SQL SERVER SCIENCE

Max Vernon

Senior SQL Server Consultant

Top-10 contributor at dba.stackexchange.com

Founder of SQLServerScience.com

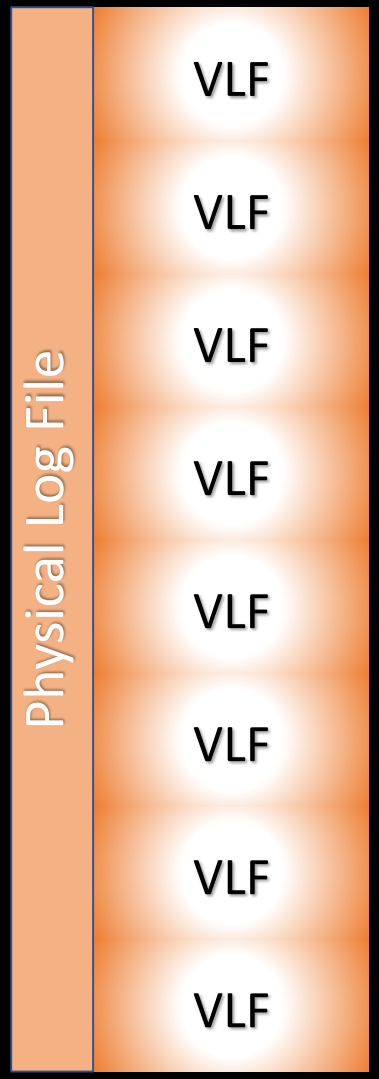
I’m sorry, I can’t let you expand that log file, Dave!

SQL Server Log Management can be tricky, but it’s worth doing it the *right* way.

Overview of the Transaction Log

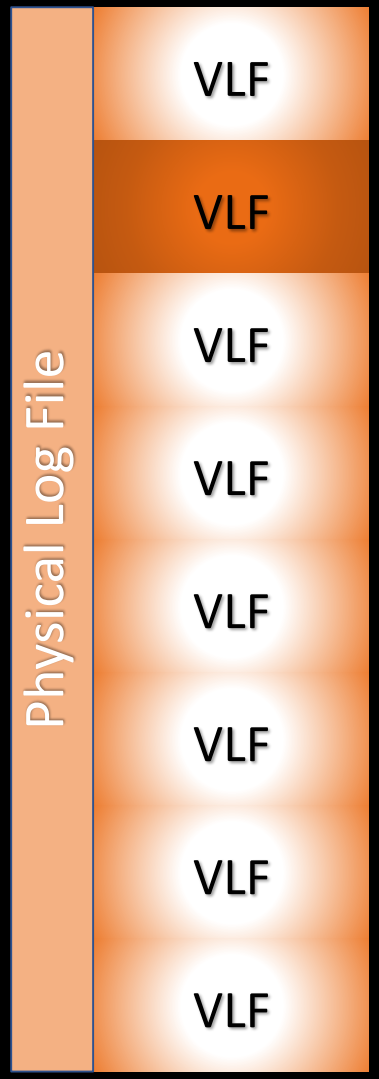
- Every database has its own transaction log.
- Every alteration to the database is logged in the transaction log. Even in simple recovery model.
- Transaction log supports point-in-time-recovery when configured correctly.
- Availability Groups, Log Shipping, Database Mirroring require full recovery model.
- Critical part of the ACID guarantees.
- Consider storing data and log files on separate disks, but realize the uptime implications.

Logical Structure of the Transaction Log



- Each physical log file is segmented into a number of Virtual Log Files, or VLFs.
- VLFs are allocated atomically, as SQL Server requires Transaction Log Space.
- Once all transactions in a VLF are committed, the VLF is marked for truncation.
- Truncated VLFs are re-used in round-robin fashion.

