# 4. CREATING SAMPLE TABLES

- What is a constraint?
- Types of constraints?
- Sample tables
- Creating integrity constraints
- Creating example table
- Inserting sample data
- Summary
- Exercises

## What is a constraint?

In the previous chapter we have seen how to create a table using CREATE TABLE command. Now we will understand how to define constraints. Constraints are used to implement standard and business rules. Data integrity of the database must be maintained. In order to ensure data has integrity we have to implement certain rules or constraints. As these constraints are used to maintain integrity they are called as integrity constraints.

### Standard rules

Standard constraints are the rules related to primary key and foreign key. Every table must have a primary key. Primary key must be unique and not null. Foreign key must derive its values from corresponding parent key. These rules are universal and are called as standard rules.

### Business rules

These rules are related to a single application. For example, in a payroll application we may have to implement a rule that prevents any row of an employee if salary of the employee is less than 2000. Another example is current balance of a bank account
Must be greater than or equal to 500.

Once the constraints are created, Oracle server makes sure that the constraints are not violated whenever a row is inserted, deleted or updated. If constraint is not satisfied then the operation will fail.

Constraints are normally defined at the time of creating table. But it is also possible to add constraints after the table is created using ALTER TABLE command. Constraints are stored in the Data Dictionary (a set of tables which stores information regarding database).

Each constraint has a name; it is either given by user using CONSTRAINT option or assigned by system. In the later case, the name is **SYS_Cn**; where ***n*** is a number.

***Note:*** *It is recommended that you use constraint name so that referring to constraint will be easier later on.*

## Types of constraints

Constraints can be given at two different levels. If the constraint is related to a single column the constraint is given at the column level otherwise constraint is to be given at the table level. Base on the where a constraint is given, constraint are of two types:

- Column Constraints
- Table Constraints

### Column Constraint

A constraint given at the column level is called as Column Constraint. It defines a rule for a single column. It cannot refer to column other than the column at which it is defined. A typical example is PRIMARY KEY constraint when a single column is the primary key of the table.

### Table Constraint

A constraint given at the table level is called as Table Constraint. It may refer to more than one column of the table. A typical example is PRIMARY KEY constraint that is used to define composite primary key. A column level constraint can be given even at the table level, but a constraint that deals with more than one column must be given only at the table level.

The following is the syntax of CONSTRAINT clause used with CREATE TABLE and ALTER TABLE commands.

```
[CONSTRAINT constraint]
{ [NOT] NULL
| {UNIQUE | PRIMARY KEY}
|  REFERENCES [schema.] table [(column)]
         [ON DELETE CASCADE]
|  CHECK (condition) }
```

The following is the syntax of table constraint.

```
[CONSTRAINT constraint]
{ {UNIQUE | PRIMARY KEY} (column [,column] ...)
|  FOREIGN KEY (column [,column] ...)
    REFERENCES  [schema.] table [(column [,column] ...)]
        [ON DELETE CASCADE]
| CHECK (condition) }
```

The main difference between column constraint and table constraint is that in table constraint we have to specify the name of the column for which the constraint is defined whereas in column constraint it is not required as constraint is given on immediately after the column.

Now let us understand sample table to be throughout this book. It is very important to understand these tables to get the best out of this book. I have made these tables to be easy to understand.

## Sample tables

The following are the sample tables used throughout the book. These tables store information about course, batches and subject. There are six tables to store the required information by typical training center.

Let us first understand the meaning of each table.

The following are the required tables of our application.

| **Table Name** | **Description** |
|---|---|
| Courses | Contains the details of all the courses offered by the institute. |
| Faculty | Contains the details of the faculty members of the institute. |
| Course_faculty | This table contains information regarding which faculty can handle which course. It also contains rating regarding how good a faculty member is in handling a particular course. The rating is based on previous experience of the faulty member with that course. |
| Batches | Contains the information about all the batches. It contains information about all the batches that started and completed, on going and scheduled but not yet started. |
| Students | Contains information about all the students. Each student is assigned a new roll number whenever he/she joins a new course. |
| Payments | Information about all the payments made by students. A single student may pay course fee in multiple installments for a single course. |

**Table 1:** Sample tables.

The following few tables will give the list of columns of each of the table given in table 1.

