



# ORACLE 数据库

## 第 4 章

### ORACLE 编程

传智播客.黑马程序员



## 一、PL/SQL

### (一) 什么是 PL/SQL

PL/SQL ( Procedure Language/SQL ) 是 Oracle 对 sql 语言的过程化扩展，指在 SQL 命令语言中增加了过程处理语句（如分支、循环等），使 SQL 语言具有过程处理能力。把 SQL 语言的数据操纵能力与过程语言的数据处理能力结合起来，使得 PLSQL 面向过程但比过程语言简单、高效、灵活和实用。

#### 基本语法结构

```
[declare  
    --声明变量  
]  
begin  
    --代码逻辑  
[exception  
    --异常处理  
]  
end;
```

### (二) 变量

声明变量的语法：

变量名 类型（长度）；

变量赋值的语法：

变量名:=变量值

变量的声明



需求：

声明变量水费单价、水费字数、吨数、金额。

对水费单价、字数、进行赋值。吨数根据水费字数换算，规则为水费字数除以1000，并且四舍五入，保留两位小数。计算金额，金额=单价\*吨数。

输出单价、数量和金额。

```
--变量的用法--
declare
    v_price    number(10,2);--水费单价
    v_usenum   number;      --水费字数
    v_usenum2  number(10,2);--吨数
    v_money    number(10,2);--金额
begin
    v_price:=2.45;--水费单价
    v_usenum:=8012;--字数
    --字数换算为吨数
    v_usenum2:= round( v_usenum/1000,2);
    --计算金额
    v_money:=round(v_price*v_usenum2,2);
    dbms_output.put_line('单价：'||v_price||'吨
数：'||v_usenum2||'金额：'||v_money);
end;
```

Select into 方式 赋值

语法结构：



```
select 列名 into 变量名 from 表名 where 条件
```

注意：结果必须是一条记录，有多条记录 and 没有记录都会报错

```
declare
    v_price number(10,2); --单价
    v_usenum number; --水费字数
    v_num0 number; --上月字数
    v_num1 number; --本月字数
    v_usenum2 number(10,2); --使用吨数
    v_money number(10,2); --水费金额
begin
    --对单价进行赋值
    v_price:=3.45;
    --变量赋值
    select usenum,num0,num1 into v_usenum,v_num0,v_num1 from
T_ACCOUNT
    where year='2012' and month='01' and owneruuid=1;

    v_usenum2:= round(v_usenum/1000,2);
    v_money:=v_price*v_usenum2;

    DBMS_OUTPUT.put_line('单价：'||v_price||'吨数：'
||v_usenum2||'金额：'||v_money||'上月字数：'||v_num0||'本月
字数'||v_num1);
end;
```

### (三) 属性类型

%TYPE 引用型



作用：引用某表某列的字段类型

```
declare

v_price number(10,2);--单价

v_usenum T_ACCOUNT.USENUM%TYPE;--水费字数

v_num0 T_ACCOUNT.NUM0%TYPE;--上月字数

v_num1 T_ACCOUNT.NUM1%TYPE;--本月字数

v_usenum2 number(10,2);--使用吨数

v_money number(10,2);--水费金额
begin

--对单价进行赋值
v_price:=3.45;
--v_usenum:=8090;
select usenum,num0,num1 into v_usenum,v_num0,v_num1 from
T_ACCOUNT
where year='2012' and month='01' and owneruuid=1;

--使用吨数
v_usenum2:= round(v_usenum/1000,2);

--计算金额
v_money:=v_price*v_usenum2;

DBMS_OUTPUT.put_line('单价：'||v_price||'吨数：'

||v_usenum2||'金额：'||v_money||'上月字数：'||v_num0||'本月
字数'||v_num1);

end;
```

%ROWTYPE 记录型 ，上例中的例子可以用下面的代码代替

作用：标识某个表的行记录类型



--变量的用法--

declare

v\_price number(10,2);--单价

v\_account T\_ACCOUNT%ROWTYPE;--记录型

v\_usenum2 number(10,2);--使用吨数

v\_money number(10,2);--水费金额

begin

--对单价进行赋值

v\_price:=3.45;

