Database Maintenance for Microsoft® SharePoint® 2010 Products

Authors:

Bill Baer

Bryan Porter

Technical Reviewer:

Paul S. Randal (SQLskills.com)

Published:

July 2011

Summary:

This paper describes the recommended maintenance strategies for the databases that host content and configuration settings for SharePoint 2010 Products.

Abstract

This white paper provides information and guidelines for maintaining the databases that host Microsoft® SharePoint® 2010 data and configurations. It describes and provides examples of the database maintenance tasks that we recommend when using SharePoint 2010.

Before you implement any database maintenance tasks or modify SharePoint 2010 databases, read the following support article: Support for changes to the databases that are used by Office server products and by Windows SharePoint Services (http://go.microsoft.com/fwlink/?LinkId=110812&clcid=0x409).

Table of Contents

To comment on this paper or request more documentation about these features, contact **SharePoint IT Docs** (itspdocs@microsoft.com).

Introduction

Routine database maintenance is essential for the smooth operation of Microsoft® SharePoint® 2010 databases. This white paper describes the database maintenance tasks supported for SharePoint 2010.

The recommended maintenance tasks for SharePoint 2010 databases include:

- Checking database integrity.
- Defragmenting indexes by either reorganizing them or rebuilding them.
- Setting the fill factor for a server.

Note: This article discusses database maintenance and not planning for capacity or performance. For information about capacity or capacity planning, see Storage and SQL Server capacity planning and configuration (SharePoint Server 2010) (http://go.microsoft.com/fwlink/?LinkId=217482).

Although previous versions of SharePoint Products and Technologies required manual intervention to perform index defragmentation and statistics maintenance, SharePoint 2010 automates this process for its databases. This is accomplished by several SharePoint Health Analyzer rules. These rules evaluate the health of database indexes and statistics daily, and will automatically address these items for these databases:

- Configuration Databases
- Content Databases
- User Profile Service Application Profile Databases
- User Profile Service Application Social Databases
- Web Analytics Service Application Reporting Databases
- Web Analytics Service Application Staging Databases
- Word Automation Services Databases

Database maintenance tasks can be also performed by either executing Transact-SQL commands, or running the Database Maintenance Wizard. We will initially present the Transact-SQL commands that you can use, and then explain how to create database maintenance plans by using the Microsoft SQL Server® Database Maintenance Wizard.

Note: In this paper, we present detailed examples only for SQL Server 2008 R2 and SQL Server 2005.

Check for consistency errors by using DBCC CHECKDB

Start your routine maintenance operations with consistency checks to ensure that your data and indexes are not corrupted. You can use the DBCC (Database Console Command) CHECKDB statement to perform an internal consistency check of the data and index pages.

The vast majority of database consistency problems are caused by I/O subsystem errors. However, database consistency may be affected when a database server is improperly shut down or a drive fails. Noticeable performance and availability issues can sometimes be symptoms of underlying database consistency problems. Database consistency checks should be performed at least once per week on your SharePoint 2010 databases, and whenever events such as database server or I/O subsystem failures occur.

About DBCC CHECKDB

DBCC CHECKDB checks the logical and physical integrity of all the objects in the specified database by performing the following operations:

- Runs the equivalent of DBCC CHECKALLOC (http://go.microsoft.com/fwlink/?LinkId=110815&clcid=0x409) to verify the allocation structures in the database.
- Runs the equivalent of DBCC CHECKTABLE (http://go.microsoft.com/fwlink/?LinkId=162093) on every table and view in the database to verify their logical and physical integrity.
- Runs the equivalent of DBCC CHECKCATALOG (http://go.microsoft.com/fwlink/?LinkId=110834&clcid=0x409) on the database to verify its metadata consistency.

This means that the DBCC CHECKALLOC, DBCC CHECKTABLE, or DBCC CHECKCATALOG commands do not have to be run separately from DBCC CHECKDB. We recommend that you run DBCC CHECKDB rather than the individual operations because it identifies the widest range of possible errors and is therefore safer to run in a production environment.

DBCC CHECKDB is very resource-intensive in terms of memory, I/O, and CPU. An alternative to running DBCC CHECKDB on your production system is to run it on a restored backup of your SharePoint databases on a different server, thus offloading the consistency-checking workload from the production system.

We recommend that you first run DBCC CHECKDB, and then, if it reveals errors, restore the affected database using your most recent backups.

Important - Running DBCC CHECKDB WITH REPAIR_ALLOW_DATA_LOSS is not supported. However, running DBCC_CHECKDB WITH REPAIR_FAST and REPAIR_REBUILD is supported, as these commands only update the indexes of the associated database.

