PRACTICAL SQL SERVER

BUSINESS INTELLIGENCE > MICROSOFT SQL SERVER

SQL Server Database Corruption, Part XI: Full Recovery Operations

overachiever | May 25, 2012

In part 9 of this series on SQL Server database corruption I defined a list of key things to do when responding to database corruption. And in that list of options and operations was the mention that in some cases you may have to revert back to using a full-blown recovery operation – meaning that

you'll need to completely restore your database from scratch. Accordingly, this post provides a bit more detail on what that kind of operation looks like – and is a supplemental 'follow-up' post (like the post before it which showed how to do page-level restores).

Restoring Entire Databases as a Means of Recovering from Corruption

Of course, in order to pull of full-blown recovery operations, you'll need a full-blown set of backups that provide complete coverage of your database since your last FULL backup. That, and it should go without saying that being able to execute a full-blown recovery operation is something that you should be already familiar with – as this operation has obvious scope and applicability outside of problems dealing with mere corruption.

However, just note that in cases where you are using a FULL recovery operation to recover from corruption, you'll be using your log-file backups to play PAST the time where corruption occurred (unlike what you MIGHT be doing as part of a point-in-time recovery where you're trying to restore to a point in time BEFORE an end user, for example, deleted an entire table). Otherwise, FULL recovery operations are pretty straight forward – and is going to be MUCH easier if you use SQL Server Management Studio's GUI or the GUI provided by your third party backup solution – as the approach outlined in Appendix 1 that involves scripting is really only useful for page-level restoration.

Accordingly, the order of operations is like so:

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

Where the number of seconds to wait before forcing a rollback is something you might want to modify.
BUT, the key thing here, obviously, is that in executing a full-blown recovery, you're OBVIOUSLY taking your database offline.

1. Set the database to SINGLE_USER_MODE – using syntax similar to that listed below:

2. Back up the tail end of the log. Again, this is CRITICAL to ensuring that data is not LOST – because you'll be using the transaction log's play-by-play record of all changes to the database to 'play through' an operation that completed and was written to disk – only to become 'lost or corrupt' when written to disk. And the hope, of course, is that when you replay this transaction or operation the second time (as part of recovery), it won't be

corrupted/lost when written to disk as whatever 'corrupted' the data previously was some sort of bug/problem/fluke at the disk subsystem level that should effectively NEVER repeat itself.
NOTE: Depending upon security settings/configuration and whether or not you're using a third party backup solution (or not) you might need to 'pop' your database out of SINGLE_USER mode in order to execute a log file backup. If you need to do this, try setting it to RESTRICTED_USER instead as that only allows dbo members access rather than what would, hopefully, be the 'unwashed masses' who might otherwise have access to the system while t-log backups are being executed.

3. Restore the database using your GUI. For an example of how to do this using native SQL Server backups, see the following video: Restoring Databases with SQL Server Management Studio. Or, use the GUI provided by your third party backup solution. In either case, this operation is something you should be VERY comfortable with – by means of regular practice.

4. If you had corrupt pages, make sure to clear msdb..suspect_pages by executing a DELETE command – just so that your NEXT problem isn't combined with this one (now that this issue or problem has been cleared up).

5. Once recovery is complete, release the database back into production by setting it to MULTI_USER as outlined above.

And, again, note that executing a FULL recovery operation is something that you should be VERY familiar with. As such, the best practices recommendation is that you execute tests on a regular basis – either by restoring copies of backup on another server (where you don't have to worry about collisions with existing databases) or where you execute these recovery operation on your existing server – but use the GUI to re-map file locations AND make sure to restore databases to a DIFFERENT database – instead of overwriting over the top of your production database. For more information and examples, see: Restoring Databases with SQL Server Management Studio.

Up Next

In the next installment of this series on corruption I'll provide a set of step-by-step instructions that you can use as a way to help practice dealing with recovery operations with corrupt databases – in the sense that I'll provide a 'demo' or 'script' you can follow-along with in order to corrupt a database, see the problems it causes, and then work through recovery operations on your own to get a good feel for what corruption looks like and how to correct it.

Part XII: Recovery Sample

TAGS: SOFTWARE DEVELOPMENT

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]';
```

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

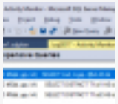
0 COMMENTS

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

Sign up for the ITPro Update newsletter.

Email address

United States

SIGN UP ▸

SECURITY > ENDPOINT SECURITY

Apple Is Ready for the Tech Backlash

The only real guarantee that companies won't go overboard in invading customers' privacy is that they have large revenue streams which make it an unnecessary risk.

Bloomberg | Mar 30, 2018

(Bloomberg View) --It's easy to understand Apple CEO Tim Cook's somewhat self-righteous comments on the Facebook-Cambridge Analytica scandal: His company has never emphasized customer data monetization, and privacy has been one of its persistent selling points. Apple still collects our data — and reserves the right to share it — but the company has a much better track record than newer tech titans.

"The truth is, we could make a ton of money if we monetized our customer — if our customer was our product," Cook said on Wednesday. "We've elected not to do that." Asked what he'd do if he were in Facebook CEO Mark Zuckerberg's shoes, he replied, "I wouldn't be in this situation."

In reality, Apple collects more information about its customers than Facebook because it offers more products and services. It knows everything from an iPhone's location to its owner's taste in