Gianluca Hotz

#sqlsat871

SQL Server 2019 CTP 2.5

Sponsors

AGIC Technology

Insight®

Datamaze
DEEP INSIDE YOUR DATA

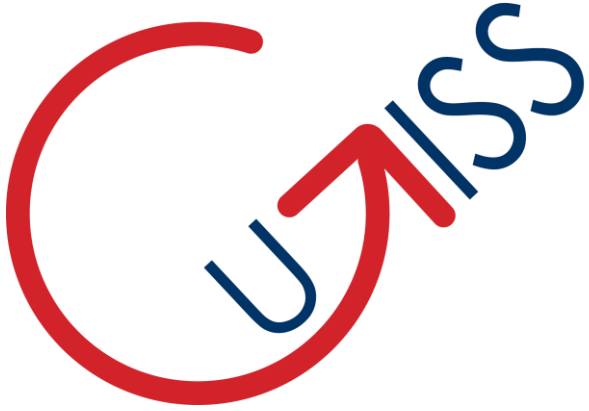
OverNet EDUCATION

Microsoft Azure

SolidQ

#sqlsat871

Organizational Sponsors



www.ugiss.org

www.unica.it

www.pass.org

Who am I?

Gianluca Hotz | @glhotz | ghotz@ugiss.org

Independent Consultant, Founder and Mentor SolidQ

- 20+ years on SQL Server (from 4.21 in 1996)
- Database modeling and development, sizing and administration, upgrade and migration, performance tuning

Interests

- Relational model, DBMS architecture, Security, High Availability and Disaster Recovery

Community

- 20 years Microsoft MVP SQL Server (from 1998)
- Founder and President UGISS
  - User Group Italiano SQL Server (PASS Chapter)

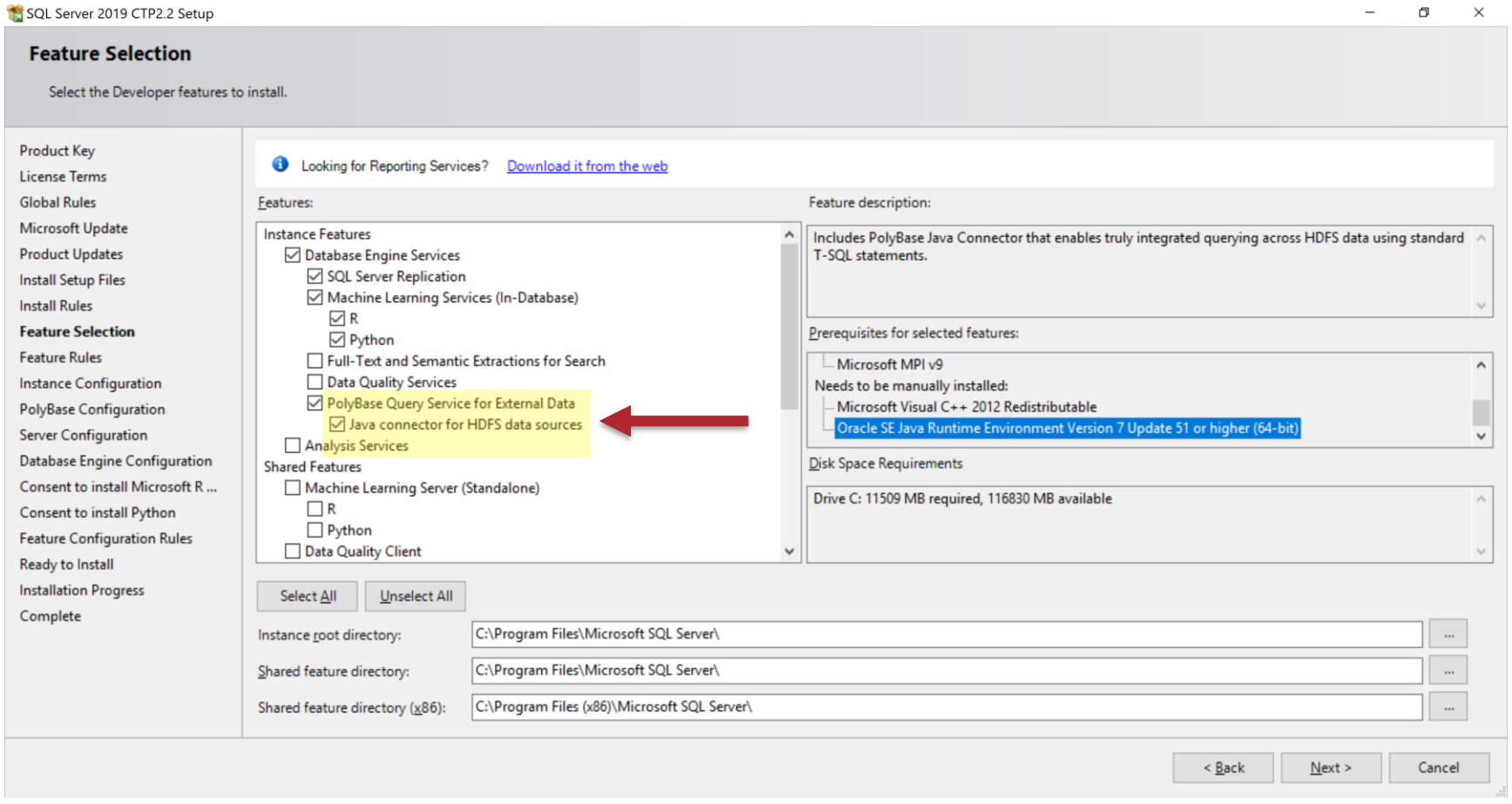
Configuration

Installation

Most new things in 2016/2017

- Separate downloads
- IFI and tempdb configuration
- Linux
- R and Python integration
- Integration Services Scale-out
- Polybase

PolyBase Java Connector for HDFS



SQL Server on Linux

Replication support

- Snapshot, Transactional and Merge

Support for Microsoft Distributed Transaction Coordinator (MSDTC)

Always On Availability Group on Docker containers with Kubernetes

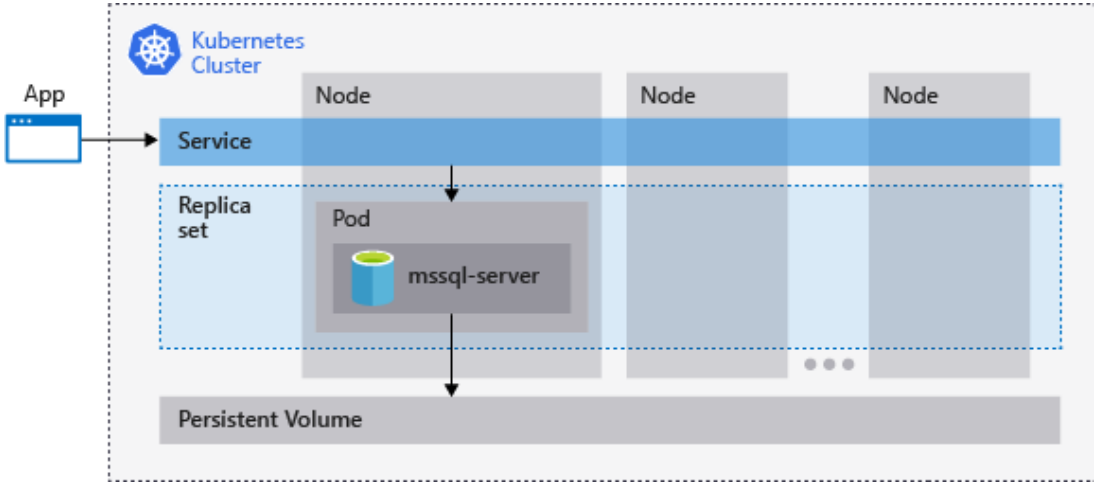
- Kubernetes operator deploys StatefulSet including container with mssql-server container and health monitor

