

系教材之二

# ORACLE关系数据库管理系统 用户手册

Pro \* C      用户指南

## 编者的话

《ORACLE数据库管理系统使用手册》系列资料是根据美国Oracle公司ORACLE5.1版翻译出版的。目前与读者见面的是该系列资料的第一批共八册书。

ORACLE关系数据库是用高级程序设计语言C编写。它采用SQL数据语言，具有广泛的兼容性和可移植性，是目前国际上唯一可在世界各主要计算机厂家70余种大中小微机（IBM的4300系列、3000系列、PS/2、PC；DEC的VAX各系列、PDP系列；DG系列；DPS系列；国产0500、0600系列微机和2000系列小型机、超级小型机等等）系统上运行的RDBMS。

ORACLE关系数据库系统可为不同类型的计算机提供整体化的标准软件环境；与SQL/DS和DB2、DB3等数据库系统兼容；能够直接使用IBM现有的数据库系统的数据和应用程序；可在MVS、VM、VMS、DOS、UNIX、XENIX、UX等十几种著名操作系统下运行，具有相同的用户界面，使用户在更换或升级系统时都不会影响已开发的软件；还为用户提供一整套包括格式化处理、菜单管理、表格图形、报表生成等第四代语言工具在内的支撑工具环境；另外还有很强的数据词典和决策支持系统等功能。

本资料完整、系统地介绍和分析了ORACLE RDBMS5.1的全部功能特点和基本原理；全面地讲解了它的操作方法及步骤，以及各有关注意事项和维护知识。

因此本资料既可为广大计算机用户及软件人员使用ORACLE数据库的培训材料和操作指南，又可作为计算机科技工作者、大专院校师生学习掌握大型数据库管理系统的一般教材。

由于时间仓促，水平有限，不当与错误之处在所难免，欢迎广大读者批评指正。

本套资料中的《SQL,Pro,C 用户指南》由杨守忠、林维达翻译，宗拔梅、马应一对。

《计算机技术》编辑部

一九八九年三月

前言

# 目 录

前言	.....
<b>第一部分：Pro*C预编译程序接口</b>	
<b>第一章 Pro*C入门</b>	(5)
· 何谓Pro*C? .....	(5)
· Pro*C的特点和长处.....	(5)
· 基本概念.....	(6)
· Pro*C命令.....	(6)
· 混合C命令和SQL语句.....	(6)
· 命令前缀 EXEC SQL.....	(6)
· 命令前缀EXEC ORACLE.....	(6)
· 可执行和说明性SQL语句.....	
· .....	(6)
· Pro*C程序的组成.....	(7)
<b>第二章 Pro*C程序的成分</b>	(7)
· 应用程序头部.....	(7)
· 头部：DECLARE段.....	(7)
· 宿主变量.....	(8)
· 宿主变量准则.....	(8)
· 指示器变量.....	(8)
· 指示器变量准则.....	(9)
· 作为宿主变量的指针说明.....	(9)
· 说明VARCHAR伪类型.....	(9)
· 说明段的注意事项.....	(10)
· 头部：说明SQL通信区.....	(11)
· ORACA SQLCA的扩展.....	(12)
· ORACA中包含的信息.....	(12)
· ORACA版本1.1的新特点.....	(12)
· 如何使用ORACA.....	(13)
· 头部：联结到ORACLE.....	(13)
· 通过自动注册的联结.....	(14)
· 通过SQL*Net的联结.....	(14)
· 应用体.....	(14)
· DECLARE STATEMENT	
· 语句.....	(15)
· DECLARE DATABASE	

<b>第三章 查询</b>	(27)
· 查询的组成.....	(28)
· 输入宿主变量.....	(28)
· 输出宿主变量.....	(28)
· 只返回一行的查询.....	(28)
· 数据转换.....	(29)
· 数值数据转换.....	(29)
· 字符数据的转换.....	(29)
· 转换错误.....	(29)
· 返回多行的查询.....	(30)
· 指针的使用.....	(30)
· DECLARECURSOR语句.....	(30)