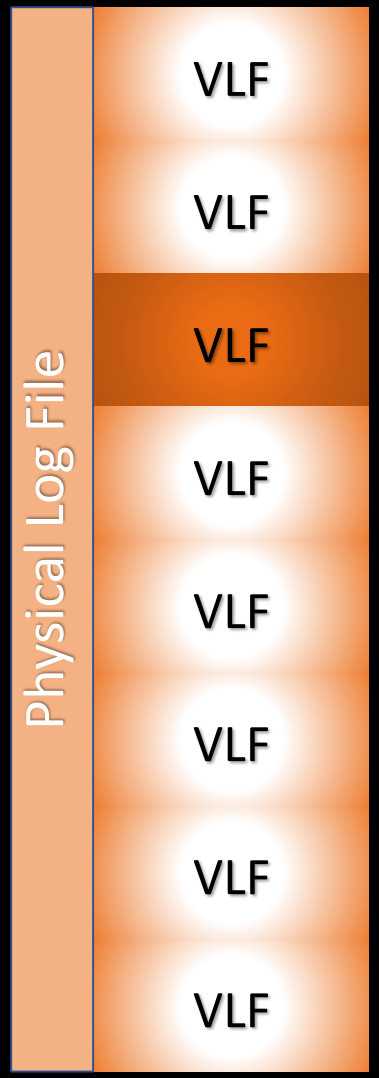
Logical Structure of the Transaction Log



- Each physical log file is segmented into a number of Virtual Log Files, or VLFs.
- VLFs are allocated atomically, as SQL Server requires Transaction Log Space.
- Once all transactions in a VLF are committed, the VLF is marked for truncation.
- Truncated VLFs are re-used in round-robin fashion.

Active VLF

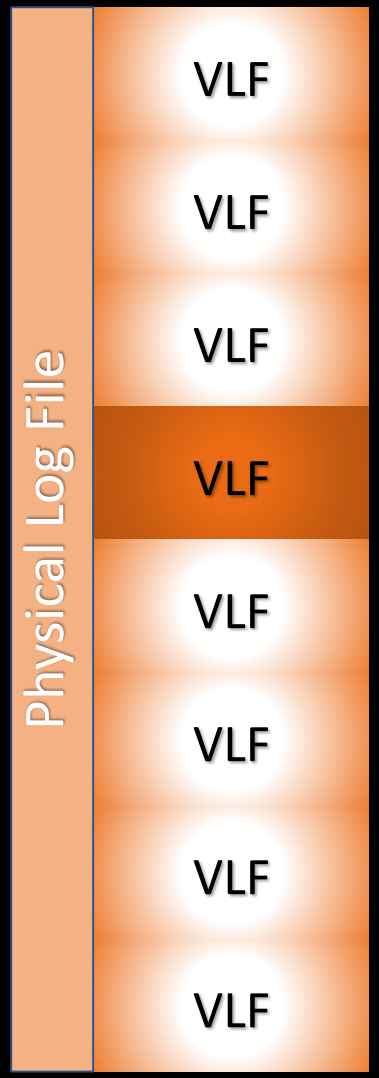
Logical Structure of the Transaction Log



- Each physical log file is segmented into a number of Virtual Log Files, or VLFs.
- VLFs are allocated atomically, as SQL Server requires Transaction Log Space.
- Once all transactions in a VLF are committed, the VLF is marked for truncation.
- Truncated VLFs are re-used in round-robin fashion.

Active VLF

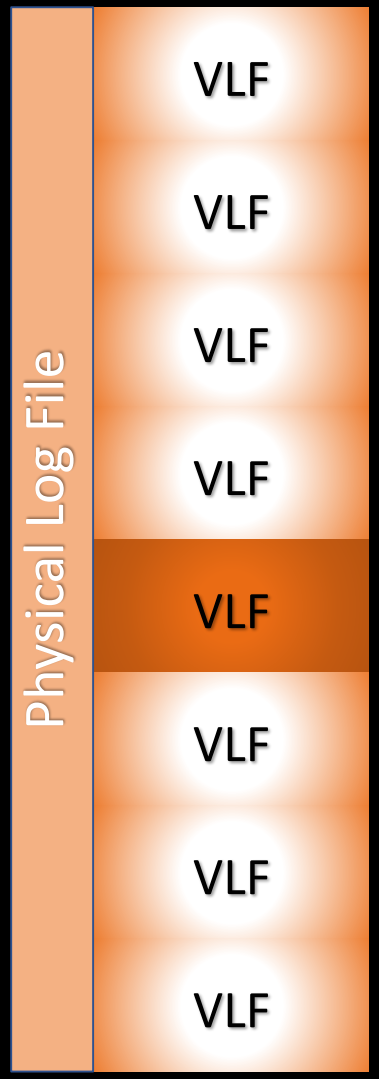
Logical Structure of the Transaction Log



- Each physical log file is segmented into a number of Virtual Log Files, or VLFs.
- VLFs are allocated atomically, as SQL Server requires Transaction Log Space.
- Once all transactions in a VLF are committed, the VLF is marked for truncation.
- Truncated VLFs are re-used in round-robin fashion.

