

Pro * C 的使用

— Pro*C 程序概述:

1. 什么是 Pro*C 程序

在 ORACLE 数据库管理和系统中，有三种访问数据库的方法：

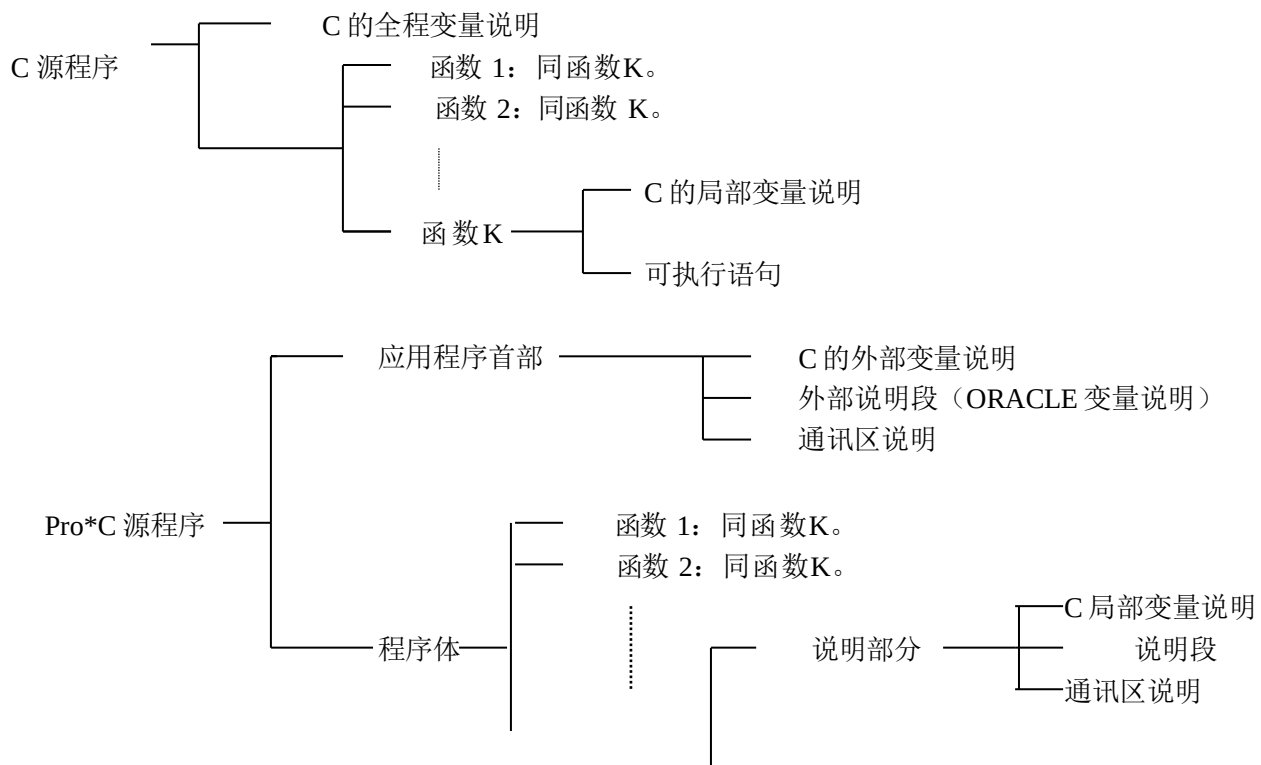
- (1) 用 SQL*Plus，它有 SQL 命令以交互的应用程序访问数据库；
- (2) 用第四代语言应用开发工具开发的应用程序访问数据库，这些工具有 SQL*Forms, QL*Reportwriter, SQL*Menu 等；
- (3) 利用在第三代语言内嵌入的 SQL 语言或 ORACLE 库函数调用来访问。

