

plsql编程

一：定义

plsql

Procedure Language 面向过程编程,java oop iop aop sop

无非三点：变量、判断、循环

二：变量

```
1 create table t_stu3(  
2     id int primary key,  
3     name varchar2(30)  
4 );  
5 select * from t_stu3;  
6 --pl中的dml语句运行在一个事务中，执行成功但没有提交,失败会回滚  
7 begin  
8     insert into t_stu3 values(5,'mike');  
9     insert into t_stu3  
10    values(6,'rosedffffffffffffffffffffffffffffffffffffffffffffffff');  
11    --commit;  
12 end;
```

```
1 --第一次开启客户端，需要启动输出命令开启服务,用于输出对象输出值  
2 set serveroutput on;  
3 declare  
4     --定义变量、常量并赋值  
5     --i number(2):=1;  
6     i number;  
7     j number:=9;  
8     --i:=3;--这是执行语句  
9     flag boolean:=true;  
10  
11 begin  
12     --执行语句  
13     --i:=3;  
14  
15     --输出结果  
16     dbms_output.put_line(i+j);  
17     --dbms_output.put_line(flag);--无法输出布尔类型  
18 end;
```

dbms database manage system

dbms_output 是oracle中内置的一个应用程序模块，专门用于控制输出

```

1 declare
2     i number:=&请输入一个值;--数值接收
3     j number:=9;
4     name varchar(10):='&请输入姓名';
5     flag boolean:=true;
6
7 begin
8     dbms_output.put_line(i+j);
9     dbms_output.put_line(name);
10    dbms_output.put_line(flag);--不能将布尔类型的值直接输出到控制台，需要自己定义函数
11
12 end;
13
14 declare
15     --布尔类型默认初始值为null,null在if逻辑判断语句中为假
16     flag boolean;
17 begin
18     --if flag then
19     --if flag=false then
20     if flag is null then
21         dbms_output.put_line('true');
22     end if;
23 end;
24
25 declare
26     --PI float:=3.14;
27     --PI float default 3.14;--这种赋值方式只在声明语句中适用
28     PI constant float default 3.14;--这是个常量
29 begin
30     --pi:=3.1415;
31     dbms_output.put_line(PI);
32 end;

```

三：判断

- if
- case

1、if

语法1:

if 条件表达式 then

.....

else

.....

end if;

```

1 declare
2     i int :=&请输入值;
3
4 begin
5     if i>=10 then
6         dbms_output.put_line('>=10');
7     else
8         dbms_output.put_line('<10');
9     end if;
10 end;

```

语法2:

if 条件表达式 then

elsif 条件表达式 then

....

else

.....

end if

```

1 declare
2     chr char(1):=lower('&请输入学生的成绩等级');
3 begin
4     if chr='a' then
5         dbms_output.put_line('优秀');
6     elsif chr='b' then
7         dbms_output.put_line('良好');
8     elsif chr='c' then
9         dbms_output.put_line('一般');
10    elsif chr='d' then
11        dbms_output.put_line('及格');
12    elsif chr='e' then
13        dbms_output.put_line('不及格');
14    else
15        dbms_output.put_line('未知');
16
17
18    end if;
19
20 end;

```

2、case

语法1:

case 变量

when 等值1 then 结果1;

when 等值2 then 结果2;

when 等值3 then 结果3;

when 等值4 then 结果4;

else 结果5;

end case;

```
1 declare
2     chr char(1):=lower('&请输入成绩等级');
3
4 begin
5     case chr
6         when 'a' then
7             dbms_output.put_line('优秀');
8         when 'b' then
9             dbms_output.put_line('良好');
10        when 'c' then
11            dbms_output.put_line('一般');
12        when 'd' then
13            dbms_output.put_line('及格');
14        when 'e' then
15            dbms_output.put_line('不及格');
16        else
17            dbms_output.put_line('未知');
18
19
20    end case;
21 end;
```

语法2:

case

when 非等值1 then 结果1;

when 非等值2 then 结果2;

when 非等值3 then 结果3;

when 非等值4 then 结果4;

else 结果5;

end case;

```
1 declare
2     score number:=&请输入学生的成绩;
3
4 begin
5     if score between 0 and 100 then
6     case
7         when score between 90 and 100 then
8             dbms_output.put_line('优秀');
9         when score between 80 and 89 then
10            dbms_output.put_line('良好');
11        when score between 70 and 79 then
12            dbms_output.put_line('一般');
13        when score between 60 and 69 then
14            dbms_output.put_line('及格');
15
16        else
17            dbms_output.put_line('不及格');
18
19
20    end case;
21    else
22        dbms_output.put_line('成绩值不在0-100之间');
23    end if;
24 end;
```

四：循环

1、loop

自己控制计数器和循环条件

```

1  --输出1-100
2  declare
3      i int default 1;
4
5  begin
6      loop
7          dbms_output.put_line(i);
8          if i=100 then
9              --退出循环
10             --exit;--类似于java中的break,退出当前循环,不是退出pl块
11             return;--退出当前pl块
12         end if;
13         --计数
14         i:=i+1;--oracle中没有i+=1,i++这种写法
15         --i++;
16
17     end loop;
18
19     dbms_output.put_line('end');
20
21 end;

```

2、for

```

1  --打印1-100
2  begin
3      for i in 1..100--注意: i值永远只能是1, -1, 不能修改自增值
4      loop
5          dbms_output.put_line(i);
6          --i:=i+3;--不能进行赋值操作
7      end loop;
8  end;
9
10 --打印100-50
11 begin
12     for i in reverse 1..100--注意: ..前后只能从小值到大值, 想-1, 加reverse关键字
13     loop
14         dbms_output.put_line(i);
15         if i=50 then
16             exit;
17         end if;
18     end loop;
19 end;

```

3、while

类似于java中的while,没有do while

```

1  --打印1-100
2  declare
3      i int :=1;
4  begin
5      while i<=100
6      loop
7          dbms_output.put_line(i);
8          i:=i+1;
9      end loop;
10 end;

```

```

1  --缓冲输出
2  begin
3      dbms_output.put(10);--缓冲输出，不换行
4      dbms_output.put(10);
5      dbms_output.new_line();--表示清空缓冲区,将值输出，同时换行
6
7      dbms_output.put(20);
8      dbms_output.put(20);
9      dbms_output.new_line();
10 end;
11
12 --九九乘法表
13 begin
14     --遍历列
15     for i in 1..9
16     loop
17         --遍历行(上三角)
18         for j in 1..i
19         loop
20             dbms_output.put(i||'*'||j||'='||i*j||' ');
21         end loop;
22         dbms_output.new_line();
23     end loop;
24
25 end;

```

```

1  --练习
2  --求100以内的质数
3  --质数: 只能被1或者其本身整除的数叫质数(素数),最小的质数为2
4  declare
5      flag boolean:=true;
6  begin
7      --除数
8      for num in 2..100
9      loop
10         --循环前初始化flag为true
11         flag:=true;
12         --被除数
13         for i in 2..num-1
14         loop
15             if mod(num,i)=0 then
16                 flag:=false;
17             end if;
18         end loop;
19         if flag then
20             dbms_output.put_line(num);
21         end if;
22     end loop;
23 end;

```

五：游标

1：基本概念

类似于java中的ResultSet

在oracle中，游标只能下行(获取下一行)，不能上行或者定位

2、游标的分类

- 单行游标---隐式游标，不需要声明就可以直接使用，自动引用名字---sql

insert update delete select into from where

- 多行游标---显示游标，必须在declare块中进行声明

3、游标中的属性

- found,notfound 布尔类型(有没有结果被影响,有, found=true,notfound=false)
- rowcount 整数(影响的行数)
- 这两个属性对单行和多行游标都适用


```

1
2 select * from emp;
3 set serveroutput on;
4 rollback;
5
6 declare
7
8 begin
9     delete from emp where deptno=20;--单行游标
10    if sql%found then --oracle中访问对象中的属性使用%
11        dbms_output.put_line('成功删除了'||sql%rowcount||'条记录');
12    else
13        dbms_output.put_line('没有匹配的记录被删除');
14    end if;
15 end;

