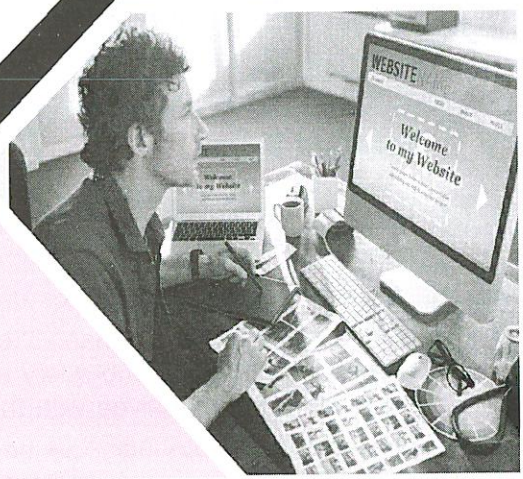
CHAPTER 28

MANAGING TABLES



CHAPTER OBJECTIVES

In this chapter you will learn:

- Creation of tables
- Different types of constraints
- Applying integrity constraints during creation of tables
- Creation of tables from existing tables
- Altering the structure of tables

28.1 INTRODUCTION TO TABLES

An Oracle database can contain multiple data structures. The table data structure stores the data in a relational database and it can be created at any time even while users are using the database. A table is composed of rows and columns. A table can represent a single entity (entity is an object whose information is stored in database) that you want to track within your system. Each entity or table has number of characteristics. The table must have a unique name through which it can be referred to after its creation.The characteristics of the table are called its attributes. These attributes can hold data. This type of a table could represent a list of the employees within your organization, or the orders placed for your company's products. A table has one attribute, which identifies each record uniquely; this attribute is called as Primary Key. Each value in the Primary Key attribute is unique and it cannot be NULL. Each record of the table is called as tuple.

28.2 STRUCTURE OF TABLE: DESIGN TABLES BEFORE CREATING THEM

Usually, the application developer is responsible for designing the elements of an application, including the tables.

Consider the following guidelines when designing your tables:

- Use descriptive names for tables, columns, indexes, and clusters.
- Be consistent in abbreviations and in the use of singular and plural forms of table names and columns.
- Document the meaning of each table and its columns with the COMMENT command.
- Normalize each table.
- Select the appropriate data type for each column.
- Define columns that allow NULL values at the last, to conserve storage space.

Before creating a table, you should also determine whether to use integrity constraints. Integrity constraints can be defined on the columns of a table to enforce the business rules of your database automatically.

28.3 TABLE CREATION RULES

There are certain standard rules for naming the tables and attributes/columns.

- Table names and column names must begin with a letter and can be 1-30 characters long.
- Names must contain only the characters A-Z, a-z, 0-9, _(underscore),$ and #.
- Names must not be an Oracle Server reserved word.
- Names must not duplicate the names of other objects owned by the same Oracle server user.
- Table name is not case sensitive.
- The table name must not be a SQL reserved word.

28.4 CREATE TABLE STATEMENT

Tables are the fundamental storage objects of the database. Tables consist of rows and columns. A single table can have a maximum of 1000 columns.

Syntax

```
Create Table tablename
(
    Column1 datatype [DEFAULT expr] [column constraint],
    column2 datatype[DEFAULT expr] [column constraint],...
    [table_constraint]}
);
```

Keywords used in the above syntax are:

Schema

It is the schema to contain the table. If it is omitted, Oracle creates the table in creator's own schema.

Table name

It is the name of the table to be created. Table name and Column name can be 1 to 30 characters long. First character must be alphabet, but name may include letters, numbers and underscores. The table name cannot be the name of another object owned by same user and cannot be a reserved word.

Column

Specifies the name of a column of the table. The number of columns in a table can range from 1 to 1000.

Datatype

It is the datatype of a column. Specifies the type and width of data to be stored in the column.

Default

It specifies a value to be assigned to the column if a subsequent INSERT statement omits a value for the column. The datatype of the expression must match the datatype of the column. A DEFAULT expression cannot contain references to other columns, the pseudo columns CURRVAL, NEXTVAL, LEVEL and ROWNUM or data constants that are not fully specified.

Example:

- Create a table student with the following fields:

Column Name	Type	Size	Description
Name	Varchar2	20	Name of the student
Class	Varchar2	15	Class of the student
Roll_no	Number	4	Roll number of the student
Address	Varchar2	30	Address of the student

```
SQL>Create table student
    (Name varchar2 (20),
    Class varchar2 (15),
    Roll_no number (4),
    Address varchar2 (30));
```

We can create tables using the SQL statement CREATE TABLE. When user SCOTT issues the following statement, it creates a table named STUDENT in its schema and stores it in the USERS tablespace.

