第6章 PL/SQL基础

6.1 PL/SQL的基本构成

6.1.1 特点

PL/SQL语言是SQL语言的扩展，具有为程序开发而设计的特性，如数据封装、异常处理、面向对象等特性。PL/SQL是嵌入到Oracle服务器和开发工具中的，所以具有很高的执行效率和同Oracle数据库的完美结合。在PL/SQL模块中可以使用查询语句和数据操纵语句(即进行DML操作)，这样就可以编写具有数据库事务处理功能的模块。

至于数据定义(DDL)和数据控制(DCL)命令的处理，需要通过Oracle提供的特殊的DBMS_SQL包来进行。PL/SQL还可以用来编写过程、函数、包及数据库触发器。过程和函数也称为子程序，在定义时要给出相应的过程名和函数名。它们可以存储在数据库中成为存储过程和存储函数，并可以由程序来调用，它们在结构上同程序模块类似。

PL/SQL过程化结构的特点是：可将逻辑上相关的语句组织在一个程序块内；通过嵌入或调用子块，构造功能强大的程序；可将一个复杂的问题分解成为一组便于管理、定义和实现的小块。

6.1.2 块结构和基本语法要求

PL/SQL程序的基本单元是块(BLOCK)，块就是实现一定功能的逻辑模块。一个PL/SQL程序由一个或多个块组成。块有固定的结构，也可以嵌套。一个块可以包括三个部分，每个部分由一个关键字标识。

块中各部分的作用解释如下：

(1) DECLARE：声明部分标志。

(2) BEGIN：可执行部分标志。

(3) EXCEPTION：异常处理部分标志。

(4) END；：程序结束标志。

在以下的训练中，将使用函数DBMS_OUTPUT.PUT_LINE显示输出结果。DBMS_OUTPUT是Oracle提供的包，该包有如下三个用于输出的函数，用于显示PL/SQL程序模块的输出信息。

第一种形式：

DBMS_OUTPUT.PUT(字符串表达式);

用于输出字符串，但不换行，括号中的参数是要输出的字符串表达式。

第二种形式：

DBMS_OUTPUT.PUT_LINE(字符串表达式);

用于输出一行字符串信息，并换行，括号中的参数是要输出的字符串表达式。

第三种形式：

DBMS_OUTPUT.NEW_LINE;

用来输出一个换行，没有参数。

调用函数时，在包名后面用一个点“.”和函数名分隔，表示隶属关系。

要使用该方法显示输出数据，在SQL*Plus环境下要先执行一次如下的环境设置命令：

SET SERVEROUTPUT ON [SIZE n]

用来打开DBMS_OUTPUT.PUT_LINE函数的屏幕输出功能，系统默认状态是OFF。其中，n表示输出缓冲区的大小。n的范围在2000～1 000 000之间，默认为2000。如果输出内容较多，需要使用SIZE n来设置较大的输出缓冲区。

在PL/SQL模块中可以使用查询语句和数据操纵语句(即进行DML操作)，所以PL/SQL程序是同SQL语言紧密结合在一起的。在PL/SQL程序中，最常见的是使用SELECT语句从数据库中获取信息，同直接执行SELECT语句不同，在程序中的SELECT语句总是和INTO相配合，INTO后跟用于接收查询结果的变量，形式如下：

SELECT 列名1，列名2... INTO 变量1，变量2... FROM 表名 WHERE 条件；

注意：接收查询结果的变量类型、顺序和个数同SELECT语句的字段的类型、顺序和个数应该完全一致。并且SELECT语句返回的数据必须是一行，否则将引发系统错误。当程序要接收返回的多行结果时，可以采用后面介绍的游标的方法。

使用INSERT、DELETE和UPDATE的语法没有变化，但在程序中要注意判断语句执行的状态，并使用COMMIT或ROLLBACK进行事务处理。

以下训练包含了按照标准结构书写的一个包含SELECT语句的PL/SQL程序示例。

【训练1】 查询雇员编号为7788的雇员姓名和工资。

步骤1：用SCOTT账户登录SQL*Plus。

步骤2：在输入区输入以下程序：

```
/*这是一个简单的示例程序*/
SET SERVEROUTPUT ON
DECLARE--定义部分标识
 v_name  VARCHAR2(10);          --定义字符串变量v_name
 v_sal   NUMBER(5);       --定义数值变量v_sal
BEGIN                           --可执行部分标识
```

```
SELECT          ename,sal
  INTO v_name,v_sal
FROM   emp
 WHERE empno=7788;
--在程序中插入的SQL语句
DBMS_OUTPUT.PUT_LINE('7788号雇员是：'||v_name||'，工资为：'||to_char(v_sal));
--输出雇员名和工资
END;
--结束标识
```

