起引用的混乱，因此，我们又重载了类的（ ）操作符。该重载类似于上述的下标重载，只是不用查找属性值，直接返回 value 的引用，可以在复杂的语句中使用作为左值。

2.4 两种多版本方法的综合

上述的两种实现多版本的方法，分别适用于不同特性的对象。比如，股票的价格这样的对象适合用多版本对象的方法，而病员（包括体温记录）这样的对象则更适合使用多版本属性的方法。

事实上两种方法在实现时是同时并存于一个系统中的，两者并没有抵触，也就是说，系统同时提供 MVPersistClass 和 PersistClass 以及各种基本属性的多版本类，应用可以按照自己的需要来选用，要注意的是，两种多版本的永久对象在具体的操作上是要区别对待的。

3 应用示例

假使我们要多次记录病人的体温，以备查询。先构造一病员类如下：

```
class    Patient:public Person{
    protected:
        void virtual DeclareValue( );
        void virtual DeclareAddress( );
    public:
        int No;
        MV_int Temp;
        Patient(char * str,int s,int n,int t){
          No=n;
          this->Temp.SetOID(OID(Temp));
          Temp[NULL]=t;
        }
    }
```

在完成 DeclareValue( ),DeclareAddress( )和 RegisterClass( )之后，下面一段 C++的应用完成记录"wang"病员体温的工作。

```
Patient wang("wang",0,17,37);
int     year,month,date,hour,t;
wang.Save( );
wang.Temp.Save( );
cout≪"请登记测量体温的时间(年  月  日  时):";
cin≫year≫month≫date≫hour;
for(;hour<=24;)
{
    cout≪"请输入"≪wang.name≪"的体温:";
    cin≫t;
    wang.Temp[TimeStamp(year,month,date,hour,0,
        0)]=t;
    wang.Temp.Save( );
    cout≪"请登记测量体温的时间(年  月  日  时):";
    cin≫year≫month≫date≫hour;
}
```

下面的 C++过程用来查询"wang"病人的体温情况：

```
cout≪"您需要"≪wang.name≪"什么时间(年  月  日
  时)的体温:";
cin≫year≫month≫date≫hour;
for(;hour<=24;){
  cout≪wang.name;

  int i=wang.time_stamp;
  cout≪"在"≪Year(i)≪"年"≪Month(i)≪"月"
      ≪Date(i)≪"日"≪Hour(i)≪"点的体温是:"
  cout≪wang.Temp[TimeStamp(year,month,date,hour,
    0,0)];
  cout≪"您还需要"≪wang.name≪"什么时间(年  月
    日  时)的体温:";
  cin≫year≫month≫date≫hour;
}
```

本文讨论了在 C++中实现多版本 OODB 的核心，无需任何元级机制来支持，使多版本 OODB 的操纵和 C++语言融为一体。在多版本永久类或者多版本属性类中增加一些时态处理方法，就可以使系统具有时态数据库的特征和功能，这样的功能设施，既是版本 OODB 应用编程的界面，也是进一步完成面向对象查询语言时需要的元级机制。

参 考 文 献

[1] W. Kim, Introduction to Object-Oriented Database, the MIT Press, London, England

[2] Edward Sciore, Using Annotations to Support *Multiple Kinds of Versioning in an Object-Oriented* Database System, ACM Trans. Database System Vol 16, No. 3. 1991

[3] 张成洪、周傲英、徐涌、施伯乐，面向对象数据库系统 FOODB 的数据模型，第十一届全国数据库学术会议论文，1993年9月

（上接第61页）

所示，在图2中同时列出了国外一 GIS 软件在同等条件下的刷新时间。从图2可以看出，在查询窗口小于64%时，增强型 BANG 文件的刷新时间比 GIS 软件快，特别是在小窗口查询时，刷新时间要快得多，这是由于增强型 BANG 文件所需的磁盘 I/O 次数少，但大窗口刷新时，性能相差不大。

从上面还可以得出：数据桶小，则点查询时间短，小窗口刷新时间快，但数据的冗余高，所占用栅格多，随着查询窗口的增大，由于磁盘 I/O 次数增加，其性能变差，数据桶大的增强型 BANG 文件，数据冗余相对小，中等查询窗口的性能好。

参 考 文 献

[1] 詹舒波、张其善，电子地图数据库存储文件的设计，计算机科学（本期）

[2] mapinfo 参考手册

[3] 李文通等，Borland C++3.1，北京航空航天出版社

计算机科学1996 Vol. 23 №. 3

增强型 BANG 文件的实现

詹舒波　张其善
（北京航空航天大学　北京100083）

摘　要　In order to shorten the time of display and query in Digital Map, a index file for the vector data is need. Based on [1], the algorithms of BANG File is given in this paper. The results of a seriers of test is presented.

