基于嵌入式系统的多模式数据访问机制①

Multi-Mode Data Access Based on Embedded System

张 烨[1] 张跃鹏[2] 柯希林[2] (1.西安理工大学 自动化与信息工程学院 陕西 西安 710048;
2.西安测绘研究所 陕西 西安 710054)

摘 要: 为了在嵌入式应用系统开发中实现多语言混合编程数据访问，比较了几种不同操作系统中数据库的访问模式，提出了在.NET Compact Framework 2.0 下 C#和 C++语言对 SQL Server CE 数据库的访问机制，特别提出了对数据库中 BLOB 数据类型的操作方法，并给出了这些方法的具体实现。该机制已经应用于实际嵌入式系统中，运行效果良好。

关键词: 嵌入式系统 多模式 数据访问

1 引言

目前嵌入式系统的应用日益广泛，尤其在 GIS(地理信息系统)方面的开发应用，由于 GIS 的数据量大，因此其应用开发与主要依靠数据库的支持。在早期使用很广的 Win CE 操作系统中，用 EVC、EVB 语言能访问嵌入式版本的 Access 数据库 Pocket Access，但数据库本身的功能较弱。目前微软推出了另一个精简数据库产品 SQL Server 2005 Compact Edition (缩写为 SQL Server CE),保留了桌面上用 SQL Server 的很多功能，大大加强了数据库自身的能力[1]。

Windows Mobile 是微软为智能移动终端设备使用的新一代操作系统，Windows Mobile 将用户熟悉的桌面 Windows 扩展到了移动设备上，是 Win CE 的升级版本。目前最新的是 Windows Mobile 6 (WM6)，其在 ROM 中内置了.NET Compact Framework 2.0 (.Net Framework 的精简版本)，从而提高了托管应用程序的性能[2]。

由于嵌入式系统受到操作系统、数据库系统及开发语言的限制，不如在 PC 机上开发灵活方便。同时新的嵌入式系统和数据库,对原有的 C++程序往往不能很好的兼容，在.NET Compact Framework 下不支持 C++通过 ADO 技术访问数据库。目前两种开发语言和相应访问技术的对比如表 1。

表 1 开发语言和访问技术对比

OS	DB	语言	访问技术
Win CE	Pocket Access	Evc	ADOCE
Win Mobile	SQL Server CE	C#	ADO.net
		C++	OLE DB

本文将介绍在 Windows Mobile 6 中，采用 C#和 C++语言实现对一个 SQL Server CE 数据库的不同访问机制，特别说明了 BLOB 数据类型访问的实现过程。

2 开发环境建立

目前在 Windows Mobile 6 版本，已经内置了 SQL Server CE。这样我们在开发程序后，不需要再考虑将 SQL Server CE 部署到目标设备上。

SQL Server CE 与 SQL Server Management Studio 和 Visual Studio 2005 完全集成。通过 SQL Server Management Studio，可以用可视化的方式来创建 SQL Server CE 数据库、查看数据库对象、创建表、修改表以及执行交互式查询等操作[4]。

现在.NET Compact Framework 只支持 C#和 VB.NET[3]，而不支持 C++。但是在 Visual Studio 2005 中发现能够用 C++来开发本地代码的移动设备

① 基金项目:国家自然科学基金(60873035)
收稿时间:2009-03-11

应用程序。同时 SQL Server CE 数据库为托管应用程序提供了一个 ADO.NET 库，并为本地应用程序提供了一个 OLE DB 库。

因此，在 PC 机上建立的开发环境为：安装 Visual Studio 2005+Windows Mobile 6 模拟器+ SQL Server CE。

3 建数据库表

存储数据的 SQL Server CE 数据库是以.sdf 为文件后缀名。可以用上述的可视化工具创建一个数据库“commdb.sdf”，再建立一个数据库表 fileinfo，字段的简单定义如表 2。

