

This Free E-Book is brought to you by Natural-Aging.com.

100% Effective Natural Hormone Treatment
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!

Indexes: An Overview and Maintenance for Performance

By Desiree Harris

Indexes: An Overview and Maintenance for Performance by Desiree Harris

Many people know the importance of creating indexes on SQL Server database tables. Indexes greatly improve the performance of a database. However, while many people create indexes on their SQL Server tables, many people don't maintain them properly to ensure queries run efficiently as possible. I'll begin by giving a quick overview of how SQL Server 2000 stores data and how indexes improve performance. Then, I'll spend quite a bit of time explaining why, when, and how to maintain indexes with DBCC SHOWCONTIG and DBCC INDEXDEFRAG to ensure queries run in the most efficient manner.

SQL Server 2000 stores data into what is known as a heap. A heap is a collection of data pages containing rows for a table. The data isn't stored in any particular order and the data pages themselves aren't in any sequential order. The data is just there with no real form or organization. When SQL Server accesses data in this form, it does a table scan. This means SQL Server starts reading at the beginning of the table and scans every page until it finds the data that meets the criteria of the query. If a table is very large, this could greatly decrease the performance of queries.

Indexes will hasten the retrieval of data. It is important to understand how data is used, the types of queries being performed and the frequency of the queries that are typically performed when planning to create indexes. An index is far more efficient when the query results return a low percentage of rows and the selectivity is high. High selectivity means a query is written so it returns the lowest number of rows possible. As a rule, indexes should be created on columns that are commonly searched; this includes primary and foreign keys. It follows that columns that contain few unique values should never be indexed; this will increase the number of rows returned in a query.

There are two types of indexes to consider when planning: Non-Clustered and Clustered Indexes.

A non-clustered index stores data comparable to the index of a text book. The index is created in a different location than the actual data. The structure creates an index with a pointer that points to the actual location of the data. Non-clustered indexes should be created on columns where the selectivity of query ranges from highly selective to unique. These indexes are useful when providing multiple

ways to search data is desired.

A clustered index stores data similar to a phone directory where all people with the same last name are grouped together. SQL Server will quickly search a table with a clustered index while the index itself determines the sequence in which rows are stored in a table. Clustered indexes are useful for columns searched frequently for ranges of values, or are accessed in sorted order.

Each table can have only one clustered index, however up to 249 clustered indexes can be added per table. For more information on how Clustered and Non-Clustered indexes store data visit http://www.sql-server-performance.com/gv_index_data_structures.asp

While I could go on and on about how SQL Server 2000 stores and accesses data in a heap and in an

Index architecture, I will move on to discuss maintaining indexes with DBCC SHOWCONTIG and DBCC INDEXDEFRAG.

Once indexes have been created, it is important to maintain indexes to ensure the best possible performance. If indexes are not maintained, over time the data will become fragmented. Fragmentation is the inefficient use of pages within an index*. There are a number of tools available that will help with optimizing indexes to ensure they are running well, however I will only discuss DBCC SHOWCONTIG and DBCC INDEXDEFRAG in this article.

The DBCC SHOWCONTIG command will provide fragmentation information on data and indexes within a specified table and it will also determine if the data and index pages are full. If a page is full, SQL Server must split the page to make room for new rows. This statement should be run on heavily modified tables, tables that contain imported data, or tables that seem to cause poor query performance. When the statement is executed, here is what will be returned:

Statistic Description

Pages Scanned Number of pages in the table or index.

Extents Scanned Number of extents in the table or index.

Extent Switches Number of times the DBCC statement moved from one extent to another while it traversed the pages of the table or index.

Avg. Pages per Extent Number of pages per extent in the page chain.

Scan Density [Best Count: Actual Count] Best count is the ideal number of extent changes if everything is contiguously linked. Actual count is the actual number of extent changes. The number in scan density is 100 if everything is contiguous; if it is less than 100, some fragmentation exists. Scan density is a percentage.