## COURSES Table

Contains information related to each course. Each course is given a unique code called course code.

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| CCODE | VARCHAR2(5) | Course Code. This is the primary key of the table. |
| NAME | VARCHAR(30) | Name of the course. |
| DURATION | NUMBER(3) | Duration of the course in no. of working days. |
| FEE | NUMBER(5) | Course fee of the course. |
| PREREQUISITE | VARCHAR2(100) | Prerequisite knowledge to do the course. |

The following are the required constraints of COURSES table.

- CCODE is primary key.
- FEE must be greater than or equal to 0.
- DURATION must be greater than or equal to 0.

## FACULTY Table

Contains information about all the faculty members. Each faculty member is given a code called as FACCODE.

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| FACCODE | VARCHAR2(5) | Faculty code. This is the primary key of the table. |
| NAME | VARCHAR2(30) | Name of the faculty. |
| QUAL | VARCHAR2(30) | Qualification of the faculty member. |
| EXP | VARCHAR2(100) | Experience of the faculty member. |

The following are the constraints of FACULTY table.

- FACCODE is primary key.

## COURSE_FACULTY table

Contains information regarding which faculty member can take which course. A single faculty member may be capable of handling multiple courses. However, each member is given a grade depending on his expertise in handling the subject. The grade will be wither A, B or C.

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| FACCODE | VARCHAR2(5) | Faculty code. |
| CCODE | VARCHAR2(5) | Course the faculty can handle. |
| GRADE | CHAR(1) | Rating of faculty’s ability to handle this particular code. A – Very good, B- Good, C- Average. |

The following are the constraints of the table.

- FACCODE is a foreign key referencing FACCODE column of FACULTY table.
- CCODE is a foreign key referencing CCODE column of COURSES table.
- Primary key is consisting of FACCODE and CCODE.
- GRADE column must contain either A, B or C.

## Batches table

Contains information about all the batches. These batches include batches that were completed, that are currently running and that are scheduled but yet to start.

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| BCODE | VARCHAR2(5) | Code that is assigned to each batch. This is the primary key of the table. |
| CCODE | VARCHAR2(5) | Course code of the course of this batch. This is a foreign key referencing CCODE of COURSES table. |
| FACCODE | VARCHAR2(5) | Code of the faculty member taking this batch. |
| STDATE | DATE | Date on which the batch has started or scheduled to start if batch has not yet started. |
| ENDDATE | DATE | Date on which the batch has completed. If batch is not completed this will be null. |
| TIMING | NUMBER(1) | Number indicating the timing of the batch. 1- morning, 2 – after noon, and 3-evening. |

The following are the required constraints of this table.

- BCODE is the primary key.
- CCODE is a foreign key referencing CCODE of COURSES table.
- FACCODE is a foreign key referencing FACCODE of FACULTY table.
- STDATA must be <= ENDDATE
- TIMING column must be 1, 2 or 3.

## STUDENTS table

Contains information about all the students of the institute. Each student is given a roll number. Roll number will be allotted to each student of each batch.

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| ROLLNO | NUMBER(5) | Roll number that is assigned to each student. This is the primary key of the table. |
| BCODE | VARCHAR2(5) | Code of the batch to which student belongs. This is the foreign key referencing BCODE of BATCHES table. |
| NAME | VARCHAR2(30) | Name of the student. |
| GENDER | CHAR(1) | Gender of the student. M for male and F for female. |
| DJ | DATE | Date on which the student has joined. |
| PHONE | VARCHAR2(10) | Contact number of the student. |
| EMAIL | VARCHAR2(30) | Email address of the student. |

The following are the constraints of the table.