步骤3：按执行按钮或F5快捷键执行程序。

输出的结果是：

7788号雇员是：SCOTT，工资为：3000

PL/SQL 过程已成功完成。

以上程序的作用是，查询雇员编号为7788的雇员姓名和工资，然后显示输出。这种方法同直接在SQL环境下执行SELECT语句显示雇员的姓名和工资比较，程序变得更复杂。那么两者究竟有什么区别呢？SQL查询的方法，只限于SQL环境，并且输出的格式基本上是固定的。而程序通过把数据取到变量中，可以进行复杂的处理，完成SQL语句不能实现的功能，并通过多种方式输出。

“--”是注释符号，后边是程序的注释部分。该部分不编译执行，所以在输入程序时可以省略。/*.....*/中间也是注释部分，同“--”注释方法不同，它可以跨越多行进行注释。

PL/SQL程序的可执行语句、SQL语句和END结束标识都要以分号结束。

6.1.3 数据类型

变量的基本数据类型同SQL部分的字段数据类型相一致，但是也有不同，如表6-1所示。

表6-1 变量的数据类型

		数据类型	子类型
纯量类型	数值	BINARY_INTEGER	NATURAL,POSITIVE
		NUMBER	DEC,DECIMAL,DOUBLE PRECISION,PLOAT,INTEGER,INT,NUMERIC, REAL,SMALLINT
	字符	CHAR	CHARACTER,STRING
		VARCHAR2	VARCHAR
		LONG	
		LONG RAW	
		RAW	
		RAWID	
	逻辑	BOOLEAN	
	日期	DATE	
组合类型	记录	RECORD	
	表	TABLE	

表6-2 数据类型说明

类型标识符	说　　明
NUMBER	数值型
INT	整数型
BINARY_INTEGER	整数型，带符号
CHAR	定长字符型，最大32 767个字符
VARCHAR2	变长字符型，最大32 767个字符
LONG	变长字符型，最长2 GB
DATE	日期型，用于存储日期和时间
BOOLEAN	布尔型，用于存储逻辑值TRUE和FALSE
LOB	大对象类型，用来存储非结构化数据，长度可达4 GB

NUMBER和VARCHAR2是最常用的数据类型。

VARCHAR2是可变长度的字符串，定义时指明最大长度，存储数据的长度是在最大长度的范围自动调节的，数据前后的空格，Oracle 9i会自动将其删去。

NUMBER型可以定义数值的总长度和小数位，如NUMBER(10,3)表示定义一个宽度为10、小数位为3的数值。整个宽度减去小数部分的宽度为整数部分的宽度，所以整数部分的宽度为7。

CHAR数据类型为固定长度的字符串，定义时要指明宽度，如不指明，默认宽度为1。定长字符串在显示输出时，有对齐的效果。

DATE类型用于存储日期数据，内部使用7个字节。其中包括年、月、日、小时、分钟和秒数。默认的格式为DD-MON-YY，如：07-8月-03表示2003年8月7日。

BOOLEAN为布尔型，用于存储逻辑值，可用于PL/SQL的控制结构。

LOB数据类型可以存储视频、音频或图片，支持随机访问，存储的数据可以位于数据库内或数据库外，具体有四种类型：BFILE、BLOB、CLOB、NCLOB。但是操纵大对象需要使用Oracle提供的DBMS_LOB包。

6.1.4 变量定义

1. 变量定义

变量的作用是用来存储数据，可以在过程语句中使用。变量在声明部分可以进行初始化，即赋予初值。变量在定义的同时也可以将其说明成常量并赋予固定的值。变量的命名规则是：以字母开头，后跟其他的字符序列，字符序列中可以包含字母、数值、下划线等符号，最大长度为30个字符，不区分大小写。不能使用Oracle的保留字作为变量名。变量名不要和在程序中引用的字段名相重，如果相重，变量名会被当作列名来使用。

变量的作用范围是在定义此变量的程序范围内，如果程序中包含子块，则变量在子块中也有效。但在子块中定义的变量，仅在定义变量的子块中有效，在主程序中无效。

变量定义的方法是：

变量名 [CONSTANT] 类型标识符 [NOT NULL][:= 值|DEFAULT 值];