OPEN CURSOR语句.....	(30)	EXECUTE IMMEDIATE 的先决条件.....	(46)
取活动集合中的行.....	(31)	EXECVTE IMMEDIATE 的例子.....	(46)
CLOSE CURSOR语句.....	(31)	方法2：使用PREPARE和 EXECVTE .....	(47)
CURRENT OF CURSOR语句 (32)		PREPARE和 EXECVTE 的限制.....	(47)
指针的种类.....	(32)	PREPARE和EXECVTE 的例子.....	(48)
程序实例.....	(33)	方法3：PREPARE, OPEN 和FETCH.....	(49)
带 WHERE 从句的查询.....	(33)	说明PREPARE, DECLARE, OPEN, FETCH的例子.....	(49)
使用提示的更复杂的查询.....	(33)	方法4：使用描述(DesCri- tor).....	(51)
<b>第四章 交付和滚回当前事务.....</b>	(35)	SQL的描述区(SQLDA) .....	(51)
工作逻辑单元.....	(35)	SQLDA各元素的意义.....	(52)
工作单元的开始.....	(35)	新的DESCRIBE的作用.....	(52)
工作单元的结束.....	(35)	处理运行查询.....	(54)
一个工作单元需要的资源.....	(35)	准备SQL语句.....	(54)
交付工作(Com MIT WORK) .....	(36)	为语句说明一个指针.....	(55)
撤销工作(ROLL BACK Work) .....	(36)	分配一个描述器.....	(55)
版本选择.....	(36)	BIND 描述器的描述.....	(55)
<b>第五章 检错和恢复.....</b>	(36)	打开指针.....	(55)
使用指示器变量中的返回值.....	(36)	Select 描述器的描述.....	(56)
使用指示器变量和空值.....	(37)	从活动集中取行.....	(56)
SQLCA结构.....	(38)	指针的关闭.....	(56)
何时查阅SQLCA.....	(38)		
SQLCA中各元素的意义.....	(38)		
WHENEVER语句.....	(40)		
WHENEVER语句的语法.....	(40)		
NOT FOUND行为的变化.....	(41)		
WHENEVER语句的 作用域.....	(41)		
WHENEVER与显示错误 校验.....	(41)		
程序实例.....	(42)		
<b>第六章 动态定义语句.....</b>	(43)		
动态定义语句的定义.....	(44)		
动态定义语句的类型.....	(44)		
接受动态SQL语句的输入.....	(45)		
使用DDL语句的注意事项.....	(45)		
方法1：EXECVTE IMMEDIATE .....	(45)		

ENDLABEL.....	(58)
ERRORS .....	(58)
FORMAT .....	(58)
HOLD-CURSOR和RELEASE	
-CVRSOR选择项.....	(58)
HOST.....	(59)
INCLVDE.....	(59)
IRECLEN.....	(59)
LITDELIM .....	(59)
LNAME.....	(59)
LRECLEN .....	(60)
LTYPE .....	(60)
MAXLITERAL .....	(60)
MAXOPENCVRSORS .....	(60)
ONAME.....	(60)
ORACA .....	(60)
ORECLEN .....	(60)
PAGELEN .....	(60)
REBIND .....	(60)
USERID .....	(61)
XREF .....	(61)
使用 REBIND 的程序例子.....	(61)
编译和联结.....	(63)
条件预编译.....	(63)
分别预编译.....	(64)
混合Pro*C和 ORACLE 调用接 口的程序.....	(65)
预编译期间应做什么? .....	(65)
运行一致性检查.....	(65)

## 第二部分：Pro\*C ORACLE调用接口

### 第八章 写ORACLE调用接口

程序引言.....	(66)
基本程序结构.....	(66)
指针数据区.....	(68)
注册数据区(LDA).....	(71)
程序接口数据区.....	(71)
一般编码的规则.....	(72)
任选参数.....	(72)
使用的替代变量.....	(73)
使用的指示器变量.....	(73)
关于使用编译程序优化器 的说明.....	(74)

### 第九章 单个程序调用的描述.....

OLON 调用.....	(74)
ORLON 调用.....	(75)
OOPEN 调用.....	(76)
OSQL3 调用.....	(77)
ODSC 调用.....	(77)
ONAME 调用.....	(79)
ODEFIN 调用.....	(79)
OBNDRV 和 OBNDRN 调用...	(81)

OOPT 调用.....	(83)
OEXEC 调用.....	(83)
OEXN 调用.....	(84)
ORES 调用.....	(84)
OFETCH 调用.....	(84)
OFEN 调用.....	(85)
OBREAK 调用.....	(86)
OCAN 调用.....	(86)
OCOM 调用.....	(87)
OROL 调用.....	(87)
OCON 调用.....	(87)
OCOF 调用.....	(88)
OERMSG 调用.....	(88)
OCMN 调用.....	(88)
OCLOSE 调用.....	(88)
OLOGOF 调用.....	(89)

### 第十章 OLDpRO\*SQL(OCI)

调用.....	(89)
OLOGON 调用.....	(90)
OSQL 调用.....	(90)
ODSRBN 调用.....	(91)

ODFINN 调用	(92)
OBIND 和 OBINDN 调用	(93)
<b>第十一章 数据类型</b>	<b>(94)</b>
数据类型的描述	(94)
数据转换	(97)
附录 A Pro*C 的错误信息	(98)
PC0 的错误信息	(99)
PC1 的错误信息	(99)
PC2 的错误信息	(100)
运行时间错误	(100)
附录 B 程序设计的一般准则 (略)	
附录 C 保留字 (略)	
附录 D Pro*C 程序实例 (略)	
附录 E 使用动态 SQL Pro*C 程序实例 (略)	
附录 F C 语言的例程序 (略)	
附录 G 程序调用参考 (略)	

