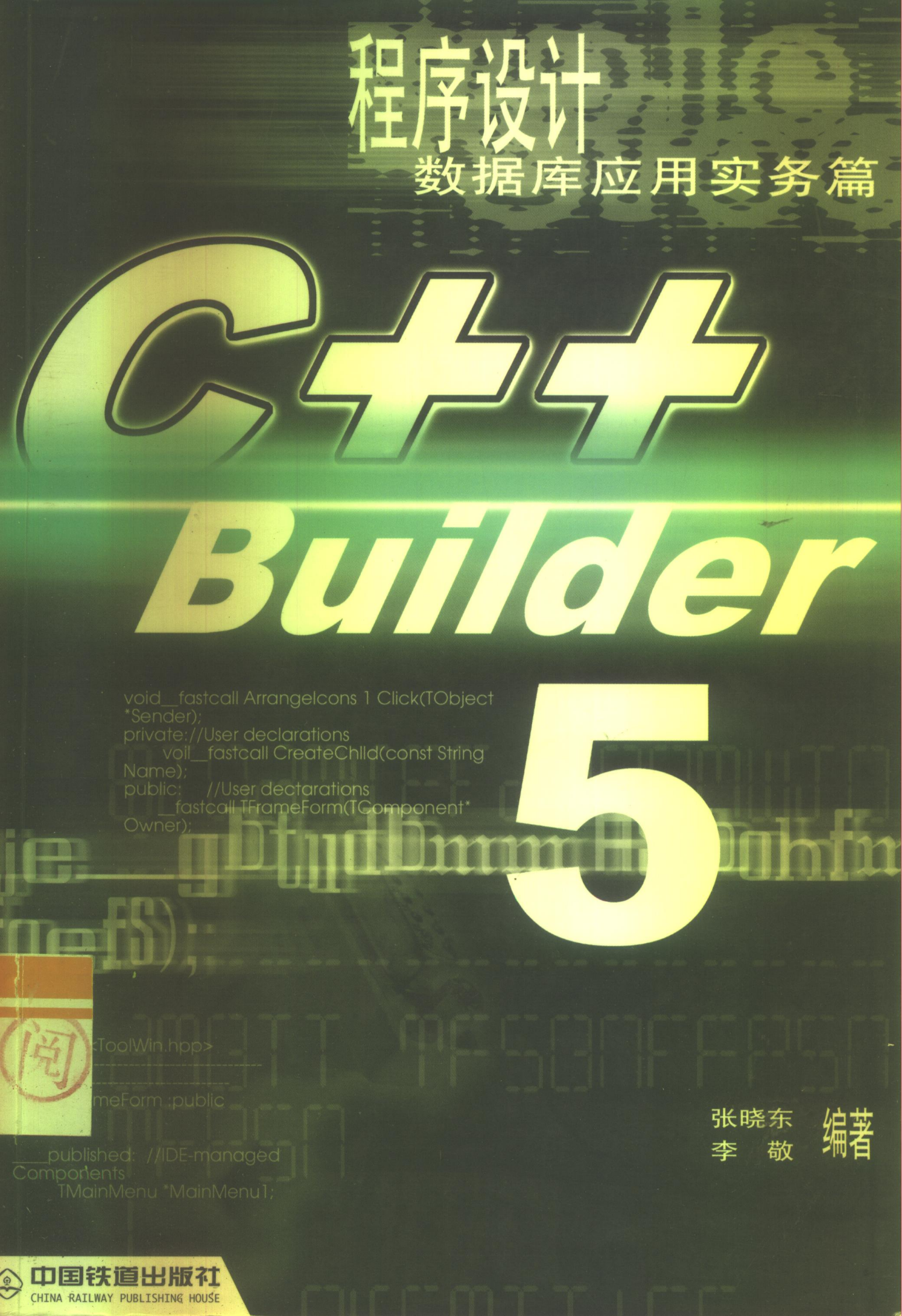
程序设计
数据库应用实务篇

C++ Builder 5



张晓东
李 敬 编著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

C++ Builder 5 程序设计

数据库应用实务篇

张晓东　李敬　编著

C++ Builder 5

中国铁道出版社

（京）新登字 063 号

内 容 简 介

C++ Builder 是 Inprise 公司推出的开发数据库的强有力工具。本书全面深入地介绍了 C++ Builder 的数据库开发技术，内容包括：数据库应用程序的体系结构，数据库工具，数据访问组件，数据控制组件，SQL 和 TQuery 组件，使用 Quick Report 创建报表，TeeChart 图表，多层 Client/Server 应用程序，MIDAS，决策支持组件以及 ADO 组件。

本书内容丰富，语言简洁，主要面向中高级 C++ Builder 程序员，也可以供具有一定编程基础的读者学习和参考。

图书在版编目（CIP）数据

C++ Builder5 程序设计. 数据库应用实务篇/张晓东等编著. —北京：中国铁道出版社，2000.9
ISBN 7-113-03285-0

Ⅰ.C… Ⅱ.张… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 43496 号

书　　名：C++ Builder5 程序设计——数据库应用实务篇
作　　者：张晓东　李 敬
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：严晓舟
特邀编辑：彭作文
封面设计：冯龙彬
印　　刷：北京兴顺印刷厂
开　　本：787×1092　1/16　印张：23.5　字数：620 千
版　　本：2000 年 9 月第 1 版　2000 年 9 月第 1 次印刷
印　　数：1～5000 册
书　　号：ISBN 7-113-03285-0 /TP・354
定　　价：38.00 元

前　言

C++ Builder 5 是 Inprise 公司推出的基于 C++语言的可视化开发平台，它不仅提供了新颖的可视化设计工具，还配有数据库引擎、可以通过 SQL 连接或 ODBC 访问多种数据库，并提供了强大的开发基于 Client/Server 模式的数据库应用程序的能力。