关键字CONSTANT用来说明定义的变量是常量，如果是常量，必须有赋值部分进行赋值。

关键值NOT NULL用来说明变量不能为空。

：=或DEFAULT用来为变量赋初值。

变量可以在程序中使用赋值语句重新赋值。通过输出语句可以查看变量的值。

在程序中为变量赋值的方法是：

变量名:=值 或 PL/SQL 表达式;

以下是有关变量定义和赋值的练习。

【训练1】 变量的定义和初始化。

输入和运行以下程序：

```
SET SERVEROUTPUT ON
DECLARE                          --声明部分标识
v_job              VARCHAR2(9);
v_count  BINARY_INTEGER DEFAULT 0;
v_total_sal        NUMBER(9,2) := 0;
v_date             DATE := SYSDATE + 7;
c_tax_rate         CONSTANT NUMBER(3,2) := 8.25;
v_valid            BOOLEAN NOT NULL := TRUE;
```

```
BEGIN
v_job:='MANAGER';
--在程序中赋值
 DBMS_OUTPUT.PUT_LINE(v_job);
--输出变量v_job的值
 DBMS_OUTPUT.PUT_LINE(v_count);
--输出变量v_count的值
DBMS_OUTPUT.PUT_LINE(v_date);
--输出变量v_date的值
DBMS_OUTPUT.PUT_LINE(c_tax_rate);
--输出变量c_tax_rate的值
END;
```

执行结果：

MANAGER

0

18-4月 -03

8.25

PL/SQL 过程已成功完成。

说明：本训练共定义了6个变量，分别用“:=”赋值运算符或DEFAULT 关键字对变量进行了初始化或赋值。其中：c_tax_rate为常量，在数据类型前加了“CONSTANT” 关键字；v_valid变量在赋值运算符前面加了关键字“NOT NULL”，强制不能为空。如果变量是布尔型，它的值只能是“TRUE”、“FALSE”或“NULL”。本练习中的变量v_valid布尔变量的值只能取“TRUE”或“FALSE”。

2．根据表的字段定义变量

变量的声明还可以根据数据库表的字段进行定义或根据已经定义的变量进行定义。方法是在表的字段名或已经定义的变量名后加%TYPE，将其当作数据类型。定义字段变量的方法如下：

变量名 表名.字段名%TYPE;

【训练2】 根据表的字段定义变量。

输入并执行以下程序：

```
SET SERVEROUTPUT ON
DECLARE
v_ename          emp.ename%TYPE;--根据字段定义变量
```

```
BEGIN
SELECT          ename
 INTO           v_ename
 FROM           emp
 WHERE          empno = 7788;
DBMS_OUTPUT.PUT_LINE(v_ename);
--输出变量的值
END;
```

执行结果：

```
SCOTT
PL/SQL 过程已成功完成。
```

说明：变量v_ename是根据表emp的ename字段定义的，两者的数据类型总是一致的。

如果我们根据数据库的字段定义了某一变量，后来数据库的字段数据类型又进行了修改，那么程序中的该变量的定义也自动使用新的数据类型。使用该种变量定义方法，变量的数据类型和大小是在编译执行时决定的，这为书写和维护程序提供了很大的便利。

3．结合变量的定义和使用

我们还可以定义SQL*Plus环境下使用的变量，称为结合变量。结合变量也可以在程序中使用，该变量是在整个SQL*Plus环境下有效的变量，在退出SQL*Plus之前始终有效，所以可以使用该变量在不同的程序之间传递信息。结合变量不是由程序定义的，而是使用系统命令VARIABLE定义的。在SQL*Plus环境下显示该变量要用系统的PRINT命令。

在SQL*Plus环境下定义结合变量的方法如下：

VARIABLE 变量名 数据类型

【训练3】 定义并使用结合变量。

步骤1：输入和执行下列命令，定义结合变量g_ename：

```
VARIABLE  g_ename VARCHAR2(100)
```

步骤2：输入和执行下列程序：

```
SET SERVEROUTPUT ON
BEGIN
 :g_ename:=:g_ename|| 'Hello~ ';
--在程序中使用结合变量
 DBMS_OUTPUT.PUT_LINE(:g_ename);
--输出结合变量的值
END;
```

输出结果：

Hello~

PL/SQL 过程已成功完成。

步骤3：重新执行程序。

输出结果：

Hello~ Hello~

PL/SQL 过程已成功完成。

步骤4：程序结束后用命令显示结合变量的内容：

PRINT g_ename

输出结果：

G_ENAME

Hello~ Hello~

说明：g_ename为结合变量，可以在程序中引用或赋值，引用时在结合变量前面要加上“：”。在程序结束后该变量的值仍然存在，其他程序可以继续引用。

4．记录变量的定义

还可以根据表或视图的一个记录中的所有字段定义变量，称为记录变量。记录变量包含若干个字段，在结构上同表的一个记录相同，定义方法是在表名后跟%ROWTYPE。记录变量的字段名就是表的字段名，数据类型也一致。