```

4、指定列值变量的单行游标

```

1 select * from emp;
2 desc emp;
3
4 declare
5     eno number(4) :=&请输入雇员编号;
6     --这三个变量用于放入查询中对应的列值
7     --采用动态类型,指定变量类型和查询的字段类型一致
8     e_name emp.ename%type;
9     e_date emp.hiredate%type;
10    e_sal emp.sal%type;
11    cnt int;
12
13 begin
14    select count(*) into cnt from emp where empno=eno;
15    if cnt=1 then
16        select ename,hiredate,sal into e_name,e_date,e_sal from emp where empno=eno;
17        dbms_output.put_line(e_name||'的入职时间: '||to_char(e_date,'yyyy-mm-dd')||'薪水
是: '||e_sal);
18    else
19        dbms_output.put_line('没有找到查询结果');
20    end if;
21 end;

```

5、指定行值变量的单行游标

```

1  select * from emp;
2  desc emp;
3
4  declare
5      eno number(4) :=&请输入雇员编号;
6      e emp%rowtype;--定义emp表的行的类型
7      cnt int;
8
9  begin
10     select count(*) into cnt from emp where empno=eno;
11     if cnt=1 then
12         select * into e from emp where empno=eno;
13         dbms_output.put_line(e.ename||'的入职时间: '||to_char(e.hiredate,'yyyy-mm-dd')||'薪水是: '||e.sal);
14     else
15         dbms_output.put_line('没有找到查询结果');
16     end if;
17 end;
18

```

6、多行游标

需要在seclare块中进行声明

6.1、精简写法(不需要声明)

```

1  begin
2      for t in (select * from user_tables)
3      loop
4          dbms_output.put_line(t.table_name||' '||t.tablespace_name);
5      end loop;
6
7  end;

```

6.2、游标的数据结构

1、结构

```

--->BOF (表名开头)  begin  of file-->0
row1-->1
row2-->2
..
rown-->n
---->EOF (表的结束)end of file-->n
....          -->n

```

2、found/notfound

指的是当前游标是不是指向下一个游标的有效可读行，指向的是有效行，值为true, 否则为false
 游标指针指向的是BOF，found/notfound的值为null
 游标指针指向的是EOF，found值为false, notfound值为true

3、rowcount

```
BOF->0,1,2,3,4...N,EOF-->n
```

4、完整定义

注意：显示游标必须打开游标，才能访问游标中的属性

```

1 declare
2     --定义一个显示游标
3     cursor cur is select * from emp where deptno=10;
4     e emp%rowtype;
5 begin
6     --打开游标
7     open cur;
8     if cur%found is null then
9         dbms_output.put_line('found is null');
10    else
11        dbms_output.put_line('found is not null');
12    end if;
13    if cur%notfound is null then
14        dbms_output.put_line('notfound is null');
15    else
16        dbms_output.put_line('notfound is not null');
17    end if;
18    dbms_output.put_line(cur%rowcount);--0
19    --将指针向下移动，指向下一条记录
20    fetch cur into e;
21    if cur%found then
22        dbms_output.put_line('found is true');
23    else
24        dbms_output.put_line('found is not true');
25    end if;
26    if cur%notfound then
27        dbms_output.put_line('notfound is true');
28    else
29        dbms_output.put_line('notfound is not true');
30    end if;
31    dbms_output.put_line(cur%rowcount);--1
32    fetch cur into e;
33    dbms_output.put_line(cur%rowcount);--2
34    fetch cur into e;
35    dbms_output.put_line(cur%rowcount);--3
36    fetch cur into e;
37    dbms_output.put_line(cur%rowcount);--3 EOF
38    if cur%found then
39        dbms_output.put_line('found is true');
40    else
41        dbms_output.put_line('found is not true');--found is not true
42    end if;
43    if cur%notfound then
44        dbms_output.put_line('notfound is true');--notfound is true
45    else
46        dbms_output.put_line('notfound is not true');
47    end if;
48    --关闭游标
49    close cur;
50
51 end;

```

使用loop循环

```

1 declare
2     cursor cur is select * from emp where deptno=&请输入部门编号;
3     e emp%rowtype;
4 begin
5     open cur;
6     loop
7         fetch cur into e;
8         --判断游标指针是否已经指向EOF,如果指向了,不需要再向下遍历,直接退出
9         if cur%notfound then
10             exit;--退出循环
11         end if;
12         dbms_output.put_line(e.empno||' '||rpad(e.ename,10,'*')||' '||e.sal);
13
14
15     end loop;
16     close cur;
17
18 end;
19

```

7、课堂练习

1、练习1

求前100个质数

```

1 declare
2     cnt int :=0;    --个数
3     num int :=2;    --质数
4     flag boolean;  --判断该值是否为质数, true-->是质数
5 begin
6     while cnt <100-->循环100次
7     loop
8         flag:=true;
9         for i in 2..num-1 --除数
10        loop
11
12            if mod(num,i)=0 then --能被整除, 该值不是质数
13                flag:=false;
14                exit;
15            end if;
16        end loop;
17        if flag then
18            cnt:=cnt+1;
19            dbms_output.put_line(cnt||' '||num);
20        end if;
21        --将该值加1, 继续判断+1后的值是否为质数
22        num:=num+1;
23
24    end loop;
25
26 end;

```

2、练习2

从控制台输入一行字符串，判断这个字符串是不是回文！

如：

abccba

123454321

```
1
2
3 declare
4     str varchar(30):='&请输入一个字符串';
5     c1 char(1);
6     c2 char(1);
7     flag boolean:=true;
8 begin
9     for i in 1..trunc(length(str)/2)
10    loop
11        --获取对应的字符
12        c1:=substr(str,i,1);
13        c2:=substr(str,-i,1);
14        if c1<>c2 then
15            flag:=false;
16            exit;--相当于break,退出当前循环,继续下一次循环
17        end if;
18    end loop;
19    if flag then
20        dbms_output.put_line(str||'是回文');
21    else
22        dbms_output.put_line(str||'不是回文');
23    end if;
24
25
26 end;
```

3、练习3

求100~999之间的水仙花数

个位立方+十位立方+百位立方=这个三位数

153=1³+5³+3³

```

1  --百位
2  select trunc(123/100) from dual;
3  --十位
4  select trunc((156-i*100)/10) from dual;
5
6  --个位
7  select mod(123,10) from dual;
8
9  declare
10     x int;
11     y int;
12     z int;
13  begin
14     for i in 100..999
15     loop
16         x:=trunc(i/100);
17         y:=trunc((i-x*100)/10);
18         z:=mod(i,10);
19         if power(x,3)+power(y,3)+power(z,3)=i then
20             dbms_output.put_line(i);
21         end if;
22     end loop;
23
24  end;

```

4、练习4

求值的算术平方根

```

1  select round(sqrt(10),6) from dual;
2
3  declare
4     x int :=&求平方根;
5     --定义数列的两个变量
6     s1 number(10,6):=1;--定义前项
7     s2 number(10,6);--定义后项
8
9
10
11  begin
12     s2:=0.5*(s1+x/s1);
13     while abs(s2-s1)>=power(10,-6)
14     loop
15         s1:=s2;--项后移
16         s2:=0.5*(s1+x/s1);
17
18     end loop;
19     dbms_output.put_line(s1||' '||round(sqrt(x),6));
20
21  end;

```

5、练习5

如果雇员薪水低于1000 加10% 但是10号部门只加5%
如果雇员薪水1000~1999 加5% 但是10号部门只加3%
如果雇员薪水2000~2999 加3% 但是10号部门只加2%
如果雇员薪水超过3000 加1% 但是10号部门不加


```

1  select * from emp;
2
3  declare
4      cursor cur is select deptno,empno,sal from emp;
5      dno emp.deptno%type;
6      eno emp.empno%type;
7      esal emp.sal%type;
8      p float;
9      cnt int :=0;
10 begin
11     open cur;
12     loop
13         fetch cur into dno,eno,esal;
14         --判断何时退出遍历
15         if cur%notfound then
16             exit;
17         end if;
18         case dno
19             when 10 then
20                 if esal between 0 and 999 then
21                     p:=1.05;
22                 elsif esal between 1000 and 1999 then
23                     p:=1.03;
24                 elsif esal between 2000 and 2999 then
25                     p:=1.02;
26                 else
27                     p:=1;
28                 end if;
29             else
30                 if esal between 0 and 999 then
31                     p:=1.1;
32                 elsif esal between 1000 and 1999 then
33                     p:=1.05;
34                 elsif esal between 2000 and 2999 then
35                     p:=1.03;
36                 else
37                     p:=1.01;
38                 end if;
39
40             end case;
41             dbms_output.put_line('雇员编号: '||eno||',所在部门: '
42             ||dno||',薪水从'||esal||',加薪后变为>>>'||esal*p);
43
44             --将新值更新到数据库
45             update emp set sal=sal*p where empno=eno;
46             cnt:=cnt+sql%rowcount;
47
48         end loop;
49
50         dbms_output.put_line('成功更新了'||cnt||'条雇员的薪水信息');
51
52     close cur;
53

