SQLskills Immersion Event
IE0: Accidental/Junior DBA

Module 6: Database Maintenance

Erin Stellato
Erin@SQLskills.com

Introduction

- **Database maintenance includes the tasks that keep the database healthy and available**
  - Backups
  - Consistency checks
  - Update statistics
  - Remove fragmentation from indexes
- **These tasks should be implemented for every database you deploy, and they should be automated**
- **Various options exist for automation:**
  - Maintenance Plans
  - Custom scripts
  - SSIS Packages
  - PowerShell scripts

Database Maintenance Plans

- **Benefits**
  - Simplifies maintenance configurations
  - Limited options for each of the most common tasks in the Maintenance Plan Wizard
    - Additional options are available as of SQL Server 2016
  - Fully integrated support in Management Studio makes it easier to create and modify tasks and schedules to meet business requirements
- **Limitations**
  - Not every task listed should be a regular part of maintenance operations
  - Default configurations often perform operations that are unnecessary
    - e.g. rebuilding all indexes in a database even if they aren't fragmented, updating statistics without any analysis
  - Maintenance Plans are not exportable to allow for ease of reuse across multiple servers

Custom Transact-SQL Scripts

- **Benefits**
  - Allow customization of maintenance tasks as all available options can be used
  - Minimize and/or eliminate unnecessary operations through programmatic analysis of index fragmentation, page density, or data change
  - Free scripts exist online that handle most maintenance requirements
  - Can handle VLDB configurations by tracking progress within a maintenance window
    - Allows you to divide the work across multiple days
  - Easy to transport scripts for standard configurations across multiple servers
- **Limitations**
  - Requires understanding Transact-SQL, if you create your own
  - Requires understanding maintenance requirements to create the optimal set of tasks
  - Requires manually creating scheduled tasks in SQL Server Agent

Deciding What Method to Use

- You can use any combination of scripts, plans, etc. to automate database maintenance
- You should choose the best option(s) based on your solution, your knowledge, and your ability to support the method going forward
- Test any scripts obtained from the internet in a test environment first
  - It is important to understand *exactly* what a script (written by someone else) does before you implement it in production

Overview

- Index management
- The impact of shrink
- Statistics management
- Consistency checking
- Planning for data purging and ETL processes

Index Review

- **An index is an ordered set of data file pages**
- **It represents a subset of information in the table**
  - Can be comprised of 1 to *n* columns from the table
- **It is a physical structure**
  - Data is stored on pages on disk, just like tables
- **Types of disk-based indexes:**
  - Clustered – all columns in the table
  - Nonclustered – subset of columns in the table
    - Filtered
  - Columnstore – data for each column stored together
    - Clustered
    - Nonclustered

What is Index Fragmentation?

- **For every modification of a row in a table, relevant indexes are updated**
  - Insert – new row added to the base table (heap or clustered index), and a row is added to each applicable nonclustered index
  - Update – row is modified in base table, rows are modified in applicable nonclustered indexes
  - Delete – row is removed from the base table, and a row is removed from each applicable nonclustered index
- **These modifications can affect the order and location of the index pages, and can create empty space on the index pages**

Logical Fragmentation Defined

- (Sometimes called "external" fragmentation)
- Occurs when the next logical page is not the next physical page
- Prevents optimal readahead
  - Reduces range scan performance
- Does not affect pages that are already in cache
  - Smaller indexes affected less (e.g. 1-5,000 pages or less)
- Reported as avg_fragmentation_in_percent in the sys.dm_db_index_physical_stats DMV for indexes

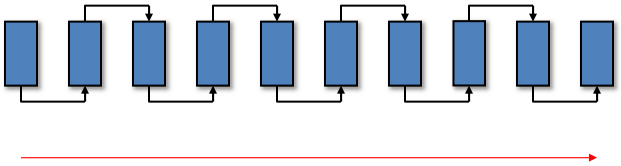
Page Density Defined

- Page density refers to the amount of data on a page
  - Sometimes called "physical" or "internal" fragmentation
- With a low page density, page fullness is below the optimal level and there can be a lot of wasted space
- Effect is:
  - Increased disk space (more pages required to hold the same number of rows)
  - Increased I/Os to read the same amount of data
  - Greater memory usage if most of the index is memory resident
- Reported as avg_page_space_used_in_percent in the sys.dm_db_index_physical_stats DMV

