"With our focused investment in performance and scale, simply upgrading to SQL 2016 could bring 25% performance improvement…"

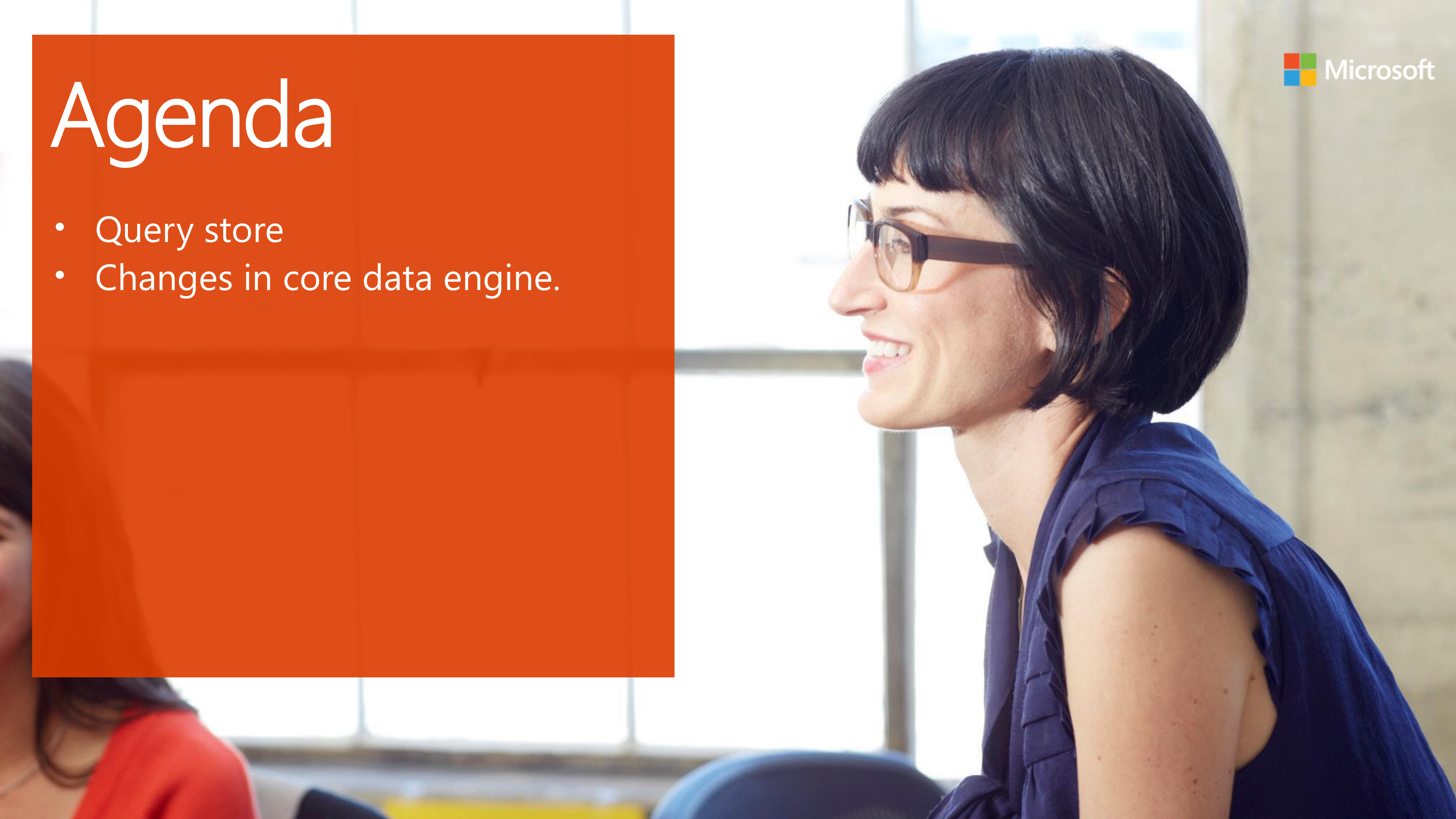
Rohan Kumar, Director of SQL Software Engineering

> "SQL Server 2016 running on the same hardware as SQL Server 2014, 2012, 2008, 2008 R2 or 2005 uses fewer resources and executes a wide range of workloads faster..."

Bob Dorr, Principle Engineer SQL Server Support

Agenda

- Query store
- Changes in core data engine.



Query Store & Live Statistics

What Will We Do?

Objectives

- The query store contains two stores: a plan store that persists the execution plans, and a run-time stats store that persists the statistics surrounding query execution, such as CPU, I/O, memory, and other metrics. SQL Server retains this data until the space allocated to Query Store is full. To reduce the impact on performance, SQL Server writes information to each of these stores asynchronously.

Walk Away with this

- *A rich persisted* db of *query execution over time* for SQL Server/Azure DB
- The *central starting point* for Query Tuning and Troubleshooting

Query Store

Stores

- The query store contains two stores:
  - A plan store that persists the execution plans
  - A run-time stats store that persists the statistics surrounding query execution, such as CPU, I/O, memory, and other metrics.
- SQL Server retains this data until the space allocated to Query Store is full (30 days). To reduce the impact on performance, SQL Server writes information to each of these stores asynchronously.

Query Store Dashboard

- **Regressed Queries** Use this dashboard to review queries that might have regressed because of execution plan changes.

- **Overall Resource Consumption** Use this dashboard to visualize overall resource consumption during the last month in four charts: duration, execution count, CPU time, and logical reads.

- **Top Resource Consuming Queries** Use this dashboard to review queries in the set of top 25 resource consumers during the last hour.

- **Tracked Queries** Use this dashboard to monitor a specify query.

The What and the Why

- A store of compiled queries, plans, and stats
    - A **database** of query history made up of memory structures and system tables
    - Contains **statements, plans, properties, and statistics**
- **A *Query* Flight Recorder**

