Database Systems

Subject Teacher: Zartasha Baloch

Creating and managing tables

Lecture # 12

Disclaimer: The material used in this presentation to deliver the lecture i.e., definitions/text and pictures/graphs etc. does not solely belong to the author/presenter. The presenter has gathered this lecture material from various sources on web/textbooks. Following sources are especially acknowledged:

1. Connolly, Thomas M., and Carolyn E. Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.
2. Gorman, Tim, Inger Jorgensen, Melanie Caffrey, and Lex deHaan. Beginning Oracle SQL: For Oracle Database 12c. Apress, 2014.
3. Greenberg, Nancy, and Instructor Guide PriyaNathan. "Introduction to Oracle9i: SQL." ORACLE, USA (2001).

Objectives

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

- Describe the main database objects
- Create tables
- Describe the data types that can be used when specifying column definition
- Alter table definitions
- Drop, rename, and truncate tables

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	Numeric value generator
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Naming Rules

Table names and column names:

- Must begin with a letter
- Must be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle server reserved word

The CREATE TABLE Statement

- You must have:
  - CREATE TABLE privilege
  - A storage area

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr][, ...]);
```

- You specify:
  - Table name
  - Column name, column data type, and column size

Referencing Another User's Tables

- Tables belonging to other users are not in the user's schema.
- You should use the owner's name as a prefix to those tables.

The DEFAULT Option

- Specify a default value for a column during an insert.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Literal values, expressions, or SQL functions are legal values.
- Another column's name or a pseudocolumn are illegal values.
- The default data type must match the column data type.

Creating Tables

- Create the table.

```
CREATE TABLE dept
            (deptno  NUMBER(2),
             dname   VARCHAR2(14),
             loc VARCHAR2(13));
Table created.
```

- Confirm table creation.

```
DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

Tables in the Oracle Database

- User Tables:
  - Are a collection of tables created and maintained by the user
  - Contain user information
- Data Dictionary:
  - Is a collection of tables created and maintained by the Oracle Server
  - Contain database information

Querying the Data Dictionary

- **See the names of tables owned by the user.**

```
SELECT table_name
FROM   user_tables ;
```

- **View distinct object types owned by the user.**

```
SELECT DISTINCT object_type
FROM   user_objects ;
```

- **View tables, views, synonyms, and sequences owned by the user.**

```
SELECT *
FROM   user_catalog ;
```

Creating a Table by Using a Subquery Syntax

- Create a table and insert rows by combining the `CREATE TABLE` statement and the `AS` *`subquery`* option.

```
CREATE TABLE table
        [(column, column...)]
AS subquery;
```

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

Creating a Table by Using a Subquery

```
CREATE TABLE  dept80
  AS
    SELECT   employee_id, last_name,
             salary*12 ANNSAL,
             hire_date
    FROM     employees
    WHERE    department_id = 80;
Table created.
```

```
 DESCRIBE dept80
```

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

The ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column
- Define a default value for the new column
- Drop a column

The ALTER TABLE Statement

Use the ALTER TABLE statement to add, modify, or drop columns.

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
DROP       (column);
```

Adding a Column

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
149	Zlotkey	126000	29-JAN-00
174	Abel	132000	11-MAY-96
176	Taylor	103200	24-MAR-98

New column

JOB_ID

"Add a new column to the DEPT80 table."

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

Adding a Column

- You use the `ADD` clause to add columns.

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));
Table altered.
```

- The new column becomes the last column.

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

Modifying a Column

- You can change a column's data type, size, and default value.

```
ALTER TABLE   dept80
MODIFY     (last_name VARCHAR2(30));
Table altered.
```

- A change to the default value affects only subsequent insertions to the table.

Dropping a Column

Use the `DROP COLUMN` clause to drop columns you no longer need from the table.

```
ALTER TABLE  dept80
DROP COLUMN  job_id;
Table altered.
```

The SET UNUSED Option

- You use the SET UNUSED option to mark one or more columns as unused.
- You use the DROP UNUSED COLUMNS option to remove the columns that are marked as unused.

```
ALTER TABLE   table
SET   UNUSED (column);
OR
ALTER TABLE  table
SET   UNUSED COLUMN column;
```

```
ALTER TABLE table
DROP  UNUSED COLUMNS;
```

Dropping a Table

- All data and structure in the table is deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- You *cannot* roll back the `DROP TABLE` statement.

```
DROP TABLE dept80;
Table dropped.
```

Changing the Name of an Object

- To change the name of a table, view, sequence, or synonym, you execute the `RENAME` statement.

```
RENAME dept TO detail_dept;
Table renamed.
```

- You must be the owner of the object.

Truncating a Table

- The `TRUNCATE TABLE` statement:
  - Removes all rows from a table
  - Releases the storage space used by that table

```
TRUNCATE TABLE detail_dept;
Table truncated.
```

- You cannot roll back row removal when using `TRUNCATE`.
- Alternatively, you can remove rows by using the `DELETE` statement.

Adding Comments to a Table

- You can add comments to a table or column by using the COMMENT statement.

```
COMMENT ON TABLE employees
IS 'Employee Information';
Comment created.
```

- Comments can be viewed through the data dictionary views:
  - ALL_COL_COMMENTS
  - USER_COL_COMMENTS
  - ALL_TAB_COMMENTS
  - USER_TAB_COMMENTS

Summary

In this lesson, you should have learned how to use DDL statements to create, alter, drop, and rename tables.

Statement	Description
`CREATE TABLE`	Creates a table
`ALTER TABLE`	Modifies table structures
`DROP TABLE`	Removes the rows and table structure
`RENAME`	Changes the name of a table, view, sequence, or synonym
`TRUNCATE`	Removes all rows from a table and releases the storage space
`COMMENT`	Adds comments to a table or view