在 Delphi 数据集中实现查找字段的动态创建

Dynamic Creation of Lookup Fields in Delphi DataSet

阮国龙　（湖北 咸宁学院信息工程学院 437000）
刘　铭　（武汉华中师范大学信息技术系 430079）

摘要：在 Delphi 环境中，我们一般只能在 IDE 中利用字段编辑器并通过设置相关属性的方法创建查找字段。但在实际应用中，我们往往需要在程序运行过程中根据数据集的不同而动态创建查找字段。本文在分析了字段创建的过程后，通过在 TCustomClientDataSet 类中添加数组属性的方法简易实现 Delphi 数据集中查找字段的动态创建。

关键词：数组属性 Delphi 数据集 动态创建 查找字段

1 引言

在数据库管理信息系统的界面设计过程中，为了使信息显示更直观（在界面中不出现意义不明的代码）和数据修改更方便（只需在下拉列表框中选择相应项即可，不用键入相应名称信息），我们经常会用到查找字段。因为查找字段还可利用数据库设计中的数据的一致性和有效节省系统资源的优点。在面向对象程序开发环境中，如 Delphi，查找字段一般是在 IDE（集成设计环境）中的字段编辑器上创建并设置相关属性。这种方法实现起来比较简单，但它要求在设计时数据集应与相关数据表关联，而且只能利用固定字段。另外这种静态创建查找字段的方法也不具灵活性，它要求设计时就要在所有需要由代码转换成名称的数据集中都事先创建好各自的查找字段，而且一旦数据表的字段定义发生改变，就有可能要重新设计。另外，当某数据集组件对应的数据表是动态变化的，用这种方法就无能为力了。因此，我们有必要寻求一种更具灵活性的动态创建查找字段的方法。以下，就从分析 Delphi 数据集中字段创建过程入手，逐步推导在数据集中动态创建查找字段的实现方法。

2 Delphi 数据集中字段创建过程分析

我们以 Delphi 中的客户端数据集（ClientDataSet）为例，通过单步跟踪程序运行过程可看到，程序在执行 TCustomClientDataSet.InternalOpen 方法中要根据 DefaultFields 值决定是否创建动态字段。相关程序代码如下：

```
procedure TCustomClientDataSet.InternalOpen;
begin
  ...
  If DefaultFields then CreateFields;
  ...
end;
```

在 IDE 中，若我们已利用字段编辑器（Fields Editor）窗口添加了若干固定字段对象，如图 1 所示。则此时的 DefaultFields 值为 False，即不执行 CreateFields 方法，字段对象已经随着数据集对象的创建而创建了。若在 IDE 中没有利用字段编辑器窗口创建固定字段，则 DefaultFields 值为 True，即会通过调用 TDataSet.CreateFields 方法并根据后台数据表字段的元数据信息（此信息保存在动态创建的 FieldDef 对象中）动态创建字段对象，TDataSet.CreateFields 再调用 TFieldDef 的 CreateField、CreateFieldComponent 等方法，直到关联数据表中包含的所有字段全部创建完毕。因此它不需要在设计时就确定字段对象的类型、大小等信息，使得数据集对象不必跟固定的后台数据表相关联。

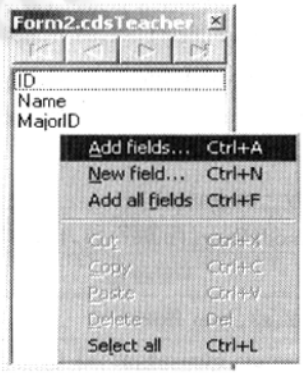

图 1　利用 Fields Editor 窗口添加固定字段

3 解决问题的思路与方法

从以上对 Delphi 数据集中字段创建过程分析可知，动态字段对象的创建是在 TCustomClientDataSet.InternalOpen 中调用了 CreateFields 方法，当 CreateFields 执行完毕，ClientDataSet 中的字段对象就已被创建，当然