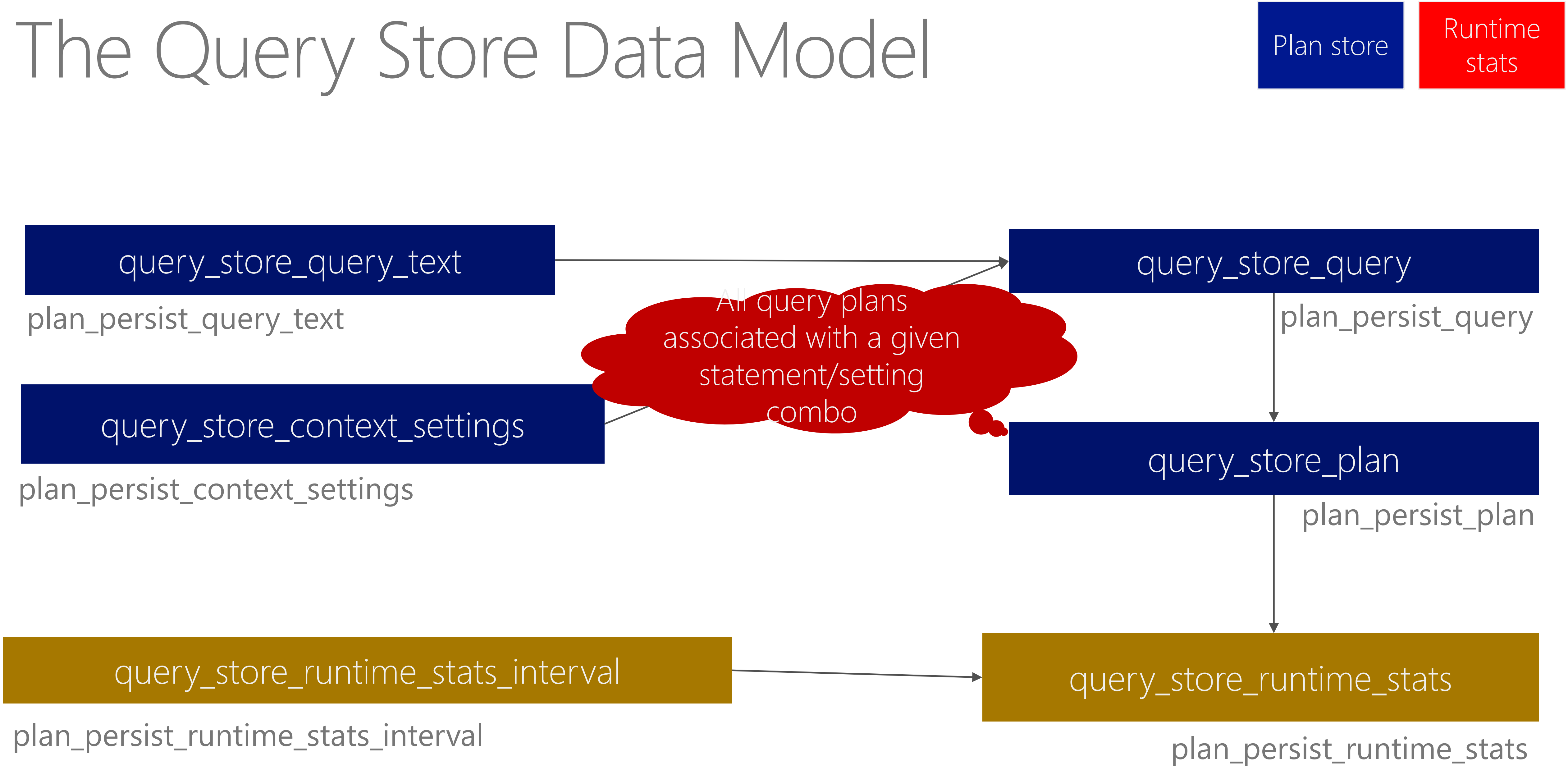
Survives crashes

Records critical information

Records for limited time

Doesn’t record everything

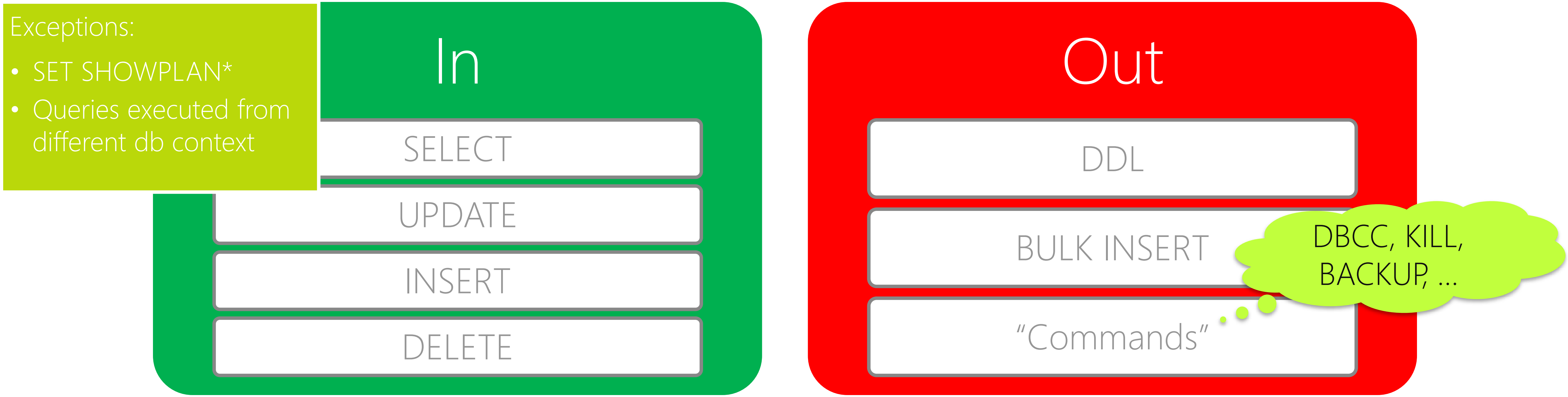
The Query Store Data Model



Plan store
Runtime stats
query_store_query_text
plan_persist_query_text
query_store_context_settings
plan_persist_context_settings
All query plans associated with a given statement/setting combo
query_store_query
plan_persist_query
query_store_plan
plan_persist_plan
query_store_runtime_stats_interval
plan_persist_runtime_stats_interval
query_store_runtime_stats
plan_persist_runtime_stats

Tracked Query

- Any T-SQL *DML statement*
- Independent of what is cached
- Each statement in an *object* is a query
- Statement text appears in parameterized form



Exceptions:

- SET SHOWPLAN*
- Queries executed from different db context

In

SELECT

UPDATE

INSERT

DELETE

Out

DDL

BULK INSERT

“Commands”

DBCC, KILL, BACKUP, ...