Logical Scan Fragmentation Percentage of out-of-order pages returned from scanning the leaf pages of an index. This number is not relevant to heaps and text indexes. An out of order page is one for which the next page indicated in an IAM is a different page than the page pointed to by the next page pointer in the leaf page.

Extent Scan Fragmentation Percentage of out-of-order extents in scanning the leaf pages of an index.

This number is not relevant to heaps. An out-of-order extent is one for which the extent containing the current page for an index is not physically the next extent after the extent containing the previous page for an index.

Avg. Bytes free per page Average number of free bytes on the pages scanned. The higher the number, the less full the pages are. Lower numbers are better. This number is also affected by row size; a large row size can result in a higher number.

Avg. Page density (full) Average page density (as a percentage). This value takes into account row size, so it is a more accurate indication of how full your pages are. The higher the percentage, the better.

The DBCC INDEXDEFAG command will rebuild a specified index or all indexes for a specific table. This command also allows use of the fillfactor option which reduces the number of page splits per data or index page. Using the fillfactor option increases performance on insert and update statements. If a data page is full, SQL Server must split the page to make room for the new rows. The fillfactor allows specification of a percentage of space to leave available on the data pages for inserts and updates.

Let's observe an example:

Running the query on a table called member:

```
&nbsp;DBCC SHOWCONTIG (member) WITH ALL_INDEXES
```

DBCC SHOWCONTIG scanning 'member' table...

Table: 'member' (786101841); index ID: 2, database ID: 14

LEAF level scan performed.

```
- Pages Scanned.....: 192
- Extents Scanned.....: 26
- Extent Switches.....: 187
- Avg. Pages per Extent.....: 7.4
- Scan Density [Best Count:Actual Count].....: 12.77% [24:188]
- Logical Scan Fragmentation .....: 48.96%
- Extent Scan Fragmentation .....: 96.15%
- Avg. Bytes Free per Page.....: 6721.0
- Avg. Page Density (full).....: 16.96%
```

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

Running the query on a table named provider:

```
&nbsp;DBCC SHOWCONTIG (provider) WITH ALL_INDEXES
```

DBCC SHOWCONTIG scanning 'provider' table...

Table: 'provider' (850102069); index ID: 2, database ID: 14

LEAF level scan performed.

```
- Pages Scanned.....: 3
- Extents Scanned.....: 1
- Extent Switches.....: 0
- Avg. Pages per Extent.....: 3.0
- Scan Density [Best Count:Actual Count].....: 100.00% [1:1]
- Logical Scan Fragmentation .....: 33.33%
- Extent Scan Fragmentation .....: 0.00%
```

Indexes: An Overview and Maintenance for Performance

– Avg. Bytes Free per Page.....: 5596.0
– Avg. Page Density (full).....: 30.86%

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

There are a few specific things to take note of and will help determine if index pages are full or if they are heavily fragmented.

The fullness of the index pages can be determined by reading the "Avg. Bytes free per page" and "Avg. Page density (full)" statistics. The "Avg. Bytes free per page" figure should be low and the "Avg. Page density (full)" figure should be high. You'll notice that both tables likely have very full pages.

The fragmentation level of an index can be determined by comparing the values of "Extent Switches" and "Extents Scanned" and having a clear understanding "Logical Scan Fragmentation" and "Extent Scan Fragmentation" values. The "Extent Switches" should be almost equal to "Extents Scanned." Based on the examples above, this is the way it should look. "Logical Scan Fragmentation" and "Extent Scan Fragmentation" values give a good indication of a table's fragmentation level. These values should be as close to zero as possible (10% may be acceptable). The 'member' table is highly fragmented and the provider table is slightly fragmented based on the numbers above.

These issues can be corrected by dropping and recreating a clustered index with the FILLFACTOR option specified. Also, the DBCC INDEXDEFRAG command will compact an index, taking into account its FILLFACTOR, which will improve the statistics.

