ITPro with deep vertical Knowledge

PRACTICAL SQL SERVER

SQL Server Database Corruption, Part X: Page Level Restore Operations

overachiever | May 18, 2012

In my previous post in my ongoing series on SQL Server database corruption I covered a list of best practices for responding to database corruption when it happens. In that post I mentioned that I'd provide a follow-up post on the specifics of how to execute page-level restore operations from within SQL Server.

Page Level Restore Operations

As documented in Books Online, SQL Server's RESTORE DATABASE command has an optional parameter or switch that instructs it to only restore one or more pages instead of an entire database – which is very handy and helpful in dealing with situations in which only a hand-full of pages have been corrupted by some sort of problem with the IO subsystem.

Accordingly, the basic order of operations for this kind of operation is as follows

1. Kick the database into single user mode. You can set the database to RESTRICTED_USER, but if you have end-users or applications connecting using logins that map to the db_owner role, they'll

still be able to connect to the database. So use SINGLE_USER mode instead and do NOT close the query window you use to set to SINGLE_USER.

To set the database to SINGLE_USER mode, execute the following:

```
ALTER DATABASE <yourDbNameHere> SET SINGLE_USER
WITH ROLLBACK AFTER 10 SECONDS
GO
```

And note that the WITH ROLLBACK AFTER n SECONDS option can/will kill connections and on-going operations. It's documented in here (towards the end of the page). Likewise note that to pull this recovery operation off you're, OBVIOUSLY, kicking everyone out of the database.

2. Then, START your recovery process by backing up the tail-end of the log file – or the portion of your log file that hasn't already been backed up. (This is CRITICAL as it ensures that all operations up to a certain point have been accounted for. And you'll want to do this AFTER switching the database to SINGLE_USER mode – otherwise you risk allowing a few operations to 'sneak' by – which means that end-users or apps will have to re-enter that data after the system comes back online.)
For example:

```
BACKUP LOG yourDbNameHere
TO DISK = N'D:\SQLBackups\yourDbNameHere_TailEnd.trn'
GO
```

3. Execute the RESTORE DATASE command with the PAGE switch and loaded with as many pages as needed (i.e., those defined/referenced in msdb..suspect_pages).
Start this operation with your last FULL backup. For example:

```
RESTORE DATABASE yourDBNameHere
PAGE = 'fileid:pageid,fileid:pageid,etc'  -- e.g. 1:5224,1:5225,etc
FROM DISK = 'D:\SQLBackups\yourDbNameHere_lastFull.BAK'
WITH
NORECOVERY
GO
```

```
GO
```

Make sure that you **DO NOT** recover the database – by using the NORECOVERY option.

4. For EACH transaction log backup since either your FULL backup or your DIFFERENTIAL backup (if you had one), go ahead and apply each transaction log – making sure that you do NOT recover the database. For Example:

```
RESTORE LOG yourDbHere
FROM DISK = 'D:\SQLBackups\yourDbName_LogFileFrom2PM.TRN'
WITH NORECOVERY
GO
RESTORE LOG yourDbHere
FROM DISK = 'D:\SQLBackups\yourDbName_LogFileFrom215PM.TRN'
WITH NORECOVERY
GO
```

-- etc (i.e. more/similar log backups in sequence)
-- and so on… up to point of disaster

```
RESTORE LOG yourDbHere
FROM DISK = 'D:\SQLBackups\ yourDbName_LogFileFrom530PM.TRN'
WITH NORECOVERY
GO
```

And, again: DO NOT RECOVER the database along the way.

5. Apply the ‘tail end’ backup that you took in step 2, and then recover the database.

```
RESTORE LOG yourDbHere
FROM DISK = ‘D:\SQLBackups\yourDBName_TailEndBackupOfYourLogFileFromStep2.TRN’
WITH RECOVERY
```

-- Note that we're now RECOVERING the database.

6. Clear out msdb..suspect_pages by running:

```
DELETE FROM msdb..suspect_pages
GO
```

Make sure to do this step BEFORE executing the next step – as cleaning this table out is YOUR responsibility (i.e., it's not automatically cleared or anything by SQL Server when you run DBCC CHECKDB() or anything else) – and you don't want to risk confusing reports in this table of bad-pages with pages you've already corrected or accounted for.

