**1、本章面试题**

     什么存储过程，有什么作用

什么函数，有什么作用，函数与存储过程有什么异同

什么是包，有什么用

**2、知识点**

**2.1、课程回顾**

什么是语句块及作用

语句块基本语法 (定义 执行 异常处理)

%type %rowtype record table用法

异常机制 (预定义异常 自定义异常)

事务（ACID特性及在oracle数据如何实现 commit rollback savepoint rollback to） spring框架事务隔离级别和传播特征（有难度）

死锁和如何防止死锁（使用oracle时一定及时的回滚或者提交数据）

**2.2、本章重点**

存储过程的概念，作用，实例

函数的概念，作用，实例

函数和存储过程的区别

包的概念，作用，实例

**3、具体内容**

**3.1 存储过程**

**3.1.1 概念**

stored procedure是一组为了完成一定业务的语句集合，经过编译后存储在数据库中。用户可以通过存储过程的名称，给出指定参数（需要参数）来执行。设计良好的一个数据库，都需要存储过程的支持。经过创建时编译一次，以后可以快速的执行，也可以被反复的调用。

**3.1.2 作用**

1，一次编译，以后执行不需编译，提高了语句的执行效率（自行编写语句的执行一次编译一次，效率低）

2，当实现的业务复杂时，可以节省网络开销（业务复杂包含很多的crud,和昨天举例10次请求10次响应，如果使用存储过程一次一次响应可以完成业务）

3， 可以被反复使用，减少数据库管理员DBA（或者开发人员）的工作量

4，安全性好（可以使用oracle提供的角色管理，来让不同的用户看到不同的存储过程）

**3.1.3 语法**

create [or replace] procedure 存储过程名称(arg1 [mode] type1,arg2 [mode] type2....)

as /is --相当于匿名块的declare

-- 定义部分

begin

-- 执行部分

exception

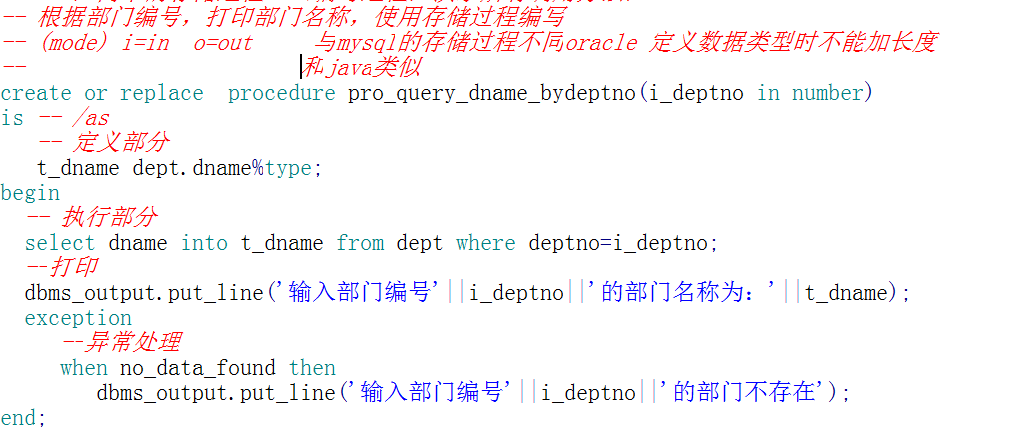
-- 异常处理

end;

