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

```
create table  t_stu3(
    id int primary key,
    name varchar2(30)
);
select * from t_stu3;
--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
begin
   insert into t_stu3 values(5,'mike');
   insert into t_stu3
values(6,'rosedfffffffffffffffffffffffffffffffffffffffffffffffffffff');
   --commit;

end;
```

```
--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
set serveroutput  on;
declare
    --定义变量、常量并赋值
    --i  number(2):=1;
    i number;
    j number:=9;
    --i:=3;--这是执行语句
    flag boolean:=true;

begin
    --执行语句
    --i:=3;

    --输出结果
    dbms_output.put_line(i+j);
    --dbms_output.put_line(flag);--无法输出布尔类型
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

```
declare
    i number:=&请输入一个值;--数值接收
    j number:=9;
    name varchar(10):='&请输入姓名';
    flag boolean:=true;

begin
    dbms_output.put_line(i+j);
    dbms_output.put_line(name);
    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

end;

declare
    --布尔类型默认初始值为null,null在if逻辑判断语句中为假
    flag boolean;
begin
    --if flag then
    --if flag=false then
    if flag is null then
        dbms_output.put_line('true');
    end if;
end;

declare
    --PI float:=3.14;
    --PI float default 3.14;--这种赋值方式只在声明语句中适用
    PI constant float default 3.14;--这是个常量
begin
    --pi:=3.1415;
    dbms_output.put_line(PI);
end;
```

三：判断

- if
- case

1、if

语法1：

if 条件表达式 then

……

else

……

end if;

```
declare
    i int :=&请输入值;

begin
    if i>=10  then
        dbms_output.put_line('>=10');
    else
        dbms_output.put_line('<10');
    end if;
end;
```

语法2：

if 条件表达式 then

elsif 条件表达式 then

....
else

.....

end if

```
declare
    chr char(1):=lower('&请输入学生的成绩等级');
begin
    if chr='a' then
        dbms_output.put_line('优秀');
    elsif chr='b' then
        dbms_output.put_line('良好');
    elsif chr='c' then
        dbms_output.put_line('一般');
    elsif chr='d' then
        dbms_output.put_line('及格');
    elsif chr='e' then
        dbms_output.put_line('不及格');
    else
        dbms_output.put_line('未知');


    end if;

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;

```
declare
    chr char(1):=lower('&请输入成绩等级');

begin
    case chr
        when 'a' then
            dbms_output.put_line('优秀');
        when 'b' then
            dbms_output.put_line('良好');
        when 'c' then
            dbms_output.put_line('一般');
        when 'd' then
            dbms_output.put_line('及格');
        when 'e' then
            dbms_output.put_line('不及格');
        else
            dbms_output.put_line('未知');


    end case;
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;

```
declare
    score number:=&请输入学生的成绩;

begin
    if score between 0 and 100 then
    case
        when score between 90 and 100 then
            dbms_output.put_line('优秀');
        when score between 80 and 89 then
            dbms_output.put_line('良好');
        when score between 70 and 79 then
            dbms_output.put_line('一般');
        when score between 60 and 69 then
            dbms_output.put_line('及格');

        else
            dbms_output.put_line('不及格');


    end case;
    else
        dbms_output.put_line('成绩值不在0-100之间');
    end if;
end;
```

四：循环

1、loop

> 自己控制计数器和循环条件

```
--输出1-100
declare
    i int default 1;

begin
loop
    dbms_output.put_line(i);
    if i=100 then
        --退出循环
        --exit;--类似于java中的break,退出当前循环，不是退出pl块
        return;--退出当前pl块
    end if;
    --计数
    i:=i+1;--oracle中没有i+=1,i++这种写法
    --i++;


end loop;

dbms_output.put_line('end');
end;
```

2、for

```
--打印1-100
begin
    for i in 1..100--注意：i值永远只能是1，-1，不能修改自增值
    loop
        dbms_output.put_line(i);
        --i:=i+3;--不能进行赋值操作
    end loop;
end;

--打印100-50
begin
    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
    loop
        dbms_output.put_line(i);
        if i=50 then
            exit;
        end if;
    end loop;
end;
```

3、while

> 类似于java中的while,没有do while

```
--打印1-100
declare
    i int :=1;
begin
    while i<=100
    loop
        dbms_output.put_line(i);
        i:=i+1;
    end loop;
end;
```

```
--缓冲输出
begin
    dbms_output.put(10);--缓冲输出，不换行
    dbms_output.put(10);
    dbms_output.new_line();--表示清空缓冲区,将值输出，同时换行

    dbms_output.put(20);
    dbms_output.put(20);
    dbms_output.new_line();
end;

--九九乘法表
begin
    --遍历列
    for i in 1..9
    loop
        --遍历列(上三角)
        for j in 1..i
        loop
            dbms_output.put(i||'*'||j||'='||i*j||' ');
        end loop;
        dbms_output.new_line();
    end loop;

end;
```

```
--练习
--求100以内的质数
--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
declare
    flag boolean:=true;
begin
    --除数
    for num in 2..100
    loop
        --循环前初始化flag为true
        flag:=true;
        --被除数
        for i in 2..num-1
        loop
            if mod(num,i)=0 then
                flag:=false;
            end if;
        end loop;
        if flag then
            dbms_output.put_line(num);
        end if;

    end loop;
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

```

select * from emp;
set serveroutput on;
rollback;

declare

begin
    delete from emp where deptno=20;--单行游标
    if sql%found then  --oracle中访问对象中的属性使用%
        dbms_output.put_line('成功删除了'||sql%rowcount||'条记录');
    else
        dbms_output.put_line('没有匹配的记录被删除');
    end if;
end;
```