在本章中，我们首先对 Pro\*C 的数据类型进行了简要的介绍，然后对如何将 C 语言的数据类型转换为 Oracle 的数据类型进行了说明。接着，我们对 Pro\*C 的错误信息进行了详细的分析，帮助读者更好地理解 Pro\*C 的运行机制。最后，我们还提供了几个附录，包括程序设计的一般准则、保留字、Pro\*C 程序实例、使用动态 SQL 的程序实例、C 语言的例程序以及程序调用参考，以便读者在实际应用中能够更好地利用这些资源。

## 第十二章 程序设计的一般准则

在本章中，我们将讨论程序设计的一般准则。首先，我们将介绍如何编写清晰、易于阅读和维护的代码。其次，我们将讨论如何有效地组织代码，使其易于理解。然后，我们将讨论如何处理常见的编程问题，如内存泄漏、线程安全性和并发控制。最后，我们将讨论如何进行单元测试和集成测试，以确保程序的质量。

## 第十三章 附录 A Pro\*C 的错误信息

在本章中，我们将提供 Pro\*C 的错误信息附录。该附录列出了所有可能的错误代码及其含义。通过阅读该附录，读者可以更好地理解 Pro\*C 在运行时遇到的各种错误，并能够更快地解决问题。同时，该附录也提供了关于如何处理这些错误的建议，帮助读者更好地利用 Pro\*C 进行开发。

# 前 言

## 目的

本指南为程序员用高级语言进行应用程序设计提供资料。该应用程序可定义和操作ORACLE关系数据库管理系统的数据。

ORACLE为应用程序设计者提供了两个程序接口，第一个是预编译程序接口，可能对开发应用最有用。通过这个接口可以使用高级语言（如FORTRAN, Pascal, C, PL/1, 或者Cobol）来编写应用程序。高级语言中包含用SQL语言书写的嵌入语句。预编译程序把嵌入程序里的SQL语句翻译成高级源代码。然后经编译、与适当的运行库联结，而成可执行代码。

使用Pro\*C预编译程序能够使应用程序设计者在某一个应用中，把C语言和SQL语言的特点最完美地结合起来。

第一个接口，即ORACLE调用接口，（以前称作高级接口或HLI）也允许高级语言应用程序在ORACLE关系数据库管理系统中存取数据。使用ORACLE调用接口的程序可以直接调用语言中指定的运行时（run-time）程序库中的ORACLE子程序。

这两个接口增强了定义和操作ORACLE数据库中的数据的应用程序的开发过程。

## 读者对象

本指南是为指导应用程序设计者如何使用ORACLE提供的高级接口而编写的，本书的读者应熟悉以下领域：

- 用C语言编写程序
- 在ORACLE RDBMS中存取数据
- SQL语言

## 本指南体系

本指南分为两个部分：第一部分包括如何使用C语言预编译程序接口的资料，第二部分说明ORACLE调用接口（OCI）。本指南的组织结构如下：

1. Pro\*C入门这一章是对Pro\*C的一般性介绍，它将讨论为什么要使用它，Pro\*C的基本概念和定义，Pro\*C程序的组成，Pro\*C程序中语句的类型，以及PCC命令的一个例子。
2. Pro\*C程序的要素这一章将讨论Pro\*C程序中使用的变量说明，并讨论应用头部（Application Prologue）里所要求的事项。另外还包括几个简短的代码实例，从进入和退出ORACLE到很短的能够建立表格或添加记录的程序。
3. 查询这一章介绍指针的概念，它用于返回查询的结果。在介绍了各种各样的指针命令之后，还举出了两个程序例子。
4. 交付和撤销工作这一章对一项工作的事务处理或逻辑单元下定义，它把几个SQL语句聚集成一个单元，程序设计者可以编写一个事务处理程序，当程序完成事务处理后，由程序把事务交付给数据库，或从数据库中撤销事务。
5. 错误检测和恢复，在这一章设计者将学会如何在各种情况下使用SQLCA和指示器

变量以检测和处理各种情况，例如空值，非条件删除，无返回行以及数据溢出或截断等。

6. 动态定义语句这一章涉及高级程序设计技巧，用于编写极灵活的程序，但要求程序设计者对C语言和SQL语言的编码有更深的理解。对于初次使用Pro\*C的用户来说，可以跳过或者略读这一章。