Since creating a table is a DDL statement, an automatic commit (or save) takes place when this statement is executed.

28.5 CREATING TABLE FROM AN EXISTING TABLE

In addition to creating table as above, we can also create a new table from the existing table also.We apply the AS sub-query clause to both create the table and insert rows returned from the subquery. For example:

```
SQL> Create table emp1
AS
Select empno, ename, hiradate, sal from EMP where comm IS NULL;
```

This statement will create a new table emp1 which contains the rows returned by another table emp which satisfies the condition comm IS NULL.

Follow the following guidelines while creating a table from another table:

- If column specifications are given, the number of columns must equal the number of columns in the subquery select statement.
- If no column specifications are given, the columns of the new table are the same as the column names in the subquery.
- The select statement in the subquery will insert the records into the new table created.

28.6 ROLE OF CONSTRAINTS TO ACHIEVE DATA INTEGRITY

To understand the role of constraints in database let us consider a case, when we appear for some interview, there are certain constraints for our qualification. Person who satisfies the given qualification constraints are eligible for interview, so these restrictions are called constraints which are to be enforced to select the correct candidate. Such limitations have to be enforced on the data to achieve the integrity (correctness).

The data, which does not, satisfies the conditions will not be stored, this ensures that the data stored in the database is valid and has gone through the integrity rules which prohibit the wrong data to be entered into the database.

Oracle allows us to apply constraints on single column or more than one column through the SQL syntax that will check data for integrity. While creating a table, constraints can be placed on the values that will be entered into the column(s). SQL will reject any value that violates the criteria that were defined.

Categories of Constraints

There are two categories of constraints, these are:

- Column constraints
- Table constraints

28.6.1 Column Constraints

When we define constraints along the definition of the column while creating or altering the table structure, they are called as column constraints. Column constraints are applied only to the individual columns. These constraints are applied to the current column. A column level constraint cannot be applied if the data constraint spans across multiple columns in a table.

For example: Empno of EMP table is a primary key, which is column level constraint.

28.6.2 Table Constraints

Table constraints are applied to more than one column of a table. These constraints are applied after defining all the table columns when creating or altering the structure of the table.

A table level constraint can be applied if the data constraint spans across multiple columns in a table.

For example: in case of bank database the account number field of account holder table is not sufficient to uniquely identify the entities because two person can have joint account and hence repeating the value of account number once for each account holder,so in this case we have to apply primary key constraint on combination of account number field and name field to achieve the uniqueness. Now, in this case the primary key constraint is applied on more than one column of the table, so this is the table level constraint.

Constraints can be added to the table at the time of creating a table with the create table command or later on with the help of ALTER TABLE command. All the details of constraints are stored in data dictionary. Each constraint is assigned a name by itself. Constraints are stored in the system tables by the Oracle engine.We can also give more user friendly names to the constraints so that they can be easily referenced, otherwise the name is automatically generated of the form SYS_Cn where n is unique number.

28.7 TYPES OF CONSTRAINTS

There are following types of constraints according to their nature.

28.7.1 NOT NULL Constraint

This constraint prevents the column from accepting NULL values. In other words, in this case it is compulsory to supply some value to the constrained column during data entry. We cannot leave the column as empty. Columns without the NOT NULL constraint allow NULL values. This constraint can be applied only at column level. But one table can have more than one column level NOT NULL constraint on different columns.

Syntax:

Columnname datatype (size) [constraint constraintname] NOT NULL

Example:

- Create a table student with the fields name, roll_no, address and phone number.

```
SQL> Create table student
(Name varchar2 (20) CONSTRAINT NN_NAME NOT NULL,
Roll_no number (5) CONSTRAINT NN_ROLL NOT NULL,
Address varchar2 (40),
phone_no varchar2 (10));
```

28.7.2 Unique Constraint

The unique key allows unique values to be entered into the column i.e. every value is a distinct value in that column. When we apply a unique constraint on a group of columns, then each value of the group should be unique, they must not be repeated. A table can have more than one unique key. This constraint can be applied both at the table level and the column level. The syntax is:

Column level syntax:

Columnname datatype (size) [constraint constraintname] UNIQUE

Example:

- Create a table student with roll_no not null, unique name, unique address and unique phone number.

```
SQL> Create table student
(Roll_no number (5) NOT NULL,
name varchar2(20) constraint un_name unique,
Address varchar2 (40) unique, phone_no varchar2 (10) unique);
```

Table level syntax of unique constraint:

```
[constraint constraintname] Unique (columnname [, columnname, .....])
```

Example:

- Create a table student with roll_no not null and the combination of name, address and phone number must be unique.

```
SQL> Create table student
(Roll_no number (5) NOT NULL,
Name varchar2 (20),
Address varchar2 (40),
phone_no varchar2 (10),
Constraint tb_un UNIQUE (name, address, phone_no));
```

28.7.3 Primary Key Constraint

A primary key is used to identify each row of the table uniquely. A primary key may be either consists of a single column or group of columns. It is a combination of both the unique key constraint as well as the not null constraint i.e. no column with the primary key constraint can be left blank and it must contain unique value in each row. We can declare the Primary key of a table with NOT NULL and UNIQUE constraint. SQL supports primary keys directly with the Primary Key constraint. When primary key is applied on a single field it is called as Simple Key and when applied on multiple columns it is called as Composite Primary Key. In a table there is only one primary key. The syntax of the Primary Key:

Column level syntax:

```
Columnname datatype(size) [constraint_name] PRIMARY KEY
```

Example:

- Create a table student with roll_no as the Primary Key, unique name, address cannot be NULL and phone number.

```
SQL> Create table student
(Roll_no number (5) Primary key,
Name varchar2 (20) Unique,
Address varchar2 (40) Not Null,
phone_no varchar2 (10));
```

Table level syntax of Primary Key constraint

PRIMARY KEY (columnname [, columnname, ...])

Example:

- Create a table student with name and class as the composite primary key, address and phone_no are the other fields.

```
SQL> Create table student
  (Name varchar2 (20),
  Class varchar2 (15),
  Address varchar2 (40) Not Null,
  phone_no varchar2 (10)
  PRIMARY KEY (name, class));
```

28.7.4 Check Constraint

Check constraints allow Oracle to verify the validity of data being entered on a table against a set of constant values. These constants act as valid values. The Check constraint consists of the keyword CHECK followed by parenthesized conditions. Check constraints must be specified as a logical expression that evaluates either to TRUE or FALSE. If an SQL statement causes the condition to be false an error message will be displayed.

Syntax of the check constraint is:

Column level syntax:

Columnname datatype (size) [constraint constraintname] CHECK (logical expression)

Example:

- Create a table EMP1 with empid as primary key, phone number as unique attribute and the salary of all the employees must be greater than 5000, address and phone_no are the other fields.

```
SQL> CREATE TABLE emp1
(empid number (10),
salary number (10, 3) CHECK (salary > 5000),
home_phone number (12),
CONSTRAINT pk_empid
PRIMARY KEY (empid),
CONSTRAINT uk_phone UNIQUE (home_phone));
```

If, for example, someone tries to create an employee row for the table defined earlier with a salary of Rs. 1000, Oracle will return an error message saying that the record data defined for the SALARY column has violated the check constraint for that column.

Example:

- Create a table emp1 with empid as primary key, phone number as unique attribute and the department of the employees must be either general or accounts with name and class as the composite primary key, address and phone_no are the other fields.

```
SQL> CREATE TABLE emp1
(empid number (10),
deptname varchar2 (20) CHECK deptname in ('general','accounts'),
home_phone number (12),
CONSTRAINT pk_empid
PRIMARY KEY (empid),
CONSTRAINT uk_phone UNIQUE (home_phone));
```

Another Example:

```
CREATE TABLE emp(
code CHAR(4) PRIMARY KEY, name VARCHAR2(15) NOT NULL,
department CHAR(10) CHECK (department IN ('mkt','sales','Acct'))
age NUMBER CHECK (age between 18 and 55),
basic NUMBER);
```

Limitations of Check Constraints

Check constraints have a number of limitations, these are:

- A column level check constraint cannot refer to another column of any table.
- It cannot refer to special keywords that can have values in them, such as user, sysdate or rowid.

Violations of Check Constraint

The check constraint in the table definition earlier is valid, but the one in the following excerpt from a table definition is not:

```
CREATE TABLE address
(.....,
    city VARCHAR2(80) check(city in (SELECT city FROM cities))
...);
```

— It is invalid because here check constraint refers a column of different table through SELECT statement.

