SQL Server
Transaction Log Internals

Ramkumar Gopal

Living For SQL Server

Chennai SQL Server User Group

Blog: http://sql-articles.com/category/cssug/

FB : http://www.facebook.com/groups/cssug/

DL : chnsqlug@googlegroups.com

Yet to Subscribe? Benefits are more, subscribe soon

Blog: http://www.sqlservercentral.com/blogs/livingforsqlserver/

http://www.facebook.com/LivingForSqlServer

CSSUG Meet – 17 Nov 2012

Agenda

- Basics
- DBCC LOGINFO & Demo
- Fn_dblog & Demo

Basics

Slno	Property	Data Files	Transaction Log Files
1	File	One primary (.MDF) and optional secondary (.NDF) data Files	Ideally one T-Log file (.LDF).
2	Access type	Random	Sequential and Circular. Will grow if required
3	Logically divided as	Pages	Virtual Log Files (VLF)
4	Size	Page size is always 8 KB	VLFs are not always in fixed size and count. Depends of Log file size and growth. If Log file size is, i) Less than 1 MB - 2 VLF ii) Between 1 MB and 64 MB - 4 VLF iii) Between 64 MB and 1 GB - 8 VLF iv) Greater than 1 GB - 16 VLF

T-Log Basics

Slno	Property	Data Files	Transaction Log Files
5	States	Page states: Free or Allocated in Mixed/Uniform extents (GAM, SGAM, IAM and PFS are used for this)	VLF states: 1. Active (2) - One or more transactions in progress 2. Recoverable (2) - VLF can be reused once CHECKPOINT (simple recovery) or T-Log backup is taken (Full and Bulk logged) 3. Reusable (0) - reusable VLFs 4. Unused (0) - Unused VLFs
6	Unique Identifier	Two part number (file_no:page_no)	3 part Log Sequence Number (LSN) - (In Hex) VLFSeqNo:OffsettoLogBlock:SlotNoInsideLockBlock
7	Link	It depends. i) Heaps - pages are not logically connected (IAM) ii) Clustered/Non Clustered Index - pages are linked (B-Tree)	In every transaction, LSN records has unique sequence numbers with one common transaction number.

T-Log Basics

Slno	Property	Data Files	Transaction Log Files
8	Purpose	**Purpose:** i) System pages (GAM, SGAM, PFS, BCM etc) ii) Data/Index pages - To store actual user data	**Multi Purpose:** 1. Crash recovery & Restore recovery 2. Point in time recovery (Log backup) 3. Replication 4. Log Shipping 5. Mirroring 6. Database snapshot creation 7. CDC and CT
9	In Memory	Buffer cache	Log Cache
10	Memory to Disk	IO to data files is asynchronous	IO to T-Log file is asynchronous

T-Log Basics

A	B	C	D
Slno	Property	Data Files	Transaction Log Files
11	from memory to disk	CHECKPOINT or Lazy-writer moves dirty pages to data files irrespective of whether chage is commited or in progress. Key: Reduce IO & Improve performance	Key: Record all DDL and DML changes in disk to fullfill ACID requirements.
12	Accuracy of data in disk	Not necessarily 100% accurate. Checkpoint or Lazywrite may even flush dirty pages impacted by un-commited transactions. See point 11 and 18	Purpose of T-Log file is to record all DDL and DML activities immediately.
13	Recovery phases	No recovery phase for data files (?)	**Recovery phases:** 1. Analysis - Dirty page table & Active transaction tables are created. 2. Redo - Committed changes in T-Log are applied in data files if not done. (if T-Log Prev. LSN = Data page LSN) 3. Undo - For Incomplete transactions, compensation log records are created

T-Log Basics

Slno	Property	Data Files	Transaction Log Files
14	Some frequently used commands/fn's	DBCC EXTENTINFO DBCC IND DBCC PAGE	CHECKPOINT DBCC LOGINFO DBCC LOG fn_dblog() fn_dump_dblog()
15	Behaviour change in Recovery Models	No major behaviour change?	i) Simple - Checkpoint truncates in-active VLFs ii) Bulk logged - Bulk logged operations are minimally logged and Only T-Log backup can truncate in-active VLFs iii) Full - Only T-Log backup can truncate in-active VLFs **Funny fact:** Database with Full recovery will act as Simple recovery until you initiate Log chain by taking first full backup.