4、指定列值变量的单行游标

```
select * from emp;
desc emp;

declare
    eno number(4) :=&请输入雇员编号;
    --这三个变量用于放入查询中对应的列值
    --采用动态类型,指定变量类型和查询的字段的类型一致
    e_name emp.ename%type;
    e_date emp.hiredate%type;
    e_sal emp.sal%type;
    cnt int;

begin
    select count(*)into cnt from emp where empno=eno;
    if cnt=1 then
        select ename,hiredate,sal into e_name,e_date,e_sal from emp where empno=eno;
        dbms_output.put_line(e_name||'的入职时间：'||to_char(e_date,'yyyy-mm-dd')||'薪水是：'||e_sal);
    else
        dbms_output.put_line('没有找到查询结果');
    end if;
end;
```

5、指定行值变量的单行游标

```
select * from emp;
desc emp;

declare
    eno number(4) :=&请输入雇员编号;
    e emp%rowtype;--定义emp表的行的类型
    cnt int;

begin
    select count(*)into cnt from emp where empno=eno;
    if cnt=1 then
        select * into e from emp where empno=eno;
        dbms_output.put_line(e.ename||'的入职时间：'||to_char(e.hiredate,'yyyy-mm-dd')||'薪水是：'||e.sal);
    else
        dbms_output.put_line('没有找到查询结果');
    end if;
end;

```

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)

```
begin
    for t in (select * from user_tables)
    loop
        dbms_output.put_line(t.table_name||' '||t.tablespace_name);
    end loop;

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

```
declare
    --定义一个显示游标
    cursor cur is select * from emp where deptno=10;
    e emp%rowtype;
begin
    --打开游标
    open cur;
    if cur%found is null then
        dbms_output.put_line('found is null');
    else
        dbms_output.put_line('found is not null');
    end if;
    if cur%notfound is null then
        dbms_output.put_line('notfound is null');
    else
        dbms_output.put_line('notfound is not null');
    end if;
    dbms_output.put_line(cur%rowcount);--0
    --将指针向下移动，指向下一条记录
    fetch cur into e;
    if cur%found then
        dbms_output.put_line('found is true');
    else
        dbms_output.put_line('found is not true');
    end if;
    if cur%notfound then
        dbms_output.put_line('notfound is true');
    else
        dbms_output.put_line('notfound is not true');
    end if;
    dbms_output.put_line(cur%rowcount);--1
    fetch cur into e;
    dbms_output.put_line(cur%rowcount);--2
    fetch cur into e;
    dbms_output.put_line(cur%rowcount);--3
    fetch cur into e;
    dbms_output.put_line(cur%rowcount);--3  EOF
    if cur%found then
        dbms_output.put_line('found is true');
    else
        dbms_output.put_line('found is not true');--found is not true
    end if;
    if cur%notfound then
        dbms_output.put_line('notfound is true');--notfound is true
    else
        dbms_output.put_line('notfound is not true');
    end if;
    --关闭游标
    close cur;

end;
```

使用loop循环

```
declare
    cursor cur is select * from emp where deptno=&请输入部门编号;
    e emp%rowtype;
begin
    open cur;
    loop
        fetch cur into e;
        --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
        if cur%notfound then
            exit;--退出循环
        end if;
        dbms_output.put_line(e.empno||' '||rpad(e.ename,10,'*')||' '||e.sal);


    end loop;
    close cur;

end;

```

7、课堂练习

1、练习1

> 求前100个质数

```
declare
    cnt int :=0;   --个数
    num int :=2;   --质数
    flag boolean;  --判断该值是否为质数，true-->是质数
begin
    while cnt <100-->循环100次
    loop
        flag:=true;
        for i in 2..num-1  --除数
        loop

            if mod(num,i)=0 then   --能被整除，该值不是质数
                flag:=false;
                exit;
            end if;
        end loop;
        if flag then
            cnt:=cnt+1;
            dbms_output.put_line(cnt||' '||num);
        end if;
        --将该值加1，继续判断+1后的值是否为质数
        num:=num+1;

    end loop;


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

```
declare
    str varchar(30):='&请输入一个字符串';
    c1 char(1);
    c2 char(1);
    flag boolean:=true;
begin
    for i in 1..trunc(length(str)/2)
    loop
        --获取对应的字符
        c1:=substr(str,i,1);
        c2:=substr(str,-i,1);
        if c1<>c2 then
            flag:=false;
            exit;--相当于break,退出当前循环，继续下一次循环
        end if;
    end loop;
    if flag then
        dbms_output.put_line(str||'是回文');
    else
        dbms_output.put_line(str||'不是回文');
    end if;


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=1*1*1+5*5*5+3*3*3

```
--百位
select trunc(123/100) from dual;
--十位
select trunc((156-i*100)/10) from dual;

--个位
select mod(123,10) from dual;

declare
    x int;
    y int;
    z int;
begin
    for i in 100..999
    loop
        x:=trunc(i/100);
        y:=trunc((i-x*100)/10);
        z:=mod(i,10);
        if power(x,3)+power(y,3)+power(z,3)=i then
            dbms_output.put_line(i);
        end if;
    end loop;

end;
```

4、练习4

> 求值的算术平方根

```
select round(sqrt(10),6) from dual;

declare
    x int :=&求平方根;
    --定义数列的两个变量
    s1 number(10,6):=1;--定义前项
    s2 number(10,6);--定义后项



begin
    s2:=0.5*(s1+x/s1);
    while abs(s2-s1)>=power(10,-6)
    loop
        s1:=s2;--项后移
        s2:=0.5*(s1+x/s1);

    end loop;
    dbms_output.put_line(s1||' '||round(sqrt(x),6));

