Adaptive Index Defrag

Rate This  
http://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.png

**[http://i1.social.s-msft.com/profile/u/avatar.jpg?displayname=pedro+azevedo+lopes&size=large](http://blogs.msdn.com/275655/ProfileUrlRedirect.ashx)**[**Pedro Azevedo Lopes**](http://blogs.msdn.com/275655/ProfileUrlRedirect.ashx)

Pedro Azevedo Lopes

FROM: <http://blogs.msdn.com/b/blogdoezequiel/archive/2011/07/03/adaptive-index-defrag.aspx?goback=%2Egde_54395_member_208944672>

1,369 Points 3 1 0

Recent Achievements

New Blog Rater Blogger II New Blog Commentator

[View Profile](http://social.msdn.microsoft.com/profile/pedro%20azevedo%20lopes/?ws=usercard-hover)

3 Jul 2011 2:00 AM

* Comments [**1**](http://blogs.msdn.com/b/blogdoezequiel/archive/2011/07/03/adaptive-index-defrag.aspx?goback=%2Egde_54395_member_208944672#comments)

Hello all, because the procedure *usp\_AdaptiveIndexDefrag* has undergone a few more updates, starting now this post will be the only source of information for it, including a **full change log at the end of this post**. It’s also reachable from the links section on the right anywhere on the blog.

Download latest version here: [**usp\_AdaptiveIndexDefrag.sql**](http://blogs.msdn.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-01-13-22-AID/3201.usp_5F00_AdaptiveIndexDefrag.sql)

**Latest update:** 28-01-2013 (Check [**change log**](http://blogs.msdn.com/b/blogdoezequiel/archive/2011/07/03/adaptive-index-defrag.aspx?goback=%2Egde_54395_member_208944672#ChangeLog))

**What’s the purpose of this?**

The purpose for this procedure to to perform an **intelligent defrag on one or more indexes for one or more databases**. The 1st release was inspired by an earlier release of Michelle Ufford’s code in SQLFOOL.com site, and has since evolved to suit different and added needs. In a nutshell, this procedure automatically chooses whether to rebuild or reorganize an index according to its fragmentation level, amongst other parameters, like if page locks are allowed or the existence of LOBs.   
All within a specified time frame you choose, defaulting to 8 hours. The defrag priority can also be set, either on size, fragmentation level or index usage (based on range scan count), which is the default.   
It also handles partitioned indexes, optional statistics update (table-wide or only those related to indexes), rebuilding with the original fill factor or index padding and online operations, to name a few options.

**On what version of SQL can I use it?**

This procedure can be used from SQL Server 2005 SP2 onwards, because of the DMVs and DMFs involved.

**How to deploy it?**

Starting with v1.3.7, on any database context you choose to create the *usp\_AdaptiveIndexDefrag* and its supporting objects, **open the attached script, and either change the *@deploymode* variable** at the top **to upgrade mode** (preserving all historic data), **or use the default for new deployments or overwrite old versions** and objects (disregarding historic data).

**How to use it?**

After executing the attached script in a user database of your choice, either run the procedure *usp\_AdaptiveIndexDefrag* with no parameters, since all are optional (If not specified, the defaults for each parameter are used), or customize its use with the following parameters, grouped by type:

***@Exec\_Print*** defaults to *1* (that is to execute the SQL code generated by this SP) or optionally *0* (just print the commands).   
***@printCmds*** defaults to *0* (do not print all commands to screen) or optionally *0* (print all commands to screen). Useful if you just want to see what commands would be executed.   
***@outputResults*** defaults to *0* (does not output fragmentation information) or optionally *0* (output fragmentation information after run completes).   
***@debugMode*** defaults to *0* (do not display debug comments) or optionally *0* (display debug comments).   
***@timeLimit*** limits how much time can be spent performing index defrags and is expressed in *minutes*. Note that the time limit is checked BEFORE an index defrag begins, thus a long index defrag can exceed the time limit. Defaults to 480m (8h).   
***@dbScope*** specifies a database name to defrag. If not specified, all non-system databases plus msdb and model will be defragmented.   
***@tblName*** specifies if you only want to defrag indexes for a specific table. The input format is *database\_name.schema.table\_name. I*f not specified, all tables will be defragmented.   
***@defragOrderColumn*** defines how to prioritize the order of defrags and is used only if *@Exec\_Print* is set to 1. The default is to order by *range\_scan\_count* (count of range and table scans on the index because these can benefit the most from defragmentation), other option are *fragmentation* (amount of fragmentation in the index) or *page\_count* (number of pages in the index).   
***@defragSortOrder*** defines the sort order of the ORDER BY clause on the above query on how to prioritize the order of defrags. Options are *ASC* (ascending) or *DESC* (descending), which is the default.   
***@forceRescan*** defaults to *0*, where a rescan will not occur until all indexes have been defragmented (this allows for a defrag run to span multiple executions over several periods of time). Other option is *1* to force a rescan.   
***@defragDelay*** specifies the time to wait between defrag commands and defaults to *5s*.

