SQL Server 数据库压缩技巧指南

SQL Server 数据库压缩技巧指南

我经常看见人们制定定期缩小数据库文件（数据或 T-Log）的 SQL Server 维护计划和定制任务。我通常也会建议我的客户和同事除非他们要研究数据库增长原因，否则不要缩小数据库文件，尤其是在一般情况下不要这么做。

检测 SQL Server 中的数据库数据文档增长

我曾经多次和那些更多关注磁盘空间而不是性能的人们讨论过这个话题，所以在这里我决定进行一下测试来证明我所说的属实。此外我还要告诉大家数据库文档在事务办理过程中增长对 SQL Server 性能会带来什么样的影响。

- 你能够最大限度地缩小数据库吗？（一）
- 你能够最大限度地缩小数据库吗？（二）
- 你能够最大限度地缩小数据库吗？（三）
- 你能够最大限度地缩小数据库吗？（四）
- 你能够最大限度地缩小数据库吗？（五）
- 你能够最大限度地缩小数据库吗？（六）

禁用 Shrink Database 任务

定期压缩你的 SQL Server 数据库通常不是节省空间的最佳方法，如果 DBA 能够意识到这一点的话，那接下来的一个步骤就是解决如何禁用这项任务的自动执行问题。SQL Server 专家 Michelle Gutzait 谈到了三种压缩数据库的方法并详细阐述了禁用过程。

- 教你在 SQL Server 2000/2005 中禁用 Shrink Database 任务（一）
- 教你在 SQL Server 2000/2005 中禁用 Shrink Database 任务（二）

压缩 SQL Server 事务日志文档

在该系列第一部分有关数据库文件压缩的文章中，我们测试并证实了事务中数据文件增长时的性能实质（performance implications）。在该事务日志情况下，还存在着潜在的性能问题。事务日志通常比数据库文件增长得要快，并且在数据库管理员也会更加经常对它们进行压缩……

- 你能最大限度压缩 SQL Server 事务日志文档吗？（一）
- 你能最大限度压缩 SQL Server 事务日志文档吗？（二）
- 你能最大限度压缩 SQL Server 事务日志文档吗？（三）
- 你能最大限度压缩 SQL Server 事务日志文档吗？（四）

你能够最大限度地缩小数据库吗？（一）

我经常看见人们制定定期缩小数据库文件（数据或 T-Log）的 SQL Server 维护计划和定制任务。我通常也会建议我的客户和同事除非他们要研究数据库增长原因，否则不要缩小数据库文件，尤其是在一般基础上不要这么做。

我曾经多次和那些更多关注磁盘空间而不是性能的人们讨论过这个话题，所以在这里我决定进行一下测试来证明我所说的属实。此外我还要告诉大家数据库文件在事务办理过程中增长对 SQL Server 性能会带来什么样的影响。

我将在以下几个增长环境中进行测试：

1、数据库数据文件增长
2、数据库 T-log 增长
3、Tempdb 增长

这是本系列中的第一篇技巧，即检测 SQL Server 中的数据库数据文件增长。

测试环境

硬件和软件配置

我进行测试的系统硬件和软件配置如下：

图 1：测试系统的硬件和软件配置

SQL Server 2005 SP2， Enterprise Evaluation Edition，计算机中有一个本地的 iSCSI LUN。

数据库配置

我创建了 ShrinkDB 数据库，它包括以下数据库文件和属性：

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
ShrinkDB	Data	PRIMARY	2	By 1 MB, unrestricted growth
ShrinkDB_log	Log	Not Applicable	10	By 1 MB, restricted growth to 2097152 MB

图 2：ShrinkDB 数据库文件和属性

以下查询：

```
select size from sysfiles where fileid = 1
```

返回：256

这是文件大小（在 8-KB 页面中）。256*8 K = 2 MB。

在这个数据库中，我创建了下表：

```
create table ExpandDB (a varchar(8000))
```

(作者：Michelle Gutzait 译者：April 来源：TT 中国)

你能够最大限度地缩小数据库吗？（二）

目标

我的目标就是在事物大小和数据文件自动增长比率的基础上测试性能：

1、 测试 1-增长率小，事务短，插入行数为 X
2、 测试 2-增长率小，事务短，插入行数为 X*Y （比测试 1 中插入的行数要多）
3、 测试 3-增长率大，事务短，插入行数为 X*Y（和测试 2 中插入的行数相等）
4、 测试 4-增长率小，一个事务中插入 X*Y 行
5、 测试 5-增长率大（如测试 3 一样），一个事务中插入的行数为 X*Y。
6、 测试 6-自动增长率很大，事务很短
7、 测试 7-自动增长率很大，事务很长

我将在结果中分析不同之处并且尝试得出最后的结论。

注：该测试中不包括并发情况。

方法和代码描述

对于每个测试来说，我先测试数据库文件的原始文件大小和自动增长率。例如，原始大小=2 MB，自动增长率= 1 MB。

我还要测定该数据库文件的目标文件大小，所以要执行的第一个循环就应该是插入行，至少直到达到数据文件的目标大小为止。

```
while (select size from sysfiles where fileid = 1) <= <X>
```

```
insert into ExpandDB select replicate ('a', 8000))
```

这个环节中我们在 ExpandDB 表中一行一行地插入了 8000 个字节，直到数据文件大小超过 X。ExpandDB 表为 heap 表，没有索引或约束避免其他特殊优化或索引。

在执行第一个循环达到目标文件大小之后，为了比较我需要执行相同量的插入，但是这些插入行不能是增长中的文件。所以我就要截去一部分表并清除 transaction log 保证 T-Log 不会过早增长。

