

基于 .NET 的 BLOB 数据存储加密技术

杨 梅^{1,2}, 郑 晗¹, 胡 斌¹

(1 西安交通大学 电子与信息工程学院, 陕西 西安 710049; 2 西北核技术研究所, 陕西 西安 710024)

摘 要: 为了解决信息系统开发中非结构化数据存储访问与加密的难题, 将这类数据以二进制对象字段 (BLOB) 存储在数据库中, 并以二进制数据流方式实现数据的加密、传输、存储与访问, 从而达到快速安全访问此类数据的目的。本文详细介绍了基于 ADO 数据访问技术操作 BLOB 数据的原理与方法, 并在此基础上给出了基于 VC++、.NET 和 ADO 数据访问技术实例应用和部分程序代码。

关键词: 二进制大对象字段; 编程模型; 结构化数据; 非结构化数据

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2013)04-0058-05

Technology of Encrypting and Storing BLOB Data Based on .NET

YANG Mei^{1,2}, ZHENG Han¹, HU Bin²

(1 The School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China;

2 Northwest Institute of Nuclear Technology, Xi'an Shanxi 710024, China)

Abstract: Managing the unstructured large object data item is a conundrum in programming management information systems. In order to encrypt, transfers, store and access the data fast and safely, the paper proposed a new spatial method which stored the data in database by BLOB (Binary Large Object), encrypted, transferred, and operated it by binary data stream. The article introduces in detail of ADO's technology structure, model and working process. The article also describes the method of storing and accessing BLOB data, gives out codes of the programs detailedly based on VC++、.NET and ADO technology.

Key words: BLOB; program model; structured data; unstructured data

1 引言

在信息系统开发中, 需要处理大量类型复杂的非结构化数据, 此类数据在数据库中以二进制对象字段 (Binary Large Object, BLOB) 存储^[1]。与结构化数据相比, 对此类数据的加密以及存储访问需要做特殊的处理。目前, 这类数据的加密及存储管理方法, 许多数据库管理系统如 Oracle 和 SQL Server 均在数据库存储中给出了不同的解决方案, 但如何在程序开发中组织调用此类数据, 不同的开发平台有不同数据访问方法和实现细节。本文介绍的是基于 VC++ 2008 语言和 ADO 数据访问技术对 BLOB 数据的加密传输及存储访问的方法。

2 ADO 数据访问技术

UDA (Universal Data Access, 一致数据访问技术) 是微软推出实现数据访问的一种标准, 包括两个层次的接口: 系统层编程接口 OLE DB (以 COM 组件的形式设计) 及应用层编程接口 ADO (ActiveX Data Objects, ActiveX 数据对象)^[2]。通过此技术能实现不用考虑数据存储地点的数据访问。

2.1 ADO 编程模型

ADO 的目标是访问、编辑和更新数据源, 其编程模型包括的类和对象提供完成该目标系列动作。ADO 编程模型中的关键元素包括:

① Connection 连接数据源, 可开始一个事务;

收稿日期: 2012-07-27; 修回日期: 2012-08-22

基金项目: 国防预研基金项目 (18051001)

- ② Command 通过指定的连接对数据源操作;
- ③ Parameter 指定列、表、值的变量参数;
- ④ Recordset 数据定位,检查、改变行中的数据;
- ⑤ Field 修改 Recordset 对象列中的数据;
- ⑥ Error 产生连接、执行或设置对象时的错误对象;
- ⑦ Property 描述或控制对象的行为;
- ⑧ Stream 提供读写连续字节和一行文本的方法;
- ⑨ Collection 将特定类型的对象包容在集合中;
- ⑩ Event 在某种操作要发生或发生后发出的通知。

2.2 ADO 读写 BLOB 数据技术

从技术实现上,处理非结构化数据有两种方式。一是将数据以文件方式存储在文件系统中,将指向文件的链接或路径存储在数据库表中,这种方式读写数据速度快,但数据管理不方便,并需要额外考虑事务处理一致性和数据的安全性;二是将数据以二进制对象字段(Binary Large Object, BLOB)存储在数据库中^[3],此方式可以充分利用数据库管理系统的事务、管理和安全特性。

BLOB 数据存储在数据库大型对象数据类型字段中时,对此类数据存取时不能与普通数据同时进行,必须将此字段分离出来单独处理,图1为将数据文件写入数据表的 BLOB 字段以及从 BLOB 字段读取数据并还原成数据文件的过程。

