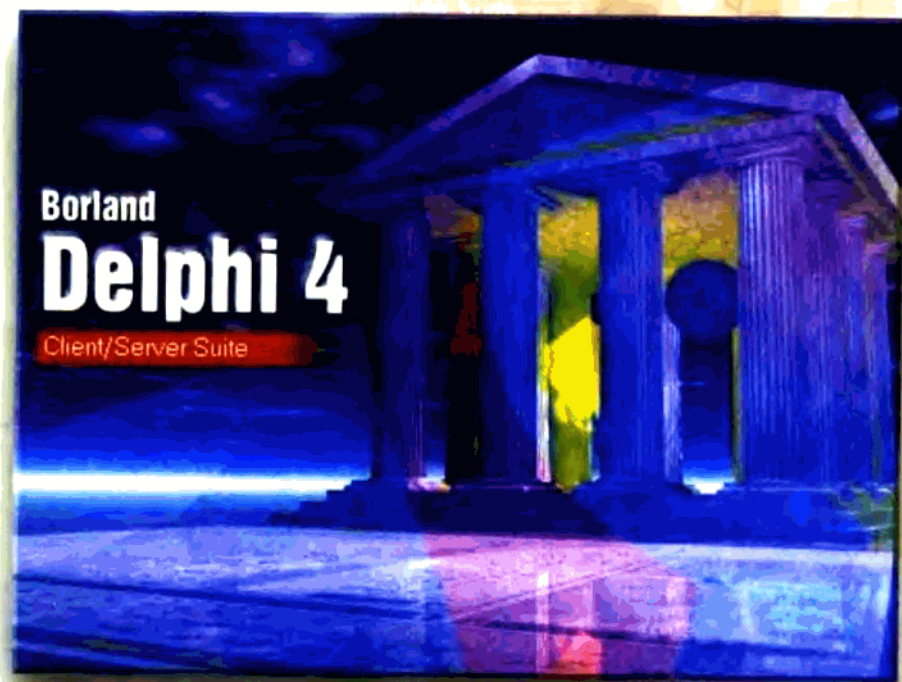


Delphi 4 高级编程丛书之三

# Database 和 MIDAS 编程技术

徐新华 编著



人民邮电出版社  
PEOPLE'S POSTS &  
TELECOMMUNICATIONS  
PUBLISHING HOUSE

**本**书是《Delphi 4 高级编程丛书》的第三册，书中全面深入地介绍了 Delphi 4 在数据库和多层 Client/Server 领域的编程技术。

本书层次清晰，实例丰富，语言通俗简洁。全书共分九章，分别介绍了数据集公共基类、怎样建立数据库访问链路、怎样显示数据库的数据、怎样用 QuickReport 制作报表、怎样制作 TeeChart 图表、怎样自定义数据集、怎样使用数据库浏览器等内容。本书的重点是第八章，详细讲述了 Delphi 4 的 MIDAS 编程技术，这也是 Delphi 4 的核心部分，读者在阅读本章内容时，可以要理清类的继承关系。

本书既可以作为广大读者学习 Delphi 4 的指导书，也可以作为程序开发设计人员的编程参考手册。

**D**elphi 4 是 Borland 公司更名为 Inprise 后推出的一个具有战略意义的产品，它标志着 Inprise 的工作重心已经从桌面转移到跨平台的分布式应用。Delphi 4 中的 MIDAS 技术奠定了它在业界的领先地位。

为了帮助广大用户全面、准确地掌握 Delphi 4 的编程思想和用法，我们编写了这套《Delphi 4 高级编程丛书》，主要是针对那些已初步掌握了 Delphi 4 的基本用法而又需要进一步提高和精通的读者。本书紧紧把握 Delphi 4 的基本特征即面向对象，重点从“类”这一层次把 Delphi 4 的编程思想讲透。我们的体会是，只要深刻领会了面向对象的编程思想，就很容易理解那些看上去高深莫测的领域，如 COM、ActiveX、CORBA、MIDAS。

