在程序中直接获取 UNIFY 数据库结构

罗辉 (湖南省双峰工商银行)

摘要:本文对 *UNIFY* 数据库的结构进行了分析,并介绍在程序中直接获取任意 *UNIFY* 数据库结构信息的通用方法。

一、UNIFY 数据库结构分析

UNIFY 是基于多用户 UNIX / XENIX 系统的数据库管理系统,它实现了数据词典和用户数据分开管理的技术。其数据词典存放在 unify.db 中,用户数据存放在 file.db 和 file.dbr 里。当用户首次启动 UNIFY 时,UNIFY 从系统的 lib 目录下的 unify.bd 原型词典库中拷贝一份 unify.db 到用户目录中。unify.db 包含了对每个记录的类型、记录名、记录注释,组成记录的每个字段的字段名、类型、长度期望长度,系统参数,屏幕信息描述等信息。用户将符合自己需要的数据库结构设计打入用户目录中的数据词典 unify.db 里,然后由 UNIFY 的"建立数据库"程序按照用户的数据库设计建立用户数据文件 file.db 和 file.dbr。UNIFY 将所有记录类型映射为 file.db。file.db 主要包括所有记录类型的记录数据及与记录数据有关的信息。而 file.dbr 文件是一个联接文件,它或是联到 file.db 文件上或是指向与 file.db 所在的同一磁盘区上。UNIFY 数据文件的存储结构对于用户来说是不透明的,其所有数据采用二进制存储方式,各字段用":"分隔符隔离。

二、直接访问 UNIFY 数据库的几个主要实用函数

UNIFY 具有宿主语言接口,为 UNIX / XENIX 系统下运行的高级语言提供了与 UNIFY 进行数据相互存取的许多功能函数,使用户通过它们可以较灵活地设计特殊的屏幕格式,方便地对数据进行操作和生成报表。

与本文有关的 UNIFY 数据库结构操作的实用功能函数有:

numrecs() :得到数据库中最大的记录类型号。

recname(recno):给出记录类型号 recno,得到记录名。其中 recno 必须满足 1<=recno<=numrecs。如果它返回 0,则 recno 就不是一个有效的记录类型编码。由于有些记录可能在数据库设计过程中被删除。在数据库未重新整理前,它们将在数据库结构中留下一些"空洞",因而并不是所有在 1 到 unmrecs 之间的记录编号都对应一个实在的记录类型。

recsyn(recno) :给出记录类型号 recno,得到长记录名。

munflds() :得到数据库中最大的字段号。

fldname(fldno):给出字段号 fldno,得到字段名。fldno 必须满足 1<=fldno<=numflds。如果它返回 0,则 fldno 就不是一个有效的字段编程。与记录一样,某些字段也有可能被删除,在数据库未重新整理前,它们也将在数据库结构中留下一些"空洞",因而并不是所有在 1 到 numflds 之间的字段编号都对应一个数据库字段。

fldsyn(fldno) :给出字段号 fldno,得到和字段名。

reckey(recno) :为特殊的记录为型 recno 得到一个主关键字字段号。其返回值,如果>0,则为记录类型的键字段号;=0,说明该记录类型无键;<0,为无效记录号。

fldesc(d,f) :返回指定字段号 K 的字段属性,将它们放在第二个参数 f 所指向的结构 FDESC 中。如同 fldname()函数一样,如果它返回 0,则 K 就不是一个有效的字段编码。

函数 fldexe(d,f)中 f 所指的结构 FDESC 内容如下:

f__rec :包含指定字段记录类型号。

f__typ :头文件 dbtypes.h 中给出的字段类型,整型。前面 8 个类型用数字 1~8 表示,而类型"COMB"

