地矿测绘 2004,20(3):20~22 CN 53-1124/TD ISSN 1007-9394
Surveying and Mapping of Geology and Mineral Resources

# 基于数据库底层接口的海量空间数据快速操作方法*

谭坤龙,傅炳荣,丁文源

(浙江省测绘大队,浙江 杭州 310030)

**摘要**:利用 VC 面向对象技术将 DB_Lib API 函数封装成类,从而屏蔽了直接调用的许多复杂工作,为其使用提供了简易接口。这种方法可供用客户/服务器结构,利用 DB_Lib 进行 GIS 应用系统开发参考。

**关键词**:DB_Lib,数据库;SQL SERVER

**中图分类号**: P 208 **文献标识码**: B **文章编号**:1007-9394(2004)03-0020-03

# Quick Access to Vast Spatial Data Based on Database API for Implementation

TAN Kun-long, FU Bing-rong, DING Wen-yuan

(*Center of Digital Mapping, Zhejiang Institute of Surveying and Mapping, Hangzhou Zhejiang* 310030, *China*)

**Abstract**: To shield directly many complex works produced by calling directly DB_Lib function, an easy interface for using is presented in this paper. The functions of DB_Lib are encapsulated to create a class by using the object-oriented technology with Visual C++. The implementation of the class has an important reference value to the developing work of the client part for SQL Server using DB_Lib.

**Key words**: DB_Lib; database; SQL SERVER

## 0 DB_library 的有关概念和特点

对于任何一个 GIS 应用系统后台的数据库而言,无论是存入的属性数据或是通过空间数据库引擎存入的空间数据,都涉及到一个数据访问速度的问题。采用诸如 ODBC,DMO,DAO 和 RDO 等方法,速度较慢,常常不能满足 GIS 应用系统对海量数据的操作要求。DB_Library 或 DB_LIB,对 C 和 VB 而言,都是一个 API,它允许用户直接处理 SQL Server。API 提供给用户不同的工具,用户用它们来发送查询到 SQL Server 和从 SQL Server 接收信息。事实上,DB_Library 是如 ODBC,DMO,DAO 和 RDO 这些更高级的 API 的底层公共接口,而它们在不同程度上屏蔽了直接调用 DB_Library 的复杂的接口工作。然而,也正是由于 DB_LIB 较之 ODBC 等数据库引擎而言更加"底层",利用它实现客户——服务器模式的应用程序的前后台通信任务具有程序代码短小、灵活快捷的特点。作者在以 VC、DB_LIB 为工具进行 SQL Server 客户端应用开发的工作中,利用 VC 的面向对象技术,将 DB_Library for C 的部分函数进行了封装,形成 1 个 CDblink 类,以类的成员函数的形式向开发者提供数据库访问的简易接口,从而较好地解决了使用难度和访问速度这一对矛盾。

## 1 CDblink——对 DB_Library for C 函数的封装

CDblink 类需实现以下主要功能:

1) 登录 SQL Server,开辟结果缓冲区。

2) 发送 SQL 语句,返回结果记录集到结果缓冲区。

3) 取结果记录集的任意记录。

4) 关闭和 SQL Server 的连接。

以下对各项功能的实现加以详述:

### 1.1 登录 SQL Server,开辟结果缓冲区——CDblink::Logtosvr()

DB_Library 函数通过 DBPROCESS 结构实现和 SQL SERVER 的通信。DBPROCESS 结构作为应用程序和 SQL Server 的连接中介,在其中记录了该应用某次访问的登录用户名、口令、服务器名、应用名称等重要信息,绝大多数的 DB_Library 函数需要以该结构作为第一个调用参量。一个应用程序可以以多个 DBPROCESS 同时对 SQL Server 进行访问,相互之间不受干扰。对于 DB_Library 来说,登录 SQL Server,就是用正确的登录信息填写 DBPROCESS 结构。为满足能够任意访问查询结果集中某一记录的需要,logtosvr() 成员函数在实现上述功能的同时,也确定了结果缓冲区的大小(如不开辟结果缓冲区,只能每次从服务器上读取 1 行数据,处理完必之后再读入下一行,这样上一行数据被覆盖,造成使用不便;开辟了一定大小的结果缓冲区后,就可一次性读入大小等于缓冲区大小的结果集存入其中,用 dbgetrow() 函数任意定位读取记录)。程序代码如下(注:以下

