

Database Maintenance Best Practices

Written By: Denny Cherry

Senior Database Administrator / Architect

Awareness Technologies

Microsoft MVP, MCDBA, MCSA, MCTS, MCITP

Published: September 8, 2009

What is Database Maintenance? 3
 Indexes 3
 Statistics 3
Why is Database Maintenance Required? 5
 Indexes 5
 Statistics 5
How do I know if I need to perform database maintenance? 7
 Indexes 7
 Statistics 7
What needs to be done?..... 9
 Index Rebuilding or Defragmenting? 9
 Statistics 10
Why don't vendors set up database maintenance automatically? 11
Why doesn't the database engine do database maintenance automatically? 12
Warranty 13

What is Database Maintenance?

Database maintenance is the process by which indexes and internal statistics are rebuilt and updated so that the SQL Server is performing at its peak performance.

Indexes

Indexes are presorted copies of specific columns of a specific table. The indexes are used to increase the speed by which the database engine can find the row or rows which were requested by the query which is being run. A single table can have multiple indexes created on it. These multiple indexes can contain the same columns, or different columns depending on the need of the queries which will be run against the database. As data is added and removed from the table, the data is also added and removed from the indexes. As data is added to the index, page splits and index fragmentation occur.¹

When data is written by the database engine to the hard drive it is written in 8k pages (8192 bytes) which are contained within 64k blocks (65536 bytes). As each 8k page is filled the database engine must begin writing to the next page. If another record needs to be inserted into a page which is already full then a page split occurs and the SQL Server must move a record from the full page to a new page in order to make room for the values which must be inserted. This can be offset by setting the FILLFACTOR setting when the index is created. In the case of a vendor provided database driven application (such as Awareness Technologies Sonar and Interguard products) the application vendor should have set the FILLFACTOR when creating the indexes for you based on the amount of data which is expected to be inserted into the index². While the FILLFACTOR can be changed it is usually best to not adjust this setting without instruction from the application vendor.

The FILLFACTOR setting tells the SQL Server how much free space to leave when initially creating the index, or when rebuilding the index. The database engine does not ensure that this amount of free space is left available during normal day to day row insert, update and delete operations.

As data is deleted from the tables, the same data is automatically removed from the index. This process leaves database pages on the disk with little to no data in the pages, but these pages are still allocated to the index to which they were initially assigned.

Statistics

Database statistics are internal metrics which the SQL Server uses to track how much data there is in each table and index within the database, and how that data is spread across the database table. This is done by reading a sample of the database table as needed. This automatic sampling of data is however not done on a regular interval as this could place undue stress on the database engine. When the amount of data in the table increases by 20% over the last time the statistics were updated the database

¹ You can read more about what indexes are and how they are used at <http://itke.techtarget.com/sql-server/back-to-basics-what-are-indexes-and-what-are-they-used-for/>

² In the case of the Sonar database which is included when the Sonar and/or Interguard products are installed on a Customer's server at their site, each index has been created with a specific fill factor to maximize performance with minimal database maintenance. While customers may change these settings as needed, Awareness Technologies cannot be held responsible for performance issues which may arise from changes made to these FILLFACTOR settings.

engine automatically updates the statistics for that table. However as tables grow larger and larger updating the statistics may need to be updated more often than that to prevent performance issues.

Why is Database Maintenance Required?

When SQL Server is operating normally, it does its best to handle the database maintenance operations itself. However over time the database engine needs assistance to do more major database maintenance operations. These operations can be configured to run automatically on a schedule so that they can be performed during off hours as to minimize the effect to end users of the application which the database serves as the data repository for.

Without performing this database maintenance users will see all sorts of slowdowns in their applications. This includes slow queries, timeouts, screens which load slowly, etc.³

Indexes

Rebuilding or defragging the indexes will decrease the amount of time taken when inserting data, as page splits (see above) require a greater amount of time to process than a regular index insert operation (which occurs as part of the regular data insert operation). By reducing the amount of empty or almost empty data pages which the index contains you will reduce the number of data pages which must be read from the disk when loading the index into memory. By reducing the number of pages which must be loaded to load the index into memory you decrease the amount of time it takes, you reduce the amount of space the index takes both on disk and in memory, and you reduce the number of pages in memory that the queries must read in order to find the row or rows they are looking for.

