DEMYSTIFYING QUERY STORE PLAN FORCING

Thank you!

SOLISYON

PASS DEUTSCHLAND e.V.
Microsoft Data Platform Community

Spot by NetApp

pmOne

smart casual datadesign

TIMEXTENDER

GDS BUSINESS INTELLIGENCE GmbH

redgate

scieneers DRIVEN BY DATA

Many thanks to our sponsors, without whom such an event would not be possible.

MILOŠ RADIVOJEVIĆ

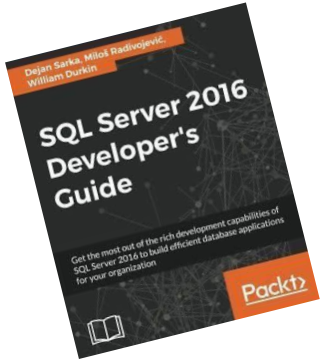
Head of Database Engineering at Entain, Austria

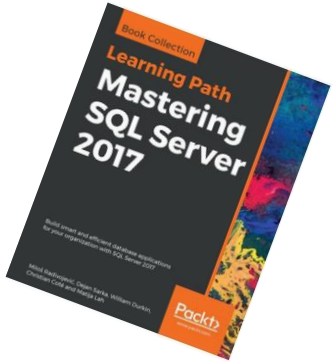
Conference speaker

Book author

Contact: milos.radivojevic@chello.at

LinkedIn: milossql





AGENDA

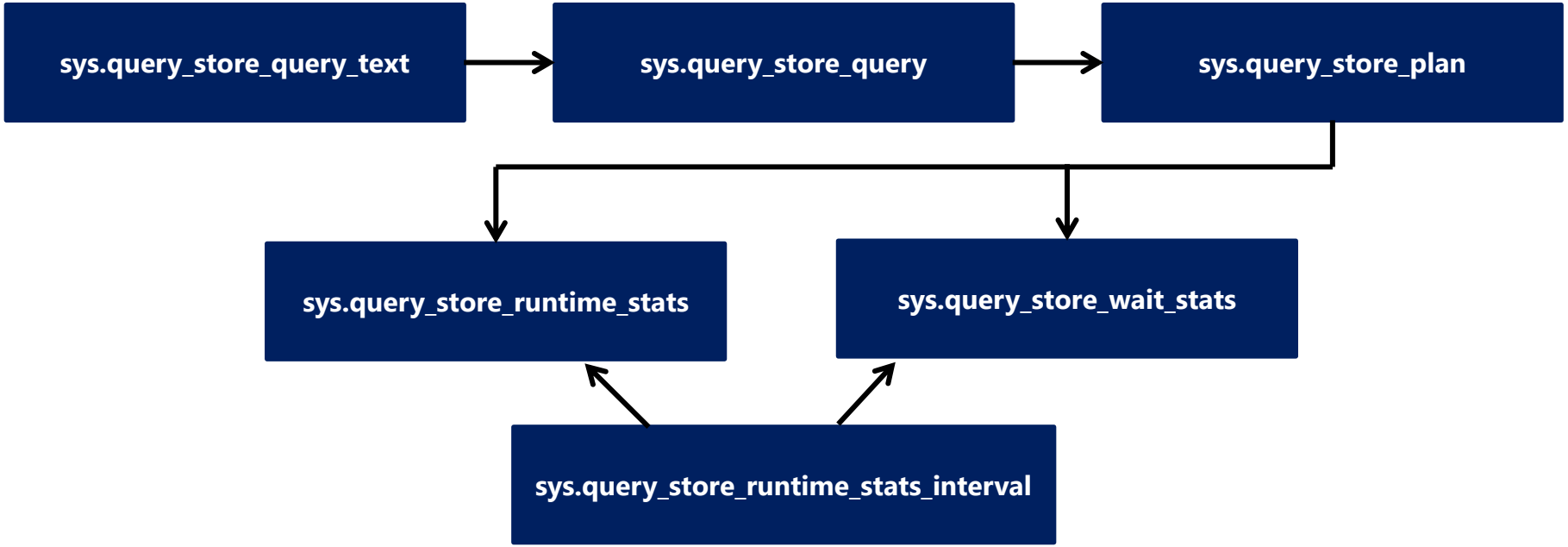
- Query Store intro
- Query Store in action
- Forcing plan
  - Details
  - When to force a plan
- Issues with forced plans
  - Forcing plan failure
  - Ignoring forced plans
- Automatic plan correction

WHAT IS QUERY STORE?