以下查询：

```
select count(*) from ExpandDB
```

返回在第一个环节中插入行的数量，

```
select size from sysfiles where fileid = 1
```

返回新数据库数据文件的大小。

以下语句将截断表和 T-Log：

```
truncate table ExpandDB
go
backup transaction ShrinkDB with truncate_only
go
```

下一步就是执行相同的插入命令：

```
declare @i int
set @i = 1
while @i <= <Z>
begin
insert into ExpandDB select replicate ('a', 8000)
set @i = @i + 1
end
```

<Z>就是在第一个环节中插入行后我们得到的表中的行数。另外一件很重要的事情就是在执行第一个环节和第二个环节中清除缓冲，消除缓冲所带来的不平等性，以便于进行性能比较：

```
DBCC DROPCLEANBUFFERS
```

最后，每个行中将每次测试将进行三次（反复三次），务必保证结果的一致性。

因此，在每次反复中的第一步就是保证能够达到你想达到的数据文件原始大小。例如，缩小数据库文件：

```
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
```

性能监控工具

用 SQL Profiler 来检测性能。

(作者：Michelle Gutzait 译者：April 来源：TT 中国)

你能够最大限度地缩小数据库吗？（三）

结果总结

将结果放在表格中进行比较，比较项目包括 CPU、读和写意及第一个循环和第二个循坏之间的间隔时间，首先是文件增长，后来就不会出现文件增长的情况。

这里有两个地方进行的比较：百分比不同（第二环节和第一个环节之间的比为*100）并且不同的处数也不同（(<First loop is >－<Second loop is >)。如果得到的是正数，那就是说第二个环节进行得比较好。如果是负数，意思就相反。在每个比较表的最后，比较得出的平均值就是这三次重复得到的结果平均值。

在这篇文章的最后，我们通过一个表来比较一下所有测试过程中所得到的平均值。

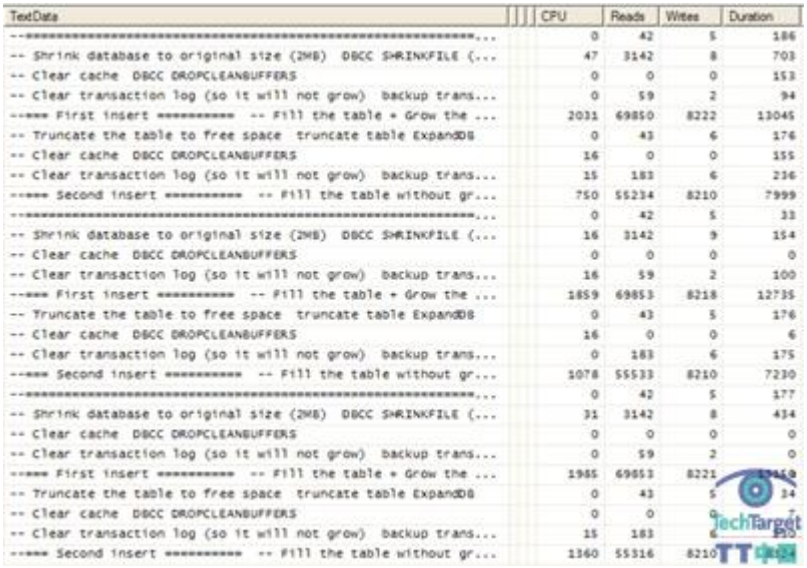
测试 1：

原始文件大小=256
目标文件大小=8448
小事务（一次 1 行），
总行数：8160
文件增长大小为 1MB

```
--====================================================================
-- Truncate the table to free space
truncate table ExpandDB
go
-- Shrink database to original size (2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Check that the size was reduced properly
select size from sysfiles where fileid = 1
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== First insert ==========
-- Fill the table + Grow the database file x 30
while (select size from sysfiles where fileid = 1) <= 8400
      insert into ExpandDB select replicate ('a',8000)
go
-- Make sure that the number of rows does not change
select count(*) from ExpandDB
go
-- What is the size of the file now?
select size from sysfiles where fileid = 1
go
-- Truncate the table to free space
truncate table ExpandDB
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== Second insert ==========
-- Fill the table without growing the file
declare @i int
set @i = 1
while @i <= 8160     -- Same number of rows inserted in the first loop
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the file now?
select size from sysfiles where fileid = 1
go
```


图 3

下面是连续执行三次上述代码所得到的 SQL Profiler 结果（我清除了调试查询）：



TextData	CPU	Reads	Writes	Duration
--==...	0	42	5	186
-- Shrink database to original size (2MB) DBCC SHRINKFILE (...	47	3142	8	703
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	153
-- Clear transaction log (so it will not grow) backup trans...	0	59	2	94
--=== First insert ========== -- Fill the table + Grow the ...	2031	69850	8222	13045
-- Truncate the table to free space truncate table ExpandDB	0	43	6	176
-- Clear cache DBCC DROPCLEANBUFFERS	16	0	0	155
-- Clear transaction log (so it will not grow) backup trans...	15	183	6	236
--=== Second insert ========== -- Fill the table without gr...	750	55234	8210	7999
--==...	0	42	5	33
-- Shrink database to original size (2MB) DBCC SHRINKFILE (...	16	3142	9	154
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	0
-- Clear transaction log (so it will not grow) backup trans...	16	59	2	100
--=== First insert ========== -- Fill the table + Grow the ...	1859	69853	8218	12735
-- Truncate the table to free space truncate table ExpandDB	0	43	5	176
-- Clear cache DBCC DROPCLEANBUFFERS	16	0	0	6
-- Clear transaction log (so it will not grow) backup trans...	0	183	6	175
--=== Second insert ========== -- Fill the table without gr...	1078	55533	8210	7230
--==...	0	42	5	177
-- Shrink database to original size (2MB) DBCC SHRINKFILE (...	31	3142	8	434
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	0
-- Clear transaction log (so it will not grow) backup trans...	0	59	2	0
--=== First insert ========== -- Fill the table + Grow the ...	1985	69853	8221	[illegible]
-- Truncate the table to free space truncate table ExpandDB	0	43	5	14
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	[illegible]
-- Clear transaction log (so it will not grow) backup trans...	15	183	6	250
--=== Second insert ========== -- Fill the table without gr...	1360	55316	8210	[illegible]