```

8、在pl块中执行DDL,DCL语句

在pl块中不能直接执行DDL,DCL语句，必须通过execute immediate 语句，动态的拼接执行语句。

```
1 begin
2     execute immediate 'create table t_stu4(id int)';
3 end;
```

注意：表名的长度是有大小限制的，最大**30**个字节

```
1 desc user_tables
2 名称                空值      类型
3  -----
4  TABLE_NAME        NOT NULL VARCHAR2(30)
5  TABLESPACE_NAME                VARCHAR2(30)
```

```
1 desc user_tables;
2 create table t_stu5aaaaaaaaaaaaaaaaabbbbbbbbbc(id int);--创建失败，表名长度最多30个字节
```

1、创建一张表

```
1 select * from user_tables;
2 select count(1)from user_tables where table_name=upper('t_113');
3 declare
4     tname varchar(30):='&请输入要创建的表名';
5     a int;
6 begin
7     select count(1) into a from user_tables where table_name=upper(tname);
8     if a=0 then
9         execute immediate 'create table '||tname||'(id int)';
10    else
11        dbms_output.put_line('该表'||tname||'已经存在，不能再次创建');
12    end if;
13 end;
```

2、删除自己名下所有的表

```

1 declare
2     cursor tbls is select table_name from user_tables;
3     tname user_tables.table_name%type;
4 begin
5     open tbls;
6     loop
7         fetch tbls into tname;
8         -- if tbls%notfound then
9         --     exit;
10        -- end if;
11        exit when tbls%notfound; --这种写法适用于循环中退出条件的判断
12        --dbms_output.put_line('drop table '||tname);
13        --主要：表名需要加上""
14        execute immediate 'drop table '||tname||'" cascade constraint';
15    end loop;
16
17    close tbls;
18 end;

```

3、练习

创建表，一定要遵守表的创建规范

--数据表，以t_开头

--备份表，以t_开头，以_bak结尾

需求：

删除备份表，再将数据表备份

```

1 select * from user_tables;
2
3 create table t_emp as select * from scott.emp;
4 create table t_emp_bak as select * from t_emp;
5 create table t_emp1 as select * from t_emp;
6 create table t_emp1_bak as select * from t_emp;
7 create table temp_emp1 as select * from t_emp;
8 create table s_emp as select * from t_emp;
9 temp_ --临时表
10 s_   统计表
11 t_   普通表
12 _bak 备份表

```

```

1 declare
2     --创建2个游标获取普通表和备份表
3     cursor baks is select table_name from user_tables where
4     regexp_like(table_name, '_bak$', 'i');
5     cursor tbls is select table_name from user_tables where
6     regexp_like(table_name, '^t_', 'i');
7     tname user_tables.table_name%type;
8 begin
9     --删除备份表
10    open baks;
11    loop
12        fetch baks into tname;
13        exit when baks%notfound;
14        execute immediate 'drop table ' || tname || ' cascade constraint';
15    end loop;
16
17    close baks;
18
19    --对普通表进行备份
20    open tbls;
21    loop
22        fetch tbls into tname;
23        exit when tbls%notfound;
24        execute immediate 'create table ' || tname || '_bak as select * from ' || tname || '';
25    end loop;
26    close tbls;
27 end;

```

9、带有参数的游标

```

1 create table emp as select * from t_emp;
2
3 declare
4     cursor cur(dno emp.deptno%type) is select * from emp where deptno=dno;
5     e emp%rowtype;
6 begin
7     open cur(&请输入部门编号);
8     loop
9         fetch cur into e;
10        exit when cur%notfound;
11        dbms_output.put_line(e.empno || ' ' || e.ename);
12    end loop;
13    close cur;
14
15 end;

```

10、使用for循环简化上面的代码

for循环的特点:

- 自动打开、关闭游标

- 自动判断游标是否结束
- 不需要进行遍历：fetch

```

1  --查询条件:
2  --1: 根据部门编号进行查询
3  --2: 根据雇员名称进行模糊查询
4  select * from emp where deptno=10 and ename like upper('%ar%');
5
6  declare
7      cursor cur(dno emp.deptno%type,en emp.ename%type) is select * from emp where deptno=dno
8      and ename like upper(en);
9      e emp%rowtype;
10 begin
11     --open cur();
12     for e in cur(&请输入部门编号,'%&姓名%')
13     loop
14         --fetch cur into e;
15         --exit when cur%notfound;
16         dbms_output.put_line(e.empno||' '||e.ename);
17     end loop;
18     --close cur;
19
20 end;
```

11、游标参数的使用问题

注意：参数的类型不能指定精度：number(4)--->number varchar(10)--->varchar

```

1  desc emp;
2
3  declare
4      cursor cur(dno NUMBER,en VARCHAR2) is select * from emp where deptno=dno
5      and ename like upper(en);
6      e emp%rowtype;
7  begin
8      --open cur();
9      for e in cur(&请输入部门编号,'%&姓名%')
10     loop
11         --fetch cur into e;
12         --exit when cur%notfound;
13         dbms_output.put_line(e.empno||' '||e.ename);
14     end loop;
15     --close cur;
16
17 end;
```

12、动态游标

1、应用场景

很多时候，在做业务时，具体要查询的语句，并不能直接在声明中确定，而是根据需求动态的去执行查询语句

2、需求

根据雇员编号查询雇员的信息

如果是10号部门，要求查询雇员的以下信息（姓名、薪水、入职时间）

如果是20,30号部门，要求查询雇员的以下信息（姓名、薪水、入职时间、奖金、所在部门名称）

注意：动态游标不支持for循环，只能使用loop进行遍历

```

1  select * from emp;
2  create table dept as select * from scott.dept;
3
4  --如果是10号部门，要求查询该雇员的下属的以下信息（姓名、薪水、入职时间）
5  --select e.ename,e.sal,e.hiredate
6  --from emp e,emp m where e.mgr=m.empno
7  --and m.empno=7839;
8  --另一种写法
9  select e.ename,e.sal,e.hiredate
10 from emp e
11 where e.mgr=7839;
12
13 --如果是20,30号部门，要求查询该部门下雇员的以下信息（姓名、薪水、入职时间、奖金、所在部门名称）
14 select e.ename,e.sal,e.hiredate,e.comm,d.dname
15 from emp e,dept d
16 where e.deptno=d.deptno
17 and e.empno=7369;
18
19
20 declare
21 --声明动态游标(在begin end块中指定查询语句)
22 cur sys_refcursor;
23 eno emp.empno%type:=&雇员编号;
24 dno emp.deptno%type;
25 e_name emp.ename%type;p.deptno%type;;e_sal emp.sal%type;;e_date
emp.hiredate%type;;e_comm emp.comm%type;;d_name dept.dname%type;
26 e_sal emp.sal%type;
27 e_date emp.hiredate%type;
28 e_comm emp.comm%type;
29 d_name dept.dname%type;
30
31 begin
32 --根据雇员编号获取对应的部门编号
33 select deptno into dno from emp where empno=eno;
34 if dno=10 then
35     open cur for select e.ename,e.sal,e.hiredate
36                     from emp e
37                     where e.mgr=eno;
38     loop
39         fetch cur into e_name,e_sal,e_date;
40         exit when cur%notfound;
41         dbms_output.put_line(e_name||' '||e_sal||' '||e_date);
42     end loop;
43     close cur;
44
45 else
46     open cur for select e.ename,e.sal,e.hiredate,e.comm,d.dname
47                     from emp e,dept d
48                     where e.deptno=d.deptno
49                     and e.empno=eno;
50     loop
51         fetch cur into e_name,e_sal,e_date,e_comm,d_name;
52         exit when cur%notfound;

```

```

53         dbms_output.put_line(e_name||' '||e_sal||' '||e_date||' '||e_comm||'
    '||d_name);
54     end loop;
55     close cur;
56 end if;
57
58 end;
```

