ORACLE Enterprise Resource Planning (ERP)

TEKSTAC Oracle ERP PL/SQL Solution

14 Solved Queries

By

Rajdeep Das

Control Structures

1. Area of Circle

Write a PL/SQL block to calculate the area of a circle for the radius ranging from 3 to 7 .Store the radius and corresponding area into the Circle table.

Circle :

Radius	Number(5)
Area	Number(7,2)

```
declare

pi constant number(4,2):=3.14;

radius number(2);

area number(14,2);

begin

radius :=3;

while radius <=7

loop

area := pi*power(radius,2);

insert into Circle values(radius,area);

radius := radius+1;

end loop;

end;

/
```

Using SQL in PL/SQL

2. Insert Record using Anonymous Block

Create a PL/SQL block to insert a new record into the Department table. Fetch the maximum department id from the Department table and add 10 to it; take this value for department id; 'TESTING' is the value for department name and CHN-102 is the value for Location ID.

Table name : Department

Column name	Data type	Constraints
DEPARTMENT_ID	NUMBER(5)	PK
DEPARTMENT_NAME	VARCHAR2(25)	NOT NULL
LOCATION_ID	VARCHAR2(15)	

Sample Output:

DEPARTMENT_ID DEPARTMENT_NAME LOCATION_ID

---------- ------------------------------ ----------

XXXX TESTING CHN-102

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
BEGIN
INSERT INTO Department(DEPARTMENT_ID,DEPARTMENT_NAME,LOCATION_ID)
select max(DEPARTMENT_ID)+10,'TESTING','CHN-102' from Department;
END;
/
```

3. Update Location

Create a PL/SQL block to update the location ID for an existing department, which has location ID preceded with 'HQ' as

'HQ-BLR-101'.

Table name : Department

Column name	Data type	Constraints
DEPARTMENT_ID	NUMBER(5)	PK
DEPARTMENT_NAME	VARCHAR2(25)	NOT NULL
LOCATION_ID	VARCHAR2(15)	

Sample Output:

DEPARTMENT_ID DEPARTMENT_NAME LOCATION_ID

---------- ---------- ----------

xxxx xxxxx HQ-BLR-101

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
begin
update department
set LOCATION_ID='HQ-BLR-101'
WHERE LOCATION_ID LIKE 'HQ%';
END;
/
```

4. Delete Record(s) at a particular location

Create a PL/SQL Block to delete all the records of department, which is located in 'CHN-102'.

Table name : Department

Column name	Data type	Constraints
DEPARTMENT_ID	NUMBER(5)	PK
DEPARTMENT_NAME	VARCHAR2(25)	NOT NULL
LOCATION_ID	VARCHAR2(15)	

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
begin

delete from department

WHERE LOCATION_ID='CHN-102';

END;

/
```

PL/SQL Cursors and Exceptions

5. Display department names using Cursors

Create a PL/SQL block to display all the department names from the Department table using cursors.

Column name	Data type	Constraints
DEPARTMENT_ID	NUMBER(5)	PK
DEPARTMENT_NAME	VARCHAR2(25)	NOT NULL
LOCATION_ID	VARCHAR2(15)	

Sample Output:

Department Names are :

ADMIN
DEVELOPMENT

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
SET SERVEROUTPUT ON;

DECLARE v_dept department.department_name%type;

CURSOR c_dept is SELECT department_name FROM department order by

department_name asc;

BEGIN dbms_output.put_line('Department Names are :');

OPEN c_dept;

LOOP

FETCH c_dept INTO v_dept;

EXIT WHEN c_dept%notfound;

dbms_output.put_line(v_dept);

END LOOP;

CLOSE c_dept; END;
```

6. Department Highest Salary Expenditure

Write a PL/SQL block to display the department name and the total salary expenditure of the department from the Employee table.

EMPLOYEE:

Column name	Data type	Constraints
EMP_ID	NUMBER(5)	PK
EMP_NAME	VARCHAR2(25)	NOT NULL
SALARY	NUMBER(10,2)	
DEPT	VARCHAR2(25)	

EMP_ID	EMP_NAME	SALARY	DEPT
101	Tom	54000	MECH
102	William	43000	CSE
103	John	34560	MECH

104	Smith	56000	CSE
105	Steve	23450	IT

Sample Output:
Department-wise salary expenditure:
IT department, total salary is 23450
CSE department, total salary is 99000
MECH department, total salary is 88560

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
set serveroutput on;

declare

   cursor c_depts is select DEPT, sum(SALARY) as sumsal from employee group by
DEPT;

   v_depts c_depts%rowtype;

begin

   open c_depts;

   dbms_output.put_line('Department-wise salary expenditure:');

   loop

      fetch c_depts into v_depts;

      exit when c_depts%notfound;

      dbms_output.put_line(v_depts.DEPT || ' department, total salary is ' ||
v_depts.sumsal );

   end loop;

   close c_depts;

end;

