MSSQLTips.com
brought to you by Edgewood Solutions

Your daily source for SQL Server Tips

Using Page Level Restore as a Disaster Recovery Procedure in SQL Server 2005

Written By: Edwin Sarmiento -- 12/12/2008

Problem
In a previous tip on Disaster Recovery Procedures in SQL Server 2005 Part 1, we have seen how we can come up with a disaster recovery procedure in SQL Server 2005. There are other ways to increase availability of your highly critical database in SQL Server 2005. What are those other options?

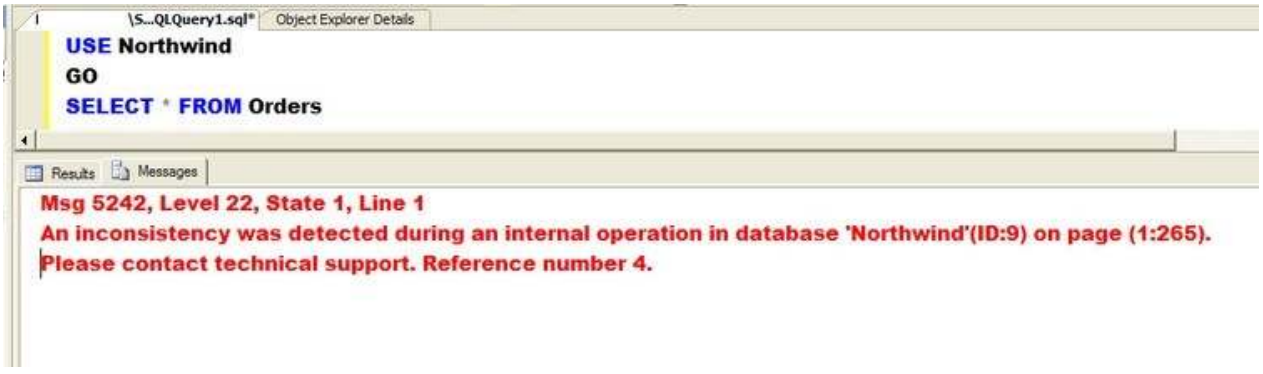
Solution
To continue the series (Disaster Recovery Procedures in SQL Server 2005 Part 1, Disaster Recovery Procedures in SQL Server 2005 Part 2 (Isolating Critical Objects), Disaster Recovery Procedures in SQL Server 2005 Part 3 (Using Partitioned Tables with Multiple Filegroups for High Availability)) on disaster recovery scenarios in SQL Server 2005, let us look at another means to increase the availability of your highly critical databases. There are cases where your very large databases may get corrupted. This may be caused by faulty disk driver or controller or sometimes by a incorrectly configured anti-virus software running on the SQL Server machine. You definitely need to have a full database backup to get the corrupted data back. But you wouldn't want to restore the entire database if only a few pages are corrupted. SQL Server 2005 introduced the concept of page-level restore. This gives you the option to restore one or more damaged pages without restoring the entire database. In the enterprise edition, page restores are performed online when conditions allow, which means higher availability for your databases.

Let's have a look at how to use page-level restores on a corrupted database. I will still be using the **Northwind** database for this tip - except that the database version that I will be using is corrupted. Note that it is not that easy to generate a corrupt database in your production environment so you would have to create one on your own to test these procedures. What I did was to use a hex editor to modify the values of the database file. This would introduce inconsistencies in the database file causing it to be corrupt. You would need to do some trial-and-error to get a specific page corrupted. In my case, I have chosen to corrupt a data page affecting the **Orders** table as it would be easy to find the text values of the records inside this table (as compared to the Order Details table that I have been using in the previous examples which contains mostly numerical data) from the hex editor.

Running a query on the corrupted **Orders** table would return partial results even though a corrupted page was detected on the Northwind database. Looking at the **Results** tab, it looks like that the damaged record pertain to row number 673 (the original Orders table contains 830 records).

671	10918	BOTTM	3	1998-03-02 00:00:00.000	1998-03-30 00:00:00.000	1998-03-11 00:00:00.000	3	48.83	B
672	10919	LINOD	2	1998-03-02 00:00:00.000	1998-03-30 00:00:00.000	1998-03-04 00:00:00.000	2	19.80	LI

In my case, from the error message, page 265 of file 1 is inaccessible.

