创建实时数据库动态图表

王有一 (西安国家电力公司热工研究院 710032)

摘要：本文根据数据集构件与图表构件组合的功能特点，说明了解决数据库图表动态显示的有效途径。并通过设置一个作为缓冲区的数据库文件，同时追加若干定时器构件与之配合，从而实现了数据库图表的动态化。

关键词：C++ Builder 构件编程 构件属性 动态图表 数据库缓冲区

1 前言

在工业控制中，将实时数据以曲线的形式动态显示，是备受欢迎的。以下我们介绍怎样在C++ Builder 4.0下实现数据的动态趋势显示。C++ Builder以构件编程的思想较之传统的Windows编程模式有无可比拟的优越性。在C++ Builder中，各种用途的软件单元被封装到构件中，程序员只需要把所需功能的构件组合在一起，通过少量手工编程就可完成应用程序的整体功能。用C++ Builder构件编程创建数据库图表是十分方便的事，然而要实现图表动态化，C++ Builder并没有现成的构件方法可用。通过实践摸索，我们找到一个行之有效的途径，实现了数据库图表的动态显示。

2 构件编程

使用C++ Builder将数据库的数据以图表曲线的形式直观地显示出来，只需要选用以下4个构件加以组合。

2.1 窗体构件 TForm

在C++ Builder主菜单中选择File New Application菜单项，新建一个应用程序。这时C++ Builder 4.0会自动生成一个窗体和它的单元文件。窗体为应用程序提供可视化界面，它是一个容器构件，其他种类的构件一般都放在窗体上。

2.2 表格构件 TTable

在C++ Builder 4.0主窗口构件栏的Data Access选项卡中，选择TTable构件并将它放置到窗体中(用鼠标单击TTable构件，再用鼠标单击窗体中某一位置，TTable构件就被放置到该处，选择其他构件的方法与此相同)。设置它的DatabaseName(数据库别名)属性为RD(由用户定义)，以指定要访问的数据库文件目录，设置它的TableName属性为Data 1.db，以指定要访问的表，最后把Active属性设为True。表格构件是最常用的数据集构件之一，应用程序将使用此构件并通过数据库引擎(BDE)来访问Data-1.ab的记录。

2.3 图表构件 TDBChart 和序列构件 TLineSeries

如果图表需要从数据集获取数据，就应采用TDBChart构件，因为该图表构件内含对BDE的支持。序列构件是包容在图表构件中使用的，所以它不能在构件栏选择，而是从图表构件的编采辑器中选取。在构件栏的Data Controls选项卡中选取一个TDBChart构件放置到Form窗体上，用鼠标右键单击此构件，在弹出的菜单中选Edit Chart命令打开图表编辑器。在编辑器Chart项Series页，单击“Add...”按钮，会弹出TeeChart Gallery图表序列库窗口，在它的Standard页下，从Point、Line、Area、Bar等众多序列类型中选择我们需要的Line类型，然后单击“OK”按钮，就会在图表上加入一个TLineSeries序列(这时一个随机数的曲线出现在图表上)。

接下来需要将TLineSeries序列与我们的数据集连接。先把图表编辑器切换到Series项Data Source页，你会看到一个下拉列表框指示出当前的数据源是Random Values随机值。展开列表框，从中选择Dataset为当前数据源，这时该页面又会出现Dataset和Labels、X、Y四个列表框。展开的Dataset列表框中只有一个数据集Table1可选，这正是前面我们在Form窗体中加入的表格构件，如果在Form中加入多个表格构件，Dataset框中会列出多个数据集供选择。

在Dataset框选择Table1，假设Table1指定的数据表

Data 1.db(它设置了日期、时间、转速三个字段)用来保存汽轮机转速每秒的采样值，那么Labels、X、Y的列表框中会列出日期、时间、转速三个字段供选择。在X框选时间字段，在Y框选转速字段，Labels框不选。注意还必须把X轴标记为DateTime，使它表示时间轴。最后单击"Close"按钮退出图表编辑器，你会发现图表上的随机数序列已更换为一个新序列，它是通过自动检索Data 1.db数据表的记录来生成序列的X轴和Y轴上的点。