六：异常

1、编号

- 内置异常编号：-19999~0
- 自定义异常编号：-20000~-20999

2、异常函数

sqlerrm(编号)-----返回异常信息

如果在pl中，不传递异常编号，直接输出sqlerrm,那么就返回当前发生的异常消息。

3、打印所有的异常消息

```

1 declare
2
3 begin
4     for errcode in -20000..0
5     loop
6         dbms_output.put_line(sqlerrm(errcode));
7     end loop;
8 end;
```

4、捕获异常

pl块中是在代码的最后捕获异常信息


```

1  --no_data_found:没有返回行
2  --too_many_rows:返回多行
3  select * from emp;
4  insert into emp(empno,ename)values(7788,'mike');
5  declare
6      e_name emp.ename%type;
7  begin
8      select ename into e_name from emp where empno=&请输入雇员编号;
9
10     exception
11     --无返回行异常
12     when no_data_found then
13         dbms_output.put_line('出错了');
14         dbms_output.put_line(sqlerrm);
15     --多行异常
16     when too_many_rows then
17         dbms_output.put_line('出错了>>');
18         dbms_output.put_line(sqlerrm);
19     when others then
20         dbms_output.put_line('出错了>>>>');
21         dbms_output.put_line(sqlerrm);
22 end;

```

5、自定义异常

1、方式1

在declare块中定义一个exception类型的变量，然后在pl块中通过raise 抛出该异常

```

1  --查询用户的薪水，看输入的薪水
2  --如果降薪，抛出不能降薪异常
3  --如果涨薪幅度超过10%，抛出涨薪幅度不能超过10%异常
4  declare
5      less0 exception;
6      plus10per exception;
7      e_sal emp.sal%type:=&加薪值;
8      s_sal emp.sal%type;--原始薪水值
9      e_no emp.empno%type:=&雇员编号;
10 begin
11     --检查是否降薪
12     if e_sal < 0 then
13         raise less0;
14     end if;
15     --检查加薪值是否超过10%
16     select sal into s_sal from emp where empno=e_no;
17     if (s_sal+e_sal)/s_sal>1.1 then
18         raise plus10per;
19     end if;
20     --dbms_output.put_line('ok');
21     update emp set sal=sal+e_sal where empno=e_no;
22     dbms_output.put_line('成功更新');
23
24     exception
25         when no_data_found then dbms_output.put_line('查无此人');
26         when too_many_rows then dbms_output.put_line('返回多行');
27         when less0 then dbms_output.put_line('不能降薪');
28         when plus10per then dbms_output.put_line('加薪幅度不能超过10%');
29 end;

```

2、方式2

直接通过raise_application_error(异常编号,异常描述)抛出

```

1  --查询用户的薪水，看输入的薪水
2  --如果降薪，抛出不能降薪异常
3  --如果涨薪幅度超过10%，抛出涨薪幅度不能超过10%异常
4  declare
5      e_sal emp.sal%type:=&加薪值;
6      s_sal emp.sal%type;--原始薪水值
7      e_no emp.empno%type:=&雇员编号;
8  begin
9      --检查是否降薪
10     if e_sal < 0 then
11         raise_application_error(-20001, '不能降薪');
12     end if;
13     --检查加薪值是否超过10%
14     select sal into s_sal from emp where empno=e_no;
15     if (s_sal+e_sal)/s_sal>1.1 then
16         raise_application_error(-20002, '加薪值是否超过10%');
17     end if;
18     --dbms_output.put_line('ok');
19     update emp set sal=sal+e_sal where empno=e_no;
20     dbms_output.put_line('成功更新');
21
22     exception
23         when no_data_found then dbms_output.put_line('查无此人');
24         when too_many_rows then dbms_output.put_line('返回多行');
25         when others then
26             dbms_output.put_line(sqlerrm);
27 end;

```

七：函数

1、定义

函数可以在查询或者pl块中使用,一定要有返回值，一定要去接收结果

```

1  select length('abc') from dual;
2  declare
3      len int;
4  begin
5      --length('abc');--不能直接执行，一定要接收结果
6      len:=length('abc');--接收该结果
7
8      --dbms_output.put_line(length('abc'));
9      dbms_output.put_line(len);
10 end;
11

```

```

1  create function plus(i int,j int) return int
2  is
3      --变量的声明
4      z int;
5  begin
6      z:=i+j;
7      return z;
8  end;
9  select length('abc') from dual;
10 select plus(45,56) from dual;
11 select nvl(sal,0)sal,nvl(comm,0)comm,plus(nvl(sal,0),nvl(comm,0))aa from emp;
12
13
14 declare
15     i int :=1;
16     j int :=2;
17     k int;
18 begin
19     plus(i,j);--不能直接执行
20     --dbms_output.put_line(plus(i,j));
21 end;

```

2、创建函数的授权

create procedure

```

1  SQL> select * from dba_sys_privs where grantee=upper('resource');--dba权限
2
3  GRANTEE                                PRIVILEGE                                ADM
4  -----                                -
5  RESOURCE                                CREATE TRIGGER                            NO
6  RESOURCE                                CREATE SEQUENCE                            NO
7  RESOURCE                                CREATE TYPE                                NO
8  RESOURCE                                CREATE PROCEDURE                           NO
9  RESOURCE                                CREATE CLUSTER                             NO
10 RESOURCE                                CREATE OPERATOR                             NO
11 RESOURCE                                CREATE INDEXTYPE                             NO
12 RESOURCE                                CREATE TABLE                               NO

```

- 函数执行授权:grant execute on plus to mike1;

在mike_账号下执行: grant execute on plus to mike1;

```

1 SQL> conn mike_/mike_
2 已连接。
3 SQL> grant execute on plus to mike1;
4
5 授权成功。
6
7 SQL> conn mike1/abc
8 已连接。
9 SQL> select mike_.plus(1,2) from dual;
10
11 MIKE_.PLUS(1,2)
12 -----
13                3

```

- 函数执行授权: revoke execute on plus from mike1 --mike_账号

```

1 SQL> conn mike_/mike_
2 已连接。
3 SQL> revoke execute on plus from mike1;
4
5 撤销成功。
6
7 SQL> conn mike1/abc
8 已连接。
9 SQL> select mike_.plus(1,2) from dual;
10 select mike_.plus(1,2) from dual
11                *
12 第 1 行出现错误:
13 ORA-00904: : 标识符无效

```

3、函数的参数、返回值不能设置精度

```

1 drop table t1;
2 create or replace table t1(id int);--没有这种写法, 删除表只能用drop命令
3
4 create or replace function plus(i number,j number) return int(3)
5 is
6     z int;
7 begin
8     z:=i+j;
9     return z;
10 end;
11
12 --在sqldeveloper工具中, 可以通过编译器工具查看, 或者执行show error命令显示错误信息
13 --在控制台, 如何运行pl块, 需要/加回车

```

4、数据字典表

```

1 select * from user_functions;--没有这个数据字典表
2 select * from user_objects where object_type=upper('table');--查询当前账号下的所有表
3 select * from user_objects where object_type=upper('sequence');--查询当前账号下的所有序列
4 select * from user_objects where object_type=upper('function');--查询当前账号下的所有函数

```

5、练习

编写一个函数：输入参：雇员编号，返回值：雇员姓名

```

1 create or replace function getEmpNameByNo(eno emp.empno%type) return emp.ename%type
2 is
3     e_name emp.ename%type;
4 begin
5     select ename into e_name from emp where empno=eno;
6     return e_name;
7
8     exception
9         when no_data_found then return '未找到记录';
10        when too_many_rows then return '返回多行';
11        when others then return '执行出错';
12 end;
13 --from dual(参数是一个明确的值)
14 select getEmpNameByNo(7369) from dual;
15 --from 表(参数是该表中的某列)
16 select getEmpNameByNo(empno) from emp where empno=7369;
17
18 select to_char(sysdate,'yyyy') from dual;
19 select to_char(hiredate,'yyyy') from emp;

