Cleveland SQLSaturday
February 2nd, 2019

Introduction to Query Store

Erin Stellato
Erin@SQLskills.com

Erin Stellato

Principal Consultant, SQLskills

Erin@SQLskills.com

@erinstellato

www.sqlskills.com/blogs/erin

Trainer/Speaker

In addition to consulting, I teach content for our IE0: Accidental DBA course, and our IEPTO2: Performance Tuning and Optimization course

PASS Volunteer

I was a member of the PASS Nomination Committee this past year, have previously served on the board of my local user group (ONSSUG) and supported the Performance Virtual Chapter

Data Platform MVP

I have been fortunate to be recognized as an MVP by Microsoft since 2012

- Team of SQL Server consultants:
  - Kimberly Tripp (@KimberlyLTripp)
  - Jonathan Kehayias (@SQLPoolBoy)
  - Glenn Berry (@GlennAlanBerry)
  - Paul Randal (@PaulRandal)
  - Erin Stellato (@ErinStellato)
  - Tim Radney (@Tim Radney)
- Instructor-led training: Immersion Events and onsite
- Online training through Pluralsight (www.pluralsight.com)
- Consulting: Health checks, design, performance, upgrades, Azure
- Remote DBA: system monitoring and troubleshooting
- Conferences: PASS Summit, SQLintersection, SQLBits
- Get our newsletter: https://www.sqlskills.com/Insider

2019 Classes and Services

- In-depth, instructor-led, technical training for SQL Server
- 2019 classes in Chicago (April/May)
  - IEPTO1/2: Performance Tuning
  - IE0: Accidental/Junior DB
  - IEAzure: Azure SQL DB, Azure VMs & MI
  - IEPowerBI: PowerBI & Report Server, SSRS
  - IECAG: Clustering and AGs
  - IEUpgrade: Upgrade/Migrate to SQL 2017
- Online, live Immersion Events through the year
  - Query Store, Columnstore, Transactions/locking/blocking, Query performance, Upgrade, Very-large tables and partitioning
- For more information: https://www.sqlskills.com/schedule/
- For more information: https://www.sqlskills.com/services/

- Email paul@sqlskills.com with the subject line: User Group Pluralsight code to get a FREE (no catches, no credit card) 30-day trial of over 150+ hours of SQLskills content (and more)
- For example:
  - https://www.pluralsight.com/courses/sqlserver-query-store-introduction
    - 3 hours providing an introduction to Query Store
  - https://app.pluralsight.com/library/courses/sqlserver-azure-database/
    - 1.5 hours on Automatic Tuning in SQL Server 2017 and Azure SQL Database
  - http://www.pluralsight.com/courses/sqlserver-optimizing-stored-procedure-performance
    - 7 hours (!) on stored procedure performance tuning (Kimberly)

Scripts for this session, along with a PDF of the slides, will be available on the CLE SQLSaturday site.

Abstract

One of the most highly anticipated new features in the SQL Server 2016 release was Query Store. If you've ever had to drop everything to troubleshoot a sudden drop in performance, then this is a feature you want in your environment. In this session we will cover the data collected, how to use the built-in reports to find problematic queries, and walk through plan forcing in Query Store. Get ready to make troubleshooting easier with this feature that's included in all editions of SQL Server 2016 and higher!

Defining Query Store

- Marketed as a flight data recorder
- Provides information about query execution
- Available in *all* editions of SQL Server
  - **Highly recommended** to run the latest release (2016 SP2 CU5 or 2017 CU13) for all editions
  - Enabled by default in Azure SQL Database since Q4 of 2016

Who should use Query Store?

You do not need to be a performance-tuning expert to use Query Store. In fact, Query Store was written so that *anyone* who uses SQL Server can find the worst-performers and regressed queries.

If you are a performance-tuning expert or have extensive experience with SQL Server, then you'll be able to work beyond the standard functionality of Query Store to fully leverage all it has to offer.

Query Store Details

- Enabled at the database level
- Data persisted in internal tables
- Cannot be enabled for master, tempdb, or model
  - http://www.sqlskills.com/blogs/erin/sql-server-query-store-default-settings/
- Requires VIEW DATABASE STATE to view the data
- Requires db_owner to force/unforce plans
- Query Store data is not captured on readable secondaries
  - https://feedback.azure.com/forums/908035-sql-server/suggestions/32899126-enable-query-store-for-collection-on-a-read-only-r