Query Store and Statistics

Compilation

Timestamp – first and last

Count

Duration – total, avg, last

Bind (CPU and Duration) – total, avg, last

Optimize (CPU and Duration) – total, avg, last

Memory – total, avg, and max

Plan Store

Execution

Timestamp – first and last

Count

Duration – min, max, last, total, avg, stdev

CPU – min, max, last, total, avg, stdev

Logical I/O (rw) – min, max, last, total, avg, stdev

Physical Reads – min, max, last, total, avg, stdev

CLR – min, max, last, total, avg, stdev

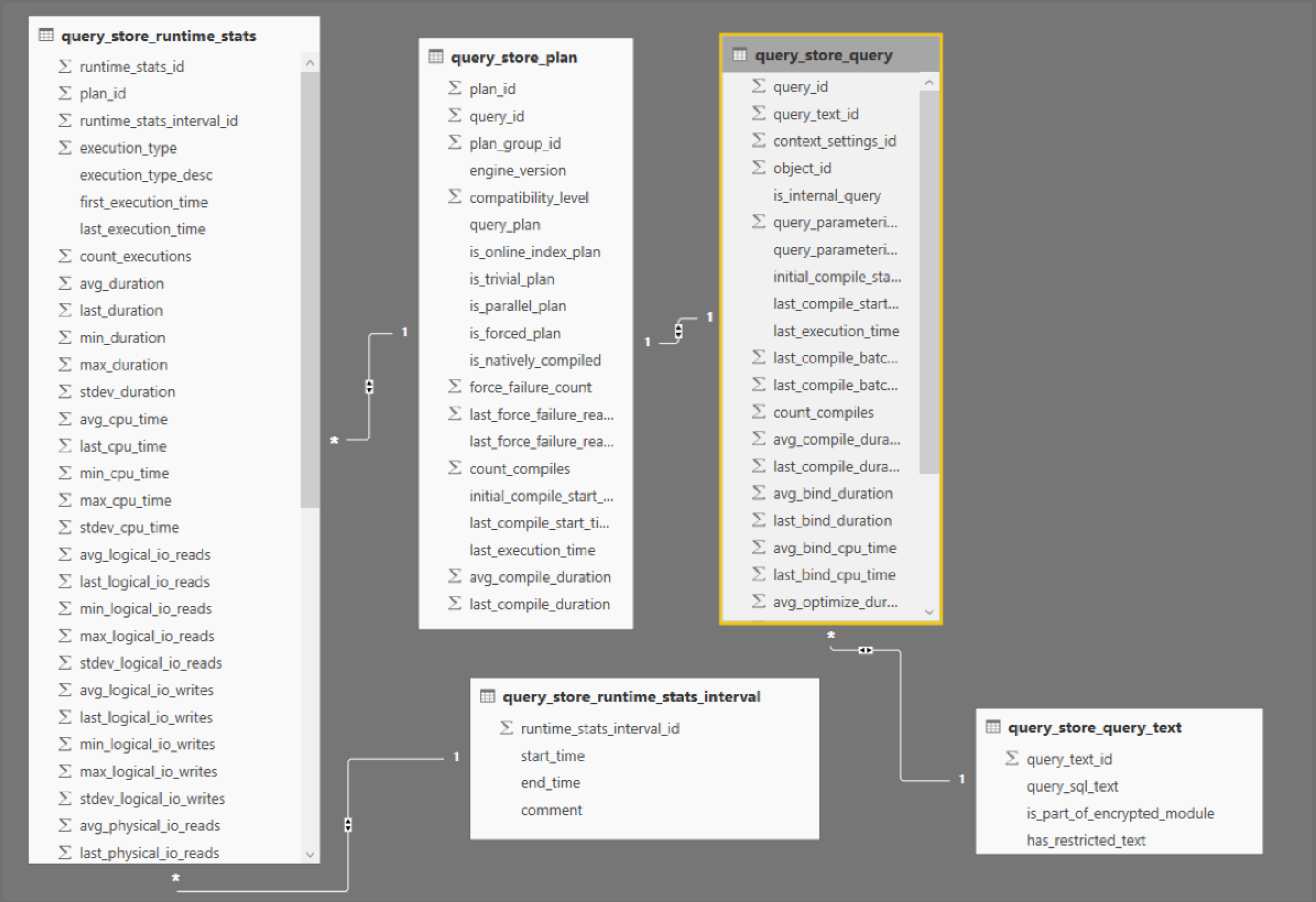
DOP – min, max, last, total, avg, stdev

Query Memory Used – min, max, last, total, avg, stdev

Rowcount – min, max, last, total, avg, stdev

Runtime Stats Store

Aggregated at interval level (default 60 mins)



query_store_runtime_stats
- runtime_stats_id
- plan_id
- runtime_stats_interval_id
- execution_type
- execution_type_desc
- first_execution_time
- last_execution_time
- count_executions
- avg_duration
- last_duration
- min_duration
- max_duration
- stdev_duration
- avg_cpu_time
- last_cpu_time
- min_cpu_time
- max_cpu_time
- stdev_cpu_time
- avg_logical_io_reads
- last_logical_io_reads
- min_logical_io_reads
- max_logical_io_reads
- stdev_logical_io_reads
- avg_logical_io_writes
- last_logical_io_writes
- min_logical_io_writes
- max_logical_io_writes
- stdev_logical_io_writes
- avg_physical_io_reads
- last_physical_io_reads

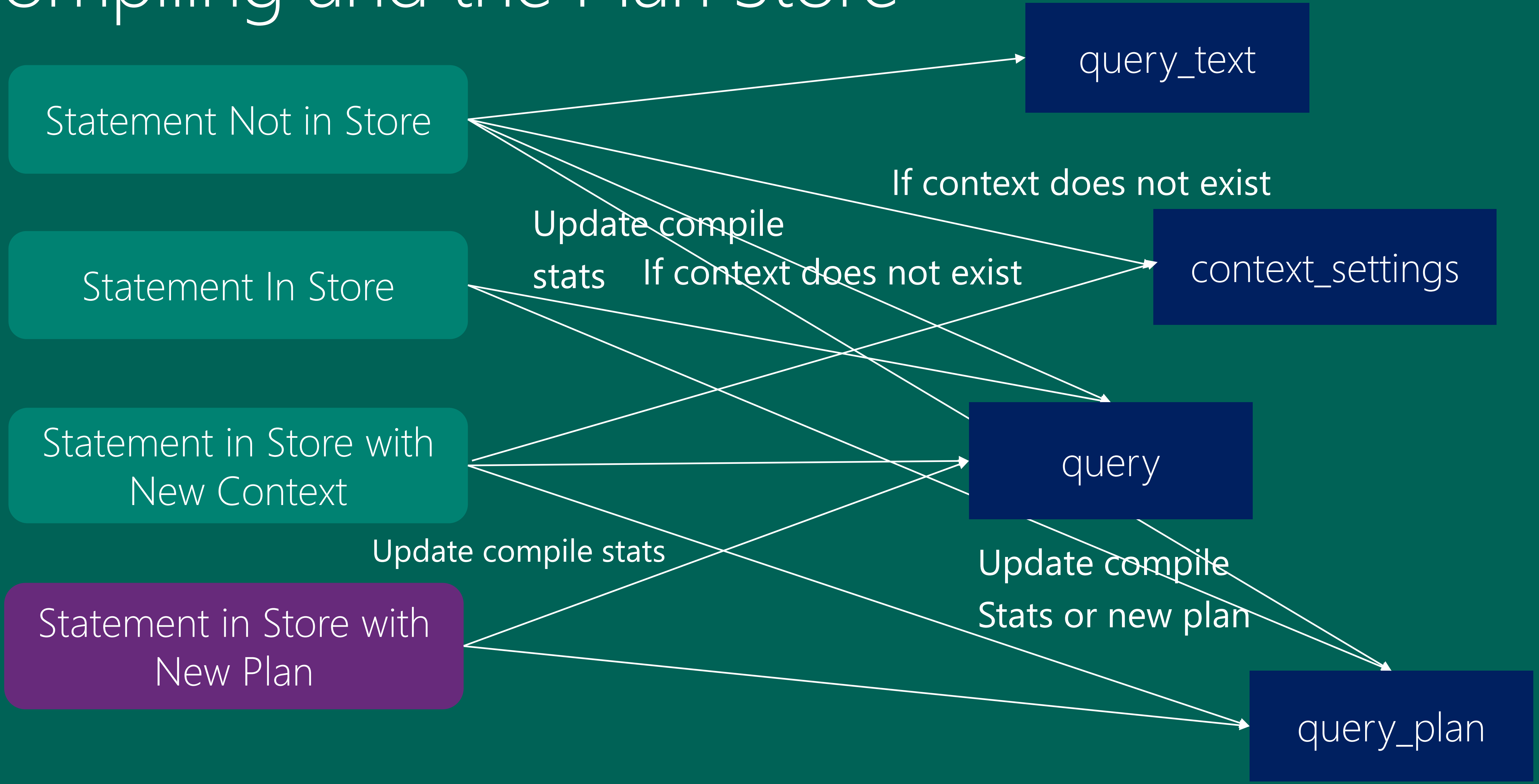
query_store_plan
- plan_id
- query_id
- plan_group_id
- engine_version
- compatibility_level
- query_plan
- is_online_index_plan
- is_trivial_plan
- is_parallel_plan
- is_forced_plan
- is_natively_compiled
- force_failure_count
- last_force_failure_rea...
- last_force_failure_rea...
- count_compiles
- initial_compile_start_...
- last_compile_start_ti...
- last_execution_time
- avg_compile_duration
- last_compile_duration

query_store_query
- query_id
- query_text_id
- context_settings_id
- object_id
- is_internal_query
- query_parameteri...
- query_parameteri...
- initial_compile_sta...
- last_compile_start...
- last_execution_time
- last_compile_batc...
- last_compile_batc...
- count_compiles
- avg_compile_dura...
- last_compile_dura...
- avg_bind_duration
- last_bind_duration
- avg_bind_cpu_time
- last_bind_cpu_time
- avg_optimize_dur...

query_store_runtime_stats_interval
- runtime_stats_interval_id
- start_time
- end_time
- comment

query_store_query_text
- query_text_id
- query_sql_text
- is_part_of_encrypted_module
- has_restricted_text

