Microsoft

Federal Election Commission

SQL Server Administration Workshop

John Deardurff

John Deardurff

Microsoft Customer Engineer (Global Technical Team)
Microsoft Certified Trainer (Regional Lead)
MVP: Data Platform (2016 – 2018)
Email: John.Deardurff@Microsoft.com
Twitter: @SQLMCT
Website: www.SQLMCT.com
GitHub: github.com\SQLMCT

MVP Microsoft® Most Valuable Professional Reconnect

Microsoft Certified Trainer Microsoft MCT Trainer 2021 - 2022

Certified Professional Microsoft MCSE Data Management and Analytics Earned 2016

Microsoft Certified Azure Database Administrator Associate ★★

Microsoft Certified Azure Data Engineer Associate ★★

Microsoft Certified Azure Solutions Architect Expert ★★★

Agenda Day 1

Module 1: SQL Server Architecture, Scheduling and Waits

- Introduction to the SQLOS
- Thread and Task Scheduling
- Waits and Queues
- CPU and Memory Configuration

Module 2: SQL Server I/O and Database Structure

- Database Files
- Data Page Structures
- Transaction Log File Structures
- SQL Server TempDB File Structure

Agenda Day 2

Module 3: Basic Disaster Recovery

- Full, Differential, and Log Backups
- Native SQL Backup Process
- Restore and Recovery Process

Module 4: Automating SQL Server Management

- SQL Server Agent and Jobs
- Monitoring with Alerts and Notifications.

Agenda Day 3

Module 5: Security - General

- Authentication and Authorization
- SQL Server Logins and Users
- SQL Server Roles and Permissions
- Dynamic Data Masking
- Row Level Security

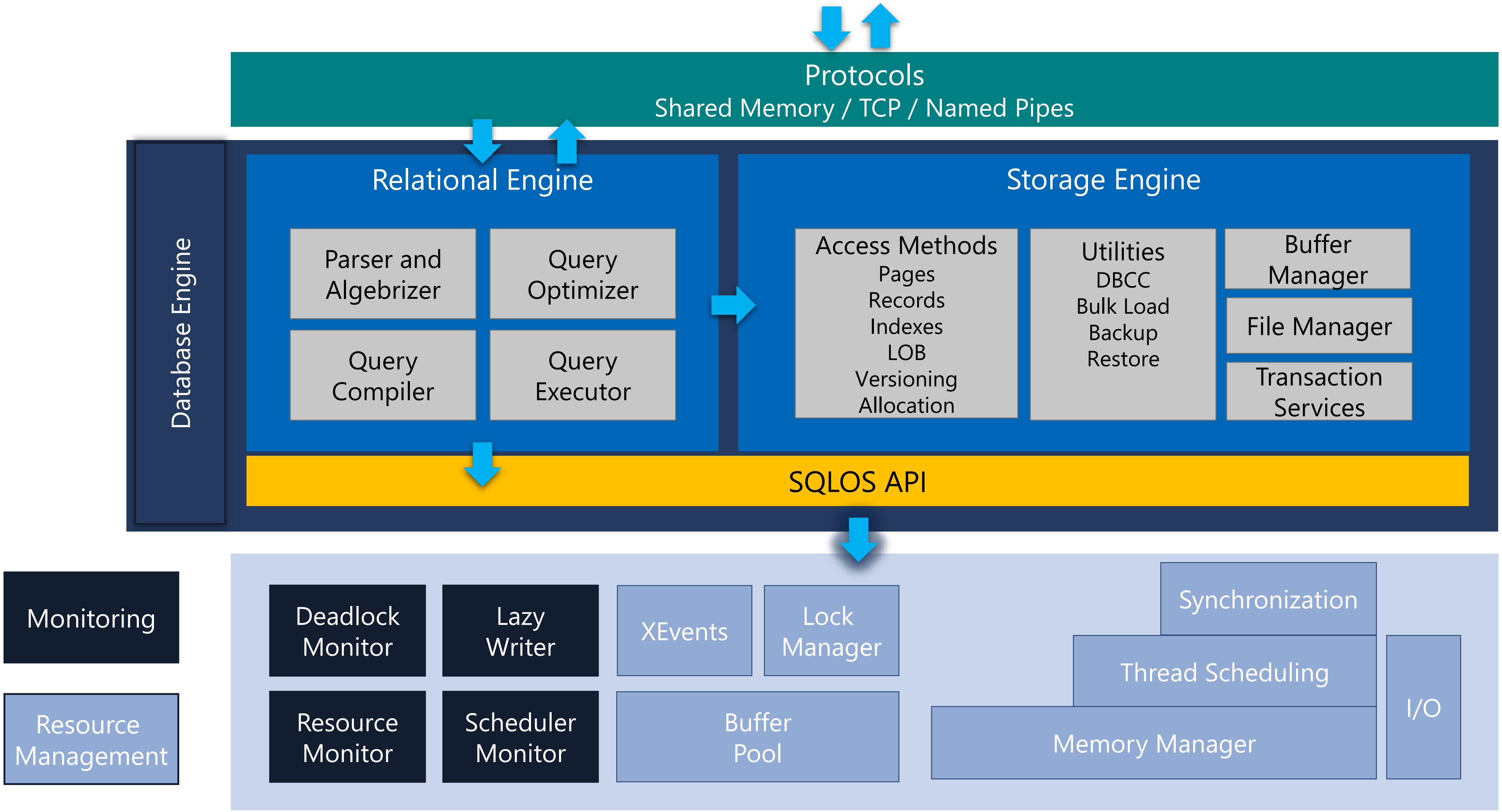
Module 6: Security - Encryption

- Manage Certificates and Keys
- Backup Encryption
- Transparent Data Encryption (TDE)
- Always Encrypted

Module 1: SQL Server Architecture, Scheduling and Waits

Lesson 1: Introduction to SQL Operating System

Inside the Database Engine



SQL Server Operating System (SQLOS)

Application layer between Microsoft SQL Server components and the Windows Operating System.

Centralizes resource allocation to provide more efficient management and accounting.

Currently, the SQLOS is used by the SQL Server relational database engine and Reporting Services for system-level services.

Abstracts the concepts of resource management from components, providing:

- **Scheduling and synchronization support**
- **Memory management and caching**
- **Resource governance**
- **Diagnostics and debug infrastructure**
- **Scalability and performance optimization**

Two Main Functions of SQLOS

Management

- Memory Manager
- Process Scheduler
- Synchronization
- I/O
- Support for Non-Uniform Memory Access (NUMA) and Resource Governor

Monitoring

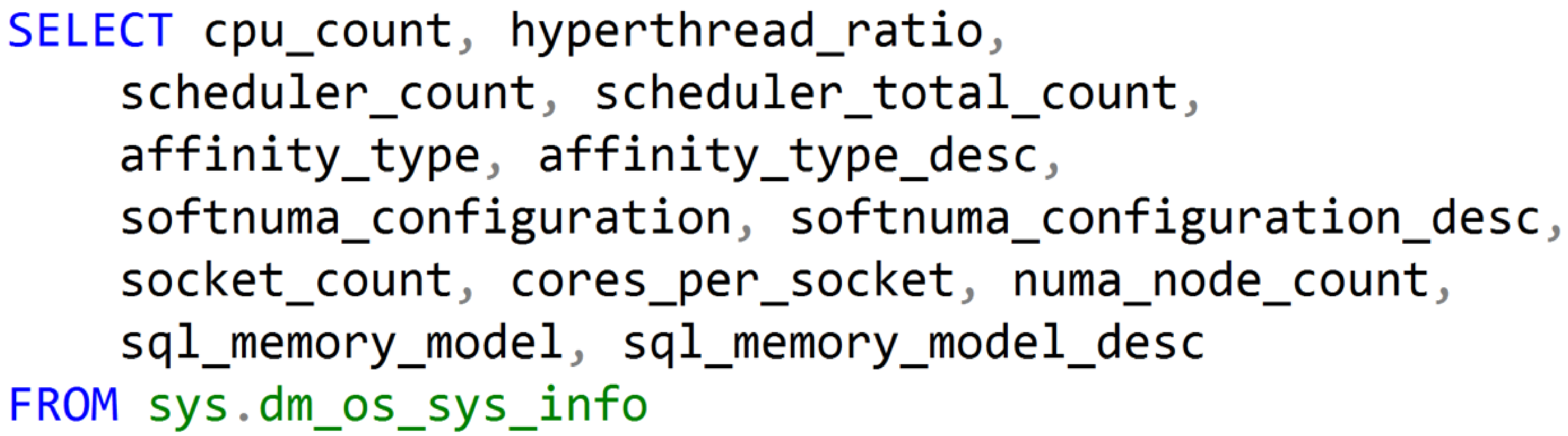
- Resource Monitor
- Deadlock Monitor
- Scheduler Monitor
- Lazy Writer (Buffer Pool management)
- Dynamic Management Views (DMVs)
- Extended Events
- Dedicated Administrator Connection (DAC)

Dynamic Management Views and Functions

Category	Description
sys.dm_exec_%	Execution and connection information
sys.dm_os_%	Operating system related information
sys.dm_tran_%	Transaction management information
sys.dm_io_%	I/O related information
sys.dm_db_%	Database information

Using Dynamic Management Objects (DMOs)

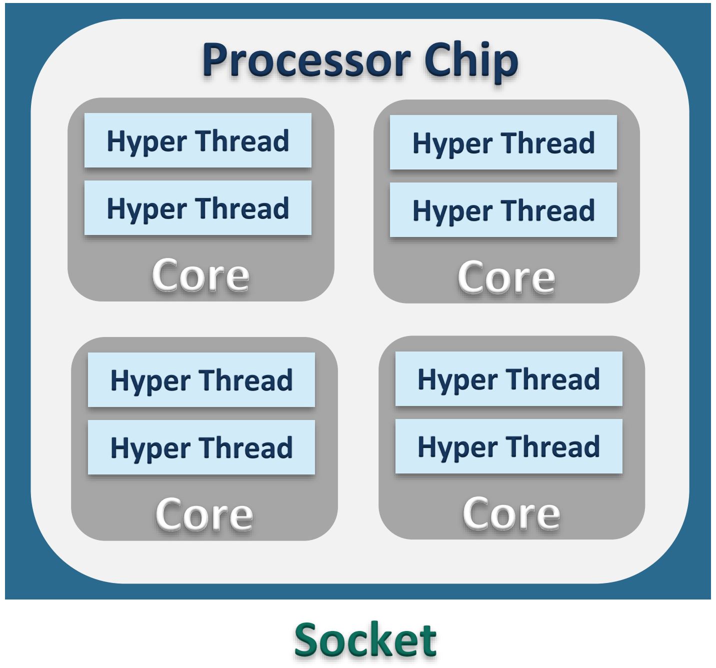
- Must reference using the sys schema
- Two basic types:
  - Real-time state information
  - Historical information



```
SELECT cpu_count, hyperthread_ratio,
    scheduler_count, scheduler_total_count,
    affinity_type, affinity_type_desc,
    softnuma_configuration, softnuma_configuration_desc,
    socket_count, cores_per_socket, numa_node_count,
    sql_memory_model, sql_memory_model_desc
FROM sys.dm_os_sys_info
```

CPU Architecture

Physical Hardware



Processor Chip

Hyper Thread

Hyper Thread

Core

Hyper Thread

Hyper Thread

Core

Hyper Thread

Hyper Thread

Core

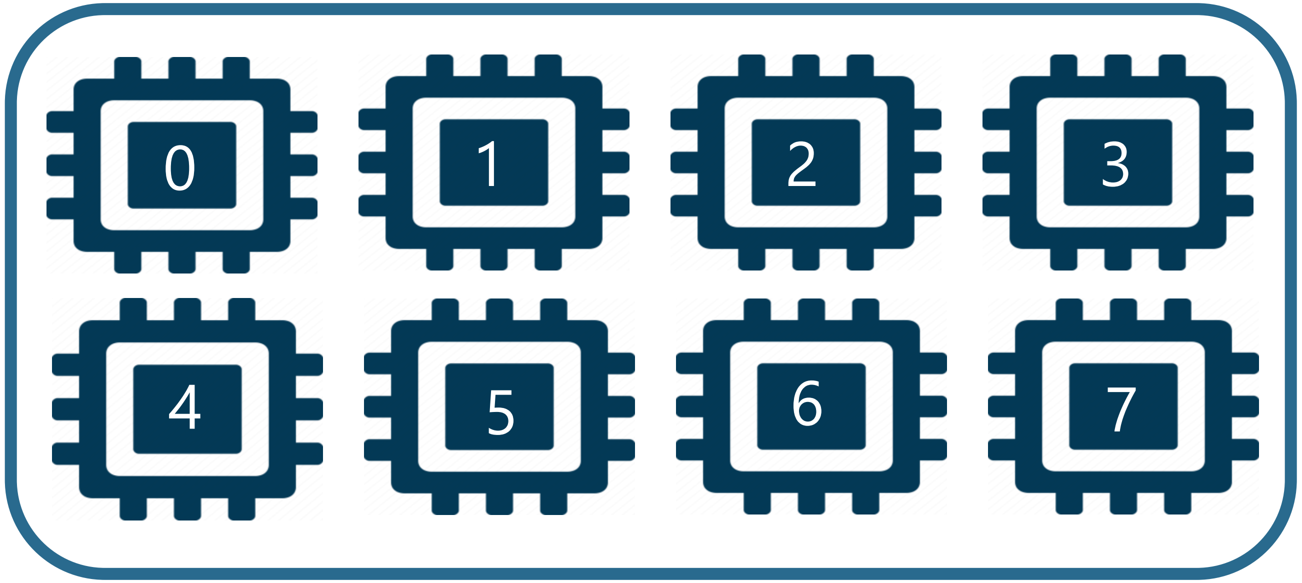
Hyper Thread

Hyper Thread

Core

Socket

Logical Processors as seen by the OS



Lesson 2: Thread and Task Scheduling

Microsoft SQL Server Scheduling Terminology

Batch

- A statement or set of statements submitted to SQL Server by the user (a query), also referred to as a request
- Monitor with sys.dm_exec_requests

Task

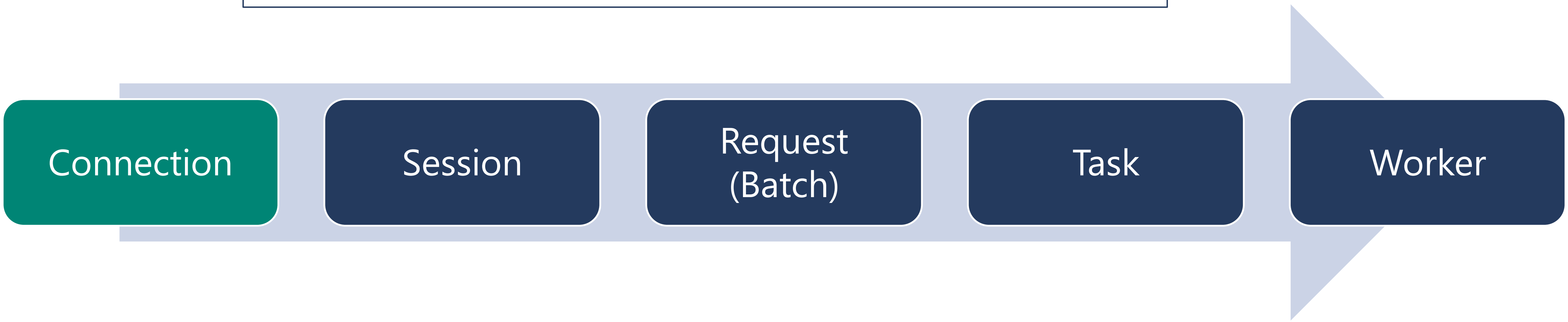
- A batch will have one or more tasks (aligns with statements)
- Monitor with sys.dm_os_tasks

Worker Thread

- Each task will be assigned to a single worker thread for the life of the task
- Monitor with sys.dm_os_workers

Hierarchy of Common Terms

```
SELECT *
FROM sys.dm_exec_connections;
-- relevant data:
-- session_id --> spid
-- most_recent_sql_handle --> last query
-- net_transport, protocol_type --> connectivity
```



Connection

Session

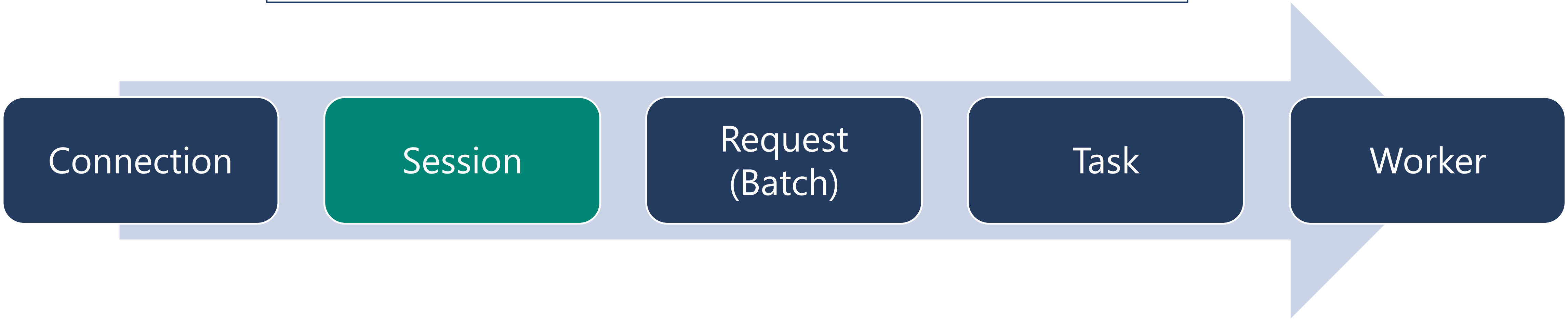
Request (Batch)

Task

Worker

Hierarchy of Common Terms

```
SELECT *
FROM sys.dm_exec_sessions;
-- relevant data:
-- session_id --> spid
-- host_name, program_name --> client identity
-- login_name, nt_user_name --> login identity
-- status --> activity
-- database_id --> database being accessed
-- open_transaction_count --> blocking identification
```



Connection

Session

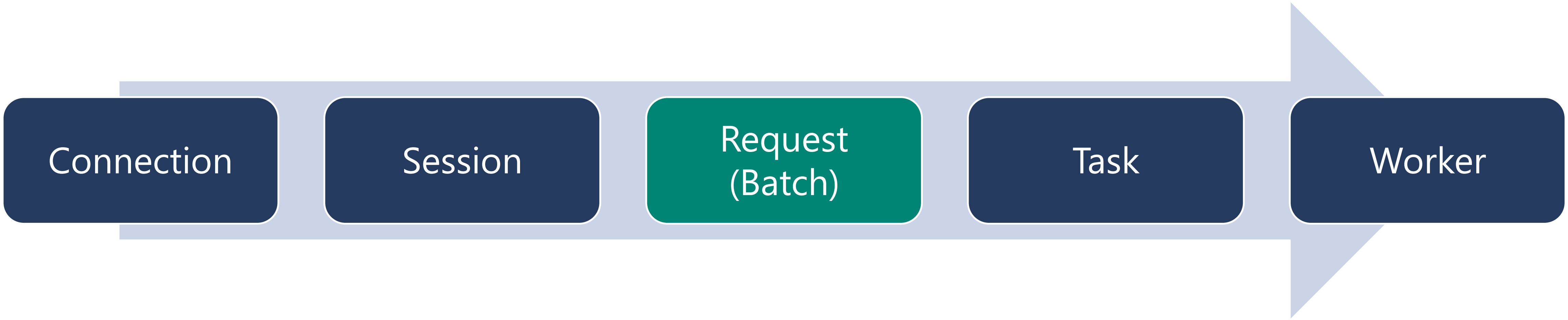
Request (Batch)

Task

Worker

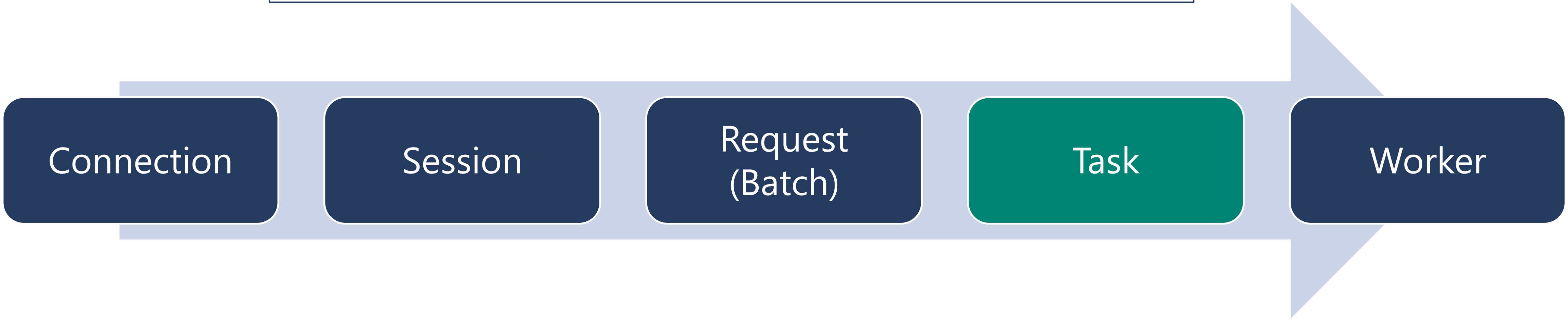
Hierarchy of Common Terms

```
SELECT *
FROM sys.dm_exec_requests;
-- relevant data:
-- session_id --> spid
-- status --> background, running, runnable, suspended
-- sql_handle, offset --> query text
-- database_id --> database being accessed
-- wait_type, wait_time --> blocking information
-- open_transaction_count --> blocking others
-- cpu_time, total_elapsed_time, reads, writes --> telemetry
```



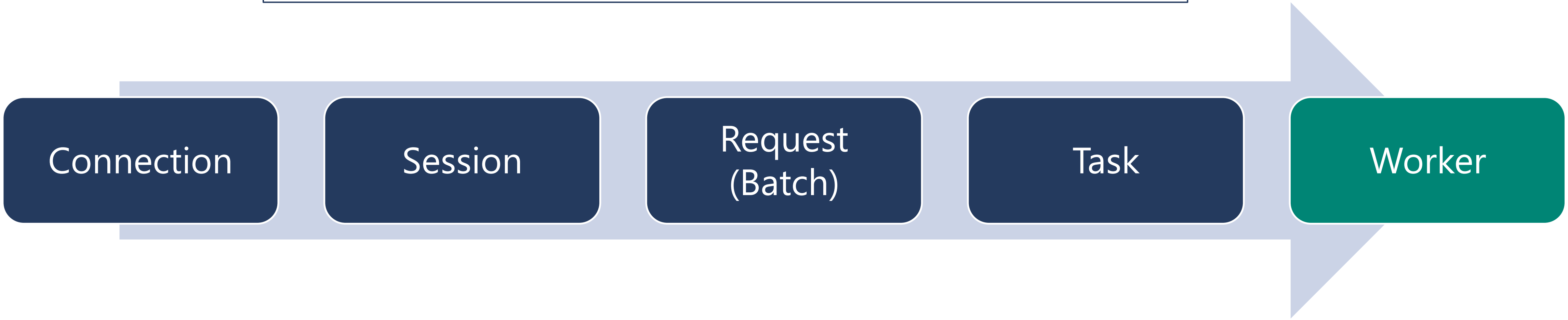
Hierarchy of Common Terms

```
SELECT *
FROM sys.dm_os_tasks;
-- relevant data:
-- task_state --> running, suspended
-- pending_io_* --> I/O activity
-- scheduler_id -->processor info
-- session_id --> spid
```



Hierarchy of Common Terms

```
SELECT *
FROM sys.dm_os_workers;
-- relevant data:
-- worker_address --> memory address of the worker
-- wait_start_ms_ticks --> Point in time worker Suspended.
-- wait_resumed_ms_ticks --> Worker in Runnable state.
-- state -- > Running, Runnable, Susspended
```



Scheduling Types

Non-Preemptive (Cooperative)

- SQL Server manages CPU scheduling for most activity (instead of the operating system).
- SQL Server decides when a thread should wait or get switched out (known as yielding).
- SQL Server developers also programmed some predetermined voluntary yields to avoid starvation of other threads

Preemptive

- Preemption is the act of an operating system temporarily interrupting an executing task.
- Higher priority tasks can preempt lower priority tasks.
- Preemptive mode used in SQL Server for external code calls, CLR with an UNSAFE assemblies, extended stored procedures

SQL Server Task Scheduling

One SQLOS Scheduler per core/logical processor

Handles scheduling tasks, I/O, and synchronization of resources

Work requests are balanced across schedulers based on the number of active tasks

Monitor using sys.dm_os_schedulers

Worker Migration – better use of schedulers in SQL Server 2019

Yielding

In SQL Server, each thread is assigned a quantum (duration 4ms), with SQL Server using a cooperative model to ensure its CPU resources are shared amongst all the threads that are in a runnable state, preventing the ‘starving’ condition of any individual thread.

By design, a worker owns the scheduler until it yields to another worker on the same scheduler.

When no worker is currently on the Runnable list, the **yielding worker is allowed another quantum** or performs the necessary idle scheduler maintenance.

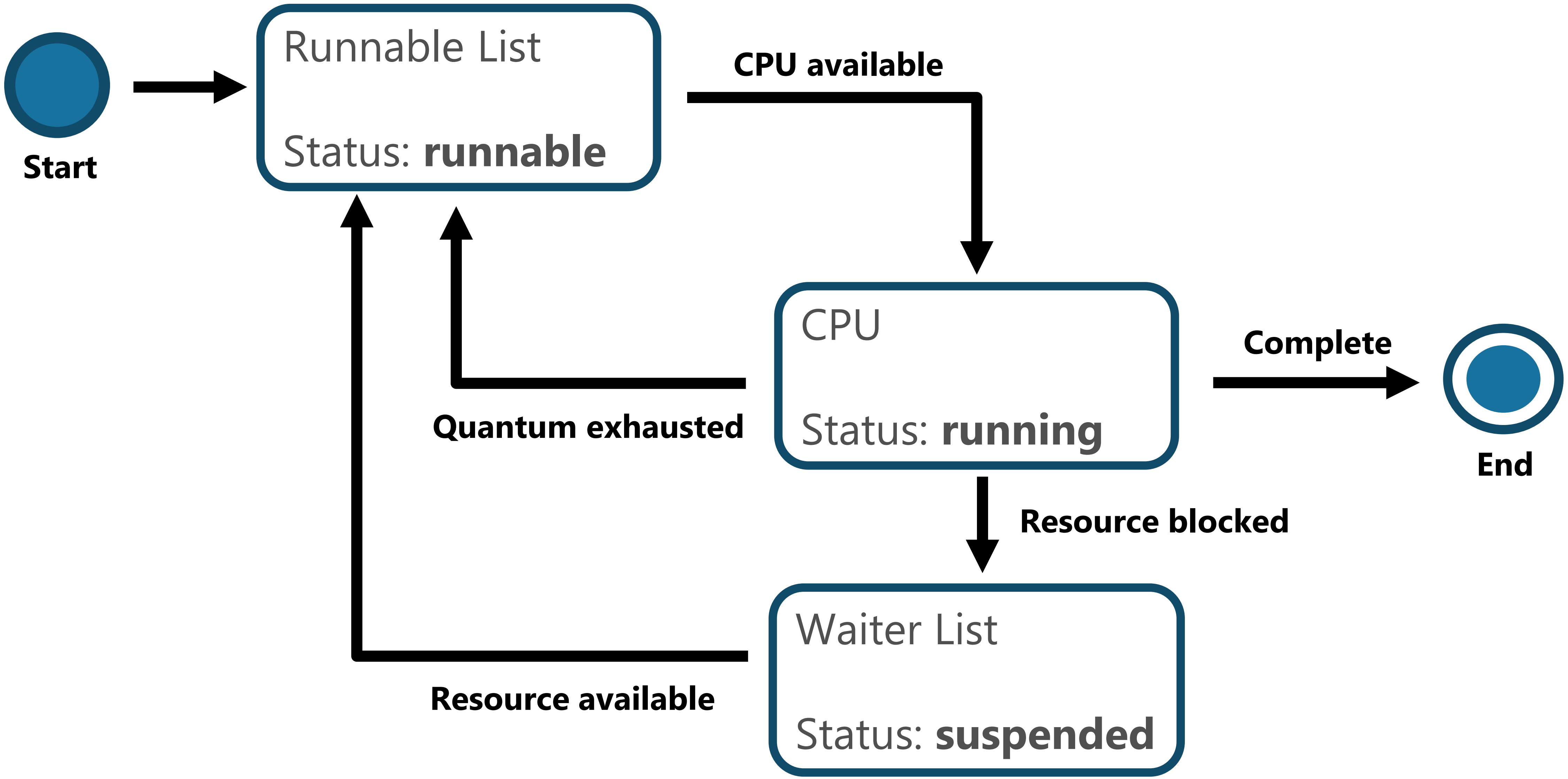
Thread States and Queues

Runnable: The thread is currently in the Runnable Queue waiting to execute. (First In, First Out).

Running: One active thread executing on a processor.

Suspended: Placed on a Waiter List waiting for a resource other than a processor. (No specific order).

Yielding



Start

Runnable List

Status: **runnable**

CPU available

CPU

Status: **running**

Quantum exhausted

Complete

End

Resource blocked

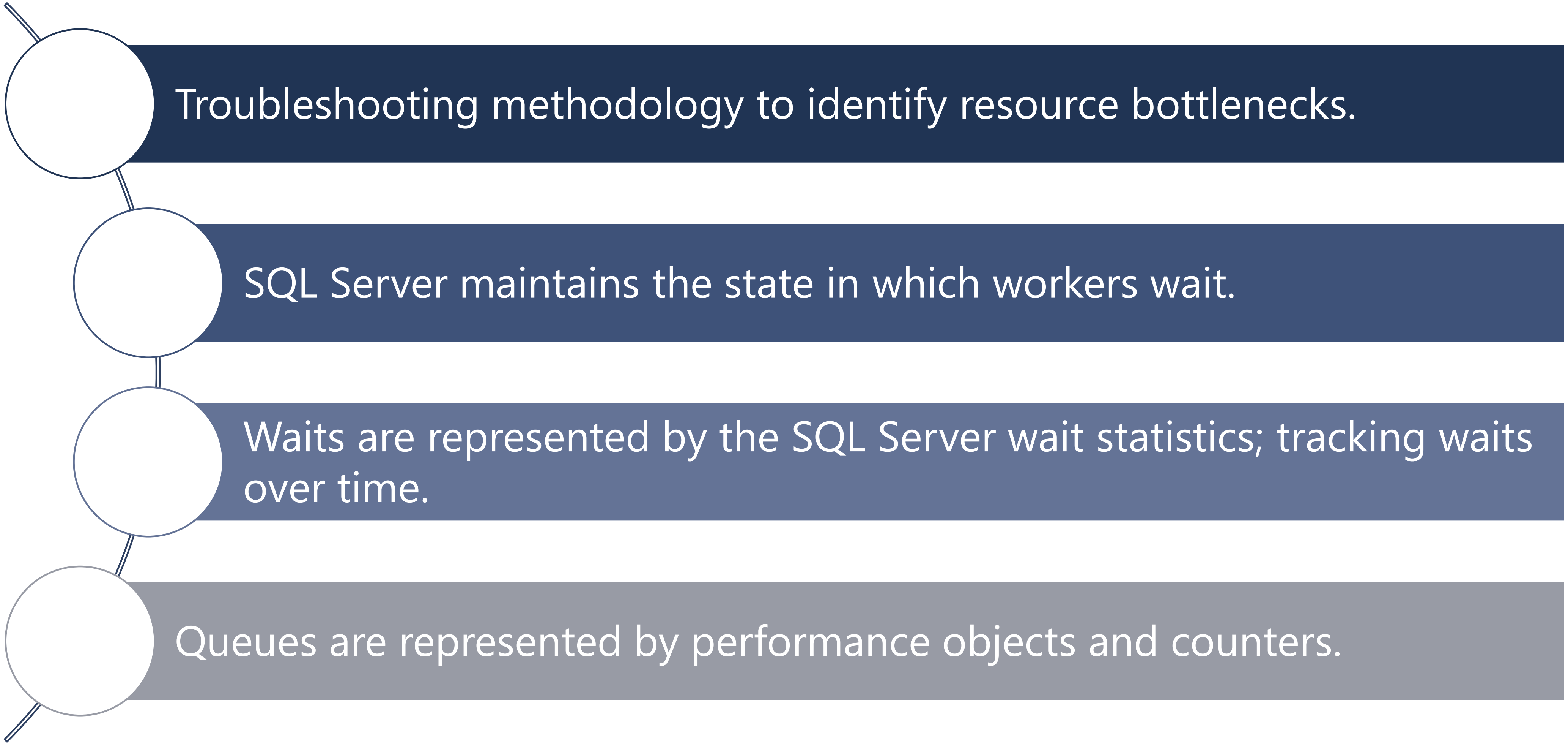
Waiter List

Status: **suspended**

Resource available

Lesson 3: Waits and Queues

Waits and Queues



Troubleshooting methodology to identify resource bottlenecks.

SQL Server maintains the state in which workers wait.

Waits are represented by the SQL Server wait statistics; tracking waits over time.

Queues are represented by performance objects and counters.

Using Waits and Queues

Useful to assist in troubleshooting an active performance issue

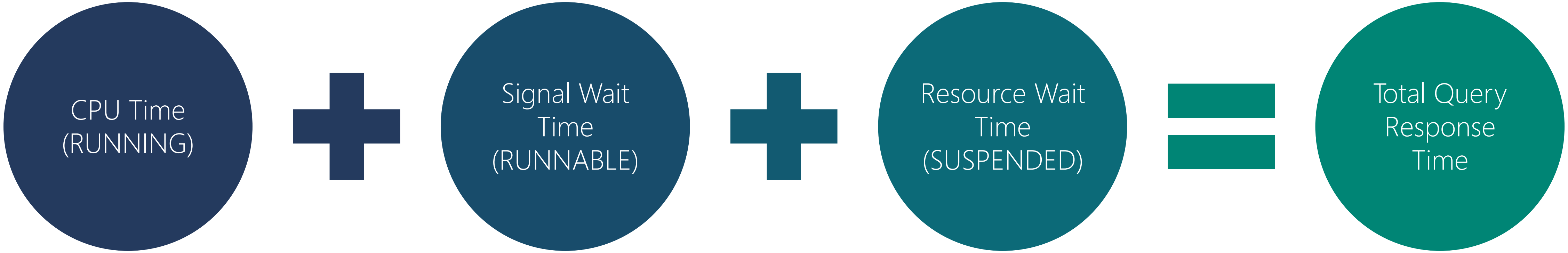
Valuable to track the resources SQL Server is regularly waiting on

Useful for workload measurements and benchmarking

Valuable for identifying performance trends

Task Execution Model

- The full cycle between the several task states, for how many times it needs to cycle, is what we experience as the total query response time.


CPU Time (RUNNING)
+
Signal Wait Time (RUNNABLE)
+
Resource Wait Time (SUSPENDED)
=
Total Query Response Time

Relevant Dynamic Management Views (DMVs)

sys.dm_os_wait_stats

- Returns information about all the waits encountered by threads that ran.
- Includes wait type, number of tasks that waited in the specific wait type, total and max wait times, and the amount of signal waits.

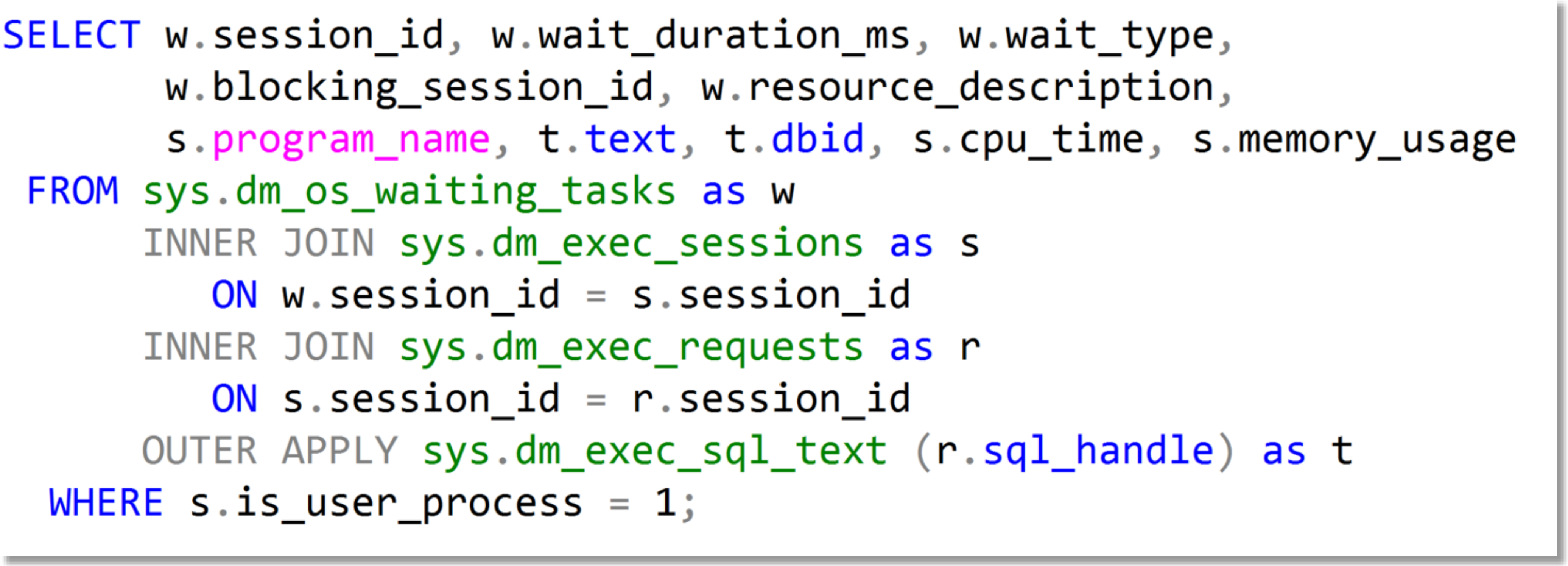
sys.dm_os_waiting_tasks

- Returns information about the wait queue of tasks actively waiting on some resource.

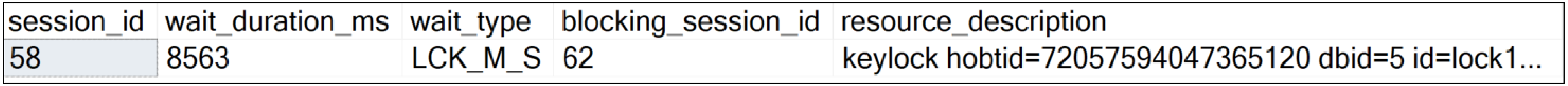
sys.dm_exec_requests

- Returns information about each request that is in-flight.
- Includes session owning the request and status of the request, which will reflect the status of one or more tasks assigned to the request.

Waiting Tasks DMV



```sql
SELECT w.session_id, w.wait_duration_ms, w.wait_type,
       w.blocking_session_id, w.resource_description,
       s.program_name, t.text, t.dbid, s.cpu_time, s.memory_usage
 FROM sys.dm_os_waiting_tasks as w
     INNER JOIN sys.dm_exec_sessions as s
        ON w.session_id = s.session_id
     INNER JOIN sys.dm_exec_requests as r
        ON s.session_id = r.session_id
     OUTER APPLY sys.dm_exec_sql_text (r.sql_handle) as t
  WHERE s.is_user_process = 1;
```

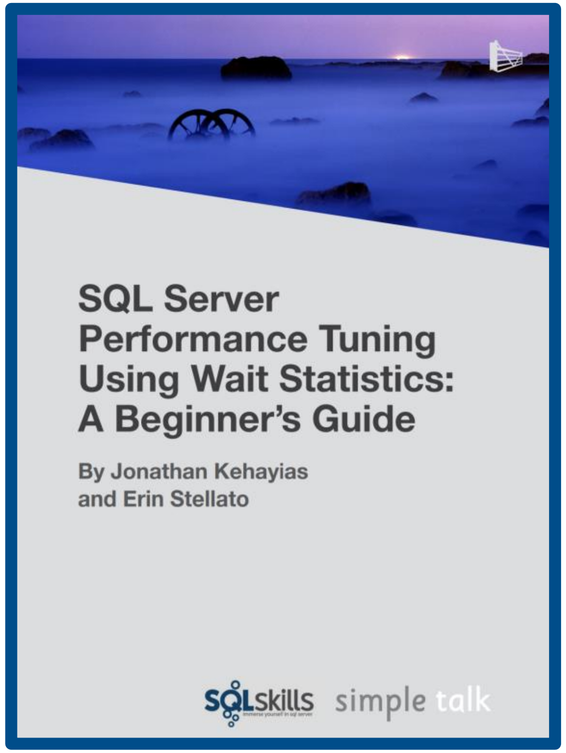


session_id	wait_duration_ms	wait_type	blocking_session_id	resource_description
58	8563	LCK_M_S	62	keylock hobtid=72057594047365120 dbid=5 id=lock1...

