基于ADO技术实现多种数据源间SQL查询功能

陈雪梅
(中国科学技术大学,安徽 合肥 230027)

摘 要:以ADO技术为数据库编程的基础,通过ADO对象的数据访问接口,用Microsoft提供的通用数据连接文件(.udl)来建立和测试ADO连接属性,动态建立可视化的数据源设置。利用Visual C++搭建的应用框架,在多种数据源间实现结构化查询语言(SQL)查询的功能。用示例说明,ADO建立的应用程序对数据库的操作与数据库系统的类型无关,它使应用程序把SQL语言作为标准语言来存取各种数据,从而进一步说明ADO技术在数据库应用领域的灵活性和高兼容性。

关键词:ADO;数据库;对象;属性;UDL;SQL
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2007)10-0007-05

Realization of SQL Query Function among Multiple Data Sources Based on ADO Technique

CHEN Xue-mei
(University of Science and Technology of China, Hefei 230027, China)

Abstract: Based on ADO technique for database programming, the ADO link attribute is set up and tested via the data access interface of ADO object by using the universal data link (.udl) supplied by Microsoft, and a visualized data source configuration is dynamically built. The SQL (Structured Query Language) query function is realized by use of the application frame constructed by C++. An example is supplied to show that the operation of databases by the application program built up of ADO is irrelevant to the types of database systems. In the example, the SQL language is used as a standard language by the application program to store and fetch various data. Thus the flexibility and high compatibility of the ADO technique in the fields of database applications are shown.

Key words: ADO; database; object; attribute; UDL; SQL

0 引 言

数据库管理系统(DBMS)是当今最流行的实用技术之一,因此就出现诸如:Access,Oracle,SQL Server,Text,及各种分布式数据库等DBMS,每一种DBMS供应商都为自己的产品提供了交互式的数据库查询工具,而这些工具仅为本系统的数据库服务。如何方便地、高效地访问由这些DBMS开发的数据库,在统一的平台上用SQL语言进行查询操作,而无须进行数据导入导出的转换服务。为实现这一目的,文中在VC环境下运用了ADO技术。

ADO(ActiveX Data Objects,ActiveX数据对象)技术是Microsoft数据库接口的组成部分,是建立在OLE DB之上的高层数据库访问技术,是对OLE DB所提供服务一个方便使用的封装并继承了ODBC的功能。它定义了一组COM对象来操作不同数据源的数据,并以统一的方式访问存储在不同信息源中的数据,为不同的数据源提供了一种统一的面向OLE DB接口的驱动。

1 ADO概述

ADO是一种面向对象的数据访问应用程序接口,具有强大的数据处理功能和简单易用的编程特点,并且得到了广泛的应用。

通过ADO提供的一组非常简单的、封装了一般通用数据访问细节的对象,可以快速创建数据库应用程序。ADO封装了OLE DB,却屏蔽了OLE DB的复杂性,利用ADO提供的API,用户在应用程序中可使用同一个接口来访问不同形式的数据源,这样的数据源不仅能够从DBMS系统中获取而且也可以从其他任何包含信息的非DBMS系统中获取。ADO应用与开发的语言无关[1],任何与COM兼容的编程语言(如Visual C++,Delphi等)都可以使用它。它们之间关系如图1所示。

收稿日期:2006-12-31
作者简介:陈雪梅(1966-),女,浙江宁波人,工程师,主要从事计算机教学与实验。

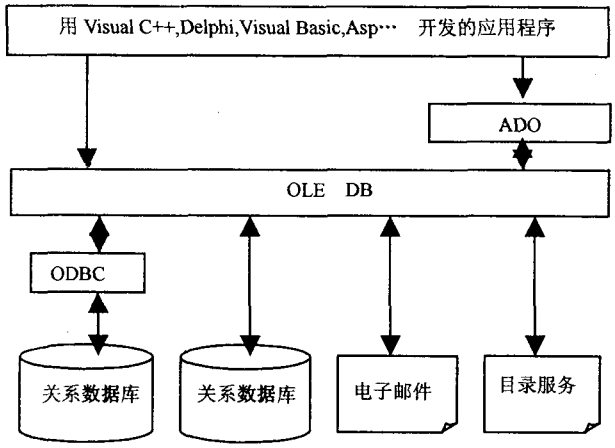

图1 应用程序和数据存储的关系

2 ADO对象

ADO对象包括:Connection Object(连接对象)、Command Object(命令对象)、Recordset Object(记录集对象)、Field Object(字段对象)、Record Object(记录对象)、Error Object(错误对象)、Parameter Object(参数对象)、Property Object(属性对象)和Stream Object(流对象),其中最常用也是最重要的是Connection,Recordset和Command这3个对象。