Active VLF

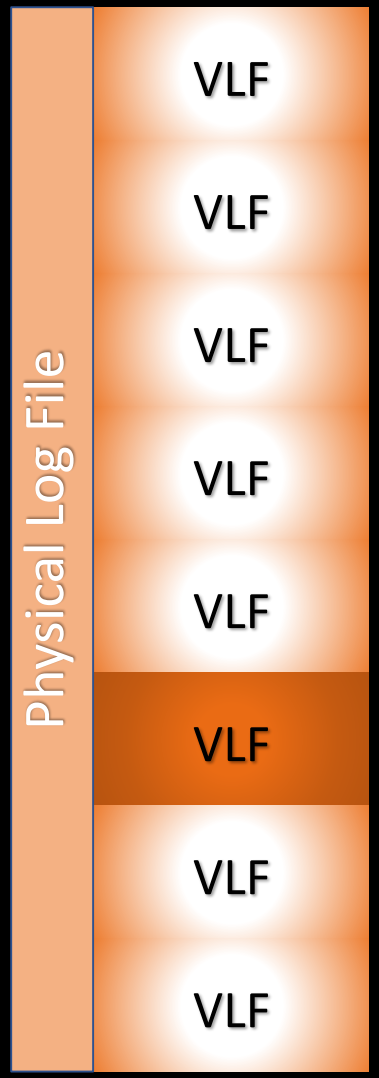
Logical Structure of the Transaction Log



- Each physical log file is segmented into a number of Virtual Log Files, or VLFs.
- VLFs are allocated atomically, as SQL Server requires Transaction Log Space.
- Once all transactions in a VLF are committed, the VLF is marked for truncation.
- Truncated VLFs are re-used in round-robin fashion.

Active VLF

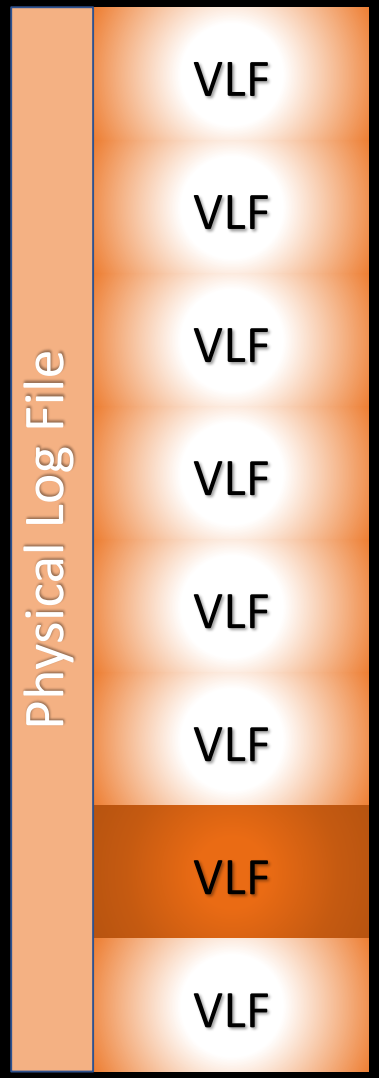
Logical Structure of the Transaction Log



- Each physical log file is segmented into a number of Virtual Log Files, or VLFs.
- VLFs are allocated atomically, as SQL Server requires Transaction Log Space.
- Once all transactions in a VLF are committed, the VLF is marked for truncation.
- Truncated VLFs are re-used in round-robin fashion.