```

编写一个函数：输入参：雇员编号，返回值：雇员信息行

```

1  create or replace function getEmpByNo(eno emp.empno%type) return emp%rowtype
2  is
3      e emp%rowtype;
4  begin
5      select * into e from emp where empno=eno;
6      return e;
7
8      exception
9          when others then return null; -- 只能返回一个对象
10 end;
11
12 --通过select查询
13 --如果函数返回结果是rowtype,那么不能通过select进行查询
14 --select getEmpByNo(7369) from dual;
15 --只能使用pl块进行查询
16 declare
17     e emp%rowtype;
18     eno emp.empno%type:=&请输入雇员编号;
19 begin
20     e:=getEmpByNo(eno);
21     --检查出错
22     --if e is null then
23     --通过访问rowtype行的某个属性,一般情况下用主键属性,判断rowtype是否有结果
24     if e.ename is null then
25
26         dbms_output.put_line('查询出错');
27     else
28         dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);
29     end if;
30
31 end;

```

6、定义无参函数

```

1  select sysdate from dual;
2
3  --定义无参函数
4  create or replace function hello return varchar
5  is
6  begin
7      return 'hello itany';
8  end;
9
10 select hello() from dual; -- 自己定义的函数,是可以+括号的
11
12 select user from dual; -- 系统无参函数不能加括号

```

7、返回游标类型的函数(多行)

注意: 这里只能使用动态游标: **sys_refcursor**

```

1  --根据部门编号，返回该部门下的所有雇员信息
2  select * from emp where deptno=10;
3  set serveroutput on;
4  create or replace function getEmpsByDeptno(dno emp.deptno%type)
5  return sys_refcursor
6  is
7      emps sys_refcursor;
8      cnt int;
9  begin
10     select count(1) into cnt from emp where deptno=dno;
11     if cnt =0 then
12         return null;
13     else
14         open emps for select * from emp where deptno=dno;
15         return emps;
16     end if;
17
18
19     exception
20         when others then return null;
21
22 end;
23 --没有任何取其中内容的方法
24 select getEmpsByDeptno(10) from dual;
25
26 --返回动态游标的函数都是通过pl进行调用
27 declare
28     cur sys_refcursor;
29     e emp%rowtype;
30 begin
31     cur:=getEmpsByDeptno(&请输入部门编号);
32     loop
33         fetch cur into e;
34         exit when cur%notfound;
35         dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);
36
37     end loop;
38     close cur;
39     exception
40         when others then dbms_output.put_line('未找到记录');
41
42
43 end;

```

8、在JDBC中访问函数


```

1  @Test
2  //根据部门编号, 返回雇员姓名(方式1)
3  public void testFun1() throws Exception{
4      Class.forName("oracle.jdbc.driver.OracleDriver");
5      Connection conn = DriverManager.getConnection(
6          "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
7      PreparedStatement pstmt = conn.prepareStatement("select getEmpNameByNo(?) from dual");
8      pstmt.setInt(1,7369);
9      ResultSet rs = pstmt.executeQuery();
10     if(rs.next()){
11         System.out.println(rs.getString(1));
12     }
13     rs.close();
14     pstmt.close();
15     conn.close();
16
17 }
18 //根据部门编号, 返回雇员姓名(方式2)
19 @Test
20 public void testFun2() throws Exception{
21     Class.forName("oracle.jdbc.driver.OracleDriver");
22     Connection conn = DriverManager.getConnection(
23         "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
24     //预定义函数的调用格式
25     CallableStatement call = conn.prepareCall("{?=call getEmpNameByNo(?) }");
26     //声明函数各个部分的类型和参数说明
27     //注册返回值参的类型
28     call.registerOutParameter(1,Types.VARCHAR);
29     //设置普通参数
30     call.setInt(2,7369);
31     //执行
32     call.execute();
33     String ename = call.getString(1);//获取第一个参数的返回值
34     System.out.println(ename);
35     call.close();
36     conn.close();
37
38 }
39
40 //如果返回值是rowtype, 目前jdbc不支持
41 //返回值是动态游标
42 @Test
43 public void testFun3() throws Exception{
44     Class.forName("oracle.jdbc.driver.OracleDriver");
45     Connection conn = DriverManager.getConnection(
46         "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
47     //预定义函数的调用格式
48     CallableStatement call = conn.prepareCall("{?=call getEmpsByDeptno(?) }");
49     //声明函数各个部分的类型和参数说明
50     //注册返回值参的类型
51     call.registerOutParameter(1,OracleTypes.CURSOR);
52     //设置普通参数
53     call.setInt(2,10);

```

```

54     call.execute();
55
56     try {
57         Object obj = call.getObject(1);
58         ResultSet rs=(ResultSet)(obj);
59         while(rs.next()){
60             System.out.println(rs.getInt("empno")+"\t"+rs.getString("ename")+
61                                 "\t"+rs.getFloat("sal"));
62         }
63     } catch (Exception e) {
64         // TODO Auto-generated catch block
65         //e.printStackTrace();
66         System.out.println("查询出错");
67     }
68     call.close();
69     conn.close();
70 }

```

9、总结

JDBC访问函数的几种类别

1. 如果你的函数返回查询结果的值,可以使用select,pl ,jdbc
2. 如果函数的返回值是rowtype,jdbc不支持, 可以使用pl
3. 如果函数的返回值是动态游标 select,pl jdbc
4. 如果函数中有insert,update,delete等语句, 不能通过select查询,可以通过pl,jdbc调用

```

1  --如果函数中有insert,update,delete等语句,
2  --根据雇员编号, 给雇员加薪10%, 返回加薪后的值
3  select * from emp;
4
5  create or replace function updateSal(eno emp.empno%type)
6  return emp.sal%type
7  is
8      s emp.sal%type;
9  begin
10     update emp set sal=sal*1.1 where empno=eno;
11     select sal into s from emp where empno=eno;
12     return s;
13
14     exception
15         when no_data_found then return -1;
16         when too_many_rows then return -2;
17         when others then return -3;
18 end;
19
20 select * from emp;
21 select updateSal(7369) from dual;--error
22 --使用pl调用
23 declare
24     eno emp.empno%type:=&请输入雇员编号;
25     esal emp.sal%type;
26 begin
27     esal:=updateSal(eno);
28     if esal=-1 then
29         dbms_output.put_line('查无此人');
30     elsif esal=-2 then
31         dbms_output.put_line('返回多行');
32     elsif esal=-3 then
33         dbms_output.put_line('查询出错');
34     else
35         dbms_output.put_line(esal);
36     end if;
37
38 end;

```

```

1  @Test
2  public void testFun4() throws Exception{
3      Class.forName("oracle.jdbc.driver.OracleDriver");
4      Connection conn = DriverManager.getConnection(
5          "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
6      //预定义函数的调用格式
7      CallableStatement call = conn.prepareCall("{?=call updateSal(?) }");
8      //声明函数各个部分的类型和参数说明
9      //注册返回值参的类型
10     call.registerOutParameter(1,Types.FLOAT);
11     //设置普通参数
12     call.setInt(2,7000);
13     call.execute();
14
15     float sal = call.getFloat(1);
16     if(sal==-1){
17         System.out.println("查无此人");
18     }
19     else if(sal==-2){
20         System.out.println("返回多行");
21     }
22     }
23     else if(sal==-3){
24         System.out.println("执行出错");
25     }
26     }
27     else{
28         System.out.println(sal);
29     }
30     call.close();
31     conn.close();
32
33 }

```

10、练习

定义一个函数

雇员编号采用自增序列

saveEmp(ename, esal, ehiredate, edeptno)

检查部门编号是否合法，返回错误标识 -1

如果函数执行出错，返回错误标识 -2

如果成功，返回新增的雇员编号

用PL和JDBC实现该函数的调用！

- 定义函数

```

1  create or replace function saveEmp(ename emp.ename%type,
2  esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type)
3  return int
4  is
5      eno int;
6      dno_cnt int;
7  begin
8      select count(deptno) into dno_cnt from emp where deptno=edeptno;
9      if dno_cnt =0 then
10         return -1;
11     else
12         select seq1.nextval into eno from dual;
13         insert into
14         emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
15         return eno;
16     end if;
17     exception
18         when others then return -2;
19 end;

```

- 在pl中调用该函数

```

1  declare
2      eno int;
3  begin
4      eno:=saveEmp('mike',1000,sysdate,30);
5      case eno
6          when -1 then dbms_output.put_line('部门编号不存在');
7          when -2 then dbms_output.put_line('查询出错');
8          else
9              dbms_output.put_line('新的雇员编号: '||eno);
10
11      end case;
12  end;
13
14  select * from emp;