--赋值

select \* into v\_account from T\_ACCOUNT  
where year='2012' and month='01' and owneruuid=1;

--使用吨数

v\_usenum2:= round(v\_account.usenum/1000,2);

--计算金额

v\_money:=v\_price\*v\_usenum2;

DBMS\_OUTPUT.put\_line('单价：'||v\_price||'吨数：'

||v\_usenum2||'金额：'||v\_money||'上月字数：

'||v\_account.num0||'本月字数'||v\_account.num1);

end;

## (四) 异常

在运行程序时出现的错误叫做异常

发生异常后，语句将停止执行，控制权转移到 PL/SQL 块的异常处理部分

异常有两种类型：

预定义异常 - 当 PL/SQL 程序违反 Oracle 规则或超越系统限制时隐式引发

用户定义异常 - 用户可以在 PL/SQL 块的声明部分定义异常，自定义的异常通过 RAISE 语句显式引发

## 预定义异常

Oracle 预定义异常 21 个

命名的系统异常	产生原因
ACCESS_INTO_NULL	未定义对象
CASE_NOT_FOUND	CASE 中若未包含相应的 WHEN，并且没有设置 ELSE 时
COLLECTION_IS_NULL	集合元素未初始化
CURSER_ALREADY_OPEN	游标已经打开
DUP_VAL_ON_INDEX	唯一索引对应的列上有重复的值
INVALID_CURSOR	在不合法的游标上进行操作
INVALID_NUMBER	内嵌的 SQL 语句不能将字符转换为数字
NO_DATA_FOUND	使用 select into 未返回行
TOO_MANY_ROWS	执行 select into 时，结果集超过一行
ZERO_DIVIDE	除数为 0
SUBSCRIPT_BEYOND_COUNT	元素下标超过嵌套表或 VARRAY 的最大值
SUBSCRIPT_OUTSIDE_LIMIT	使用嵌套表或 VARRAY 时，将下标指定为负数
VALUE_ERROR	赋值时，变量长度不足以容纳实际数据
LOGIN_DENIED	PL/SQL 应用程序连接到 oracle 数据库时，提供了不正确的用户名或密码
NOT_LOGGED_ON	PL/SQL 应用程序在没有连接 oracle 数据库的情况下访问数据
PROGRAM_ERROR	PL/SQL 内部问题，可能需要重装数据字典& pl./SQL 系统包
ROWTYPE_MISMATCH	宿主游标变量与 PL/SQL 游标变量的返回类型不兼容
SELF_IS_NULL	使用对象类型时，在 null 对象上调用对象方法
STORAGE_ERROR	运行 PL/SQL 时，超出内存空间
SYS_INVALID_ID	无效的 ROWID 字符串
TIMEOUT_ON_RESOURCE	Oracle 在等待资源时超时



语法结构：

```
exception

when 异常类型 then

    异常处理逻辑
```

根据上例中的代码，添加异常处理部分

```
--变量的用法--
declare

v_price    number(10,2);--水费单价

v_usenum   T_ACCOUNT.USENUM%type; --水费字数

v_usenum2  number(10,3);--吨数

v_money    number(10,2);--金额
begin

v_price:=2.45;--水费单价

select usenum into v_usenum from T_ACCOUNT where
owneruuid=1 and year='2012' and month='01';

--字数换算为吨数
v_usenum2:= round( v_usenum/1000,3);

--计算金额
v_money:=round(v_price*v_usenum2,2);

dbms_output.put_line('单价：'||v_price||'吨
数：'||v_usenum2||'金额：'||v_money);
exception
when NO_DATA_FOUND then

    dbms_output.put_line('未找到数据，请核实');

when TOO_MANY_ROWS then
```





```
dbms_output.put_line('查询条件有误，返回多条信息，请核实');  
end;
```