mode 使用in或者out in代表输入参数可以省略不写 out相当于存储过程的返回值

**3.1.4 实例**

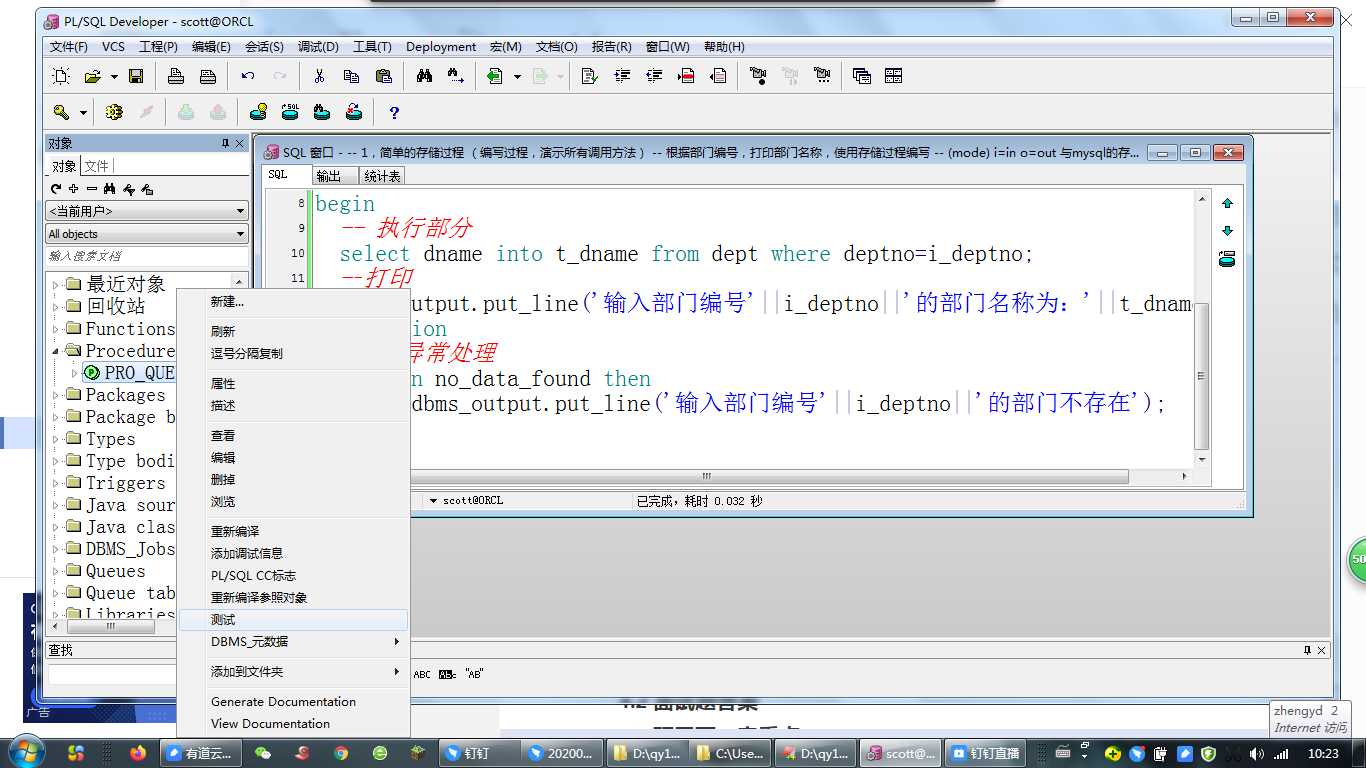
**1，简单的存储过程 （编写过程，演示所有调用方法）**

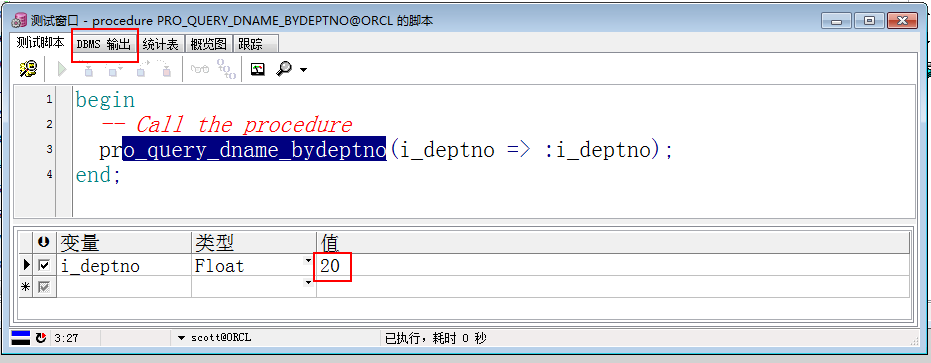


编写执行完存储过程后，有红叉说明语法有错误，点击该存储过程右键点编辑，再次执行，会有错误行提示（位置不太准确）

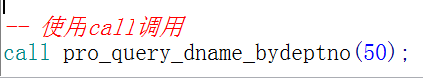
**存储过程的调用：**

1)，使用可视化工具（plsql直接调用）

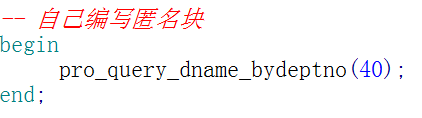