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

```plsql
select * from emp;

declare
    cursor cur is select deptno,empno,sal from emp;
    dno emp.deptno%type;
    eno emp.empno%type;
    esal emp.sal%type;
    p float;
    cnt int :=0;
begin
    open cur;
    loop
        fetch cur into dno,eno,esal;
        --判断何时退出遍历
        if cur%notfound then
            exit;
        end if;
        case dno
            when 10 then
                if esal between 0 and 999 then
                    p:=1.05;
                elsif esal between 1000 and 1999 then
                    p:=1.03;
                elsif esal between 2000 and 2999 then
                    p:=1.02;
                else
                    p:=1;
                end if;
            else
                if esal between 0 and 999 then
                    p:=1.1;
                elsif esal between 1000 and 1999 then
                    p:=1.05;
                elsif esal between 2000 and 2999 then
                    p:=1.03;
                else
                    p:=1.01;
                end if;

        end case;
        dbms_output.put_line('雇员编号：'||eno||',所在部门：'
        ||dno||',薪水从'||esal||',加薪后变为>>>'||esal*p);

        --将新值更新到数据库
        update emp set sal=sal*p where empno=eno;
        cnt:=cnt+sql%rowcount;

    end loop;

    dbms_output.put_line('成功更新了'||cnt||'条雇员的薪水信息');

    close cur;

```

```
end;
```

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.

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

注意：表名的长度是有大小限制的，最大**30**个字节

```
desc user_tables
名称                          空值       类型
----------------------------- -------- ------------
TABLE_NAME                    NOT NULL VARCHAR2(30)
TABLESPACE_NAME                        VARCHAR2(30)
```

```
desc user_tables;
create table t_stu5aaaaaaaaaaaaaabbbbbbbbbbc(id int);--创建失败，表名长度最多30个字节
```

1、创建一张表

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

2、删除自己名下所有的表

```
declare
    cursor tbls is select table_name from user_tables;
    tname user_tables.table_name%type;
begin
    open tbls;
    loop
        fetch tbls into tname;
--      if tbls%notfound then
--          exit;
--      end if;
        exit when tbls%notfound;--这种写法适用于循环中退出条件的判断
        --dbms_output.put_line('drop table '||tname);
        --主要：表名需要加上""
        execute immediate 'drop table "'||tname||'" cascade constraint';
    end loop;

    close tbls;
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
```

```
select * from user_tables;

create table t_emp as select * from scott.emp;
create table t_emp_bak as select * from t_emp;
create table t_emp1 as select * from t_emp;
create table t_emp1_bak as select * from t_emp;
create table temp_emp1 as select * from t_emp;
create table s_emp as select * from t_emp;
temp_  --临时表
s_   统计表
t_   普通表
_bak 备份表
```

```
declare
    --创建2个游标获取普通表和备份表
    cursor baks is select table_name from user_tables where
regexp_like(table_name,'_bak$','i');
    cursor tbls is select table_name from user_tables where
regexp_like(table_name,'^t_','i');
    tname user_tables.table_name%type;
begin
    --删除备份表
    open baks;
    loop
        fetch baks into tname;
        exit when baks%notfound;
        execute immediate 'drop table "'||tname||'" cascade constraint';
    end loop;

    close baks;

    --对普通表进行备份
    open tbls;
    loop
        fetch tbls into tname;
        exit when tbls%notfound;
        execute immediate 'create table "'||tname||'_bak" as select * from "'||tname||'"';
    end loop;
    close tbls;

end;
```

9、带有参数的游标

```
create table emp as select * from t_emp;

declare
    cursor cur(dno emp.deptno%type) is select * from emp where deptno=dno;
    e emp%rowtype;
begin
    open cur(&请输入部门编号);
    loop
        fetch cur into e;
        exit when cur%notfound;
        dbms_output.put_line(e.empno||' '||e.ename);
    end loop;
    close cur;

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

```
--查询条件：
--1：根据部门编号进行查询
--2：根据雇员名称进行模糊查询
 select * from emp where deptno=10 and ename like upper('%ar%');

declare
    cursor cur(dno emp.deptno%type,en emp.ename%type) is select * from emp where deptno=dno
    and ename like upper(en);
    e emp%rowtype;
begin
    --open cur();
    for e in  cur(&请输入部门编号,'%&姓名%')
    loop
        --fetch cur into e;
        --exit when cur%notfound;
        dbms_output.put_line(e.empno||' '||e.ename);
    end loop;
    --close cur;

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**

```
desc emp;

declare
    cursor cur(dno NUMBER,en VARCHAR2) is select * from emp where deptno=dno
    and ename like upper(en);
    e emp%rowtype;
begin
    --open cur();
    for e in  cur(&请输入部门编号,'%&姓名%')
    loop
        --fetch cur into e;
        --exit when cur%notfound;
        dbms_output.put_line(e.empno||' '||e.ename);
    end loop;
    --close cur;

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

```
select * from emp;
create table dept as select * from scott.dept;

--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）
--select e.ename,e.sal,e.hiredate
--from emp e,emp m where e.mgr=m.empno
--and m.empno=7839;
--另一种写法
select e.ename,e.sal,e.hiredate
from emp e
where e.mgr=7839;

--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）
select e.ename,e.sal,e.hiredate,e.comm,d.dname
from emp e,dept d
where e.deptno=d.deptno
and e.empno=7369;


declare
--声明动态游标(在begin  end块中指定查询语句)
cur sys_refcursor;
eno emp.empno%type:=&雇员编号;
dno emp.deptno%type;
e_name emp.ename%type;p.deptno%type;,e_name emp.ename%type;,e_sal emp.sal%type;,e_date
emp.hiredate%type;,e_comm emp.comm%type;,d_name dept.dname%type;
e_sal emp.sal%type;
e_date emp.hiredate%type;
e_comm emp.comm%type;
d_name dept.dname%type;

begin
    --根据雇员编号获取对应的部门编号
    select deptno into dno from emp where empno=eno;
    if dno=10 then
        open cur for select e.ename,e.sal,e.hiredate
                            from emp e
                            where e.mgr=eno;
        loop
            fetch cur into e_name,e_sal,e_date;
            exit when cur%notfound;
            dbms_output.put_line(e_name||' '||e_sal||' '||e_date);
        end loop;
        close cur;

    else
        open cur for select e.ename,e.sal,e.hiredate,e.comm,d.dname
                            from emp e,dept d
                            where e.deptno=d.deptno
                            and e.empno=eno;
        loop
            fetch cur into e_name,e_sal,e_date,e_comm,d_name;
            exit when cur%notfound;
```