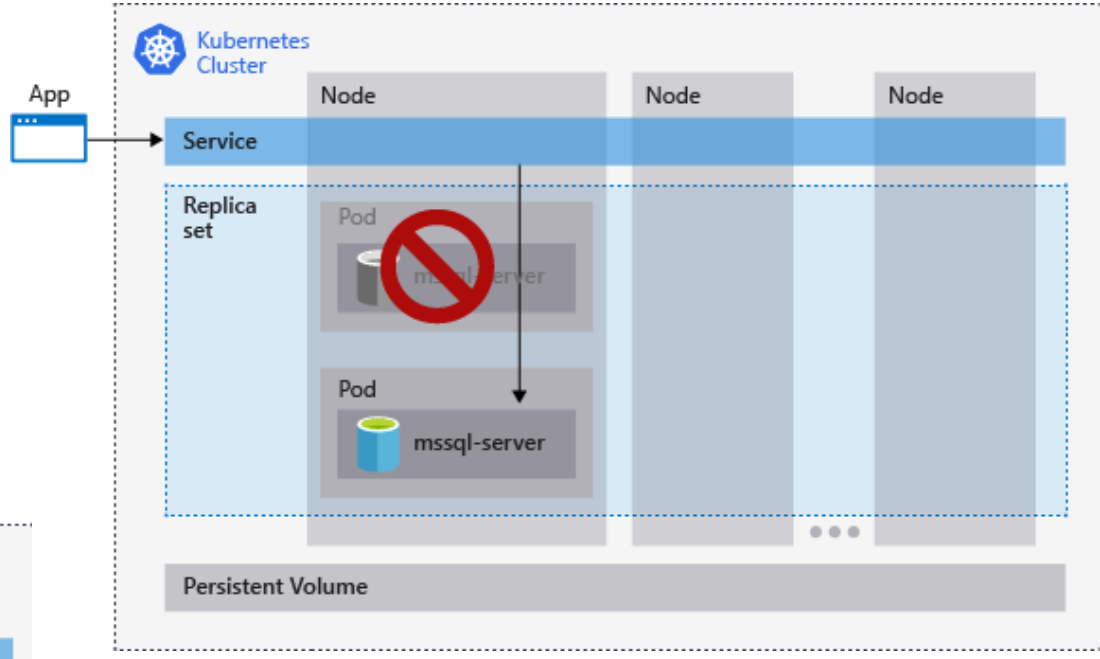
OpenLDAP support for third-party AD providers

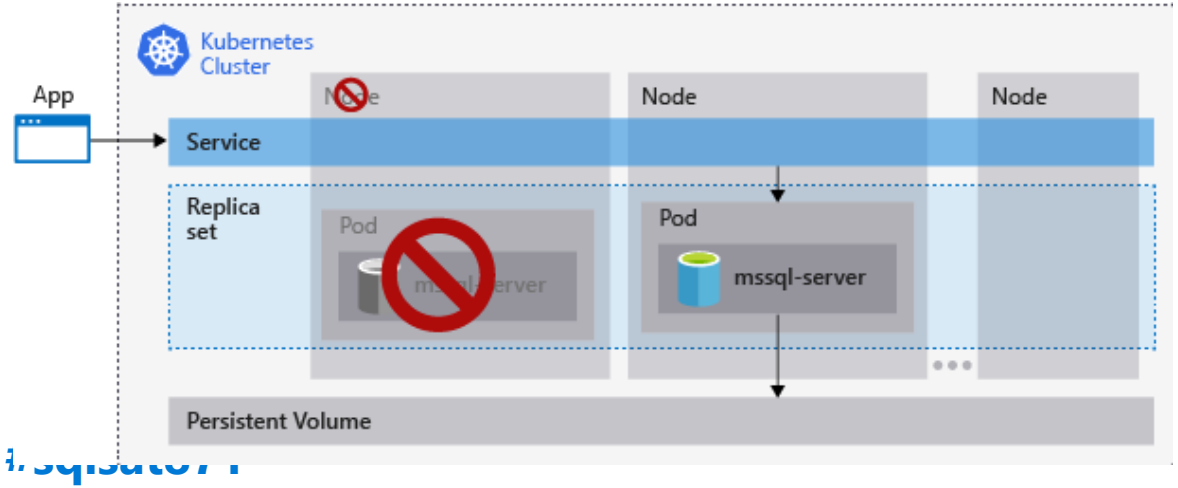
Machine Learning Services (In-Database) on Linux

New container registry

HA solution in Azure Kubernetes Service







Polybase

Data Virtualization

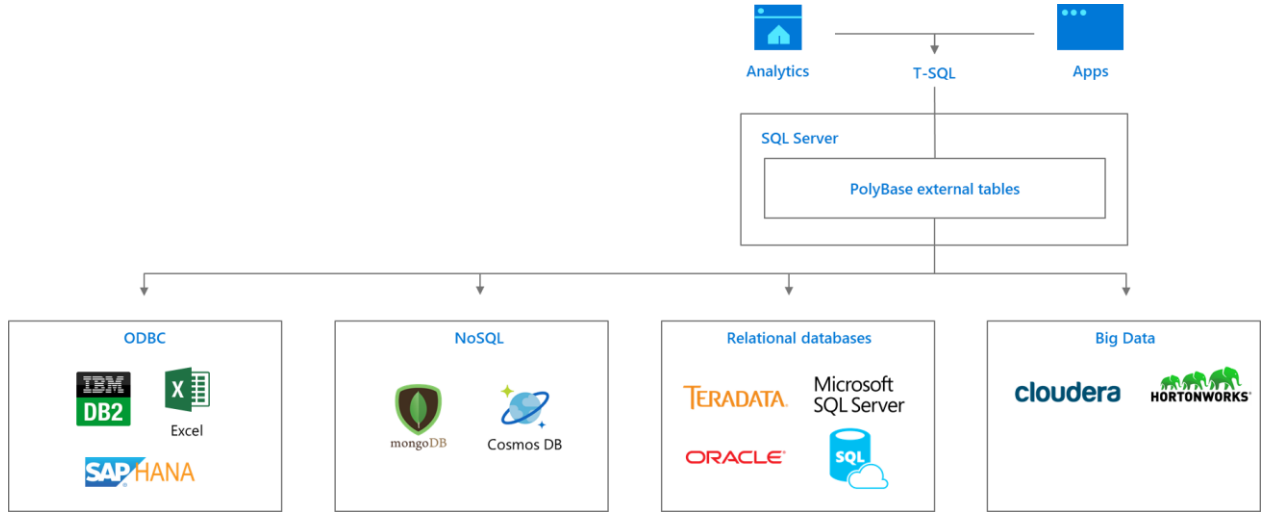
- External Data Source
- External Tables

SQL Server 2016+

- Azure Blob Storage, Hadoop

SQL Server 2019

- SQL Server (all flavors), Oracle, Teradata, MongoDB, CosmosDB (Mongo API)
- ODBC Generic Type (e.g. DB2, SAP Hana, Excel, ...)