2) 使用call关键字调用（sql窗口）



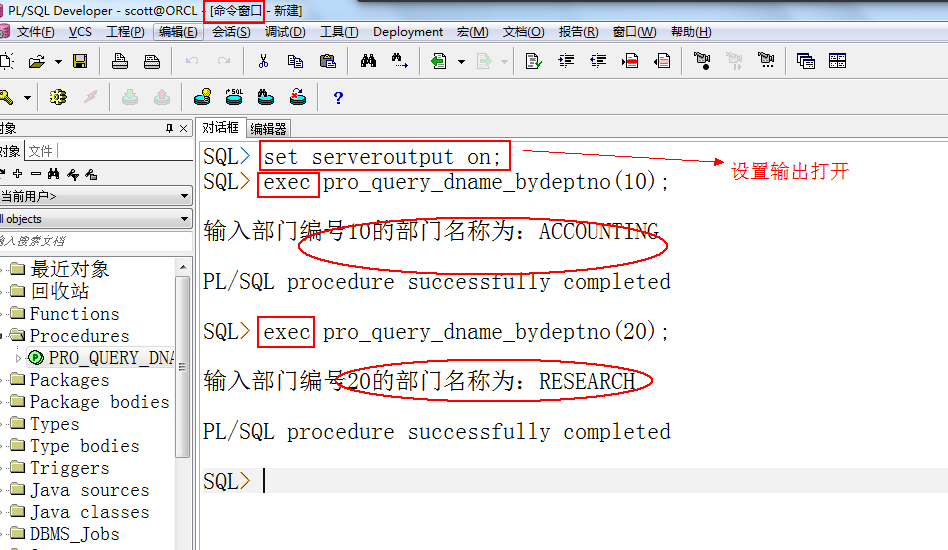
3） 自己编写匿名块调用

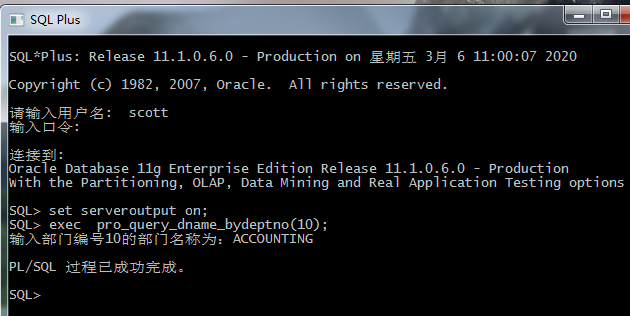


4）使用exec命令窗口调用

set serveroutput on

exec 存储过程名称(参数)





**2，存储过程 (讲解if elsif用法 带流程控制，带入参和出参)**

员工奖金发放：

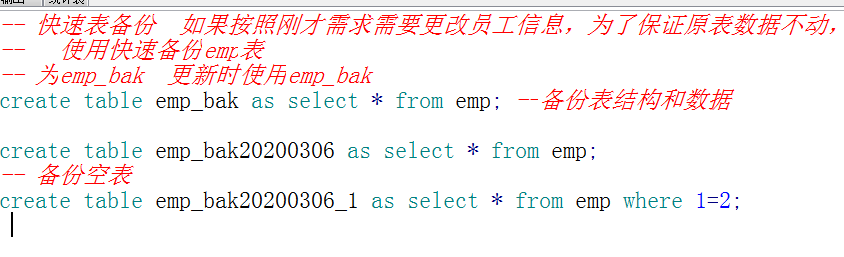
输入员工编号，如果该员工

原来没有奖金，则按照工资的10%发放

原来有奖金但不超过1000的，补到1000；

其余的按照原来奖金基础再加上10%发放；

更新成功后,返回更新后的奖金;



