

Capítulo 8

Adicionando Componentes de Banco de Dados à uma Aplicação

Overview

Os componentes de banco de dados e suas capacidades descritas neste capítulo fornecem as ferramentas necessárias para a criação de aplicações que utilizem bancos de dados local, baseado em PC e remotos, baseados em servidores SQL.

Este capítulo aplica os componentes de banco de dados (descritos no capítulo anterior) no desenvolvimento de uma aplicação. Estes componentes estão nas páginas Data Access e Data Controls da Component Palette.

Conceito de um DataSet no Delphi

Introdução

Para manipular e consultar bancos de dados, você precisa entender o conceito de dataset. Um dataset no Delphi é um objeto que consiste de uma série de registros, cada um contendo qualquer quantidade de campos e um ponteiro para o registro atual. O dataset pode ter uma correspondência direta, um-para-um, com uma tabela física, ou, como um resultado de uma query, pode ser um subconjunto de uma tabela ou uma junção de diversas tabelas.

Tipo de Objeto TDataSet

Um dataset no Delphi é o tipo de objeto TDataSet e é uma classe abstrata. Uma classe abstrata é uma classe de onde você pode derivar outras classes, mas não pode criar uma variável desta classe. Por exemplo, ambos os componentes Query e Table classificam-se como componentes TDataSet porque cada um foi derivado do objeto TDataSet. Note que você não encontrará nenhum componente chamado TDataSet na Component Palette. O TDataSet contém as abstrações necessárias para manipular diretamente uma tabela. É a ferramenta utilizada para abrir uma tabela e navegar por suas colunas e linhas.

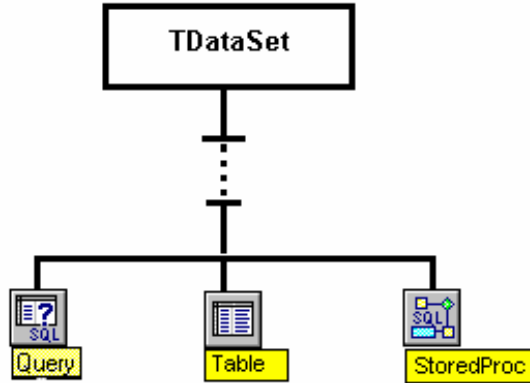
Os componentes neste capítulo são referenciados pelo seu tipo de objeto (identificado no Delphi pelo seu nome com o prefixo T). O termo componente DataSet é utilizado para referenciar um componente Table, Query, ou StoredProc. TTable, TQuery, e TStoredProc são descendentes de TDataSet, ou seja, eles herdam as propriedades de TDataSet.



O conceito de herança é discutido em detalhes no capítulo Object-Oriented Programming in Delphi de seu manual.

Diagrama dos componentes DataSet

O diagrama a seguir mostra o conceito dos componentes DataSet, que são componentes derivados do tipo de objeto TDataSet:



Utilizando o Componente e DataSource

Introdução

O componente DataSource atua como intermediário entre o componente DataSet (TTable, TQuery, ou TStoredProc) e os componentes Data Control. Dentre os componentes Data Control incluem DBGrid, e DBText entre outros. Esta seção explica:

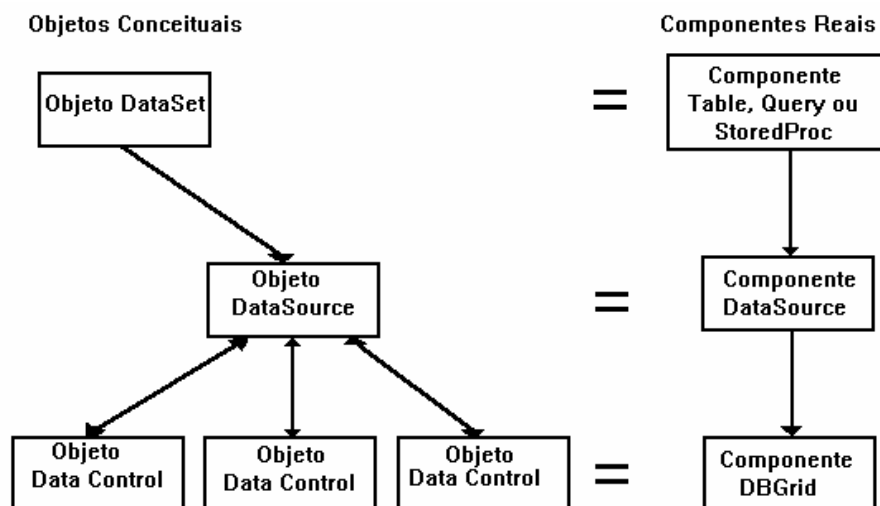
- Qual o papel do componente DataSource em uma aplicação de banco de dados
- Como utilizar propriedades e eventos do DataSource
- Como adicionar componentes DataSource à sua aplicação

