# oracle pl/sql 实例练习

第一部分：oracle pl/sql 实例练习(1)

一、使用 scott/tiger 用户下的 emp 表和 dept 表完成下列练习，表的结构说明如下

emp 员工表(empno 员工号/ename 员工姓名/job 工作/mgr 上级编号/hiredate 受雇日期/sal 薪金/comm 佣金/deptno 部门编号)

dept 部门表(deptno 部门编号/dname 部门名称/loc 地点)

工资 = 薪金 + 佣金

也可以通过以下脚本测试：

```
create table DEPT
(
DEPTNO NUMBER(2) not null,
DNAME VARCHAR2(14),
LOC     VARCHAR2(13)
)
tablespace USERS;
alter table DEPT add constraint PK_DEPT primary key (DEPTNO);
insert into DEPT (DEPTNO, DNAME, LOC)
values (10, 'ACCOUNTING', 'NEW YORK');
insert into DEPT (DEPTNO, DNAME, LOC)
values (20, 'RESEARCH', 'DALLAS');
insert into DEPT (DEPTNO, DNAME, LOC)
values (30, 'SALES', 'CHICAGO');
insert into DEPT (DEPTNO, DNAME, LOC)
values (40, 'OPERATIONS', 'BOSTON');
commit;
create table EMP
(
EMPNO     NUMBER(4) not null,
ENAME     VARCHAR2(10),
JOB       VARCHAR2(9),
MGR       NUMBER(4),
HIREDATE DATE,
SAL       NUMBER(7,2),
COMM      NUMBER(7,2),
DEPTNO    NUMBER(2)
)
tablespace USERS;
alter table EMP add constraint PK_EMP primary key (EMPNO);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7369, 'SMITH', 'CLERK', 7902, to_date('17-12-1980', 'dd-mm-yyyy'), 800, null, 20);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7499, 'ALLEN', 'SALESMAN', 7698, to_date('20-02-1981', 'dd-mm-yyyy'), 1600, 300,
30);
```

```
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7521, 'WARD', 'SALESMAN', 7698, to_date('22-02-1981', 'dd-mm-yyyy'), 1250, 500,
30);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7566, 'JONES', 'MANAGER', 7839, to_date('02-04-1981', 'dd-mm-yyyy'), 2975, null,
20);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7654, 'MARTIN', 'SALESMAN', 7698, to_date('28-09-1981', 'dd-mm-yyyy'), 1250,
1400, 30);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7698, 'BLAKE', 'MANAGER', 7839, to_date('01-05-1981', 'dd-mm-yyyy'), 2850, null,
30);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7782, 'CLARK', 'MANAGER', 7839, to_date('09-06-1981', 'dd-mm-yyyy'), 2450, null,
10);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7788, 'SCOTT', 'ANALYST', 7566, to_date('19-04-1987', 'dd-mm-yyyy'), 3000, null,
20);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7839, 'KING', 'PRESIDENT', null, to_date('17-11-1981', 'dd-mm-yyyy'), 5000, null,
10);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7844, 'TURNER', 'SALESMAN', 7698, to_date('08-09-1981', 'dd-mm-yyyy'), 1500, 0,
30);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7876, 'ADAMS', 'CLERK', 7788, to_date('23-05-1987', 'dd-mm-yyyy'), 1100, null,
20);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7900, 'JAMES', 'CLERK', 7698, to_date('03-12-1981', 'dd-mm-yyyy'), 950, null, 30);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7902, 'FORD', 'ANALYST', 7566, to_date('03-12-1981', 'dd-mm-yyyy'), 3000, null,
20);
insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values (7934, 'MILLER', 'CLERK', 7782, to_date('23-01-1982', 'dd-mm-yyyy'), 1300, null,
10);
commit;
```