External Tables vs Linked Servers

External Tables

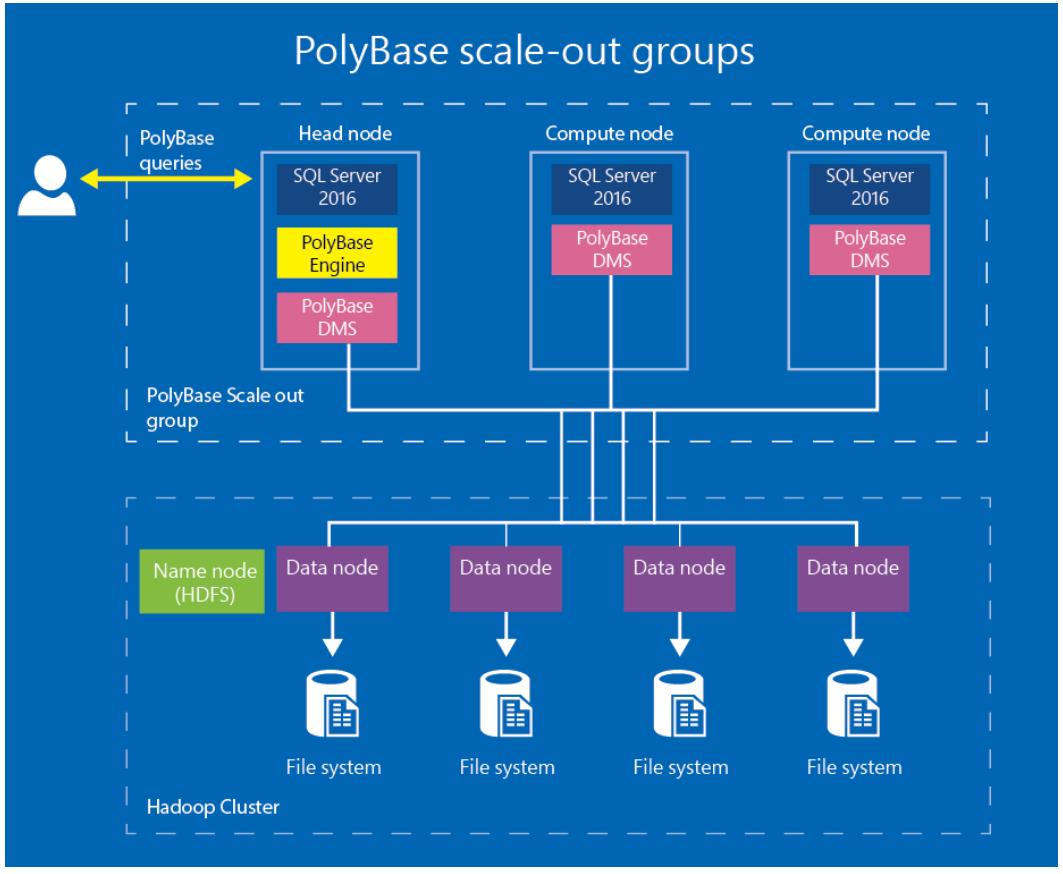
- Database Scoped
- ODBC Drivers
- Read-only*
- Scale out queries with push-down
- Failover with AG
- Basic authentication
- Distributed Transactions not supported

Linked Servers

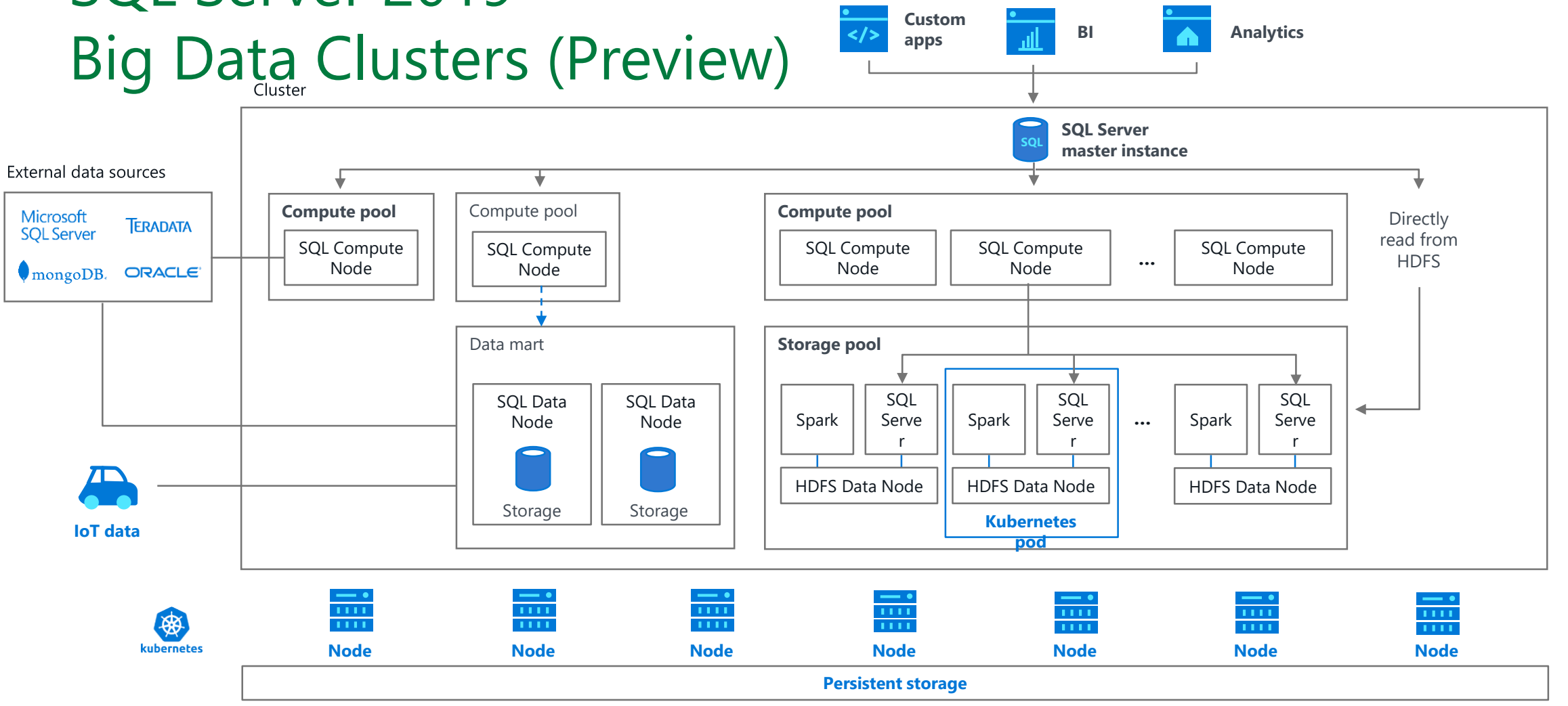
- Instance Scoped
- OLEDB Providers
- Read/write
- Single threaded queries with push-down
- Requires separate config from AG
- Basic and integrated authentication
- Distributed Transactions supported

* Insert into HDFS allowed

Polybase Scale-Out Groups



SQL Server 2019
Big Data Clusters (Preview)



#sqlsat871

Polybase in Big Data Cluster vs Stand-alone

Feature	Big data cluster	Stand alone
Create external data source for SQL Server, Oracle, Teradata, and Mongo DB	X	X
Create external data source using a compatible third-party ODBC Driver		X
Create external data source for HADOOP data source	X	X
Create external data source for Azure Blob Storage	X	X
Create external table on a SQL Server data pool	X	
Create external table on a SQL Server storage pool	X	
Scale-out query execution	X	X

Other Services

Master Data Services

- Silverlight controls replaced with HTML

SQL Server Machine Learning Services

- Windows Server Failover Cluster support
- Partition-Based modeling

Administration

Tools

SQL Server Management Studio V18

Azure Data Studio (was Operations Studio)

Resumable online index operations

Resume after failure (e.g. out of disk space)

Pause and resume later (e.g. free temporarily resources)

Manage large indexes using less log an shorter transactions

Fit rebuild operations into limited maintenance windows

```
REBUILD
WITH (ONLINE = ON, RESUMABLE = ON, MAX_DURATION = 30 MINUTES);
```