## (五) 条件判断

### 基本语法 1

```
if 条件 then  
    业务逻辑  
end if;
```

### 基本语法 2

```
if 条件 then  
    业务逻辑  
else  
    业务逻辑  
end if;
```

### 基本语法 3

```
if 条件 then  
    业务逻辑  
elsif 条件 then  
    业务逻辑  
else  
    业务逻辑  
end if;
```

需求 :设置三个等级的水费 5 吨以下 2.45 元/吨 5 吨到 10 吨部分 3.45 元/吨 ,  
超过 10 吨部分 4.45 , 根据使用水费的量来计算阶梯水费。

```
declare  
  
    v_price1 number(10,2);--不足 5 吨的单价  
  
    v_price2 number(10,2);--超过 5 吨不足 10 吨单价  
  
    v_price3 number(10,2);--超过 10 吨单价
```



```
v_account T_ACCOUNT%ROWTYPE;--记录型

v_usenum2 number(10,2);--使用吨数

v_money number(10,2);--水费金额
begin

--对单价进行赋值
v_price1:=2.45;
v_price2:=3.45;
v_price3:=4.45;

--赋值
select * into v_account from T_ACCOUNT
where year='2012' and month='01' and owneruuid=1;

--使用吨数
v_usenum2:= round(v_account.usenum/1000,2);

--计算金额(阶梯水费)

if v_usenum2<=5 then--第一个阶梯
    v_money:=v_price1*v_usenum2;
elsif v_usenum2>5 and v_usenum2<=10 then --第二个阶梯
    v_money:=v_price1*5 + v_price2*( v_usenum2-5);
else --第三个阶梯
    v_money:=v_price1*5 +v_price2*5 +
v_price3*( v_usenum2-10 );
end if;

DBMS_OUTPUT.put_line('吨数：'

||v_usenum2||'金额：'||v_money||'上月字数：'

' ||v_account.num0||'本月字数'||v_account.num1);
```



```
exception

    when NO_DATA_FOUND then

        DBMS_OUTPUT.put_line('没有找到数据');

    when TOO_MANY_ROWS then

        DBMS_OUTPUT.put_line('返回的数据有多行');

end;
```

## (六) 循环

### 1. 无条件循环

#### 语法结构

```
loop
    --循环语句
end loop;
```

范例：输出从1开始的100个数

```
declare
v_num number:=1;
begin
    loop
        dbms_output.put_line(v_num);
        v_num:=v_num+1;
        exit when v_num>100;
    end loop;
end ;
```

### 2、条件循环

#### 语法结构

```
while 条件
loop
```



```
end loop;
```

范例：输出从1开始的100个数

```
declare
v_num number:=1;
begin
    while v_num<=100
    loop
        dbms_output.put_line(v_num);
        v_num:=v_num+1;
    end loop;
end ;
```

### 3、for循环

#### 基本语法

```
for 变量 in 起始值..终止值
loop

end loop;
```

范例：输出从1开始的100个数

```
begin
    for v_num in 1..100
    loop
        dbms_output.put_line(v_num);
    end loop;
end;
```

## （七）游标

### 1.什么是游标

游标是系统为用户开设的一个数据缓冲区,存放 SQL 语句的执行结果。我们可以把游标理解为 PL/SQL 中的结果集。



## 2.语法结构及示例

在声明区声明游标，语法如下：

```
cursor 游标名称 is SQL 语句;
```

使用游标语法

```
open 游标名称
```

```
loop
```

```
    fetch 游标名称 into 变量
```

```
    exit when 游标名称%notfound
```

```
end loop;
```

```
close 游标名称
```

需求：打印业主类型为 1 的价格表

代码：

```
declare

    v_pricetable T_PRICETABLE%rowtype; --价格行对象

    cursor cur_pricetable is select * from T_PRICETABLE where
    ownertypeid=1; --定义游标

begin
```



```

open cur_pricetable;--打开游标
loop
    fetch cur_pricetable into v_pricetable;--提取游标到变量

    exit when cur_pricetable%notfound;--当游标到最后一行下面退出循环

    dbms_output.put_line( '价格:'

        ||v_pricetable.price ||'吨位 :

        '||v_pricetable.minnum||'-'||v_pricetable.maxnum );
end loop;

close cur_pricetable;--关闭游标
end ;

