C++ Builder 5

高级编程技术

—Database与MIDAS编程

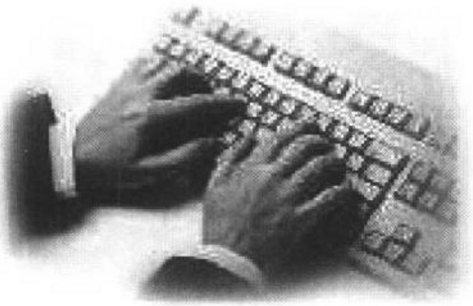
徐新华　等编著

人民邮电出版社
www.pptph.com.cn

C++ Builder 5
高级编程技术

—— Database 与 MIDAS 编程

徐新华　等 编著



人民邮电出版社

内容提要

本书全面深入地介绍 C++ Builder 5 的数据集、数据库访问链路、数据感知控件、ADO、访问 Interbase、Client/Server、MIDAS、决策支持、QuickReport 报表、TeeChart 图表、数据库浏览器等内容。

C++ Builder 5 是一个完全面向对象的编程工具。众多长期从事编程的人员从实践中体会到，只要真正领会了面向对象的编程思想，即使是很高深的编程领域，诸如 COM、ActiveX、CORBA、MIDAS 都不难掌握；所以，本书的重点是面向对象编程。

本书内容全面而又不失简洁，例子丰富，既可以作为广大读者学习 C++Builder 5 的入门指导书，也可以作为程序员编程时的参考手册。

C++ Builder 5 高级编程技术
——Database 与 MIDAS 编程

◆ 编　　著　徐新华　等
责任编辑　王晓明
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@pptph.com.cn
网址　http://www.pptph.com.cn
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销
◆ 开本：787×1092　1/16
印张：21.5
字数：536 千字　　2000 年 12 月第 1 版
印数：1－5 000 册　　2000 年 12 月北京第 1 次印刷
ISBN 7-115-09012-2/TP·1988
定价：32.00 元

前　言

C++ Builder 5 是用于电子商务、Internet 应用和数据库编程等开发工作的最佳工具之一。C++ Builder 5 最接近 ISO 的 C++标准；同时支持 COM 与 CORBA 两大分布式计算规范；C++ Builder 5 支持 ADO，从而可以访问更多的数据；C++ Builder 5 增加了数据模块设计器，可以轻松地创建和维护数据模块；通过 InterBase Express(IBX)，C++ Builder 5 集成了对 InterBase 数据库的访问，不再需要借助于 BDE；MIDAS 与新增加的 Internet Express 配合使用，使客户(Web 浏览器)能更方便地与 MIDAS 服务器交互；C++ Builder 5 内建了全球 ORB 分发数量最多的 VisiBroker 4.0，集成了 CORBA IDL 编译器，单一步骤就能生成 CORBA 对象；TeamSource 是一个集成的工作流程管理工具，大大简化了团队开发；运用 Integrated Translation Environment(ITE)，可以方便地进行软件的本地化和国际化。另外，C++ Builder 5 可以很方便地操纵 Microsoft Office 97/2000 的文档和程序，这在进行企业级开发时是很重要的。

为了帮助广大用户全面、准确地掌握 C++ Builder 5 的编程思想和用法，作者专门编写了这套《C++ Builder 5 高级编程技术》。这套丛书与作者以前编写的《C++ Builder 4 高级编程丛书》之间有着继承性，但在内容上，因该软件版本的升级又有很大的区别。这套《C++ Builder 5 高级编程技术》可以使读者在掌握了 C++ Builder 4 编程技术以后，进一步学习使用新版本软件编程的方法和技巧。考虑到很多程序员已经初步掌握了 C++ Builder 5 的基本用法，因此本套丛书内容的重点放在了编程技术的进一步精通和提高上。

本套丛书分为 4 册。第 1 册介绍 IDE 与 OOP 编程，第 2 册介绍 GUI 编程，第 3 册介绍 Database 与 MIDAS 编程，第 4 册介绍 COM、CORBA 与 Internet 编程。

本书是丛书的第 3 册，全面深入地介绍了 C++ Builder 5 的数据集、数据库访问链路、数据感知控件、ADO、访问 Interbase、Client/Server、MIDAS、决策支持、QuickReport 报表、TeeChart 图表、数据库浏览器等内容。