SQL Server 2017+ Resumable Index Rebuild

SQL Server 2019+ Resumable Index Create

Transparent Data Encryption

ALTER DATABASE … SET ENCRYPTION **SUSPEND**

ALTER DATABASE … SET ENCRYPTION **RESUME**

Columnstore Indexes

Online build/rebuild Clustered Columnstore

E.g. conversion from row-store to Columnstore

DBCC CLONEDATABASE

Instantaneous schema-only copy of a database for troubleshooting

- No data, full-schema, statistics and Query Store
- Non-blocking
- Read-only by default (can be changed)
- Optionally **NO_STATISTICS**, **NO_QUERYSTORE**
- SQL Server 2012 SP4, 2014 SP2 CU3, 2016 SP1, 2017

New in SQL Server 2019

- Columnstore Statistics

Accelerated Database Recovery

Benefits

- Fast and consistent database recovery
  - Number/size of active transactions don't impact recovery time
- Instantaneous transaction rollback
  - Active time and number of updates don't impact rollback time
- Aggressive Log Truncation
  - Even with long running transactions, prevents growing out of control

High level

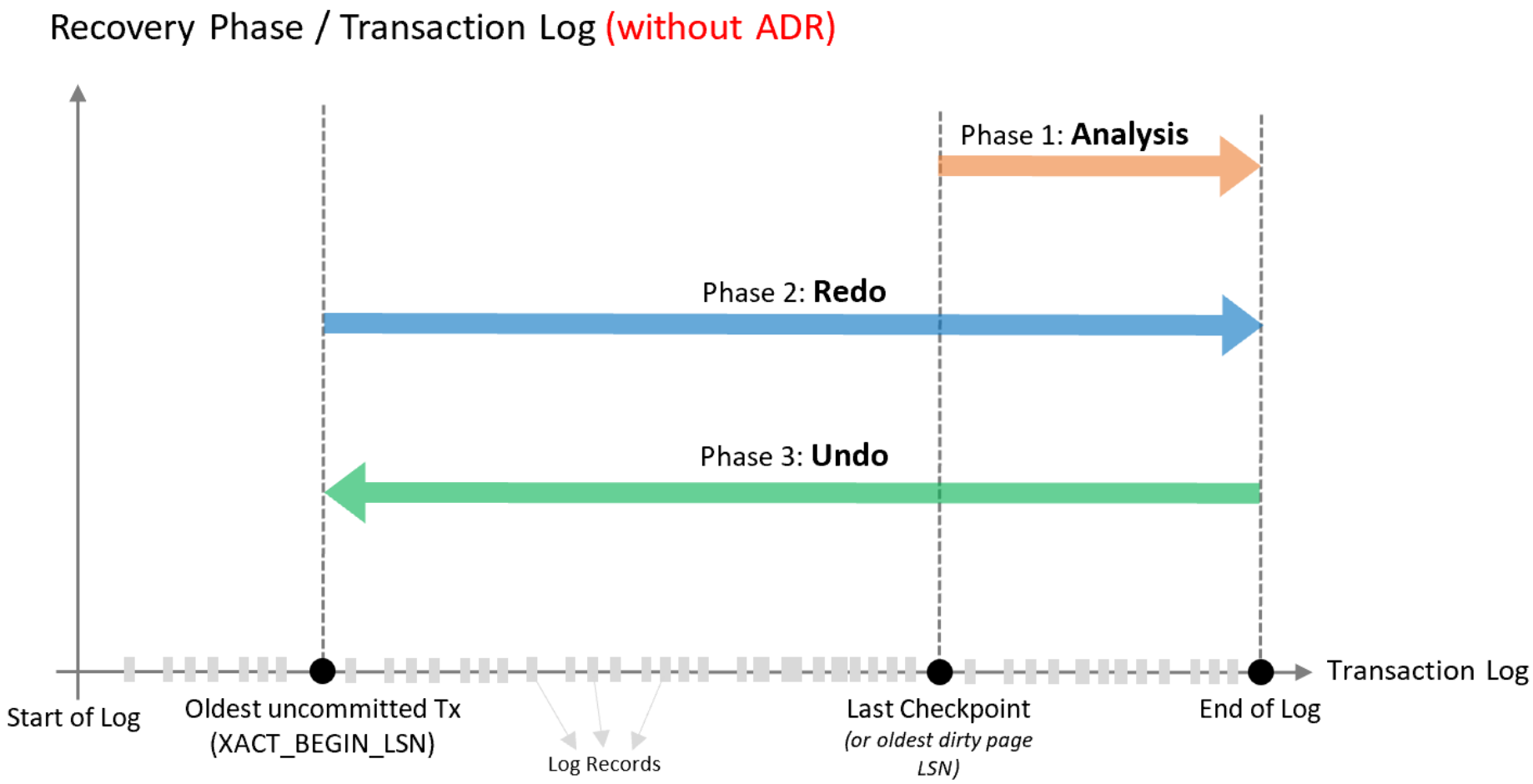
- Versioning all physical database modifications
- Only logical operations undone (are limited and can be undone instantly)
- Active transactions at crash time are marked as aborted
- Any versions generated aborted transactions can be ignored user queries

Currently available in Preview in Azure SQL Database and CTP2.3+!