The following table contains sample output from DBCC CHECKDB.

```
DBCC results for 'Contoso_Content_1'.
Service Broker Msg 9675, State 1: Message Types analyzed: 14.
Service Broker Msg 9676, State 1: Service Contracts analyzed: 6.
Service Broker Msg 9667, State 1: Services analyzed: 3.
Service Broker Msg 9668, State 1: Service Queues analyzed: 3.
Service Broker Msg 9669, State 1: Conversation Endpoints analyzed: 0.
Service Broker Msg 9674, State 1: Conversation Groups analyzed: 0.
Service Broker Msg 9670, State 1: Remote Service Bindings analyzed:
0.
DBCC results for 'sys.sysrowsetcolumns'.
There are 2663 rows in 21 pages for object "sys.sysrowsetcolumns".
DBCC results for 'sys.sysrowsets'.
There are 309 rows in 4 pages for object "sys.sysrowsets".
```

```
…more


CHECKDB found 0 allocation errors and 0 consistency errors in
database 'Contoso_Content_1'.

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

Table 1. DBCC CHECKDB Sample Output

For more information about using DBCC CHECKDB with SQL Server 2008, see DBCC CHECKDB (Transact-SQL) (http://go.microsoft.com/fwlink/?LinkId=217483).

DBCC CHECKDB and performance

We recommend that you run consistency checks during non-production hours, because DBCC CHECKDB is extremely resource-intensive (in terms of I/O, CPU, memory, and tempdb space). There is a common misconception that DBCC CHECKDB acquires blocking locks; this has not been the case since before SQL Server 2000. For more information about DBCC CHECKDB not acquiring blocking locks, see "A SQL Server DBA myth a day: (2/30) DBCC CHECKDB causes blocking" (http://www.sqlskills.com/BLOGS/PAUL/post/A-SQL-Server-DBA-myth-a-day-(230)-DBCC-CHECKDB-causes-blocking.aspx).

You may find that the resource overhead of running DBCC CHECKDB is too high for your production system. In that case, do not attempt to run consistency checks one table at a time as this will be more problematic overall. The best ways to reduce the integrity-checking overhead on the production system is to do one of the following:

- Use the WITH PHYSICAL_ONLY option to reduce the CPU and memory usage.
- Restore a database backup on a separate SQL Server and run consistency checks on the restored copy of the database.

For more information about these options, see this blog post by Paul S. Randal.

Measure and reduce index fragmentation

Index fragmentation occurs when the logical order of pages in a table or index (as defined by the index key) is not the same as the physical order of the pages in the data files. It can also mean that the data density on data file pages is low, resulting in wasted disk space, memory, and I/Os. Index fragmentation can be the result of many inserts, updates, or deletes to a table. The following figures illustrate a newly built, non-fragmented index and then a fragmented index after many inserts, updates, and deletes. The red arrow shows the physical order of the index and the black arrows show the logical ordering of the index pages.

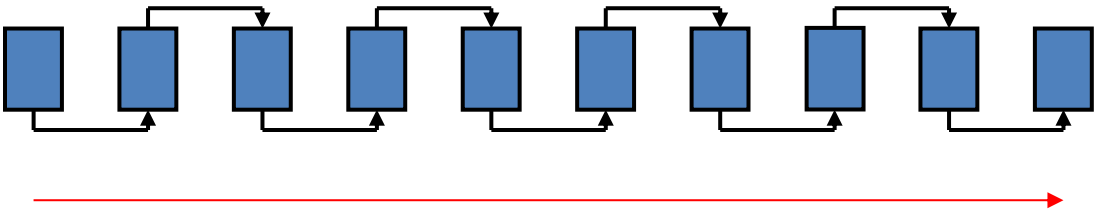


Figure 1. Non-fragmented index (Image source: Paul S. Randal)

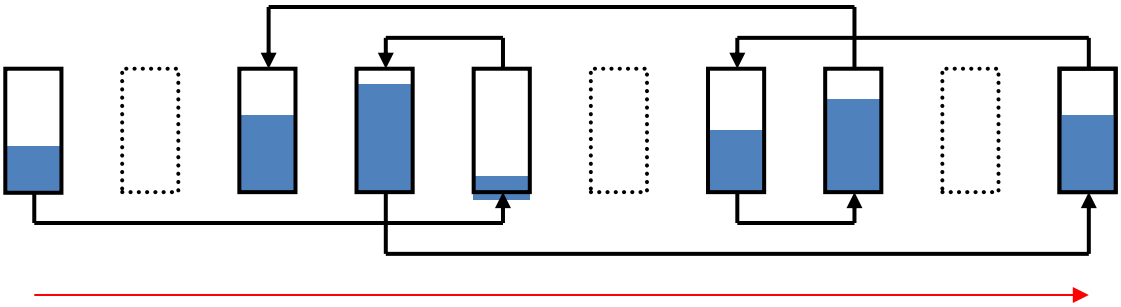


Figure 2. Fragmented Index (Image source: Paul S. Randal)

Because inserts, updates, and deletes are not distributed equally among the rows of the table and indexes, the fullness (or data density) of each page can vary over time. For queries that scan part or all of the indexes of a table, fragmentation can cause additional page reads, which hinders parallel scanning of data and can significantly affect search performance.

Index fragmentation can result in performance degradation and inefficient space utilization, and indexes may become quickly fragmented on even moderately used databases.

Before implementing an index fragmentation maintenance plan, you need to understand which tables and indexes are most fragmented and then create a maintenance plan to rebuild or reorganize those indexes.

In SharePoint 2010, an example of a table that often becomes fragmented is **AllDocs**, which contains document libraries, their associated documents and lists and list items, and their respective metadata.

The fragmentation level of an index is the percentage of index pages that are not in the same logical and physical order.

Online vs. offline index rebuilds

Online index rebuilding is only available in SQL Server Enterprise, Developer, and Evaluation editions. The methods outlined in this white paper account for this. The procedures shown will fall back to an offline index rebuild if the edition of SQL Server that is hosting a specific database does not support online index rebuilds, or if the index that is being rebuilt is not eligible for an online index rebuild. An index might not be eligible for an online rebuild due to the presence of LOB (large object) columns, such as columns with a data type of NVARCHAR(MAX), IMAGE, etc.

For information about online index rebuilds, see How Online Index Operations Work (http://go.microsoft.com/fwlink/?LinkId=217492). When an offline index rebuild is performed, table level locks will be taken during the rebuild process. This may prevent the table from being written to or even accessed at all. Many of the indexes in SharePoint databases will always be rebuilt using an offline index rebuild due to the presence of LOB columns.

Even if online index rebuild is used, there are still two points in the operation where table locks are held momentarily, and these could cause blocking. As a result, we recommend that you always schedule index rebuild activities during periods of low activity.

Measure fragmentation in a SQL Server 2008 or 2005 database (sys.dm_db_index_physical_stats)

In SQL Server 2008 or SQL Server 2005, use the **sys.dm_db_index_physical_stats** dynamic management view to determine fragmentation for the indexes on a specified table or view.

For measuring fragmentation, we recommend that you monitor the column **avg_fragmentation_in_percent**. The value for **avg_fragmentation_in_percent** should be as

close to zero as possible for maximum performance. However, values from 0 percent through 10 percent may be acceptable. For more information see sys.dm_db_index_physical_stats (http://go.microsoft.com/fwlink/?LinkId=110839&clcid=0x409).

Table 2 shows sample results from **sys.dm_db_index_physical_stats**, with a value of 9.375 for **avg_fragmentation_in_percent** in one row.

database_id	index_type_desc	alloc_unit_type_desc	avg_fragmentation_in_percent
10	CLUSTERED INDEX	IN_ROW_DATA	0
10	NONCLUSTERED INDEX	IN_ROW_DATA	0
10	NONCLUSTERED INDEX	IN_ROW_DATA	0
10	CLUSTERED INDEX	IN_ROW_DATA	0
10	NONCLUSTERED INDEX	IN_ROW_DATA	0
10	CLUSTERED INDEX	IN_ROW_DATA	9.375

Table 2. Sample results from sys.dm_db_index_physical_stats

To use the **sys.dm_db_index_physical_stats** dynamic management view

1. On the taskbar, click **Start**, point to **All Programs**, point to **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.

   To use **sys.dm_db_index_physical_stats** with a database object, you must know the database ID and object ID.

2. Select the content database in the Object Explorer, and then click **New Query**. Execute the following script.

   ```
   SELECT DB_ID() AS [Database ID];
   ```

   Note: When using DB_ID without specifying a database name, the compatibility level of the current database must be 100 (a SQL Server 2008 database) or 90 (a SQL Server 2005 database). If you have upgraded from a previous version of SQL Server, you must specify a database name in the DB_ID statement. For more information about compatibility levels, see sp_dbcmptlevel (Transact-SQL) (http://go.microsoft.com/fwlink/?LinkId=110840&clcid=0x409).

3. Execute **sys.dm_db_index_physical_stats** on the database or object you have selected. You can specify not only the database, but a table or index.

   Syntax:

   ```
   sys.dm_db_index_physical_stats (
       { database_id | NULL | 0 | DEFAULT }
       , { object_id | NULL | 0 | DEFAULT }
       , { index_id | NULL | 0 | -1 | DEFAULT }
       , { partition_number | NULL | 0 | DEFAULT }
   ```