Refer to [**Reorganizing and Rebuilding Indexes**](http://msdn.microsoft.com/en-us/library/ms189858.aspx) for documentation on the following parameters. ***@ixtypeOption*** defaults to *NULL* (all indexes will be defragmented). Other options are *1* (only Clustered indexes) or *0* (only Non-Clustered indexes, including XML and Spatial Indexes).   
***@minFragmentation*** defaults to *5*%, will not defrag if fragmentation is less.   
***@rebuildThreshold*** defaults to *30*%. Higher than 30% will result in a rebuild operations instead of reorganize.   
***@minPageCount*** specifies how many pages must exist in an index in order to be considered for a defrag. Defaults to one extent (8 pages).    
***@maxPageCount*** specifies the maximum number of pages that can exist in an index and still be considered for a defrag run. Useful for scheduling small indexes during business hours and large indexes for non-business hours.   
***@fillfactor*** defaults to *1* (the original FF from when the index was created or last defragmented) or optional *0* (uses the default FF of 0).    
***@scanMode*** specifies which scan mode to use to determine fragmentation levels. *LIMITED* mode is the default. Scans the smallest number of pages. For an index, only the parent-level pages of the B-tree (that is, the pages above the leaf level) are scanned. For a heap, only the associated PFS and IAM pages are examined. The data pages of the heap are not scanned. Other options include *SAMPLED* (returns statistics based on a 1 percent sample of all the pages in the index or heap) or *DETAILED* (scans all pages and returns all statistics. Can cause performance issues). If the index or heap has fewer than 10,000 pages, DETAILED mode is automatically used instead of SAMPLED.

***@onlineRebuild*** defaults to *0* (offline rebuild) or optionally *1* (online rebuild if possible)   
***@sortInTempDB*** defaults to *0* (perform sort operation in the index's database) or optionally *1* (perform sort operation in TempDB). If a sort operation is not required, or if the sort can be performed in memory, SORT\_IN\_TEMPDB is ignored. Setting this option to 1 can result in faster defrags and prevent database file size inflation. The caveat is you have to monitor TempDB closely.   
***@maxDopRestriction*** specifies a processor limit for index rebuilds. If not specified, defrag operations will use the system MaxDOP setting, up to a limit of 4.

***@updateStats*** defaults to *1* (updates stats when reorganizing) or optionally *0* (does not update stats when reorganizing)   
***@updateStatsWhere*** defaults to *1* (updates only index related stats) or optionally *0* (updates all stats in entire table). Even if you choose to update stats, and if the *@scanMode* option was NOT set in LIMITED mode, only those within certain thresholds will be updated. Refer to [**Statistical maintenance functionality (autostats) in SQL Server**](http://support.microsoft.com/kb/195565/en-us) to check the defaults for auto-update statistics.                  
***@statsSample*** defaults to *NULL* (performs a sample scan on the target table or indexed view where the database engine automatically computes the required sample size), or optionally *FULLSCAN* (all rows in table or view should be read to gather the statistics) or *RESAMPLE* (statistics will be gathered using an inherited sampling ratio for all existing statistics including indexes).   
***@ix\_statsnorecomp*** defaults to *0* (run with STATISTICS\_NORECOMPUTE OFF). Refer to [**ALTER INDEX (Transact-SQL)**](http://technet.microsoft.com/en-us/library/ms188388.aspx) for information on the option STATISTICS\_NORECOMPUTE. Optionally use *1* (run with STATISTICS\_NORECOMPUTE ON will disable the auto update statistics on index related statistics). If you are dealing with stats update with a custom job (or even with this code by updating statistics), you may use this option.