表 2 字段定义

字段名	数据类型
FileID	bigint
FileName	nvarchar(20)
FileLen	int
FileContent	Image

不同的数据库对 BLOB 类型有不同的关键字定义。SQL Server 中定义为 Image 数据类型；在 Access 数据库中定义为 OLE 对象类型；SQL Server CE 中是 Image 或 varBinary 类型。这里的 FileContent 为一个 Image 类型的字段。

4 C#的访问方法

由于.NET Compact Framework 支持 C#语言的开发，SQL Server CE 数据库又为托管应用程序提供了一个 ADO.NET 库，是对 OLE DB 的接口作了封装，使程序开发得到简化，ADO.NET 技术属于数据库访问的高层接口[5]。

要采用 ADO.NET 库，就必须使用一个新的命名空间——System.Data.SqlServerCe。此命名空间在建立工程时不是自动存在的，这就需要手动添加对 SqlServerCe.dll 文件的引用。

4.1 数据库的连接

数据源要写数据库的绝对路径，采用 SqlCeCon－nection 类，代码如下：

```
// 数据库存放的路径
str="Data Source=Storage Card\\commdb.sdf;";
// 实例化连接对象
SqlCeConnection conn = new SqlCeConnection(str);
Try
{
conn.Open();  // 连接数据库
}
catch (System.Exception e)
{
MessageBox.Show("打开数据库失败);
}
```

4.2 数据的保存

采用 sqlDataAdapter 类的更新数据集的方法增加记录，下面是增加的一个简单例子，记录的内容在程序中直接给定。

可以看到各类型的数据可以采用同一种形式赋给字段，只是二进制数据要先保存在 byte 类型的数组中，而赋给 BLOB 类型字段的是 byte 型的指针。

```
// 在数据库中增加一条记录，包括二进制数据
byte[] pcont = new byte[4];
pcont[0] = 65;
pcont[1] = 66;
// 建立数据适配对象，构造 sql 语句
SqlCeDataAdapter sqlDataAdapter = new SqlCeDataAdapter("SELECT * FROM fileinfo", conn);
// 建立命令字，执行 sql 语句
SqlCeCommandBuilder sqlCommandBuilder = new SqlCeCommandBuilder(sqlDataAdapter);
// 构造数据集，保存记录值
DataSet dataSet = new DataSet("fileinfo");
sqlDataAdapter.Fill(dataSet, "fileinfo");
// 数据集增加一行，每个字段填值
System.Data.DataRow dataRow;
dataRow=dataSet.Tables["fileinfo"].NewRow();
```

```
    dataRow["FileID"] = 1;
    dataRow["FileName"] = "test.txt";
    dataRow["FileLen"] = 2;
    dataRow["FileContent"] = pcont;
    // 增加一行，将数据集更新到数据库表中
    dataSet.Tables["fileinfo"].Rows.Add(dataRo
w);
sqlDataAdapter.Update(dataSet, "fileinfo");
```

4.3 数据的查询

数据检索利用 SqlCeCommand 类对象执行查询 SQL，如需要更新等其它操作，应采用可绑定游标的 SqlCeResultSet 类对象保存记录结果。如只是要查询结果，用 SqlCeDataReader 类对象得到记录集，再用 dataread["字段名"]就可以得到一条记录的各数据，但应注意各字段的数据类型，需要用强制转换为相应类型。

```
    // 建立命令字，执行 sql 语句
    SqlCeCommand command;
    command = conn.CreateCommand();
    command.CommandText = "SELECT * FROM
fileinfo";
    // 执行 sql 语句的结果保存到数据集中
    SqlCeDataReader dataread = command.
ExecuteReader();
    while (dataread.Read())  // 下一个还有记录
    {
    long id = (long)dataread["FileID"];
    user[idx] = (string)dataread["FileName"];
    }
    dataread.Close();
```

