Microsoft

SQL Server I/O and Database Structure

Module 2

Learning Units covered in this Module

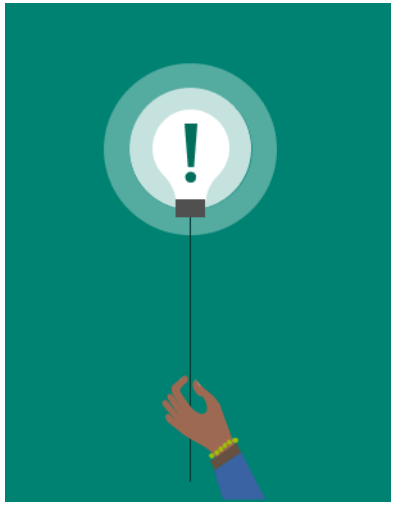
- Lesson 1: SQL Server Disk I/O
- Lesson 2: SQL Server Log File Structure

Lesson 1: SQL Server Disk I/O

Objectives

After completing this learning, you will be able to:

- Identify SQL Server disk I/O operations.
- Identify I/O patterns for data files and log files.
- Monitor SQL Server I/O using:
  - Performance Monitor Counters
  - Dynamic Management Views
  - Wait Statistics



Database files and filegroups

Database files

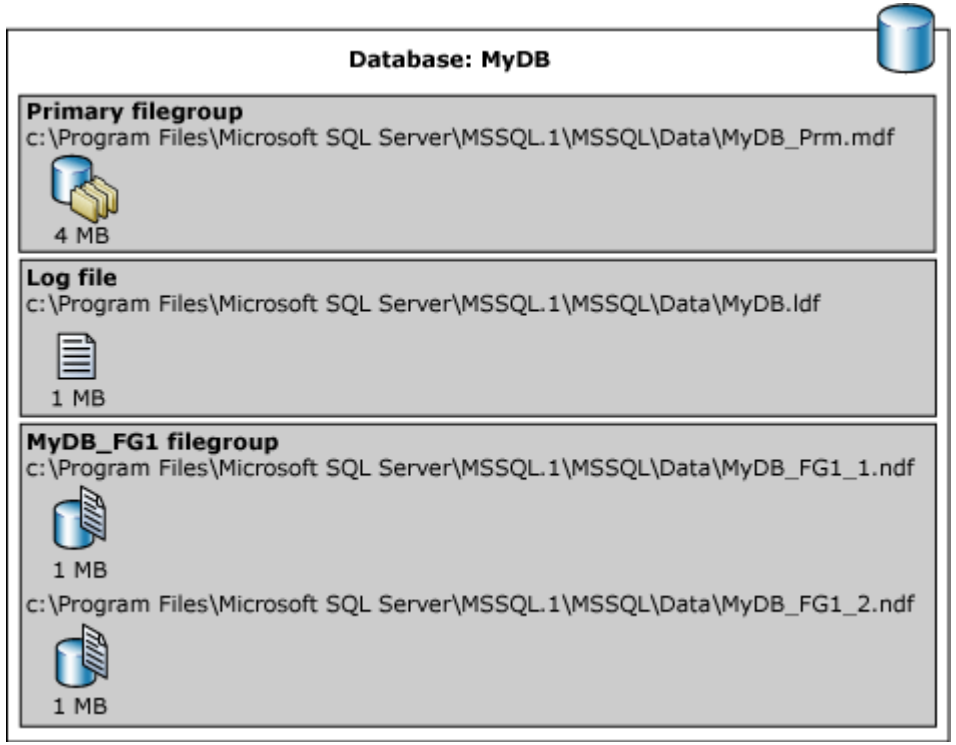
A database is composed by at least two operating system files:

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



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

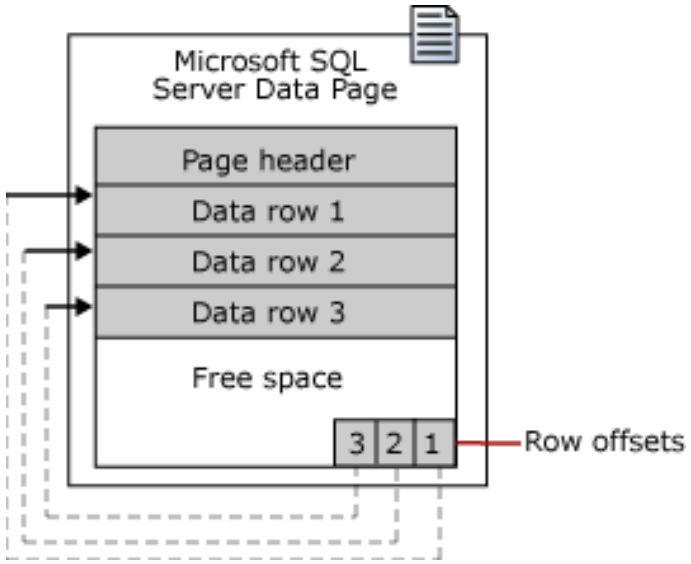
Log Files

- 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

Pages and Extents architecture

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

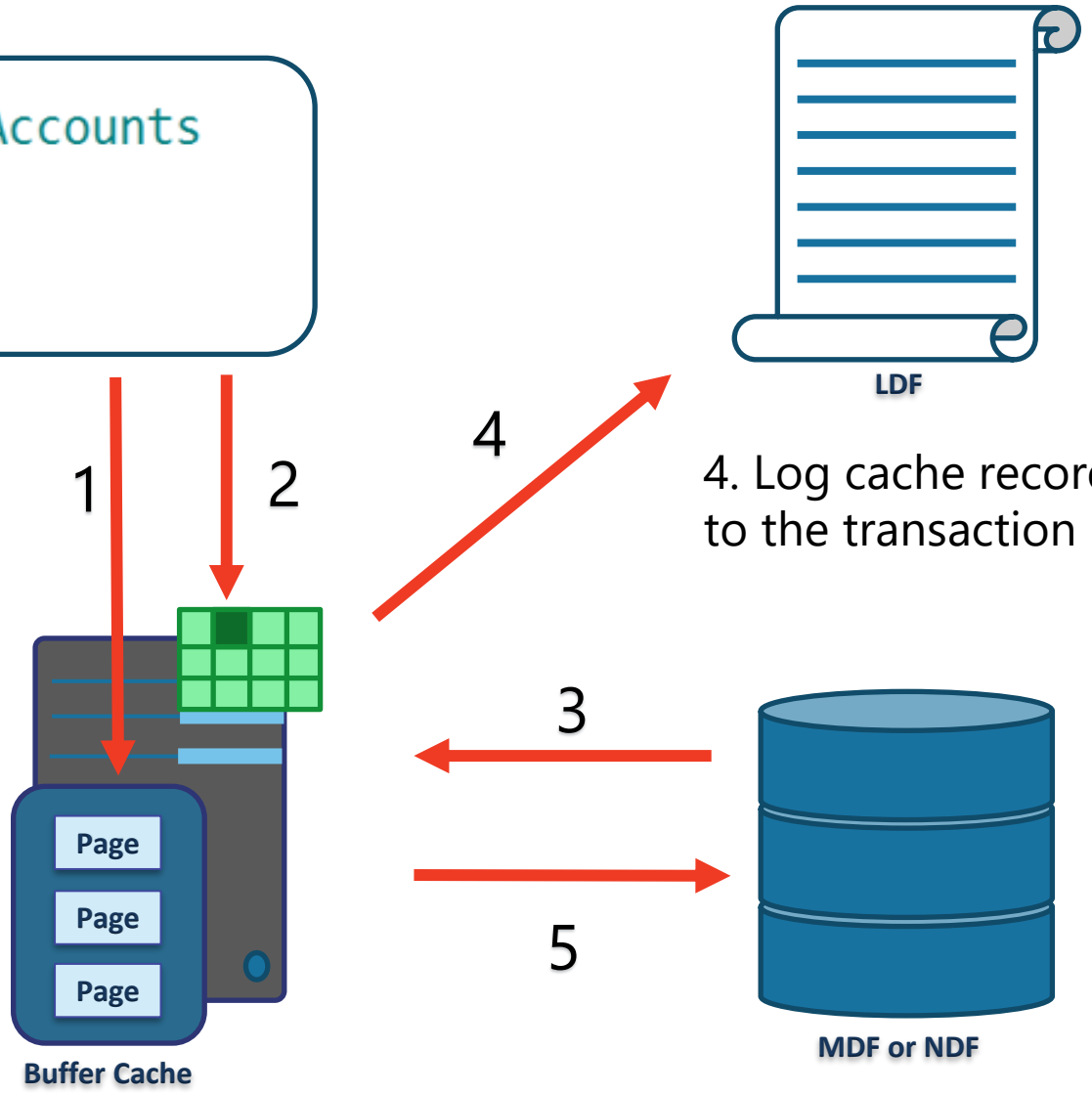
- The disk space allocated to a data file (.mdf or .ndf) is logically divided into pages.
- Each page is 8 KB in size
- Pages are numbered contiguously from 0 to n.
- Disk I/O operations are performed at the page level.