Troubleshooting Wait Types

Aaron Bertrand – Top Wait Types
https://sqlperformance.com/2018/10/sql-performance/top-wait-stats

Paul Randal – SQL Skills Wait Types Library
https://www.sqlskills.com/help/waits/



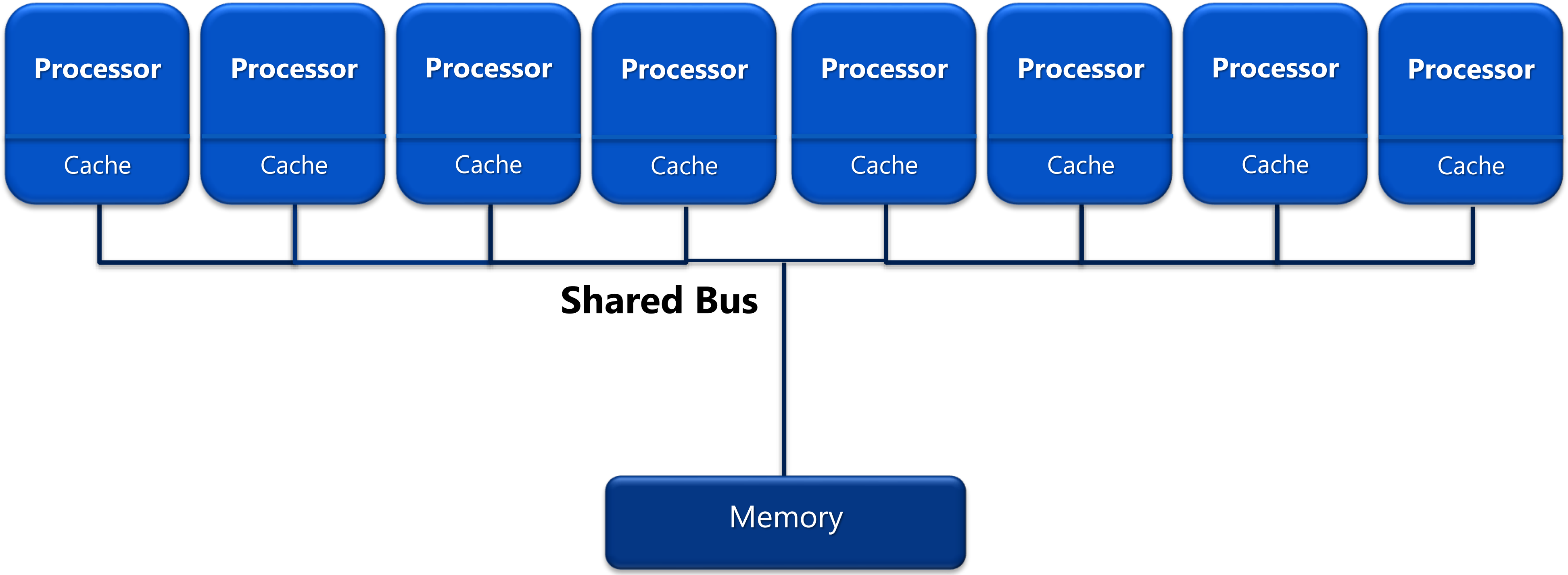
SQL Server
Performance Tuning
Using Wait Statistics:
A Beginner's Guide

By Jonathan Kehayias
and Erin Stellato

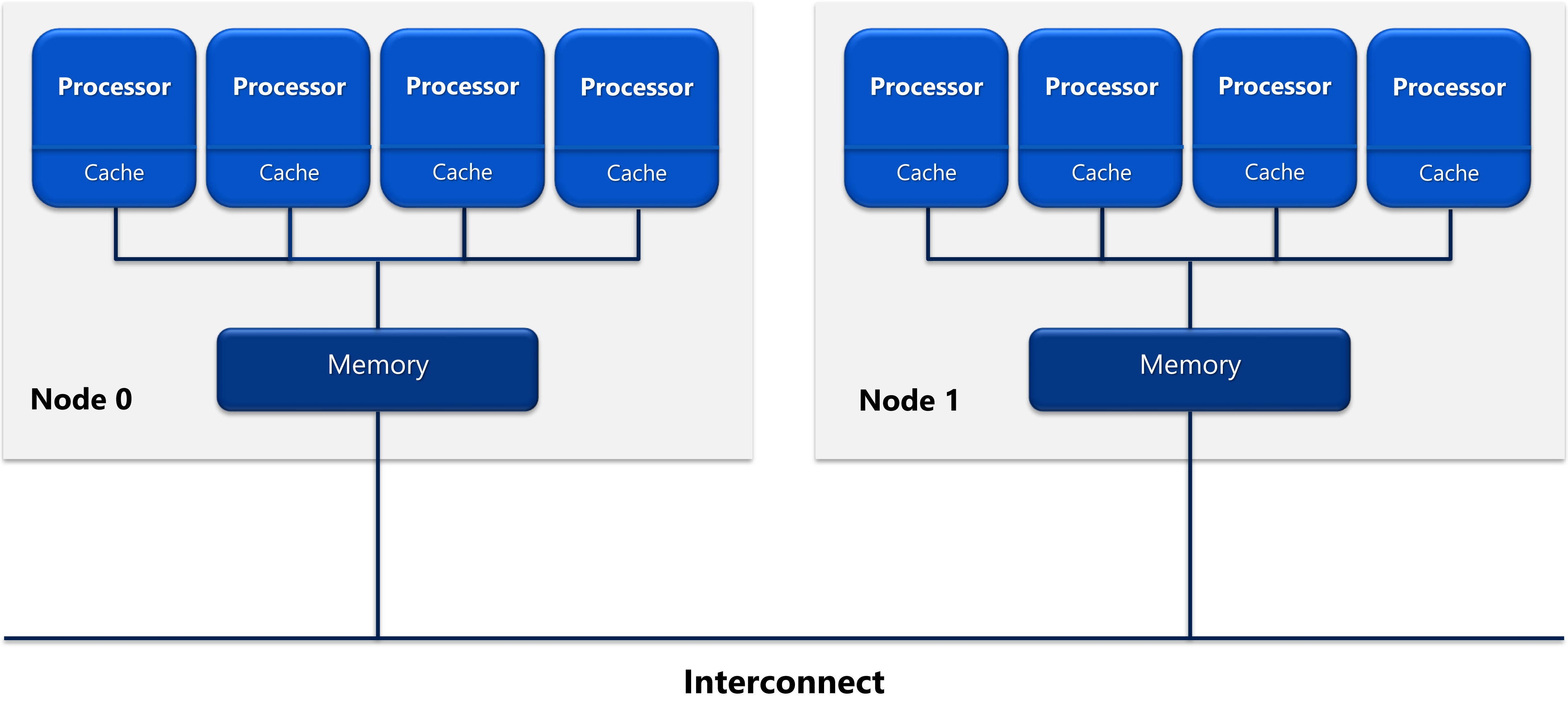
SQLskills simple talk

Lesson 4: CPU and Memory Configuration

Symmetric Multi-Processing (SMP)



Non-Uniform Memory Access (NUMA)



NUMA (Non-Uniform Memory Access) Architecture

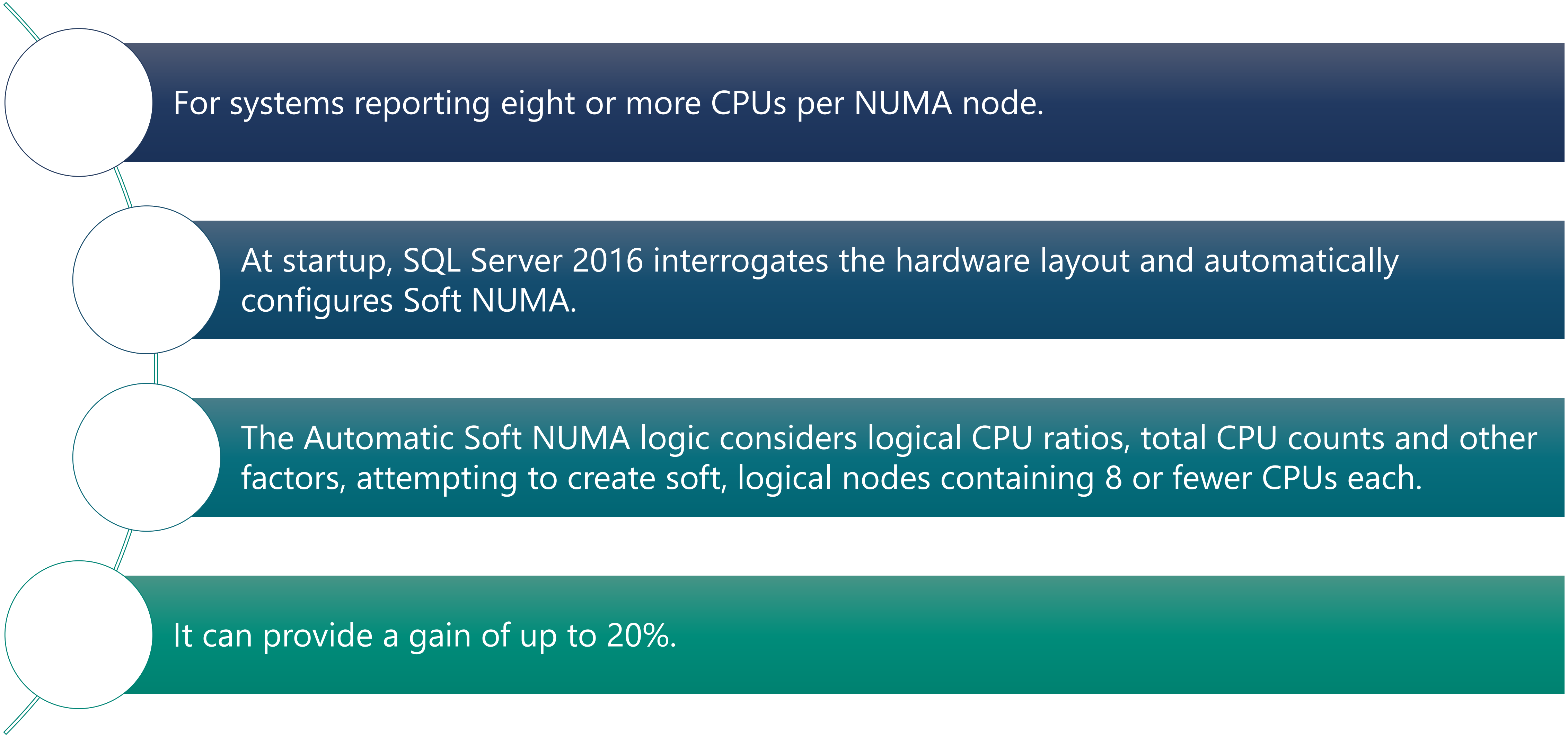
Offers nodes of processors each with its own bus for access for local memory.

Interconnect between nodes allows one node to get to other’s memory.

Offers scalability for NUMA-aware applications.

NUMA-aware applications such as SQL Server try to avoid remote or foreign memory access.

Automatic Soft NUMA



For systems reporting eight or more CPUs per NUMA node.

At startup, SQL Server 2016 interrogates the hardware layout and automatically configures Soft NUMA.

The Automatic Soft NUMA logic considers logical CPU ratios, total CPU counts and other factors, attempting to create soft, logical nodes containing 8 or fewer CPUs each.

It can provide a gain of up to 20%.

SQL Server Configuration

Processor Configuration Settings And Best Practices

Affinity Mask

- Assigns CPUs for SQL Server use
- Set via sp_configure or Alter Server Configuration
- Only required in specific scenarios

Max Degree of Parallelism (MAXDOP)

- Maximum number of processors that are used for the execution of a query in a parallel plan. This option determines the number of threads that are used for the query plan operators that perform the work in parallel.

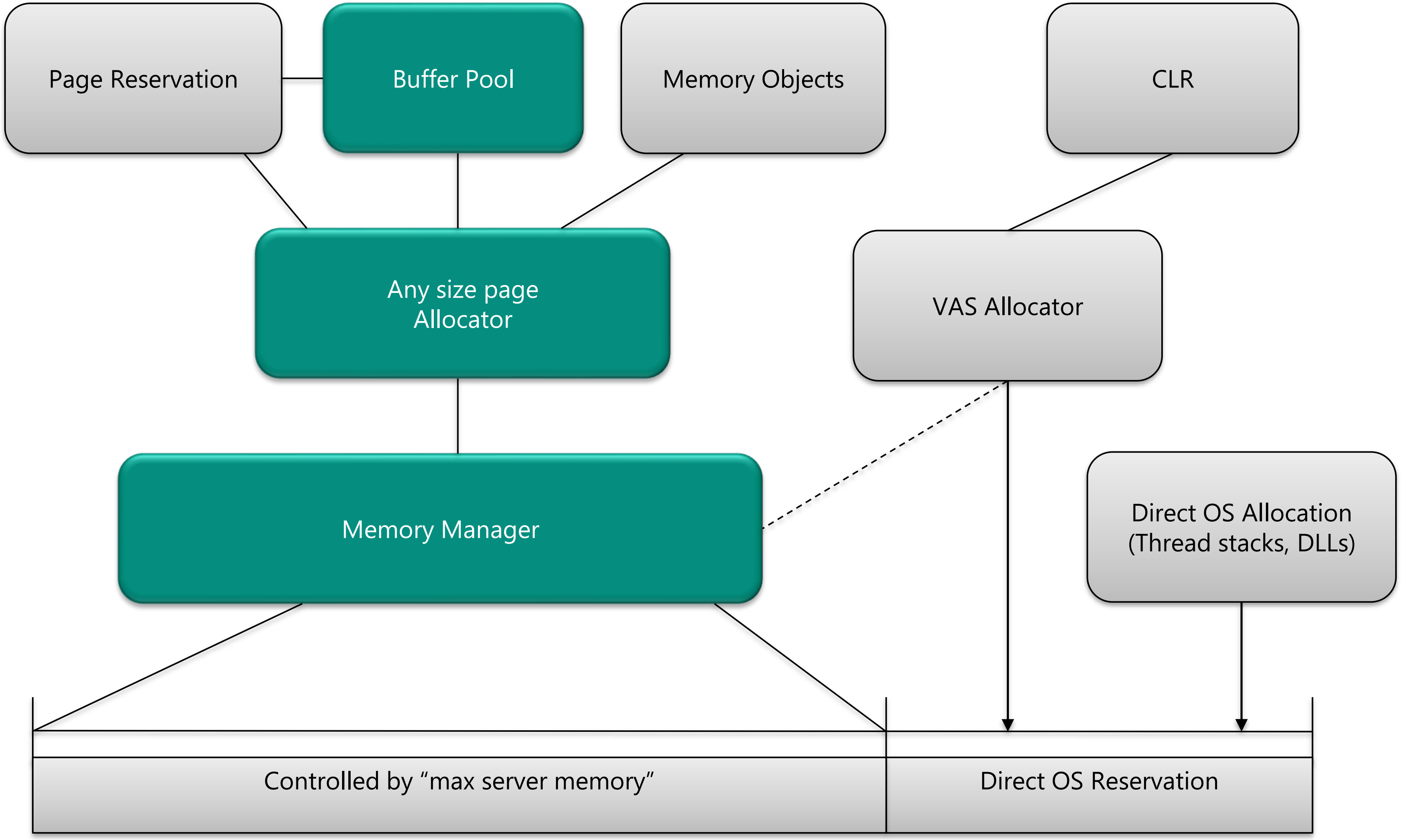
Cost Threshold for Parallelism

- Only queries with a cost that is higher than this value will be considered for parallelism
- Only required when dealing with excessive parallelism

Max Worker Threads

- Number of threads SQL Server can allocate
- Recommended value is 0. SQL Server will dynamically set the Max based on CPUs and CPU architecture

Memory Manager SQL Server 2012 and later



How to determine Thread Stack Memory

Maximum Worker Threads
512 + (Processors -4) *16

*

2mb per thread

Cores	Threads	Memory (MB)
4	512	1,024
8	576	1,152
16	704	1,408
32	960	1,920
64	1,472	2,944
80	1,728	3,456

SQL Server Configuration

MAXDOP Setting and Best Practices

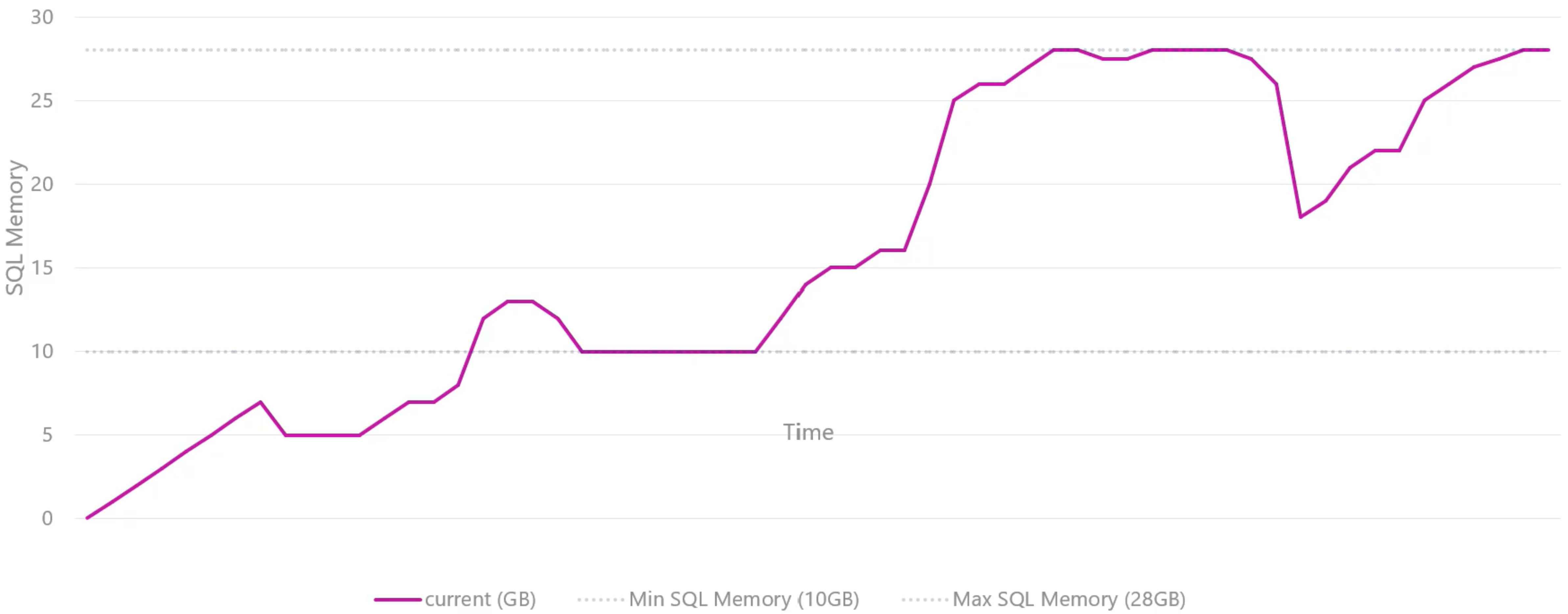
Best Practice Recommendations (documented in KB 2806535):

Server with single NUMA node	Less than or equal to 8 logical processors	Keep MAXDOP at or below # of logical processors
Server with single NUMA node	Greater than 8 logical processors	Keep MAXDOP at 8
Server with multiple NUMA nodes	Less than or equal to 16 logical processors per NUMA node	Keep MAXDOP at or below # of logical processors per NUMA node
Server with multiple NUMA nodes	Greater than 16 logical processors per NUMA node	Keep MAXDOP at half the number of logical processors per NUMA node with a MAX value of 16

Dynamic Memory Management

- Max Server Memory defines the maximum amount of memory the SQL Server process can allocate.
- Min Server Memory defines the level down to which SQL Server may trim in the event of memory pressure.
- Memory is allocated as-needed after the SQL Server service startup.
- After memory is acquired, it will not be released unless the operating system reports memory pressure.
- In the event of pressure, SQL Server will reduce its memory footprint to avoid operating system paging.

Dynamic Memory Management



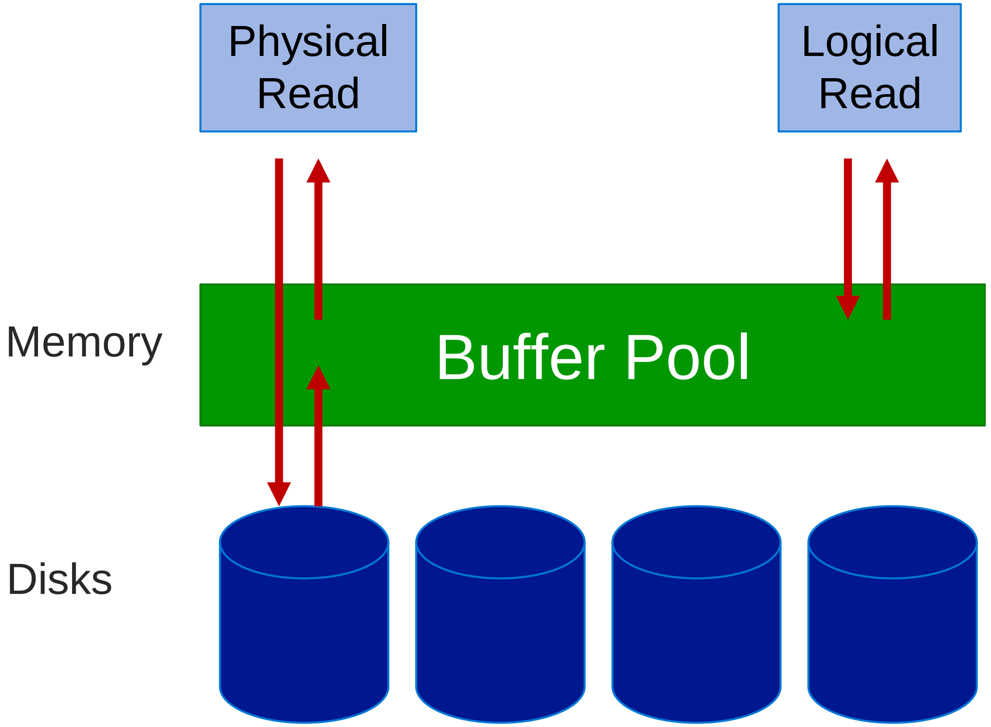
SQL Server Buffer Pool

Stores 8 kilobytes (KB) pages of data to avoid repeated disk I/O.

- Pages held in the buffer until the space is needed by something else.

Largest percentage of SQL Server memory.

- Separate buffer pool nodes for each hardware NUMA node.



```
/* physical Reads & Logical Reads can be obtained with */
SET STATISTICS IO ON
```

Lock Pages in Memory

Special operating system API for memory allocations.

Memory allocated through this API cannot be paged out by the operating system.

Needed to support large page allocations.

Configured by granting the Lock pages in memory security privilege to the SQL Server service account.

Microsoft

Module 2: SQL Server I/O and Database Structure

Lesson 1: Database Files

Database files and filegroups

Database files

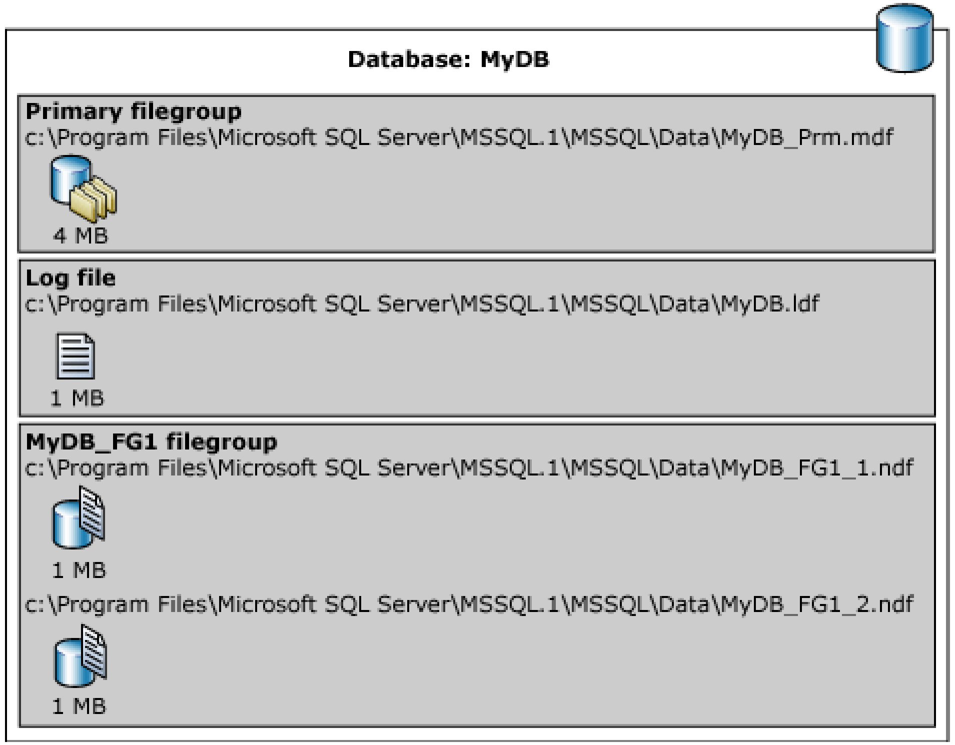
A database is composed by at least two operating system files:

Data files

- Contain database objects and data
- First data file is called primary data file. This file has a .mdf extension
- A database may have additional data files, known as secondary data files. They use .ndf extension
- Can be grouped together in filegroups for allocation and administration purposes

Log file

- Contain Log Records and entries are sequenced



Database: MyDB

Primary filegroup
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB_Prm.mdf
4 MB

Log file
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB.ldf
1 MB

MyDB_FG1 filegroup
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB_FG1_1.ndf
1 MB
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB_FG1_2.ndf
1 MB

SQL Server disk I/O patterns:

Data Files

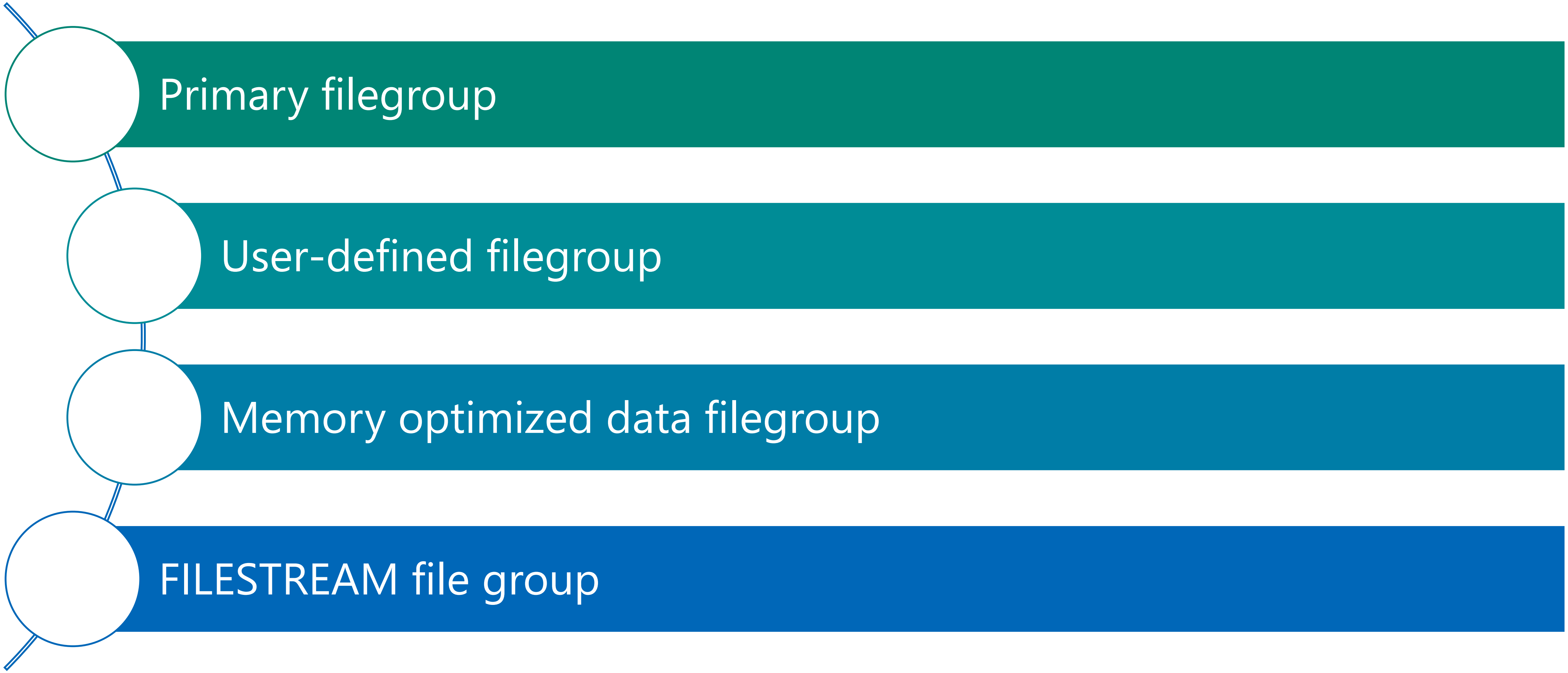
- One .mdf file per database
- May have one or more .ndf file
- Random reads and writes
  - Read activity during backups; other activity varies depending on query activity and buffer pool size
- Write activity during checkpoints, recovery, and lazy writes

Log Files

- One* .ldf file per database
- Sequential reads and writes
- Write activity during the log buffer flush operations
- Read activity during checkpoints, backups, and recovery
- Features such as database mirroring and replication will increase read and write activity

Database files and filegroups

Types of filegroups



- Primary filegroup
- User-defined filegroup
- Memory optimized data filegroup
- FILESTREAM file group

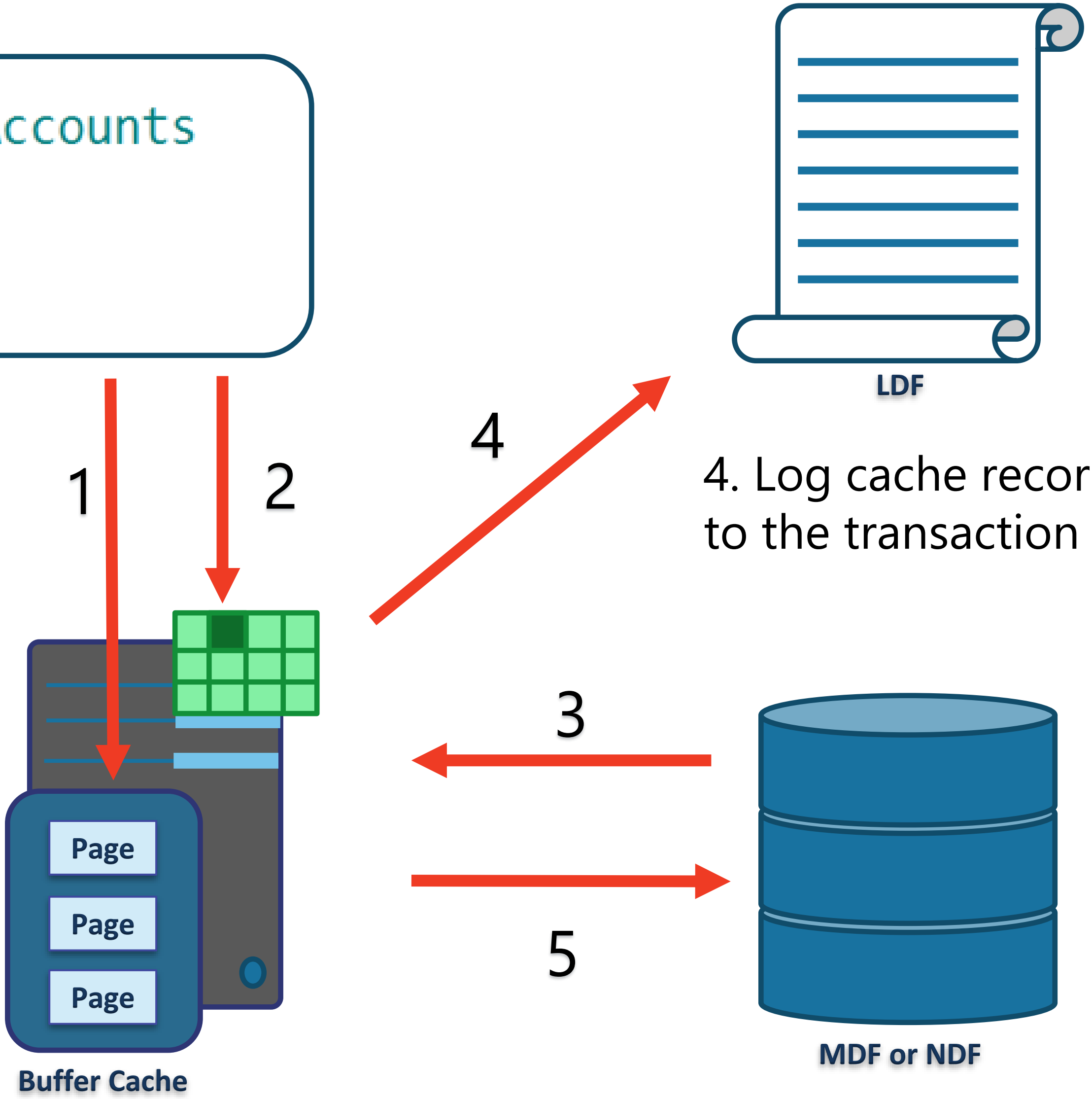
SQL Server Disk I/O (Write-Ahead Logging)

```
UPDATE Accounting.BankAccounts
SET Balance -= 200
WHERE AcctID = 1
```

1. Data modification is sent to buffer cache in memory.

2. Modification is recorded in the log cache.

3. Data pages are located or read into the buffer cache and then modified.



4. Log cache record is flushed to the transaction log

5. At checkpoint, dirty data pages are written to the database file.

Log Buffer Flushing

SQL Server will flush the log buffer to the log file

- SQL Server gets a commit request of a transaction that changes data.
- The log buffer fills up. (Max size 60kb.)
- SQL Server needs to harden dirty data pages (checkpoints)
- Manually request a log buffer flush using the sys.sp_flush_log procedure

Log buffer flushing results in a WRITELOG wait type.

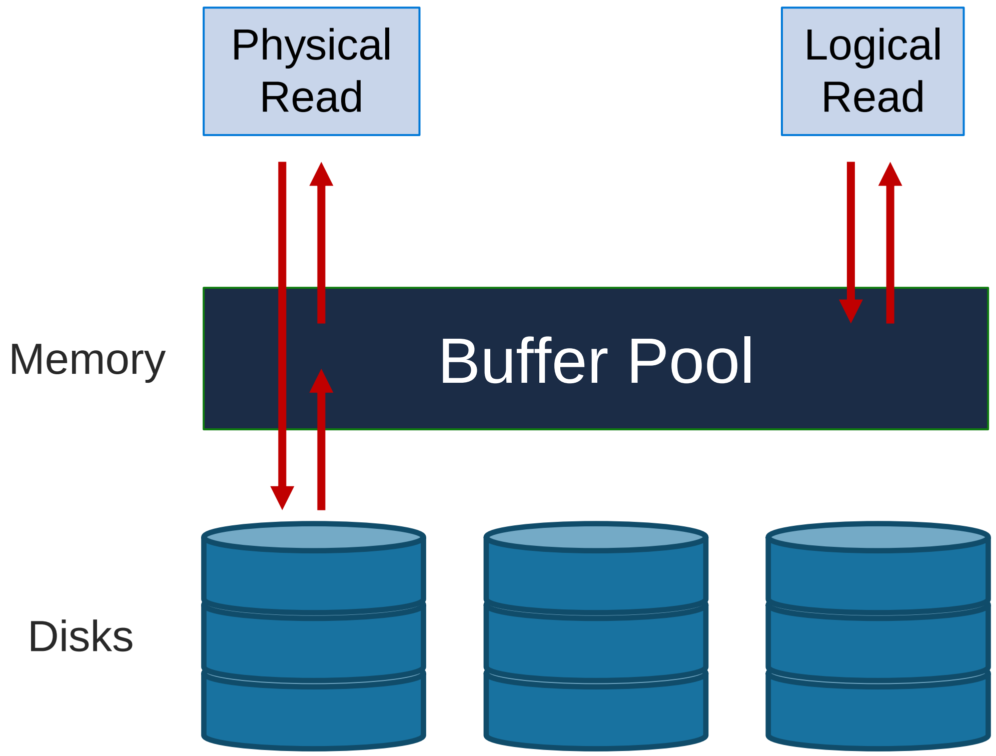
SQL Server Buffer Pool

Stores 8 kilobytes (KB) pages of data to avoid repeated disk I/O.

- Pages held in the buffer until the space is needed by something else.

Lazy Writer searches for eligible buffers.

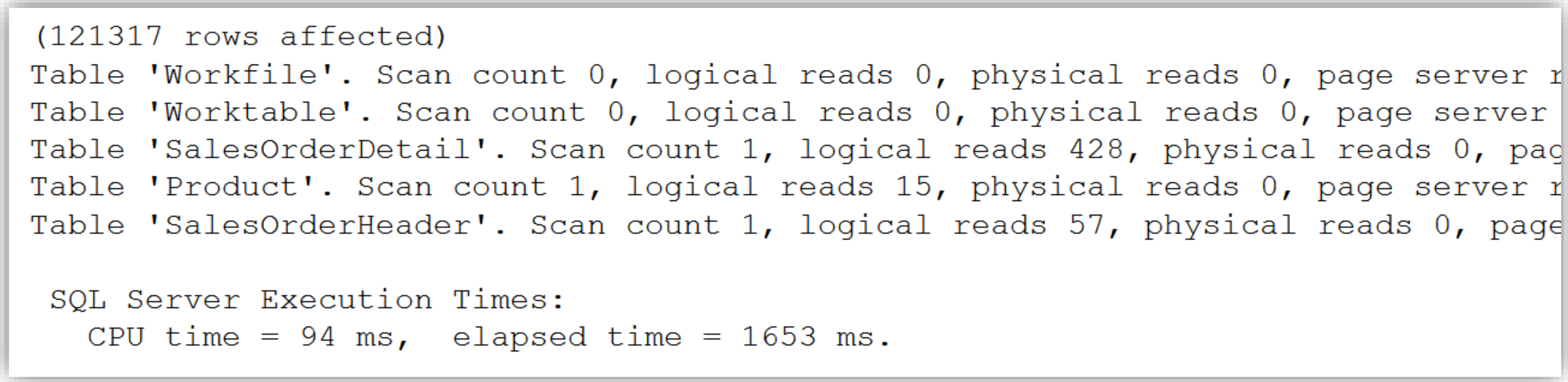
- If the buffer is dirty, an asynchronous write (lazy write) is posted so that the buffer can later be freed.
- If the buffer is not dirty, it is freed.