Fragmentation in Action (1)

Index pages of newly built index

(All movies nicely ordered on shelves)

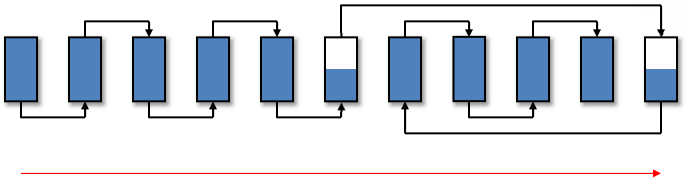


Long arrow is the allocation order
Small arrows are following the logical order

Fragmentation in Action (2)

Newly built index after adding one new row…that could not fit on a page

(There isn't space on the shelf for the new movie you bought…where do you put it?)

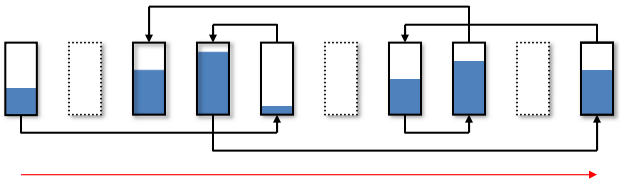


Long arrow is the allocation order
Small arrows are following the logical order

Fragmentation in Action (3)

Index pages after random inserts/deletes

(How your movie shelves look after people keep borrowing movies, and you keep buying new ones.)



Long arrow is the allocation order
Small arrows are following the logical order

Finding Fragmentation

- Fragmentation occurs as a result of table design and/or workloads that cause page splits on full pages
- It is expected in a SQL Server database
- Use the sys.dm_db_index_physical_stats DMV to see the level of fragmentation for an index
  - Logical fragmentation
    - avg_fragmentation_in_percent (should be low)
  - Page density
    - avg_page_space_used_in_percent
      - Should be high for data warehouse
      - Should have some free space for OLTP

Demo

Detecting fragmentation using sys.dm_db_index_physical_stats

How Do You Remove Fragmentation?

- **There are two realistic choices**
  - Rebuild the index
  - Defrag the index (also known as reorganize)
- **Can also choose not to remove fragmentation**
  - If the index isn't used for range scans, and page density isn't an issue, why spend the resources?
- **In general, you want to avoid rebuilding every single index every single day (or even every week)**
  - However, if you have the maintenance window and the resources, this is better than doing nothing at all

ALTER INDEX … REBUILD

- **Pros**
  - No knowledge of index schema or constraints required
  - Can use multiple CPUs, and control MAXDOP
  - Rebuilds index statistics (with equivalent of full scan, or sampled if partitioned index)
  - Can rebuild a single partition (online from 2014) or all partitions
  - Can be performed online
    - 2012: Indexes with LOB columns (plus clustered index on table with LOB column)
  - Can be minimally-logged (but log backup will be the same size)
  - SORT_IN_TEMPDB can reduce logging and produce a perf boost from 2012 onward
    - Can reduce amount of log sent to mirror or AG replica(s)
- **Cons**
  - Atomic operation – potentially long rollback on interrupt, all or nothing semantics
  - Requires creating complete new index before dropping old one
  - When offline – SCH-M table lock for nonclustered or clustered index rebuild
    - 2014: blocking lock can be postponed to allow better concurrency

ALTER INDEX … REORGANIZE

- **Pros**
  - ALWAYS online – only requires table IX lock
  - Interruptible with no loss of work – stops instantly
  - Has progress reporting in sys.dm_exec_requests / percent_complete
  - Optionally compacts LOB storage (on by default)
  - Usually faster for a lightly fragmented index
  - Can reorganize one or all partitions
  - Does not require any extra disk space
- **Cons**
  - Usually slower for a heavily fragmented index
  - Always fully-logged, single CPU only, does not update statistics
  - Does not do as good a job as removing fragmentation when there is plenty of contiguous free space

