# USING DDL STATEMENTS

## Using DDL Statements to Create and Manage Tables

A schema is the collection of multiple database objects,which are known as schema objects.These objects have direct access by their owner schema.Below table lists the schema objects.

- Table - to store data
- View - to project data in a desired format from one or more tables
- Sequence - to generate numeric values
- Index - to improve performance of queries on the tables
- Synonym - alternative name of an object

One of the first steps in creating a database is to create the tables that will store an organization's data.Database design involves identifying system user requirements for various organizational systems such as order entry, inventory management, and accounts receivable. Regardless of database size and complexity, each database is comprised of tables.

## Creating the table

To create a table in the database,a DBA must have certain information in hand - the table name, column name, column data types, and column sizes. All this information can be modified later using DDL commands.

## Table Naming Conventions -

- The name you choose for a table must follow these standard rules:
- The name must begin with a letter A-Z or a-z
- Can contain numbers and underscores
- Can be in UPPER of lower case
- Can be up to 30 characters in length
- Cannot use the same name of another existing object in your schema
- Must not be a SQL reserved word

Following the above guidelines, 'EMP85' can be a valid table name.But 85EMP is not.Similarly, UPDATE cannot be a chosen as a table name since it a SQL reserved keyword.

## CREATE TABLE statement

The CREATE TABLE is a DDL statement which is used to create tables in the database.The table gets created as soon as the CREATE TABLE script is executed and is ready to hold the data onwards.The user must have the CREATE TABLE system privilege to create the table in its own schema.But to create a table in any user's schema, user must have CREATE ANY TABLE schema.

Here is the syntax of a basic CREATE TABLE statement.There may be many additional clauses to explicitly provide the storage specifications or segment values.

```
CREATE TABLE [schema.]table
           ( { column datatype [DEFAULT expr] [column_constraint] ...
             | table_constraint}
          [, { column datatype [DEFAULT expr] [column_constraint] ...
             | table_constraint} ]...)
          [AS subquery]
```

In the above syntax, DEFAULT specifies default value which can be used during INSERT statement if the column is ignored. It cannot contain references to other table columns or pseudo columns *CURRVAL*, *NEXTVAL*, *LEVEL*, *andROWNUM* except SYSDATE and USER, or date constants that are not fully specified.

Constraints are the rules defined optionally at the column level or table level *coveredlaterinthischapter* .These rules are checked during any data action *Insert*, *update* on the table and raise error to abort the action upon its violation.

For example, the CREATE TABLE statement below creates a table EMP_TEST. Note the column specifications, data type and precision.

```
CREATE TABLE SCOTT.EMP_TEST
(EMPID NUMBER,
ENAME VARCHAR2(100),
DEPARTMENT_ID NUMBER,
SALARY NUMBER,
JOB_ID VARCHAR2(3),
HIREDATE DATE,
COMM NUMBER);
```

A user can refer the tables from other user's schema by prefixing the username or schema with the table name.For example, a user GUEST wishes to query the employee name and salary from the EMP_TEST table which is owned by SCOTT. He can issue the below query -

```
SELECT  ENAME, SALARY,
FROM  GUEST.EMP_TEST;
```

A column can hold a default value during the time of table creation.It helps to restrict the NULL values getting into the column. Default value can be deduced from either a literal, expression or SQL function which must return a compatible data type to the column. In the below CREATE TABLE statement, note that the LOCATION_ID column has default value 100.

```
CREATE TABLE SCOTT.DEPARTMENT
(DEPARTMENT_ID NUMBER,
   DNAME VARCHAR2 (100),
   LOCATION_ID NUMBER DEFAULT 100);
```

## CTAS - Create table using subquery

A table can be created from an existing table in the database using a subquery option.It copies the table structure as well as the data from the table. Data can also be copied based on conditions.The column data type definitions including the explicitly imposed NOT NULL constraints are copied into the new table.

The below CTAS script creates a new table EMP_BACKUP. Employee data of department 20 gets copied into the new table.

```
CREATE TABLE EMP_BACKUP
AS
SELECT * FROM EMP_TEST
WHERE department_id=20;
```

## Data types

Data types are used to specify the basic behavior of a column in the table.On a broader basis,column behavior can either belong to number,character or a date family.There are multiple other subtypes which belong to these families.