```
            dbms_output.put_line(e_name||' '||e_sal||' '||e_date||' '||e_comm||'
'||d_name);
        end loop;
        close cur;
    end if;

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

```
declare

begin
    for errcode in -20000..0
    loop
        dbms_output.put_line(sqlerrm(errcode));
    end loop;
end;
```

4、捕获异常

pl块中是在代码的最后捕获异常信息

```
--no_data_found:没有返回行
--too_many_rows:返回多行
select * from emp;
insert into emp(empno,ename)values(7788,'mike');
declare
    e_name emp.ename%type;
begin
   select ename into e_name from emp where empno=&请输入雇员编号;

   exception
   --无返回行异常
   when no_data_found then
        dbms_output.put_line('出错了');
        dbms_output.put_line(sqlerrm);
   --多行异常
     when too_many_rows then
        dbms_output.put_line('出错了>>');
        dbms_output.put_line(sqlerrm);
   when others then
        dbms_output.put_line('出错了>>>>');
        dbms_output.put_line(sqlerrm);
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

```
--查询用户的薪水，看输入的薪水
--如果降薪，抛出不能降薪异常
--如果涨薪幅度超过10%，抛出涨薪幅度不能超过10%异常
declare
    less0 exception;
    plus10per exception;
    e_sal emp.sal%type:=&加薪值;
    s_sal emp.sal%type;--原始薪水值
    e_no emp.empno%type:=&雇员编号;
begin
    --检查是否降薪
    if e_sal <0 then
        raise less0;
    end if;
    --检查加薪值是否超过10%
    select sal into s_sal from emp where empno=e_no;
    if (s_sal+e_sal)/s_sal>1.1 then
        raise plus10per;
    end if;
    --dbms_output.put_line('ok');
    update emp set sal=sal+e_sal where empno=e_no;
    dbms_output.put_line('成功更新');

    exception
        when no_data_found then dbms_output.put_line('查无此人');
        when too_many_rows then dbms_output.put_line('返回多行');
        when less0 then dbms_output.put_line('不能降薪');
        when plus10per then dbms_output.put_line('加薪幅度不能超过10%');
end;
```

2、方式2

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

```
--查询用户的薪水，看输入的薪水
--如果降薪，抛出不能降薪异常
--如果涨薪幅度超过10%，抛出涨薪幅度不能超过10%异常
declare
    e_sal emp.sal%type:=&加薪值;
    s_sal emp.sal%type;--原始薪水值
    e_no emp.empno%type:=&雇员编号;
begin
    --检查是否降薪
    if e_sal <0 then
        raise_application_error(-20001,'不能降薪');
    end if;
    --检查加薪值是否超过10%
    select sal into s_sal from emp where empno=e_no;
    if (s_sal+e_sal)/s_sal>1.1 then
        raise_application_error(-20002,'加薪值是否超过10%');
    end if;
    --dbms_output.put_line('ok');
    update emp set sal=sal+e_sal where empno=e_no;
    dbms_output.put_line('成功更新');

    exception
        when no_data_found then dbms_output.put_line('查无此人');
        when too_many_rows then dbms_output.put_line('返回多行');
        when others then
            dbms_output.put_line(sqlerrm);
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

```
select length('abc') from dual;
declare
  len int;
begin
--length('abc');--不能直接执行，一定要接收结果
len:=length('abc');--接收该结果

--dbms_output.put_line(length('abc'));
dbms_output.put_line(len);
end;

```

```
create function plus(i int,j int) return int
is
    --变量的声明
    z int;
begin
    z:=i+j;
    return z;
end;
select length('abc') from dual;
select plus(45,56) from dual;
select nvl(sal,0)sal,nvl(comm,0)comm,plus(nvl(sal,0),nvl(comm,0))aa from emp;


declare
    i int :=1;
    j int :=2;
    k int;
begin
    plus(i,j);--不能直接执行
    --dbms_output.put_line(plus(i,j));
end;
```

2、创建函数的授权

> create procedure

```
SQL> select * from dba_sys_privs where grantee=upper('resource');--dba权限

GRANTEE                        PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
RESOURCE                       CREATE TRIGGER                           NO
RESOURCE                       CREATE SEQUENCE                          NO
RESOURCE                       CREATE TYPE                              NO
RESOURCE                       CREATE PROCEDURE                         NO
RESOURCE                       CREATE CLUSTER                           NO
RESOURCE                       CREATE OPERATOR                          NO
RESOURCE                       CREATE INDEXTYPE                         NO
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;

```
SQL> conn mike_/mike_
已连接。
SQL> grant execute on plus to mike1;

授权成功。

SQL> conn mike1/abc
已连接。
SQL> select mike_.plus(1,2) from dual;

MIKE_.PLUS(1,2)
---------------
              3
```

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

```
SQL> conn mike_/mike_
已连接。
SQL> revoke execute on plus from mike1;

撤销成功。

SQL> conn mike1/abc
已连接。
SQL> select mike_.plus(1,2) from dual;
select mike_.plus(1,2) from dual
             *
第 1 行出现错误:
ORA-00904: : 标识符无效
```

3、函数的参数、返回值不能设置精度