* 收稿日期:2004-07-10

所用的 DB_library 函数均以'db'打头):

```
//以合法的用户身份登陆 SQL Server
bool CDblink::Logtosvr()
{
/*获取 login 结构,该结构作为 dbopen()(打开数据库,创建一个新的 DBPROECSS 结构)的参量. */
PLOGINREC login = dblogin ();
/*用用户输入的登录信息:登录用户名(loginfo. User)、口令(loginfo. PassWord)、登录应用名
(loginfo. AppName)填充 login 结构*/
DBSETLUSER (login, loginfo. User);
DBSETLPWD (login, loginfo. PassWord);
DBSETLAPP (login, loginfo. AppName);
//设置登录延时
dbsetlogintime (2);
// 登录数据库,得到 dbproc(一个指向新建 DBPROCESS 结构的指针)
dbproc = dbopen (login, loginfo. Server);
if(! dbproc)
{
SendMsg("登录 SQLSERVER 失败");
return false;
}
//把结果缓冲区设为 MaxLineNum 行
char buflen[10];
wsprintf(buflen,"%d",MaxLineNum+1);
dbsetopt(dbproc, DBBUFFER, buflen);
return true;
}
```

### 1.2 发送 SQL 语句,返回结果集——CDblink::ExecSQL()

登录 SQL Server 之后,就可以向 SQL Server 发送 SQL 语句,包括 select ,insert,update,delete 等。DB_Library 也允许用户输入以'Exec'打头的 SQL 语句,执行存于服务器上的一个存储过程,这样就极大地扩展了 DB_Library 的功能。DB_Library 将用户的 SQL 语句存入命令缓冲区 ,用户 1 次可发送数个 SQL 语句,称为 1 个"命令批次"(command batch)。缺省状态下,用户在执行了 1 个命令批次后 DB_LIB 将清除命令缓冲区中的已有内容,并提供函数供用户返回命令批次中每个查询的结果集。由于在实际应用中有时并不知道本次查询将有几个 SQL 语句(例如在执行 1 个存储过程时),为方便起见,作者设计的 ExecSQL()成员函数只取最后一个查询的结果集(这种方法能够满足绝大多数情况下的查询需求),实现的方法就是遍历每一个 SQL 语句的结果集,将结果取出到结果缓冲区直到最后一个,由于取下 1 个 SQL 语句的结果集都将清除上一个的结果集,结果缓冲区中最终保留的将是最后一个 SQL 语句的结果集,程序如下:

```
/*向服务器发送命令,lpszStatament 参量是用户输入的 SQL 语句。*/
bool CDblink::ExecSQL(LPCTSTR lpszStatement)
{
//查询返回值
RETCODE result_code,retcode;
//清除命令缓冲区
dbfreebuf (dbproc);
//向 dbproc 添加 SQL 串,用户 1 次可添加多个查询语句
if(dbcmd(dbproc, lpszStatement) ! = SUCCEED)
return false;
//DB_LIB 执行该 SQL 语句
if(dbsqlexec(dbproc) ! = SUCCEED)
return false;
//从服务器取数据到结果缓冲区
m_Rows = 0;
loopID = FALSE;
/*遍历每个 SQL 语句的结果集直到最后一个(NO_MORE_RESULTS 代表已无结果集可取)*/
while((result_code = dbresults(dbproc)) ! = NO_MORE_RESULTS)
{
//取某个结果集操作成功
if (result_code = = SUCCEED)
{
m_Rows = 0; //记录某一结果集的行数
//遍历结果集中的每一行
while ((retcode = dbnextrow(dbproc))! = NO_MORE_ROWS)
{
m_Rows + +;
/*省略 MaxLineNum 行以上的行(MaxLineNum 代表缓冲区的大小),如果缓冲区已填满,DB_LIB 将不再读取该结果集的下一行,因此需清除缓冲区的一行,读入下一行,直到读完。*/
if(m_Rows > MaxLineNum)
{dbclrbuf(dbproc, 1);m_Rows - -;}
}
}
else break;
}
/*读入当前缓冲区的列数(也就是当前查询表的列数),m_Cols 是 CDblink 类的成员*/
m_Cols = dbnumcols(dbproc);
return true;
}
```