记录变量的定义方法是：

记录变量名 表名%ROWTYPE;

获得记录变量的字段的方法是：记录变量名.字段名，如emp_record.ename。

如下练习中定义并使用了记录变量。

【训练4】 根据表定义记录变量。

输入并执行如下程序：

```
SET SERVEROUTPUT ON
DECLARE
emp_record                emp%ROWTYPE;--定义记录变量
BEGIN
 SELECT * INTO            emp_record
 FROM                     emp
 WHERE                    mpno = 7788;--取出一条记录
DBMS_OUTPUT.PUT_LINE(emp_record.ename);   --输出记录变量的某个字段
END;
```

执行结果为：

SCOTT

PL/SQL 过程已成功完成。

说明：在以上的练习中定义了记录变量emp_record，它是根据表emp的全部字段定义的。SELECT语句将编号为7788的雇员的全部字段对应地存入该记录变量，最后输出记录变量的雇员名称字段emp_record.ename的内容。如果要获得其他字段的内容，比如要获得编号为7788的雇员的工资，可以通过变量emp_record.sal获得，依此类推。

5．TABLE类型变量

在PL/SQL中可以定义TABLE类型的变量。 TABLE数据类型用来存储可变长度的一维数组数据，即数组中的数据动态地增长。要定义TABLE变量，需要先定义TABLE数据类型。通过使用下标来引用TABLE变量的元素。

TABLE数据类型的定义形式如下：

TYPE 类型名 IS TABLE OF 数据类型[NOT NULL] INDEX BY BINARY_INTEGER;

此数据类型自动带有BINARY_INTEGER型的索引。

【训练5】 定义和使用TABLE变量：

```
SET SERVEROUTPUT ON
DECLARE
TYPE type_table IS TABLE OF VARCHAR2(10) INDEX BY BINARY_INTEGER;  --类型说明
 v_t        type_table;    --定义TABLE变量
BEGIN
 v_t(1):='MONDAY';
 v_t(2):='TUESDAY';
v_t(3):='WEDNESDAY';
 v_t(4):='THURSDAY';
v_t(5):='FRIDAY';
```

```
DBMS_OUTPUT.PUT_LINE(v_t(3)); --输出变量的内容
END;
```

执行结果为：

```
WEDNESDAY
PL/SQL 过程已成功完成。
```

说明：本例定义了长度为10的字符型TABLE变量，通过赋值语句为前五个元素赋值，最后输出第三个元素。

6.1.5 运算符和函数

PL/SQL常见的运算符和函数包括以下方面(这里只做简单的总结，可参见SQL部分的例子)：

* 算术运算：加(+)、减(-)、乘(*)、除(/)、指数(**)。

* 关系运算：小于(<)、小于等于(<=)、大于(>)、大于等于(>=)、等于(=)、不等于(!=或<>)。

* 字符运算：连接(||)。

* 逻辑运算：与(AND)、或(OR)、非(NOT)。

还有如表6-3所示的特殊运算。

表6-3 特殊运算符

运算符	操　　作
IS NULL	用来判断运算对象是否为空，为空则返回 TRUE
LIKE	用来判断字符串是否与模式匹配
BETWEEN…AND…	判断值是否位于一个区间
IN(…)	测试运算对象是否在一组值的列表中

IS NULL或IS NOT NULL用来判断运算对象的值是否为空，不能用“=”去判断。另外，对空值的运算也必须注意，对空值的算术和比较运算的结果都是空，但对空值可以进行连接运算，结果是另外一部分的字符串。例如：

NULL+5的结果为NULL。

NULL>5的结果为NULL。

NULL|| 'ABC' 的结果为'ABC'。

在PL/SQL中可以使用绝大部分Oracle函数，但是组函数(如AVG( )、MIN( )、MAX( )等)只能出现在SQL语句中，不能在其他语句中使用。还有GREATEST( )、LEAST( )也不能使用。类型转换在很多情况下是自动的，在不能进行自动类型转换的场合需要使用转换函数。

6.2 结构控制语句

6.2.1 分支结构

分支结构是最基本的程序结构，分支结构由IF语句实现。

使用IF语句，根据条件可以改变程序的逻辑流程。IF语句有如下的形式:

IF 条件1 THEN

语句序列1;

[ELSIF 条件2 THEN

语句序列2;

ELSE

语句序列n；]

END IF;

其中：

条件部分是一个逻辑表达式，值只能是真(TRUE)、假(FALSE)或空(NULL)。

语句序列为多条可执行的语句。

根据具体情况，分支结构可以有以下几种形式：

IF-THEN-END IF

IF-THEN-ELSE-END IF

IF-THEN-ELSIF-ELSE-END IF

1．IF-THEN-END IF形式

这是最简单的IF结构，练习如下：

【训练1】 如果温度大于30℃，则显示“温度偏高”。

输入并执行以下程序：

```
SET SERVEROUTPUT ON
DECLARE
 V_temprature              NUMBER(5):=32;
 V_result          BOOLEAN:=false;
```

```
BEGIN
  V_result:= v_temprature >30;
 IF V_result THEN
DBMS_OUTPUT.PUT_LINE('温度'|| V_temprature ||'度,偏高');
 END IF;
END;
```

执行结果为：

温度32度，偏高

PL/SQL过程已成功完成。

说明：该程序中使用了布尔变量，初值为false，表示温度低于30℃。表达式v_temprature >30返回值为布尔型，赋给逻辑变量V_result。如果变量v_temprature的值大于30，则返回值为真，否则为假。V_result值为真就会执行IF到 END IF之间的输出语句，否则没有输出结果。

试修改温度的初值为25℃，重新执行，观察结果。

2．IF-THEN-ELSE-END IF形式

这种形式的练习如下：

【训练2】 根据性别，显示尊称。

输入并执行以下程序：

```
SET SERVEROUTPUT ON
DECLARE
 v_sex    VARCHAR2(2);
v_titil    VARCHAR2(10);
```

```
BEGIN
  v_sex:='男';
  IF v_sex ='男' THEN
    v_titil:='先生';
  ELSE
    v_titil:='女士';
  END IF;
  DBMS_OUTPUT.PUT_LINE(v_titil||'您好！ ');
END;
```

执行结果为：

先生您好！

PL/SQL 过程已成功完成。

说明：该程序根据性别显示尊称和问候，无论性别的值为何，总会有显示结果输出。如果V_sex的值不是‘男’和‘女’，那么输出结果会是什么？

【练习1】对以上程序进行补充修改，在ELSE部分嵌入一个IF结构，如果V_sex的值不是'女'，则显示“朋友你好”。

3．IF-THEN-ELSIF-ELSE-END IF形式

这种形式的练习如下：

【训练3】 根据雇员工资分级显示税金。

输入并运行以下程序：

```
SET SERVEROUTPUT ON
DECLARE
  v_sal  NUMBER(5);
  v_tax  NUMBER(5,2);
BEGIN
  SELECT sal INTO v_sal
  FROM emp
  WHERE empno=7788;
```

```
IF v_sal >=3000 THEN
        V_tax:= v_sal*0.08;--税率8%
 ELSIF v_sal>=1500 THEN
        V_tax:= v_sal*0.06; --税率6%
ELSE
        V_tax:= v_sal*0.04; --税率4%
 END IF;
DBMS_OUTPUT.PUT_LINE('应缴税金:'||V_tax);
END;
```

执行结果为：

```
应缴税金:240
PL/SQL 过程已成功完成。
```

说明：该程序根据工资计算7788号雇员应缴税金，不同工资级别的税率不同。

6.2.2 选择结构

CASE语句适用于分情况的多分支处理，可有以下三种用法。

1．基本CASE结构

语句的语法如下：

```
CASE 选择变量名
WHEN 表达式1 THEN
 语句序列1
WHEN 表达式2 THEN
语句序列2
```

WHEN 表达式n THEN

语句序列n

ELSE

语句序列n+1

END CASE;

在整个结构中，选择变量的值同表达式的值进行顺序匹配，如果相等，则执行相应的语句序列，如果不等，则执行ELSE部分的语句序列。

以下是一个使用CASE选择结构的练习。

【训练1】 使用CASE结构实现职务转换。

输入并执行程序：

```
SET SERVEROUTPUT ON
DECLARE
v_job  VARCHAR2(10);
BEGIN
SELECT job INTO v_job
FROM emp
WHERE empno=7788;
CASE v_job
```

```
WHEN 'PRESIDENT' THEN
 DBMS_OUTPUT.PUT_LINE('雇员职务:总裁');
WHEN 'MANAGER' THEN
 DBMS_OUTPUT.PUT_LINE('雇员职务:经理');
WHEN 'SALESMAN' THEN
 DBMS_OUTPUT.PUT_LINE('雇员职务:推销员');
WHEN 'ANALYST' THEN
 DBMS_OUTPUT.PUT_LINE('雇员职务:系统分析员');
WHEN 'CLERK' THEN
 DBMS_OUTPUT.PUT_LINE('雇员职务:职员');
```

```
ELSE
 DBMS_OUTPUT.PUT_LINE('雇员职务:未知');
END CASE;
END;
```