SET STATISTICS IO

```sql
SET STATISTICS IO ON
GO
SET STATISTICS TIME ON
SELECT SOH.SalesOrderID, SOH.CustomerID,
OrderQty, UnitPrice, P.Name
FROM Sales.SalesOrderHeader AS SOH
JOIN Sales.SalesOrderDetail AS SOD
ON SOH.SalesOrderID = SOD.SalesOrderID
JOIN Production.Product AS P
ON P.ProductID = SOD.ProductID
SET STATISTICS IO, TIME OFF
```

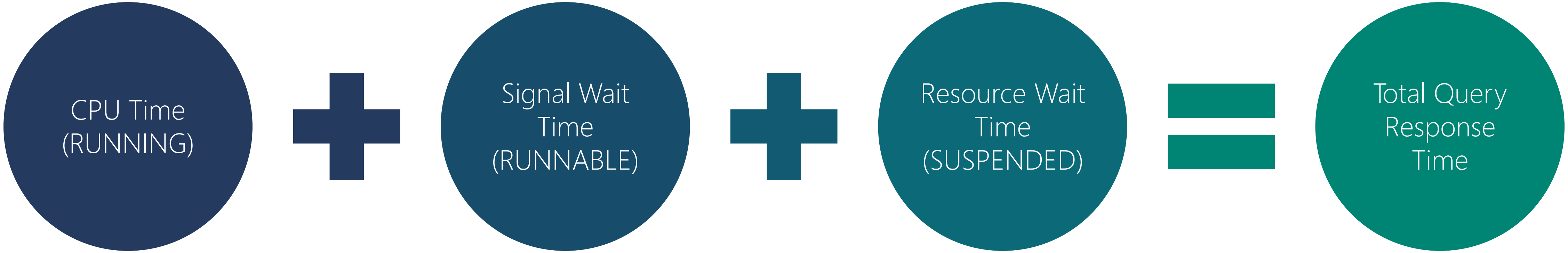
Used to identify physical reads and logical reads for a query



```
(121317 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server r
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server
Table 'SalesOrderDetail'. Scan count 1, logical reads 428, physical reads 0, pag
Table 'Product'. Scan count 1, logical reads 15, physical reads 0, page server r
Table 'SalesOrderHeader'. Scan count 1, logical reads 57, physical reads 0, page

 SQL Server Execution Times:
   CPU time = 94 ms,  elapsed time = 1653 ms.
```

Total Query Response Time

- The full cycle between the several task states, for how many times it needs to cycle, is what we experience as the total query response time.


CPU Time (RUNNING)
+
Signal Wait Time (RUNNABLE)
+
Resource Wait Time (SUSPENDED)
=
Total Query Response Time

Checkpoints

Flushes dirty pages from the buffer pool to the disk. Frequency of checkpoints varies based on the database activity and recovery interval.

Automatic (default) – Database engine issues checkpoints automatically based on the server level "recovery interval" configuration option

Indirect (new in SQL Server 2012) – Database engine issues checkpoints automatically based on the database level TARGET_RECOVERY_TIME

```sql
ALTER DATABASE [AdventureWorksPTO] SET TARGET_RECOVERY_TIME = 60 SECONDS
```

Manual – Issued in the current database for your connection when you execute the T-SQL CHECKPOINT command

Internal – Issued by various server operations

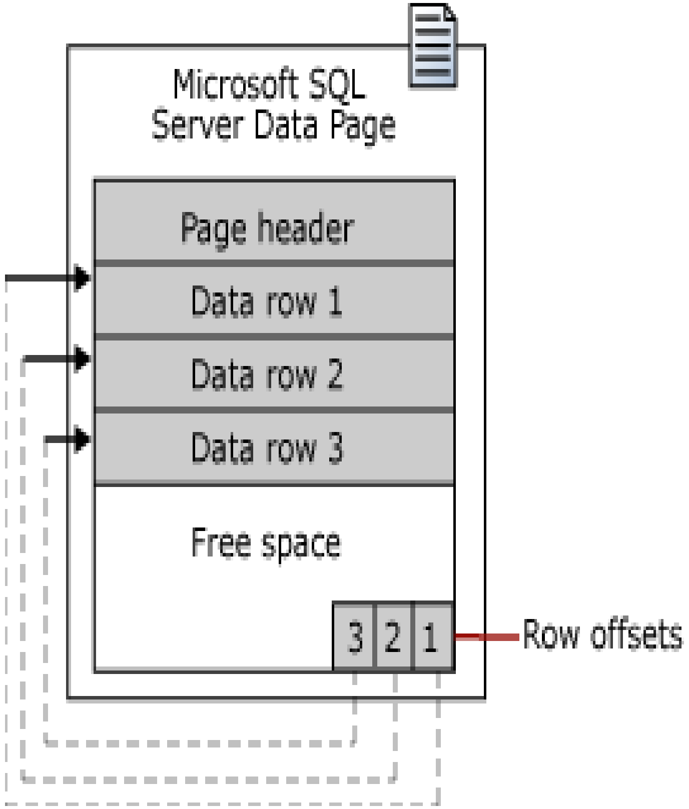
Lesson 2: Data Page Structures

Pages and Extents architecture

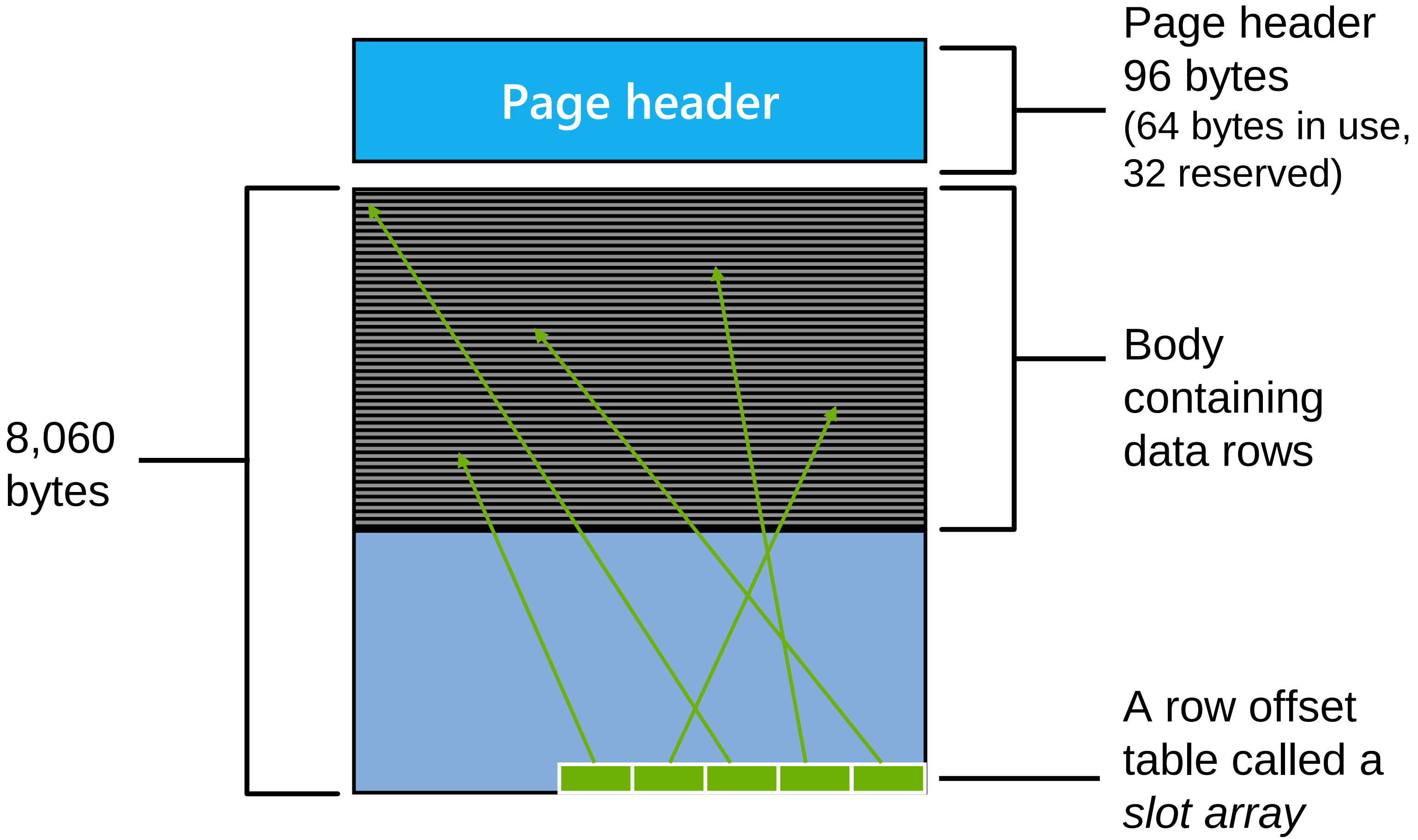
A data page is the fundamental unit of data storage in SQL Server.

- The disk space allocated to a data file (.mdf or .ndf) is logically divided into pages.
- Each page is 8 KB in size
- Pages are numbered contiguously from 0 to n.
- Disk I/O operations are performed at the page level.

Extents are a collection of eight physically contiguous pages (64KB) and are used to efficiently manage the pages.



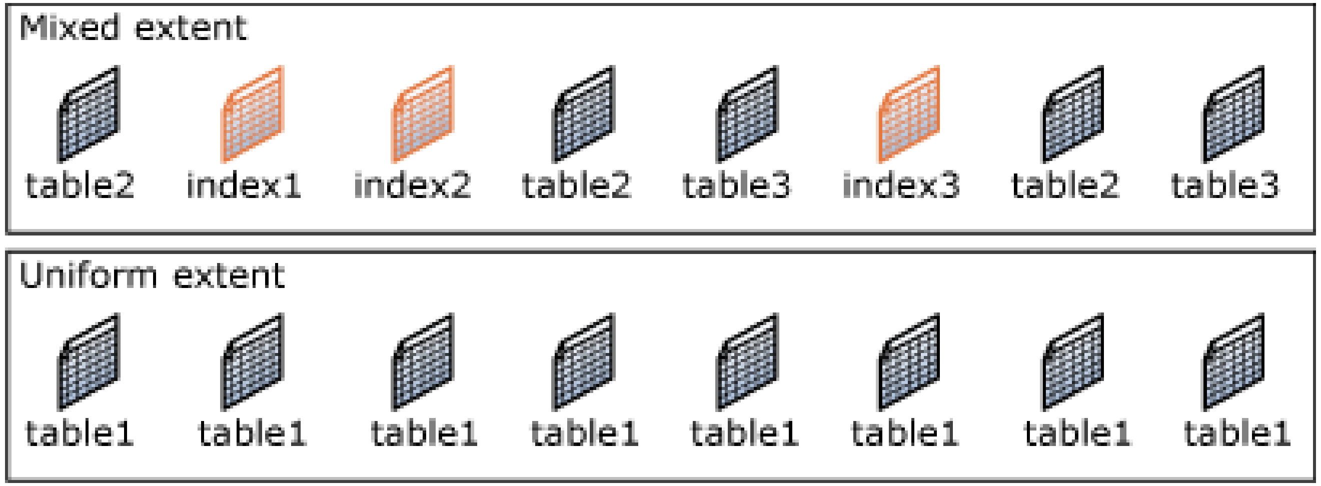
Data Page structure



SQL Server object allocation

Two types of extents:

- Mixed extents
  - Up to 8 distinct objects can reside in the same Mixed Extent.
  - Controlled by SGAM (Shared Global Allocation Map) Object.
- Uniform extents
  - All pages belong to the same object.
  - Controlled by GAM (Global Allocation Map) object.



Page types

Page Type (ID)	Description
Data (1)	Data rows with all data, except text, ntext, image, nvarchar(max), varchar(max), varbinary(max), and xml data, when **text in row** is set to **ON**
Index (2)	Index Entries
Text/Image (3 or 4)	Large Object Data Type, variable length columns when the data row exceeds 8 kilobytes (KB)
GAM, SGAM (8 and 9)	Extent Allocation information
PFS (11)	Information about page allocation and free space available on pages
IAM (10)	Information about extents used by a table or index per allocation unit
Bulk Changed Map (17)	Information about extents modified by bulk operations since the last BACKUP LOG statement per allocation unit
Differential Changed Map (16)	Information about extents that have changed since the last BACKUP DATABASE statement per allocation unit
Boot (13)	Information about the database; Each database has only one Boot page
File Header (15)	Information about the file. It is the first page (page 0) in every file

The Role of Allocation Pages in Object Allocation

PFS and IAM are used to determine when an object needs a new extent allocated

GAMs and SGAMs are used to allocate the extent

Allocation objects

Page Free Space (PFS)

- Tracks free space on a page uses 1 Byte/Page
- Covers 64 MB worth of pages

Global Allocation Map (GAM)

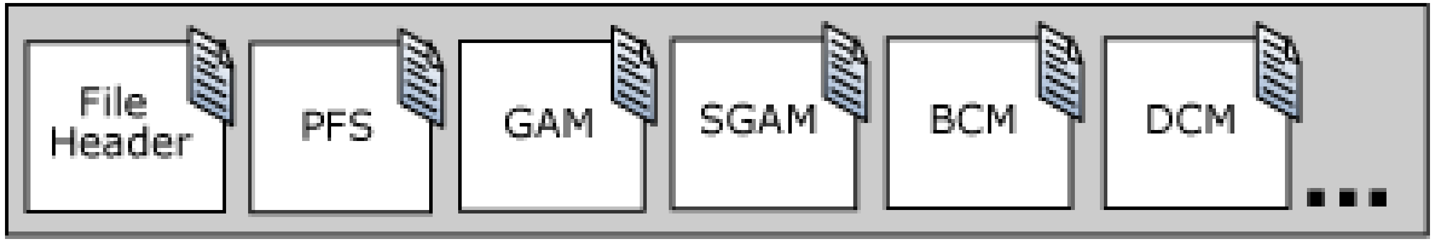
- Tracks Uniform extents that have been allocated uses 1 Bit/extent
- Covers 64,000 extents (4 GB worth of data)

Shared Global Allocation Map (SGAM)

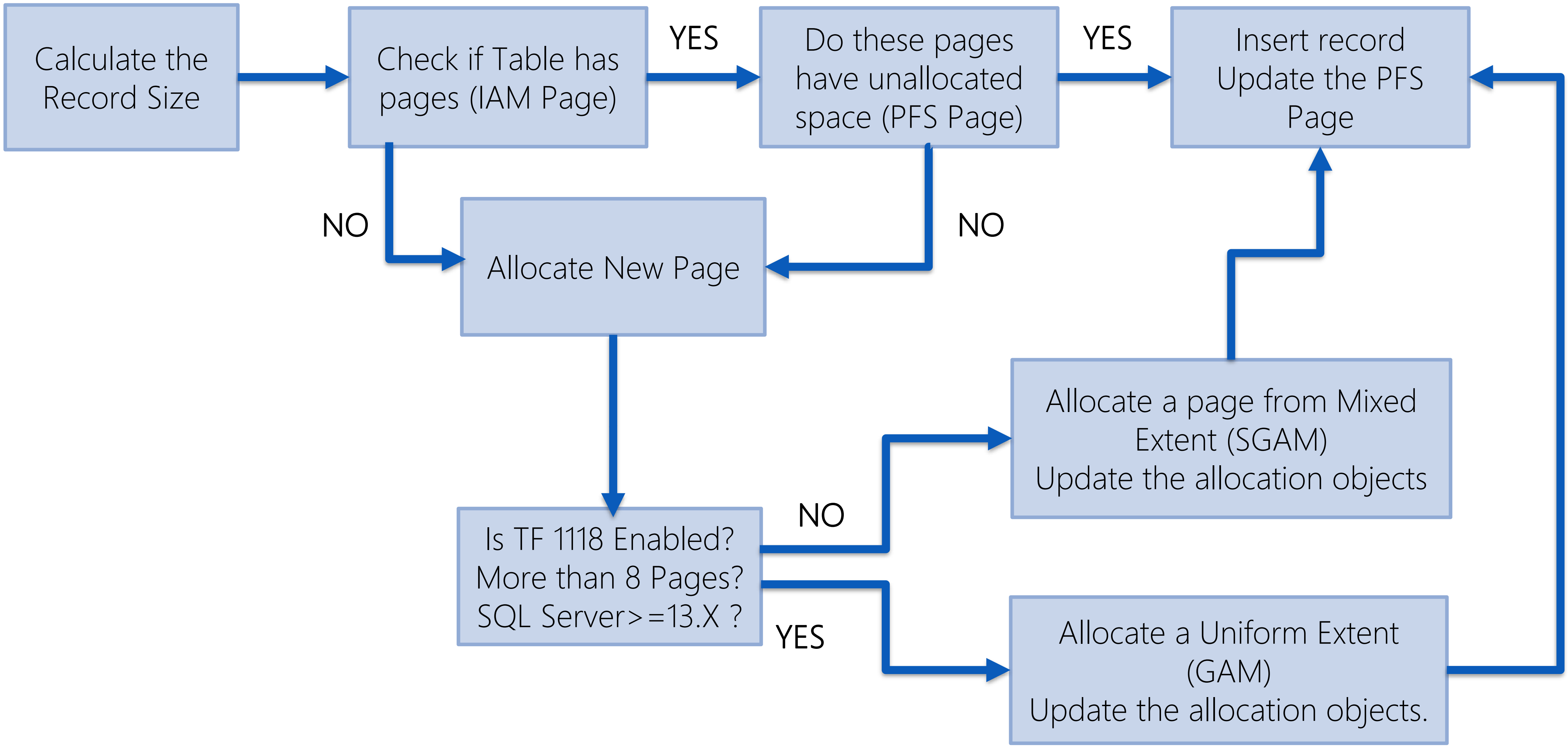
- Tracks Mixed extents that have all 8 pages allocated uses 1 Bit/extent
- Covers 64,000 extents (4 GB worth of data)

Index Allocation Map (IAM)

- Tracks extents that are allocated to an allocation unit
- Covers 4 GB worth of data
- One IAM chain per table, per index, per partition, per allocation unit type



Tying it all together



Calculate the Record Size

Check if Table has pages (IAM Page)

YES

Do these pages have unallocated space (PFS Page)

YES

Insert record Update the PFS Page

NO

Allocate New Page

NO

Is TF 1118 Enabled? More than 8 Pages? SQL Server>=13.X ?

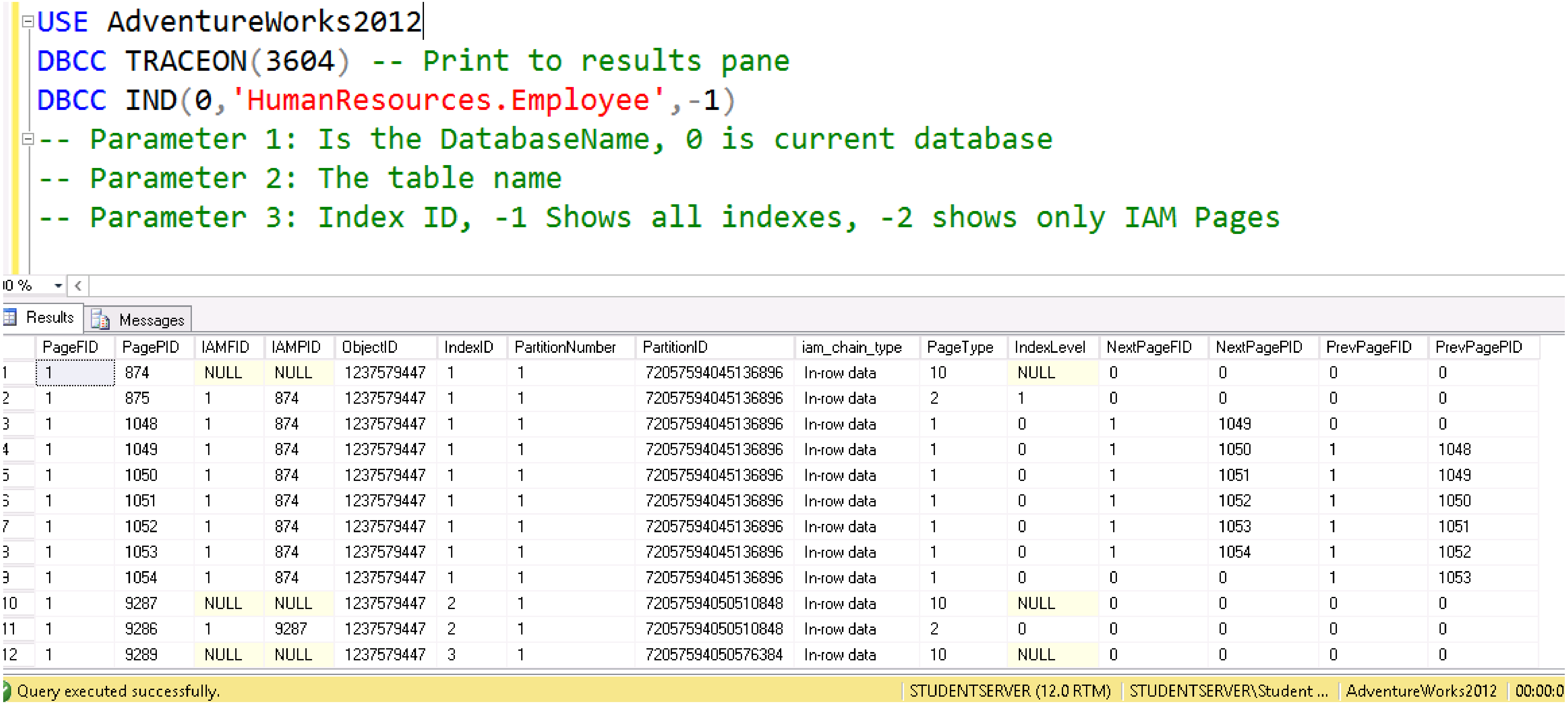
NO

Allocate a page from Mixed Extent (SGAM) Update the allocation objects

YES

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

DBCC IND

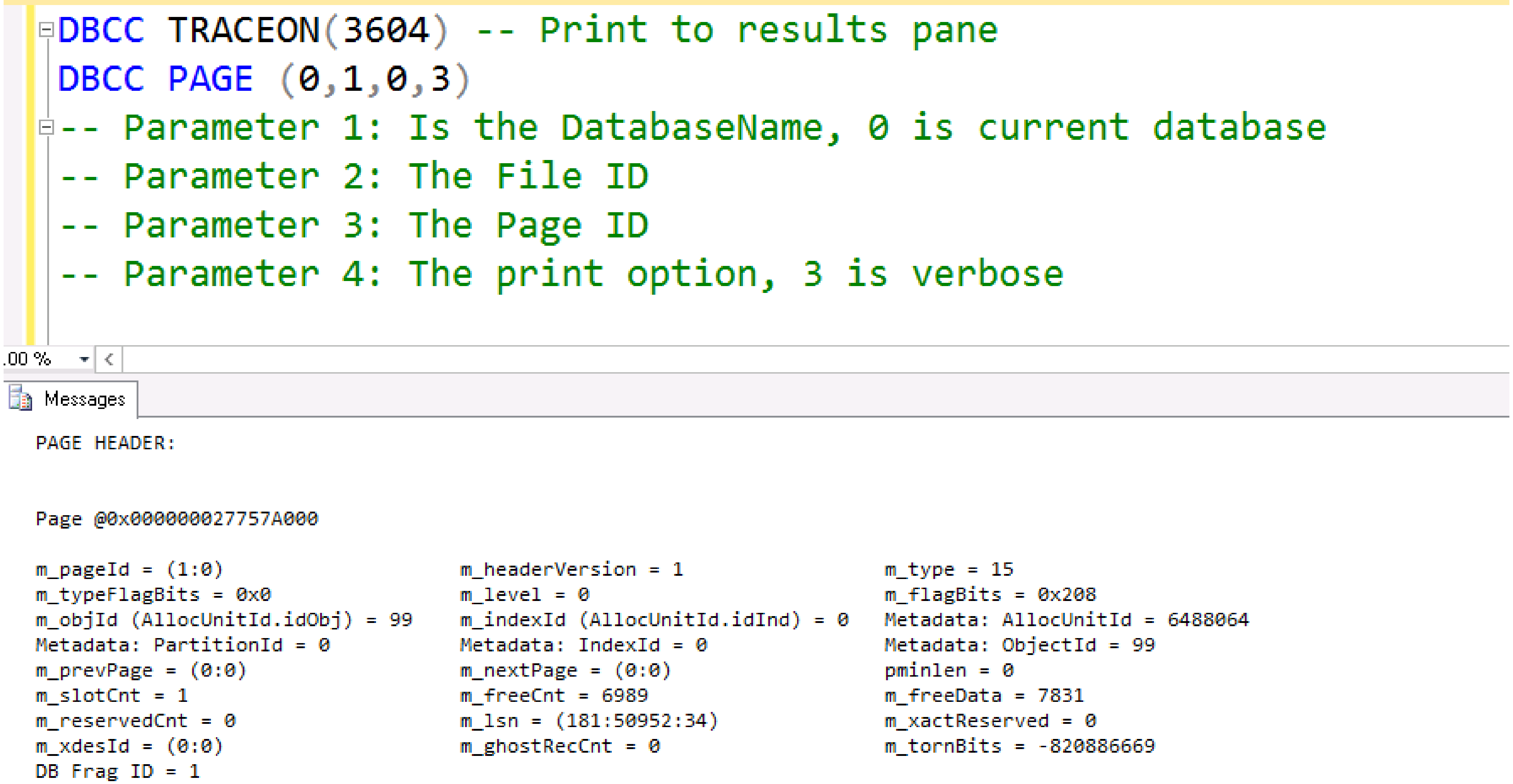


```sql
USE AdventureWorks2012
DBCC TRACEON(3604) -- Print to results pane
DBCC IND(0,'HumanResources.Employee',-1)
-- Parameter 1: Is the DatabaseName, 0 is current database
-- Parameter 2: The table name
-- Parameter 3: Index ID, -1 Shows all indexes, -2 shows only IAM Pages
```

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	874	NULL	NULL	1237579447	1	1	72057594045136896	In-row data	10	NULL	0	0	0	0
2	1	875	1	874	1237579447	1	1	72057594045136896	In-row data	2	1	0	0	0	0
3	1	1048	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1049	0	0
4	1	1049	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1050	1	1048
5	1	1050	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1051	1	1049
6	1	1051	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1052	1	1050
7	1	1052	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1053	1	1051
8	1	1053	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1054	1	1052
9	1	1054	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	0	0	1	1053
10	1	9287	NULL	NULL	1237579447	2	1	72057594050510848	In-row data	10	NULL	0	0	0	0
11	1	9286	1	9287	1237579447	2	1	72057594050510848	In-row data	2	0	0	0	0	0
12	1	9289	NULL	NULL	1237579447	3	1	72057594050576384	In-row data	10	NULL	0	0	0	0

Query executed successfully. | STUDENTSERVER (12.0 RTM) | STUDENTSERVER\Student ... | AdventureWorks2012 | 00:00:0

DBCC PAGE



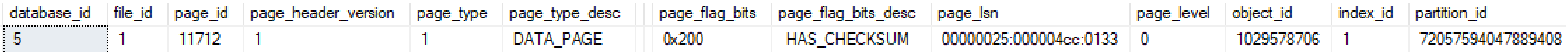
```sql
DBCC TRACEON(3604) -- Print to results pane
DBCC PAGE (0,1,0,3)
-- Parameter 1: Is the DatabaseName, 0 is current database
-- Parameter 2: The File ID
-- Parameter 3: The Page ID
-- Parameter 4: The print option, 3 is verbose
```

```
PAGE HEADER:


Page @0x000000027757A000

m_pageId = (1:0)                    m_headerVersion = 1                 m_type = 15
m_typeFlagBits = 0x0                m_level = 0                         m_flagBits = 0x208
m_objId (AllocUnitId.idObj) = 99    m_indexId (AllocUnitId.idInd) = 0   Metadata: AllocUnitId = 6488064
Metadata: PartitionId = 0           Metadata: IndexId = 0               Metadata: ObjectId = 99
m_prevPage = (0:0)                  m_nextPage = (0:0)                  pminlen = 0
m_slotCnt = 1                       m_freeCnt = 6989                    m_freeData = 7831
m_reservedCnt = 0                   m_lsn = (181:50952:34)              m_xactReserved = 0
m_xdesId = (0:0)                    m_ghostRecCnt = 0                   m_tornBits = -820886669
DB Frag ID = 1
```

New DMF

sys.dm_db_page_info

- Returns information about a page in a database.
- The function returns one row that contains the **header** information from the page, including the object_id, index_id, and partition_id.
- This function replaces the need to use DBCC PAGE in most cases.
- Output sample:

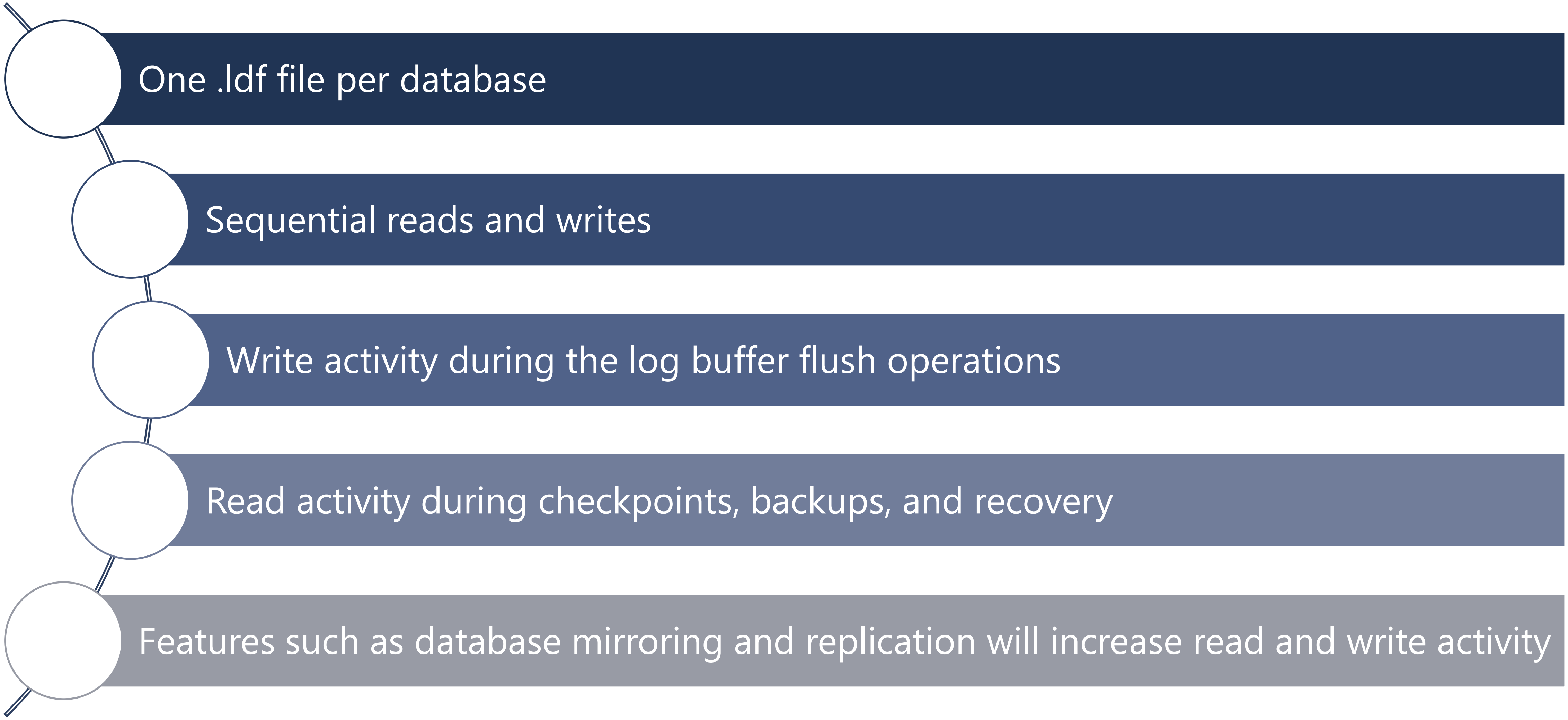


database_id	file_id	page_id	page_header_version	page_type	page_type_desc	page_flag_bits	page_flag_bits_desc	page_lsn	page_level	object_id	index_id	partition_id
5	1	11712	1	1	DATA_PAGE	0x200	HAS_CHECKSUM	00000025:000004cc:0133	0	1029578706	1	72057594047889408

sys.dm_db_page_info is currently supported only in SQL Server 2019 (15.x) and later.

Lesson 3: Transaction Log File Structures

SQL Server disk I/O patterns: transaction log



- One .ldf file per database
- Sequential reads and writes
- Write activity during the log buffer flush operations
- Read activity during checkpoints, backups, and recovery
- Features such as database mirroring and replication will increase read and write activity

Transaction Log

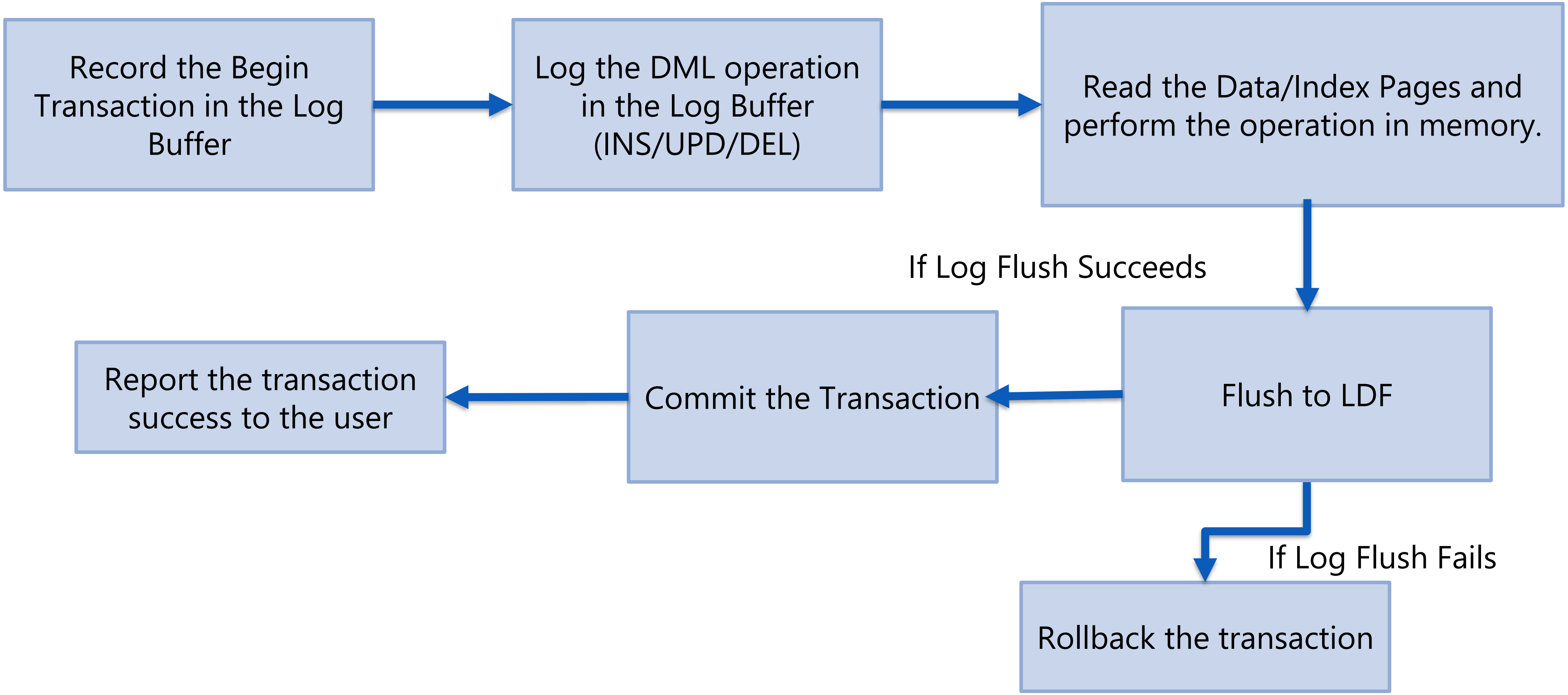
Sequence of log records contained in one or more physical files

Records identified by increasing Logical Sequence Numbers (LSNs)

Recorded operations:

- Start and end of each transaction
- All database modifications
- Extent and page allocation or deallocation
- Creating and dropping objects

Microsoft SQL Server Transaction Logging



Record the Begin Transaction in the Log Buffer
Log the DML operation in the Log Buffer (INS/UPD/DEL)
Read the Data/Index Pages and perform the operation in memory.
If Log Flush Succeeds
Flush to LDF
Commit the Transaction
Report the transaction success to the user
If Log Flush Fails
Rollback the transaction

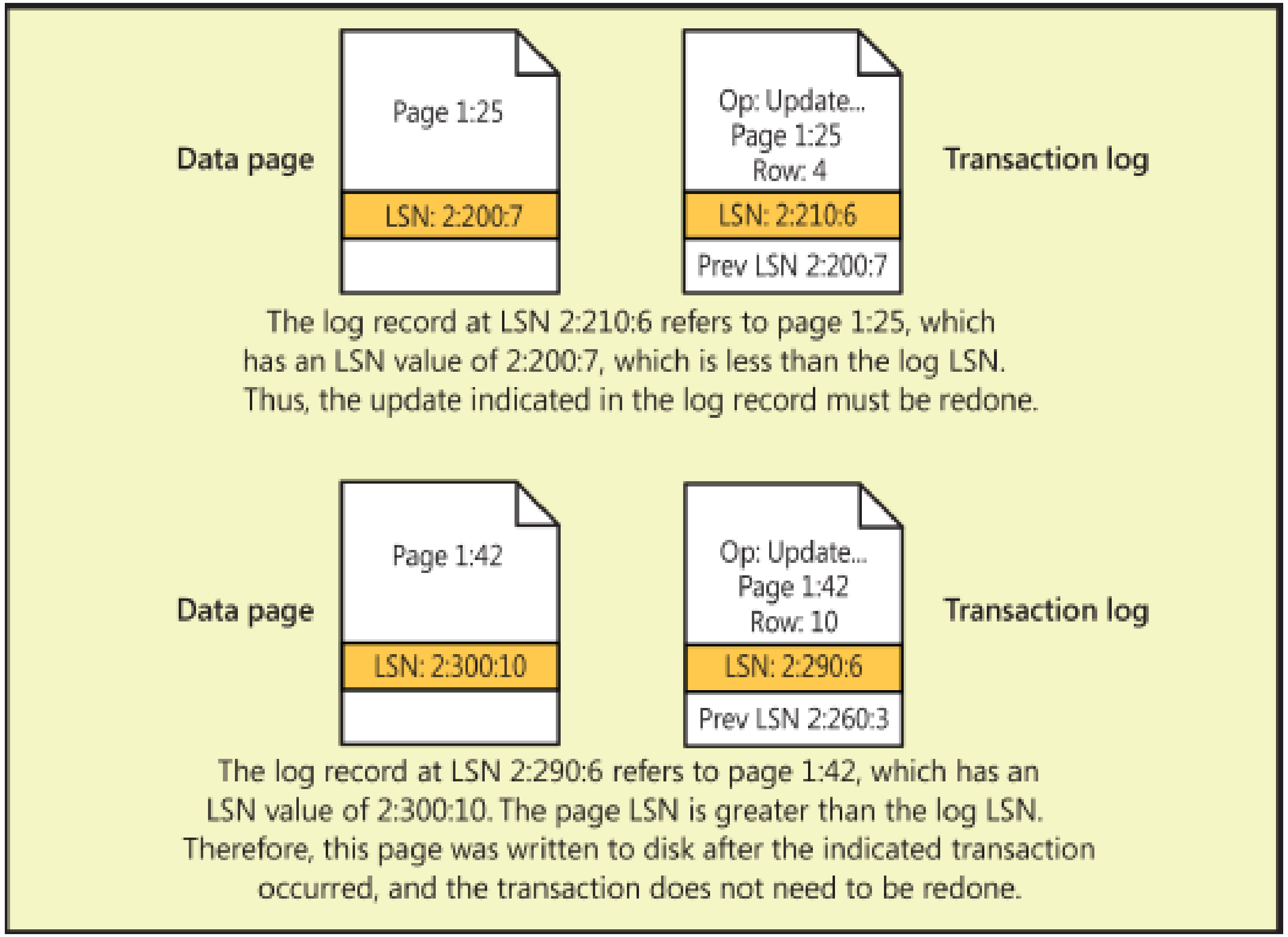
Page LSN and recovery

Last LSN in the page header

Log Record has two LSNs

- Previous LSN from the data page
- Current log record LSN

All used for recovery

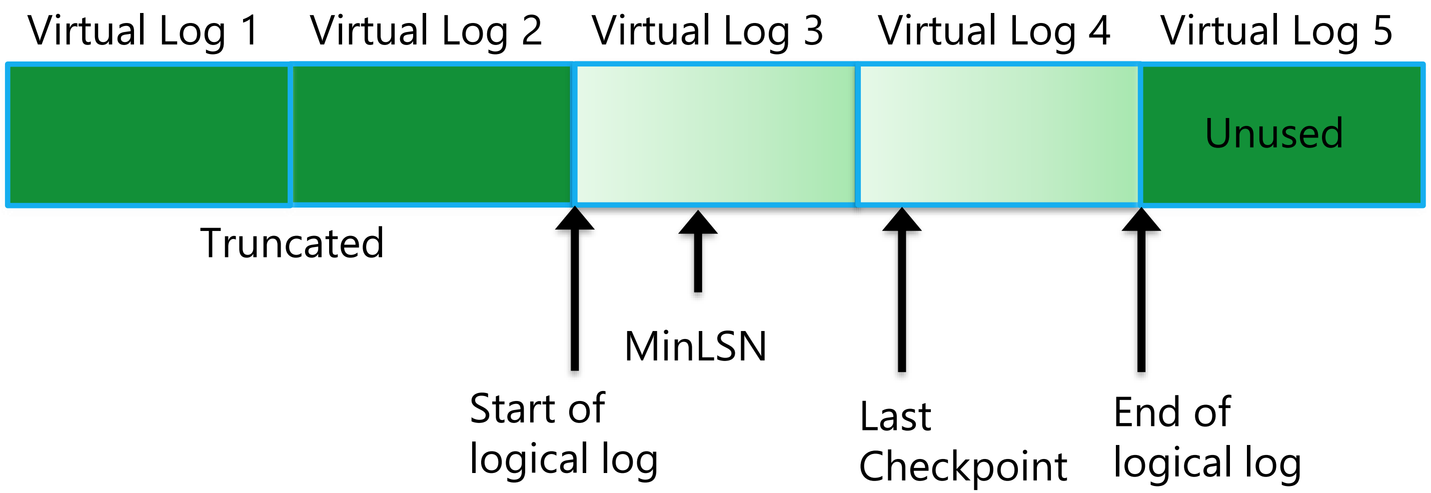


The log record at LSN 2:210:6 refers to page 1:25, which has an LSN value of 2:200:7, which is less than the log LSN. Thus, the update indicated in the log record must be redone.