```

运行结果如下：

```

2.45  0-5
3.45  5-10
4.45  10-

```

### 3.带参数的游标

我们的查询语句的条件值有可能是在运行时才能决定的，比如性业主类型，可能是运行时才可以决定，那如何实现呢？我们接下来学习带参数的游标，修改上述案例

```

declare

    v_pricetable T_PRICETABLE%rowtype;--价格行对象

    cursor cur_pricetable(v_ownertypeid number) is select *
from T_PRICETABLE where ownertypeid=v_ownertypeid;--定义游标

begin

```



```
open cur_pricetable(2);--打开游标
loop
    fetch cur_pricetable into v_pricetable;--提取游标到变量
    exit when cur_pricetable%notfound;--当游标到最后一行下面退出循环
    dbms_output.put_line('价格:'||v_pricetable.price ||'吨位: '||v_pricetable.minnum||'-'||v_pricetable.maxnum );
end loop;
close cur_pricetable;--关闭游标
end ;
```

#### 4. for 循环提取游标值

我们每次提取游标,需要打开游标 关闭游标 循环游标 提取游标 控制循环的退出等等,好麻烦!有没有更简单的写法呢?有!用 for 循环一切都那么简单,上例的代码可以改造为下列形式

```
declare
    cursor cur_pricetable(v_ownertypeid number) is select *
from T_PRICETABLE where ownertypeid=v_ownertypeid;--定义游标
begin
    for v_pricetable in cur_pricetable(3)
    loop
        dbms_output.put_line('价格:'||v_pricetable.price ||'吨位: '||v_pricetable.minnum||'-'||v_pricetable.maxnum );
    end loop;
end ;
```



## 二、存储函数

### （一）什么是存储函数

存储函数又称为自定义函数。可以接收一个或多个参数，返回一个结果。

在函数中我们可以使用 P/SQL 进行逻辑的处理。

### （二）存储函数语法结构

创建或修改存储过程的语法如下：

```
CREATE [ OR REPLACE ] FUNCTION 函数名称  
  
    ( 参数名称 参数类型, 参数名称 参数类型, ... )  
  
RETURN 结果变量数据类型  
  
IS  
  
    变量声明部分;  
  
BEGIN  
  
    逻辑部分;  
  
    RETURN 结果变量;  
  
[EXCEPTION  
  
    异常处理部分]  
  
END;
```





### (三) 案例

需求：创建存储函数，根据地址 ID 查询地址名称。

语句:

```
create function fn_getaddress(v_id number)
return varchar2
is
    v_name varchar2(30);
begin
    select name into v_name from t_address where id=v_id;
    return v_name;
end;
```

测试此函数：

```
select fn_getaddress(3) from dual
```

输出内容

	FN_GETADDRESS(3)
1	华龙苑南里小区

需求：查询业主 ID，业主名称，业主地址，业主地址使用刚才我们创建的函数来实现。

```
select id 编号, name 业主名称, fn_getaddress(addressid) 地址
from t_owners
```

查询结果如下：

	业主编号	业主名称	地址
1	1	范小冰	明兴花园
2	2	王强	明兴花园
3	3	马腾	明兴花园
4	4	林玲玲	鑫源秋墅
5	5	刘华	鑫源秋墅
6	6	刘东	鑫源秋墅
7	7	周健	华龙苑南里小区
8	8	张哲	河畔花园
9	9	昌平区中西医结合医院	霍营
10	10	美廉美超市	霍营



## 三、存储过程

### （一）什么是存储过程

存储过程是被命名的 PL/SQL 块，存储于数据库中，是数据库对象的一种。

应用程序可以调用存储过程，执行相应的逻辑。

存储过程与存储函数都可以封装一定的业务逻辑并返回结果，存在区别如下：

- 1、存储函数中有返回值，且必须返回；而存储过程没有返回值，可以通过传出参数返回多个值。
- 2、存储函数可以在 `select 语句` 中直接使用，而存储过程不能。过程多数是被应用程序所调用。
- 3、存储函数一般都是封装一个查询结果，而存储过程一般都封装一段事务代码。

### （二）存储过程语法结构

创建或修改存储过程的语法如下：

```
CREATE [ OR REPLACE ] PROCEDURE 存储过程名称  
  
    ( 参数名 类型, 参数名 类型, 参数名 类型 )  
  
IS|AS  
  
    变量声明部分;
```



```
BEGIN
```