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

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.

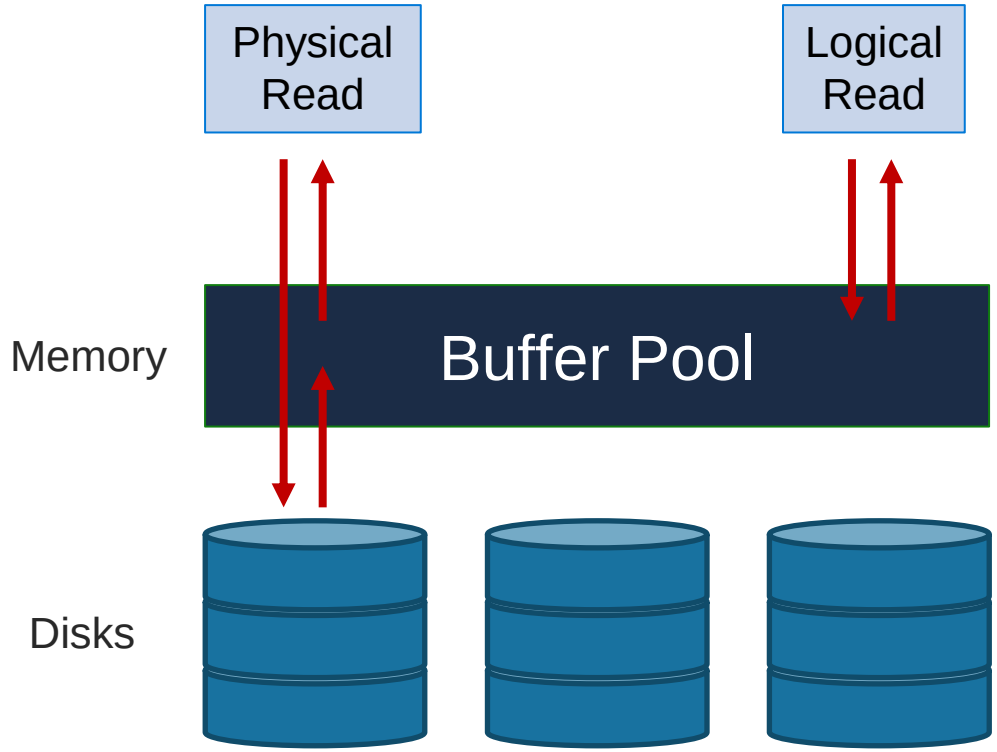
SQL Server Buffer Pool

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

```
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
```

