

## OTHER DATABASE OBJECTS

### -- CHAPTER 13 --

- What is a Sequence?
  - Automatically generates unique numbers
  - Is a sharable object
  - Is typically used to create a primary key value
  - Replaces application code
  - Speeds up the efficiency of accessing sequence values when cached in memory
- The CREATE SEQUENCE statement: Define a sequence to generate sequential numbers automatically.

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

- INCREMENT BY n
  - Is the name of the sequence generator
- START WITH n
  - Specifies the interval between sequence numbers where n is an integer (If this clause is omitted, the sequence will increment by 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 (This is the default option).
- 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 (This is the default option).
- CYCLE | NOCYCLE
  - Specifies that the sequence continues to generate values after reaching either its maximum or minimum value or does not generate additional values (NOCYCLE is the default option).
- CACHE n | NOCACHE
  - Specifies how many values the Oracle Server will pre-allocate and keep in memory (By default, the Oracle Server will cache 20 values).
- The following is an example of the creation of a sequence named DEPT\_DEPTNO to be used for the primary key of the dept table. Do not use the CYCLE option.

```
SQL> CREATE SEQUENCE dept_deptno
2 INCREMENT BY 1
3 START WITH 91
```

- 4 MAXVALUE 100
- 5 NOCACHE
- 6 NOCYCLE;

Sequence created.

- Confirming your sequence values in the USER\_SEQUENCES data dictionary table.

```
SQL> SELECT sequence_name, min_value, max_value, increment_by, last_number
       2  from user_sequences;
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPARTMENT_ID_SEQ	1	1.000E+27	1	21
<b>DEPT_DEPTNO</b>	<b>1</b>	<b>100</b>	<b>1</b>	<b>91</b>
DEPT_ID_SEQ	1	80	1	78
MEMBER_ID	1	9999999	1	108
S_CUSTOMER_ID	1	9999999	1	216
S_DEPT_ID	1	9999999	1	51
S_EMP_ID	1	9999999	1	26
S_IMAGE_ID	1	9999999	1	1981
S_LONGTEXT_ID	1	9999999	1	1369
S_ORD_ID	1	9999999	1	115
S_PRODUCT_ID	1	9999999	1	50537
S_REGION_ID	1	9999999	1	6
S_WAREHOUSE_ID	1	9999999	1	10502
TEST_SEQ	1	1.000E+27	1	1
TITLE_ID	1	9999999	1	100
WORKER_ID_SEQ	1	9999999	1	204

16 rows selected.

- Note that there are 15 other sequences that were created by some one else in the Scott/Tiger schema...
- Using a sequence: Once you create your sequence, you can use the sequence to generate sequential numbers for use in your tables. Reference the sequence values by using the NEXTVAL and CURRVAL pseudocolumns.
- The NEXTVAL pseudocolumn is used to extract successive sequence numbers from a specified sequence. You must qualify ENXTVAL with the sequence name. When you reference sequence. ENXTVAL, a new sequence number is generated and the current sequence number is placed in CURRVAL.
- The CURRVAL pseudocolumn 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 returned to that user's process is displayed.
- Rules for using NEXTVAL and CURRVAL
  - You can use NEXTVAL and CURRVAL in the following:
    - The SELECT LIST OF A SELECT statement that is not part of the subquery
    - The SELECT list of a subquery in the INSERT statement.
    - The VALUES clause of an INSERT statement
    - The SET clause of an UPDATE statement

- You cannot use NEXTVAL and CURRVAL in the following:
  - A SELECT list of a view
  - A SELECT statement with the DISTINCT keyword
  - A SELECT statement with the GROUP BY, HAVING, or ORDER BY clauses
  - A subquery in a SELECT, DELETE, or UPDATE statement
  - A DEFAULT expression in a CREATE TABLE or ALTER TABLE statement
- Using a sequence: Insert a new department named “MARKETING” in San Diego.

```
SQL> INSERT INTO dept(deptno, dname, loc)
  2  values (dept_deptno.NEXTVAL,
  3          'MARKETING', 'SAN DIEGO');
```

1 row created.

```
SQL> select * from dept
  2  where deptno = 91;
```

DEPTNO	DNAME	LOC
91	MARKETING	SAN DIEGO

- View the current value for the dept\_deptno sequence...

```
SQL> select dept_deptno.CURRVAL
  2  from dual;
```

CURRVAL
91

- Here’s another example...

```
SQL> r
  1  INSERT INTO dept(deptno, dname, loc)
  2* values (dept_deptno.NEXTVAL, 'SCIENCE', 'SANTA ANA')
```

1 row created.