***@dealMaxPartition*** specifies whether to exclude the right-most populated partition (if an index is partitioned), or act only on that same partition, excluding all others. Typically, this is the partition that is currently being written to in a sliding-window scenario. Enabling this feature may reduce contention. This may not be applicable in other types of partitioning scenarios. Non-partitioned indexes are unaffected by this option. This parameter defaults to *0* (only right-most populated partition is defragmented). If the partition is smaller than *@minPageCount*, it won't be considered. Other options are *1* (to exclude the right-most populated partition) or *NULL* (all partitions are defragmented).                        
***@dealLOB*** specifies if all pages that contain large object (LOB) data are compacted or not. The LOB data types are image, text, ntext, varchar(max), nvarchar(max), varbinary(max), and xml. Compacting this data can improve disk space use. Reorganizing a specified clustered index compacts all LOB columns that are contained in the clustered index. Reorganizing a non-clustered index compacts all LOB columns that are non-key (included) columns in the index. Default is *0* (compact LOBs when reorganizing) and optional *1* (does not compact LOBs when reorganizing).

**@ignoreDropObj** specifies if a table or index is dropped after the defrag cycle has begun, you can choose to ignore those errors in the overall outcome, thus not showing a job as failed if the only errors present refer to dropped database objects. Default is *0* (includes errors about objects that have been dropped since the defrag cycle began) and optional *1* (for error reporting purposes, ignores the fact that objects have been dropped since the defrag cycle began).

**@disableNCIX**specifies if non-clustered indexes are to be disabled before a rebuild. If disk space is limited, it may be helpful to disable the non-clustered index before rebuilding it. When a non-clustered index is not disabled, the rebuild operation requires enough temporary disk space to store both the old and new index. However, by disabling and rebuilding a non-clustered index in separate transactions, the disk space made available by disabling the index can be reused by the subsequent rebuild or any other operation. Hence, no additional space is required except for temporary disk space for sorting (this is typically 20 percent of the index size according to BOL). Note that it does not disable indexes on partitioned tables when defragging a subset of existing partitions. Also, the procedure keeps track of whatever indexes were disabled by the defrag cycle. In case the defrag is canceled, it will account for these disabled indexes in the next run. Default is *0* (does NOT disable non-clustered indexes prior to a rebuild)  and optional *1* (disables non-clustered indexes prior to a rebuild).

