**DML Statements:**

## INSERT statement

The INSERT command is used to store data in tables.

### **Syntax:**

INSERT INTO table (column1 name, column2 name,)

VALUES (column1 value, column2 value,);

The INSERT statement below creates a new employee record in the EMPLOYEES table. Note that it inserts the values for the primary columns EMPLOYEE\_ID, FIRST\_NAME, SALARY and DEPARTMENT\_ID.

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, SALARY, DEPARTMENT\_ID)

VALUES (130, 'KEMP', 3800, 10);

## UPDATE statement

The UPDATE command modifies the data stored in a column. It can update single or multiple rows at a time depending on the result set filtered by conditions specified in WHERE clause.

The UPDATE command changes data in the table, not the table structure.

### **Syntax:**

UPDATE table

SET column = value [, column = value ...]

[WHERE condition]

The UPDATE statement below updates the salary of employee JOHN to 5000.

UPDATE employees

SET salary = 5000

WHERE UPPER (first\_name) = 'JOHN';

## DELETE statement

The DELETE command is one of the simplest of the SQL statements. It removes one or more rows from a table. Multiple table delete operations are not allowed in SQL.The syntax of the DELETE command is as below.

DELETE FROM table\_name

[WHERE condition];

The DELETE command deletes all rows in the table that satisfy the condition in the optional WHERE clause. Since the WHERE clause is optional, one can easily delete all rows from a table by omitting a WHERE clause since the WHERE clause limits the scope of the DELETE operation.

The below DELETE statement would remove EDWIN's details from EMP table.

DELETE employees

WHERE UPPER (ENAME) = 'EDWIN'

**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.

Syntax:

CREATE TABLE [schema.]table

(({ column datatype [DEFAULT expr] [column\_constraint] ...

| table\_constraint}

[, { column datatype [DEFAULT expr] [column\_constraint] ...

| table\_constraint} ]...)

**Constraints:**

Constraints are the rules defined optionally at the column level or table level.

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.

These rules are checked during any data action (Insert, update) on the table and raise error to abort the action upon its violation.

CREATE TABLE SCOTT.EMP\_TEST

(EMPID NUMBER,

ENAME VARCHAR2 (100),

DEPARTMENT\_ID NUMBER,

SALARY NUMBER,

JOB\_ID VARCHAR2 (3),

HIREDATE DATE,

COMM NUMBER);

## 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)

);

## 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 (column name)

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)

);

## 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.

### **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),

...

);

## 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"

## ALTER TABLE statement

.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;

## 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;