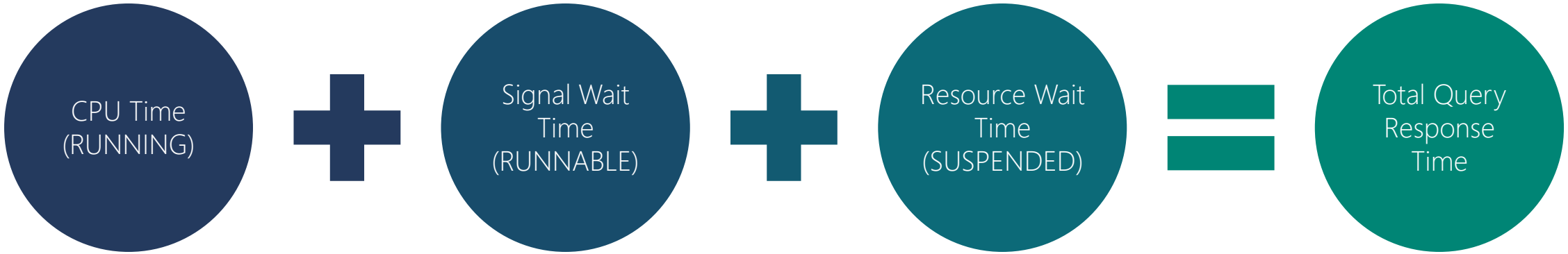
Used to identify physical reads and logical reads for a query

```
(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.



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

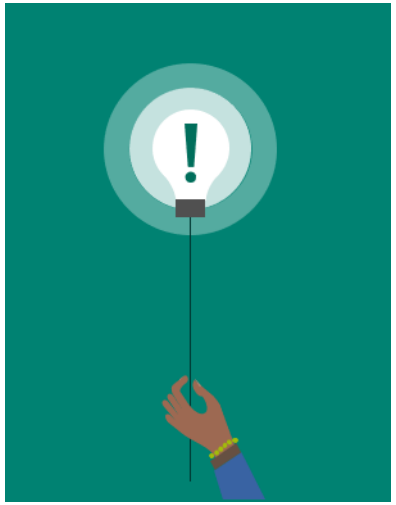
Internal – Issued by various server operations