- Troubleshooting feature introduced with SQL Server 2016
  - It stores the history of queries, plans, execution details and waits statistics
- Belongs to database
  - it is persistent – survives after restart, failover etc.
- Disabled by default in on-prem databases (until SQL Server 2022), in Azure SQL enabled
- Available in all editions (one feature requires Enterprise Edition)

QUERY STORE INTERFACE

- 8 public catalog views & 7 system stored procedures
- most important views:

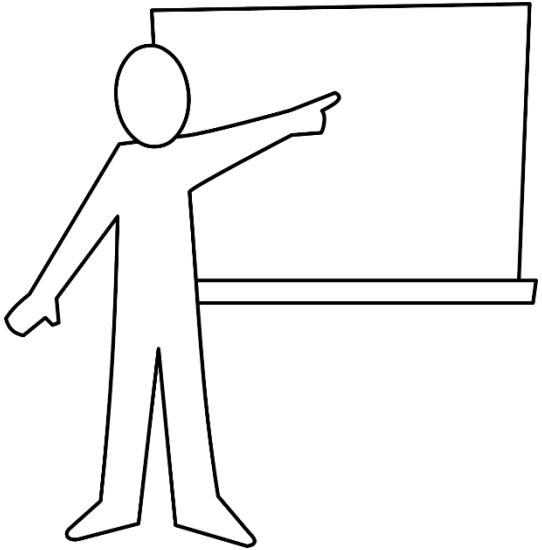


WHAT CAN WE DO WITH QUERY STORE?

- Identify and fix query plan regressions
  - Reduce the risk of upgrading, patching and reconfiguring
- Support troubleshouting process
  - Was this query slow last weekend?
  - Why my query was slow last Saturday?
  - What are unstable queries (with multiple plans)?
  - Find out unfinished queries or queries that ended with an exception
- Analyze workload patterns

Query Store in Action

QUERY STORE IN ACTION - DEMO SETUP



https://github.com/Microsoft/sql-server-samples/releases/download/wide-world-importers-v1.0

```sql
USE [master]
GO
RESTORE DATABASE WideWorldImporters FROM DISK = 'C:\Temp\WideWorldImporters-Full.bak' WITH
MOVE 'WWI_Primary' TO 'D:\MSSQL2019\DATA\WideWorldImporters.mdf',
MOVE 'WWI_UserData' TO 'D:\MSSQL2019\DATA\WideWorldImporters_Userdata.ndf',
MOVE 'WWI_Log' TO 'D:\MSSQL2019\DATA\WideWorldImporters.ldf',
MOVE 'WWI_InMemory_Data_1' TO 'D:\MSSQL2019\DATA\WWI_InMemory_Data_1'
```