这些字段对象对应于数据表中已定义的字段。而需要的查找、计算等自定义字段对象还没开始创建。我们的思路就是在 CreateFields 执行完毕后，接着就要动态创建查找字段。而查找字段必须给定数据集的 LookupDataSet、Size、FieldName、FieldKind、KeyFields、LookupKeyFields、LookupResultField、DataSet 等属性。

为了体现面向对象程序设计思想，应该把动态创建查找字段的代码整合进 TCustomClientDataSet 的内部源代码中。因此我们想到了在 TCustomClientDataSet 中添加属性的方法，使其他程序设计者只需指定相应的属性值，就能实现查找字段的创建。但问题是添加了这几个属性后，在一个数据集中只能创建一个查找字段，这显然不符合实际的需要。因此我们想到了用数组类型的属性。

数组属性通过属性名后跟参数表进行说明，参数表中的参数又称为下标，访问属性时通过下标来进行。数组属性声明的一般形式为：

property 属性名[参数表]：类型 read 方法名 write 方法名 <default>

其中访问说明中 read 和 write 后必须跟 Get/Set 方法名，不允许出现域名。因此除要添加几个相应的域名外，还要定义并实现相应的 function 和 procedure 方法。最后是要更改源代码中的 TCustomClientDataSet.InternalOpen 的实现方法。整个步骤如下：

(1) 在 TCustomClientDataSet 类中添加数组类型的域名及定义相应的 Get/Set 方法

```
FExFLookupDataSet: array [0.. MaxNum] of TDataSet;
//MaxNum 为在数据集中可能要添加的查找字段最大个数
FExFClientDataSet: array[0.. MaxNum]of TClientDataSet;
//FExFClientDataSet 为查找字段的 LookupDataSet
FExFSize: array[0.. MaxNum] of Integer;
FExFFieldName: array[0.. MaxNum] of string;
FExFFieldKind: array[0.. MaxNum] of TFieldKind;
FExFKeyFields: array[0.. MaxNum]of string;
FExFLookupKeyFields: array[0.. MaxNum] of string;
FExFLookupResultField: array[0.. MaxNum] of string;
.....
function TCustomClientDataSet.GetExFLookupDataSet(I: Integer): TDataSet;
procedure TCustomClientDataSet.SetExFLookupDataSet(I: Integer; Value: TDataSet);
```

其他 7 对 Get/Set 方法的定义类似，在此略。

(2) 在 TCustomClientDataSet 中添加 8 个数组属性

```
property ExFLookupDataSet[I: integer]: TDataSet read GetExFLookupDataSet write SetExFLookupDataSet;
```

其他属性 ExFClientDataSet[I: Integer]、ExFSize[I: Integer]、ExFFieldName[I: Integer]、ExFFieldKind[I: Integer]、ExFKeyFields[I: Integer]、ExFLookupKeyFields[I: Integer]、ExFLookupKeyFields[I: Integer]、ExFLookupResultField[I: Integer]的定义类似，在此略。

(3) 相应的 function 和 procedure 方法的实现

```
function TCustomClientDataSet.GetExFLookupDataSet(I: Integer): TDataSet;
  begin
    Result := FExFLookupDataSet[I];
  end;
  procedure TCustomClientDataSet.SetExFLookupDataSet(I: Integer; Value: TDataSet);
  begin
    FExFLookupDataSet[I] := Value;
  end;
```

