How To Create a Corrupt SQL Server Database for Test Purposes

By Pezzar 16. April 2012 08:52

It's quite possible that at some point you may want to have the use of a corrupted SQL Server database for test or DR practice purposes. This is very easy to achieve as I will detail below.

For this exercise we merely need a Hex editor and the use of a SQL Server instance.
Note: do not use a Production SQL Server instance!

I have chosen XVI32 as this editor is free of charge and requires no installation to take place, simply place the files into a folder and create a shortcut to the program.

The core database will be created using the following simple script. We'll go through the process in stages with diagrams to see exactly what's happening. Start with the code below;

Don't forget to modify any drive letters and paths before executing the script ;-)

```
USE [master]
CREATE DATABASE [Corrupt2K8] ON PRIMARY
( NAME =N'Corrupt2K8', FILENAME=N'C:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\Corrupt2K8.mdf',
SIZE = 524288KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON
( NAME = N'Corrupt2K8_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\Corrupt2K8_log.ldf',
SIZE = 262144KB , MAXSIZE = 2048GB , FILEGROWTH = 1024KB)
GO
USE [Corrupt2K8]
IF OBJECT_ID('dbo.NoddyTable','U') IS NOT NULL
BEGIN
DROP TABLE dbo.NoddyTable
END
CREATE TABLE dbo.NoddyTable(
NoddyID UNIQUEIDENTIFIER NOT NULL DEFAULT NEWID()
, NoddyName VARCHAR(128) NULL
, NoddyInt BIGINT NULL
, NoddyDate DATETIME NULL
)
INSERT INTO dbo.NoddyTable
SELECT NEWID(), name, ROUND(RAND(object_id)*856542, 0), GETDATE() FROM sys.columns
UNION ALL
SELECT NEWID(), name, ROUND(RAND(object_id)* 1048576, 0), GETDATE() FROM sys.columns

ALTER TABLE dbo.NoddyTable ADD CONSTRAINT PK_NoddyID
PRIMARY KEY CLUSTERED (NoddyID)
WITH (IGNORE_DUP_KEY=OFF)

CREATE NONCLUSTERED INDEX IDX_NoddyName_NoddyDate
ON dbo.NoddyTable(NoddyName, NoddyDate)
WHERE NoddyName IN ('password','length','created','crtype','offset','intprop')

CREATE NONCLUSTERED INDEX IDX_NoddyDate
ON dbo.NoddyTable(NoddyDate)
```

Enter search term Search

RecentPosts

Moving Database Files In SQL Server
Comments: 0
Not rated yet

AlwaysOn Availability Groups
Comments: 0
Rating: 5 / 1

How To Create A Corrupt SQL Server Database For Test Purposes
Comments: 0
Not rated yet

Using Sp_Change_Users_Login To Fix Orphaned Logins
Comments: 0
Rating: 3 / 1

Page List

Combining AlwaysOn AG With Failover Cluster Instances
Encrypted Backups Feature In SQL Server 2014
Implementing Microsoft ISCSI Initiator Policies & Multi Pathing
Moving Database Files In SQL Server
Repairing A Broken Log Shipping Plan From A Primary Differential Backup
SQL Server AlwaysOn Groups And FCIs Part1
SQL Server AlwaysOn Groups And FCIs Part2
SQL Server AlwaysOn Groups And FCIs Part3
SQL Server AlwaysOn Groups And FCIs Part4
Stairway To AlwaysOn HA Level 1
Stairway To AlwaysOn HA Level 2
Stairway To AlwaysOn HA Level 3
Stairway To AlwaysOn HA Level 4
Transparent Data Encryption On SQL Server
Using & Creating Mount Points In SQL Server

Category list

AlwaysOn Availability Groups (1)
Database Corruption (1)
File And Filegroups (1)
User Accounts (1)

Once you have the database created, take a full backup followed by a differential and then a transaction log backup, you may then use these in future testing scenarios.