二、问题：

1．列出至少有一个员工的所有部门。

2．列出薪金比“SMITH”多的所有员工。

3．列出所有员工的姓名及其直接上级的姓名。

4．列出受雇日期早于其直接上级的所有员工。

5．列出最低薪金大于 1500 的各种工作。

6．列出在每个部门工作的员工数量、平均工资和平均服务期限。

答案：

1．列出至少有一个员工的所有部门。

```
select deptno from emp group by deptno having count(*)>1;
```

解析：该语句主要学习分组函数 group by，以及对分组后过滤的条件函数 having。需要注意的是使用了分组函数语句的 select 字段中不能包含 group by 后没有的字段，原因是如果显示非分组的字段就可能显示多条记录，就达不到按某字段分组的目的。Having 只能用在分组函数 group by 后，相当于对分组后的记录做 where 条件过滤。

2．列出薪金比“SMITH”多的所有员工。

```
select * from emp where sal>(select sal from emp where ename='SMITH');
```

解析：该语句主要学习子查询，且子查询在 where 条件的后面。注意和 3 的区别。

3．列出所有员工的姓名及其直接上级的姓名。

```
select ename,(select ename from emp where empno=a.mgr) from emp a;
```

解析：该语句主要学习子查询，且子查询在 select 和 from 的中间，同时内部语句使用了外部语句的字段。注意和 2 的区别。

4．列出受雇日期早于其直接上级的所有员工。

```
select ename from emp a where hiredate>(select hiredate from emp where empno=a.mgr);
```

解析：该语句主要学习子查询，且子查询在 where 条件的后面。同时内部语句使用了外部语句的字段。注意和 2 的区别。注意和 2 的区别。

5．列出最低薪金大于 1500 的各种工作。

```
select job,min(sal) msal from emp group by job having min(sal)>1500;
```

解析：该语句主要学习分组函数，功能同 1。

6．列出在每个部门工作的员工数量、平均工资和平均服务期限。

```
select deptno,count(*), trunc(avg(sal+nvl(comm,0))) avgsal, trunc(avg(sysdate-hiredate))
avgday from emp group by deptno;
```

解析：该语句主要学习分组函数，同时学习其相关的统计函数 count, avg 等。

三、以下是过程、函数的学习事例。所有的过程和函数都放到了 package 下面。

包声明：

```
create or replace package MyPack1 is
--过程模板
procedure P_Templet(i_myNumber In Number,o_myVarchar Out varchar2);
function f_Templet(i_a1 varchar2,i_a2 Number) return number;
--声明一个游标
type Ref_Cursor_Type is ref cursor;
--列出至少有一个员工的所有部门
procedure p_test1;
--列出薪金比“SMITH”多的所有员工
function f_test2(i_ename emp.ename%type) return number;
end MyPack1;
```

包实现：

```
create or replace package body MyPack1 is
--过程模板
    procedure P_Templet(i_myNumber In Number,
      o_myVarchar Out varchar2) is
    myDate      date;
```

```
    myNumber    Number(10);
    myVarchar   varchar2(50);
begin
       null;
       exception
       when others then
       dbms_output.put_line('程序出现了异常！'||Sqlcode||Sqlerrm);
end;
-------------------------------------------------
--函数模板
function f_Templet(i_a1 varchar2,i_a2 Number) return number is
    myDate      date;
    myNumber    Number(10);
    myVarchar   varchar2(50);
begin
       null;
       exception
       when others then
      dbms_output.put_line('程序出现了异常！'||Sqlcode||Sqlerrm);
       return myNumber;
end;
-----------------------------------------------------
--列出至少有一个员工的所有部门
--用一个 sql 语句实现：select count(*),deptno from emp group by deptno having count(*)>1;
--主要学习过程的写法，以及游标的用法。
procedure p_test1 is
   v_deptno         dept.deptno%type;
   v_deptname       dept.dname%type;
   v_count1    Number(10);
   C1             Ref_Cursor_Type;
begin
       open C1 for
         select distinct deptno,dname from dept;
        loop
       fetch C1
         into v_deptno,v_deptname;
          exit when C1%notfound;
            begin
            select count(*) into v_count1 from emp e where e.deptno=v_deptno;
            if(v_count1>0) then
              Dbms_Output.Put_Line(v_deptname);
            else null;
            end if;
            end;
```

```
        end loop;
       exception
      when others then
       null;
end;
```