7. Re-run DBCC CHECKDB() against your database – making sure to use the ALL_ERRORMSGS (and NO_INFOMSGS) options.

8. Switch the database back to multi-user mode (i.e. release it to production usage again) as follows:

```
ALTER DATABASE AdventureWorks SET MULTI_USER
GO
```

9. Make sure to clear out msdb..suspect_pages again IF needed (i.e., if you ran into any other issues along the way).

NOTE that with this Page-Level restoration approach, you're going to need to potentially do a LOT of manual scripting of Transaction log file restoration operations – something that can get a bit tedious. (It can also be potentially error-prone in a stressful situation. Therefore, if you're not sure you've applied a transaction log file already or not, just RE-APPLY it – there is NO penalty (other than time) for duplicate applications whereas you'll get big, scary/ugly, errors if you miss a log file.

To clarify that a bit: assume you've got log files A, B, C, D, E, and F after your last full/diff backup. And you apply A, B, and C – and then can't remember if you last applied C or D. In a case like this, just re-apply C if in doubt as it will NOT error-out on you if it's already been applied and you'll just waste the time involved to apply it at worst. (Whereas if you can't remember and jump from C to E, for example, you WILL get an error telling you that the LSNs are out of sequence – which is a big, ugly, scary message that will usually freak you out.))

Up Next

ENGAGE

As mentioned in a previous post, PAGE level recovery is a fantastic option – IF you've just got a small-ish number of pages that are damaged. If gobs of pages are damaged or if you've got large numbers of log files that you'd need to apply manually (since the SSMS GUI prior to SQL Server 2012 doesn't allow PAGE level restores – meaning that you have to manually handle all t-log applications), then you'll potentially want to look at doing a full-blown FULL recovery operation. Then, I'll provide a working set of examples that you can use for a full-blown 'corruption example' that you can play with and do page-level restores on in your own environment, and follow up with a summary link to all of the posts in this series.

Part XI: Full Recovery Operations

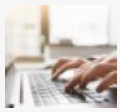
1 COMMENT

RELATED

The Challenge Is on: Maintain a Custom Letter-Based Sequence
APR 01, 2018

SPONSORED CONTENT
Accelerate Database Efficiency with SQL Performance Investigator (PI)
MAR 06, 2018

Hire a PI to solve the mystery behind plan change
MAR 06, 2018


SPONSORED CONTENT
Hire a PI to solve blocking locks mysteries
MAR 06, 2018

```
SELECT STRING_AGG(CHAR(n), '')
  WITHIN GROUP (ORDER BY CHAR(n) COLLATE Latin1_General_CS_
FROM dbo.GetNums(0, 255)
WHERE CHAR(n) COLLATE Latin1_General_CS_AS LIKE '[A-Z]';

SELECT STRING_AGG(CHAR(n), '')
  WITHIN GROUP (ORDER BY CHAR(n) COLLATE Latin1_General_BIN
FROM dbo.GetNums(0, 255)
WHERE CHAR(n) COLLATE Latin1_General_BIN LIKE '[A-Z]';
```

BUSINESS INTELLIGENCE > MICROSOFT SQL SERVER

The Challenge Is on: Maintain a Custom Letter-Based Sequence

If you need to implement a custom sequence solution, and you want it to perform as well as possible, you need to ask yourself which guarantees the solution needs to provide versus which it doesn't.

Itzik Ben-Gan | Apr 01, 2018

Recently I delivered an advanced T-SQL class in Europe, and one of my students, Nicolas Barreau of Avanade France SAS, presented a nice little puzzle. The task wasn't very complex, but it was interesting and fun to work on, and I hope you will enjoy working on it, too.

The Challenge

The challenge Nicola presented was to maintain a custom, letter-based, cycling, gap-allowing, sequence, with values in the form LLL, where L is an alphabet letter in the range A through Z (26 symbols). Nicola's company uses the sequence value as a match flag for settlement, for instance, between payment and invoice. The first value produced should be AAA, and it should advance as if the letters were digits in base 26, like so:

AAA

AAB

…

AAY

AAZ

READ MORE

TAGS: SOFTWARE DEVELOPMENT

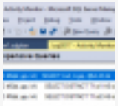
0 COMMENTS

RELATED

The Challenge Is on: Multi-Way Number Partitioning with T-SQL and SQL CLR
FEB 21, 2018

SQL Server: Avoiding a Sort with Descending Order
JAN 19, 2018

SQL Server: Avoiding a Sort with Merge Join Concatenation
DEC 18, 2017



T-SQL Puzzle Challenge: Grouping Connected Items
NOV 27, 2017