7. 调用Pro\*C (PCC命令) 这一章描述了PCC的命令和它的选择项。

8. 编写ORACLE调用接口程序入门这一章介绍了编写Pro\*SQL程序所使用的概念，例如指针数据区，注册数据区 (LDA)，返回代码，参数类型以及程序结构。

9. 单个的程序调用的描述这一章介绍了各个OCI调用，每个调用都有一节说明一般句法，并用C语言给出了一个例子。调用是按照它们出现的一般顺序讨论的，而没有按字母表顺序（在附录中可以查到按字母表顺序排列的（调用一览表）。

10. 过去的OCI (Pro\*SQL) 这一章概述了早些时候的调用，它们已被第二章的那些元素所代替，但出于完整起见一并收编在此。各个调用都有一节表明一般句法，包括关于它的用途和参数的讨论，并用C语言给出了一个例子。调用是按它们在一个程序中出现的一般顺序进行讨论的，没有按照字母表的顺序（在附录中按字母表顺序列出了这些调用）。

11. 数据类型这一章对OCI和ORACLE RDBMS中使用的数据类型以及可能用到的数据转换进行了讨论。

附录A：Pro\*C的错误信息，该附录列举了Pro\*C的错误信息。

附录B：C语言程序设计准则，该附录对编写C语言程序给予了一般性的提示。

附录C：保留字，该附录列举了专用于ORACLE RDBMS或Pro\*C的保留字，它们不允许用于用户命名的ORACLE目标，例如表格，栏目或视图。

附录D：Pro\*C程序实例，该附录列出了分发给Pro\*C用户的目地代码介质上的一个实例程序清单。

附录E：用动态SQL语言编写的Pro\*C程序实例，该附录列出了一个以动态SQL语言编写的实例程序的清单，它也包括在分发给Pro\*C用户的目地代码介质上。

附录F：使用ORACLE调用接口的C语言程序一例，该附录列举了一个完整的C语言程序，以提示用户在校验之后如何在数据库上输入（雇员信息）和增加新的雇员记录。

附录G：程序调用一览表，该附录按字母表顺序排列了本指南中所有的OCI调用，对于快速查找是很有用的。

## 有关文献

阅读本指南时你可能想要参考ORACLE公司出版的下列文献。你会收到和你购买的产品有关的文献题录，你没有必要也不可能得到全部文件。因为版本经常更新以反映产品的最新变化。

Oracle公司的出版物有：

ORACLE RDBMS的文件部分是：

- ORACLE RDBMS版本发行说明 · ORACLE 实用程序用户指南
- ORACLE 综述与SQL入门 · ORACLE 报告用户指南；RPF/RPT
- ORACLE 数据库管理员指南 · ORACLE 的错误信息与代码

各种预编译程序产品 (Pro\*C, Pro\*FORTRAN, Pro\*COBOL, Pro\*PL/11) 的资料是：

- Pro\*ORACLE 版本发行说明 · Pro\* PL/1 用户指南
- Pro\*C 用户指南 · Pro\* Pascal 用户指南
- Pro\*COBOL 用户指南 · Pro\* Ada 用户指南
- Pro\*FORTRAN 用户指南
- SQL\*PLus 产品文件部分是:
- SQL\*PLus 版本发行说明 · SQL\*PLus 的快速流览
- SQL\*PLus 用户指南 · SQL\*PLus 快速参考
- SQL\*Forms 产品的文件是:
- SQL\*Forms 版本发行说明 · SQL\*Forms 用户快速参考
- SQL\*Forms 操作人员指南 · SQL\*Forms 设计者快速参考
- SQL\*Forms 设计者指南 · SQL\*Forms 快速流览
- SQL\*Forms 设计者参考
- SQL\*Menu 产品的文件是:
- SQL\*Menu 版本发行说明 · SQL\*Menu 用户指南
- Easy\*SQL 产品的文件是:
- Easy\*SQL 入门 · Easy\*SQL 快速流览
- Easy\*SQL 用户指南 · Easy\*SQL 参考卡
- SQL\*Graph 产品的文件是:
- SQL\*Graph 版本发行说明 · SQL\*Graph 快速流览
- SQL\*Graph 用户指南 · SQL\*Graph 快速参考
- SQL\*CaLe 产品的文件是:
- Lotus1-2-3 用户SQL\*CaLe入门 · SQL\*CaLe 的快速流览
- SQL\*CaLe 版本发行说明 · SQL\*CaLe 快速参考
- SQL\*CaLe 用户指南

对支持ORACLE的任何操作系统要提供安装和用户指南，如

- DEC VAX/VMS ORACLE安装和用户指南, IBM VM/SP ORACLE安装和用户指南

### **本指南使用的约定**

在ORACLE RDBMS以及相关产品的文件说明中可以看到下列约定:

文件名: 文件名以大写字母出现, 如INIT.ORA。文件名中可能变化的部分以小写字母表示, 如SGADEFx.ORA。

保留字与关键字 这些字也是以大写字母出现在例子中和正文中, 表示它们将照原样录入, 并且在ORACLE里保留意义。

键名: 键名以大写字母出现, 并用括号括起来, 如 [RETURN]。

### **命令句法**

命令: 这种字体用于标识正文, 必须严格遵照所示录入。

SELECT \* FROM

变量: 变量以斜体字出现, 用户必须用适当的值代替。

选择项 选择项总是以竖杆隔开。如果所选之项是要求的, 选择项集合由大括号括起, 如果是任意的, 则由中括号括起。(要求项和任意项的惯用标志见下)

- 要求项** 要求项由大括号括起，用户必须在选择项中选出一个。
- 任选项** 任选项由中括号括起。注意：中括号也可以只括起字的一部分，如S [elect]。

**缺省值** 缺省值用下划线标识，缺省值经常表示相同类型变量的一种选择。

**重复项** 表示有任何多个相似项的省略符号。

**新资料或修改资料** 对于新增加或修改的材料在边界处加线加以突出。

当下列符号在命令格式中出现时就要如实录入

. 句号 ; 分号

, 逗号 : 冒号

— 连字号 = 等号

## 第一部分 Pro\*c预编译程序接口

下面的章节对Pro\*C预编译程序接口进行了完整的描述。第一部分全面描述了Pro\*C程序的结构，给出了Pro\*C语句的句法，另外还涉及错误处理，事务控制、指针、预编译程序使用、命令选择和程序设计准则，对于每个概念都举出大量的例子加以解释。

### 第一章 Pro\*c入门

本章是对Pro\*c的一般性介绍。它将谈到你为什么要用pro\*c、pro\*c的基本概念和定义，Pro\*c程序的组成以及其中出现的语句类型。

SQL数据语言是非过程语言。就是说大多数语句都是独立执行、与下文无关。而程序设计语言如C，Fortran、COBOL、PL/I等，叫过程语言。是建立在“循环”、“分枝”或“if/then”对这样的结构基础之上的。而SQL语言是一种很有效的语言；不过它也有缺乏过程能力的局限性。

SQL语言的创造人明确地将SQL设计成非过程语言，他也很清楚这样一种语言的局限性，于是把SQL设计成可嵌套在过程性程序语言之中，如C语言，利用这种结构。程序设计者在应用时，就可以把SQL的长处和程序语言（宿主语言）的长处结合起来，这样的应用程序无论比单用C语言还是单用SQL语言的应用程序都更有效也更灵活。

ORACLE关系数据库管理系统提供了几种工具，为设计者用宿主语言编写的程序在ORACLE数据库中取存数据，现有用于FORTRAN、COBOL、Pascal和PL/I等语言的工具。

#### 何谓Pro\*c？

Pro\*C工具是由ORACLE RDBMS提供的，它把含有SQL语句的C语言程序转换成C语言程序，能在ORACLE数据库中存取和操作数据。作为预编译程序，Pro\*C把输入文件中的EXEC SQL语句转换成输出文件中适当的ORACLE调用，输出文件随后就可以以C语言程序的正常方式进行编译，联结和执行。Pro\*C（或类似产品：如Pro\* Fortran和Pro\*PL/I）相当于ORACLE调用接口（以前称Pro\*SQL）。ORACLE调用接口（OCI）是ORACLE数据库的调用接口，它允许用户把ORACLE调用直接嵌入高级语言中，如C，FORTRAN或者COBOL，每一处理都是通过多个调用和使用指针来完成的。

#### Pro\*C的特点和长处

Pro\*C预编译程序的一些特点如下：

- Pro\*C调用的概念性较强，因而比Pro\*SQL（OCI）调用易于理解。

- 一个Pro\*C（OCI）调用自动翻译成等效的几个运行时程序库调用，减少了程序设计的时间。

- 可以用一个程序对应不同数据库中的数据。

- 多道程序可以分别编译共同执行。

## **基本概念**

使用Pro\*C工具给设计者设计程序的正常步骤增加了一步，不过这附加的一步却使 Pro\*C 工具替设计者做了相当数量的工作。编写和执行C语言程序的正常步骤如图 1 所示，

1. 编写C语言程序。
2. 编译此程序，产生一个输出目标文件。
3. 连接编辑此目标文件，产生一个可执行文件。
4. 运行程序完成任务。

**图1 编写C程序的步骤**

如果设计者在原来的程序中加入了Pro\*C语句，那么在编程的开始要增加一个步骤，如图2所示。

1. 编写Pro\*C程序
2. 利用Pro\*C对程序进行预编译产生输出文件
3. 编译程序，产生目标文件
4. 连接编辑目标文件，产生可执行文件
5. 运行程序，完成任务

