International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 24189-24200

http://www.ripublication.com

Analyzing Integral Components of SQL Server Databases

Dr. Muthukumar Murugesan[1], Dr.K.Karthikeyan[2], Dr.K.Sivakumar[3]
[1] *Delivery Project Lead, Mphasis Limited, Bangalore, Karnataka, India, amgmuthu@yahoo.com*
[2]*Assistant Professor, Departmentof Computer Application in Anna University, Regional Office, Madurai, Tamilnadu, India., adithyakarthi@gmail.com*
[3]*Professor-CSE, Hindusthan Institute of Technology, Coimbatore, Tamilnadu, India, rksivakumar@gmail.com*

Abstract

A database is the integral part of all data work flow and it is a defined space stored in the system, which helps in grouping user objects. This space can be split into several files organized in groups.SQL Server has a limit of 32,767 databases per instance and each database can store more than 2 billion objects, by reaching a physical size up to 524,272 terabytes. Database file is acts as a normal operating system file. Each database must have minimum one data file and one log file.Understanding the internal architecture of this file structure, how integral parts are performing,is most important activity. This helps database architects and administrators to get more clarity and will give positive drive to design, manage and handle the database in correct manner. If database is not properly designed or maintained,it will not attain better performance and have to face the challenges almost everywhere starting from data retrieval to generating reports. This paper emphasizes the physical structure of a SQL server integral files and how they are structured internally.

Keywords: SQL server architecture, file system, data files, log files, file anatomy, database structure

Introduction

Database design is the part of the database development process that involves analysis of a problem definition (specifications and requirements) and provides all necessary findings for building a logical structure of data. The problem definition specifies more or less formally the purpose, needs, requirements and constraints for data expected to support some organizational operations.The logical structure of data may initially be expressed in a plain language and later transformed into a meaningful data model. Each and every database is used as individual file to store data and log information.

These information are never mixed in the same file. Generally, SQL server has two type of files such as “Data Files and Log Files”.The data files are categorized into two types: Primary Data File and Secondary Data File(s).

Primary data file is the starting point of the database and has the reference to the other files in the database. User data and objects can be stored either in this file or in secondary data files.Each database should have only one primary data file. By convention, the Primary Data File has the .mdf extension.

Secondary data files make up all the data files, other than the primary data file. This is optional and some databases may not have any secondary data files, while others have several secondary data files. The recommended file name extension for secondary data files is .ndf.

Log files,each database must have at least one log file and extension of this log file as .ldf.The transaction log files hold the log information and that is used to recover the database.

SQL Server does not enforce the file extension of .mdf, .ndf, and .ldf, but these extensions help to identify the different kinds of files and their use.Data will be stored in SQL server special structure called data pages. Each data page will have 8192 bytes size (8kb) and 128 data pages can be stored in 1 MB size. Rows cannot span pages and however, portions of the row may be moved off the row's page so that the row can actually be very large. INSERT or UPDATE operations can increase the total size of the row beyond 8060 byte limit. When the total row size of all fixed and variable columns in a table exceeds 8060 byte limitation, SQL Server dynamically moves one or more variable length columns to pages to the ROW_OVERFLOW_DATA allocation unit, starting with the column with the largest width.SQL server has some set of default page setup as below:

Table 1: Default Page Setup

Page Description	Usage
Page Size	8KB
Max. Bytes Stored	8192 Bytes
Bytes Used for Page Header (Fixed)	96 Bytes
Maximum Bytes Used for User’s Data	8053 Bytes + 7 Bytes row overhead
Minimum Bytes Used for Row Offset	36 Bytes
Default Bytes Used for Each Offset Element	2 Bytes
Default Number of Row Offset	18
Default Total Number of Records per Page	18
Default Bytes per Record/Row	447.38 Bytes (8053/18)

Database Physical Structure

Each instance of SQL Server has four system databases (master, model, tempdb, and msdb) and one or more user databases.Single instance of the SQL Server is capable of handling thousands of users working in multiple databases at the same time.

Each instance of SQL Server makes all databases in the instance available to all users and that connects to the instance. SQL Server has two files as “Logical and Physical”.

Logical file is the namereferred to the physical file in all Transact-SQL statements. The logical file must comply with the rules of SQL Server identifiers and must be unique among logical file names in the database.

Physical file should follow the rules of the operating system file names. The physical file includes the directory path.