The log record at LSN 2:290:6 refers to page 1:42, which has an LSN value of 2:300:10. The page LSN is greater than the log LSN. Therefore, this page was written to disk after the indicated transaction occurred, and the transaction does not need to be redone.

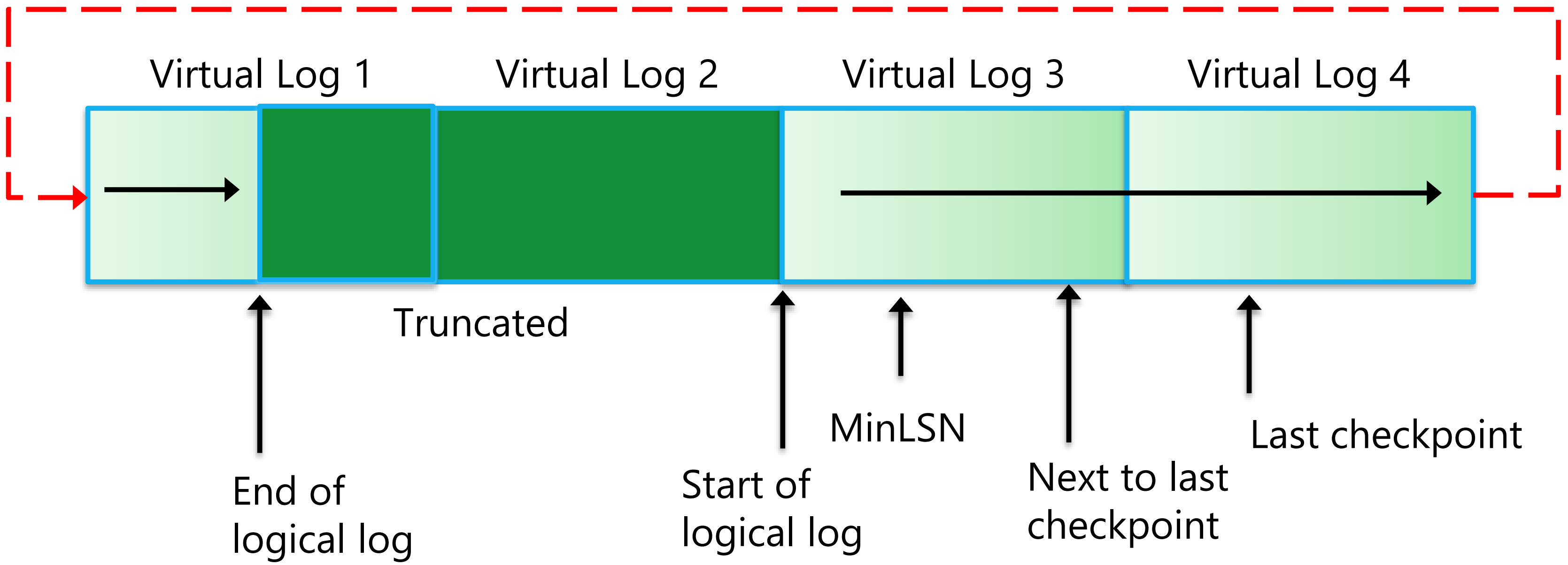
Physical Log File Structure

- The physical file is divided into virtual log files (VLFs).
- SQL Server works to keep the number of VLFs small.
- VLF size and number is dynamic and cannot be configured or set.



Physical Log File Structure (Continued)

- Log file is circular.
- When the end of logical log meets the beginning, the physical log file is extended.



Addressing VLF Counts

VLF count T-LOG file creation

Size <= 64 MB → 4 VLFs

Between 64 MB and 1 GB → 8 VLFs

Size > 1GB →16 VLFs

VLF count at T-LOG file growth

Growth <= 64 MB → 4 VLFs

Between 64 MB and 1GB → 8 VLFs

Growth > 1GB → 16 VLFs

(SQL 2014 and higher)

If growth is less than 1/8th of current size → add 1 VLF

SQL Server 2014 VLF Growth Improvement

- Is the growth size less than 1/8 the size of the current log size?
  - Yes: create 1 new VLF equal to the growth size
  - No: use the previous formula
- Example of a 256 MB log file with an autogrowth setting of 5 MB
  - 2012 and earlier: 10 auto-grows of 5MB would add 4 VLFs x 10 auto-grows
  - 2014 and later: 10 auto-grows of 5MB each would only create 10 VLFs

Grow Iterations + Log size	Up to SQL Server 2012	From SQL Server 2014
0 (256 MB)	8	8
10 (306 MB)	48	18
20 (356 MB)	88	28
80 (656 MB)	328	88
250 (1.2 GB)	1008	258
3020 (15 GB)	12091	3028

Addressing VLF Counts

To avoid VLF fragmentation

- Pre-size log files to avoid unplanned file growth
- Set autogrowth to an appropriate fixed size

To view VLFs

- DBCC LOGINFO (<database_id>)
- sys.dm_db_log_info (SQL Server 2016 SP2 and higher)
- Active VLFs have a status of 2
- SQL Server Error Log (SQL 2012 and higher)

Lesson 4: SQL Server TempDB File Structure

TempDB database

System database

- Available to all users with the same structure as user databases.
- Operations are minimally logged.
- Re-created every time SQL Server is started.

Workload

- Used for temporary (non-durable) storage.
- Object and data frequently being created and destroyed.
- Very high concurrency.
- Backup and restore operations are not allowed on TempDB.

What is stored in TempDB?

Temporary user objects

- Global or local temporary tables and indexes
- Temporary stored procedures
- Table variables
- Tables returned in table-valued functions, or cursors

Internal objects

- Worktables to store intermediate results for spools, cursors, sorts, and temporary LOB storage.
- Work files for hash join or hash aggregate operations.
- Intermediate sort results for operations such as creating or rebuilding indexes (if SORT_IN_TEMPDB is specified), or certain GROUP BY, ORDER BY, or UNION queries.

Version stores

- Common version store
- Online-index-build version store

Physical properties of TempDB

File	Logical name	Physical name	Initial size	File growth
Primary data	tempdev	tempdb.mdf	8 MB	Autogrow by 64 MB until the disk is full
Secondary data files*	temp#	tempdb_mssql_#.ndf	8 MB	Autogrow by 64 MB until the disk is full
Log	templog	templog.ldf	8 MB	Autogrow by 64 megabytes to a maximum of 2 terabytes

If the number of logical processors is less than or equal to eight (8), use the same number of data files as logical processors.

If the number of logical processors is greater than eight (8), use eight data files.

If contention continues, increase the number of data files by multiples of four (4) up to the number of logical processors

Alternatively, make changes to the workload or code.

Optimizing TempDB performance

- Consider instant file initialization
- Pre-allocate space for all TempDB files
- Divide TempDB into multiple data files of equal size
- Put the TempDB database on a fast I/O subsystem
- Use disk striping if there are many directly attached disks.
- Put the TempDB database on separate disks from user databases

Performance improvements in TempDB

Starting with SQL Server 2016 (13.x)

Trace Flags 1117 and 1118 behavior enabled by default for TempDB

Temporary tables and table variables are cached.

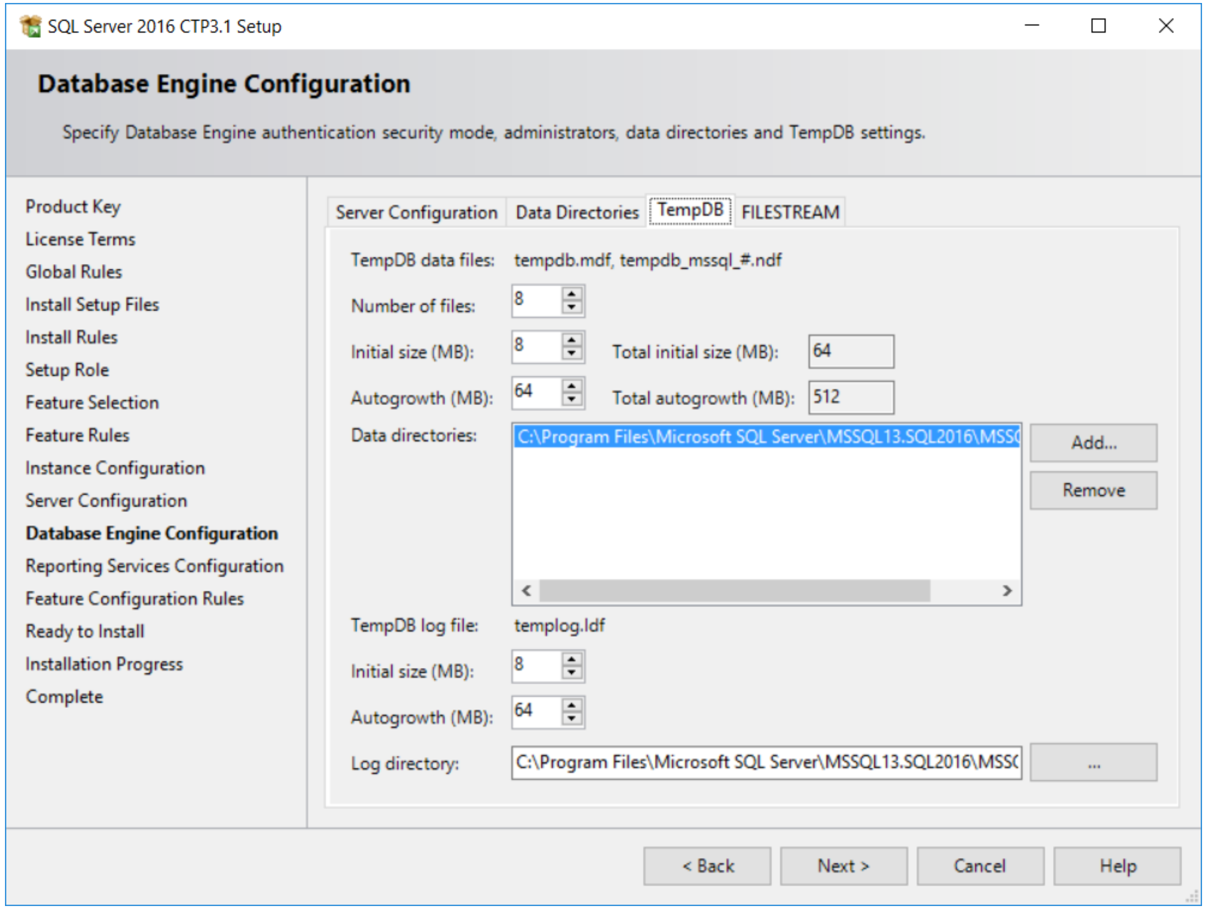
Allocation page latching protocol improved.

Logging overhead for TempDB is reduced.

Performance improvements in TempDB

Starting with SQL Server 2016 (13.x)

- Setup adds multiple TempDB data files during instance installation.
- By default, setup adds as many TempDB data files as the logical processor count or eight, whichever is lower.



SQL Server 2016 CTP3.1 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators, data directories and TempDB settings.

TempDB data files: tempdb.mdf, tempdb_mssql_#.ndf

Number of files: 8

Initial size (MB): 8 Total initial size (MB): 64

Autogrowth (MB): 64 Total autogrowth (MB): 512

Data directories: C:\Program Files\Microsoft SQL Server\MSSQL13.SQL2016\MSS(

TempDB log file: templog.ldf

Initial size (MB): 8

Autogrowth (MB): 64

Log directory: C:\Program Files\Microsoft SQL Server\MSSQL13.SQL2016\MSS(

Performance improvements in TempDB

Starting with SQL Server 2019 (15.x)

Default

- Temp table cache improvements
- Concurrent PFS updates

Opt-in

- Memory-Optimized TempDB Metadata

TEMPDB Files, Trace Flags, and Updates by Pam Lahoud

How (and When) To: Memory Optimized TempDB

Microsoft

Module 3: Basic Disaster Recovery

Lesson 1: Full, Differential, and Log Backups

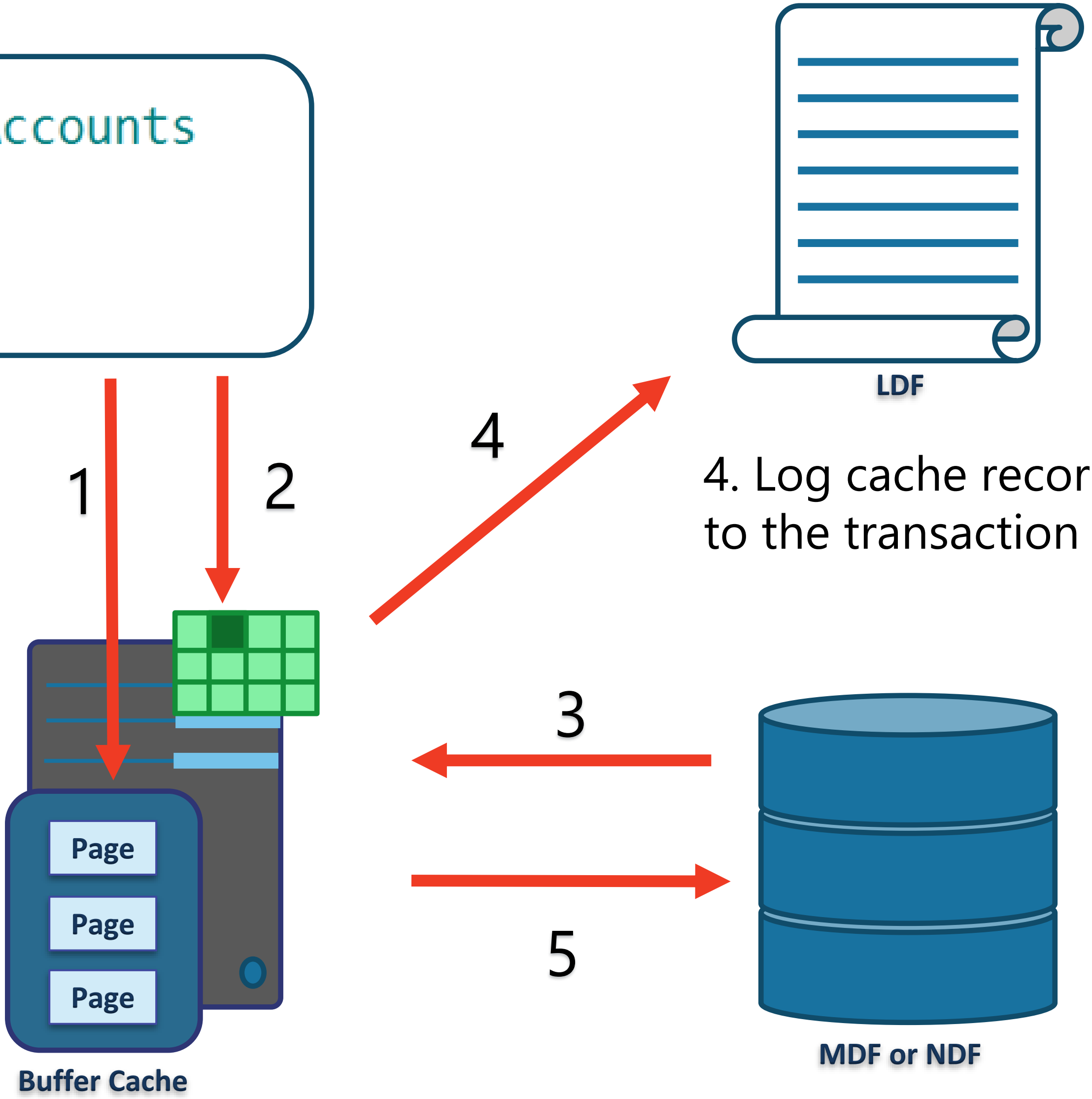
Overview of SQL Server Transaction Logs

```
UPDATE Accounting.BankAccounts
SET Balance -= 200
WHERE AcctID = 1
```

1. Data modification is sent to buffer cache in memory.

2. Modification is recorded in the log cache.

3. Data pages are located or read into the buffer cache and then modified.



4. Log cache record is flushed to the transaction log

5. At checkpoint, dirty data pages are written to the database file.

Checkpoints

Flushes dirty pages from the buffer pool to the disk. Frequency of checkpoints varies based on the database activity and recovery interval.

Automatic (default) – Database engine issues checkpoints automatically based on the server level "recovery interval" configuration option

Indirect (new in SQL Server 2012) – Database engine issues checkpoints automatically based on the database level TARGET_RECOVERY_TIME

```
ALTER DATABASE [AdventureWorksPTO] SET TARGET_RECOVERY_TIME = 60 SECONDS
```

Manual – Issued in the current database for your connection when you execute the T-SQL CHECKPOINT command

Internal – Issued by various server operations

Working with Recovery Models

Recovery Model	Description
Simple	• Does not permit or require log backups • Automatically truncates log to keep space requirements small
Full	• Requires log backups for manageability • Avoids data loss due to a damaged or missing data file • Permits recovery to a specified point in time
Bulk Logged	• Requires log backups for manageability • Can enhance the performance of bulk copy operations • Reduces log space usage by using minimal logging for many bulk operations

Backup Types

Backup type	Description
Full	All data files and the active part of the transaction log
Differential	The parts of the database that have changed since the last full database backup
Transaction Log	Any database changes recorded in the log files
File/File Group	Specified files or filegroups
Partial	The primary filegroup, every read/write filegroup, and any specified read-only filegroups
Tail-log Backup	Log backup taken of the tail of the log just before a restore operation
Copy Only	The database or log (without affecting the backup sequence)

Determining Recovery Objectives

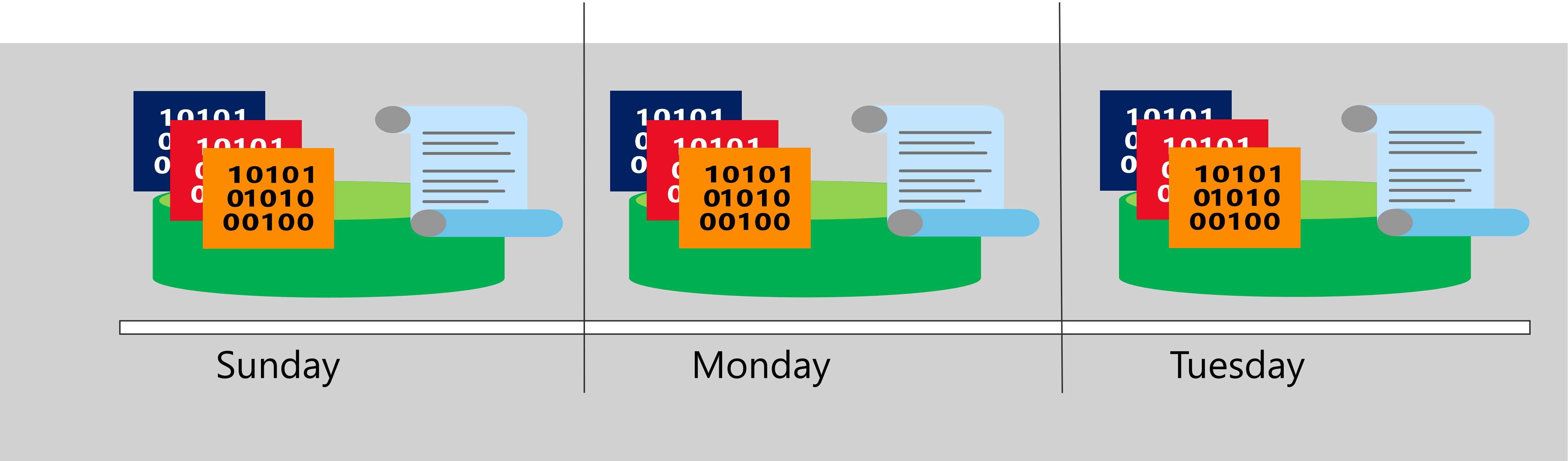
Determine safety levels:

- How long can recovering take? (RTO)
- How much data is it acceptable to lose? (RPO)
- Is it possible to recover the data from other sources?

Backup strategy should map to requirements:

- Types and frequency of backups
- Backup media to use
- Retention period for backups and for media
- Backup testing policy

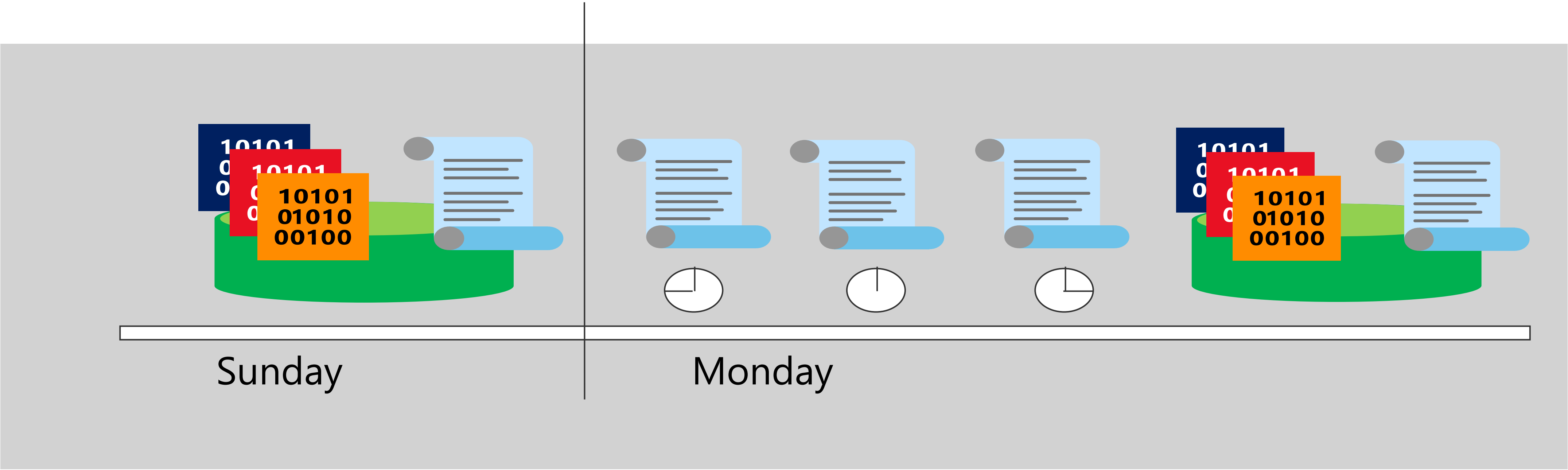
Full Database Backup Strategies



Full database backups:

- Back up all data and part of the log records
- Can be used to restore the whole database
- Permit recovery to backup times only

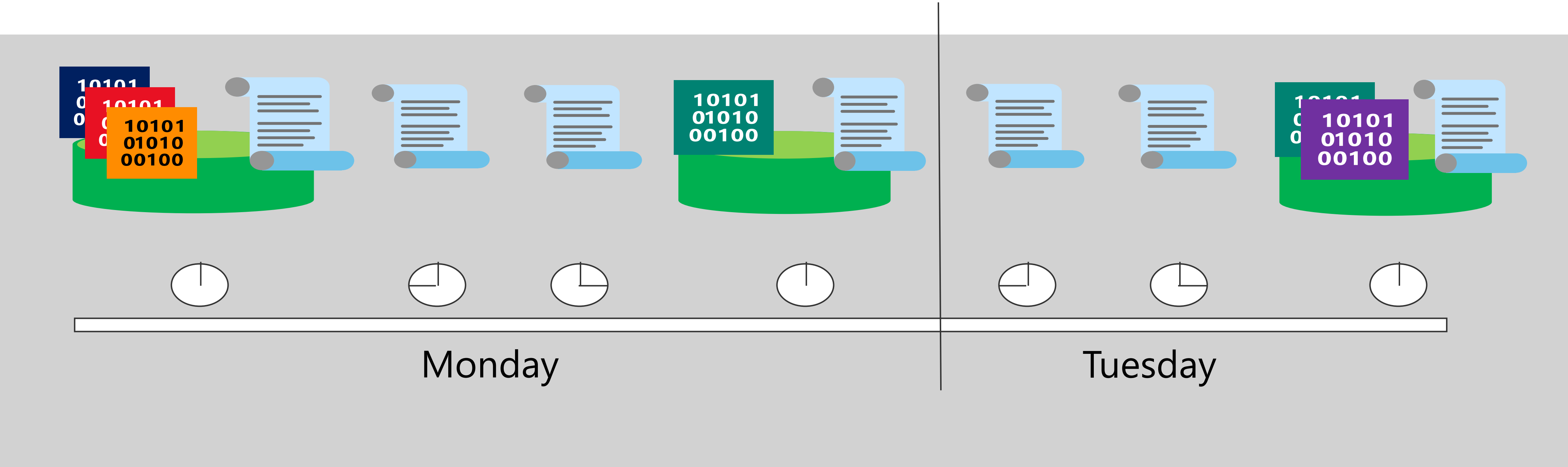
Transaction Log Backup Strategies



A database and transaction log backup strategy:

- Involves at least full and transaction log backups
- Enables point-in-time recovery
- Allows the database to be fully restored in the case of data file loss

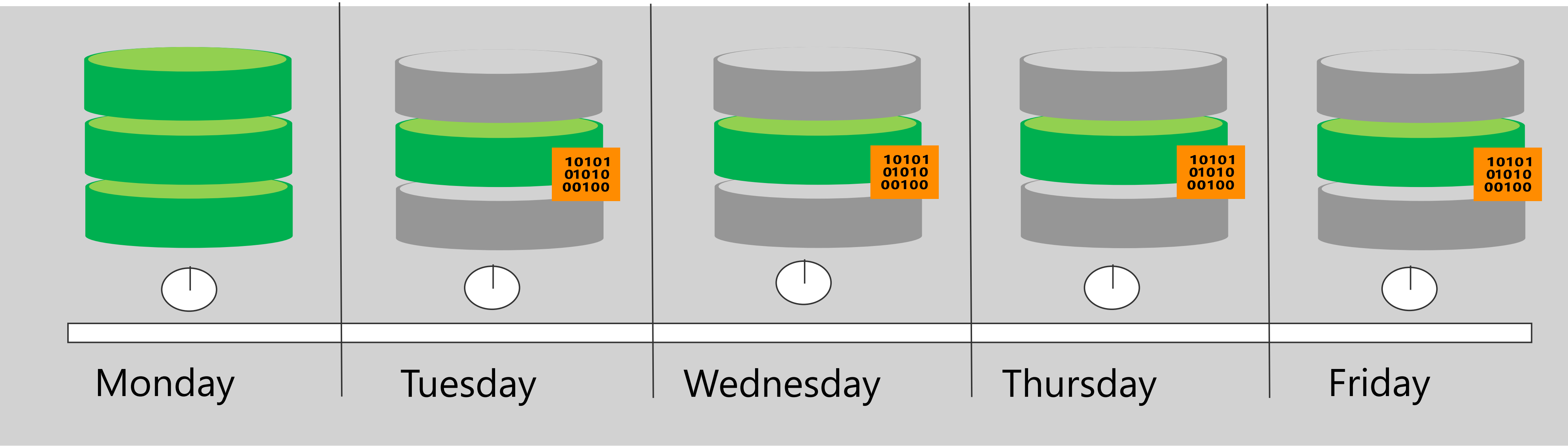
Differential Backup Strategies



A differential backup strategy:

- Involves performing full and differential database backups
- Includes differential backups with only changed data
- Is useful if only a subset of a database is modified more frequently than the rest of the database

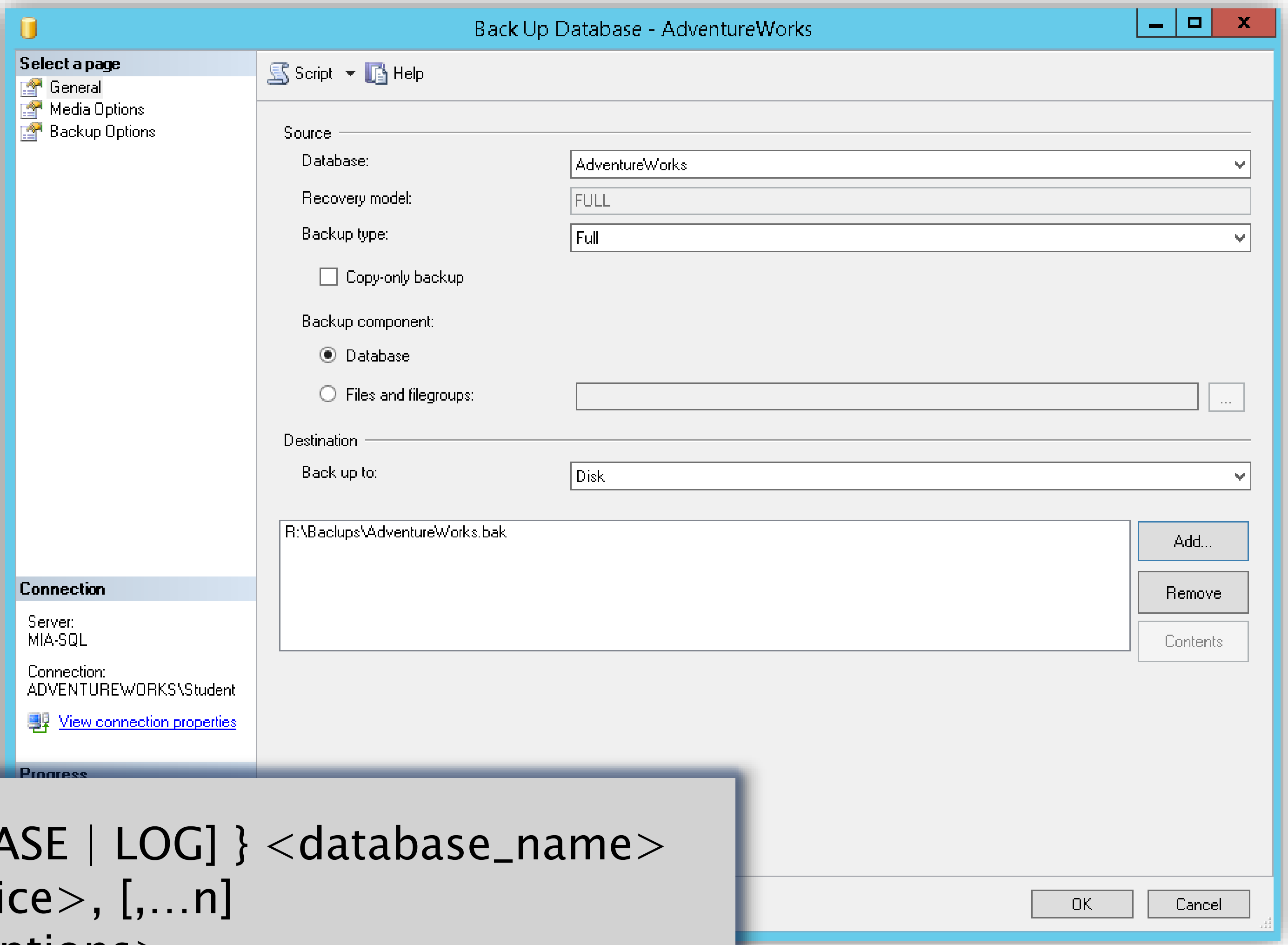
Partial Backup Strategies



- Faster backup and restore for very large databases
- Can be complex to set up and manage

Lesson 2: Native SQL Backup Process

Introduction to SQL Server Backup


Back Up Database - AdventureWorks

Source

Database: AdventureWorks

Recovery model: FULL

Backup type: Full

□ Copy-only backup

Backup component:

● Database

○ Files and filegroups:

Destination

Back up to: Disk

R:\Baclups\AdventureWorks.bak

Server:
MIA-SQL

Connection:
ADVENTUREWORKS\Student

View connection properties

```
BACKUP { DATABASE | LOG] } <database_name>
TO <backup_device>, [,…n]
WITH <general_options>
```

Media Sets and Backup Sets

Media sets consist of one or more disk backup devices

- Data is striped across multiple devices

A backup set represents one backup of any type

Backup sets are written to media sets

- A media set can contain multiple backup sets

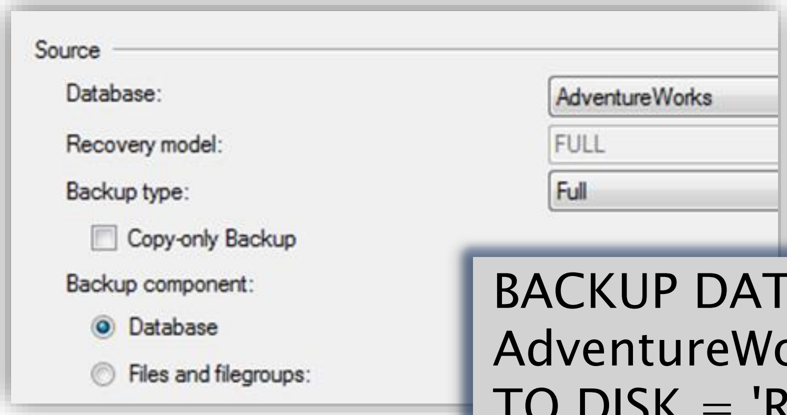
Backup devices and media sets are created at first use

- Use FORMAT to overwrite an existing media set
- Use INIT to overwrite existing backup sets in a media set
- Use the FORMAT option with caution

Performing Database Backups

Full Backup:

- Entire database
- Active portion of log file



Source

Database: AdventureWorks

Recovery model: FULL

Backup type: Full

□ Copy-only Backup

Backup component:

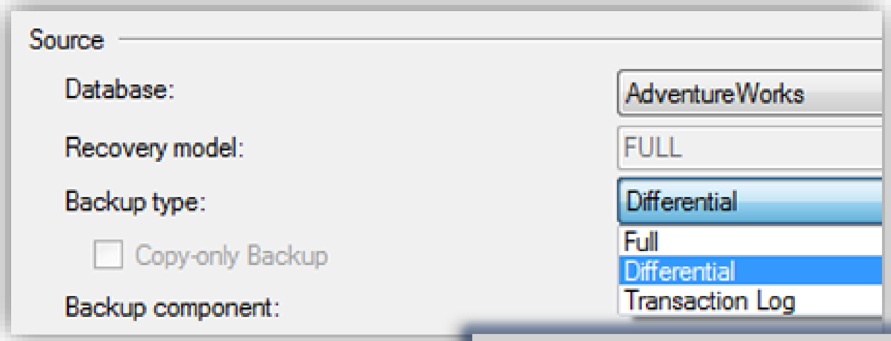
● Database

○ Files and filegroups:

```
BACKUP DATABASE
AdventureWorks
TO DISK = 'R:\Backups\AW.bak'
WITH INIT;
```

Differential Backup

- Extents modified since the last full database backup
- Active portion of log file



Source

Database: AdventureWorks

Recovery model: FULL

Backup type: Differential

□ Copy-only Backup

Backup component:

```
BACKUP DATABASE
   AdventureWorks
TO DISK = 'R:\Backups\AW.bak'
WITH DIFFERENTIAL, NOINIT;
```

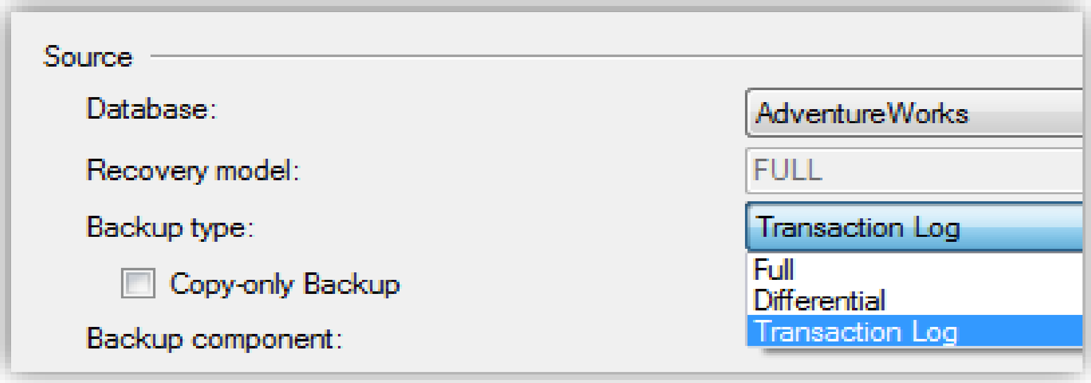
Performing Transaction Log Backups

Backs up only the transaction log

Backs up the log from the last successfully executed log backup to the current end of the log

Truncates inactive log records unless options specified

Database must be in full or bulk-logged recovery model



Source

Database: AdventureWorks

Recovery model: FULL

Backup type: Transaction Log

☐ Copy-only Backup

Backup component:

```
BACKUP LOG AdventureWorks
TO DISK = 'R:\Backups\AW.bak'
WITH NOINIT;
```

```
BACKUP LOG AdventureWorks
TO DISK = 'R:\Backups\AW.bak'
WITH [NORECOVERY | NO_TRUNCATE |
CONTINUE_ON_ERROR];
```