```
        , { mode | NULL | DEFAULT }
    )
```

You should be careful when using the sys.dm_db_index_physical_stats DMV as it can be very resource-intensive. A comprehensive guide that explains the various ways to use it is at Inside sys.dm_db_index_physical_stats(http://www.sqlskills.com/BLOGS/PAUL/post/Inside-sysdm_db_index_physical_stats.aspx).

Reducing fragmentation for a database

To reduce the level of index fragmentation, follow the guidance below.

Run database maintenance Health Analyzer rules

SharePoint 2010 ships with the Health Analyzer rules framework. This rules framework is configured with a number of rules to monitor the health and well-being of a SharePoint environment and in some instances takes action to correct certain types of issues.

SharePoint 2010 ships with several rules pertinent to content database maintenance. There are rules that automatically reduce index fragmentation for some SharePoint databases, and rules that check for outdated statistics, updating them if necessary. These Health Analyzer rules replace the updated Database Statistics timer job introduced in Service Pack 2 for SharePoint Products and Technologies. By default, these rules are configured to execute on a schedule that varies from daily, weekly, to on-demand depending on the rule target.

All Health Analyzer rules that are configured to execute daily and associated with a particular SharePoint service are executed by the same timer job. Adjusting the scheduling of this timer job will adjust when Health Analyzer rules configured for daily execution and associated to that service will execute during the day. All rules discussed in this white paper are associated to the SharePoint Timer service.

Health Analyzer rules configured to execute on a different time interval (such as weekly) or associated with a different service have distinct timer jobs. Configuring a Health Analyzer rule to execute weekly would mean that that Health Analyzer rule will execute with the timer job configured to execute weekly for the specific service that the Health Analyzer rule is associated to. This execution will occur on whatever schedule has been defined for that timer job.

Health Analyzer rules may be run manually by selecting “Run Now” from the ribbon from within the Health Analyzer Rules page in Central Administration. Running these rules will cause the health of indexes and statistics to be evaluated, and cause index rebuilds and recalculations as appropriate.

Databases used by SharePoint have fragmented indices - When you run this rule, the following tasks are performed:

- The rule reports indexes as being fragmented. This is because evaluating index health is an expensive operation. As a result of the details of Health Analyzer rule execution, this rule will always reports indexes as being fragmented in order to trigger the corrective action.
- For each SharePoint database, the rule action looks for, and if found, executes the *proc_DefragmentIndices* stored procedure. During the execution of this stored procedure, a listing of all indexes within the database is built. Each index is evaluated as to its present level of fragmentation. Any indexes fragmented in excess of 30 percent are considered for rebuild.
- Assuming the edition of SQL Server supports online index rebuilds, an online index rebuild is attempted for each index. Should this fail, perhaps because the underlying

index does not support online rebuilds due to the use of LOB columns, an offline index rebuild will be performed.

As noted above, not every database in a SharePoint environment is serviced by this rule. Certain databases use different rules to perform similar maintenance activities.

Search – One or more property databases have fragmented indices - This rule maintains the indexes within the SharePoint 2010 Enterprise Search Property Databases. This rule is configured by default to execute weekly on any server in the farm. All processing for this rule – including corrective actions – occurs during the *Check* phase of rule execution. This means that if you want to manage index rebuilds for the Enterprise Search Property Database, it is not enough to simply configure this rule to not rebuild indexes automatically. You must disable the rule entirely in order to avoid the execution of index maintenance operations automatically by SharePoint 2010.

When you run the '*Search – One or more property databases have fragmented indices*' the following tasks are performed:

- The rule confirms that the environment is in a state in which performing an index rebuild is safe.
- For each property database configured for search applications within the local farm, the rule executes the *proc_MSS_DefragSearchIndexes* stored procedure. During the execution of this stored procedure, a listing of all indexes with average fragmentation in excess of 10% is built.
- Each index in the list that affects the performance of the Property database is rebuilt. If the edition of SQL Server supports online index rebuilds, an online index rebuild is performed. If an online index rebuild is attempted, but fails, the index will be rebuilt offline.

Search - One or more crawl databases may have fragmented indices - This rule maintains the indexes within the SharePoint 2010 Enterprise Search Crawl Databases. This rule is configured by default to only execute on demand. When executed, it will execute from any server in the farm.
This rule, when executed, will always report indexes in the crawl databases as being fragmented. This is due to the expensive nature of checking for fragmentation within a database. As a result, simply disabling the 'Repair' activity for this rule will result in all crawl databases being reported as unhealthy, even when the crawl databases have had their indexes recently rebuilt.

To manually manage the maintenance of indexes within crawl databases, you should disable the '*Search - One or more crawl databases may have fragmented indices*' rule in its entirety.

When you run the '*Search - One or more crawl databases may have fragmented indices*' the following tasks are performed:

- The rule confirms that the environment is in a state in which performing an index rebuild is safe.
- For each Crawl database configured for search applications within the local farm, the rule executes the *proc_MSS_DefragGathererIndexes* stored procedure.
- Each index within the Crawl database performance in the list is rebuilt. If the edition of SQL Server supports online index rebuilds, an online index rebuild is performed. If an online index rebuild is attempted, but fails, the index will be rebuilt offline.

It is important to note that the '*Search - One or more crawl databases may have fragmented indices*' rule will rebuild every index within all Crawl databases regardless of fragmentation level. It will also enable page level data compression, if supported by the edition of SQL Server that is hosting the Crawl database.

Due to the nature of the Crawl database, it is not anticipated that you will need to defragment this database frequently. Execute this rule after you have first performed a full crawl over your

content. Afterwards, monitor the indexes within the Crawl database for fragmentation, and execute this rule whenever index fragmentation grows. This may occur as a result of the sudden addition or removal of a large amount of crawled content – for example, during content expulsion as a result of environmental cleanup, or after the onboarding of a new content source, such as a file share or large SharePoint Web application.

The following databases do not have an automated mechanism in place for their maintenance. These databases are not anticipated to encounter a great deal of fragmentation. Monitor these databases for fragmentation, and rebuild indexes within these databases when fragmentation exceeds 30%.

- Search Administration Database
- Secure Store Database
- State Service Database
- Profile Sync Database
- Usage Database
- Managed Metadata Database
- Business Connectivity Services Database
- PerformancePoint Services Database

For more information about the changes that are supported for SharePoint 2010 databases, see Support for changes to the databases that are used by Office server products and by Windows SharePoint Services (http://go.microsoft.com/fwlink/?LinkId=110844&clcid=0x409) in the Microsoft Knowledge Base.

If the performance of a heavily fragmented database or table is not measurably improved by frequent defragmentation, you should check the performance of the I/O subsystem.

Reducing fragmentation for a specific table and its indexes

If you want to defragment an index associated with a particular table rather than an entire database, you can either reorganize or rebuild the index.

- Reorganizing an index specifies that the index leaf level will be reorganized. Index reorganization defragments and compacts clustered and non-clustered indexes on tables and views and can significantly improve index scanning performance. Reorganizing an index makes use of the existing space allocated to the index. Reorganization is always performed online, so that the underlying table is available to users.

- Rebuilding an index specifies that an entirely new copy of the index will be rebuilt. This means that a rebuild operation requires enough extra space to build the new copy of the index before removing the old, fragmented index. Rebuilding improves the performance of index scans and seeks. You can rebuild the index with a table either online or offline.

The fragmentation level of an index determines the method you should use to defragment it, and whether it can remain online, or should be taken offline. The following table describes the defragmentation method that is recommended for different fragmentation levels.

Fragmentation level	Defragmentation method
Up to 10%	Reorganize (online)
10-75%	Rebuild (online)
75%	Rebuild (offline)

Note: Using the DROP INDEX and CREATE INDEX commands is not supported on SharePoint 2010 databases.

You can reorganize and rebuild indexes by using the SQL Server 2008 or SQL Server 2005 ALTER INDEX statement, or the SQL Server 2008 or SQL Server 2005 Maintenance Plan Wizard. This paper presents only the SQL Server 2008 or SQL Server 2005 options in detail.

Using ALTER INDEX

ALTER INDEX permits a database administrator to perform maintenance operations against an index on a table or view. It can be used to disable, rebuild, and reorganize indexes. Optionally, it may be used to set options on the index. In most cases you can rebuild indexes while the database is online, which keeps the data more available than an offline index rebuild.

Important: SQL Server 2000 supported the use of DBCC DBREINDEX and DBCC INDEXDEFRAG for index maintenance. These commands have been deprecated from SQL Server 2005 onwards and will be removed in a future version of SQL Server. Do not use these commands to perform index maintenance on a SharePoint 2010 database.

Note: When an index is being rebuilt offline, a shared table lock is put on the table, preventing all operations with the exception of `SELECT` operations from being performed. SharePoint 2010 databases use clustered indexes specifically. When a clustered index is being rebuilt offline, an exclusive table lock is put on the table, preventing any table access by end users.

You can customize the following sample script to rebuild all indexes on a table.