图 4：三次执行后的 SQL Profiler 结果

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows	2031	69850	8222	13045
	File does NOT grow	750	55234	8210	7999
	% improvement	63.07237814	20.92483894	0.145949891	38.68148716
	Difference	1281	14616	12	5046
2	File grows	1859	69853	8218	12735
	File	1078	55533	8210	7230

	does NOT grow				
	% improvement	42.01183432	20.50019326	0.097347286	43.22732627
	Difference	781	14320	8	5505
3	File grows	1985	69853	8221	13150
	File does NOT grow	1360	55316	8210	8324
	% improvement	31.486146 1	20.81084563	0.133803674	36.69961977
	Difference	625	14537	11	4826
	Average % improvement	45.52345285	20.74529261	0.125700284	39.536144 4
	Average difference	895.666667	14491	10.33333333	5125.666667

(作者：Michelle Gutzait 译者：April 来源：TT 中国)

你能够最大限度地缩小数据库吗？（四）

结果总结

测试 2

原始文件大小=256

目标文件大小=34816

小事务（一次 1 行），

总行数：33373

文件增长大小为 1MB

现在我们来测试一下插入更多数据。代码为：

```
--============================================================
-- Truncate the table to free space
truncate table ExpandDB
go
-- Shrink database to original size (2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Check that the size was reduced properly
select size from sysfiles where fileid = 1
go
-- Clear cache
DBCC DROPCLEANBUFFERS
Go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== First insert ==========
-- Fill the table + Grow the database file x 30
while (select size from sysfiles where fileid = 1) <= 33600
      insert into ExpandDB select replicate ('a',8000)
go
-- Truncate the table to free space
truncate table ExpandDB
go
-- Clear cache
DBCC DROPCLEANBUFFERS
Go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== Second insert ==========
-- Fill the table without growing the file
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
```

图 5：测试更多条插入的 SQL 代码

结果总结：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows	6730	285208	33618	56148
	File does NOT grow	4250	225591	33567	35003
	% improvement	**36.84992571**	**20.9029901**	**0.151704444**	**37.65940016**
	Difference	**2480**	**59617**	**51**	**21145**

2	File grows	7031	285223	33619	53373
	File does NOT grow	4204	225364	33564	32472
	% improvement	**40.20765183**	**20.98673669**	**0.163597965**	**39.16024956**
	Difference	**2827**	**59859**	**55**	**20901**
3	File grows	6453	285278	33618	54189
	File does NOT grow	3844	225362	33564	33530
	% improvement	**40.43080738**	**21.00267108**	**0.160628235**	**38.1239735**
	Difference	**2609**	**59916**	**54**	**20659**
	Average % improvement	**39.16279497**	**20.96413262**	**0.158643548**	**38.31454107**
	Average difference	**2638.666667**	**59797.33333**	**53.33333333**	**20901.66667**

测试 3

原始文件大小=256

目标文件大小=34816

小事务（一次 1 行），

文件增长大小为 10MB

这次测试中的代码和 Test 2 中的代码一样，但是数据库数据文件现在设定为自动增长到 10MB。

结果总结

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows	8907	283930	33588	41354
	File does NOT grow	6297	225358	33565	37755
	% improvement	**29.30279555**	**20.62902828**	**0.068476837**	**8.702906611**
	Difference	**2610**	**58572**	**23**	**3599**
2	File grows	9078	283902	33588	46607
	File does NOT grow	5578	216971	33560	41202
	% improvement	**38.55474774**	**23.57538869**	**0.083363106**	**11.59697041**
	Difference	**3500**	**66931**	**28**	**5405**
3	File grows	9016	283909	33587	52515
	File does NOT grow	6015	225358	33565	38807
	% improvement	**33.28527063**	**20.62315742**	**0.065501533**	**26.10301819**

	Difference	**3001**	**58551**	**22**	**13708**
	Average % improvement	**33.71427131**	**21.60919146**	**0.072447159**	**15.46763174**
	Average difference	**3037**	**61351.33333**	**24.33333333**	**7570.666667**

测试 4

原始文件大小=33664

目标文件大小=66944

大事务（33373 行），

文件增长大小为 1MB

在该测试中，我将 ShrinkDB 表再次插入到大事务中的 ShrinkTable 里（每次 33373 行）。文档增长率为 1 MB，我必须将 T-Log 文档增加到 200 MB，这样执行事务时才它才不会增长。我现在在测试文档的增长是如何影响事务的。

代码：

```
------------------------------------------------------
-- Truncate the table
truncate table ExpandDB
go
-- Shrink database as much as possible (to 2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== First insert ==========
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Truncate the table again
truncate table ExpandDB
go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== Second insert ==========
-- Fill the table without growing the file
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
```

结果总结：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows	3969	880471	33381	36097
	File does NOT grow	3720	879426	33380	18104
	% improvement	**6.273620559**	**0.118686476**	**0.002995716**	**49.84624761**
	Difference	**249**	**1045**	**1**	**17993**
2	File grows	3750	880473	33381	35962
	File does NOT grow	3578	879435	33380	20006
	% improvement	**4.586666667**	**0.117891179**	**0.002995716**	**44.36905623**
	Difference	**172**	**1038**	**1**	**15956**
3	File grows	3657	880471	33381	36544
	File does NOT grow	3999	879422	33380	18802
	% improvement	**-9.35192781**	**0.119140778**	**0.002995716**	**48.54969352**
	Difference	**-342**	**1049**	**1**	**17742**
	Average % improvement	**0.502786472**	**0.118572811**	**0.002995716**	**47.58833245**
	Average difference	**26.33333333**	**1044**	**1**	**17230.33333**