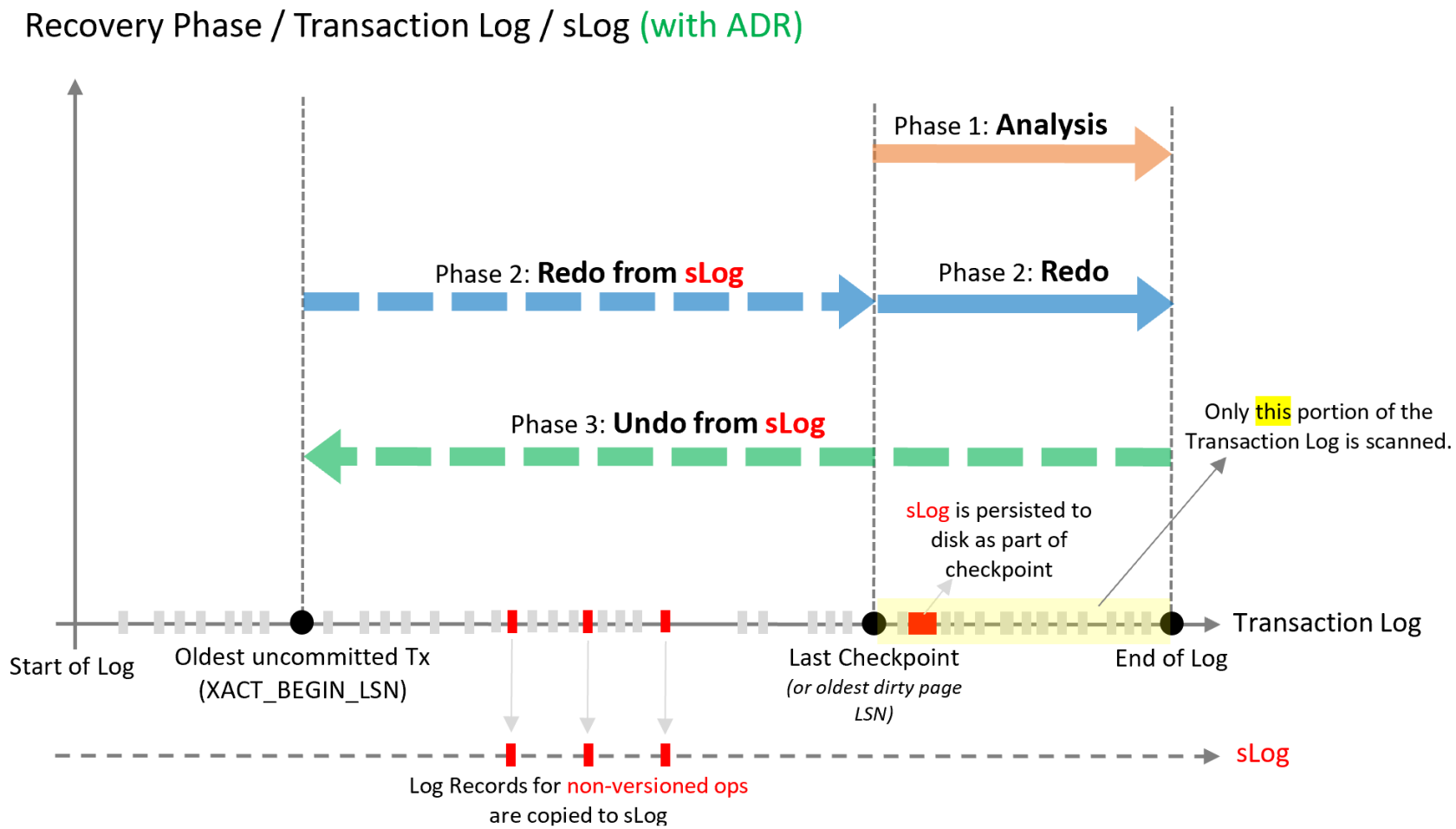
ADR Components

- Persisted Version Store (PVS)
  - New version store, stored in the database instead of tempdb
  - Enable also resource isolation
- Logical Revert
  - Asynchronous process performing row level version based undo
    - Keeps track of all aborted transactions
    - Performs rollback using PVS
    - Releases all locks immediately after transaction abort
- sLog
  - Secondary log stream storing log records for non versioned operations
    - Low volume and in-memory
    - Serialized on disk during CHECKPOINT
    - Enables aggressive transaction log truncation
- Cleaner
  - Asynchronous process that cleans page versions

Current database recovery process



Accelerated database recovery process



Improved diagnostic data for stats blocking

Query Waiting for synchronous update operations

Now **sys.dm_exec_requests** shows **SELECT (STATMAN)**

New **WAIT_ON_SYNC_STATISTICS_REFRESH** wait stat

session_id	request_id	start_time	status	command
59	0	2018-11-13 11:50:36.617	suspended	SELECT (STATMAN)

wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
WAIT_ON_SYNC_STATISTICS_REFRESH	1	18781	18781	0

Availability Groups Enhancements

More synchronous replicas

- SQL Server 2012 4 replicas, 2 synchronous
- SQL Server 2014 8 replicas, 2 synchronous
- SQL Server 2017 8 replicas, 3 synchronous
- SQL Server 2019 8 replicas, **5 synchronous**

Secondary-to-primary read/write redirection

- READ_WRITE_ROUTING_URL and ApplicationIntent=ReadWrite (default)
- Killer feature to replace Listener
  - Cluster technology not offering listener-like features
  - Multi-subnet scenarios too complex to setup/maintain (e.g. Pacemaker)
  - Read scale-out or DR with cluster type NONE

Storage Class Memory / PMEM

Allows low latency I/O

- memory-mapped memcpy-like operations in user mode

SQL Server 2016 SP1

- NVDIMM-N for tail of the log caching

SQL Server 2019

- PMEM devices Linux
- support for data, log In-Memory OLTP checkpoint files placement

Hybrid Buffer Pool

- Clean pages direct referenced on PMEM devices without copy
- Dirty pages still kept in DRAM

Estimating Data Compression savings

sp_estimate_data_compression_savings

- Returns specified object's current size and estimates

SQL Server < 2019

- ROW and PAGE compression

SQL Server 2019+

- Adds **COLUMNSTORE** and **COLUMNSTORE_ARCHIVE** compression
- Object type determines Columnstore type
  - E.g. Heap -> Clustered, Clustered index -> Clustered

Query Profiling Infrastructure

- Run-time information on query execution plans
  - E.g. live query statistics
- Lightweight profiling
  - introduced in SQL Server 2014 SP2, 2016 SP1+
  - 2% expected CPU overhead vs. 75%
  - SQL Server 2019+
    - Enabled by default
    - XEvents **query_post_execution_plan_profile** (vs **query_post_execution_showplan**)
    - DMF **sys.dm_exec_query_plan_stats()** last known ***actual*** execution plan

Internal database pages information

Undocumented DBCC PAGE

New in SQL Server 2019

sys.dm_db_page_info(DatabaseId, FileId, PageId, Mode)
page_resource column in **sys.dm_exec_requests** and **sys.sysprocesses**
sys.fn_PageResCracker(page_resource) to get **db_id, file_id, page_id**

Results Messages

	database_id	file_id	page_id	page_header_version	page_type	page_type_desc	page_type_flag_bits	page_type_flag_bits_desc	page_flag_bits	page_flag_bits_desc
1	2	1	1	1	11	PFS_PAGE	0x0		0x0	
2	2	1	2	1	8	GAM_PAGE	0x0		0x0	
3	2	1	3	1	9	SGAM_PAGE	0x0		0x0	
4	2	1	4	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	2	1	5	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6	2	1	6	1	16	DIFF_MAP_PAGE	0x0		0x200	HAS_CHECKSUM
7	2	1	7	1	17	ML_MAP_PAGE	0x0		0x0	
8	2	1	8	1	1	DATA_PAGE	0x4		0x200	HAS_CHECKSUM
9	2	1	9	1	13	BOOT_PAGE	0x0		0x0	
10	2	1	10	1	10	IAM_PAGE	0x0		0x200	HAS_CHECKSUM
11	2	1	11	1	2	INDEX_PAGE	0x4		0x200	HAS_CHECKSUM

Demo

Compression Savings

Internal Pages Information

Development

Truncation error messages

Error message 8152 too generic

- String or binary data would be truncated

SQL Server 2019 introduces message 2628

- String or binary data would be truncated in **table** '%.*ls', **column** '%.*ls'. Truncated **value**: '%.*ls'

Enabled with Trace Flag 460

- Opt-in required to avoid breaking parsing applications

Extensibility Framework

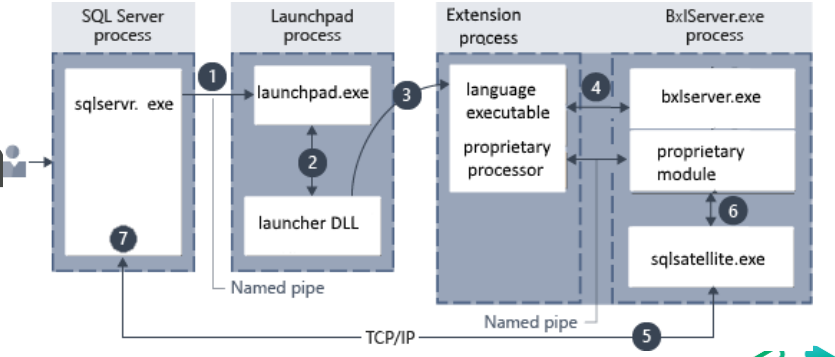
Secure external script execution

Scale/optimization opportunities

SQL Server integration (e.g. store procedures, PREDICT)

Language Support

- SQL Server 2016+ Support for R
- SQL Server 2017+ Support for Python
- SQL Server 2019+ Support for Java