(1)Connection对象是到本地或远程数据源的连接,它管理应用程序和数据源之间的通信,以后的数据库操作都是建立在这个连接上的。

(2)Recordset对象是用来连接到一个记录集,从而获取查询的结果,这些结果由数据的行(记录)和列(字段)组成,记录集对象能方便地实现记录的添加、修改和删除操作。

(3)Command对象用于定义将对数据源执行的命令,可以运用SQL语句或存储过程,以便执行大量操作或处理数据库结构。

在VC环境下使用这3个对象,需要定义与之对应的3个智能指针,分别为_ConnectionPtr,_RecordsetPtr和_CommandPtr,然后调用它们的CreateInstance方法进行实例化,从而创建这3个对象的实例。本示例中只用到_ConnectionPtr,_RecordsetPtr二个接口指针。

3 ADO的基于UDL连接

ADO与数据源的连接是通过连接智能指针_ConnectionPtr来创建的,在ADO的数据源连接对象属性中较重要的是Provider和ConnectionString属性,使用Provider属性可设置或返回连接提供者的名称,也可以通过ConnectionString属性的内容或Open方法的ConnectionString参数设置该属性。ConnectionString是一个包含建立到数据源连接信息的字符串,该属性提供所要连接的数据库类型、数据所处服务器、要访问的数据库和数据库访问的安全认证信息[2]。通常的方法是在ConnectionString中直接提供以上信息,其大致过程如下:

```
_ConnectionPtr m_pConnection;
m_pConnection.CreateInstance(__uuidof(Connection));//创建连接实例
```

对Access数据库连接:

```
m_pConnection->Open(LPCSTR(Provider=Microsoft.Jet.OLEDB.4.0;DataSource=student.mdb),"","",adModeUnknown);
```

对SQL Server数据库连接:

```
m_pConnection->Open(LPCSTR(Provider=SQLOLEDB.1;User ID=student;Password='';
Initial Catalog=pubs;Data Source=SERVER2003),"","",adConnectUnspecified)
```

在这段代码中是通过Connection对象的Open方法来进行连接数据库的,Open的原型:HRESULT Connection15::Open(_bstr_t ConnectionString,_bstr_t UserID,_bstr_t Password,long Options),其中,ConnectionString为连接字符串,User ID是用户名,Password是登陆密码。可以看出上述的连接属性设置标准随着数据源的类型不同而变化,当用户需要在多种不同的数据源中进行SQL语言操作时,每次都需修改代码的设置方式,这是很不方便的,于是希望有可视化的数据源设置方法。为此Microsoft提供了通用数据连接文件(.UDL)来建立和测试ADO连接属性,用可视化方式来定义要连接的数据源,通过"属性"中的"提供者"、"连接"、"高级"、"所有"各选项,实现数据访问的透明性。文中示例中通过设置以上各项,建立了A.udl文件。

ADO连接对象可以很方便地使用UDL文件来连接数据源,A.udl文件与ADO连接过程:

```
_ConnectionPtr m_pConnection;
m_pConnection.CreateInstance(__uuidof(Connection));//创建连接实例
m_pConnection->ConnectionString="File Name=A.udl";
m_pConnection->Open("","","",NULL);
```

这样一来无论数据源如何变化,在代码中都可以用统一的方法编程。当数据源改变时,只要双击相应的A.udl文件即可可视化地设置数据源,无需再更改代码,对于那些独特的数据文件或更新的、更先进的数据库系统来说,同样可以选用相应的OLE DB Provider来访问,从而很方便地实现跨多种数据源数据SQL查询操作。

4　创建数据库、表

为了检测所创建实例的需要,分别在 Access,SQL Server 的 DBMS 中创建数据库、表。先在 Access 中建立一个空数据库 student.mdb,手动添加各种数据库对象,并将它保存到"d:\student.mdb"中。在数据库 student.mdb 中新建一个 student 表,它包括"学号"、"姓名"、"性别"、"年龄"、"系别",分别用"Sno","Sname","Ssex","Sage","Sdept"代替,数据类型分别为文本和数字,并把"Sno"设为"主键"。然后再在 SQL Server 的 pubs 中创建表 student1,也有"Sno","Sname","Ssex","Sage","Sdept"字段,其中设 Sno 为 NOT NULL UNIQUE,并插入一些与上个表不同的值。当然也可以打开 SQL Server 企业管理器来创建新的数据库,再在新库中添加表。