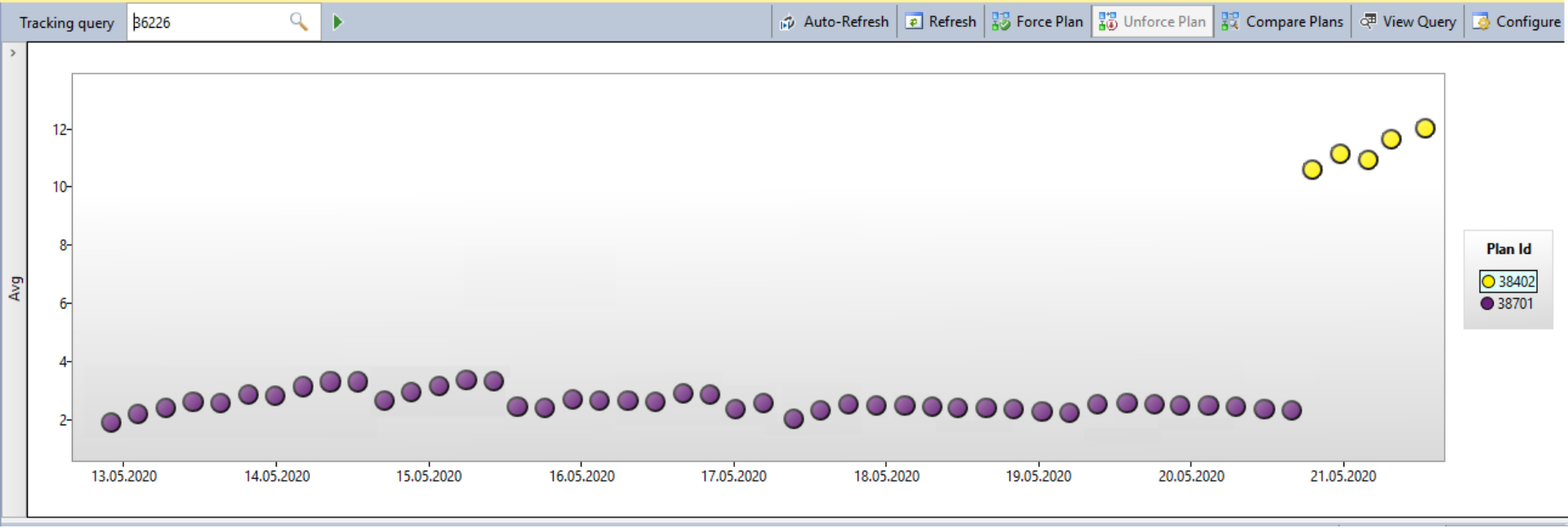
117 %

Messages

```
Processed 1464 pages for database 'WideWorldImporters', file 'WWI_Primary' on file 1.
Processed 53096 pages for database 'WideWorldImporters', file 'WWI_UserData' on file 1.
Processed 33 pages for database 'WideWorldImporters', file 'WWI_Log' on file 1.
Processed 3862 pages for database 'WideWorldImporters', file 'WWI_InMemory_Data_1' on file 1.
Converting database 'WideWorldImporters' from version 852 to the current version 869.
Database 'WideWorldImporters' running the upgrade step from version 852 to version 853.
Database 'WideWorldImporters' running the upgrade step from version 853 to version 854.
Database 'WideWorldImporters' running the upgrade step from version 854 to version 855.
Database 'WideWorldImporters' running the upgrade step from version 855 to version 856.
Database 'WideWorldImporters' running the upgrade step from version 856 to version 857.
Database 'WideWorldImporters' running the upgrade step from version 857 to version 858.
Database 'WideWorldImporters' running the upgrade step from version 858 to version 859.
Database 'WideWorldImporters' running the upgrade step from version 859 to version 860.
Database 'WideWorldImporters' running the upgrade step from version 860 to version 861.
Database 'WideWorldImporters' running the upgrade step from version 861 to version 862.
Database 'WideWorldImporters' running the upgrade step from version 862 to version 863.
Database 'WideWorldImporters' running the upgrade step from version 863 to version 864.
Database 'WideWorldImporters' running the upgrade step from version 864 to version 865.
Database 'WideWorldImporters' running the upgrade step from version 865 to version 866.
Database 'WideWorldImporters' running the upgrade step from version 866 to version 867.
Database 'WideWorldImporters' running the upgrade step from version 867 to version 868.
Database 'WideWorldImporters' running the upgrade step from version 868 to version 869.
RESTORE DATABASE successfully processed 58455 pages in 0.829 seconds (550.874 MB/sec).
```

WHEN SHOULD WE FORCE A PLAN?

- A good candidate for plan forcing



WHEN SHOULD WE FORCE A PLAN?

- Which plan to force here?

Tracking query 869305

Auto-Refresh | Refresh | Force Plan | Unforce Plan | Compare Plans | View Query | Configure

Avg: 150, 100, 50, 0

30.06.2019 07.07.2019 14.07.2019 21.07.2019 28.07.2019 04.08.2019 11.08.2019 18.08.2019 25.08.2019 01.09.2019 08.09.2019 15.09.2019 22.09.2019 29.09.2019 06.10.2019

Plan Id
892399
893164
893432
892971
894749
978147
978103
895637
1047365
982309
1066289
1068533

Plan 894749 [not forced]

Force Plan | Unforce Plan

Query 1: Query cost (relative to the batch): 100%

SELECT Cost: 0 % ← Sort Cost: 2 % ← Compute Scalar Cost: 0 % ← Stream Aggregate (Aggregate) Cost: 0 % ← Sort Cost: 2 % ← Compute Scalar Cost: 0 % ← Nested Loops (Left Outer Join) Cost: 0 % ← Nested Loops (Inner Join) Cost: 0 %

PLAN FORCING - SUMMARY

- Should be perceived as a temporary solution!
- SQL Server creates a new plan with a shape of the forced plan
- Huge responsibility – db_owner permission required!
- Plan forcing is not always a good idea!
- Check carefully plan history for a query_id before you force the plan
- Use sp_query_store_force_plan rather than GUI

THE MOST IMPORTANT FACT ABOUT PLAN FORCING

Execution plan **is not forced for a given query**, but for a given **query_id!**

WHEN PLAN FORCING DOESN'T WORK?