- ❑ ROLLNO is the primary key.
- ❑ BCODE is a foreign key referencing BCODE of BATCHES table.
- ❑ GENDER may be either M or F.

## PAYMENTS table

Contains information about all the payment made by students of all bathes.

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| ROLLNO | NUMBER(5) | Roll number of the student paying the fee. |
| DP | DATE | Date on which the amount is paid. |
| AMOUNT | NUMBER(5) | The amount paid by student. |

The following are the constraints.

- ❑ Primary key is consisting of ROLLNO and DP.
- ❑ AMOUNT must be >= 25

## Creating Integrity Constraints

In the following few sections we will see how to integrity constraints.

## NOT NULL Constraint

Used to prevent any null value from entering into column. This is automatically defined for column with PRIMARY KEY constraint.

The following example shows how you can define course name as not null column using NOT NULL constraint.

```
CREATE TABLE COURSES
( ...,
  name varchar2(20)
          CONSTRAINT courses_name_nn NOT NULL,
  ...
);
```

CONSTRAINT option is used to given a name to constraint.  The convention followed here is TABLENAME_COLUMN_TYPE.

## PRIMARY KEY Constraint

This constraint is used to define the primary key of the table. A primary key is used to uniquely identify rows in a table. There can be only one primary key in a table.  It may consist of more than one column. If primary key is consisting of only one column, it can be given as column constraints otherwise it is to be given as table constraint.

***Note:*** *You have to use table constraint to define composite primary key.*

Oracle does the following for the column that  has PRIMARY KEY constraint.

- Creates a unique index to enforce uniqueness. We will discuss about indexes later in this book.
- Defines NOT NULL constraint to prevent null values.

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

```
CREATE TABLE COURSES
( ccode  varchar2(5) CONSTRAINT courses_pk PRIMARY KEY,
  ...  );
```

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

```
CREATE TABLE COURSE_FACULTY
 ( ...,
   CONSTRAINT COURSE_FACULTY_PK PRIMARY KEY (ccode,faccode)
 );
```

## UNIQUE Constraint

Enforces uniqueness in the given column(s). Oracle automatically creates a unique index for this column.

The following example creates unique constraint on NAME column of COURSES table.

```
CREATE TABLE courses
( ... ,
  name varchar2(20)
          CONSTRAINT courses_name_u  UNIQUE,
  ... );
```

If two or more columns collective should be unique then UNIQUE constraint must be given at the table level.

## FOREIGN KEY Constraint

A foreign key is used to join the child table with parent table. FOREIGN KEY constraint is used to provide *referential integrity,* which makes sure that the values of a foreign key are derived from parent key. It can be defined either at the table level or at the column level.

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.

The following example defines foreign key constraint for CCODE of COURSE_FACULTY table.

```
CREATE TABLE course_faculty
(ccode varchar2(5)
     CONSTRAINT course_faculty_ccode_fk REFERENCES  courses(ccode),
   ...
);
```

***Note:*** *When the name of the column in the referenced table is same as the foreign key then column need not be given after the table name. It means* ***REFERENCES courses*** *in the above example will suffice.*

Table level constraint is used when foreign key is a composite foreign key.

## ON DELETE CASCADE option

As mentioned earlier, after a foreign key is defined, Oracle will NOT allow any parent row to be deleted if it has dependent rows in the child table.

For example, if CCODE in COURSE_FACULTY table is defined as foreign key referencing CCODE column of COURSES table then it is NOT possible to delete rows from COURSES table if dependent rows exists in COURSE_FACULTY table.

However, by using ON DELETE CASCADE it is possible to delete all child rows while parent row is being deleted.

The following code shows how to use ON DELETE CASCADE option.

```
CREATE TABLE course_faculty
(ccode varchar2(5)
     CONSTRAINT course_faculty_ccode_fk REFERENCES  courses(ccode)
               ON DELETE CASCADE,
   ...
);
```

## CHECK Constraint

Defines the condition that should be satisfied before insertion or updation is done.

