Oracle Application E-BUSINESS SUITE

Oracle Applications Technical Training

PLSQL Exercise

Author:	Consultant
Creation Date:	2016/01/01
Last Updated:	2016/01/01
Document Ref:	
Version:	1.0

Approve	

1. PLSQL练习题目

1.1 PLSQL简介

1. 定义变数v_sex 为字元型态(只有一个字元)，试举出三种方式

2. 定义v_name 为varchar2(40)、v_empno 为Number，v_hiredate 为Date 资料型态

3. 定义v_sal 变数为Number 型态且不可为空值

4. 同上例，v_sal 预设值为1000

5. 定义变数v_tax 为常数值(34.5)

6. 定义变数v_hiredate 继承原emp 表格中hiredate 栏位资料型态

7. 同上例，定义一变数v_hiredate2 继承v_hiredate 变数资料型态

8. 定义变数dept_rec 继承原Dept 表格所有栏位资料型态

9. 宣告dept_rec_type Record 资料型态，包含deptno number 及dname varchar2(40)

10. 由上例资料型态，定义Record 变数dept_rec

1.2 执行语法说明

1. 写出下列语法产出结果

```
DECLARE
    v_name VARCHAR2(40) :='KEN';
    v_mgr NUMBER :=7688;
    v_times NUMBER :=10;
BEGIN
  DECLARE
      v_name VARCHAR2(40) :='JOHN';
      v_times NUMBER :=20;
  BEGIN
      dbms_output.put_line(v_name||''||v_mgr ||''||v_times) ;
  END ;
  dbms_output.put_line(v_name||''||v_mgr||''||v_times) ;
END;
```

2. 于PL/SQL 语法中，使用SQL 语法将'KING'薪水资料带给变数v_sal

3. 使用IF 逻辑运算判断成绩，当成绩(v_score)>=90 则v_grade 为A，介于90(不含)至80(含)为B，介于80(不含)至70(含)为C，70 以下为D

4. 使用LOOP 方式，计算由1+2...10 所得结果

1.3 副程式

1. 建立Function 名称为get_loc，传入员工代号，回传部门所在地

2. 使用已建立好get_loc 函数，传入7839 员工代号，并将回传值传给v_loc 变数

3. 同范例1，建立Procedure 名称为get_loc1，使用参数(OUT)方式回传

4. 使用已建立好get_loc1 Procedure，传入7839 员工代号将值带给v_loc 变数

5. 建立一Package Spec handle_loc 包含上述两个副程式

6. 建立一Package Body handle_loc 包含上述两个副程式

7. 使用handle_loc Package 中handle_loc Function，传入7839 员工代号，并将值传给v_loc 变数

1.4 Cursor功能说明

1. 宣告一个Cursor 名为 emp_dept_cur，资料集为员工代号、员工姓名、部门代号及部门名称

2. 同上例，范围缩小为部门代码不包含10 及依照薪水由大至小排序

3. 同上例，使用参数(p_deptno)方式传入

4. 同上例，建立一显性(Explicit) Cursor 将资料(empno、ename、deptno、dname)写入emp_dept_temp 中

5. 同上例，使用Cursor for loop 方式，将资料(empno、ename、deptno、dname)写入emp_dept_temp 中

1.5 例外状况处理

1. 判断下列语法，会产生结果为何?

```
DECLARE
   i NUMBER :=1;
   j NUMBER :=0;
BEGIN
      BEGIN
         i:= i/j;
      EXCEPTION
          WHEN no_data_found THEN
          dbms_output.put_line('no_data_found');
      END;
EXCEPTION
 WHEN too_many_rows THEN
    dbms_output.put_line('no_data_found');
WHEN zero_divide THEN
    dbms_output.put_line('zero_divide');
WHEN OTHERS THEN
    dbms_output.put_line('others');
END;
```

2. 建立一 Exception 当sal/comm 发生zero_divide 时，将错误代码及写入讯息记录档emp_error 中

3. 同上例，当 comm 为0 时，于执行语法中使用raise_application_error 触发 Exception 错误代码为-20011 及错误讯息"dividor can not be zero!!"

2. SQL 练习答案

2.1 PLSQL简介

1. 定义变数v_sex 为字元型态(只有一个字元)，试举出三种方式

```
v_sex CHAR;
v_sex CHAR(1);
v_sex VARCHAR2(1);
```

2. 定义v_name 为varchar2(40)、v_empno 为Number，v_hiredate 为Date 资料型态

```
v_name VARCHAR2(40);
v_empno NUMBER;
v_hiredate DATE;
```

3. 定义v_sal 变数为Number 型态且不可为空值

```
v_sal NUMBER NOT NULL;
```

4. 同上例，v_sal 预设值为1000

```
v_sal NUMBER NOT NULL DEFAULT 1000;
```

5. 定义变数v_tax 为常数值(34.5)

```
v_tax CONSTANT NUMBER := 34.5;
```

6. 定义变数v_hiredate 继承原emp 表格中hiredate 栏位资料型态

```
v_hiredate emp.hiredate%TYPE;
```

7. 同上例，定义一变数v_hiredate2 继承v_hiredate 变数资料型态

```
v_hiredate2 v_hiredate%TYPE;
```

