http://www.sqlservercentral.com/articles/65804/
Printed 2009/02/16 06:56AM

Help, my database is corrupt. Now what?

By Gail Shaw, 2009/02/09

A corrupt database is probably one of most DBA's worst nightmares. It results in downtime, managers shouting and all other sorts of unpleasant things

In this article, I'm going to explain some of the things not to do to a corrupt database, and then go through some of the things that should be done, some of the scenarios and the fixes for those.

How to identify corruption

Corruption's typically pretty obvious when someone runs across the damaged pages. Queries fail with high severity errors. Backups or reindex jobs fail with high severity errors. Some messages that indicate corruption within a database are:

```
SQL Server detected a logical consistency-based I/O error: incorrect checksum (expected: 0xfdff74c9; actual: 0xfdff74cb). It occurred during a re
```

```
Attempt to fetch logical page 1:69965 in database 13 failed. It belongs to allocation unit 72057594049069056 not to 281474980642816.
```

The main problem is that, if regular database integrity checks are not being done, the corruption may be picked up hours, days or even months after it occurred, at which point it may be difficult to resolve

I'm not going to cover the situation where the database is in a suspect state. Covering the possible reasons why a database is suspect, the methods to discover why it is suspect and the various means of fixing that are a whole article in themselves, if not a full book.

What to do when the database is corrupt.

1. Don't panic
2. Don't detach the database
3. Don't restart SQL
4. Don't just run repair.
5. Run an integrity check
6. Afterwards, do a root-cause analysis

Don't panic

The most important thing when dealing with database corruption of any form is not to panic. Any decisions made or actions taken should be carefully thought through and made after careful consideration with all factors taken into account. It's very easy to make the situation worse with ill-thought through decisions.

Don't detach the database

While it is possible that the corruption message describes a transient condition, that is not the usual situation. In the vast majority of cases if SQL detects corruption within a database it means that there really are some damaged pages within the DB. Trying to trick SQL into not seeing that, by detaching and reattaching the database, backing up then restoring the database, restarting the SQL Service or rebooting the machine is not going to make corruption go away.

If there is corruption in the database, and SQL detects that corruption when it attaches the database, the attach will fail. There are ways to hack the database back into SQL, but it's much better to simply not detach the database in the first place

Don't restart SQL

Similar to detaching, restarting the SQL service will not fix corruption if it is present.

As with detaching the database, restarting the service may make matters worse. If SQL Server encounters corruption while performing the restart-recovery on a database, that database will be marked suspect, making any necessary repairs much harder to achieve.

Don't just run repair

It may be tempting to just run CheckDB with one of the repair options (typically allow data loss) and believe that it will make everything better. In many cases running repair is not the recommended fix. It is not guaranteed to fix all errors and it may result in unacceptable data loss.

Repair is, in most cases, the last resort for fixing corruption. It should be done only when none of the alternatives are possible, not done as the first thing tried.

Run an integrity check

To decide on a method of repairing the corruption, the details of exactly what is damaged must be known. The only way to get this information is to run CheckDB with the All_ErrorMsgs option (on 2005 SP3, this is a default option in CheckDB and does not need to be specified. It does need to be specified on SQL 2008). Additionally, the No_Infomsgs option removes all of the information about the number of rows and number of pages in the table, which is unnecessary when dealing with corruption.

CheckDB can take quite a while on larger databases, but it is necessary to let it run to completion. A repair strategy should not be considered without knowing all of the problems in the database

Root cause

Once the corruption has been resolved, the work isn't over. If the root cause of the corruption isn't found, it may happen again. Typically the leading cause of corruption is problems with the IO subsystem. Other possible causes are misbehaving filter drivers (like an antivirus), human intervention or a bug in SQL.

Next steps

The steps to resolve corruption depend entirely on the results of CheckDB. I'm going to go through some of the more common scenarios here. This is by no means a comprehensive document of all possible corruptions within a database. [1]

This list is in approximate order of severity, from the least severe problem to the most. Each has an example of possible error messages that indicate that particular problem. In general, the most severe error that CheckDB finds determines the methods available to resolve the corruption problem.

If anyone encounters an error that is not detailed here, see the last section – Obtaining Help.

Inaccurate space metadata

```
Msg 2508, Level 16, State 3, Line 1
 The In-row data RSVD page count for object "Broken1", index ID 0, partition ID 76911687695381, alloc unit ID 76911687695381 (type In-row data) i
```

This error indicates that the page has an incorrect value on it for the reserved space. In SQL 2000 it was possible for the row and page counts for a table or index to be incorrect, even negative. CheckDB did not pick this up. On SQL 2005, the counts should be correctly kept and CheckDB gives a warning when it finds this scenario

This is not a serious problem and it's trivial to fix. As the message says, run DBCC UPDATEUSAGE on the database in question and the warnings will disappear. This is common on databases upgraded from SQL 2000, for the reasons mentioned above, and should not occur on database created in SQL 2005/2008.

```
Msg 8914, Level 16, State 1, Line 1
 Incorrect PFS free space information for page (1:26839) in object ID 181575685, index ID 1, partition ID 293374720802816, alloc unit ID 76911687
```

This error indicates that the PFS page (page free space) that tracks how full pages are has incorrect values. This, like the above error, is not serious. The algorithm that tracked this in SQL 2000 was not always accurate. While fixing this does require running CheckDB with the Repair_Allow_Data_Loss option, if this is the only error that there is, it will not actually delete any data.