/\* 员工奖金发放：

输入员工编号，如果该员工

原来没有奖金，则按照工资的10%发放

原来有奖金但不超过1000的，补到1000；

其余的按照原来奖金基础再加上10%发放；

更新成功后,返回更新后的奖金;

if 判断条件 then

处理业务

end if;

if 判断条件 then

处理业务

elsif 判断条件 then

处理业务

elsif 判断条件 then

处理业务

......

else

处理业务

end if;

\*/

create or replace procedure pro\_update\_comm\_by\_empno

(i\_empno number,o\_comm\_info out varchar2)

as -- is 也可以

-- 定义部分

t\_comm emp.comm%type; --临时奖金变量

t\_updated\_comm emp.comm%type;

begin

-- 根据员工编号查询该员工原来的奖金 赋值

select comm into t\_comm from emp\_bak where empno=i\_empno;

-- 打印

dbms\_output.put\_line('编号为'||i\_empno||'员工的原奖金为：'||t\_comm);

-- 条件判断 java中= 赋值操作 在oracle 作为判断条件时，可以使用=

if t\_comm is null or t\_comm=0 then

-- 更新该员工的奖金为原来工资的%10

update emp\_bak set comm=0.1\*sal where empno=i\_empno;

elsif t\_comm<1000 then --与java语法不一样

-- 更新该员工的奖金为1000

update emp\_bak set comm=1000 where empno=i\_empno;

else

-- 更新为员工的奖金为员工奖金的1.1倍

update emp\_bak set comm=1.1\*comm where empno=i\_empno;

end if;

-- 提交事务

commit;

-- 查询更新后的奖金，返回

select comm into t\_updated\_comm from emp\_bak where empno=i\_empno;

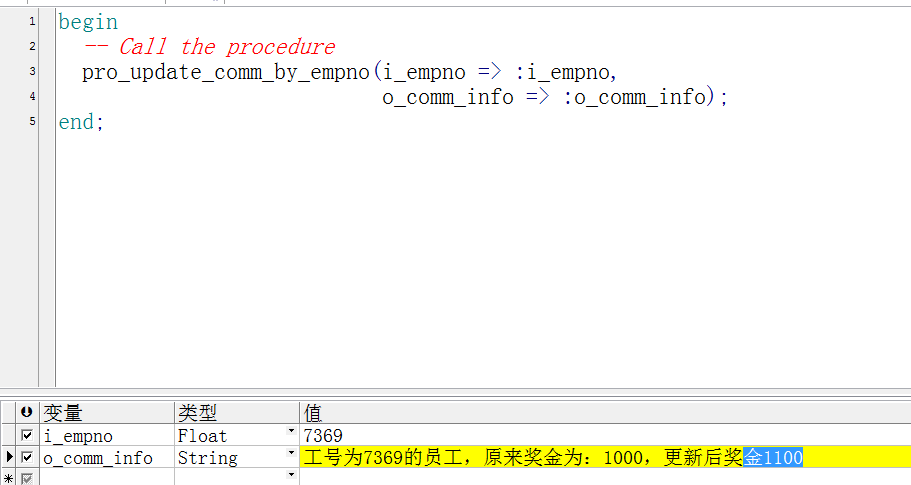
-- 给出参赋值

o\_comm\_info:='工号为'||i\_empno||'的员工，原来奖金为：'||nvl(t\_comm,0)||'，更新后奖金'

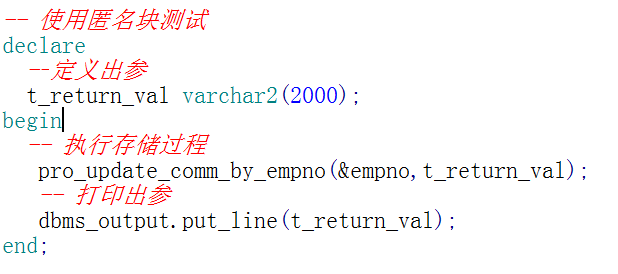
||t\_updated\_comm;

end;