T-Log Basics

Slno	Property	Data Files	Transaction Log Files
16	Shrink	It is not a good practice to shrink data files. This will lead to fragmentation and poor performance	We can shrink un-used VLF portions if required. *Read more on this
17	Friendliness	Two frequently accessed data files are not friends. If possible, keep them in seperate disk drives :-) Key: Performance	Data and Log files are not friends. If possible, Seperate them and keep them in seperate disk drives.
18	Delete	Deleted user records are marked as ghost records and not physically removed immediately. Ghost clean up will happen at frequent interval	T-Log records are not deleted. During Rollbacks compensation records are created to undo the changes made.
19	Documentation	Anatomy of data and index pages are documented	Anatomy un-documented. You may get some clues if you are good at page anatomy.

DBCC LOGINFO

Create LSN
0 if VLF is added by statement
LSN - if VLF created for LSN~

Log File Id

VLF Size – Initially total log size is best possible* equal blocks

Offset - First 8K page header.

FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
2	2555904	8192	25	2	64	0
2	2555904	2564096	0	0	0	0
2	2555904	5120000	0	0	0	0
2	2809856	7675904	0	0	0	0

VLF Sequence No.
Starts at 25.
0 if unused

Status
0 (Reusable or Unused)
2 (Active or Recoverable)

Parity
0 Unused~
64 Active~
128 Reused~

~ As observed

FN_DBLOG(lsn_begin, lsn_end)

```
SELECT
    [Previous LSN],         [Current LSN],      Operation,          CONTEXT,
    [Transaction ID],       AllocUnitName,      [Page ID],          [Slot ID]
    [Offset in Row],        [Begin Time],       [Transaction Name], [End Time],
    [Number of Locks],      [Lock Information], [RowLog Contents 0], [RowLog Contents 1],
    [RowLog Contents 2],    [RowLog Contents 3], [RowLog Contents 4], [Log Record],
    [Log Record Fixed Length], [Log Record Length]
FROM fn_dblog(null,null)
```

FN_DBLOG(lsn_begin, lsn_end) Operation

Source: http://www.sqlservercentral.com/blogs/hugo/2012/11/06/another-dive-into-transaction

ABORT_XACT Indicates that a transaction was aborted and rolled back.

BEGIN_CKPT A checkpoint has begun.

BEGIN_XACT Indicates the start of a transaction.

BUF_WRITE Writing to Buffer.

COMMIT_XACT Indicates that a transaction has committed.

CREATE_INDEX Creating an index.

DELETE_ROWS Rows were deleted from a table.

FN_DBLOG(lsn_begin, lsn_end) Operation

Source: http://www.sqlservercentral.com/blogs/hugo/2012/11/06/another-dive-into-transaction

DELETE_SPLIT	A page split has occurred. Rows have moved physically.
DELTA_SYSIND	SYSINDEXES table has been modified.
DROP_INDEX	Dropping an index.
END_CKPT	Checkpoint has finished.
EXPUNGE_ROWS	Row physically expunged from a page, now free for new rows.
FILE_HDR_MODIF	SQL Server has grown a database file.
FORGET_XACT	Shows that a 2-phase commit transaction was rolled back.

FN_DBLOG(lsn_begin, lsn_end) Operation

Source: http://www.sqlservercentral.com/blogs/hugo/2012/11/06/another-dive-into-transaction

FORMAT_PAGE Write a header of a newly allocated database page.

INSERT_ROWS Insert a row into a user or system table.

MARK_DDL Data Definition Language change - table schema was modified.

MARK_SAVEPOINT designate that an application has issued a 'SAVE TRANSACTION' command.

MODIFY_COLUMNS Designates that a row was modified as the result of an Update command.

MODIFY_HEADER A new data page created and has initialized The header of that page.

MODIFY_ROW Row modification as a result of an Update command.

FN_DBLOG(lsn_begin, lsn_end) Operation

Source: http://www.sqlservercentral.com/blogs/hugo/2012/11/06/another-dive-into-transaction

PREP_XACT Transaction is in a 2-phase commit protocol.

