PL/SQL简介

1. 简介

- PL/SQL也是一种程序语言，叫做过程化SQL语言(Procedural Language/SQL)。PL/SQL是Oracle数据库对标准数据库语言SQL的扩展。
- 在普通SQL语句的使用上增加了编程语言的特点，所以PL/SQL就是把数据操作和查询语句组织在PL/SQL代码的过程性单元中，通过逻辑判断、循环等操作实现复杂的功能或者计算的程序语言。

声明部分、执行部分、异常处理部分

```
DECLARE
/* 声明部分：在此声明PL/SQL用到的变量，类型及游标，以
   及局部的存储过程和函数 */
BEGIN
    /*执行部分：过程及SQL语句，即程序的主要部分*/
EXCEPTION
    /* 执行异常部分：错误处理*/
END;
```

2. 变量及数据类型

（1）变量声明

在declare部分声明变量，格式如下：

declare

变量名　数据类型；

[变量名　数据类型；……]

（2）变量赋值

变量赋值格式如下：

变量名:＝常量或表达式

- 可以在声明变量的同时给变量赋值，也可以在执行部分单独给变量赋值。

```
DECLARE
    v_sno VARCHAR2(10) :='04001';
    v_cno VARCHAR2(10) :='001';
    v_grade NUMBER :=90;
BEGIN
    INSERT INTO SC VALUES (v_sno,v_cno,v_grade);
    CONNIT;
END;
```

2. 变量及数据类型

- 除了常用的数据类型外，在Oracle中还有两个比较特殊的数据类型：%TYPE 和%ROWTYPE 。
- %TYPE
  - 定义一个变量，其数据类型与已经定义的某个数据变量的类型相同，或者与数据库表的某个列的数据类型相同，这时可以使用%TYPE。
  - 使用%TYPE特性的优点在于：
    - 所引用的数据库列的数据类型可以不必知道；
    - 所引用的数据库列的数据类型可以实时改变。

练习实例

```
DECLARE
    m sc.sno%type;
    n sc.cno%type;
    t sc.grade%type;//没有必要提前知道数据列的数据类型
BEGIN
    m='04001';
    n='001';
    t='90';

    INSERT
    INTO SC
    VALUES (m,n,t);
    CONNIT;
END;
```

2. 变量及数据类型

- %ROWTYPE
  - PL/SQL提供%ROWTYPE操作符，返回一个记录类型，其数据类型和数据库表的数据结构相一致。
  - 使用%ROWTYPE特性的优点在于：
    - 所引用的数据库中列的个数和数据类型可以不必知道；
    - 所引用的数据库中列的个数和数据类型可以实时改变。

```
DECLARE
    m sc.sno%rowtype;
BEGIN
    m.sno='04002';
    m.cno='001';
    m.grade='91';
    INSERT
    INTO SC
    VALUES (m.sno,m.cno,m.grade);
    CONNIT;
END;
```

运算符与表达式

一般运算符

3. 运算符与表达式

- 一般运算符

运算符	意义
+	加号
-	减号
*	乘号
/	除号
:=	赋值号
=>	关系号
..	范围运算符
||	字符连接符

关系符号作用：调用存储过程中为形参赋值

关系运算符

3. 运算符与表达式

- 关系运算符

运算符	意义
=	等于
<> , != , ~= , ^=	不等于
<	小于
>	大于
<=	小于或等于
>=	大于或等于

逻辑运算符

- 逻辑运算符

运算符	意义
IS NULL	是空值
BETWEEN	介于两者之间
IN	在一列值中间
AND	逻辑与
OR	逻辑或
NOT	取返,如IS NOT NULL, NOT IN

流程控制语句

4. pl/sql流程控制语句

- PL/SQL的流程控制语句，包括如下三类:
- 顺序语句:
  - GOTO语句：无条件跳转到指定的标号
  - NULL语句：说明“不用做任何事情”的意思，相当于一个占位符
- 条件语句: IF 语句
- 循环语句:
  - LOOP…END LOOP语句
  - WHILE语句
  - FOR循环语句

条件语句

条件语句

```
IF <布尔表达式> THEN
    PL/SQL 和 SQL语句
END IF;
```

```
IF <布尔表达式> THEN
    PL/SQL 和 SQL语句
ELSE
    其它语句
END IF;
```

```
IF <布尔表达式> THEN
    PL/SQL 和 SQL语句
ELSIF < 其它布尔表达式> THEN
    其它语句
ELSIF < 其它布尔表达式> THEN
    其它语句
ELSE
    其它语句
END IF;
```

```
DECLARE
    m sc.grade%type;
BEGIN
    select grade into m
    from sc
    where sno ='04001' and cno = '005';
    if m<60 then
        dbms_output.put_line('不及格');
    elsif m>=60 and m<80 then
        dbms_output.put_line('及格');
    elsif m>=80 and m<90 then
        dbms_output.put_line('良好');
    else
        dbms_output.put_line('优秀');
    end if;
END;
```