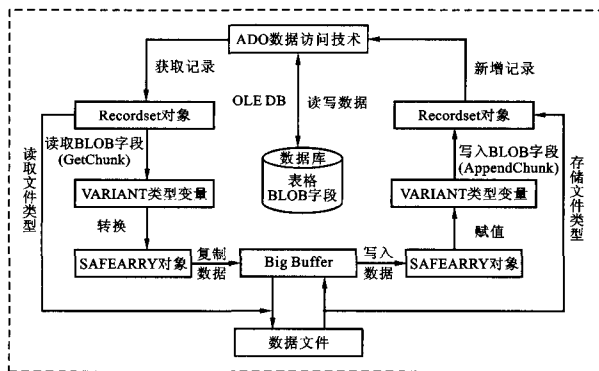


图1 数据库存取非结构化数据过程

可以分别使用 ADO 的 Fields 对象中的 Append Chunk 和 GetChunk 方法写入和读取 BLOB 类型的数据^[4],原型如下:

```
_variant_t GetChunk(longLength);
HRESULT AppendChunk(const _variant_t
```

```
&data);
```

用 AppendChunk 方法写 BLOB 数据的关键问题是把二进制数据赋值给 VARIANT 类型的变量;读数据时使用 GetChunk 获得数据长度,进而返回包含数据的 VARIANT 的类型变量,然后可以利用 SafeArrayAccessData 函数得到 VARIANT 变量中指向数据的 char * 类型的指针,进而将二进制数据还原成存储时的数据文件。

代码1 存储 BLOB 数据过程

```
m_filelen = file.GetLength(); //获得文件长度
m_pBuffer = new char[m_filelen + 1]; //定义缓冲区大小
rgsabound[0].lLbound = 0; //数组元素的起始计数为0
rgsabound[0].cElements = m_filelen; //数组元素个数为文件长度
psa = SafeArrayCreate(VT_UI1, 1, rgsabound); //创建 SAFE ARRAY 对象
//将 m_buffer 指向的二进制数据保存到 SAFEARRAY 对象 pas 中
for(long i=0; i<(long)m_filelen; i++)
    SafeArrayPutElement(psa, &i, m_pBuffer++); //第 i 个元素赋值
varblob.vt = VT_ARRAY|VT_UI1; //将 varblob 的类型设置为 BYTE 类型数组,VT_ARRAY 为指向 SAFEARRAY 的指针,VT_UI1 指示 BYTE
varblob.parray = psa; //为 varblob 变量赋值
m_pRecordset->AddNew(); //增加一条记录
m_pRecordset->GetFields()->GetItem("Book")->AppendChunk(varblob); //将 BLOB 数据写入数据库字段
m_pRecordset->Update(); //更新记录数据
```

代码2 读取 BLOB 数据过程

```
long lDataSize = m_pRecordset->GetFields()->GetItem("Book")->ActualSize; //获得二进制数据长度
_variant_t varBLOB = m_pRecordset->GetFields()->GetItem("BaseModel")->GetChunk(lDataSize); //从字段读取数据到 varBLOB 中
char * m_pBuffer; //定义缓冲变量
if(m_pBuffer = new char[lDataSize+1]) //分配空间
{
    char * pBuf = NULL;
    SafeArrayAccessData(varBLOB.parray, (void **)&pBuf); //得到指向数据的指针
    memcpy(m_pBuffer, pBuf, lDataSize); //复制数据
    SafeArrayUnaccessData(varBLOB.parray);
} //释放数据
```

```
CString strPath=filePath.GetPathName(); //文件路径名称
```

```
CFile file(strPath, CFile::modeCreate | CFile::modeWrite); //写文件
```

```
file.Write(m_pBuffer, lDataSize); //将缓冲区数据写入文件
```

3 非结构数据加密/解密

对数据库数据加密的目的是保护数据表中数据不被非法访问和更新^[5],并防止数据的泄露和丢失,使得数据以密文方式存储,确保数据库的安全。

3.1 数据库加密技术

从加密方式来分数据库加密包括:操作系统层、数据管理系统内核层、数据管理系统外层,其中前两种方式需要修改数据管理系统内核,操作难度大。从加密粒度来分包括:数据库级、表级、记录级、字段级、数据项级。从加密算法来分包括:对称加密算法、非对称加密算法、混合加密算法。通过对加密方式、加密粒度、加密算法分析,在对信息系统中的非结构化数据进行加密时,加密方式一般采用数据管理系统外部加密,加密粒度选择数据项级,加密技术使用对称加密算法。

3.2 非结构化数据加密

与普通结构化数据相比较,非结构化数据由于在存储时需要分离出来单独处理,所以对数据的加密、解密同样需要进行特殊处理,图2为非结构化数据加密、解密过程。

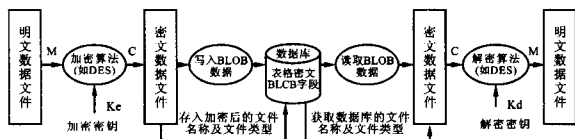


图2 为非结构化数据加密、解密过程

从图2可以看出,非结构化数据加密与结构化数据加密相比,由于非结构化数据的存储以文件方式进行,所以加密数据也是以文件的方式进行,数据的存储访问操作过程为:明文数据文件—密文数据文件—BLOB数据存储到数据库—读取BLOB数据—密文数据文件—明文数据文件。

