Should I Use A Unique Identifier As A Primary Key?

Take the following sample code. Four similar tables with an INT IDENTITY, BIGINT IDENTITY, and two with UNIQUEIDENTIFIERS, one using newid() and the other with newsequentialid().

Once the tables are created and we insert 100,000 rows into each, they are very similar, but they take up very different amounts of disk space.

Using the Database Health Reports application, we can easily check in on the disk space used.

From this you can see that the total spaced used by either of the uniqueidentifiers is 31% larger than the BIGINT examples.

But looking back at the sample code you might argue that it isn't a very realistic example because there is not primary key with a clustered index (common practice).

So lets try it again with a primary key with a clustered index on each of the tables.

Again, using the Database Health Reports application, I check in on the disk space used, which shows the following:

Here you can see that the amount of space taken up by the uniqueidentifier compared to BIGINT is over twice the size. This additional size is mostly accounted for in the fragmentation caused by clustering on a uniqueidentifier.

Lets take a look at the index fragmentation, again using the Database Health Reports.

Here you can see that the uniqueidentifer test table is 99.4% fragmented. As you continue to add more to this table, it will continue to be defragmented.

Summary

Using uniqueidentifers as primary keys, and as clustered indexes can lead to trouble over time.

1. I suggest that if you can avoid it, you should avoid a uniqueidentifier for a primary key.

2. If you must use a uniqueidentifier, the use NEWSEQUENTIALID ( ) to generate the unique identifer, not NEWID().