Statistics

Table and Index Statistics automatically update, so long as the “Auto Create Statistics” and “Auto Update Statistics” options are enabled in the database properties as shown in Figure 1.

While the database engine will do its best to keep these statistics up to date, the engine is programmed with specific thresholds which must be reached before the automatic statistics update logic is triggered. The trigger for this operation is that the table must have 20% more records than it had the last time the statistics for the table were updated. For smaller tables this 20% change requirement is fine, however as the table grows this requirement of 20% change takes longer and longer to achieve. For a table with 10000 rows in it a 20% change rate means adding in 2000, however for a table which

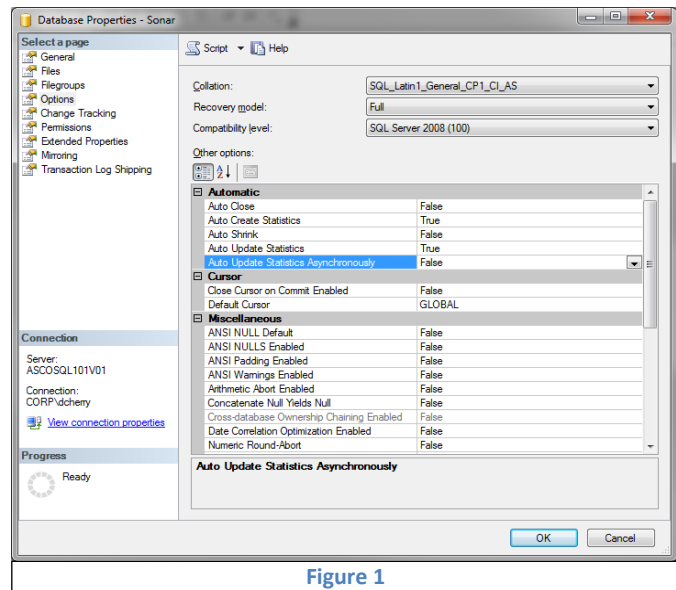


Figure 1

³ For example when using our Sonar or Interguard products on your own server without performing this database maintenance on the Sonar database timeouts will be experienced when viewing recorded data in the data grid.

has 10,000,000 rows in it a 20% change rate means adding in 2,000,000 rows which can take several days to complete at the minimum. There is no way to change this setting from 20% to a smaller number.

Now the catch to the update statistics logic is that if you are deleting data from the tables, you will never get to the next trigger point to update statistics, or at least it will take a much longer amount of time before you reach that point. This is because the logic doesn't count all inserts to the table, is only counts the number of rows in the table⁴. So as you remove data the statistics actually become more and more out of date, eventually with no hope of ever catching up unless you have a radical data growth.

⁴ By default the Sonar database which comes with our Sonar and Interguard products have the data deletion functions disabled. They can be enabled by the customer by simply enabling the database job and changing the setting within the settings table. A future release of the product will provide a settings screen within the UI to enable and disable the data deletion, as well as control how much data is being deleted.

How do I know if I need to perform database maintenance?

If you have a database, it needs regular database maintenance performed on it. There really is no “if” involved here. Depending on the size of the database, and how quickly your data grows it may take time (a very long time in some cases) before you begin to see problems, but eventually you will see problems.

Back when SQL Server 7 and SQL Server 2000 were release the “Microsoft Marketing Machine” loved to advertise that SQL Server was now a no maintenance database platform. This wasn’t ever the case, and shouldn’t ever be thought to be the case. While it may have helped sell SQL Server licenses in the short term, it didn’t to a whole to bolster faith in the product in the long run.

Indexes

When your indexes need to have maintenance performed on them you will see an increase load on the disks which can be seen via performance monitor. This will require that you have a baseline set of numbers to compare your current real time performance monitor data to. As each environment and database is different we can’t provide sample load numbers that would be within an acceptable range. Numbers which are fine for one database may be completely unacceptable for another environment simply due to system design, available resources, database size, hard drive speed, as well as many other factors.

However high disk load numbers in performance monitor don’t necessarily mean that your indexes need rebuilding of defragmenting, it can also mean that your database load has grown beyond the performance capacity of your storage subsystem of your database server, however index tuning and maintenance is a good first step to take as index tuning and maintenance is much less expensive than a hardware upgrade⁵.

