Database Systems LAB # 14

Procedures, Functions and Triggers in Oracle RDBMS

Procedure & Function:

A procedure is a block that can take parameters (sometimes referred to as arguments) and be invoked. Procedures promote reusability and maintainability.

Difference:

Procedure:
No return.
PROCEDURE name as

Function:
Returns a value
FUNCTION name RETURN data-type as

Syntax for procedure: Create PROCEDURE procedur_name (parameter1 in datatype1, parameter2 in datatype2,…)
IS|AS
PL/SQL Block;

Examples:

```
Create PROCEDURE leave_emp (v_id IN emp.empno%TYPE) as
BEGIN
DELETE from emp WHERE empno=v_id;
END;
/

CREATE procedure print_nme(Fname in t1.name%TYPE) as Cname t1.name%TYPE;
BEGIN
select t1.name into Cname from t1 where t1.age=20;
end;
/
```

Call a procedure: exec print_nme('SA');

Syntax for function: Create function function_name (parameter1 in datatype1, parameter2 in datatype2,…) return type
IS|AS
PL/SQL Block;

Examples:

```
CREATE or replace function print_id(Fname in t1.name%TYPE) return t1.name%TYPE as
Cname t1.name%TYPE;
BEGIN
select t1.name into Cname from t1 where t1.age=20;
return(Cname);
end;
/

declare
name varchar(2);
begin
name:=fun('CA');
dbms_output.put_line(name);
end;
/
```

Database Triggers:

Database trigger is a PL/SQL block that is executed on an event in the database. The event is related to a particular data manipulation of a table such as inserting, deleting or updating a row of a table.

Triggers may be used for any of the following:

To implement complex business rule, which cannot be implemented using integrity constraints.
To automatically perform an action when another concerned action takes place. For example, updating a table whenever there is an insertion or a row into another table.

Sample Table to be Triggered

```
CREATE TABLE PERSON (
ID INT,
NAME VARCHAR(30),
DOB DATE,
PRIMARY KEY(ID)
);
```

1. Before Insert Trigger

```
CREATE TRIGGER PERSON_INSERT_BEFORE
BEFORE INSERT ON PERSON
FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE ('BEFORE INSERT OF'||:NEW.NAME);
END;
/
```

2. After Insert Trigger

```
CREATE TRIGGER PERSON_INSERT_AFTER
AFTER INSERT ON PERSON
FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE ('AFTER INSERT OF'||:NEW.NAME);
END;
/
```

3. Before Update Statement Trigger

```
CREATE TRIGGER PERSON_UPDATE_S_BEFORE
BEFORE UPDATE
ON PERSON
BEGIN
DBMS_OUTPUT.PUT_LINE ('BEFORE UPDATING SOME PERSON(S)');
END; /
```

4. Dropping Triggers

```
DROP TRIGGER trigger_name;
```