Compiling and the Plan Store



Statement Not in Store

Statement In Store

Statement in Store with New Context

Statement in Store with New Plan

query_text

context_settings

query

query_plan

If context does not exist

Update compile stats

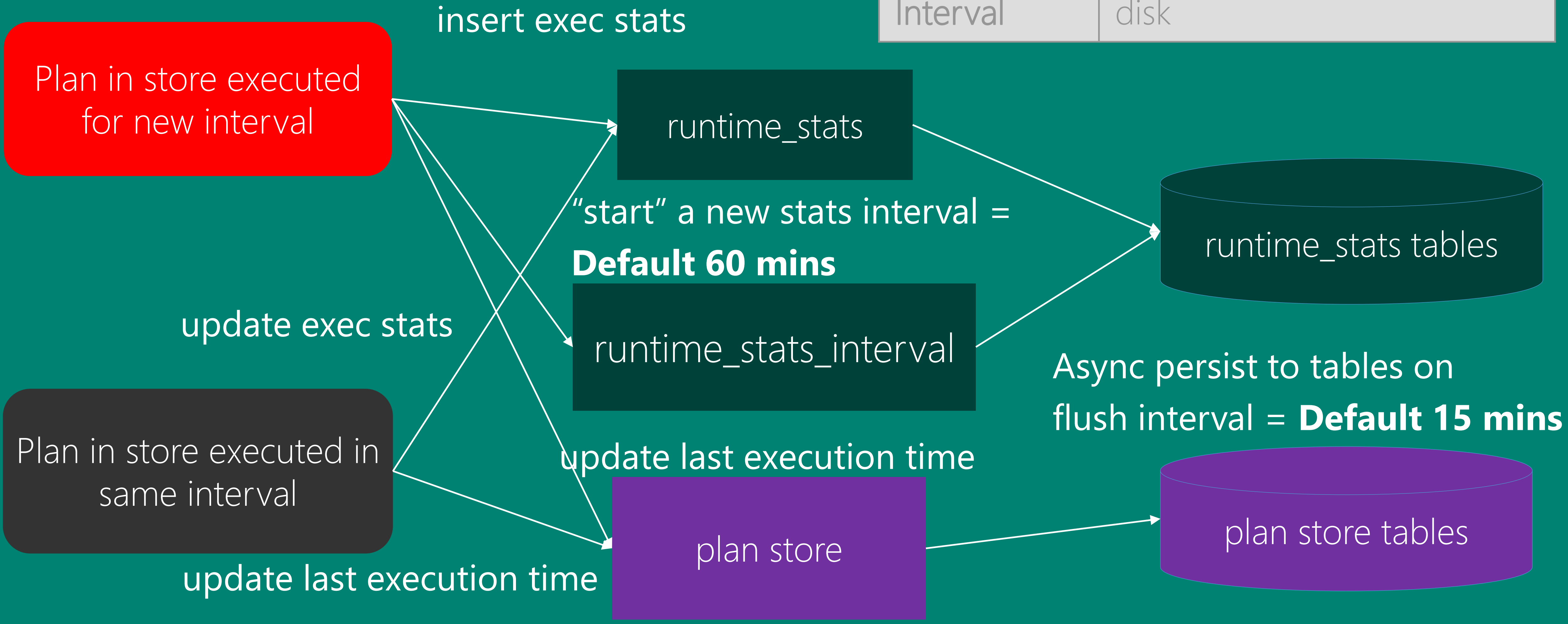
If context does not exist

Update compile stats

Update compile Stats or new plan

Execution and Runtime Stats

Stats Interval	Timeframe for aggregation of stats for plan
Flush Interval	How often do we persist to disk



insert exec stats

Plan in store executed for new interval

runtime_stats

"start" a new stats interval = **Default 60 mins**

runtime_stats tables

update exec stats

runtime_stats_interval

Async persist to tables on flush interval = **Default 15 mins**

Plan in store executed in same interval

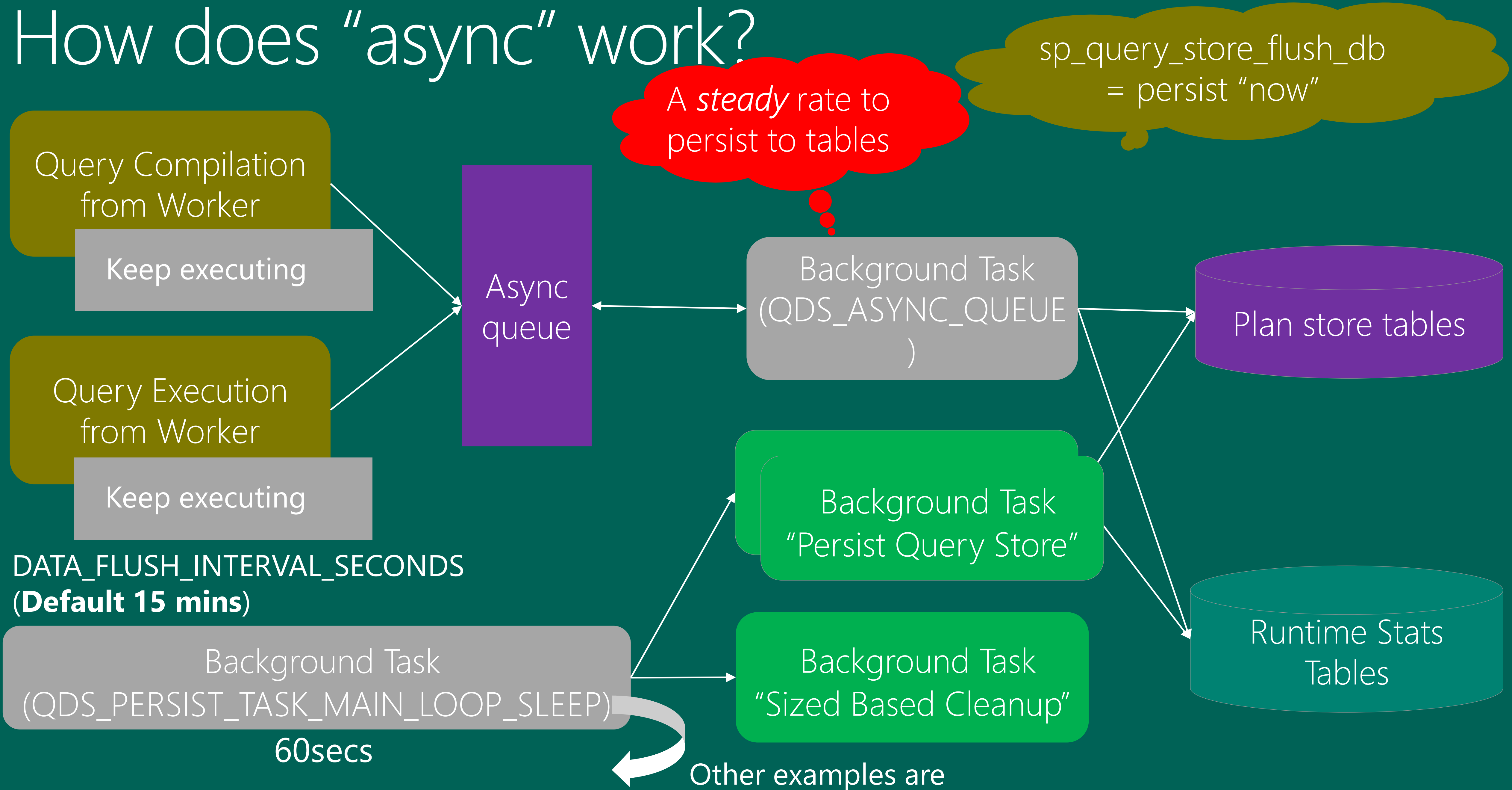
update last execution time

plan store

plan store tables

update last execution time

How does "async" work?



sp_query_store_flush_db
= persist "now"
A *steady* rate to persist to tables
Query Compilation from Worker
Keep executing
Query Execution from Worker
Keep executing
Async queue
Background Task (QDS_ASYNC_QUEUE)
Plan store tables
Background Task "Persist Query Store"
Background Task "Sized Based Cleanup"
Runtime Stats Tables
Background Task (QDS_PERSIST_TASK_MAIN_LOOP_SLEEP)
60secs

DATA_FLUSH_INTERVAL_SECONDS (**Default 15 mins**)

Other examples are

Time based cleanup/change read-write

Query Store Memory Structures

Memory Clerk/Object	Description
MEMORYCLERK_QUERYDISKSTORE_HASHMAP	• Hash table of queries and plans for instance/node • Largest memory consumer • Uses MEMOBJ_QUERYDISKSTORE (NUMA enabled) • Repopulated from disk at startup
MEMORYCLERK_QUERYDISKSTORE	• General clerk for overall Query Store for instance • Should be fairly fixed in size and small • Uses MEMOBJ_QUERYDISKSTORE (NUMA enabled)
MEMOBJ_QUERYSTOREPARTITIONEDHEAP	• CPU partitioned heap for execution stats for instance
USERSTORE_QDSSTMT	• Temp buffers to store statements before persisted
CACHESTORE_QDSCONTEXTSETTINGS	• Track unique context settings across all queries
CACHESTORE_QDSRUNTIMESTATS	• Cache of aggregated runtime stats before persisted

Query Store Maintenance

sys.database_query_store_options

readonly_reason

Enabling, Clearing, and State

ON = Enable; **OFF** = Disabled (existing state and data kept); **CLEAR** = TRUNCATE tables

READ_WRITE = Default when ON; **READ_ONLY** = intentional or *problem* (desired != actual)

Size, limits, and retention

Default **max size** = 100Mb (limited by overall database size). If you hit max, state = READ_ONLY

Default **max plans per query** = 200 (this is silent but its well 200!...)