本套丛书分为四册，第一册是《IDE 和 Object Pascal 语言》，第二册是《GUI 编程技术》，第三册是《Database 和 MIDAS 编程技术》，第四册是《COM、ActiveX、CORBA、Internet/Intranet 编程技术》。

本书是此套丛书的第三册，主要介绍 Delphi 4 在数据库和多层 Client/Server 领域的编程技术，各章内容如下：

第一章详细介绍了 Delphi 4 的几个数据集公共基类，包括 TDataSet、TBDEDataSet、TDBDataSet、TField、TFieldDef，另外还介绍了字段编辑器。

第二章介绍怎样建立数据库访问链路，包括表、查询、数据源、存储过程、连接数据库、BDE 会话期、批量移动数据、缓存更新、嵌套表等内容。

第三章详细介绍了几个数据控件，包括 TDBGrid、TDBNavigator、TDBText、TDBEdit、TDBMemo、TDBImage、TDBListBox、TDBComboBox、TDBCheckBox、

TDBRadioGroup、TDBLookupListBox、TDBLookupComboBox、TDBRichEdit、TDBCtrlGrid 等。其中，TDBGrid 支持 Oracle8 的对象字段和嵌套表。

第四章介绍怎样制作 QuickReport 报表。

第五章介绍怎样制作 TeeChart 图表。

第六章介绍怎样编写决策支持程序，包括引入数据集、建立数据仓库、决策源、数据透视表、决策栅格等内容。

第七章介绍怎样自定义数据集，初学者可以不看这部分内容。

第八章是本书的核心，详细介绍 Delphi 4 的 MIDAS 编程技术。首先讲述怎样创建应用服务器，然后具体介绍了应用服务器上的两个关键部件：远程数据模块和 TDataSetProvider 或 TProvider。接着介绍了“瘦”客户，这也是本章重点要讲述的内容。接下来，按照类的继承关系依次介绍了几个 MIDAS 连接元件，如 TDCOMConnection、TOLEnterpriseConnection、TCorbaConnection、TSocketConnection。本章还介绍了 TSimpleObjectBroker，结合应用服务器的冗余配置，可以保证关键业务 7×24 小时不中断地运行。最后，详细介绍了 TClientDataSet 元件，这是建立“瘦”客户程序的关键。

第九章介绍了一个很有用的工具数据库浏览器。

本书由徐新华执笔，参加编写的人员有郭平、周学成、徐存聪等人。

由于水平有限，再加上时间很紧，尽管我们作了严格的审核和测试，书中可能还是难免有一些错误，敬请广大读者不吝赐教，我们谨在此表示感谢。

考虑到 Delphi 4 中增加了许多崭新的技术，有些技术具有相当的难度，为了帮助广大程序员更好地掌握这个优秀的开发工具，我们愿意为购买此书的读者提供技术咨询。我们将热情、及时地答复大家提出的问题。

电子函件：p\_inprise@mail.263.net.cn

作者  
1998 年 8 月

## 第一章

### 数据集

1.1 TDataSet .....	2
1.2 TBDEDataSet.....	20
1.3 TDBDataSet.....	25
1.4 TField .....	27
1.4.1 具体的字段对象.....	27
1.4.2 TField的特性、方法和事件.....	28
1.5 TFieldDef.....	39
1.6 字段编辑器.....	40

## 第二章

### 建立数据库访问链路

2.1 访问数据库表.....	45
2.1.1 访问数据库表的一般步骤.....	45
2.1.2 TTable的特性、方法和事件.....	46
2.1.3 Master/Detail 关系的一个示例.....	57
2.1.4 TIndexDef.....	58
2.1.5 TCheckConstraint.....	59
2.2 对数据库查询.....	60
2.2.1 查询数据库的一般步骤.....	60
2.2.2 TQuery的特性和方法.....	61
2.2.3 TParam对象.....	68
2.2.4 TParams.....	70
2.2.5 SQL Builder.....	73
2.3 数据源.....	74
2.4 存储过程.....	77
2.4.1 使用 TStoredProc的一般步骤.....	77
2.4.2 存储过程的参数.....	78