本书主要由徐新华编写，另外，参加本书编写工作的还有顾洪均、凌晨、张莉、郭平等人。由于我们的水平有限，再加上时间很紧，因此，尽管我们作了比较严格的审核和测试，但书中还是难免会有一些错误，敬

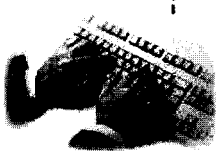
请广大读者不吝赐教，我们谨在此表示感谢。

为了帮助广大程序员更好地掌握 C++ Builder 5 这个优秀的开发工具，北京东大阿尔发软件技术有限公司愿意为购买此书的读者提供咨询。

北 京 东 大 阿 尔 发 软 件 技 术 有 限 公 司
地址：北京市上地信息产业基地上地村路 1 号(100085)
电话：(010)62987260　传真：(010)62985141
网址：http://www.allfa.com.cn　邮件：books@allfa.com.cn

作 者
2000 年 8 月

目　录



第一章　数据集

数据集是一组离散的数据记录的集合。在 C++ Builder 5 中，数据集有三种表现形式：表、查询、存储过程，这三种形式的数据集分别用 TTable、TQuery、TStoredProc 来操纵。

TTable、TQuery、TStoredProc 的直接上级是 TDBDataSet，TDBDataSet 是从 TBDEDataSet 派生的，而 TBDEDataSet 又是从 TDataSet 派生的。

C++ Builder 5 用 TClientDataSet 来实现和操纵分布式数据集，而 TClientDataSet 也是从 TDataSet 派生的。

C++ Builder 5 还允许通过 ActiveX Data Objects(ADO)来访问数据，而 TADODataSet 或更具体地说是 TADOTable 和 TADOQuery 也是从 TDataSet 派生的。

可见，不管是基于 BDE 的数据集，还是分布式数据集，或者是 ADO 数据集，TDataSet 是它们的共同基类。这些数据集之间的继承关系如图 1.1 所示。

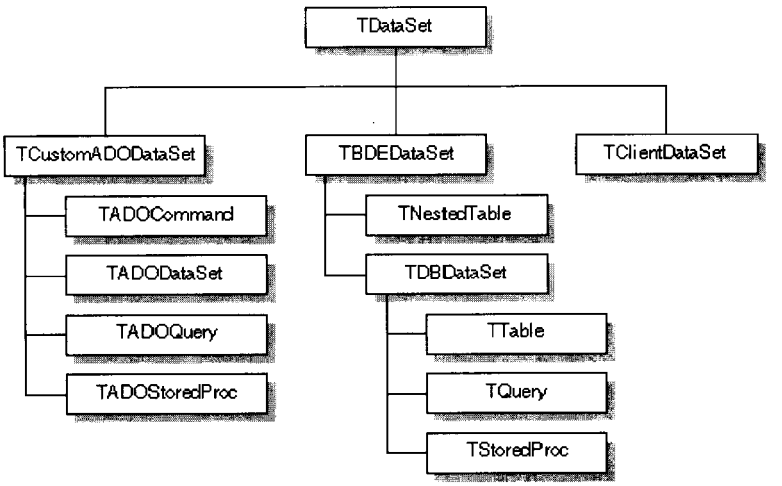


图 1.1　数据集的继承关系

1.1　TDataSet

TDataSet 是所有数据集的虚拟基类。不能直接创建它的对象实例，也不能直接访问它的特性和方法，因为这些特性和方法大部分是虚拟的或抽象的。

如果从功能上划分，TDataSet 的成员可以分为这么几大块：打开和关闭数据集、浏览记录、编辑数据、书签管理、控制连接、访问字段、记录缓冲区管理、过滤、事件。

Active 特性

声明：_property bool Active;

如果这个特性设为 true，相当于调用 Open()来打开数据集；如果这个特性设为 false，相当于调用 Close()来关闭数据集。

也可以通过 Active 特性来判断数据集是否打开。程序示例如下：

```
void _fastcall TForm1::Button1Click(TObject *Sender)
{
    Table1->Active = !(Table1->Active);
    if(Table1->Active)
        Caption = "表已打开";
    else
        Caption = "表还没有打开";
}
```

