系列读物之二

ORACLE关系数据库管理系统

用 户 手 册

（三）

PRO * COBOL

用户指南

《计 算 机 技 术》编 辑 部

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二批共九册书。

ORACLE关系数据库是用高级程序设计语言C编写的。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；HP系列；国产0500、0690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，且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 RDBMS 5.1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，以及有关注意事项和维护知识。

因此本资料既可作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《PRO * COBOL用户指南》由安桂洁翻译，焦极宸译校。

《计算机技术》编辑部

一九八九年三月

目　录

前　言

目的

本手册为在开发应用程序中使用程序设计接口提供参考资料，这些应用程序是在ORACLE RDBMS中定义和操纵数据的。

ORACLE为应用程序员提供了两种与高级语言的接口方法。第一种，也许是对开发应用最有效的，是预编译程序接口。通过使用预编译程序接口，可以在高级语言（例如FORTRAN，Pascal，C，PL/1，或者Cobol）编写的应用程序中包含嵌入式的SQL语句。这使得在某个应用程序中高级语言和SQL特性的最恰当结合成为可能。

第二种接口，被称为ORACLE调用接口　（以前称为高级接口（HLI））也允许高级语言应用程序访问ORACLE关系数据库管理系统中的数据，使用ORACLE调用接口的程序可以直接调用在语言专用运行时库（the language-specific run-time libraries）中的ORACLE子程序。

这两种接口加快了开发应用程序的进程，这些应用程序是用来对ORACLE 数据库中的数据进行操作的。

读者对象

这本手册是为那些使用ORACLE提供的程序设计接口的应用程序员设计的。

本手册的读者应熟悉下列领域：

· 熟练的用COBOL编写程序

· 对在ORACLE RDBMS中存取数据有一定的经验。

· 熟悉SQL语言

对于SQL语言语法的详细介绍请看《SQL＊Plus用户指南》和《ORACLE数据库管理员指南》

内容概述

本手册仅介绍使用Pro＊COBOL预编译程序接口的有关信息，关于使用ORACLE调用接口的信息从略。

本手册的内容要点如下：

1．Pro＊COBOL简介

这一章是对Pro＊COBOL的一般性介绍。它会说明为什么要使用Pro＊COBOL，一般性概念和定义，Pro＊COBOL程序的组成成分，语句种类和PCC命令的例子。

2．Pro＊COBOL程序的元素

这一章将讨论在Pro＊COBOL程序中使用的变量的说明，应用程序开头（Application Prologue）所要求的项目。本章中还包括几个简短的代码实例，从进入和退出ORACLE到能够建立表或插入记录。

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，

还提供了两个实例程序。

4. 提交和撤销工作 (Committing and Rolling Back Work)

这一章对一项工作的事务处理 (transaction) 或逻辑单元给出了定义，即可以把几个SQL语句组合在一起形成一个单元。程序员可以编写这样的程序，该程序能够在事务处理完成时，用程序控制把该事务提交给数据库，使其在数据库中成为永久的，或者把该事务从数据库中撤销，使其从数据库中被删除掉。

5. 错误检测和恢复

在这一章中程序员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，无条件删除，无返回行以及数据的溢出或截断。

6. 动态定义语句

这一章中涉及较高级的程序设计技巧，这对于编写非常灵活的程序是很有用的，但是要求程序设计者对COBOL和SQL语言的编码有较深入的了解。对于刚刚开始使用 Pro * COBOL的用户来说应该跳过或者略读这一章。

7. 调用Pro * COBOL (PCC命令)

