12

Other Database Objects

Objectives

After completing this lesson, you should be able to do the following:

- Create, maintain, and use sequences
- Create and maintain indexes
- Create private and public synonyms

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Generates primary key values
Index	Improves the performance of some queries
Synonym	Alternative name for an object

What Is a Sequence?

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 Syntax

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}];
```

Creating a Sequence

- Create a sequence named `DEPT_DEPTID_SEQ` to be used for the primary key of the `DEPARTMENTS` table.
- Do not use the `CYCLE` option.

```
CREATE SEQUENCE dept_deptid_seq
                INCREMENT BY 10
                START WITH 120
                MAXVALUE 9999
                NOCACHE
                NOCYCLE;
Sequence created.
```

Confirming Sequences

- Verify your sequence values in the `USER_SEQUENCES` data dictionary table.

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

- The `LAST_NUMBER` column displays the next available sequence number if `NOCACHE` is specified.

NEXTVAL and CURRVAL Pseudocolumns

- NEXTVAL returns the next available sequence value. It returns a unique value every time it is referenced, even for different users.
- CURRVAL obtains the current sequence value.
- NEXTVAL must be issued for that sequence before CURRVAL contains a value.

Using a Sequence

- Insert a new department named “Support” in location ID 2500.

```
INSERT INTO departments(department_id,
            department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
            'Support', 2500);
1 row created.
```

- View the current value for the DEPT_DEPTID_SEQ sequence.

```
SELECT   dept_deptid_seq.CURRVAL
FROM     dual;
```

Using a Sequence

- Caching sequence values in memory gives faster access to those values.
- Gaps in sequence values can occur when:
  - A rollback occurs
  - The system crashes
  - A sequence is used in another table
- If the sequence was created with `NOCACHE`, view the next available value, by querying the `USER_SEQUENCES` table.

Modifying a Sequence

Change the increment value, maximum value, minimum value, cycle option, or cache option.

```
ALTER SEQUENCE dept_deptid_seq
               INCREMENT BY 20
               MAXVALUE 999999
               NOCACHE
               NOCYCLE;
Sequence altered.
```

Guidelines for Modifying a Sequence

- You must be the owner or 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.

```
DROP SEQUENCE dept_deptid_seq;
Sequence dropped.
```

What is an Index?

An index:

- Is a schema object
- Is used by the Oracle server to speed up the retrieval of rows by using a pointer
- Can reduce disk I/O by using a rapid path access method to locate 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 automatically when you define a `PRIMARY KEY` or `UNIQUE` constraint in a table definition.**
- **Manually: Users can create nonunique indexes on columns to speed up access to the rows.**

Creating an Index

- Create an index on one or more columns.

```
CREATE INDEX index
ON table (column[, column]...);
```

- Improve the speed of query access to the LAST_NAME column in the EMPLOYEES table.

```
CREATE INDEX  emp_last_name_idx
ON            employees(last_name);
Index created.
```

When to Create an Index

You should create an index if:

- A column contains a wide range of values
- A column contains a large number of null values
- One or more columns are frequently used together in a `WHERE` clause or a join condition
- The table is large and most queries are expected to retrieve less than 2 to 4 percent of the rows

When Not to Create an Index

It is usually not worth creating an index if:

- The table is small
- The columns are not often used as a condition in the query
- Most queries are expected to retrieve more than 2 to 4 percent of the rows in the table
- The table is updated frequently
- The indexed columns are referenced as part of an expression

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.

```
SELECT   ic.index_name, ic.column_name,
         ic.column_position col_pos,ix.uniqueness
FROM     user_indexes ix, user_ind_columns ic
WHERE    ic.index_name = ix.index_name
AND      ic.table_name = 'EMPLOYEES';
```

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.

```
CREATE INDEX upper_dept_name_idx
ON departments(UPPER(department_name));

Index created.

SELECT *
FROM   departments
WHERE  UPPER(department_name) = 'SALES';
```

Removing an Index

- Remove an index from the data dictionary by using the `DROP INDEX` command.

```
DROP INDEX index;
```

- Remove the `UPPER_LAST_NAME_IDX` index from the data dictionary.

```
DROP INDEX upper_last_name_idx;
Index dropped.
```

- To drop an index, you must be the owner of the index or have the `DROP ANY INDEX` privilege.

Synonyms

Simplify access to objects by creating a synonym (another name for an object). With synonyms, you can:

- Ease referring to a table owned by another user
- Shorten lengthy object names

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

Creating and Removing Synonyms

- Create a shortened name for the `DEPT_SUM_VU` view.

```
CREATE SYNONYM  d_sum
FOR  dept_sum_vu;
Synonym Created.
```

- Drop a synonym.

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

Summary

In this lesson, you should have learned how to:

- Automatically generate sequence numbers by using a sequence generator
- View sequence information in the `USER_SEQUENCES` data dictionary table
- Create indexes to improve query retrieval speed
- View index information in the `USER_INDEXES` dictionary table
- Use synonyms to provide alternative names for objects

Practice 12 Overview

This practice covers the following topics:

- Creating sequences
- Using sequences
- Creating nonunique indexes
- Displaying data dictionary information about sequences and indexes
- Dropping indexes