Lesson 2: SQL Server Log File Structure

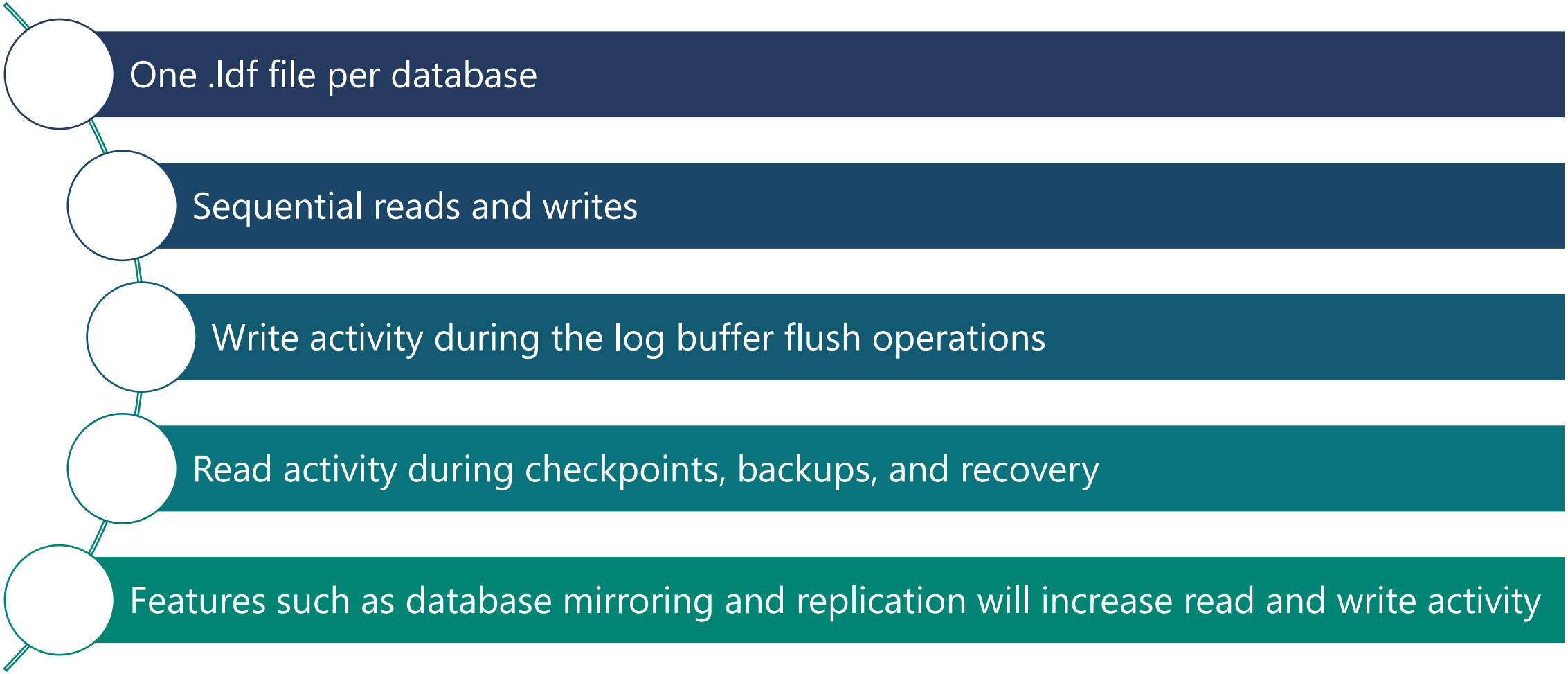
Objectives

After completing this learning, you will be able to:

- Understand the log file structure.
- Explain SQL Server logging process.



SQL Server disk I/O patterns: transaction log



Transaction Log

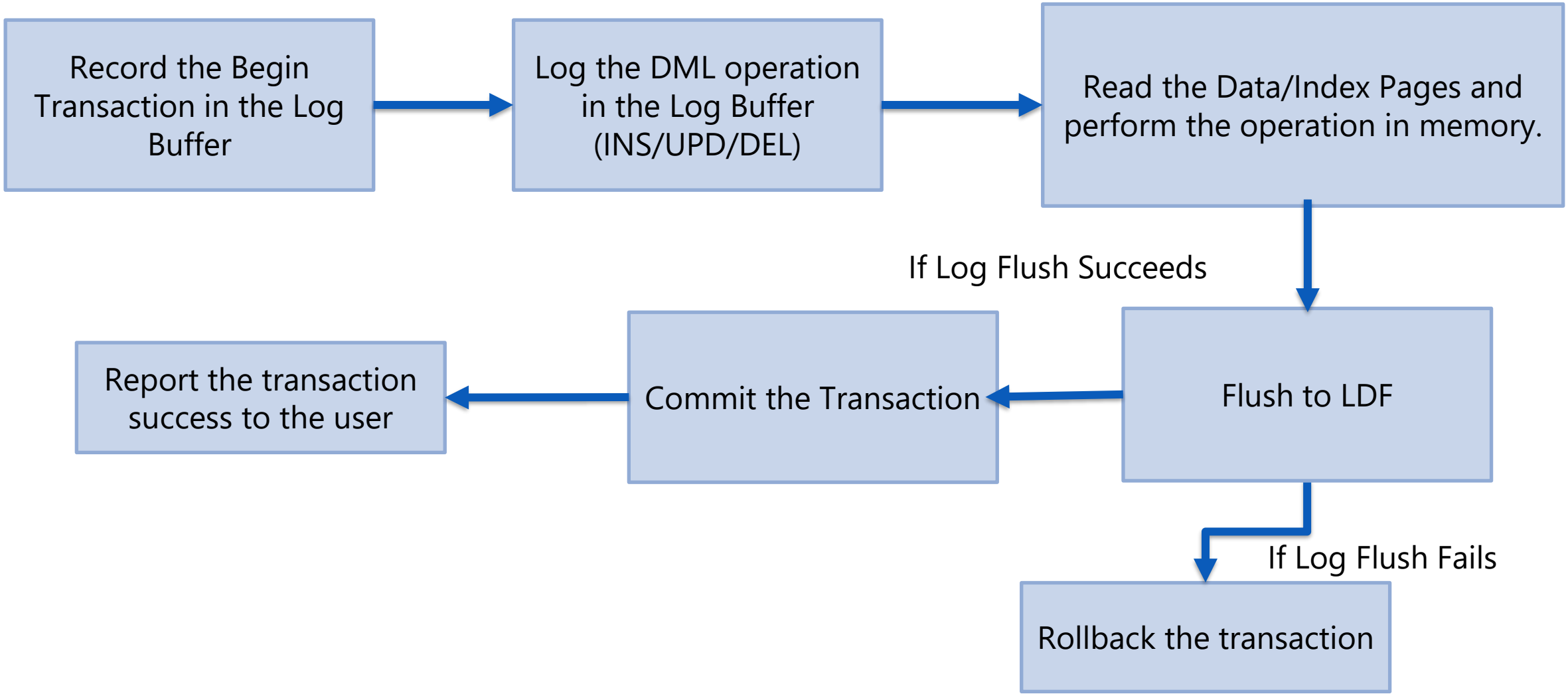