### 1.3 取结果缓冲区的任意记录——CDblink::GetResultRow(int num)

当结果集从服务器取到结果缓冲区之后,就可以取结果缓冲区的任意 1 条记录。为此,专门定义了 1 个 RESULT 结构,为 CDblink 添加成员变量 RESULT result[100], 用来存储返回记录的每列的数据指针和列的数据长度(result 数组开 100 大小对绝大多数表已足够)。这样在应用时就可通过强制数据类型转换,将其转化为 VC 能够接受的数据类型。RESULT 结构如下:

```
typedef struct
{
```

```
LPBYTE p; //列数据指针
int n; //列数据长度
} RESULT;
程序实现如下:
//取第 num 行数据
RESULT * CDblink::GetResultRow(int num)
{
//如果 dbproc 非法,拒绝执行该操作
if(! dbproc) {execID = false;return NULL;}
//如果索取行 <0 行或超出返回的结果集的行总数,拒绝执行该操作。
if(num <0 || num > m_Rows) return NULL;
//返回的是正常行(非统计行(compute row)
if(dbgetrow(dbproc, num) == REG_ROW)
{
//将该行的每一列的数据指针、长度信息存入对应的 result 数组的对应项中。
for(int i=0; i< m_Cols ; i++)
{
//第 i+1 列的数据指针
result[i].p = dbdata(dbproc, i+1);
//第 i+1 列的数据的长度
result[i].n = dbdatlen(dbproc, i+1);
}
return result;
}
else
{
return NULL;
}
}
```

**1.4 关闭和 SQL Server 的连接——CDblink::~CDblink()**

```
CDblink::~CDblink()
{
//关闭与 SQL Server 的连接。
dbclose(dbproc);
dbwinexit();
}
```

## 2 实例

利用 CDblink 类,可以方便地实现对 SQL Server 的访问。举一应用实例:

1) CDblink * SqlDB = new CDblink;

2) SqlDB→ExecSQL("select * from ServerName. dbo. TableName");

3) RESULT * res = SqlDB⇔GetResultRow(1);

4) 读取 res 中的每一列,进行强制类型转换。如第一列在 SQL Server 为整型,长度为 4 个字节,则对应 VC 的数据类型也为整型,转换方式为 int VC_var = *(int *)res[1].p,余者类推。

5) delete SqlDB。

## 3 总结

CDblink 类实现了对 DB_LIB 部分重要函数的封装,将其和服务器端存储过程协调使用,能够满足绝大多数情况下客户端对服务器端的访问需求,且使用简单灵活,方便快捷,优点是明显的;然而对于其中如结果缓冲区大小限制、只能对正常行进行操作等不足之处,则需在今后的科研实践工作中进一步加以改进。

## [参 考 文 献]


[1] 陈述彭,鲁学军,周成虎. 地理信息系统导论[M]. 北京:科学出版社,2000.

[2] 付继彬,范群波,刘晓亮. Oracle8 入门与提高[M]. 北京:清华大学出版社,2001.


(上接第 7 页)

```
    Numpoints As Long
    Numpoints As Long
End Type          //面文件记录信息的定义
```

对二进制文件的解译需要很大的耐心。字节的顺序位置一定要清晰,逐个字节的进行读取、解译。至此,将解译的信息记录作为书写 mif 文件的数据源。

.shp 文件的解译是整个程序的主体,现仅对其进行描述。.shx,.dbf 解译的原理与其大致相同。在此不再赘述。

2.3.2 写 mif 文件

写 mif 文件和 mid 文件属于对文本文件的操作,相对比较简单,在此不再赘述。

## 3 结束语

关于异构数据模型数据之间转换和互操作是当今 GIS 领域普遍关注的问题,此问题的解决办法也出现很多种,如直接读取转换,中间数据格式转换,语义转换等。加拿大的 FME 数据通用转换器基于语义层次的转换,中国超图的基于直接读取的转换都是很好的例子。本数据转换模块是在前者的先进经验的启发下,结合共享平台的具体应用进行组织开发的。

## [参 考 文 献]


[1] 陈常松. 地理信息共享的理论与政策研究[M]. 北京:科学出版社,2003.

[2] 周晓萍,赵宇红. Visual FoxPro 数据库文件的结构分析与实用方法[J]. 南华大学学报(理工版),2001,(2).

[3] 王晓武,等. MapBasic 程序设计[M]. 北京:电子工业出版社,2000.

[4] 华一新. 地理信息系统原理与技术[M]. 北京:解放军出版社,2001.


**作者简介**:刘雪凯(1978~),男,河北省徐水县人,硕士研究生。主要从事 GIS、测绘及其在土地管理中的应用等方面的研究工作。