In SQL Server, data and log files can be put on either FAT or NTFS file systems. However, NTFS file system has more security. SQL Server files can grow automatically from their original specified size. When a file is defined a specific growth increment can be specified. Every time,when the file is filled, it increases its size default by the growth increment. If there are multiple files in a file group, they will not auto grow until all the files are full. Each file can also have a maximum size specified. If a maximum size is not specified, the file can continue to grow until it has used all available space on the disk. The user can let the files auto grow as required to reduce the administrative burden of monitoring free space in the database and manually allocate additional space. In SQL Server, the locations of all the files in a database are recorded in the primary file of the database and in the master database. Mostly, the SQL Server Database Engine uses the file location information from themaster database most of the time.

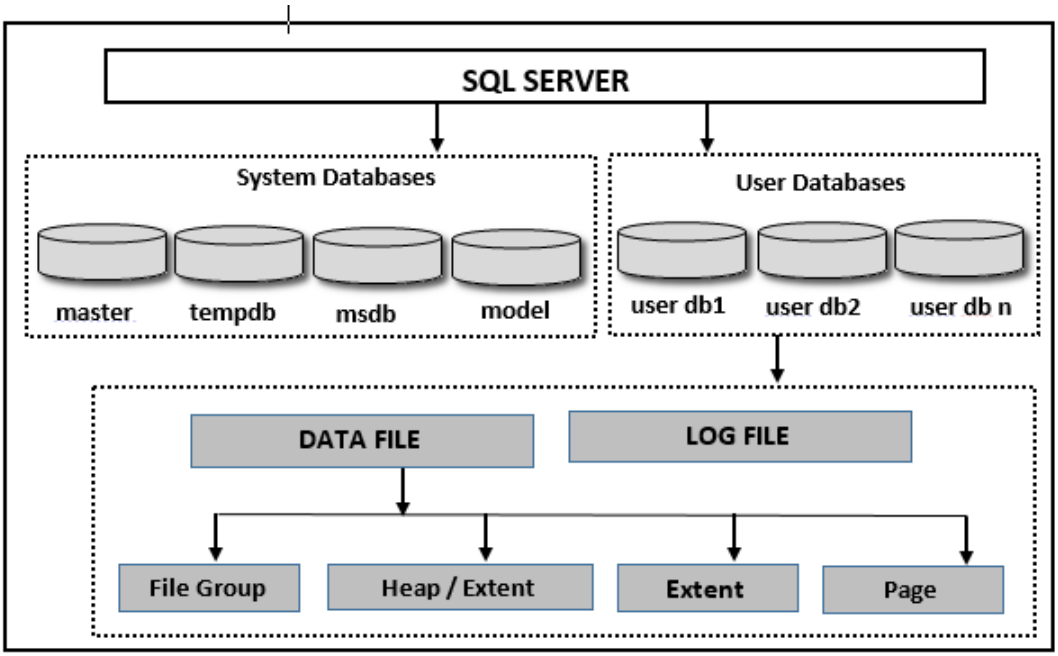


Figure 1: SQL Server Database Architecture

Important Elements of Data File

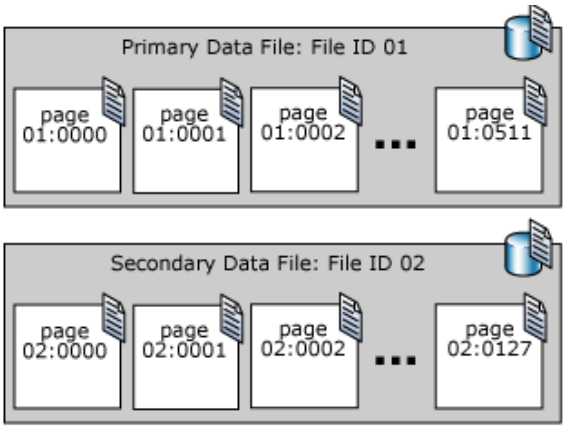


Figure 2: Architecture Diagram - Data File

Page

The page is fundamental unit of data storage and internally, SQL server organizes and stores data in smaller units known as pages. The SQL server uses different types of pages like DATA, GAM, SGAM and etc. to store different types of data like data, index data, text, sort, BLOB and etc. SQL server usesabout 14 different types of pages in data files andshould specify the narrowest columns as much as possible to maximize the number of rows that can fit into one data page.