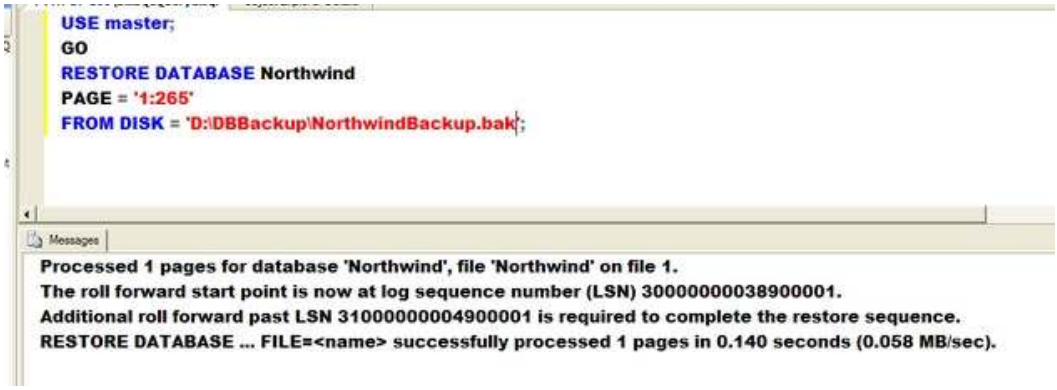


Let's run the **DBCC CHECKDB** command to check the integrity of all the objects in the **Northwind** database. The result simply confirms that page 265 - the **Orders** table - of the database cannot be processed.

```
There are 0 rows in 0 pages for object "sys.sysbinsubobjs".
DBCC results for 'Orders'.
Msg 8928, Level 16, State 1, Line 1
Object ID 21575115, index ID 1, partition ID 282888923447296,
alloc unit ID 1413946736640 (type In-row data): Page (1:265) could not be processed.  See other errors for details.
Msg 8944, Level 16, State 18, Line 1
Table error: Object ID 21575115, index ID 1, partition ID 282888923447296,
alloc unit ID 282888923447296 (type In-row data), page (1:265), row 4.
Test (columnOffsets->offTbl [varColumnNumber] >= priorOffset) failed. Values are 0 and 194.
Msg 8944, Level 16, State 18, Line 1
Table error: Object ID 21575115, index ID 1, partition ID 282888923447296,
alloc unit ID 282888923447296 (type In-row data), page (1:265), row 4.
Test (columnOffsets->offTbl [varColumnNumber] >= priorOffset) failed. Values are 0 and 194.
```

Now, if this was a very large database, you wouldn't just restore the full database backup. That would render your database inaccessible during the time that it is being restored and that's the last thing you want to do, especially if only a few pages are damaged. Notice that even though there are corrupt pages in the database, all the other objects are still accessible and we want to keep it that way. For this scenario, we can just use the page-level restore option in SQL Server 2005. To do that, we specify the file and page numbers that we got from the **DBCC CHECK** command in our **RESTORE DATABASE** command.

```
USE master
GO
RESTORE DATABASE Northwind
PAGE = '1:265'
  FROM DISK = N'D:\DBBackup\NorthwindBackup.bak'
GO
```



If your database contains more than one damaged page, simply add the file and page numbers accordingly in the **PAGE** parameter and separated by commas.. You can use either a full, file or filegroup backup to do the page restore provided that the backups that you use are valid and contain copies of the damaged pages. I simply used a full database backup in this example to demonstrate how to use the page-level restore option. This restore process simply replaces the pages specified in the **RESTORE DATABASE** command. Notice that the message returned by the **RESTORE DATABASE** command gives us a hint on what to do next. As always, we backup the tail of the log to complete the roll forward process.

```
BACKUP LOG Northwind
  TO DISK = N'D:\DBBackup\Northwind_log.TRN'
  WITH INIT, NO_TRUNCATE,
  STATS = 10
GO
```

Finally, we restore the tail of the log which we backed up as part of the recovery process to get the database back online.

```
RESTORE LOG Northwind
  FROM DISK = N'D:\DBBackup\Northwind_log.TRN'
  WITH RECOVERY,
  STATS = 10
GO
```

You can validate if the restore process is successful by running a query on the **Orders** table. The page-level restore option enables us to restore a damaged portion of the database as quickly as possible without having to restore the entire database. Your restore sequence will depend on the types of backups you are doing as this is a more granular approach in recovering a database. Let's say you have regular transaction log backups running every hour, you will need to restore those log backups in sequence before backing up the tail of the log as you need to keep the unbroken chain of log backups up to the current log file so that the page is brought up to date with the current log file.

Next Steps

- Simulate this particular process by going thru the steps outlined above.
- You can download the corrupt Northwind database files and backups used in the sample here. The database backup is so that you have a consistent database state before you proceed with testing the procedures outlined above.
- Check out the earlier tips in this series:
  - Disaster Recovery Procedures in SQL Server 2005 Part 1
  - Disaster Recovery Procedures in SQL Server 2005 Part 2 (Isolating Critical Objects)
  - Disaster Recovery Procedures in SQL Server 2005 Part 3 (Using Partitioned Tables with Multiple Filegroups for High Availability)