- When a forced plan fails
  - When a plan based on the forced plan is not possible
  - Index, tables or other database objects are dropped or renamed
  - Forced plan has conflict with query hints or SET options
- When SQL Server ignores a forced plan

DEMO

Forced Plan Failure

FORCED PLAN FAILURE – WHAT SQL SERVER WILL DO?

- It will try to create a plan based on the forced plan

- After realizing that this is not possible
  1. It increments the counter of forced plan failures
  2. It sets a reason for the last failure
  3. It creates a new plan as it would do without forcing

FORCED PLAN FAILURE - REASONS

error_number	last_force_failure_reason
8712	NO_INDEX
8698	NO_PLAN
8689	NO_DB
8684	TIME_OUT
8637	ONLINE_INDEX_BUILD
8683	INVALID_STARJOIN
8690	HINT_CONFLICT
8691	SETOPT_CONFLICT
8713	VIEW_COMPILE_FAILED
8694	DQ_NO_FORCING_SUPPORTED
8695	GENERAL_FAILURE

ACCELERATED_PLAN_FORCING in SQL Server 2019 – a solution for TIME_OUT

DEMO

Where is my forced plan?

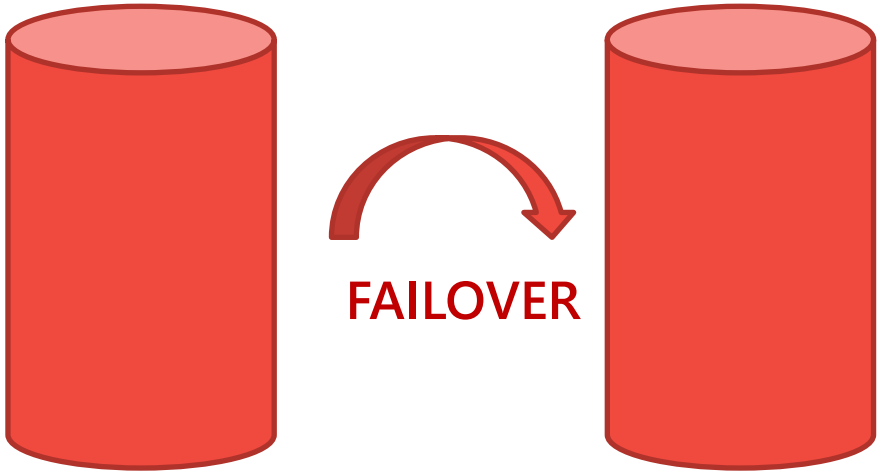
WHERE IS MY FORCED PLAN?

```sql
CREATE TYPE dbo.Ids AS TABLE(
            Id INT NOT NULL PRIMARY KEY CLUSTERED
)
GO

CREATE OR ALTER PROCEDURE dbo.GetList
(@tvp AS dbo.Ids READONLY)
AS
            SELECT t.*
            FROM dbo.T t
            INNER JOIN @tvp tvp ON t.id = tvp.Id;
GO

DECLARE @t AS dbo.Tvp;
INSERT @t SELECT TOP (10) id FROM dbo.T ORDER BY 1 DESC;
EXEC dbo.GetList @t;
GO 3
```

+ Plan Forcing



WHERE IS MY FORCED PLAN?

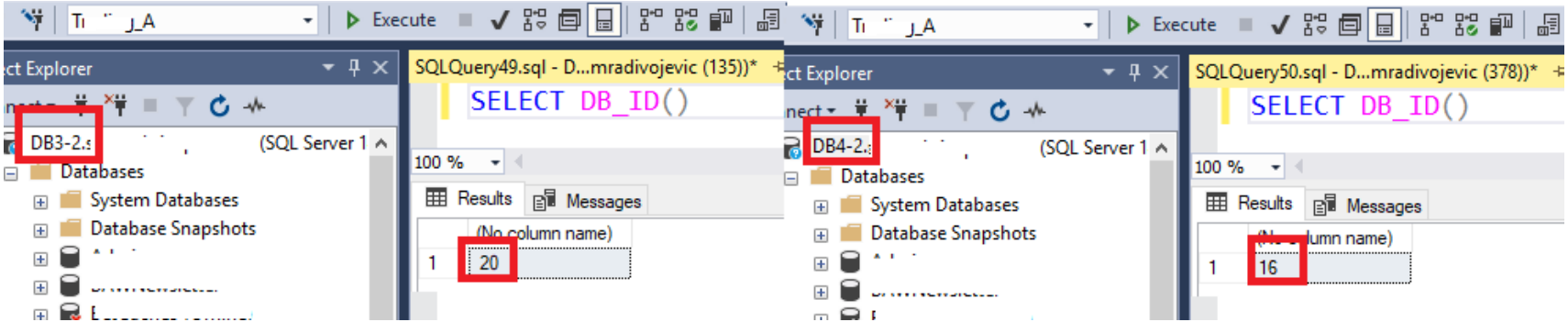
```
SELECT * FROM sys.query_store_query WHERE query_hash = (SELECT query_hash FROM sys.query_store_query WHERE query_id = 35);
```