提示： 如果要访问 SQL 服务器，在打开数据库中的一个数据集之前，首先要先连接数据库。当关闭了数据库中的最后一个数据集，连接就会终止。连接或断开数据库会有一些开销，因此，最好用 TDatabase 元件来管理数据库的连接。

AutoCalcFields 特性

声明：_property bool AutoCalcFields;

当从数据集中检索记录时，将触发 OnCalcFields 事件。如果 AutoCalcFields 特性设为 true，当计算字段的值被更新时，也将触发这个事件。

注意： 如果 AutoCalcFields 设为 true，在处理 OnCalcFields 事件的句柄中不能再修改数据集；否则，将老是触发这个事件，从而导致无限循环。

BlockReadSize 特性

声明：_property int BlockReadSize;

如果这个特性设为 0，每当调用 Next()时，关联的数据感知控件将刷新。为了提高应用程序的性能，可以把 BlockReadSize 特性设为一个大于 0 的数，这样，只有当调用了 Next()若干次后，关联的数据感知控件才刷新一次。

Bof 特性

声明：_property bool Bof;

如果当前记录是数据集的第一条记录，则这个只读的特性将返回 true。当程序第一次打开一个数据集或者调用了 First()后，当前记录就是第一条记录。程序示例如下：

```
void _fastcall TForm1::Button1Click(TObject *Sender)
{
    Table1->First();
```

```
    if(Table1->Bof)
        ShowMessage("当前是在表的开头");
}
```

Bookmark 特性

声明：_property System::AnsiString Bookmark;

书签用于保存数据集中某个记录位置。这样，就可以方便地回到书签所标记的地方。

如果读这个特性的话，这个特性将返回当前记录的书签(如果有的话)。也可以写这个特性来指定一个已有的书签，使该书签标记的记录成为当前记录。

注意：C++ Builder 5 仍然支持 GetBookmark()、GotoBookmark()和 FreeBookmark()。不过，最好不要用这些方法，除非要保持对 16 位程序的兼容。

CanModify 特性

声明：_property bool CanModify;

如果这个只读的特性返回 false，表示数据集是只读的，不能修改。不过，即使 CanModify 特性返回 true，也并不意味着数据集一定能修改。例如，要修改 SQL 表，需要有访问权限。

如果 ReadOnly 特性设为 true，CanModify 特性肯定返回 false。但如果 ReadOnly 特性设为 false，只有当获得了对数据集的修改权限后，CanModify 特性才返回 true。

Constraints 特性

声明：_property TCheckConstraints Constraints;

这个特性用于给数据集建立纠错规则(仅限于记录级)。在运行期，可以通过 TCheckConstraint 对象和 TCheckConstraints 对象来操纵每一个纠错规则。

DataSetField 特性

声明：_property TDataSetField* DataSetField;

这个特性是一个 TDataSetField 对象，它代表一个嵌套的数据集。

DefaultFields 特性

声明：_property bool DefaultFields;

如果数据集中包含运行期动态生成的字段(TField 对象)，则这个只读的特性将返回 true。如果用字段编辑器创建了永久的字段，则这个特性将返回 false。

Eof 特性

声明：_property bool Eof;

如果当前记录是数据集的最后一条记录，则这个只读的特性将返回 true。当程序打开一个空的数据集或者调用了 Last()时，当前记录就是最后一条记录。程序示例如下：

```
void _fastcall TForm1::DataSource1DataChange(TObject *Sender,TField *Field)
{
```

```
    StatusBar1->SimplePanel = true;
    if(Table1->Eof)
        StatusBar1->SimpleText = "现在是在表的最后";
}
```

FieldCount 特性

声明：_property int FieldCount;

这个只读的特性返回数据集中的字段数。由于数据集中可能包含动态生成的字段，因此，这个特性返回的字段数与物理字段数可能不同。程序示例如下：

```
void _fastcall TForm1::Button1Click(TObject *Sender)
{
    String Info("表中的字段有: ");
    for(int i = 0; i < Table1->FieldCount; i++)
        Info = Info + Table1->Fields[i]->FieldName + ", ";
    ShowMessage(Info);
}
```

FieldDefs 特性

声明：_property TFieldDefs* FieldDefs;