用 100 表示。

f__len　:字段长。

f__par　:COMB 字段的父字段烽,本字段为一子字段。

f__rprec:父记录类型(如存在的话)。

f__rpfld:父记录类型中字段的数量。

三、直接访问 UNIFY 数据库结构的通用程序设计

大型商业应用软件都应保证系统的通用性能。开发基于 UNIFY 的数据库管理系统的通用软件,最起码的一点是,要能使程序自动适应不同的数据库结构。通过上面对 UNIFY 的结构和功能函数的简单分析,相信要使程序实现这一点并不困难。为此,笔者编制了一个读取任意 UNIFY 数据库结构的通用演示程序。它直接从 unify.db 数据库中读取数据结构。对每一记录类型,打印它的记录名和记录长名;主关键字的字段名和字段长名;其所属的每一字段的字段名、字段长名、字段长度及字段类型。

用 vi 将程序以文件名 unifystru.c 编辑存盘。然后用 ucc-c unifystru.c 和 uld unifystru unifystru.o 编译连接成可执行程序 unifystru,置入 UNIFY 公用目录 /usr/unify/bin 下,即可在任意 UNIFY 应用环境下演示执行了。

这里需注意三点:

1.程序中引入 UNIFY 头文件时,必须用

include 而不是用#include。这是因为必须用 ucc 而不 cc 编译命令编译该程序。ucc 通过调用 UNIFY 预处理程序 upp 预处理 include 文件。而 upp 将在程序中寻找

include 而不是#include,以决定哪里需要预处理。

2.程序必须在 UNIFY 应用环境下才能执行,这时 UNIFY 自动将 unify.db 库打开了(当然此时 file.db 也已打开)。该程序并没有打开数据库的语句,而是默认 unify.db 已被打开,直接进行读取的。

3.用 fldexe()返回的 FDESC 结构中的字段类型 f__typ 是用数字表示,其与类型名对应关系在 dbtypes.h 文件中定义。因此程序中使用了一个指针数组来指出类型名。其中的算式[.f.f__typ%91-1]就是用于将类型号折算为相应的数组下标的。

四、结束语

在获取数据库的结构信息后,进行数据存取操作的通用程序设计就不再是难事了。通过 UNIFY 丰富的记录函数、关系函数、I/O 函数、事务记录函数等众多实用函数,就可以实现任意的通用性设计。正如其它通用数据库管理系统设计一样。

```
#cat unifystru.c<CR>

include"/usr/unify/include/fdesc.h"

include"/usr/unify/include/dbtypes.h"
static char * fldtype[]={"INT","LONG","DATE-,"AMT","STRNG","HAMT","HR","FLT","COMB"};
main()
{
  int i;
  int j;
  int k;
  int limitrec,limitfld;
  char * name, * sname;
  char * fldname(), * fldsyn();
  char * recname(), * recsyn();
  struct fdesc f;
  limitrec=numrecs();
  limitfld=numflds();
  for(i=1;i<=limitrec;i++)        /* 全部记录循环
                                  */
  {
    printf("\nRecord#%d-----",i);
    if ((namd=recname(i)==0)      /* 剔出无效记录名
                                  */
      continue;
    sname=recsyn(i);              /* 返回记录长名
                                  */
    printf("Record__name:%s-----Long__record__na
```

```
me: %s", name,sname);
        (namd = fldname(j)! = 0)        /* 返回记录的主关
                                        键字名 */
      {
        sname = fldsyn(j);              /* 返回记录的主关
                                        键字长名 */
        printf("\n\t      Key__     field__     name:
%s-----",name);
        printf("Long__key__field__name;%s\n",sname);
      }
      else                              /* 记录无关键字
                                        */
        printf("\n\t The record has no key\n");
      printf("Fild__        No\tFiled__        name
\tField__long__name");
      printf(" Field__len Field__Tupe\n");
      for (d = 0;k < = 1imitfld;k++)    /* 某一记录的全部
                                        字段循环 */
        {
          if ((name = fldname(k){
                                        */
            continue;
          if (f.f__rec! = i)            /* 剔出非本记录的
                                        字段 */
            continue;
          printf("#  %5d:%18s--------%20s-----",k,na
me,fldsyn(k));
          printf("%2d------%4s\n",f.f__  len,fldtpe[f.f
__typ%91-1]);
                                        /* 打印字段的长度
                                        和类型 */
        }
      }
    }
  }
```