逻辑部分

```
[EXCEPTION
```

异常处理部分]

```
END;
```

参数只指定类型，不指定长度

过程参数的三种模式：

IN 传入参数（默认）

OUT 传出参数，主要用于返回程序运行结果

IN OUT 传入传出参数

### （三）案例

#### 1. 创建不带传出参数的存储过程：添加业主信息

```
--增加业主信息序列
create sequence seq_owners start with 11;

--增加业主信息存储过程
create or replace procedure pro_owners_add
(
    v_name varchar2,
    v_addressid number,
    v_housenumber varchar2,
    v_watermeter varchar2,
    v_type number
)
is
begin
```



```
insert into T_OWNERS
values( seq_owners.nextval,v_name,v_addressid,v_housenumb
er,v_watermeter,sysdate,v_type );
commit;
end;
```

### PL/SQL 中调用存储过程

```
call pro_owners_add('赵伟',1,'999-3','132-7',1);
```

### JDBC 调用存储过程

```
/**
 * 增加
 * @param owners
 */
public static void add(Owners owners){

    java.sql.Connection conn=null;
    java.sql.CallableStatement stmt=null;

    try {
        conn=BaseDao.getConnection();
        stmt=conn.prepareCall("{call
pro_owners_add(?,?,?,?,?)}");

        stmt.setString(1, owners.getName());
        stmt.setLong(2, owners.getAddressid());
        stmt.setString(3, owners.getHousenumber());
        stmt.setString(4, owners.getWatermeter());
        stmt.setLong(5, owners.getOwnertypeid());

        stmt.execute();
    } catch (SQLException e) {

        e.printStackTrace();
    }finally {
        BaseDao.closeAll(null, stmt, conn);
    }
}
```



## 2 创建带传出参数的存储过程

需求：添加业主信息，传出参数为新增业主的 ID

```
--增加业主信息存储过程

create or replace procedure pro_owners_add
(
    v_name varchar2,
    v_addressid number,
    v_housenumber varchar2,
    v_watermeter varchar2,
    v_type number,
    v_id out number
)
is
begin
    select seq_owners.nextval into v_id from dual;
    insert into T_OWNERS
values( v_id,v_name,v_addressid,v_housenumber,v_watermete
r,sysdate,v_type );
    commit;
end;
```

PL/SQL 调用该存储过程

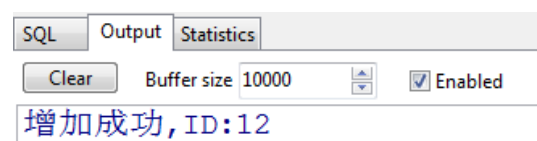
```
declare

    v_id number;--定义传出参数的变量
begin

    pro_owners_add('王旺旺',1,'922-3','133-7',1,v_id);

    DBMS_OUTPUT.put_line('增加成功,ID:'||v_id);
end;
```

执行成功后输出结果：



The screenshot shows the 'Output' tab of a SQL tool. It contains a 'Clear' button, a 'Buffer size' of 10000, and an 'Enabled' checkbox. Below these, the text '增加成功, ID:12' is displayed in blue.

JDBC 调用存储过程



```
/**
 * 增加
 * @param owners
 */
public static long add(Owners owners){
    long id=0;
    java.sql.Connection conn=null;
    java.sql.CallableStatement stmt=null;
    try {
        conn=BaseDao.getConnection();
        stmt=conn.prepareCall("{call
pro_owners_add(?,?,?,?,?,?)}");
        stmt.setString(1, owners.getName());
        stmt.setLong(2, owners.getAddressid());
        stmt.setString(3, owners.getHousenumber());
        stmt.setString(4, owners.getWatermeter());
        stmt.setLong(5, owners.getOwnertypeid());
        stmt.registerOutParameter(6, OracleTypes.NUMBER);//注
册传出参数类型
        stmt.execute();
        id=stmt.getLong(6);//提取传出参数
    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        BaseDao.closeAll(null, stmt, conn);
    }
    return id;
}
```

