

Sequences



What Will I Learn?

In this lesson, you will learn to:

- List at least three useful characteristics of a sequence
- Write and execute a SQL statement that creates a sequence
- Query the data dictionary using USER_SEQUENCES to confirm a sequence definition
- Apply the rules for using NEXTVAL to generate sequential unique numbers in a table
- List the advantages and disadvantages of caching sequence values
- Name three reasons why gaps can occur in a sequence



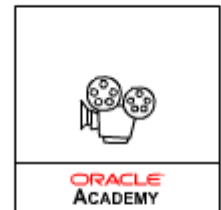
Why Learn It?

Can you image how tedious it would be to have to enter the names of the 30,000 people who enter the London Marathon into a database, while making sure that no one was given the same identification number?

What if you went to lunch and when you returned, someone else had entered some of the runners' applications?

How would you know where to start again?

Fortunately, SQL has a process for automatically generating unique numbers that eliminates the worry about the details of duplicate numbers. The numbering process is handled through a database object called a **SEQUENCE**.



Tell Me / Show Me

THE SEQUENCE OBJECT

You already know how to create two kinds of database objects, the TABLE and the VIEW.

A third database object is the SEQUENCE. A SEQUENCE is a shareable object used to automatically generate unique numbers.

Because it is a shareable object, multiple users can access it. Typically, sequences are used to create a primary-key value.

As you'll recall, primary keys must be unique for each row. The sequence is generated and incremented (or decremented) by an internal Oracle routine. This can be a time-saving object because it reduces the amount of code you need to write.





Tell Me / Show Me

THE SEQUENCE OBJECT (continued)

Sequence numbers are stored and generated independently of tables. Therefore, the same sequence can be used for multiple tables.

To create a **SEQUENCE**:

```
CREATE SEQUENCE sequence  
  [INCREMENT BY n]  
  [START WITH n]  
  [{MAXVALUE n | NOMAXVALUE}]  
  [{MINVALUE n | NOMINVALUE}]  
  [{CYCLE | NOCYCLE}]  
  [{CACHE n | NOCACHE}];
```

Tell Me / Show Me

CREATE SEQUENCE
sequence

[INCREMENT BY n]

[START WITH n]

[{MAXVALUE n |
NOMAXVALUE}]

[{MINVALUE n |
NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{CACHE n | NOCACHE}];

sequence	is the name of the sequence generator (object)
INCREMENT BY n	specifies the interval between sequence numbers where n is an integer (If this clause is omitted, the sequence increments by 1.)
START WITH n	specifies the first sequence number to be generated (If this clause is omitted, the sequence starts with 1.)
MAXVALUE n	specifies the maximum value the sequence can generate
NOMAXVALUE	specifies a maximum value of 10^{27} for an ascending sequence and -1 for a descending sequence (default)
MINVALUE n	specifies the minimum sequence value
NOMINVALUE	specifies a minimum value of 1 for an ascending sequence and $-(10^{26})$ for a descending sequence (default)
CYCLE NOCYCLE	specifies whether the sequence continues to generate values after reaching its maximum or minimum value (NOCYCLE is the default option.)
CAHCE n NOCACHE	specifies how many values the Oracle server pre-allocates and keep in memory. (By default, the Oracle server caches 20 values.) If the system crashes, the values are lost.



Tell Me / Show Me

CREATING A SEQUENCE

In the SEQUENCE created for the London Marathon runners, the numbers will increment by 1, starting with the number 1. In this case, beginning the sequence with 1 is probably the best starting point. It is a tradition that the best runner in the elite group wears number 1. For other situations, such as department IDs and employee IDs, the starting number may be assigned differently.

Because there will be at least 30,000 runners, the sequence's maximum value was set well above the expected number of runners.

```
CREATE SEQUENCE runner_id_seq  
  INCREMENT BY 1  
  START WITH 1  
  MAXVALUE 50000  
  NOCACHE  
  NOCYCLE;
```



Tell Me / Show Me

CREATING A SEQUENCE (continued)

The NOCACHE option prevents values in the SEQUENCE from being cached in memory, which in the event of system failure prevents numbers preallocated and held in memory from being lost.