The condition used in CHECK constraint may NOT contain:

- A reference to pseudo column SYSDATE
- Subquery

If it is given as column constraint, it can refer only to current column. But if it is given as table constraint, it can refer to more than one column of the table. In neither case it can refer to a column of other tables.

The following example shows how to create CHECK constraint to make sure GRADE column of COURSE_FACULTY contains letters A, B and C only.

```
CREATE TABLE course_faculty
 ( ...,
   grade char(1) CONSTRAINT course_faculty_grade_chk
          CHECK  ( grade in ('A','B','C') ),
   ...
 );
```

The above CHECK constraint does not allow any other characters other than A, B and C. It must be noted that character comparison is always case sensitive. So to ignore case differences you can convert GRADE to uppercase before comparison made as follows:

```
CREATE TABLE course_faculty
 ( ...,
   grade char(1) CONSTRAINT course_faculty_grade_chk
          CHECK  ( upper(grade) in ('A','B','C') ),
   ...
 );
```

The following is an example of CHECK constraint at table level. The constraint makes sure the starting date (STDATE) of a batch is less than or equal to ending date (ENDDATE) of the batch.

```
CREATE TABLE batches
 ( ...,

   CONSTRAINT batches_dates_chk
        CHECK  ( stdate <=  enddate),
 );
```

## Creating sample tables

Here is the script to create all six tables required in this application. Just run this script from SQL> prompt of SQL*PLUS using START command. You can down load this and next script, which is used to insert sample data, from my web site using the URL *www.srikanthtechnologies.com/books/oraclebook.asp*.

```
rem *******  script to create tables for oracle book   ****************
rem
rem                   Author : P.Srikanth
rem                   Date   : 4-aug-2001
rem                   place  : Vizag.
rem
rem *******************************************************************
rem --------- first drop all existing tables ------------------

drop table payments cascade constraints;
drop table students cascade constraints;
drop table batches cascade constraints;
drop table course_faculty cascade constraints;
drop table faculty cascade constraints;
drop table courses cascade constraints;


create table  courses
(
   ccode        varchar2(5)  constraint courses_pk  primary key,
   Name         varchar2(30) constraint courses_name_u unique,
   Duration     number(3)    constraint courses_duration_chk
                             check( duration >= 1),
   fee           number(5)   constraint courses_fee_chk
                             check( fee >= 0 ),
   Prerequisite varchar2(100)
);


create table  faculty
(
   fCODE     Varchar2(5) constraint faculty_pk  primary key,
   Name      varchar2(30),
   qual      varchar2(30),
   exp       Varchar2(100)
);


create table  course_faculty
(
   fcode   varchar2(5) constraint course_faculty_fcode_fk
                          references faculty(fcode),
   CCODE   Varchar2(5) constraint course_faculty_ccode_fk
                          references courses(ccode),
   grade   char(1)  constraint course_faculty_grade_chk
                          check ( upper(grade) in ('A','B','C') ),
   constraint course_faculty_pk  primary key(ccode,fcode)
);
```

```
create table batches
(
   bCODE   Varchar2(5) constraint batches_pk primary key,
   ccode   varchar2(5) constraint batches_ccode_fk
                         references courses(ccode),
   fcode   varchar2(5) constraint baches_fcode_fk
                         references faculty(fcode),
   stdate  date        constraint batches_stdate_nn not null,
   enddate date,
   timing  number(1)   constraint batches_timing_chk
                           check( timing in (1,2,3) ),
   constraint batches_dates_chk  check ( stdate <= enddate)
);
```

```
create table students
(
   rollno   number(5)   constraint students_pk primary key,
   bcode    varchar2(5) constraint students_bcode_fk
                          references batches(bcode),
   name     varchar2(30),
   gender   char(1)      constraint students_gender_chk
                              check( upper(gender)  in ('M','F')),
   dj       date,
   phone    varchar2(10),
   email    varchar2(30)
);
```

```
create table payments
(
   rollno    number(5) constraint payments_rollno_fk
                         references students(rollno),
   dp        date      constraint payments_dp_nn  not null,
   amount    number(5) constraint payments_amount_chk
                                 check ( amount > 0 ),
   constraint payments_pk primary key (rollno,dp)
);
```

## Getting information about tables

Data dictionary keeps track of the entire information about the database. It stores information about tables, constraints, procedures etc. Oracle provides a set of data dictionary views, which can be used to get information about these objects.

