VISCOSITY
NORTH AMERICA

On the Second Day of 18c/19c, Viscosity Gave to me...

Scalable Sequences

December 9, 2020

Technology Overview

Database: 18c / 19c

Company: Oracle Corporation

Topic: Scalable Sequences

Viscosity can help with any of your Database Upgrade needs

Viscosity has performed numerous zero-downtime database migrations and upgrades over the years and has a proven track record with business critical and mission critical databases.

Viscosity's Database Migration & Upgrade Services can plan, upgrade, validate and migrate all database content - quickly and effectively with our automated approach and proven methodology.

Learn more about how you can maintain and maximize your investments at viscosityna.com or email us at hello@viscosityna.com.

www.viscosityna.com

Most companies leverage database sequences as monotonically increasing number generators to populate synthetic primary key and unique indexes. As applications leverage sequentially generated integers and continuously access the right-most leaf block during heavy workloads, performance can degrade significantly on RAC databases as buffer block contention is introduced.

Historically, Viscosity has implemented Reverse Key Indexes or Hash Partitioned Indexes, and have even changed the block size of the tablespace/database, but the application/schema had to be refactored to recognize the performance gains. Oracle introduced and documented scalable sequences in Oracle Database 18c, to eliminate the hot index contention issue in RAC during large scale data insertions.

Before we begin our review of scalable sequences, let's leverage the Oracle built-in namespace, sys_context, to retrieve information about our current session, instance, and session id, as they are relevant to our scalable sequence discussions:

```
SQL> col instance for a20
SQL> col sid for a20
  1  select sys_context('USERENV', 'INSTANCE')
Instance,
  2     sys_context('USERENV', 'SID') SID
  3* from dual
SQL> /

INSTANCE             SID
-------------------- --------------------
1                    47
```

Next, let's create a scalable sequence called myseq with the *scale* keyword.

```
SQL> create sequence myseq scale;

Sequence created.

SQL> set numwidth 32
SQL> select myseq.nextval from dual;

                         NEXTVAL
--------------------------------
    10104700000000000000000004
```

We can dissect the first 6 digits of the sequence number offset to 2 categories:

- Instance ID
- Session ID

The first 3 digits represent the instance offset number: [(instance id % 100) + 100] => 101
The second 3 digits represent the session offset number:
(session id from session information % 1000) = 1047/1000 = remainder of 047

The remaining 22 digits have 0's padded with the last digit of our sequence number being 4.

When we CREATE or ALTER a sequence, we have the option to specify SCALE|NOSCALE. NOSCALE is the default meaning that it will be the normal sequence. Another option, is to specify EXTEND | NOEXTEND where the NOEXTEND option is the default. In the example below, we have created a sequence called hca_seq_scale_noextend_01 with the SCALE option, but with the max value of 12 digits:

```
create sequence hca_seq_scale_noextend_01 scale maxvalue
999999999999;
```

Remember that NOEXTEND is the default, so we do not need to specify that option. Next, let's select the next value from the sequence:

```
SQL> select hca_seq_scale_noextend_01.nextval from dual;

                         NEXTVAL
--------------------------------
                   101047000001
```

If you look at this output, the numeric values have a total of 12 digits. Remember, the first 6-digit offset represents the instance and session id information. The remaining 6-digits represent the sequence values.

Let's create another sequence called hca_seq_scale_extend_01 with both SCALE and EXTEND keywords:

```
SQL> create sequence hca_seq_scale_extend_01 scale extend maxvalue
999999999999;

Sequence created.

SQL> select hca_seq_scale_extend_01.nextval from dual;

                         NEXTVAL
--------------------------------
              101047000000000001
```

Dissecting the output of 101047 000000000001:

- 101047 represents the 6-digit offset
- 000000000001 represents the 12-digit sequence which represents the 12 x 9's in our example for maximum value.

We can query the dictionary view dba_sequences to identify scalable sequences:

```
  1  SELECT  sequence_name
  2       ,scale_flag
  3       ,extend_flag
  4       ,max_value
  5  FROM   dba_sequences
  6  where sequence_name in
  7       ('MYSEQ', 'HCA_SEQ_SCALE_NOEXTEND_01',
  8*       'HCA_SEQ_SCALE_EXTEND_01')
SQL> /

SEQUENCE_NAME                SCALE_FLAG      EXTEND_FLAG     MAX_VALUE
---------------------------  --------------- --------------- --------------------------------
MYSEQ                         Y              N               99999999999999999999999999999
HCA_SEQ_SCALE_EXTEND_01       Y              Y               999999999999
HCA_SEQ_SCALE_NOEXTEND_01     Y              N               999999999999
```

The two columns of particular interest are the scale_flag and extend_flag colums. As you can see, the default max value is 28 digits. If we create a scalable sequence with the default options, although it does not create with the EXTEND option, the default digit size is 28 for the max value.

We can alter a scalable sequence to have SCALE|NOSCALE and to EXTEND|NOEXTEND. Below is an example with the sequences that we created earlier:

```
SQL> alter sequence hca_seq_scale_extend_01 scale noextend;

Sequence altered.

SQL> alter sequence myseq noscale;

Sequence altered.
```

In summary, high concurrency of inserts with right-hand leaf nodes of the B-tree indexes leads to performance issues. With Oracle RAC, this performance bottleneck is exponentially magnified, as hot blocks are transferred across the interconnect. With scalable sequences, because we are generating unordered values for the primary or unique keys, we will notice significant reduction to index block contention caused by the right-handed indexes. Thus, yielding better throughput, scalability with batch jobs, and data loads resulting in better performance and response times. For all new applications on RAC, consider scalable sequences as your new standard.

One of the sayings we have at Viscosity is our customer's, "have four aces in their pocket". Over the next 11 days, the talented staff at Viscosity along with our Oracle ACEs will address more Oracle Database 18c and 19c new features. Continue to join us next year, as we continue our Oracle Database 19c hands-on-lab workshops.

Happy Holidays!