---

--列出薪金比“SMITH”多的所有员工

--用一个 sql 语句实现：select * from emp where sal>(select sal from emp where ename='SMITH');

-- 调用该函数的语句：select * from (select ename,mypack1.f_test2(ename) as count1 from(select --distinct t.ename from emp t) ) where count1>0

--主要学习函数的写法，学习外部 sql 如何调用该函数，因为这样调用对条件控制或分页等更灵活，同--时降低了函数的复杂性。

```
function f_test2(i_ename emp.ename%type) return number is
   v_sal     Number(10);
   v_sal2     Number(10);
   v_return     Number(10);
begin
      begin
          select sal into v_sal from emp where ename='SMITH';
          exception
              when others then
              v_sal :=0;
      end;
      begin
          select sal into v_sal2 from emp where ename=i_ename;
          exception
              when others then
              v_sal2 :=0;
      end;
      v_return := v_sal2-v_sal;
      return v_return;
end;
end MyPack1;
```

---

第二部分：oracle pl/sql 实例练习(2)

--创建测试表

```
create table TESTTABLE
(
ID          NUMBER(4) not null,
CURRENTDATE DATE not null
)
```

---

实例 1

```
--该实例要学习如何通过 pl/sql 批量插入数据，以及常量、For 的用法。
Declare
maxrecords Constant Int:=100;
i Int:=1;
Begin
      For i In 1..maxrecords Loop
         Insert Into testtable(id,currentdate)
         Values(i,Sysdate);
         dbms_output.put_line('现在插入的内容是：'||i||'   '||Sysdate);
         Commit;          --这里切记要 commit 否则将不会将数据提交到表中
      End Loop;
      dbms_output.put_line(maxrecords||'条记录已经插入。');
End;
-----------------------------------------------------------------
实例 2
--这种写法是为了简化多个选择用 if 来做判断。
Declare
      v_test Int:=60;
      Begin
         Case v_test
         When 90 Then
         dbms_output.put_line('v_test 的值为：90！');
         When 80 Then
         dbms_output.put_line('v_test 的值为：80！');
         Else
         dbms_output.put_line('v_test 的值我不知道！');
         End Case;
      End;

-------------------------------------------------------
实例 3
--完整定义了一个记录类型的变量，简单应用。这里要学习如何使用 Record 类型变量。
Declare
      Type myrecord Is Record(
         r_Id Number(4),
         r_currentdate Date
         );
      v_myrecord myrecord;
      Begin
         Select * Into v_myrecord From testtable Where id=10;
         dbms_output.put_line('用记录类型的变量取出来的值为：'||
                        v_myrecord.r_Id||' '||v_myrecord.r_currentdate);
      End;
```

---

实例 4

```
--学习使用%Rowtype 定义变量
Declare
      v_myrow testtable%Rowtype;
      Begin
          Select * Into v_myrow From testtable Where id=20;
          dbms_output.put_line('用 rowtype 查询的结果是：
'||v_myrow.id||v_myrow.currentdate);
      End;
```

---

实例 5

```
--学习如何定义一维表变量，以及如何给一维表变量赋值。这种变量类似编程语言中的一维数组
Declare
     Type mytbtype1 Is Table Of Varchar2(4) Index By Binary_Integer;
     Type mytbtype2 Is Table Of testtable.id%Type Index By Binary_Integer;
     tb1 mytbtype1;
     tb2 mytbtype2;
     Begin
         tb1(1):='大学';
         tb1(2):='大专';
         dbms_output.put_line(tb1(1)||tb1(2));
         select id BULK COLLECT into tb2 from testtable where id<10 order by id;
         For i In 1..tb2.count Loop
           dbms_output.put_line(tb2(i));
         End Loop;
     End;
```