Chart选项用于从总体上设置图表的属性，包括序列、坐标轴、标题、图例、分页、背景等，而Series选项则用于设置某个序列的属性(顺便说明，一个图表上可以有多个序列，每个序列可以单独指定各自的数据集)，包括格式、标注、数据源以及其他杂项。我们可以根据需要给图表加上标题说明，给坐标轴添加文字标注。实际上图表和序列构件的属性还可以在对象浏览器上设置，只不过图表编辑器将属性按功能归类，对用户更清晰和方便。

到目前为止，我们还没有动手编写一条程序语句，数据库图表就已经基本完成。现在选择 C++ Builder 主菜单RUN|RUN菜单项，开始编译、连接并运行这个应用程序，数据库图表就会被创建并显示出来。

3 实现图表动态化

如果数据库文件Data 1.db中已有300个记录，应用程序将会显示出一个静态的数据曲线，布满在时间轴为0～5分钟的整个图表。而我们的数据库是实时的，新采集的工业数据在不断地加入，却在图表上看不到。怎样显示出这些新采集的数据呢?通过对象浏览器，我们试着改一下DBChart构件中影响显示区间的属性参数：

ScaleLastPage: true—改为—> false(设置最后一页的曲线不按数据集实有数据点布满图表，而是依据MaxPointsPerPage属性值而定)

MaxPointsPerPage：0—改为—>600(设置每页最大点数为600，初始值0的意思是通过检索数据集实时记录自动确定每页最大点数)

再看运行效果：这时图表的X轴时间刻度增加到10分钟，代表转速值的曲线没有布满图表，而是分布在0～5分钟时间段，并开始缓慢前移(每秒移动一次)，动态地指示出了新采集的数据。但是，当反映实时数据的动态曲线走完所设置的时间段(设置值600相当于10分钟)，就看不见了。再增大时间段显然毫无意义。根据TTable、TDBChart和TLineSeries构件组合的特点，可看到它们的功能主要是：将与序列构件相连接的数据集数据映射到图表中的曲线。所以再更改属性参数也不可能从根本上解决问题，只有从数据库本身的设置想办法。如果我们将数据库设置为动态的(连同时间字段)，那么映射到图表的曲线(连同X轴时间刻度)也会相应地呈现动态变化。

4 手工编程

为了实现数据库自身的动态更新，还需要在窗体中增加3个定时器构件。同时还需要增加一个表格构件Table2，它的TableName属性指向Data 2.db，Data 2.db代替Data 1.db成为数据库主文件，而Data 1.db只作为一个数据库缓冲区使用。

定时器1的作用是将每秒新采集的数据存入缓冲区文件Data 1.db，它的Interval属性设置为1000(定时1秒)，以下是它的事件响应函数：

```
void fastcall TForm1::Timer 1 Timer(TObject * Sender)
{
    ⋮
// 检测数据表是否处于关闭状态
if (Table1->State==dsInactive)
// 打开数据表
Table1->Active=true;
// 在数据表末尾添加一条空记录
Table1->Append();
// 给空记录各字段赋值
Table1->Fields->Fields［0］->Value=Date();
Table1->Fields->Fields［1］->Value=Time();
// 全局变量NewVal暂存转速采集值
Table1->Fields->Fields［2］->Value=NewVal;
// 将添加的记录保存到数据表
Table1->Post();
    ⋮
}
```

定时器2的作用是：当数据曲线走完最初3分钟时间段即触发定时事件响应函数：启动定时器3，同时停止自身定时，它的Interval属性设置为180000。

```
void fastcall TForm1::Timer2Timer(TObject*Sender)
{
// 启动定时器3
Timer3->Enabled=true;
// 停止定时器2
```

```
Timer2->Enabled=false;
}
```