 /
```

7. Department Details

Create a PL/SQL block to retrieve all the information about each department from the DEPARTMENT table and display the information on the screen, incorporating a PL/SQL table of records.

Column name	Data type	Constraints
DEPARTMENT_ID	NUMBER(5)	PK
DEPARTMENT_NAME	VARCHAR2(25)	NOT NULL
LOCATION_ID	VARCHAR2(15)	

Sample Output:

Department Details are :

1000, ADMIN, HQ-101

1010, DEVELOPMENT, CBE-103

1020, TESTING, CHN-102

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
SET SERVEROUTPUT ON;

DECLARE

v_dept department.department_id%type;

w_dept department.department_name%type;

x_dept department.location_id%type;

CURSOR c_dept is SELECT department_id, department_name, location_id FROM
department order by

department_id asc;

BEGIN dbms_output.put_line('Department Details are :');

OPEN c_dept;

LOOP

FETCH c_dept INTO v_dept, w_dept, x_dept;
```

```
EXIT WHEN c_dept%notfound;

dbms_output.put_line(v_dept || ', ' || w_dept || ', ' || x_dept);

END LOOP;

CLOSE c_dept;

END;

/
```

PL/SQL Functions & Procedures

8. Procedure with Exception Handling

Create a PL/SQL Procedure to insert employee details into Employee table. Before inserting, check whether the employee age is eligible or not. Employee age should be 18 or greater. Values are passed as argument to the procedure.

If age valid, insert employee record into table and print the message "Age valid - Record inserted", else print the message "Age invalid - Record not inserted" by raising an exception.

Table: EMPLOYEE

Column name	Data type	Constraints
EMP_ID	NUMBER(5)	PK
EMP_NAME	VARCHAR2(25)	NOT NULL
AGE	NUMBER(3)	

Functional Requirement:

```
PROCEDURE CHECK_AGE_ELIGIBILITY(

 v_id IN EMPLOYEE.EMPID%TYPE,

 v_name IN EMPLOYEE.EMPNAME%TYPE,

 v_age IN EMPLOYEE.AGE%TYPE)
```

Sample Input 1 :

```
CHECK_AGE_ELIGIBILITY(103, 'Robert',  24 ) ;
```

Sample Output 1:

Age valid - Record inserted

Sample Input 2:

CHECK_AGE_ELIGIBILITY(104,'Riya', 4);

Sample Output 2:

Age invalid - Record not inserted

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
create or replace PROCEDURE CHECK_AGE_ELIGIBILITY(
    v_id IN EMPLOYEE.EMPID%TYPE,
    v_name IN EMPLOYEE.EMPNAME%TYPE,
    v_age IN EMPLOYEE.AGE%TYPE) as

not_of_age exception;

    begin
        if v_age>=18 then
            insert into EMPLOYEE(empid,empname,age) values(v_age,v_name,v_age);
            dbms_output.put_line('Age valid - Record inserted');
        else
            raise not_of_age;

        end if;
    exception
```

```
        when not_of_age then
        dbms_output.put_line('Age invalid - Record not inserted');
end;
/
```

9. Calculate increment using Function

Create a PL/SQL Function to calculate increment for the employees. Function will take increment percentage and salary as input and return increment amount as output.

Employee:

Column name	Data type	Constraints
EMP_ID	NUMBER(5)	PK
EMP_NAME	VARCHAR2(25)	NOT NULL
SALARY	NUMBER(10,2)	

Functional Requirement:
FUNCTION calculate_Increment(incperc number, salary number)

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
create or replace function calculate_increment
(incperc in number,
salary  in number
)
return number
as
begin
return salary * incperc/100;
end;
/
```

10. Remove Employee records – Procedures

Create a procedure that deletes employee records from the Employee table. Get the department name as an input parameter. Delete the employee records who belongs to that department.

Display the count of employee records that were deleted. If the respective department was not found, then raise "DeptNotFoundException" and print the message 'No Records found.'

Assume the Employee table has been already created and few records has been inserted.