Data dictionary views are not actually tables instead they are relational views. However, at this stage you can treat data dictionary views tables for the time being.

We get list of tables from our schema using the followings:

```
select  * from tab;
```

It is also possible to get information about all constraints of all tables in your account by using:

```
select * from user_constraints;
```

If you want to get information about constraints of a single table, you can give the following to get names of constraints of BATCHES table.

CONSTRAINT_TYPE column of USER_CONSTRAINTS may contain any of the following characters.

| Character | Meaning |
|---|---|
| C | Check |
| U | Unique |
| R | Reference |
| P | Primary key |
| V | Check on view |

***Note:*** *The table name must be given in uppercase while searching based on table name as all object names (table is an object) are stored in uppercase in data dictionary.*

```
select constraint_name from user_constraints
where table_name = 'BATCHES';

CONSTRAINT_NAME
------------------------------
BATCHES_STDATE_NN
BATCHES_TIMING_CHK
BATCHES_DATES_CHK
BATCHES_PK
BATCHES_CCODE_FK
BACHES_FCODE_FK
```

## Inserting data into sample tables

The following is the script to insert a set of sample rows into all six tables.

```
rem ****script to insert sample data into table of oracle book*****
rem
rem    Author : P.Srikanth
rem    Date   : 4-aug-2001
rem     Place  : Vizag.
rem
rem *************************************************************************


rem --------- delete existing data from all tables  ------------------

delete from payments;
delete from students;
delete from batches;
delete from course_faculty;
delete from faculty;
delete from courses;



rem ------------------------ COURSES ----------------------------


insert into courses values('ora','Oracle database',25,4500,'Windows');

insert into courses values('vbnet','VB.NET',30,5500,'Windows and
programming');

insert into courses values('c','C programming',20,3500,'Computer Awareness');

insert into courses values('asp','ASP.NET',25,5000,'Internet and programming');

insert into courses values('java','Java Language',25,4500,'C language');

insert into courses values('xml','XML Programming', 15, 4000, 'HTML,Scripting,
ASP/JSP');



rem --------------------------- FACULTY ----------------------------


insert into faculty values('gk','George Koch','MS Computer Science','15 years
with databases');
```

```
insert into faculty values('da','Dan Appleman','CS and EE graduate',
'Extensively worked with COM');

insert into faculty values('hs','Herbert Schildt','MS Computer Science', 'Author
of several books');

insert into faculty values('dh','David Hunter','MS Electronics', 'Extensively
worked with Internet Tehnologees');

insert into faculty values('sw','Stephen Walther','Ph.D. in Philosophy',
'Extensively worked with Internet Tehnologees');

insert into faculty values('kl','Kevin Loney', 'MS Eletronics', 'Specialized in
Oracle DBA');

insert into faculty values('jj','Jamie Jaworski','Bachlors of Electrical'
,'Developed programs for US defense department');

insert into faculty values('jc','Jason Couchman','OCP DBA','Published articles
on Oracle');



rem ------------------------- COURSE_FACULTY ------------------------

insert into course_faculty values('gk','ora','A');

insert into course_faculty values('kl','ora','A');

insert into course_faculty values('jc','ora','A');

insert into course_faculty values('da','vbnet','A');

insert into course_faculty values('sw','asp','A');

insert into course_faculty values('da','asp','B');

insert into course_faculty values('hs','c','A');

insert into course_faculty values('dh','xml','A');

insert into course_faculty values('jj','java','A');

insert into course_faculty values('hs','java','B');

insert into course_faculty values('jj','c','A');

