在上一篇文章【数据库事务日志碎片原理分析与方案】-分析篇 中，已经普及了一些与日志文件相关的内容，下面，我们就进一步的理解。

日志增长与 VLF 文件的个数

通过上面的相关内容的介绍，我们已经知道了日志文件自动的增长会到了一些问题，而事实确实如此，下面，我们就来更加清楚的看看这些问题。

很显然，我们不希望日志文件任意的增长，我们更加希望这个增长是受我们控制的。我们先看看自动增长的一些问题：

1.**导致过多的 VLF**。因为自动增长会在需要的时候去增加日志文件所在磁盘空间的大小，而且每次分配的空间又会被分成多个 VLF，如果每次增长的磁盘空间不大，而数据库的操作（指的是那些写日志的操作）又非常的频繁，最后就结果就是数据库的日志文件一点点的增长，从而导致磁盘的文件碎片和数据库内部日志文件碎片，这样就会极大的降低了读取日志的速度。另外，大家应该还记得之前我们说过：数据库的日志是循环写入的，如果日志文件内部碎片，那么在重新将日志写入日志文件的时候，就需要去需找文件的位置，会使得相对应的数据库操作更加的慢。

2.**每次日志文件的增长会消耗 CPU 和 I/O**：日志自动增长不会使用实例文件初始化，因此，SQL Server 需要去每次都去经历日志文件进行初始化的全过程，而这个过程是非常消耗资源的。这一点，大家只要知道有这么回事就行了，不用深究。

另外，如果我们在建立数据库的时候，采用了默认的配置，那么自动增长是被默认开启的，而且默认的数据文件和日志文件的大小都很小。如果我们就这样使用数据库，最后会发现：日志文件中包含成百上千个 VLF。

每次在新建一个数据库的时候，默认的大小以及自动增长的设置都是从 model 这个系统数据库中继承而来的。一般而言，日志文件的初始大小 1MB，自动增长为 10%。

这个时候，可能有朋友就要问了：**这个 1MB 日志中包含多少个 VLF，或者说日志文件的大小与 VLF 的个数之间是什么关系**？

我们下面的一个表反映了日志文件的大小和 VLF 个数之间的关系。

Number of VLFs	Growth Size
4	<64MB
8	>=64MB and < 1GB
16	>=1GB

很显然，在默认的情况下，数据库的日志文件将会包含 4 个 VLF，因为默认的时候大小是 1MB<64MB 的。

所以，这样有一个建议：除非你需要用很多的小的数据库，否则，就调整 Model 数据库的一些设置，使得以后你新建的数据库从继承你设定的设置数据。

下面，我们就通过例子来看看一些问题。

我们下面的代码，创建了一个新的数据库，采用的默认的设置。同时，也将恢复模型设置为 Full，如下所示：

```
-- TEST #1 ----------------------------------------
USE master;
GO
CREATE DATABASE VLFTest1 ON
   (NAME = VLFTest1,
    FILENAME = 'C:\SQLData\VLFTest1_data.mdf',
    SIZE = 3MB,
    FILEGROWTH = 1) LOG ON
   (NAME = VLFTest1_log,
    FILENAME = 'D:\SQLData\VLFTest1_log.ldf',
    SIZE = 1MB,
    FILEGROWTH = 10%);
GO

ALTER DATABASE VLFTest1 SET RECOVERY FULL;
GO

BACKUP DATABASE VLFTest1
TO DISK = 'E:\SQLBackups\VLFTest1_Full.bak';
GO
```


在下面的一段代码中，我们使用 **DBCC LOGINFO** 来查看日志文件的 VLF 的个数，如下：

```
USE VLFTest1
DBCC LOGINFO () WITH NO_INFOMSGS;
GO
/*OUTPUT:
FileId      FileSize           StartOffset       FSeqNo      Status      Parity CreateLSN
----------- ------------------ ----------------- ----------- ----------- ------ -
2           253952               8192              26          2           64     0
2           253952               262144            0           0           0      0
2           253952               516096            0           0           0      0
2           278528               770048            0           0           0      0
*/
```


