# CONSTRSAINTS

- SQL allow us to define the ways in which we can automatically enforce the integrity of a database.

- Constraints define rules regarding permissible values allowed in columns and are the standard mechanism for enforcing integrity.

- Using constraints is preferred to using triggers, stored procedures, rules, and defaults, as a method of implementing data integrity rules.

## Types of Constraints:

There are a number of different ways to implement constraints, but each of them falls into one of these three categories: entity, domain, and referential integrity constraints.

**Domain Constraints:** A Domain constraint deals with one or more columns. It is important to ensure that a particular column or a set of columns meets particular criteria. When you insert or update a row, the constraint is applied without respect to any other row in the table. The focus is on the data that is in the column. These kinds of constraints will resurface when we deal with Check constraints, Default constraints and rules and defaults.

**Entity Constraints:** Entity constraints are all about individual rows. This constraint ignores the column as a whole and focuses on a particular row. This can be best exemplified by a constraint that requires every row to have a unique value for a column or a combination of columns. This is to ensure that for a particular row, the same value does not already exist in some other row. We'll see this kind of constraint in dealing with Primary key and Unique constraints.

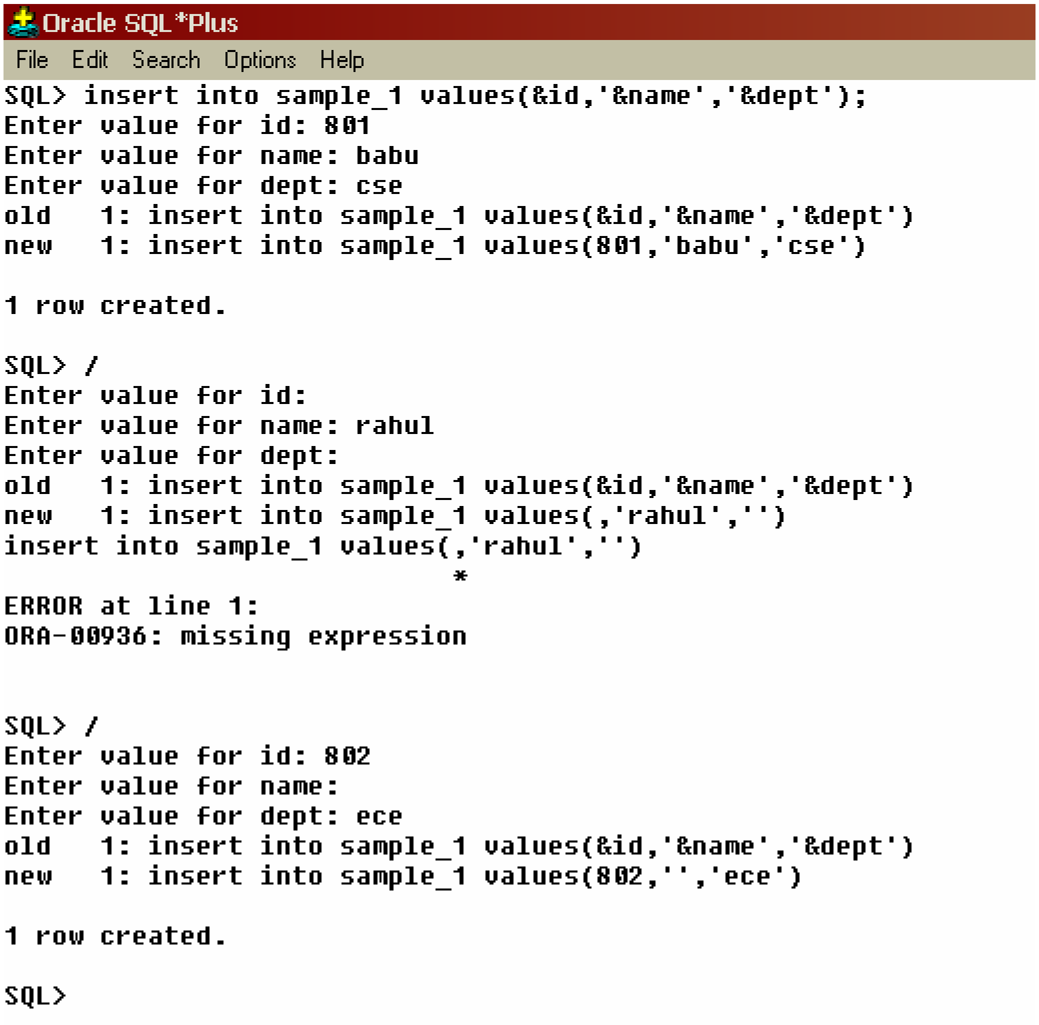
**Referential Integrity Constraints:** Referential integrity constraints are created when a value in one column must match the value in another column. It can either be in the same table or more typically, a different tableReferential integrity constraints allow us to build what we would call a domain table. A **domain table** is table whose sole purpose is to provide a limited list of acceptable values. For each row we insert into the referencing table, it will have a value that is in our domain list. We will see more of this when we learn about Foreign key constraints.

## Key constraints:

1) **NOT NULL**
2) **UNIQUE**
3) **CHECK**
4) **Primary Key**
5) **Foreign Key**

**NOT NULL:**

```
create table sample_1
( id int not null,
  name char(20),
  dept char(5) not null
 );
```



```
SQL> insert into sample_1 values(&id,'&name','&dept');
Enter value for id: 801
Enter value for name: babu
Enter value for dept: cse
old   1: insert into sample_1 values(&id,'&name','&dept')
new   1: insert into sample_1 values(801,'babu','cse')

1 row created.

SQL> /
Enter value for id:
Enter value for name: rahul
Enter value for dept:
old   1: insert into sample_1 values(&id,'&name','&dept')
new   1: insert into sample_1 values(,'rahul','')
insert into sample_1 values(,'rahul','')
                            *
ERROR at line 1:
ORA-00936: missing expression


SQL> /
Enter value for id: 802
Enter value for name:
Enter value for dept: ece
old   1: insert into sample_1 values(&id,'&name','&dept')
new   1: insert into sample_1 values(802,'','ece')

1 row created.

SQL>
```

By Dinesh

There are three cases shown in above pic.

1. Normal way of inserting into a table.
2. Leaving blank the constrained column/ attribute:
   By doing this you can see the row is not inserted.
3. Inserting row into the table by satisfying the constraints

## **UNIQUE:**

The UNIQUE constraint ensures that all values in a column are distinct.

Often, Unique constraints are referred to as ***Alternate Keys***. Alternate keys are not considered to be the unique identifier of a record in a table.