```

- 在jdbc中调用该函数

```

1  @Test
2  public void testFun5() throws Exception{
3      Class.forName("oracle.jdbc.driver.OracleDriver");
4      Connection conn = DriverManager.getConnection(
5          "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
6      //预定义函数的调用格式
7      CallableStatement call = conn.prepareCall("{?=call saveEmp(?,?,?,?) }");
8      //声明函数各个部分的类型和参数说明
9      //注册返回值参的类型
10     call.registerOutParameter(1,Types.INTEGER);
11     //设置普通参数
12     call.setString(2,"rose");
13     call.setFloat(3,2000);
14     call.setTimestamp(4,Timestamp.valueOf("2017-3-3 10:10:10"));
15     call.setInt(5,50);
16     call.execute();
17
18     int eno = call.getInt(1);
19     if(eno==-1){
20         System.out.println("部门编号不存在");
21     }
22     else if(eno==-2){
23         System.out.println("查询出错");
24     }
25     else{
26         System.out.println(eno);
27     }
28
29
30 }

```

11、返回参

可以在函数的参数中定义返回参

```

1 hello(name varchar)===hello(name in varchar)--name只做输入,默认参数为输入参
2 hello(name out varchar) ---name只能输出
3 hello(name in out varchar)--name即做输入, 也做输出
4
5
6 create or replace function hello(name in out varchar) return varchar
7 is
8
9 begin
10     name:=upper(name);
11     return 'hello' || ' ' || name;
12 end;
13
14 declare
15     n varchar(10):='tom';
16     r varchar(100);--不设置类型精度只有两个地方: 参数, 返回值
17 begin
18     r:=hello(n);
19     dbms_output.put_line(n);--TOM
20     dbms_output.put_line(r);
21 end;

```

除了返回雇员编号, 还要返回该雇员所在部门的总人数

```

1 create or replace function saveEmp1(ename emp.ename%type,
2 esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type,dept_total_emps out
3 int)
4 is
5     eno int;
6     dno_cnt int;
7 begin
8     select count(deptno) into dno_cnt from emp where deptno=edeptno;
9     if dno_cnt =0 then
10         return -1;
11     else
12         select seq1.nextval into eno from dual;
13         insert into
14 emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
15         --设置返回参
16         select count(deptno)into dept_total_emps from emp where deptno=edeptno;
17         return eno;
18     end if;
19 exception
20     when others then return -2;
21 end;

```

```

1 declare
2     eno int;
3     demps int;--返回参，该部门下雇员的人数
4 begin
5     eno:=saveEmp1('mike1',1000,sysdate,30,demps);
6     case eno
7         when -1 then dbms_output.put_line('部门编号不存在');
8         when -2 then dbms_output.put_line('查询出错');
9         else
10             dbms_output.put_line('新的雇员编号: '||eno||',该部门下已有'||demps||'位同事');
11
12     end case;
13 end;
14
15 select count(*) from emp where deptno=30;

```

```

1 @Test
2 public void testFun6() throws Exception{
3     Class.forName("oracle.jdbc.driver.OracleDriver");
4     Connection conn = DriverManager.getConnection(
5         "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
6     //预定义函数的调用格式
7     CallableStatement call = conn.prepareCall("{?=call saveEmp1(?,?,?,?) }");
8     //声明函数各个部分的类型和参数说明
9     //注册返回值参的类型
10    call.registerOutParameter(1,Types.INTEGER);
11    //设置普通参数
12    call.setString(2,"rose");
13    call.setFloat(3,2000);
14    call.setTimestamp(4,Timestamp.valueOf("2017-3-3 10:10:10"));
15    call.setInt(5,10);
16    call.registerOutParameter(6,Types.INTEGER);
17    call.execute();
18
19    int eno = call.getInt(1);
20    if(eno==-1){
21        System.out.println("部门编号不存在");
22    }
23    else if(eno==-2){
24        System.out.println("查询出错");
25    }
26    else{
27        System.out.println("新增雇员编号: "+eno);
28        System.out.println("该部门下已经有"+call.getInt(6)+"位同事");
29    }
30
31
32 }

```

除了返回雇员编号，还要返回部门内雇员的信息


```

1  create or replace function saveEmp2(ename emp.ename%type,
2  esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type,dept_ems out
   sys_refcursor)
3  return int
4  is
5      eno int;
6      dno_cnt int;
7  begin
8      select count(deptno) into dno_cnt from emp where deptno=edeptno;
9      if dno_cnt =0 then
10         return -1;
11     else
12         select seq1.nextval into eno from dual;
13         insert into
emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
14         --设置返回参
15         open dept_ems for select * from emp where deptno=edeptno;
16         return eno;
17     end if;
18     exception
19         when others then return -2;
20
21 end;

```

```

1  declare
2      eno int;
3      emps sys_refcursor;--返回参，该部门下所有雇员的信息
4      e emp%rowtype;
5  begin
6      eno:=saveEmp2('mike2',1000,sysdate,30,emps);
7      case eno
8          when -1 then dbms_output.put_line('部门编号不存在');
9          when -2 then dbms_output.put_line('查询出错');
10         else
11             dbms_output.put_line('新的雇员编号: '||eno);
12             loop
13                 fetch emps into e;
14                 exit when emps%notfound;
15                 dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);
16
17             end loop;
18             close emps;--一定要关闭该游标
19
20         end case;
21     end;
22
23     select empno,ename,sal from emp where deptno=30;

```

```

1  @Test
2  public void testFun7() throws Exception{
3      Class.forName("oracle.jdbc.driver.OracleDriver");
4      Connection conn = DriverManager.getConnection(
5          "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
6      //预定义函数的调用格式
7      CallableStatement call = conn.prepareCall("{?=call saveEmp2(?,?,?,?) }");
8      //声明函数各个部分的类型和参数说明
9      //注册返回值参的类型
10     call.registerOutParameter(1,Types.INTEGER);
11     //设置普通参数
12     call.setString(2,"rose");
13     call.setFloat(3,2000);
14     call.setTimestamp(4,Timestamp.valueOf("2017-3-3 10:10:10"));
15     call.setInt(5,10);
16     call.registerOutParameter(6,OracleTypes.CURSOR);
17     call.execute();
18
19     int eno = call.getInt(1);
20     if(eno==-1){
21         System.out.println("部门编号不存在");
22     }
23     else if(eno==-2){
24         System.out.println("查询出错");
25     }
26     else{
27         System.out.println("新增雇员编号: "+eno);
28         ResultSet rs=(ResultSet)call.getObject(6);
29         while(rs.next()){
30             System.out.println(rs.getInt("empno")+"\t"+
31                 rs.getString("ename")+"\t"+rs.getFloat("sal"));
32         }
33     }
34 }

```

八：存储过程

1、定义

procedure: 存储过程，跟函数类似，都是预编译在数据库中的一段pl代码块，可以被调用

存储过程没有返回值，只通过返回参（可以定义多个返回参，返回多个结果）去实现结果的返回。创建和执行权限和函数一致，create procedure,在resource角色中有

```

1  create or replace procedure proHello(name varchar)
2  is
3  begin
4      dbms_output.put_line('hello' || name);
5  end;
6
7  execute proHello('tom');--存储过程没有返回参，可以使用execute直接调用
8
9
10 create or replace procedure proHello1(name varchar,n_out out varchar)
11 is
12
13 begin
14     dbms_output.put_line('hello' || name);
15     n_out:=upper(name);
16 end;
17
18
19 execute proHello1('张三');--有返回参，不能直接调用，需要使用pl中进行调用

```

2、在pl块中调用存储过程

```

1  declare
2
3  begin
4      proHello('tom');
5  end;
6
7  declare
8      n varchar(10);
9  begin
10     proHello1('tom',n);--hellotom
11     dbms_output.put_line(n);--TOM
12 end;
13

```

3、数据字典表

```

1  select * from user_functions;--不存在
2  select * from user_procedures;--包括函数和存储过程
3  select * from user_objects where object_type=upper('procedure');--只包含存储过程

```

注意：表名、函数名、存储过程名、序列名.....在oracle中不能重名，因为他们都属于oracle的对象名。通过 **select * from user_objects** 查询当前用户下的所有对象(表名、函数名、存储过程名、序列名....)