C++ Builder 从推出 1.0 版以来，一直受到广大用户和程序员的喜爱。当 Inprise 公司在 2000 年 1 月 26 日一经推出 C++ Builder 5，立刻就受到用户的一片好评。新推出的 C++ Builder 5 在数据库和 Internet 应用程序开发等方面的性能都有了很大的提高。

在 C++ Builder 5 中，原先支持多层数据库应用程序的结构现在又支持无界的远程数据模块，新增加的 InternetExpress 组件可以让用户很容易地建立起基于 Web 的应用程序，并且该程序中的浏览器可以与 MIDAS 应用服务器交换数据。

C++ Builder 5 还重新设计了编译器，将编译置成后台运行，这样大大提高了开发效率以及应用程序的性能。同时，还优化了 IDE，允许用户定制并保存自己的环境设置，有效地解决了多人共享的问题。

C++ Builder 5 支持 XML，从而简化了数据分布，优化了数据交换。C++ Builder 5 中增加了对 ADO（ActiveX Data Objects）的支持，用户可以迅速实现对终端用户做商业决策的数据的一致性访问。通过 ADO，用户还可以快速访问关系型或非关系型数据库以及 E-mail 和文件系统。

InterBase Express(IBX)组件可以通过 InterBase API 访问本地的 InterBase 数据库，因为 API 可以直接被组件使用，而 BDE 不需要。利用 InterBase Express，系统集成商和独立软件商可以快速地开发出高性能的应用系统。将 InterBase Express 与 C++ Builder 5 结合，可以在只需少量维护工作而且只占用很少系统资源的数据库中进行高效率、高性能的开发工作。

鉴于 C++ Builder 5 卓越的性能和广大用户的要求，我们在有着丰富 C++ Builder 数据库开发经验的基础上编写了这本书。由于本书不是 C++ Builder 的入门书籍，有关语言等其它方面的内容请读者自行参看相关手册。

有关程序源代码的下载地址：

http://www.bookposter.com/download.htm

本书由张晓东、李敬、芦丽丹、乔林、林杜等合力编写，他们都十分熟悉 C++ Builder 程序设计和数据库开发，并有着丰富的数据库开发经验。限于作者水平，缺点错误在所难免，敬请广大读者对本书提出批评和建议。