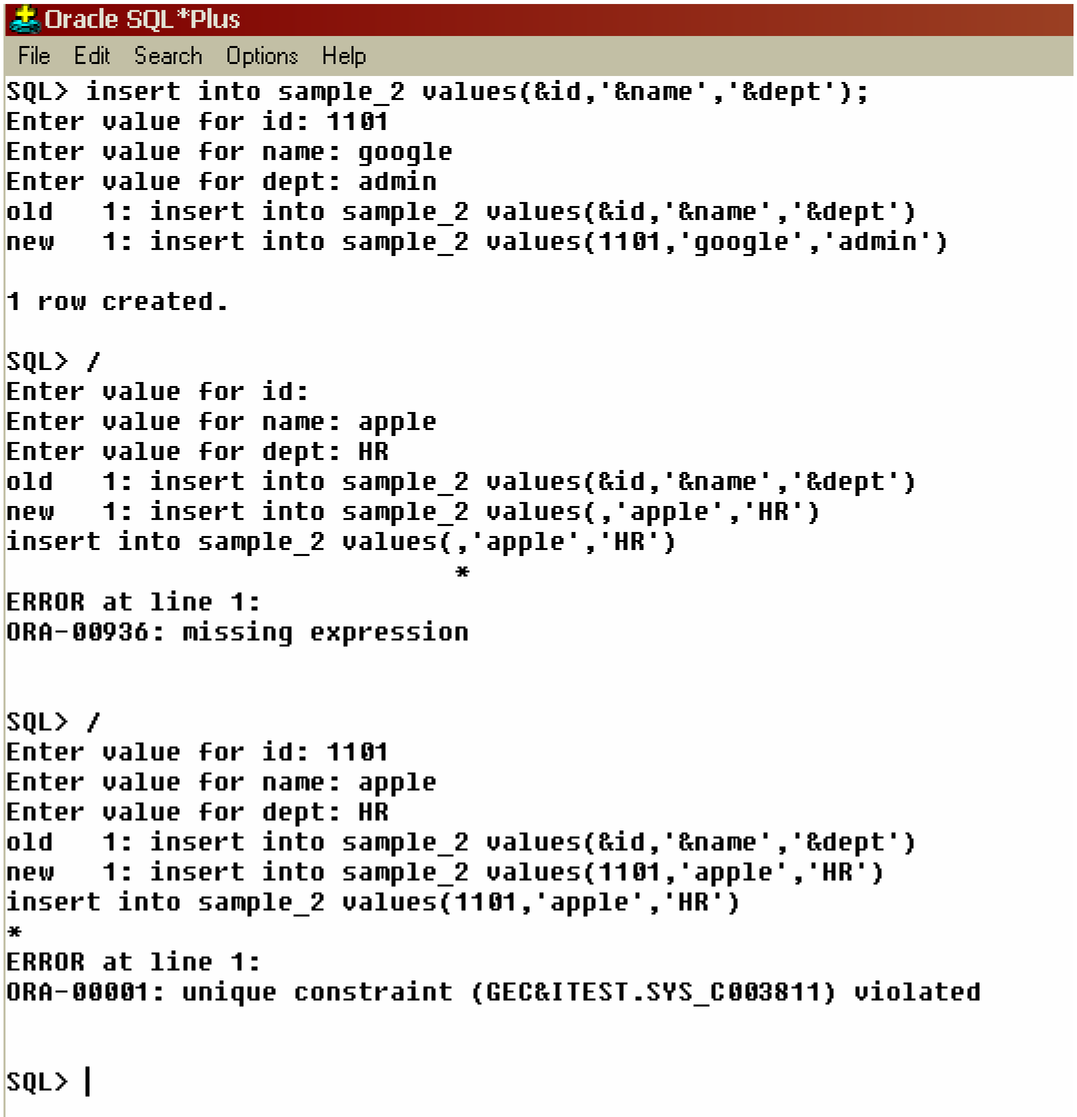
Once we establish a Unique constraint, every value in the named column must be unique. SQL Server will show an error if you try to update or insert a row with a value that already exists in a column with a Unique constraint.

A Unique key will not automatically prevent us from entering NULL values.

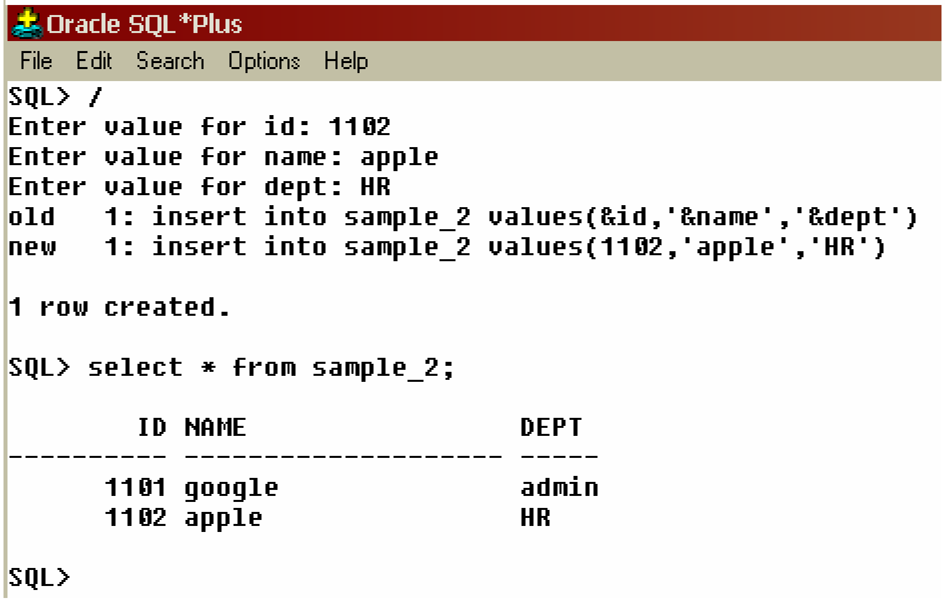
```
create table sample_2
( id int unique,
  name char(20),
  dept char(5)
);
```

In the Screenshot below I have tried to insert a null value for ID. But as per the unique constraint NULL values cannot be entered into the table for that attribute.