(作者：Michelle Gutzait 译者：April 来源：TT 中国)

你能够最大限度地缩小数据库吗？（五）

测试 5

原始文件大小=33664

目标文件大小=66944

大事务（33373 行），

文件增长大小为 10MB

在该测试中，代码和 Test 4 中运行的代码是一样的，但是文档自动增长大小设置为 10MB。

结果总结：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows	4016	879535	33381	23003
	File does NOT grow	4016	879432	33380	21177
	% improvement	0	0.011710734	0.002995716	7.938095031
	Difference	0	103	1	1826
2	File grows	3672	879531	33381	22501
	File does NOT grow	3673	879439	33380	19965
	% improvement	-0.027233115	0.01046012	0.002995716	11.2706102
	Difference	-1	92	1	2536
3	File grows	3798	879544	33381	22366
	File does NOT grow	3782	879426	33380	18702
	% improvement	0.421274355	0.013416043	0.002995716	16.38200841
	Difference	16	118	1	3664
	Average % improvement	0.13134708	0.011862299	0.002995716	11.86357121
	Average	5	104.3333333	1	2675.333333

	difference				

测试 6

原始文件大小=256

目标文件大小=128256

小事务（每次一行），

文件增长大小为 1000MB

在该测试中，我插入的是 33373 行，其中有一次让文档增长，另一次不让它增长。

代码：

```
--============================================================
-- Truncate the table
truncate table ExpandDB
go
-- Shrink database as much as possible (2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== First insert ==========
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Truncate the table
truncate table ExpandDB
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== Second insert ==========
-- Insert all rows from ExpandDB table again
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
```


图 7：允许 SQL 数据文档增长测试

结果总结：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)

1	File grows	3343	217006	33382	19335
	File does NOT grow	3500	216989	33383	18020
	% improvement	**-4.696380497**	**0.007833885**	**-0.002995626**	**6.801137833**
	Difference	**-157**	**17**	**-1**	**1315**
2	File grows	3344	217006	33385	30126
	File does NOT grow	3359	217001	33378	16499
	% improvement	**-0.448564593**	**0.002304084**	**0.0209675**	**45.23335325**
	Difference	**-15**	**5**	**7**	**13627**
3	File grows	3578	217006	33382	16884
	File does NOT grow	3234	216989	33382	21183
	% improvement	**9.61430967**	**0.007833885**	**0**	**-25.46197584**
	Difference	**344**	**17**	**0**	**-4299**
	Average % improvement	**1.489788193**	**0.005990618**	**0.005990625**	**8.857505083**
	Average difference	**57.33333333**	**13**	**2**	**3547.666667**

测试 7

原始文件大小=34816

目标文件大小=1314816

大事务（33373 行），

文件增长大小为 1000MB

代码：

```
--=====================================================
-- Set autogrowth to 10MB
ALTER DATABASE [ShrinkDB] MODIFY FILE ( NAME = N'ShrinkDB', FILEGROWTH
= 10240KB )
go
-- Truncate the table
truncate table ExpandDB
go
-- Shrink database as much as possible (to 256)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Set autogrowth to 10000MB
ALTER DATABASE [ShrinkDB] MODIFY FILE ( NAME = N'ShrinkDB', FILEGROWTH
= 10240000KB )
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== First insert ==========
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Truncate the table again
truncate table ExpandDB
go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== Second insert ==========
-- Fill the table without growing the file
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
```

图 8：允许 SQL 数据文档增长测试

结果总结：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows	3548	867705	45151	24096
	File does NOT grow	3250	878488	34317	20707
	% improvement	**8.399098083**	**-1.242703453**	**23.99503887**	**14.06457503**
	Difference	**298**	**-10783**	**10834**	**3389**
2	File grows	3579	879331	33542	21753
	File does NOT grow	3765	872841	39965	22775
	% improvement	**-5.196982397**	**0.738061094**	**-19.14912647**	**-4.698202547**
	Difference	**-186**	**6490**	**-6423**	**-1022**
3	File grows	3781	879411	33452	25014
	File does NOT grow	3422	870858	41918	21235
	% improvement	**9.494842634**	**0.972582786**	**-25.30790386**	**15.10753978**
	Difference	**359**	**8553**	**-8466**	**3779**
	Average % improvement	**4.23231944**	**0.155980142**	**6.82066382**	**8.157970755**
	Average difference	**157**	**1420**	**1351.66667**	**2048.666667**

（作者：Michelle Gutzait 译者：April 来源：TT 中国）

你能够最大限度地缩小数据库吗？（六）

总结：

分析平均数：

测试 1-小事务、小增长

	CPU	Reads	Writes	Duration (ms)
Average % improvement	45.52345285	20.74529261	0.125700284	39.5361444
Average difference	895.6666667	14491	10.33333333	5125.666667

测试 2-小事务、小增长、行数更多

	CPU	Reads	Writes	Duration (ms)
Average % improvement	39.16279497	20.96413262	0.158643548	38.31454107
Average difference	2638.666667	59797.33333	53.33333333	20901.66667

测试 3-小事务、大增长、行数和测试 2 中的一样

	CPU	Reads	Writes	Duration (ms)
Average % improvement	33.71427131	21.60919146	0.072447159	15.46763174
Average difference	3037	61351.33333	24.33333333	7570.666667

测试 4-大事务、小增长、行数和测试 2 中的一样多

	CPU	Reads	Writes	Duration (ms)
Average % improvement	0.502786472	0.118572811	0.002995716	47.58833245
Average difference	26.33333333	1044	1	17230.33333