2.4.3 TStoredProc 的特性、方法和事件 .....	79
2.5 连接数据库 .....	81
2.6 BDE 会话期 .....	88
2.6.1 TSession 的特性、方法和事件 .....	88
2.6.2 TSessionList .....	98
2.6.3 动态创建 TDatabase 和 TSession 的例子 .....	99
2.7 批量移动数据 .....	100
2.8 缓存更新 .....	104
2.9 访问嵌套表 .....	107
2.10 数据模块 .....	107
2.10.1 为什么要使用数据模块 .....	108
2.10.2 怎样把数据模块加到工程中 .....	108
2.10.3 数据模块上的快捷菜单 .....	109
2.10.4 给数据模块命名 .....	109
2.10.5 重用数据模块 .....	110

### 第三章

#### 使用数据控件

3.1 显示和编辑数据的一般步骤 .....	111
3.2 TDBGrid .....	112
3.2.1 TDBGrid 的特性、方法和事件 .....	113
3.2.2 TDBGridColumns .....	117
3.2.3 在设计期设置列的属性 .....	118
3.2.4 在运行期操纵列的属性 .....	119
3.3 TDBNavigator .....	122
3.4 TDBText .....	126
3.5 TDBEdit .....	127
3.6 TDBMemo .....	127
3.7 TDBImage .....	128
3.8 TDBListBox .....	130
3.9 TDBComboBox .....	131
3.10 TDBCheckBox .....	131
3.11 TDBRadioGroup .....	132
3.12 TDBLookupListBox .....	133
3.13 TDBLookupComboBox .....	134
3.14 TDBRichEdit .....	136
3.15 TDBCtrlGrid .....	136

## 第四章

### 用 Quick Report 制作报表

4.1 QuickReport 概述 .....	141
4.2 建立报表的一般步骤 .....	142
4.2.1 一个最简单的报表 .....	142
4.2.2 一个稍复杂的报表 .....	143
4.2.3 基于自定义的文本建立报表 .....	143
4.2.4 自定义预览窗口 .....	143
4.3 TQuickRep .....	144
4.4 TQRSubDetail .....	151
4.5 TQRBand .....	153
4.6 TQRChildBand .....	155
4.7 TQRGroup .....	155
4.8 TQRLabel .....	156
4.9 TQRDBText .....	156
4.10 TQRExpr .....	156
4.11 TQRSysData .....	158
4.12 TQRMemo .....	158
4.13 TQRRichText .....	159
4.14 TQRDBRichText .....	159
4.15 TQRShape .....	159
4.16 TQRImage、TQRDBImage .....	160
4.17 TQRCompositeReport .....	160
4.18 TQRPreview .....	160
4.19 TQRPrinter .....	161

## 第五章

### TeeChart 图表

5.1 制作 TeeChart 图表的一般步骤 .....	165
5.2 TeeChart 向导 .....	167
5.3 图表编辑器 .....	169
5.4 怎样引出图表 .....	169
5.5 预览和打印图表 .....	170
5.6 创建数据库图表的一般步骤 .....	171
5.7 在 QuickReport 报表上创建图表的一般步骤 .....	172
5.8 创建决策图表的一般步骤 .....	173

## 第六章

### 决策方

6.1 使用决策支持元件的一般步骤 .....	176
-------------------------	-----

6.2 引入数据集 .....	176
6.3 建立数据仓库.....	178
6.3.1 TDecisionCube 的特性、方法和事件 .....	178
6.3.2 决策方编辑器 .....	183
6.4 决策源.....	185
6.5 数据透视表 .....	192
6.6 决策栅格.....	194

## 第七章

### 自定义数据集

7.1 创建自定义数据集 .....	199
7.2 打开、初始化和关闭数据集.....	202
7.3 书签管理.....	204
7.4 记录管理.....	205
7.5 字段管理.....	208
7.6 记录导航.....	211
7.7 异常处理.....	212
7.8 建立 FieldDefs 列表.....	212
7.9 注册数据集元件 .....	212