```
USE Contoso_Content_1

GO

ALTER INDEX ALL ON [database_name. [ schema_name ] . | schema_name.
]table_or_view_name

REBUILD WITH (FILLFACTOR = 80, SORT_IN_TEMPDB = ON,

STATISTICS_NORECOMPUTE = ON)

GO
```

Fine tuning index performance by setting fill factor

Fill factor can be used to further improve index data storage and performance. When indexes are created or rebuilt, the fill factor value (1-100) determines the percentage of space on each leaf level page that can be filled with data. The remaining space is reserved for future growth. For many situations the default server-wide fill factor level of 0 is optimal (which means fill each page to 100% full). However, for SharePoint 2010, a server-wide setting of 80 is optimal to support growth and minimize fragmentation.

Note: We do not recommend that you set the fill factor for individual tables or indexes. Although this is the preferred method for non-SharePoint SQL Server databases, testing has shown that SharePoint databases work best with an 80% fill factor.

To view the fill factor value of one or more indexes, query the **sys.indexes** catalog view. For more information about the view, see sys.indexes (Transact-SQL) (http://go.microsoft.com/fwlink/?LinkId=110850&clcid=0x409).

To configure the server-wide fill factor value, use the **sp_configure** system stored procedure. For more information, see spconfigure (Transact-SQL) (http://go.microsoft.com/fwlink/?LinkId=110851&clcid=0x409).

Shrinking data files

In SQL Server 2008 and SQL Server 2005, you can shrink each file within a database (extensions .mdf, .ldf, and .ndf) to remove unused pages and recover disk space. SharePoint 2010 databases do not automatically shrink data files, although many activities create unused space in the database. Activities that can create unused space include running the Move-SPSite (http://technet.microsoft.com/en-us/library/ff607915.aspx) Windows PowerShell command, and deleting documents, document libraries, lists, list items, and sites.

Figure 3. Database Allocation

Free space is only released from the end of the file — for example, a content database file of 60 GB with a specified target size of 40 GB will free as much space as possible from the ending (conceptually 'right-hand' end) 20 GB of the database file. If used pages are included in the ending 20 GB, those pages will subsequently be relocated to the beginning 40 GB of the file that is retained. You can shrink database files individually or as a group.

Shrink operations should be executed rarely and only after executing an operation that removes a very large quantity of data from a database, and then only when that free space is not anticipated to be used again. Data file shrink operations cause heavy index fragmentation and are extremely resource-intensive. Examples of when it may be acceptable to shrink the database data files are the relocation of a large of number of site collections from one content database to another content database, or the deletion of a large list, either of which can create large amounts of unused space. Database files can only be reduced to the point where there is no free space remaining. Therefore, a content database in which content is infrequently deleted may see minimal benefit from shrinking, and will likely suffer performance consequences when that database needs to grow to accommodate additional data without specific accommodations. For more information, see Database File Initialization (http://msdn.microsoft.com/en-us/library/ms175935.aspx).

As shrinking causes index fragmentation, you should never shrink database files on a regular basis; databases should only be shrunk in response to large quantities of unused space appearing as a result of operations dramatically impacting the relative amount of used space within a database. If at all possible, shrinking a database should be avoided.

Use the following guidelines for shrinking databases:

- Do not auto-shrink databases or configure a maintenance plan that programmatically shrinks your databases.
- Shrink a database only when 50% or more of the content in it has been removed by user or administrator deletions and you do not anticipate the unused space being re-used by more data.
- We recommend that you shrink only content databases. The configuration database, Central Administration content database, and various service application databases do not usually undergo enough deletions to contain significant free space.

- Shrinking databases is an extremely resource-intensive operation. Therefore, if you absolutely must shrink a database, carefully consider when you schedule the shrink operation.

- After a database shrink operation, the indexes within that database are now fragmented. Address this fragmentation by using ALTER INDEX… REORGANIZE. If you are not configured to allow instant file initialization, you should shrink the database to a target size that accommodates any space needed for near term expected growth. For more information, see Database File Initialization (http://msdn.microsoft.com/en-us/library/ms175935.aspx). If you remove fragmentation by rebuilding the indexes, this will cause the database to grow again, leaving you with unused space.

Databases and database files can be shrunk manually to recover space by executing the DBCC SHRINKFILE and DBCC SHRINKDATABASE statements, by using SQL Server 2008 or SQL Server 2005 Management Studio.

For more information about why shrinking a database is detrimental to performance and should not be done unless absolutely necessary, see Why you should not shrink your data files (http://www.sqlskills.com/BLOGS/PAUL/post/Why-you-should-not-shrink-your-data-files.aspx).

Shrinking a database by using Transact-SQL commands

DBCC SHRINKDATABASE shrinks the data and log files for a specific database. To shrink individual files, use DBCC SHRINKFILE.

DBCC SHRINKDATABASE

Syntax:

```
DBCC SHRINKDATABASE
( 'database_name' | database_id | 0
     [ ,target_percent ]
     [ , { NOTRUNCATE | TRUNCATEONLY } ]
)
[ WITH NO_INFOMSGS ]
```

database_name **|** *database_id* **| 0** specifies the database name or ID. To select the current database, use 0.

target_percent is the free space in a percentage you want to retain after the database has been shrunk.

NOTRUNCATE compacts the data in data files by moving allocated pages from the end of a file to unallocated pages in the front of the file.

TRUNCATEONLY releases all free space at the end of the file to the operating system but does not perform any page movement inside the file.

Note: Using the TRUNCATEONLY option is not supported for SharePoint 2010 content databases.

For more information, see DBCC SHRINKDATABASE (Transact-SQL) (http://go.microsoft.com/fwlink/?LinkId=110852&clcid=0x409).

DBCC SHRINKFILE

Syntax:

```
DBCC SHRINKFILE
(
     { 'file_name' | file_id }
    { [ , EMPTYFILE ]
    | [ [ , target_size ] [ , { NOTRUNCATE | TRUNCATEONLY } ] ]
    }
)
[ WITH NO_INFOMSGS ]
```

file_name **|** *file_id* specifies the file name or ID.

EMPTYFILE Migrates all data from the specified file to other files in the same filegroup.

Note: Using the EMPTYFILE option is not supported for SharePoint 2010 database files.

target_size is the target size for the file in megabytes, expressed as an integer.

NOTRUNCATE compacts the data in data files by moving allocated pages from the end of a file to unallocated pages in the front of the file.

TRUNCATEONLY releases all free space at the end of the file to the operating system but does not perform any page movement inside the file.

Note: Using the TRUNCATEONLY option is not supported for SharePoint 2010 content databases.

For more information, see DBCC SHRINKFILE (Transact-SQL) (http://go.microsoft.com/fwlink/?LinkId=110853&clcid=0x409).

To shrink a database by using SQL Server 2008 Management Studio

1. On the taskbar, click **Start**, point to **All Programs**, point to **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.
2. In Object Explorer, connect to an instance of the SQL Server 2008 Database Engine and then expand that instance.
3. Expand **Databases**, right-click the database that you want to shrink, point to **Tasks**, point to **Shrink**, and then click **Files**.
4. Select the file type and file name.
5. Select **Reorganize files before releasing unused space.** You must also set the **Shrink file to** value. Selecting this option causes any unused space in the file to be released to the operating system and tries to relocate rows to unallocated pages.
6. Click **OK**.

Creating SQL Server 2008 maintenance plans

