如何选用 ORACLE 产品实现数据库管理

总参防化指挥工程学院　谭继尧　张玉起

一年多来，为了运用 ORACLE 实现防化情报数据库管理，我们阅读了大量资料，经过反复实践，感到 ORACLE 产品丰富、功能强大、资料纷繁。许多产品如 SQL * FORMS、SQL * PLUS、SQL * Menu、Pro * C、Pro * FORTRAN 等各自都能单独或利用其它产品实现数据的输入、输出、修改、删除、查询和统计等项功能。但是每种产品都有各自的优点和缺点。单独选用某一种产品都有各自的优点和缺点。单独选用某一种产品都很难设计出理想的管理系统。在我们国家，像数据管理系统这种最终产品，往往是给那些不懂或不很懂 ORACLE 的人使用，因此要求操作过程越简单越好，用户要学习的东西越少越好，这就要求我们选用了在 SQL * PLUS 下运用 SQL 语句建立和修改管理数据库的结构；选用 SQL * FORMS 实现数据的输入、修改和删除；选用预编译接口 PRO * C 实现查询、统计和输出。

一、选用 SQL * FORMS 实现数据输入、修改和删除

SQL * FORMS 能很快地建立起一个独立的系统，它有丰富的表格处理功能。可以进行全屏幕操作，它的触发器的用户出口能实现用户所须的多种功能。FORMS 对表的管理分为三级：即表、块和字段。每个表可有若干个块，各块之间互相独立，每块可有若干字段，设计时可为每个表，块和字段设计出各种触发器，以便实现所需要的功能。可以通过菜单选择，规定各个字段的属性在屏幕处理上，FORMS 将每个表分为若干页，一页上可有若干块，一块的内容也可放在若干页上。运行时就能很便利的对数据库的数进行查询、修改、删除和输入

FORMS 的块分为两种：一种叫控制块，这种块不对 ORACLE 数据库进行直接操作，只用它来完成功能菜单的显示，根据数据用户选择，使用触发器及宏指令交控制转向相应的块中，以完成指定的功能，或者用来完成某些数据运算及数据校验等。另一种叫基本块，用来对 ORACLE 数据为直接进行操作。应用它配合相应的触发器和宏指令甚至用户出口，可实现对数据的插入、删除、修改等务项维护功能。因为各块相对独立，可将一块设计成插入操作，另一块设计成删除或修改，由控制块来协调完成。此外，FORMS 的宏指令提供了从一个表调用另一个表的功能，这样就可以在一个维护管理系统下实现对多个表的维护管理。

二、用 Pro * C 查询、统计和输出

用 FORMS 开发表格也能实现查询，否则便不能对指定记录实现修改或删除，但由 FORMS 开发的表格必须在 IAP 的支持下运行，其处理过程复杂，速度自然要慢，而且其查询输出方式也不够灵活。

用 SQL 语言能直接对数据库查询、统计，速度比之作高级语言访问文本文件要快得多，但它不是过程语言，不能编程，无法建立友好的人机界面。难以实现理想的屏幕管理和打印输出，要求用户熟练掌握它，这是不现实的。

而各种高级语言普遍能实现自顶向下的开发顺序，使程序模块化、控制统结构化。尤其是 C 语言，不仅描述问题的能力强、灵活性好，目标质量高，特别是它丰富的函数调用，为设计高质量的，人机界面友好的程序提供了理想的工具。但是如同其它高级语言一样，它不能直接访问 ORACLE 数据库。这就需要 ORACLE 提供的预编写的程序译接口--Pro * C 有了它就可以编写的程序中嵌套 SQL 语句，直接访问数据库，把查询、统计的结果存入 C 语言的宿主变量中，由 C 语言的函数或语句加

工输出。这样就把 C 和 SQL 有机地结合在一起、充分发挥各自的优势。用 PRO * C 来实现查询统计和输出，至少在以下几个方面是有利的：

1.C 有丰富的屏幕管理功能

在 VAXC 中，屏幕管理软件包 Curses 是由 VAX RTL 函数和宏组成的，利用这些函数和宏可以建立和修改终端屏幕区域，实现光标在屏幕上的自由移动，利用 Curses 的 move()函数和各种 I／O 函数，可以在屏幕的任何位置进行输入、输出操作。利用 getch ()函数可以灵活地控制每个 I／O 操作出现的时机，从而设计出美观实用的屏幕 I／O 格式，构成友好的人机界面。在这一方面，C 比 FORTRAN 和 FORMS 都强，前者只能滚动式地输入、输出，很难构成理想的画面，例如想把某个输入数据显示在相应提示信息的右边就很困难。FORMS 虽然可以全屏幕操作，但它的屏幕管理是按页显示，每页都是一次出现的先后时机，或根据前面的输入信息确定后面的显示内容。而且从运行环境上看，C 编写的程序经预编译-编译-链接后，可以在操作系统下运行，比 FORMS 事发的系统要在 IAP 支持下运行，速度要快。不是在利用 C 的屏幕管理功能时，要特别注意不可使用标准 I／O 设备的 I／O 函数，如 scanf () prinf ()等。

2.C 能利用动态定义语句实现灵活、快速的查询和统计。

前面已提到，在 C 语言编写的程序中，可以嵌套 SQL 语句直接访问数据库，实现快速的查询、统计。其目的做法之一是利用下述五条语句：

```
PREPARE 语句名 FROM 宿主字符串
DECLARE 指针名 FOR 语句名
OPELARE 指针名 [USINC Uarl (, Uar2 , ……)]
FETCH    指针名 INTO Uarl [, Uar2 , ……]
CLOSE    指针名
```

在程序中，每条语句前面都冠以 EXEC SQL。其中“宿主字符串”包含了整个 SQL 查询(统计)语句，语句中除了输出栏目和数据库名须在程序设计时固定外，整个 WHERE 子名中的查询(统计)条件和输出时的计录排序标准(ORDER BY 子句)都可以在程序运行时临时指定。由于用户现场指定查询(统计)条件和输出记录的排序标准是不难的。

3.C 语言有动态内存分配功能

一个实用的数据库，通常要包括几千几万其至更多的记录，当使用动态定义语句查询时，符合条件的记录，也就是进入活动集的记录，少则一两条，甚至没有，多则成千上万条。一般的高级语言都必须把从活动集中取出记录放到预先定义好的数组中，数组的大小是程序设计时确定的。如果定义得太大，则在大多数情况下都浪费存储空间，而且因为这些数组必须在外部说明。因此一直受到整个程序运行结束才能释放，很不合算。如果数组设计得太小，则每次从活动集中取出的记录太少，而活动集中的记录只能顺序取出，再想取已取过的记录只有关闭指针，重新打开，从头重取，当然很不方便，更不便于“二次查询”和“多次查询”，如果使用 C 的动态存储分配函数 ca110c()或 ma110c()，就可以较好地解决这一矛盾。具体做法是设计一个记录类型，使之包括数据库记录的全部栏目，查询时先用 COUNT 函数统计出符合条件的记录数，根据这一结果申请适当的存储空间，用以存放活动集中的全部，从而可以按任何顺序取出符合条件的记录，再配合键的定义，就可以实现向前或向后的“翻页”。在此基本上还可以进行“二次查询”，这样进行要快得多。使用完毕之后，用 free()或 cfree ()函数释放。这样既可以尽量少占存储空间，又可使占用的时间尽量的短。当然，申请动态存储空间也不是无限的，要受到机器内存大小的限制，但比用固定的数组却要好得多。

以上是我们在如何选择 ORACLE 产品实现数据管理的一些粗浅体会，好多问题我们也还正在摸索和实践，有待进一步认识。

▲