Extensibility Framework Components

Launchpad Service

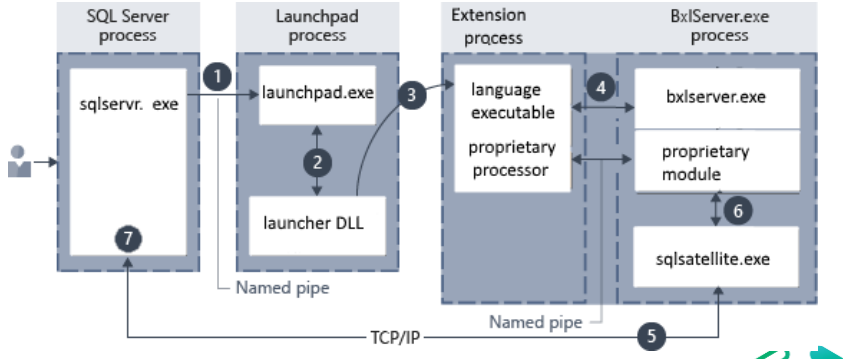
- One per SQL Server instance (with Machine Learning Services)
- Provides security isolation

BxlServer

- Manage communications between SQL Server and external processes
- Binary Exchange Language data format

SQLSatellite

- Extensibility API used by BxlServer
- I/O data/arguments , error handling



Java Language extension

Leverages Extensibility Framework

- Through **sp_execute_external_script**

Current support

- On Windows version 1.10 (JRE 10, JDK 10)
- On Linux version 1.8 (JRE 8, JDK 8)

UTF-8 Support

Full support for import/export, collations, replication, …

- Still not for Linked Servers, In-Memory OLTP, External Table (Polybase)

CHAR and VARCHAR support (Windows collations only)

UTF8 in collation names

- E.g. LATIN1_GENERAL_100_CI_AS_SC_UTF8

Can provide storage savings

- E.g. up to 50% from NCHAR(10) to CHAR(10) with UTF8 vs UTF16

SQL Graph enhancements

Derived tables and view support in MATCH queries

- Set of nodes/edges using UNION ALL
- Useful for heterogeneous entities or connections between them

MATCH support in MERGE

Edge Constraints

- CONNECTION constraint

Demo

Java integration

Security

Certificate Management

Extended in SQL Server Configuration manager

- View and validate certificates installed
- View certificates close to expiration
- Deploy Certificates across machine in Availability Groups
- Deploy Certificates across machine in Failover Cluster Instances

SQL Data Discovery and Classification

SQL Server management Studio Tool (V17.5)

Discovery & Recommendations, Labeling, Reporting

Metadata con be persisted and queried

- Based on Extended Properties
  - sys_information_type_name, sys_sensitivity_label_name

Support for SQL Server 2008+ and Azure SQL Database

SQL Server Sensitivity Classification

SQL Server 2019+

(already available in Azure SQL Database)

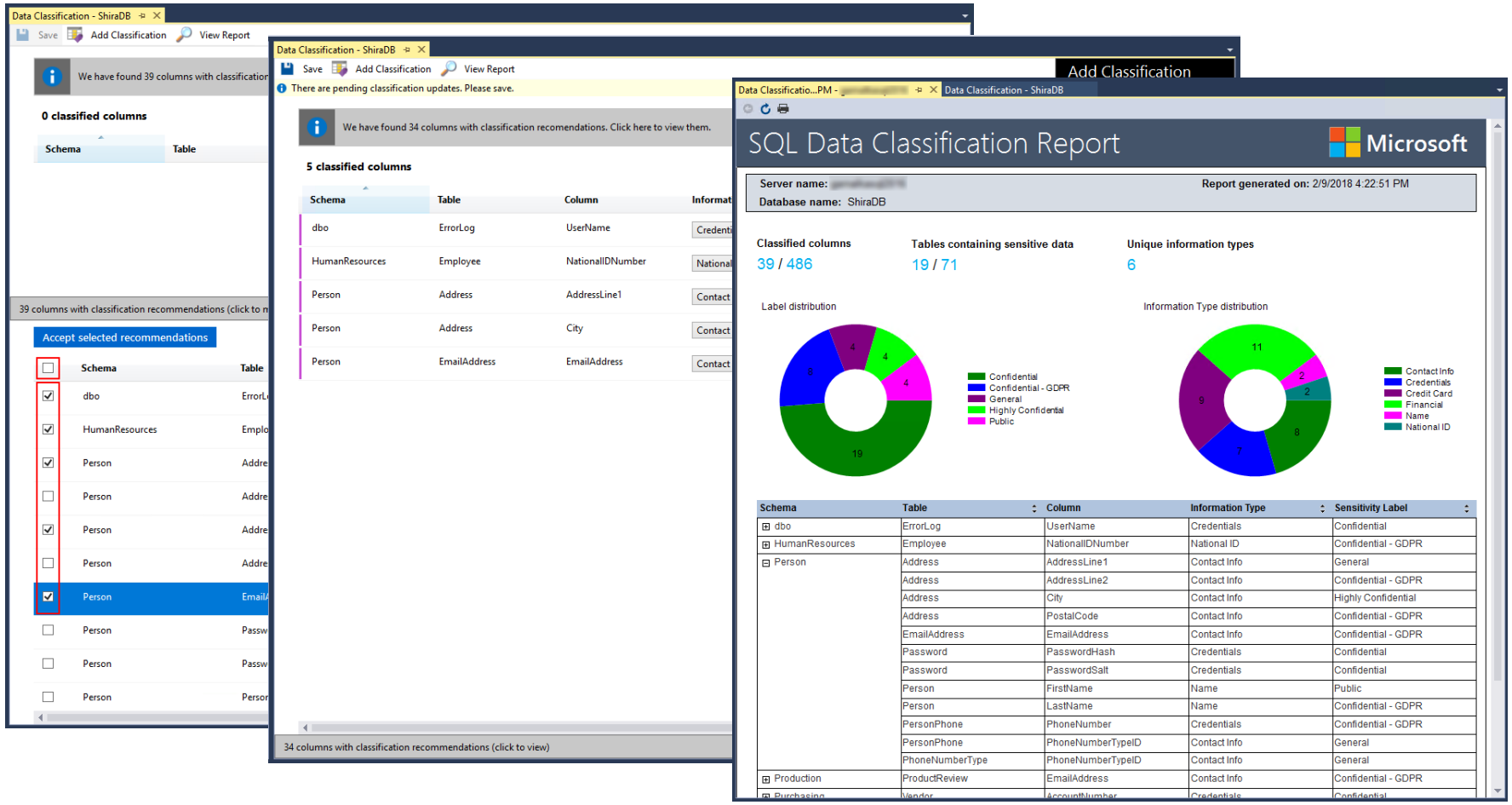
T-SQL command **ADD|DROP SENSITIVITY CLASSIFICATION**

applies to tables, columns
LABEL, **LAtBEL_ID**, **INFORMATION_TYPE**, **INFORMATION_TYPE_ID**

Metadata stored in **sys.sensitivity_classifications**

SQL Server Audit add column **data_sensitivity_information**

Data Discovery and Classification Demo



#sqlsat871

Always Encrypted with Secure Enclaves

Basic architecture as SQL Server 2016+ implementation

Now allows server-side computation on encrypted columns

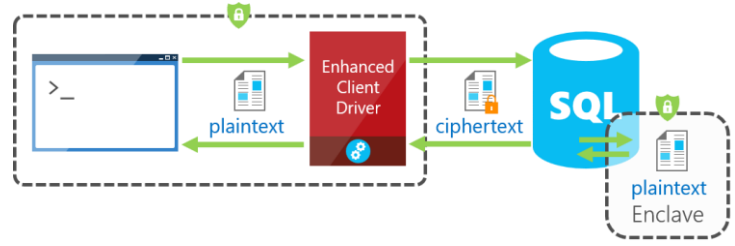
- In-Place Encryption (ALTER TABLE for initial encryption)
- Rich computations (e.g. range comparisons, LIKE predicates, …)

Inside secure enclaves

- Virtualization-based Security (VBS) secure memory enclaves
  - also known as Virtual Secure Mode(VSM) enclaves
- Operation on plaintexts cannot be disclosed outside enclave
- Column Master Keys sent over secure channel by client driver