**图2 编写Pro\*C程序的步骤**

## **Pro\*C命令**

运行Pro\*C预编译程序的句法和选择在“调用Pro\*C (PCC命令)”中有详细的叙述。在那一章中还可提供了预编译几个一起运行的文件以及Pro\*C和OCI一块使用的资料。

### **混合C命令和SQL语句**

任何有效的SQL语句均可以在C语言程序中执行。尽管Pro\*C程序有一些必要的“组成部份”或语句而且按基本的顺序出现，但C语言编写的程序行可以出现在Pro\*C程序的任何地方（当然要符合C程序设计的标准）。必要的组成部份在“Pro\*C程序的成分”中有介绍，在其后的章节中介绍得更详细。

#### **命令前缀EXEC SQL**

SQL语句散布在很多不同的宿主语言中可能会带来很多语言上的困难。为了尽量减少这些困难。SQL语句都以 EXEC SQL 作前缀，于是。预编译程序的一个作用，就是把以 EXEC SQL 开头的语句翻译成适当的可调用数据库的源语言代码（本指南为C语言），

#### **命令前缀EXEC ORACLE**

大多数Pro\*C语句以EXEC SQL为前缀，还有一些语句以EXEC ORACLE为前缀。SQL语句都要求以EXEC SQL作前缀，但有一小部分固定的命令要求以EXEC ORACLE作前缀。参看“EXEC ORACLE选择项”，其中列举了这些语句并作了描述。这些语句不与SQL/DS或DB2兼容。对ORACLE预编译程序是唯一的。

#### **可执行和说明性SQL语句**

Pro\*C程序中的SQL语句可分为两大类，一类是可执行的，另一类是说明性的，全部语句，无论是可执行语句还是说明性语句，皆以EXEC SQL作前缀。

可执行SQL语句就是产生对数据库实际调用的SQL语句，它们包括数据操纵语句(DML)，数据定义语句(DDL)和数据控制语句(DCL)，但并不局限于这些。一个可执行SQL语句执行之后，SQLCA (SQL通信区) 就得到一批返回代码。

一项工作的逻辑单元以执行第一个可执行SQL语句开始，但CONNECT语句除外。因此，在CONNECT，COMMIT，或者ROLLBACK WORK语句之后碰到第一个可执行SQL语句将开始一项新的工作逻辑单元。

说明性SQL语句不产生代码，也不对逻辑工作单元发生作用。SQLCA不受说明性语句的影响。说明性SQL语句如图3所示。

```
# SQL说明性语句
BEGIN DECLARE SECTION
END DECLARE SECTION
WHENEVER .....
DECLARE CURSOR .....
INCLUDE .....
```

图3 SQL说明性语句

### Pro\*C程序的组成

一个Pro\*C程序包括两个组成部分，这两部分是Pro\*C程序处理程序所要求的：

- 应用程序头部（见“应用程序头部”）。
- 应用程序体（见“应用程序体”）。

在应用程序头部定义各个变量，并为Pro\*C程序作好一般性的准备工作。应用程序体基本上是Pro\*C调用，包括SQL语句。如INSERT或UPDATE，对ORACLE数据进行操作。C语言代码可以包围任何Pro\*C处理器所要求的代码段。

下一章：“Pro\*C程序的成分”，描述了应用程序的头部，并列举了一些短小的Pro\*C程序实例来解释Pro\*C程序的基本原则。

## 第二章 Pro\*C程序的成分

本章通过定义几个Pro\*C专用名词和举例帮助读者识别Pro\*C程序中必不可少的成分，结尾部分是几个短小的Pro\*C程序实例，它们将告诉你一个Pro\*C程序起码的要求，和编写第一个程序时应建立的一些概念。

本章中，C程序指的是需进行预编译的模块，即使该模块定义了多种功能或仅仅是应用程序中若干模块之一。

### 应用程序头部

Pro\*C程序开始后紧接的是应用头部，它总是包括三部分：

1. DECLARE段，命名宿主变量
2. INCLUDE SQLCA语句，命名SQL通信区，为错误处理作准备。
3. CONNECT语句，联结ORACLE RDBMS。在Pro\*C程序中只有C程序设计语句可以放在这些语句的前面。

### 头部：DECLARE段

C程序中要用到的所有宿主变量都要在DECLARE段中命名（说明）。DECLARE段以下面语句开始：

```
EXEC SQL BEGIN DECLARE SECTION;
```

以下面语句结束：

```
EXEC SQL END DECLARE SECTION;
```

在这两个语句之间仅允许有一种用于说明宿主变量，或说明指示器变量的语句类型。

每个预编译程序单元只允许一个BEGIN/END DECLARE段，但一个程序可以包含多个独立的预编译程序单元。DECLARE段可以是全局说明段，也可以是局部说明段。