```
drop table t1;
create  or replace table t1(id int);--没有这种写法，删除表只能用drop命令

create or replace function plus(i number,j number) return int(3)
is
    z int;
begin
    z:=i+j;
    return z;
end;

--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
--在控制台，如何运行pl块,需要/加回车
```

4、数据字典表

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

```sql
create or replace function getEmpNameByNo(eno emp.empno%type)return emp.ename%type
is
    e_name emp.ename%type;
begin
    select ename into e_name from emp where empno=eno;
    return e_name;

    exception
        when no_data_found then return '未找到记录';
        when too_many_rows then return '返回多行';
        when others then return '执行出错';
end;
--from dual(参数是一个明确的值)
select getEmpNameByNo(7369) from dual;
--from 表(参数是该表中的某列)
select getEmpNameByNo(empno) from emp where empno=7369;

select to_char(sysdate,'yyyy') from dual;
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

```
create or replace function getEmpByNo(eno emp.empno%type)return emp%rowtype
is
    e emp%rowtype;
begin
    select * into e from emp where empno=eno;
    return e;

    exception
        when others then return null;--只能返回一个对象
end;

--通过select查询
--如果函数返回结果是rowtype,那么不能通过select进行查询
--select getEmpByNo(7369) from dual;
--只能使用pl块进行查询
declare
    e emp%rowtype;
    eno emp.empno%type:=&请输入雇员编号;
begin
    e:=getEmpByNo(eno);
    --检查出错
    --if e is null then
    --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
    if e.ename is null then

        dbms_output.put_line('查询出错');
    else
        dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);
    end if;

end;
```

6、定义无参函数

```
select sysdate from dual;

--定义无参函数
create or replace function hello return varchar
is
begin
    return 'hello itany';
end;

select hello() from dual;--自己定义的函数，是可以+括号的

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**

```
--根据部门编号，返回该部门下的所有雇员信息
select * from emp where deptno=10;
set serveroutput on;
create or replace function getEmpsByDeptno(dno emp.deptno%type)
return sys_refcursor
is
    emps sys_refcursor;
    cnt int;
begin
    select count(1)into cnt from emp where deptno=dno;
    if cnt =0 then
        return null;
    else
        open emps for select * from emp where deptno=dno;
        return emps;
    end if;


    exception
        when others then return null;

end;
--没有任何取其中内容的方法
select getEmpsByDeptno(10) from dual;

--返回动态游标的函数都是通过pl进行调用
declare
    cur sys_refcursor;
    e emp%rowtype;
begin
    cur:=getEmpsByDeptno(&请输入部门编号);
    loop
        fetch cur into e;
        exit when cur%notfound;
        dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);

    end loop;
    close cur;
    exception
        when others then dbms_output.put_line('未找到记录');


end;
```

8、在JDBC中访问函数

```java
@Test
//根据部门编号，返回雇员姓名(方式1)
public void testFun1() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  PreparedStatement pstmt = conn.prepareStatement("select getEmpNameByNo(?) from dual");
  pstmt.setInt(1,7369);
  ResultSet rs = pstmt.executeQuery();
  if(rs.next()){
    System.out.println(rs.getString(1));
  }
  rs.close();
  pstmt.close();
  conn.close();

}
//根据部门编号，返回雇员姓名(方式2)
@Test
public void testFun2() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  //预定义函数的调用格式
  CallableStatement call = conn.prepareCall("{?=call getEmpNameByNo(?) }");
  //声明函数各个部分的类型和参数说明
  //注册返回值参的类型
  call.registerOutParameter(1,Types.VARCHAR);
  //设置普通参数
  call.setInt(2,7369);
  //执行
  call.execute();
  String ename = call.getString(1);//获取第一个参数的返回值
  System.out.println(ename);
  call.close();
  conn.close();

}

//如果返回值是rowtype,目前jdbc不支持
//返回值是动态游标
@Test
public void testFun3() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  //预定义函数的调用格式
  CallableStatement call = conn.prepareCall("{?=call getEmpsByDeptno(?) }");
  //声明函数各个部分的类型和参数说明
  //注册返回值参的类型
  call.registerOutParameter(1,OracleTypes.CURSOR);
  //设置普通参数
  call.setInt(2,10);
```

```
  call.execute();

  try {
    Object obj = call.getObject(1);
    ResultSet rs=(ResultSet)(obj);
    while(rs.next()){
      System.out.println(rs.getInt("empno")+"\t"+rs.getString("ename")+
                         "\t"+rs.getFloat("sal"));
    }
  } catch (Exception e) {
    // TODO Auto-generated catch block
    //e.printStackTrace();
    System.out.println("查询出错");
  }
  call.close();
  conn.close();
}
```

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

```
--如果函数中有insert,update,delete等语句，
--根据雇员编号，给雇员加薪10%，返回加薪后的值
select * from emp;

create or replace function updateSal(eno emp.empno%type)
return emp.sal%type
is
    s  emp.sal%type;
begin
    update emp set sal=sal*1.1 where empno=eno;
    select sal into s from emp where empno=eno;
    return s;

    exception
        when no_data_found then return -1;
        when too_many_rows then return -2;
        when others then return -3;
end;

select * from emp;
select updateSal(7369) from dual;--error
--使用pl调用
declare
    eno emp.empno%type:=&请输入雇员编号;
    esal emp.sal%type;
begin
    esal:=updateSal(eno);
    if esal=-1 then
        dbms_output.put_line('查无此人');
    elsif esal=-2 then
        dbms_output.put_line('返回多行');
    elsif esal=-3 then
        dbms_output.put_line('查询出错');
    else
        dbms_output.put_line(esal);
    end if;

end;
```

```java
@Test
public void testFun4() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  //预定义函数的调用格式
  CallableStatement call = conn.prepareCall("{?=call updateSal(?) }");
  //声明函数各个部分的类型和参数说明
  //注册返回值参的类型
  call.registerOutParameter(1,Types.FLOAT);
  //设置普通参数
  call.setInt(2,7000);
  call.execute();

  float sal = call.getFloat(1);
  if(sal==-1){
    System.out.println("查无此人");
  }
  else if(sal==-2){
    System.out.println("返回多行");