In other case I tried to insert same value for ID, as a result the row was not inserted.



```
SQL> insert into sample_2 values(&id,'&name','&dept');
Enter value for id: 1101
Enter value for name: google
Enter value for dept: admin
old   1: insert into sample_2 values(&id,'&name','&dept')
new   1: insert into sample_2 values(1101,'google','admin')

1 row created.

SQL> /
Enter value for id:
Enter value for name: apple
Enter value for dept: HR
old   1: insert into sample_2 values(&id,'&name','&dept')
new   1: insert into sample_2 values(,'apple','HR')
insert into sample_2 values(,'apple','HR')
                            *
ERROR at line 1:
ORA-00936: missing expression


SQL> /
Enter value for id: 1101
Enter value for name: apple
Enter value for dept: HR
old   1: insert into sample_2 values(&id,'&name','&dept')
new   1: insert into sample_2 values(1101,'apple','HR')
insert into sample_2 values(1101,'apple','HR')
*
ERROR at line 1:
ORA-00001: unique constraint (GEC&ITEST.SYS_C003811) violated


SQL> |
```

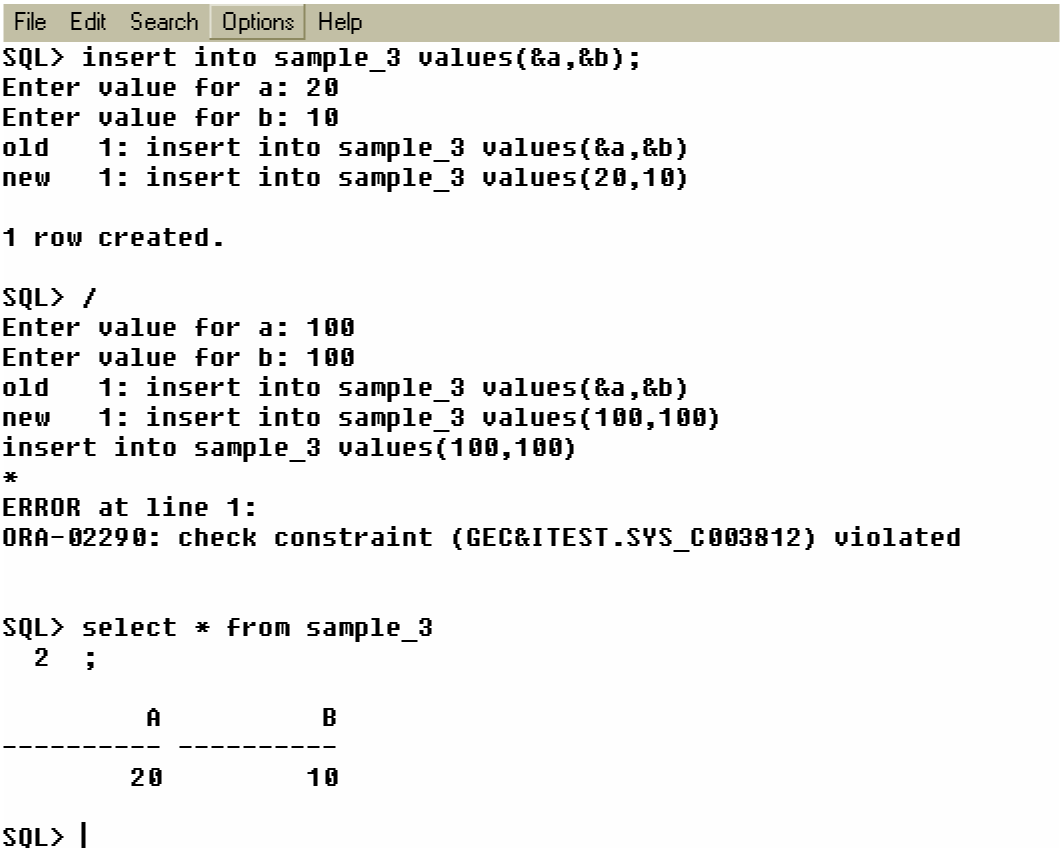


```
SQL> /
Enter value for id: 1102
Enter value for name: apple
Enter value for dept: HR
old   1: insert into sample_2 values(&id,'&name','&dept')
new   1: insert into sample_2 values(1102,'apple','HR')

1 row created.

SQL> select * from sample_2;

        ID NAME                 DEPT
---------- -------------------- -----
      1101 google               admin
      1102 apple                HR

SQL>
```

## CHECK:

The CHECK constraint ensures that all values in a column satisfy certain conditions.

```
create table sample_3
( a int Check (a between 1 and 50) ,
  b int
);
```

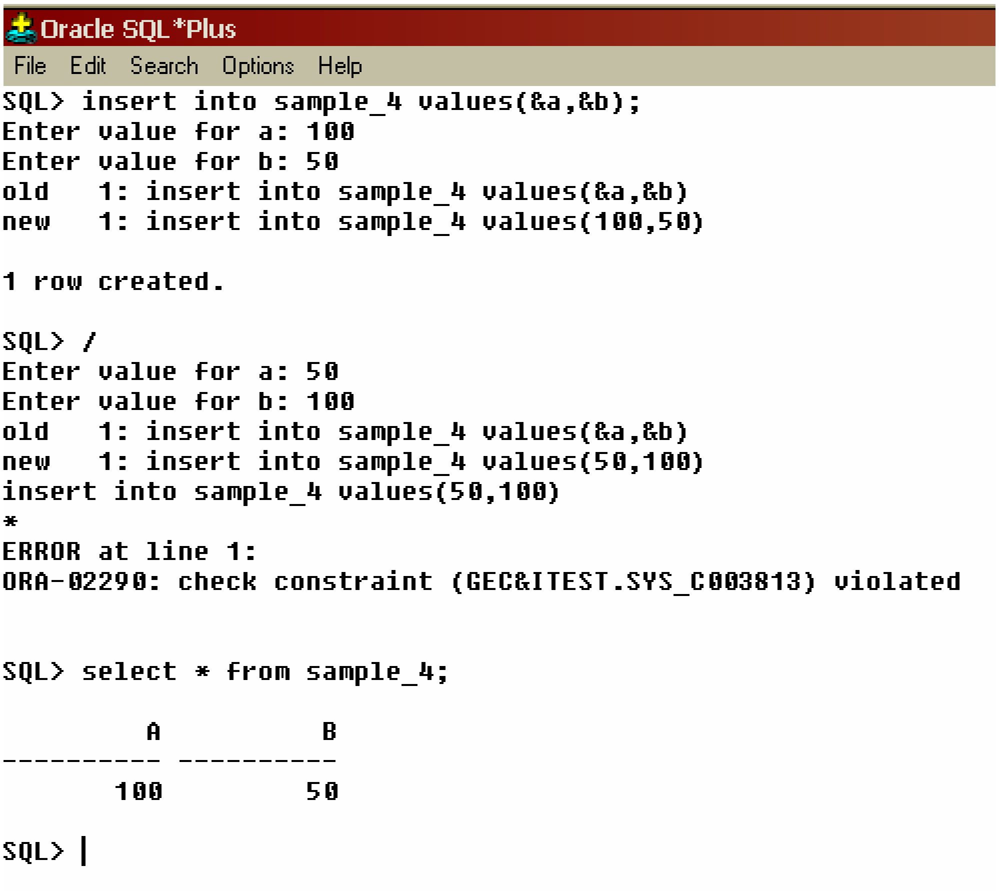


```
SQL> insert into sample_3 values(&a,&b);
Enter value for a: 20
Enter value for b: 10
old   1: insert into sample_3 values(&a,&b)
new   1: insert into sample_3 values(20,10)

1 row created.

SQL> /
Enter value for a: 100
Enter value for b: 100
old   1: insert into sample_3 values(&a,&b)
new   1: insert into sample_3 values(100,100)
insert into sample_3 values(100,100)
*
ERROR at line 1:
ORA-02290: check constraint (GEC&ITEST.SYS_C003812) violated


SQL> select * from sample_3
  2  ;

         A          B
---------- ----------
        20         10

SQL> 
```