If your indexes have become so fragmented across the data files, the SQL Server may be using other indexes than the ones expected to query the data, because the cost of searching through a less than optimal index is less than loading an index which is mostly empty (and therefore has many extra data pages than needed) from disk. This can be identified by comparing the used execution plan with a known good execution plan.

Statistics

It is much trickier to determine if the statistics are too far out of date. The first indicator will be that queries are taking much longer than expected to take⁶. However when you look at the execution plan, it looks the same as the known good plan.

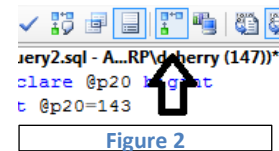


Figure 2

⁵ While index tuning is mentioned briefly in this document, proper tuning techniques are outside the scope of this document.

⁶ In the cases of some customer’s servers which I’ve looked into, queries before database maintenance was configured were taking 2-3 minutes to complete. After the database maintenance was completed the exact same query would complete with 1-2 seconds.

The next thing to look into will be the row estimates for each operation that the procedure is completing. The easiest way to do this is to capture the query using SQL Profiler, and then run the query in the SQL Server Management Studio (Query Analyzer for anyone who is still using SQL Server 2000⁷) with the “Include Actual Execution Plan” button depressed as shown in Figure 2. After running the query you can then navigate to the “Execution Plan” tab and mouse over each operator’s icon. When you place your mouse over the various operators you’ll get a popup which looks much like the one shown in Figure 3⁸. You know that you have a problem with your statistics when the Estimated Number of Rows is far less than the Actual Number of Rows. In many cases when the statistics are complete useless to the database engine, the estimated number of rows will show as 1, where the actual number of rows value will be much higher. In a perfect world we want these values to match, but in a highly dynamic database system this isn’t a likely situation to fall into as that would require almost constant updating of statistics on the database tables.

Index Seek (NonClustered)	
Scan a particular range of rows from a nonclustered index.	
Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Number of Rows	2387
Estimated I/O Cost	1.95127
Estimated CPU Cost	0.0767794
Number of Executions	1
Estimated Number of Executions	1
Estimated Operator Cost	2.02805 (70%)
Estimated Subtree Cost	2.02805
Estimated Number of Rows	69656.8
Estimated Row Size	35 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	19

Figure 3

If database statistics are out of date, you can also see incorrect execution plans being generated which will use less than optimal indexes to query for the data from the table much in the same way that indexes which require rebuilding or defragmentation can.

⁷ While our product is not supported on a SQL Server 2000 database, this whitepaper is not designed specifically to address issues which can arise with our Sonar or Interguard products but instead as a general discussion of database maintenance issues which can arise.

⁸ The image shown in Figure 3 is simply an example of good statistics being used to locate the data. The numbers which are shown are simply the numbers which were returned when a query was run against a sample database for the purposes of taking the screenshot. These are not target numbers, and should not be used in a comparison against actual numbers returned from your database system. In the case of this query, this Index Seek operation is one portion of a much larger and very complex query. The index shown in the example is built on a table with 223 million records in it at the time of query execution.

What needs to be done?

Databases need regular tuning in order to perform at their peak performance. This can be done by setting up database maintenance jobs to rebuild or defragment indexes and update statistics as needed.

Index Rebuilding or Defragmenting?

When you perform maintenance on the indexes on your tables you have two options. The first is to rebuild the index, and the second is to defragment the index. When you rebuild the index, the database engine effectively removes the entire index and recreates it from scratch. When you defragment the index the index is left in place, and the data is simply moved around the disks to get it into the optimal position.

When using the Enterprise Edition of SQL Server 2005 or higher you can rebuild indexes online⁹. This means that users will still be able to access the table while the index is being rebuilt. If you are not using the Enterprise Edition of SQL Server 2005 or higher you do not have the option of doing online index rebuilds and will have to use offline rebuilds which require that users not be able to access the table while it is being re-indexed. On smaller tables this lockout may only be a few milliseconds to a few seconds (depending on table size) however on large tables this lockout could take hours or days.

When using the online index rebuilding option on the Enterprise Edition of SQL Server 2005 and higher one thing to note is that the online rebuild will take longer than an offline rebuild. This is for a couple of different reasons. The first being that an offline index rebuild is a multi-threaded operation, while an online index rebuild is a single threaded operation. This means that when the offline operation is running several different threads within the database engine which are used to read data from the disk at the same time, sorting that data and writing the data to the disk, an in the online operation uses only a single thread which removes the speed improvement which is gained by the offline operation running several threads in parallel. The second reason is that users are actively using the system which can cause locking and blocking issues if the users run a long running query which locks the table or index causing the online index operation to stall as it is attempting to read the data which is being modified or viewed.