Default **days queries kept in store** = 30 days

Capture and cleanup efficiently

Query capture mode of AUTO (Default is ALL) = execution count and resource consumption

Sized based cleanup of AUTO (Default is AUTO) = Remove oldest and least expensive

Maintenance at a deeper level

sp_query_store_remove_plan = delete specific plan and associated runtime stats

sp_query_store_remove_query = delete query, associated query text, plans and runtime stats

sp_query_store_reset_exec_stats = Delete runtime stats for specific plan

Considerations

- Use ALTER to modify objects or “duplicate” queries will be tracked
- Forcing plans is better than Plan Guides

  Force by a simple id\You can change the text of a procedure
- There is a cost (ad-hoc and In-Memory workloads) -> use Auto as Capture Mode
- stmt_sql_handle matches statement not batch

  Join with dm_exec_query_stats
- Encrypted procs hides text as with DMVs or catalog views

Final Comments

It is a *rich persisted* database of *query execution over time* information for SQL Server and Azure

It is the *central starting point* for Query Performance Tuning and Troubleshooting

Understanding how it works can help you more effectively use it

Monitor and tune configuration based on your application needs

Query Live Statistics

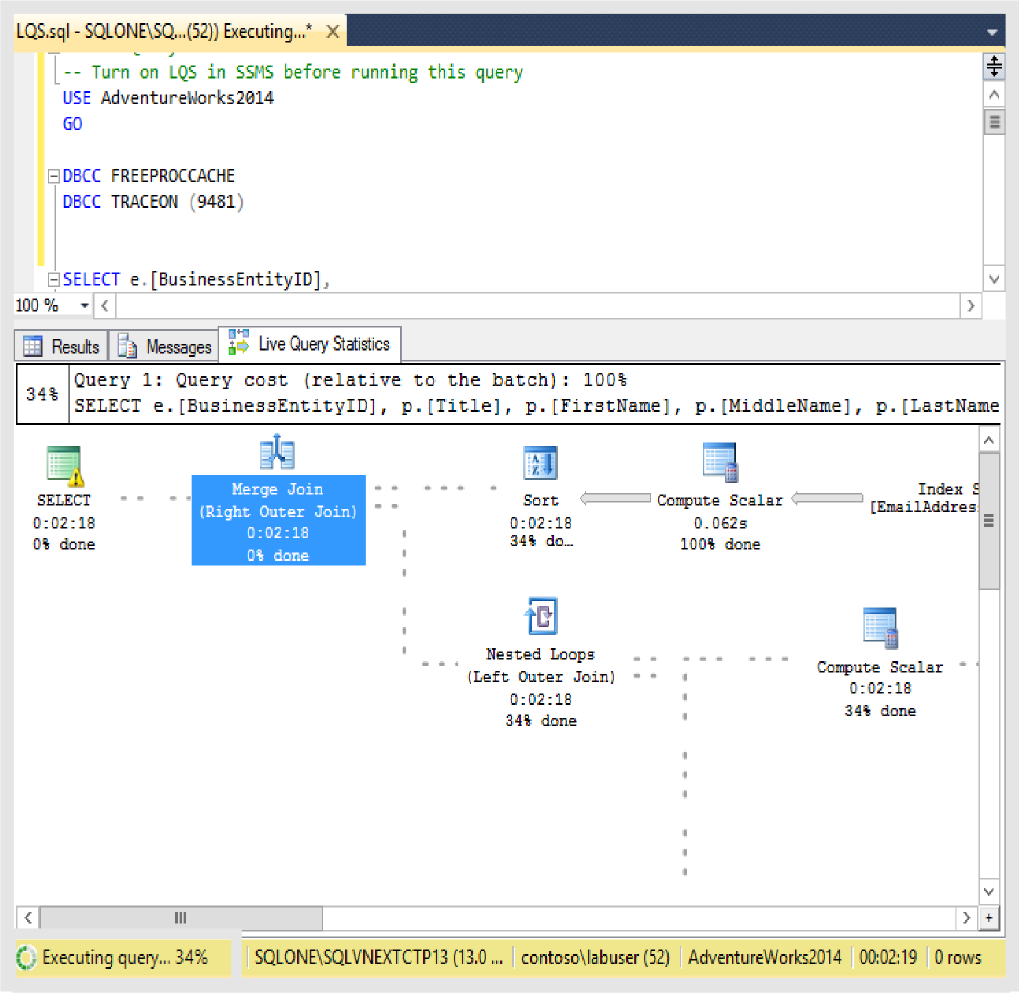
Live Stats

- Ability to view the live execution plan of an active query.
- This live query plan provides real-time insights into the query execution process as the controls flow from one query plan operator to another

Limitations

- Columnstore indexes are not supported
- Memory-optimized tables are not supported
- Natively compiled stored procedures are not supported

Live Query Statistics



LQS.sql - SQLONE\SQ...(52)) Executing...*

```
-- Turn on LQS in SSMS before running this query
USE AdventureWorks2014
GO

DBCC FREEPROCCACHE
DBCC TRACEON (9481)


SELECT e.[BusinessEntityID],
```

100 %

Results | Messages | Live Query Statistics

34%
Query 1: Query cost (relative to the batch): 100%
SELECT e.[BusinessEntityID], p.[Title], p.[FirstName], p.[MiddleName], p.[LastName

Executing query... 34% | SQLONE\SQLVNEXTCTP13 (13.0 ... | contoso\labuser (52) | AdventureWorks2014 | 00:02:19 | 0 rows

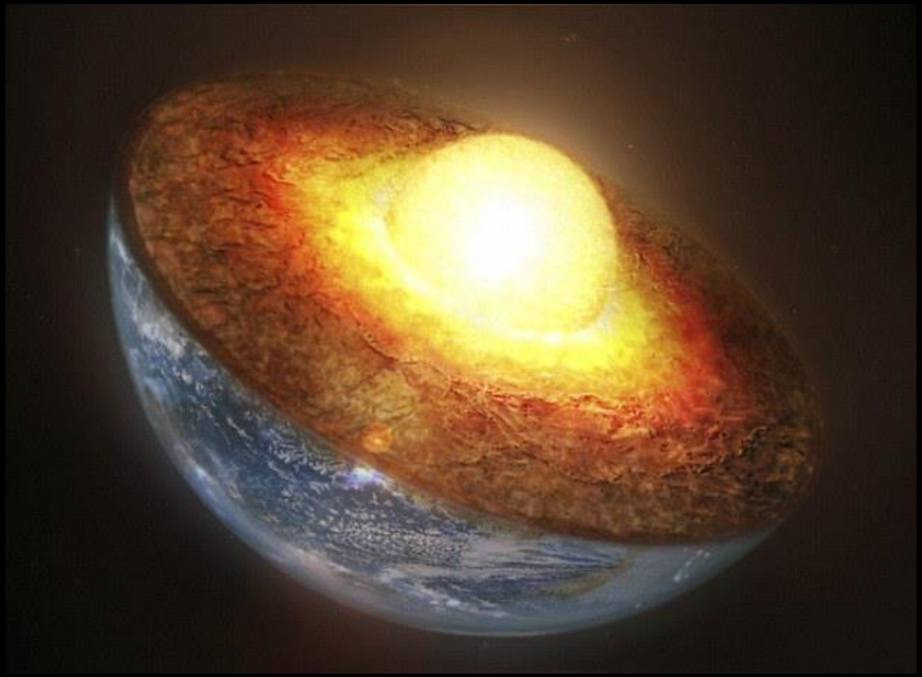
View CPU/memory usage, execution time, query progress, and more