Query and Performance Data in Query Store

sys.dm_exec_query_plan
sys.dm_exec_cached_plans
sys.dm_exec_sql_text

Plan Store

sys.dm_exec_query_stats

Runtime Stats Store

sys.dm_exec_session_wait_stats*

Wait Stats Store

SQL Server 2017 and Azure SQL Database

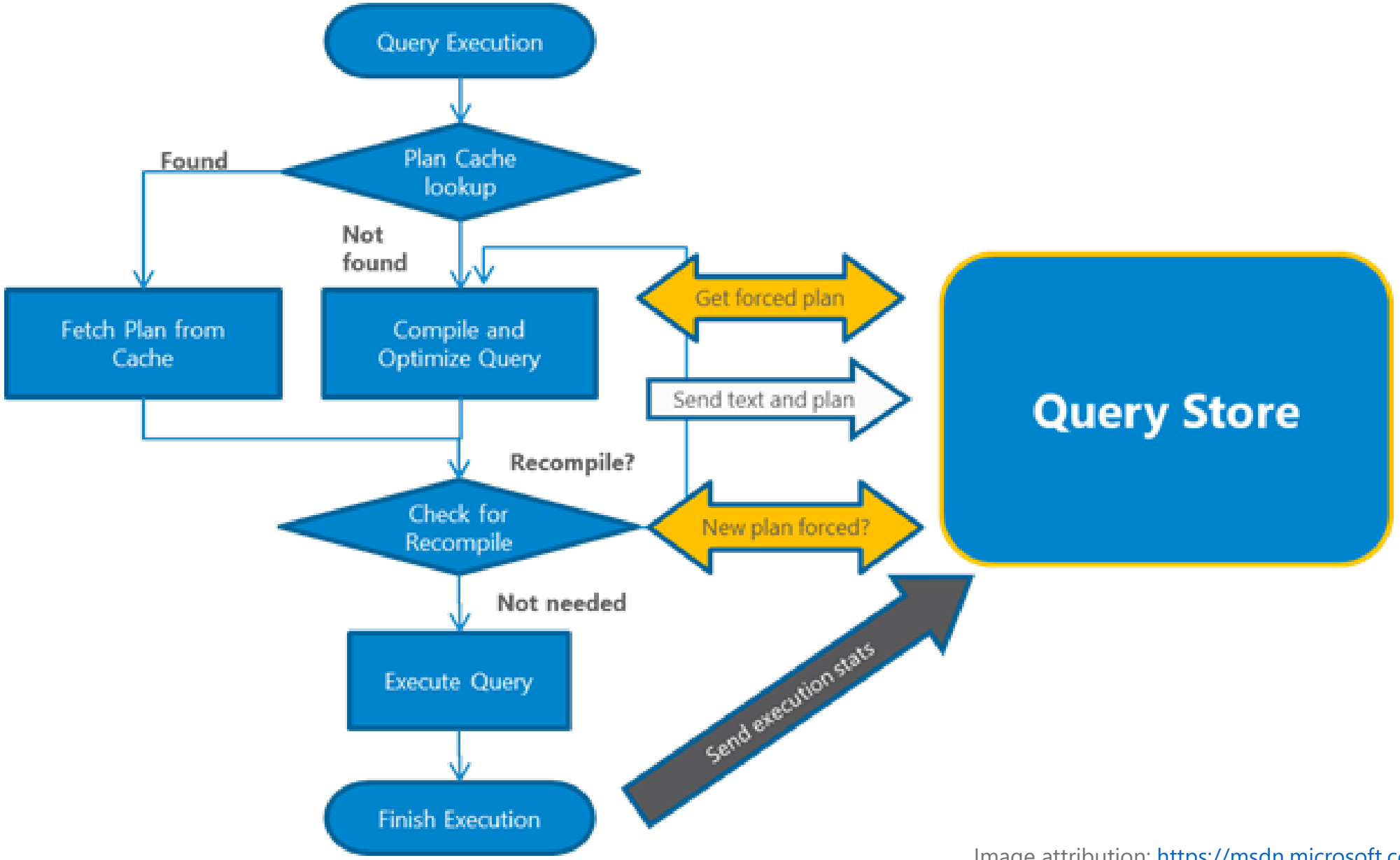


Image attribution: https://msdn.microsoft.com/en-us/library/mt631173.aspx

Finding the “Worst Performers” in Query Store

Query Store Settings(1)

- Query Store is *not* enabled by default in SQL Server 2016 or SQL Server 2017
- Query Store *is* enabled by default in for new databases in Azure SQL Database
- There are nine settings related to Query Store configuration, and they affect what data is collected and how it is stored
- Improper configuration can cause data to be removed from Query Store before expected, or Query Store can stop collecting data entirely
- Stored in sys.database_query_store_options, along with current and desired status

Query Store Settings (2)

OPERATION_MODE = [READ_WRITE | READ_ONLY]

Why it matters: Do you want to capture new data?