**Check constraint that check relation between two columns:**

```
create table sample_4
( a int,
  b int,
  Check ( a >=b)
);
```

By Dinesh



```
SQL> insert into sample_4 values(&a,&b);
Enter value for a: 100
Enter value for b: 50
old   1: insert into sample_4 values(&a,&b)
new   1: insert into sample_4 values(100,50)

1 row created.

SQL> /
Enter value for a: 50
Enter value for b: 100
old   1: insert into sample_4 values(&a,&b)
new   1: insert into sample_4 values(50,100)
insert into sample_4 values(50,100)
*
ERROR at line 1:
ORA-02290: check constraint (GEC&ITEST.SYS_C003813) violated


SQL> select * from sample_4;

         A          B
---------- ----------
       100         50

SQL> |
```

**Constraint chect using UPPER fn:**

```
create table sample_4
( a int,
  b int,
  Check ( a >=b)
);
```

## DEFAULT:

Like all constraints, Default constraints become an integral part of the table definition.

It defines what to do when a row is inserted with data for the column on which you have not determined a Default constraint.

```
CREATE TABLE SAMPLE_6
( id int unique,
  name char(20),
  dept char(20) default 'not allocated'
);
```

By Dinesh

## PRIMARY KEY:

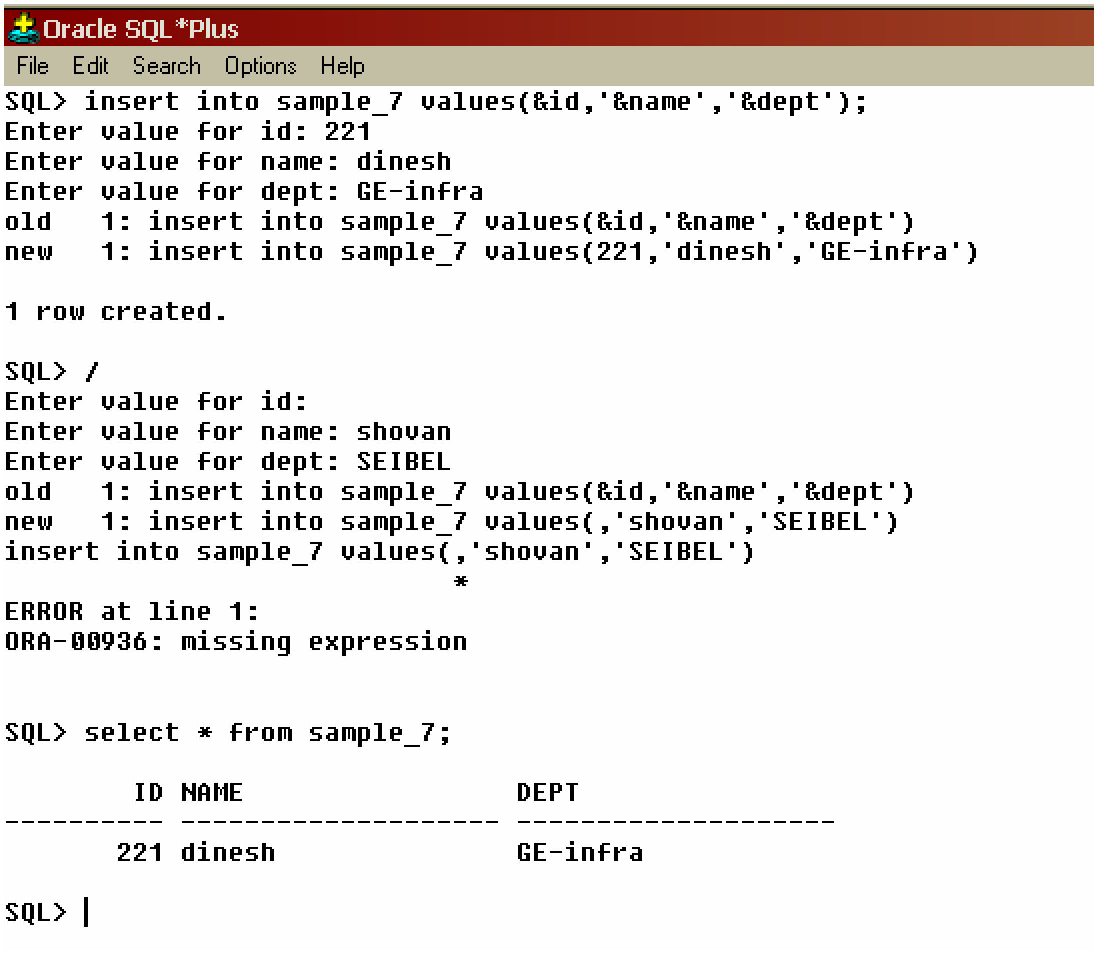
- A primary key is used to uniquely identify each row in a table.
- A primary key can consist of one or more fields on a table.
- When multiple fields are used as a primary key, they are called a **composite key**.
- Primary keys can be specified either when the table is created (using **CREATE TABLE**) or by changing the existing table structure (using **ALTER TABLE**).

```
CREATE TABLE sample_6
( id int PRIMARY KEY ,
  name char(20),
  dept char(20)
);
```

**(or)**

```
CREATE TABLE sample_6
( id int ,
  name char(20),
  dept char(20),
  CONSTRAINT id_pk PRIMARY KEY(id)
);
```