The disk space allocated to a data file (.mdf or .ndf) in a database is logically divided into pages numbered contiguously from 0 to n. Disk I/O operations are performed at the page level.Data rows are put on the page serially, starting immediately after the header. All the pages in SQL Server have the same structure.A data page has three sections as "Page header, Actual data and Row offset array". A schematic diagram of data pages looks like as below:

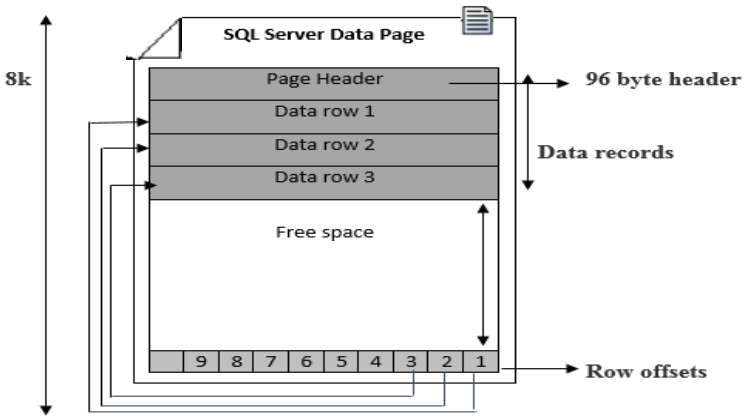


Figure 3: Data Pages - Schematic Diagram

Page Header

Each page begins with a fixed size of 96 bytes header that is used to store system information about the page. This information includes the page number, page type, the amount of free space on the page, and the allocation unit ID of the object that owns the page. The size of page header will be same for all pages. DBCC command will give the output with 4 different sections about the database. The first section is BUFFER, which gives the details about the memory allocation. Next section gives page header information. Third section gives the details about slots where the actual user's data to be stored. Last section of page provides details about row offset table. SQL statement DBCC PAGE will give header information.

For experimental purpose, database called "Data Pages" have been used. From the figure4it can be made sure that the number of free bytes (here 4292 bytes) on the page uses variable **m_freeCnt**. This means, the data up to 4292 bytes can be stored in same page. Ifmore bytes of data are stored, SQL server will automaticallycreate the new page to accommodate the new data.

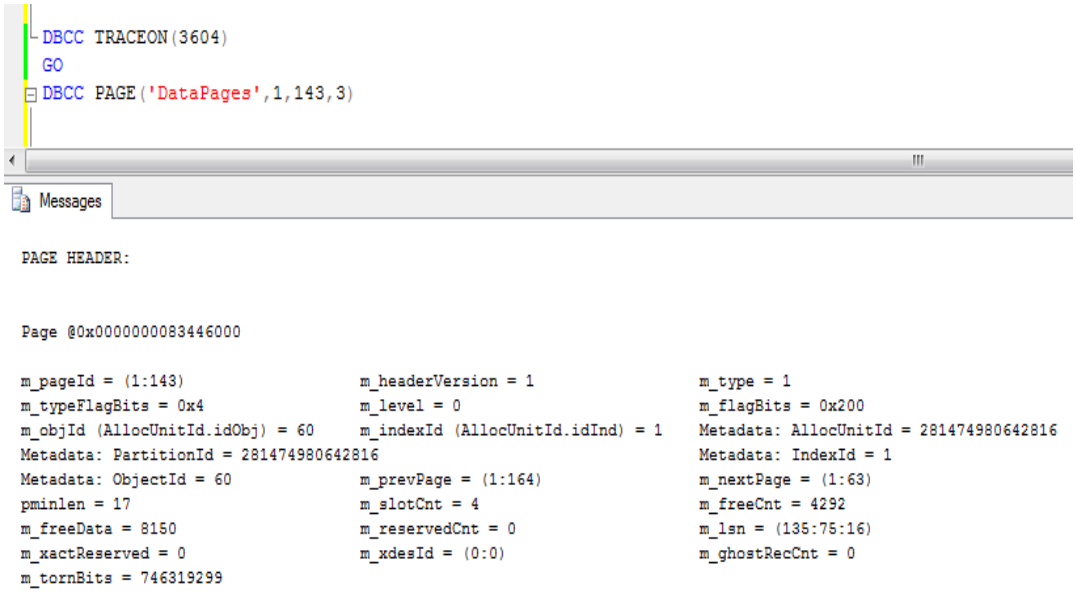


Figure 4: Page Row Header

User's Data

The maximum amount of 8060 bytes users' data can be stored in each page which includes the 7 bytes row overhead. So in a single record you can store maximum of 8053 bytes users' can bestored. If it is tried to create the table with exceeding the size of 8053 bytes, the SQL server will not permit to create and it will give the error message. The below figure5 shows this restriction clearly and throws the error message.

