PLSQL PACKAGE & TRIGGER

BY- N K SHUKLA

TRIGGER

- Trigger is invoked by Oracle engine automatically whenever a specified event occurs. Trigger is stored into database and invoked repeatedly, when specific condition match.
- Triggers are stored programs, which are automatically executed or fired when some event occurs.
- Triggers are written to be executed in response to any of the following events.
  - A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
  - A database definition (DDL) statement (CREATE, ALTER, or DROP).
  - A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).
- Triggers could be defined on the table, view, schema, or database with which the event is associated.

TRIGGER Advantages:

- ❑Trigger generates some derived column values automatically
- ❑Enforces referential integrity
- ❑Event logging and storing information on table access
- ❑Auditing
- ❑Synchronous replication of tables
- ❑Imposing security authorizations
- ❑Preventing invalid transactions

TRIGGER Types:

❑A trigger's type is defined by the type of triggering transaction and by the level at which the trigger is executed. Oracle has the following types of triggers depending on the different applications.

- Row Level Triggers
- Statement Level Triggers
- Before Triggers
- After Triggers

❖**Dropping Triggers:**

❑Triggers may be dropped via the drop trigger command. In order to drop a trigger, you must either own the trigger or have the DROP ANY TRIGGER system privilege.

❑Syntax:

➢DROP TRIGGER trigger_name;

Trigger Syntax:

Function BODY:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Program 1:Create a trigger that prevents any changes to the salary column of the employees table for employees whose job title is President.

```
create or replace trigger chk_sal before update of salary on emp for
EACH ROW when (new.job_t = 'precident’)
DECLARE

BEGIN

if :new.job_t='precident' then
 RAISE_APPLICATION_ERROR(-20501,'You cannot update SALARY of
Precident');
end if;

END;
```

Program 1: trigger invoked automatically when u edit salary of precident

Employee E01 is president if you update his salary trigger get invoked.

Autocommit Display 10 Save Run

```
update emp set salary=78000 where eid='E01';
```

Results Explain Describe Saved SQL History

```
ORA-20501: You cannot update SALARY of Precident
ORA-06512: at "SYSTEM.CHK_SAL", line 5
ORA-04088: error during execution of trigger 'SYSTEM.CHK_SAL'
```

Program 2:Create a trigger that automatically updates the salary_grade column of the employees table based on the employee's salary.

```
create or replace trigger UP_GRADE after update of salary on emp
DECLARE

BEGIN

UPDATE EMP SET SAL_GRADE='PB1' WHERE SALARY>80000;

END;
```

Program 2:Create a trigger that automatically updates the salary_grade column of the employees table based on the employee's salary.

Autocommit Display 10 Save Run

```
update emp set salary=98888 where eid='E02';
select * from emp;
```

Results Explain Describe Saved SQL History

EID	ENAME	SALARY	DNAME	DID	JOB_T	SAL_GRADE
E01	Ajay Patel	78000	Computer	CE	precident	-
E02	jay Patel	98888	Computer	CE	mngr	PB1
E03	Amit Shah	75000	Electronics	EC	mngr	-
E04	Mit Shakariya	78000	Electronics	EC	precident	-
E05	Hamit Shakhiya	50000	Instrumental	IC	mngr	-
E07	ada	5222	w	w	w	w
E06	disha shah	222222	computer	ce	prog	PB1

7 rows returned in 0.00 seconds CSV Export

PACKAGE

- package is a container for other database objects.
- A package can hold other database objects such as variables, constants, cursors, exceptions,
- procedures, functions and sub-programs.
- A package has usually two components, a specification and a body.
- A package's specification declares the types (variables of the record type), memory
- variables, constants, exceptions, cursors, and subprograms that are available for use.
- A package's body fully defines cursors, functions, and procedures and thus implements the specification.
- Package Define in 2 Steps:
    1. Package Specification
    2. Package Body

PACKAGE

❖ PACKAGE SPECIFICATION:

- The package specification contains:
  1. Name of the package
  2. Names of the data types of any arguments
- This declaration is local to the database and global to the package

- Syntax:

```
Create [or replace] package package_name
{is | as}
Package_specification
End package_name;
```

PACKAGE

❑ PACKAGE BODY:

❑The body of a package contains the definition of public objects that are declared in the specification.

❑The body can also contain other object declarations that are private to the package.

❑The objects declared privately in the package body are not accessible to other objects outside the package.

❑Unlike package specification, the package body can contain subprogram bodies.

❑After the package is written, debugged, compiled and stored in the database applications can reference the package's types, call its subprograms, use its cursors, or raise its exceptions.

❑Syntax:

```
Create [or replace] package body package_name
{is | as}
Package_body
End package_name;
```

PROGRAM 1: Create a package called "Employee_Info" that contains two functions, "get_salary" and "get_job_title". The "get_salary" function takes an employee ID as input and returns the employee's salary. The "get_job_title" function takes an employee ID as input and returns the employee's job title.

Two steps:

1. Package specification of emp_info
2. Package body of emp_info

1. Package Specification:

```
create or replace package emp_info is
    function get_salary(id varchar2) return number;
    function get_job_title(id varchar2) return varchar2;
end emp_info;
```

Package Body:

```
create or replace package body emp_info is

function get_salary(id varchar2) return number is
sal number;
begin
 select salary into sal from emp where eid=id;
 return sal;
end;

function get_job_title(id varchar2) return varchar2 is
title varchar2(20);
begin
 select job_t into title from emp where eid=id;
 return title;
end;
end emp_info;
```

Use Package from PL/SQL block for employee E01:

Autocommit Display 10

```
DECLARE
    id emp.eid%type;
    sal emp.salary%type;
    title emp.job_t%type;
BEGIN
    id:=:id;
    sal:=emp_info.get_salary(id);
    DBMS_OUTPUT.PUT_LINE('Salary::'||sal);
    title:=emp_info.get_job_title(id);
    DBMS_OUTPUT.PUT_LINE('Job Title::'||title);
END;
```

Results Explain Describe Saved SQL History

```
Salary::78000
Job Title::precident

Statement processed.
```