3.3 DES 加密算法

DES (Data Encryption Standard, 数据加密标准算法) 是一种对二元数字分组加密的分组密码算法^[6],其分组长度为 64 比特(bit),密钥长度为 64bit。DES 是一个对称算法,算法经过一系列的步骤将 64 bit 明文变换成 64bit 密文,解密过程中使用同样的步骤和密钥。DES 综合应用了置换、代替、移

位多种密码技术,是一种乘积密码。DES 在算法上采用迭代机制,结构紧凑,条理清楚,而且算法为对合运算,便于实现,图3为加密算法的迭代变换过程。

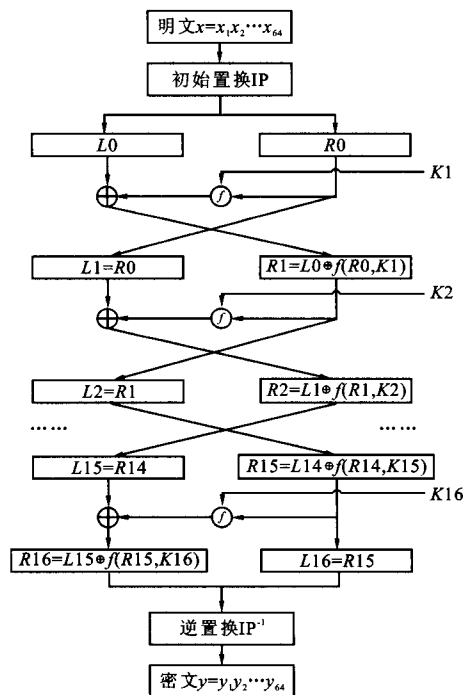


图3 DES 加密算法迭代过程

DES 是一种比较成熟的加密算法,具有加解密速度快、操作简单的优点,适合大量数据的加密和解密。DES 算法已经有了相当成熟的程序实现方法,可以直接设计加密函数对数据文件进行加密。

代码3 数据加密/解密与数据存储过程

```
bool DES::Des_Go(char * Out, char * In, long data-
len, const char * Key, int keylen, bool Type) //定义加密
函数
```

```
{ SetKey(Key, keylen);
if( ! Is3DES )
{ // 1 次 DES 加密
for(long i=0,j=dataLen>>3; i<j; ++i, Out += 8,
In += 8)
SDes(Out, In, &SubKey[0], Type);
}
Else
{ // 3 次 DES 加密:加(key0)—解(key1)—加(key0);
解密:解(key0)—加(key1)—解(key0)
for(long i=0,j=dataLen>>3; i<j; ++i, Out += 8,
In += 8) {
SDes(Out, In, &SubKey[0], Type);
SDes(Out, Out, &SubKey[1], ! Type);
SDes(Out, Out, &SubKey[0], Type); }
```

```

    }
    return true;
}
DES cEncrypt,cDecrypt; //定义加密/解密函数类实例
CFile ff(FileNamePath,CFile::modeRead); //打开文件
ff.Read(inbuff,8); //读明文文件
cEncrypt.Des_Go(oubuff,inbuff,sizeof(inbuff),skey,sizeof(skey),ENCRYPT); //加密文件
fp.Write(oubuff,8); //写密文文件
//将密文文件赋值给BLOB变量,见上文BLOB数据的存取
m_pRecordsetUp->GetFields()->GetItem("Book")->AppendChunk(varblob); //将密文数据写入数据库

```

4 实例应用

4.1 创建MFC应用程序并建立后台数据库

首先创建一个基于对话框的MFC应用程序,为了编程方便在项目属性中将字符集设定为使用多字节字符集。然后在Microsoft SQL Server 2008数据库管理系统中建立数据库,名称为DOOKDB,在该数据库下建一个名称为Books的数据表格,表格字段如表1所示。

表1 Books表格详细信息

字段名称	字段类型	说明
BookID	Int	图书ID,自动产生
BookName	Varchar	图书名称
BookType	Varchar	图书类型
Book	Varbinary	图书
Date	Date	时间
Memo	Varchar	备注

建立一个用于数据库连接的VC++普通类,类的名称为CADOConn。默认情况下,Visual C++不支持ADO对象的直接使用,使用时需要将ADO库文件导入到工程中^[7]。方法为:在CADOConn类的头文件中添加代码: #import "c:\ProgramFiles\Microsoft SQL Server\80\Tools\Binn\sqlcmd.exe" no_namespace rename("EOF","adoEOF") rename("BOF","adoboF")。其中参数no_namespace表示不使用ADO名字空间。为了避免出现常量名冲突,将EOF更改为adoEOF,将BOF更改为adoboF。现在就可以使用ADO访问数据库了。