Table level syntax:

Check (logical expression)

Example:

```
SQL> CREATE TABLE emp1
(empid number (10),
salary number (10, 3),
comm number (10, 3),
home_phone number (12),
CONSTRAINT pk_empid PRIMARY KEY (empid),
```

```
    CONSTRAINT uk_phone UNIQUE (home_phone),
    CHECK (salary > comm));
```

Example:

```
    SQL> CREATE TABLE emp1
    (empid number (10),
    salary number (10, 3),Home_phone number (12),
    CONSTRAINT pk_empid PRIMARY KEY (empid),
    CONSTRAINT uk_phone UNIQUE (home_phone),
  CHECK (salary < 500000));
```

28.7.5 Default Constraint

The default constraint allows the user to insert the values in the columns where the user do not want to insert the value.The most common example of this constraint is NULL value, which is inserted in columns not defined with the NOT NULL constraint. This is not actually a constraint, it only specifies that a value should be inserted in a column if the user does not enter a value. The default value assignments are defined at the time of create table command. The datatype of the default value should match the datatype of the column.

The syntax is:

```
Columnname datatype (size) DEFAULT value;
```

Example:

- Create a table student with roll number of the student as the primary key, the default value of marks secured by the student should be 100 the other fields are the name and address of the student.

```
SQL> Create table student
(roll_no number(5) primary key,
Name varchar2 (20) not null,
Marks number (3) default 100,
address varchar2 (30) not null));
```

28.7.6 Foreign Key Constraint

A foreign key is a kind of constraint which establishes a relationship among tables. A foreign key may be a single column or the combination of columns which derive their values based on the primary key or unique key values from another table. A foreign key constraint is also known as the referential integrity constraint, as its values correspond to the actual values of the primary key of other table.

A table in which there is a foreign key, it is called as the detail table or the child table and the table from which it refers the values of the primary key, it is called as the master table or the parent table. A foreign key must have the corresponding primary key or unique key value in the master table. Whenever you insert a value in the foreign key column, it checks the value from the primary key of the parent table. If the value is not present, it gives an error. The parent table can be referenced in the foreign key by using the References option.

Consider two tables (EMP, DEPT) given below.

Target Attribute

EMP → Foreign

Primary Key → DEPT

Empno	Ename	Job	Sal	Deptno
1	A	Clerk	4000	10
2	A	Clerk	4000	30
3	B	Mgr	8000	20
4	C	Peon	2000	40
5	D	Clerk	4000	10
6	E	Mgr	8000	50

Deptno	Dname	Loc
10	A	Asr
20	B	Jal
30	C	Qadian
40	D	-

If we try to insert information of employee with deptno 50, then this is an invalid information, because there is no deptno 50 exists in the company(as shown in table DEPT).Then, this invalid information should be prevented from insertion, which would only be possible if deptno of EMP table refer the deptno of DEPT table. It means that only those values are permitted in deptno of EMP table, which appears in the deptno attribute of the DEPT table. Thus, we can say that deptno of EMP table is the foreign key which refers the primary key deptno of DEPT table. Thus, we can insert the empno 6 with any deptno from 10,20,30 and 40.

Null may also be permitted in the deptno of EMP table. Here, deptno of DEPT table is Target attribute and DEPT is the target table.

Consider another example:

Student

Rno	Name	Class_Code
1	A	2
2	B	1
3	C	-

Class

Class_Code	Name
1	B.TECH
2	B.TECH
3	BBA

Here, college has three valid classes with class code 1,2 and 3. The class_code(foreign key) of student table refers class_code of class table.

The syntax of the foreign key at the column level is:

Columnname datatype (size) references tablename [(columnname)]

Example: Create a table emp_detail in which the deptno field refers its value from the deptno field of the dept table.

```
SQL> Create table emp_detail
    (Empno number (4) primary key,
    Ename varchar2 (20) not null,
```

Hiredate date not null,
Deptno number (2) references dept (deptno),
Salary number (6, 2));

So, in the emp_detail table deptno is a foreign key, which gets its values from the parent table i.e. the dept table.

The syntax of the foreign key at the table level is:

Foreign key (columnname [,columnname,.....]) references tablename (columnname [,columnname.....])

Example:

- Create a table emp_detail in which the deptno and dname together refers its value from deptno and dname field of dept table.

```
SQL> Create table emp_detail
  (Empno number (4) primary key,
  Ename varchar2 (20) not null,
  Hiredate date not null,
  Deptno number (2),
  Dname varchar2 (10),
  Salary number (6, 2)
  Foreign key (deptno,dname) references dept(deptno,dname)) ;
```

So in this table deptno and dname is the foreign key which references the corresponding fields of the dept table.