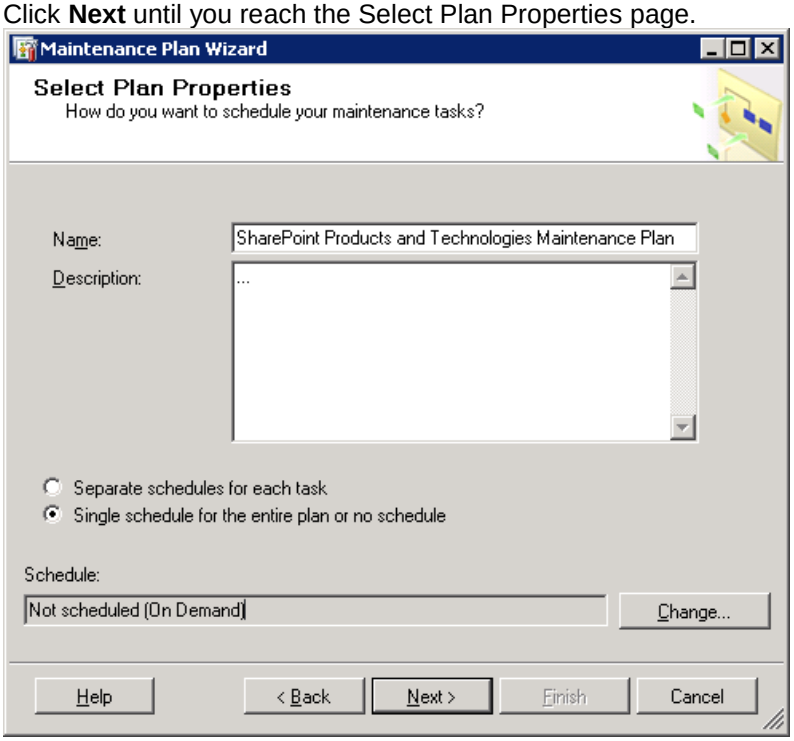
Many of the database maintenance operations covered in this white paper can be programmatically applied through the implementation of SQL Server maintenance plans. Maintenance plans can both automate and schedule essential tasks to protect your data. By

using maintenance plans in SQL Server 2008 or SQL Server 2005, an administrator can schedule such operations as running database consistency checks, and reorganizing or rebuilding indexes. For more information, see the following resources:

- Maintenance Plan Wizard (http://go.microsoft.com/fwlink/?LinkId=110855&clcid=0x409) for SQL Server 2008
- Database Maintenance Plan Wizard (http://go.microsoft.com/fwlink/?LinkId=217494) for SQL Server 2005

To configure a SQL Server 2008 database maintenance plan

1. On the taskbar, click **Start**, point to **All Programs**, point to **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.
2. In Object Explorer, connect to an instance of the SQL Server 2008 Database Engine and then expand that instance.
3. Click **Management**, right-click **Maintenance Plans**, and then click **Maintenance Plan Wizard**.
4. Click **Next** until you reach the Select Plan Properties page.



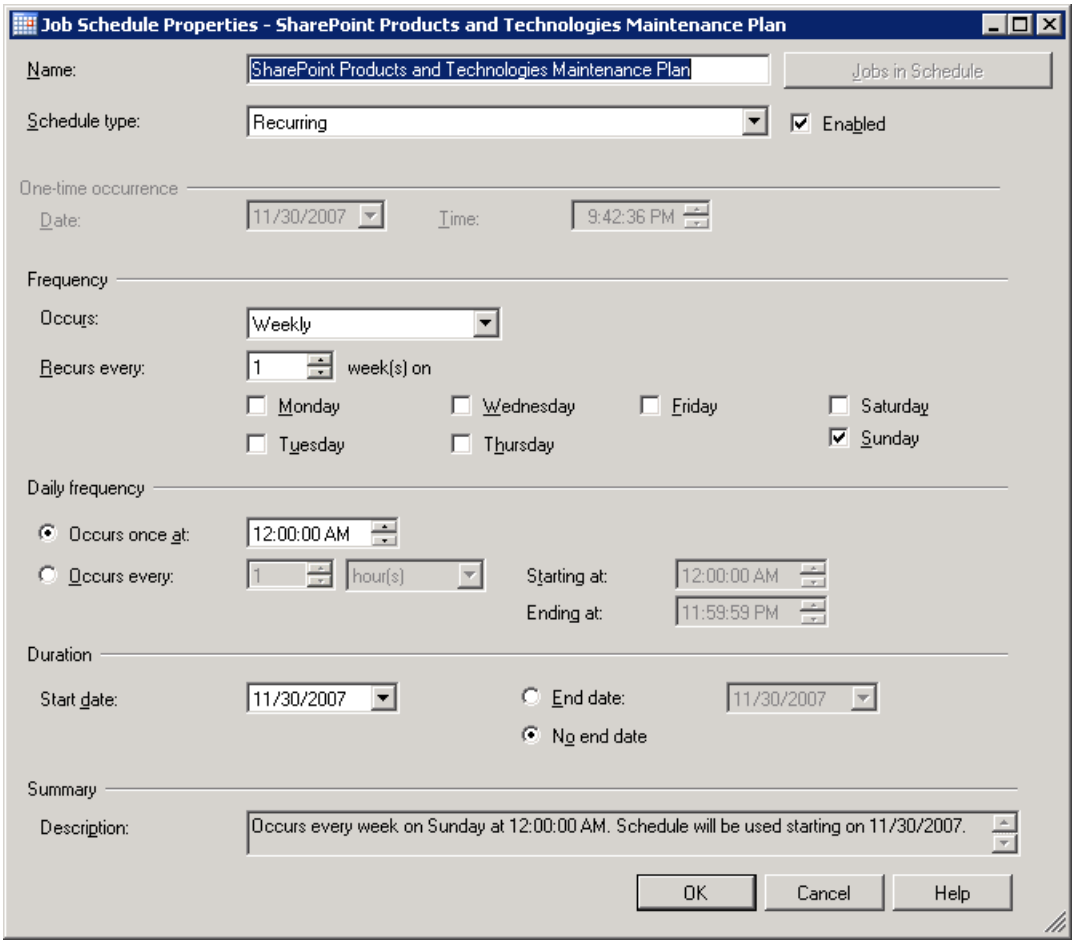
5. In the **Name** and **Description** fields, type a name and description.
6. Decide whether to configure one or more maintenance plans.
   - To configure a single maintenance plan, select **Single schedule for the entire plan or no schedule**.
   - To configure multiple maintenance plans with specific tasks, select **Separate schedules for each task**.

   If you have an environment with 10 or more content databases or more than 200 GB of content, we recommend that you configure separate maintenance plans to provide appropriate specificity and to maximize the maintenance window.