在这一章中将介绍PCC命令和它的可选项。

附录A：预编译程序错误信息

这个附录列举了预编译程序的错误信息。

附录B：程序设计准则

这个附录给出了编写Pro * COBOL程序的一般性要领。

附录C：保留字

这个附录列举了ORACLE RDBMS和Pro * COBOL 使用的全部保留字，这些保留字不能在用户命名的ORACLE目标中使用，例如表名，栏名或视图名。

附录D：Pro * COBOL程序实例

这个附录中包含了样本程序的列表清单，这个程序在给Pro * COBOL用户的目标代码介质上。

附录E：带有动态SQL的Pro * COBOL程序实例

这个附录中包含了使用动态SQL的样本程序的列表清单，这个程序是在用户的Pro * COBOL目标代码介质上。

有关文献

阅读本手册时你可能想要参考ORACLE公司出版的下列文献。

关于ORACLE RDBMS的文件是：

- ORACLE RDBMS版本发行说明
- ORACLE综述与SQL入门
- ORACLE数据库管理员指南
- ORACLE实用程序用户指南
- ORACLE报表编写用户指南：RPF/RPT
- ORACLE错误信息与代码

各种预编译程序产品 (Pro * C，Pro * FORTRAN，Pro * COBOL，Pro * PL/1) 的有关资料是：

· Pro * ORACLE版本发行说明
· Pro * C用户指南
· Pro * COBOL用户指南
· Pro * FORTRAN用户指南
· Pro * PL/1用户指南
· Pro * Pascal用户指南
· Pro * Ada用户指南

关于SQL * Plus的资料是：
· SQL * Plus版本发行说明
· SQL * Plus用户指南
· SQL * Plus快速浏览
· SQL * PLus快速参考

关于SQL * Forms的资料是：
· SQL * Forms版本发行说明
· SQL * Forms用户指南
· SQL * Forms设计者指南
· SQL * Forms用户快速参考
· SQL * Forms设计者快速参考
· SQL * Forms快速浏览

关于SQL * Menu的资料是：
· SQL * Menu版本发行说明
· SQL * Menu用户指南
· SQL * Menu快速浏览

关于Easy * SQL的资料是：
· Easy * SQL版本发行说明
· Easy * SQL入门
· Easy * SQL用户指南
· Easy * SQL快速浏览
· Easy * SQL参考卡

关于SQL * Graph的资料是：
· SQL * Graph版本发行说明
· SQL * Graph用户指南
· SQL * Graph快速浏览
· SQL * Graph快速参考

关于SQL * Calc的资料是：
· Lotus 1-2-3 用户SQL * Calc入门
· SQL * Calc用户指南
· SQL * Calc快速浏览
· SQL * Calc快速参考

对于支持ORACLE的任何操作系统，还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

本手册中使用的约定

在这个手册中给出的实例和程序段都是用 COBOL 语言写的。这些实例都是在 VAX/VMS下开发且运行检验过的，只要经过较少的修改就可以在另一个操作系统下运行。

实例中的命令或语句句法的关键字总是用大写字母给出的，以小写字母出现的名字表示在程序中该名字可依具体情况变化。可选的参数被括在括号中。

这本手册的读者将会看到在实例中有两个表被频繁使用，即SCOTT.EMP表和SCOTT.DEPT表，在《SQL * Plus用户指南》一书中会找到关于这两个表的介绍。

第一部分　PRO*COBOL预编译程序接口

下面的章节将对Pro*COBOL预编译程序接口进行完整的描述。第一部分全面描述Pro*COBOL程序的结构，给出Pro*COBOL语句的句法，另外还涉及错误处理，事务控制，指针，预编译程序的使用、命令选件和程序设计准则，对于每个概念都给出大量实例加以解释。

第一章　PRO*COBOL简介

这一章是对Pro*COBOL 的一般性介绍。它将会告诉你为什么要使用 Pro*COBOL，Pro*COBOL的基本概念和定义，Pro*COBOL程序的组成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、与上下文无关的。而程序设计语言例如C，Fortran，COBOL，PL/1都叫作过程语言，其程序结构都是以“循环”、“分枝”和“if/then”对为基础的。尽管SQL语言在过程能力上有一些缺陷，它仍是一种功能很强，具有强大生命力的语言。

SQL语言的发明者明确地将SQL设计成非过程语言的同时，他也非常清楚这种语言的局限性，于是就把SQL构造成在过程性程序设计语言中可嵌入的，例如COBOL。利用这种结构，程序员在设计应用程序时能够把SQL的最佳特性与程序设计语言（宿主语言）的最佳特性进行结合，这样的应用程序比单独使用COBOL或SQL编写的程序更有效，更灵活。