作者

2000 年 8 月

目 录

1 建立数据库应用程序

C++ Builder 是 Inprise 公司推出的开发数据库应用程序的强有力工具。它不仅允许用户创建能访问 dBASE、Paradox 和 Local InterBase 服务器的数据库应用程序，还支持具有 ODBC 接口的数据库系统。使用 C++ Builder，用户可以很方便地开发出基于大型数据库系统的应用程序。

本章我们主要介绍一下数据库的基础知识和有关数据集的概念，并通过一个简单的实例讲一下如何创建数据库应用程序。

1.1 数据库基础知识

数据库主要由三个部分组成：数据库，即按一定结构组织在一起的相关数据的集合；数据库管理系统（DBMS），专门负责组织和管理数据信息的程序；以及数据库应用程序，使我们能获取、显示和更新由 DBMS 存储的数据。

一、数据库

1. 本地数据库

本地数据库位于本地磁盘或局域网上。如果有几个用户同时访问数据库，本地数据库采取基于文件的锁定策略，因此，本地数据库又叫基于文件的数据库。本地数据库往往与数据库应用程序在同一个系统中，因此访问本地数据库的速度比访问远程数据库的速度要快。但本地数据库所能存储的数据没有远程数据库所能存储的数据多，在选择使用本地数据库还是远程数据库时必须考虑到这一点。

使用本地数据库的应用程序也称为单层应用程序，因为数据库和应用程序都在同一个文件系统中。典型的本地数据库有 Paradox，dBASE，FoxPro 和 Access。

2. 远程数据库

远程数据库通常位于远程计算机上，用户通过 SQL(Structured QueryLanguage，结构化查询语言)来访问远程数据库中的数据。因此远程数据库有时候也称 SQL 服务器或者 RDBMS(Remote Database Management System，远程数据库管理系统)。远程数据库非常适合于几个用户同时访问。远程数据库提供基于事务的多用户支持，而且存储的数据也比本地数

据库多得多。有时数据不只在一个服务器上，而是分布在几个服务器上。

使用远程数据库的应用程序称为两层或多层应用程序，因为数据库和应用程序位于彼此独立的系统(层)中。典型的远程数据库有 Interbase，Oracle，Sybase，Informix，Microsoft SQL Server 和 DB2。

二、数据库管理系统

数据库管理系统是数据库系统的核心部分，它建立在操作系统的基础上，对数据库进行统一的管理和控制。

DBMS 的主要功能有：

- 描述数据库：描述数据库的逻辑结构、存储结构、语义信息和保密要求等。
- 管理数据库：控制整个数据库系统的运行，控制用户的并发性访问，检验数据的安全、保密与完整性，执行数据检索、插入、删除和修改等操作。
- 维护数据库：控制数据库初始数据的装入，记录工作日志，监视数据库的性能，修改更新数据库，重新组织数据库以及恢复出现故障的数据库。
- 数据通信：组织数据的传输。

数据库管理系统主要有四种类型：文件管理系统，层次数据库系统，网状数据库系统和关系数据库系统。目前最流行、应用最广泛的是关系数据库系统。关系数据库以行和列的形式来组织信息。一个关系数据库由若干表组成。一个表就是一组相关数据按行排列，像一张表格一样。表中的每一列称为一个字段。每一个字段都有相应的描述信息，如数据类型、数据宽度等。表中的每一行称为一条记录。为了加快访问数据库的速度，很多数据库都使用索引。

三、数据库应用程序

数据库应用程序是一个允许用户插入、修改、删除并报告数据库中数据的计算机程序。传统的数据库应用程序是由程序员用一种或多种通用或专用的程序设计语言编写的，近年来出现了多种面向用户的数据库应用程序开发工具，这些工具可以简化使用 DBMS 的过程，不需要专门编程。C++ Builder 就是一种强有力的数据库应用程序开发工具。