测试 5-大事务、更大增长、插入行数和测试 2 中的一样多

	CPU	Reads	Writes	Duration (ms)
Average % improvement	0.13134708	0.011862299	0.002995716	11.86357121

Average difference	**5**	**104.3333333**	**1**	**2675.333333**

测试 6-小事务、大增长、行数和测试 2 中的一样多

	CPU	Reads	Writes	Duration (ms)
Average % improvement	**1.489788193**	**0.005990618**	**0.005990625**	**8.857505083**
Average difference	**57.33333333**	**13**	**2**	**3547.666667**

测定 7-大事务、大增长、行数和测试 2 中的一样多

	CPU	Reads	Writes	Duration (ms)
Average % improvement	**4.23231944**	**0.155980142**	**-6.82066382**	**8.157970755**
Average difference	**157**	**1420**	**-1351.66667**	**2048.666667**

结果证明在性能在随着文档增长而减弱，尤其是当事务短并且增长率小的时候。同样，自动增长率越高，那么性能就越好。事务时间越长，性能就受自动增长率的影响越少，所以在你压缩你的数据库文档的时候，你应该问问自己是不是值得这么做，也就是说文档不会在将来再次增长。此外，你应该根据你所期望的数据库活动和增长率设置一个门槛。

即使是 CPU 中出现了空白，事务间的读取和写入造成的自动增长以及没有自动自动增长的事务不能够不是很广，事务的持续时间在文档增长时还是会受到影响。

(作者：Michelle Gutzait 译者：April 来源：TT 中国)

教你在 SQL Server 2000/2005 中禁用 Shrink Database 任务（一）

经常压缩 SQL Server 数据库并不是节省空间的最佳策略，并且 DBA 已经意识到下一步就是解决如何终止自动执行任务。SQL Server 专家 Michelle Gutzait 探讨了三种常见的压缩数据库的方法以及如何停止它们。

问题：

我的数据库正在自动地压缩，但是我不知道如何停止它。请给我一些建议。

作者注：

我们并不总提倡在例行程序上压缩数据库或数据库文件。当数据库文件在事务进程中需要增长时，压缩数据库可能会降低性能。更多信息，请查询我的另一篇技巧文章《你能最大限度缩小数据库吗？》。

自动压缩数据库方法：

1、自动压缩数据库选项

2、压缩数据库维护任务/计划

3、自定义任务

如何停止数据库自动压缩

1）自动压缩数据库选项

这里是关于如何在 SQL Server 和 SQL Server 中手动终止自动压缩数据库选项的方法：

```
ALTER DATABASE dbname SET AUTO_SHRINK OFF
```

另外一个选择是使用 GUI (Enterprise Manager/SQL Server Management Studio)：

在 SQL Server2000 中，右击数据库 - Properties - Options：

图 1：在 SQL Server 2000 中，通过置空 Auto Shrink 选项来停止它。

在 SQL Server2005 中，右击数据库 - Properties - Options：

图 2：在 SQL Server 2005 中，通过设置 Auto Shrink 选项为 False 来停止它。

(作者：Michelle Gutzait 译者：曾少宁／陈柳 来源：TT 中国)

教你在 SQL Server 2000/2005 中禁用 Shrink Database 任务（二）

2）压缩数据库维护任务/计划

通过 SQL Server 2000 的 Enterprise Manager，双击打开维护计划：

图 3：打开 SQL Server Enterprise Manager 中的维护计划

在 Optimization 标签中停止数据库选项：

图 4：在 Optimization 标签中停止压缩数据库选项

你还可以拥有以下选项：

如果压缩数据库任务是维护计划的唯一任务，那么我们可以任意做出以下选择：

- 删除维护计划
- 停止运行维护计划中的任务

- 删除维护计划的调度——这将删除任务。我们可以点击 Schedule - Change 按钮，然后停止右上方的调度设置选项。

如果压缩数据库任务不是维护计划的唯一任务，那么我们可以在以下选择其中一种方法：

- 停止上面描述的选项
- 停止调度（如果不需要重新组织索引）

在 SQL Server 2005 的 Management Studio 中，双击打开维护计划：

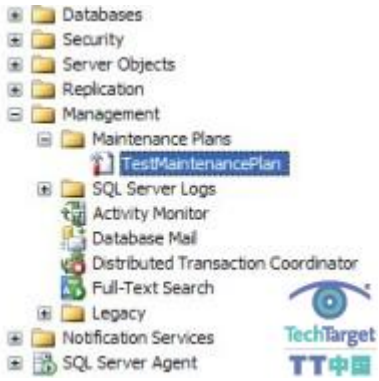


图 5：在 SQL Server 2005 Management Studio 中，双击维护计划

查找负责压缩数据库的任务并双击它：

图 6：查找负责压缩数据库的 SQL Server Management Studio 并双击它。

现在我们有一下几个选择：

1、如果压缩数据库任务是维护计划的唯一任务，那么我们可以选择以下任意方法：

- 删除维护计划
- 停止运行维护计划的任务
- 删除维护计划的调度——这就删除了任务。

2、如果压缩数据库任务不是维护计划的唯一任务，那么我们可以选择以下一种方法：

- 从维护计划中删除任务
- 置空（没有选择数据库）

3）自定义任务

我曾经见过通过自定义任务自动执行压缩数据库任务的案例。我们可以通过名称((Content component not found.))查找用以压缩数据库的任务，或者我们将查找文本（两种方法都可以在 SQL Server 2000/2005 上进行）：

```
DBCC SHRINKDATABASE
Or
DBCC SHRINKFILE
```