Sequence of log records contained in one or more physical files

Records identified by increasing Logical Sequence Numbers (LSNs)

Recorded operations:

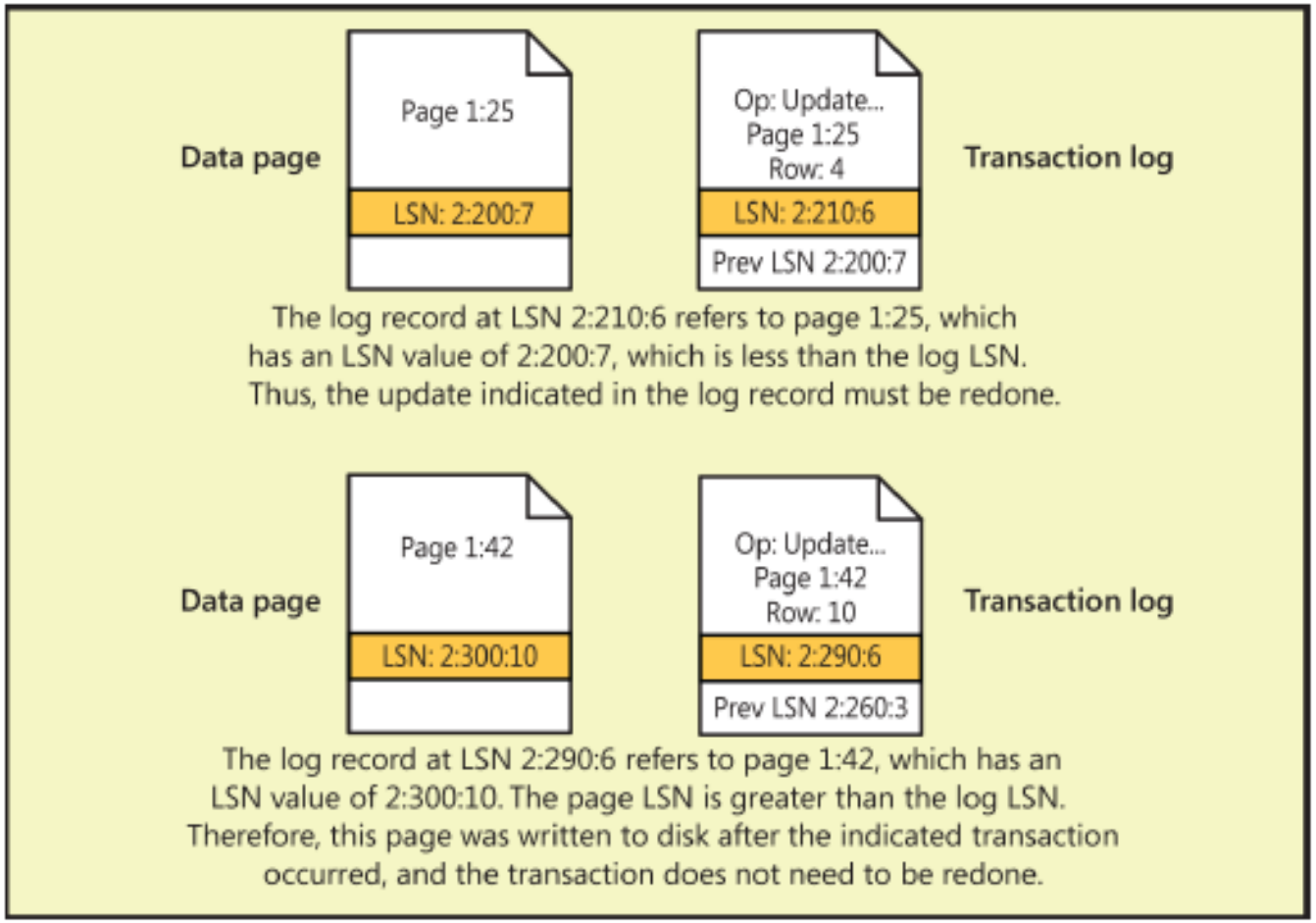
- Start and end of each transaction
- All database modifications
- Extent and page allocation or deallocation
- Creating and dropping objects