5　ADO 应用实例

5.1　建立工程

利用 MFC App Wizard 建立一个基于对话框的工程 Cxm1,建立过程中均使用默认选择。在对话框中加入控件及属性,见表 1。

5.2　ADO 库、OLE/COM 库环境、全局变量 m_pConnection

在默认情况下,Visual C++不支持 ADO 对象。ADO 类的定义是作为一种资源存储在 ADO DLL(msado15.dll)中,在其内部称为类型库。它描述了自治接口,以及 C++使用的 COM vtable 接口。当使用 ADO 时需用 #import 指令从 ADO DLL 中读取这个库文件。将 ADO 库文件导入到工程中,一般是在 stdafx.h 文件的最后加入如下语句来实现:#import "C:\Program Files\common files\system\ado\msado15.dll" no_namespace rename("EOF","adoEOF")。上述代码"C:\Program Files\common files\system\ado\msado15.dll"指明了文件 msado15.dll 的具体位置,当 VC++读出 msado15.dll 中的类型库信息时,在当前编译目录下自动生成两个该类型库的头文件和实现文件 msado15.tlh 和 msado15.tli,这两个文件为每一个接口产生了智能指针,并为各种接口方法、枚举类型以及 CLSID 等进行声明并创建一系列包装方法[3]。no_namespace 指明 ADO 对象不使用命名空间。为了避免出现常量名冲突将 EOF 改名为 adoEOF。

由于 ADO 库是一组 COM 动态库,因此在应用程序调用 ADO 前,必须初始化 OLE/COM 库环境。在 MFC 应用程序中,一般在应用程序主类的 InitInstance()成员函数里初始化 OLE/COM 库环境。

定义一个在"CCxm1Dlg"类内有效的全局变量 m_pConnection,切换到类视图,右键单击"CCxm1Dlg"类,选择"Add Member Variable…"打开添加成员变量对话框,变量类型输入"_ConnectionPtr",变量名输入"m_pConnection",使用选"Public",点击"OK",这样就声明了一个"m_pConnection"全局变量。

5.3　连接数据库

要操作数据库中的数据,需要连接到数据库中去。ADO 通过_ConnectionPtr 智能指针与数据源进行连接,并调用 CreateInstance()来创建一个连接对象的实例,再调用 Open()函数来创建与数据源的连接。在 OnButtonConnect()函数中加入以下代码,完成上述连接过程:

```
void CCxm1Dlg::OnButtonConnect()
{
```

表 1　控件及属性

对象	ID	属性修改(需要设置的值)	变量(类型、变量名)	事件处理(事件、映射函数)
DataGrid	IDC_DATAGRID_SHOW	Caption(查询结果显示)	CDataGrid m_cdatagridShow	
Static Text	IDC_STATIC	Caption(SQL 语句输入)		
Edit Box	IDC_EDIT_SQL	Multiline(True),WantReutn(True) Border (false),Static edge(True)	CString m_cstringeditSQL	EN_CHANGE CCxmlDlg::OnChangeEditSql
Static Text	IDC_STATIC	Caption(操作结果:)		
Static Text	IDC_STATIC_RESULT	Caption()	CString m_cstringstaticResult	
Button	IDC_BUTTON_CONNECT	Caption(连接)		BN_CLICKED CCxmlDlg::OnButtonConnect
Button	IDCB_BUTTONB_EXECUTE	Caption(执行)		BN_CLICKED CCxmlDlg::OnButtonExecute
Button	IDC_BUTTON_DISCONNECT	Caption(断开)		BN_CLICKED CCxmlDlg::OnButtonDisConnect
Button	IDC_BUTTON_EXIT	Caption(退出)		BN_CLICKED CCxmlDlg::OnButtonExit

```
//先判断连接是否存在以及连接的状态,已经连接则返回
if((m_pConnection ! =NULL) & & (m_pConnection->State ! =adStateClosed))
{
  m_cstringstaticResult = _T("请先断开正在连接的数据库");
  UpdateData(false);
  return;
}
//创建连接实例
HRESULT hr=m_pConnection.CreateInstance(_uuidof(Connection));
if( FAILED(hr))
{
  m_cstringstaticResult=_T("创建 Connect 实例失败");
  UpdateData(false);
  return;
}
  //设置等待时间
  m_pConnection->put_ConnectionTimeout(long(5));
  //用 try…catch()来捕获错误信息
  try
  {
    m_pConnection->ConnectionString="File Name=A.udl";
    m_pConnection->Open("","","",NULL);
  }
  catch(_com_error& e)
  {
    m_cstringstaticResult=_T("数据库连接失败:");
    m_cstringstaticResult += e.ErrorMessage();
    UpdateData(false);
    return;
  }
  m_cstringstaticResult=_T("数据库连接成功");
  UpdateData(false);
}
```