Important Note:

It is important to consider what will happen to child record if parent record is updated or deleted.

For example, let us consider the following database:

Emp (eno, ename, job, dno) Having eno as primary key and dno as forign key referencing dept (dno).

Dept (dno, dname) having dno as primary key.

Here, dept table is parent as shown below:

EMP

ENO	ENAME	JOB	DNO
1	RAJ	CLERK	10
2	RAM	PROF	20
3	RAHAT	PROF	10
4	RISHAN	ASSOC PROF	20

DEPT

DNO	DNAME
10	COMPUTER
20	CIVIL

Now, it is important to consider what will happen, if user tries to delete record of deptno 20 from DEPT table. Then of course record of emp no 2 and 4 become invalid as they belong to this department.

There are four options to handle this situation.

- Prevent the dept being deleted until all its employees are re allotted to some other dept
- Automatic delete the employees who belong to that dept
- Set the dept column for the employees who belong to that dept to NULL
- Set the dept column of these employees to some default value which indicate they are currently not allotted to any dept

To achieve this, four options van be set in CREATE TABLE command

ON DEELTE RESTRICT
ON DELETE CASCADE
ON DELETE SET NULL
ON DELETE SET DEFAULT

The syntax to each of these cases has been given below:

```
SQL> CRAETE TABEL DEPT(DNO NUMBER(2) PRIMARY KEY,
DNAME CHAR(20));
```

Case-I

```
SQL> CREATE TABLE EMP(ENO NUMBER(2) PRIMARY KEY,
ENAME CHAR(20), JOB CHAR(20)
DNO NUMBER(2) REFERENCES DEPT(DNO) ON DELETE RESTRICT;
```

It will prevent the dept being deleted until all its employees are re allotted to some other dept. This is the default case and will be applicable if did not give any option. It means the below command will also has the same effect.

```
SQL> CREATE TABLE EMP(ENO NUMBER(2) PRIMARY KEY,
ENAME CHAR(20), JOB CHAR(20)
DNO NUMBER(2) REFERENCES DEPT(DNO);
```

Case-II

```
SQL> CREATE TABLE EMP(ENO NUMBER(2) PRIMARY KEY,
ENAME CHAR(20), JOB CHAR(20)
DNO NUMBER(2) REFERENCES DEPT(DNO) ON DELETE CASCADE;
```

It will also delete the employees who belong to that dept.

Case-III

```
SQL> CREATE TABLE EMP(ENO NUMBER(2) PRIMARY KEY,
ENAME CHAR(20), JOB CHAR(20)
DNO NUMBER(2) REFERENCES DEPT(DNO) ON DELETE SET NULL;
```

It will set the dept column for the employees who belong to that dept to NULL.

Case-IV

```
SQL> CREATE TABLE EMP(ENO NUMBER(2) PRIMARY KEY,
ENAME CHAR(20), JOB CHAR(20)
```

DNO NUMBER(2) REFERENCES DEPT(DNO) ON DELETE SET DEFAULT;

It will set the dept column of these employees to some default value which indicate they are currently not allotted to any dept. This concept is not implemented in Oracle.

28.8 CREATING A TABLE WITH ROWS FROM ANOTHER TABLE

A table is created using CREATE TABLE statement with rows in place, derived from another table.

Syntax

```
CREATE TABLE table_name
[(Column name.....)]
AS SELECT statement;
```

The table will be created with the specified columns and the rows retrieved by the SELECT statement inserted into it. If all the columns in the SELECT statement have well defined names, (that is, no expressions, and so on), the column specifications may be omitted. If column specifications are given, the number of columns must equal the number of items in the SELECT list. Constraints information is inherited from the selected table. Data type for the column cannot be specified.

Example

- Create a table emp2 from emp table having employee number, name, mgr, and sal of employees in department 10 and 20.

```
SQL>  CREATE TABLE emp2
      AS
      SELECT empno, ename, mgr, sal FROM emp
      WHERE deptno in (10,20);
```

Table created

The above table contains all the records present in emp table having deptno 10 or deptno 20. Also the constraints attached to columns in emp table are attached to the columns of emp2 table.