When To Rebuild vs. Defrag

- **Much debate on this…and basically it depends**
- **Points to consider:**
  - Level of fragmentation
  - Amount of transaction log generated
  - Amount of space available
  - Accessibilty of index during the operation
  - Can you interrupt the process?
- **Your mileage may (and will) vary**
- **Deciding between rebuild vs. reorganize often comes down to fragmentation level**
  - < 10% do nothing
  - 10% <> 30% defrag/reorganize
  - 30%+ rebuild
  - And don't do anything if the index has < 1-5,000 pages

FILLFACTOR

- **Makes the Storage Engine leave space on each leaf-level page to allow row inserts/expansions without causing page splits**
- **Specified at index creation or rebuild time**
  - NOT maintained during regular data modifications
- **Do not change the sp_configure option**
- **0 = 100 = default value with special meaning of 'leave no space'**
  - Excellent for data warehouse, but not ideal for OLTP

Choosing the "Right" FILLFACTOR

- **Balancing act between how often page splits occur and how often you can rebuild/defrag the index**
- **What is going to cause page splits in your schema?**
  - UPDATEs to variable-width data types?
  - Random INSERTs?
    - The more volatile ⇨ lower FILLFACTOR
- **How often can you rebuild/defrag?**
  - The more frequent ⇨ higher FILLFACTOR
- **Pick a value, try it, monitor fragmentation, tweak it**
  - Use DMVs to see how fast the fragmentation increases
  - The faster fragmentation occurs ⇨ lower FILLFACTOR or decreased time between rebuilds/defrag
  - 70% is a common first guess

Demo

Removing fragmentation

Maintenance Plan Options (before SQL Server 2016)

- **Rebuild Index Task**
  - Will rebuild *all* indexes in a database
  - The actual level of fragmentation is not considered
  - Fillfactor can be configured
  - Options to sort results tempdb and keep indexes online can be used
- **Reorganize Index Task**
  - Will reorganize *all* indexes in a database
  - The actual level of fragmentation is not considered
- **Maintenance tasks that address fragmentation are "all or none"**
- **If you have a large maintenance window and spare resources, this might be the easiest thing to do**
- **For databases that are mirrored, participate in Availability groups, or are log shipped, this can result in a large amount of log generation that has no benefit whatsoever**

Maintenance Plan Options in SQL Server 2016+

- **Rebuild Index Task**
  - Will rebuild indexes in one or more selected databases based on fragmentation level, page count, and use
  - Fillfactor can be configured
  - Options to sort results tempdb and keep indexes online can be used
  - Lock wait and priority options can be set
- **Reorganize Index Task**
  - Will rebuild indexes in one or more selected databases based on fragmentation level, page count, and use
- **Tasks can be configured to make better use of system resources and potentially generate less transaction log**
- **All objects (indexes and/or views) are considered**
- **If you have a large maintenance window and spare resources, this still might be the easiest thing to do**

Custom Script Options

- **Writing your own script to manage fragmentation provides significant flexibility**
- **However, such a script could be time-consuming to write**
  - Create a driver table that lists indexes
  - Use DMF to determine the level of fragmentation
  - Based on threshold, rebuild, reorg, or do nothing
  - Log what was done for future reference
  - Bonus: keep a counter of how many times in succession an index is rebuilt and programmatically reduce fillfactor
- **If you want to tackle the task for the sake of learning, go for it**
  - Use the sys.dm_db_index_physical_stats DMF for fragmentation analysis
- **If you don't want to reinvent the wheel…**

Ola Hallengren's Maintenance Scripts

- **Provide multiple database selection options:**
  - System databases only
  - User databases only
  - All databases or individual databases
  - Databases with inclusions or exclusions
  - Databases in select Availability Groups
- **Low, medium and high fragmentation thresholds that determine the specific operation to be performed based on the threshold values**
- **Options to set fragmentation levels, minimum page counts, whether to sort in tempdb, maxdop, and fillfactors for index operations**
- **The ability to perform index operations at the partition level**
- **The ability to specify certain indexes or exclude Microsoft provided indexes**
- **Set lock wait, priority, and resumable options**

Overview

- Index management
- The impact of shrink
- Statistics management
- Consistency checking
- Planning for data purging and ETL processes

When to Use Data File Shrink

- Shrinking data files should be a very rare operation
  - Primarily because it causes index fragmentation
- Three scenarios:
  - Emptying a file before removing it
  - When a large amount of data has been deleted AND the space won't be reused
  - Moving a file/filegroup/database to read-only
- Even then, shrink may not be the best method
  - Remember – it causes fragmentation
- So, how to perform a data file shrink?