如果C程序中的EXEC SQL语句用到一个宿主或指示器变量，而此变量未在DECLARE段中说明，那么在程序预编译时将显示下面信息：

```
Undeclared host variable a at line b in file C
```

其中a为变量名，b为行号，c为文件名。

#### 宿主变量

SQL语句和程序语句中用到的每个值都必须说明为宿主变量。宿主变量的数据类型必须用宿主语言在DECLARE段中说明。其类型不必与定义表时使用的ORACLE数据类型相匹配；ORACLE将完成宿主语言数据类型和ORACLE数据类型之间的合理的转换。

```
EXEC SQL BEGIN DECLARE SECTION
    int Pempno;           /* 雇员号码 */
    char Pname [11];       /* 雇员姓名 */
    int Pdeptno;          /* 部门号码 */
EXEC SQL END DECLARE SECTION
```

图4 DECLARE段实例

图4中的DECLARE段定了三个将在下面的SQL语句中出现的宿主变量(PEMPNO, PNAME, PDEPTNO)，该SQL语句可能在以后的程序中出现(在应用体中)：

```
EXEC SQL SELECT DEPTNO, ENAME
    INTO :PDEPTNO, :PNAME
    FROM EMP
    WHERE :EMPNO = :PEMPNO
```

#### 宿主变量准则

##### 一个宿主变量

- 必须在DECLARE段明确说明
- 说明时必须用一致的大写或小写格式
- 在SQL语句里必须以冒号(：)作前缀
- 在C语句中不得以冒号作前缀
- 不得与SQL保留字同名(见附录C)
- 只能在允许使用常量的地方使用
- 可以有一个相关的指示器变量  
(指示器变量将在下节描述) 注意，在版本1.1中Pro\*C不用NULL终结字符串。

##### 指示器变量

指示器变量是与DECLARE段中定义的宿主变量一一对应的任选变量，主要用于处理空值，用来存贮各个字段返回代码，用其指示“返回空值”或“截断字符字段”。  
也可以用指示器变量打名空值。(“见使用指示器变量中的返回值”)

## 指示器变量准则

### 一个指示器变量

- 必须在DECLARE段中明确说明
- 在说明时必须使用一致的大小写格式
- 必须说明为双字节整型（是采取“short”还是“short int”，取决于给定条件下C语言的实现）
- 在SQL语句中使用时必须以冒号（:）作前缀
- 在C语句中使用时不得以冒号作前缀
- 不得与SQL保留字同名（见附录C）
- 在SQL语句中必须以与其相关的宿主变量作前缀

例如，现在我们在前面的查询中加入两个指示器变量DEPTNOI和NAMEI：

```
EXEC SQL BEGIN DECLARE SECTION;
    DEPTNOI :PDEPTNO; DEPTNOI :PNAME; NAMEI :PNAME;
    EMPNO :PEMPNO;
```

用指示器变量的另一些情况详见“使用指示器变量的返回值”。

### 作为宿主变量的指针说明

在DECLARE段中可以使用指针变量（简单指针），这些变量以正常的C语言方式定义，即用一个星号加上变量名，举例如下：

```
EXEC SQL BEGIN DECLARE SECTION;
    int * i, * j, * intptr;
    char * CP;
EXEC SQL END DECLARE SECTION;
```

在SQL语句中，变量名前有冒号而无星号，则认为星号隐含；

```
SELECT INTFIELD INTO :intptr FROM ...;
```

对于所有非字符指针，被访问目标所占空间由说明中指定的基本类型所占的空间来决定，对于字符指针，被访问目标假定为空值终结的字符串，其所占空间在运行中由字符串长度决定。

注意：如果用字符变量指针作为输入宿主变量（在USING子句中），其长度仍由字符串长度调用决定。它不会产生预期结果，因为它服从于现有长度，而不是最大长度。

### 说明VARCHAR伪类型

Pro\*C允许使用VARCHAR伪类型来容纳可变长的字符串。它只在DECLARE段中使用，可以当作扩展C语言类型或预说明结构。下面的说明：

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR JobDesc [40];
EXEC SQL END DECLARE SECTION;
```

可以展开为下面的结构变量：

```
Struct {
    Unsigned /* 两字节 */ Short int len;
```

```
unsigned char arr [40]
} jobDesc;
```

字符数组由VARCHAR定义中指定的大小来说明。len字段确定了变长字符串的当前长度。

根据定义，VARCHAR变量不再是空值终结的(hull-terminated)，VARCHAR结构中的Len部分包含了字符串的长度，因此，当用作输出变量时由ORACLE设置，len作输入变量时必须由用户程序设置。arr数组中的所有空值均忽略或复写。字符串有以下两种定义方式：

Char Str [100];	作为字符数组或
Char *Str;	作为字符串指针

当用作数组时，其长度由字符串长度决定(如100)。无论输入和输出，都是这个长度。ORACLE将空出(blank-pad)输出缓冲器，相当于用户程序空出(blank-pad)输入字符串。

VARCHAR变量在SQL语句中使用，用集合名，前缀冒号，跟其它变量一样，例如，在上面的说明之后可能跟有下面的两个查询之一：