Enables rapid identification of potential bottlenecks for troubleshooting query performance issues

Allows drill down to live operator level statistics:

- Number of generated rows
- Elapsed time
- Operator progress
- Live warnings

Changes in core engine



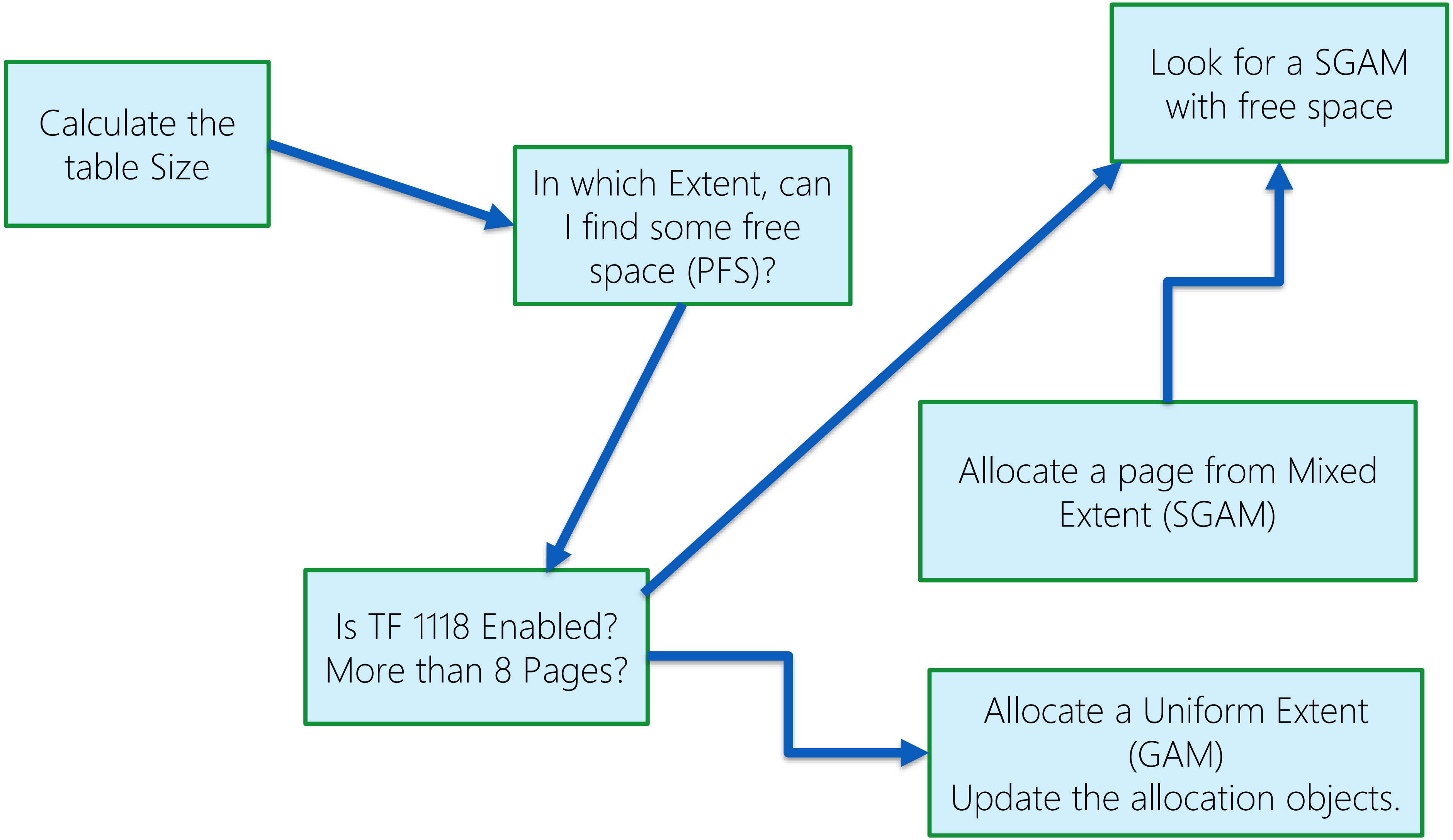
TempDB Allocation Bottlenecks

TempDB has more frequent allocation and de-allocation of objects than user database

Object allocation in TempDB can become a bottleneck. To alleviate this bottleneck:

- Use multiple TempDB files
- Size TempDB appropriately
- Make data files the same size
- Place TempDB on separate disks

In a high level!



Calculate the table Size

In which Extent, can I find some free space (PFS)?

Is TF 1118 Enabled? More than 8 Pages?

Look for a SGAM with free space

Allocate a page from Mixed Extent (SGAM)

Allocate a Uniform Extent (GAM)
Update the allocation objects.

TEMPDB

Best practices:

- New install creates multiple files by default (TEMPDB Tab)
- Fix for metadata contention (SH instead of EX latch)
- Uniform extent allocation (-T1118) on by default
- Grow all files at same time (-T1117) on by default

To disable the traceflag behavior

Syntax:

- (1118)ALTER DATABASE <dbname> SET MIXED_PAGE_ALLOCATION { ON | OFF }
- (1117)ALTER DATABASE <dbname> MODIFY FILEGROUP <filegroup> { AUTOGROW_ALL_FILES | AUTOGROW_SINGLE_FILE }

DBCC

- MULTI_OBJECT_SCANNER waits while running DBCC CHECKS

(checkdb, checktable, …)

- Internally DBCC CHECK uses a page scanning coordinator design (MultiObjectScanner).

SQL Server 2016 changes the internal design (CheckScanner)

- Applying no lock semantics and a design similar to those used with In-Memory Optimized (Hekaton) objects.
- Allowing DBCC operations to scale far better than previous releases.
- Lockless implementation
- Sustained scalability to 64 CPUs

Log Writer

- SQL Server 2016 extended the log writer by allowing up to 4 workers for the instance
- The number of log writers created during the SQL Server instance startup depends on the number of hardware

  NUMA nodes present on the system (up to 4).

Indirect Checkpoint

- Checkpoint: Flushes dirty pages (in-memory modified) from buffer pool to disk.

Frequency of checkpoints varies based on database activity and recovery interval. Database engine issues checkpoints automatically based on the server level “recovery interval” configuration option

Indirect Checkpoints -> Default in SQL 2016

- Changed algorithm:

Based on number of dirty pages, not number of transactions

More predictable database recovery time. Database engine issues checkpoints automatically based on the database level TARGET_RECOVERY_TIME