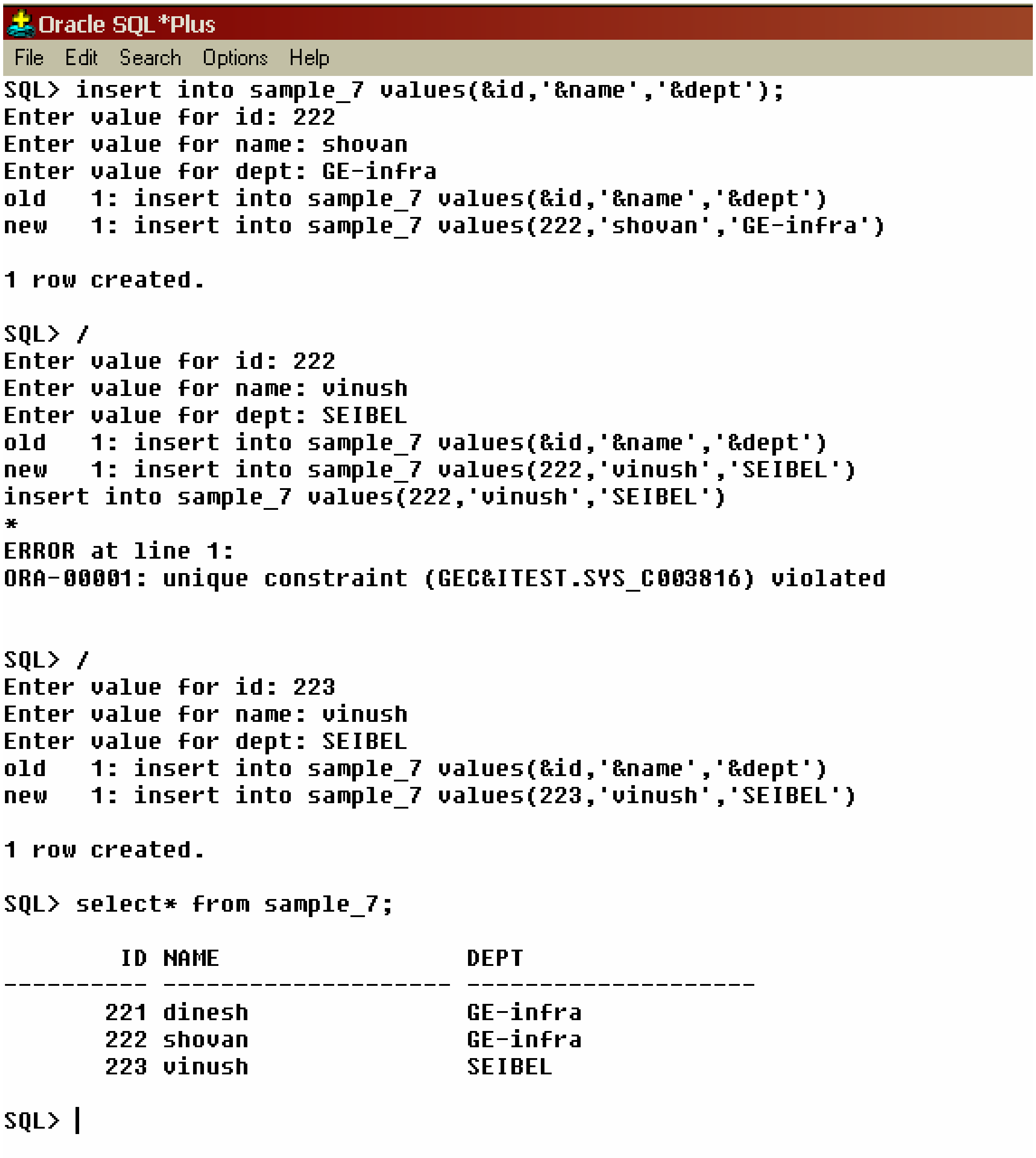
**NOTE: Primary key includes some constraints within itself like, NOT NULL, UNIQUE.**



```
SQL> insert into sample_7 values(&id,'&name','&dept');
Enter value for id: 221
Enter value for name: dinesh
Enter value for dept: GE-infra
old   1: insert into sample_7 values(&id,'&name','&dept')
new   1: insert into sample_7 values(221,'dinesh','GE-infra')

1 row created.

SQL> /
Enter value for id:
Enter value for name: shovan
Enter value for dept: SEIBEL
old   1: insert into sample_7 values(&id,'&name','&dept')
new   1: insert into sample_7 values(,'shovan','SEIBEL')
insert into sample_7 values(,'shovan','SEIBEL')
                            *
ERROR at line 1:
ORA-00936: missing expression


SQL> select * from sample_7;

        ID NAME                 DEPT
---------- -------------------- --------------------
       221 dinesh               GE-infra

SQL> 
```

By Dinesh

In the above case Primary constraint satisfy **NOT NULL** property.



```
SQL> insert into sample_7 values(&id,'&name','&dept');
Enter value for id: 222
Enter value for name: shovan
Enter value for dept: GE-infra
old   1: insert into sample_7 values(&id,'&name','&dept')
new   1: insert into sample_7 values(222,'shovan','GE-infra')

1 row created.

SQL> /
Enter value for id: 222
Enter value for name: vinush
Enter value for dept: SEIBEL
old   1: insert into sample_7 values(&id,'&name','&dept')
new   1: insert into sample_7 values(222,'vinush','SEIBEL')
insert into sample_7 values(222,'vinush','SEIBEL')
*
ERROR at line 1:
ORA-00001: unique constraint (GEC&ITEST.SYS_C003816) violated


SQL> /
Enter value for id: 223
Enter value for name: vinush
Enter value for dept: SEIBEL
old   1: insert into sample_7 values(&id,'&name','&dept')
new   1: insert into sample_7 values(223,'vinush','SEIBEL')

1 row created.

SQL> select* from sample_7;

        ID NAME                 DEPT
---------- -------------------- --------------------
       221 dinesh               GE-infra
       222 shovan               GE-infra
       223 vinush               SEIBEL

SQL> |
```