Still some limitations (no indexing)

Performance optimizations pending…



Data Masking

Dynamic Data Masking	Static Data Masking
On the original database	On a copy of the database
Original data intact	Original data not retrievable
On-the-fly at query time	At storage level
Based on user permissions	Masked for everyone

Static Data Masking

Component of SQL Server Management Studio V18 Preview5+

Define per-column masking configuration

- NULL, Single-Value, Shuffle, Group Shuffle, String Composite
- Can save and load it

It's basically a backup/restore and modify data according to config

No automation yet ☹

Static Data Masking Limitations

No temporal and memory-optimzed tables

No computed and identity columns

No geometry and geography types

Azure SQL Database Hyperscale service tier not supported

Statistics not updated

No cleanup in case of error

- can leave sensitive data copies (backupset)

Data and log files may contain sensitive data

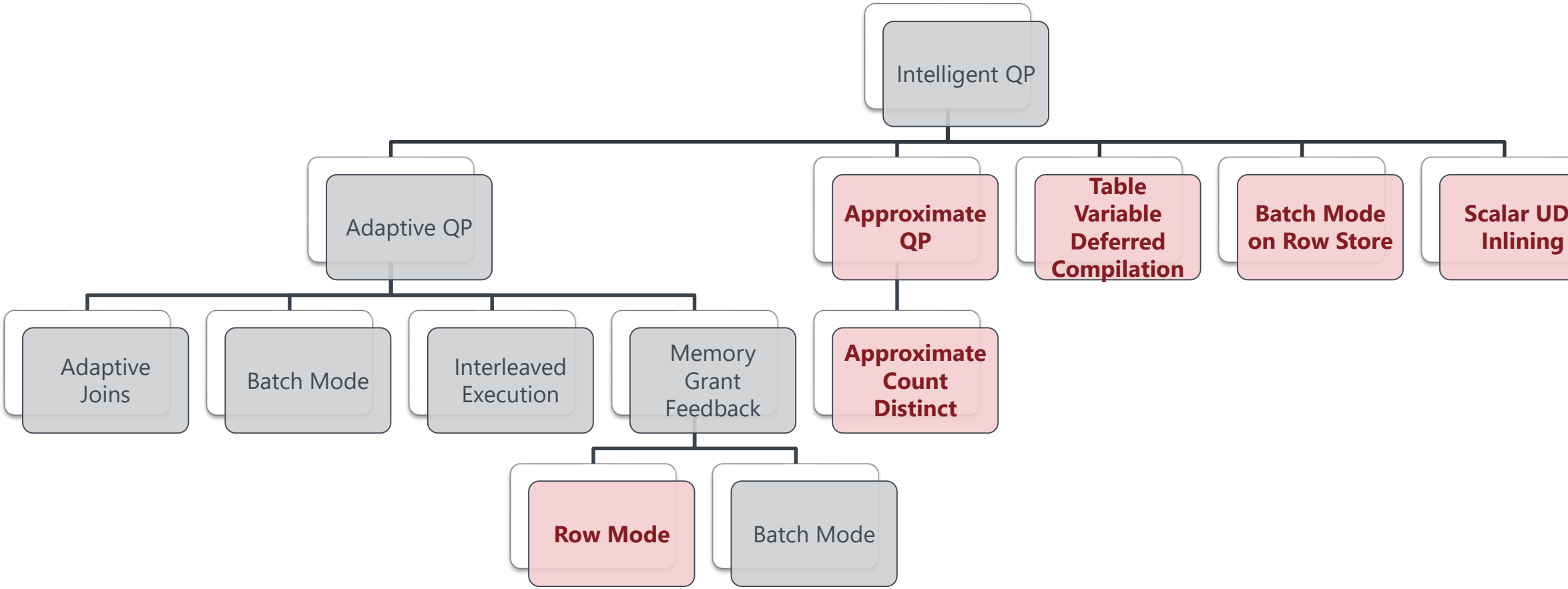
- retrievable with hex editor

Demo

Data Classification

Performance

Intelligent Query Processing



Execution Modes

Row Mode

- Execution tree iterators consume 1 row at a time
- Traditional execution mode for Rowstore

Batch Mode

- Execution tree iterators consume a batch of rows at a time
- Optimal with large scan operations (e.g. large table aggregates or joins)
- SQL Server 2012 introduced to leverage Columnstore Indexes
- SQL Server 2016/2017 extended usage scenarios for CI
- SQL Server 2019 extended usage scenarios to Rowstore

Batch Mode on RowStore

Help reducing CPU Consumption

Columnstore still a better choice

- for OLAP workload that is I/O bound
- can't always create it (e.g. impact on OLTP, features not supported)

Limitations

- In-Memory tables not supported (only heaps & disk-based b-trees)
- Not used when fetching/filtering LOB columns
  - (including sparse columns sets & XML)

Batch Mode on Rowstore Control

SQL Server < 2019

Some scenarios covered with tricks… (article part1, part2, part3)

SQL Server 2019+

Scenarios supported directly by Query Processor

On by default with database compatibility level **150+**

```
ALTER DATABASE SCOPED CONFIGURATION
SET BATCH_MODE_ON_ROWSTORE = ON|OFF
OPTION (USE HINT ('ALLOW_BATCH_MODE'));
OPTION (USE HINT ('DISALLOW_BATCH_MODE'));
```


Memory Grant

Excessive Grant

Too much memory allocated vs. memory used
Impact: blocking, out-of-memory, reduced concurrency

Poor Grant

Not enough memory allocated resulting in data spill to tempdb
Impact: slow query, excessive disk usage (tempdb)

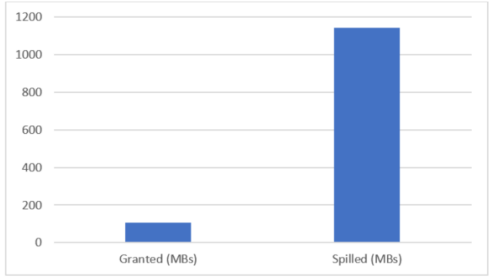
Grant increase

dynamic grants increase allocation too much
impact: server instability, unpredictable performance

Memory Grant Feedback

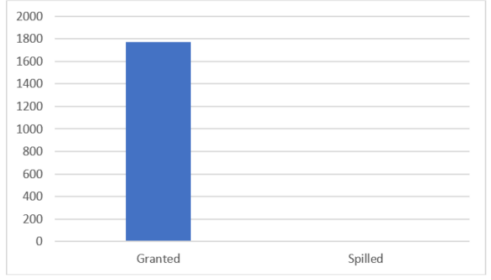
Post-execution evaluation

- Updates grant value for cached plan
- E.g. more memory if spilled, less if excessive grant



Version support

- SQL Server 2017+ Batch Mode
- SQL Server 2019+ Row Mode



Plan caching

- Not persistent (i.e. not save in Query Store)
- OPTION(RECOMPILE) prevents caching and memory grant feedback

Memory Grant Feedback Control

Batch Mode

On by default with database compatibility level **140+**

```
ALTER DATABASE SCOPED CONFIGURATION
SET BATCH_MODE_MEMORY_GRANT_FEEDBACK = ON|OFF
OPTION (USE HINT('DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK'));
```

Row Mode

On by default with database compatibility level **150+**

```
ALTER DATABASE SCOPED CONFIGURATION
SET ROW_MODE_MEMORY_GRANT_FEEDBACK = ON|OFF
OPTION (USE HINT ('DISABLE_ROW_MODE_MEMORY_GRANT_FEEDBACK'));
```


Troubleshooting Memory Grant Feedback

Parameter sensitive scenarios

- Some queries requires different plans with different grants
- Memory grant feedback will disable itself when unstable