C++ Builder 5 用 TFieldDef 对象来操纵数据集中的每一个字段(计算字段除外)。

一个数据集中有几个字段，就有几个 TFieldDef 对象。这些 TFieldDef 对象由 FieldDefs 特性(TFieldDefs 对象)来管理。

一般情况下，不需要修改字段的定义，除非要在运行期动态创建一个表或字段。下面的代码动态创建了一个表：

```
void _fastcall TForm1::Button1Click(TObject *Sender)
{
    Table1->Active = false;

    Table1->DatabaseName = "BCDEMOS";     // 先描述表的类型和名称
    Table1->TableType := ttParadox;
    Table1->TableName := "CustInfo";

    Table1->FieldDefs->Clear();           // 给出字段的定义
    Table1->FieldDefs->Add("Field1", ftInteger, 0, true);
    Table1->FieldDefs->Add("Field2", ftString, 30, false);

    Table1->IndexDefs->Clear();           // 描述表的索引
    Table1->IndexDefs->Add("","Field1",TIndexOptions()<<ixPrimary<<ixUnique);
    Table1->IndexDefs->Add("Fld2Index","Field2",
    TIndexOptions() << ixCaseInsensitive);
```

```
    Table1->CreateTable();                    // 创建这个表
}
```

FieldList 特性

声明：_property TFieldList* FieldList;
通过这个特性可以访问一个数据集中所有字段的名称。

Fields 特性

声明：_property TField* Fields[int Index];
通过这个特性，可以访问数据集中的每一个字段(TField 对象)，序号从 0 开始。
下面的代码演示了如何读一个字段的值：

```
Edit1->Text = Table1->Fields[0]->AsString;
```

下面的代码演示了如何对一个字段赋值：

```
Table1->Edit();
Table1->Fields[0]->AsString = Edit1->Text;
Table1->Post();
```

Fields 特性往往与 FieldCount 特性配合起来使用，程序示例如下：

```
void _fastcall TForm1::Button1Click(TObject *Sender)
{
    for(int i = 0; i < Table1->FieldCount; i++)
        ListBox1->Items->Add(Table1->Fields[i]->FieldName);
}
```

如果仅仅要访问字段的值的话，也可以使用 FieldValues 特性。

FieldValues 特性

声明：_property System::Variant FieldValues[System::AnsiString FieldName];
通过这个特性，可以按字段名来访问字段的值。程序示例如下：

```
Customers->Edit();
Customers->FieldValues["CustNo"] = Edit1->Text;
Customers->Post();
```

由于 FieldValues 是 TDataSet 的默认特性，上述代码可以简化为：

```
Customers->Edit();
Customers["CustNo"] = Edit1->Text;
Customers->Post();
```

FieldValues 特性的数据类型是 Variant，它可以表达任何数据类型的值。因此，不需要用诸如 AsString、AsInteger 来转换字段的值。

Filtered 特性

声明：_property bool Filtered;

很多情况下，程序往往只关心数据集中的部分记录，这就需要把满足特定条件的记录过滤出来。要过滤记录，有两种方式：

1. 调用 Locate()来寻找匹配的记录。

2. 把 Filtered 特性设为 true，并且通过 Filter 特性来指定一个过滤条件。这样，数据集中的指针在移动时，就会触发 OnFilterRecord 事件。通过 Accept 参数可以控制是否过滤某记录。下面的代码过滤了 State 字段的值是“CA”的记录：

```
void _fastcall TForm1::Table1FilterRecord(TDataSet *DataSet, bool &Accept)
{
    Accept = DataSet["State"] = = "CA";
}
```

在运行期，可以给 OnFilterRecord 事件动态指定一个句柄，从而使程序按照不同的条件来对记录进行过滤。程序示例如下：

```
Table1->OnFilterRecord = AnotherFilterRecord;
```

要取消过滤，可以把 Filtered 特性设为 false，这样，OnFilterRecord 事件不再触发。不过，要显示所有的记录，必须调用 Refresh()。

要说明的是，如果记录很多，最好还是通过 SQL 查询或设置范围来检索特定的记录。

Filter 特性

声明：_property System::AnsiString Filter;

这个特性用于设置过滤条件，实际上是一个字符串，其格式类似于 SQL Select 语句中的 Where 部分。过滤条件中可以含有比较运算符，诸如<、>、>=、<=、= 、<>等。