1.2 C++ Builder 数据库组件

一个数据库应用程序通常要包含三个基本的数据库组件：数据集组件，负责与 BDE 的联系；数据源组件，连接数据集组件和用户界面的桥梁；数据控制组件，为用户提供浏览和编辑数据的界面。

一、数据访问组件

数据访问组件位于组件选项板的“Data Access”页上，如图 1-1 所示。

图 1-1 “Data Access”选项卡上的数据访问组件

数据访问组件用于直接访问数据库中的数据库表，它为数据控制组件和数据源提供了一条通道。数据访问组件在程序运行时是不可见的。表 1.1 列出了数据访问组件的主要功能。

表 1.1 数据访问组件

组　件	主 要 功 能
TDataSouce	作为数据集组件 TTable 和 TQuery 与数据浏览组件 TDBGrid 和 TDBEdit 之间传送数据的通道
TTable	存取磁盘上数据库表的媒介，它通过 BDE 存取数据库表中的数据，数据浏览组件通过 TDataSouce 从 TTable 中访问数据并显示和编辑其中的数据
TQuery	利用 SQL 语言访问数据库表中的数据，通过 TDataSouce 实现数据浏览组件对数据库的访问
TStoredProc	用来访问远程服务器中的存储过程
TDatabase	用来建立应用程序与要登录的远程服务器上的数据库的永久性连接
TSession	用来控制数据库应用程序和数据库的连接
TBatchMove	用来复制数据库表的结构或表中的记录
TUpdateSQL	根据预定义的 SQL 语句来修正数据集
TNestedTable	通过 TDataSource 组件存取嵌套数据集字段中的数据并提供给数据浏览组件

二、数据控制组件

数据控制组件位于组件选项板的“Data Controls”页上，如图 1-2 所示。

图 1-2 “Data Controls”选项卡上的数据控制组件

数据控制组件主要用于设计用户界面，对数据库中的数据进行浏览、编辑、插入和删除等操作。数据控制组件又称为数据浏览组件，它其实是在 Standard 页上的标准组件的基础上，相应地增加了数据浏览功能，使得它们能显示和编辑数据库中的数据。数据控制组件不仅能显示数据库中的数据，还能将其自身经过修改的数据写入数据库中。表 1.2 列出了数据控制组件的主要功能。

表 1.2 数据控制组件的主要功能

组　件	主 要 功 能
TDBGrid	浏览数据库中数据的网络，它以网格的方式显示数据库中的数据，在网格中可以对数据进行编辑。利用 Fields Editor 可以对数据库表中字段的显示格式、显示顺序及是否显示等进行控制
TDBNavigator	用来向前或向后移动记录指针，可以对单条记录进行编辑，还可以用来插入、删除记录以及刷新显示和取消上一次操作
TDBText	用来显示数据库中数据的文本框，但它只能显示数据库表中当前记录的字段值，用户不能对数据进行修改

续上表

组　件	主 要 功 能
TDBEdit	用来显示和编辑数据库表中数据的编辑框，可以显示和编辑数据库表中当前记录的字段值
TDBCheckBox	浏览数据库中数据的检查框，可以用来显示和编辑数据库中布尔型字段的字段值
TDBListBox	浏览数据库中数据的列表框，可以用一个列表框来显示数据库表中一个字段的值
TDBComboBox	浏览数据库中数据的组合框，可以用一个组合框来显示数据库表中一个字段的值
TDBRadioGroup	浏览数据库中数据的单选框，用一组单选钮来确定显示数据库表中哪一个字段
TDBMemo	用来浏览数据库中备注型的字段，可以显示数据库表中当前记录中的 BLOB 型字段
TDBImage	用来浏览数据库中数据的图像框，可以显示、拷贝、粘贴数据库表中图像类型的字段
TDBLookUpList	浏览数据库表中数据的列表框，在基于一个数据库表的应用中，用它可以显示另一个数据库表中一个指定的字段值
TDBLookUpCombo	浏览数据库表中数据的组合框，在基于一个数据库表的应用中，用它可以显示另一个数据库表中一个指定的字段值
TDBCtrlGrid	用来显示和编辑数据库表中查询的网格。与 TDBGrid 组件不同，它不是在一行上显示一条记录，而是允许数据集中数据自由布局，每一条记录都显示在各自的面板上
TDBRichEdit	用来显示和编辑数据库表中数据的编辑框，可以多行文本来显示和编辑 RTF 格式的备注类型字段的内容
TDBChart	以图表的形式显示数据库中的数据，用法与 TChart 组件相似

