Ex. No 14

Creation of Triggers, Procedures and Functions

AIM: Design and implement triggers and cursors

Trigger
A trigger is a procedure that is automatically invoked by the DBMS in response to specified changes to the database, and is typically specified by the DBA.. A database that has a set of associated triggers is called an Active Database.

A trigger description contains three parts:
Event : A change to the database that activates the trigger.
Condition : A query or test that is run when the trigger is activated.
Action : A procedure that is executed when the trigger is activated and its condition is true.
An insert, delete, or update statement could activate a trigger, regardless of which user or application invoked the activating statement; users may not even be aware that a trigger was executed as a side effect of their program.

Sample 1.

Create a table Account with fields id, name and balance. Primary key = id

1. Write a **MY SQL** program in Workbench to create a database trigger to check while insert a new row, if the balance should not go less than zero if so generate necessary notification.

   Ans:

```
create table Account(id int primary key, name varchar(20), bal int);
DELIMITER //
create trigger trigger1 before insert on Account
for each row
begin
if (new.bal < 0 ) then
        signal sqlstate '45000' set message_text = 'Error..! Transaction Not
Allowed Balance  ';
end if;
end //
DELIMITER ;
insert into Account values(100, 'Alice', 200);
insert into Account values(101, 'Bob', 500);
select * from Account;
/* Test query to validate Trigger */
insert into Account values(102, 'Cindy', -500);
```

2. Write a **PL/ SQL** program in Workbench to create a database trigger to check while insert a new row, if the balance should not go less than zero if so generate necessary notification.

```
drop table Account;
create table Account(id int primary key, name varchar(20), bal int);
CREATE  TRIGGER trigger2
BEFORE INSERT ON Account
FOR EACH ROW
DECLARE
 min_bal integer := 0;
BEGIN
IF (:new.bal < min_bal ) THEN
   dbms_output.put_line('Error..! Balance cannot be less than zero');
   raise_application_error(-20000, 'Balance cannot be less than zero');
END IF;
END;
insert into Account values(100, 'Alice', 200);
insert into Account values(101, 'Bob', 500);
select * from Account;
/* Test query to validate Trigger */
insert into Account values(102, 'Cindy', -500);
/
```

Sample 2.

Create a table Student with fields Rollno, name, age and address. Primary key = Rollno
Write a PL/SQL program to create a trigger before insert for each row and NOT allowing transaction on weekends.

```
create table Student(Rollno int primary key, name varchar(20), address varchar(50));
insert into Student values(104, 'Bob', 'Address1');
CREATE OR REPLACE TRIGGER trigger3
BEFORE INSERT ON Student
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
BEGIN
  IF TO_CHAR(SYSDATE, 'D') <> '7' THEN
```

```
    dbms_output.put_line('Error..! Cannot insert record on weekdays');
    RAISE_APPLICATION_ERROR(-20000, 'Cannot insert record on weekdays');
  END IF;
END;
/* Test query to validate Trigger */
insert into Student values(103, 'Alice', 'Address2');
```

Sample 3:

A Library database contain the following tables.

Book_avail (bookid, title, no_of _copies, price)
Student (st_id,name,class,fine)
Issue_tab (st_id, book_id, issuedate, returndate)

Fill tables with below Data

```
create table Book_avail(bookid int primary key, title varchar(20), no_of_copies int, price int);

create table Student(st_id int primary key, name varchar(20), class varchar(10), fine int);

create table Issue_tab(st_id int, book_id int, issue_date date , return_date date, primary key(st_id, book_id));

insert into Student values(100, 'Alice', 'CSE', 0);

insert into Student values(101, 'Bob', 'CSE', 0);

insert into Book_avail values(1, 'Data Structure', 10, 1000);

insert into Book_avail values(2, 'Java - Complete ref', 10, 1000);

insert into Issue_tab values(100, 1, TO_DATE('2022/01/01','%yyyy-%mm-%dd'), TO_DATE('2022/02/01','yyyy-%mm-%dd'));

insert into Issue_tab values(101, 2, TO_DATE('2022/01/01','%yyyy-%mm-%dd'), TO_DATE('2022/03/01','yyyy-%mm-%dd'));
```

Question 1

Create a database trigger in PL/SQL to calculate the fine based on the rules given below.

After 1 month 5% of price
After 2 month 10% of price
After 3 month 20% of price.

Sample 4:

Create a Table

Customer(id int primary key, name varchar(20), age int, address varchar(30), sal int);

Fill tables with below Data

create table Customer(id int primary key, name varchar(20), age int, address varchar(30), sal int);

insert into Customer values(1, 'Alice', 20, 'Address1', 10000);

insert into Customer values(2, 'Bob', 30, 'Address2', 20000);

insert into Customer values(3, 'Cindy', 40, 'Address3', 30000);

insert into Customer values(4, 'Sam', 50, 'Address4', 40000);

insert into Customer values(5, 'Eric', 60, 'Address5', 50000);

Question 1

Create a trigger to display the salary difference between new and old values before - insert, delete or update of values on the table