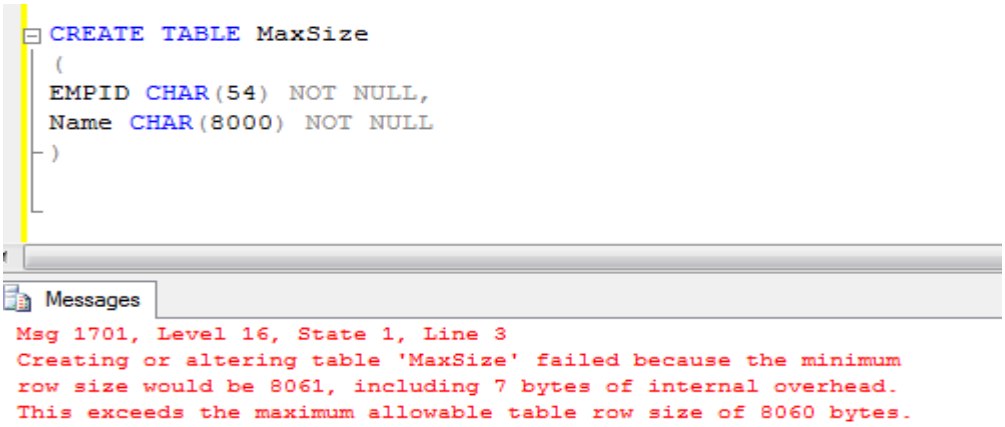


Figure 5: Data Pages - Schematic Diagram

Row Offset

Remaining 36 bytes from 8192 bytes are used for row offset. A row offset table starts at the end of the page, and each row offset table contains one entry for each row on the page. The entries in the row offset table are in reverse sequence from the sequence of the rows on the page. Count of records (size of array) is saved in the header. The size of each element in the offset array is 2 bytes.Slot array can grow from bottom to top based on the size of the records. If the size of records is small, more records can be accommodated in a page and offset table will take more space from bottom to top. Each slot in the record offset array points to a byte index in the page where a record begins.

Extent

The space allocation in SQL Server is managed in portions called “extents”, which are basically a group of eight physically contiguous pages. Hence, each extent will have 64 KB (8*8 KB/Page)size.Extents are used to efficiently manage the pages and make the allocation system more efficient. All the information about the extents are tracked in GAM, SGAM, and IAM pages.

Generally, extents are two types such as Uniform Extent and Mixed Extent”. Uniform Extent consists of 8 pages in the extent and used by a single object. Mixed Extentconsists of 8 pages in the extent may be used by different objects.

A new table of index generally allocates pages from mixed extents. When the table or index grows to the point that it has eight pages, it then switches to use uniform extents for subsequent allocations. While creating an index on an existing table that has enough rows to generate eight pages in the index, index is allocated in uniform extents.

Heap and Index

A heap is a table without a clustered index. One or more nonclustered indexes can be created on tables and stored as a heap. Data are stored in the heap without specifying an order.If a table is a heap and does not have any nonclustered indexes, then the

entire table must be examined (a table scan) to find any row. A clustered table providesfew benefits over a heap such as physically storing the data based on the clustered index, the ability to use the index to find the rows quickly and the ability to reorganize the data by rebuilding the clustered index. Depending on the INSERT, UPDATE and DELETE activity against the tables,the physical data can become very fragmented.When a table is stored as a heap, individual rows are identified by reference to a Row Identifier (RID) consisting of file number, data page number, and slot on the page. The row id is a small and efficient structure. Sometimes data architects use heaps, when data are always accessed through nonclustered indexes and the RID is smaller than a clustered index key.

File Group

Collections of files are called as Filegroups and are used to helpdata placement and administrative tasks such as backup and restore operations.Database objects and files can be grouped together in file groups for allocation and administration purposes. There are two types of file groups **Primary and User defined**. No file can be a member of more than one file group. One file group in each database is designated as the default file group. If no default file group is specified, the primary file group is the default file group. SQL Server default file extensions .mdf, .ndf, and .ldf are helpedto identify different kinds of files and their use. In SQL Server, the locations of all the files in a database are recorded in the primary file of the database, called master database. The SQL Server Database Engine uses the file location information from the master database by default.