数据控制组件在设计时具有数据感知的特点，如果正确设置 DataSource 属性指定一个数据源，用户可以马上就能看到数据，不需要编译和运行程序。数据控制组件都是可见的，也就是说，如果用户修改了这些组件的属性，在窗体上马上就能反映出来。

三、数据集组件

TTable、TQuery、TSoredProc 和 TClientDataSet 组件负责与实际的数据库表联系，并从中获取数据信息，因而常常被称为数据集组件。数据集组件在设计时是可见的，但在运行时是不可见的。它们通过 BDE 为应用程序提供与数据库的连接。

这四个数据集组件是 C++ Builder 中四种类型的标准数据集组件，它们都是从一个共同的基类 TDataSet 继承来的。其中，TClientDataSet 是直接从 TDataSet 继承来的，TTable、TQuery 和 TSoredProc 是从 TDBDataSet 继承来的，TDBDataSet 又从 TBDEDataSet 继承而来，TBDEDataSet 才是直接继承于 TDataSet。它们的继承关系如图 1-3 所示。

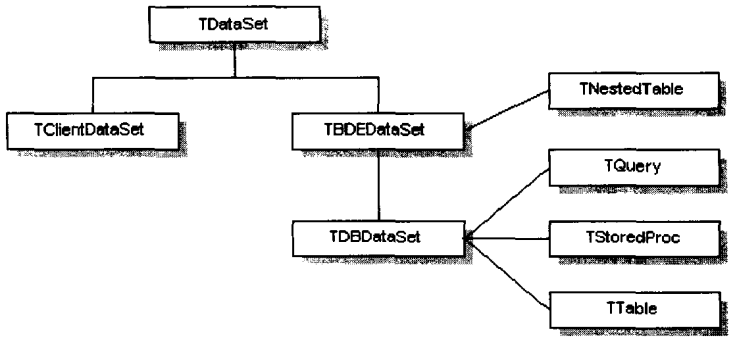


图 1-3 数据集组件的继承关系

数据集组件 TDataSet 是不可见组件，它是所有数据集的抽象基类，它的大部分属性和方法是抽象或虚拟的。所谓抽象，是指这些方法只有声明而没有定义，派生类必须给出定义后才能调用这些方法，不同的派生类可以有不同的定义。所谓虚拟，是指这些方法可以被派生类重载。用户不能直接创建 TDataSet 的实例，否则会引起运行期错误。

1. 数据集组件的几种状态

数据集组件的状态(State 属性)决定了当前能够对数据集进行何种操作。表 1.3 列出了数据集组件的状态及其含义。

表 1.3 数据集组件的状态

状　　态	含　　义
非活动状态（dsInactive）	数据集是关闭的，不能访问数据
浏览状态（dsBrowse）	数据集是打开的，只能浏览表中的数据，不能修改或删除记录，缺省情况下数据集打开时即处于该状态
编辑状态（dsEdit）	可以浏览表中的记录并能修改当前的记录或删除记录
插入状态（dsInert）	可以在表中插入一条记录
查找状态（dsSetKey）	在该状态下，可以调用 FindKey、GotoKey、GotoNearest 和 FindNearest 方法来查找表中的记录，此状态只适用于 TTable 组件
处理计算字段状态（dsCalcFields）	在该状态下，当数据集组件的 OnCalcFields 事件被执行时，C++ Builder 会自动地将数据集组件置为该状态，以防止其他非计算字段的值被修改
过滤状态（dsFilter）	正在进行过滤操作，可以查看受限的数据集，但不能编辑和插入记录
当前值状态（dsCurValue）	只能内部使用
新值状态（dsNewValue）	只能内部使用
旧值状态（dsOldValue）	只能内部使用