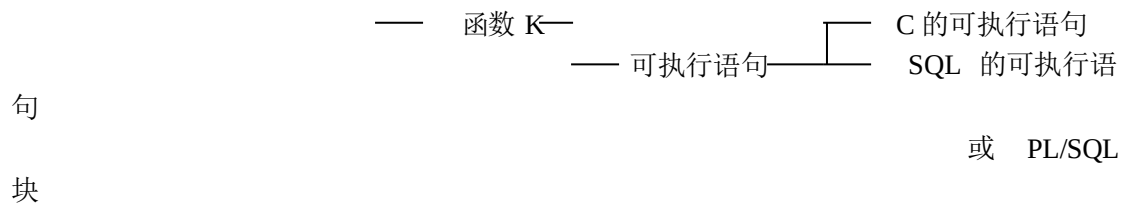
Pro*C 就属于第三种开发工具之一，它把过程化语言 C 和非过程化语言 SQL 最完善地结合起来，具有完备的过程处理能力，又能完成任何数据库的处理品任务，使用户可以通过编程完成各种类型的报表。在 Pro*C 程序中可以嵌入 SQL 语言，利用这些 SQL 语言可以完成动态地建立、修改和删除数据库中的表，也可以查询、插入、修改和删除数据库表中的行，还可以实现事务的提交和回滚。

在 Pro*C 程序中还可以嵌入 PL/SQL 块，以改进应用程序的性能，特别是在网络环境下，可以减少网络传输和处理的总开销。

2. Pro*C 的程序结构图

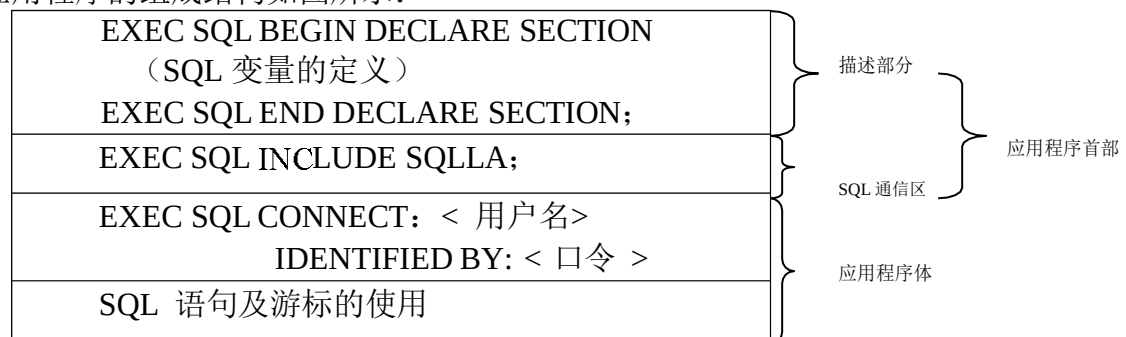
通俗来说，Pro*C 程序实际是内嵌有 SQL 语句或 PL/SQL 块的 C 程序，因此它的组成很类似 C 程序。但因为它内嵌有 SQL 语句或 PL/SQL 块，所以它还含有与之不同的成份。为了让大家对 Pro*C 有个感性的认识，特将二者差别比较如下：





二. Pro*C 程序的组成结构

每一个Pro*C 程序都包括两部分：（1）应用程序首部；（2）应用程序体
应用程序首部定义了 ORACLE 数据库的有关变量， 为在 C 语言中操纵
ORACLE 数据库做好了准备。应用程序体基本上由Pro*C的SQL 语句调用组成。
主要指查询SELECT、INSERT、UPDATE、DELETE 等语句。
应用程序的组成结构如图所示：



1. 应用程序首部

应用程序的首部就是Pro*C 的开始部分。它包括以下三部分：

- C 变量描述部分；
- SQL 变量描述部分（DECLARE 部分）；
- SQL 通信区。

(1).DECLARE 部分（描述部分）

描述部分说明程序的SQL 变量， 定义部分以 EXEC SQL BEGIN DECLARE SECTION ;开始和以 EXEC SQL END DECLARE SECTION ；结束的。它可以出现在程序的主部，也可出现在局部

- SQL 变量的说明和使用

在说明段能为SQL 变量指定的数据类型如表所示：

数据类型	描述
CHAR	单字符
CHAR(n)	n 个字符数组
INT	整数
SHORT	短整数
LONG	单精度浮点数
FLOAT	双精度浮点数
DOUBLE	变长字符串
VARCHAR	

这些数据类型的扩充。这在以后会谈到。

SQL 变量的使用应注意以下几点：

- 必须在描述部分明确定义
- 必须使用与其定义相同的大小写格式
- 在SQL 语句中使用，必须在其之前加一个“：”(冒号)，但在 C 语句中引用时不需加冒号。
- 不能是SQL 命令中的保留字。
- 可以带指示变量。

例如：EXEC SQL BEGIN DECLARE SECTIONS;

 VARCHAR programe[30];

 Int porgsal, pempno;

EXEC SQL END DECLARE SECTION;

...