When you defragment an index the operation is completed online which allows users to access the table, no matter the edition and version of SQL Server that you are using. This defragmentation process however has a couple of drawbacks.

If your database is configured to store index data in more than one physical file within the same file group, the defragmentation process will not move data from one file to another. So if your index is more heavily stored in one data file, the defragmentation process will leave it unbalanced while the index rebuilding process will balance the data of the index between all the data files in the file group which the index has been built in.

⁹ Online index building and rebuilding were first introduced in SQL Server 2005, and up through SQL Server 2008 R2 require the Enterprise Edition of SQL server to use.

All things being equal the defragmentation will typically take longer than an index rebuild. This is because the SQL Server needs to read the index, figure out where to split the data in the page (if needed) then write the new pages. While an index rebuild simply reads the data table, sorts the data and begins writing the new index. However if only a small part of the index values have changed then a defragmentation will be quicker as the engine will only need to move data on a small percentage of the data pages within the index. Most typically I've found that if a table is over 25% fragmented then an index rebuild will be a faster operation than an index defragmentation¹⁰.

A large advantage of the database defragmentation process over the index rebuilding process is that if a defragmentation process is stopped for some reason when it next starts up it will resume about where it left off. However if an index rebuild is stopped and restarted, the entire operation is restarted. This also means that the index rebuild requires more space in the transaction log and the database files to handle the rebuild than the index defragmentation does.

Statistics

There is only one operation that can be performed against statistics and that is to update the statistics. However there are a couple of options which can be used. You can update the statistics using a sampling of data from the table, or a full scan. The sampling of data is a much quicker operation, because the database engine only scans a sample of the values in the column or columns which make up the statistic, and as such it is not as accurate. A full scan will take much longer as the SQL Server must read every value in the column or columns which make up the statistic before it can write the values to the statistic.

Updating of statistics is an online operation on all versions and editions of SQL Server going back to at least SQL Server 6.5 if not earlier.

If you perform index rebuilds as part of your database maintenance (see above), you will only need to update some of the statistics in your database as the statistics which go with the indexes which you rebuild will be automatically updated when the index is rebuilt. If you only perform index defragmentation you will need to update your statistics as well as the index defragmentation operation does not update the statistics of the index.

¹⁰ The 25% fragmentation number is only an example. Each table and database will be different depending on several factors including disk speed, system load, memory amount, etc. You should perform testing on your system to see exactly where the balance point is that you should switch from defragmenting indexes to rebuilding them.

Why don't vendors set up database maintenance automatically?

Some vendors choose to, other do not. Generally the reason to not setup database maintenance is the vendor would have to guess as to the schedule that the database maintenance needs to be performed at. This includes the frequency as well as the time to begin the database maintenance. Because the vendor doesn't want to trigger database maintenance to begin running to late in the night, and have the maintenance continue running during the business day when it would adversely affect the performance of the application which uses the database. They usually choose not to setup maintenance automatically instead leaving that task to the customers Database Administrator or as a professional services engagement so that the database maintenance tasks can be tuned to the specific requirements of the customers environment.

Why doesn't the database engine do database maintenance automatically?

Microsoft choose to make database maintenance a manual process for much the same reason that vendors typically don't like to setup the database maintenance automatically. There's no way to know when the best time to run the maintenance is, or what kind of maintenance needs to be run.

Some databases require that the maintenance be done in the middle of the night, while some require that the maintenance be done in the middle of the afternoon. It all depends on what the system is and where the server is located compared to where the users of the system are. For example a server located in Los Angeles, CA wouldn't want to do database maintenance in the middle of the night if the bulk of the customers were in Australia as that would put the database maintenance in the middle of the business day for the users, even though it was the middle of the night for the server.

Warranty

The information provided in this document is provided free of charge. The author (Denny Cherry) and employer (Awareness Technologies) make no warranty against the information provided in this document. Use of the information provided in this document is at your own risk. Should performance issues arise with your database after using the information provided in this document you should contact a local database professional.

Denny Cherry and Awareness Technologies are not liable for any performance issues, broken systems, lost revenue, or any other consequence of using the information provided in this document.