## 第八章

### MIDAS 技术

8.1 应用服务器 .....	216
8.1.1 创建应用服务器的一般步骤 .....	216
8.1.2 与“瘦”客户连接 .....	218
8.2 TDataSetProvider.....	219
8.3 TProvider.....	226
8.4 “瘦”客户.....	227
8.4.1 创建“瘦”客户的一般步骤 .....	227
8.4.2 与应用服务器连接 .....	228
8.5 TCustomRemoteServer .....	229
8.6 TDispatchConnection.....	230
8.7 TCOMConnection .....	232
8.8 TDCOMConnection.....	232
8.9 TOLEnterpriseConnection .....	232
8.10 TCorbaConnection .....	233
8.11 TSocketConnection.....	234
8.12 TRemoteServer 和 TMIDASConnection .....	235
8.13 TSimpleObjectBroker.....	235
8.14 TClientDataSet.....	238



8.15 “公文包”模式 .....	251
8.16 把“瘦”客户程序作为 ActiveForm 发布 .....	251

## 第九章

### 数据库浏览器

9.1 数据库浏览器的窗口 .....	253
9.2 建立和维护数据库别名 .....	254
9.3 信息窗格 .....	255
9.4 访问数据库表 .....	258
9.5 数据字典 .....	258

# 第一章

## 数据集

数据集是一个抽象而又具体的概念。说它抽象，是因为它并不直接描述数据库的任何记录和字段。说它具体，是因为在 Delphi 4 中数据集有三种确切的表现形式：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，而 TDBDataSet 的上级是 TDataSet，TDataSet 又是从 TDataSet 继承下来的。

Delphi 4 引入了分布式数据集的概念，并且用 TClientDataSet 来实现和操纵分布式数据集，而 TClientDataSet 也是从 TDataSet 继承下来的。

由此可见，不管是传统的基于 BDE 的数据集，还是分布式数据集，TDataSet 是它们的共同基类。它们之间的继承关系如图 1.1 所示。

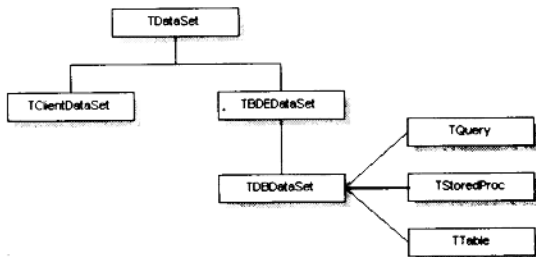


图 1.1 数据集的继承关系

## 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 Active: Boolean;

如果这个特性设为 True，相当于调用 Open 打开数据集；如果这个特性设为 False，相当于调用 Close 关闭数据集。

如果修改了数据集的某个记录后，在还没有 Post 之前就把 Active 特性设为 False，修改将作废。因此，建议您响应 BeforeClose 事件，在数据集变成非活动状态之前使修改有效。

```
Procedure TForm1.Button1Click(Sender: TObject);
```

```
Begin
```

```
    Table1.Active := False;
```

```
    Try
```

```
        Table1.DatabaseName := 'Delphi_Demos';
```

```
        Table1.Active := True;
```

```
    Except
```

```
        On EDatabaseError Do
```

```
            Table1.DatabaseName := 'c:\delphi\demos\database';
```

```
            Table1.Active := True;
```

```
    End;
```

```
End;
```

### AutoCalcFields 特性

声明：Property AutoCalcFields: Boolean;

当程序从数据集中检索记录时，将触发 OnCalcFields 事件。如果 AutoCalcFields 特性设为 True，当程序修改了数据集的某个字段时，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，可能变成死循环。

### BOF 特性

声明：Property BOF: Boolean;

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

```

Procedure TForm1.Table1DataChange(Sender: TObject; Field: TField);
Begin
    If Table1.BOF then
        CopyData.Enabled := False
    Else
        CopyData.Enabled := True;
End;

```

### BlockReadSize 特性

声明: Property BlockReadSize: Integer;

如果这个特性设为 0, 程序每调用一次 Next, 相应的数据控件将刷新一次。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, 当程序调用 Next 若干次后, 相应的数据控件才刷新一次。

### Bookmark 特性

声明: Property Bookmark: TBookmarkStr;