How to Shrink a Data File

- **Create a new filegroup and move all indexes into it**
  - Use CREATE INDEX… WITH DROP_EXISTING and specify the location to be the new filegroup
    - Doesn't move LOB data, but see http://bit.ly/H75AWL
  - Can be performed online
- **If any table are heaps, use shrink to move them**
  - Heaps are unordered so shrink does not fragment them
  - Shrink is very slow for heaps and LOB data
- **Drop old filegroup**
- **If you have to shrink a data file, use ALTER INDEX … REORGANIZE to remove index fragmentation afterwards**
  - Rebuilding the indexes will grow the files again

Common Shrink Misuse

- **Using shrink in a regular maintenance plan (or auto-shrink and auto-grow enabled)**
  - Leads to shrink-grow-shrink-grow cycle
  - Causes massive fragmentation
  - MS Dynamics CRM does this – be aware!
    - http://bit.ly/x9a8Pa
- **Using shrink after an index rebuild to reclaim space**
  - An index rebuild needs space to build a new copy of the index, maybe growing the file
  - Shrinking after a rebuild will reclaim that space BUT will undo the effects of the index rebuild, wasting a lot of time and resources

A Time When Shrink is OK…

- **Do not be surprised if, at some point, you have transaction log file that grows very large unexpectedly**
  - Transaction log backups stop running
  - An extremely large transaction runs, causing the log to grow
- **In this case, you will want to shrink the file back to its normal size**
  - Otherwise, if you have to restore, you will have to wait longer than usual for the log file to zero initialize
- **Shrink the log file down to a small size, then grow in 8GB increments until it is back to the usual size**
  - Use 8000MB if on SQL server 2008/2005, because of a bug

Demo

Impact of shrink on index fragmentation

Overview

- Index management
- The impact of shrink
- Statistics management
- Consistency checking
- Planning for data purging and ETL processes

Statistics Review

- Statistics provide details about the selectivity of a column, and the distribution of data in a column
- Statistics are important because they provide valuable information to the query optimizer, which decides how it will access data
- Consider a query that joins to two tables, where each table has two nonclustered indexes
  - Should the tables be scanned?
  - Should the nonclustered indexes be used?
    - Can the indexes be seeked, or do they need to be scanned?

When Are Statistics Created?

- **Automatically**
  - For all indexes
  - When "auto create statistics" is ON, statistics are created when a column is in a SARG or a join and does not have an index with that column as the high-order element
- **Manually**
  - sp_createstats
  - Using CREATE STATISTICS

*Reminder: Leave Auto Create Statistics ON**

**However, Sharepoint does not allow this to be enabled*

What If the Data Changes?

- **Automatically updated!**
  - If auto update statistics is ON (for both the DB and the table – didn't set "norecompute")
  - If roughly 500 + 20% of the data changes
    - Complete details in: Statistics Used by the Query Optimizer in Microsoft SQL Server 2008
      - http://technet.microsoft.com/en-us/library/dd535534%28v=sql.100%29.aspx
    - Threshold changes in SQL Server 2016
  - Update more frequently with TF 2371
    - Available in 2008 R2 SP1+
    - http://support.microsoft.com/kb/2754171
    - Not needed in SQL Server 2016+ if compatibility mode set to 130
- **Manually update statistics**
  - Execute UPDATE STATISTICS
  - Execute sp_updatestats