Transaction Log File

A log file is the initial repository of all data contained in the data base. That is, when data are first passed from the user to the data base, they are written to the log file.After the transaction is marked as completed, the data are then eligible to be written to the data file.Every SQL Server database has a transaction log that records all transactions and the database modifications made by each transaction. The transaction log is a critical component of the database and, if there is a system failure, the transaction log will be required to bring the database back to a consistent state. The transaction log should never be deleted or moved, unless the ramifications are fully understood.Log files do not contain pages but they contain a series of log records. Log files are never part of a filegroup and log space is managed separately from data space.Logically, transaction log is a set of log records. Each record is identified by a Log Sequence Number (LSN). The new log record is always written at the logical end of log file with a LSN and which is greater than the previous one.

It holds all the log information and which are used to recover the database. There must be at least one log file for each database, although there can be maximum 16 log files. The recommended file name extension for log files is .ldf.The data written to log files are written in a serial manner.Data will be written first to the log, and then to the data file. If logs are placed on the same partition as the data files, the same write-head which writes the log must also write to the data partition. If logs are placed on their

own partition, logs and data can be written in parallel. The data written to a data file is usually much slower, as the drive head has to seek and find the location within each logical table where to put the data.When a data base is backed up, the transaction log becomes part of the backup file. It will not be a separate backup file. However, new data added during the backup isn't included in the backup file being generated.

Internal Structure of Virtual Log File

SQL server uses the transaction log in sequential manner. As like the data file divided into pages, log files are also divided into Virtual Log File (VLF). The size of the VLFs in a log file may not be in equal size. SQL server decides the size and number of VLF in a log file based on the size of the log file growth as given below.

Table 2: Log File Allocation

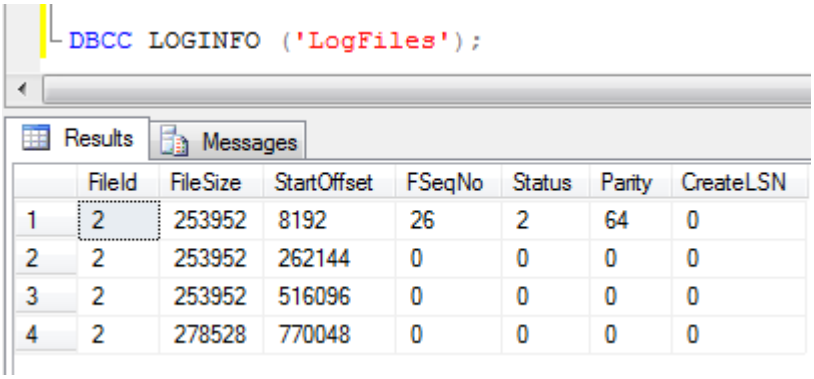
Log File Growth Size	Number of VLF
Growth up to 64 MB	4
From 64 MB to 1 GB	8
Larger than 1 GB	16

Let us create a database with 1 MB initial log size and later, increase it to 65 GB.

```
USEMASTER;
GO
CREATEDATABASE LogFiles
ON
 ( NAME = LogFiles,FILENAME='D:\MyDb\LogFiles.mdf',
 SIZE = 3MB, MAXSIZE = 3072MB, FILEGROWTH = 5MB
)

LOGON ( NAME = LOGFiles_log,FILENAME='D:\MyDb\LOGFiles_log.ldf',
SIZE = 1MB, MAXSIZE = 2048MB, FILEGROWTH = 5MB );
GO
```

According to above SQL script, new database has been created with 1 MB initial log size. As per SQL server specification, only four VLF will be created till 64 MB size. Using DBCC Log info statement will find out number of VLF created for the database.



```
DBCC LOGINFO ('LogFiles');
```