现在，我们再看到下面的代码，在代码中创建了一个新表 AgileSharpTable，并且开启事务，然后插入 10000 行数据：

```
SET NOCOUNT ON;
USE VLFTest1;
GO
CREATE TABLE MyTable
    (
      Id INT IDENTITY ,
      MyDesc CHAR(8000)
    );
DECLARE @I INT = 0
DECLARE @S DATETIME = GETDATE();
BEGIN TRANSACTION;
WHILE @I < 100000
    BEGIN
        SET @I += 1;
        INSERT  INTO MyTable
        VALUES  ( REPLICATE('ABCD', 2000) );
    END
COMMIT TRANSACTION;
SELECT  'Time to run Test #1: '
        + CAST(DATEDIFF(SS, @S, GETDATE()) AS VARCHAR(4)) + 'Seconds';
```


此时，这个测试花了将近 50 秒的时间，此时，我们在查看 VLF 的信息，此时发现已经有了 251 个。

下面，我们在新建一个数据库，此时我们自己设置日志文件的大小和增长情况，如下：

```
-- TEST #2 ----------------------------------------
USE master;
GO
CREATE DATABASE VLFTest2 ON
   (NAME = VLFTest2,
    FILENAME = 'C:\SQLData\VLFTest2_data.mdf',
    SIZE = 3MB,
    FILEGROWTH = 1) LOG ON
   (NAME = VLFTest2_log,
    FILENAME = 'D:\SQLData\VLFTest2_log.ldf',
    SIZE = 1000MB,
    FILEGROWTH = 250MB);

GO

ALTER DATABASE VLFTest2 SET RECOVERY FULL;
GO

BACKUP DATABASE VLFTest2
TO DISK = 'E:\SQLBackups\VLFTest2_Full.bak';
GO
```


此时，我们还是运行 DBCC LOGINFO，此时发现里面包含了 8 个 VLF，因为我们的日志文件的初始大小>=64MB 并且<1GB。

像第一个例子，我们此时在新的数据库中创建新的表，并且插入 10000 条数据。然后我们在运行 DBCC LOGINFO，此时发现，它的 VLF 还是 8 个。而且插入的数据也快了很多，只有 27 秒，这个主要是因为日志文件初始化的过程变少，提高了性能。

到现在为止，我们做了一些简单的演示，也看出一些性能的问题，下面我们就进一步的看看 VLF，那么在此之前，我们先具体的看看刚刚使用的 DBCC LOGINFO 命令。

深入 DBCC LOGINFO

通过刚刚的使用，我们已经发现这个命令相当的有用了，通过它我们可以找到日志文件中的 VLF 的个数。另外，它还可以告诉我们更多的信息。为了更加的清楚，我们删除之前的那个 VLFTest1 数据库，然后重新建立它。

然后运行 DBCC LOGINFO，得到如下的结果：

FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
2	253952	8192	30	2	64	0
2	253952	262144	0	0	0	0
2	253952	516096	0	0	0	0
2	278528	770048	0	0	0	0

列名的解释如下：

FileID：在 sysfiles 中对用的 FileID，可以通过它表明日志文件存在的位置。

FileSize：VLF 的大小，单位是字节。

StartOffSet：VLF 文件在日志文件中起始的字节数。

Status：表明 VLF 是否包含活动的日志记录。2 表示这个 VLF 是活动的，不能被覆盖写入。

Parity：它的之范围是 0,64 或者 128.

CreateLSN:VLF 最先创建的 LSN。如果两个 VLF 文件有相同的这个数字，就表明这两个 VLF 是同时创建的，并且通过自动增长

简单的介绍之后，我们再来运行下面的一段代码：

```
SET NOCOUNT ON;
USE VLFTest1;
GO
CREATE TABLE MyTable
    (
      Id INT IDENTITY ,
      MyDesc CHAR(8000)
    );
DECLARE @I INT = 0
WHILE @I < 100000
    BEGIN
        SET @I += 1;
        INSERT  INTO MyTable
        VALUES  ( REPLICATE('A', 8000) );
    END
```