How to Update Statistics

- **Manually: but automated through a job**
  - Executing sp_updatestats
    - Sledgehammer maintenance. Only one row has to have been modified to execute this… not very granular
  - Ola Hallengren's script
  - Roll your own?
    - Programmatically evaluate the stat_header and amount of data changed
      - Update statistics based on date (e.g. stats_date older than 1 week) or percentage of data changed
- **Auto update stats: only as a safety measure**
  - As a fallback if your code doesn't catch EVERY statistic
- **Asynchronous update stats**
  - *Unlikely* to cause a problem

Understanding Sample Rate and What To Use

- **The sample rate is how much of the data SQL Server reads to create the statistic**
- **Statistics updated with FULLSCAN require reading all the data**
  - This can be time-consuming
  - However, it can create a more accurate statistic than the default sample
  - Remember that an index rebuild will update statistics with FULLSCAN
- **For any table greater than 8MB in size, a default sample is used**
  - This is typically a small percentage of the table
- **The default sample used by SQL Server is often adequate**
  - In cases where query performance is poor AND the actual number of rows returned from a query is significantly different from the estimate, verify the sample rate used for statistics creation (rows v. rows scanned) and consider using FULLSCAN
  - Option to persist a sampling rate (PERSIST_SAMPLE_PERCENT) available in 2016 SP1 CU4 and 2017 CU1

Demo

Updating statistics

Using the Update Statistics Task

- Update statistics for select databases
- Can choose to update:
  - All statistics
  - Column statistics only
  - Index statistics only
- Can specify whether the update does a FULLSCAN, or uses a specific Sample Rate
  - Option to persist sample rate is not available
- Statistics are updated regardless of whether any rows have been modified, and regardless of last update date
- As with index rebuilds, this can waste resources

Ola Hallengren's Maintenance Scripts

- **Provide multiple database selection options:**
  - System databases only
  - User databases only
  - All databases
  - Individual databases
  - Databases with inclusions or exclusions
- **The ability to update column statistics**
- **The ability to set sample rates for statistics**
- **The ability to update only modified statistics**
  - The ability to update only if modifications exceed a threshold

Overview

- **Index management**
- **The impact of shrink**
- **Statistics management**
- **Consistency checking**
- **Planning for data purging and ETL processes**

Database Corruption Review

- Structural inconsistency in the database
- A problem in the physical and/or logical structure of the database which prevents SQL Server from accessing some part of the database, causes incorrect behavior, or returns incorrect data

How Does Corruption Occur?

- It's the I/O subsystem, usually always
  - Very small percentage of corruptions are bad memory
  - Miniscule percentage corruptions are SQL Server bugs
- Consider the MTBF of disks, you're going to have a problem at some point in your career
- Jim Gray likened a disk head in a 15,000rpm disk to a 747 flying at 500mph about ¼ inch above the ground
  - What happens in a crash?
- Corruptions cannot be caused by:
  - Anything an application can do
  - Interrupting a shrink, rebuild, or long-running batch
  - Propagation by log-shipping, replication, mirroring

Page Protection Options

- SQL Server allows pages to be 'protected' on disk from corruptions
- Allows fast detection of corruptions
- Set using
  - ALTER DATABASE SET PAGE_VERIFY <option>
- Three options:
  - NONE (don't do this…)
  - TORN_PAGE_DETECTION
  - CHECKSUM

Page Checksums

- Per-page checksum
  - Written as the very last thing SQL Server does on a physical write
  - Checked as the very first thing SQL Server does on a physical read
- Provides the 'smoking gun' that the error is not due to SQL Server
- On by default for new databases in SQL Server 2005+
  - Added to tempdb for 2008+
  - Switching it on doesn't do anything until pages are written…
- Checksum failures result in an 824 error
- Negligible CPU overhead
- Error-detecting, not error correcting
- Blog post: How to tell if the IO subsytem is causing corruptions?
  - https://www.sqlskills.com/ie0/iosubsystemcorruption

