47_ YASH SARANG EXPERIMENT 9 DBMS

Aim: Triggers / Functions and Procedure.

Theory:

TRIGGER:-

① A trigger is a piece of code that is run before or after a database table is modified.

② A trigger can be defined on:
- Tables/Views : DML (insert/update/delete)
- Schema : DDL Trigger and logon/logoff.
- Database : System event triggers (startup/shutdown)

③ A trigger is used to
- prevent invalid transaction.
- keep an audit trail of table.
- ensure rollback if database is inconsistent.

④ The different parts of a trigger are trigger statements, trigger restriction and action.

⑤ A row trigger is fired each time a row in the table is affected. It uses 'for each row' clause.

⑥ Statement triggers are fired once on behalf of the statement, independant of the number of rows the trigger statement affects.

⑦ Syntax for creating a trigger is as follows:

CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE / AFTER / INSTEAD OF} → specifies when executed.
{INSERT / UPDATE / DELETE} → specifies DML operation.
[of col_name] → specifies col_name that will be updated.

PROCEDURE & FUNCTIONS.

1) PL/SQL allows writing subprograms made up of logically grouped SQL and PL/SQL statements.

2) If we want to update data on table, a procedure is preffered while for retrieving information, a function is preffered.

3) A select SQL query can call a function but cannot call a procedure.

4) Procedure may return one or more values through parameters or may not return values at all. A function always returns a value using the return statement.

SQL QUERIES:

1. Create a trigger that fires before inserting or delete of a row in the emp table and displays the count of rows.

```
SQL> create trigger t
  2  before insert or delete on employee
  3  declare
  4  val number;
  5  begin
  6  select count(*) into val from employee;
  7  dbms_output.put_line('Number of rows are: ' || val );
  8  end;
  9  /

Trigger created.
```

```
SQL> insert into employee values(421,'Harish',5000,101,'Clerk');
Number of rows are: 17

1 row created.
```

2. Create a trigger that stops the user from entering Dept no in emp table if that dept no doesn't exist in dept table. The trigger should display the contents of dept table.

```
SQL> create trigger checker2
  2  before insert on employee
  3  for each row
  4  declare
  5  val number;
  6  begin
  7  select count(*) into val from department d where d.department_no = :new.department_no;
  8  if val = 0 then
  9  dbms_output.put_line('Department does not exist');
 10  else
 11  dbms_output.put_line('Value Inserted');
 12  end if;
 13  end;
 14  /

Trigger created.
```

```
SQL> insert into employee values(219,'Lata',7000,999,'Artist');
Number of rows are: 19
Department does not exist
```

```
SQL> insert into employee values(219,'Lata',7000,401,'Artist');
Number of rows are: 19
Value Inserted
```

3. Write a procedure that: Accepts department number and percentage of raise in sal Updates the sal of all those employees under that department

```
SQL> create procedure bonus2( deptno in number, inc in float)
  2  as
  3  begin
  4  update employee set salary = salary + salary*inc where department_no = deptno;
  5  dbms_output.put_line('Values Updated');
  6  end;
  7  /

Procedure created.
```

```
SQL> EXEC bonus2(201,0.15);
Values Updated

PL/SQL procedure successfully completed.

SQL> select * from employee where department_no = 201;

EMPLOYEE_ID EMPLOYEE_NAME            SALARY DEPARTMENT_NO JOB
----------- -------------------- ---------- ------------- --------------------
        211 John                      69000           201
        212 Shweta                     5750           201
        213 Amit                      49450           201
```

4. Write a function that accepts: Dept no and returns the total sal of all employees in that department.

```
SQL> create function dept (deptno in number)
  2  return number is totalsum employee.salary%type;
  3  begin
  4  select sum(salary) into totalsum from employee where department_no = deptno;
  5  return(totalsum);
  6  end;
  7  /

Function created.
```

```
SQL> declare
  2  answer number;
  3  deptno number;
  4  begin
  5  deptno := 101;
  6  answer := dept(deptno);
  7  dbms_output.put_line('department: '|| deptno || ' Sum is ' || answer);
  8  end;
  9  /
department: 101 Sum is 165290

PL/SQL procedure successfully completed.
```

Conclusion: Triggers / Functions and procedures have been implemented.