ORACLE关系数据库管理系统提供了若干工具，供设计者用宿主语言编写程序在ORACLE 数据库中存取数据，现在，在C，Fortran，COBOL和Pascal语言中使用了这些工具。

什么是PRO*COBOL

Pro*COBOL工具是由ORACLE RDBMS提供的，它把含有SQL语句的COBOL程序转换成COBOL程序，并使该COBOL程序能够对ORACLE 数据库中的数据进行存取和操作。作为预编译程序，Pro*COBOL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，随后这个输出文件就能以COBOL语言程序的正常方式进行编译，连接和运行。

Pro*COBOL与ORACLE调用接口相比，PRO*COBOL是ORACLE RDBMS的一部分（类似的产品还有PRO*FORTRAN，PRO*PL/1），而OCI是对于ORACLE数据库的调用接口，它允许用户在高级语言，如C，FORTRAN或COBOL中直接嵌入ORACLE调用。每一个事务都是通过多个调用和使用指针来完成的。注意这里的ORACLE 调用接口是调用Pro*SQL（或宿主语言接口）。

PRO*COBOL的特点和长处

Pro*COBOL预编译程序的一些特点如下：

· Pro*COBOL调用的概念性较强，因而比OCI调用易于理解。

· 对于运行时库来说，一个Pro*COBOL调用自动地被转换成几个调用，从而减少了程序设计时间。

· 对于不同数据库中的数据能够使用同一程序。

· 多个程序可以分别编译共同执行。

基本概念

使用Pro*COBOL工具在程序设计者正常的程序设计步骤中增加了一步，不过这附加的一步却使Pro*COBOL工具为设计者作了相当数量的工作。编写和运行 COBOL 程序的正常步骤如图 1 所示。

1. 编写源程序
2. 编译源程序
3. 连接编辑目标文件，产生可执行文件。
4. 运行程序，完成想要做的工作。

图1 编写COBOL程序的步骤

如果程序设计者在源程序中使用了Pro*COBOL 语句，那么，这些步骤的开头，将增加一个步骤，如图 2 所示。

1. 编写源程序
2. 用Pro*COBOL 对源程序进行预编译，其结果在输出文件中。
3. 对预编译的结果进行编译，结果在输出文件中。
4. 连接编辑上述目标文件，结果是可执行文件。
5. 运行程序，完成要想做的工作。

图2 编写Pro*COBOL程序的步骤

Pro*COBOL命令

运行Pro*COBOL预编译程序的句法和选件在“调用Pro*COBOL（预编译程序命令）”中有详细介绍。你还能在那一章中找到关于为了一起运行而对几个文件进行预编译的信息，以及一起使用Pro*COBOL 和OCI的信息。

混合使用COBOL命令和SQL语句

任何合法的SQL语句均可在COBOL程序中执行。尽管Pro*COBOL 程序中有一些必要的组成成份或语句，且要求按基本顺序出现，但COBOL程序设计行可以在 Pro*COBOL 程序的任何地方出现（当然要遵循COBOL 程序设计标准）。必要的组成成份在“Pro*COBOL程序的组成”中介绍，其详细的描述在随后的几节中。

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，所有的SQL语句都以前缀EXEC SQL开头，以END-EXEC为结尾。如下：

EXEC SQL UPDATE DEPT SET LOC='BELMONT' WHERE LOC='MENLO OPARK' END-EXEC

END-EXEC后面的句点是可选的。预编译程序的一个作用就是把以EXEC SQL为开头的所有语句翻译成适当的可调用数据库的源语言代码（在本手册中为COBOL语言）。这些语句与SQL是不兼容的，但对ORACLE的预编译程序来说是唯一的。