After running the queries below:

```
&nbsp;DBCC DBREINDEX (member, ", 80)  
&nbsp;DBCC DBREINDEX (provider, ", 80)
```

Running the queries:

```
&nbsp;DBCC SHOWCONTIG (member) WITH ALL_INDEXES  
&nbsp;DBCC SHOWCONTIG (provider) WITH ALL_INDEXES
```

DBCC SHOWCONTIG scanning 'member' table...

Table: 'member' (786101841); index ID: 2, database ID: 14

LEAF level scan performed.

– Pages Scanned.....: 41
– Extents Scanned.....: 6
– Extent Switches.....: 5
– Avg. Pages per Extent.....: 6.8
– Scan Density [Best Count:Actual Count].....: 100.00% [6:6]
– Logical Scan Fragmentation: 0.00%
– Extent Scan Fragmentation: 0.00%
– Avg. Bytes Free per Page.....: 1657.0
– Avg. Page Density (full).....: 79.53%

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

DBCC SHOWCONTIG scanning 'provider' table...

Table: 'provider' (850102069); index ID: 2, database ID: 14

LEAF level scan performed.

- Pages Scanned.....: 2
- Extents Scanned.....: 2
- Extent Switches.....: 1
- Avg. Pages per Extent.....: 1.0
- Scan Density [Best Count:Actual Count].....: 50.00% [1:2]
- Logical Scan Fragmentation: 0.00%
- Extent Scan Fragmentation: 0.00%
- Avg. Bytes Free per Page.....: 4346.0
- Avg. Page Density (full).....: 46.31%

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

As a result of running the DBCC SHOWCONTIG and DBCC INDEXDEFRAG commands, we were able to diagnose and greatly reduce fragmentation on the 'member' and 'provider' tables. The member table is almost perfect and the 'provider' table shows great improvement. This will result in an extraordinary performance increase on queries that are run against these tables.

*For more information on fragmentation, visit this link:

www.sql-server-performance.com/

Desiree Harris is a support specialist with ORCS Web, Inc. – a company that provides managed hosting services for clients who develop and deploy their applications on Microsoft Windows platforms.

Desiree Harris is a support specialist with ORCS Web, Inc. – a company that provides managed

hosting services for clients who develop and deploy their applications on Microsoft Windows platforms.

The Right Mutual Funds For Baby Boomers

By C.C. Collins

The Right Mutual Funds For Baby Boomers by C.C. Collins

The Right Mutual Funds For Baby Boomers By C.C. Collins, Wealth Strategist,
<http://wealthscientist.com>

If you are a baby boomer, time is not on your side. Many baby boomers see retirement age fast approaching with little to nothing in the way of retirement assets that will allow them to actually retire and live a comfortable lifestyle.

With the benefit of time in short supply, substantial investment performance in a shorter than normal time frame becomes strikingly important.

Indexes: An Overview and Maintenance for Performance

Mutual Fund Advice A case could be made that a special type of mutual fund, an index mutual fund, in conjunction with careful market trend analysis (not predictive market timing) could be used to achieve higher returns faster than a standard mutual fund.

As to the specific type of index fund to consider using, investors would do well to "keep it simple" and use an index fund that tracks well known indexes like the S&P 500, Nasdaq100, and Wilshire 2000.

Index funds that track any of the major indexes are just taking advantage of the concept of diversification. The only remaining risk is whether the entire market goes up or goes down and one can switch to a fund that is designed to profit from a down market when such action is called for.

There are very few active investment managers that outperform index funds or exchange traded funds over a five year or greater period. This is why an index fund is recommended in the case of baby boomer-aged investors who need stellar performance over shorter time frames.

Mutual Fund Selection

Mutual Fund Action plan

Mutual Fund Research

Mutual Fund Investment tools

C.C. Collins is a Financial Planning Advisor and Author of "Scientific Wealth Strategies" at <http://wealthscientist.com>. Find more information at <http://networthpublishing.com>



This Free E-Book has been brought to you by Natural-Aging.com.

[100% Effective Natural Hormone Treatment](#)
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!