运行 DBCC LOGINFO,结果如下：

FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
2	253952	8192	30	0	64	0
2	253952	262144	31	0	64	0
2	253952	516096	32	0	64	0
2	278528	770048	33	0	64	0
2	262144	1048576	34	0	64	32000000037600016
2	262144	1310720	35	0	64	33000000025500044
2	262144	1572864	36	0	64	34000000013600036
2	262144	1835008	37	0	64	35000000006700028
2	262144	2097152	38	0	64	35000000048700008
2	262144	2359296	39	0	64	36000000037600052
2	262144	2621440	40	0	64	37000000029600022
2	327680	2883584	41	0	64	38000000025600016
2	327680	3211264	42	2	64	39000000025500044
2	393216	3538944	0	0	0	40000000025600060
2	393216	3932160	0	0	0	41000000049500016

从中可以看到，此时我们有一个活动的 VLF，而且还新建了 11 个 VLF。

控制 VLF

到这里，已经讲了很多与 VLF 相关的内容了，相信大家都应该心里不再害怕这些内容了。其实日志文件碎片产生最主要的原因就是日志文件的增长次数过多，特别是如果每次只是增长一点点的时候。

所以，当我们在创建数据库的时候需要设置合适的文件的大小，使得文件的大小起码可以应付一段时间的增长。同时，也不要一下子就去创建一个很大的日志文件，因为里面可能只包含很少的 VLF，最后却发挥不了太大作用，反而导致磁盘空间不足的错误发生。

很多的朋友都问我有没有一些好的数据或者法则来判断导致应该创建多少个 VLF，这里的回答是没有，如果非得要答案，只有一个：视情况而定。如果我以前说这话，可能要被人批死，但是如果做过这方面工作的朋友，应该清楚：这才是真理。没有万能的钥匙。这主要取决于数据库的读写的比例，还有就是日志文件备份的方式。

另外，也不是说：自动增长就是罪恶的，只要用的合适，依然不错，而且它还是保护措施，特别是当我们把日志的增长速度估算错误的时候，自动增长可以来帮助我们弥补这个问题。例如，假设我们把日志大小设置为 40GB，并且禁用了自动增长，但是数据库的读写速度不是我们当初想的那样，我们当初以为 40GB 可以抗住 3 个月的数据库日志的写入，但是结果却不是，如果我们禁用自动增长，结果可能导致数据库无法写入。

一般是通过一些数据进行分析了，这里我给朋友们一些可以参考的数据吧，但是不能盲目的套用，但是思路是这样的。我一般会把日志文件的大小设置和所在的磁盘大小相差 90%，并且开启自动增长。而且这个大小的设置也不是一下子就设置上去的，例如，磁盘空间是 100GB，我不会一下子就设置 90GB 的日志文件，因为那样，里面包含的 VLF 太少，会导致严重的性能问题。所以要一步步的设置，例如，开始设置日志大小为 1GB，然后在将其增加到 2GB（也就是每次增加 1GB，也有人推荐 512MB），一次类推，一直最后到 90GB。当然，每一次增加的大小可以调整，需要自己去测一下。

修复日志碎片

到这里，对日志文件剩下的话题就是整理它的碎片了。

如果我们发现数据库中包含了非常多的 VLF，此时我们就需要去整理。在这，朋友又要问了：**你说的“非常多的碎片”，到底“非常多”是多少**？说实在的，我还真不好说了，但是可以这样思考：如果是你设置日志文件的大小和 VLF，你会根据我们之前告诉大家的方法来设置 VLF 的个数（例如，每次增加 1GB 的日志文件，增加 8 个 VLF），得到你设置的 VLF 的个数，然后比较现在的 VLF 个数，如果两个数据之间相差的十几倍以上，那么就可能存在 VLF 问了，就需要整理碎片了。

整理的方法简单，就是运行 DBCC SHRINKFILE 命令，如下：

```
DBCC SHRINKFILE(<transaction log logical file name>,TRUNCATEONLY);
```

然后再想之前那么，把日志文件的大小设置回去。

好了，到这里，讨论就完了，有机会，我们在讨论这个话题。

文件推荐：

大话扩展事件第一篇：概述（上)

大话扩展事件第一篇：概述（下)

大话扩展事件第二篇：查询扩展事件的元数据信息（上）

SQL Server 十大最佳存储实践

针对数据库开发人员的性能调优小提示