Microsoft SQL Server Transaction Logging



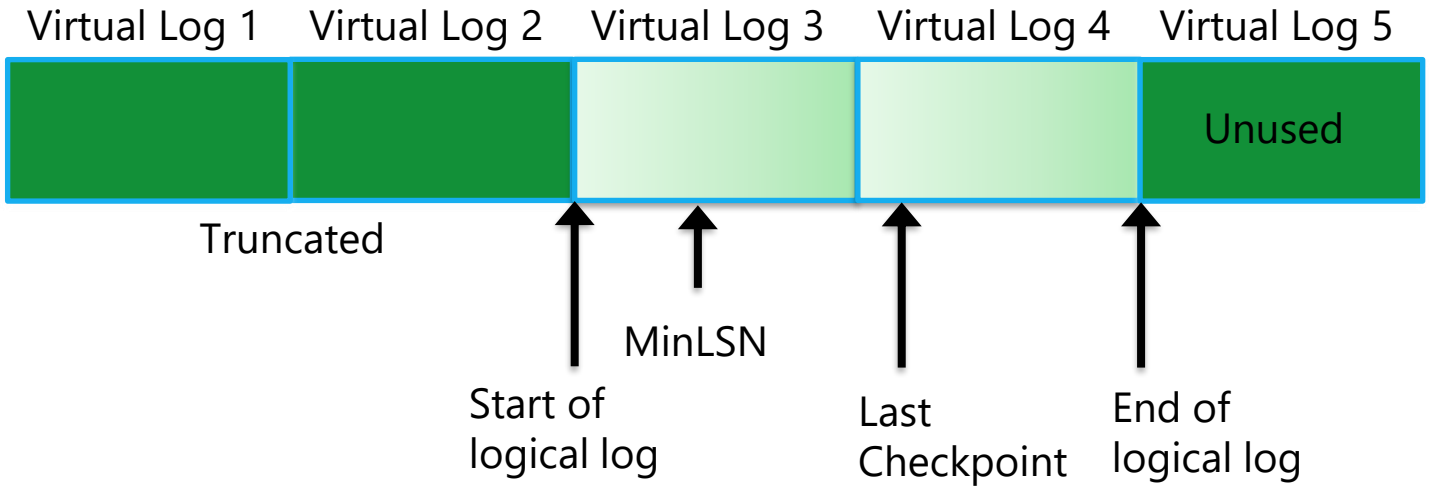
Page LSN and recovery

- 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



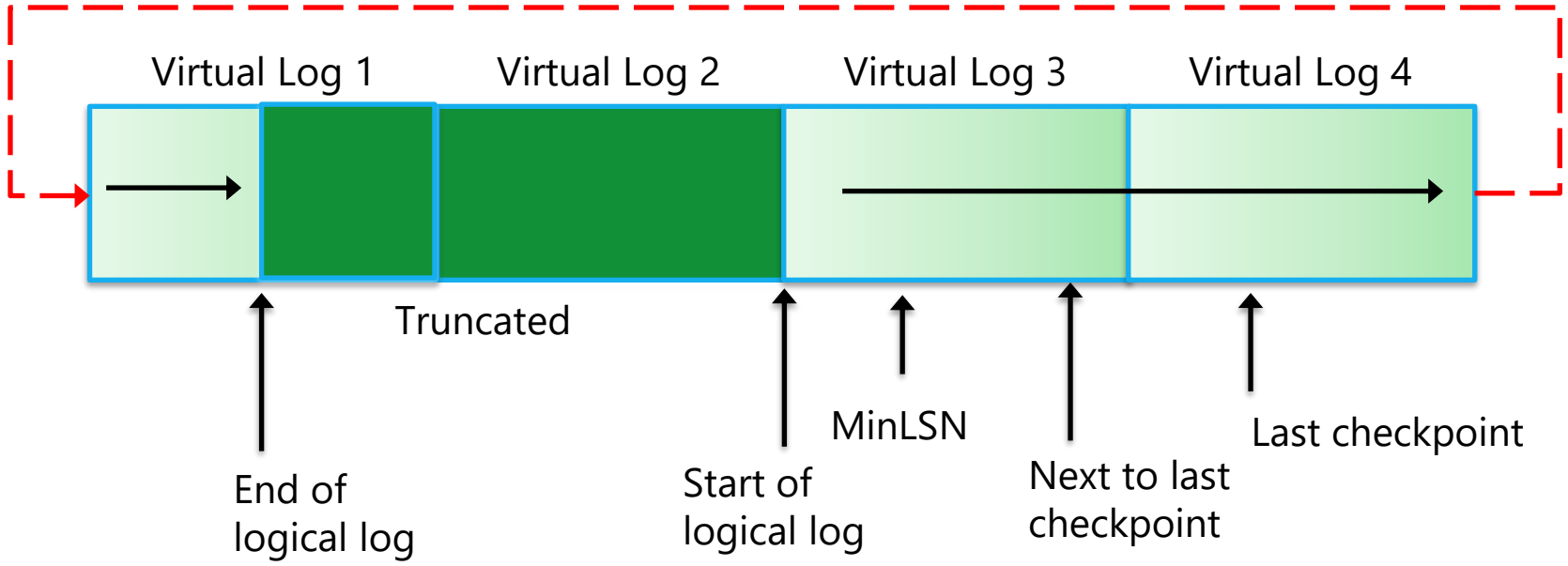
Physical Log File Structure

- 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)

Demonstration

Examining log use

Questions?