  }
  else if(sal==-3){
    System.out.println("执行出错");

  }
  else{
    System.out.println(sal);
  }
  call.close();
  conn.close();

}
```

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

```
create or replace function saveEmp(ename emp.ename%type,
esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type)
return int
is
    eno int;
    dno_cnt int;
begin
    select count(deptno) into dno_cnt from emp where deptno=edeptno;
    if dno_cnt =0 then
        return -1;
    else
        select seq1.nextval into eno from dual;
        insert into
emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
        return eno;
    end if;
    exception
        when others then return -2;

end;
```

- 在pl中调用该函数

```
declare
    eno int;
begin
    eno:=saveEmp('mike',1000,sysdate,30);
    case eno
        when -1 then dbms_output.put_line('部门编号不存在');
        when -2 then dbms_output.put_line('查询出错');
        else
            dbms_output.put_line('新的雇员编号：'||eno);

    end case;
end;

select * from emp;
```

- 在jdbc中调用该函数

```java
@Test
public void testFun5() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  //预定义函数的调用格式
  CallableStatement call = conn.prepareCall("{?=call saveEmp(?,?,?,?) }");
  //声明函数各个部分的类型和参数说明
  //注册返回值参的类型
  call.registerOutParameter(1,Types.INTEGER);
  //设置普通参数
  call.setString(2,"rose");
  call.setFloat(3,2000);
  call.setTimestamp(4,Timestamp.valueOf("2017-3-3 10:10:10"));
  call.setInt(5,50);
  call.execute();

  int eno = call.getInt(1);
  if(eno==-1){
    System.out.println("部门编号不存在");
  }
  else if(eno==-2){
    System.out.println("查询出错");
  }
  else{
    System.out.println(eno);
  }


}
```

11、返回参

> 可以在函数的参数中定义返回参

```
hello(name varchar)===hello(name in varchar)--name只做输入,默认参数为输入参
hello(name out varchar)   ---name只能输出
hello(name in out varchar)--name即做输入，也做输出


create or replace function hello(name in out varchar) return varchar
is

begin
    name:=upper(name);
    return 'hello'||' '||name;
end;

declare
    n varchar(10):='tom';
    r varchar(100);--不设置类型精度只有两个地方：参数，返回值
begin
    r:=hello(n);
    dbms_output.put_line(n);--TOM
    dbms_output.put_line(r);
end;
```

> 除了返回雇员编号，还要返回该雇员所在部门的总人数

```
create or replace function saveEmp1(ename emp.ename%type,
esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type,dept_total_emps out
int)
return int
is
    eno int;
    dno_cnt int;
begin
    select count(deptno) into dno_cnt from emp where deptno=edeptno;
    if dno_cnt =0 then
        return -1;
    else
        select seq1.nextval into eno from dual;
        insert into
emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
        --设置返回参
        select count(deptno)into dept_total_emps from emp where deptno=edeptno;
        return eno;
    end if;
    exception
        when others then return -2;

end;
```

```
declare
    eno int;
    demps int;--返回参，该部门下雇员的人数
begin
    eno:=saveEmp1('mike1',1000,sysdate,30,demps);
    case eno
        when -1 then dbms_output.put_line('部门编号不存在');
        when -2 then dbms_output.put_line('查询出错');
        else
            dbms_output.put_line('新的雇员编号：'||eno||',该部门下已有'||demps||'位同事');

    end case;
end;

select count(*) from emp  where deptno=30;
```

```
@Test
public void testFun6() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  //预定义函数的调用格式
  CallableStatement call = conn.prepareCall("{?=call saveEmp1(?,?,?,?,?) }");
  //声明函数各个部分的类型和参数说明
  //注册返回值参的类型
  call.registerOutParameter(1,Types.INTEGER);
  //设置普通参数
  call.setString(2,"rose");
  call.setFloat(3,2000);
  call.setTimestamp(4,Timestamp.valueOf("2017-3-3 10:10:10"));
  call.setInt(5,10);
  call.registerOutParameter(6,Types.INTEGER);
  call.execute();

  int eno = call.getInt(1);
  if(eno==-1){
    System.out.println("部门编号不存在");
  }
  else if(eno==-2){
    System.out.println("查询出错");
  }
  else{
    System.out.println("新增雇员编号："+eno);
    System.out.println("该部门下已经有"+call.getInt(6)+"位同事");
  }


}
```

> 除了返回雇员编号，还要返回部门内雇员的信息

```
create or replace function saveEmp2(ename emp.ename%type,
esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type,dept_emps out
sys_refcursor)
return int
is
    eno int;
    dno_cnt int;
begin
    select count(deptno) into dno_cnt from emp where deptno=edeptno;
    if dno_cnt =0 then
        return -1;
    else
        select seq1.nextval into eno from dual;
        insert into
emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
        --设置返回参
        open dept_emps for select *  from emp where deptno=edeptno;
        return eno;
    end if;
    exception
        when others then return -2;

end;
```

```
declare
    eno int;
    emps sys_refcursor;--返回参，该部门下所有雇员的信息
    e emp%rowtype;
begin
    eno:=saveEmp2('mike2',1000,sysdate,30,emps);
    case eno
        when -1 then dbms_output.put_line('部门编号不存在');
        when -2 then dbms_output.put_line('查询出错');
        else
            dbms_output.put_line('新的雇员编号：'||eno);
            loop
                fetch emps into e;
                exit when emps%notfound;
                dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);