## Number data type

The NUMBER datatype encompasses both integer,fixed-point,and floating-point numeric values.Early versions of Oracle defined different datatypes for each of these different types of numbers,but now the NUMBER datatype serves all of these purposes.Choose the NUMBER datatype when a column must store numerical data that can be used in mathematical calculations.Occasionally,the NUMBER datatype is used to store identification numbers where those numbers are generated by the DBMS as sequential numbers.

NUMBER *p, s*, where p is the precision up to 38 digits and s is the scale *numberofdigitstotherightofthedecimalpoint*.The scale can range between -84 to 127.

NUMBER *p*,is a fixed-point number with a scale of zero and a precision of p.

FLOAT [*p*],where p is the binary precision that can range from 1 to 126. If p is not specified the default value is binary 126.

## Date data type

For each DATE data type, Century, Year, Month, Day, Hour, Minute, Second are stored in database. Every database system has a default date format that is defined by the initialization parameter NLS_DATE_FORMAT. This parameter is usually set to DD-MON-YY.If you do not specify a time, the default time is 12:00:00 a.m.

## Character data type

Oracle supports three predefined character datatypes including CHAR, VARCHAR, VARCHAR2, and LONG.VARCHAR and VARCHAR2 are actually synonymous, and Oracle recommends using VARCHAR2 instead of VARCHAR.Use the CHAR datatype when the column will store character values that are fixed-length.For example, a Social Security number *SSN* in the United States is assigned to every citizen and is always 9 characters in size *eventhoughanSSNisstrictlycomposedofdigits, thedigitsaretreatedascharacters*, and would be specified as CHAR9. Use the VARCHAR2 datatype to store alphanumeric data that is variable-length.For example, a customer name or address will vary considerably in terms of the number of characters to be stored.The maximum size of a VARCHAR2 column is 4,000 characters.

## LOB data type

Oracle provides several different LOB datatypes, including CLOB *characterlargeobject* and BLOB *binarylargeobject*.Columns of these datatypes can store unstructured data including text, image, video, and spatial data.The CLOB datatype can store up to eight terabytes of character data using the CHAR database character set.The BLOB datatype is used to store unstructured binary large objects such as those associated with image and video data where the data is simply a stream of "bit" values.A BLOB datatype can store up to eight terabytes of binary data.The NCLOB data type can store character large objects in multibyte national character set up to 8TB to 128TB.The BFILE data type value works as a file locator or pointer to file on the server's file system. The maximum file size supported is 8TB to 128TB.

## Constraints

Constraints are the set of rules defined in Oracle tables to ensure data integrity.These rules are enforced placed for each column or set of columns.Whenever the table participates in data action, these rules are validated and raise exception upon violation. The available constraint types are NOT NULL, Primary Key, Unique, Check, and Foreign Key.

The below syntax can be used to impose constraint at the column level.

## Syntax:

```
column [data type] [CONSTRAINT constraint_name] constraint_type
```

All constraints except NOT NULL, can also be defined at the table level. Composite constraints can only be specified at the table level.

## NOT NULL Constraint

A NOT NULL constraint means that a data row must have a value for the column specified as NOT NULL.If a column is specified as NOT NULL,the Oracle RDBMS will not allow rows to be stored to the employee table that violate this constraint.It can only be defined at column level, and not at the table level.

## Syntax:

```
COLUMN [data type] [NOT NULL]
```

## UNIQUE constraint

Sometimes it is necessary to enforce uniqueness for a column value that is not a primary key column.The UNIQUE constraint can be used to enforce this rule and Oracle will reject any rows that violate the unique constraint.Unique constraint ensures that the column values are distinct, without any duplicates.