下面我们讲一下这些状态的使用。

（1）dsInactive 状态

当数据集已关闭时就处于 dsInactive 状态，此时，不能访问它的任何数据。

要使数据集进入 dsInactive 状态，可以把 Active 属性设为 false，或者调用 Close 函数。在数据集将要关闭之前，会触发 BeforeClose 事件。当数据集刚刚关闭，会触发 AfterClose 事件。

如果在数据集处于 dsEdit 或 dsInsert 状态时调用 Close 函数，应当在 BeforeClose 事件处理程序中提示用户是认可还是取消。

程序代码如下：

```
void __fastcall TForm1::VerifyBeforeClose(TDataSet *DataSet)
{
  if (DataSet->State == dsEdit || DataSet->State == dsInsert)
  {
    TMsgDlgButtons btns;
    btns << mbYes << mbNo;
if (MessageDlg("真的要修改吗？", mtConfirmation, btns, 0) == mrYes)
      DataSet->Post();
    else
      DataSet->Cancel();
  }
}
```

（2）dsBrowse 状态

当一个数据集刚刚打开时，数据集就处于 dsBrowse 状态。此时，可以显示数据集中的记录，但不能编辑和插入记录。

dsBrowse 状态可以认为是数据集的基本状态，在此状态下，可以进入其他状态。例如，调用 Insert 或 Append 将使数据集的状态从 dsBrowse 变成 dsInsert。当然，这还取决于其他因素，如 CanModify 属性的值。

下列程序代码使数据集 Table1 进入 dsInert 状态，用户可以输入新记录并把新记录写入数据库。

```
Table1->Insert();
AddressPromptDialog->ShowModal();
if (AddressPromptDialog->ModalResult == mrOK)
  Table1->Post();
else
  Table1->Cancel();
```

使数据集回到 dsBrowse 状态有两种方法：

- Post 函数

  该函数试图把修改了的数据保存到数据集中。如果成功，数据集将回到 dsBrowse 状态；如果没有成功，数据集仍然保持原来的状态。

- Cancel 函数

  该函数取消当前正在进行的编辑、插入、搜索等操作，使数据集回到 dsBrowse 状态。

（3）dsEdit 状态

如果应用程序要修改数据库的数据，必须先进入 dsEdit 状态。

要进入 dsEdit 状态，可以调用 Edit 函数。例如：

```
Table1->Edit();
```

当然，这还取决于 CanModify 属性的值。如果 CanModify 属性返回 true，表示数据集可以读和写。对于 TTable 组件，如果 ReadOnly 属性设为 true，CanModify 属性肯定返回 false。对于 TQuery 组件，如果 RequestLive 属性设为 false，CanModify 属性肯定返回 false。即使数据集进入了 dsEdit 状态，用户也不一定就能修改数据，数据控制组件的 ReadOnly 属性还必须设为 false。对于 SQL 数据库，用户能不能修改数据还取决于用户有没有修改数据的权限。

要从 dsEdit 状态返回到 dsBrowse 状态，可以调用 Cancel()，Post()或 Delete()。当然 Post()和 Delete()要调用成功，否则就仍然保持 dsEdit 状态。数据控制组件中，如果用户修改了数据后把输入焦点移走，就相当于调用了 Post()函数，将会使数据集返回 dsBrowse 状态。

（4）dsInsert 状态

要插入新的记录，必须首先进入 dsInsert 状态。要进入 dsInsert 状态，可以调用 Insert 函数或 Append 函数。

当然 Insert 函数或 Append 函数并不能保证一定能进入 dsInsert 状态，这还取决于 CanModify 属性的值是否返回 true。即使数据集进入了 dsInsert 状态，用户也不一定就能插入记录，数据控制组件的 ReadOnly 属性还必须设为 false。对于 SQL 数据库，用户能不能插入记录还取决于用户有没有修改数据的权限。