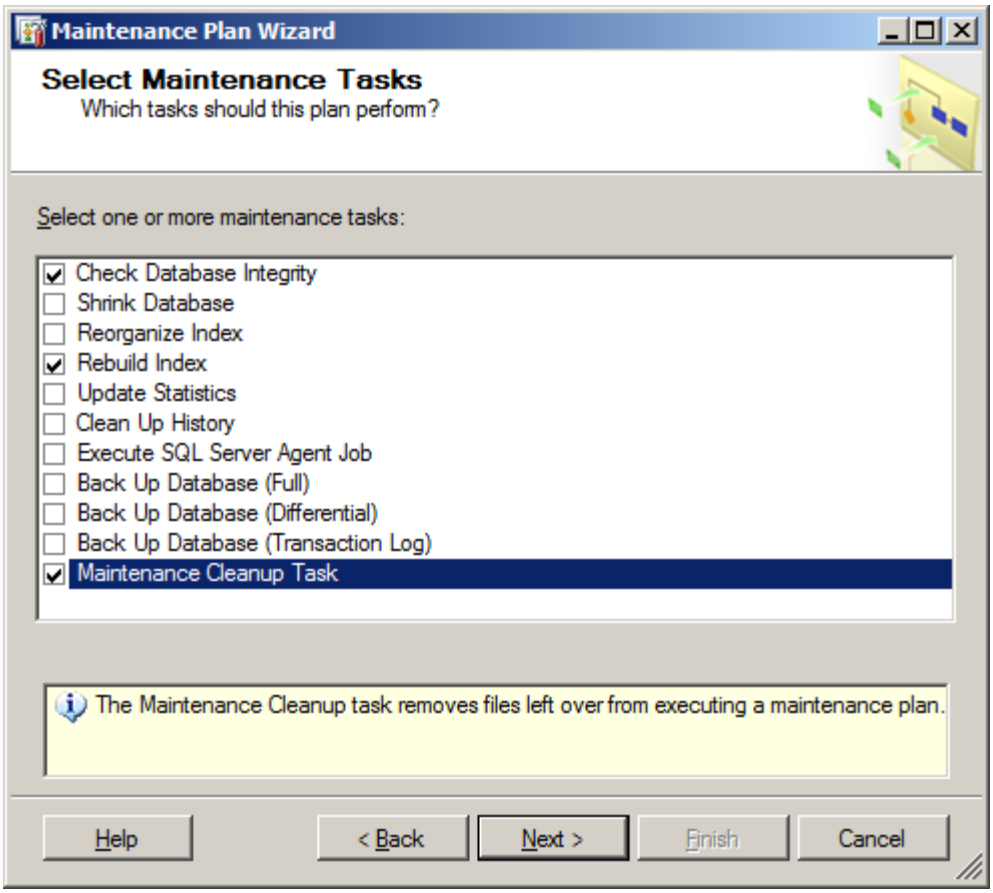
If you configure multiple maintenance plans for a database, specify a name or description that enables you to differentiate the plans and their purposes, including their schedules.

7. Click **Change** to set a schedule for one or more plans.

   The **Job Schedule Properties** dialog box appears.



8. Complete the schedule, click **OK**, and then click **Next**.
9. On the Select Maintenance Tasks page, select the maintenance tasks to include in the plan, and then click **Next**.

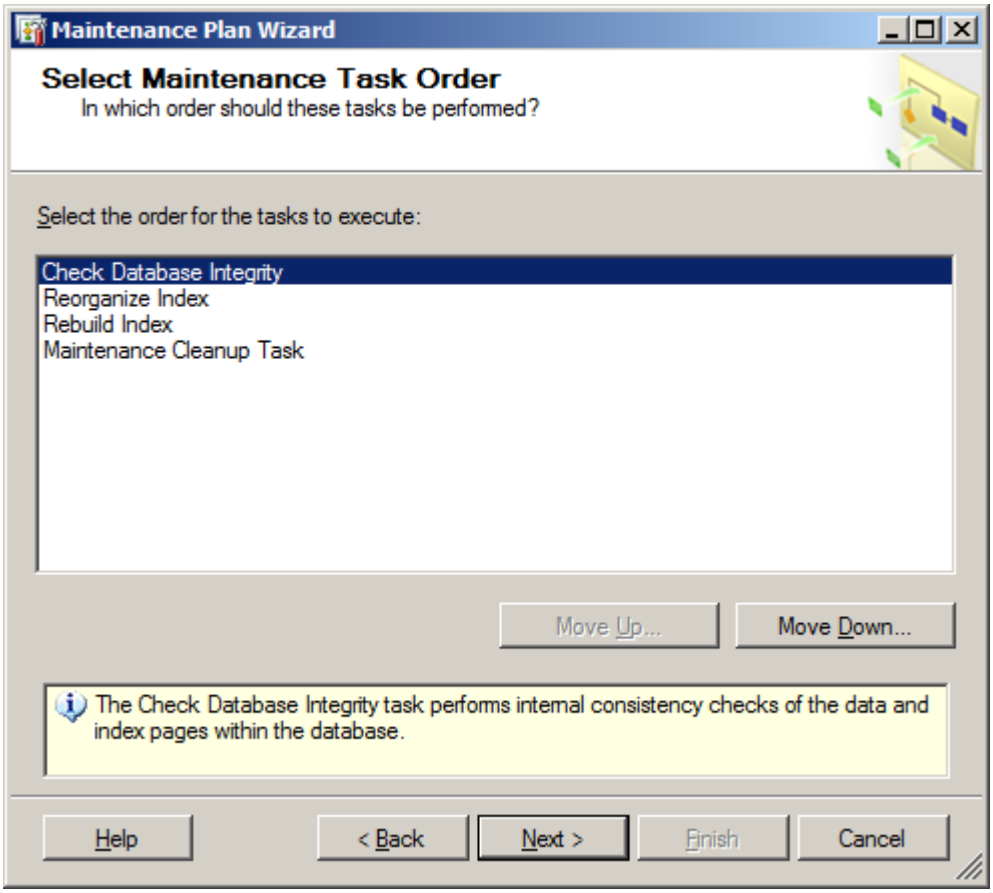


Notes:

- A maintenance plan should include either index reorganization or index rebuilding; not both.
- A maintenance plan should never include shrinking a database.
- To determine the duration of each task, test each task individually before combining tasks into a single plan. You may need to define several maintenance plans on separate schedules to allow tasks to complete during hours when end-user operations will not be negatively affected.
- The Maintenance Cleanup task removes files left over from executing a maintenance plan.

10. On the Select Maintenance Task Order page, change the order of the maintenance plan tasks if needed. Select a task, and then click **Move up** or **Move down**. When tasks are correctly ordered, click **Next**.

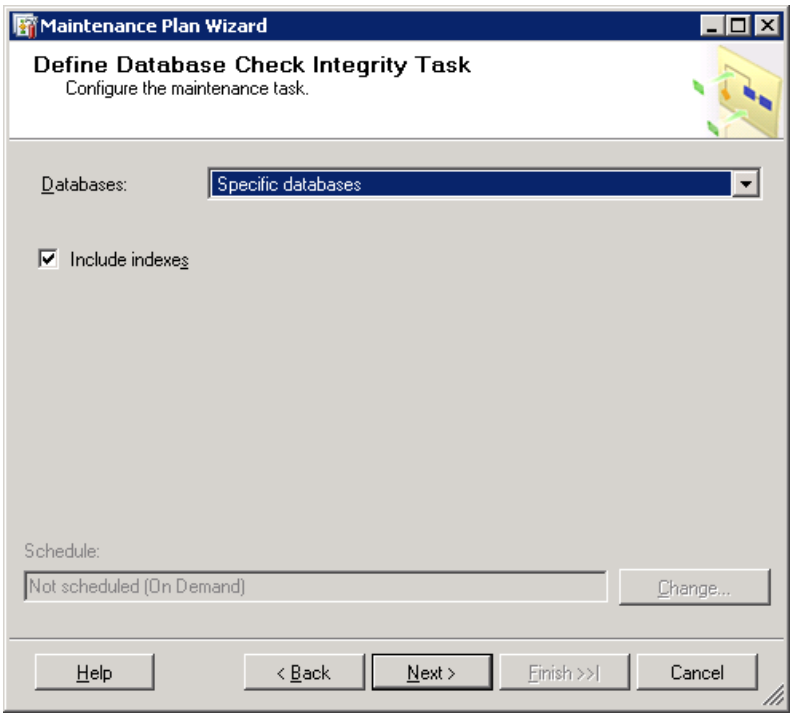
    Note: If your databases are very large, you may want to create a separate maintenance plan for checking database integrity less frequently than index maintenance.



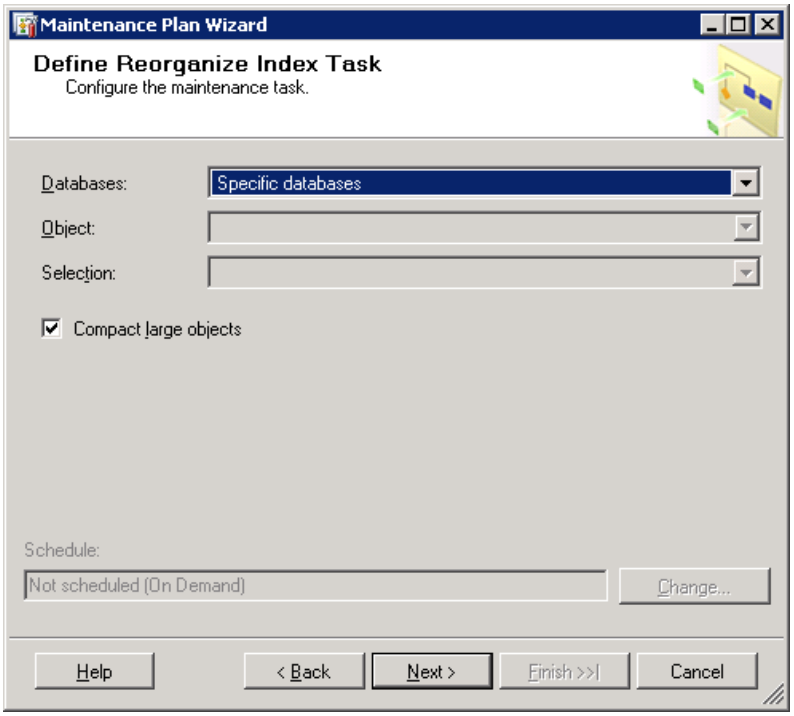
Next, the wizard guides you through setting the details for each task.

11. On the Define Database Check Integrity Task page, select the databases to check for integrity, and then click **Next**.