对文件内容的 BLOB 类型的读取有所不同，需要先得到 BLOB 字段的索引，以及内容的长度，用 GetBytes()函数读到一个 Buffer 中，部分实现的代码如下：

```
    // 得到 BLOB 类型的字段序号
    colidx=dataread.GetOrdinal("FileContent");
    // 如二进制内容不为空
    if (dataread.IsDBNull(colidx) == false)
    {
    // 得到内容的长度
    int len = (int)dataread["FileLen"];
    pBuff = new byte[len];
    // 从数据库中得到二进制的内容    dataread.
GetBytes(colidx, 0, pBuff, 0, len);
}
```

5 C++的访问方法

由于.NET Compact Framework 不支持 C++通过 ADO.net 技术访问数据库的方法，因此就不能使用 C#中的相关类，而要通过底层的接口 OLE DB 方法来访问 SQL Server CE，则在程序中必须包含“ssceoledb30.h”头文件。使用 OLE DB 来实现数据库应用程序相对复杂一些，需要较多的代码。

5.1 数据库连接

首先创建一个 CLSID_SQLSERVERCE_3_0 的 OLE DB 提供者的实例，初始化一个接口指针，再通过设置属性 DBPROPSET 结构和数据库“Storage Card\\commdb.sdf”联系起来，最后创建一个数据库访问的会话指针。核心的语句如下：

```
    // 提供者的接口指针
    IDBInitialize   *pIDBInitialize = NULL;
    // 创建一个 OLE DB 提供者的实例
    hr =CoCreateInstance(CLSID_SQLSERVERCE_
3_0, 0, CLSCTX_INPROC_SERVER,
    IID_IDBInitialize,
    (void**)&pIDBInitialize);
                ⋮
    // 根据 DBPROPSET 属性，初始化数据库源
pIDBInitialize->Initialize();
    pIDBInitialize->QueryInterface(IID_IDBCreat
eSession, (void**)&m_pIDBCreateSession);
```

5.2 常规数据的操作

利用前面初始化访问会话指针，初始化命令指针 pIDBCrtCmd，再创建一个带参数设置的命令，SQL 语句的执行就通过设置命令的文本参数，pwszSQL 为要执行的 SQL 语句。

这样对数据库的查询、增加、更新、删除操作的用法都一样，只要写对相应的 SQL 语句。执行 SQL 后得到的查询结果数据在 pIRowset 指针中，影响的记录数在 cRowsAffected 变量中。核心的语句如下：

```
// 创建一个会话实例
m_pIDBCreateSession->CreateSession(NULL, IID_IDBCreateCommand,
(IUnknown**) &pIDBCrtCmd);
// 创建一个用参数的命令
pIDBCrtCmd->CreateCommand(NULL, IID_ICommandWithParameters,
(IUnknown**) &pICmdWParams);
// 参数是字符串型
pICmdWParams->QueryInterface(IID_ICommandText, (void**) &pICmdText);
// 绑定 pwszSQL 的 sql 语句
pICmdText->SetCommandText(DBGUID_SQL, pwszSQL);
// 执行 SQL 语句
pICmdText->Execute(NULL,IID_IRowset,NULL,&cRowsAffected,(IUnknown**)&pIRowset);
```

5.3 BLOB 数据的保存

BLOB 类型的数据无法用上述普通的方式进行存储，实现相对复杂，主要是先得到要写入的记录行，再使用 ISequentialStream 类的 Write 方法。大致流程如图 1。确定记录行有两种情况：一种是按条件查询到已存在的要更新的记录；另一种是插入新记录，先插入非 BLOB 类型的数据，再写入 BLOB 类型的数据。