**测试结果**



使用匿名块调用：



**3，存储过程 (讲解case when ...then ... else end case用法 带流程控制，带入参和出参)**

-- 根据员工编号输出员工工资级别（1000以下D ，1000-2000 C ，2000以上B,3000 以上为A）

/\*

case when 条件判断 then

执行业务

when 条件判断 then

执行业务

......

else

执行业务

end case;

\*/

create or replace procedure pro\_query\_sal\_level\_byempno

(i\_empno in number,o\_sallevel\_info out varchar2)

is

t\_sal emp.sal%type;

t\_sal\_level varchar2(1);

begin

-- 查询员工工资 赋值给变量

select sal into t\_sal from emp where empno=i\_empno;

-- 判断 （使用case when ） 和if elsif... 一样，满足第一个条件后返回

case when t\_sal<1000 then

t\_sal\_level := 'D';

when t\_sal<2000 then

t\_sal\_level:='C';

when t\_sal<3000 then

t\_sal\_level:='B';

else

t\_sal\_level:='A';

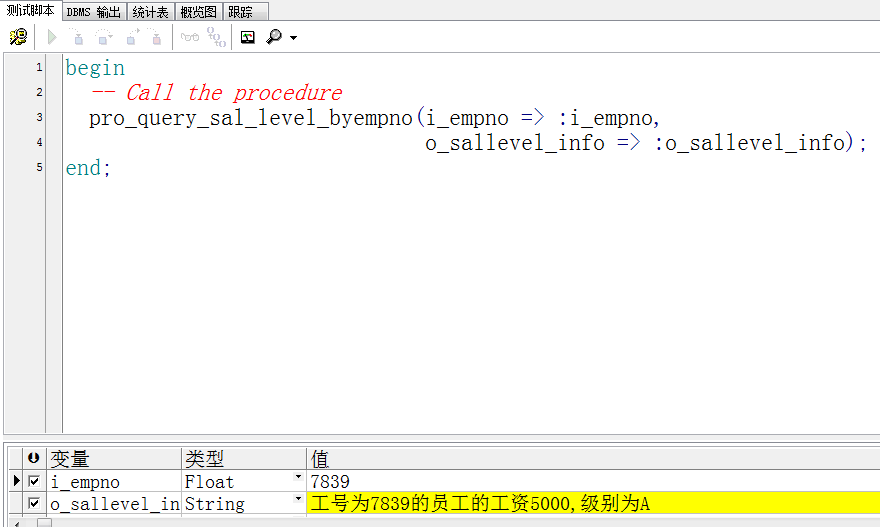
end case;

-- 给出参赋值

o\_sallevel\_info:='工号为'||i\_empno||'的员工的工资'||t\_sal||',级别为'||t\_sal\_level;

end;

**测试：**



4 ,存储过程（循环用法,使用不同的循环向dept的备份表插入数据）

-- 创建dept的备份表

create table dept\_bak as select \* from dept where 'a'='b';

select \* from dept\_bak;

-- 插入数据时要求，存储过程有2个参数（插入数据条数，返回成功插入提示）

-- 练习循环的循环

-- 基本循环语法 loop 判断退出 执行循环业务 end loop;

create or replace procedure pro\_batch\_insert\_dept\_a

(i\_add\_count number,o\_exec\_result out varchar2)

is

-- 定义变量 用作循环变量 默认开始值为1

t\_i number(4):=1;

begin

-- 判断输入参数

if i\_add\_count <=0 then

-- 错误提示

-- dbms\_output.put\_line('输入的添加数量不能小于1');

o\_exec\_result:='输入的添加数量不能小于1';

-- 直接返回，让程序不再继续执行

return;

end if;

-- 删除原有数据

delete from dept\_bak;