**What objects are created when running the attached script?**

1. *tbl\_AdaptiveIndexDefrag\_Working*, used to keep track of which objects to act on, and crucial information that influence how those objects are handled. It also keeps track of which indexes were already defragged in a previous run, if your defrag cycle must span several days due to time constraints.
2. *tbl\_AdaptiveIndexDefrag\_Stats\_Working*, the statistics counterpart of the above table.
3. *tbl\_AdaptiveIndexDefrag\_log*, an index operations logging table, where all the index operations are logged.
4. *tbl\_AdaptiveIndexDefrag\_Stats\_log*, a statistics operations logging table, where all the statistics operations are logged.You might want to cleanup this and the above table after awhile using the procedure *usp\_AdaptiveIndexDefrag\_PurgeLogs*.
5. *tbl\_AdaptiveIndexDefrag\_Exceptions*, an exceptions table where you can set the restrictions on which days certain objects are handled (mask just like in *sysschedules* system table). You can also set exceptions for specific indexes, tables or entire databases.   
   Say you have a specific table that you only want to defrag on weekends, you can set it in the exceptions table so that all indexes on that table will only be defragged on Saturdays and Sundays. Or you want to exclude one database or table from ever being defragged. These are just examples of how to manage specific needs.
6. *tbl\_AdaptiveIndexDefrag\_IxDisableStatus*, where indexes that were disabled are logged, so that an interruption in the defrag cycle can account for these indexes has being disabled by the defrag cycle itself and not the user.
7. *usp\_AdaptiveIndexDefrag\_PurgeLogs*, which will purge the log tables of data older than 90 days, to avoid indefinite growth.   
   The 90 days is just the default, change @daystokeep input parameter to a value you deem fit. I recommend executing this in a job.
8. *usp\_AdaptiveIndexDefrag\_Exclusions*, which is will help in setting on which days (if any) you allow for a specific index, or even all indexes on a given table, to be defragmented. In the previous post [**here**](http://blogs.msdn.com/b/blogdoezequiel/archive/2011/03/10/sql-swiss-army-knife-7-adaptive-index-defrag.aspx) there was an example query of how you could set the exclusions embedded in the script, but due to some feedback, I’ve turned it into an SP.   
   This sproc takes 4 input parameters:
   * *@exclusionMask\_DB*, enter only one database name at a time.
   * *@exclusionMask\_days*, enter weekdays in short form, between commas. Keep only relevant weekdays on which you DO NOT want to allow defragmentation to occur. Order is not mandatory, but weekday short names are important AS IS ('Sun,Mon,Tue,Wed,Thu,Fri,Sat').  
     Conversely, leave the default value of NULL to exclude the object(s) from ever being defragged, equivalent to choosing every day of the week.
   * *@exclusionMask\_tables* (optional) enter table names separated by commas ('table\_name\_1, table\_name\_2, table\_name\_3').
   * *@exclusionMask\_indexes* (optional) enter index names separated by commas ('index\_name\_1, index\_name\_2, index\_name\_3'). If you want to exclude all indexes in a given table, enter its name but don't add index names.
9. *usp\_CurrentExecStats*, which can be used to keep track of which indexes were already defragged thus far in the current execution.
10. *usp\_AdaptiveIndexDefrag*, the main procedure that handles index defragmentation and statistics updates. Takes the input parameters shown before.
11. And several views for miscellaneous purposes:   
    *vw\_ErrLst30Days*, to check all known execution errors in the last 30 days.   
    *vw\_ErrLst24Hrs*, to check all known execution errors in the last 24 hours.   
    *vw\_AvgTimeLst30Days*, to check the average execution time for each index in the last 30 days.   
    *vw\_AvgFragLst30Days*, to check the average fragmentation found for each index in the last 30 days.   
    *vw\_AvgLargestLst30Days*, to check the average size for each index in the last 30 days.   
    *vw\_AvgMostUsedLst30Days*, to check the average usage of each index in the last 30 days.  
    v*w\_LastRun\_Log*, to check in the log tables how the last execution did.