Create a table Salary from table emp having empno, ename and salary details.

```
SQL> CREATE TABLE salary(employee_number,Name, Salary)
     AS
     SELECT empno, ename, sal FROM emp;
```

Table created

The above table contains records of table emp having empno, ename, salary details.

28.9 REQUIRED PERMISSIONS TO CREATE A TABLE

To create a table in users own schema, user must have CREATE TABLE system privilege. To create a table in another user's schema, user must have CREATE ANY TABLE system privilege. Also the owner of the schema to contain the table must have either

space quota on the table space to contain the table or UNLIMITED TABLESPACE system privilege.

28.10 TO DISPLAY INFORMATION ABOUT TABLE

To display the structure of the table we can use DESCRIBE statement. It displays the information about the column names of the table, their data types and special attributes associated with them like NOT NULL.

Syntax:

```
DESCRIBE table_name;
```

Example:

```
DESCRIBE EMP
```

The output is:

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

It can also works with first four characters of DESCRIBE i.e. DESC EMP also works. It is a SQL*PLUS statement so not available in the buffer.

In order to display all the tables stored in database, we can use the following statement:

```
SQL>SELECT * FROM TAB;
```

The output is:

TNAME	TABTYPE CLUSTERID
ACCOUNT	TABLE
BONUS	TABLE
DEPT	TABLE
EMP	TABLE

NOTE: The CLUSTERID is not displayed by the SELECT * FROM CAT:

28.11 THE USER_CONSTRAINTS TABLE

A user can define number of constraints in a table and if the user wishes to see the name of the constraints,they are available in the USER_CONSTRINTS table.The DESCRIBE option only gives the name of the fields in the table and NOT NULL constraint associated with the column. So the user_constraints table provides information pertaining to the names

of all the constraints on the table. Following are some of the fields in the user_constraints table:

Column name	Description
Owner	The owner of the constraint
Constraint_type	The type of constraint
Constraint_name	Name of the constraint
Table_name	The name of the associated table
Search_condition	The search condition used(for check constraint)
R_owner	The owner of the table referenced by the foreign key
R_constraint_name	The name of the constraint referenced by a foreign key

So you can see the name of the constraints associated with the table by writing a simple SQL query.

```
SQL>Select owner, constraint_name from user_constraints
where table_name = 'EMP';
```

The Output is:

OWNER	CONSTRAINT_NAME
SCOTT	PK_EMP
SCOTT	FK_DEPTNO

28.12 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 >];
```

28.12.1 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) ;
```

28.12.2 To MODIFY columns

Several conditions apply to modify the datatypes of existing columns or to add columns to a table in the database. The general thumb rule is that increases are generally OK, while decreases are usually a little trickier. Some examples of increases that are generally acceptable are listed as follows:

- Increases to the size of a VARCHAR2 or CHAR column.
- Increases in size of a NUMBER column.
- Increasing the number of columns in the table.

The following list details the allowable operations that decrease various aspects of the database:

- Reducing the number of columns in a table (empty table only)
- Reducing the size of a NUMBER column (empty column for all rows only)
- Reducing the length of a VARCHAR2 or CHAR column (empty column for all rows only)
- Changing the datatype of a column (empty column for all rows only)

The Modify option changes the following of the existing column

- Datatype

- Column width
- Constraints i.e. DEFAULT, NOT NULL or NULL.

There are certain restrictions while making these modifications:

- The type and/or size of a column can be changed, if every row for the column is NULL.
- An existing column can be modified to NOT NULL only if it has a non-null value in every row.

The syntax of the MODIFY option is:

```
Alter table <tablename >
[Modify < columnname >.....];
```

Examples:

- Modify the job column of the emp table by increasing its width to varchar2 (40).

```
SQL> Alter table emp
Modify job varchar2 (40);
```

- Modify the job column of the emp table by increasing its width to varchar2 (35) and applying the constraint of NOT NULL.

```
SQL> Alter table emp
Modify job varchar2 (35) NOT NULL;
```

28.12.3 The DROP option

This option removes the constraints from a table. When a constraint is dropped, any associated index with that constraint (if there is one) is also dropped.

The syntax is:

```
Alter table < tablename >
[DROP <constraints > .....] ;
```

Example:

- Remove the primary key constraint from the emp table.

```
SQL> Alter table emp
DROP primary key;
```

- Remove the constraint on the deptno field of the dept table.

```
SQL> Alter table dept DROP constraint pk_deptno ;
```

28.12.4 Drop column option

Alter table command can also be used to drop the existing columns of the tables, but this option is available only in Oracle 8i or versions after that.