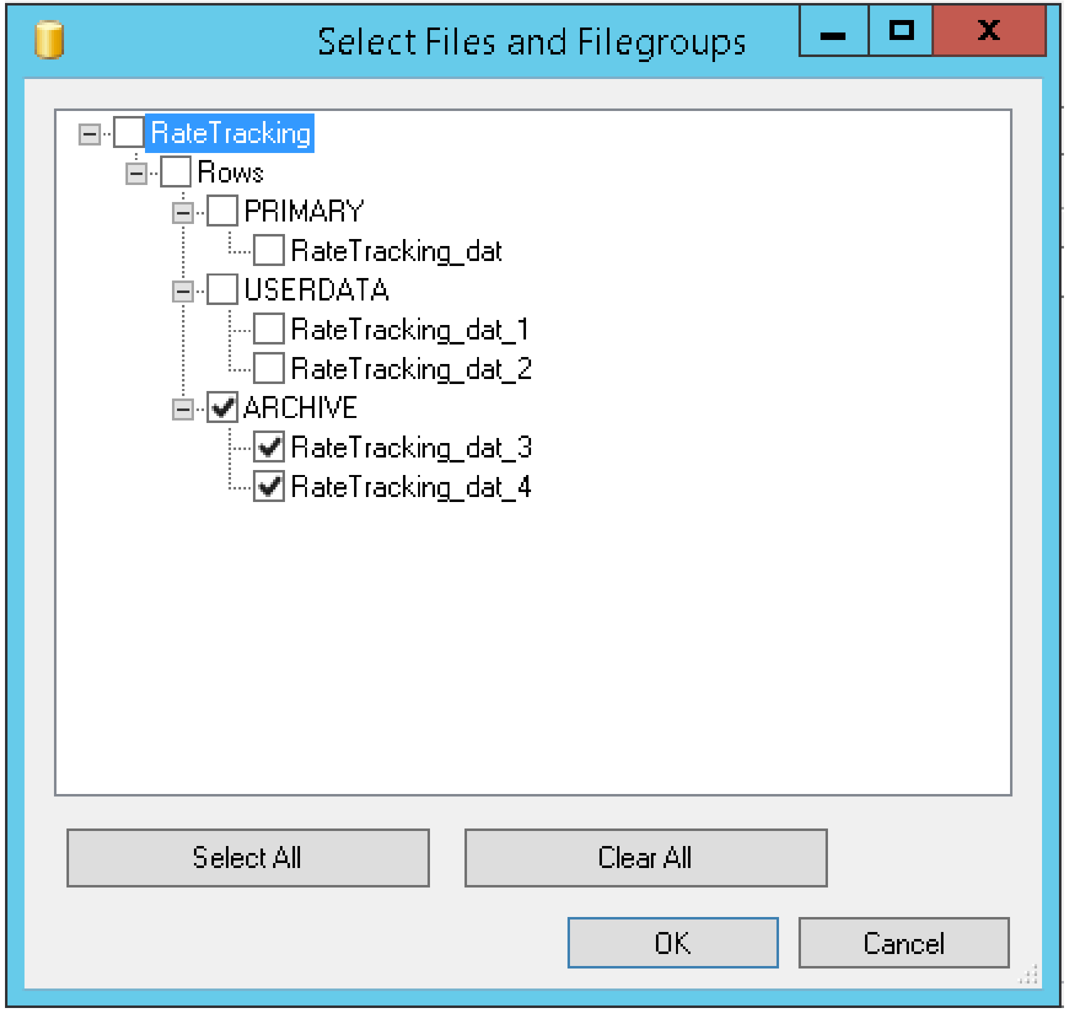
Performing Partial and Filegroup Backups

- Partial Backup
  - Primary filegroup
  - Read/Write filegroups

```
BACKUP DATABASE LargeDB
READ_WRITE_FILEGROUPS
TO DISK = 'R:\Backups\LrgRW.bak'
WITH INIT;
```

- File or Filegroup backup
  - Specific files or filegroups

```
BACKUP DATABASE LargeDB
FILEGROUP = 'FG2'
TO DISK = 'R:\Backups\LrgFG2.bak'
```



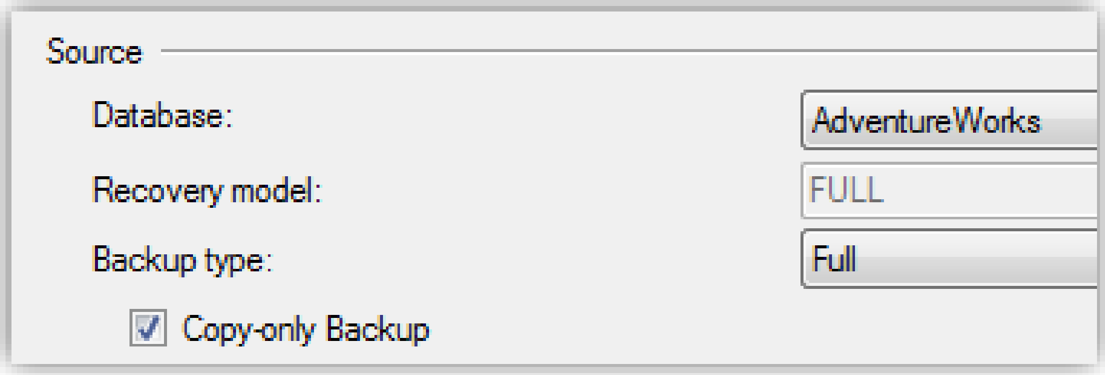
- ☐ RateTracking
  - ☐ Rows
    - ☐ PRIMARY
      - ☐ RateTracking_dat
    - ☐ USERDATA
      - ☐ RateTracking_dat_1
      - ☐ RateTracking_dat_2
    - ☑ ARCHIVE
      - ☑ RateTracking_dat_3
      - ☑ RateTracking_dat_4

Copy-Only Backups

Back up the database without changing the restore order

Copy-only transaction log backups do not truncate the log

Copy-only full database backups do not affect the differential base



Source

Database: AdventureWorks

Recovery model: FULL

Backup type: Full

☑ Copy-only Backup

```
BACKUP DATABASE AdventureWorks
TO DISK = 'Q:\Backups\AW_Copy.bak'
WITH COPY_ONLY, INIT;
```

Options for Ensuring Backup Integrity

Mirrored media sets:

- Can mirror a backup set to up to four media sets
- Mirrors require the same number of backup devices
- Only in Enterprise Edition

CHECKSUM backup option

- Available for all backup types
- Generates a checksum over the backup stream
- Use to verify the backup

Backup verification

- Can use RESTORE VERIFYONLY for backup verification
- Useful when combined with the CHECKSUM option

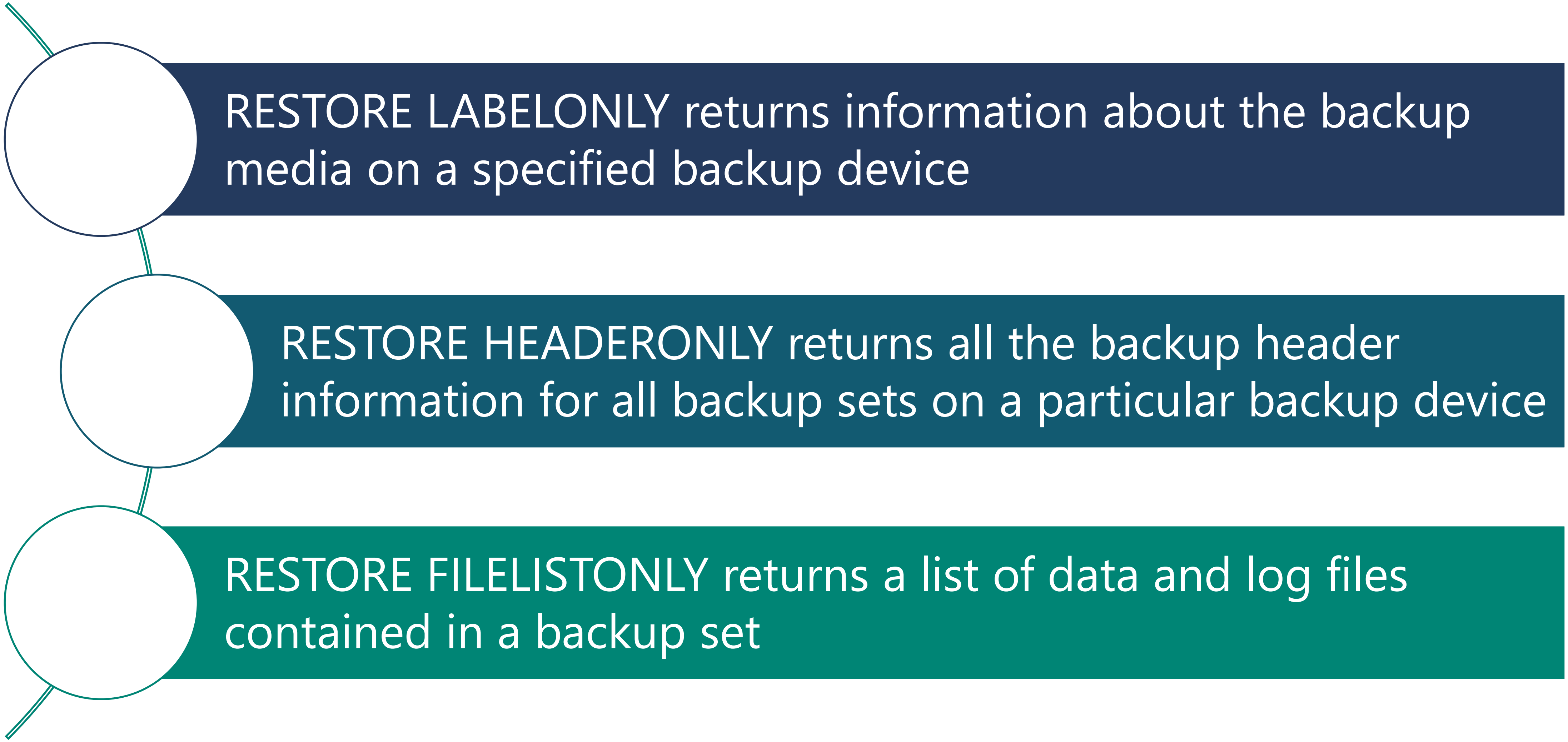
Viewing Backup History

- SQL Server tracks all backup activity in the **msdb** database

```
SELECT        bs.media_set_id, bs.backup_finish_date,  bs.type,
              bs.backup_size, bs.compressed_backup_size,
              mf.physical_device_name
FROM dbo.backupset AS bs
INNER JOIN dbo.backupmediafamily AS mf
ON bs.media_set_id = mf.media_set_id
WHERE database_name = 'AdventureWorks'
ORDER BY backup_finish_date DESC;
```

- The **Backup and Restore Events** report in SQL Server Management Studio displays detailed backup history information

Retrieving Backup Metadata



RESTORE LABELONLY returns information about the backup media on a specified backup device

RESTORE HEADERONLY returns all the backup header information for all backup sets on a particular backup device

RESTORE FILELISTONLY returns a list of data and log files contained in a backup set

Lesson 3: Restore and Recovery Process

Phases of the Restore Process

The restore process of a SQL Server database consists of three phases

Redo and undo are also known as recovery

Phase	Description
Data copy	Creates files and copies data to the files
Redo	Applies committed transactions from restored log entries
Undo	Rolls back transactions that were uncommitted at the recovery point

Types of Restores

Complete database restores:

- Simple recovery model
- Full recovery model

System database restore

Advanced restores:

- File or filegroup restore
- Piecemeal restore
- Encrypted backup restore
- Page restore

Preparations for Restoring Backups

Perform a tail-log backup if using full or bulk-logged recovery model

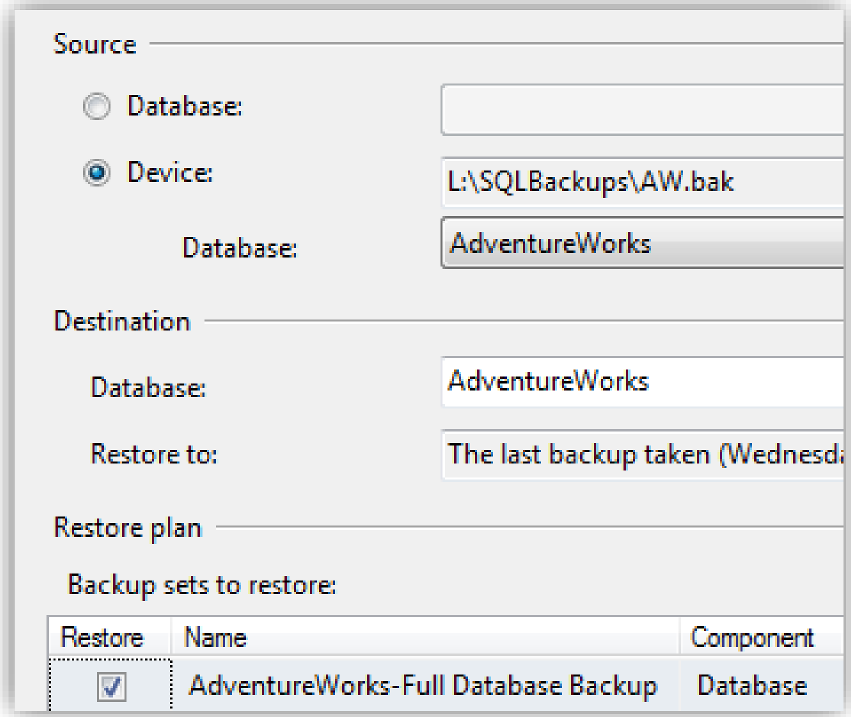
Identify the backups to restore:

- Last full, file, or filegroup backup
- Last differential backup, if exists
- Log backups if using full or bulk-logged recovery model

Restoring a Full Database Backup

- Restore databases in SQL Server Management Studio, or use the RESTORE DATABASE statement
  - Use WITH REPLACE to overwrite an existing database
  - Use WITH MOVE to relocate database files

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'R:\Backups\AW.bak';
```



Source

○ Database:

● Device: L:\SQLBackups\AW.bak

Database: AdventureWorks

Destination

Database: AdventureWorks

Restore to: The last backup taken (Wednesd

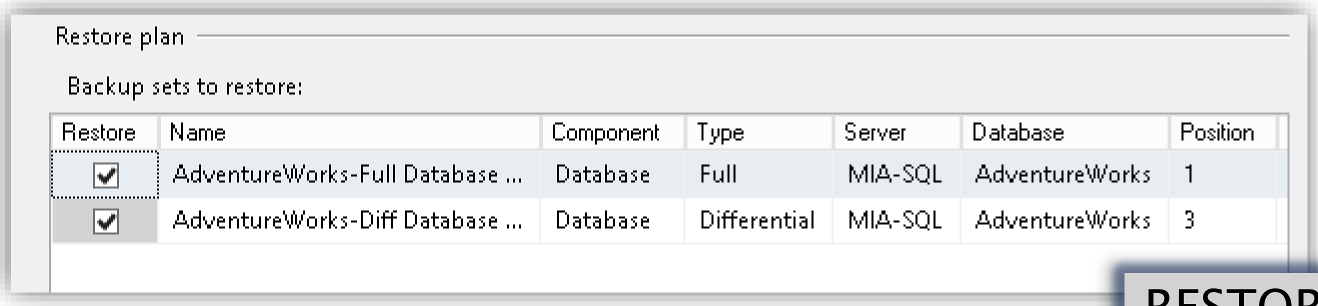
Restore plan

Backup sets to restore:

Restore	Name	Component
☑	AdventureWorks-Full Database Backup	Database

Restoring a Differential Backup

- Restore the latest full database backup WITH NORECOVERY
- Restore the latest differential backup WITH RECOVERY



Restore plan

Backup sets to restore:

Restore	Name	Component	Type	Server	Database	Position
☑	AdventureWorks-Full Database ...	Database	Full	MIA-SQL	AdventureWorks	1
☑	AdventureWorks-Diff Database ...	Database	Differential	MIA-SQL	AdventureWorks	3

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'R:\Backups\AW.bak'
WITH FILE = 1, NORECOVERY;

RESTORE DATABASE AdventureWorks
FROM DISK = 'R:\Backups\AW.bak'
WITH FILE = 3,  RECOVERY;
```

Restoring Transaction Log Backups

- Restore transaction logs by using the RESTORE LOG statement
- Restore the log chain chronologically
  - Use NORECOVERY for all but the last backup
  - Use RECOVERY for the last backup (often the tail-log backup)

```
-- Restore last full and differential database backups...
-- Restore planned log backups
RESTORE LOG AdventureWorks FROM DISK = 'R:\Backups\AW.bak'
WITH FILE = 5,  NORECOVERY;

-- Restore tail-log backup
RESTORE LOG AdventureWorks FROM DISK = 'R:\Backups\AW-TailLog.bak'
WITH RECOVERY;
```

Recovering System Databases

System database	Description
master	Backup required: Yes Recovery model: Simple Restore using single user mode
model	Backup required: Yes Recovery model: User configurable Restore using –T3608 trace flag
msdb	Backup required: Yes Recovery model: Simple (default) Restore like any user database
tempdb /resource	No backups can be performed tempdb is created during instance startup Restore resource using file restore or setup

Overview of Point-in-Time Recovery

- Enables recovery of a database up to any arbitrary point in time that is contained in the transaction log backups
- Point in time can be defined by a datetime value or a named transaction
- Database must be in FULL recovery model

STOPAT Option

Provide the STOPAT and WITH RECOVERY options as part of all RESTORE statements in the sequence:

- No need to know in which transaction log backup the requested point in time resides
- If the point in time is after the time included in the backup, a warning will be issued, and the database will not be recovered after the restore completes
- If the point in time is before the time included in the backup, the RESTORE statement fails
- If the point in time provided is within the time frame of the backup, the database is recovered up to that point

STOPATMARK Option

Transactions marked using:

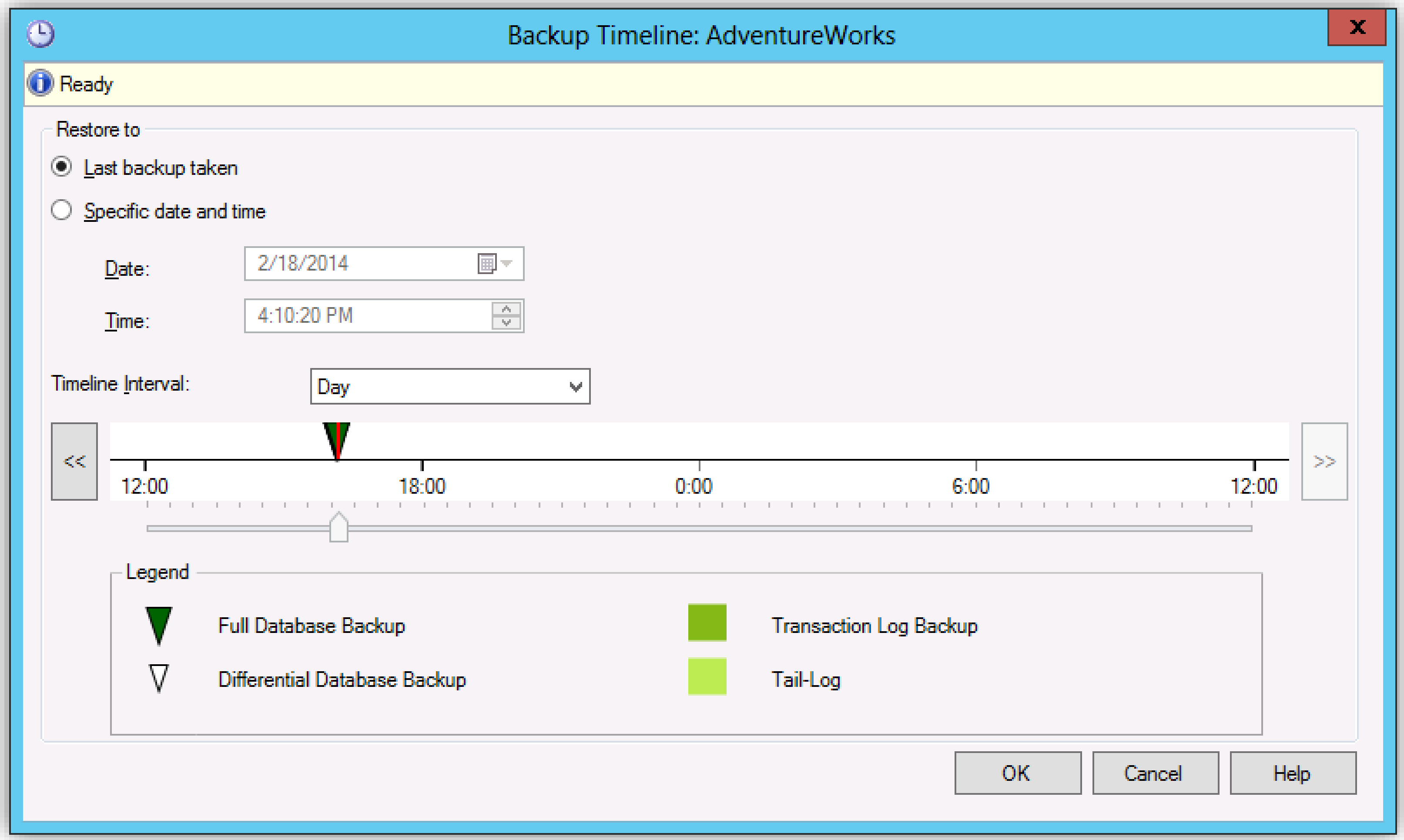
- BEGIN TRAN <name> WITH MARK <description>

Restore has two related options:

- STOPATMARK rolls forward to the mark and includes the marked transaction in the roll forward
- STOPBEFOREMARK rolls forward to the mark and excludes the marked transaction from the roll forward

If the mark is not present in the transaction log backup, the backup is restored, but the database is not recovered

Performing a Point-in-Time Recovery by Using SQL Server Management Studio



Backup Timeline: AdventureWorks

Ready

Restore to

● Last backup taken

○ Specific date and time

Date: 2/18/2014

Time: 4:10:20 PM

Timeline Interval: Day

Legend

▼ Full Database Backup

🟩 Transaction Log Backup

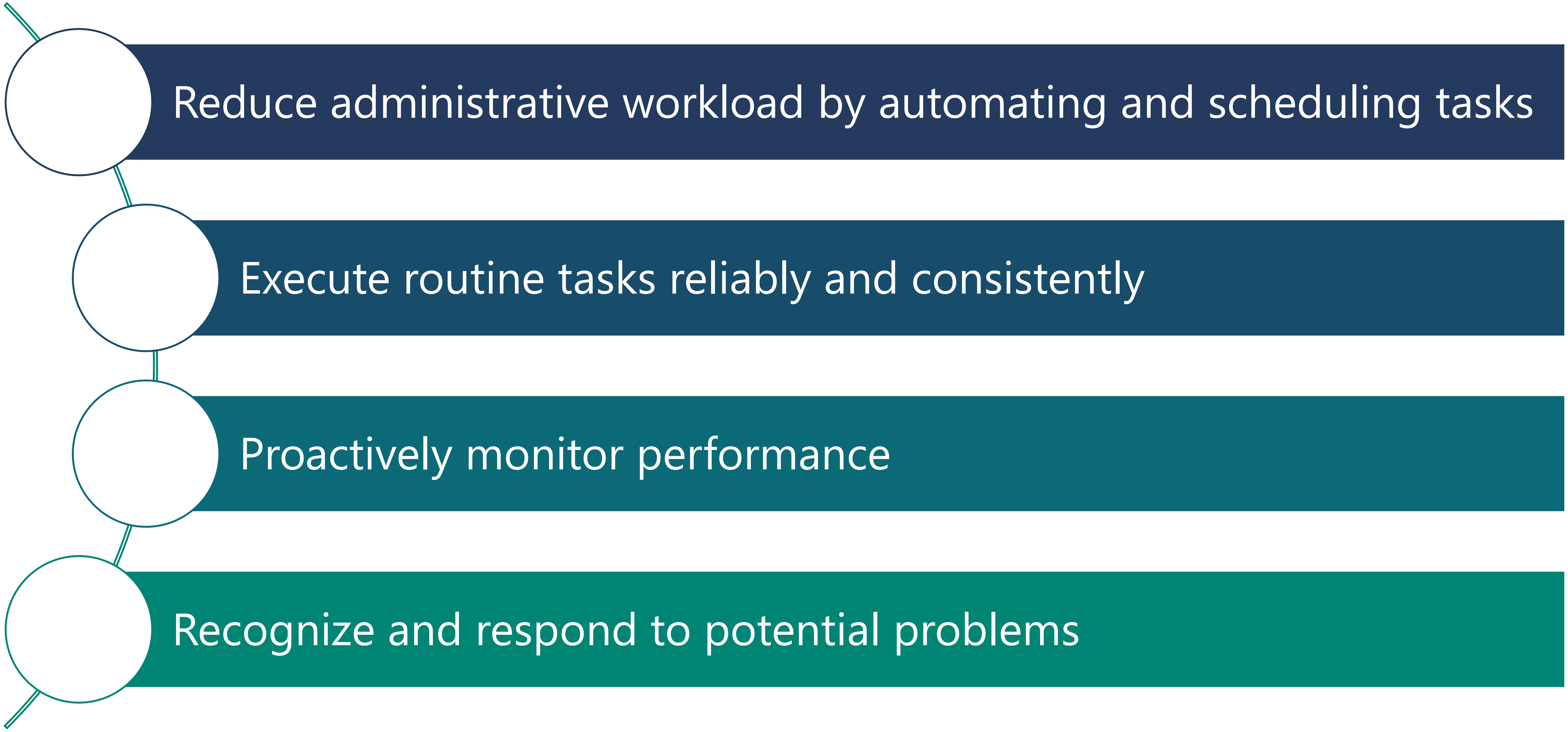
▽ Differential Database Backup

🟩 Tail-Log

Module 4: Automating SQL Server Management

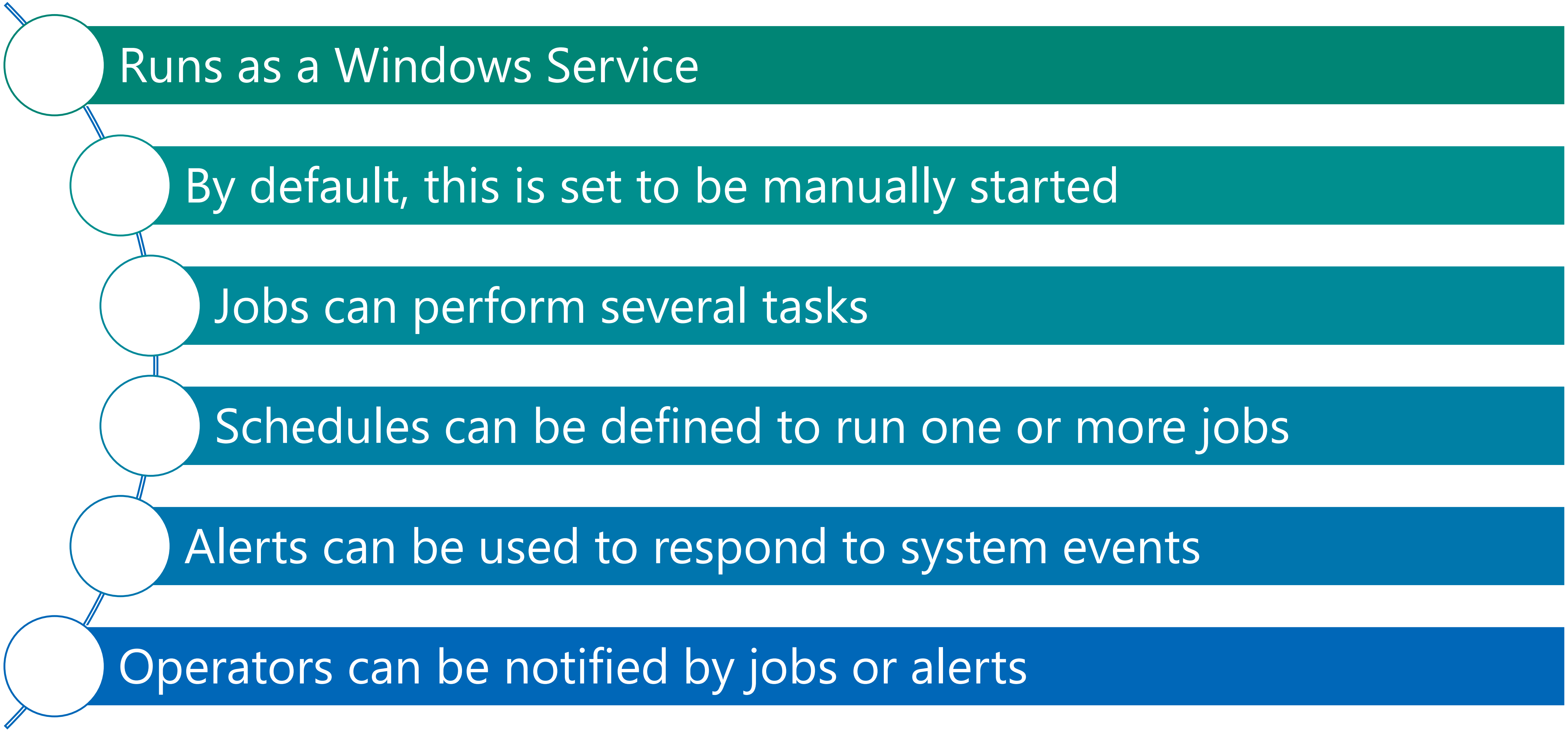
Lesson 1: SQL Server Agent and Jobs

Benefits of Automating SQL Server Management



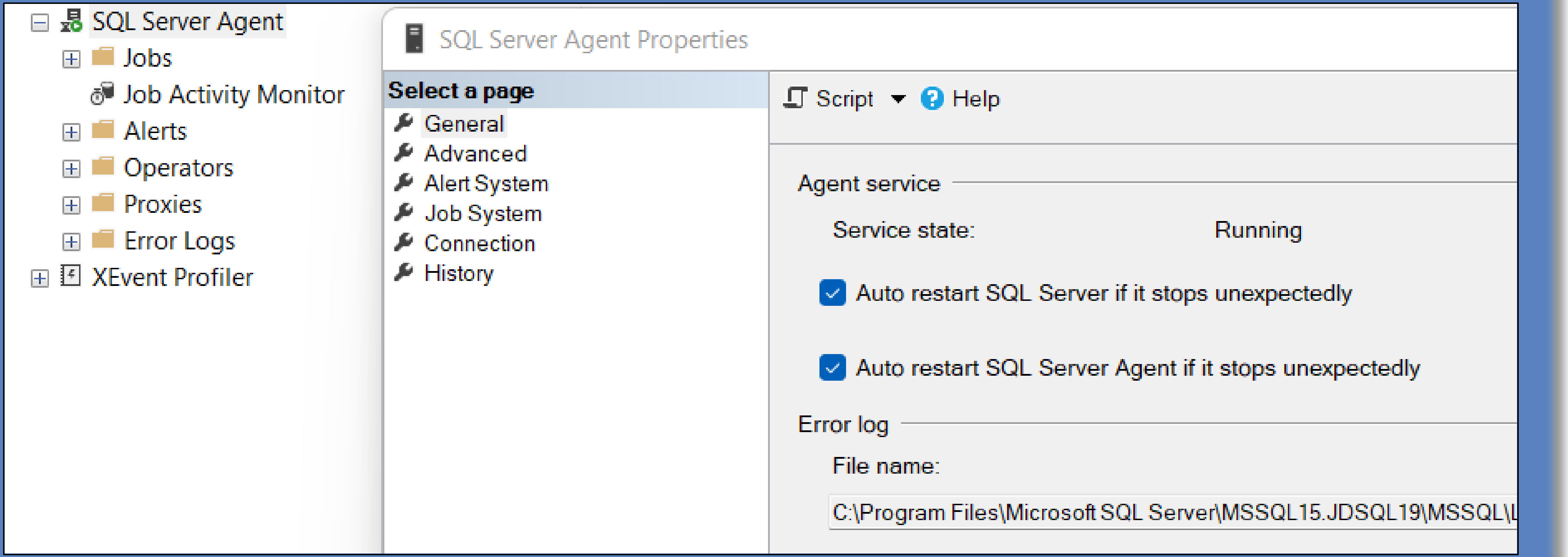
- Reduce administrative workload by automating and scheduling tasks
- Execute routine tasks reliably and consistently
- Proactively monitor performance
- Recognize and respond to potential problems

Overviews of the SQL Server Agent Service



- Runs as a Windows Service
- By default, this is set to be manually started
- Jobs can perform several tasks
- Schedules can be defined to run one or more jobs
- Alerts can be used to respond to system events
- Operators can be notified by jobs or alerts

SQL Server Agent Properties



Agent service

Service state: Running

☑ Auto restart SQL Server if it stops unexpectedly

☑ Auto restart SQL Server Agent if it stops unexpectedly

Error log

File name:

C:\Program Files\Microsoft SQL Server\MSSQL15.JDSQL19\MSSQL\L

Defining Jobs, Job Types, and Job Categories

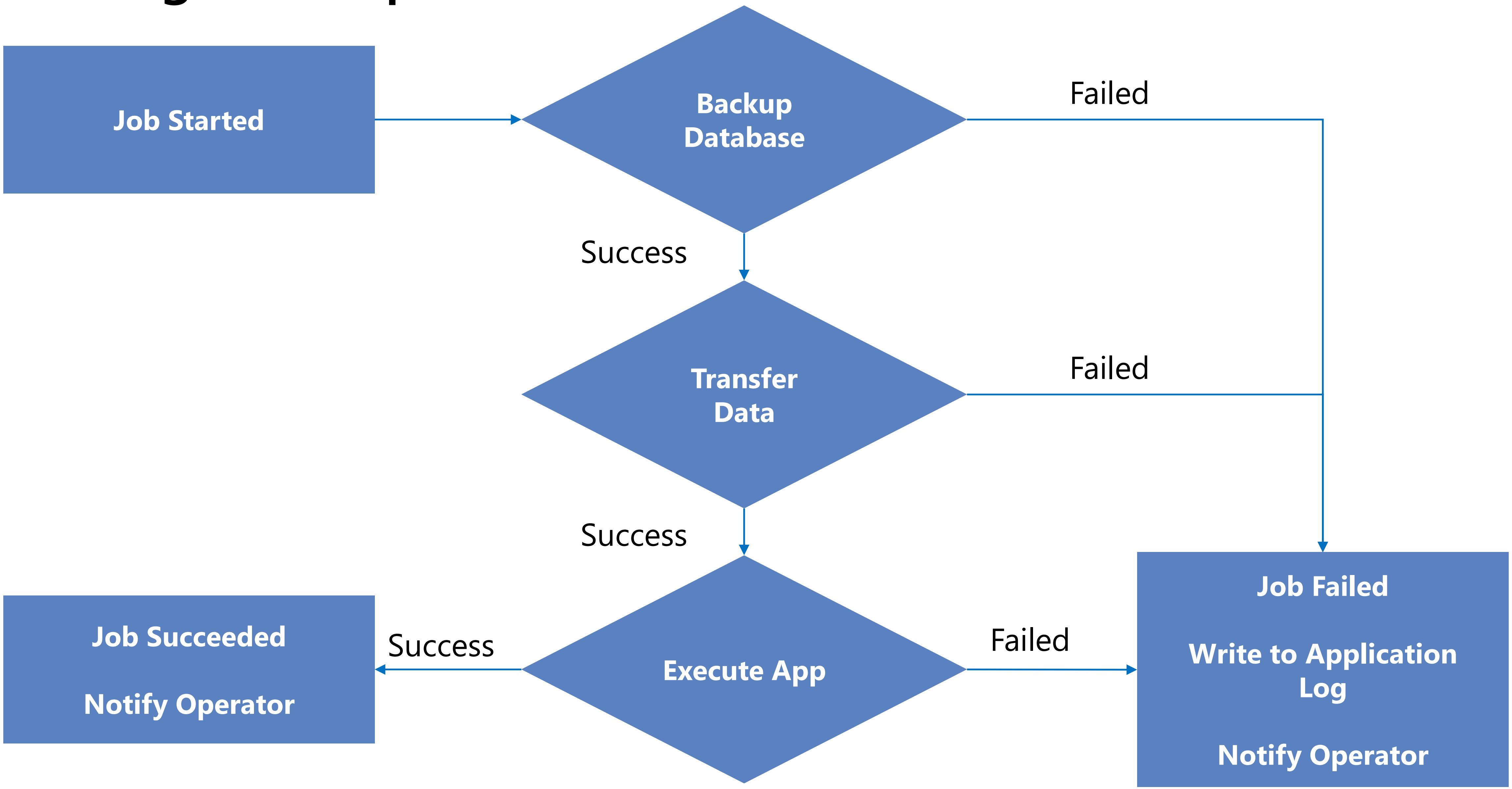
Jobs consist of a series of steps

Job step types include:

- Command-line script, batch of commands or application
- Transact-SQL statement
- PowerShell script
- SSIS and SSAS commands and queries

Jobs can be assigned to categories

Creating Job Steps



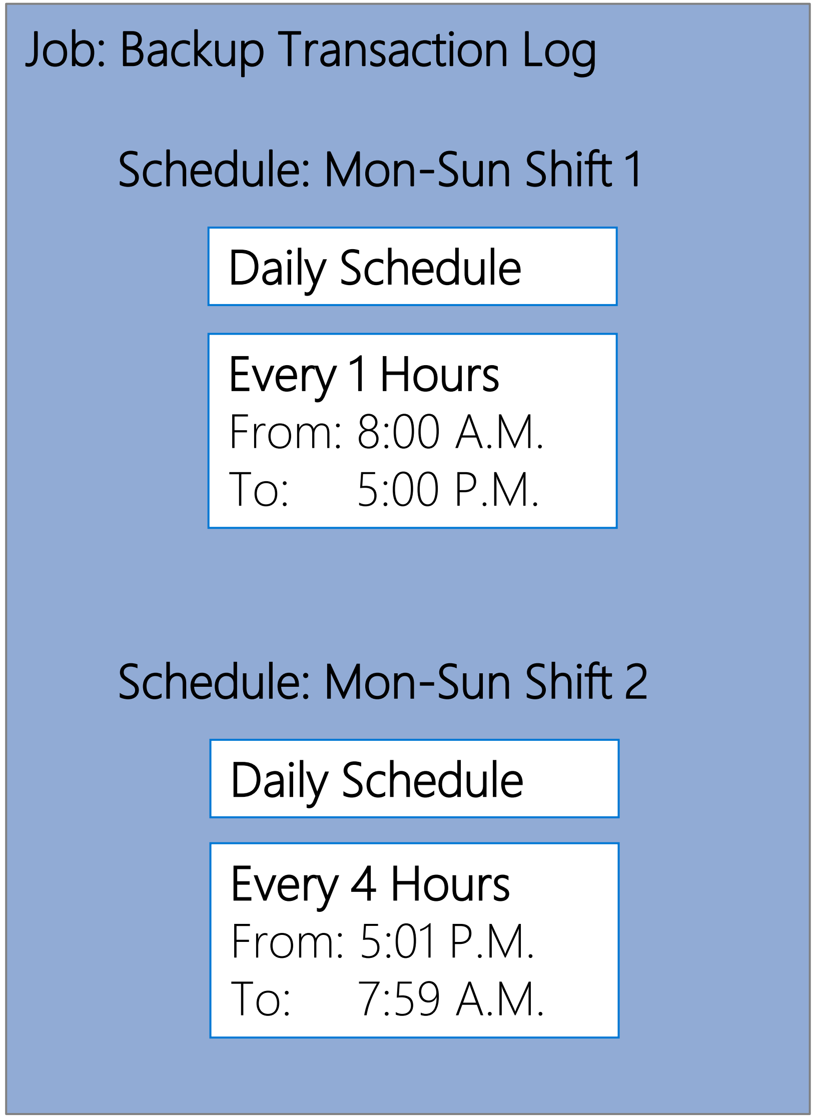
Scheduling Jobs for Execution

Recurrence:

- One time
- When SQL Server Agent starts
- Whenever the CPU is idle

One job can have multiple schedules

Multiple jobs can share one schedule



Job: Backup Transaction Log

Schedule: Mon-Sun Shift 1

Daily Schedule

Every 1 Hours
From: 8:00 A.M.
To: 5:00 P.M.

Schedule: Mon-Sun Shift 2

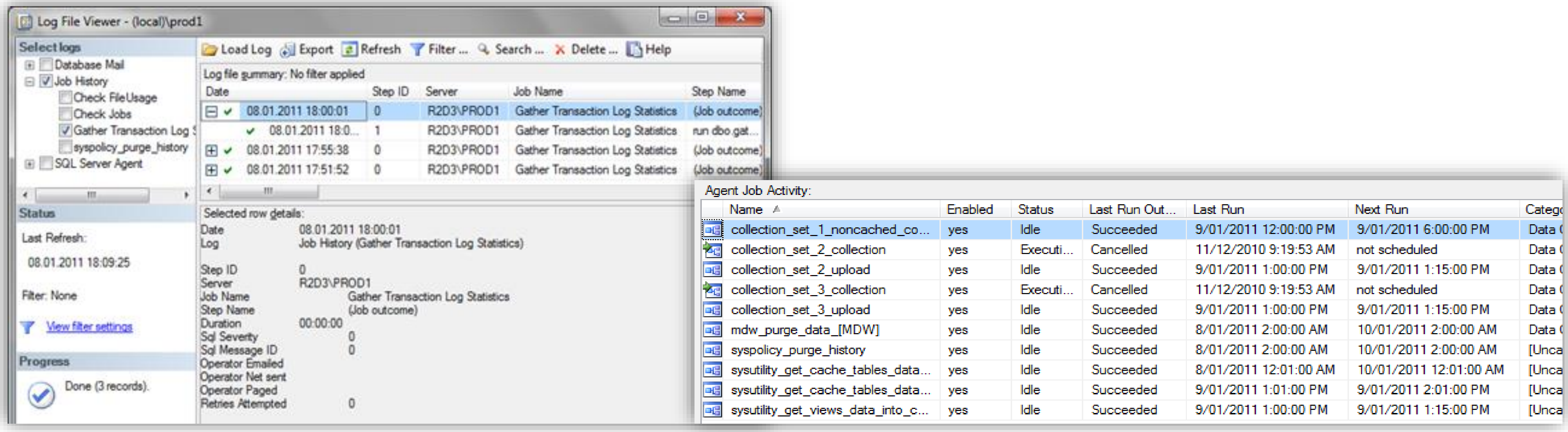
Daily Schedule

Every 4 Hours
From: 5:01 P.M.
To: 7:59 A.M.