insert into course_faculty values('jj','vbnet','B');
```

```
rem ------------------------- BATCHES ---------------------------

insert into batches values('b1','ora','gk','12-jan-2001','20-feb-2001', 1);

insert into batches values('b2','asp','da','15-jan-2001','5-mar-2001', 2);
insert into batches values ('b3','c','hs','20-jan-2001','27-feb-2001',3);

insert into batches values ('b4','xml','dh','2-mar-2001','30-mar-2001', 3);

insert into batches values ('b5','java','hs','5-apr-2001','10-may-2001', 1);

insert into batches values ('b6','vbnet','da','12-july-2001',null,1);

insert into batches values ('b7','ora','jc','15-aug-2001',null,2);



rem ---------------------------- STUDENTS ----------------------------


insert into students values (1,'b1','George Micheal','m','10-jan-2001',
'488333','gm@yahoo.com');

insert into students values (2,'b1','Micheal Douglas','m','11-jan-2001',
'334333','md@hotmail.com');

insert into students values (3,'b2','Andy Roberts','m','11-jan-2001',
'433554','ar@yahoo.com');

insert into students values (4,'b2','Malcom Marshall','m','16-jan-2001',
'653345','mm@usa.net');

insert into students values (5,'b2','Vivan Richards','m','16-jan-2001',
'641238','vr@yahoo.com');

insert into students values (6,'b3','Chirs Evert','f','14-jan-2001',
null,'ce@yahoo.com');

insert into students values (7,'b3','Ivan Lendal','m','15-jan-2001',
'431212','il@hotmail.com');

insert into students values (8,'b4','George Micheal','m','1-mar-2001',
'488333','gm@hotmail.com');
```

```
insert into students values (9,'b5','Richard Marx','m','6-apr-2001',
'876567','rm@hotmail.com');

insert into students values (10,'b5','Tina Turner','f','6-apr-2001',
'565678','tinat@hotmail.com');

insert into students values (11,'b5','Jody Foster','f','7-apr-2001',
'234344','jody@hotmail.com');




rem ---------------------------- PAYMENTS --------------------------

insert into payments values (1,'10-jan-2001',4500);
insert into payments values (2,'11-jan-2001',3500);
insert into payments values (2,'17-jan-2001',1000);

insert into payments values (3,'13-jan-2001',2000);
insert into payments values (3,'20-jan-2001',3000);

insert into payments values (4,'16-jan-2001',3000);
insert into payments values (4,'30-jan-2001',2000);

insert into payments values (5,'16-jan-2001',5000);

insert into payments values (6,'14-jan-2001',3500);
insert into payments values (7,'15-jan-2001',3500);

insert into payments values (8,'1-mar-2001',2000);
insert into payments values (8,'2-mar-2001',2000);

insert into payments values (9,'7-apr-2001',3000);

insert into payments values (10,'10-apr-2001',4500);

insert into payments values (11,'7-apr-2001',1000);
insert into payments values (11,'10-apr-2001',3500);


commit;
```

## Summary

In this chapter we have seen how to use integrity constraints implement standard and business rules. Constraints can be related to a single column – column constraints, or related to multiple columns – table constraints.

We have understood six tables that we need to store information about courses, students and payments. It is very important to understand what is stored in each table and the relationship among six tables. One script is used to create tables and another script is used to insert sample data into it.

## Exercises

1. __________ constraint can be used to implements business rules.
2. __________ option of REFERENCES constraint is used to delete all child rows when parent row is being deleted.
3. Data dictionary view used to get information about constraints is _____________.
4. When a table has a composite primary key, where the PRIMARY KEY constraint is defined?__________
5. What is the relationship between COURSES and COURSE_FACULTY table?
6. How do you get details of all CHECK constraints of all tables?
7. Is it possible to create a constraint to prevent a data that is less than the system date?