SET_BITS Designates that the DBMS modified space allocation bits as the result of allocating a new extent.

SET_FREE_SPACE Designates that a previously allocated extent returned to the free poo

SORT_BEGIN A sort begins with index creation.
SORT_END end of the sorting while creating an index.
SORT_EXTENT Sorting extents as part of building an index.

UNDO_DELETE_SPLIT The page split process has been dumped.

XACT_CKPT during the Checkpoint, open transactions were detected.

FN_DBLOG(lsn_begin, lsn_end)

Output 1/6

	Previous LSN	Current LSN	Transaction ID
1	00000000:00000000:0000	00000019:0000001a:0001	0000:0000025e
2	00000019:0000001a:0001	00000019:0000001a:0002	0000:0000025e
3	00000019:0000001a:0002	00000019:0000001a:0003	0000:0000025e
4	00000019:0000001a:0003	00000019:0000001b:0001	0000:0000025e
5	00000000:00000000:0000	00000019:0000001b:0002	0000:00000000
6	00000000:00000000:0000	00000019:0000001b:0003	0000:00000000
7	00000019:0000001b:0001	00000019:0000001b:0004	0000:0000025e
8	00000000:00000000:0000	00000019:0000001b:0005	0000:00000000

FN_DBLOG(lsn_begin, lsn_end)

Output 2/6

Operation	CONTEXT	AllocUnitName	Page ID
LOP_BEGIN_XACT	LCX_NULL	NULL	NULL
LOP_MODIFY_ROW	LCX_BOOT_PAGE	Unknown Alloc Unit	0001:00000009
LOP_MODIFY_ROW	LCX_BOOT_PAGE	Unknown Alloc Unit	0001:00000009
LOP_DELETE_ROWS	LCX_MARK_AS_GHOST	sys.sysprufiles.clst	0001:0000001d
LOP_MODIFY_HEADER	LCX_PFS	Unknown Alloc Unit	0001:00000001
LOP_SET_BITS	LCX_PFS	sys.sysprufiles.clst	0001:00000001
LOP_INSERT_ROWS	LCX_CLUSTERED	sys.sysprufiles.clst	0001:0000001d
LOP_SET_BITS	LCX_PFS	sys.sysprufiles.clst	0001:00000001
LOP_DELETE_ROWS	LCX_MARK_AS_GHOST	sys.sysprufiles.clst	0001:0000001d

FN_DBLOG(lsn_begin, lsn_end)

Output 3/6

Offset in Row	Begin Time	Transaction Name	End Time
NULL	NULL	NULL	2012/11/16 22:31:49
NULL	2012/11/16 22:31:59:203	INSERT	NULL
NULL	2012/11/16 22:31:59:203	Allocate Root	NULL
NULL	2012/11/16 22:31:59:213	FirstPage Alloc	NULL
0	NULL	NULL	NULL
0	NULL	NULL	NULL

FN_DBLOG(lsn_begin, lsn_end)

Output 4/6

Number of Locks	Lock Information	RowLog Contents 0
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
3	HoBt 72057594038779904:ACQUIRE_L...	0x30000800E9030000020000100130041414141
NULL	NULL	NULL

FN_DBLOG(lsn_begin, lsn_end)

Output 5/6

RowLog Contents 0	RowLog Contents 1	RowLog Contents 2	RowLog Contents 3	RowLog Conte
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
0x30000800E903000...	0x	0x0101000C0000E7A4787...	0x	0x
NULL	NULL	NULL	NULL	NULL

FN_DBLOG(lsn_begin, lsn_end)

Output 6/6

Log Record	Log Record Fixed Length	Log Record Length
0x00006000190000003D0000000C00020061020000000...	96	96
0x00006000190000003D0000000D00020061020000000...	96	96
0x00005000190000003D0000000E00020061020000000...	80	84
0x00006000190000003D0000000F00020061020000000...	96	96
0x00003000190000003D000000020002006102000000...	48	52
0x00003E00190000003D000000010002006002000000...	62	120
0x00003000190000003D000000010002006002000000...	48	52

Reference

Paul Randal & Kalen Delaney