书签用于标记数据集的某个记录, 以后可以快速地回到书签所标记的记录。

如果读这个特性的话, 这个特性返回当前记录的书签(如果有的话)。如果写这个特性, 通常用于指定一个已有的书签, 书签标注的记录就变成当前记录。

### CanModify 特性

声明: Property CanModify: Boolean;

如果这个只读的特性返回 False, 表示数据集的数据不能修改。不过, 即使 CanModify 特性返回 True, 也并不意味着数据集一定能修改, 因为 SQL 表还有个访问权限的问题。

千万不要把 ReadOnly 特性与 CanModify 特性混淆起来。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读写权限后, CanModify 特性才返回 True。

### DefaultFields 特性

声明: Property DefaultFields Boolean;

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

### EOF 特性

声明: Property EOF: Boolean;

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

```

With CustTable Do
Begin
    DisableControls;

```

```

Try
  First;
  While not EOF Do
  Begin
    { Process each record here }
    ...
    Next;
  End;
Finally
  EnableControls;
End;
End;

```

### FieldCount 特性

声明: Property FieldCount: Integer;

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数据集物理存储时的字段数可能不同。程序示例如下:

```

Procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  Info: String;
Begin
  Info := 'The fields of table ' + Table1.TableName + ' are:#13#10#13#10;
  For I := 0 to Table1.FieldCount - 1 Do
    Info := Info + Table1.Fields[I].FieldName + #13#10;
  ShowMessage(Info);
End;

```

### FieldDefs 特性

声明: Property FieldDefs: TFieldDefs;

Delphi 4 用 TFieldDef 对象来定义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 特性返回的 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者字段时才需要给出字段的定义。下面的程序示例创建了一个表:

```

If not Table1.Exists then
Begin
  With Table1 Do
  Begin
    Active := False;

```

```

DatabaseName := 'DBDEMOS';
TableType := ttParadox;
TableName := 'CustInfo';

With FieldDefs do
Begin
    Clear;
    Add('Field1', ftInteger, 0, True);
    Add('Field2', ftString, 30, False);
End;
With IndexDefs do
Begin
    Clear;
    Add(", 'Field1', [ixPrimary, ixUnique]);
    Add('Fld2Idx', 'Field2', [ixCaseInsensitive]);
End;

CreateTable;
End;
End;

```

### Fields 特性

声明: Property Fields[Index: Integer]: TField;

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。

一般来说, 要访问字段的值最好通过 FieldValues 特性, 不过, 如果知道字段的数据类型的话, 也可以通过 Fields 特性来访问字段的值, 例如:

```
Edit1.Text := CustTable.Fields[6].AsString;
```

上面这行代码用于读字段的值。如果要对字段赋值, 可以参考下面的例子:

```
CustTable.Edit;
```

```
CustTable.Fields[6].AsString := Edit1.Text;
```

```
CustTable.Post;
```

### FieldValues 特性

声明: Property FieldValues[const FieldName: String]: Variant; default;

这个特性可以通过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, 因此上述程序可以简化为:

```
Customers.Edit;
Customers['CustNo'] := Edit1.Text;
Customers.Post;
```

由于 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 Filtered: Boolean;

如果这个特性设为 True，表示将对数据集中的记录进行过滤，过滤条件由 Filter 特性设置，过滤时将触发 OnFilterRecord 事件；如果这个特性设为 False，表示不进行过滤。

为什么要使用过滤？因为程序关心的往往只是数据集中满足特定条件的部分记录，这就需要把不满足条件的记录过滤掉。要过滤记录，有两种方法：一是在运行期调用 Locate 过程寻找匹配的记录，二是把 Filtered 特性设为 True。这样，数据集的每条记录都会产生 OnFilterRecord 事件，在处理 OnFilterRecord 事件的句柄中，把 Accept 参数设为 False 表示过滤掉该记录。例如，要只显示 State 字段的值是“CA”的记录，程序示例如下：

```
Procedure TForm1.Table1FilterRecord(DataSet: TDataSet; Var Accept: Boolean);
Begin
    Accept := DataSet['State'] = 'CA';
End;
```