Results | Messages

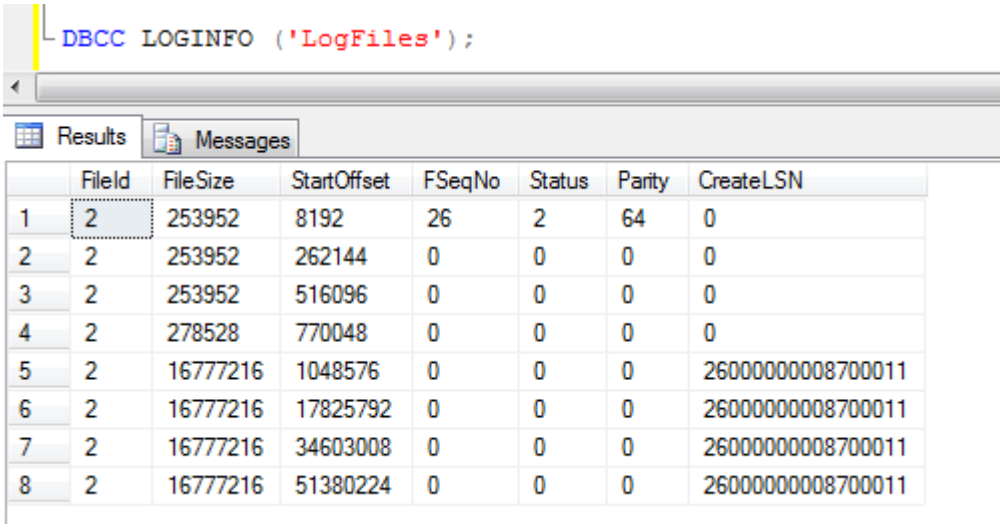
	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	2	253952	8192	26	2	64	0
2	2	253952	262144	0	0	0	0
3	2	253952	516096	0	0	0	0
4	2	278528	770048	0	0	0	0

Figure 6: VLF Details for 1 MB Size

Now, use the below commands to increase the LOG file size from 1 MB to 65 MB.

```
ALTERDATABASE LogFiles
MODIFYFILE ( NAME = LOGFiles_log,
FILENAME='D:\MyDb\LOGFiles_log.ldf',SIZE = 65MB)
```

Use DBCC Loginfo statement to find out number of VLF created for the database.



```
DBCC LOGINFO ('LogFiles');
```

Results | Messages

	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	2	253952	8192	26	2	64	0
2	2	253952	262144	0	0	0	0
3	2	253952	516096	0	0	0	0
4	2	278528	770048	0	0	0	0
5	2	16777216	1048576	0	0	0	26000000008700011
6	2	16777216	17825792	0	0	0	26000000008700011
7	2	16777216	34603008	0	0	0	26000000008700011
8	2	16777216	51380224	0	0	0	26000000008700011

Figure 7: VLF Details for 65 MB Size

As shown in the above figure7, the above records represent a VLF specification for current database. From the above figure7, fields are classified as:

File Id: This is the unique file id of the log file and will be same for all records, as there is only one log file. If we have multiple log file then this FileID will have multiple numbers.

File Size: This is representing the size of the VLF. The first four are seen, all have same size except the fourth one. This is because the first 8KB of the log file is used for file header.

As shown in figure 7, If filesize value of first four records is added along with 8192(8KB),1 MB will be obtained, which is the initial size of the log file what has been specified during the database creation.

253952 + 253952 + 253952 + 278528 = 1040384 + 8192 = 1048576 bytes

1048576 / 1024 = 1024 KB = 1 MB

As shown in figure 7, If filesize value of first eight records is added along with 8192(8KB), 65 MB will be received. The growth is happened because of ALTER statement.

68149248 + 8192 = 68157440

68157440 / 1024 = 66560 KB = 65 MB

Start Off Set: The first VLF always starts from 8192 bytes. This means the first 8KB (8192 bytes) will be used for file header.

FSeq No: The file sequence number indicates the order of usage of the VLFs. The row with the highest FSeqNo value is the VLF, where current log records are being written. FSeqNo values are not consistent. It will keep changing each time, when VLFs are getting reused.

Status: Status has two possible values : 0 and 2. A value 2 means, the VLF is not reusable and value 0 means, it can be reused.

Parity: Parity has three possible values 0,64 and 128. If the VLF is not used yet, it will have a value 0 and will be set to 64 on first use.Every time, a VLF is reused and the parity value is switched between 64 and 128.

Create LSN: The value indicates when the VLF is created or group the VLF based on the creation. A values 0 indicates, those VLFs are created as part of database creation.

In thecase as shown in figure 8, first four records have a value 0 which indicates that these VLFs are created as part of database creation with 1 MB log size. The remaining 4 records have the same value. These VLFs are created as part of alter database statement to increase the size of the log file from 1 MB to 65 MB.