Corruption only in the nonclustered indexes

If all the errors that checkDB returns refer to indexes with IDs of 2 or greater, then it indicates that all of the corruption is within the nonclustered indexes. Since the data in a nonclustered index is redundant these corruptions can be repaired without data loss.

If all of the errors that CheckDB picks up are in the nonclustered indexes, the recommended repair level will be Repair_Rebuild.

```
Msg 8941, Level 16, State 1, Line 1
 Table error: Object ID 181575685, index ID 4, page (3:224866). Test (sorted [i].offset >= PAGEHEADSIZE) failed. Slot 159, offset 0x1 is invalid.

Msg 8942, Level 16, State 1, Line 1
 Table error: Object ID 181575685, index ID 4, page (3:224866). Test (sorted[i].offset >= max) failed. Slot 0, offset 0x9f overlaps with the prio
```

Those are just examples; there are many more possible errors.

In this case the corruption can be completely repaired by dropping the damaged nonclustered indexes and recreating them. Online index rebuilds (and some of the offline index rebuilds) read the old index to create the new and hence will encounter the corruption. Hence it's necessary to drop the old index completely and create a new one.

This is mostly what CheckDB with the repair_rebuild option will do, however the database must be in single user mode for the repair to be done. Hence it's usually better to manually rebuild the indexes as the database can remain online and in use while the affected indexes are recreated.

If there is insufficient time available to rebuild the affected index and there is a clean backup with an unbroken log chain, the damaged pages can be restored from backup.

Corruption in the LOB pages

```
Msg 8964, Level 16, State 1, Line 1
 Table error: Object ID 181575685, index ID 1, partition ID 72057594145669120, alloc unit ID 72057594087800832 (type LOB data). The off-row data
```

This indicates that there are LOB (large object) pages that are not referenced by any data row. This can come about if there was corruption of the clustered index or heap and the damaged pages were deallocated.

If these are the only errors that CheckDB returns, then running repair with the Allow_Data_Loss will simply deallocate these pages. Since the data rows that the LOB data belongs to does not exist, this will not result in further data loss.

Data Purity errors

```
Msg 2570, Sev 16, State 3, Line 17
 Page (1:1103587), slot 24 in object ID 34, index ID 1, partition ID 281474978938880, alloc unit ID 281474978938880 (type "In-row data"). Column
```

The data purity error indicates that there is a value in the column that's outside of the acceptable range of the column. That can be a datetime where the minutes past midnight exceed 1440, a Unicode string where the number of bytes is not a multiple of 2, or a float or real with an invalid floating point value.

These errors are not checked for by default on a database upgraded from SQL 2000 or lower. CheckDB must run successfully once, with the DATA_PURITY option before

CheckDB will not fix this. It doesn't know what values to put in the column to replace the invalid ones. The fix for this is fairly easy, but manual. The bad values have to annually updated to something meaningful. The main challenge is finding the bad rows. This kb article goes over the steps in detail. http://support.microsoft.com/kb/923247

Corruption in the clustered index or heap

If there is corruption to the clustered index's leaf pages or to the heap, it means that data has been lost. The leaf pages of the clustered index are the actual data pages of the table, and hence this is not redundant information.

If any of the errors that CheckDB picks up are in the leaf pages of the clustered index, the recommended repair level will be Repair_Allow_Data_Loss

```
Server: Msg 8976, Level 16, State 1, Line 2
 Table error: Object ID 181575685, index ID 1, partition ID 76911687695381, alloc unit ID 76911687695381 (type In-row data). Page (1:22417) was n

Server: Msg 8939, Level 16, State 1, Line 2
 Table error: Object ID 181575685, index ID 0, page (1:168576). Test (m_freeData >= PAGEHEADSIZE && m_freeData <= (UINT)PAGESIZE - m_slotCnt * si
```

Those are just examples; there are many more possible errors. The thing to notice is that the index ID mentioned is either 0 or 1. If any of the errors returned by CheckDB have an index id of 0 or 1, it means that there's damage to the base tables.

This kind of damage is repairable, but repairing it involves discarding rows or entire pages. If CheckDB deletes data to fix corruption, it will not check foreign keys and it will not fire triggers. The rows or pages will simply be deallocated. This can result in data integrity violations (child records without a parent) and it can result in logical database inconsistencies (nonclustered index rows or LOB pages that no longer reference a row). As such, repair is not the recommended route.

If there is a clean backup, restoring from the backup is usually the recommended method of fixing these errors. If the database is in full recovery and there is an unbroken log chain since the clean database backup, then it is possible backup the tail of the transaction log and to restore, either the entire database or just the damaged pages, with no data loss at all.