QUERY_CAPTURE_MODE = [ALL | AUTO | NONE]

Why it matters: Do you want to capture all queries, or just those most relevant to your workload?

MAX_PLANS_PER_QUERY =

Why it matters: You may have more than 200 unique plans for a query (?) (!)

Query Store Settings (3)

MAX_STORAGE_SIZE_MB =

Why it matters: The default is 1GB or less, "right" value depends on multiple factors

CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = #)

Why it matters: You need to think about how much data you want to keep

SIZE_BASED_CLEANUP_MODE = [AUTO | OFF]

Why it mattes: If this is OFF and MAX_STORAGE_SIZE_MB is reached, Query Store will switch to READ_ONLY

Query Store Settings (4)

DATA_FLUSH_INTERVAL_SECONDS =

Why it matters: Determines how frequently Query Store is written to disk

INTERVAL_LENGTH_MINUTES =

Why it matters: Affects the space needed by Query Store and the time windows across which you can analyze data

WAIT_STATS_CAPTURE_MODE [ON | OFF]

Why it matters: Enabled by default (and enabled when upgrade to SQL 2017)

SQL Server 2017 and Azure SQL Database

Default Values

Setting	2016/2017	SQL DB Basic	SQL DB Standard	SQL DB Premium
QUERY_CAPTURE_MODE	ALL	AUTO	AUTO	AUTO
MAX_STORAGE_SIZE_MB	100	10	100	1024
CLEANUP_POLICY	30	7	30	30
SIZE_BASED_CLEANUP_MODE	AUTO	AUTO	AUTO	AUTO
DATA_FLUSH_INTERVAL_ SECONDS	900	900	900	900

- Tier change in Azure SQL Databases causes a change in Query Store settings, unless they were modified using ALTER DATABASE

Demo: Configuring Query Store

How Much Space Do I Need? (1)

- There are multiple factors to consider:
  - How long are you keeping the data?
  - Across what interval are you capturing data?
  - What does your workload look like?
    - Number of unique queries?
    - Number of unique plans?
  - Are you capturing wait statistics (2017+)

How Much Space Do I Need? (2)

- Default size for Azure SQL Database is either 10MB, 100MB, or 1GB (depends on tier)
  - Counts against total space for the database
  - Interval for all is 60 minutes
- Start with 2GB and monitor
  - current_storage_size_mb in sys.database_query_store_options
    - https://www.sqlskills.com/blogs/erin/monitoring-space-used-by-query-store/
  - Extended Events
    - query_store_database_out_of_disk_space
    - query_store_disk_size_over_limit

Query Store vs. Third-Party Monitoring Tools

Query Store

- Must enable for each database
- Retention policy affects size of data in user DB
- Does not capture runtime parameters
- SQL Server 2016+
- Ability to force plans
- Captures all queries

3rd Party

- Captures information for all databases on an instance
- Data stored in a separate DB (on a separate instance)
- Captures runtime parameters with queries
- All SQL Server versions
- Cannot force plans
- Captures long-running queries

How do you fix a poorly-performing query?

- Change code and/or schema
- Add RECOMPILE
- Manually get the “best” plan in cache
- Use a plan guide
- Force a plan in Query Store

Forcing Plans with Query Store

- Query Store allows you to easily find queries with multiple plans and force one plan
- Not schema-bound
- Monitor failures with Extended events
- If a plan is no longer optimal, Query Store will continue to use it, unless you un-force it or forcing fails

A bad plan is not the one which failed, but the one which succeeded at the greatest cost.

-Anonymous DBA

Demo: Creating Plan Stability

Points to Remember with Plan Forcing(1)