命令前缀EXEC ORACLE

大多数Pro*COBOL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用EXEC ORACLE作前缀，这些语句的列表和介绍请看“EXEC ORACLE的选件”。以EXEC ORACLE为前缀的Pro*COBOL语句也是以END-EXEC作结尾。

可执行的和说明性的SQL语句

Pro*COBOL程序中的SQL语句可分为两大类：可执行的和说明性的。所有的语句，不论是可执行的还是说明性的，都是以EXEC SQL为开头。

可执行的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。一个可执行的SQL语句执行以后，SQLCA（SQL通信区）就得到一组返回的代码。

逻辑工作单元从第一个可执行的SQL语句开始，但CONNECT语句除外。因此，在CONNECT，COMMIT或ROLLBACK WORK语句之后碰到的第一个可执行的SQL语句将开始一个新的逻辑工作单元。

说明性的SQL语句不生成代码，也不对逻辑工作单元发生作用。SQLCA不受说明性语句的影响。说明性的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的说明性语句

Pro*COBOL程序的组成

Pro*COBOL程序由两部分组成，这两部分是Pro*COBOL处理程序所要求的。

- 应用程序头部（在“应用程序头部中介绍”）
- 应用程序体（在“应用程序体中介绍”）

应用程序头部用来定义变量，并为Pro*COBOL程序作一般性的准备工作。应用程序体基本上是对ORACLE数据进行操作的Pro*COBOL调用，其中包括如INSERT或UPDATE的SQL语句。COBOL代码可以出现在Pro*COBOL处理程序要求的任何代码节周围。

下一章，“Pro*COBOL程序的元素”，将介绍应用程序头部，其中包含 Pro*COBOL程序的一些较短的实例，用以解释Pro*COBOL程序的基本原理。

第二章　PRO*COBOL程序的元素

这一章通过定义几个PRO*COBOL专用名词和举例帮助读者识别Pro*COBOL程序中不可少的元素，结尾部分是几个短小的Pro*COBOL程序实例，它们将告诉你一个Pro*COBOL程序起码的要求，以及编写第一个程序时应建立的一些概念。

应用程序头部

每一个Pro*COBOL程序开始后紧接着的是应用程序头部，它总是包括三部分：

1. DECLARE节，命名宿主变量。
2. SQLCA数据结构的说明，通常用EXEC SQL INCLUDE语句来说明。
3. CONNECT语句，连接到ORACLE RDBMS。

在Pro*COBOL程序中只有COBOL程序设计语句能够放在上述语句前面。

头部：DECLARE节

在COBOL程序中所有要使用的宿主变量都要在DECLARE节中命名（或者“说明”）。DECLARE节用下面的语句开始：

EXEC SQL BEGIN DECLARE SECTION END-EXEC.

用下面的语句结束：

EXEC SQL END DECLARE SECTION END-EXEC.

在这两个语句之间所允许的语句类型只有一个，用来说明宿主变量或指示器变量。

每个预编译单位（尽管一个程序可以包含几个独立的预编译单位）只允许有一个BEGIN/END DECLARE节，而且必须在工作存贮节中说明。

如果在COBOL程序的EXEC SQL语句中使用了宿主变量或指示器变量，但没有在DECLARE节中说明，当该程序被预编译时就会出现下面的信息：

*****S***** Undeclared host variable a at line b in file c

这里的a是变量的名字，b是使用变量a的行号，c是使用变量a的文件名。相反，如果变量已在DECLARE节中说明，但程序中又没有使用，也会出现错误。

宿主变量

在SQL语句和程序设计语句中所使用的每一个值都应该说明为宿主变量，这些宿主变量的数据类型必须在DECLARE节中用宿主语言来说明，其数据类型不必与所使用的ORACLE表中的数据类型相匹配；ORACLE将自动完成宿主语言与ORACLE数据类型之间的转换。

图4中说明了三个宿主变量（PEMPNO，PNAME，PDEPTNO），它们在下面的

```
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
77  PENAME          PIC X(9) EALUE SPACES.
77  PSAL            PIC S99999V99 COMP-3
77  PEMPNO          PIC S9(5) COMP VALUE ZERO
    EXEC SQL END DECLARE SECTION END-EXEC.
```