-- 循环插入数据

loop

-- 根据判断条件，让循环退出，没有的话就是死循环

if t\_i>i\_add\_count then

exit;-- 退出 相当于java中的break跳出循环 （continue 终止本次循环，进行下次）

end if;

-- 插入数据 （执行业务）

insert into dept\_bak values(t\_i,'devTeam'||t\_i,t\_i||'floor');

-- 让循环变量自增

t\_i:=t\_i+1;

end loop;

-- 退出循环后，提交事务

commit;

-- 返回插入成功提示

o\_exec\_result:='成功插入'||(t\_i-1)||'条';

end;

-- 查看

select \* from dept\_bak order by deptno;

-- while 循环用法

create or replace procedure pro\_batch\_insert\_dept\_b

(i\_add\_count number,o\_exec\_result out varchar2)

is

-- 定义变量 用作循环变量 默认开始值为1

t\_i number(4):=1;

begin

-- 判断输入参数

if i\_add\_count <=0 then

-- 错误提示

-- dbms\_output.put\_line('输入的添加数量不能小于1');

o\_exec\_result:='输入的添加数量不能小于1';

-- 直接返回，让程序不再继续执行

return;

end if;

-- 删除原有数据

delete from dept\_bak;

-- 使用while循环插入数据

while t\_i<=i\_add\_count loop -- 满足while判断条件才可以循环

-- 判断退出可以不要，满足条件才循环

-- 插入数据 （执行业务）

insert into dept\_bak values(t\_i,'devTeam'||t\_i,t\_i||'floor');

-- 让循环变量自增

t\_i:=t\_i+1;

end loop;

-- 提交事务

commit;

-- 返回成功提示

o\_exec\_result:='成功插入'||(t\_i-1)||'条';

end;

-- for 循环用法

create or replace procedure pro\_batch\_insert\_dept\_c

(i\_add\_count number,o\_exec\_result out varchar2)

is

begin

-- 判断输入参数

if i\_add\_count <=0 then

-- 错误提示

-- dbms\_output.put\_line('输入的添加数量不能小于1');

o\_exec\_result:='输入的添加数量不能小于1';

-- 直接返回，让程序不再继续执行

return;

end if;

-- 删除原有数据

delete from dept\_bak;

-- 使用while循环插入数据 (for(int i=0;i<arr.length;i++)){}

-- 1..i\_add\_count i\_add\_count=5 1..5 [1,2,3,4,5]

for t\_i in 1..i\_add\_count loop -- 在1到i\_add\_count之间每次取一个值

-- 插入数据 （执行业务）

insert into dept\_bak values(t\_i,'devTeam'||t\_i,t\_i||'floor');

end loop;

-- 提交事务

commit;

-- 返回成功提示

o\_exec\_result:='成功插入'||i\_add\_count||'条';

end;

**3.2 函数**

**3.2.1 概念：**

函数用于返回特定值，在实际应用中，如果一个语句的集合，完成特定功能后需要返回一个特定值，就可以使用函数。

**3.2.2 作用：**

如同 存储过程一样。

**3.2.3 语法：**

create or replace function 函数名称(arg1 [mode] datatype,arg2 [mode] datatype......)

return datatype -- 带返回值 datatype数据类型，根据实际需求，不同函数返回值类型不同（返回值不能加长度 varchar2 number date....）

is/as

-- 定义部分

begin

-- 执行部分

exception

-- 异常处理

end;

mode = in 入参 out 出参 参数类型

**3.2.4 实例**

-- 实例1

-- 根据函数的入参员工的职位job，返回该职位的平均工薪sal

select job,round(avg(sal+nvl(comm,0)),2) as 平均工资 from emp

group by job ;

select \* from emp;

