C 语言、FMS 与 ORACLE 的联用初探

军事医学科学院计算中心　段　强

目前,功能强、应用面广的 ORACLE 关系型数据库管理系统正广泛地应用于人事信息管理、财务管理等许多事务型的管理信息系统和数据库应用系统。SQL * PLUS 是友好的、非过程化地操纵 ORACLE 数据库中数据的命令语言;PRO * C、PRO * FORTRAN 为高级语言的预编译接口,其中动态定语句的使用增加了开发的灵活性;IMP、EXP 等多种实用程序为数据加载、备份提供了方便,等等。这些优点使 ORACLE 成为开发应用系统的有力工具。

需重点提一下的是 SQL * FORMS。它是全屏幕交互方式下、规范化设计用户界面的环境,在开发应用系统时占有很重要的位置。SQL * FORMS 用第四代语言编程,语言清晰精炼,功能很强。设计的格式近似于实际的表格,可在上面直接对数据进行输入、修改、检索等处理,操作很直观。此外,SQL * FORMS 本身难于实现的功能,大大增强了 ORACLE 的处理能力。

本中心在 VAX 系列机器上开发一个人事信息管理系统,用 ORACLE 实现,宿主语言用 C 语言。在开发过程中,用 SQL * FORMS 设计系统控制菜单、数据录入卡、数据查询显示等模块时,发现 SQL * FORMS 在窗口管理方面存在不足,只好借助于 VMS / FMS 格式管理系统加以补充。下面就结合开发过程,谈谈用 C 语言、FMS 实现并扩充 SQL * FORMS 某些功能所作的一些尝试。

一、菜单显示及选择方面的改进

由于 SQL * FORMS 不提供由用户随意控制字段值来显示属性的功能,菜单选择项均以背景正文方式排列在格式中,因而不能用方向键以光条方式直观地点菜单。点菜单的方式大致可分为两种,一是利用数字键如 1、2 等来选择菜单,只需设一个字段来接受,用一步(step)触发器步来转移,一一对应,无须作复杂的转换。二是在每个选择项后面增加一个反显字段,用 NXTFLD(前一字段)和 PRVFLD(后一字段)功能键在各字段间移动,然后用功能键 ENTER(也可是别的功能键)加以确认。这样,如果还需一个标志以区分光标所在位置,那么每个字段的触发器必须有三步,即字段前显示、字段后清除及确认选择转移。虽然后者比前者要形象,但实现复杂,而且菜单屏幕显得冗余。为此,我们用 FMS 设计了菜单格式,用一个通用的功能子程序来显示菜单并返回选择的值。具体的做法是设计格式时,字段的多少由所有菜单中选择项最多的数目来确定,而处理每个菜单时从确定的菜单格式文件读取选择项信息,并计算出该菜单的实际选择项数目,以便光标移动时确定光标的位置。每次移动光标即把所在选择项改为反显,其余的选择项为正显。这样,利用 FMS 字段的显示属性直观地实现了所有菜单的选择,使之符合有关菜单选择的最新潮流。

二、多窗口方面的改进

在设计人事信息录入卡时,我们有这样的要求:人事基本信息固定有窗口的某部分作为提示信息,而输入工作经历或其他履历时,显示变动的信息。也就是说,屏幕至少分成两部分,一部分不变而另一些部分则根据需要显示不同的信息。这在 SQL * FORMS 当中难以做到,因为格式中不同页面各块之间转换时要清屏。最直接的做法是,在格式的每一个页面的同一位置复制基本信息这一块,每进入一块时重新刷新显示信息,这无疑增加了格式的设计工作量,并且大大减慢了运行速度,而利用 FMS 却能方便地实现。为此,我们利用 FMS 设计了多个格式分别对应上述信息块,并且把每个格式的清除区按照设计的需要加以重新设置,每个格式占用一个屏幕区间,使他们在屏幕上各自位置不重叠地进行显示,这样