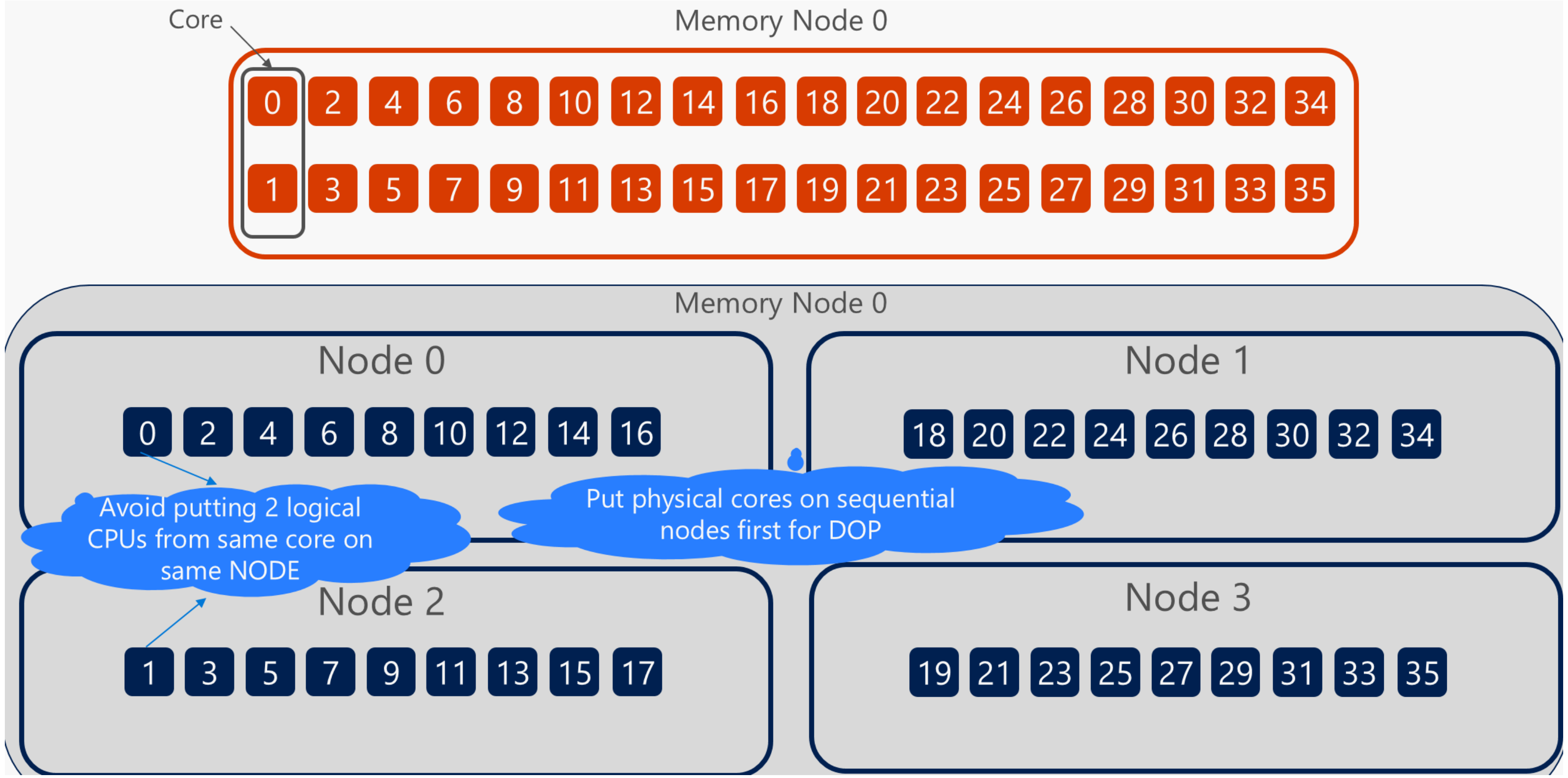
Automatic Soft Numa

The Answer…. Partition NUMA Nodes = Soft NUMA

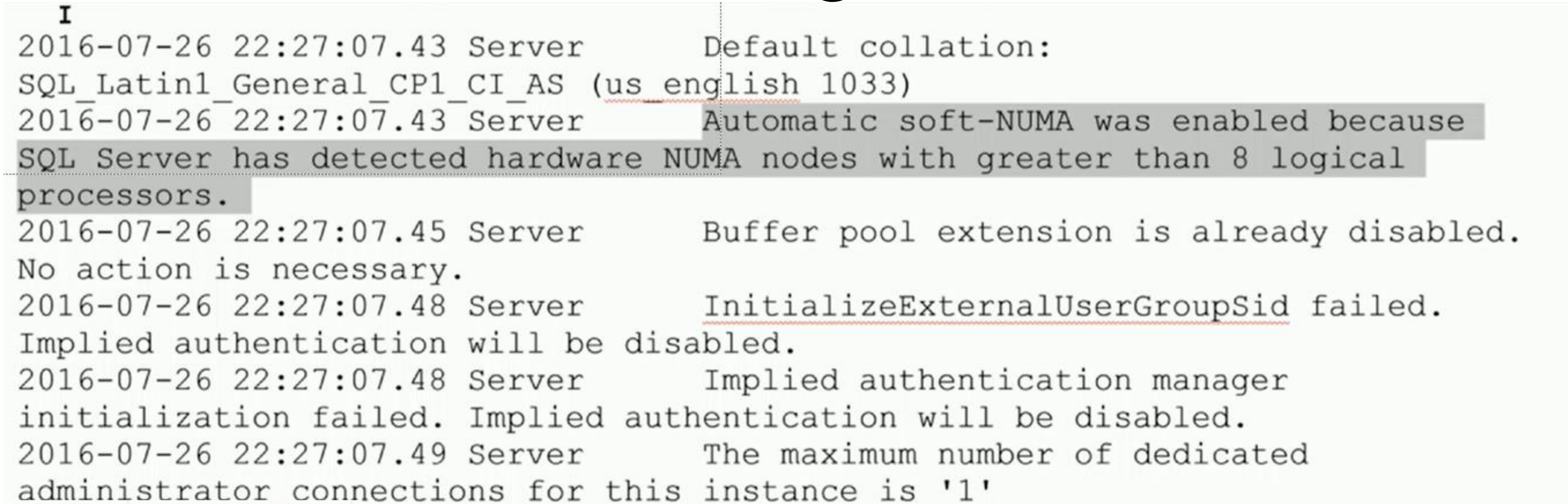
Split up HW NUMA nodes when we detect > *8 physical processors per NUMA node*

On by default in 2016 (Change with ALTER SERVER CONFIGURATION)

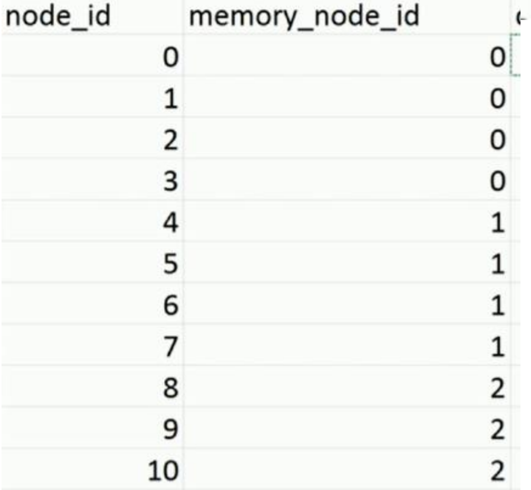
Code in engine that benefits from NUMA partitioning gets a boost



Soft Numa node – SQL Log



```
2016-07-26 22:27:07.43 Server      Default collation:
SQL_Latin1_General_CP1_CI_AS (us_english 1033)
2016-07-26 22:27:07.43 Server      Automatic soft-NUMA was enabled because
SQL Server has detected hardware NUMA nodes with greater than 8 logical
processors.
2016-07-26 22:27:07.45 Server      Buffer pool extension is already disabled.
No action is necessary.
2016-07-26 22:27:07.48 Server      InitializeExternalUserGroupSid failed.
Implied authentication will be disabled.
2016-07-26 22:27:07.48 Server      Implied authentication manager
initialization failed. Implied authentication will be disabled.
2016-07-26 22:27:07.49 Server      The maximum number of dedicated
administrator connections for this instance is '1'
```



node_id	memory_node_id
0	0
1	0
2	0
3	0
4	1
5	1
6	1
7	1
8	2
9	2
10	2

Sys.dm_os_nodes

CMenthread

- Memory Objects = "Heaps" in SQL Server

Most of these are "global" or "thread safe"
When any thread is waiting on another for access to allocate, waittype = CMEMTHREAD
Some waits with small average wait time is normal
Infrastructure allows for ***partitioning*** by NODE or CPU during creation. Requires more memory

- CMEMTHREAD Waits Over the Years

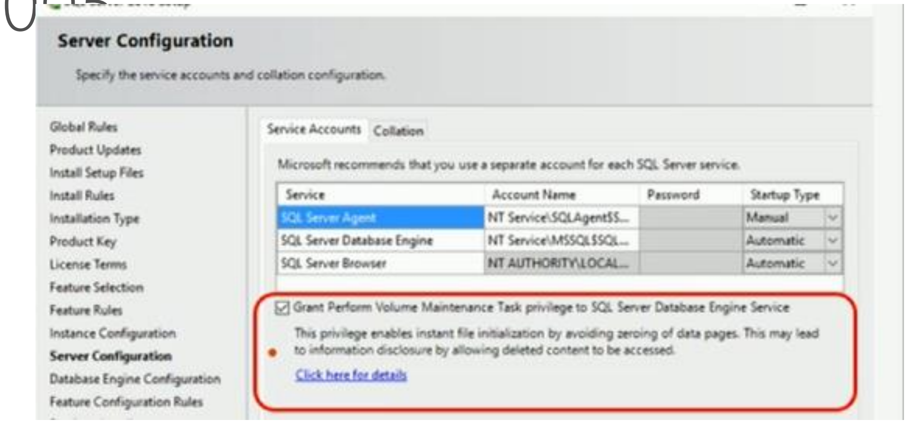
Larger SMP and NUMA systems allow more threads to allocate so this can become a bottleneck
Over the years we discover a "hot" memory object and change partition creation to NODE (rare cases CPU)
-T8048 introduced to change a NODE partitioned object to CPU partitioned

Instant File Initilization

- This has been around since SQL Server 2005

We initialize data files when creating a database

Speed to create database largely = speed of writing to disk



Server Configuration

Specify the service accounts and collation configuration.