-- oracle 语法严谨，在group by语句中，select中出现列，

-- 必须在group by后面出现，除非你是聚合函数

select deptno,job,round(avg(sal+nvl(comm,0)),2) as 平均工资 from emp

group by job; -- 语法错误

select min(deptno),job,round(avg(sal+nvl(comm,0)),2) as 平均工资 from emp

group by job; -- min max avg sum count....聚合 min(deptno) 在这地方没有实际意义

-- 函数

create or replace function fun\_query\_avgsalary\_byjob

(i\_job in varchar2)

return varchar2

is -- /as

-- 定义部分

t\_avg\_salary number(8,2);

begin

-- 执行部分

select avg(sal) into t\_avg\_salary from emp where job=i\_job;

return '输入职位的'||i\_job||'的平均工资为:'||t\_avg\_salary;

exception

-- 异常处理

when no\_data\_found then -- no\_data\_found 预定义异常

return '输入职位的'||i\_job||'不存在！';

end;

-- 函数的调用

declare

t\_result varchar2(200);

begin

-- 类似java中的方法调用

t\_result:=fun\_query\_avgsalary\_byjob('SALESMAN');

-- 打印

dbms\_output.put\_line(t\_result);

end;

**-- 实例2 根据部门名称查询部门人数（带出参返回字符串信息，带返回值直接为部门人数）**

-- 根据部门名称查询部门人数（带出参返回字符串信息，带返回值直接为部门人数）

-- 查询部门人数显示部门名称和该部门人数

select d.dname,count(\*) from dept d join emp e

on d.deptno=e.deptno group by d.dname;

-- 查询部门人数显示部门名称和该部门人数 没有员工的部门也显示

-- count(column) column 为空不计数 按人数倒序排序

select d.dname,count(e.empno) from dept d left join emp e

on d.deptno=e.deptno group by d.dname order by count(e.empno) desc;

-- 查询指定名称的人数

select d.dname,count(e.empno) from dept d left join emp e

on d.deptno=e.deptno group by d.dname having d.dname='SALES';

--order by count(e.empno) desc; -- 没有意义

-- 获取部门人数排名前2名的部门

select rownum,t.\* from (

select d.dname,count(e.empno) from dept d left join emp e

on d.deptno=e.deptno group by d.dname order by count(e.empno) desc) t

where rownum<3;

-- 函数实例2

create or replace function fun\_query\_deptnum\_by\_dname

(i\_dname in varchar2,o\_result out varchar2)

return number

as

-- 定义变量

t\_deptno dept.deptno%type;

t\_dept\_num number(3);

begin

-- 根据部门名称查询部门编号

select deptno into t\_deptno from dept where dname=i\_dname;

-- 根据查询的部门编号 查询部门人数

select count(\*) into t\_dept\_num from emp where deptno=t\_deptno;

-- 给出参赋值

o\_result:='名称为'||i\_dname||'的部门人数为：'||t\_dept\_num;

return t\_dept\_num;

exception

when no\_data\_found then

o\_result:='名称为'||i\_dname||'不存在';

return -1;

end;

-- 第2种写法，使用 group by +having

create or replace function fun\_query\_deptnum\_by\_dname\_a

(i\_dname in varchar2,o\_result out varchar2)

return number

as

-- 定义变量

t\_dept\_num number(3);

begin

-- 根据部门名称直接查询部门人数

select count(e.empno) into t\_dept\_num from dept d left join emp e

on d.deptno=e.deptno group by d.dname having d.dname=i\_dname;

-- 给出参赋值

o\_result:='名称为'||i\_dname||'的部门人数为：'||t\_dept\_num;

return t\_dept\_num;

exception

when no\_data\_found then

o\_result:='名称为'||i\_dname||'不存在';

return -1;

end;

**3.3 函数和存储过程有什么异同**

**相同：**

1,都可以有in和out模式传递或者返回参数

2,输入参数都可以接受默认值,都可以传值

3,语法相似，都有定义部分，执行部分和异常处理部分

**不同点：**

一般如果业务需要有多个返回值或者没有返回值时建议使用存储过程，如果是只返回一个参数时，建议使用函数。

如果返回多个或者不返回值，使用了函数，就是编程中的坏味道（不是错误，不好的编程习惯）。

一般处理大量业务时，使用存储过程，如果是小功能，或者小计算或者工具函数（类似java）建议使用函数。

**3.4 包**

**3.4.1 概念**

是oracle一种对象，讲逻辑上的oracle对象（存储过程，函数等等）组合成一个更大逻辑对象，可以有变量，常量，存储过程，函数，等等。。。

**3.4.2 作用**

类似JAVA中的包，在一个大的应用系统开发过程中，要使用很多模块，需要使用大量类似于存储过程和函数这样的对象，使用包可以让按照模块分门别类，更好对我们程序进行管理和维护。

com.aaa.demo1.A a = new A();

com.aaa.demo2.A a = new A();