Viewing Job History

SQL Server Agent writes job history to **msdb** and optionally to log files

Job Activity Monitor shows current job activity



Log file summary: No filter applied

Date	Step ID	Server	Job Name	Step Name
✓ 08.01.2011 18:00:01	0	R2D3\PROD1	Gather Transaction Log Statistics	(Job outcome)
✓ 08.01.2011 18:0...	1	R2D3\PROD1	Gather Transaction Log Statistics	run dbo.gat...
✓ 08.01.2011 17:55:38	0	R2D3\PROD1	Gather Transaction Log Statistics	(Job outcome)
✓ 08.01.2011 17:51:52	0	R2D3\PROD1	Gather Transaction Log Statistics	(Job outcome)

Last Refresh:

08.01.2011 18:09:25

Filter: None

Done (3 records).

Selected row details:

```
Date                08.01.2011 18:00:01
Log                 Job History (Gather Transaction Log Statistics)

Step ID             0
Server              R2D3\PROD1
Job Name                    Gather Transaction Log Statistics
Step Name                   (Job outcome)
Duration            00:00:00
Sql Severity                0
Sql Message ID              0
Operator Emailed
Operator Net sent
Operator Paged
Retries Attempted           0
```

Agent Job Activity:

Name	Enabled	Status	Last Run Out...	Last Run	Next Run	Categ
collection_set_1_noncached_co...	yes	Idle	Succeeded	9/01/2011 12:00:00 PM	9/01/2011 6:00:00 PM	Data (
collection_set_2_collection	yes	Executi...	Cancelled	11/12/2010 9:19:53 AM	not scheduled	Data (
collection_set_2_upload	yes	Idle	Succeeded	9/01/2011 1:00:00 PM	9/01/2011 1:15:00 PM	Data (
collection_set_3_collection	yes	Executi...	Cancelled	11/12/2010 9:19:53 AM	not scheduled	Data (
collection_set_3_upload	yes	Idle	Succeeded	9/01/2011 1:00:00 PM	9/01/2011 1:15:00 PM	Data (
mdw_purge_data_[MDW]	yes	Idle	Succeeded	8/01/2011 2:00:00 AM	10/01/2011 2:00:00 AM	Data (
syspolicy_purge_history	yes	Idle	Succeeded	8/01/2011 2:00:00 AM	10/01/2011 2:00:00 AM	[Unca
sysutility_get_cache_tables_data...	yes	Idle	Succeeded	8/01/2011 12:01:00 AM	10/01/2011 12:01:00 AM	[Unca
sysutility_get_cache_tables_data...	yes	Idle	Succeeded	9/01/2011 1:01:00 PM	9/01/2011 2:01:00 PM	[Unca
sysutility_get_views_data_into_c...	yes	Idle	Succeeded	9/01/2011 1:00:00 PM	9/01/2011 1:15:00 PM	[Unca

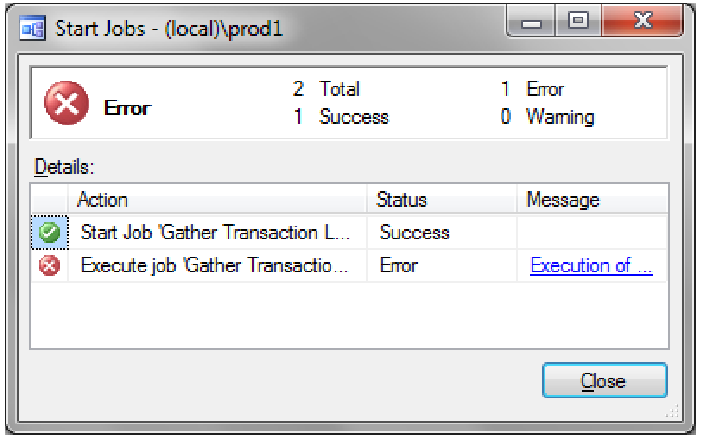
Querying SQL Server Agent-related System Tables

- Configuration and history in **msdb.dbo**
- Use history tables to automate collection of job history over several systems

```
SELECT j.name, jh.run_date, jh.run_time, jh.message
FROM msdb.dbo.sysjobhistory AS jh
INNER JOIN msdb.dbo.sysjobs AS j
ON jh.job_id = j.job_id
WHERE jh.step_id = 0;
GO
```

Troubleshooting Failed Jobs

- SQL Server Agent status:
  - Is the service account valid?
  - Is the **msdb** database online?
- Job history:
  - Job outcome identifies the last step to execute
  - Job step outcome identifies why the step failed
- Job execution:
  - Is the job enabled?
  - Is the job scheduled?
  - Is the schedule enabled?
- Access to dependencies:
  - Are all dependent objects available?



Start Jobs - (local)\prod1

Error

2	Total	1	Error
1	Success	0	Warning

Details:

	Action	Status	Message
✓	Start Job 'Gather Transaction L...	Success	
☒	Execute job 'Gather Transactio...	Error	Execution of ...

Lesson 2: Monitoring with Alerts and Notifications

Overview of SQL Server Alerts

An alert is a predefined response to an event

Alerts can be triggered by:

- Logged SQL Server events
- SQL Server performance conditions
- WMI events

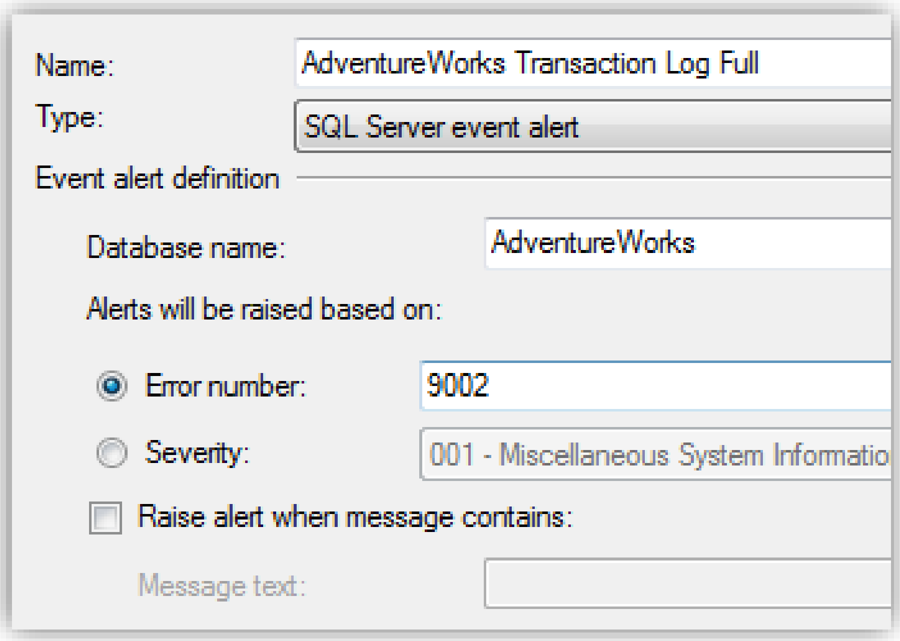
Alerts can:

- Notify an operator
- Start a job

Creating Alerts

- Create by using SQL Server Management Studio or **sp_add_alert**

- Define action to take



Name: AdventureWorks Transaction Log Full

Type: SQL Server event alert

Event alert definition

Database name: AdventureWorks

Alerts will be raised based on:

● Error number: 9002

○ Severity: 001 - Miscellaneous System Informatio

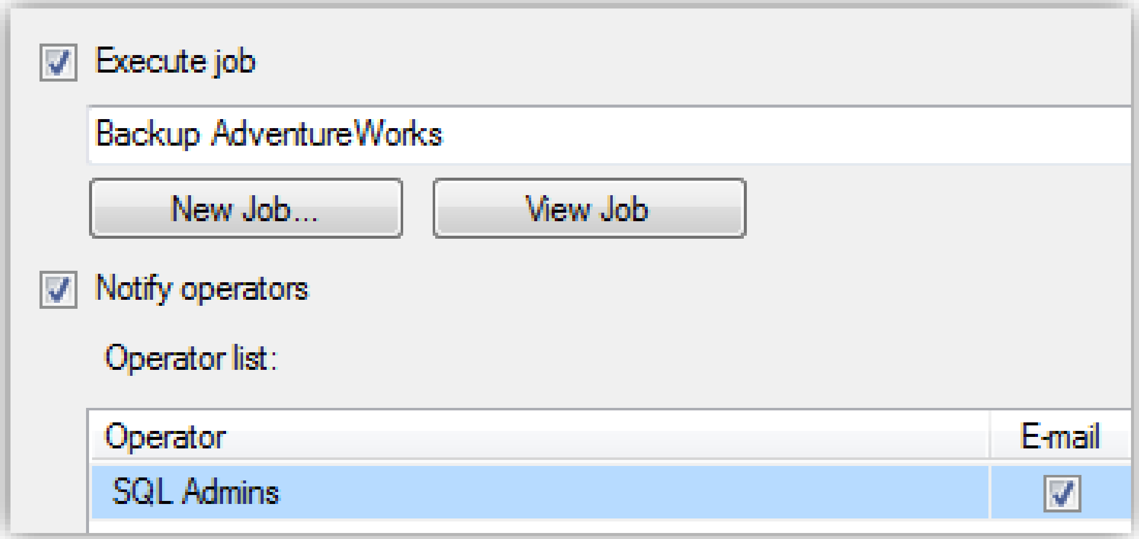
□ Raise alert when message contains:

Message text:

```
EXEC msdb.dbo.sp_add_alert
 @name=N'AdventureWorks Transaction Log Full',
 @message_id=9002, @delay_between_responses=0,
 @database_name=N'AdventureWorks';
GO
```

Configuring Alert Actions

- Actions:
  - Execute a job
  - Notify an operator



☑ Execute job

Backup AdventureWorks

☑ Notify operators

Operator list:

Operator	E-mail
SQL Admins	☑

- Create Notifications

```
EXEC msdb.dbo.sp_add_notification
 @alert_name
    = N'AdventureWorks Transaction Log Full',
 @operator_name=N'SQL Admins',
 @notification_method = 1;
GO
```

Microsoft

Module 5: Security - General

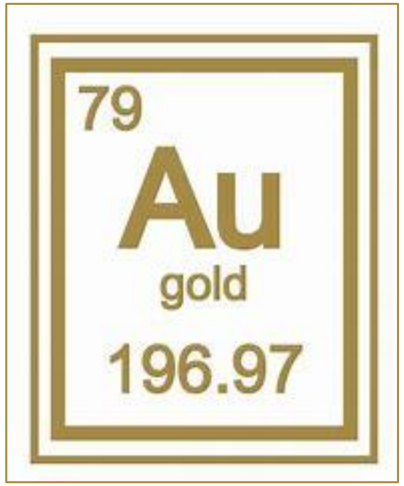
Lesson 1: Authentication and Authorization

The Three AU's of Security

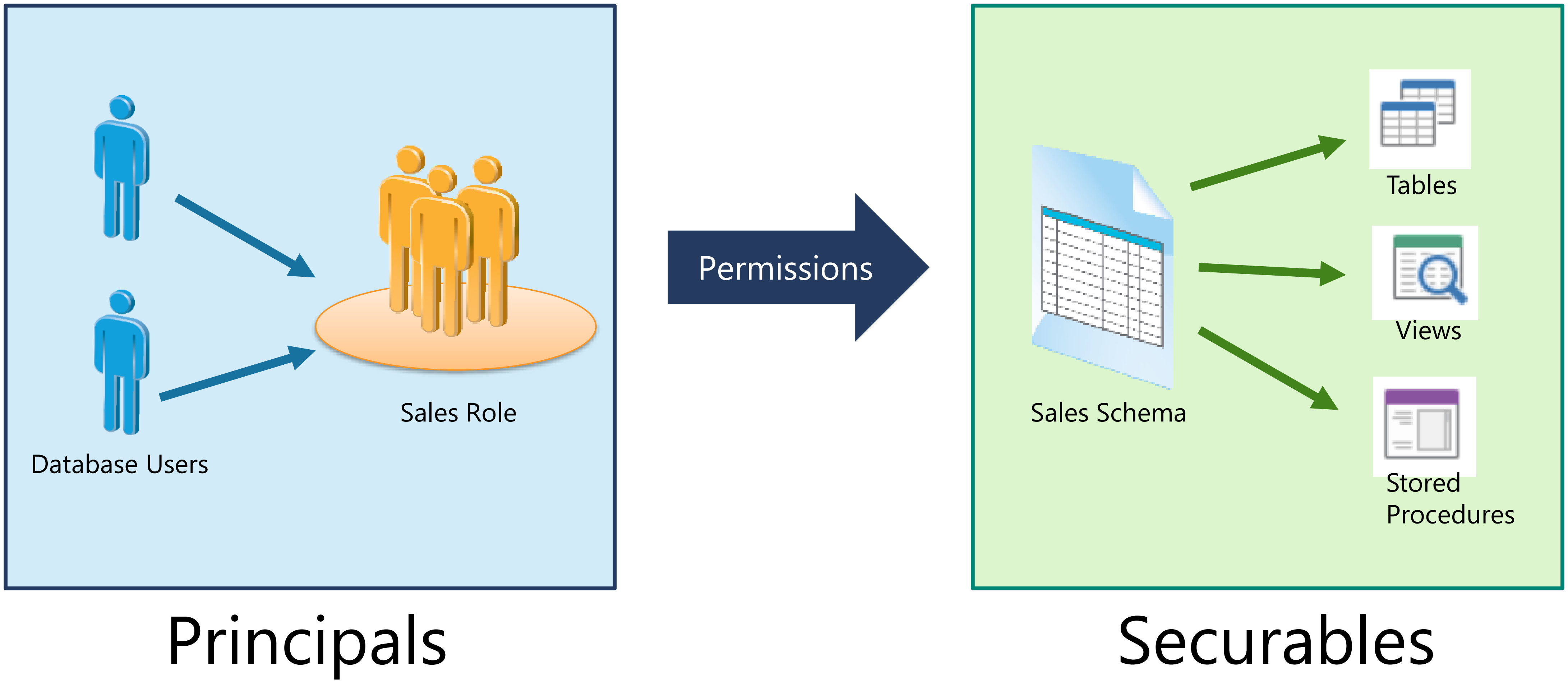
AUTHENTICATION – Verifies who you are

AUTHORIZATION – Assigns what you can do

AUDITING – Monitors what you did



Principals vs Securables



Database Users

Sales Role

Permissions

Sales Schema

Tables

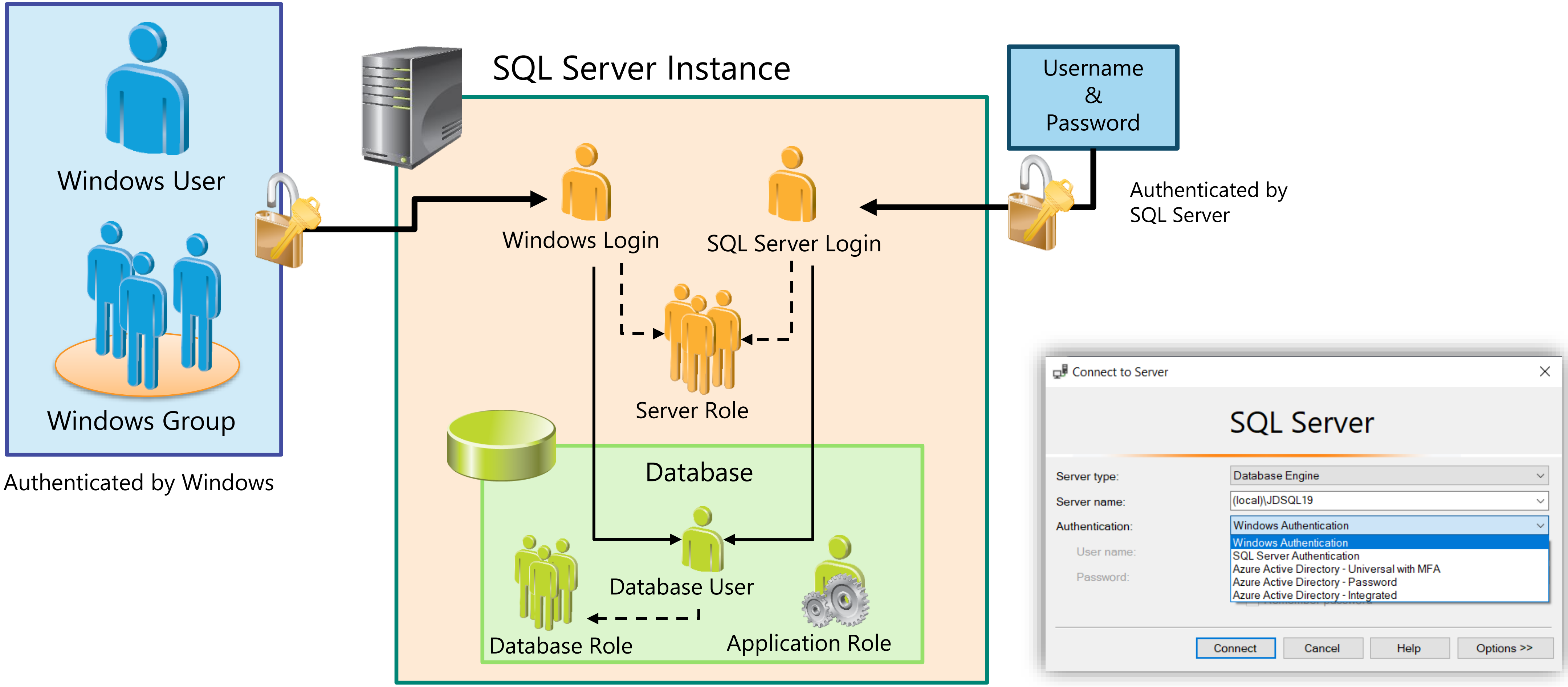
Views

Stored Procedures

Principals

Securables

Security Principals



Connect to Server

SQL Server

Server type: Database Engine

Server name: (local)\JDSQL19

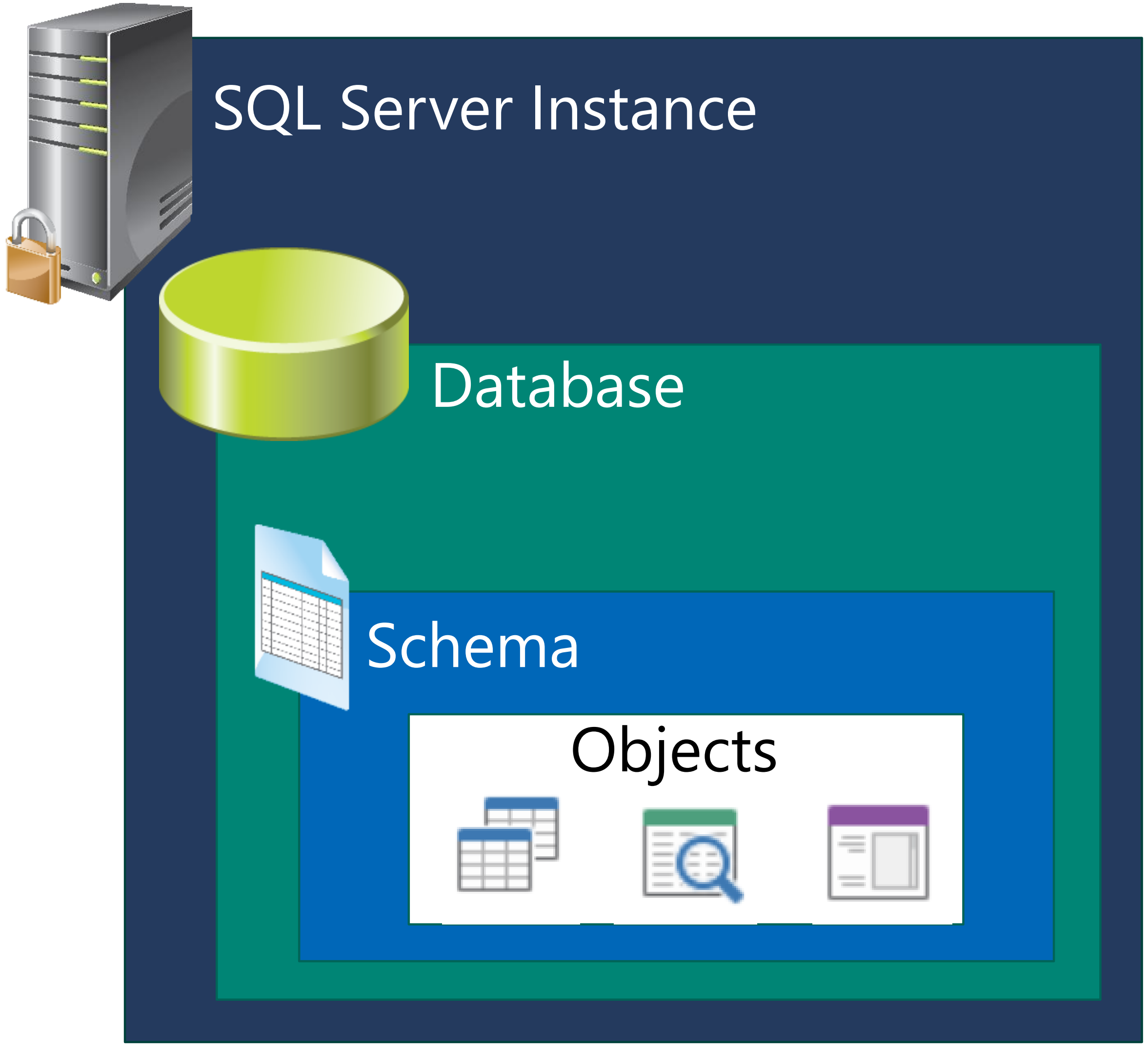
Authentication: Windows Authentication

- Windows Authentication
- SQL Server Authentication
- Azure Active Directory - Universal with MFA
- Azure Active Directory - Password
- Azure Active Directory - Integrated

User name:

Password:

Securables and the Four-Part Name of Objects



Server.Database.Schema.Object

Authentication

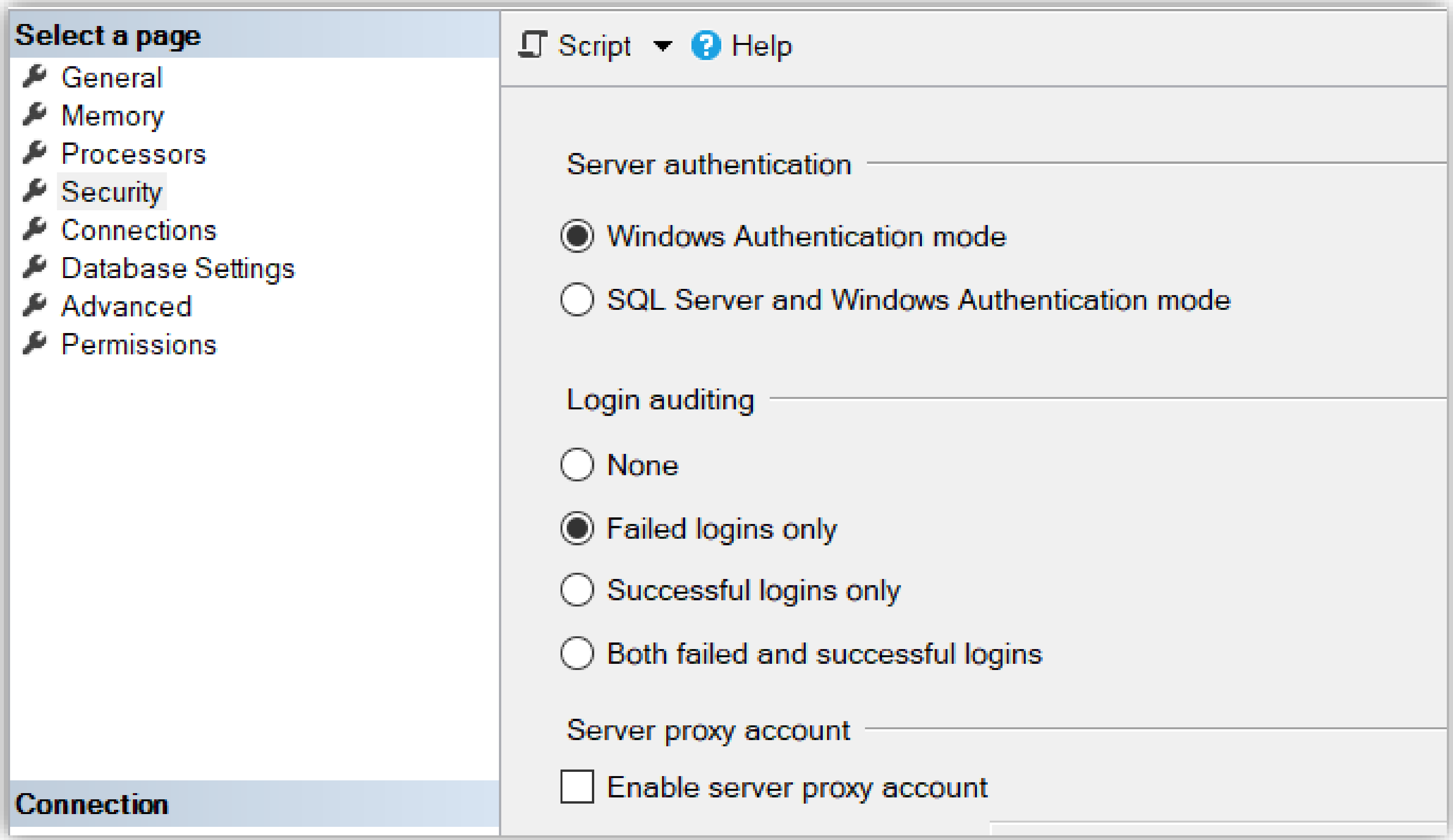
Windows Authentication

- SQL Server validates credentials using Active Directory and then verifies if it has permissions to connect.

SQL Authentication

- SQL Server validates the password against a hash stored in master and then verifies if it has permissions to connect.

Server Authentication



Server authentication

● Windows Authentication mode

○ SQL Server and Windows Authentication mode

Login auditing

○ None

● Failed logins only

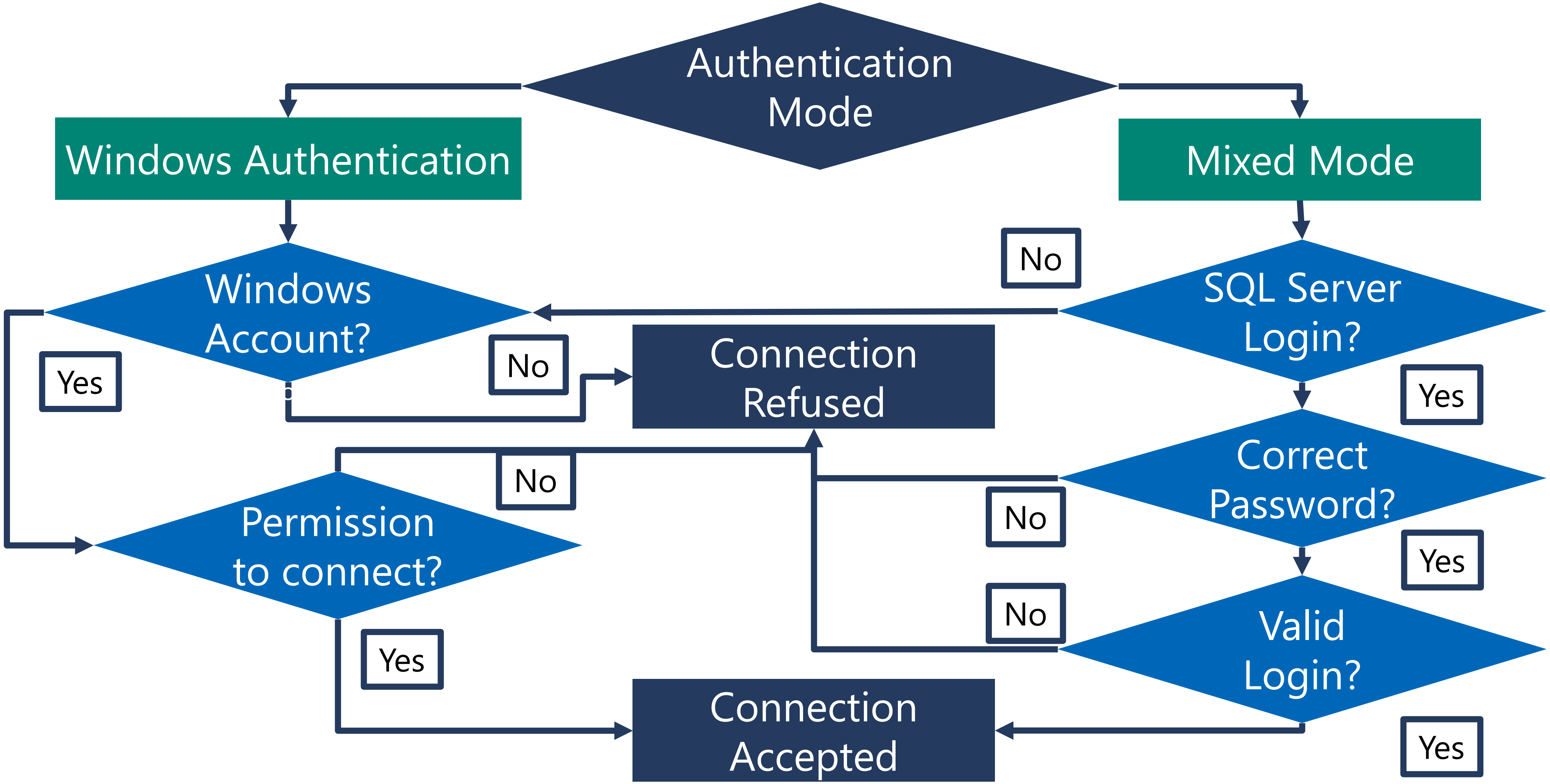
○ Successful logins only

○ Both failed and successful logins

Server proxy account

□ Enable server proxy account

Authentication



Advantages of SQL Server Authentication

- Allows applications that require SQL Server Authentication
- Allows SQL Server to support environments with mixed operating systems, where all users are not authenticated by a Windows domain
- Allows users to connect from unknown or untrusted domains
- Allows SQL Server to support web-based applications where users create their own identities
- Allows software developers to distribute their applications by using a complex permission hierarchy

Disadvantages of SQL Server Authentication

If there is an Active Directory in the organization, users cannot take advantage of their Windows credentials.

SQL Server Authentication cannot use Kerberos security protocol.

Windows offers additional password policies that are not available for SQL Server logins.

The encrypted SQL Server Authentication login password must be passed over the network at the time of the connection.

Authorization

- Process by which SQL server decides whether a given principal can access a resource
- Allows granting the specific permissions required rather than granting membership in a fixed role
- Provides structural information and metadata of a securable only to those principals who have permission to access the securable
- Allows creating custom permission sets
- Works on the principle of *least privilege*

Lesson 2: SQL Server Logins and Users

Creating Logins

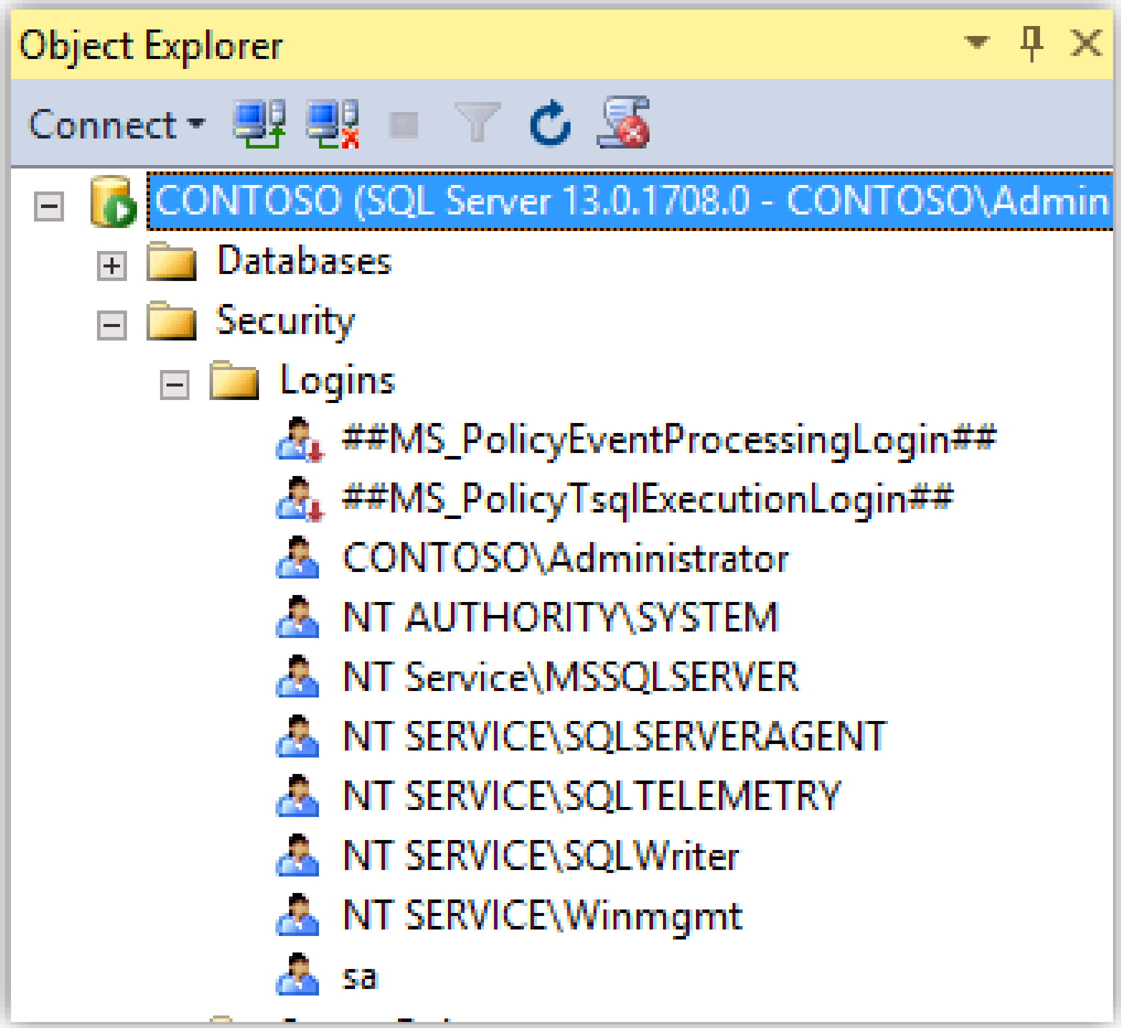
Allows connection to a SQL Server Instance

Two type of logins:

- SQL Login
- Windows Login

Can be created by:

- CREATE LOGIN statement in T-SQL
- SQL Server Management Studio



CONTOSO (SQL Server 13.0.1708.0 - CONTOSO\Admin
Databases
Security
Logins
##MS_PolicyEventProcessingLogin##
##MS_PolicyTsqlExecutionLogin##
CONTOSO\Administrator
NT AUTHORITY\SYSTEM
NT Service\MSSQLSERVER
NT SERVICE\SQLSERVERAGENT
NT SERVICE\SQLTELEMETRY
NT SERVICE\SQLWriter
NT SERVICE\Winmgmt
sa

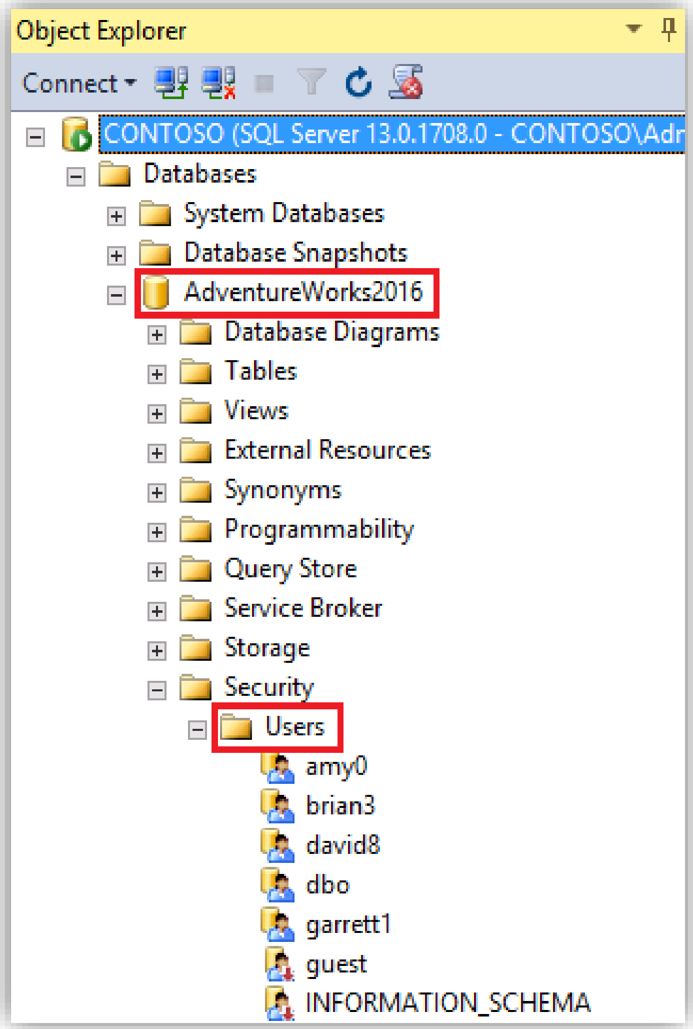
Creating Users

Allow access to a database

Specific to a single database

Type of users:

- Windows user
- SQL User with Password
- SQL User with Login
- SQL User without Password
- User mapped to a certificate
- User mapped to an asymmetric key



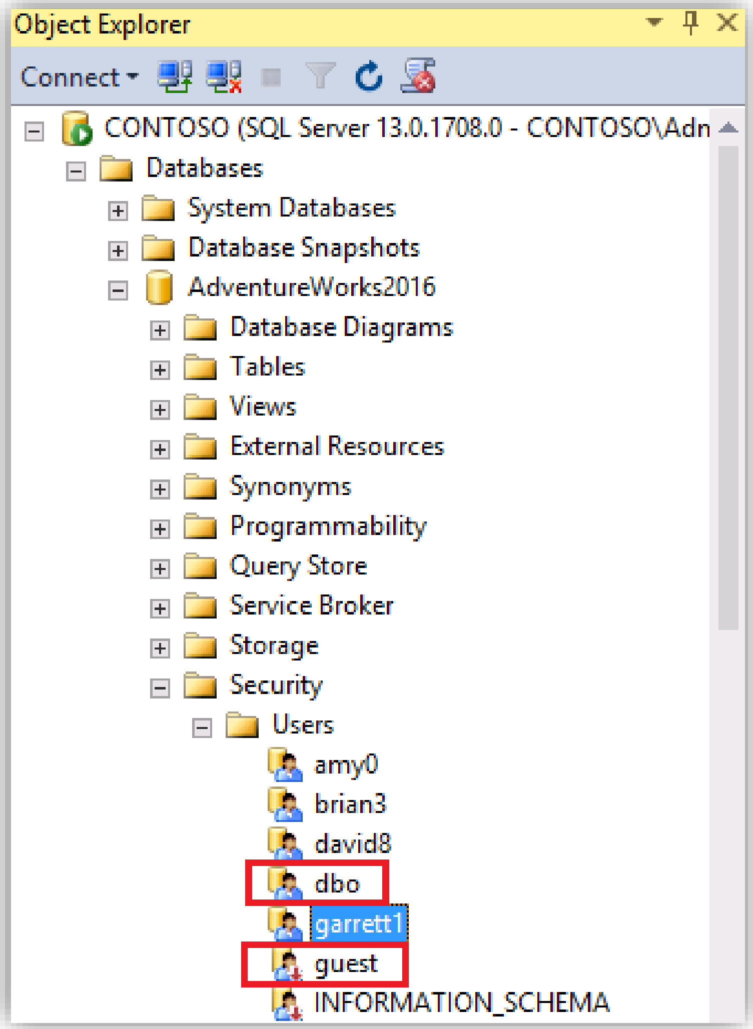
DBO and Guest User

DBO

- Performs all activities in the database
- Members of sysadmin role, SA login, and database owner are mapped to DBO.
- Cannot be deleted

Guest

- Allows logins without user accounts to access database
- Disabled by default in user databases
- Cannot be dropped but you can prevent it from accessing a database
- Must NOT be disabled in master and tempdb



Lesson 3: SQL Server Roles and Permissions

Roles

Server Roles

- Fixed server roles
- User-defined server roles

Database Roles

- Fixed database roles
- User-defined database roles

Application roles

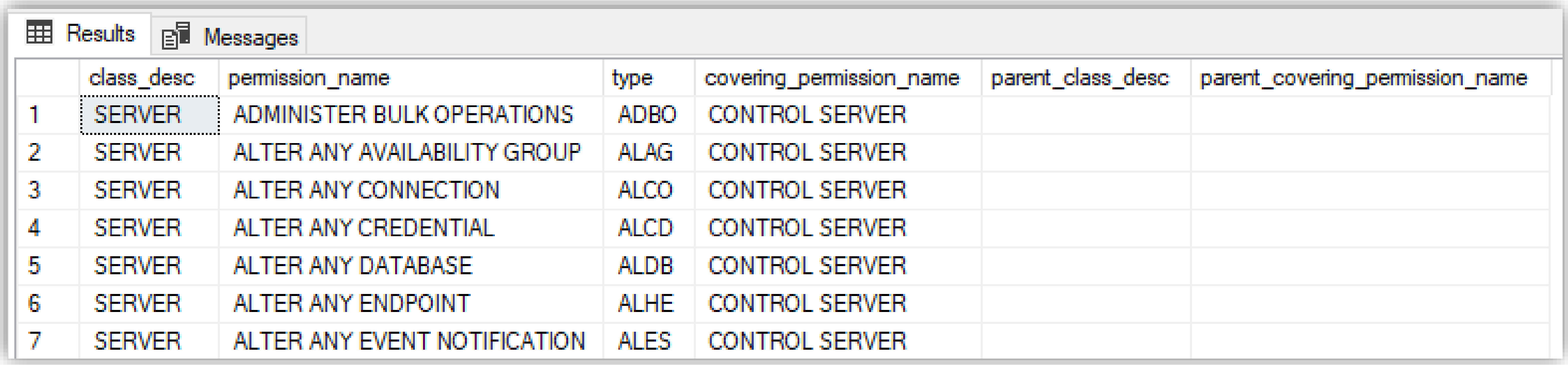
- Assign rights to applications instead of users

Fixed Server Level Roles and Permissions

Role	Description	Server-level Permission
sysadmin	Perform any activity	CONTROL SERVER (with GRANT option)
dbcreator	Create and alter databases	ALTER ANY DATABASE
diskadmin	Manage disk files	ALTER RESOURCES
serveradmin	Configure server-wide settings	ALTER ANY ENDPOINT, ALTER RESOURCES, ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN, VIEW SERVER STATE
securityadmin	Manage and audit server logins	ALTER ANY LOGIN
processadmin	Manage SQL Server processes	ALTER ANY CONNECTION ALTER SERVER STATE
bulkadmin	Run the BULK INSERT statement	ADMINISTER BULK OPERATIONS
setupadmin	Configure replication and linked servers	ALTER ANY LINKED SERVER

Listing Server Level Permissions

```
SELECT * FROM sys.fn_builtin_permissions('SERVER’)
ORDER BY permission_name;
```



	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	SERVER	ADMINISTER BULK OPERATIONS	ADBO	CONTROL SERVER		
2	SERVER	ALTER ANY AVAILABILITY GROUP	ALAG	CONTROL SERVER		
3	SERVER	ALTER ANY CONNECTION	ALCO	CONTROL SERVER		
4	SERVER	ALTER ANY CREDENTIAL	ALCD	CONTROL SERVER		
5	SERVER	ALTER ANY DATABASE	ALDB	CONTROL SERVER		
6	SERVER	ALTER ANY ENDPOINT	ALHE	CONTROL SERVER		
7	SERVER	ALTER ANY EVENT NOTIFICATION	ALES	CONTROL SERVER		

Public Role

Public is a special role that is at the server and database level.

