ADO/OLE DB数据访问技术初探

姜恩波 (中国科学院成都文献情报中心 610041)

摘要：本文介绍了微软新近推出的通用数据访问模型UDA和它的两层标准数据接口OLE DB和ADO。通过介绍表明了基于组件对象模型(COM)的UDA技术是将来数据访问技术的发展方向。另外还通过与ODBC的比较和笔者的开发实例来说明这一点。

关键词：ADO OLE DB 数据库

1 概 述

通用数据访问技术(Universal Data Access简称UDA)是一种新的数据访问模型。它的核心作用是为不同的数据源提供通用的数据访问接口。这里不同的数据源，既包括最流行的关系数据库，也包括非关系数据库数据，如E-mail、电子表格、分布式文件和用户自定义格式数据等多种数据系统，这样就增加了程序的可移植性，也减少了编程代码的数量和难度。UDA模型是一个具有两层标准软件接口的数据访问模型，它以COM(组件对象模型)技术为基础。OLE DB是其底层接口，继承了COM的特性；ADO则是建立在OLE DB上的高层应用模型。它们的简单关系如图。

COM技术并不特指一种编程语言，它是一种编程规范，它具有面向对象的特性，支持封装、多态和继承性。因此一些遵守COM规范的软件模块可以在程序中进行重用、组合和相互作用。“COM运行不依赖机器环境，因此，它能够与任何一种支持COM对象的语言一起工作”。对象和接口是COM的组成结构。COM“对象”在意义基本等同于面向对象中的“对象”，即有自己的属性和方法。在应用中，COM对象是被封装起来的，用户不用了解其内容的构成和运行情况，对COM对象的调用是通过COM“接口”来实现的。COM接口可以理解为虚函数，虚函数中声明的方法可以理解为指针。应用程序调用接口中声明的方法，实际上指向了对象的方法，从而获得对象的功能。不难看出，程序只能通过接口而不能直接调用对象中的方法，使COM对象具有更好的安全性。

在ADO/OLE DB中COM编程的一般结构是：

(1) 初始化COM接口，为以后的操作得到一个环境；

(2) 获得COM对象实例；

(3) 通过接口对对象进行操作；

(4) 清除COM接口。

2 ADO/OLE DB技术

OLE DB是UDA的底层软件接口，它封装了ODBC的功能，但又有所扩展。它是UDA技术的核心，是系统级的访问接口，它由一组COM对象组成。每种对象分别支持数据库应用中的不同内容，如数据源对象、命令对象、事务对象。每种对象有自己的方法和支持的接口。OLE DB数据提供程序(OLE DB Data Provider)支持多种数据库引擎，能够对多种数据源提供一致的接口。用户的数据应用程序(Data Consumer Program)通过数据提供程序访问其中的数据。和ODBC相比，“OLE DB内置了SQL处理程序，不用额外建立SQL驱动引擎”；另外，它还支持一些不规则数据系统，而ODBC只支持关系数据库。根本一点，ODBC是基于传统的API编程，而OLE DB上基于COM的应用组件(模块)编程，是一种更先进、更灵活的方法。

基于COM的OLE DB技术是数据库管理的发展方向。它为各种应用程序提供最佳的访问接口，所以它的特点是应用灵活、速度快。但是OLE DB的一个缺陷是编程代码量大，对底层的操作比较复杂。笔者曾编过OLE DB程序，虽然活动模板库(ATL)的应用可以减少这一缺陷，但实际应用起来还是比较困难。(限于篇幅这里不介绍ATL)而继承了OLE DB功能并对它进行封装、简化的接口就是ADO。

ADO是UDA的高层应用级接口。“它最主要的优点

是易于使用、速度快、内存支出少和使用较少的网络流量”。ADO以OLE DB为基础又对它进行了简化和扩展。OLE DB编程在创建行集对象前必须创建会话对象和命令对象，命令对象是行集对象的父对象，而会话对象和命令对象也具有这种父子关系，这一过程不能颠倒，ADO编程则不用依赖这种对象的层次关系。例如，程序可以在开始就用行集对象的Open ()函数打开行集对象。系统在执行Open()时，会隐式地建立一个到数据源的连接。当然父对象的显式和隐式创建是有不同的作用的，但ADO给用户提供了这样一条选择途径。另外，ADO支持脚本语言，如VBscript、JavaScript。这使它可以直接应用到网页中，和Web结合更加的方便、紧密。ADO是一种理想的Web应用开发工具，特别适合C/S结构。ASP和ADO相结合是当前Web数据库应用的一大热点。

利用ADO编程，用户不用了解OLE DB底层复杂的数据接口。ADO有七种对象。它们分别是连接对象(Coxnection)、命令对象(Commind)、行集对象(RecordSet)、域对象(Field)、参数对象(Parmeter)、错误对象(Error)和属性对象(Property)。这些对象的功能大都可以通过它们的名称体现。其中，属性对象并不单独存在，连接对象、命令对象、域对象和行集对象都包含有属性对象。属性对象实例表明相关对象所具有的一些性质。ADO是基于COM的。所以它的编程步骤也是“初始化COM环境→获得对象指针→操作对象→清除环境”。

	ODBC	OLE DB	ADO
方便、易用	不好	利用ATL较好	好
面象对象性质	不支持利用API编程	好	好
控制层面	底层	底层	高层
访问速度	视供应商的优化质量	很快	快
开放性	只限关系数据库，但可支持多种RDBMS	封装了ODBC，支持COM接口能够与非关系型数据源进行通信，扩展了ODBC的功能	是建立在OLE DB上的高层接口，支持COM接口，支持不同数据源
支持脚本语言	不支持	不支持	支持