要从 dsInsert 状态返回到 dsBrowse 状态，可以调用 Cancel()，Post()或 Delete()。当然 Post()要调用成功，否则就仍然保持 dsInsert 状态。数据控制组件中，当用户修改数据后把输入焦点移走，就相当于调用 Post()函数，将会使数据集返回 dsBrowse 状态。

（5）dsSetKey 状态

Locate 函数和 Lookup 函数可以在数据集中搜索特定的记录。对于 TTable 组件，还可以调用 GotoKey，GotoNearest，FindKey 或 FindNearest 函数在表格中搜索特定的记录。在调用上述方法前，必须首先使数据集进入 dsSetKey 状态。

要使数据集进入 dsSetKey 状态，可以调用 SetKey 函数。调用上述方法后，数据集又回到 dsBrowse 状态。此外，在对数据集进行过滤和设置范围前，也要先进入 dsSetKey 状态。

（6）dsCalcFields 状态

当 OnCalcFields 事件被触发时，就会使数据集进入 dsCalcFields 状态。

在处理 OnCalcFields 事件的句柄中，应当给出“计算字段”的值。在 dsCalcFields 状态下，应用程序不能修改非“计算字段”的值。因为如果其他字段的值发生变化，又将触发 OnCalcFields 事件，从而导致无限循环。

当 OnCalcFields 事件处理完后，数据集又返回 dsBrowse 状态。

（7）dsFilter 状态

当 OnFilterRecord 事件被触发时，就会使数据集进入 dsFilter 状态。该状态下不允许修改

数据集的记录，否则过滤就无法正确执行。

当 OnFilterRecord 事件处理完后，数据集回到 dsBrowse 状态。

（8）dsNewValue，dsOldValue 和 dsCurValue 状态

在允许缓存更新的情况下，当用户修改数据集中的记录时，有可能进入 dsNewValue，dsOldValue 或 dsCurValue 状态。在这三种状态下，可以通过字段(TField 对象)的 NewValue，OldValue 或 CurValue 属性来访问当前的值。

这三种状态都是由 C++ Builder 内部使用的，应用程序无法主动进入这三种状态。

2. 数据集的打开和关闭

（1）数据集的打开

数据集组件虽然直接与数据库表发生联系，但应用程序通过数据集组件访问数据库中的数据时，必须把要访问的数据库表调入内存，这也被称为数据集的打开。

打开数据集有两种方法：

- 通过设置数据集组件的 Active 属性为 true 来打开与数据集相连的数据库表，例如：

```
Table1->Active=true;
```

这种方法也可以在程序运行时期通过程序进行。

- 调用数据集的 Open 函数，例如：

```
Table1->Open();
```

这种方法只能在程序运行时期通过程序进行。

（2）数据集的关闭

同理，关闭数据集也有两种方法：

- 通过设置数据集组件的 Active 属性为 false 来关闭与数据集相连的数据库表，例如：

```
Table1->Active=false;
```

这种方法也可以在程序运行时期通过程序进行。

- 调用数据集的 Close 函数，例如：

```
Table1->Close();
```

这种方法只能在程序运行时期通过程序进行。

3. 数据集的导航

数据集中存放有数据库表中的多条记录，用户常常要将记录定位到特定的记录上，这个定位过程也称为数据库表的导航。C++ Builder 为用户在数据集中定位记录指针提供了下列函数及属性，如表 1.4 所示。

表 1.4 记录定位函数

函　　数	功　　能
First	移动记录指针到表中的第一条记录
Last	移动记录指针到表中的最后一条记录
Prior	移动记录指针到表中的前一条记录
Next	移动记录指针到表中的下一条记录
MoveBy	向前或向后移动记录指针到用户指定的地方