Every SQL Server login and user belongs to the Public role

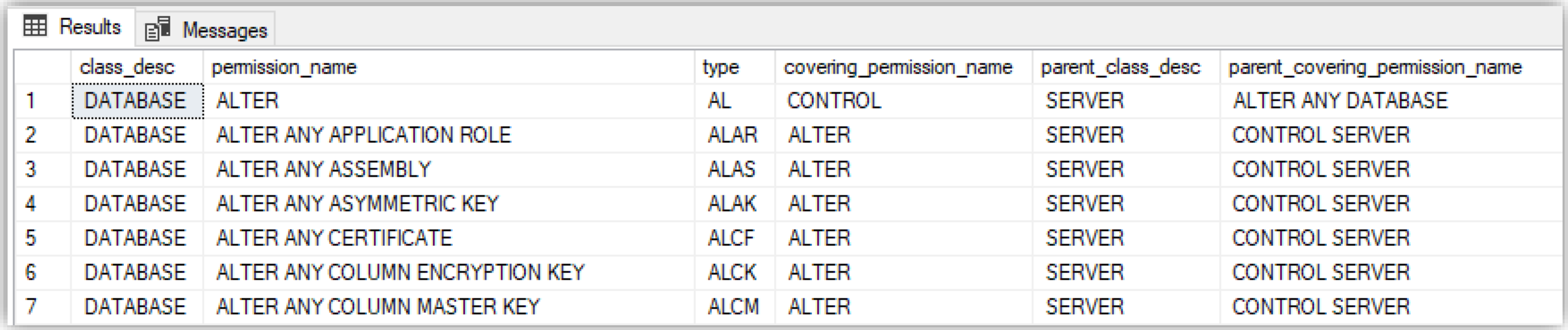
Care must be taken when granting permissions to Public server role especially when granting server-level **permissions.**

Fixed Database Level Roles and Permissions

Role	Description
db_owner	Perform any configuration and maintenance activities on the DB and can drop it
db_securityadmin	Modify role membership and manage permissions
db_accessadmin	Add or remove access to the DB for logins
db_backupoperator	Back up the DB
db_ddladmin	Run any DDL command in the DB
db_datawriter	Add, delete, or change data in all user tables
db_datareader	Read all data from all user tables
db_denydatawriter	Cannot add, delete, or change data in user tables
db_denydatareader	Cannot read any data in user tables

Listing Database level permissions

SELECT * FROM sys.fn_builtin_permissions(‘Database’)
ORDER BY permission_name;



	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	DATABASE	ALTER	AL	CONTROL	SERVER	ALTER ANY DATABASE
2	DATABASE	ALTER ANY APPLICATION ROLE	ALAR	ALTER	SERVER	CONTROL SERVER
3	DATABASE	ALTER ANY ASSEMBLY	ALAS	ALTER	SERVER	CONTROL SERVER
4	DATABASE	ALTER ANY ASYMMETRIC KEY	ALAK	ALTER	SERVER	CONTROL SERVER
5	DATABASE	ALTER ANY CERTIFICATE	ALCF	ALTER	SERVER	CONTROL SERVER
6	DATABASE	ALTER ANY COLUMN ENCRYPTION KEY	ALCK	ALTER	SERVER	CONTROL SERVER
7	DATABASE	ALTER ANY COLUMN MASTER KEY	ALCM	ALTER	SERVER	CONTROL SERVER

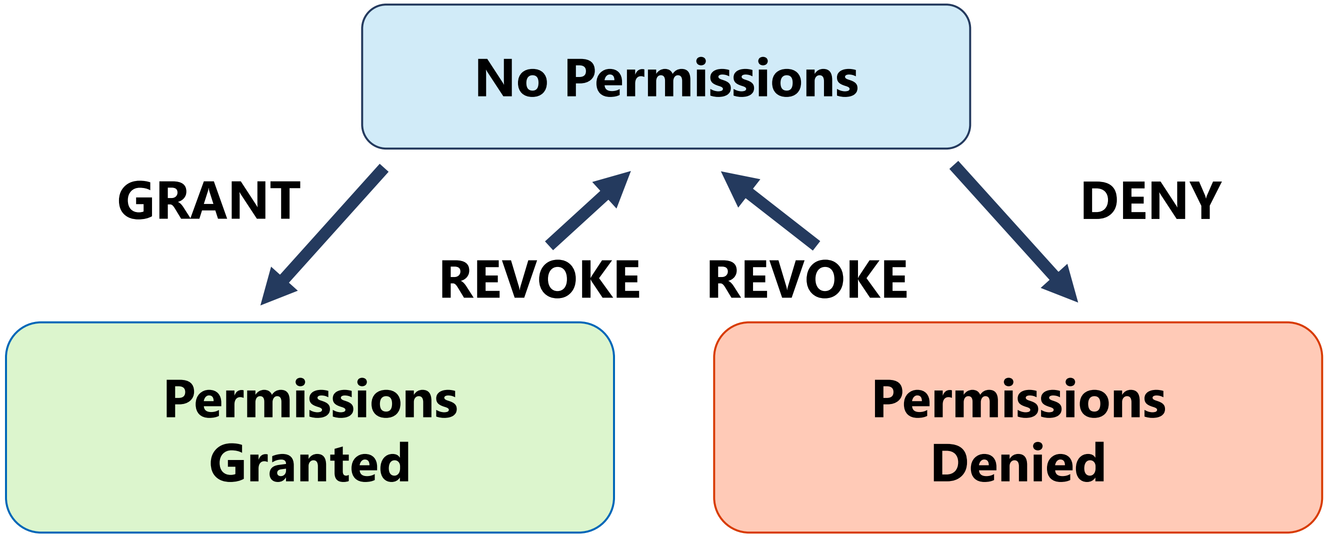
Assigning Permissions

GRANT is used to assign a permission

DENY is used to explicitly deny a permission

- Used where permissions inherited through group or role membership
- Should only be used in exceptional circumstances

REVOKE removes either a GRANT or a DENY



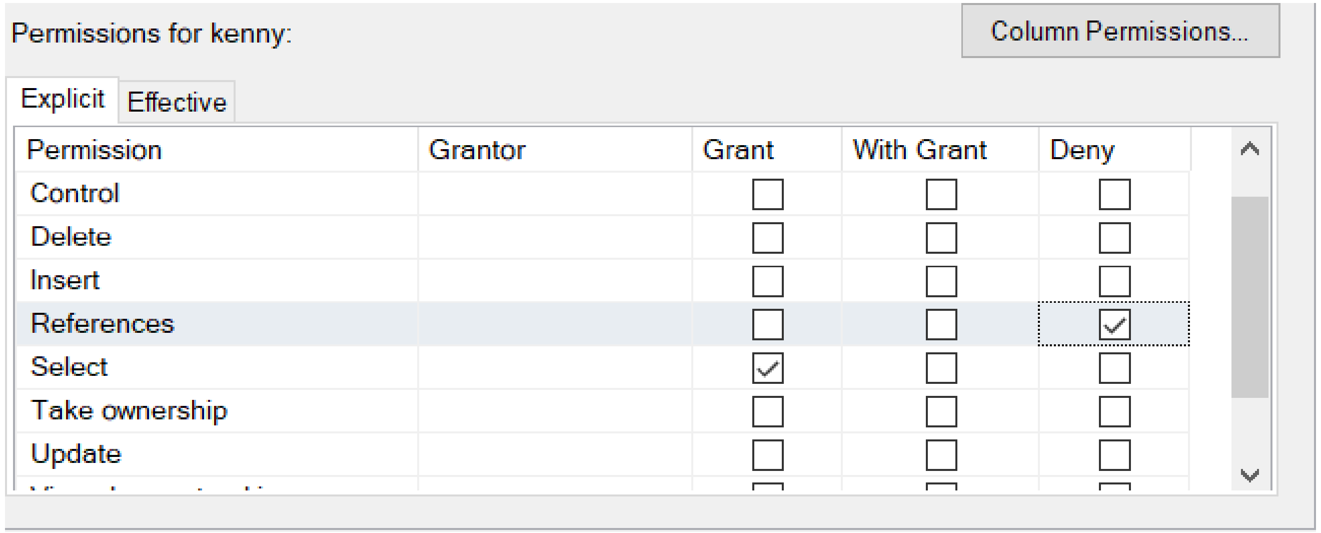
Assigning Permissions to Tables and Views

Grant with Grant allows the user to assign that permission.

Tables and Views can be assigned the same permissions.

Permissions for SELECT, UPDATE, and REFERENCES can also be set at the column level.

Select the Effective Tab to see what permissions have been granted.



Permissions for kenny:

Permission	Grantor	Grant	With Grant	Deny
Control		☐	☐	☐
Delete		☐	☐	☐
Insert		☐	☐	☐
References		☐	☐	☑
Select		☑	☐	☐
Take ownership		☐	☐	☐
Update		☐	☐	☐

Security with Schemas

FQN has the form: ***server.database.schema.object***

In a database, all objects are created within a schema (dbo is default).

Allow their owners full control over objects within the schema

Permissions can be granted at the schema level.

Can contain objects owned by multiple database users

Can be owned by any database principal

Creating a Schema

```sql
CREATE SCHEMA Sprockets AUTHORIZATION Annik
CREATE TABLE NineProngs (source int, cost int)

GRANT SELECT ON SCHEMA::Sprockets TO Mandar
DENY SELECT ON SCHEMA::Sprockets TO Prasanna;
GO

ALTER SCHEMA Sprockets TRANSFER dbo.FourSporks
GO
```

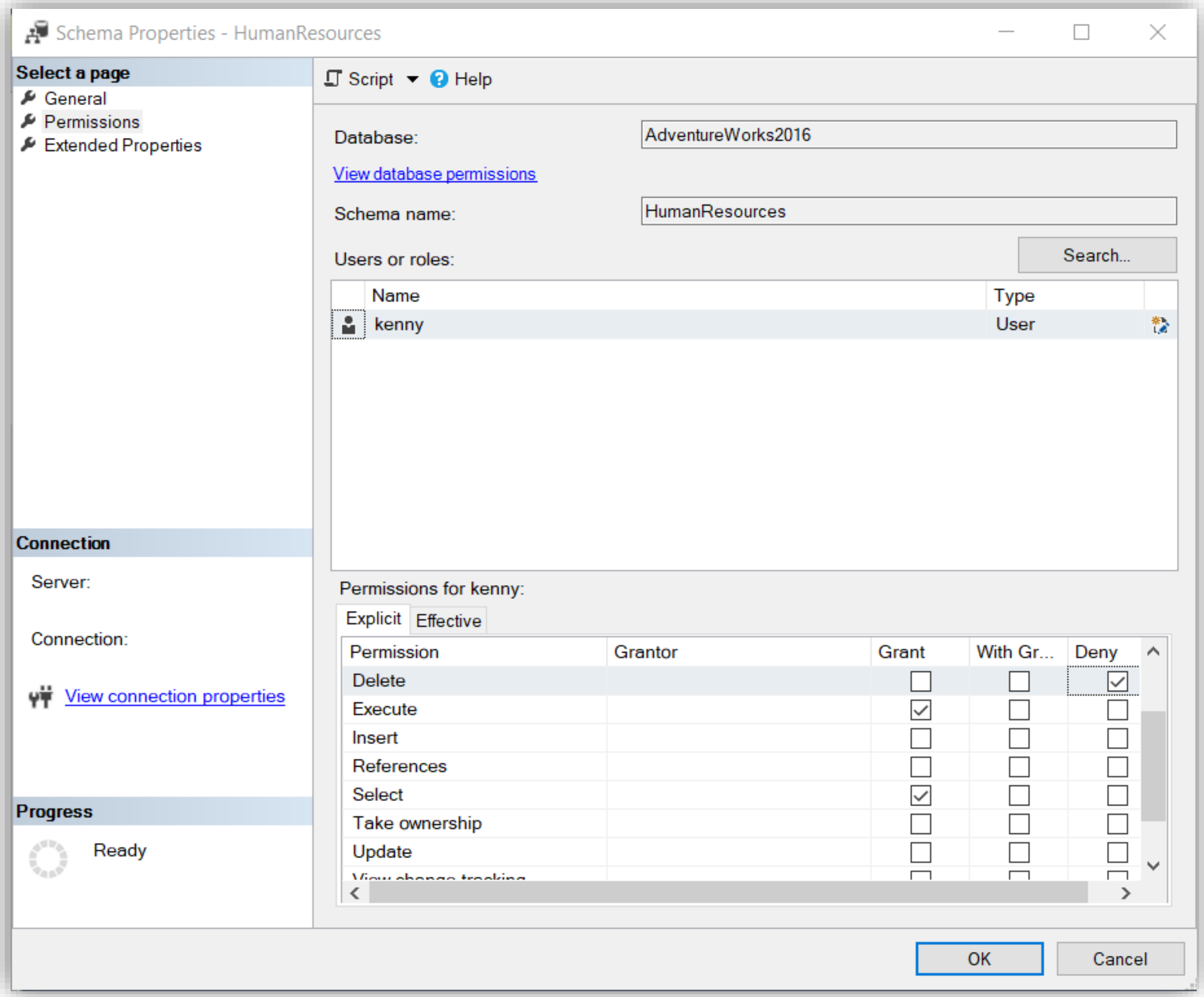
Assign Permissions to a Schema

Permissions assigned at the schema level affect all objects belonging to that schema.

Tables and Views can be assigned the same permissions.

The Execute permission will be applied to all Stored Procedures in the schema.

Select the Effective Tab to see what permissions have been granted.



Database: AdventureWorks2016

Schema name: HumanResources

Users or roles:

Name	Type
kenny	User

Permissions for kenny:

Permission	Grantor	Grant	With Gr...	Deny
Delete		☐	☐	☑
Execute		☑	☐	☐
Insert		☐	☐	☐
References		☐	☐	☐
Select		☑	☐	☐
Take ownership		☐	☐	☐
Update		☐	☐	☐
View change tracking		☐	☐	☐

Creating Synonyms

Creating a synonym will allow you to reference an object by an alternate name.

Useful for legacy applications that referenced objects with a dbo owner.

```
CREATE SYNONYM dbo.Employee FOR HumanResources.Employee
```

```
SELECT * FROM HumanResources.Employee
OR
SELECT * FROM dbo.Employee
```

Lesson 4: Row-Level Security

What is Row-Level Security?

A security feature available in SQL Server 2016 or later that will restrict access to specific rows in a table based on values in a column.

Row Level Security Scenarios

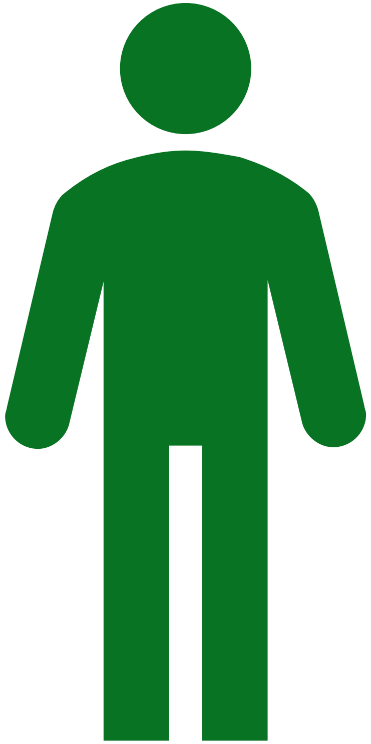
A hospital can restrict doctors and nurses to only view data about their specific patients.

A bank can restrict access to data based on the location of their branch offices.

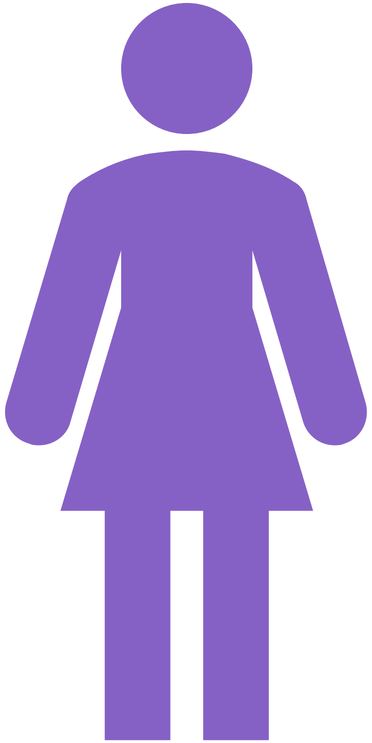
A bicycle company can restrict sales leads to only specific salespeople.

Salespeople for Adventure Works Bicycle Company

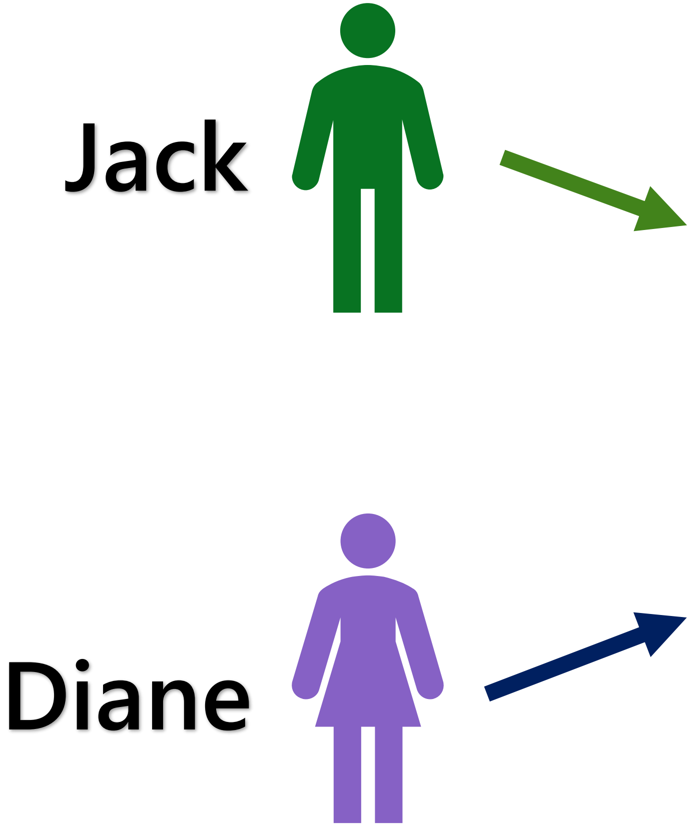
Jack



Diane

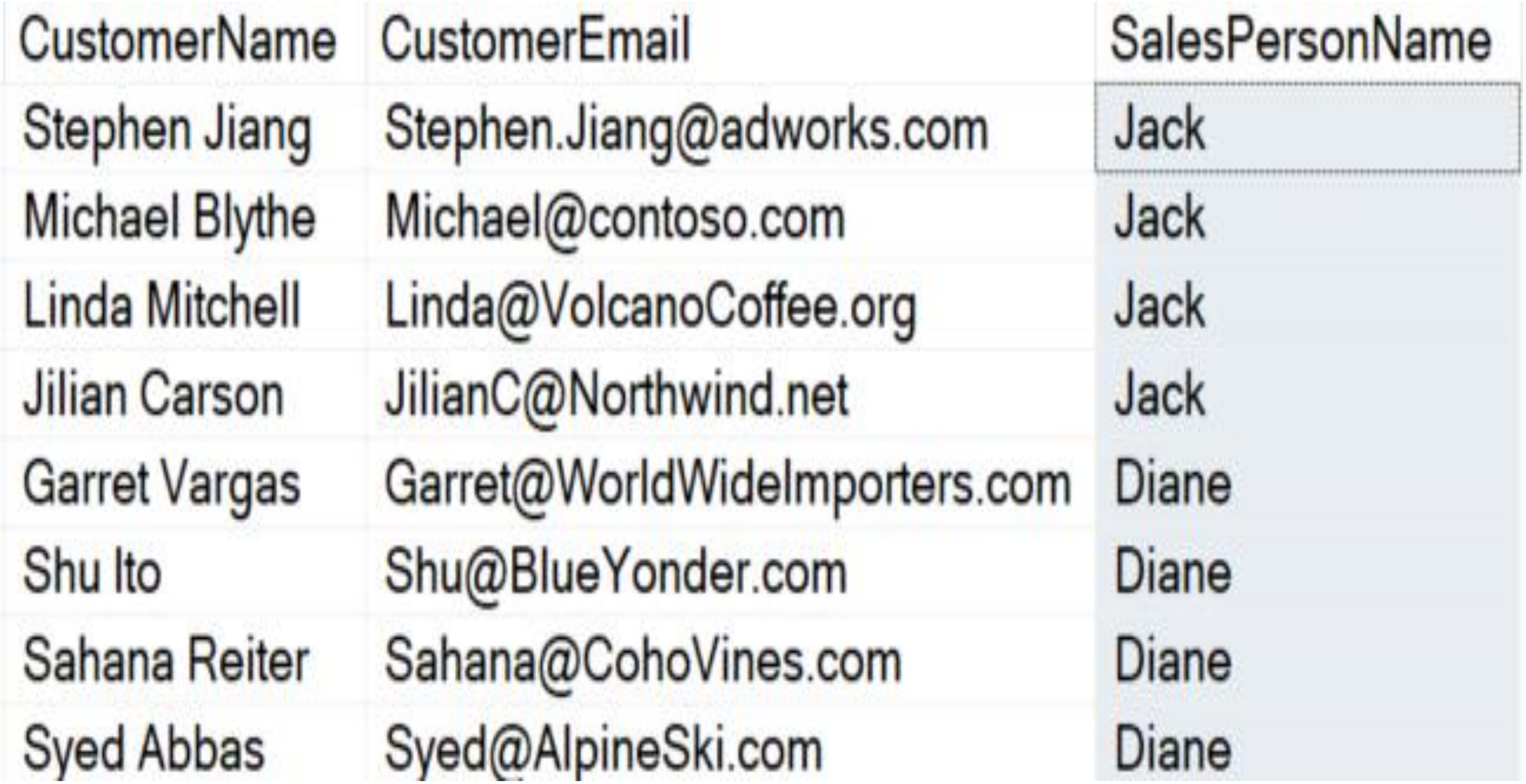


Salespeople for Adventure Works Bicycle Company



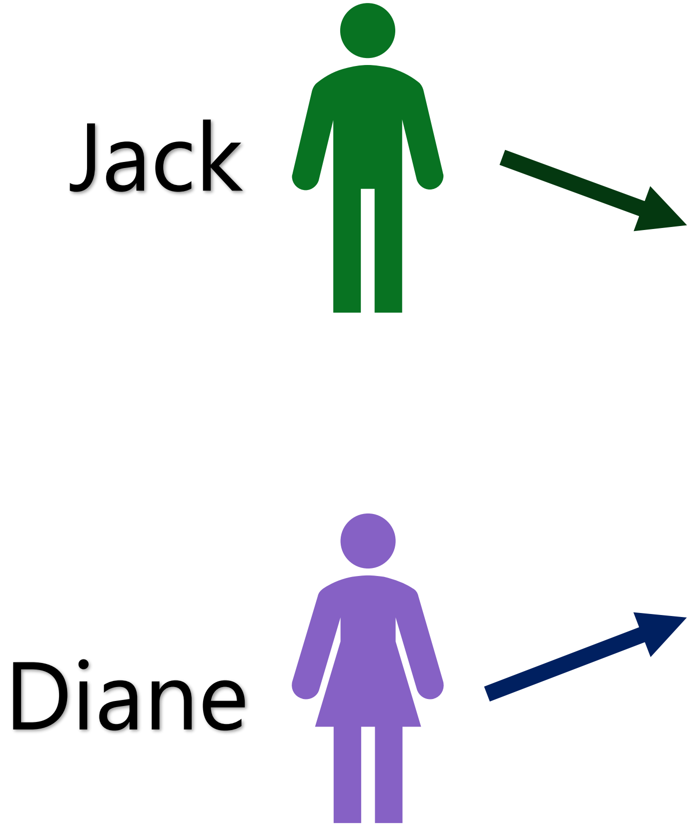
Jack

Diane



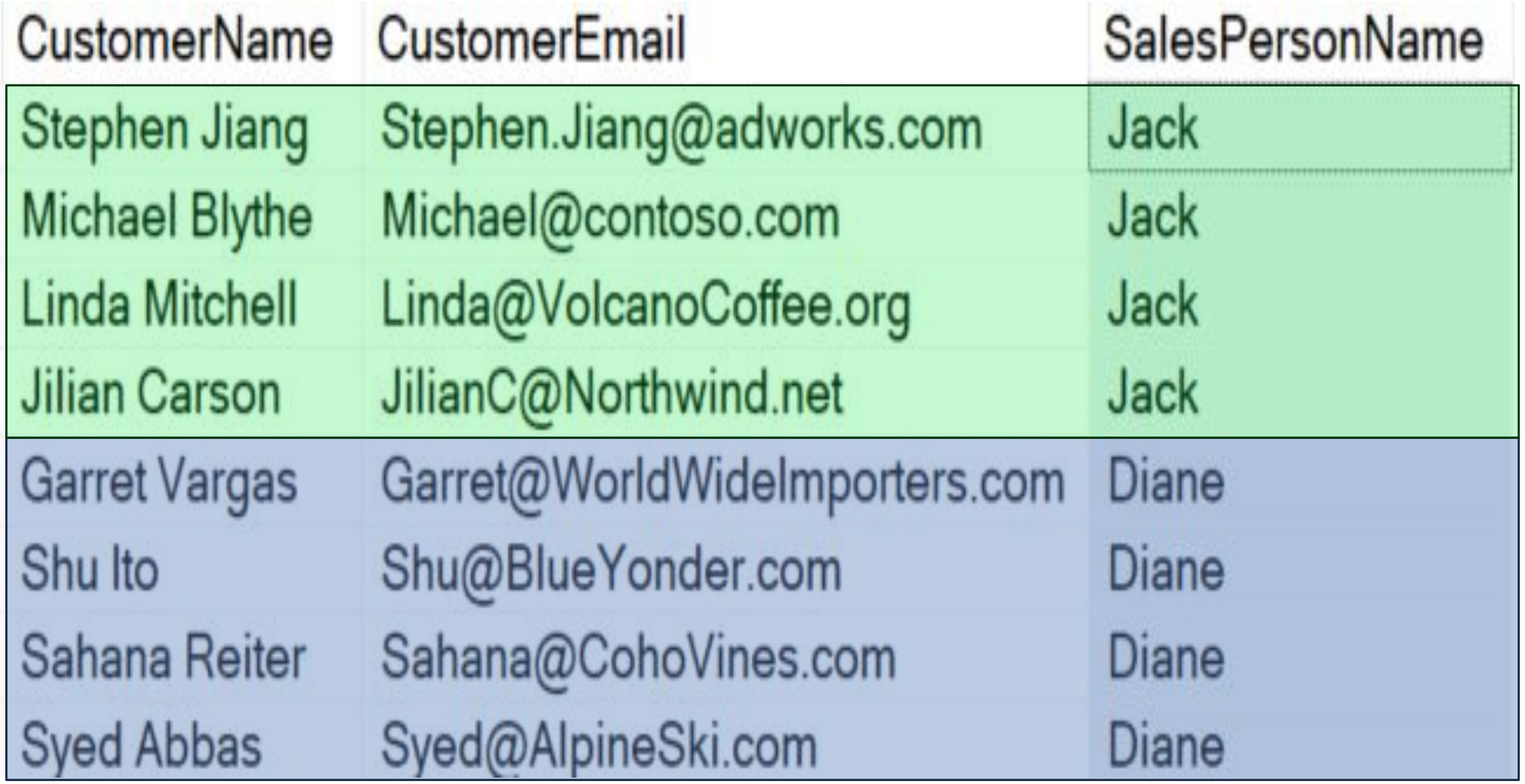
CustomerName	CustomerEmail	SalesPersonName
Stephen Jiang	Stephen.Jiang@adworks.com	Jack
Michael Blythe	Michael@contoso.com	Jack
Linda Mitchell	Linda@VolcanoCoffee.org	Jack
Jilian Carson	JilianC@Northwind.net	Jack
Garret Vargas	Garret@WorldWideImporters.com	Diane
Shu Ito	Shu@BlueYonder.com	Diane
Sahana Reiter	Sahana@CohoVines.com	Diane
Syed Abbas	Syed@AlpineSki.com	Diane

Salespeople for Adventure Works Bicycle Company



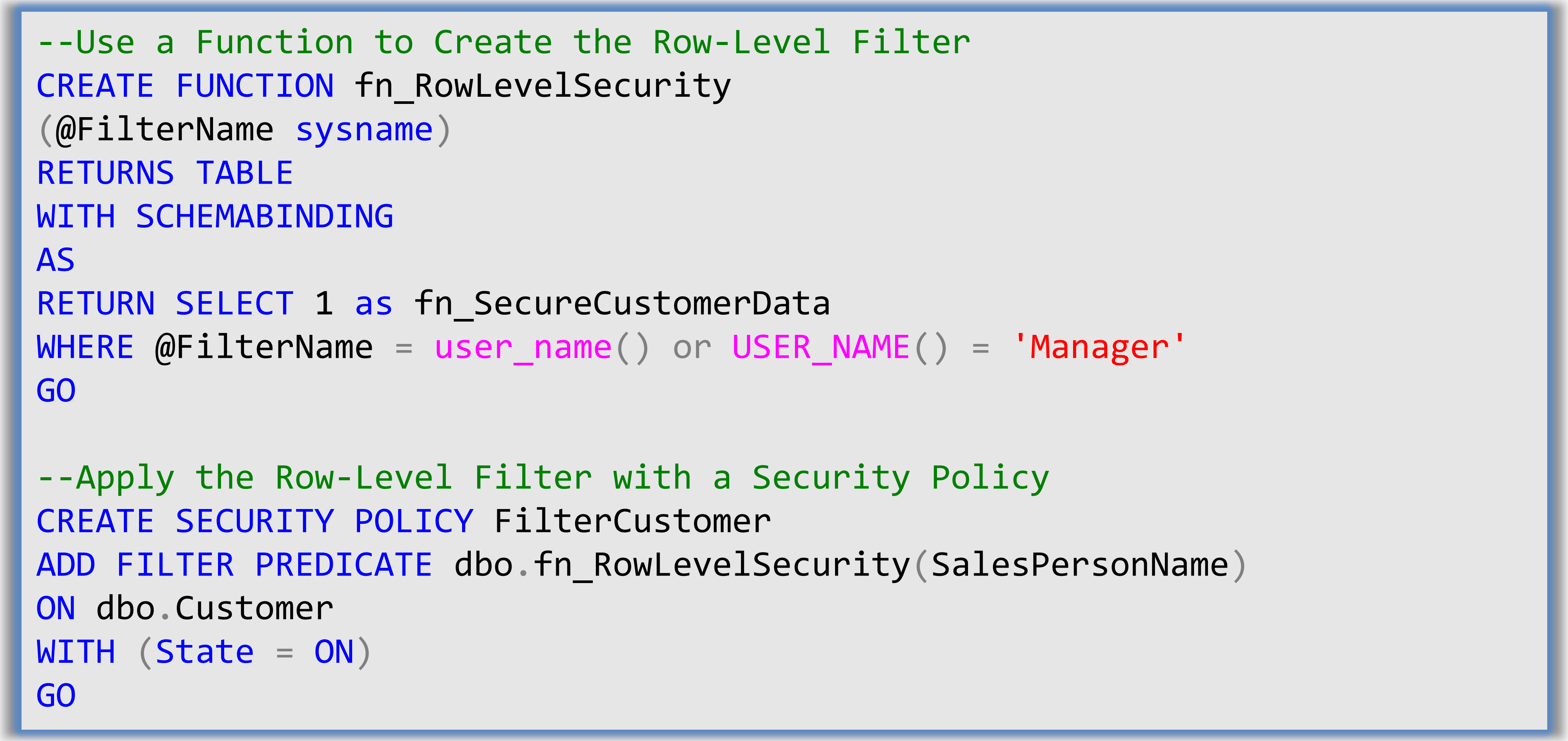
Jack

Diane



CustomerName	CustomerEmail	SalesPersonName
Stephen Jiang	Stephen.Jiang@adworks.com	Jack
Michael Blythe	Michael@contoso.com	Jack
Linda Mitchell	Linda@VolcanoCoffee.org	Jack
Jilian Carson	JilianC@Northwind.net	Jack
Garret Vargas	Garret@WorldWideImporters.com	Diane
Shu Ito	Shu@BlueYonder.com	Diane
Sahana Reiter	Sahana@CohoVines.com	Diane
Syed Abbas	Syed@AlpineSki.com	Diane

Create Function and Security Policy



```
--Use a Function to Create the Row-Level Filter
CREATE FUNCTION fn_RowLevelSecurity
(@FilterName sysname)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN SELECT 1 as fn_SecureCustomerData
WHERE @FilterName = user_name() or USER_NAME() = 'Manager'
GO

--Apply the Row-Level Filter with a Security Policy
CREATE SECURITY POLICY FilterCustomer
ADD FILTER PREDICATE dbo.fn_RowLevelSecurity(SalesPersonName)
ON dbo.Customer
WITH (State = ON)
GO
```

Lesson 5: Dynamic Data Masking

Dynamic Data Masking scenarios

Developers can troubleshoot production data without viewing sensitive information.

Customer Service representatives can view parts of sensitive data like credit card information.

Reports can be distributed with sensitive data obfuscated at the data layer.

Dynamic Data Masking Functions

DEFAULT

RANDOM

CUSTOM

EMAIL

Default Data Masking Function

DEFAULT

Masking is according to the data types of the specified column.

- For string data types, uses XXXX
- For numeric data types use a zero value.
- For date and time data types use 01.01.1900 00:00:00.0000000

Random Data Masking Function

RANDOM

The Random Data Masking function is only applied on numeric data types. It displays a random value for the specified range.

- **Syntax**: Random([start], [end])

- **Actual syntax:** MASKED WITH (FUNCTION = ‘Random(1, 12)')

Custom Masking Function

The Custom masking function allows the ability to create a custom mask using the Partial function.

CUSTOM

- **Syntax**: Partial(prefix,[padding],suffix)

- **Prefix –** Starting characters to display.
- **Padding** –Custom string for masking.
- **Suffix –** Last characters to be displayed.

Email Data Masking Function

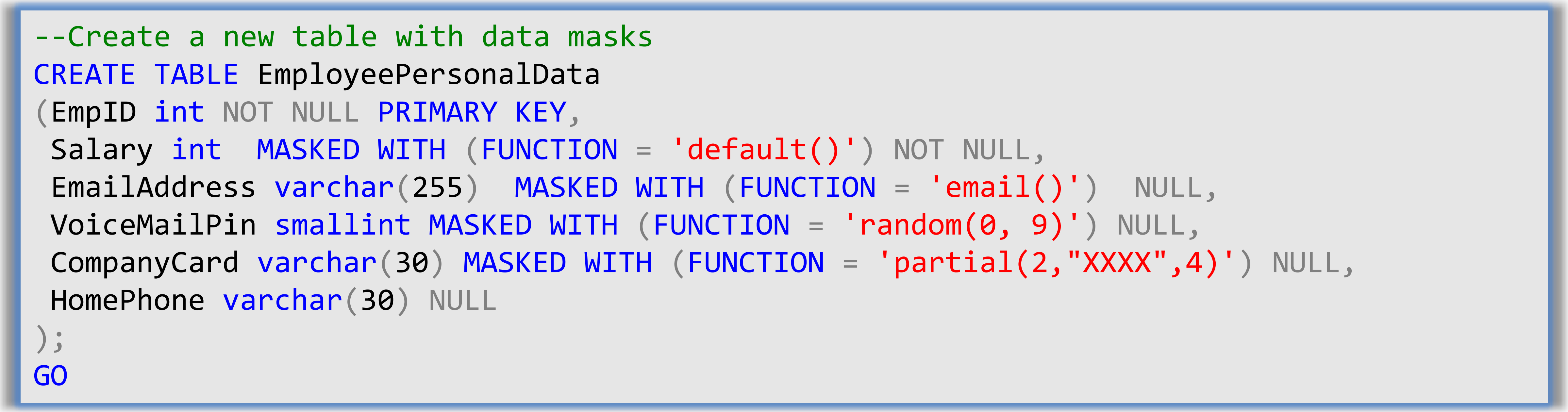
EMAIL

Masking will display the first character of an email address and mask the rest of the address with XXX@XXXX and will use the .com email suffix.