            end loop;
            close emps;--一定要关闭该游标

    end case;
end;

select empno,ename,sal from emp where deptno=30;
```

```java
@Test
public void testFun7() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  //预定义函数的调用格式
  CallableStatement call = conn.prepareCall("{?=call saveEmp2(?,?,?,?,?) }");
  //声明函数各个部分的类型和参数说明
  //注册返回值参的类型
  call.registerOutParameter(1,Types.INTEGER);
  //设置普通参数
  call.setString(2,"rose");
  call.setFloat(3,2000);
  call.setTimestamp(4,Timestamp.valueOf("2017-3-3 10:10:10"));
  call.setInt(5,10);
  call.registerOutParameter(6,OracleTypes.CURSOR);
  call.execute();

  int eno = call.getInt(1);
  if(eno==-1){
    System.out.println("部门编号不存在");
  }
  else if(eno==-2){
    System.out.println("查询出错");
  }
  else{
    System.out.println("新增雇员编号："+eno);
    ResultSet rs=(ResultSet)call.getObject(6);
    while(rs.next()){
      System.out.println(rs.getInt("empno")+"\t"+
                         rs.getString("ename")+"\t"+rs.getFloat("sal"));
    }
  }
}
```

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

```
create or replace procedure proHello(name varchar)
is
begin
    dbms_output.put_line('hello'||name);
end;

execute proHello('tom');--存储过程没有返回参，可以使用execute直接调用


create or replace procedure proHello1(name varchar,n_out out varchar)
is

begin
    dbms_output.put_line('hello'||name);
    n_out:=upper(name);
end;


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

```
declare

begin
    proHello('tom');
end;

declare
    n varchar(10);
begin
    proHello1('tom',n);--hellotom
    dbms_output.put_line(n);--TOM
end;

```

3、数据字典表

```
select * from user_functions;--不存在
select * from user_procedures;--包括函数和存储过程
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

```
create or replace procedure prcSaveEmp2(ename emp.ename%type,
esal emp.sal%type,ehiredate emp.hiredate%type,edeptno emp.deptno%type,dept_emps out
sys_refcursor,eno out int)

is
    dno_cnt int;
begin
    select count(deptno) into dno_cnt from emp where deptno=edeptno;
    if dno_cnt =0 then
        eno:=-1;
    else
        select seq1.nextval into eno from dual;
        insert into
emp(empno,ename,sal,hiredate,deptno)values(eno,ename,esal,ehiredate,edeptno);
        --设置返回参
        open dept_emps for select *  from emp where deptno=edeptno;
        eno:=eno;
    end if;
    exception
        when others then eno:=-2;

end;
```

```
select * from emp;
declare
    eno int;
    emps sys_refcursor;--返回参，该部门下所有雇员的信息
    e emp%rowtype;
begin
    prcSaveEmp2('mike3',1000,sysdate,30,emps,eno);
    case eno
        when -1 then dbms_output.put_line('部门编号不存在');
        when -2 then dbms_output.put_line('查询出错');
        else
            dbms_output.put_line('新的雇员编号：'||eno);
            loop
                fetch emps into e;
                exit when emps%notfound;
                dbms_output.put_line(e.empno||' '||e.ename||' '||e.sal);

            end loop;
            close emps;--一定要关闭该游标

    end case;
end;

select empno,ename,sal from emp where deptno=30;
```

5、在java中调用存储过程

```java
@Test
public void testProcedure1() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  //预定义函数的调用格式
  CallableStatement call = conn.prepareCall("{call prcSaveEmp2(?,?,?,?,?,?) }");
  //声明函数各个部分的类型和参数说明
  //设置普通参数
  call.setString(1,"rose1");
  call.setFloat(2,2000);
  call.setTimestamp(3,Timestamp.valueOf("2017-3-3 10:10:10"));
  call.setInt(4,10);
  //注册返回参：游标
  call.registerOutParameter(5,OracleTypes.CURSOR);

  //注册返回参：雇员编号
  call.registerOutParameter(6,Types.INTEGER);
  call.execute();

  int eno = call.getInt(6);
  if(eno==-1){
    System.out.println("部门编号不存在");
  }
  else if(eno==-2){
    System.out.println("查询出错");
  }
  else{
    System.out.println("新增雇员编号："+eno);
    ResultSet rs=(ResultSet)call.getObject(5);
    while(rs.next()){
      System.out.println(rs.getInt("empno")+"\t"+
                         rs.getString("ename")+"\t"+rs.getFloat("sal"));
    }
  }
  call.close();
  conn.close();
}
```

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

```
drop table t_test;
create table t_test(id int,name varchar(14));

begin
    for i in 1..100000
    loop
        insert into t_test values(i,'mike_'||i);
    end loop;
end;

select count(1) from t_test;
```

```
--查看表占有的表空间大小
select segment_name,bytes/1024||'k' from user_segments;
commit;

--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
--oracle会通过自己的算法，会在系统空闲(CPU,IO)的时候,会自动扫描磁盘空间中的无效数据，进行回收
delete from t_test where id<50000;
--truncate 可以自动收缩表空间
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

```
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

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

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

```
select t.* from(
select empno,ename,sal from emp) t;

create view v_1
as
select empno,ename,sal from emp;

select * from v_1;
```

3、视图的数据字典表

```
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

```
insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
values(111,'mike',sysdate,'SALE',7369,20);

select * from v_emp;
--10号部门与视图的查询冲突，但默认可以插入
insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
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

```
--drop view v_emp;
create or replace view v_emp
as
select empno,ename,job,mgr,hiredate,comm,deptno from emp
where deptno<>10
with check option;

insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
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。

```
select * from emp;
insert into v_emp(empno,ename,hiredate,job,mgr,deptno)
values(112,'mike1',sysdate,'SALE',7369,10);--报错，无效数据

update v_emp set ename=lower(ename) where ename='KING';--修改不了
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