下面的代码表示要过滤 OrderNum 字段的值大于等于 16 的记录：

```
OrderNum >= 16;
```

您可以通过 and、not、or 等关系运算符来构成多重条件。下面的代码表示要过滤 OrderNum 字段的值大于等于 16，并且 Sex 字段为“Man”的记录：

```
(OrderNum>=16) and (Sex = 'Man');
```

如果字段的名称本身含有空格，要用一对方括号把字段名称括起来。例如：

```
[Home State] = 'CA' or [Home State] = 'MA'
```

FilterOptions 特性

声明：_property TFilterOptions FilterOptions;

这个特性用于设置过滤选项。TFilterOptions 是一个集合，可以包含下列元素：

- foCaseInsensitive　　大小写不敏感；
- foNoPartialCompare　　对于字符串类型的字段必须全字匹配。

Found 特性

声明：_property bool Found

如果这个只读的特性返回 true，表示 FindFirst()、FindLast()、FindNext()或 FindPrior()等函数调用成功。

Modified 特性

声明：_property bool Modified;

如果这个只读的特性返回 true，表示当前记录被修改了，但还没有被写到数据集中。调用了 Cancel()或 Post()后，这个特性将恢复为 false。程序示例如下：

```
if (Table1->Modified)
{
    if (MessageDlg("要保存当前记录吗？", mtConfirmation,
        TMsgDlgButtons() << bYes << mbNo, 0) = mrYes)
        Table1->Post();
    else
        Table1->Cancel();
}
```

ObjectView 特性

声明：_property bool ObjectView;

对于含有对象字段(TADTField、TArrayField、TDataSetField、TReferenceField)的数据集来说，如果这个特性设为 true，数据集中的字段将以树状结构存储。否则，数据集中的字段将以线性存储，子字段紧邻着对象字段。

RecNo 特性

声明：_property int RecNo;

这个特性返回当前记录在数据集中的序号。这个特性与 RecordCount 特性配合使用，可以遍历数据集的所有记录。对于 Paradox 表来说，通过 RecNo 特性可以指定当前记录。

RecordCount 特性

声明：_property int RecordCount;

这个只读的特性返回数据集的记录数。下面的代码用一个百分数来显示处理数据集记录的进度：

```
int I = 1;
Table1->First();
While (!Table1->Eof)
{
    StatusBar1->Panels[0]->Text = IntToStr((I*100) div RecordCount);
    I +=1;
    Table1->Next();
}
```

注意：如果对数据集进行过滤或者设置了范围，则 RecordCount 特性返回的记录数与物理记录数可能不同。不过，对于 dBASE 表来说，RecordCount 特性总是返回物理记录数。

对于一个很大的数据集尤其是 SQL 表来说，访问 RecordCount 特性要耗费相当长的时间。因此，一般只对 Paradox 表和 dBASE 表来使用 RecordCount 特性。

RecordSize 特性

声明：_property Word RecordSize;
这个只读的特性返回当前记录缓冲区的长度(以字节为单位)。

SparseArrays 特性

声明：_property bool SparseArrays;
如果 SparseArrays 特性设为 false，当打开一个含有数组字段(TArrayField)的数据集时，将为数组的每个元素创建一个字段对象，这样，数据感知控件就可以单独显示其中的一个元素。把 SparseArrays 特性设为 true，可以节省内存，但无法引用数组的每一个元素。

State 特性

声明：_property TDataSetState State;
这个只读的特性返回数据集当前的状态。可以是以下值：

- dsInactive　　数据集已关闭，不能访问；
- dsBrowse　　数据集处于活动状态，能够浏览它的数据但不能编辑；
- dsEdit　　数据集处于活动状态，能够编辑；
- dsInsert　　数据集处于活动状态，能够插入记录；
- dsSetKey　　数据集处于活动状态，正在调用 SetRange；
- dsCalcFields　　数据集处于活动状态，正在处理 OnCalcFields 事件；
- dsFilter　　数据集处于活动状态，正在处理 OnFilterRecord 事件。

下面的代码修改了数据集中的一个 BLOB 字段：

```
void _fastcall TForm1::Button1Click(TObject *Sender)
{
    TBlobStream *Stream1 = new TBlobStream(Table1Notes, bmReadWrite);
    try
    {
        Table1->Edit();
        if (Table1->State = = dsEdit)
        {
            Stream1->Seek(60, 0);
            Stream1->Truncate();
            Table1->Post();
        }
    }
    catch (...)
    {
        delete Stream1;
```