第一的命令压缩数据库，而第二个压缩一个指定的文件。下面是 SQL Server 中终止自动压缩自定义任务的选择：

1、删除任务的步骤（如果它不是唯一的步骤）。

2、注释掉压缩数据库/文件的代码部分。

3、停止任务（如果它仅仅用于压缩数据库/文件）。

4、删除任务调度（如果它仅仅用于压缩数据库/文件）。

（作者：Michelle Gutzait 译者：曾少宁/陈柳 来源：TT 中国）

你能最大限度压缩 SQL Server 事务日志文档吗？（一）

在该系列第一部分有关数据库文件压缩的文章中，我们测试并证实了事务中数据文件增长时的性能实质（performance implications）。在该事务日志情况下，还存在着潜在的性能问题。

事务日志通常比数据库文件增长得要快，并且在数据库管理员也会更加经常对它们进行压缩。在本篇技巧中，我将测试事务中事务日志文件的影响。在本文的示例中，我用到的是和前一篇技巧相同的环境和配置，唯一不同的地方就是我在 SQL Server 上安装的 SP3。

数据库

我测试的数据库为 ShrinkDB，我将它设置为在事务过程中不允许数据库文件增长。但是事务日志文件大小仅仅为 2 MB，测试目标为让每个事务增长。

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
ShrinkDB	Rows Data	PRIMARY	200000	None
ShrinkDB_log	Log	Not Applicable	2	By 1 MB, restricted growth to 2097152 MB

ShrinkDB 数据库恢复模式：设置为 FULL。

```
select size from sysfiles where fileid = 2
```

以上查询中，fileid 为事务日志文件，返回值为 256。这就是说事务日志文件页数为 8 KB，大小为 2 MB。

在我上一篇文章中，再一次用到了 ExpandDB 表。

```
create table ExpandDB (a varchar(8000))
```

测试

测试目标为了解 INSERT、UPDATE 和 DELETE 命令和事务日志之间是如何相互影响的。

我将整个测试分为两个阶段：

在第一阶段，我监测到了事务日志增长相关操作程序修改的一般行为。

在第二阶段，我只用到了在第一阶段中促使事务日志增长的那些命令。在这一阶段，我测试了和第一部分中相同的自动增长情况。

由于更新、插入和删除表现各不相同，每个阶段都包含三个部分，一下每项操作就是一个部分：UPDATES， INSERTS 和 DELETE。

第一阶段：事务日志增长行为

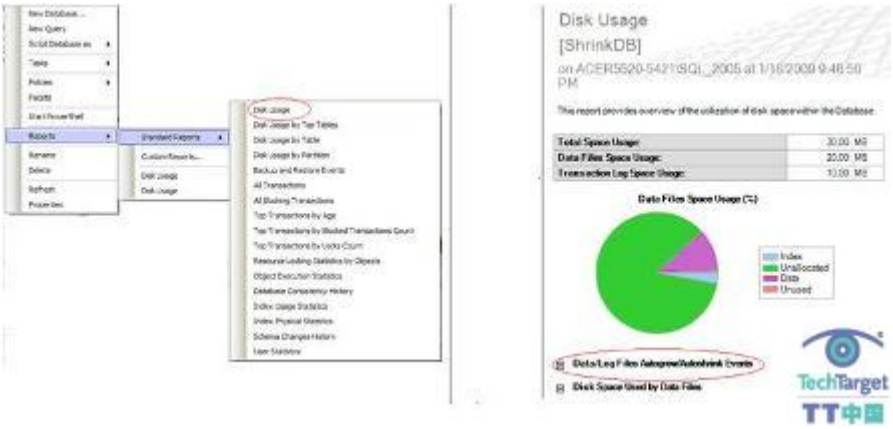
UPDATE 命令和事务日志增长

在本次测试中，代码仅仅在表中插入一行，并且在事务日志文件增长到 8 MB（1，024 页）时，代码将自动更新这个行。

```
-- Insert one row into the table
insert into ExpandDB select replicate ('a',8000)
--============================================================
-- Update the table until T-Log reaches 8MB
while (select size from sysfiles where fileid = 2) <= 1024
      update ExpandDB set a = replicate ('a',8000)
go
```


结果让人觉得很吃惊。事务日志文件并没有增长并且事务日志中闲置空间所占百分比一直为 30%以上（这个百分比是由 dbcc sqlperf(logspace)计算出来的）。在事务中，事

务日志会一直填充至 70%，然后减少至 50%左右。为了保证我能从 dbcc sqlperf 中获得正确数据，我还可以运行报告（report）查询数据库文件的自动增长率。



（作者：Michelle Gutzait 译者：April 来源：TT 中国）

你能最大限度压缩 SQL Server 事务日志文档吗？（二）

这一次它并没有报告数据库的自动增长数据。

于是我决定用短时间更新（一次一行）检查大事务，多以我增加了一个开始事务：

```
-- Big transaction
begin tran
while (select size from sysfiles where fileid = 2) <= 1024
      update ExpandDB set a = replicate ('a',8000)
go
commit tran
```


这次的结果让人觉得非常迷惑。事务日志文档在最初 2 MB 的基础上没有增长，并且事务日志中的空余空间比例大于 60%。我删除了这个环节并制定了事务。之后，事务日志文档中已使用的空间大小甚至还减小了。

在 UPDATE 命令另外一个测试中，我在表中插入了 10，000 行并运行以下代码：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go

-- Check the size of the T-Log file (RESULT = 2MB)
select size from sysfiles where fileid = 2
go