执行结果：

```
雇员职务:系统分析员
PL/SQL 过程已成功完成。
```

说明：以上实例检索雇员7788的职务，通过CASE结构转换成中文输出。

【练习1】将雇员号修改成其他已知雇员号，重新执行。

2．表达式结构CASE语句

在Oracle中，CASE结构还能以赋值表达式的形式出现，它根据选择变量的值求得不同的结果。

它的基本结构如下：

变量=CASE 选择变量名

WHEN 表达式1 THEN 值1

WHEN 表达式2 THEN 值2

WHEN 表达式n THEN 值n

ELSE值n+1

END;

【训练2】 使用CASE的表达式结构。

```
SET SERVEROUTPUT ON
DECLARE
        v_grade         VARCHAR2(10);
        v_result VARCHAR2(10);
BEGIN
        v_grade:='B';
        v_result:=CASE v_grade
         WHEN 'A' THEN '优'
```

```
WHEN 'B' THEN '良'
        WHEN 'C' THEN '中'
         WHEN 'D' THEN '差'
        ELSE '未知'
END;
DBMS_OUTPUT.PUT_LINE('评价等级:'||V_result);
END;
```

执行结果为：

```
评价等级:良

PL/SQL 过程已成功完成。
```

说明：该CASE表达式通过判断变量v_grade的值，对变量V_result赋予不同的值。

3．搜索CASE结构

Oracle还提供了一种搜索CASE结构，它没有选择变量，直接判断条件表达式的值，根据条件表达式决定转向。

CASE

WHEN 条件表达式1 THEN

语句序列1

WHEN 条件表达式2 THEN

语句序列2

```
WHEN 条件表达式n THEN
  语句序列n
ELSE
  语句序列n+1
END CASE;
```

【训练3】 使用CASE的搜索结构。

```
SET SERVEROUTPUT ON
DECLARE
  v_sal   NUMBER(5);
BEGIN
  SELECT sal INTO v_sal FROM emp
          WHERE empno=7788;
CASE
          WHEN v_sal>=3000 THEN
DBMS_OUTPUT.PUT_LINE('工资等级：高');
          WHEN v_sal>=1500 THEN
```

```
 DBMS_OUTPUT.PUT_LINE('工资等级：中');
    ELSE
  DBMS_OUTPUT.PUT_LINE('工资等级：低');
END CASE;
END;
```

执行结果为：

```
工资等级：高
PL/SQL 过程已成功完成。
```

说明：此结构类似于IF-THEN-ELSIF-ELSE-END IF结构。本训练判断7788雇员的工资等级。

6.2.3 循环结构

循环结构是最重要的程序控制结构，用来控制反复执行一段程序。比如我们要进行累加，则可以通过适当的循环程序实现。PL/SQL循环结构可划分为以下3种：

* 基本LOOP循环。
* FOR LOOP循环。
* WHILE LOOP循环。

1．基本LOOP循环

基本循环的结构如下：

```
LOOP                --循环起始标识
 语句1;
 语句2;
 EXIT [WHEN 条件];
END LOOP;        --循环结束标识
```

该循环的作用是反复执行LOOP与END LOOP之间的语句。

EXIT用于在循环过程中退出循环，WHEN用于定义EXIT的退出条件。如果没有WHEN条件，遇到EXIT语句则无条件退出循环。

【训练1】 求：$1^2+3^2+5^2+...+15^2$ 的值。

输入并执行以下程序：

```
SET SERVEROUTPUT ON
DECLARE
 v_total          NUMBER(5):=0;
v_count           NUMBER(5):=1;
BEGIN
LOOP
   v_total:=v_total+v_count**2;
EXIT WHEN v_count=15;--条件退出
```

```
v_count:=v_count+2;
END LOOP;
 DBMS_OUTPUT.PUT_LINE(v_total);
END;
```