I/O Errors

- **Three types**
  - 823: a hard I/O error
  - 824: a soft I/O error
  - 825: a read-retry error
- **Helpful error messages:**

```
Msg 824, Level 24, State 2, Line 1
SQL Server detected a logical consistency-based I/O error: incorrect
checksum (expected: 0x7232c940; actual: 0x720e4940). It occurred during
a read of page (1:143) in database ID 8 at offset 0x0000000011e000 in
file 'c:\sqlskills\broken.mdf'. Additional messages in the SQL Server
error log or system event log may provide more detail. This is a severe
error condition that threatens database integrity and must be corrected
immediately. Complete a full database consistency check (DBCC CHECKDB).
This error can be caused by many factors; for more information, see SQL
Server Books Online.
```

- **Logged in msdb.dbo.suspect_pages**
  - Input into single-page restore operations

Read-Retry

- **Present in SQL Server 2000 for limited circumstances**
- **Extended in SQL Server 2005 for data pages**
- **Reads that fail because of I/O errors are retried 4 times**
- **Eventual failure results in an I/O error**
- **Eventual 'success' results in an error in the error log:**
  - A read of the file <<FILE NAME>> at offset <<PHYSICAL OFFSET>> succeeded after failing <<RETRY COUNT>> time(s) with error: <<DETAILED ERROR INFORMATION>>. Additional messages in the SQL Server error log and system event log may provide more detail. This error condition threatens database integrity and must be corrected. Complete a full database consistency check (DBCC CHECKDB). This error can be caused by many factors; for more information, see SQL Server Books Online.
- **Early warning of impending doom as read-retry means the I/O subsystem is returning incorrect data to SQL Server**
- **KB article: http://support.microsoft.com/kb/2015757**

DBCC CHECKDB

- **The only way to read all allocated pages in the database**
  - Use to force page checksums to be checked
- **Choose between full checks and WITH PHYSICAL_ONLY**
- **It runs online by default, using a hidden database snapshot**
- **Use DBCC CHECKDB (yourdb) WITH NO_INFOMSGS**
- **If it's taking longer than usual, that usually means that it found some corruption**
- **Most importantly, wait for it to complete!**

- **Blog post series from Paul: CHECKDB From Every Angle**
  - https://www.sqlskills.com/ie0/checkdbpr

Things to Be Aware Of

- **CHECKDB can use tempdb extensively, so tempdb must be sized appropriately**
  - Use WITH ESTIMATEONLY to determine how much space is needed
  - Be aware of a bug related to this option in SQL Server 2008R2
- **Internal database snapshots use NTFS alternate streams on existing database files**
  - Some 3rd party software use kernel filter drivers that do not cope with these and so CHECKDB thinks the snapshot is corrupt
  - Not applicable as of SQL Server 2014, no longer uses alternate streams
- **Beware of sudden very long run-times**
  - Most likely a corruption has triggered a deep-dive algorithm

Demo

Running DBCC CHECKDB

DBCC CHECKDB Questions

- **How often should I run DBCC CHECKDB?**
  - As often as possible
  - At least once per week
- **How long will it take?**
  - Depends on size of the database, concurrent workload, server speed, and many other factors
  - See Paul's post for more details – How long will CHECKDB take to run?
    - https://www.sqlskills.com/ie0/checkdbduration
- **How to consistency check a very large database?**
  - Take the backup to another server, restore, and run DBCC CHECKDB
  - Break up the checks using DBCC CHECKFILEGROUP/CHECKTABLE
  - See Paul's post for more details – Consistency Checking Options for a VLDB
    - https://www.sqlskills.com/ie0/checkdbvldb

Using the Check Database Integrity Task (before SQL Server 2016)

- Performs DBCC CHECKDB for all selected databases
- No ability to specify other parameters
  - e.g. PHYSICAL_ONLY
- Has option to include indexes or not
  - Recommended to include them

Using the Check Database Integrity Task in SQL Server 2016+

- Performs DBCC CHECKDB for all selected databases
- Can run using PHYSICAL_ONLY parameter
- Can set Max Degree of Parallelism (applicable in Enterprise only)
- Has option to include indexes or not
  - Recommended to include them

Ola Hallengren's Maintenance Scripts

- Provides the ability to perform DBCC CHECKDB against select databases, select tables within a database, and select databases based on Availability Group
- Provides the ability to run variations of integrity checks
  - e.g. CHECKFILEGROUP, CHECKCATALOG
- Provides the ability to specify options
  - e.g. PHYSICAL_ONLY
- Provides the option to include indexes or not
- Provides the option to check databases in a specific order, or in parallel
- Ability to set MAXDOP