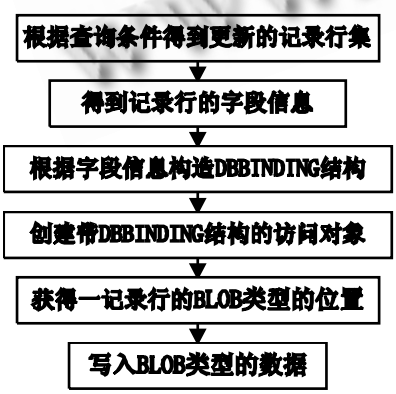

图 1　写入 BLOB 类型数据的流程图

下面给出主要代码：

```
// 打开表，得到行集
pIOpenRowset->OpenRowset( NULL,&TableID, NULL,
IID_IRowsetChange,
sizeof(rowsetpropset)/sizeof(rowsetpropset[0]), rowsetpropset,
(IUnknown**) &pIRowsetChange );
// 得到字段信息
pIRowsetChange->QueryInterface(IID_IRowset, (void**) &pIRowset);
pIRowset->QueryInterface(IID_IColumnsInfo, (void **)&pIColumnsInfo);
pIColumnsInfo->GetColumnInfo(&ulNumCols, &pDBColumnInfo, &pStringsBuffer);
// 构造结构，并赋值
DBBINDING *prgBinding;
// BLOB 字段的位置，从 1 开始
prgBinding[dwIndex].iOrdinal = dwOrdinal;
// 创建访问对象 hAccessor
pIAccessor->CreateAccessor(DBACCESSOR_ROWDATA,
dwBindingSize, prgBinding,
0, &hAccessor, NULL);
// 获取行数据及 BLOB 字段位置
pIRowset->GetNextRows(DB_NULL_HCHAPTER, 0, 1, &cRowsObtained, &prghRows);
pIRowset->GetData(prghRows[0], hAccessor, pData);
pISequentialStream=(*(ISequentialStream**)(pData + prgBinding[nConditionCols].obValue));
// 写入数据
ISequentialStream* pISequentialStream;
pISequentialStream->Write(pbBLOBData,
unBLOBSize, &dwWritten);
```

5.4 BLOB 数据的读取

实现方法和保存数据时相类似，前面步骤都一样，直到用 GetData()函数获得一行记录的数据，再依据

(下转第 227 页)

(上接第 200 页)

绑定的结构内容定位到 BLOB 类型的字段，调用 ISequentialStream 类的 Read 方法。主要代码如下：

```
// 得到一行记录数据，指针为 pRowValues
pIRowset->GetData(hRow[0], hAccessor, pRowValues);
// 存放 BLOB 数据的起始地址 pRealData
CHAR* pRealData = pRowValues + unOffset;
// 读取 BLOB 数据到 bBuffer
ISequentialStream*pStream=*((ISequentialStream**)pRealData);
pStream->Read((void*)bBuffer, cnBufferLen, &unRead);
```

6 结论

综上所述，在嵌入式系统开发中，不同语言对数据库的访问机制不同，尤其是 BLOB 类型的操作各有特点。因此，在开发一项新课题时，首先选用 C#语言来开发，程序实现相对简单。如果是原 C++程序需要移植到 WM6 上，或要调用 C++程序的动态库，那么 C++中原来访问数据库的程序就需要改写。总之，随着嵌入式系统地发展，结合.NET Compact Framework 的开发应用也将越来越多，有利于更简便地开发出更好的数据库应用程序。该方法已经实际应用于指挥系统中，效果良好。

参考文献

1 吴飞,王昕.嵌入式移动数据库 SQL Server for Windows CE 的应用研究.微计算机信息, 2006,22(6-2):122－124.

2 Lim EP, Siau K. Advances in Mobile Commerce Technologies. Idea Group Publishing,2003.

3 Srinivasa Sivakumar,Craig Morris,Peter Stanski.NET Compact Framework.Wrox Publisher, 2002.

4 Dove D. A Technical Comparison of Replication and Remote Data Access Features in SQL Server 2005 Mobile Edition3.0.http://www.mocrosoft.com

5 Boling D. Microsoft Windows CE 程序设计.北京:北京大学出版社, 1999.