query_id	query_text_id	context_settings_id	object_id	batch_sql_handle	query_hash
35	32	3	981578535	0x0300140027B3813AA714A200F5AA0000010000000000000000000000000000000000000000000000000000	0x54277B2880A5E863
57	32	3	981578535	0x0300100027B3813AA714A200F5AA0000010000000000000000000000000000000000000000000000000000	0x54277B2880A5E863

query_text_id	context_settings_id	object_id	batch_sql_handle
32	3	981578535	0x0300140027B3813AA714A200F5AA0000010000000000000000000000000000000000000000000000000000
32	3	981578535	0x0300100027B3813AA714A200F5AA0000010000000000000000000000000000000000000000000000000000

WHERE IS MY FORCED PLAN?

- Query_id has been changed because of different database_id attributes on the primary and secondary!



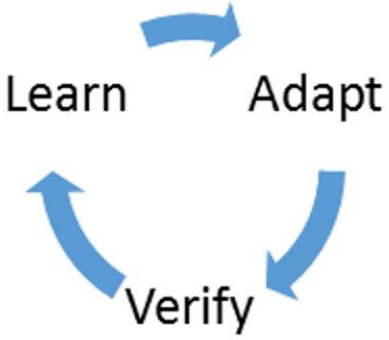
PLAN FORCING AND TEMPORARY OBJECTS

- **Forced plan is associated to a query_id and not to a query!**
- Query_Id depends on
  - query_text_id
  - object_id
  - context_settings_id
  - query_parameterization_type
  - batch_sql_handle
- When query_id is changed, forced plan is not applied!

WHEN PLAN FORCING DOESN'T WORK?

- When you make any change in a query (incl. comments)
- When you DROP/CREATE objects
- Environment variables changed
- Parametrization type changed
- When your procedure use table valued parameter and
  - ALTER PROCEDURE is executed
  - After a failover, when dbs in AG have different db_ids

AUTOMATIC TUNING



- A very ambitious SQL Server feature. It:
  - continuously monitors, analysis and learns about the workload
  - identifies potential issues and improvements
  - can automatically fix issues and apply improvements
  - verifies applied improvements

ENTERPRISE EDITION

AUTOMATIC TUNING

SQL AZURE	ON-PREM DATABASE
AUTOMATIC INDEX MANAGEMENT	AUTOMATIC PLAN CORRECTION
AUTOMATIC PLAN CORRECTION	

- On-prem (CURRENTLY): Automatic Tuning = Automatic Plan Correction
- Automatic Plan Correction:
  - offline plan regression recommendations
  - automatic correction

AUTOMATIC TUNING

- When Query Store is enabled, and you have Enterprise Edition AT is always turned ON
- If **FORCE_LAST_GOOD_PLAN** parameter is set to
  - OFF => plan regression recommendations and manual plan correction

    The results of the analysis are exposed via

  sys.dm_db_tuning_recommendations ON => automatic plan correction

    Query Store applies and verifies applied changes

DEMO

Automatic Tuning

AUTOMATIC TUNING AND PLAN FORCING

- **Offline**
  - Plan is forced manually, and it is persistent
- **Online**
  - Automatic forced plans are usually not persisted between restarts of the SQL Server instance!

sys.dm_db_tuning_recommendations

- Notification on the top of the DMV

ALERT: Query plans in the database have been changed

A

To Milos Radivojevic; **All_**

The actual sender of this message is different than the normal sender. Click here to learn more.

Reply | Reply All | Forward

Do. 22.10.2020 1

The following plans in the database have been recently changed.

objectname	query_id	valid_since	oldCPUTime	newCPUTime
usp_CT	52572	2020-10-22 08:09:12.6933333	73	121

PLAN FORCING - SUMMARY

- Allows you to fix plan regressions without code changes
- Use it as workaround, not as a solution!
- Huge responsibility (db_owner)
- Forced plan is associated to a **query_id** and not a query
- Be careful with parameter sensitive queries
- Be careful with queries using table variables and TVPs
- Check whether plan forcing is respected **(Queries With Forced Plans** report)

Feedback please!

PASS DEUTSCHLAND e.V.
Microsoft Data Platform Community

https://feedback.dsmuc.de/