```
create or replace view v_emp
as
select empno,ename,job,mgr,hiredate,comm,deptno from emp
where deptno<>10
with read only;

insert into v_emp(empno,ename,hiredate,job,mgr,deptno)

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

```
create table salgrade as select * from scott.salgrade;
create or replace view v_emp_infos
as
select e.empno,e.ename,e.sal,d.dname,g.grade
from emp e,dept d,salgrade g
where e.deptno=d.deptno
and e.sal between g.losal and g.HISAL;

select * from v_emp_infos;
```

该视图也可以作为子查询的连接表

```
select v.*,e.sal,e.comm
from emp e,V_EMP_INFOS v
where e.empno=v.empno;

select v.*,e.sal,e.comm
from emp e,V_EMP_INFOS v
where e.empno=v.empno;
--多表关联的视图无法进行增、删、改操作，只能查询
delete from V_EMP_INFOS where empno=7369;--报错
insert into V_EMP_INFOS (empno,ename)values(123,'jack');--报错
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

```
select * from v_1;
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

```
select to_char(sysdate,'day') from dual;
select to_char(sysdate,'hh24') from dual;

create or replace trigger tri_emp_dml
before update or delete on emp
declare
    --定义一个异常
    --oper_date_error exception;
begin
    --如果不是工作时间，抛异常
    if to_char(sysdate,'day')in('星期六','星期日') or to_char(sysdate,'hh24') not between 8
and 18 then
        --raise oper_date_error;
        --使用系统异常
        raise_application_error(-20001,'不允许在非工作日时间进行更新和删除操作');
    end if;
    --exception
        --when oper_date_error
        --  then dbms_output.put_line('不允许在非工作日时间进行更新和删除操作');

end;

select * from emp;
insert into emp(empno,ename)values(1,'mike');
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

```
select * from emp;

create or replace trigger tri_emp_dml
after insert or update or delete on emp
for each row
declare

begin
    if inserting then
        dbms_output.put_line('insert');
    elsif updating then
        dbms_output.put_line('update');
    elsif deleting then
        dbms_output.put_line('delete');
    end if;

end;

insert into emp(empno,ename)values(2,'rose');
delete from emp where deptno is null;

```

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

```
create or replace trigger tri_emp_dml
after insert or update or delete on emp
for each row
declare

begin
    if inserting then
        dbms_output.put_line('insert'||:new.empno||' '||:new.ename);
    elsif updating then
        dbms_output.put_line('update'||:old.ename||' '||:new.ename);
    elsif deleting then
        dbms_output.put_line('delete'||:old.empno||' '||:old.ename);
    end if;

end;

insert into emp(empno,ename)values(1,'mike');--insert1 mike
update emp set ename='张三' where empno=1;--updatemike 张三
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

```
create or replace trigger tri_emp_dml
before insert or update or delete on emp
for each row
declare

begin
    if inserting then
        dbms_output.put_line('insert'||:new.empno||' '||:new.ename);
        :new.ename:=upper(:new.ename);
    elsif updating then
        dbms_output.put_line('update'||:old.ename||' '||:new.ename);
        :new.ename:=upper(:new.ename);
    elsif deleting then
        dbms_output.put_line('delete'||:old.empno||' '||:old.ename);

    end if;

end;

insert into emp(empno,ename)values(9001,'hello1');--90001  hello1

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

```
select count(1) from emp;

create or replace trigger tri_emp_dml
before insert or update or delete on emp
for each row
declare
    cnt int;
begin
    if inserting then
        dbms_output.put_line('insert'||:new.empno||' '||:new.ename);
        select count(1) into cnt from emp;--成功
        dbms_output.put_line(cnt);--插入之前的记录数，比实际值-1
    elsif updating then
        dbms_output.put_line('update'||:old.ename||' '||:new.ename);
        select count(1) into cnt from emp;--报错
        dbms_output.put_line(cnt);
    elsif deleting then
        dbms_output.put_line('delete'||:old.empno||' '||:old.ename);
        select count(1) into cnt from emp;--报错
        dbms_output.put_line(cnt);
    end if;

end;

insert into emp(empno,ename)values(9002,'rose');
update emp set ename=upper(ename) where empno=9002;
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)

```
select * from emp;
select * from emp_bak;
rollback;
delete from emp where deptno=10;
select * from emp_bak;
create table emp_bak as select * from emp where 1=2;

alter table emp_bak add(username varchar(20),oper_d date,oper_type varchar(10));


create or replace trigger trig_emp_del_bak
before delete on emp
for each row
declare
begin
    if deleting then
        --将删除的数据备份到备份表中
        --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
        insert into emp_bak values(:old.empno,:old.ename,:old.job,:old.mgr,:old.hiredate,:old.sal,:old.comm,:old.deptno,user,sysdate,'delete');
    end if;
end;

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

```
truncate table emp;
select * from emp;
select count(empno) from emp;--0
select max(empno) from emp;--null


create or replace trigger trig_emp_autoid
before insert on emp
for each row
declare
    cnt int;
begin
    select max(empno) into cnt from emp;
    if cnt is null then
        :new.empno:=1;
    else
        :new.empno:=cnt+1;
    end if;
end;

insert into emp (ename,sal)values('李四1',2000);
```

6、禁用，启用触发器

```
--禁用某一个触发器（行级，表级触发器）
alter trigger trig_emp_autoid disable;

--启用某一个触发器（行级，表级触发器）
alter trigger trig_emp_autoid enable;

--禁用某个表上的所有触发器(行级，表级触发器)
alter table emp disable all triggers;
--启用某个表上的所有触发器(行级，表级触发器)
alter table emp enable all triggers;
```