输出结果为：

```
680
PL/SQL 过程已成功完成。
```

说明：基本循环一定要使用EXIT退出，否则就会成为死循环。

【练习1】求1*2*3*4*...*10的值。

2．FOR LOOP循环

FOR循环是固定次数循环，格式如下：

```
FOR 控制变量 in [REVERSE] 下限..上限
LOOP
 语句1;
语句2;
END LOOP;
```

循环控制变量是隐含定义的，不需要声明。

下限和上限用于指明循环次数。正常情况下循环控制变量的取值由下限到上限递增，REVERSE关键字表示循环控制变量的取值由上限到下限递减。

以下是FOR循环结构的练习。

【训练2】 用FOR循环输出图形。

```
SET SERVEROUTPUT ON
BEGIN
FOR I IN 1..8
LOOP
DBMS_OUTPUT.PUT_LINE(to_char(i)||rpad('*',I,'*'));
END LOOP;
END;
```

输出结果为：

```
1*
2**
3***
4****
5*****
6******
7*******
8********
```

PL/SQL 过程已成功完成。

说明：该程序在循环中使用了循环控制变量I，该变量隐含定义。在每次循环中根据循环控制变量I的值，使用RPAD函数控制显示相应个数的“*”。

【练习2】为以上程序增加REVERSE关键字，观察执行结果。

【训练3】 输出一个空心三角形。

```
BEGIN
FOR I IN 1..9
LOOP
 IF I=1 OR I=9 THEN
 DBMS_OUTPUT.PUT_LINE(to_char(I)||rpad('                ',12-I,'
')||rpad('*',2*i-1,'*'));
```

```
ELSE
  DBMS_OUTPUT.PUT_LINE(to_char(I)||rpad('    ',12-I,'    ')||'*'||rpad('
',I*2-3,' ')||'*');
        END IF;
        END LOOP;
        END;
```

输出结果为：

```
1        *
2       * *
3      *   *
4     *     *
5    *       *
6   *         *
7  *           *
8 *             *
9 *****************
PL/SQL 过程已成功完成。
```

说明：该实例采用循环和IF结构相结合，对第1行和第9行(I=1 OR I=9)执行同样的输出语句，其他行执行另外的输出语句。

【练习3】修改程序，输出一个实心三角形。

3．WHILE LOOP循环

WHILE循环是有条件循环，其格式如下：

```
WHILE 条件
   LOOP
    语句1;
    语句2;
   END LOOP;
```

当条件满足时，执行循环体；当条件不满足时，则结束循环。如果第一次判断条件为假，则不执行循环体。

以下是WHILE循环结构的练习。

【训练3】 使用WHILE 循环向emp表连续插入5个记录。

步骤1：执行下面的程序：

```
SET SERVEROUTPUT ON
DECLARE
v_count        NUMBER(2) := 1;
BEGIN
  WHILE v_count <6 LOOP
    INSERT INTO emp(empno, ename)
    VALUES (5000+v_count, '临时');
```

```
v_count := v_count + 1;
  END LOOP;
  COMMIT;
END;
```

输出结果为：

```
PL/SQL 过程已成功完成。
```

步骤2：显示插入的记录：

```
SELECT empno,ename FROM emp WHERE ename='临时';
```

输出结果为：

```
 EMPNO ENAME
---------- ----------
      5001 临时
      5002 临时
      5003 临时
      5004 临时
      5005 临时
已选择5行。
```

步骤3：删除插入的记录：

```
DELETE FROM emp WHERE ename='临时';
COMMIT;
```

输出结果为：

```
已删除5行。
提交完成。
```

说明：该练习使用WHILE循环向emp表插入5个新记录(雇员编号根据循环变量生成)，并通过查询语句显示新插入的记录，然后删除。

4．多重循环

循环可以嵌套，以下是一个二重循环的练习。

【训练4】 使用二重循环求1！+2！+...+10！的值。

步骤1：第1种算法：

```
SET SERVEROUTPUT ON
DECLARE
  v_total       NUMBER(8):=0;
  v_ni          NUMBER(8):=0;
  J    NUMBER(5);
BEGIN
```

```
FOR I IN 1..10
  LOOP
   J:=1;
    v_ni:=1;
   WHILE J<=I
   LOOP
    v_ni:= v_ni*J;
    J:=J+1;
   END LOOP;--内循环求n!
```

```
v_total:=v_total+v_ni;
  END LOOP;--外循环求总和
  DBMS_OUTPUT.PUT_LINE(v_total);
END;
```