-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Update 10000 rows at a time
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
```

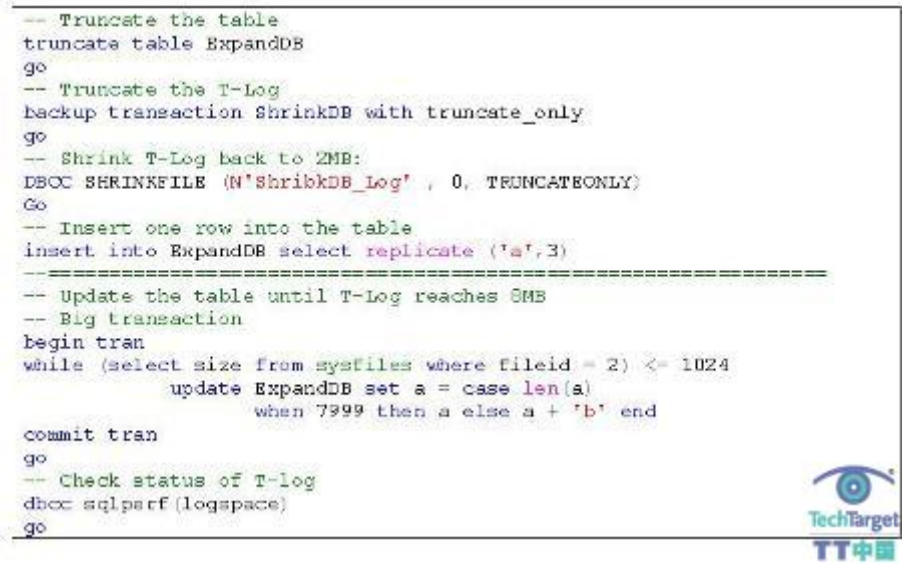
所得结果相同。该事务日志文档增长了 1 MB 至 10 MB，但是达到这个大小之后就会原地不动。无论我执行同样的更新多少次，空间都会被先添满然后再减少：

Database Name	Log Size (MB)	Log Space Used (%)	Status
ShrinkDB	9.992188	40.39777	0
ShrinkDB	9.992188	62.96423	0
ShrinkDB	9.992188	37.40227	0

我查询了这个问题的相关答案但是在 SQL Server 2005 文档里并没有找到。我又在另外一台 SQL Server 2005 SP3 的机器上进行了这项测试，还是遇到了相同的情况。

用 ApexSQL Log 工具分析事务日志内容，结果都证实了每次只要我查看事务日志文件，我都会获得不同的更新数据。这种情况和事务日志大小一样，开始时增长、后来又压缩，但是文档大小没有变化。

另外我还想了解一下事务日志行为，我执行的这个事务中，UPDATE 命令修改了行中的值：



```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShribkDB_Log' , 0, TRUNCATEONLY)
Go
-- Insert one row into the table
insert into ExpandDB select replicate ('a',3)
--============================================================
-- Update the table until T-Log reaches 8MB
-- Big transaction
begin tran
while (select size from sysfiles where fileid = 2) <= 1024
            update ExpandDB set a = case len(a)
                  when 7999 then a else a + 'b' end
commit tran
go
-- Check status of T-log
dbcc sqlperf(logspace)
go
```

在这种情况下，事务日志增长到了 4 MB，空间利用率还是在很长一段时间内为 94%。但是我没有获得任何提示错误信息向我说明字符串被切断或者溢出。如同 ApexSQL Log 工具所检测到的，事务日志的内容在停止增长时约为 13，380 行。此外，SQL Profiler 监控还显示在事务停止增长后就不会执行任何命令，也不会出现任何错误。取消事务后，在错误日志中不会出现显示自动增长操作已被取消或失败等具体信息，但是事务还是正常进行。

在相似的实例中，我有时还能发现错误日志中以下信息：

```
"Autogrow of file '<DB logical file name' in database '<Database name>' was c
anceled by user or timed out after 469 milliseconds. Use ALTER DATABASE to set
 a smaller FILEGROWTH value for this file or to explicitly set a new file size
."
```

但是无论我执行多少次上述代码，在测试过程中都不会发生那种情况。

取消事务以后，事务日志的空间利用率减到了约 45%。

我决定在事务日志中行数不变的情况下运行上述 UPDATE 命令：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go

-- Check the size of the T-Log file (RESULT = 2MB)
select size from sysfiles where fileid = 2
go

-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Update 10000 rows at a time
update ExpandDB set a = case len(a)
                    when 7999 then a else a + 'b' end
go
update ExpandDB set a = case len(a)
                    when 7999 then a else a + 'b' end
go
update ExpandDB set a = case len(a)
                    when 7999 then a else a + 'b' end
go
update ExpandDB set a = case len(a)
                    when 7999 then a else a + 'b' end
go
update ExpandDB set a = case len(a)
                    when 7999 then a else a + 'b' end
go
update ExpandDB set a = case len(a)
                    when 7999 then a else a + 'b' end
go
update ExpandDB set a = case len(a)
                    when 7999 then a else a + 'b' end
go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
```


(作者：Michelle Gutzait 译者：April 来源：TT 中国)

你能最大限度压缩 SQL Server 事务日志文档吗？（三）

结果显示：如果我一个一个进行更新，中间再等待一两秒钟，事务日志就不会增长、空间就会和以前一样空，不会被填满。我在每个行里执行两个更新时，在第二个事务运行之前，空间并没有填满或腾空。在这种情况下，事务日志就会增长。

INSERT 命令和事务日志增长

这次测试的目标就是让事务日志增长到 8MB，然后比较数据文档增长率和事务日志增长率。数据文档设置为 200CB，没有增长，事务日志设为 2MB，增长大小为 1MB。我执行了一下代码：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log' , 0, TRUNCATEONLY)
Go
-- Check the size of the T-Log file
select size from sysfiles where fileid = 2
go
-- Insert narrow rows into the table
while (select size from sysfiles where fileid = 2) <= 1024
      insert into ExpandDB select replicate ('a',3)
go
```

同时，我还执行了以下代码：