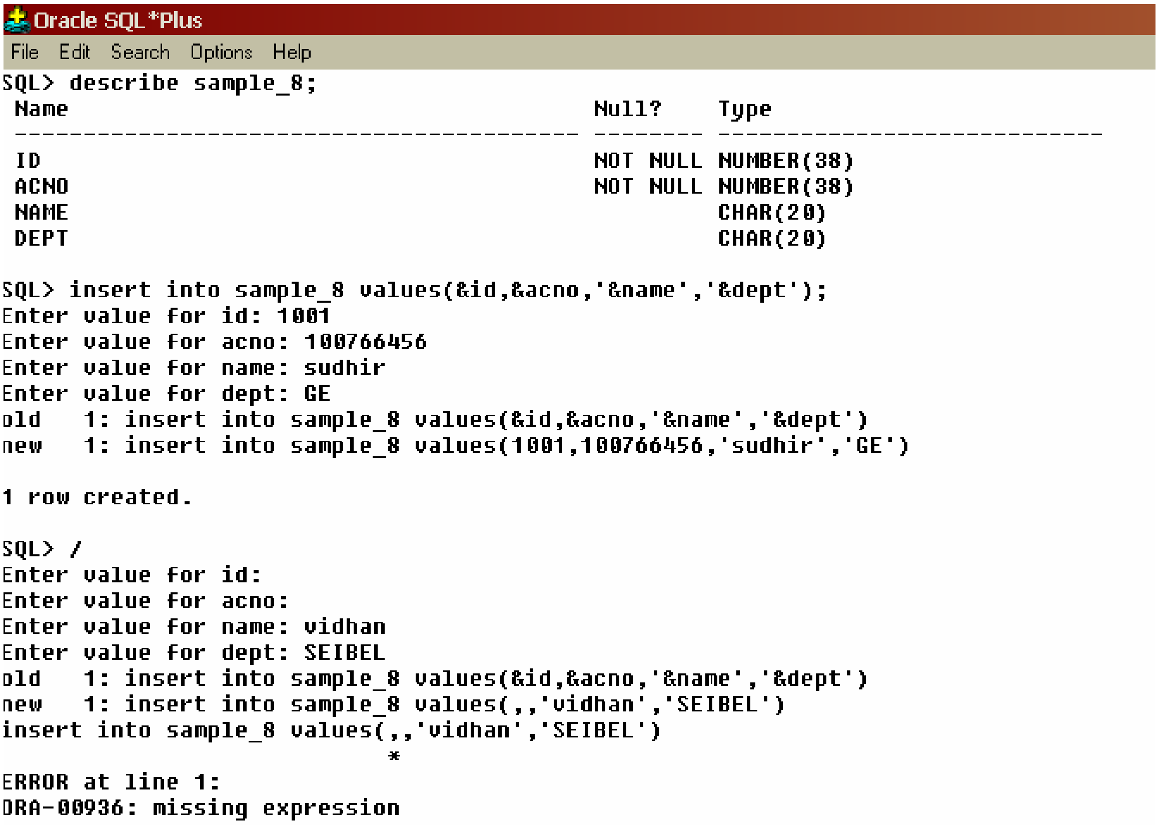
In the above case Primary Key satisfies the **UNIQUE** property.

## Creating PRIMARY KEY with more than one field/attribute:(composite keys)

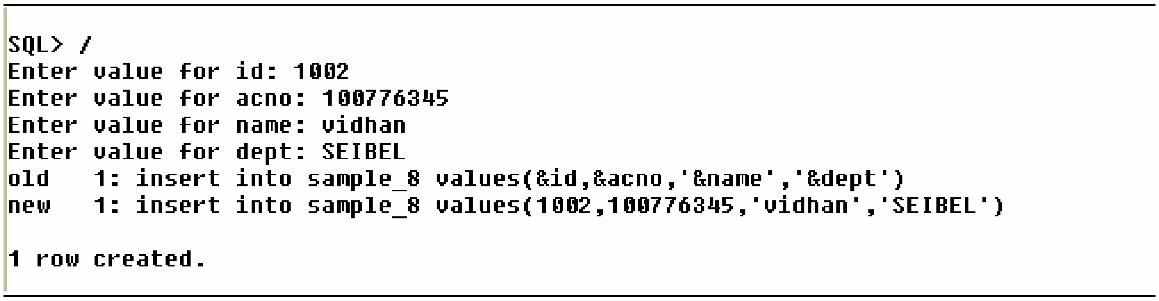
**Note we cannot have multiple PRIMARY KEY inside the sql table.**

```
CREATE TABLE sample_8
( id int,
  acno int,
  name char(20),
  dept char(20),
  CONSTRAINT pri_no_key PRIMARY KEY(id,acno)
);
```

Here “id’ & “acno” are two primary key for the table sample_8.

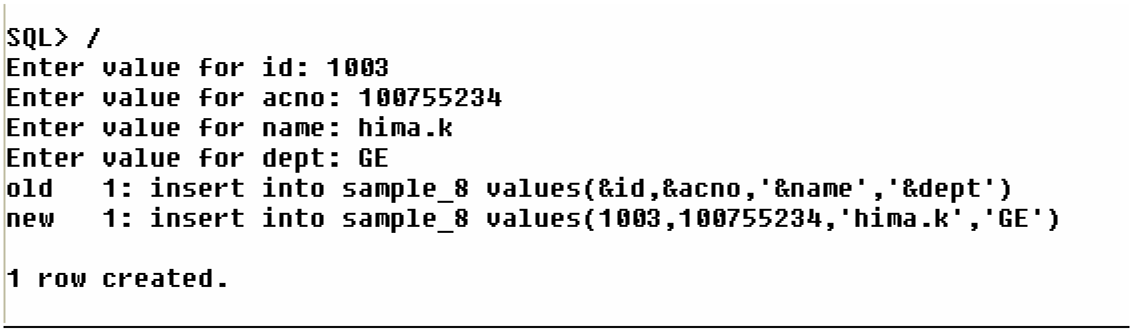


```
SQL> describe sample_8;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 ACNO                                      NOT NULL NUMBER(38)
 NAME                                               CHAR(20)
 DEPT                                               CHAR(20)

SQL> insert into sample_8 values(&id,&acno,'&name','&dept');
Enter value for id: 1001
Enter value for acno: 100766456
Enter value for name: sudhir
Enter value for dept: GE
old   1: insert into sample_8 values(&id,&acno,'&name','&dept')
new   1: insert into sample_8 values(1001,100766456,'sudhir','GE')

1 row created.

SQL> /
Enter value for id:
Enter value for acno:
Enter value for name: vidhan
Enter value for dept: SEIBEL
old   1: insert into sample_8 values(&id,&acno,'&name','&dept')
new   1: insert into sample_8 values(,,'vidhan','SEIBEL')
insert into sample_8 values(,,'vidhan','SEIBEL')
                            *
ERROR at line 1:
ORA-00936: missing expression
```

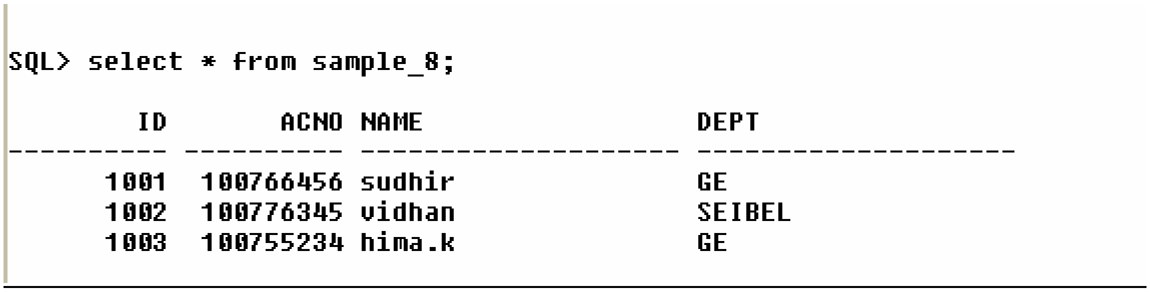


```
SQL> /
Enter value for id: 1002
Enter value for acno: 100776345
Enter value for name: vidhan
Enter value for dept: SEIBEL
old   1: insert into sample_8 values(&id,&acno,'&name','&dept')
new   1: insert into sample_8 values(1002,100776345,'vidhan','SEIBEL')

1 row created.
```