其他相应 Get/Set 方法的实现类似，在此略。

(4) 对源代码中 TCustomClientDataSet.InternalOpen 方法的改进

通过上面的分析结果可知，查找字段对象的动态创建过程应封装在 TCustomClientDataSet.InternalOpen 方法中完成。在此我们把 TCustomClientDataSet.InternalOpen 过程中的 if DefaultFields then CreateFields 改为：

```
if DefaultFields then
begin
 CreateFields;
 for I:= 0 to MaxNum do //开始对数据集查找字段对象的动态创建
 begin
  if Assigned(ExFFieldKind[I]) then
  //如果已指定了字段类型 ExFFieldKind[i]属性，说明有可能要添加字段对象
   try
    ExField[I] := TStringField.Create(self); //创建字符类型的查找字段对象
    with ExField[I]do
    begin //指定动态字段对象相应属性值
     LookupDataSet := ExFLookupDataSet[I];
```

```
        Size := ExFSize[i];
        FieldName := ExFFieldName[i];
        FieldKind := ExFFieldKind[i];
        KeyFields := ExFKeyFields[i];
        LookupKeyFields := ExFLookupKeyFields[i];
        LookupResultField := ExFLookupResultField[i];
      end;
      Integer((@ExField[i].DataSet)^) := Integer(TDataSet(self)); //指定 DataSet 属性值
      TDataSet(self).Fields.Add(ExField[i]); //向数据集添加字段对象
      TDataSet(self).FieldByName(ExFKeyFields[i]).Visible := False;
      //添加了查找字段对象后,把相应的代码字段隐藏
      TDataSet(self).FieldByName(ExFFieldName[i]).Index :=
        TDataSet(self).FieldByName(ExFKeyFields[i]).Index;
      //把所添加的字段对象移到数据集中相应代码字段对象的位置
    except
      on E: Exception do
        //错误处理代码
    end;
  end;
end;
```

注意在给 ExField 指定 DataSet 属性值时,需要一定技巧,我们通常想到的方法就是直接给它赋值,如 ExField.DataSet := DataSet,这时,会同时创建两个相同的查找字段对象。这是因为在执行 ExField.DataSet := DataSet 语句时,会调用 TField 派生类的 SetDataSet 方法,而该方法会再一次打开 DataSet 数据集,因此共打开了两次,也就导致了创建两次查找字段对象。我们对此解决的办法是直接把存放 DataSet 的内存值赋给 ExField.DataSet 所指的内存值,赋值方法为 Integer((@ExField.DataSet)^) := Integer(DataSet)。

至此,我们的底层设计工作已完成。使用时只需在应用程序的相应事件中指定数据集的以上所添加的几个属性值即可。

4 实践应用

以下为在一教职工数据集(CdsTeachers)中动态创建"专业"查找字段对象的过程。

```
procedure TForm1.BtnOpenClick(Sender: TObject);
  begin
    TempData := Intf.ReadInfo('', '001', 1, -1, OutRecs1);
    //读主数据集数据包并存放在一个临时变量中。ReadInfo 方法为服务器自定义接口中//的读取数据包的方法,在不同的程序中可用其它的数据读取方法,如 DataRequest
    with CdsTeachers do
    begin
      ExFClientDataSet[0] := TLMClientDataSet.Create(nil);
      ExFClientDataSet[0].Data := Intf.ReadInfo('', '003', 1, -1, OutRecs1);
      //ExFClientDataSet[0]为第一个查找字段的 LookupDataSet
      ExFLookupDataSet[0] := ExFClientDataSet[0];
      ExFSize[0] := ClientDataSetEx.FieldByName('Name').Size;
      ExFFieldName[0] := '专业';
      ExFFieldKind[0] := fkLookup;
      ExFKeyFields[0]:= 'MajorID';
      ExFLookupKeyFields[0] := 'ID';
      ExFLookupResultField[0]:= 'Name';
      //若有多个查找字段,依次改变下标 0 即可
      ...
      Data := TempData;
    end;
  end;
```

从以上过程看出,为数据集指定我们所定义的几个新属性值即可轻易实现查找字段的动态创建。需要注意的是:我们这里不是直接用 CdsTeachers.Open 方法打开数据集,而是先指定相关属性值,最后给定它的 Data 属性值。

5 结束语

以上介绍的方法已在 Delphi 7 环境中调试通过,当然在 C++ Builder 中也能使用。另外计算字段的动态创建、以及其他数据集组件(虽然本文大多是以 ClientDataSet 为例)如 ADO 数据集组件要实现动态创建字段,也可以参照此方法实现。

参考文献

1 Steve Teixeira, Xavier Pacheco. Delphi 6 开发人员指南[M],机械工业出版社,2003。
2 乔林,参透 Delphi / Kylix [M],中国铁道出版社,2003。