图4 DECLARE节实例

SQL语句中将出现，在后面的程序（应用程序体）中也会出现：

```
EXEC SQL SELECT DEPTNO, ENAME
         INTO :FDEPTNO, :PNAME
         FROM EMP
         WHERE EMPNO=:PEMPNO END-EXEC.
```

Pro*COBOL支持的COBOL数据类型

在DECLARE节中，只能说明下述的COBOL数据类型：

S9(4) COMP	COMP-1
S9(9) COMP	COMP-1
S9[*9]V9[*9]COMP-3	COMP-2
X(n)	COMP-2
COMP-2	COMP-1

变量必须用一个层号（01，77，05）来说明。象在“数据转换”一节中所说的，Pro*COBOL自动地完成从ORACLE数据类型到Pro*COBOL数据类型间的转换。

宿主变量准则

宿主变量

- 必须在DECLARE节中显式地说明
- 必须在SQL语句中将冒号（:）放在前面
- 在COBOL语句中不能把冒号放在前面
- 不能用SQL保留字（见附录C）命名
- 仅能在使用常数的地方使用
- 可以有相关联的指示器变量

（指示器变量在上一节中介绍）。

指示器变量

指示器变量是可选的变量，它与DECLARE节中说明的宿主变量一一对应。它们主要用于处理NULL值。一般用在存贮各个字段时返回的代码，用其指示“返回空值”或“字符字段被截断”。

指示器变量也可在插入NULL值时使用。（见“在指示器变量中使用的返回值”一节。）

指示器变量准则

指示器变量

- 必须在DECLARE节中显式地说明
- 必须说明为两字节整数，如：

VAR-NAME S9(4) COMP.

· 在SQL语句中使用时必须把冒号（：）放在前面

· 在COBOL语句中使用时不能把冒号放在前面

· 不能用SQL保留字（见附录C）命名

· 在SQL语句中必须把与它相关联的输入宿主变量放在前面。

例如，我们对前面的查询增加了两个指示器变量DEPTNOI和NAMEI：

```
EXEC SQL SELECT DEPTNO, ENAME
         INTO :PDEPTNO:DEPTNOI, :PNAME:NAMEI
         FROM EMP
         WHERE EMPNO=:PEMPNO END-EXEC.
```

关于使用指示器变量的详细情况见“在指示器变量中使用的返回值”一节。

说明VARCHAR伪类型

Pro*COBOL允许使用VARCHAR伪类型，以容纳可变长的字符串。它只涉及到DECLARE节，可以当作扩展的COBOL类型或者预说明结构。下述说明：

```
EXEC SQL BEGIN DECLARE SECTION;
   01 ENAME    VARCHAR
EXEC SQL END DECLARE SECTION;
```

可以扩展为下面的结构变量：

```
01  ENAME.
    05  ELENGTH    PIC S9(4) COMP.
05  NAME        PIC X(40).
```

在VARCHAR定义中用指定的大小来说明字符数组。len字段指出了可变字符串的当前长度。

在SQL语句中引用的VARCHAR 变量，象其他变量一样，使用该集合的名字，前面冠以冒号。例如，下述两个查询中的一个可以跟在上面的说明之后：

```
EXEC SQL SELECT JOBDESC INTO :JOBDESC FROM EMP
WHERE EMPNO=:EMPNO END EXEC.
```

或者：

```
ACCEPT TSTRING.
UNSTRING TSTRING
DELIMITED BY ' ' NAME INTO COUNT IN ELENGTH
```

当VARCHAR作为输出宿主变量使用时（在INTO子句中），ORACLE RDBMS设置长度（length）字段。当VARCHAR 作为输入宿主变量使用时，程序应该设置长度字段。

头部：说明SQL通讯区

每一个Pro*COBOL程序的应用头部都必须包含一个SQL通讯区，供事件处理时访问，这只需使用一行语句：

EXEC SQL INCLUDE SQLCA END-EXEC.