**A few common usage scenarios for this script:**

1. ***EXEC dbo.usp\_AdaptiveIndexDefrag***The defaults are to defragment indexes with fragmentation greater than 5%; rebuild indexes with fragmentation greater than 30%; defragment ALL indexes; commands WILL be executed automatically; defragment indexes in DESC order of the RANGE\_SCAN\_COUNT value; time limit was specified and is 480 minutes (8 hours); ALL databases will be defragmented; ALL tables will be defragmented; WILL be rescanning indexes; the scan will be performed in LIMITED mode; LOBs will be compacted; limit defrags to indexes with more than 8 pages; indexes will be defragmented OFFLINE; indexes will be sorted in the DATABASE; indexes will have its ORIGINAL Fill Factor; only the right-most populated partitions will be considered if greater than 8 page(s); statistics WILL be updated on reorganized indexes; defragmentation will use system defaults for processors; does NOT print the t-sql commands; does NOT output fragmentation levels; waits 5s between index operations;
2. ***EXEC dbo.usp\_AdaptiveIndexDefrag @dbScope = 'AdventureWorks2008R2'***Same as above, except its scope is only the 'AdventureWorks2008R2' database.
3. ***EXEC dbo.usp\_AdaptiveIndexDefrag @dbScope = 'AdventureWorks2008R2', @tblName = 'AdventureWorks2008R2.Production.BillOfMaterials'***Same as above but only acting on the BillOfMaterials table.
4. ***EXEC dbo.usp\_AdaptiveIndexDefrag @Exec\_Print = 0, @printCmds = 1***Using the operating defaults in 1, this will not execute any commands. Instead, just prints them to the screen. Useful if you want to check what it will be doing behind the scenes.
5. ***EXEC dbo.usp\_AdaptiveIndexDefrag @Exec\_Print = 0, @printCmds = 1, @scanMode = 'DETAILED', @updateStatsWhere = 0***Same as above, but adding the DETAILED scanMode to allow for finer thresholds in stats update, and forcing the update statistics to run on all stats found on table, instead of just the ones related to the indexes.
6. ***EXEC dbo.usp\_AdaptiveIndexDefrag @scanMode = 'DETAILED', @updateStatsWhere = 0 , @disableNCIX = 1***Differs from the above just because it will execute the comands instead of printing them and will disable non-clustered indexes prior to a rebuild operation.
7. ***EXEC dbo.usp\_AdaptiveIndexDefrag @minFragmentation = 3, @rebuildThreshold = 20***Using the operating defaults in 1, this will lower the minimum fragmentation that allows the defrag to include a given index to 3%, and the rebuild vs. reorganize threshold to just 20%.
8. ***EXEC dbo.usp\_AdaptiveIndexDefrag @onlineRebuild = 1***Using the operating defaults in 1, this will try to do online rebuild operations whenever possible.
9. ***EXEC dbo.usp\_AdaptiveIndexDefrag @onlineRebuild = 1, @updateStatsWhere = 0, @statsSample = 'FULLSCAN'***Similar to the above, this will also force update statistics to run on all stats found on table with FULLSCAN.
10. ***EXEC dbo.usp\_AdaptiveIndexDefrag @onlineRebuild = 1, @updateStatsWhere = 0, @dbScope = 'AdventureWorks2008R2', @defragOrderColumn = 'fragmentation', @timeLimit = 240, @scanMode = 'DETAILED'***Similar to the above, this will also restrict all defrag operations to the 'AdventureWorks2008R2' database, giving priority to the most fragmented indexes (instead of the most used, which is the default), limiting the time window for defrag operations to just 4 hours, and using the DETAILED scanMode to allow for finer thresholds in stats update.
11. ***EXEC dbo.usp\_AdaptiveIndexDefrag @timeLimit = 360***Using the operating defaults in 1, will set the running window to just 6 hours.
12. ***EXEC dbo.usp\_AdaptiveIndexDefrag @rebuildThreshold******= 99, @dealMaxPartition = 1, @onlineRebuild = 1***Using the operating defaults in 1, this will try to do online rebuild operations whenever possible and exclude from the defrag run the right-most partition will while setting a rebuild threshold of 99%, essentially forcing a reorganize instead of a rebuild.   
    Useful if you consider the scenario where you had all your indexes with a low fill factor for some purpose in a partitioned table, but then had to rebuild them using a higher fill factor and reclaim the space using DBCC SHRINKFILE (yes, not advisable but can happen on occasion). Forcing a reorganize on all but the right-most partition (active) is the most efficient way of defragmenting your indexes again with minimum impact on the server availability.