3 实例操作

笔者在开发一个小型搜索引擎的过程中，接触到ADO/OLE DB。感觉与以前的数据访问技术OD BC相比要方便易用。上表从应用角度对ADO/OLE DB和ODBC的使用性能作一些简单比较：

笔者的编程环境是Windows98、VC6.0 MFC和ADO1.5。因为ADO在MFC中并没有支持类，而是以COM动态链接库的形式存在，所以必须首先导入ADO的支持类。

导入ADO支持类的方法有两种。一种是利用#import指令；另一种是包括头文件adoid.h和adoint.h。后一种方法会使代码增加，这里不介绍了。

#import 指令格式:

#import "ADO DLL path”

引号中的字符串是指ADO在Visual Studio安装时的安装注册目录。#import指令产生两个文件.TLH和.TLI。它们分别是ADO支持类的头文件和实现文件。导入ADO DLL后，还需要包含一些头文件、预定义和声明名称空间。

```
#import "C:/program files/common files/system/ado/msado/s.dll"
        rename-namespace("ADOCG")
rename("EOF","Endofile")
using namespace ADOCG
#define INITG ID
#include "icrsint.h"
```

下面就可以进行ADO的编程了。笔者的意图是想要把一个URL数组hreflink中的内容放到一个数据库表中，以备检索。

首先利用MFC的Application Wizard创建一个基于对话框的程序框架，并且在对话框上添加显示控件，在“类创建向导”中添加相应的变量和函数。对表中记录进行检索和更新，需要先获取行集。笔者是在初始化对话框的时候取回行集。

定义全局变量

```
_ConnectionPtr pCon;
_CommandPtr pCmd;
_RecordsetPtr pRes;
```

在OnInitDialog()中添加获取行集的代码

```
CoInitialize(NULL);// 首先 初始化 COM 环境
pCon.CreateInstance(_uuidof(Connection));// 获取连结对象的指针实例
pCon->Open("DSN=jebtest","","",-1);// 与数据源建立连接
pCmd.CreateInstance(_uuidof(Command));// 获取命令对象指针实例
pCmd->ActiveConnection=pCon;//设置活动连结属性
pCmd->CommandText="select*from superlink";//设置要执行的 SQL 语句
pRst=pCmd->Execute(NULL,NULL,adCmdUnknown);// 执行 SQL 语句并返回记录集
```

pCon、pCmd和pSet是全局变量，分别表示连结对象、命令对象和行集对象的对象指针。这样在程序启动时，记录集就已经取回来了。

添加记录用的是记录集对象的 AddNew()方法。

```
COleSafeArray fieldlist; // 定义一个 Ole 数组对象
CString field [MAX_LINK] ,value [MAX_LINK];
fieldlist.CreateOneDim(VT_VARIANT,1);// 创建一个一维 ColeSafeArray 对象。表示表的列，因此数组中元素个数为 1
long lArrayIndex [1];
lArrayIndex [0] =0;
fieldlist.PutElement(lArrayIndex,&(_variant_t("url"))); // 第一个元素的内容为“url”
COleSafeArray valuelist;
valuelist.CreateOneDim(VT_VARIANT,1);
lArrayIndex [0] =0;
valuelist.PutElement(lArrayIndex,&(_variant_t(hrelink[0]))); //创建一个一维ColeSafeArray对象，表示列的值。
hr=pRset->AddNew(fieldlist,valuelist); //调用AddNew()添加记录
pRset->Update();
pRset->Close(); // 关闭记录集对象
```

另外如果出错，必须对错误进行捕获和处理。程序采用 try--catch 结构。下面是错误处理代码

```
catch(_com_error &e)
{
_bstr_t bstrSource(e.Source());
_bstr_t bstrDescription(e.Description());
TRACE("/tCode=%081x/n",e.Error());
TRACE("/tCode meaning=%s/n",e.ErrorMessage());
TRACE("/tSource=%s/n",(LPCTSTR)bstrSource);
TRACE("/tDescription=&s/n",(LPSTR)bstrDescription);
}
catch(...)
{
TRACE("There are some UNHANDLED EXCEPTION");
}
CoUninitialize(); // 清除 COM 环境
```

这样就能够把新记录添加到数据库中了。

不管是ODBC还是ADO/OLE DB，它们方法的参数多，并且参数类型复杂，而参考例程往往只是具体实例的参数情况。VC6.0提供了ADO/OLE DB对象和方法库以供编程参考。它的位置是“TOOL” "OLE DB/COM Object Viewer” ALL Objects”中。它包含了每种对象的ID和它详细的类型信息(type info.)。方法说明就在类型信息中。

还有一个问题就是数据类型的转换。数据库数据类型一般和C++数据类型不同，需要在两者之间进行翻译。这个工作是由COM支持类来完成的。_variant_t用于处理数据库字段数据，_bstr_t处理C++数据类型。它们分别封装了VARIANT和BSTR类，可以利用两者之间的相互转化来进行从数据库到C++的转换。下面是笔者找到的一个转化例程。

```
_variant_t vEquipname;
Cstring sEquipname;
vEquipname=pRes->GetCollect(_variant_t("equipname));
vEquipment.ChangeType(VT_BSTR)
sEquipname=cEquipname.bstrVal;
```

通过这些程序，我们可以感到用ADO编程条理是很清楚、简洁的。从建立结构、执行命令到取回记录再对数据进行操作，每一步不需要太多属性设定，并且比较符合一般的思路。■

参考文献

1 David Bennett等著 徐军等译. Visual C++ 5开发人员指南. 北京: 机械工业出版社, 1998

2 Lyn Robison著, 黄惠菊等译。轻松掌握用 Visual C++6 对数据库编程. 北京: 电子工业出版社, 1999

3 http://www.microsoft.com/data/ado/default.htm