If there is no clean backup, then it's necessary to run CheckDB with the Repair_allow_data_loss option. This requires that the database be in single user mode for the duration of the repair.

It may be possible to determine what CheckDB will delete, for a clustered index. See this blog post - http://sqlskills.com/BLOGS/PAUL/post/CHECKDB-From-Every-Angle-Using-DBCC-PAGE-to-find-what-repair-will-delete.aspx

Corruption in the Metadata

```
Msg 3853, Level 16, State 1, Line 1
 Attribute (object_id=181575685) of row (object_id=181575685,column_id=1) in sys.columns does not have a matching row (object_id=181575685) in sy
```

This type of error usually appears in a database upgraded from SQL 2000, where someone did direct updates to the system tables.

There are no foreign keys enforced among the system tables in any version of SQL, so it was possible on SQL 2000 to delete a row from sysobjects (for example a table) and leave the rows in syscolumns and sysindexes that reference the deleted row.

On SQL 2000, CheckDB did not do a check of the system catalog, so this kind of problem often went completely unnoticed. On SQL 2005, CheckDB does do consistency checks of the system catalog, and so these errors can appear.

Fixing these is not trivial. CheckDB will not repair them, as the only fix is to delete records from the system tables, which may cause major data loss. If there's a backup of the database from before it was upgraded to SQL 2005 and the upgrade was very recent, then that backup can be restored to SQL 2000, the system tables fixed on SQL 2000 and then the database upgraded again.

If there is no SQL 2000 backup, or the upgrade was too long ago and the data loss is unacceptable, then there are two possible fixes. First, edit the system tables in SQL 2005, which is a complex and very risky process, as the system tables are not documented and are much more complex than they were on previous versions. See this blog post for details - http://www.sqlskills.com/BLOGS/PAUL/post/TechEd-Demo-Using-the-SQL-2005-Dedicated-Admin-Connection-to-fix-Msg-8992-corrupt-system-tables.aspx

The other solution is to generate scripts of all the objects in the database and export all of the data. Create a new database, recreate the objects and reload the data.

The second option is usually the recommended one.

Irreparable Corruption

CheckDB can't repair everything. Any errors like these are irreparable and the only way to resolve them is to restore a backup of the database that does not have the corruption. If there is a full, unbroken log chain from that backup up until the current time, then the log tail can be backed up and the database can be restored without any data loss.

If there is no clean backup, then the only remaining option is to generate scripts of the database objects and export the data that is accessible. It is quite likely, due to the corruption, that not all of the data will be accessible, and most likely not all of the objects will script without error.

Damaged system tables

```
Msg 7985, Level 16, State 2, Line 1
 System table pre-checks: Object ID 4. Could not read and latch page (1:358) with latch type SH.
 Check statement terminated due to unrepairable error.

Msg 8921, Level 16, State 1, Line 1
 Check terminated. A failure was detected while collecting facts. Possibly tempdb out of space or a system table is inconsistent.
```

CheckDB depends on a few of the critical system tables to get a view of what should be in the database. If those tables themselves are damaged, then CheckDB cannot know what the database should look like, and won't even be able to analyze it, let alone repair.

Damaged allocation pages

```
Msg 8946, Level 16, State 12, Line 1
 Table error: Allocation page (1:2264640) has invalid PFS_PAGE page header values. Type is 0. Check type, alloc unit ID and page ID on the page.
 Msg 8998, Level 16, State 2, Line 1
 Page errors on the GAM, SGAM, or PFS pages prevent allocation integrity checks in database ID 13 pages from (1:2264640) to (1:2272727)
```

In this case, one or more of the database allocation pages are damaged. The allocation pages are used to mark which pages and extents in the database are allocated and which are free. CheckDB will not repair damage to the allocation pages, as it is extremely difficult to work out, without those pages, what extents are allocated and which are not. Dropping the allocation page is not an option as that would discard up to 4GB of data.

Obtaining Help

If you're not sure what to do, get help. If you've run into a corruption message that you don't understand, that isn't described here, get help. If you're not sure the best way to recover, get help.

Help can come in many forms. If there's a senior DBA, ask them. If you have a mentor, ask them. Ask on a forum; the forums here, the Microsoft newsgroups or forums, or another forum if you prefer. Just be aware that not all advice given on the forums is good advice. In fact, there's some downright dangerous suggestions posted from time to time.

Finally, consider calling Microsoft's customer support people. They will charge, but they do know how to deal with corruption, and if it's a critical system database that's down, the cost of support may be far less than the cost of downtime while searching for a solution.

Conclusion

In this article I've given some suggestions on how to deal with corruption and, more importantly, how not to deal with corruption. I hope that people have a better understanding of the available methods of fixing such problems and of the importance of having good backups.

Footnote

1) Paul Randal has written an 80 page chapter for the forthcoming SQL Server 2008 Internals book that covers, in detail, how CheckDB works and has a comprehensive list of all errors it can produce.