可以单独地对各格式进行操作,在同一屏幕上同时显示两个或更多的格式,实现多窗口功能,满足了上述的要求。

三、C 语言、FMS 与 ORACLE 三者的结合运用

ORACLE 为高级语言留下了预编译接口,使得用高级语言直接调用数据库数据成为可能。C 语言是当今世界流行极广的第三代高级语言,具有以下几个特点:

1.面向过程,具有丰富的数据类型及控制流可以弥补ORACLE 在控制过程方面的不足。

2.编程效率高,运算速度快。比如在 PRO * C 中,可用数组存贮多条记录一次存取数据,在联接数据库进行大批量的信息处理时,能加快执行速度。

3.具有很强的字符处理功能,包括字符数据与数值数据之间相互转换,这对于使用 FMS 有极大的好处。

由于 FMS 只用 ASCII 字符串来显示数据,在终端屏幕上显示的所有信息以及从终端操作员那里接收的信息均是作为 ASCII 字符值来表示的,所以要操纵数据值,如用宿主语言从屏幕上接收 FMS 数值字段值,必须先把该字段的 ASCII 字符串转化成数值数据,反之亦然。基于上述原因,我们选择 C 语言作为宿主语言,协调 FMS 与 ORACLE 之间的数据转换换,实现并扩展 SQL * FORMS 的功能。

为了使数据一致,我们把 FMS 格式上各显示字段的长度设成与 ORACLE 数据库中相应字段的长度一样,并且对于浮点数据字段通常在字段格式中固定住小数点的位置,分别输入整数部分与小数部分,这样设计的用户界面相对比较友好。当然,这主要是我们的系统不需要保证数据的高精度,把一些日期值设成是浮点数。

在用 PRO * C 从数据库中取数据时,把各字段的输出宿主变量均定义为相应长度的字符串变量。同时,因格式本身只认识 ASCII 值,为了把一个 FMS 格式上所以字段当作一个整体来读写,定义了不同的结构变量对应不同的格式,每个结构的成员都顺序对应格式中的每个字段。提取结构成员时,需考虑成员的长度用 strnopy()而不用 strcpy(),因为 FMS 向变量传值时,不足部分用空格填充而不用 C 语言的'\O'字符串结束符。当需要处理非字符数据时,用 C 语言提供的标准函数 atoi()、atof()、sscanf() 和 sprintf()当作前处理器和后处理器进行字符数据与数值数据之间的变换。由于高版本的 C 语言提供了结构变量的直接整体赋值,在查询数据库返回多行记录时,可以用 C 语言的动态分配空间功参建立线性链表,每个结点存贮一个记录,能方便地实现往回检索,解决了 PRO * C 回读记录时先得关闭指针,然后再打开、从头开始读记录的毛病。用高级宿主语言直接操纵数据库,对于那些能熟练掌握高级语言的开发人员来说,使用得心应手,能充分增加应用程序的灵活性与可视性。不过,在这三者的结合过程中,我们也发现了下面一些弊端:

1.用 FMS 与 C 语言来实现 SQL * FORMS 的一些查询、修改等功能时,略显有些绕圈。虽然总体实现比较清晰,最好也只是在 SQL * FORMS 难以实现时才这么做。

2.PRO * C 定义宿主变量时不认识结构定义。这样在向 FMS 传送数据时无疑要增加很多变量。若能解决上述问题,用 FMS 将方便得多。还有,PRO * C 的宿主输入变量必须用 VARCHAR 伪类型,也使 C 语言的实现受到限制。

3.用 C 语言调用 FMS 的驱动子程序时也有些问题,比如说对每个与 FMS 有关的字符变量,必须都定义一个描述结构,并且要单独读取一个字段值时,C 语言定义的字符串的长度必须比 FMS 格式定义中字段的域长多一位。

以上写的是我们在用 ORACLE 开发人事管理系统过程中,为了美化用户界面,优化程序设计所做的一些有益的、成果管理系统打下了良好的基础。我们衷心希望能够获得广大同行在这方面的开发经验。

▲