关键词　Digital Map, Index Structure, Vector Data.

在文[1]中，我们基于电子地图应用的特点，分析了常规存贮文件和几种空间数据存贮结构（如：R-树、K-D-B 树，BANG 文件等）在处理空间复杂对象上所存在的缺点和局限，并提出了适用于电子地图数据库的存贮文件——增强型 BANG 文件结构。本文在文[1]的基础上，将论讨增强型 BANG 文件的操作，给出其查询、插入、分裂等操作算法。

1　增强型 BANG 文件算法

1.1　查找算法

查找算法包括点查询（精确查询）、窗口查询（范围查询）。

1.1.1　**点查询**，是最基本类查询，它完成：当指定数据区域中一点坐标（x，y）时，计算出包括该点的空间对象及其属性，其算法代码如图1(a)所示，其中 Get-Dir()返回点（x，y）所在的目录项；Get-Records-Num 计算目录项 dir-Item 数据桶所包括的记录总数，这次操作将涉及磁盘 I/O；Get-Ploygon-Obj()返回目录项 dir-item 数据桶中第 i 个空间对象（obj）；Am-Contained 判断点（x，y）是否包含在 obj 中，如果是则返回 OK。

1.1.2　**窗口查询**，也叫范围查询，它完成：当指定数据区域中一矩形范围时，返回该范围包含的全部空间对象及属性，其代码如图1(b)所示，其中，win 为窗口结构，它包括窗口起始点 x、y 和窗口宽 width，高 heigh；objs 为返回对象数据。由于窗口可能跨越栅格，所以 Get-Dirs()返回跨越的栅格区集（dir-set），其个数由（dir-num()）返回。Am-Intersection()判断窗口（win）与对象（obj）是否有相交或覆盖。如果是则返回 OK。obj-num 为窗口包括 obj 的个数。

```
;点查询 Point-Query(x,y,objs)
;返回查询到对象(obj)及数目(obj-unm)
Point-Query(x,y,objs)
{
  obj-num=0;
  dir-item=Get-Dir(x,y);
  num-record=Get-Record-Num(dir-item);
  for(i=0;i<num-record;i++){
  obj=Get-Polygon-Obj(dir-item[i]);
  ret-code=Am-Contained(x,y,obj);
  if (ret-code==OK){
  objs[obj-num]=obj;
  obj-num++;}}
  return(obj-num)
}
```

(a)

```
;窗口查询 Window-Query(win,objs)
;返回查询窗口包括的对象(objs)及数目
Window-Query(win,objs)
{
  obj-num=0;
  dir-set=Get-Dirs(W);
  dir-num=Get-Dir-Num(dir-set);
  for(i=0;i<dir-num;i++){
  num-record=Get-Record-Num(dir-set[i]);
  for(j=0;j<num-record;j++){
  obj=Get-Polygon-Obj(dir-set[i][j]
  ret-code=Am-Intersection (win,obj)
  if(ret-code=OK){
  objs[obj-num]=obj;
  obj-num++;}}}
  return(obj-num)};
```

(b)

图1

1.2　插入算法

插入算法为数据库基本操作，在数据库建立时就涉及它。

对增强型 BANG 文件，由于其复杂可能跨越多个栅格，所以 Get-Dirs()返回跨越的栅格区集（Dir-Set），其个数由 Get-Dir-Num()返回。与对象（Obj）相交的每个栅格都保留该对象拷贝 mov-obj，如果此时栅格发生溢出（Overflow()返回 OK），则栅格调用分裂（Split-Grid()）子程序。

```
;插入算法
;无返回
insert(Obj)
{
Dir-Set=Get-Dirs(Obj);
Dir-Num=Get-Dir-Num(Dir-Set);
for (i=0;i<Dir-Num;i++){
```

```
mov-obj(Dir-Set[i],obj);
ret-code=Overflow(Dir-Set[i]);
IF(Ret-Code=OK)
Split-Grid(Dir-Set[i])}
```