EXEC SQL SELECT ENAME , SAL

 INTO: programe, : porgsal

 FROM EMP

 WHERE EMPNO = : pempno;

(2). 指示器变量的说明和引用

指示变量实际上也是一类 SQL 变量,它被用来管理与其相关联的宿主变量(即在 SQL 语句中充当输入或输出的变量)。每一个宿主变量都可定义一个指示器变量，主要用于处理空值(NULL)

指示器变量的说明基本同一般 SQL 变量一样，但必须定义成 2 字节的整型，如SHORT、INT。在 SQL 语句中引用时，其前也应加“：”(冒号)，而且必须附在其相关联的宿主变量之后，在 C 语句中，可独立使用。当指示器变量为-1 时，表示空值。例如：

EXEC SQL BEGIN DECLARE SECTION ;

INT dept- number;

SHORT ind - num;

CHAR emp -name;

EXEC SQL END DECLARE SECTION ;

...

Scanf(“90d %s”, & dept- number , dept - name);

 If (dept - number ==0)

 Ind - num = -1;

 Else

 Ind - num = 0;

EXEC SQL INSERT INTO DEPT (DEPTNO, DNAME)

VALUES(:dept - number : ind- num , :dept - name);

其中ind - num 是 dept - number 的指示器变量。当输入的dept - number 值是 0 时，则向 DEPT 表的 DEPTNO 列插入空值。

(3) .指针 SQL 变量的说明和使用

指针 SQL 变量在引用前也必须在 DECLARE 部分先说明。其说明格式同 C 语言。在 SQL 语句中引用时，指针名字前要加前缀“:”（冒号）而不加“*”（星号）。在 C 语句中用法如同 C 语言的指针变量。

(4) .数组 SQL 变更的说明和引用

在 SQL 语句中引用数组时，只需写数组名（名字前加冒号），不需写下标，在 C 语句中用法如同 C 语言的数组变量。

使用数组可大大降低网络传输开销。如要向一表插入 100 行数据，如果没有数组，就要重复 100 次，而引用后，只须执行一次 insert 语句、便可一次性插入。例如：

```
EXEC SQL BEGIN DECLARE SECTION;
    Int  emp_number[100];
    Char  emp_name[100][15];
    Float salary[100],commission[100];
    Int  dept_number;
EXEC SQL END DECLARE SECTION;

....

EXEC SQL SELECT EMPNO,ENAME,SAL,COMM
    INTO :emp_number,:emp_name,:salary,:commission
FROM EMP
WHERE DEPTNO=:dept_number;
```

在使用数组时，应注意以下几点：

- 不支持指针数组
- 只支持一维数组，而 emp-name [100][15] 视为一维字符串
- 数组最大维数为 32767
- 在一条 SQL 语句中引用多个数组时，这些数组维数应相同
- 在 VALUES, SET, INTO 或 WHERE 子句中，不允许把简单 SQL 变量与数组 SQL 变量混用
- 不能在 DECLARE 部分初始化数组

例如：下面的引用是非法的

```
EXEC SQL BEGIN DECLARE SECTION;
Int  dept - num [3] = {10,20,30};
EXEC SQL END DECLARE SECTION ;

EXEC SQL SELECT EMPNO, ENAME , SAL
    INTO : emp - num [ i ], : emp - name [ i ], : salarg [ i ]
FROM EMP
```

(5) 伪类型 VARCHAR 的说明和引用

VARCHAR 变量在引用之前也必须在说明段说明，说明时必须指出串的最

最大

长度，如：

```
EXEC SQL BEGIN DECLARE SECTION;
    Int  book - number;
    VARCHAR  book - name [ 50 ];
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/057034050106006035>