Active VLF

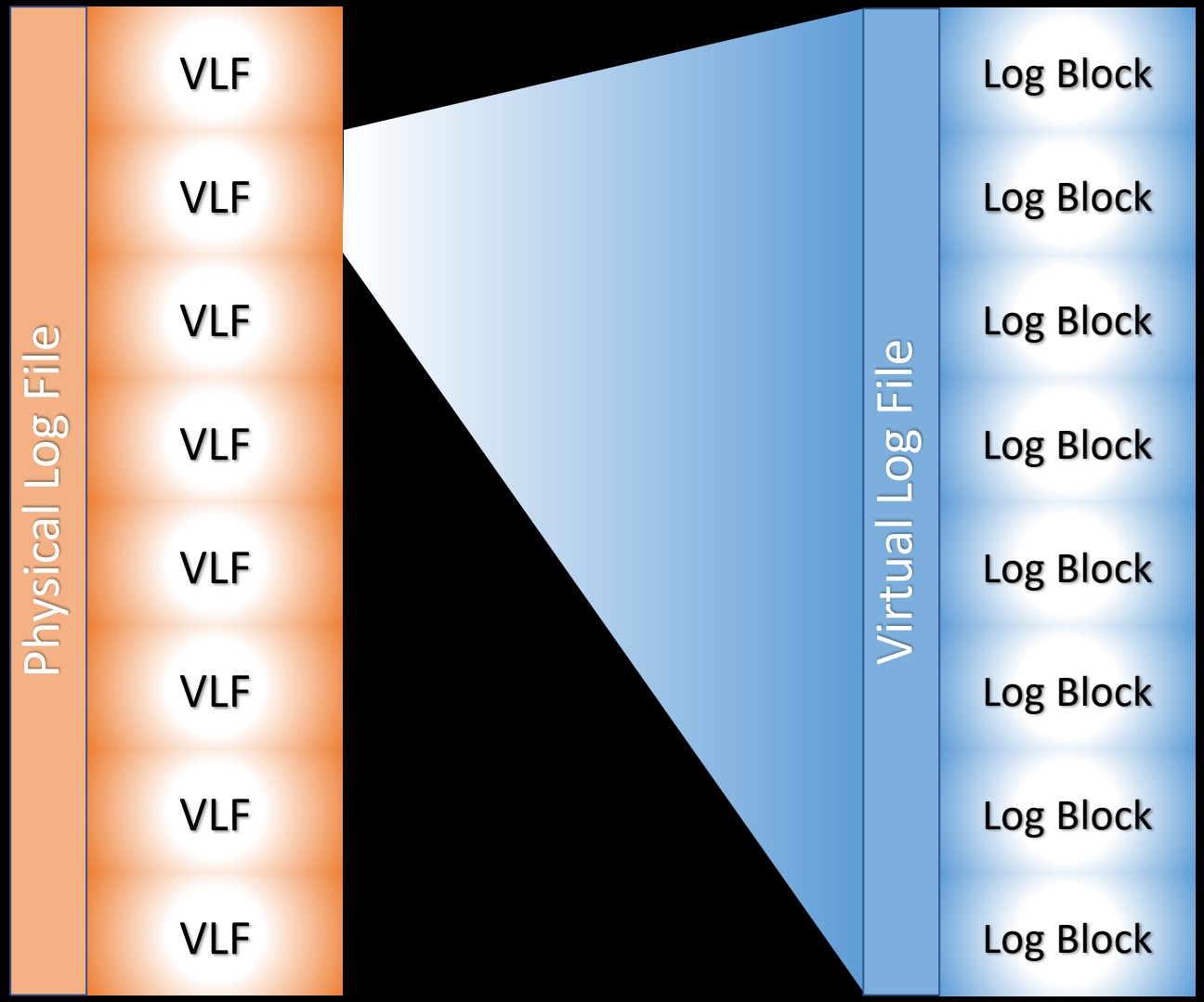
Logical Structure of the Transaction Log



- Each physical log file is segmented into a number of Virtual Log Files, or VLFs.
- VLFs are allocated atomically, as SQL Server requires Transaction Log Space.
- Once all transactions in a VLF are committed, the VLF is marked for truncation.
- Truncated VLFs are re-used in round-robin fashion.