## Syntax:

### Column Level:

```
COLUMN [data type] [CONSTRAINT <name>] [UNIQUE]
```

**Table Level:** CONSTRAINT [constraint name] UNIQUE *columnname*

Note: Oracle internally creates unique index to prevent duplication in the column values.Indexes would be discussed later in PL/SQL.

```
CREATE TABLE TEST
( ... ,
  NAME VARCHAR2(20)
          CONSTRAINT TEST_NAME_UK UNIQUE,
  ... );
```

In case of composite unique key,it must be defined at table level as below.

```
CREATE TABLE TEST
( ... ,
  NAME VARCHAR2(20),
  STD VARCHAR2(20) ,
      CONSTRAINT TEST_NAME_UK UNIQUE (NAME, STD)
 );
```

## Primary Key

Each table must normally contain a column or set of columns that uniquely identifies rows of data that are stored in the table.This column or set of columns is referred to as the primary key.Most tables have a single column as the primary key.Primary key columns are restricted against NULLs and duplicate values.

## Points to be noted -

- A table can have only one primary key.
- Multiple columns can be clubbed under a composite primary key.
- Oracle internally creates unique index to prevent duplication in the column values.Indexes would be discussed later in PL/SQL.

## Syntax:

### Column level:

```
COLUMN [data type] [CONSTRAINT <constraint name> PRIMARY KEY]
```

### Table level:

```
CONSTRAINT [constraint name] PRIMARY KEY [column (s)]
```

The following example shows how to use PRIMARY KEY constraint at column level.

```
CREATE TABLE TEST
( ID  NUMBER CONSTRAINT TEST_PK PRIMARY KEY,
  ...  );
```

The following example shows how to define composite primary key using PRIMARY KEY constraint at the table level.

```
CREATE TABLE TEST
 ( ...,
   CONSTRAINT TEST_PK PRIMARY KEY (ID)
 );
```

## Foreign Key

When two tables share the parent child relationship based on specific column, the joining column in the child table is known as Foreign Key.This property of corresponding column in the parent table is known as Referential integrity.Foreign Key column values in the child table can either be null or must be the existing values of the parent table.Please note that only primary key columns of the referenced table are eligible to enforce referential integrity.

If a foreign key is defined on the column in child table then Oracle does not allow the parent row to be deleted,if it contains any child rows.However,if ON DELETE CASCADE option is given at the time of defining foreign key,Oracle deletes all child rows while parent row is being deleted.Similarly,ON DELETE SET NULL indicates that when a row in the parent table is deleted, the foreign key values are set to null.

## Syntax:

### Column Level:

```
COLUMN [data type] [CONSTRAINT] [constraint name] [REFERENCES] [table name (column
name)]
```

### Table level:

```
CONSTRAINT [constraint name] [FOREIGN KEY (foreign key column name) REFERENCES]
[referenced table name (referenced column name)]
```

The following example shows how to use FOREIGN KEY constraint at column level.

```
CREATE TABLE TEST
(ccode varchar2(5)
     CONSTRAINT TEST_FK REFERENCES PARENT_TEST(ccode),
    ...
);
```

### Usage of ON DELETE CASCADE clause

```
CREATE TABLE TEST
(ccode varchar2(5)
   CONSTRAINT TEST_FK REFERENCES PARENT_TEST (ccode)
   ON DELETE CASCADE,
   ...
);
```

## Check constraint

Sometimes the data values stored in a specific column must fall within some acceptable range of values.A CHECK constraint requires that the specified check condition is either true or unknown for each row stored in the table.Check constraint allows to impose a conditional rule on a column, which must be validated before data is inserted into the column. The condition must not contain a sub query or pseudo column CURRVAL NEXTVAL, LEVEL, ROWNUM, or SYSDATE.

Oracle allows a single column to have more than one CHECK constraint. In fact, there is no practical limit to the number of CHECK constraints that can be defined for a column.