定时器3的作用是：当数据曲线走完后面7分钟时间段时触发定时事件响应函数：先将Table1表中记录添加到Table2表的末尾，然后删除Table1表的前420个记录，从而使映射Table1表的数据曲线在图表上后退了一个7分钟时间段，你会发现X轴上的时间刻度值随着曲线一起后移，然后曲线在图表上继续前移…。定时器3将使这个动态过程周而复始地进行，它的Interval属性设置为420000。

```
void fastcall TForm1::Timer3Timer(TObject*Sender)
{
  if (Table1->State==dsInactive)
    Table1->Active=true;
  if(Table2->State==dsInactive)
  Table2->Active=true;
//将Table1表中记录添加到Table2表的末尾，重复部分不添加
  Table2->BatchMove(Table1,batAppendUpdate);
//取表记录总数(应为600)
  int iRecCount=Table1->RecordCount;
//将记录指针指向表首
  Table1->First();
//从记录总数中删除前420个记录
  for (int i=0;i<iRecCount-180;i++)
  Table1->Delete();
}
```

现在，应用程序已实现数据库曲线的动态显示，还有一些初始化的工作需要做，我们将它填写在建立窗体的事件响应函数中：

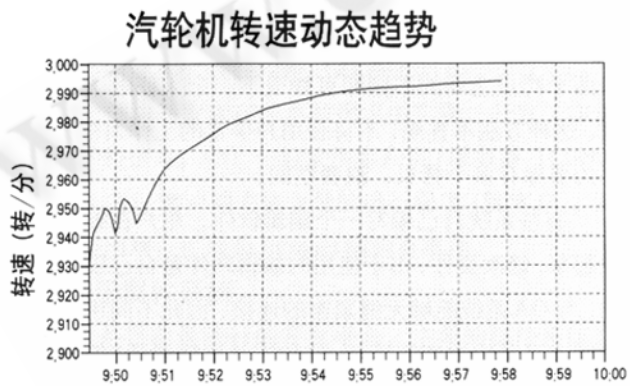
```
void fastcall TForm1::FormCreate(TObject*Sender)
{
//检测数据表是否存在
  if(!Table1->Exists)
  {
//关闭数据表
  Table1->Active=false;
//指定数据库别名
  Table1->DatabaseName="RD";
//指定数据库类型
  Table1->TableType=ttParadox;
//指定数据表名
  Table1->TableName="Data _1";
//清除Table1原有字段定义
  Table1->FieldDefs->Clear();
//添加新的字段定义
  Table1->FieldDefs->Add(" 日期 ",ftDate,0,true);
  Table1->FieldDefs->Add(" 时间 ",ftTime,0,false);
  Table1->FieldDefs->Add(" 转速 ",ftFloat,0,false);
//清除Table1原有索引字段定义
  Table1->IndexDefs->Clear();
//添加新的索引字段定义
  Table1->IndexDefs->Add("","" 日期；时间 ",
      TIndexOptions()<<ixPrimary<<ixUnique);
//建立新表
  Table1->CreateTable();
  }
  if (!Table2->Exists)
   {
  Table2->Active=false;
  Table2->DatabaseName="RD";
  Table2->TableType=ttParadox;
  Table2->TableName="Data 2";
  Table2->FieldDefs->Clear();
  Table2->FieldDefs->Add(" 日期”，ftDate,0,true);
  Table2->FieldDefs->Add(“时间”，ftTime,0,false);
  Table2->FieldDefs->Add(" 转速 ",ftFloat,0,false);
  Table2->IndexDefs->Clear();
  Table2->IndexDefs->Add("","" 日期；时间 ",
  TIndexOptions()<<ixPrimary<<ixUnique);
  Table2->CreateTable();
  }
  if (Table1->State==dsInactive)
  Table1->Active=true;
  if(Table2->State==dsInactive)
  Table2->Active=true;
//将记录指针指向表尾
  Table1->Last();
//删除表的全部记录，该表用作数据库缓冲区
  while (!Table1->Bof)
  Table1->Delete();
```

(下转第76页)

(上接第 63 页)

5 图例

数据库图表的图例，是一条反映汽轮机升速过程的动态曲线(见附图)。



最后要说明一下 DBChart 构件的 Align(排列)属性，它应当被设置为 alClient(充满全部客户区)，这样，当用户改变窗口的大小时，图表的形状大小也会随着变化。■

参考文献

1 *徐新华 著，C++Builder 3数据库编程指南，清华大学出版社，1999.2*

2 *刘海涛 著，Borland C++ Builder 3入门与提高，清华大学出版社，1999.4*

3 *刘文圣，刘光，权元聪 编著，C++ Builder 指南，人民邮电出版社，1999.1*