在这段代码中用到了 ConnectionString 和 A.udl 文件,代码中 ConnectionTimeout 属性指示在终止尝试和产生错误之前执行命令需等待的时间,默认值为 15s,这里设定 5s。在应用程序中,通常用 try…catch()来捕获错误信息,任何涉及 ADO 对象的操作都会生成一个或多个提供者错误,在每个错误出现时,一个或多个 Error 对象将被放到 Connection 对象的 Error 集合中,当另一个 ADO 操作产生错误时,Error 集合将被清空,并在其中放入新的 Error 对象集,需要指出的是,每个 Error 对象代表的是特定的提供者错误而并非是 ADO 错误。

5.4 执行 SQL 语句

数据库连接成功后,如何使编辑框的 SQL 语句通过"执行"按钮,在 CDataGrid 窗口中以图形化的方式显示出来,这就要用到 ADO 中的 Recordset 对象和控件。ADO 将查询的结果存放在 Recordest 对象中,Recordest 对象将这些结果进行诸如更新、排序和过滤等其它功能操作,然后通过使用 DataGrid 控件将当前执行后的结果显示出来。通过创建一个对象实例接口 m_pRecordsetPtr,用 Open()打开代表基本表、查询结果或以前保存的 Recordest 中记录的游标的集合。下面是实现这一功能的源代码:

```
void CCxm1Dlg::OnButtonExecute()
{
  //先判断是否已经建立和数据库的连接
  if((m_pConnection==NULL) || (m_pConnection->State ! =adStateOpen))
  {
    m_cstringstaticResult=_T("请先连接数据库");
    UpdateData(false);
    return;
  }
  //打开,首先创建一个_RecordsetPtr 实例,然后调用 Open()得到一条 SQL 语句的执行结果
  _RecordsetPtr m_pRecordsetPtr;
  HRESULT hr = m_pRecordsetPtr.CreateInstance(_uuidof(Recordset));
  if( FAILED(hr) )
  {
    m_cstringstaticResult=_T("创建 RecordSet 实例失败");
    UpdateData(false);
    return;
  }
  try
  {
    m_pRecordsetPtr->CursorLocation=adUseClient;
    m_pRecordsetPtr->Open((LPCSTR)m_cstringeditSQL,m_pConnection.GetInterfacePtr(),adOpenDynamic,adLockOptimistic,adCmdText);
  }
  catch(_com_error& e)
  {
    m_cstringstaticResult=_T("操作失败:");
    m_cstringstaticResult += e.ErrorMessage();
    UpdateData(false);
    return;
  }
  m_cstringstaticResult=_T("操作成功");
```

```
    UpdateData(false);
    //显示查询结果
    m_cdatagridShow.SetRefDataSource(NULL);
    m_cdatagridShow.SetRefDataSource((LPUNKNOWN)m_pRecordsetPtr);
    m_cdatagridShow.Refresh();
}
```

其中,在代码 Open((LPCSTR)m_cstringeditSQL, m_pConnection.GetInterfacePtr(), adOpenDynamic, adLockOptimistic, adCmdText)中,m_cstringeditSQL 是记录源(Source),这里代表编辑框中所写入的一条 SQL 语句,m_pConnection.GetInterfacePtr()指定相应的 Connection 对象,adOpenDynamic 指定打开 Recordset 时使用的游标类型,它的值选项[4]见表 2。adLockOptimistic 指定打开 Recordset 时应该使用的锁定类型,adCmdText 指定将 m_cstringeditSQL 语句视为命令,它的其它可选属性值[4]见表 3。

表 2　游标属性的值

常量	说　明
adOpenForwardOnly	(默认值)打开仅向前类型游标
adOpenKeyset	打开键集类型游标
adOpenDynamic	打开动态类型游标
adOpenStatic	打开静态类型游标