4、在pl中调用存储过程

```

1  create or replace procedure prcSaveEmp2(ename emp.ename%type,
2  esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type,dept_emp out
   sys_refcursor,eno out int)
3
4  is
5      dno_cnt int;
6  begin
7      select count(deptno) into dno_cnt from emp where deptno=edeptno;
8      if dno_cnt =0 then
9          eno:=-1;
10     else
11         select seq1.nextval into eno from dual;
12         insert into
emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
13         --设置返回参
14         open dept_emp for select * from emp where deptno=edeptno;
15         eno:=eno;
16     end if;
17     exception
18         when others then eno:=-2;
19
20 end;

```

```

1  select * from emp;
2  declare
3      eno int;
4      emp sys_refcursor;--返回参, 该部门下所有雇员的信息
5      e emp%rowtype;
6  begin
7      prcSaveEmp2('mike3',1000,sysdate,30,emp,eno);
8      case eno
9          when -1 then dbms_output.put_line('部门编号不存在');
10         when -2 then dbms_output.put_line('查询出错');
11         else
12             dbms_output.put_line('新的雇员编号: '||eno);
13             loop
14                 fetch emp into e;
15                 exit when emp%notfound;
16                 dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);
17             end loop;
18             close emp;--一定要关闭该游标
19
20         end case;
21     end;
22
23
24 select empno,ename,sal from emp where deptno=30;

```

5、在java中调用存储过程

```

1  @Test
2  public void testProcedure1() throws Exception{
3      Class.forName("oracle.jdbc.driver.OracleDriver");
4      Connection conn = DriverManager.getConnection(
5          "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
6      //预定义函数的调用格式
7      CallableStatement call = conn.prepareCall("{call prcSaveEmp2(?,?,?,?,?) }");
8      //声明函数各个部分的类型和参数说明
9      //设置普通参数
10     call.setString(1,"rose1");
11     call.setFloat(2,2000);
12     call.setTimestamp(3,Timestamp.valueOf("2017-3-3 10:10:10"));
13     call.setInt(4,10);
14     //注册返回参：游标
15     call.registerOutParameter(5,OracleTypes.CURSOR);
16
17     //注册返回参：雇员编号
18     call.registerOutParameter(6,Types.INTEGER);
19     call.execute();
20
21     int eno = call.getInt(6);
22     if(eno==--1){
23         System.out.println("部门编号不存在");
24     }
25     else if(eno==--2){
26         System.out.println("查询出错");
27     }
28     else{
29         System.out.println("新增雇员编号: "+eno);
30         ResultSet rs=(ResultSet)call.getObject(5);
31         while(rs.next()){
32             System.out.println(rs.getInt("empno")+"\t"+
33                 rs.getString("ename")+"\t"+rs.getFloat("sal"));
34         }
35     }
36     call.close();
37     conn.close();
38 }

```

6、何时使用存储过程？

在实际开发中，是采用函数/存储过程,还是使用JDBC去操作数据库？

1、3W1H

what why where how

函数/存储过程：业务后端编程---依赖于数据库

JDBC:业务前端编程---依赖于高级语言

2、优点

1. 调用方便
2. 执行快捷
 - 如果你传递的是sql语句，那么数据库需要经过语法校验--解析--编译--进行数据库各种口令--执行
 - 如果你使用函数或者存储过程，那么直接执行
3. 保护业务(黑盒编程)

3、缺点

1. 完全依赖于数据库(不同的数据库/升级都需要)
2. 不容易调试(编码调试，性能调试)

九：删除数据的2种方式

- delete

DML

支持事务

delete from emp where 条件(选择删除行)

需要手动收缩表空间

- truncate

DDL

不支持事务

truncate table 表名(删除整张表，不能进行行选择)

可以自动收缩表空间

```
1 drop table t_test;
2 create table t_test(id int,name varchar(14));
3
4 begin
5     for i in 1..100000
6     loop
7         insert into t_test values(i,'mike_'||i);
8     end loop;
9 end;
10
11 select count(1) from t_test;
```

```
1 --查看表占有的表空间大小
2 select segment_name,bytes/1024||'k' from user_segments;
3 commit;
4
5 --如果你删除数据，无论你删除了多少数据，表空间大小并没有发生变化
6 --oracle会通过自己的算法，会在系统空闲(CPU,I/O)的时候,会自动扫描磁盘空间中的无效数据，进行回收
7 delete from t_test where id<50000;
8 --truncate 可以自动收缩表空间
9 truncate table t_test;
```

十：视图

1、为什么要使用视图？

将一个表的查询权限授予mike

```
1 grant select,insert,update,delete on emp to mike;
```

需求：如何将一张表的部分信息（部分行，部分列）开放给其他人？

我不允许10号部门的雇员访问，也不允许对薪水字段进行访问，除此之外可以对任何数据访问

```
1 --需要create view权限
2 create view v_emp
3 as
4 select empno,ename,job,mgr,hiredate,comm,deptno from emp
5 where deptno<>10;
6 --就可以将这个视图授权给某个mike用户
7 grant select,insert on v_emp to mike;
8 select * from v_emp;
```

对于同一个基表，可以创建任意多个视图

例如：创建一个视图：不让修改修改薪水列

```
1 create view v_emp1
2 as
3 select empno,ename,job,mgr,hiredate,comm,deptno from emp;
4 --就可以将这个视图授权给某个mike用户
5 grant select,insert on v_emp1 to mike;
6 select * from v_emp1;
```

通过视图，可以有选择的对部分行和列进行操作

2、视图和子查询之间的关系

视图是一个逻辑表，它只是基表的一个子查询，在数据库并不占有表空间

```
1 select t.* from(
2 select empno,ename,sal from emp) t;
3
4 create view v_1
5 as
6 select empno,ename,sal from emp;
7
8 select * from v_1;
```

3、视图的数据字典表

```
1 | select * from user_views;
```

4、视图的CRUD操作

视图虽然只是一张逻辑表，但是仍然可以对其进行crud操作

```
1 | insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
2 | values(111,'mike',sysdate,'SALE',7369,20);
3 |
4 | select * from v_emp;
5 | --10号部门与视图的查询冲突，但默认可以插入
6 | insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
7 | values(112,'mike1',sysdate,'SALE',7369,10);
```

问题：

无效数据应该不能插入，需要对视图增加一个检查约束，不允许通过视图去修改和保存与视图相冲突的数据。

with check option

```
1 | --drop view v_emp;
2 | create or replace view v_emp
3 | as
4 | select empno,ename,job,mgr,hiredate,comm,deptno from emp
5 | where deptno<>10
6 | with check option;
7 |
8 | insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
9 | values(112,'mike1',sysdate,'SALE',7369,10);--报错，无效数据
```

原理：根据视图去修改数据，要求该视图对数据一定是可见的，否则没有意义。

```
1 | select * from emp;
2 | insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
3 | values(112,'mike1',sysdate,'SALE',7369,10);--报错，无效数据
4 |
5 | update v_emp set ename=lower(ename) where ename='KING';--修改不了
6 | update v_emp set sal=sal*1.1 where empno=7369;--报错，sal列不可见
```

如果不允许用户对视图做任何修改，可以设置成只读视图

with read only

```
1 | create or replace view v_emp
2 | as
3 | select empno,ename,job,mgr,hiredate,comm,deptno from emp
4 | where deptno<>10
5 | with read only;
6 |
7 | insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
8 | values(112,'mike1',sysdate,'SALE',7369,20);
```


对于一个复杂的查询，也可以定义一个视图，然后暴露该视图给用户

```
1 create table salgrade as select * from scott.salgrade;
2 create or replace view v_emp_infos
3 as
4 select e.empno,e.ename,e.sal,d.dname,g.grade
5 from emp e,dept d,salgrade g
6 where e.deptno=d.deptno
7 and e.sal between g.losal and g.HISAL;
8
9 select * from v_emp_infos;
```

该视图也可以作为子查询的连接表

```
1 select v.*,e.sal,e.comm
2 from emp e,V_EMP_INFOS v
3 where e.empno=v.empno;
4
5 select v.*,e.sal,e.comm
6 from emp e,V_EMP_INFOS v
7 where e.empno=v.empno;
8 --多表关联的视图无法进行增、删、改操作，只能查询
9 delete from V_EMP_INFOS where empno=7369;--报错
10 insert into V_EMP_INFOS (empno,ename)values(123,'jack');--报错
11 update V_EMP_INFOS set ename=lower(ename) where empno=7369;
```

