重建索引是为了减少数据碎片。数据碎片会导致SQL Server进行不必要的数据读，降低SQL Server的性能。重建索引也会同时更新列统计，而如果查询所使用的列缺少或遗漏统计信息，这可能导致SQL Server内部的优化器选择比预期效率低的查询计划。

如果您重建了某张表上的聚集索引，该表上的非聚集索引也同时会被更新。

要更新索引，您可以使用Maintenance Wizard（相关内容您可以参考http://msdn.microsoft.com/en-us/library/ms180074.aspx），或在SQL Server代理（Agent）中运行如下的自定义代码来更新某个数据库中所有表上的索引：

您可以根据您的需求修改DBREINDEX的参数。

需要注意的是，重建非聚集索引时该表会暂时加上共享锁，对用户不可进行SELECT以外的操作；重建聚集索引时该表会暂时加上排外锁，不允许任何用户访问。因此需要制定好计划来预防可能的访问问题。

REBUILD有一个fill factor参数，如果fill factor设置为100%，这意味着每一个索引页都是完全满的，如果fill factor设置为50%意味着每个索引页都是半满的。对于fill factor 100%，每次新插入或更新一个记录，由于当前页没有空间可用，可能有分页情况产生。过多的分页会降低SQL Server的性能。下面具体举个例子：

假设您在一张表上建立了一个使用默认fill factor的新索引。当SQL Server创建索引时，它会把索引放置在连续的物理页上，以使数据顺序地被读，I/O访问最优化。但当表因INSERT，UPDATE，DELETE等操作增长改变时，分页发生，SQL Server在磁盘的其他地方分配新的页，导致新的页与原物理页不连续，增加了随机I/O，访问索引页变慢。

那么fill factor的合适值应该为多少？这取决于表的读/写比：

低更新表（读/写比：100比1）：100% fill factor

高更新表（写超过读）：50%-70% fill factor

居中：80%-90% fill factor

过低的fill factor会增加页的数量，也会导致更多的页需要被移至缓存，缓存中有用的数据减少。默认的fill factor为0（即100% fill factor），通常这不是个好的选择，特别是对于聚集索引。

如果您无法判断设置什么fill factor,您首先需要确定磁盘的读/写比.方法就是使用如下两个计数器:

Physical Disk Object: % Disk Read Time 和 Physical Disk Object: % Write Time。另外一个可能有用的计数器就是：SQL Server Access Methods: Pages Splits/Sec。这个计数器测量SQL Server内每秒分页的次数。如果该数值过高，您需要降低fill factor防止新的分页。

如果您想确认您的索引因分页产生的碎片程度，您可以运行DBCC SHOWCONTIG命令。如果看特定表和特定索引，您可以运行如下代码：

结果集中最重要的参数是Scan Density，越接近100%越好。如果Scan Density小于75%，那么您可能需要重建表中的索引。

对于小于100数据页，重建索引并不会有明显的性能改善。这是因为物理硬件缓存，SQL Server缓存和SQL Server预读机制隐藏了碎片的负面作用。但对于非常大的表，重建索引会使它受益匪浅，因为涉及大量磁盘I/O操作。

**标签：** <无>

**代码片段(4)**[[全屏查看所有代码]](http://www.oschina.net/code/piece_full?code=16436)

1. [代码]**重建索引**

[?](http://www.oschina.net/code/snippet_222150_16436#)

|  |  |
| --- | --- |
| 1 | ALTER INDEX Idx\_name ON tblCompany REBUILD |

2. [代码]**[SQL]代码**

[?](http://www.oschina.net/code/snippet_222150_16436#)

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | USE databasename --Enter the name of the database you want to reindex DECLARE @TableName VARCHAR(255) DECLARE tablecursor CURSOR FOR SELECT table\_name FROM information\_schema.tables WHERE table\_type = 'base table' OPEN tablecursor FETCH next FROM tablecursor INTO @TableName WHILE @@FETCH\_STATUS = 0 BEGIN DBCC dbreindex(@TableName, '', 90) FETCH next FROM tablecursor INTO @TableName END CLOSE tablecursor DEALLOCATE tablecursor |

3. [代码]**[SQL]代码**

[?](http://www.oschina.net/code/snippet_222150_16436#)

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | --Script to identify table fragmentation --Declare variables DECLARE @ID INT, @IndexID INT, @IndexName VARCHAR(128) --Set the table and index to be examined SELECT @IndexName = 'index\_name' --enter name of index SET @ID = Object\_id('table\_name') --enter name of table --Get the Index Values SELECT @IndexID = indid FROM sysindexes WHERE id = @ID AND name = @IndexName --Display the fragmentation DBCC showcontig (@id, @IndexID) |

4. [代码]**重建数据表索引和更新统计信息**

[?](http://www.oschina.net/code/snippet_222150_16436#)

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 | --方法一： USE adventureworks go EXEC Sp\_msforeachtable @command1="print '?' DBCC DBREINDEX ('?', '', 90)" go EXEC Sp\_updatestats go --方法二： USE adventureworks go CREATE PROCEDURE Usp\_reindex\_updatestats AS DECLARE @MyTable VARCHAR(255) DECLARE mycursor CURSOR FOR SELECT table\_name FROM information\_schema.tables WHERE table\_type = 'base table' AND table\_name NOT LIKE 'sys%' OPEN mycursor FETCH next FROM mycursor INTO @MyTable WHILE @@FETCH\_STATUS = 0 BEGIN PRINT 'Reindexing Table: ' + @MyTable DBCC dbreindex(@MyTable, '', 90) FETCH next FROM mycursor INTO @MyTable END CLOSE mycursor DEALLOCATE mycursor EXEC Sp\_updatestats go |

[**SQLServer 重建索引前后对比**](http://www.cnblogs.com/mingl12/p/5730178.html)

在做维护项目的时，我们经常会遇到索引维护的问题，通过语句，我们就可以判断某个表的索引是否需要重建。

执行一下语句：先分析表的索引

分析表的索引建立情况：DBCC showcontig('Table')

DBCC SHOWCONTIG 正在扫描 'Table'' 表...

表: 'Table'' (53575229)；索引 ID: 1，数据库 ID: 14

已执行 TABLE 级别的扫描。

- 扫描页数................................: 228

- 扫描区数..............................: 52

- 区切换次数..............................: 225

- 每个区的平均页数........................: 4.4

- 扫描密度 [最佳计数:实际计数].......: 12.83% [29:226]

- 逻辑扫描碎片 ..................: 97.37%

- 区扫描碎片 ..................: 98.08%

- 每页的平均可用字节数........................: 2686.3

- 平均页密度(满).....................: 66.81%

当你发现，扫描密度行，最佳计数和实际计数的比例已经严重失调，逻辑扫描碎片占了非常大的百分比，每页平均可用字节数非常大时，就说明

你的索引需要重新整理一下了。

执行重建索引命令：

DBCC DBREINDEX('Table'')

后分析的情况

DBCC SHOWCONTIG 正在扫描 'Table'' 表...

表: 'Table'' (53575229)；索引 ID: 1，数据库 ID: 14

已执行 TABLE 级别的扫描。

- 扫描页数................................: 154

- 扫描区数..............................: 20

- 区切换次数..............................: 19

- 每个区的平均页数........................: 7.7

- 扫描密度 [最佳计数:实际计数].......: 100.00% [20:20]

- 逻辑扫描碎片 ..................: 0.00%

- 区扫描碎片 ..................: 55.00%

- 每页的平均可用字节数........................: 86.8

- 平均页密度(满).....................: 98.93%