By Dinesh



```
SQL> /
Enter value for id: 1003
Enter value for acno: 100755234
Enter value for name: hima.k
Enter value for dept: GE
old   1: insert into sample_8 values(&id,&acno,'&name','&dept')
new   1: insert into sample_8 values(1003,100755234,'hima.k','GE')

1 row created.
```



```
SQL> select * from sample_8;

        ID       ACNO NAME                 DEPT
---------- ---------- -------------------- --------------------
      1001  100766456 sudhir               GE
      1002  100776345 vidhan               SEIBEL
      1003  100755234 hima.k               GE
```

## Disable / Enable PRIMARY KEY:

**Syntax:**

ALTER TABLE **table_name ENABLE CONSTRAINT Constraint_name.**

ALTER TABLE **table_name DISABLE CONSTRAINT Constraint_name.**

**NOTE:**

**If you Drop the table which contains primary key, the primary key will also be dropped.**

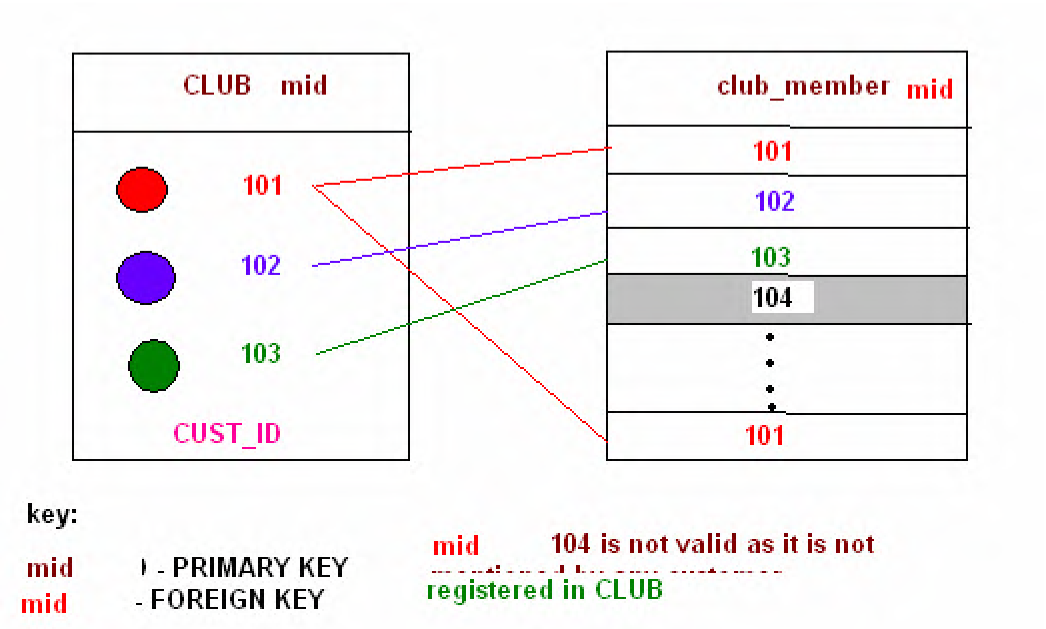
## Naming PRIMARY KEY:

**CREATE TABLE sample_10**<br>
**( id int,**<br>
**Name varchar2(20),**<br>
**CONSTRAINT sample_10_id_key PRIMARY KEY(id)**<br>
**)**

## FOREIGN KEY:

A foreign key is a field (or fields) that points to the primary key of another table.

The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted.



Here is a sample program for implementing foreign key:

**Parent table:**

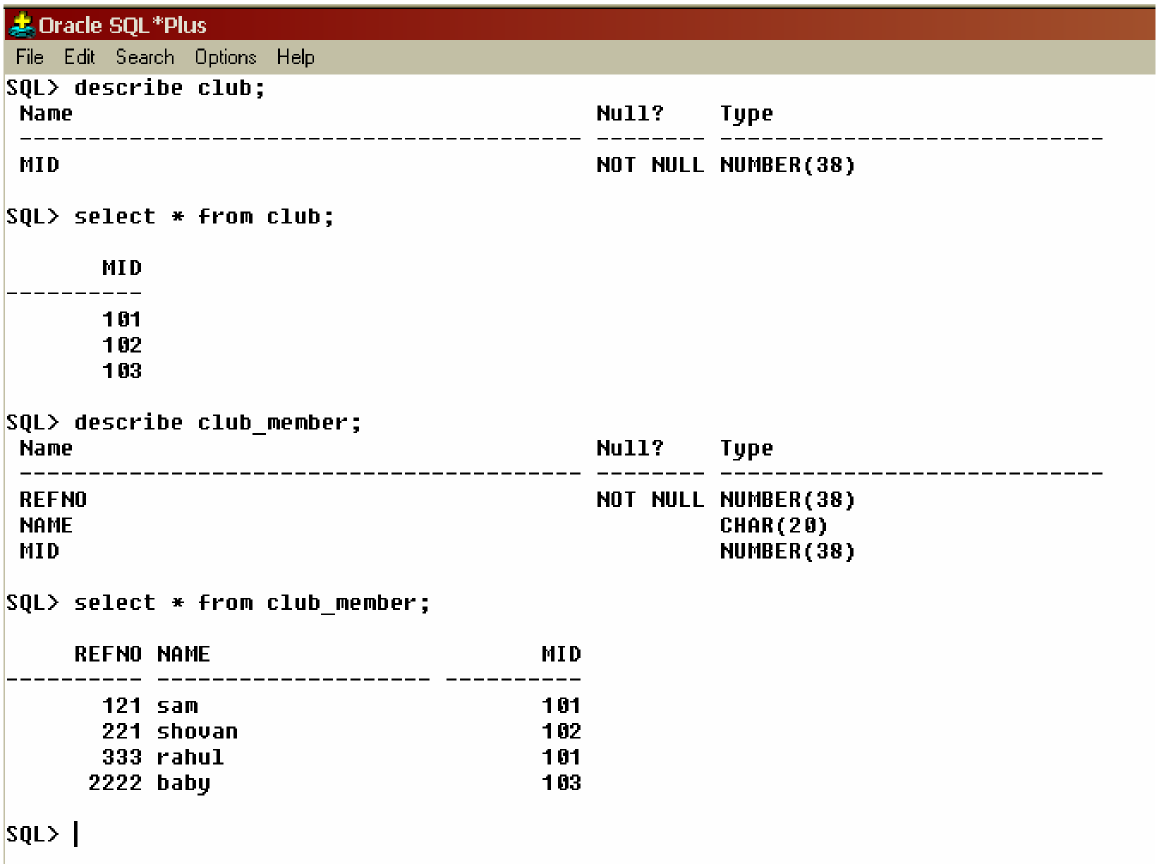
```
create table club
( mid int PRIMARY KEY

);
```