```
select getdate()
dbcc sqlperf (logspace)
go
waitfor delay '00:00:02'
go
select getdate()
dbcc sqlperf (logspace)
go
select getdate()
go
dbcc sqlperf (logspace)
go
waitfor delay '00:00:02'
go
```

这次执行了很长一段时间，数据文档填充和腾空的速度很慢，但是事务日志文档大小约为 2MB。以上代码返回下面这些结果：

Results Messages

	(No column name)
1	2009-01-26 23:13:51.233

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	69.84127	0
2	tempdb	0.7421875	54.47368	0
3	model	0.4921875	50	0
4	msdb	1.992188	36.07843	0
5	ShrinkDB	1.992188	56.4951	0

	(No column name)
1	2009-01-26 23:13:53.420

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	69.84127	0
2	tempdb	0.7421875	54.47368	0
3	model	0.4921875	50	0
4	msdb	1.992188	36.07843	0
5	ShrinkDB	1.992188	39.33823	0

	(No column name)
1	2009-01-26 23:13:53.650

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	69.84127	0
2	tempdb	0.7421875	54.47368	0
3	model	0.4921875	50	0
4	msdb	1.992188	36.07843	0
5	ShrinkDB	1.992188	43.06372	0

很显然，剩余空间减少了。记在删除测试 45 分钟插入时间之前，我在表中大约有 180 万个行，并且事务日志仍然为 2MB。在删除之后，事务日志还是 2MB 并且只利用了 39.88% 的空间。数据文档的利用率为 31%。

然后我又尝试用大事务一次插入一行进行测试。

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log' , 0, TRUNCATEONLY)
Go
-- Insert 100000 rows
-- Big transaction
begin tran
while (select size from sysfiles where fileid = 2) <= 1024
      insert into ExpandDB select replicate ('a',3)
go
commit tran
go
dbcc sqlperf(logspace)
go
select count(*) from ExpandDB
go
```

事务日志增长到 9MB，并且最后表中的行数达到了 14，572。

Results | Messages

	DbId	FileId	CurrentSize	MinimumSize	UsedPages	EstimatedPages
1	5	2	256	256	256	256

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	46.03175	0
2	tempdb	0.7421875	69.67105	0
3	model	0.4921875	54.76191	0
4	msdb	1.992188	57.05882	0
5	ShrinkDB	8.992188	38.76521	0

	(No column name)
1	14572

要测试量比较大的升级，我在表中插入了 10，000 并执行了以下代码：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go

-- Check the size of the T-Log file (RESULT = 2MB)
select size from sysfiles where fileid = 2
go

-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB
Go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB
Go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB
Go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB
Go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
```


这一次，每行插入后事务日志翻了一番。

DELETE 命令和事务日志增长

现在我的目标和 INSERT 以及 UPDATE 命令相同，我在表中插入了 10，000 行，并执行以下代码每次删除移行：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log' , 0, TRUNCATEONLY)
Go
-- Insert 10000 rows
declare @i int
set @i = 1
while @i <= 10000
begin
      insert into ExpandDB select replicate ('a',1000)
      set @i = @i + 1
end
go
-- Delete rows one by one:
set rowcount 1
while (select size from sysfiles where fileid = 2) <= 1024
      delete from ExpandDB
set rowcount 0
go
```

TechTarget
TT中国

(作者：Michelle Gutzait 译者：April 来源：TT 中国)

你能最大限度压缩 SQL Server 事务日志文档吗？（四）

相同的情况表示：事务没有增长，并且空间也进行了一轮的填补或腾空。

然后我再尝试进行一次大规模删除：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log' , 0, TRUNCATEONLY)
Go
-- Insert 100,000 rows
declare @i int
set @i = 1
while @i <= 100000
begin
      insert into ExpandDB select replicate ('a',1000)
      set @i = @i + 1
end
go
-- Delete 10,000 at a time and monitor the T-Log size and free space:
begin tran
set rowcount 10000
while exists (select 1 from ExpandDB)
begin
      delete from ExpandDB
      dbcc sqlperf(logspace)
end
set rowcount 0
commit tran
go
```


结果显示事务日志增长了：

Database Name	Log Size (MB)	Log Space Used (%)	Status
ShrinkDB	29.99219	54.32241	0
ShrinkDB	32.99219	51.68571	0
ShrinkDB	34.99219	50.18001	0
ShrinkDB	37.99219	48.60683	0
ShrinkDB	39.99219	46.85608	0
ShrinkDB	40.99219	66.63094	0
ShrinkDB	40.99219	58.30355	0
ShrinkDB	41.99219	53.7686	0
ShrinkDB	44.99219	70.38548	0

最后，我用一些小操作运行大型事务：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShribkDB_Log' , 0, TRUNCATEONLY)
Go
-- Insert 100,000 rows
declare @i int
set @i = 1
while @i <= 100000
begin
      insert into ExpandDB select replicate ('a',1000)
      set @i = @i + 1
end
go
-- Delete 10,000 at a time and monitor the T-Log size and free space:
begin tran
set rowcount 1
while exists (select 1 from ExpandDB)
begin
      delete from ExpandDB
end
set rowcount 0
commit tran
go
```

显示结果表明事务日志虽然增长得很慢，但还是出现了增长。

总结

结果表明：在进行短时间、分别 INSERT、UPDATE 和 DELETE 命令时，事务日志并没有增长很多。但是和我看到的事务日志文档一样，这次测试有一些地方很奇怪。文档空间使用率在达到 70%后又开始腾空而不是继续增长。

在执行大型操作（包括事务中数量比较多的一些数据）时，事务就会出现增长，但是并不是我们想象的那样成直线增长。似乎事务有些数据后来又进行了压缩。

（作者：Michelle Gutzait 译者：April 来源：TT 中国）