Microsoft recommends that you use a separate account for each SQL Server service.

Service	Account Name	Password	Startup Type
SQL Server Agent	NT Service\SQLAgent$S...		Manual
SQL Server Database Engine	NT Service\MSSQL$SQL...		Automatic
SQL Server Browser	NT AUTHORITY\LOCAL...		Automatic

☑ Grant Perform Volume Maintenance Task privilege to SQL Server Database Engine Service

This privilege enables instant file initialization by avoiding zeroing of data pages. This may lead to information disclosure by allowing deleted content to be accessed.

Click here for details

- Along comes a Faster Way

Windows introduces the SetFileValidData API

Give a length and "Your Good!"

CREATE DATABASE is now "just faster" by factors of 200%+

Creating the file for the database is *almost* the same speed regardless of size

- I don't care how slow CREATE DATABASE is

But you do for autogrow or adding a file

- What's the Catch?

You must have the Performance Volume Maintenance Tasks privilege

Anyone with this privilege can see the bytes on disk for the length you specify

Anyone else will only see 0s

SOS RWLock

- Spinlock contention because it increase the number of CPUs
- So, now the spinlocks for readers does not block the other readers.
- For “reader” scenarios, less collisions, lower CPU, better throughput

- Core Synchronization Primitive used in the Engine
  - Used by various places in the code to implement multiple readers and a single writer
  - Not visible as a wait_type. You will see some other wait_type (Ex. COMMIT_TABLE)
  - Uses built-in SOS “Events” to wait

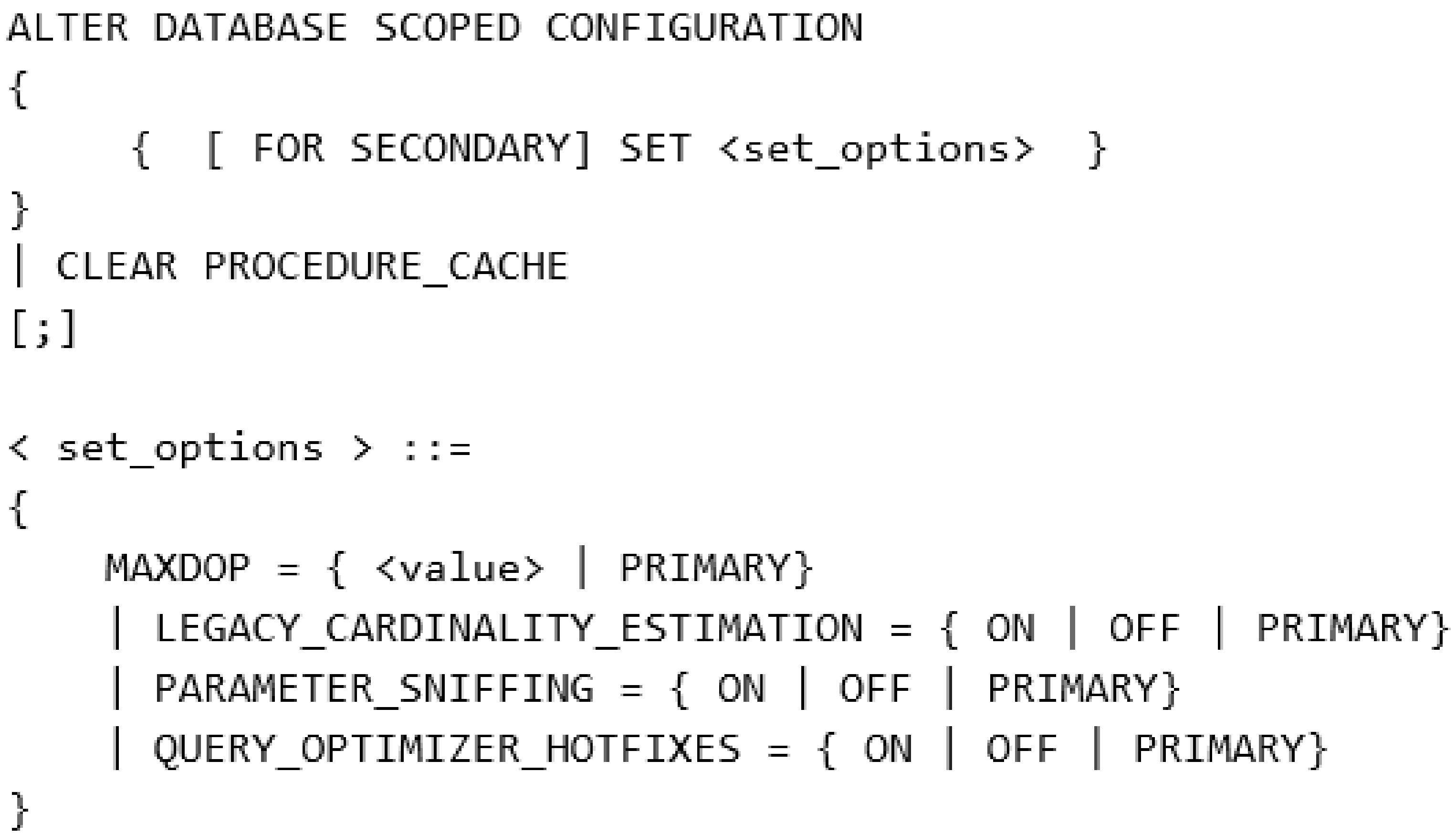
https://blogs.msdn.microsoft.com/bobsql/2016/07/23/how-it-works-reader-writer-synchronization/

Trace flag 4199 hotfixes made to previous releases will be enabled under compatibility level 130

Trace flag 4199 will be used to release any future hotfixes for databases under compatibility level 130

[SQL Server query optimizer hotfix trace flag 4199 servicing model (974006)]()

Database Scoped Configurations



```
ALTER DATABASE SCOPED CONFIGURATION
{
     {  [ FOR SECONDARY] SET <set_options>  }
}
| CLEAR PROCEDURE_CACHE
[;]

< set_options > ::=
{
    MAXDOP = { <value> | PRIMARY}
    | LEGACY_CARDINALITY_ESTIMATION = { ON | OFF | PRIMARY}
    | PARAMETER_SNIFFING = { ON | OFF | PRIMARY}
    | QUERY_OPTIMIZER_HOTFIXES = { ON | OFF | PRIMARY}
}
```

Updated Scheduling Algorithms

- A large and a short CPU quantum worker can receive unbalanced access to scheduling resources
- Previous SQL Server versions rely only on load factor and NUMA nodes
- SQL Server 2016 monitors the quantum usage patterns allowing workers to get fair treatment

High Performance Workloads

Prior to SQL Server 2016, most workloads require some tuning effort:

- Activate Instant File Initialization
- Adjust tempdb configuration
- Turn on trace flags
- Change configuration parameters

Recommended updates and configuration options for SQL Server 2012 and SQL Server 2014 with high-performance workloads (2964518)