The NOCYCLE option prevents the numbering from starting over at 1 if the value 50,000 is exceeded. Don't use the CYCLE option if the sequence is used to generate primary-key values unless there is a reliable mechanism that deletes old rows faster than new ones are added.

```
CREATE SEQUENCE runner_id_seq  
  INCREMENT BY 1  
  START WITH 1  
  MAXVALUE 50000  
  NOCACHE  
  NOCYCLE;
```




Tell Me / Show Me

CONFIRMING SEQUENCES

To verify that a sequence was created, query the USER_OBJECTS data dictionary. To see all of the SEQUENCE settings, query the USER_SEQUENCES data dictionary as shown below. List the value names in the SELECT statement as shown below.

```
SELECT sequence_name, min_value, max_value, increment_by, last_number  
FROM user_sequences;
```

If NOCACHE is specified, the last_number column in the above query displays the next available sequence number.

If CACHE is specified, the last_number column displays the next available number in the sequence which has not been cached into memory.

Tell Me / Show Me

NEXTVAL AND CURRVAL PSEUDOCOLUMNS

The NEXTVAL pseudocolumn is used to extract successive sequence numbers from a specified sequence. You must qualify NEXTVAL with the sequence name.

When you reference sequence.NEXTVAL, a new sequence number is generated and the current sequence number is placed in CURRVAL.



Tell Me / Show Me

NEXTVAL AND CURRVAL PSEUDOCOLUMNS (continued)

The example below inserts a new department in the DEPARTMENTS table. It uses the DEPARTMENTS_SEQ sequence for generating a new department number as follows:

```
INSERT INTO departments  
      (department_id, department_name, location_id)  
VALUES (departments_seq.NEXTVAL, 'Support', 2500);
```



Tell Me / Show Me

NEXTVAL AND CURRVAL PSEUDOCOLUMNS (continued)

Suppose now you want to hire employees to staff the new department. The INSERT statement to be executed for all new employees can include the following code:

```
INSERT INTO employees (employee_id,  
                        department_id, ...)  
VALUES (employees_seq.NEXTVAL,  
        dept_deptid_seq .CURRVAL, ...);
```

Note: The preceding example assumes that a sequence called EMPLOYEE_SEQ has already been created for generating new employee numbers.



Tell Me / Show Me

NEXTVAL AND CURRVAL PSEUDOCOLUMNS (continued)

The CURRVAL pseudocolumn in the example below is used to refer to a sequence number that the current user has just generated. NEXTVAL must be used to generate a sequence number in the current user's session before CURRVAL can be referenced. You must qualify CURRVAL with the sequence name. When sequence.CURRVAL is referenced, the last value generated by that user's process is returned.

```
INSERT INTO employees (employee_id,  
                        department_id, ...)  
VALUES (employees_seq.NEXTVAL,  
        dept_deptid_seq .CURRVAL, ...);
```



Tell Me / Show Me

USING A SEQUENCE

After you create a sequence, it generates sequential numbers for use in your tables. Reference the sequence values by using the NEXTVAL and CURRVAL pseudocolumns.



You can use NEXTVAL and CURRVAL in the following contexts:

- The SELECT list of a SELECT statement that is not part of a subquery
- The SELECT list of a subquery in an INSERT statement
- The VALUES clause of an INSERT statement
- The SET clause of an UPDATE statement

Tell Me / Show Me

USING A SEQUENCE (continued)

You cannot use NEXTVAL and CURRVAL in the following contexts:

- The SELECT list of a view
- A SELECT statement with the DISTINCT keyword
- A SELECT statement with GROUP BY, HAVING, or ORDER BY clauses
- A subquery in a SELECT, DELETE, or UPDATE statement
- The DEFAULT expression in a CREATE TABLE or ALTER TABLE statement

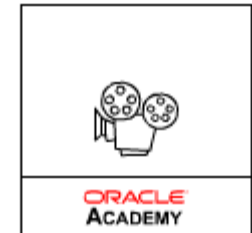


Tell Me / Show Me

USING A SEQUENCE (continued)

To continue our London Marathon example, the following syntax would be used to insert a new participant's information into the runners' table. The runner's identification number would be generated by retrieving the NEXTVAL from the sequence.

```
INSERT INTO runners(runner_id, first_name, last_name,  
                    address, city, state/province, country)  
VALUES  (runner_id_seq.NEXTVAL, 'Joanne', 'Everely',  
        '1945 Brookside Landing', 'New  
York','NY','USA');
```



Tell Me / Show Me

USING A SEQUENCE (continued)

To view the current value for the `runners_id_seq`, `CURRVAL` is used. Note the use of the `DUAL` table in this example. Oracle Application Developer will not execute this query, but you should understand how this works.

```
SELECT runner_id_seq.CURRVAL  
FROM dual;
```

Cache sequences in memory provide faster access to sequence values. The cache is populated the first time you refer to the sequence. Each request for the next sequence value is retrieved from the cached sequence. After the last sequence value is used, the next request for the sequence pulls another cache of sequences into memory. 20 is the default number of sequence numbers cached.