---

实例 6

```
--定义一个多维表变量，这就像一个二维数组
--当然这个二维的数组的下表就有些区别与我们在编程语言中熟悉的二维数组了
--可以理解为一维存储的列名，而另一维则是存储与一维列名相对应的数据
Declare
     --这里区别与一维表变量的定义
     Type multbtype Is Table Of testtable%Rowtype Index By Binary_Integer;
     multb multbtype;
     Begin
         Select * Into multb(1)
         From testtable
         Where id=88;
         dbms_output.put_line('multb(1).id='||
                              multb(1).id||
```

```
                    ' multb(1).currentedate'||
                    multb(1).currentdate
                    );
    End;
```

---

实例 7

```
--这里主要学习 oracle 中的‘数组’的属性和方法，有点类似数据结构中的链表
Declare
    Type mytabletype Is Table Of Varchar2(9) Index By Binary_Integer;
    tb mytabletype;
    Begin
        tb(1):='成都市';
        tb(2):='太原市';
        tb(3):='北京市';
        dbms_output.put_line('记录总数：'||to_char(tb.Count));
        dbms_output.put_line('第一条记录为：'||tb.First||'其值为：'||tb(tb.First));
        dbms_output.put_line('最后条记录为：'||tb.Last||'其值为：'||tb(tb.Last));
        dbms_output.put_line('第二条的前一条记录为：'||tb.Prior(2)||'其值为：
'||tb(tb.Prior(2)));
        dbms_output.put_line('第二条的后一条记录为：'||tb.Next(2)||'其值为：
'||tb(tb.Next(2)));
        dbms_output.put_line('第二条记录为：'||tb(2));
        tb.Delete(2);
        dbms_output.put_line('删除第二条记录后的第二条记录为:'||tb(3));
        --exists 存在的问题，不知到怎么使用
        --tb.Exists('太原市');
    End;
```

---

实例 8

```
--学习如何定义和使用游标，这里展示了%isopen,%found,%Rowcount
Declare
    Cursor mycursor Is
        Select * From testtable e
        Where e.id<100;
    currentrecord mycursor%Rowtype;
    Begin
        --打开游标
        If mycursor%Isopen Then
          dbms_output.put_line('该游标已经打开了，正在关闭！');
          Close mycursor;
```

```
        Else
          dbms_output.put_line('游标已经关闭');
        End If;
        --读取数据
        Open mycursor;--打开游标时最好做判断当前游标是否是关闭的，这里略写
        Loop
        Fetch mycursor Into currentrecord;
        Exit When mycursor%Notfound;
        Begin
           dbms_output.put_line('游标已经取到数据,查询结果是：');
           dbms_output.put_line(to_char(currentrecord.id||currentrecord.currentdate));
   End;
   End Loop;
   Close mycursor;--关闭游标时最好做判断当前游标是否是打开的，这里略写
        --读取记录总条数
        Open mycursor;
        Loop
           Fetch mycursor Into currentrecord;
        Exit When mycursor%Notfound;
        End Loop;
        dbms_output.put_line('查结果总共有：'||mycursor%Rowcount);
        --关闭游标
        If mycursor%Isopen Then
         dbms_output.put_line('正在关闭游标。');
        Close mycursor;
        End If;
     End;
--------------------------------------------------------------------------
实例 9
--学习存储过程如何带输入参数和输出参数
Create Or Replace Procedure myprocedure(
      i_id In testtable.id%Type,
      o_currentdate Out testtable.currentdate%Type) Is
      v_currentdate testtable.currentdate%Type;
      Begin
        Select currentdate Into v_currentdate From testtable t Where t.id=i_id;
        dbms_output.put_line('currentdate：'||v_currentdate);
        o_currentdate:=v_currentdate;
        Exception
        When Others Then
        dbms_output.put_line('程序出现了异常！'||Sqlcode||Sqlerrm);
      End;
```