问题：多表关联的视图无法进行增、删、改操作，只能查询

解决方案：通过触发器可以实现多表连接的视图的增删改

5、视图的重命名

```
1 select * from v_1;
2 rename v_1 to v_2;
```

十一：触发器

1、种类

- 系统级触发器(具有dba角色的用户才能定义)

对系统级事件进行触发

- **表级触发器**(只要对表进行了insert,update,delete,select操作，就会触发，它不管你影响了多少行，只会触发一次)

对DML进行操作时触发

- **行级触发器**

对DML进行操作时触发

A.表上的行级触发器(只要表上的行被影响,就会触发多少次,如果没有影响,不触发)

B.视图上的行级触发器(只作用在视图上)

- DDL触发器(当你对数据进行ddl操作, create drop alter truncate)

对DDL进行操作时触发

2、表级触发器

- 例子

```
1
2  --创建一个触发器
3  --表级触发器: after和before没有什么区别
4  create or replace trigger tri_emp_dml
5  after insert or update or delete on emp
6  declare
7
8  begin
9      if inserting then
10         dbms_output.put_line('insert');
11     elsif updating then
12         dbms_output.put_line('update');
13     elsif deleting then
14         dbms_output.put_line('delete');
15     end if;
16
17 end;
18 select * from emp;
19 insert into emp(empno,ename)values(113,'mayer');
20 update emp set sal=2300 where empno=7369;
21 delete from emp where empno=1;
```

- 练习

需求:

允许用户在任何时间段录入新的数据,但是你只能在工作时间(周一到周五8:00-18:00)对数据进行更新和删除

```

1  select to_char(sysdate,'day') from dual;
2  select to_char(sysdate,'hh24') from dual;
3
4  create or replace trigger tri_emp_dml
5  before update or delete on emp
6  declare
7      --定义一个异常
8      --oper_date_error exception;
9  begin
10     --如果不是工作时间，抛异常
11     if to_char(sysdate,'day')in('星期六','星期日') or to_char(sysdate,'hh24') not between 8
and 18 then
12         --raise oper_date_error;
13         --使用系统异常
14         raise_application_error(-20001,'不允许在非工作日时间进行更新和删除操作');
15     end if;
16     --exception
17     --when oper_date_error
18     -- then dbms_output.put_line('不允许在非工作日时间进行更新和删除操作');
19
20 end;
21
22 select * from emp;
23 insert into emp(empno,ename)values(1,'mike');
24 update emp set ename=lower(ename) where empno=1;

```

问题：使用自定义异常，虽然可以捕获该异常，但操作并没有拦截，仍然更新/删除成功

解决方案：使用系统异常

3、行级触发器

1、概念

```

1  select * from emp;
2
3  create or replace trigger tri_emp_dml
4  after insert or update or delete on emp
5  for each row
6  declare
7
8  begin
9      if inserting then
10         dbms_output.put_line('insert');
11     elsif updating then
12         dbms_output.put_line('update');
13     elsif deleting then
14         dbms_output.put_line('delete');
15     end if;
16
17 end;
18
19 insert into emp(empno,ename)values(2,'rose');
20 delete from emp where deptno is null;
21

```

2、行级游标

在做行级触发器中，有两个默认的行级游标，分别用来引用该行数据在更新前、后的值

:new ---insert的行或者update后的值

:old ---update前的值或者是delete的值

```

1  create or replace trigger tri_emp_dml
2  after insert or update or delete on emp
3  for each row
4  declare
5
6  begin
7      if inserting then
8         dbms_output.put_line('insert'||:new.empno||' '||:new.ename);
9     elsif updating then
10         dbms_output.put_line('update'||:old.ename||' '||:new.ename);
11     elsif deleting then
12         dbms_output.put_line('delete'||:old.empno||' '||:old.ename);
13     end if;
14
15 end;
16
17 insert into emp(empno,ename)values(1,'mike');--insert1 mike
18 update emp set ename='张三' where empno=1;--updatemike 张三
19 delete from emp where sal is null;

```

3、before和after的区别

在表级触发器中before和after其实没有什么区别，在行级触发器中我们是可以修改:new值的，前提是只能是before

```

1  create or replace trigger tri_emp_dml
2  before insert or update or delete on emp
3  for each row
4  declare
5
6  begin
7      if inserting then
8          dbms_output.put_line('insert'||:new.empno||' '||:new.ename);
9          :new.ename:=upper(:new.ename);
10     elsif updating then
11         dbms_output.put_line('update'||:old.ename||' '||:new.ename);
12         :new.ename:=upper(:new.ename);
13     elsif deleting then
14         dbms_output.put_line('delete'||:old.empno||' '||:old.ename);
15
16     end if;
17
18 end;
19
20 insert into emp(empno,ename)values(9001,'hello1');--90001  hello1
21
22 update emp set ename='zhangsan' where empno=9001;--HELLO1  zhangsan

```

4、特点

在行级触发器中，如果是insert操作（必须before）我们还以查询触发的表，如果你是update,delete,不能查询触发的表

原因：

- insert不会对之前的数据有影响，可以查询insert之前的数据
- update,delete会对之前的数据有影响，不可以查询update,delete之前的数据

```

1  select count(1) from emp;
2
3  create or replace trigger tri_emp_dml
4  before insert or update or delete on emp
5  for each row
6  declare
7      cnt int;
8  begin
9      if inserting then
10         dbms_output.put_line('insert'||:new.empno||' '||:new.ename);
11         select count(1) into cnt from emp;--成功
12         dbms_output.put_line(cnt);--插入之前的记录数，比实际值-1
13     elsif updating then
14         dbms_output.put_line('update'||:old.ename||' '||:new.ename);
15         select count(1) into cnt from emp;--报错
16         dbms_output.put_line(cnt);
17     elsif deleting then
18         dbms_output.put_line('delete'||:old.empno||' '||:old.ename);
19         select count(1) into cnt from emp;--报错
20         dbms_output.put_line(cnt);
21     end if;
22
23 end;
24
25 insert into emp(empno,ename)values(9002,'rose');
26 update emp set ename=upper(ename) where empno=9002;
27 delete from emp where empno=9002;

```

5、行级触发器的应用

1、应用1

删除，更新备份(将删除的数据自动备份到备份表中)

```

1  select * from emp;
2  select * from emp_bak;
3  rollback;
4  delete from emp where deptno=10;
5  select * from emp_bak;
6  create table emp_bak as select * from emp where 1=2;
7
8  alter table emp_bak add(username varchar(20),oper_d date,oper_type varchar(10));
9
10
11 create or replace trigger trig_emp_del_bak
12 before delete on emp
13 for each row
14 declare
15 begin
16     if deleting then
17         --将删除的数据备份到备份表中
18         --insert into emp_bak as select * from emp where empnp=:old.empno;--不可以,
delete,update操作, 不能查询触发的表
19         insert into emp_bak
values(:old.empno,:old.ename,:old.job,:old.mgr,:old.hiredate,:old.sal,:old.comm,:old.deptno
,user,sysdate,'delete');
20     end if;
21 end;
22
23 delete from emp where empno=61;

```

2、应用2

使用触发器自己实现一个主键列的自增

原理:

如果当前表中没有记录, 默认从1开始, 否则, 最大键值+1

```

1 truncate table emp;
2 select * from emp;
3 select count(empno) from emp;--0
4 select max(empno) from emp;--null
5
6
7 create or replace trigger trig_emp_autoid
8 before insert on emp
9 for each row
10 declare
11     cnt int;
12 begin
13     select max(empno) into cnt from emp;
14     if cnt is null then
15         :new.empno:=1;
16     else
17         :new.empno:=cnt+1;
18     end if;
19 end;
20
21 insert into emp (ename,sal)values('李四1',2000);

```

6、禁用，启用触发器

```

1 --禁用某一个触发器（行级，表级触发器）
2 alter trigger trig_emp_autoid disable;
3
4 --启用某一个触发器（行级，表级触发器）
5 alter trigger trig_emp_autoid enable;
6
7 --禁用某个表上的所有触发器(行级，表级触发器)
8 alter table emp disable all triggers;
9 --启用某个表上的所有触发器(行级，表级触发器)
10 alter table emp enable all triggers;

```