Overview

- Index management
- The impact of shrink
- Statistics management
- Consistency checking
- Planning for data purging and ETL processes

Planning for ETL and Data Purging

- **ETL = Extract, Transform, Load**
  - A common term for moving data between systems
  - SSIS is often involved, but a variety of methods can be used
- **Data can need to be purged for any number of reasons**
  - May be a regular, scheduled process
  - May be a one-time event
- **Whether ETL and/or data purging occurs on an ad-hoc or recurring basis, you must be prepared for the effects of the data addition or data removal**
  - Index fragmentation
  - Additional logging
  - Invalidated statistics

SQLskills immerse yourself in sql server

Maintenance Tasks and ETL/Data Purging

- **Maintenance tasks should be scheduled around ETL and data purge tasks when possible**
- **Should a full backup be performed before the load/purge, or after?**
  - Before: creates a restore point in the event of a problem
  - After: captures all newly loaded information
- **Should transaction log backups be performed during the ETL/Data Purge?**
  - If the database is in FULL RECOVERY, yes. Transaction log size management is critical during operations that add or remove significant amounts of data
    - BULK_LOGGED is an option
- **What about index fragmentation and statistics?**
  - Both maintenance tasks should run after ETL/Data purge events
  - Address fragmentation introduced by the process
  - Provide accurate information to the optimizer if data distribution changes

SQLskills immerse yourself in sql server

Key Takeaways

- **In addition to backups, the essential maintenance tasks a DBA needs to setup and monitor are:**
  - Integrity checks
  - Fragmentation removal
  - Statistics updates
- **You can set up a maintenance plan for any of these tasks, but you will have better control (and typically less resource use) if you use custom scripts**
- **Don't re-invent the wheel, use Ola's scripts**
- **Understand the typical runtime for tasks, particularly integrity checks**

Resources (1)

- **Posts:**
  - SQLskills Accidental DBA Series
    - https://www.sqlskills.com/help/accidental-dba/
  - Importance of index maintenance
    - https://www.sqlskills.com/ie0/indexmaintenance
  - Inside sys.dm_db_index_physical_stats
    - https://www.sqlskills.com/ie0/indexphysicalstats
  - Database Maintenance Best Practices Part I – clarifying ambiguous recommendations for Sharepoint
    - https://www.sqlskills.com/ie0/maintenancesharepoint
  - Understanding When Statistics Will Automatically Update
    - https://www.sqlskills.com/ie0/whenstatsautoupdate
  - Managing SQL Server Statistics
    - https://www.sqlskills.com/ie0/managingstats

Resources (2)

- **Posts:**
  - Corruption demo databases and scripts
    - https://www.sqlskills.com/ie0/corruptiondbs
  - Tips and tricks for interpreting CHECKDB output
    - https://www.sqlskills.com/ie0/checkdboutput
  - Paul's CHECKDB From Every Angle Category:
    - https://www.sqlskills.com/ie0/checkdbpr
- **Whitepapers:**
  - Microsoft SQL Server 2000 Index Defragmentation Best Practices
    - http://technet.microsoft.com/en-gb/library/cc966523.aspx
    - Based on SQL Server 2000, so discusses DBREINDEX vs. INDEXDEFRAG
    - Concepts translate to 2005+
  - Online Indexing Operations in SQL Server 2005
    - http://technet.microsoft.com/library/Cc966402

SQLskills immerse yourself in sql server

Resources (3)

- **Pluralsight:**
  - SQL Server: Detecting and Correcting Database Corruption
    - https://www.pluralsight.com/courses/sqlserver-database-corruption
  - SQL Server: Advanced Corruption Recovery Techniques
    - https://www.pluralsight.com/courses/sqlserver-advanced-corruption-recovery-techniques
- **Scripts to automate maintenance tasks:**
  - Ola Hallengren
    - http://ola.hallengren.com/

SQLskills immerse yourself in sql server