表 3　常用 Source 参数的类型

常量	说　明
adCmdText	将 Source 视为命令
adCmdTable	生成 SQL 查询从在 Source 中命名的表中返回所有行
adCmdTableDirect	直接从在 Source 中命名的表中返回所有行
adCmdStoredProc	将 Source 视为存储过程
adCmdUnknown	Source 参数中的命令类型为未知
adCmdFile	从在 Source 中命名的文件中恢复保留(保存的)Recordset

5.5　断开数据库

访问完数据库后,为了节省资源,通常需要将数据源的连接关闭,执行 OnButtonDisconnect()函数,代码如下:

```
void CCxm1Dlg::OnButtonDisconnect()
{
    //断开连接
    if((m_pConnection != NULL) && (m_pConnection->State != adStateClosed))
    {
        m_pConnection->Close();
        m_pConnection=NULL;
        m_cstringstaticResult=_T("已经断开正在连接的数据库");
        UpdateData(false);
    }
}
```

5.6　退出功能界面

```
void CCxm1Dlg::OnButtonExit()
{
    //断开连接
    OnButtonDisconnect();
    //退出
    OnOK();
}
```

到此为止,利用 ADO 在 Visual C++中进行数据库编程的一个具体应用程序就完成了,运行该程序效果如图 2 所示。这样就可以在 VC 开发的平台上用 SQL 语句来操纵这些不同数据源的数据库。

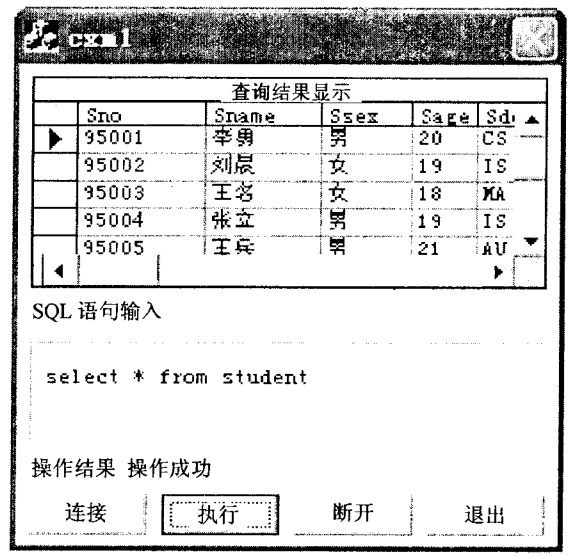

图 2　程序运行效果

6　结束语

在 Visual C++开发环境下,基于访问数据库采用 ADO 或 ADO.NET 技术,这样的应用程序开发比较简便灵活,而且访问效率高、速度快、易使用、内存支出少、磁盘遗迹小,同时也使程序员更容易控制对数据库的访问。ADO 技术中 OLE DB 为任何数据源提供了高性能的访问,这些数据源不仅仅只是 Access,SQL,Oracle 等数据库,还包括可以使用 OLE DB Provider 来访问的任何数据源,如:电子邮件和文件系统、目录服务和图形等。另外,ADO 还支持各种客户/服务器模块与基于 Web 的应用程序[5],具有远程数据服务的特性。正是 ADO 具有的这些特点,该技术已成为现今数据库开发应用的热点,逐渐被广大的用户所接受。

(下转第 68 页)

工给出。

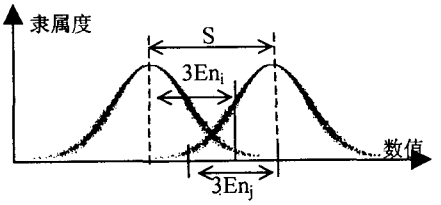

图 4 边缘事物的模糊判断(Ⅱ)

3 改进后的算法与原算法的比较

改进后的算法 3 与算法 1、2 相比有以下优点:

(1) 算法 3 比算法 1 和算法 2 更符合人类认识事物的规律。算法 3 只对那些少数处于各概念边缘且和邻近概念交合的地方的事物进行不确定性判断,而不是对每个事物进行不确定性判断。