**Change log:**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Comments** |
| 1 | 08-02-2011 | Initial release; |
| 1.1 | 15-02-2011 | Added support for maintaining current index padding options; Added logic for exclusion of hypothetical objects; Deal with LOB compaction when reorganizing; Corrected bug with update stats kicking in when not supposed to; Corrected options not compatible with partitioned indexes; |
| 1.2 | 10-03-2011 | Increased control over new or changed database handling; |
| 1.2.1 | 22-03-2011 | Corrected method of finding available processors; |
| 1.3 | 21-06-2011 | Added more options to act upon statistics (IX related or Table-wide); Added finer thresholds for updates on table-wide statistics when reorganizing (when SAMPLED or DETAILED scanMode is selected) **\***; Added option for no\_recompute on index REBUILD; Added restrictions for spatial and XML indexes; Always rebuild filtered indexes; If found, output list of disabled or hypothetical indexes so that you can act on them; Added range scan count to logging table for comparison; Added update index related stats (with defaults) before rebuild operations. This provides better cardinality estimation, and thus a more time-efficient operation when rebuilding; Bug fix in Reorganize statements. Bug fix in one Rescanning condition. |
| 1.3.1 | 28-06-2011 | Corrected issue with commands running on multiple partitions. Changed behaviour of update statistics when tables have multiple partitions. |
| 1.3.2 | 01-07-2011 | Changed objects named %Exclusions to %Exceptions. When re-deploying, existing %Exclusions table will be renamed and not recreated. Added procedure to check current batch execution progress (usp\_CurrentExecStats). |
| 1.3.3 | 08-07-2011 | Corrected issue where explicit change in database scope parameter did not trigger rescan under certain conditions. Corrected statistics update thresholds. |
| 1.3.4 | 22-07-2011 | Bug fix in indexes information when working in SQL 2005. |
| 1.3.5 | 15-11-2011 | Bug fix in logging showing as NULL on some issued commands.  Optimizations on support SP usp\_AdaptiveIndexDefrag\_Exceptions. |
| 1.3.6 | 17-02-2012 | Allow large object names in tables and indexes. |
| 1.3.7 | 27-02-2012 | Enhanced error reporting view to incorporate stats updates. Bug fix when certain index options were chosen together. |
| 1.3.8 | 28-02-2012 | Corrected view that reports last run. Added upgrade mode. |
| 1.3.9 | 12-03-2012 | Fixed upgrade mode in case old data cannot be copied back. |
| 1.4.0 | 12-04-2012 | Fixed issue with case sensitive servers. |
| 1.4.1 | 17-05-2012 | Fixed issue on support SP usp\_AdaptiveIndexDefrag\_Exceptions. |
| 1.4.2 | 29-05-2012 | Fixed issue on support SP usp\_AdaptiveIndexDefrag\_CurrentExecStats. Fixed issue with large object IDs. |
| 1.4.3 | 29-08-2012 | Fixed issue with upgrade mode data retention. Fixed issue with format dependent conversions. |
| 1.4.4 | 10-09-2012 | Fixed issue where running the procedure to print commands only, previous execution errors would still be reported. |
| 1.4.5 | 12-10-2012 | Added support for ignoring errors regarding database objects that were dropped since the defrag cycle began;  Added support for disabling indexes before rebuilding (space saving feature) - see notes on parameter @disableNCIX. |
| 1.4.6 | 23-01-2013 | Added hard limit of 4 for MaxDOP setting; Changed default for statistics update to updates all stats in table, as opposed to just index related stats; Fixed issue on support SP usp\_AdaptiveIndexDefrag\_CurrentExecStats reporting incorrect number of already defraged indexes; Fixed null elimination message with vw\_LastRun\_Log; Incremented debug mode output; Redesigned table wide statistics update (see notes on parameter @updateStatsWhere); Fixed issue with upgrade mode leaving old tables behind. |
| 1.4.7 | 28-01-2013 | Fixed issue with exceptions not working with on some days i.e, on a day that should not be doing anything, it did; Tuned online rebuild options; Redesigned support SP usp\_AdaptiveIndexDefrag\_Exceptions. |

**\*** As you may know, the default thresholds for [**auto update statistics**](http://support.microsoft.com/kb/195565/en-us) differ. If the [**cardinality**](http://blogs.msdn.com/b/bartd/archive/2011/01/25/query_5f00_tuning_5f00_key_5f00_terms.aspx) for a table is greater than 6, but less than or equal to 500, update statistics every 500 modifications. If the cardinality for a table is greater than 500, update statistics when (500 + 20 percent of the table) changes have occurred. On the other hand, the [**sp\_updatestats**](http://msdn.microsoft.com/en-us/library/ms173804.aspx) method is only sensitive to the row modification counter calculated in the [**sys.sysindexes**](http://msdn.microsoft.com/en-us/library/ms190283.aspx) catalog view, which counts the total number of inserted, deleted, or updated rows since the last time statistics were updated for the table. This can be can be over simplistic in itself.   
So, when executing this script (thru a daily job, for example), and you choose a scan mode option other than ‘Limited’ (pertaining to [**sys.dm\_db\_index\_physical\_stats**](http://msdn.microsoft.com/en-us/library/ms188917.aspx)), the default for this script (refer to the parameters section in the script for other options), some finer thresholds somewhere in between both methods can be forced when dealing with statistics update.   
One other enhancement is to issue an UPDATE STATISTICS command on the index just before rebuilding it. This can yield better performance in the rebuild phase itself. Does not apply when an index is reorganized instead of rebuilt.   
You can look in the changelog section of the usp\_AdaptiveIndexDefrag for detailed information on the changes done in every version so far.

Until next time!

**Disclaimer:** I hope that the information on these pages is valuable to you. Your use of the information contained in these pages, however, is at your sole risk. All information on these pages is provided "as -is", without any warranty, whether express or implied, of its accuracy, completeness, fitness for a particular purpose, title or non-infringement, and none of the third-party products or information mentioned in the work are authored, recommended, supported or guaranteed by Ezequiel. Further, Ezequiel shall not be liable for any damages you may sustain by using this information, whether direct, indirect, special, incidental or consequential, even if it has been advised of the possibility of such damages.