We now want to choose an object as the target of our corruption exercise. I am going to choose a non clustered index on the table 'dbo.NoddyTable', to get a list of indexes on this table use the following query;

```
SELECT OBJECT_NAME(object_id), name, index_id, type_desc FROM sys.indexes
ORDER BY 1
```

I will be using the non-clustered index 'IDX_NoddyDate', this has an index id of 3. To find details of the page numbers in use by this index we now need to turn to an undocumented DBCC command 'DBCC IND'. Full details of how to use this command may be found at the links below but basically this is used as follows;

```
DBCC IND (DatabaseName,'tablename', index_id)
```

So, I have

```
DBCC IND (Corrupt2K8,'dbo.NoddyTable', 3)
```

Below is the output from the command above

```
DBCC IND(Corrupt2K8, 'dbo.NoddyTable', 3)

DBCC TRACEON(3604)
DBCC PAGE(Corrupt2K8, 1, 173, 1)
```

Results | Messages

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType
1	1	175	NULL	NULL	2105058535	3	1	72057594038976512	In-row data	10
2	1	174	1	175	2105058535	3	1	72057594038976512	In-row data	2
3	1	78	1	175	2105058535	3	1	72057594038976512	In-row data	2
4	1	79	1	175	2105058535	3	1	72057594038976512	In-row data	2
5	1	80	1	175	2105058535	3	1	72057594038976512	In-row data	2
6	1	89	1	175	2105058535	3	1	72057594038976512	In-row data	2

I'm going to pick a page of page type 2 (an index page), my chosen page number here is 174.

Next I need to go view a dump of this page just to have a look at the records it contains. This requires the use of another undocumented DBCC command called DBCC PAGE. Again full details of this are in the links below but basically it's used as follows;

```
DBCC PAGE (DatabaseName, filenumber, pagenumber, printoption)
```

So, I have the following code

```
--switch on client output first
DBCC TRACEON(3604)
--now read the page
DBCC PAGE (Corrupt2K8, 1, 174, 1)
```

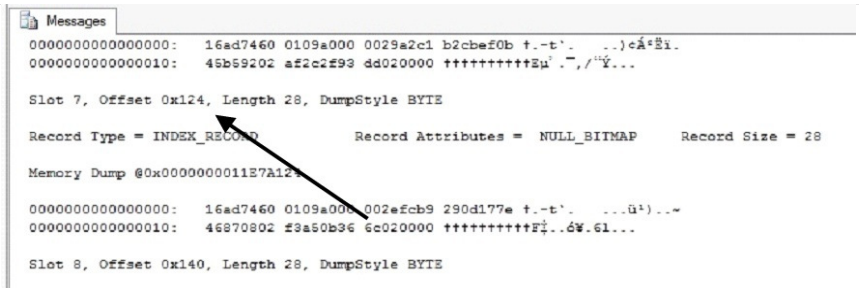
This is the page header;

```
DBCC TRACEON(3604)
DBCC PAGE(Corrupt2K8, 1, 174, 1)
```

Messages

```
PAGE: (1:174)

BUFFER:

BUF @0x000000008BFB7E80

bpage = 0x000000008B3FC000          bhash = 0x0000000000000000          bpageno = (1:174)
bdbid = 6                           breferences = 0                     bcputicks = 0
bsampleCount = 0                    bUse1 = 13634                       bstat = 0xc0010b
blog = 0x1212121b                   bnext = 0x0000000000000000

PAGE HEADER:

Page @0x000000008B3FC000

m_pageId = (1:174)                  m_headerVersion = 1                 m_type = 2
m_typeFlagBits = 0x0                m_level = 0                         m_flagBits = 0x4
m_objId (AllocUnitId.idObj) = 32    m_indexId (AllocUnitId.idInd) = 256
Metadata: AllocUnitId = 72057594040025088
Metadata: PartitionId = 72057594038976512                               Metadata: IndexId = 3
Metadata: ObjectId = 2105058535     m_prevPage = (0:0)                  m_nextPage = (1:78)
pminlen = 25                        m_slotCnt = 269                     m_freeCnt = 26
m_freeData = 7628                   m_reservedCnt = 0                   m_lsn = (35:530:24)
m_xactReserved = 0                  m_xdesId = (0:0)                    m_ghostRecCnt = 0
m_tornBits = 0
```

