Database Corruption Challenge Week 7 - Alternate Solution

The alternate solution to the Database Corruption Challenge this week was created by Patrick Flynn. This solution is the only solution to successfully recover all the data without using any of the backups. If the challenge had been structured differently and all you had was just the database files and no backups, this solution would work just fine.

As you read through this solution, you will notice the use of Common Table Expressions throughout the solution. Patrick is the first person that I have seen write TSQL code that uses Common Table Expressions more than I do. (I mean that as a compliment Patrick)

To start with let's add the UDF_Convert_Hex_to_Binary function into the master database to match what Patrick had in his system.

All code provided by Patrick, comments and screenshots by me.

Once that function is created, let's drop the previous CorruptionChallenge7 database, because this is probably not our first time running through the solution. If you have a different way of checking for a database and dropping it, please see my blog post related to this topic earlier today.

Now we will use the MDF, LDF, and NDF files provided in the solution, and attach them as the CorruptionChallenge7 database.

Now we have a database that we are ready to work on figuring out where the rows in the [Record1000] table went. So let's see what we have in the current database log that hasn't yet been backed up.

Now there is a lot there to sort out.

Now to calculate the number of columns in each row deleted.

Now that the number of columns has been calculated we calculate the NullBitMapLength. All of these will be used later to extract the actual rows from the log data.

Next the NullBytes and TotalNoofVarColumsn fields are calculated and filled in.

Three more fields to calculate, the ColumnOffsetArray, the VarColumnStart, and the NullBitmap.

Now we need to figure out the columns for this table.

Now to add some more column information.

Next we create a place to store the recovered data and fill it in.

Now let’s test the query to pull data out of the #RecoverdData table and PIVOT it to something we can use.

Now its starting to look like something we can use.

Since the data looks good, we now put it back into the table it was originally deleted from.

Data restored, and all is good, now we run the checksum query to confirm, and … we get the right results.

Clean things up.

And there you have it, recovering 3010 delete rows without going to a backup. In fact this solution, assuming the database already existed and didn't need to be attached, could be done with no downtime, and without bumping any users out of the system.

You can download the PatrickFlynn2NoRestore.zip.

Nice solution Patrick!