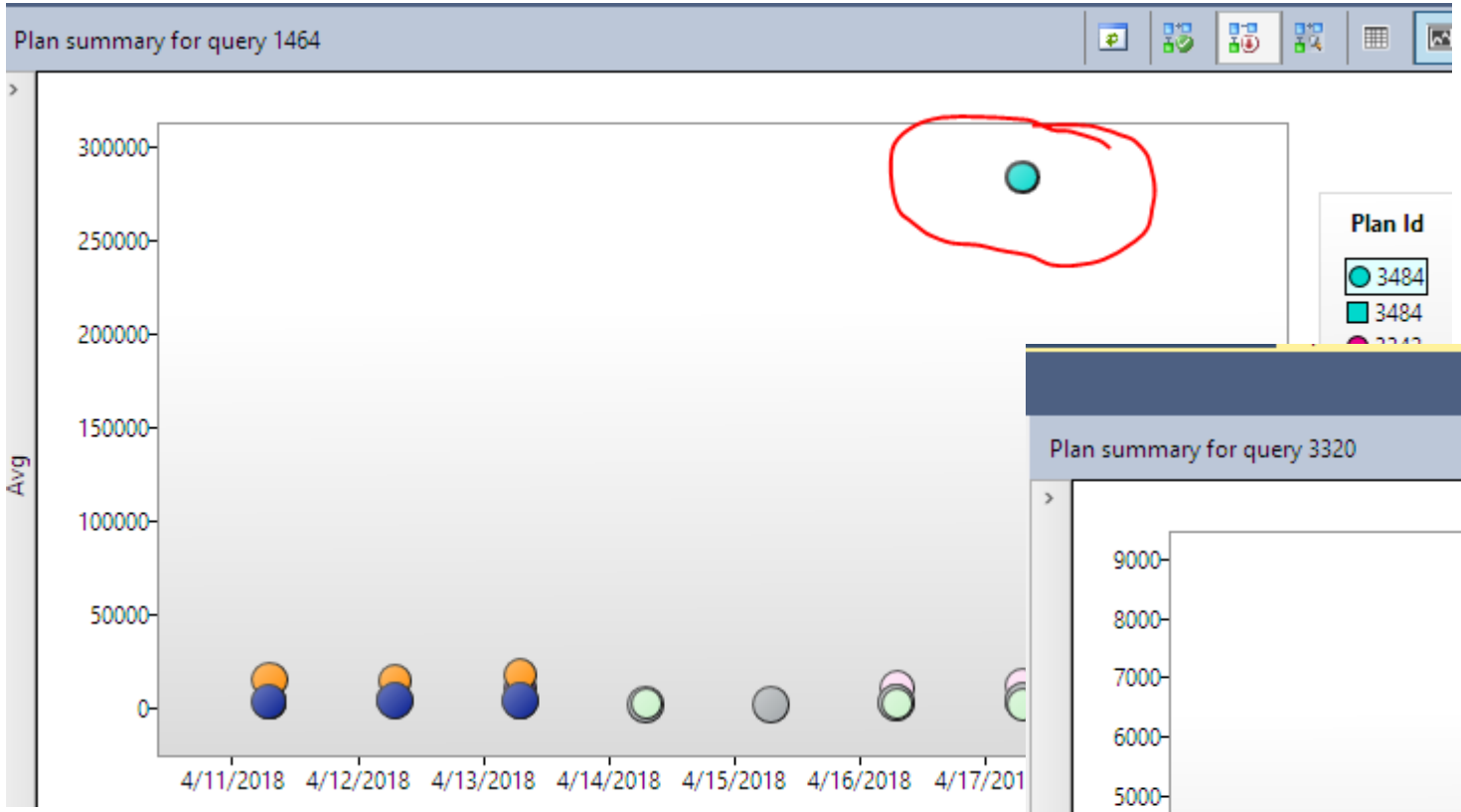
- It may not always be obvious that a plan is forced – check the actual plan and Query Store to determine
- Turning off Query Store negates the ability to use forced plans
- Forced plans will not be aged out of Query Store
- Pay attention to forced plans when testing code and schema changes
  - Changing index names
  - Changing object names

Points to Remember with Plan Forcing (2)