输出结果为：

```
4037913
PL/SQL 过程已成功完成。
```

步骤2：第2种算法：

```
SET SERVEROUTPUT ON
DECLARE
  v_total                NUMBER(8):=0;
  v_ni                   NUMBER(8):=1;
BEGIN
  FOR I IN 1..10
  LOOP
    v_ni:= v_ni*I;        --求n!
    v_total:= v_total+v_ni;
  END LOOP;                          --循环求总和
  DBMS_OUTPUT.PUT_LINE(v_total);
END;
```

输出结果为：

409114

PL/SQL 过程已成功完成。

说明：第1种算法的程序内循环使用WHILE循环求阶层，外循环使用FOR循环求总和。第2种算法是简化的算法，根据是：n!=n*(n?1)!。

6.3 阶段训练

【训练1】 插入雇员，如果雇员已经存在，则输出提示信息。

```
SET SERVEROUTPUT ON
DECLARE
v_empno NUMBER(5):=7788;
v_num VARCHAR2(10);
i NUMBER(3):=0;
BEGIN
 SELECT count(*) INTO v_num FROM SCOTT.emp WHERE empno=v_empno;
```

```
IF v_num=1 THEN
  DBMS_OUTPUT.PUT_LINE('雇员'||v_empno||'已经存在！');
  ELSE
  INSERT INTO emp(empno,ename)
   VALUES(v_empno,'TOM');
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('成功插入新雇员！');
END IF;
END;
```

说明：在本程序中，使用了一个技巧来判断一个雇员是否存在。如果一个雇员不存在，那么使用SELECT...INTO来获取雇员信息就会失败，因为SELECT...INTO形式要求查询必须返回一行。但如果使用COUNT统计查询，返回满足条件的雇员人数，则该查询总是返回一行，所以任何情况都不会失败。COUNT返回的统计人数为0说明雇员不存在，返回的统计人数为1说明雇员存在，返回的统计人数大于1说明有多个满足条件的雇员存在。本例在雇员不存在时进行插入操作，如果雇员已经存在则不进行插入。

【训练2】 输出由符号“*”构成的正弦曲线的一个周期(0～360°)。

```
SET SERVEROUTPUT ON SIZE 10000
SET LINESIZE 100
SET PAGESIZE 100
DECLARE
  v_a      NUMBER(8,3);
  v_p      NUMBER(8,3);
BEGIN
  FOR I IN 1..18
```

```
LOOP
v_a:=I*20*3.14159/180;--生成角度，并转换为弧度
 v_p:=SIN(v_a)*20+25;--求SIN函数值，20为放大倍数，25为水平位移
DBMS_OUTPUT.PUT_LINE(to_char(i)||lpad('*',v_p,' '));--输出记录变量的某个字段
END LOOP;
END;
```

输出结果如下：

```
1                  *
2                    *
3                      *
4                        *
5                        *
6                      *
7                    *
8                  *
9                *
10             *
```

```
11          *
12      *
13  *
14  *
15      *
16          *
17              *
18                  *
```

PL/SQL 过程已成功完成。

说明：在本程序中使用到了固定次数的循环以及SIN和LPAD函数，通过正确地设置步长、幅度和位移的参数，在屏幕上可正确地显示图形。

6.4 练习

1. 用来存放可变长度字符串的函数是：

A. CHAR B. VARCHAR2

C. NUMBER D. BOOLEAN

2. 在程序中必须书写的语句是：

A. SET SERVEROUTPUT ON

B. DECLARE

C. BEGIN

D. EXCEPTION

3. 在程序中正确的变量定义语句是：

A. emp_record emp.ename%ROWTYPE

B. emp_record emp%ROWTYPE

C. v_ename emp%TYPE

D. v_ename ename%TYPE

4. 在程序中最有可能发生错误的语句是：

A. INSERT INTO emp(empno,ename) VALUES(8888,'Jone')

B. UPDATE emp SET sal=sal+100

C. DELETE FROM emp

D. SELECT * FROM emp

5. 关于以下分支结构，如果i的初值是15，环循结束后j的值是：

```
IF i>20 THEN
  j:= i*2;
ELSIF i>15 THEN
  j:= i*3;
ELSE
  j:= i*4;
END IF;
```

A. 15　　B. 30　　C. 45　　D. 60

6. 关于以下循环，如果I的初值是3，则循环的次数是：

```
WHILE I<6 LOOP
 I:= I + 1;
END LOOP;
```

A. 3　　B. 4

C. 5　　D. 6

7. 以下表达式的结果非空的是：

A. NULL||NULL　　B. 'NULL'||NULL

C. 3+NULL　　D. (5>NULL)