## Syntax:

### Column level:

```
COLUMN [data type] CONSTRAINT [name] [CHECK (condition)]
```

### Table level:

```
CONSTRAINT [name] CHECK (condition)
```

The following example shows how to use CHECK constraint at column level.

```
CREATE TABLE TEST
( ...,
   GRADE char (1) CONSTRAINT TEST_CHK
   CHECK (upper (GRADE) in ('A','B','C')),
   ...
);
```

The following example shows how to use CHECK constraint at table level.

```
CREATE TABLE TEST
( ...,
   CONSTRAINT TEST_CHK
   CHECK (stdate < = enddate),
);
```

## ALTER TABLE statement

A DBA can make changes to the table structure or column definitions after the table has been created in the database.The DDL command ALTER TABLE is used to perform such actions.Alter command provides multiple utilities exclusive for schema objects.The ALTER TABLE statement is used to add, drop, rename, and modify a column in a table.

The below ALTER TABLE statement renames the table EMP to EMP_NEW.

```
ALTER TABLE EMP RENAME TO EMP_NEW;
```

The below ALTER TABLE statement adds a new column TESTCOL to the EMP_NEW table

```
ALTER TABLE EMP_NEW ADD (TESTCOL VARCHAR2 (100))
```

The below ALTER TABLE statement renames the column TESTCOL to TESTNEW.

```
ALTER TABLE EMP_NEW RENAME COLUMN TESTCOL TO TESTNEW
```

The below ALTER TABLE statement drop the column TESTNEW from EMP_NEW table

```
ALTER TABLE EMP_NEW DROP COLUMN TESTNEW;
```

The below ALTER TABLE statement adds primary key on the EMPLOYEE_ID column.

```
ALTER TABLE EMP_NEW ADD PRIMARY KEY (EMPLOYEE_ID)
```

The below ALTER TABLE statement drop the primary key.

```
ALTER TABLE EMP_NEW DROP PRIMARY KEY;
```

The below ALTER TABLE statement switches the table mode to read only.

```
ALTER TABLE EMP_NEW READ ONLY;
```

## Read Only Tables

Read only tables came as an enhancement in Oracle 11g.It allows the tables to be used for read only purpose. In earlier oracle versions, tables were made read only by granting SELECT privilege to the other users, but owner still had the read write privilege.But now,if a table is set as Read only,even owner doesn't have access on data manipulation.

## Syntax:

```
ALTER TALE [TABLE NAME] READ ONLY
```

```
ALTER TALE [TABLE NAME] READ WRITE
```

## Illustration

```
SQL>CREATE TABLE ORATEST (id NUMBER)

SQL>INSERT INTO ORATEST VALUES (1);

SQL>ALTER TABLE ORATEST READ ONLY;

SQL> INSERT INTO ORATEST VALUES (2);
INSERT INTO ORATEST VALUES (2)
            *
ERROR at line 1:
ORA-12081: update operation not allowed on table "TEST"."ORATEST"

SQL> UPDATE ORATEST SET id = 2;
UPDATE ORATEST SET id = 2
       *
ERROR at line 1:
ORA-12081: update operation not allowed on table "TEST"."ORATEST"

SQL> DELETE FROM ORATEST;
DELETE FROM ORATEST
            *
ERROR at line 1:
ORA-12081: update operation not allowed on table "TEST"."ORATEST"

SQL> TRUNCATE TABLE ORATEST;
TRUNCATE TABLE ORATEST
               *
ERROR at line 1:
ORA-12081: update operation not allowed on table "TEST"."ORATEST"

SQL> ALTER TABLE ORATEST ADD (description VARCHAR2 (50));
ALTER TABLE ORATEST ADD (description VARCHAR2 (50))
*
ERROR at line 1:
ORA-12081: update operation not allowed on table "TEST"."ORATEST"

SQL> ALTER TABLE ORATEST READ WRITE;

Table altered.

SQL> DELETE FROM ORATEST;
```

```
1 row deleted.
```

## DROP TABLE statement

The DROP TABLE statement is used to remove a table from the database. The dropped table and its data remain no longer available for selection.Dropped table can be recovered using FLASHBACK utility,if available in recyclebin.Dropping a table drops the index and triggers associated with it.

### Syntax:

```
DROP TABLE [TABLE NAME] [PURGE]
```

The below statement will drop the table and place it into the recyclebin.

```
DROP TABLE emp_new;
```

The below statement will drop the table and flush it out from the recyclebin also.

```
DROP TABLE emp_new PURGE;
```

Processing math: 100%