EMPLOYEE:

Column name	Data type	Constraints
EMP_ID	NUMBER(5)	PK
EMP_NAME	VARCHAR2(25)	NOT NULL
SALARY	NUMBER(10,2)	
DEPT	VARCHAR2(25)	

EMP_ID	EMP_NAME	SALARY	DEPT
101	Tom	54000	MECH
102	William	43000	CSE
103	John	34560	MECH
104	Smith	56000	CSE
105	Steve	23450	IT

Functional Requirements:

PROCEDURE DELETE_EMPLOYEE(v_dept IN EMPLOYEE.dept%TYPE)

Sample Output:

2 Employee record(s) got deleted.

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE DELETE_EMPLOYEE(V_DEPT IN
EMPLOYEE.DEPT%TYPE)
```

```
IS
DeptNotFoundException EXCEPTION;
BEGIN
DELETE FROM EMPLOYEE WHERE DEPT = V_DEPT;
IF SQL%FOUND THEN
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' Employee record(s) got deleted.');
ELSE
RAISE DeptNotFoundException;
END IF;
EXCEPTION
WHEN DeptNotFoundException
THEN
DBMS_OUTPUT.PUT_LINE( 'No Records found.');
END;
/
```

11. Concatenate Employee names – Functions

Write a PL/SQL function to concatenate first name and last name of an employee. Pass employee id as an input to the function CONCAT_NAME.
EMPLOYEE:

Column name	Data type	Constraints
EMP_ID	NUMBER(5)	PK
FIRST_NAME	VARCHAR2(25)	NOT NULL
LAST_NAME	VARCHAR2(25)	
DEPT	VARCHAR2(25)	

Functional Requirement:
FUNCTION CONCAT_NAME(v_id employee.emp_id%type)

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
create or replace function concat_name(v_id employee.emp_id%type)

return varchar2

is

retval varchar2(25);

begin

select first_name || last_name

into retval

from employee

where emp_id = v_id;

return retval;

end;

/
```

PL/SQL Packages

12. Package with a Procedure to update salary

Create a PL/SQL Package with Procedure in it. Procedure will take designation and incentive as input and update the employee salary by adding the incentive for the given designation. Display the number of employee records that have got updated, e.g. '3 employee record(s) are updated'.

Employee:

Column name	Data type	Constraints
EMP_ID	NUMBER(5)	PK
EMP_NAME	VARCHAR2(25)	NOT NULL
SALARY	NUMBER(10,2)	
DESIGNATION	VARCHAR2(25)	

EMP_ID	EMP_NAME	SALARY	DESIGNATION
101	Mathew	45000	PROGRAMMER
102	Sam	54000	MANAGER
103	John	35000	TEAM LEAD
104	James	48000	PROGRAMMER
105	Josh	25000	TESTER

Functional Requirements:
Package name as EMP_DESIGNATION, and
Procedure signature:
EMP_DETAILS(design employee.designation%TYPE, incentive number);

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
CREATE OR REPLACE PACKAGE emp_designation

AS

PROCEDURE emp_details (design employee.designation%TYPE, incentive NUMBER);

END emp_designation;

/

CREATE OR REPLACE PACKAGE BODY emp_designation

AS

PROCEDURE emp_details (design employee.designation%TYPE, incentive NUMBER)
```

```
IS

BEGIN

UPDATE employee

SET employee.salary = employee.salary + incentive

WHERE employee.designation = design;

DBMS_OUTPUT.put_line(SQL%ROWCOUNT || ' employee(s) are updated.');

END emp_details;

END emp_designation;

/
```

PL/SQL Triggers

13. Insert a Record – Triggers

Create a PL/SQL Trigger to display the message “NEW EMPLOYEE DETAILS INSERTED”, whenever a new record is inserted into Employee table.

Column name	Data type	Constraints
EMP_ID	NUMBER(5)	PK
EMP_NAME	VARCHAR2(25)	NOT NULL
SALARY	NUMBER(10,2)	

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
create or replace trigger trigger1

after insert on employee

for each row

begin

dbms_output.put_line('NEW EMPLOYEE DETAILS INSERTED');

end;
```

14. Delete a Record – Triggers

An HR system has an Employee table that holds a row for each employee within the company. Each record in the table has a employee id, employee name and manager column, that holds the id for the employee's manager. Write a trigger so that when an employee record is deleted, the record details need to be inserted into a table called Employee_archive along with the deleted date.

EMPLOYEE:

EMPID	NUMBER	PRIMARY KEY
EMPNAME	VARCHAR2(25)	
MANAGERID	NUMBER	

EMPLOYEE_ARCHIVE:

EMPID	NUMBER	PRIMARY KEY
EMPNAME	VARCHAR2(25)	
MANAGERID	NUMBER	
DELETED_DATE	DATE	

(Hint: Data is case sensitive. Use '/' to terminate the PLSQL block)

```
create or replace trigger trg_bd_emp
  before delete on employee
  for each row
begin
  insert into employee_archive (empid, empname, managerid, deleted_date)
    values (:old.empid, :old.empname, :old.managerid, sysdate);
end trg_bd_emp;
/
```