循环语句

- 循环语句

```
LOOP
        要执行的语句;
        EXIT WHEN <条件语句>
END LOOP;

WHILE <布尔表达式> LOOP
    要执行的语句;
END LOOP;
```

- 循环语句

```
FOR 循环变量 IN 下限 .. 上限 LOOP
        要执行的语句;
END LOOP;
```

游标

- 在 PL/SQL 程序中，对于处理多行记录的事务经常使用游标来实现。
- 游标是系统为用户开设的一个数据缓冲区，内存中一段连续的存储单元，存放SQL语句的执行结果。每个游标区都有一个名字。
- PL/SQL通过游标提供了一个对结果集进行逐行处理的能力，游标可以看作一种特殊的指针，它与某个查询结果相联系，可以指向结果集的任意位置，以便对指定位置的数据进行处理。
- 使用游标可以在查询数据的同时对数据进行处理。

声明游标

```
Declare
Cursor c1 is
  Select s.sno,sname,cname,grade
  From s,sc,c
  Where  s.sno=sc.sno
         and c.cno=sc.cno
         and college='信息'
```

打开游标

```
begin
  open c1;
```

提取游标数据

移动游标指针

```
fetch c1 into a, b, c, d;
```

```
DECLARE
    a varchar2(10);
    b varchar2(20);
    c varchar2(40);
    d number;
    m number;
    cursor c is select s.sno,sname,cname,grade;
        from s,sc,c
        where s.sno=sc.sno and c.cno=sc.cno and college='信息';
BEGIN
    open c1;
    loop
        fetch c1 into a,b,c,d;
        exit when c1%notfound;
        if(d>60)then
            m:=m+d;
            dbms_output.put_line(a||b||c||d||m);
        end if;
    end loop;
    close c1;
END;
```

- **声明游标：**就是定义一个游标名，以及与其相对应的SELECT 语句。格式：

```
CURSOR cursor_name[(parameter[, parameter]…)]
        IS select_statement;
```

- 游标参数只能为输入参数，其格式为：

```
parameter_name [IN] datatype [{:= | DEFAULT} expression]
```

  - 在指定数据类型时，不能使用长度约束。如NUMBER(4)、CHAR(10) 等都是错误的。

- **打开游标：**就是执行游标所对应的SELECT 语句，将其查询结果放入工作区，并且游标指针指向工作区的首部，标识游标结果集合。
- 格式：

```
OPEN cursor_name[([parameter =>] value[,
[parameter =>] value]…)];
```

- **推进游标指针并取当前记录 ：**向后推动游标指针，然后将工作区中游标指针当前所指向的记录取出放入指定的输出变量中。
- 格式：

```
FETCH cursor_name INTO {variable_list |
                        record_variable };
```

- 说明
  - (1) 变量必须与SELECT语句中的目标列表达式具有一一对应关系；
  - (2) FETCH语句通常用在一个循环结构中，通过循环执行FETCH语句逐条取出结果集中的行进行处理；
  - (3) 为进一步方便用户处理数据，现在一些关系数据库管理系统对FETCH语句做了扩充，允许用户向任意方向以任意步长移动游标指针。

- **游标属性**
  - %FOUND：布尔型属性，当最近一次读记录时成功返回，则值为TRUE；
  - %NOTFOUND：布尔型属性，与%FOUND相反；
  - %ISOPEN：布尔型属性，当游标已打开时返回 TRUE；
  - %ROWCOUNT：数字型属性，返回已从游标中读取的记录数 。

存储过程

- ORACLE 提供可以把PL/SQL 程序存储在数据库中，并可以在任何地方来运行它。这样就叫存储过程或函数。
- 过程和函数统称为PL/SQL子程序，他们是被命名的PL/SQL块，均存储在数据库中，并通过输入参数、输出参数或输入/输出参数与其调用者交换信息。
- 过程和函数的唯一区别是函数通过函数名向调用者返回数据，而过程则不返回数据。

创建过程语法格式:

```
CREATE [OR REPLACE] PROCEDURE Procedure_name
    [ (argment [ { IN | IN OUT }] Type,
          argment [ { IN | OUT | IN OUT } ] Type ]
    { IS | AS }
        <类型.变量的说明>
BEGIN
        <执行部分>
EXCEPTION
        <可选的异常错误处理程序>
END;
```

- 调用存储过程：

```
begin
    Procedure_name( parameter1, parameter2…);
end;
```

```
DECLARE
    a varchar2(10);
    b varchar2(20);
    c varchar2(40);
    d number;
    m number;
    cursor c is select s.sno,sname,cname,grade;
        from s,sc,c
        where s.sno=sc.sno and c.cno=sc.cno and college='信息';
BEGIN
    open c1;
    loop
        fetch c1 into a,b,c,d;
        exit when c1%notfound;
        if(d>60)then
            m:=m+d;
            dbms_output.put_line(a||b||c||d||m);
        end if;
    end loop;
    close c1;
END;
```