8. 定义变数dept_rec 继承原Dept 表格所有栏位资料型态

```
dept_rec dept%ROWTYPE;
```

9. 宣告dept_rec_type Record 资料型态，包含deptno number 及dname varchar2(40)

```
TYPE dept_rec_type IS RECORD
(deptno NUMBER,
dname VARCHAR2(40)
);
```

10. 由上例资料型态，定义Record 变数dept_rec

```
dept_rec dept_rec_type;
```

2.2 执行语法说明

1. 写出下列语法产出结果

```
DECLARE
```

```
    v_name VARCHAR2(40) :='KEN';
    v_mgr NUMBER :=7688;
    v_times NUMBER :=10;
BEGIN
   DECLARE
      v_name VARCHAR2(40) :='JOHN';
      v_times NUMBER :=20;
   BEGIN
      dbms_output.put_line(v_name||' '||v_mgr ||' '||v_times) ;
   END ;
   dbms_output.put_line(v_name||' '||v_mgr||' '||v_times) ;
END;
```

结果:

```
JOHN 7688 20
KEN 7688 10
```

2. 于PL/SQL 语法中，使用SQL 语法将'KING'薪水资料带给变数v_sal

```
SELECT sal  INTO v_sal
  FROM emp
WHERE ename = 'KING';
```

3. 使用IF 逻辑运算判断成绩，当成绩(v_score)>=90 则v_grade 为A，介于90(不含)至80(含)为B，介于80(不含)至70(含)为C，70 以下为D

```
IF v_score >= 90 THEN
   v_grade := 'A';
ELSIF v_score >=80 THEN
   v_grade := 'B';
ELSIF v_score >=70 THEN
   v_grade := 'C';
ELSE
   v_grade := 'D';
END IF;
```

4. 使用LOOP 方式，计算由1+2...10 所得结果

```
v_acc:=0;
v_time:=1;
LOOP
   v_acc := v_acc + v_time;
   EXIT WHEN v_time>=10;
   v_time := v_time + 1;
   dbms_output.put_line(v_acc);
```

```
END LOOP;
```

2.3 副程式

1. 建立Function 名称为get_loc，传入员工代号，回传部门所在地

```
CREATE OR REPLACE FUNCTION get_loc(p_empno IN NUMBER)
RETURN VARCHAR2
IS
v_loc VARCHAR2(120);
BEGIN
SELECT dept.loc INTO v_loc
FROM emp,dept
WHERE emp.deptno = dept.deptno
AND empno = p_empno;
RETURN v_loc;
EXCEPTION WHEN OTHERS THEN
RETURN NULL;
END get_loc;
```

2. 使用已建立好get_loc 函数，传入7839 员工代号，并将回传值传给v_loc 变数

```
v_loc := get_loc(7839);
```

3. 同范例1，建立Procedure 名称为get_loc1，使用参数(OUT)方式回传

```
CREATE OR REPLACE PROCEDURE get_loc1(p_empno IN NUMBER,x_loc
OUT VARCHAR2)
IS
BEGIN
   SELECT dept.loc INTO x_loc
    FROM emp,dept
   WHERE emp.deptno = dept.deptno
      AND empno = p_empno;
EXCEPTION
   WHEN OTHERS THEN
   x_loc := NULL;
END get_loc1;
```

4. 使用已建立好get_loc1 Procedure，传入7839 员工代号将值带给v_loc 变数

```
get_loc1(7839,v_loc);
```

5. 建立一Package Spec handle_loc 包含上述两个副程式

6. 建立一Package Body handle_loc 包含上述两个副程式

7. 使用handle_loc Package 中handle_loc Function，传入7839 员工代号，并将值传给v_loc 变数

2.4 Cursor功能说明

1. 宣告一个Cursor 名为 emp_dept_cur，资料集为员工代号、员工姓名、部门代号及部门名称

2. 同上例，范围缩小为部门代码不包含10 及依照薪水由大至小排序

3. 同上例，使用参数(p_deptno)方式传入

4. 同上例，建立一显性(Explicit) Cursor 将资料(empno、ename、deptno、dname)写入emp_dept_temp 中

5. 同上例，使用Cursor for loop 方式，将资料(empno、ename、deptno、dname)写入emp_dept_temp 中

2.5 例外状况处理

1. 判断下列语法，会产生结果为何?

```
DECLARE
   i NUMBER :=1;
   j NUMBER :=0;
BEGIN
      BEGIN
        i:= i/j;
      EXCEPTION
          WHEN no_data_found THEN
          dbms_output.put_line('no_data_found');
      END;
EXCEPTION
 WHEN too_many_rows THEN

    dbms_output.put_line('no_data_found');
WHEN zero_divide THEN
    dbms_output.put_line('zero_divide');
WHEN OTHERS THEN
    dbms_output.put_line('others');
END;
```

2. 建立一 Exception 当sal/comm 发生zero_divide 时，将错误代码及写入讯息记录档emp_error 中

3. 同上例，当 comm 为0 时，于执行语法中使用raise_application_error 触发 Exception 错误代码为-20011 及错误讯息"dividor can not be zero!!"

3. Open and Closed Issues

Add open issues that you identify while writing or reviewing this document to the open issues section. As you resolve issues, move them to the closed issues section and keep the issue ID the same. Include an explanation of the reSolution.

Open Issues

ID	Issue	ReSolution	Responsibility	Target Date	Impact Date

Closed Issues

ID	Issue	ReSolution	Responsibility	Target Date	Impact Date
		1			