```
EXEC SQL SELECT JOBDESC INTO :job Desc FROM EMP
Where empno= :emPno;
```

或者

```
get String (jobDesc · arr);
/* Get job description from user */
job Desc · len=Strlen (jobDesc · arr);
/* Get Length of String */
EXEC SQL SELECT...INTO...FROM EMP
WHERE JOBDESC= :jobDesc;
```

当VARCHAR用作输出宿主变量(在INTO从句中)时，ORACLE RDBMS设置其长度字段；当用作输入宿主变量(在WHERE从句中)时，程序应设置其长度。

可以用VARCHAR \* ID把指针定义为VARCHAR数据类型，对于标准C语言类型还可以重复定义，如：VARCHAR ID1 [10]，\*ID2，…，IDN [4]

它等价于：

```
VARCHAR ID1 [10];
VACHAR ID2;
VARCHAR IDN [4];
...等等
```

处理指针时字符串缓冲器的长度可引用Strlen( )函数确定，这对于输入很方便，因为用户程序不必设置len(VARCHARS)或者腾空缓冲器(字符串数组)，然而输出时要麻烦一些，因为ORACLE根据strlen( )函数确定缓冲器长度，它没有必要向用户显示。缓冲器作为字符数组(str [100])比作为字符指针更易于说明和输出。

#### 说明段的注意事项

勿用类型定义语句中定义过的名字。因为Pro\*C不能识别这些名字，使用的结果必然导

致变量来说明等一系列错误。如果试图在结构中使用一个已经定义过的结构，或者勿用前面定义过的结构，同样不能定义一个具有某种结构类型的变量，不过允许通过下面三种方式使用结构：

- 在结构元素与简单变量之间往复拷贝数据
- 为了存取类型T的一个字段，在说明段中说明一个T类型的变量，然后用某个结构元素的地址初始化指针
- 使用描述器和动态语句，这使宿主变量的定位具有灵活性

### 头部：说明SQL通信区

每个Pro\*C程序的应用头部都必须包含一个SQL通信区，供事件处理时访问。只需用一行语句：EXEC SQL INCLUDE SQLCA;

EXEC SQL INCLUDE用于在Pro\*C程序中包含其它文件，为了跨语言兼容此语句代替了C语言中#include约定。EXEC SQL INCLUDE files语句中的文件本身可以有EXEC SQL INCLUDE语句，嵌套的层数受操作系统打开文件能力的限制。

INCLUDE语句应与实际.h文件的字体（大写或小写）相匹配（通常为大写），Pro\*C必须在预编译程序时定位好SQLCA文件，有三个选择：

- 使用INCLUDE=命令行选择

完整地描述文件名以便Pro\*C能够在公共O/S区中找到它，如（VMS中）SYS\$ORACLE7(SQLCA.H）或（Windows中）ORACLE7(SQLCA.H）。

- 把SQLCA文件拷贝到将调用PCC的目录或磁盘上。

如果遇到问题，则与DBA核查。

该语句引导Pro\*C在程序中包含SQL通信区，SQLCA可以全程嵌套也可以局部嵌套。SQLCA中定义的变量用于与ORACLE通信。程序进行预编译时，ORACLE以几种宿主语言变量说明来代替INCLUDE语句。这些说明在程序运行过程中通过传递各种操作信息，使ORACLE与程序实现对口。

SQLCA包括如下信息：状态、消息、时间、日期、日志记录、缓冲区、连接数等。

- 警告标志和事件信息
- 错误代码
- 诊断正文
- 处理的行数

举个例子，你可以核查SQL语句是否运行成功。如果成功，那么插入或修正了多少行，或者，如果语句由于出错而中断，你可以获得更多的信息。

这样，SQLCA在程序各处，根据ORACLE传来的工作情况的反馈，为设计者提供采取不同行动的选择。

Pro\*C程序在可能情况下将暂不处理错误，继续运行。设计者可以通过SQLCA中的变量控制某种情况下采取的行动。例如，可以借助WHENEVER语句规定出错、警告或某些特殊事件时采取的行动。行动可以包括结束程序，分支转移或继续。（WHENEVER语句及选择将在“WHENEVER语句”中讨论）

虽然SQLCA中有好几个变量，我们现在只谈两个：

Sqlca.Sqlcode变量里装着每个可执行SQL语句运行之后的结果代码。代码是一个表