The email address of Jane.Smith@AdventureWorks.com will be masked as JXXX@XXXX.com

The email address of Susan.Jones@Contoso.net will be masked as SXXX@XXXX.com

Email Data Masking Function



```sql
--Create a new table with data masks
CREATE TABLE EmployeePersonalData
(EmpID int NOT NULL PRIMARY KEY,
 Salary int  MASKED WITH (FUNCTION = 'default()') NOT NULL,
 EmailAddress varchar(255)  MASKED WITH (FUNCTION = 'email()')  NULL,
 VoiceMailPin smallint MASKED WITH (FUNCTION = 'random(0, 9)') NULL,
 CompanyCard varchar(30) MASKED WITH (FUNCTION = 'partial(2,"XXXX",4)') NULL,
 HomePhone varchar(30) NULL
);
GO
```

Microsoft

Module 6: Security - Encryption

Lesson 1: Manage Certificates and Keys

What are certificates?

Digitally signed documents containing a public/private key pair

Certificates can be used to authenticate and/or encrypt messages between two parties.

Contain information that can either verify the sender of a message or encryption and decryption algorithm

Encryption and Cryptography

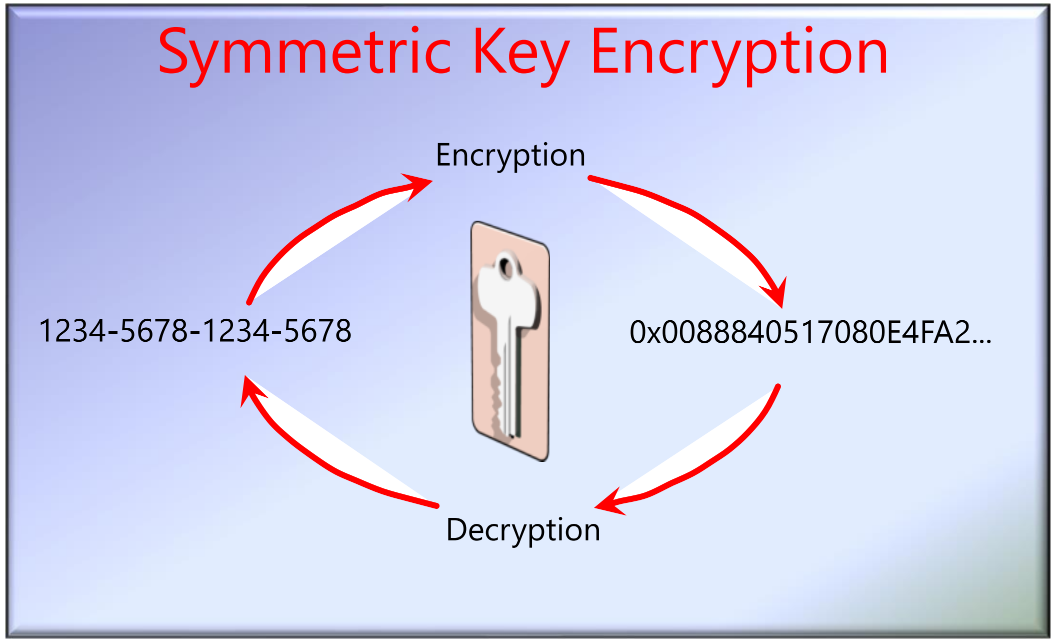
Encryption is the process of obscuring information to make it unreadable. Reversal depends on a key.

Cryptography is the science of keeping secrets.

Plaintext uses an encryption algorithm to generate ciphertext.

A decryption algorithm will revert ciphertext back to plaintext.

Symmetric vs Asymmetric Keys



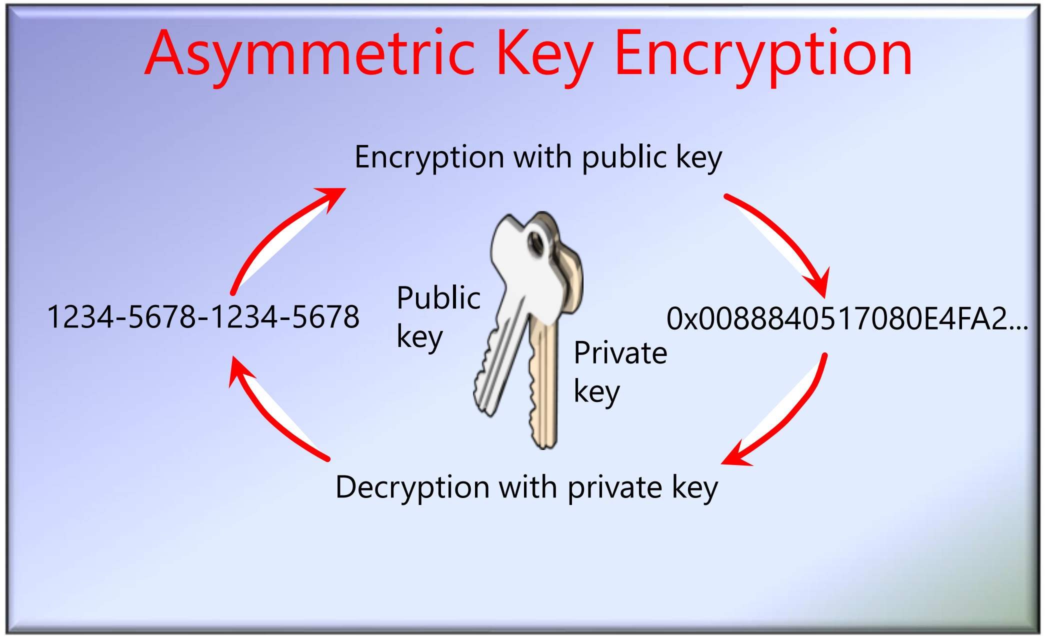
Symmetric Key Encryption

Encryption

1234-5678-1234-5678

0x0088840517080E4FA2...

Decryption



Asymmetric Key Encryption

Encryption with public key

1234-5678-1234-5678

Public key

Private key

0x0088840517080E4FA2...

Decryption with private key

When both algorithms depend on the same key, it is called a **symmetric key encryption.**

When encryption uses a pair of cryptographic keys – a public key and a private key – it is called an **asymmetric key encryption.**

Encryption Usage

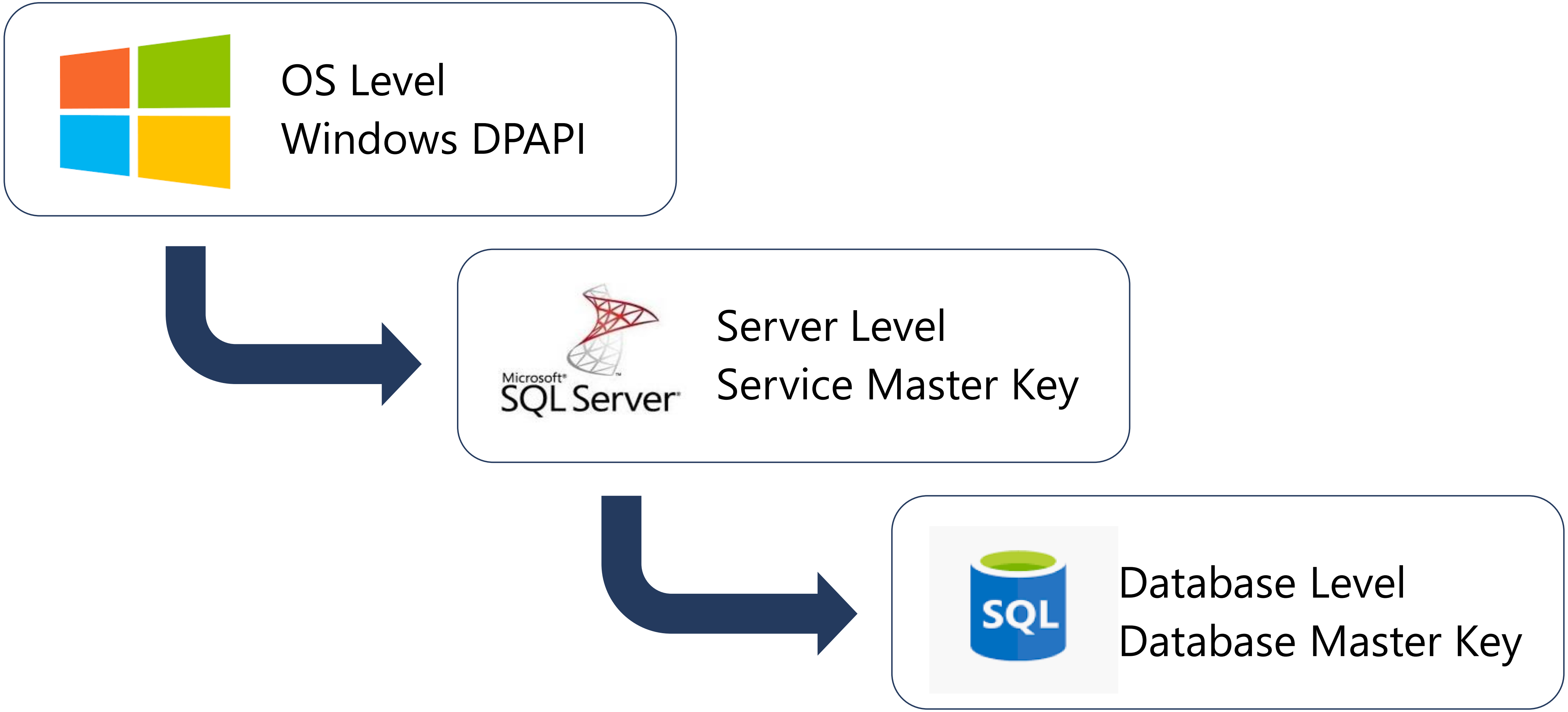
The symmetric and asymmetric keys are used in an encryption hierarchy that parallels the hierarchy of securable objects.

Encryption can be used with any level of SQL Server securable, data, files, and connection.

SQL Server allows administrators and developers to choose among several encryption algorithms

The algorithms are implemented using the Windows Crypto API.

Encryption Hierarchy



OS Level
Windows DPAPI

Server Level
Service Master Key

Database Level
Database Master Key

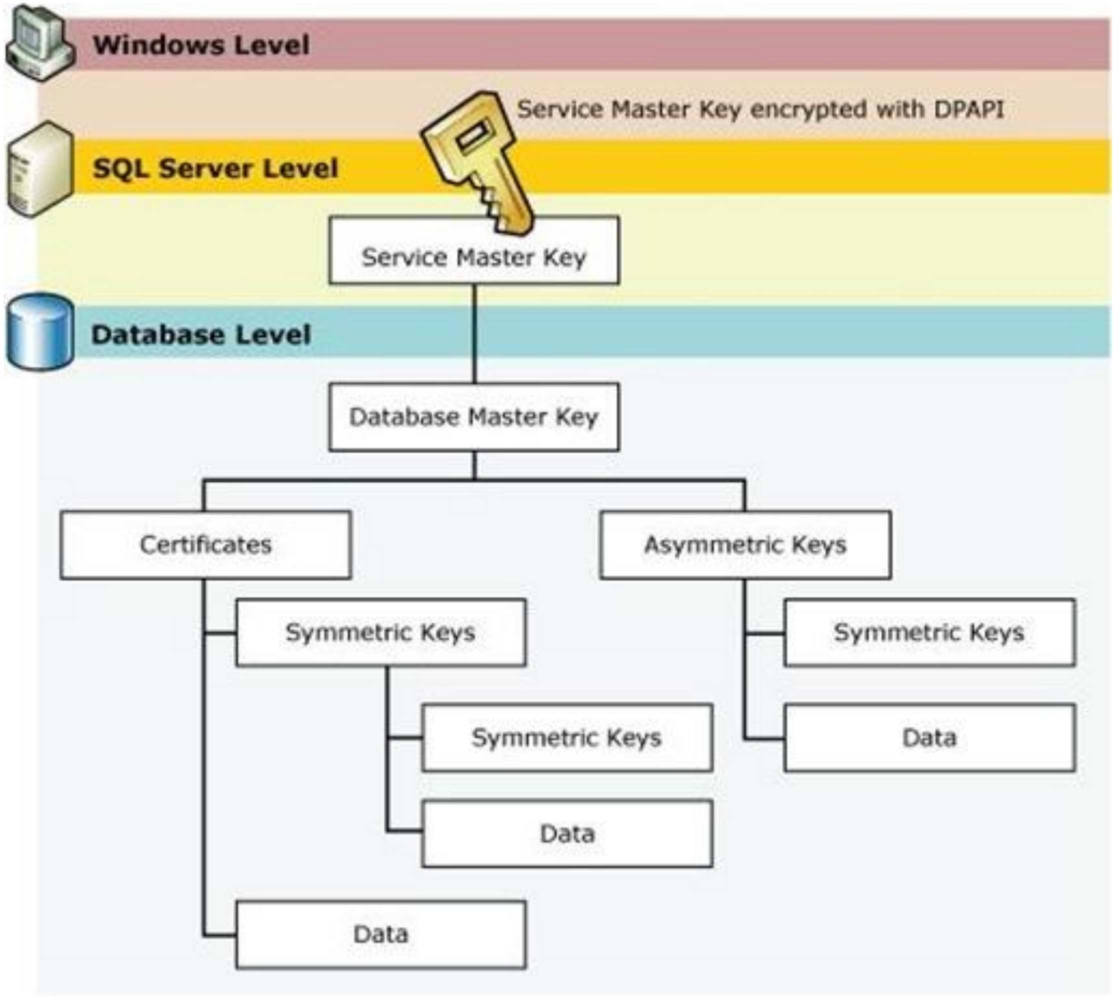
Encryption Hierarchy

Service Master Key

Database Master Key

Certificates

Database Encryption Key (DEK)



Service Master Key

The Service Master Key is the root of the SQL Server encryption hierarchy and is specific to each instance.

Encrypted by Windows DPAPI that uses a key derived from the SQL Server service account and the computer credentials

Generated automatically the first time it is need to encrypt another key

Can be regenerated by using ALTER SERVICE MASTER KEY statement

Database Master Key

- Used to encrypt database level certificates and asymmetric keys
- There can be only one Database Master Key per database
- Encrypted by the Service Master Key by default
- Can also be encrypted by using a password

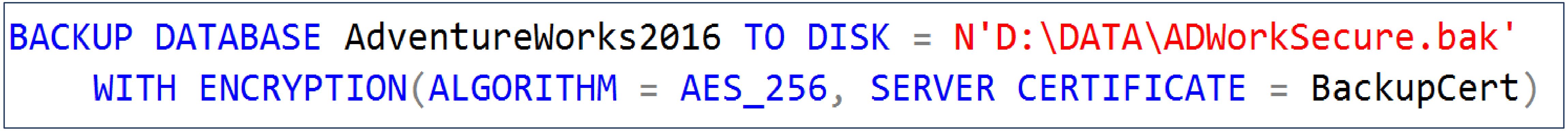
Lesson 2: Backup Encryption

Backup Encryption

- Available in Standard or Enterprise Edition
- Encrypts native backup files
- Based on certificate stored securely within SQL Server engine
- Supported algorithms are AES (128, 192, 256) and Triple DES

Backup Encryption Prerequisites

- Create a Database Master Key for the master database
- Create a certificate or asymmetric key to use for backup encryption
- It is very important to back up the certificate or asymmetric key



```
BACKUP DATABASE AdventureWorks2016 TO DISK = N'D:\DATA\ADWorkSecure.bak'
    WITH ENCRYPTION(ALGORITHM = AES_256, SERVER CERTIFICATE = BackupCert)
```

Backup Encryption Restrictions

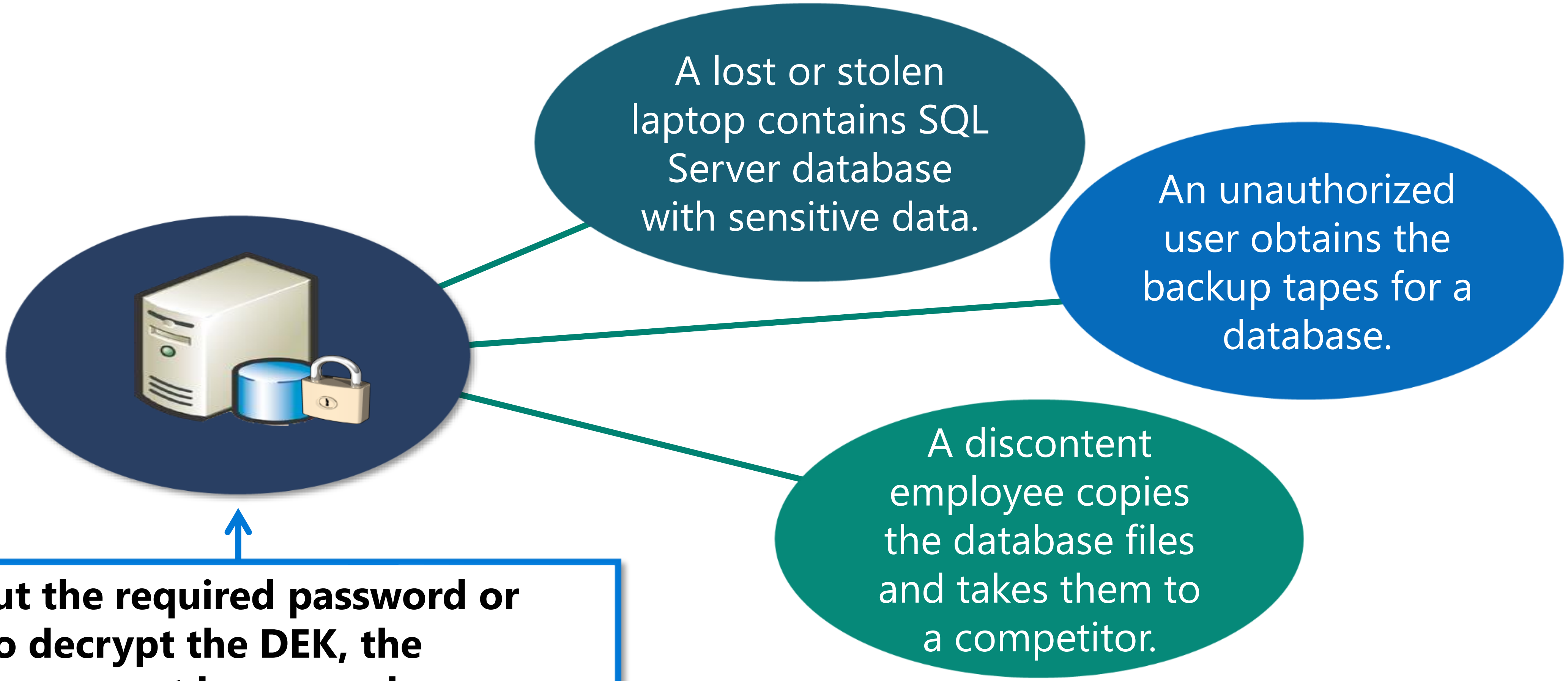
- If using asymmetric keys, must store on an EKM provider
- Express and Web editions do not support backup encryption.
- Restoring encrypted backups to Express and Web editions is allowed
- Appending to existing backup sets in not supported

Lesson 3: Transparent Data Encryption (TDE)

Transparent Data Encryption Benefits

- Performs all the cryptographic operations at the database level
- Removes any need for application developers to create custom code to encrypt and decrypt data
- Data is encrypted as it is written to disk and decrypted as it is read from disk
- Because SQL Server manages encryption and decryption transparently, there is no need for application changes
- Protects data, files, and backups at rest

Scenarios



A lost or stolen laptop contains SQL Server database with sensitive data.

An unauthorized user obtains the backup tapes for a database.

A discontent employee copies the database files and takes them to a competitor.

Without the required password or HSM to decrypt the DEK, the database cannot be opened.

How Transparent Data Encryption works

Entire database is encrypted

Protects data, files, and backups at rest

Tempdb encrypted by default when any database has TDE enabled

Backups are also encrypted for TDE databases

How Transparent Data Encryption works

Works at storage I/O level (encryption at rest)

Check the status of encryption using **sys.dm_database_encryption_keys**

Encryption happens before writing to disk and performed by background threads

- Page protection (checksum/torn page) is applied after encryption
- Page protection (e.g. checksums) is checked before decryption
- Database pages are decrypted when read into memory

Why Transparent Data Encryption

Securing data at rest

No changes in the application layer

Performance should not be affected

Scalability

Space should not be increased or affected

Supports AES and 3DES encryption algorithms

Impact of Transparent Data Encryption

Performance Impact

- Encryption or decryption scan
- Query impact

Backup/Restore and Detach/Attach

- Certificate should have two files
- Backup both files

Key Management

- If unwanted access to the key should happen, consider changing the certificate

High Availability

- Create a Master Key on the mirror (secondary replica or stand by server)
- Backup the certificate on the principal and restore it on the mirror.

Lesson 4: Overview of Always Encrypted

Always Encrypted - Benefits

Allows customers to securely store sensitive data outside of their trust boundary while protecting data from highly privileged users.

Prevention of data disclosure

- Client-side encryption of sensitive data using keys that are never given to database system

Queries on encrypted data

- Support for equality comparison, including join, group by, and distinct operators

Application transparency

- Minimal application changes through server and client library enhancements

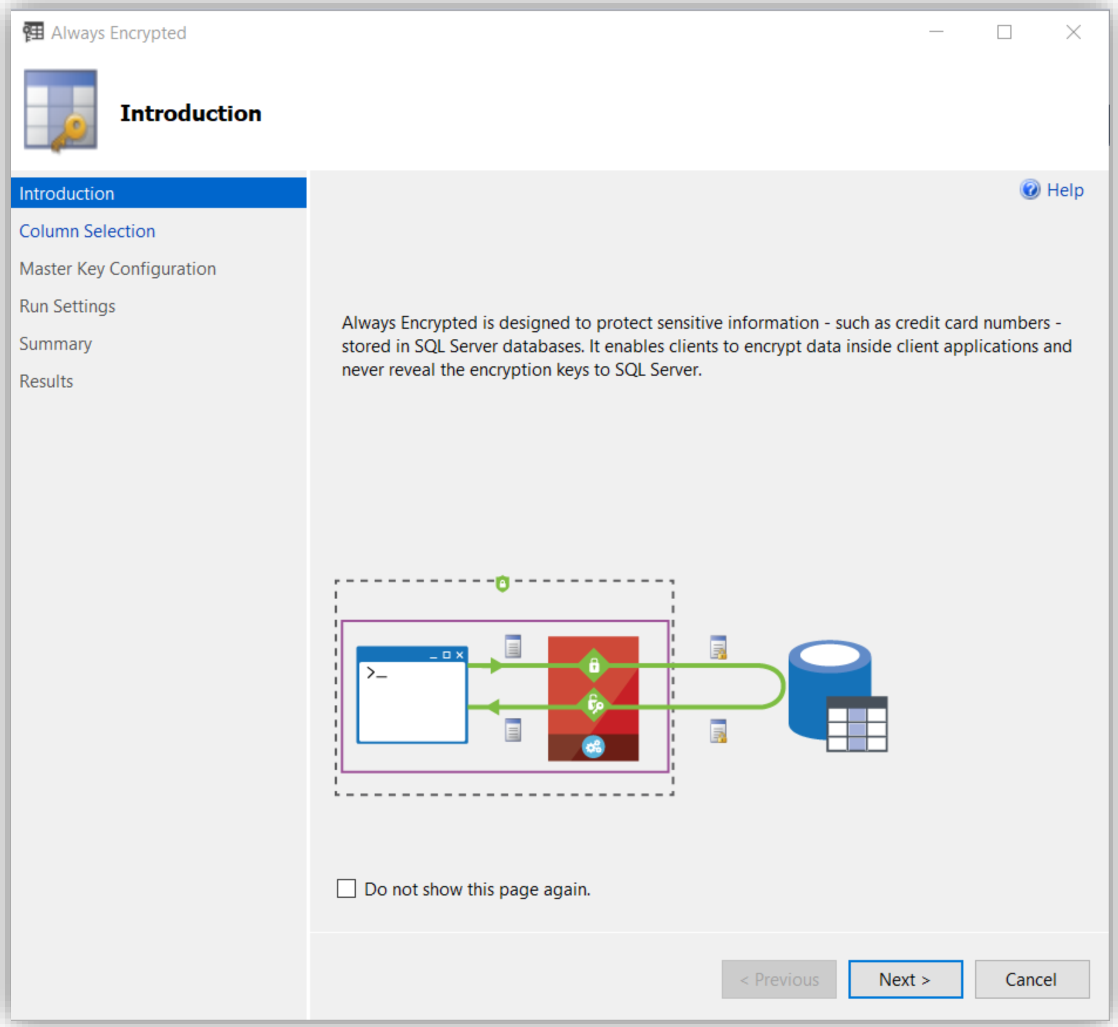
What is Always Encrypted?

Capability

- The ADO.NET client library provides transparent client-side encryption
- Microsoft SQL Server executes T-SQL queries on encrypted data

Benefits

- Sensitive data remains encrypted and can always be queried , on-premises and in the cloud
- Unauthorized users never have access to data or keys
- No application changes



Always Encrypted is designed to protect sensitive information - such as credit card numbers - stored in SQL Server databases. It enables clients to encrypt data inside client applications and never reveal the encryption keys to SQL Server.

□ Do not show this page again.

Always Encrypted - Capabilities and Functions

Migration of sensitive data in application

- SQL Server only handles encrypted data—not plain text values

Automatic encryption and decryption of sensitive data

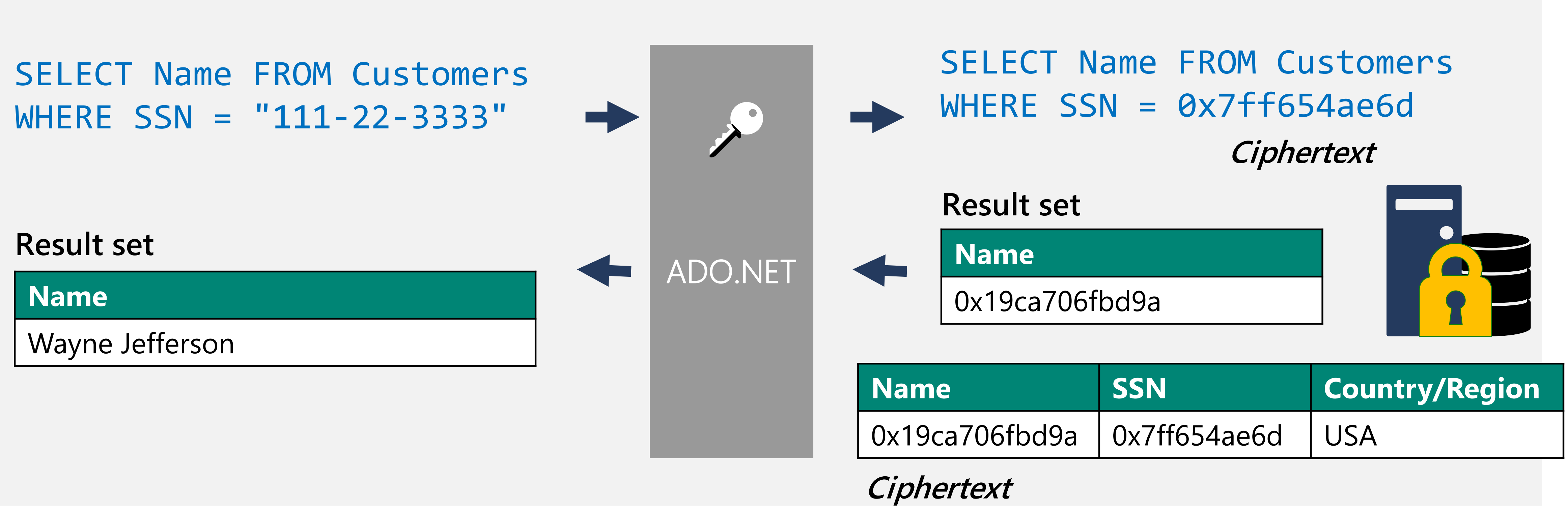
- Automatically rewrites queries to preserve semantics to application
- Driver transparently decrypts data

Bulk loading of encrypted data

- Use ALLOW_ENCRYPTED_VALUE_MODIFICATIONS option for bulk loading

How does Always Encrypted work?

Encrypted sensitive data and corresponding keys are never seen in plain text in SQL Server



```
SELECT Name FROM Customers
WHERE SSN = "111-22-3333"
```

ADO.NET

```
SELECT Name FROM Customers
WHERE SSN = 0x7ff654ae6d
```

Ciphertext

Result set

Name
Wayne Jefferson

Result set

Name
0x19ca706fbd9a

Name	SSN	Country/Region
0x19ca706fbd9a	0x7ff654ae6d	USA

Ciphertext

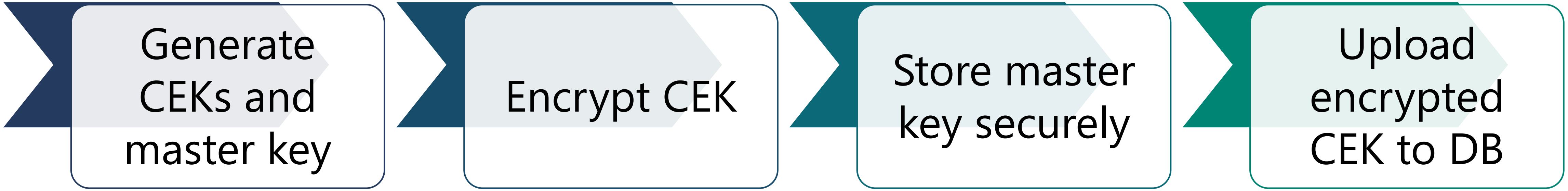
Column Keys

CMK – Column Master Key is used to encrypt other keys, always in client's control, and in an external key store

- Azure Key Vault
- Windows Certificate Store
- Hardware Security Sections

CEK – Column Encryption Key is a content encryption key

Key provisioning



Generate CEKs and master key

Encrypt CEK

Store master key securely

Upload encrypted CEK to DB

Encryption Types

Randomized

- Unpredictable results, more secure
- No support for equality searches, joins, grouping, or indexing
- Use for data that is returned, but not queried

Deterministic

- Predictable results, less secure
- Use for data that must be queried (equality support only)
- Easier to guess by examining encryption results
  - Increased risk for small value sets (True/False)

Encryption Methodologies

Randomized encryption

- Encrypt (‘123-45-6789’) = 0x0123A99C
- Repeat: Encrypt (‘123-45-6789’) = 0x01EB449B

Deterministic encryption

- Encrypt (‘123-45-6789’) = 0x17cfd50a
- Repeat: Encrypt (‘123-45-6789’) = 0x17cfd50a

Data Encryption Algorithm

Always Encrypted uses the AEAD_AES_256_CBC_HMAC_SHA_256 algorithm to encrypt data in the database

Ciphertext length varies depends on the data type

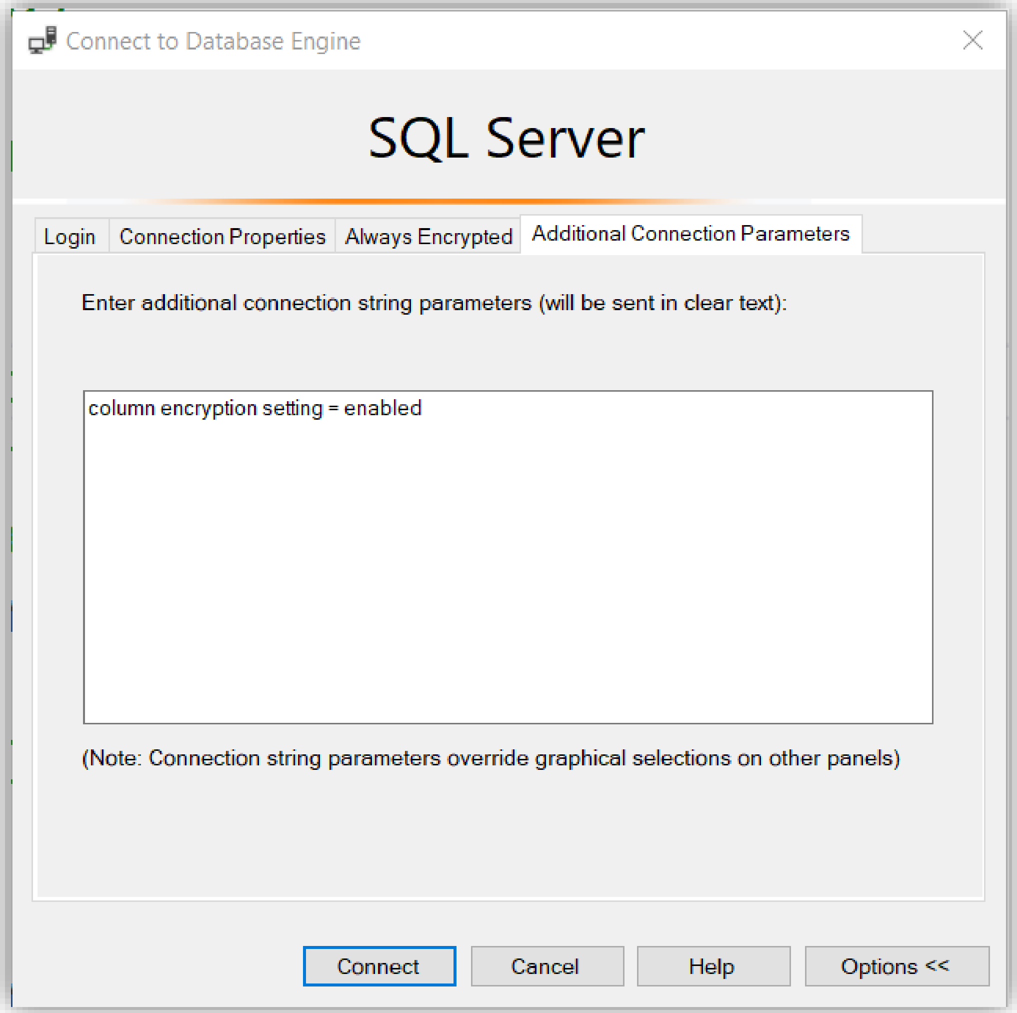
Always Encrypted Catalog Views

```
--Always Encrypted catalog views
SELECT * FROM sys.column_master_keys
SELECT * FROM sys.column_encryption_keys
SELECT * FROM sys.column_encryption_key_values
```

Find Always Encrypted Columns

```sql
--Find columns protected by Always Encrypted
SELECT c.name as E_Column, c.column_encryption_key_id,
   cek.name as E_Key, encryption_type_desc,
   encryption_algorithm_name
FROM sys.columns as c
JOIN sys.column_encryption_keys as cek
ON c.column_encryption_key_id =
   cek.column_encryption_key_id
WHERE c.column_encryption_key_id IS NOT NULL
```

Column Encryption Setting = Enabled



Enter additional connection string parameters (will be sent in clear text):

```
column encryption setting = enabled
```

(Note: Connection string parameters override graphical selections on other panels)

Permissions

ALTER ANY COLUMN MASTER KEY

- Required to create and delete a column master key

ALTER ANY COLUMN ENCRYPTION KEY

- Required to create and delete a column encryption key

VIEW ANY COLUMN MASTER KEY DEFINITION

- Required to access and read column master key metadata objects while managing keys or querying encrypting columns

VIEW ANY COLUMN ENCRYTPION KEY DEFINITION

- Required to access and read column encryption key metadata objects while managing keys or querying encrypting columns

Limitations

Not supported when columns use any of these datatypes

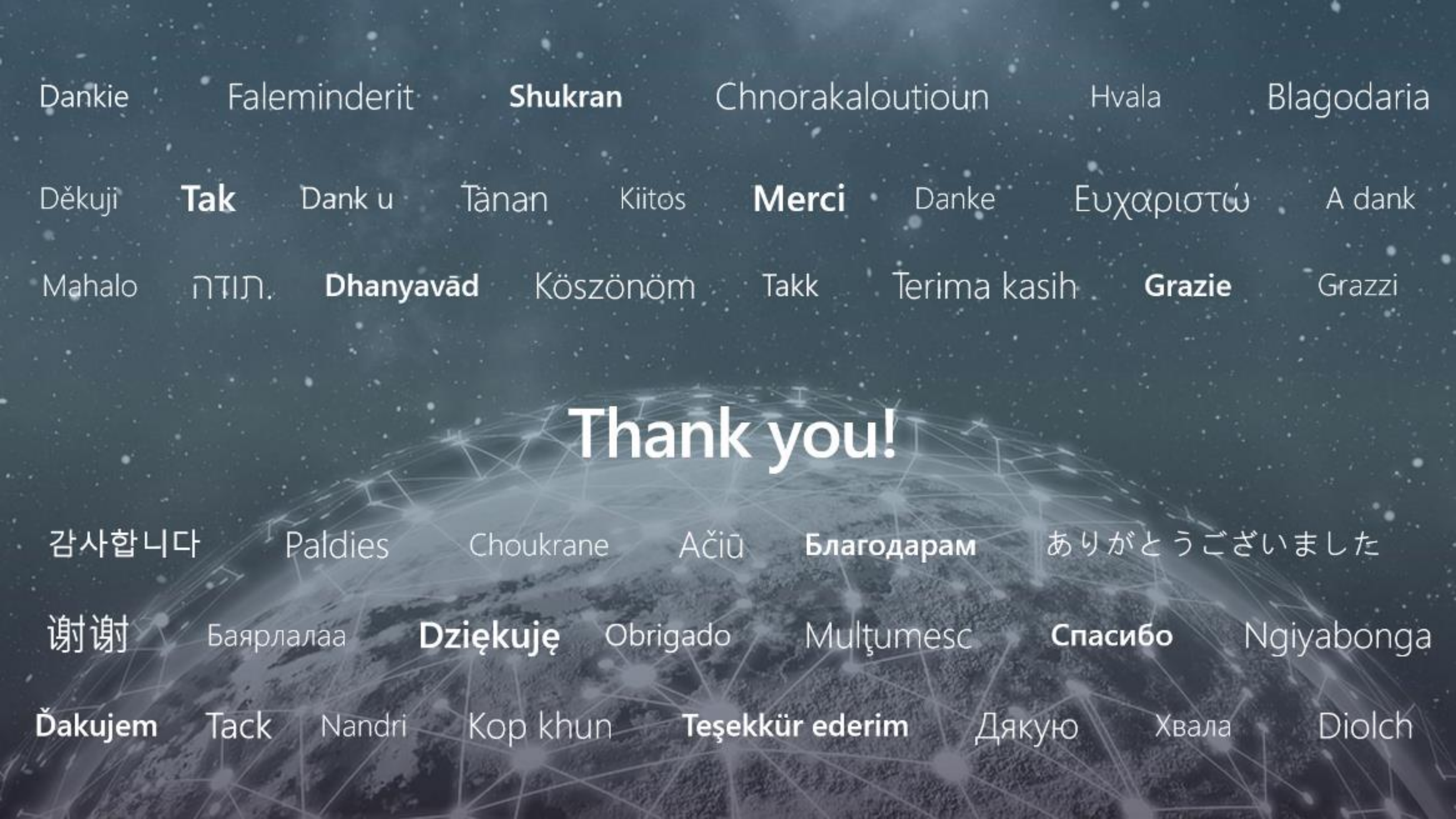
- xml, hierarchyid, rowversion, image, text, ntext, geography, geometry, user-defined types, or sql_variant

Clauses that cannot be used for encrypted columns

- FOR XML
- FOR JSON PATH

Features that do not work on encrypted columns

- Transactional or merge replication
- Distributed queries (linked servers)



Dankie Faleminderit **Shukran** Chnorakaloutioun Hvala Blagodaria

Děkuji **Tak** Dank u Tänan Kiitos **Merci** Danke Ευχαριστώ A dank

Mahalo תודה. **Dhanyavād** Köszönöm Takk Terima kasih **Grazie** Grazzi

Thank you!

감사합니다 Paldies Choukrane Ačiū **Благодарам** ありがとうございました

谢谢 Баярлалаа **Dziękuję** Obrigado Mulţumesc **Спасибо** Ngiyabonga

Ďakujem Tack Nandri Kop khun **Teşekkür ederim** Дякую Хвала Diolch

Microsoft