在运行期，可以把 Filtered 特性设为 False 从而使 OnFilterRecord 事件不再触发，也就是说不进行过滤。不过，如果要显示所有的记录，程序必须显式地调用 Refresh。

可以在运行期给 OnFilterRecord 事件重新指定一个事件句柄，从而使程序按不同的过滤条件对记录进行过滤。程序示例如下：

```
Table1.OnFilterRecord := AnotherFilterRecord;
Refresh;
```

最后要说明的是，尽管使用过滤能够只显示一部分记录，但如果数据集中记录很多，最好还是使用查询或者设置范围。

### Filter 特性

声明：Property Filter: String;

这个特性用于设置过滤条件。过滤有点类似于查询，用于从数据集中选取一些满足特定条件的记录，当然，查询的功能要强大得多。

Filter 特性是字符串，其格式非常相似于 SQL 语句的 WHERE 部分，可以使用比较操作符包括 <、>、>=、<=、=、<> 等。程序示例如下：

```
OrderNum >= 16;
```

表示只选取 OrderNum 字段的值大于等于 16 的记录。

如果需要指定多重条件，可以使用关系操作符包括 AND、NOT、OR 等把多个条件连接起来。程序示例如下：

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

表示只选取 OrderNum 字段的值大于等于 16 并且 Sex 字段为“Man”的记录。

如果字段的名称本身含有空格，要用一对方括号把字段名括起来，示例如下：

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

### FilterOptions 特性

声明：Property FilterOptions: TFilterOptions;

这个特性设置过滤的选项，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，不允许部分匹配。

### Found 特性

声明：Property Found: Boolean;

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

### Modified 特性

声明：Property Modified: Boolean;

这是个运行期只读的特性，如果返回 True 表示当前记录中某个字段被修改了但还没有被写到数据集中。当程序调用 Cancel 或 Post 后，这个特性又恢复为 False。程序示例如下：

```
If Table1.Modified Then
```

```
Begin
```

```
    If MessageDlg('要保存当前记录吗?', mtConfirmation, [mbYes,mbNo], 0) = mrYes
```

```
        Then Table1.Post
```

```
        Else Table1.Cancel;
```

```
End;
```

### NestedDataSets 特性

声明：Property NestedDataSets: TList;

Delphi 4 支持 Oracle8 的嵌套表功能。这个特性返回一个列表，该列表列出了数据集的所有嵌套数据集，也称子数据集。

### ObjectView 特性

声明：Property ObjectView: Boolean;

如果这个特性设为 False，数据集中的字段按照它们在 Fields 数组中的顺序存储。如果这个特性设为 True，数据集中的字段按照字段之间的继承关系以树状存储。

### RecordCount 特性

声明：Property RecordCount: Longint;

这个只读的特性返回数据集的记录数。注意：RecordCount 特性返回的记录数与数据集物理存储的记录数可能不同，因为有可能使用了过滤。不过，对于 dBASE 表来说，RecordCount 特性返回的总是数据集物理存储的全部记录数，即使调用了 SetRange。



下面用一个百分数来显示处理数据集记录的进度。程序示例如下：

```
Var I : Integer;  
I := 1;  
With Table1 Do  
Begin  
    First;  
    While Not EOF Do  
    Begin  
        StatusBar1.Panels[0].Text := IntToStr((I*100) div RecordCount);  
        Inc(I);  
        Next;  
    End;  
End;
```

### RecordSize 特性

声明：Property RecordSize: Longint;

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)。

### State 特性

声明：Property State: TDataSetState;

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。

### ActiveBuffer 函数

声明：Function ActiveBuffer: PChar;

这个函数返回指向当前记录缓冲区的指针。

### Append 过程

声明：Procedure Append;

这个过程在数据集的末尾添加一个新的空记录。在调用这个过程之前，最好先访问 CanModify 特性，当 CanModify 特性返回 True 时再调用 Append 过程。

调用了这个过程以后，还必须调用 Post 过程才真正把一条新记录插入到数据集中。如果调用 Cancel 过程将取消刚才的操作。程序示例如下：

```
Table1.Append;
```