newid() vs newsequentialid() What are the differences/pros and cons?

29

In a database where all of your primary keys are GUIDs, what are the differences/implications and/or pros and cons using newid() versus newsequentialid() as the "default value or binding".

The only difference that I know of is that newid() creates a new random GUID as opposed to newsequentialid() creates a new GUID based on the last one that is in the table in an incremented fashion.

sql-server-2005 primary-key identity

8

asked Oct 19 '09 at 6:08

7wp
7,113 17 64 97

4 Answers

28

As I understand it, when you perform an insert in a row the DB, it will be inserted in order relative to the other PK's in the table. With a normal guid, this could be anywhere in the table. A newsequentialid() will always be added to the end of the table.

So the performance of inserts is improved.

This site explains the differences and benchmarks between the two different methods.

Update - the blog post referenced has been moved. The link now refers to an web.archive.org link. Here is the key takeaway:

> Most striking is the number of writes required by the NEWID system function. This, coupled with the average page density of 69%, is evidence of the page splitting caused by the random distribution of inserts at the leaf level. As soon as a page fills up, it needs to be split into 2 pages of 50% each for the insert to complete. Not only has page splitting resulted in poor page density, it has fragmented the data pages quite badly (there is a 99% probability that the next data page is not next to the current one). In our tests the most likely place for a free page required for the page split is at the end of the table irrespective of where the row is being inserted. Therefore to read the rows in order the scan needs to keep jumping back and forth between widely distributed split pages, hence the appalling fragmentation.

--Stefan Delmarco

edited Jul 23 '16 at 4:02

answered Oct 19 '09 at 6:14

ChadT

6,398 2 35 54

Huh? If they are generated sequantially How can it be guarneteed that the GUID generated is actually unique, instead of just being something that looks like a GUID? – Justin Feb 18 '10 at 15:00

4 @Justing - There is no *guarantee* that a GUID is unique, it's just very likely to be. – Hans Kesting Dec 29 '12 at 19:02

1 Links are volatile... Fotia now 404s the link in the answer. – Marc L. May 13 '14 at 18:52

2 web.archive.org/web/20110526141832/http://www.fotia.co.uk/fotia/… – RoboJ1M Sep 25 '14 at 9:01

13

Regarding the use of sequential keys (as with identity, sequence, and NEWSEQUENTIALID) vs. nonsequential ones (as with NEWID or a custom randomized key generator), there are several aspects to consider.

Starting with sequential keys, all rows go into the right end of the index. When a page is full, SQL Server allocates a new page and fills it. This results in less fragmentation in the index, which is beneficial for read performance. Also, insertions can be faster when a single session is loading the data, and the data resides on a single drive or a small number of drives.

Consider nonsequential keys, such as random ones generated with NEWID or with a custom solution. When trying to force a row into an already full page, SQL Server performs a classic page split—it allocates a new page and moves half the rows from the original page to the new one. A page split has a cost, plus it results in index fragmentation. Index fragmentation can have a negative impact on the performance of reads. However, in terms of insert performance, if the storage subsystem contains many spindles and you're loading data from multiple sessions, the random order can actually be better than sequential despite the splits.

That's because there's no hot spot at the right end of the index, and you use the storage subsystem's available throughput better. A good example for a benchmark demonstrating this strategy can be found in a blog by Thomas Kejser at http://blog.kejser.org/2011/10/05/boosting-insert-speed-by-generating-scalable-keys/.

Source: Querying Microsoft® SQL Server® 2012 Exam 70-461 Training Kit

answered Sep 9 '13 at 15:45

Farzin
131 1 3

1

From my understanding, when the SQL instance fires up the NEWSEQUENTIALID GUID is initialised to a random value. Then for the life of its operation GUIDs are incremented on the central GUID, NOT by looking at the last GUID generated for the table.

answered Jun 24 '11 at 5:15

Todd
11 1

1 The benefit of doing this is in the indexes. Complete random GUIDs fragment the indexes too much, sequential GUIDs are good for insertion and indexing performance. – Todd Jun 24 '11 at 5:17

Microsoft states that it is sequential with regards to the last GUID generated since the host machine was last started. – Lars Gyrup Brink Nielsen Dec 15 '16 at 21:31

As I know, `NEWID()` generates the `GUID` in random order and `NEWSEQUENTIALID()` generates the `GUID` in sequential order.

5 You didn't add anything to what was already written in the question. – Dale Burrell Feb 12 '14 at 3:14