**3.4.3 语法**

**包的头部(类似于java中的接口，只定义规范，有方法名称参数返回值，没有具体的实现)**

create or replace package 包的名称

is

end 包的名称;

**包的身体（类似于java中接口的实现类，除了要遵循接口中的实现规范之外，加上具体的实现）**

create or replace package body 包的名称

end 包的名称;

**3.4.4 实现**

-- 包的头部(接口)

/\*\*

讲课包示例a

\*/

create or replace package pkg\_demo\_a

is

/\*根据职位名称查询该职位的平均工资

i\_job 职位名称

return 平均工资信息

\*/

function fun\_query\_avgsalary\_byjob(i\_job in varchar2) return varchar2;

-- 批量插入dept表的数据

procedure pro\_batch\_insert\_dept\_a (i\_add\_count number default 3,o\_exec\_result out varchar2);

end pkg\_demo\_a;

-- 包的身体（实现类）

create or replace package body pkg\_demo\_a

is

/\*根据职位名称查询该职位的平均工资

i\_job 职位名称

return 平均工资信息

\*/

function fun\_query\_avgsalary\_byjob(i\_job in varchar2) return varchar2

is -- /as

-- 定义部分

t\_avg\_salary number(8,2);

begin

-- 执行部分

select avg(sal) into t\_avg\_salary from emp where job=i\_job;

return '输入职位的'||i\_job||'的平均工资为:'||t\_avg\_salary;

exception

-- 异常处理

when no\_data\_found then -- no\_data\_found 预定义异常

return '输入职位的'||i\_job||'不存在！';

end;

-- 批量插入dept表的数据

procedure pro\_batch\_insert\_dept\_a(i\_add\_count number default 3,o\_exec\_result out varchar2)

is

-- 定义变量 用作循环变量 默认开始值为1

t\_i number(4):=1;

begin

-- 判断输入参数

if i\_add\_count <=0 then

-- 错误提示

-- dbms\_output.put\_line('输入的添加数量不能小于1');

o\_exec\_result:='输入的添加数量不能小于1';

-- 直接返回，让程序不再继续执行

return;

end if;

-- 删除原有数据

delete from dept\_bak;

-- 循环插入数据

loop

-- 根据判断条件，让循环退出，没有的话就是死循环

if t\_i>i\_add\_count then

exit;-- 退出 相当于java中的break跳出循环 （continue 终止本次循环，进行下次）

end if;

-- 插入数据 （执行业务）

insert into dept\_bak values(t\_i,'devTeam'||t\_i,t\_i||'floor');

-- 让循环变量自增

t\_i:=t\_i+1;

end loop;

-- 退出循环后，提交事务

commit;

-- 返回插入成功提示

o\_exec\_result:='成功插入'||(t\_i-1)||'条';

end;

end pkg\_demo\_a;

测试：

-- 使用匿名块测试

declare

t\_result varchar2(200);

begin

-- 执行函数，赋值给结果变量

t\_result:= pkg\_demo\_a.fun\_query\_avgsalary\_byjob('CLERK');

-- 打印

dbms\_output.put\_line(t\_result);

end;

-- 测试存储过程

declare

t\_result varchar2(200);

begin

-- 执行存储过程，赋值给结果变量

pkg\_demo\_a.pro\_batch\_insert\_dept\_a(38,t\_result);

-- 打印

dbms\_output.put\_line(t\_result);

end;

**4、本章总结**

**4.1 总结本章知识点**

**4.2 面试题答案**

**4.3 预习下一章重点**