I'm going to home in on slot7 or record 7. I'll use the Hex editor to modify this record in the page which will then generate an error when DBCC CHECKDB is run. The detail for slot 7 looks as follows;

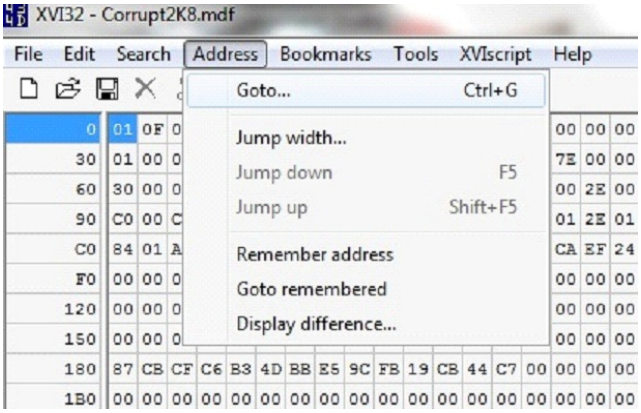


So, to hack the record at slot 7 on page 174, I first need to work out some figures to find the address locations within the file. Convert the record offset from hex to decimal first and then the address for slot 7 is calculated as follows

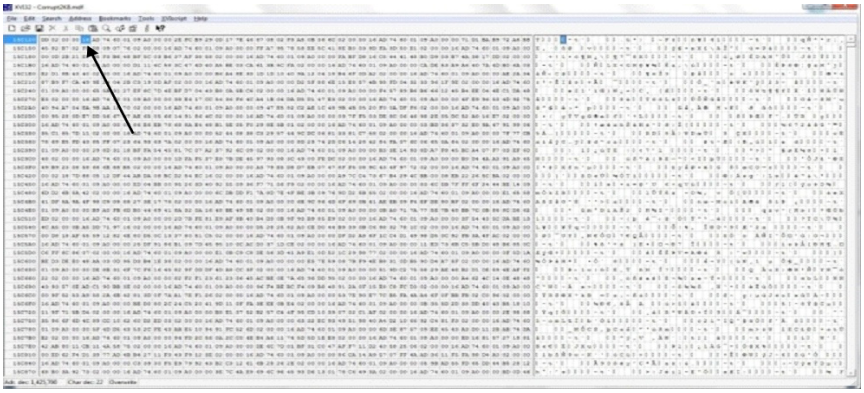
page number x num of bytes per page + record offset

This equates to 174 x 8192 + 292 = 1425700

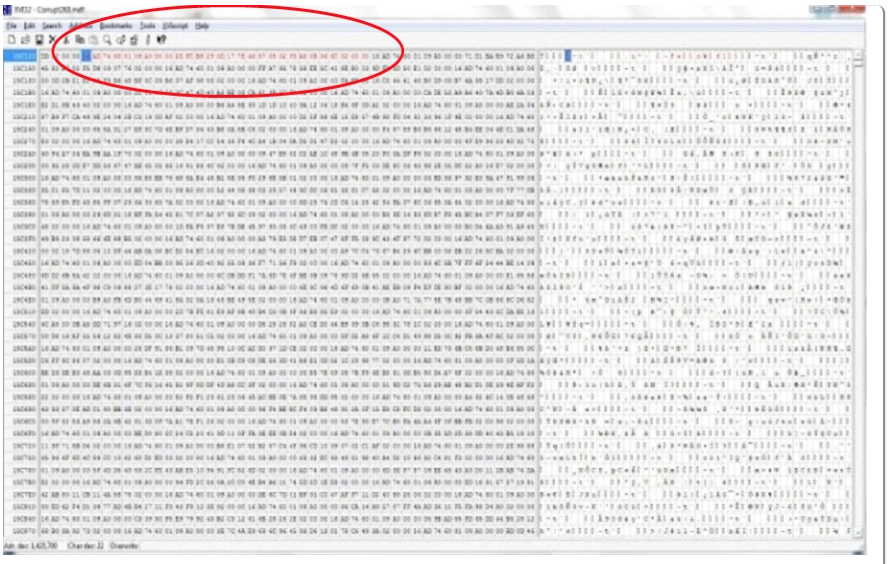
Take the database offline and now open the primary data file using XVI32. From the File menu select open and then browse to the MDF file.