Syntax:

```
ALTER TABLE <table_name> DROP COLUMN <column_name>;
```

Example:

- Drop the column ename of emp table.

```
SQL>  ALTER TABLE emp DROP COLUMN ename;
```

28.13 REMOVING TABLES

In order to remove a table from the database, the drop table command must be executed.

The syntax is:

DROP table <tablename >;

Example:

SQL> DROP TABLE emp;

However, dropping tables may not always be that easy. Recall from the earlier lesson that when you disable constraints like primary keys that have foreign-key constraints in other tables depending on their existence, you may have some errors. The same thing happens when you try to drop the table that has a primary key referenced by enabled foreign keys in another table. If you try to drop a table that has other tables' foreign keys referring to it, the following error will ensue:

ORA-02266: unique/primary keys in table referenced by enabled foreign keys

When there are foreign-key constraints on other tables that reference the table to be dropped, then you can use on delete cascade option. The constraints in other tables that refer to the table being dropped are also dropped with cascade option. There are usually some associated objects that exist in a database along with the table. These objects may include the index that is created by the primary key or the unique constraint that is associated with columns in the table. If the table is dropped, Oracle automatically drops any index associated with the table as well.

SQL> DROP TABLE dept
CASCADE CONSTRAINTS;

Alternately, you can disable or drop the foreign key in the other table first and then issue the drop table statement without the cascade constraints option. However, with this method you run the risk that many other tables having foreign keys that relate back to the primary key in the table you want to drop will each error out, one at a time, until you disable or drop every foreign-key constraint referring to the table. If there are several, your drop table activity may get extremely frustrating.

28.14 AVAILABLE TABLES AS DATA DICTIONARY

User tables are tables created by the user such as student. There is another collection of tables, which are owned by the SYS user in the Oracle database known as data dictionary. These tables are maintained and created by the Oracle Server and contains information about the database. Information stored in the data dictionary include name of the Oracle Server users, privileges granted to users, database objects names, table constraints and other information. You can query the following data dictionary tables to view various database objects owned by you.The data dictionary tables frequently used are:

- USER_TABLES
- USER_OBJECTS
- USER _CATALOG

FLASH BACK

The table data structure stores the data and it is composed of rows and columns. A table can also represent a relation between two entities. First step before creating tables is to design them so that it gives a meaningful picture. One must follow rules to create a table as the name of the should not be a SQL reserve word, not more than 30 characters, must contain only the characters A-Z, a-z, 0-9, _, $, #. the name of the table is case insensitive.

The table can be created with the create statement and the table can be created from the existing table also which may or may not contain data. We can impose number of constraints on the table as a whole as well as on the columns also. these constraints provides data integrity and are divided into two categories:

- Column constraints(applied on one column)
- Table constraints(applied on more than one column)

The different types of constraints are;

- Not null constraint
- Primary key constraint
- Foreign key constraint
- Unique constraint
- Default constraint
- Check constraint

The user can see the names of the constraints in the user_constraints table. Finally, if we want to change the structure of the table we have the ALTER table command. With the alter statement we can add column or constraints, modify the datatypes or to add more columns, and drop the column or table itself.

REVIEW QUESTIONS

1. What is meant by table and what are creation rules?
2. Explain the syntax of Create table statement.
3. What is the importance of constraints and what are the different categories of constraints.
4. Differentiate between column level and table level constraints.
5. What are the different types of constraints? Explain the importance of each constraint with suitable example.
6. Explain the following:
   (a) NOT NULL constraint
   (b) Unique constraint
   (c) Primary Key constraint
   (d) Foreign Key constraint
   (e) Check constraint
7. How you can establish relationship between the different tables.
8. What is the use of DEFAULT constraint?
9. Explain the need to alter the table structure. What is the syntax to change the structure of existing table?
10. Explain the following with alter table statement:
    (a) To add new columns.
    (b) To modify columns.
    (c) To drop columns.