## 四、触发器

### (一) 什么是触发器

数据库触发器是一个与表相关联的、存储的 PL/SQL 程序。每当一个特定的数据操作语句(Insert,update,delete)在指定的表上发出时，Oracle 自动地执行触发器中定义的语句序列。



## 触发器可用于

- 数据确认
- 实施复杂的安全性检查
- 做审计，跟踪表上所做的数据操作等
- 数据的备份和同步

## 触发器分类

- 前置触发器（BEFORE）
- 后置触发器（AFTER）

## （二）创建触发器的语法

语法：

```
CREATE [or REPLACE] TRIGGER 触发器名
    BEFORE | AFTER
    [DELETE ] [[or] INSERT] [[or]UPDATE [OF 列名]]
    ON 表名
    [FOR EACH ROW ] [WHEN (条件) ]
declare
    .....
begin
    PLSQL 块
End ;
```

FOR EACH ROW 作用是标注此触发器是行级触发器      语句级触发器

在触发器中触发语句与**伪记录变量**的值

触发语句	:old	:new
Insert	所有字段都是空 (null)	将要插入的数据
Update	更新以前该行的值	更新后的值



delete

删除以前该行的值

所有字段都是空(null)

## (三) 案例

### 1. 前置触发器

需求：当用户输入本月累计表数后，自动计算出本月使用数。

代码：

```
create or replace trigger tri_account_updatenum1
before
update of num1
on t_account
for each row
declare
begin
    :new.usenum:=:new.num1-:new.num0;
end;
```

### 2. 后置触发器

需求：当用户修改了业主信息表的数据时记录修改前与修改后的值

*--创建业主名称修改日志表:用于记录业主更改前后的名称*

```
create table t_owners_log
(
    updatetime date,
    ownerid number,
    oldname varchar2(30),
    newname varchar2(30)
);
```

*--创建后置触发器，自动记录业主更改前后日志*

```
create trigger tri_owners_log
after
update of name
on t_owners
```





```
for each row
declare

begin
    insert into t_owners_log
values(sysdate,:old.id,:old.name,:new.name);
end;
```

测试：

```
--更新数据

update t_owners set name='杨小花' where id=3;
commit;

--查询日志表

select * from t_owners_log;
```

## 五、综合案例

1. 编写 PL/SQL ，用水吨数 12 吨，业主类型为 1，计算阶梯水费。

思路分析：

水费是实行阶梯计算的，我们查询价格表中业主类型为 1 的水费价格记录

1	select * from t_pricetable where ownertypeid=1 order by minnum					
	ID	PRICE	OWNERTYPEID	MINNUM	MAXNUM	
1	1	2.45	1	0	5	
2	2	3.45	1	5	10	
3	3	4.45	1	10		

minnum 为下限值 ， maxnum 为上限值。上边的记录的含义是

5 吨以下的价格为 2.45

超过 5 吨不足 10 吨的价格为 3.45

超过 10 吨以上的价格为 4.45



如果吨数为 12。计算如下：

价格	下限值	上限值	吨数	金额
2.45	0	5	5	12.25
3.45	5	10	5	17.25
4.45	10		2	8.9
合计			12	38.4

考虑到阶梯的层次可能是不确定的，所以我们需要通过游标查询出阶梯价格记录，然后计算每一阶梯的水费，然后相加。伪代码如下：

```
金额=0  
  
循环价格表{  
  
    if( 上限值为空 或者 总吨数<上限值 ) -- 最高阶梯  
    {  
  
        //此为最后阶梯 ，数量为超过上限值部分的吨数  
  
        金额=金额+ 价格* ( 总吨数- 上限值 )  
  
        退出循环  
    }  
    else  
    {  
  
        //此为最后阶梯 ，数量为区间内的吨数  
  
        金额=金额+ 价格* ( 上限值- 下限值 )  
  
    }  
  
}
```