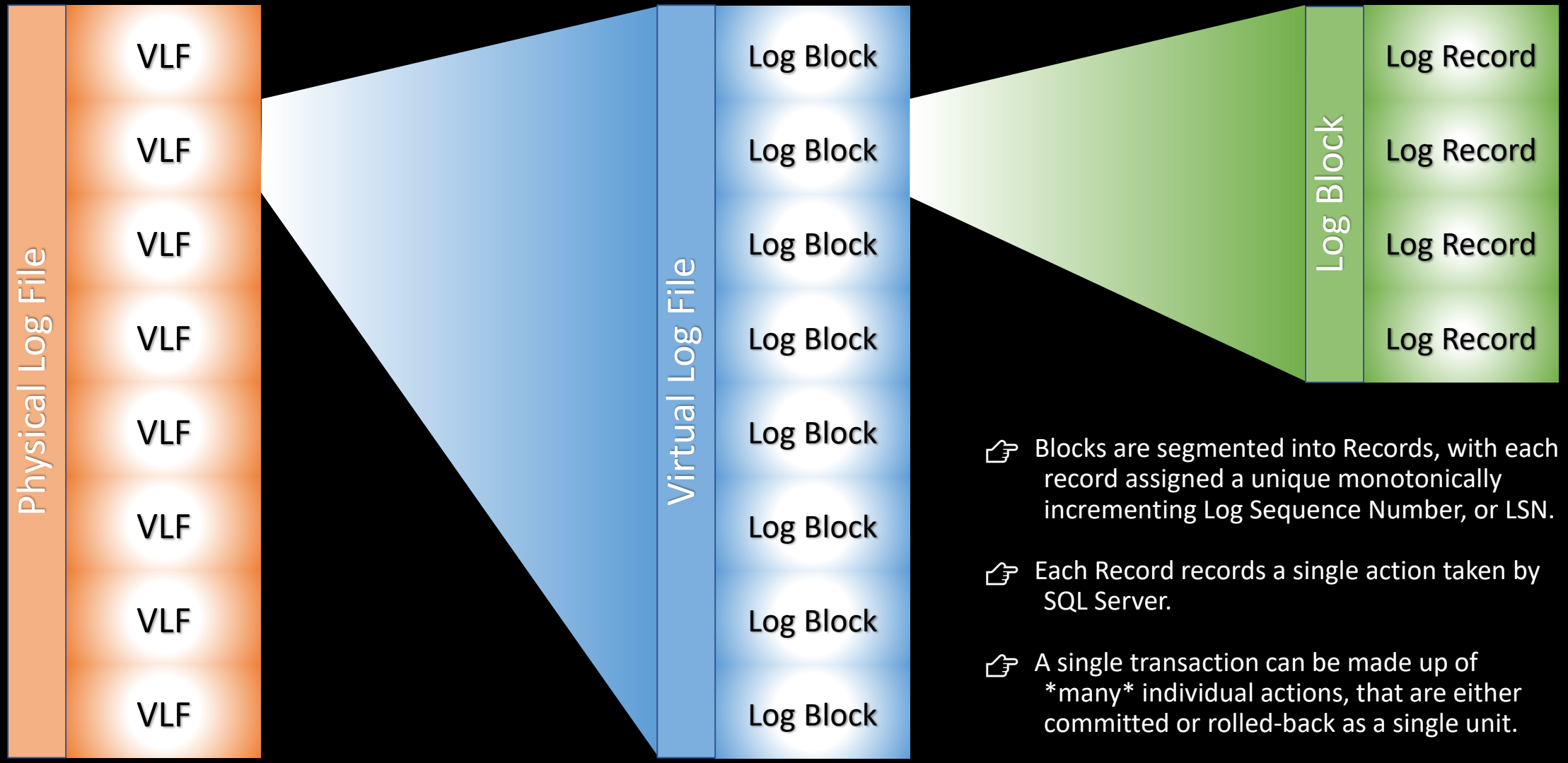
Active VLF

Logical Structure of the Transaction Log



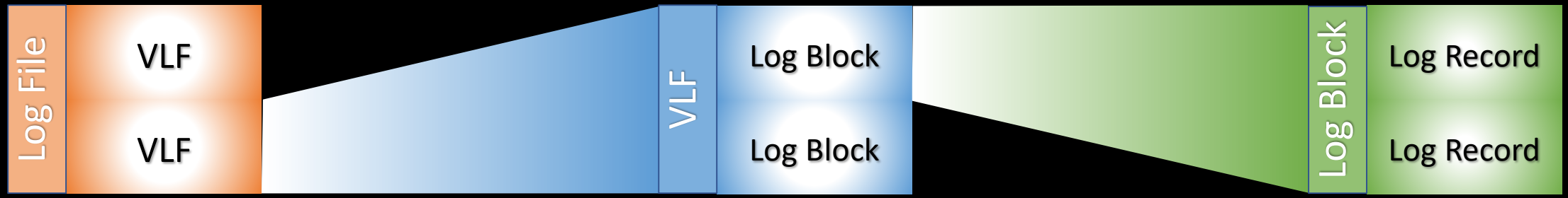
- VLFs are segmented into blocks that are allocated atomically, in serial order.
- Each Block consumes 512 bytes to 60 KB.
- Individual Blocks are written to disk in a single write operation, ahead of changes being written to any data file.

