ALTERING TABLES

After you have created the tables, there may be time when you want to change the structure of the table. Either ou want to add a column or change the definition of the existing columns. We can do this with the help of ALTER TABLE statement. To alter a table, the table must be contained in your schema, or you must have either the ALTER object privilege for the table or the ALTER ANY TABLE system privilege. A table in an Oracle database can be altered for the following reasons:

- To add or drop one or more columns to or from the table.
- To add or modify an integrity constraint on a table.
- To modify an existing column's definition (datatype, length, default value, and NOT NULL integrity constraint).
- To enable or disable integrity constraints or triggers associated with the table.
- To drop integrity constraints associated with the table.

You can increase the length of an existing column. However, you cannot decrease it unless there are no rows in the table. Furthermore, if you are modifying a table to increase the length of a column of datatype CHAR, realize that this may be a time consuming operation and may require substantial additional storage, especially if the table contains many rows. This is because the CHAR value in each row must be blank-padded to satisfy the new column length.

Before altering a table, familiarize yourself with the consequences of doing so.

- **If a new column is added to a table, the column initially has NULL values.**
- **You can add a column with a NOT NULL constraint to a table only if the table does not contain any rows.**
- **If a view or PL/SQL program unit depends on a base table, the alteration of the base table may affect the dependent object.**

To change the definition of the table ALTER TABLE statement is used.

The syntax of the ALTER TABLE statement is

```
ALTER TABLE < table_name >
[ADD   < columnname > | <constraints >…….]
[MODIFY   <columnname>……]
[DROP <options   >];
```

To ADD new columns

If you want to add new columns to your table than use the ALTER TABLE statement with the ADD clause. You can add one or more than one columns at the same time. The database developer will need to understand how to implement changes on the database in an effective and simple manner. The developer can do one of two things when a request to add some columns to a table comes in. One is to add the columns, and the other is to re-create the entire table from scratch. Obviously, there is a great deal of value in knowing the right way to perform the first approach. Columns can be added and modified in the Oracle database with ease using the alter table statement and its many options for changing the number of columns in the database. The following code block is an example of using the alter table statement. If the developer or the DBA needs to add a column that will have a NOT NULL constraint on it, then several things need to happen. The column should first be created without the constraint, and then the column should have a value for all rows populated. After all column values are NOT NULL, the NOT NULL constraint can be applied to it. If the user tries to add a column with a NOT NULL constraint on it, the developer will encounter an error stating that the table must be empty.

Syntax

```
ALTER TABLE <table_name>
[ADD <column_name datatype (size) | <constraints> ,…….];
```

Here in the syntax:

table_name	Name of the table
Column_name	Name of the new column
datatype	Datatype of the column
size	Size of the column
constraints	Any type of constraint you want to put

The ADD option allows you to add PRIMARY KEY, CHECK, REFERENCES constraint to the table definition.

Example:

Adding columns

- Add the fields address and phone number in the emp table of width 50 and 10 charcater respectively.

```
SQL> ALTER table emp
  Add (address varchar2 (50),phone_no varchar2(10)) ;
```

Adding Primary Key

- Alter the dept table add the Primary Key constraint to deptno.

```
SQL> ALTER table dept add Primary Key (deptno);
```

- Alter the dept table add the Primary Key constraint with name PK_deptno to deptno.

```
SQL> ALTER table dept add constraint
  PK_deptno Primary Key (deptno);
```

Adding foreign key

- Alter the emp table, add foreign key constrain to deptno column so that refers to deptno of dept table.

```
SQL> ALTER TABLE emp add foreign key(deptno)REFERENCES dept(deptno);
```

OR

```
SQL> ALTER TABLE emp
    add constraint  FK_DEPTNO foreign key (deptno)
     REFERENCES dept(deptno) ;
```