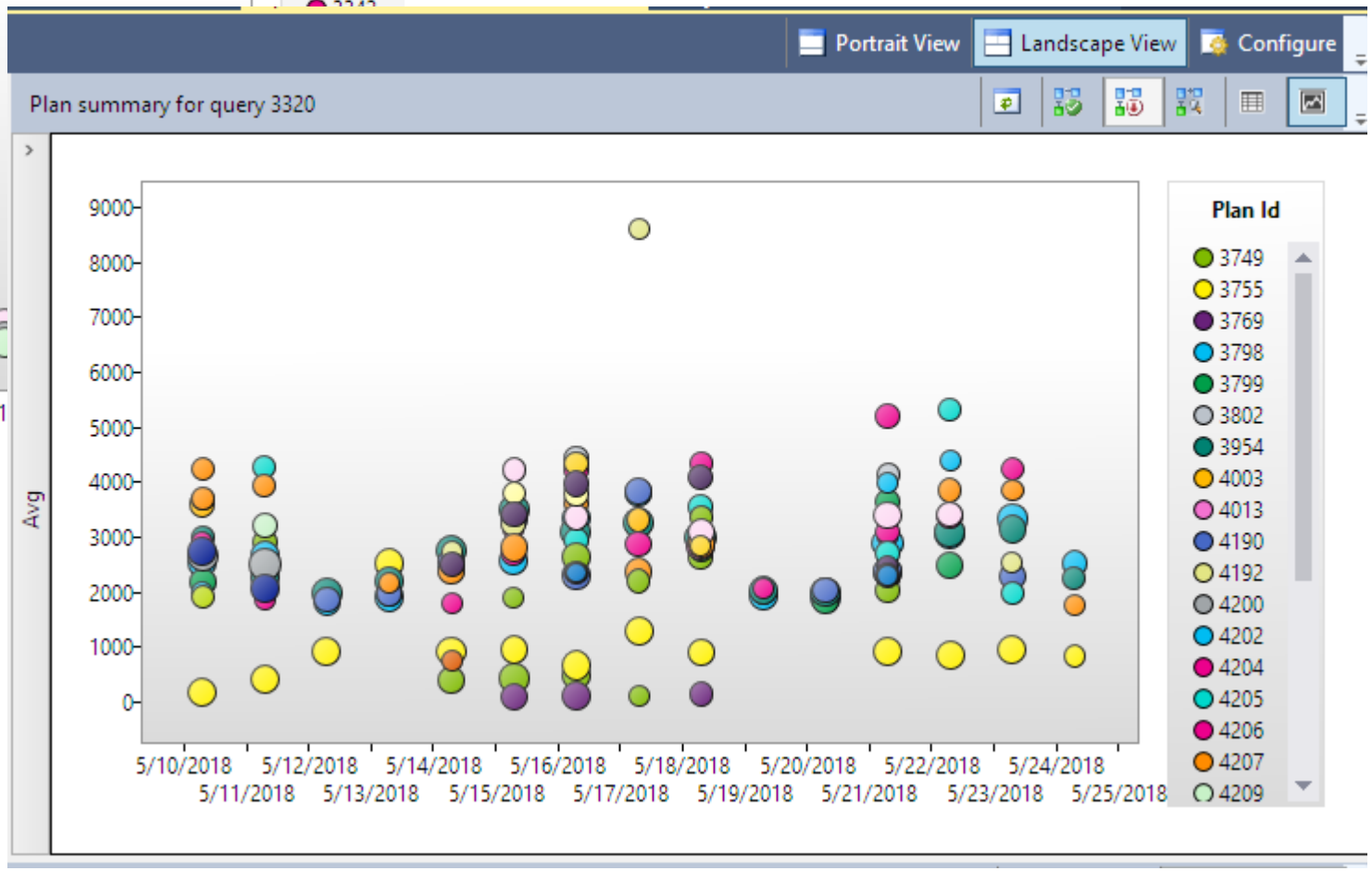
- You cannot force a plan on read-only secondary by forcing the plan on the primary
- You cannot force a plan for a query if it hasn't been generated by that query
- Query performance can be different across environments for multiple reasons – including forced plans!
- *Be aware that forced plans are removed if you installed SQL Server 2017 CU2 and then upgrade to a later CU*
  - https://www.sqlskills.com/blogs/erin/query-store-fix-in-sql-server-2017/

Typical Reasons Forcing Can Fail

- Drop an index (NO_INDEX)
- Change an index name (NO_INDEX)
- Remove columns from an index (NO_PLAN)
- Change the object_id due to DROP/CREATE rather than ALTER



Query Store plan example from class:
Query 1464 would run well the majority of the time, but then it would generate plan 3484 which was extremely slow comparatively. The query executed on a regular basis as part of a screen refresh, so this created problems for users and created high CPU on the server.



Developers decided to add an OPTION (RECOMPILE) to the query (which created a new query_id, 3320). This seemed to address the issue based on the new plans.

Solutions here could also include forcing the plan (e.g. force plan 1838 for the original query), or re-writing the query. It's important to understand WHY different plans were generated.

Uses of Query Store

- Troubleshooting query performance
- Proactively analyze workload patterns
- Understand waits at the plan level (2017)
- Reduce the risk of upgrading
  - Testing query performance before an upgrade (hardware, software, application)
  - Testing changes in the Cardinality Estimator
- Determine patterns in coding and plan execution

Resources

- Blog Posts
  - Query Store Settings
    - https://www.sqlskills.com/blogs/erin/query-store-settings/
  - Query Store Requests
    - https://www.sqlskills.com/blogs/erin/query-store-requests/
  - All related to QS:
    - https://www.sqlskills.com/blogs/erin/category/query-store/

Resources

- SQL Server: Introduction to Query Store
  - https://www.pluralsight.com/courses/sqlserver-query-store-introduction

- Automatic Tuning in SQL Server 2017 and Azure SQL Database
  - https://app.pluralsight.com/library/courses/sqlserver-azure-database/table-of-contents

Thank you!