Extended Events to monitor changes

- SQL Server 2017+ **memory_grant_feedback_loop_disabled**
- SQL Server 2019+ **memory_grant_updated_by_feedback**

SQL Server 2019+ execution plan attributes

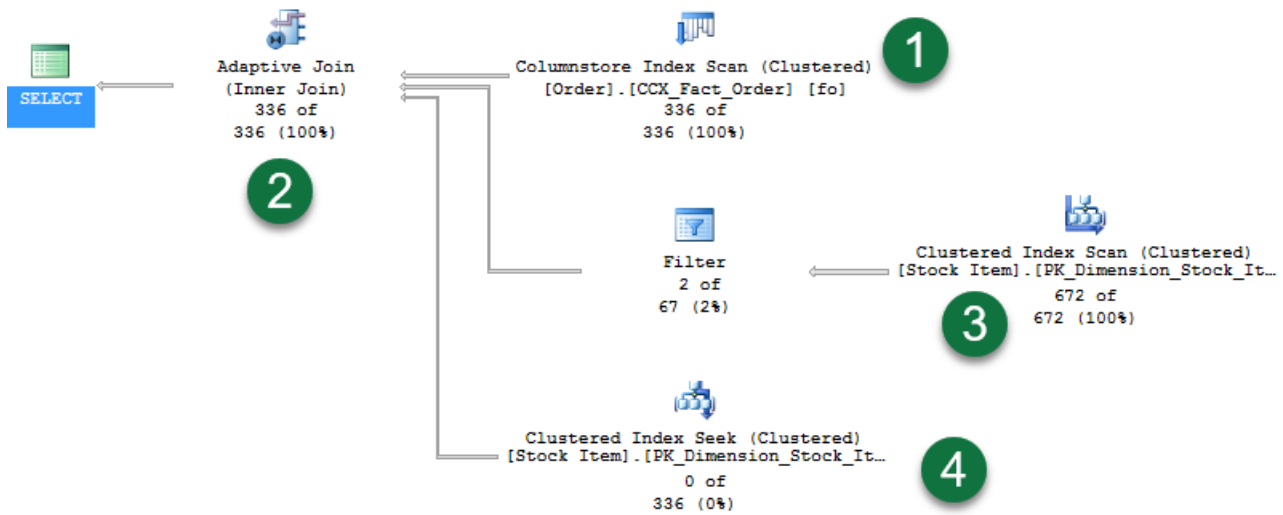
- **IsMemoryGrantFeedbackAdjusted**
  - No: First Execution, Accurate Grant, Feedback disabled
  - Yes: Adjusting, Stable
- **LastRequestedMemory**

Batch mode adaptive joins

Scenario

- Nested loop algorithm better for small build join inputs
- Hash algorithm better for bigger inputs

Adaptive joins defer choice after first input scanned



Interleaved Execution

Problem with multi-statement table valued functions (MSTVFs)

- SQL Server <= 2012 optimize with cardinality = 1
- SQL Server 2014 & 2016 optimize with cardinality = 100

SQL Server >= 2017

- Start optimization
- Pause and executes MSTVFs if candidate
- Resume optimization with correct cardinality

Table Variable vs Temporary Tables

Area	Temporary Tables	Table Variables
Manual statistics creation and update	Yes	No\
Indexes	Yes	Only inline index definitions allowed
Constraints	Yes	Only PRIMARY KEY, UNIQUE and CHECK
Automatic statistics creation	Yes	No
Creating and using a temporary object in a single batch	Compilation of a statement that references a temporary table that doesn't exist is deferred until the first execution of the statement	A statement that references a table variable is compiled along with all other statements before any statement that populates the Table Variable is executed, so compilation sees it as 1

Table Variable Deferred Compilation

Before SQL Server 2019

- Statement referencing TV compiled before population
- Number of row estimate fixed at 1

Starting with SQL Server 2019

- Behaves like Temporary Tables
- Statement referencing non existing TV is deferred until first execution
- Number of row estimate much better

Control

- On by default with database compatibility level **150+**
- ALTER DATABASE SCOPED CONFIGURATION
  SET **DEFERRED_COMPILATION_TV** = ON|OFF
  OPTION (USE HINT ('**DISABLE_DEFERRED_COMPILATION_TV**'));

Scalar UDF inlining

T-SQL user defined functions that returns a single data value

Performance problems

- Iterative invocation
  - once per row, context switching especially with query execution
- Lack of costing
  - before, only relational operators were costed, assumption to be cheap…
- Interpreted execution
  - each statement executes in isolation, no cross-statement optimizations
- Serial execution
  - Intra-query parallelism not allowed

Scalar UDF Automatic inlining

In SQL Server 2019 Scalar UDF automatically transformed into

- Scalar Expressions
- Scalar Subqueries

Optimize the whole plan (UDFs no longer visible)

Control

- On by default with database compatibility level **150+**
- ALTER DATABASE SCOPED CONFIGURATION
  SET **TSQL_SCALAR_UDF_INLINING** = ON|OFF
- OPTION (USE HINT ('**DISABLE_TSQL_SCALAR_UDF_INLINING**'));
- CREATE FUNCTION … WITH **INLINE** = ON | OFF

Scalar UDF inlining example

```
CREATE FUNCTION dbo.discount_price(@price DECIMAL(12,2), @discount DECIMAL(12,2))
RETURNS DECIMAL (12,2) AS BEGIN RETURN @price * (1 - @discount); END

SELECT L_SHIPDATE, O_SHIPPRIORITY
, SUM(dbo.discount_price(L_EXTENDEDPRICE, L_DISCOUNT))
FROM LINEITEM, ORDERS
WHERE O_ORDERKEY = L_ORDERKEY
GROUP BY L_SHIPDATE, O_SHIPPRIORITY
ORDER BY L_SHIPDATE
```

10GB CCI compressed TPC-H Schema, 2 x CPUs (12 cores), 96GB RAM, SSD storage

	Query without UDF	Query with UDF (no inlining)	Query with UDF (inlining)
Execution time	1.6 seconds	29 minutes 11 seconds	1.6 seconds

Scalar UDF inlining requirements

Written using the following constructs
- DECLARE, SET (var declaration/assignments)
- SELECT (single/multiple var assignments)
- IF/ELSE (arbitrary nesting levels)
- RETURN (single or multiple)
- UDF nested/recursive function calls
- Relational operations like EXISTS, ISNULL

No invocation of functions that are
- time-dependent (GETDATE())
- has side effects (NEWSEQUENTIALID())

Uses EXECUTE AS CALLER (default)

No table variables references

No table-valued parameters references

No user-defined types references

Not natively compiled
- interop supported

Not a partition function

Not referenced in
- GROUP BY clauses
- computed columns
- check constraints

No signatures added to it

Scalar UDF inlining troubleshoting

Column **is_inlineable** in **sys.sql_modules**

- Doesn't imply it will always be inlined! (e.g. 1000s lines of code)

Execution Plan

- If inlined successfully, xml node **<UserDefinedFunction>** will be missing

Extended Events

- **tsql_scalar_udf_not_inlineable**

APPROX_COUNT_DISTINCT

Returns **approximate** number of unique non-null values in groups

- HyperLogLog algorithm guarantees
  up to 2% error rate within 97% probability

Fast data exploration with low memory footprint

- E.g. dashboards, trend analysis, feature selection, etc.
- Think 10 billion rows,
  1 user using 1,5GB memory vs 100 users using 12MBs
- Tradeoff: precision, only scenarios where exact values are not necessary!

Demo

Scalar UDF inlining

APPROX_COUNT_DISTINCT

PASS SQLSATURDAY SARDEGNA | 18 MAY 2019

Ask your Questions

#sqlsat871

Thank you very much for attending SQL Saturday Sardegna 2019!