数据控制组件中有一个 TDBNavigator 组件，专门用来浏览记录，它把上述方法用按钮来实现。

TDataSet 还有两个只读的布尔类型的属性可以用来判断当前记录的位置：

- Bof 属性

  如果这个属性返回 true，表示现在已到了数据集的开始位置。

- Eof 属性

  如果这个属性返回 true，表示现在已到了数据集的末尾。

（1）First 和 Last

First 函数能够使数据集的第一条记录成为当前记录，并且把 Bof 属性设为 true。

如果第一条记录是当前记录，First 函数什么也不做。

程序示例如下：

```
Table1->First();
```

Last 函数能够使数据集的最后一条记录成为当前记录，并且把 Eof 属性设为 true。

如果最后一条记录是当前记录，Last 函数什么也不做。

程序示例如下：

```
Table1->Last();
```

TDBNavigator 组件上有两个按钮，分别对应着 First 函数和 Last 函数。

（2）Next 和 Prior

Next 函数能够使下一条记录成为当前记录。

如果当前记录是数据集的最后一条记录，Next 函数就什么也不做。

程序示例如下：

```
Table1->Next();
```

Prior 函数能够使前一条记录成为当前记录。

如果当前记录是数据集的第一条记录，Prior 函数什么也不做。

程序示例如下：

```
Table1->Prior();
```

（2）MoveBy

MoveBy 函数使数据集中的另一条记录成为当前记录，该记录距原先的当前记录若干行。

MoveBy 需要传递一个参数，指定相隔的行数，正数表示往记录编号增大的方向移动，负数表示往记录编号减小的方向移动。

程序示例如下：

```
Table1->MoveBy(-3);
```

MoveBy 返回实际移动的行数，返回值与传递给 MoveBy 的参数有可能不同。

（2）Eof 和 Bof 属性

TDataSet 有两个只读的属性 Eof 和 Bof，分别用于判断是否到达数据集的末尾和开头。在遍历数据集的所有记录时经常要用到这两个属性。

如果 Eof 属性返回 true，表示现在已到数据集的末尾。下列操作会把 Eof 属性设为 true：

- 打开一个空的数据集；
- 调用了 Last；

- 调用了 Next，且现在已经在数据集的最后一条记录;
- 调用了 SetRange，且范围是无效的。

除此以外，Eof 属性都将返回 false。

Eof 属性通常用在一个循环中，每调用一次 Next 就要判断一下 Eof 属性，以避免对不存在的记录进行操作。

程序示例如下：

```
Table1->DisableControls(); // 切断 Table1 与其他数据浏览组件的联系
try
{
  Table1->First(); // 将当前指针移动到表中的第一条记录
  while (!Table1->Eof) // 遍历表中的记录，直到最后一条记录
  (
    // 处理当前的记录
    ...
    Table1->Next(); // 移动当前指针到表中的下一条记录
  }
}
__finally
{
  Table1->EnableControls(); // 恢复 Table1 与其他数据浏览组件的联系
}
```

这段程序代码同时演示了在遍历数据集的所有记录时怎样暂时禁止数据控制组件跟着刷新。在遍历数据集的所有记录前应当调用 DisableControls 函数禁止刷新，这样能够加快遍历的速度，因为刷新也要花时间。遍历结束后，应当调用 EnableControls 函数恢复刷新。EnableControls 最好在 try..._finally 语句的_finally 部分调用，这样即使在遍历时出现异常，也能保证刷新得到恢复。

如果 Bof 属性返回 true，表示现在已到数据集的开头。

下列操作会把 Bof 属性设为 true：

- 打开一个非空的数据集;
- 调用了 First;
- 调用了 Prior，且现在已经是数据集的第一条记录。

除此以外，Bof 属性都将返回 false。与 Eof 属性一样，Bof 属性通常用在一个循环中，每调用一次 Prior 就要判断一下 Bof 属性，以避免对不存在的记录进行操作。

程序示例如下：

```
Table1->DisableControls();
try
{
  while (!Table1->Bof)
  (
    // 处理当前的记录
    ...
```