**Child table:**

```
create table club_member
( refno int primary key,
  name char(20),
  mid int references club(mid)
);
```

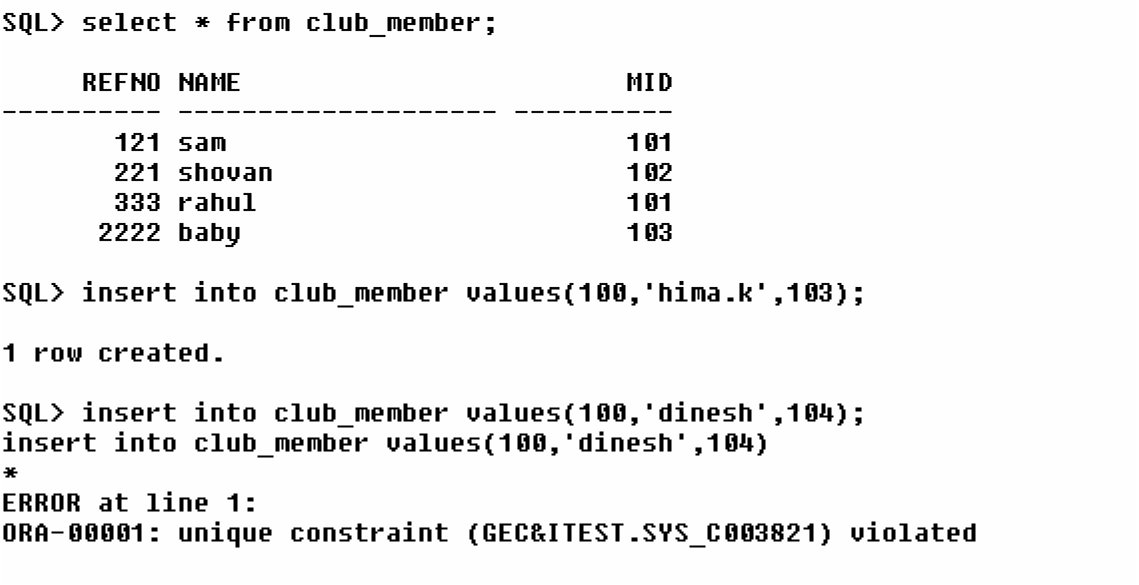
**(or)**

```
create table club_member
( refno int primary key,
  name char(20),
  mid int,
  CONSTRAINT mid_fk FOREIGN KEY(mid) REFERENCES club(mid)
);
```



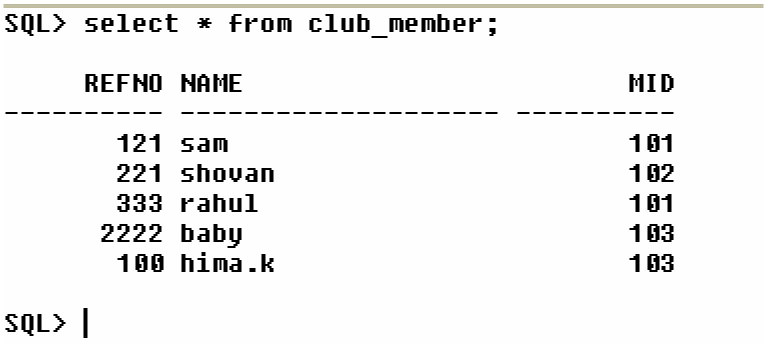
```
SQL> describe club;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MID                                       NOT NULL NUMBER(38)

SQL> select * from club;

       MID
----------
       101
       102
       103

SQL> describe club_member;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REFNO                                     NOT NULL NUMBER(38)
 NAME                                               CHAR(20)
 MID                                                NUMBER(38)

SQL> select * from club_member;

     REFNO NAME                        MID
---------- -------------------- ----------
       121 sam                         101
       221 shovan                      102
       333 rahul                       101
      2222 baby                        103

SQL> |
```

When trying to insert member whose id is not defined in table **"CLUB".**



```
SQL> select * from club_member;

     REFNO NAME                        MID
---------- -------------------- ----------
       121 sam                         101
       221 shovan                      102
       333 rahul                       101
      2222 baby                        103

SQL> insert into club_member values(100,'hima.k',103);

1 row created.

SQL> insert into club_member values(100,'dinesh',104);
insert into club_member values(100,'dinesh',104)
*
ERROR at line 1:
ORA-00001: unique constraint (GEC&ITEST.SYS_C003821) violated
```

By Dinesh



```
SQL> select * from club_member;

     REFNO NAME                        MID
---------- -------------------- ----------
       121 sam                         101
       221 shovan                      102
       333 rahul                       101
      2222 baby                        103
       100 hima.k                      103

SQL> |
```

**<u>Drop FOREIGN KEY</u>:**

```
ALTER TABLE table_name drop CONSTRAINT Constraint_name
```

## <u>CASCADING CONSTRAINTS:</u>

- The CASCADE CONSTRAINTS clause is used along with the DROP COLUMN clause.
- The CASCADE CONSTRAINTS clause drops all referential integrity constraints that refer to the primary and unique keys defined on the dropped columns.
- The CASCADE CONSTRAINTS clause also drops all multicolumn constraints defined on the dropped columns.

Eg:

```
Alter table club_member DROP (PK) CASCADE CONSTRAINT

Alter table club_member DROP (PK, FK, Column_1) CASCADE CONSTRAINT
```

## <u>VIEWING CONSTRAINTS:</u>

```
SELECT constraint_name, constraint_type, search_condition
FROM user_constraints
WHERE table_name = 'CLUB_MEMBERS';
```

Viewing column associated with constraint:

```
SELECT constraint_name, column_name
FROM user_cons_columns
WHERE table_name = 'CLUB_MEMBERS';
```

By Dinesh