Logical Structure of the Transaction Log



- Blocks are segmented into Records, with each record assigned a unique monotonically incrementing Log Sequence Number, or LSN.
- Each Record records a single action taken by SQL Server.
- A single transaction can be made up of *many* individual actions, that are either committed or rolled-back as a single unit.

Logical Structure of the Transaction Log



- Recovery consists of reading each individual log record, and performing either roll-forward, or roll-back.
- As the number of active log records increases, so does recovery time.
- VLFs can be variable size. A large number of small VLFs may result in vastly increased recovery times. Large VLFs take time to create since the log file must be “zeroed” at each growth increment.

Transaction Log Growth

Check how many VLFs you have with `DBCC LOGINFO`
Is the Log set to grow by a small percentage?

- When growing a transaction log file, SQL Server uses the following rules to determine how many VLFs are created:
  - If the next growth is less than 1/8 of current log physical size, then create 1 VLF that covers the growth size (Starting with SQL Server 2014 (12.x))
  - If the next growth is more than 1/8 of the current log size, then use the pre-2014 method:
    - If growth is less than 64MB, create 4 VLFs that cover the growth size (e.g. for 1 MB growth, create four 256KB VLFs)
    - If growth is from 64MB up to 1GB, create 8 VLFs that cover the growth size (e.g. for 512MB growth, create eight 64MB VLFs)
    - If growth is larger than 1GB, create 16 VLFs that cover the growth size (e.g. for 8 GB growth, create sixteen 512MB VLFs)

When You Don’t Manage Growth 😢

- Configured with 1MB growth “for efficiency”
- Lots of small transactions; a classic “OLTP” system.
- Full Recovery Model with *daily* transaction log backups
- Database Mirroring
- Serial single-threaded Recovery Analysis Phase

… Disaster!

When You Don’t Manage Growth 😢

- Configured with 1MB growth “for efficiency”
- Lots of small transactions; a classic “OLTP” system.
- Full Recovery Model with *daily* transaction log backups
- Database Mirroring
- Serial single-threaded Recovery Analysis Phase

… Disaster!

20 hours of Database Recovery

When You Don’t Manage Growth 😢

Changes in Modern Versions of SQL Server 😎

- SQL Server 2014+ has slightly more “sane” VLF creation algorithm.
- SQL Server 2016+ never configures the log with 10% growth by default
- SQL Server 2017+ shows the following in the SQL Server Error Log when it detects a database with excessive VLFs:
  Database <name> has more than 10000 virtual log files which is excessive. Too many virtual log files can cause long startup and backup times. Consider shrinking the log and using a different growth increment to reduce the number of virtual log files.
- SQL Server 2019 has Accelerated Database Recovery and Persistent Version Store in 2019
  - Analysis phase - The process remains the same as today with the addition of reconstructing sLog and copying log records for non-versioned ops.
  - sLog is a secondary in-memory log stream that stores log records for non-versioned operations (such as metadata cache invalidation, lock acquisitions, and so on). The sLog is:
    - Low volume and in-memory
    - Persisted on disk by being serialized during the checkpoint process
    - Periodically truncated as transactions commit
    - Accelerates redo and undo by processing only the non-versioned operations
    - Enables aggressive transaction log truncation by preserving only the required log records
    - Preliminary testing against CTP versions shows up to 10% decrease in OLTP performance with ADR
    - Will likely have considerable changes before RTM

Summary

- 👉 Ensure you treat the transaction log like the first-class citizen it is.
- 👉 Don’t assume SQL Server is just going to “look after it” for you.
- 👉 Find databases with high VLF counts using the code at https://www.sqlserverscience.com/recovery/detect-databases-high-vlf-count
- 👉 Fix databases with high VLF counts using the code at https://www.sqlserverscience.com/recovery/fix-high-vlf-count