4.2 建立与数据库连接

在做好了以上准备后,下一步可以建立与数据库的连接了。

代码4 连接数据库

```

_ConnectionPtr m_pConnection; //定义指向连接对象指针
_RecordsetPtr m_pRecordset; //定义指向记录对象指针
_bstr_t ADOConnStr; //定义连接字符串,BSTR是ADO专有结构化数据类型,包括串及串的长度
m_pConnection.CreateInstance(_T("ADODB.Connection")); //创建对象
ADOConnStr="Driver=SQLServer;Server=192.168.1.156;Database=AGMDB;UID=sa;PWD=123456"; //定义连接字符串
m_pConnection->Open(ConnStr,"", "", adModeUnknown); //连数据库
return m_pRecordset; //返回记录集

```

4.3 操作数据库

在建立好数据库的连接后,为方便地操作数据库及表格中的数据,需要为数据库的表格创建一个类,类的成员变量对应表的列(字段),成员函数是对成员变量和记录的操作。

代码5 为数据表Books建立一个独立的VC++普通类CBookTable,CBookTable类的成员变量为Book表的列,成员函数为基本数据库操作函数。

```

Long BookID; //图书ID
CString BookName; //图书名称
CString BookType; //图书类型
CString Date; //建立时间
CString Memo; //备注
//以下定义数据库操作
int Books_HaveName(CString cBooksName); //检查重复性
long Books_SQL_Insert(VARIANT cBook); //插入记录
void Books_SQL_Update(CString cBooksID, VARIANT cBook); //更新数据
void Books_SQL_Delete(CString cBooksID); //删除数据
void Books_GetData(CString cBooksID); //读取记录值

```

在建立了数据库的连接,编写了数据库操作通用函数后,下一步可在控件响应函数和对话框中调用和显示数据库中的数据了,其中BLOB数据的加密、解密和存储访问使用代码1-3中的方法处理。至此,本节从工程实现的角度为操作数据库中的数据进行了技术和步骤上的准备,在实际应用中用户

可以根据系统实际需要灵活组合、动态调用,从而达到安全有效管理数据的目的.

5 结束语

本文介绍了在 .NET 框架下,利用 VC++ 2008 和 ADO 数据访问技术实现对 SQL Server LOB 数据的加密及存取访问方法,此种方法可存储访问任何类型的非结构化数据.在程序设计时将每个数据库表的字段和操作封装到类中,使应用程序可以共享对表的操作,无需重复编码.这样既可以大大提高程序开发效率,增加代码的可重用性,又可以规范程序代码,增加代码的可读性,使程序易于维护,从而将面向对象的程序设计思想应用在系统开发过程中.

参考文献:

- [1] Mickey Williams. Visual C++ .NET 技术内幕[M]. 北京:清华大学出版社,2007:453-456.
- [2] 启明工作室. Visual C++ + SQL Server 数据库应用实例完全解析[M]. 北京:人民邮电出版社,2007.

- [3] 王 姝. 数据库图像字段数据存储与读取技术应用研究[J]. 科学技术与工程,2010(1):682-685.
- [4] 杨勇. 数据库系统中 BLOB 对象的管理[J]. 微电子学与计算机,2006,23(7):148-150.
- [5] 昕晨,DES[CP/OL]. <http://www.hur.cn/soft/2010/8049.html/detail118292.html>.
- [6] 赵卓,刘博,厉京运. 基于 DBMS 外层的数据库加密系统研究与设计[J]. 计算机工程与设计,2008(12):63-66.
- [7] 王井阳,张晓明,秦敏,等. 在 VC 中利用 ADO 技术操作数据库中的 BLOB 数据[J]. 计算机应用研究,2004(10):79-81.

作者简介:



杨 梅 女,(1976-),硕士,工程师.研究方向为数据库应用开发.

(上接第 57 页)

- [9] Gang Zhang, Zeng shantian. A chaotic secure communication scheme based on logistic map[C]//International Conference on Computer Application and System Modeling. Chongqing, China, 2010:589-591.
- [10] National institute of standards and technology. FIPS 140-2-2001 security requirements for cryptographic modules [S]. Washington DC: American National Standards Institute, 2001.

作者简介:

樊 勇 男,(1987-),硕士研究生.研究方向为信息安全、密码系统.

顾 剑 男,(1958-),硕士,教授.研究方向为信息安全系统.

杜文才 男,(1953-),博士.教授,博士生导师.研究方向为计算机网络通信、通信安全.