1.3 分裂算法

分裂沿 n 维空间的某一坐标进行，把原栅格分成大小相等的两个子栅格。考虑到复杂对象引起的冗余，分裂坐标的选取原则为：①使两个子栅格具有相同对象的数目尽可能小，这样可减少冗余；②使两个子栅格尽可能平衡。这两个原则有时相互矛盾，这时就需要找到一折中方案，但优先考虑的是减少子栅格中相同对象的数目。

分裂算法采用比较法，以二维空间为例，我们先选取 x 轴为分裂线，计算出此时两个子栅格各自包含的对象数及重复对象数。再计算出以 y 轴为分裂线的上述二参数，把这些参数进行比较，决定最后的分裂线。

1.4 目录项查找

增强型 BANG 文件的目录管理占有重要位置，它的主要任务为：维护栅格与数据桶之间的动态关系。

目录项查找也在许多操作中应用，以二维空间为例，坐标(x,y)与栅格序号(nx,ny)具有如下映射关系：

$$nx=\left|\frac{x}{DX}\cdot 2^{level.x}\right| \qquad ny=\left|\frac{y}{DY}\cdot 2^{level.y}\right|$$

其中：DX、DY 为整体空间区域(界限)。level. x,level. y 为 x,y 方向上的分区数。

目录项查找算法代码如下：

```
;查找目录项 Get-Dir(x,y)
;返回目录项 dir-item
Get-Dir(x,y)
{
Get-level(& level x,& level y);
for(i=level y;i>0;i--)
for(j=level x,;j>=0;j--){
Get Region(x,y,j,i,&nx,&ny);
ret-code=Am-Exist(j,i,nx,ny);
if (ret-code==ok){
dir-item=Make-Item(j,i,nx,ny);
return(dir-item)}}
```

目录项查找首先从最小栅格区开始(leve x,level y)，随后判断该栅格序号是否存在(Am-Exist())，如果存在则返回目录项序号(j,i,nx,ny)，否则找上一级(level-1)包含的栅格号，直到找到为止。

Get-Region()按点坐标(x,y)，分区级(j,i)计算出栅格号(nx,ny)，Am-Exist()判断栅格号(j,i,nx,ny)是否在目录中存在，如果是则返回 OK。

2 测试结果及分析

我们把增强型 BANG 文件应用于实际数据，并就电子地图主要涉及的操作：窗口图形放大、信息检索等性能进行了测试。测试的硬件环境为：①486/33PC 机；②8M 内存；③硬盘540M；④VGA 显示，1M 显示内存。软件环境为：①Windows3.1；②编程语言 Borland C++3.1。

测试数据为数据化北京市地图，该数据化地图包括1063个曲面，图形数据文件全长72,462字节，其中曲面面积大于整个空间面积1%的曲面有3个，大于0.1%面积的有195个。

测试过程中，我们采用两种类型的 BANG 文件数据桶：①4K 字节，②8K 字节。相对应定义每个栅格可包含的最大曲面数为：①50个；②100个。

地图数据存贮成 BANG 文件结构后占有的栅格数及冗余如表1所示。

表1 BANG 文件结构存贮文件

数据桶	最大记录	占用栅格数	重复存贮曲面数
4K	50	32	73
8K	100	16	17

BANG 文件点检索平均延迟时间如表2所示。

表2 点检索延迟时间

数据桶	检索时间(20次平均数)
4K	75ms
8K	110ms

BANG 文件的窗口查询性能以查询窗口内图形刷新时间为指标。

设整体区域范围为100×100，按窗口查询的大小分成下列5种：

①查询窗口为80×80，即占整体区域的64%

②查询窗口为50×100，即占整体区域的50%

③查询窗口为50×50，即占整体区域的25%

④查询窗口为20×50，即占整体区域的10%

⑤查询窗口为20×20，即占整体区域的4%

上述每种查询窗口在任意位置对地图进行放大，共进行20次。其刷新时间为20次的平均值，如图2

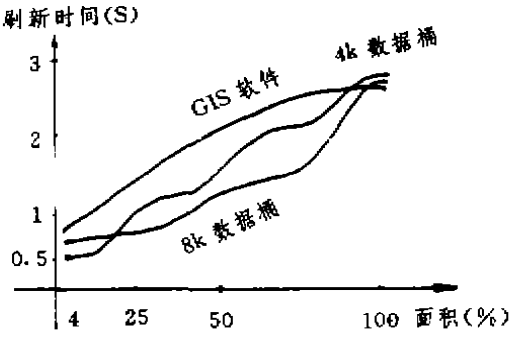

图2 刷新时间

(下转第51页)