(2) 算法 3 比算法 1 和算法 2 对事物属于哪个概念的判断更准确。虽然,云模型强调随机性和模糊性,但并不是没有规律的没有标准的把一个 100% 或 90% 多的属于 B 的事物判定为属于 20% 或者 10% 属于 A 或 C 的概念,而算法 3 把那些相对比较明确的属于某概念直接判给它应该属于的概念,这样就可以避免确定属于 B 概念的事物被判给隶属度较小的 A 或 C 概念。这里通过算法 1、算法 2 和算法 3 对 $x(20)$ 这个对象属于概念集{cloude1(10.0,5,0.2), cloude2(20.0,5,0.2),cloude3(28.0,5,0.2),cloude4(35.0,5,0.2)}中哪个概念分别进行了 100 次判定,其中四云概念图及三判定算法的结果如图 5、图 6 所示。

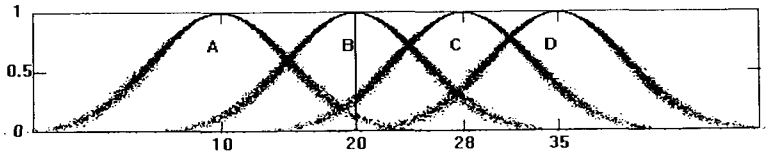

图 5 四云概念图

从四云图(图 5)中可以看出 $x(20)$ 的对象正处于 B 概念的中心,隶属度为 100%,但是,从以上的三判定结果(图 6)可以看出,算法 1 和算法 2 有可能将对 B 概念隶属度为 100% 的对象 $x(20)$ 判定为属于 A 或 C,这明显是错误的。而算法 3 则不会出现这种错误。

(3) 算法 3 比算法 1 和算法 2 降低了算法的时间复杂度。毕竟在划分概念时,概念之间交合的部分只占少数。因此,同时属于两个或多个概念的事物的数量较少。本算法把大多数明确属于某概念的事物直接判给该概念。这样通过简单的"明确属于某概念的判断",省去了对大量事物的复杂的不确定性判定,降低了算法的时间复杂度。

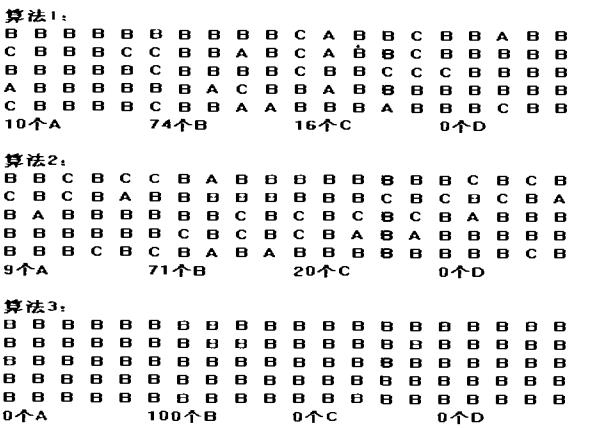

图 6 三判定算法对 $x(20)$ 的 100 次判定结果

4 结束语

分析了基于云模型的隶属概念判定的两种算法存在的不符合人类对客观事物的认识规律、错误判定和时间复杂度高等不足,并给出了改进算法,从而避免了"确定事物的不确定的判定"和"不确定事物的确定判定"的错误,增加了算法的准确性和科学性,使算法更加符合人类对客观事物的认识规律,并且降低了算法的时间复杂度。

参考文献:

[1] 邸凯昌.空间数据发掘与知识发现[M].武汉:武汉大学出版社,2001:1-121.

[2] 蒋 嵘,李德毅,范建华. 数值型数据的泛概念树的自动生成方法[J].计算机学报,2000,23(5):471-476.

[3] 李兴生,李德毅.一种基于云模型的决策表连续性离散化方法[J].模式识别与人工智能,2003,16(1):33-38.

[4] 张家精,姜其岩,陈金兰. 基于云理论和 Rough 集理论的数据发掘和知识发现[J].贵州大学学报:自然科学版,2004,21(增刊 2):14-17.

[5] 李德毅,杜 鹢.不确定性人工智能[M].北京:国防工业出版社,2005.

(上接第 11 页)

参考文献:

[1] 刘 瑞,吴跃进,王宗跃. Visual C++项目开发实用案例[M].北京:科学出版社,2006.

[2] 求是科技. Visual C++6.0 程序设计与开发技术大全[M].北京:人民邮电出版社,2004.

[3] 郎 锐. Visual C++数据库开发基础及实例解析[M].北京:机械工业出版社,2005.

[4] 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,2006.

[5] 扬富国. Visual C++程序开发案例解析[M].北京:清华大学出版社,北京交通大学出版社,2006.