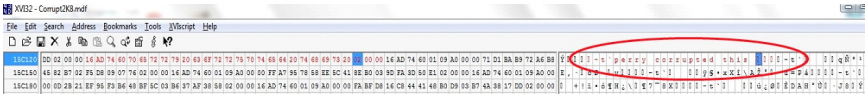
Now, from the File menu select "Address" > "Goto". In the dialog box which appears ensure you select the decimal radio button and enter the address which was calculated above, in my case 1425700. As you can see from the screenshot below, the editor has placed me at the start of my chosen record in page 174.



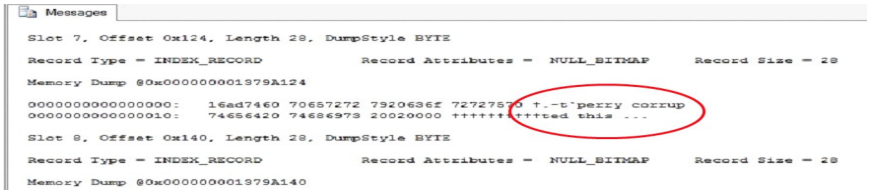
This record has a length of 28 bytes which was detailed in the page dump we did earlier, now to modify the record. First switch the editor to Text and Overwrite Mode if it isn't already. From the File menu ensure "Tools" > "Text Mode" and "Tools" > "Overwrite" are selected. Now I'll mark the blocks I wish to mangle. To do this, from the File menu select "Edit" > "Block <n> chars". Switching to decimal, I enter the record length of 28. The blocks have now been marked in red as shown below



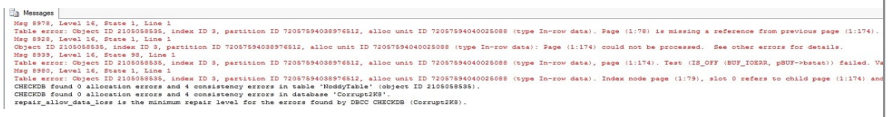
Now, to overwrite the record in slot 7 as shown below, in text mode type a simple string



Now click "File" > "Exit" and save the file when prompted to do so. In SQL Server you may now bring the database back online. We'll re run the page dump and check the results which are shown below;



Well, as we can see above the record was modified in the anticipated location, of course the only part hosed here is the non clustered index which is easily fixed by dropping and re creating it. What does DBCC CHECKDB show us?



Now you have a corrupt database which you may use for your DR and script tests. Give this a go in your test systems and by all means post back if you're stuck.

Credits

Information on these undocumented procedures was digested from Paul Randal's blogs at the following links

http://blogs.msdn.com/b/sqlserverstorageengine/archive/2006/12/13/more-undocumented-fun_3a00_-dbcc-ind_2c00_-dbcc-page_2c00_-and-off_2d00_row-columns.aspx

http://blogs.msdn.com/b/sqlserverstorageengine/archive/2006/08/09/692806.aspx

http://blogs.msdn.com/b/sqlserverstorageengine/archive/2006/06/10/625659.aspx

Warnings

Obviously you should not perform this on your production databases\servers, complete all tests in your offline environments. I will not be held responsible for those who wish to apply this is online environments

Tags: Database Corruption

E-Mail | Kick it! | DZone it! | del.icio.us Permalink | Comments (0)

Related posts

SQL Server AlwaysOn Groups and FCIs Part4
Welcome to Part 4 of my article detailing combining a Failover Cluster Instance of SQL Server into a...

SQL Server AlwaysOn Groups and FCIs Part1
Welcome to my latest article, which looks in detail at combining a Failover Cluster Instan...

Moving Database Files in SQL Server
In this article I will be discussing the moving of database files within a SQL Server instance. We'l...

BlogEngine.NET 2.5.0.6
Titanium X Theme by MGD King