Now,the internal structure of log file has 8 transaction log files and structure of the transaction log look like as below:

8KB Page Header	**1 MB Log of initial DB creation**				**65 MB Log as part of ALTER Database**			
Header	**VLF1**	**VLF2**	**VLF3**	**VLF4**	**VLF5**	**VLF6**	**VLF7**	**VLF8**

Figure 8: Log File Internal Structure

Importance of Log File

When particular data are queried by the user, SQL Server reads the required data pages from the disk into memory containing the requested data from the data file. In case SQL Server needs to make any modification in the existing data, it reads the

required data pages into the buffer cache; updates those cached data pages in memory; writes modifications to the log file, when the transaction is committed, and then writes the updated data pages back to the disk, when the checkpoint operation is performed. In-memory modified data, pages are called dirty pages. When a checkpoint is performed, it permanently writes these dirty pages on disk.

A log is a physical file in which SQL server stores the details of all transactions and data modifications performed on the database.The log file is used to record any change that is made to the database. It's intended for recovery of the database in case of disaster or failure. Because a log file is intended to record the changes, it is designed to be written and accessed in a sequential manner.

DML operations (INSERT, DELET and UPDATE) will generate transaction logs. DDL operations, such as ALTER TABLE, and CREATE PROCEDURE which cause changes to the system tables, will also generate transaction logs. Transaction logs are generated by each transactions. Any data modification to persistent data in the data files will be at least implicitly involved in a transaction.Transaction log will store separate log entries for each operation.For example, while inserting a record into a table, transaction log will store separate log entry for inserting into clustered index and other non clustered index. In the same way, if a single update statement is updating 10 records, transaction log will capture 10 separate log entries. Each and every transaction may store multiple records in log files.

SQL Server is designed to handle and process multiple I/O requests simultaneously; if we have enough hardware resources are available. Even if SQL Server is capable of handling simultaneous I/O requests in parallel.It may face the issue of disk contention, while reading large amounts of data from data files and writing large a number of transaction logs to log files in parallel with two different requests, if data files and log files reside on the same physical disk. However, if data file and log file are located on separate physical disks, SQL Server gracefully handles and processes such requests in parallel.Generally, placing data files and log files on separate physical drives greatly improves the performance of the database.

Conclusion

This paper has taken a look back at the essential parts of database to establish their importance.SQL server is a ubiquitous and critical component of modern database system.The techniques analyzed in this paper can augment traditional forensics skills. This paper presents the architectural discussion, several internal components of data files and log files and significance of those components. Understanding of these integral components will help the developers and administrators to design the database in a proper manner and will help them to improve the performance.

References

[1] Dmitri Korotkevitch., June 2014, “Pro SQL Server Internals,” Apress.

[2] Christian Bolton., Rob Farley., Glenn Berry., Justin Langford., and Gavin Payne., Nov 2012, "Professional SQL Server 2012 Internals and Troubleshooting Paperback," Wrox.

[3] XueLian Feng., and HaiYan Liu., April, 2013 "Design of the Database of Library Information," International Journal of Database Theory and Application, Vol. 6, No. 2.

[4] Kristi L. Berg., Tom Seymour and RichaGoel., 2013., "History of Databases,"International Journal of Management & Information Systems.

[5] Joseph M. Hellerstein., Michael Stonebraker., and James Hamilton., 2007 "Architecture of Database System," Foundation and Trends in Databases, Volume1 Issue 2, ISSN 1931-7889.

[6] Kalen Delaney., 2007, "Inside SQL Server 2005 the Storage Engine," Syngress Publishing.

[7] Date, C. J., 2004, "An Introduction to Database Systems,"Fifth Edition, Boston, Addison Wesley.

[8] Kalen Delaney., 2007, "Inside SQL Server 2008 the Storage Engine," Microsoft Press.

[9] Kevvie Fowler., 2007, "Forensic Analysis of a SQL Server 2008 Database Server," SANS Institute InfoSec Reading Room.

[10] Arnold, S., 2012, "The Future of Database. Retrieved" fromhttp://arnoldit.com/wordpress/2008/10/21/ the-future-of-database/.

[11] KroenkeD., and Auer, D., 2007,"Database Concepts.,"3rd ed. New York, NY: Prentice.

[12] Microsoft SQL Server Books Online, https://technet.microsoft.com/en-us/library/ms190969(v=sql.105).aspx