    Note: You can safely check all SharePoint 2010 databases for integrity.

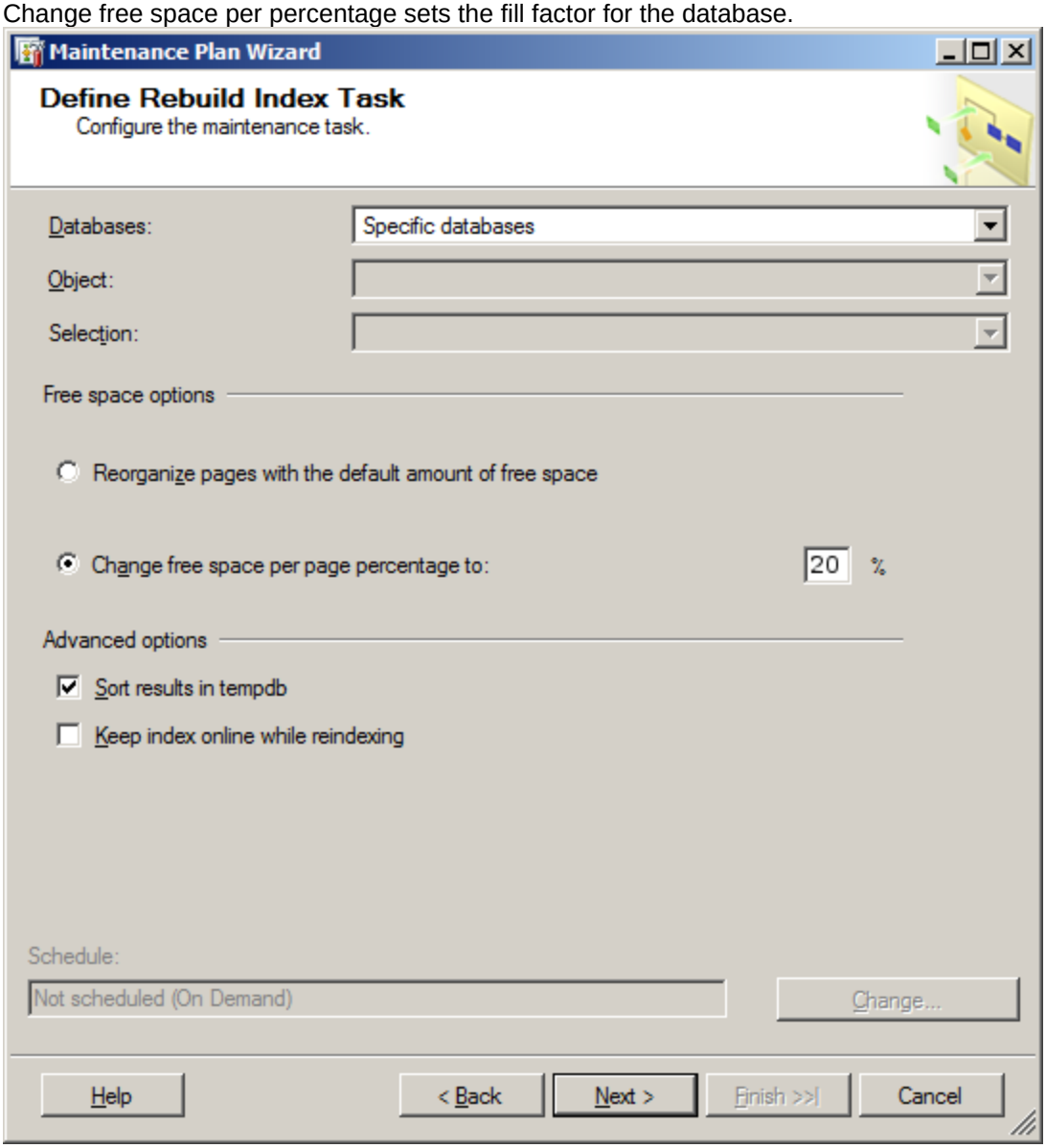


12. On the Define Reorganize Index Task page, in the **Databases** list, specify the databases to reorganize the indexes for, select the **Compact large objects** check box, and then click **Next**.



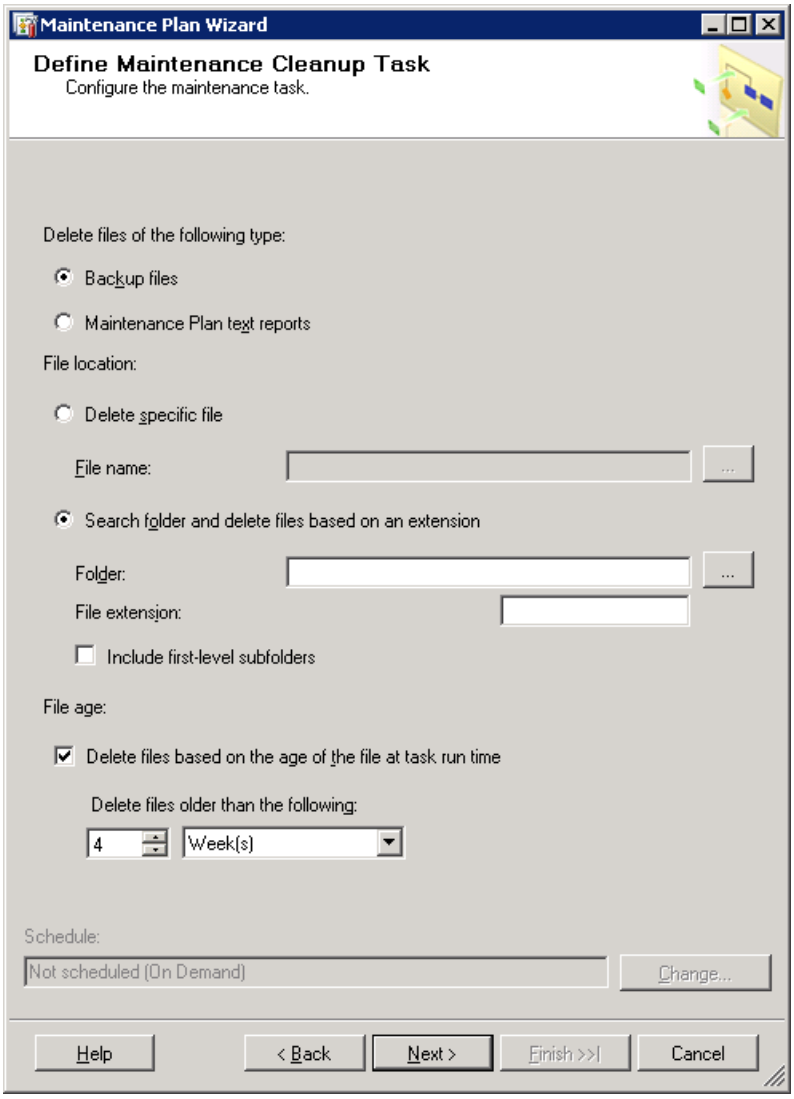
13. On the Define Rebuild Index Task page, if you have chosen to rebuild indexes instead of reorganize those, in the **Databases** list; specify the databases to reorganize the indexes for.
14. Select **Change free space per page percentage**, type **20**, and then click **next**.

Change free space per percentage sets the fill factor for the database.

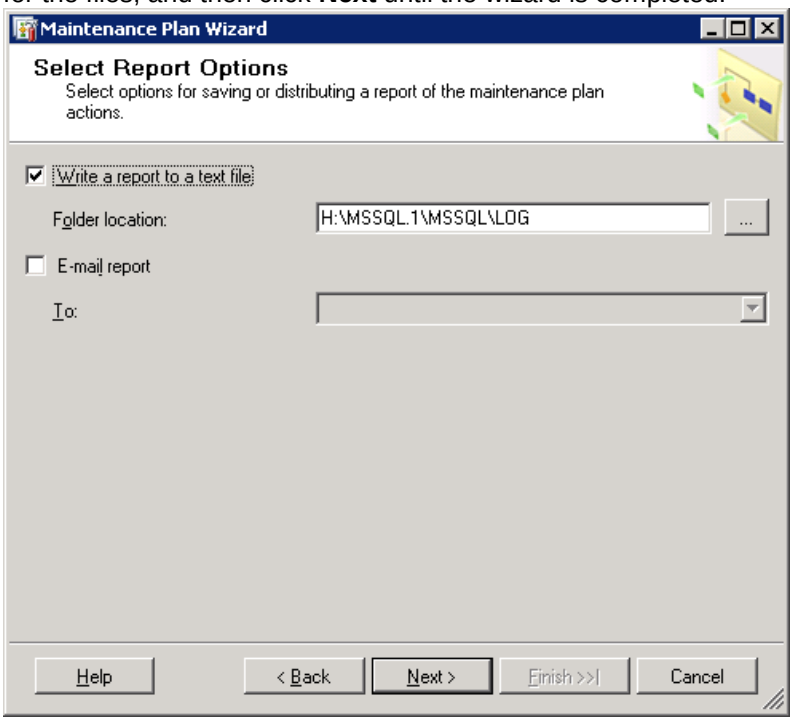


15. On the Define Maintenance Cleanup Task page, set the values that meet your needs, and then click **Next**.

    We recommend that you delete Maintenance Plan text reports.



16. On the Select Report Options page, select **Write a report to a text file**, select a location for the files, and then click **Next** until the wizard is completed.



Summary

Whichever method you choose to use, consistently maintaining the databases that host SharePoint 2010 can significantly improve the health and performance of your system.

Ensure that you have reliable backups for all databases before you implement maintenance operations and maintenance plans.

Before you implement consistently running maintenance operations or a maintenance plan, test the impact of the operations on your system, and the time required to run them.

As much as possible, set any maintenance operations or maintenance plans to run during off hours to minimize the performance impact to users.