SQL Server: Detecting and Correcting Database Corruption

Module 5: DBCC CHECKDB and Related Commands

Paul S. Randal
http://www.SQLskills.com/blogs/paul/
Paul@SQLskills.com

pluralsight
hardcore developer training

Introduction

- **Consistency checking is done using the DBCC CHECKDB command or by using some of the DBCC commands that run a subset of the DBCC CHECKDB functionality**
- **SQL Server Books Online for DBCC in general is at http://bit.ly/c2pV9l**

- **In this module we'll cover:**
  - DBCC CHECKDB overview
  - Other consistency checking commands
  - Consistency checks that DBCC CHECKDB performs
  - DBCC CHECK* options and behaviors
  - Potential problems
  - How to break up consistency checks over time

DBCC CHECKDB (1)

- **This is the main consistency checking command**
- **Runs all consistency checks against a single database**
- **Very simple to execute:**
  - DBCC CHECKDB (N'database')
  - Or change context to the desired database and just do DBCC CHECKDB
- **The only way to read all allocated pages in the database**
  - Use to force page checksums to be checked
- **Possible to change some of the behavior using a variety of options**
  - We'll cover these later
- **Very resource intensive with CPU, memory, and I/O**
- **Will use multiple threads on Enterprise Edition**
  - Parallelism can be disabled using documented trace flag 2528

DBCC CHECKDB (2)

- **Many algorithms have been added to minimize runtime and run online without blocking locks since SQL Server 2000**
  - Compared with 6.5 or 7.0, which used locking all the time
  - Performance increases from version to version
- **Features worth mentioning from SQL Server 2005 onward:**
  - Progress reporting
  - Data purity checks
  - "Last known good" stored in the database boot page
  - No false failures like in SQL Server 2000
- **My comprehensive blog post series is at http://bit.ly/TFvxmG**
- **Detailed information (~90 pages) in the *SQL Server 2008 Internals* and *SQL Server 2012 Internals* books**

Other DBCC CHECK* Commands (1)

- **DBCC CHECKALLOC**
  - Checks consistency of allocation structures in a database
    - After first checking the consistency of critical system tables
  - Examples:
    - Ensure allocation structures in the database are valid
    - Ensure no two tables have the same data file pages allocated
  - Functionality is part of DBCC CHECKDB
- **DBCC CHECKTABLE**
  - Checks consistency of a table and its indexes
  - Examples:
    - Ensure each data file page in the table and its indexes can be read and has a valid structure
    - Ensure each nonclustered index row has a matching table row
    - Ensure order of keys in indexes are correct
    - Ensure that any FILESTREAM data is correctly linked to the table
  - Functionality is part of DBCC CHECKDB

Other DBCC CHECK* Commands (2)

- **DBCC CHECKCATALOG**
  - Check relationships between system catalogs
    - E.g. column metadata must exist for all columns listed in a table's metadata, and vice-versa
  - Functionality is part of DBCC CHECKDB
- **DBCC CHECKFILEGROUP**
  - Perform consistency checks for a filegroup
  - Runs allocation checks for the database, then logical consistency checks for table and index partitions in the specified filegroup
  - Similar functionality to DBCC CHECKDB

Other DBCC CHECK* Commands (3)

- **DBCC CHECKIDENT**
  - Checks and can reset the identity value for a table
  - Functionality is not part of DBCC CHECKDB
- **DBCC CHECKCONSTRAINTS**
  - Checks FOREIGN KEY and CHECK constraints for validity
  - Functionality is not part of DBCC CHECKDB

What Exactly Does DBCC CHECKDB Do? (1)

- **Primitive checks of critical system tables**
  - If any problems are found, the only option is to restore from backups or try to extract data manually
- **Allocation checks**
- **Logical checks of critical system tables**
- **Logical checks of all other tables**
- **Metadata checks**
- **Service Broker data validation**
- **Indexed view, XML index, spatial index checks**
  - Only when the WITH EXTENDED_LOGICAL_CHECKS option is used, from SQL Server 2008 onward
- **If a repair option is specified, repairs are done at each stage if necessary and possible**

What Exactly Does DBCC CHECKDB Do? (2)

- **Another way to think of it is that DBCC CHECKDB does:**
  - Primitive system table checks, then
  - DBCC CHECKALLOC, then
  - DBCC CHECKTABLE of system tables, then
  - DBCC CHECKTABLE of user tables, then
  - DBCC CHECKCATALOG, then
  - Service Broker checks and other optional checks

- **It does not do:**
  - DBCC CHECKIDENT
  - DBCC CHECKCONSTRAINTS

Does DBCC CHECKDB Run During Startup?