11. How we can create a table from existing table?
12. How we can drop a table from database?

Lab Assignment-7

1. Create emp table with given constraints.
   employee table(empno, ename, job, sal, deptno)
   Here, empno is primary key, ename is unique, job in (Prof, AP, and Lect), sal is not NULL, and deptno default is 10.
2. Insert appropriate records, check error messages in case of violation
3. List all the constraint names for emp table.
4. Create table book with
   Rno number—PK
   DateOfIssue-date
   DateOfReturn-date
   DateOfReturn > DateOfIssue
5. Insert appropriate records, check error messages in case of violation and list all the constraint names for book table.
6. Create table STUDENT with
   Rno-Number Class-Char Marks-Number Primary key(rno,class) Marks>0
7. Insert appropriate records, check error messages in case of violation and list all the constraint names for STUDENT table.
8. Create table S which has the following attributes (Salesperson table) (sno, sname, city) Where sno is primary key
9. Create table P which has the following attributes (Part table) (pno, pname, color) Where pno is primary key
10. Create table SP which has the following attributes (sno, pno qty)
    Where combination of (sno, pno) is primary key, also sno and pno are foreign keys
11. Create table dept which has the following attributes (department table) (deptno, dname)
    Where deptno is primary key, dname in (Acc, comp, elect)
12. Create table emp which has the following attributes (employee table)
    (empno, ename, job, sal, deptno) Where empno is primary key, ename is unique, job in (Prof, AP, and Lect), sal is not NULL, and deptno is foreign key
13. Drop the unique constraint on ENAME
14. Drop the Foreign Key constraint on DEPTNO
15. Add Foreign Key constraint on DEPTNO
16. Change Data type of ENAME
17. Change width of JOB
18. Add COMMISSION column in EMP table
19. Drop CITY column from S table
20. Create duplicate copy of EMP table
21. Copy structure of DEPT table in another table with different column names

22. Change the name and job of the employee whose EMPNO =100 during creation of duplicate table.
23. Delete the record of employee who belong to computer department
24. Drop DEPT Table
25. Drop duplicate table of EMP table

HANDS ON SESSION

1. User would like to insert a row into the EMPLOYEE table, which has three columns: EMPID, LASTNAME, and SALARY. The user would like to enter data for EMPID 59694, LASTNAME Harris, but no salary. Which statement would work best?
   (a) insert into EMPLOYEE values (59694,'HARRIS', NULL);
   (b) insert into EMPLOYEE values (59694,'HARRIS');
   (c) insert into EMPLOYEE (EMPID, LASTNAME, SALARY) values (59694,'HARRIS');
   (d) insert into EMPLOYEE (select 59694 from 'HARRIS');
2. Which line of the following statement will produce an error?
   (a) create table GOODS (b) GOODNO number,
   (c) GOOD_NAME varchar2 check(GOOD_NAME in (select NAME FROM AVAIL_GOODS)),
   (d) constraint PK_GOODS_01 (e) primary key (GOODNO));
   (f) There are no errors in this statement.
3. Developer ANJU executes the following statement: CREATE TABLE animals AS SELECT * from MASTER.ANIMALS; What is the effect of this statement?
   (a) A table named ANIMALS will be created in the MASTER schema with the same data as the ANIMALS table owned by ANJU.
   (b) A table named ANJU will be created in the ANIMALS schema with the same data as the ANIMALS table owned by MASTER.
   (c) A table named ANIMALS will be created.
   (d) in the ANJU schema with the same data as the ANIMALS table owned by MASTER.A table named MASTER will be created in the ANIMALS schema with the same data as the ANJU table owned by ANIMALS.
4. To increase the number of NULLable columns for a table:
   (a) Use the alter table statement
   (b) Ensure that all column values are NULL for all rows
   (c) First increase the size of adjacent column datatypes, then add the column
   (d) Add the column, populate the column, then add the NOT NULL constraint
5. Which command shows the structure of a table?
   (a) START (b) DESCRIBE (c) APPEND (d) GET
6. When using the DESCRIBE command, which is NOT displayed on the screen?
   (a) Constraints. (b) column types.

(c) whether a column must be not null. (d) column names.

7. In a table, there is a column called GPA defined as NUMBER (5,3). What does the 5 represent?
   (a) Largest exponent that can be used with a GPA.
   (b) Total number of digits for the entire value
   (c) Number of digits to the right of the decimal point.
   (d) Maximum number of digits to the left of the decimal point.

Solution Keys

1. **(a)** The positional criteria for not specifying column order is met by the data in the values clause.
2. **(c)** A check constraint cannot contain a reference to another table, nor can it reference a virtual column such as ROWID or SYSDATE
3. **(c)** A table is always created in the schema of the user who created it, and that since the create table as select clause was used, ANJU.ANIMALS will have the same data as MASTER.ANIMALS.
4. **(a)** The alter table statement is the only choice offered that allows the developer to increase the number of columns per table.

5. (b) **6. (a)** **7. (b)**

○○○○○