Tell Me / Show Me

Although sequence generators issue sequential numbers without gaps, this action occurs independent of a database commit or rollback. Gaps (nonsequential numbers) can be generated by:



- rolling back a statement containing a sequence, the number is lost.
- a system crash. If the sequence caches values into the memory and the system crashes, those values are lost.
- the same sequence being used for multiple tables. If you do so, each table can contain gaps in the sequential numbers.

Tell Me / Show Me

Viewing the Next Value

If the sequence was created with NOCACHE, it is possible to view the next available sequence value without incrementing it by querying the `USER_SEQUENCES` table.



Tell Me / Show Me

MODIFYING A SEQUENCE

As with the other database objects you've created, a SEQUENCE can also be changed using the ALTER SEQUENCE statement. What if the London Marathon exceeded the 50,000 runner registrations and you needed to add more numbers? The sequence could be changed to increase the MAXVALUE without changing the existing number order.



```
ALTER SEQUENCE runner_id_seq  
    INCREMENT BY 1  
    MAXVALUE 999999  
    NOCACHE  
    NOCYCLE;
```

Tell Me / Show Me

MODIFYING A SEQUENCE (continued)

Some validation is performed when you alter a sequence. For example, a new MAXVALUE that is less than the current sequence number cannot be executed.



```
ALTER SEQUENCE runner_id_seq  
    INCREMENT BY 1  
    MAXVALUE 90  
    NOCACHE  
    NOCYCLE;
```

ERROR at line 1:

ORA-04009: MAXVALUE cannot be made to
be less than the current value

Tell Me / Show Me

ALTER SEQUENCE GUIDELINES

A few guidelines apply when executing an ALTER SEQUENCE statement. They are:

- You must be the owner or have the ALTER privilege for the sequence in order to modify it.
- Only future sequence numbers are affected by the ALTER SEQUENCE statement.
- The START WITH option cannot be changed using ALTER SEQUENCE. The sequence must be dropped and re-created in order to restart the sequence at a different number.



Tell Me / Show Me

REMOVING A SEQUENCE

To remove a sequence from the data dictionary, use the **DROP SEQUENCE** statement. You must be the owner of the sequence or have **DROP ANY SEQUENCE** privileges to remove it. Once removed, the sequence can no longer be referenced.

```
DROP SEQUENCE dept_deptid_seq;
```



ORACLE
ACADEMY



Tell Me / Show Me

Terminology

Key terms used in this lesson include:

NEXTVAL

CURRVAL

INCREMENT BY

Sequences

CACHE/ NOCACHE

MAXVALUE

NOMAXVALUE

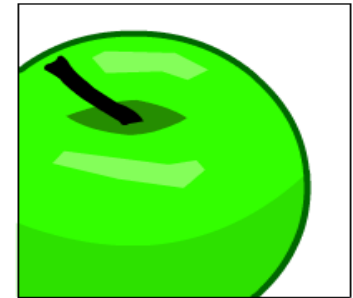
CREATE SEQUENCE

STARTS WITH

CYCLE/ NOCYCLE

MINVALUE

NO MINVALUE

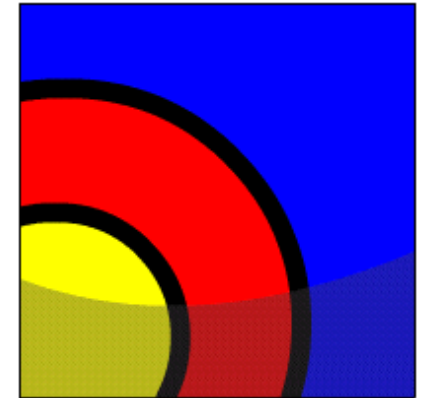




Summary

In this lesson you have learned to:

- List at least three useful characteristics of a sequence
- Write and execute a SQL statement that creates a sequence
- Query the data dictionary using `USER_SEQUENCES` to confirm a sequence definition
- Apply the rules for using `NEXTVAL` to generate sequential unique numbers in a table
- List the advantages and disadvantages of caching sequence values
- Name three reasons why gaps can occur in a sequence



Summary

Practice Guide

The link for the lesson practice guide can be found in the course resources in Section 0.