语句：

```
declare  
  
    v_ownertypeid number; -- 业主类型 ID
```



```
v_usenum2 number(10,2);--总吨数

v_money number(10,2);--总金额

cursor cur_pricetable(v_type number) is select * from
t_pricetable where ownertypeid=v_type;--价格游标

v_pricetable t_pricetable%rowtype;--每阶梯价格对象

begin

v_ownertypeid:=1;
v_usenum2:=12;
v_money:=0;

for v_pricetable in cur_pricetable(v_ownertypeid)
loop
    if v_pricetable.maxnum is null or
v_usenum2<=v_pricetable.maxnum then

        --最后阶梯    (总吨数-下限值)*价格

        v_money:=v_money+
v_pricetable.price*(v_usenum2-v_pricetable.minnum);
        exit;
    else

        --非最后阶梯    (上限值-下限值)* 价格

        v_money:=v_money+
v_pricetable.price*(v_pricetable.maxnum-v_pricetable.minn
um);
    end if;

end loop;

DBMS_OUTPUT.put_line('阶梯水费金额：'||v_money);

end;
```

## 2. 存储函数综合案例：创建计算阶梯水费的函数，参数为业主类型、吨数。

```
create or replace function fn_calmoney(v_ownertypeid
```



```
number,v_usenum2 number)
return number
is

    v_pricetable t_pricetable%rowtype;--价格行对象

    v_money number(10,2);--金额

    cursor cur_pricetable(v_type number) is select * from
t_pricetable where ownertypeid=v_type order by minnum;--定义游标

begin

    v_money:=0;--金额

    for v_pricetable in cur_pricetable(v_ownertypeid)
    loop

        --计算阶梯水费

        --如果水费小于最大值，或最大值为null 表示此阶梯为最后一个阶梯，

        --价格*（总吨数-此阶梯下限值）

        if v_usenum2<= v_pricetable.maxnum or
v_pricetable.maxnum is null then
            v_money:=v_money+ v_pricetable.price* ( v_usenum2 -
v_pricetable.minnum);
            exit;

        else -- 价格*（此阶梯上限值-此阶梯下限值）

            v_money:=v_money+ v_pricetable.price*
(v_pricetable.maxnum-v_pricetable.minnum );
        end if;
    end loop;
    return v_money;
end;
```

测试此函数：

```
select fn_calmoney(1,12) from dual;
```



### 3. 触发器综合案例：当用户输入本月累计数后，自动计算阶梯水费。

```
create or replace trigger tri_account_updatenum1
before
update of num1
on t_account
for each row
declare

    v_usenum2 number(10,2);--吨数
begin

    --使用数赋值

    :new.usenum:=:new.num1-:new.num0;

    v_usenum2:= round( :new.usenum/1000,3);--计算吨数

    :new.money:=fn_calmoney(:new.ownertype,v_usenum2);--对金额列赋值
end ;
```

修改某记录，观察结果。

### 4. 存储过程综合案例。

需求：增加业主信息时，同时在账务表（account）增加一条记录，年份与月份为当前日期的年月，初始值（num0）为0,其它字段信息（区域）与t\_owners表一致

难点分析：

1. 如何取得年和月 用to\_char()函数
2. 如何取得区域ID 参数中没有直接提供区域ID，我们可以通过addressid到address表查询

创建存储过程语句：

```
create or replace procedure pro_owners_add
```



```
(
    v_name varchar2,
    v_addressid number,
    v_housenumber varchar2,
    v_watermeter varchar2,
    v_type number,
    v_ownersuuid out number
)
is

    v_area number;--区域编号

    v_year char(4);--年份

    v_month char(2);--月份
begin
    --提取序列值到变量
    select seq_owners.nextval into v_ownersuuid from dual;

    --根据地址编号查询区域编号
    select areaid into v_area from t_address where
id=v_addressid;

    --年份
    v_year:=to_char(sysdate , 'yyyy');

    --月份
    v_month:=to_char(sysdate, 'mm');

    --增加业主信息
    insert into t_owners
values( v_ownersuuid,v_name,v_addressid,v_housenumber,v_w
atermeter,sysdate,v_type );

    --增加账务表信息
    insert into t_account
(id,owneruuid,ownertype,areaid,year,month,num0)
values
(seq_account.nextval,v_ownersuuid,v_type,v_area,v_year,
v_month,0 );
    commit;
```



```
exception
when NO_DATA_FOUND then
    v_ownersuuid:=-1;
    rollback;
end;
```

## 六、总结

### (一) 知识点总结

### (二) 上机任务布置