```
SQL> select * from dept
  2  where deptno = dept_deptno.currval;
```

DEPTNO	DNAME	LOC
92	SCIENCE	SANTA ANA

- Note here that we selected the record whose deptno is the same as the current value selected in the sequence dept\_deptno.
- Modifying a sequence: Change the increment value, maximum, minimum, cycle option or cache option...\

```
SQL> ALTER SEQUENCE dept_deptno
  2  INCREMENT BY 5
```

```

3  MAXVALUE 999999
4  MINVALUE 20
5  NOCACHE
6  NOCYCLE;

```

Sequence altered.

```

SQL> SELECT sequence_name, min_value, max_value, increment_by, last_number
2  from user_sequences
3* where sequence_name = 'DEPT_DEPTNO'

```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_DEPTNO	20	999999	5	97

- Guidelines for Modifying a Sequence:
  - You must be the owner of have the ALTER privilege for the sequence.
  - Only future sequence numbers are affected
  - The sequence must be dropped and re-created to restart the sequence at a different number
  - Some validation is performed.
- Removing a Sequence: Remove a sequence from the data dictionary by using the DROP SEQUENCE statement. Once removed, the sequence can no longer be referenced.

```

SQL> DROP SEQUENCE dept_deptno;

```

Sequence dropped.

- What a is an Index?
  - A schema object
  - Is issued y the Oracle Server to speed up the retrieval of rows by using a pointer
  - Can reduce disk I/O by using rapid path access method to locate the data quickly
  - Is independent of the table it indexes
  - Is used and maintained automatically by the Oracle Server.
- How are indexes created?
  - Automatically:
    - A unique index is created when you define a PRIMARY KEY or UNIQUE constraint in a table definition.
  - Manually:
    - Users can create non-unique indexes on columns to speed up access time to the rows.
- Creating an Index: Create an index on one or more columns.

```

SQL> r
1  CREATE INDEX emp_ename_idx
2* ON emp(ename)

```

Index created.

- More Is Not Always Better: More indexes on a table does not mean it will speed up queries. Each DML operation that is committed on a table with indexes means that the indexes must be updated. The

more indexes you have associated with a table, the more effort the Oracle Server must make to update all the indexes after a DML.

- When to Create an Index:
  - The column is used frequently in the WHERE clause or in a join condition
  - The column contains a wide range of values
  - The column contains a large number of null values.
  - Two or more columns are frequently used together in a WHERE clause or join condition.
  - The table is large and most queries are expected to retrieve less than 2-4% of the rows.
- Confirming Indexes: The USER\_INDEXES data dictionary view contains the name of the index and its uniqueness. The USER\_IND\_COLUMNS view contains the index name, the table name, and the column name.

```
SQL> r
 1 SELECT ic.index_name, ic.column_name,
 2 ic.column_position col_pos, ix.uniqueness
 3 FROM user_indexes ix, user_ind_columns ic
 4 WHERE ic.index_name = ix.index_name
 5* AND ic.table_name = 'EMP'
```

INDEX_NAME	COLUMN_NAME	COL_POS	UNIQUENES
SYS_C008971	EMPNO	1	UNIQUE
EMP_ENAME_IDX	ENAME	1	NONUNIQUE

- Function-Based Indexes: A function-based index is an index based on expressions. The index expression is built from table columns, constants, SQL functions, and user-defined functions.
- Removing an Index

```
SQL> DROP INDEX indexName;
```

- You cannot modify indexes. To change an index, you must drop it and then re-create it. Remove an index definition from the data dictionary by issuing the DROP INDEX statement. You must be the owner of the index or have the DROP ANY INDEX privilege.
- Synonyms: Simplify access to objects by creating a synonym. Refer to a table owned by another user. Shorten lengthy object names.

```
CREATE [PUBLIC] SYNONYM synonym
FOR object;
```

- PUBLIC: creates a synonym that is accessible to all users
- Synonym: is the name of the synonym to be created
- Object: Identifies the object for which the synonym is created.
  - The object cannot be contained in a package
  - A private synonym name must be distinct from all other objects owned by the same user.
- Creating and Removing Synonyms: Create a shortened name for the DEPT\_SUM\_VU view.

```
SQL> CREATE SYNONYM d_sum
 2 FOR dept_sum_vu;
```

Synonym created.

- Removing a Synonym: To drop a synonym, use the DROP SYNONYM statement. Only the DBA can drop a public synonym.

```
SQL> DROP SYNONYM d_sum;  
Synonym dropped.
```

\*