- **When SQL Server starts, you may see messages in the error log about DBCC CHECKDB**
  - 2013-08-18 12:02:34.23 spid4s CHECKDB for database 'master' finished without errors on 2013-08-16 03:10:47.163 (local time). This is an informational message only; no user action is required.
- **This does not mean that DBCC CHECKDB was performed during instance startup**
- **It is reporting the "last known good" time for databases on the instance**
- **This is the time that DBCC CHECKDB last finished without finding any corruptions**
  - Stored in the boot page of the database (page 9 in file 1)
  - Not propagated to availability group replicas
  - Can be viewed using DBCC DBINFO (covered in the advanced course)

DBCC CHECK* Options (1)

- **NO_INFOMSGS**
  - Skip printing informational messages
  - Recommended to always use
- **ALL_ERRORMSGS**
  - Print all error messages
  - Not required from SQL Server 2008 SP1 onward
- **NOINDEX**
  - Skip checking nonclustered indexes
  - Not recommended
- **DATA_PURITY**
  - Perform column data validation
  - Default for all databases created on SQL Server 2005 or higher
    - Except master and model (it's a bug, see my blog at http://bit.ly/17mAGYQ)
  - Used only after upgrading from SQL Server 2000

DBCC CHECK* Options (2)

- **ESTIMATEONLY**
  - Estimate how much *tempdb* space is required
    - This functionality is broken in current builds of SQL Server 2008 R2 and 2012
- **TABLOCK**
  - Use locks instead of a database snapshot
  - Not permitted on the *master* database
  - Default for *tempdb*, where it causes allocation checks to be skipped
- **EXTENDED_LOGICAL_CHECKS**
  - Perform extra validation of some data structures
    - Indexed views, XML indexes, spatial indexes
- **PHYSICAL_ONLY**
  - Skips the majority of logical consistency checks
  - Reduces the time necessary for completion
  - Reduces CPU and memory resources necessary
- **Repair options will be discussed in Module 8**

How Are DBCC CHECK* Commands Online?

- **SQL Server 2000 used transaction log analysis, which was slow**
- **Since SQL Server 2005, DBCC commands use a database snapshot, which is a transactionally-consistent, point-in-time view of the database**
  - Not required if the database is read-only, single-user, or the target database is itself a database snapshot
- **The database snapshot is "hidden"**
  - Created as NTFS alternate streams of the existing data files
- **Sometimes problems can occur:**
  - The database snapshot runs out of space
  - No permissions to create the database snapshot
  - 3rd-party NTFS filter drivers that do not support alternate streams correctly
- **Solution: create your own database snapshot and check that**

How Long Will DBCC CHECKDB Take to Run?

- **Depends on many factors:**
  - Size of the database
  - Concurrent I/O load on the server
  - Concurrent CPU activity on the server
  - Concurrent update activity on the database
  - Throughput capabilities of the I/O subsystem
  - Number of CPUs on the server
  - Speed of the disks where *tempdb* is placed
  - Complexity of the database schema
  - Which options were specified
  - Number and type of corruptions that exist
- **See my blog post for more details at http://bit.ly/RRL570**

Potential Problems with DBCC CHECKDB (1)

- **Hidden database snapshots use NTFS alternate streams on existing database files**
  - Some 3rd-party software uses kernel filter drivers that do not cope with these correctly and return garbage data so DBCC CHECKDB reports corruption
- **The initial phase where the database snapshot is created cannot be interrupted due to limitations in the Engine**
  - This can make killing a DBCC CHECKDB impossible, and the SPID to appear as if it is in rollback
- **Database snapshots in general have an issue where they can run into NTFS limitations on size due to file-system fragmentation**
  - See http://support.microsoft.com/kb/2002606 for more details

Potential Problems with DBCC CHECKDB (2)

- **Beware of unexpected very long run times**
  - Most likely a corruption has triggered an algorithm to look deeper into the database structure to determine exactly where the problem is
- **Consistency checks make use of *tempdb* for temporary data storage, so *tempdb* must be sized appropriately**
  - Use WITH ESTIMATEONLY to get an estimate of how much space is required
  - Beware that in some current builds that functionality is broken

Breaking Up Consistency Checks Over Time

- **Several options for doing the equivalent of DBCC CHECKDB if it's not possible to restore to another server and run DBCC CHECKDB there**
  - E.g. no other system to use or the database is too large
- **Option 1:**
  - Use DBCC CHECKFILEGROUP to perform per-filegroup checks of all filegroups over a number of days, plus DBCC CHECKCATALOG
    - This is another benefit of using multiple filegroups when your database size becomes very large (e.g. more than 100 GB)
- **Option 2:**
  - Use DBCC CHECKTABLE to perform consistency checks of all tables over a number of days, plus DBCC CHECKALLOC and DBCC CHECKCATALOG
  - Can be based on time, table size, or some other factor

Summary

- **Most people just use DBCC CHECKDB but it is useful to understand the different consistency checking commands and their options**
- **Also useful to know what DBCC CHECKDB is doing and some of the problems that can occur with how it is implemented**

- **In the next module, we'll discuss:**
  - Interpreting the output of DBCC CHECKDB
  - Simple examples of corruption