Papel de um Componente DataSource

Um Componente DataSource gerencia o relacionamento entre uma tabela de banco de dados e a representação deste dado em seu form. É um intermediário entre os componentes DataSet e DataControl. Componentes DataSet gerenciam a comunicação com o Borland Database Engine (BDE), e o componente DataSource gerencia a comunicação com componentes data-aware Data Control.

Em uma típica aplicação de banco de dados, um componente DataSource é associado com um componente DataSet (Table ou Query) e um ou mais componentes Data Control (tais como DBGrid).

O diagrama a seguir mostra este relacionamento:



Utilizando Propriedades e Eventos DataSource

As propriedades e eventos chave do componente DataSource são:

- Propriedade DataSet
- Propriedade Enabled
- Propriedade AutoEdit
- Evento onDataChange
- Evento onUpdateData
- Evento onStateChange

Cada uma destas propriedades e eventos são discutidas nas seções a seguir:

Utilizando a propriedade DataSet do Componente DataSource

A propriedade DataSet identifica o nome de um componente DataSet.

Você pode atribuir à propriedade DataSet através de programação ou utilizando o Object Inspector.

O valor atribuído à propriedade DataSet é o nome de um objeto TDataSet.

Por exemplo, a linha de código a seguir atribui um nome ou objeto TQuery à propriedade DataSet de um componente ou objeto DataSource.

```
DataSource1.DataSet := Query1
```

Você pode inserir diversos componentes Query, Table, e StoredProc em um form a atribuir a propriedade DataSet baseado em uma condição no programa. Você também pode atribuir a propriedade DataSet a um nome ou objeto TQuery, TTable, ou TStoredProc encontrado em outro form utilizando o identificador da unit do form. Por exemplo, após incluir a Unit1 na cláusula uses, você pode digitar o seguinte:

```
DataSource1.DataSet := Unit1.Form1.Table1;
```

Utilizando a Propriedade Enable do TDataSource

A propriedade Enable inicia ou termina a comunicação entre os componentes TDataSource e DataControl.

Os valores da propriedade Enable são:

- True
Os componentes Data Control conectados ao TDataSource "enxergam" as alterações do dataset.
- False
Você pode manipular o TDataSet através de programação sem que os componentes Data Control "enxerguem" as alterações.

Por exemplo, o código a seguir desabilita o TDataSource, procura por um valor coincidente de número do cliente, e depois habilita o TDataSource para que ou a linha do cliente ou a última linha seja exibida.

```
DataSource1.Enabled := False;  
Table.First;  
While not Table1.EOF do  
begin  
if Table1.FieldName('NoCliente').AsString = LookupCust then  
Break;  
Table1.Next;  
end;  
DataSource1.Enabled := True;
```

Utilizando a propriedade `Enable` permite que você desconecte temporariamente o componente visual `Data Control` do `TDataSource`.

No exemplo anterior, se a tabela contiver 2000 linhas e o `TDataSource` estiver habilitado, o usuário de sua aplicação veria 2000 linhas rolando na tela durante esta operação. Desabilitando `TDataSource` é uma maneira mais eficiente de se pesquisar em um grande número de linhas, pois o componente `Data Control` não atrasará a procura exibindo cada linha conforme esta for sendo alterada.

Utilizando a Propriedade `AutoEdit` de `TDataSource`

A propriedade `AutoEdit` controla como a edição é iniciada nos componentes `Data Control`.

Os valores das propriedades `AutoEdit` são:

- `True`
O modo de edição é iniciado sempre que o usuário comece a digitar dentro de um componente `Data Control`.
- `False`
O modo de edição é iniciado quando o método `Edit` é invocado, por exemplo, após o usuário dar um clique sobre o botão **Edit** do `Navigator`. Este parâmetro controla a edição.

Dentro do seu código, você pode utilizar os seguintes métodos para controlar as alterações nos dados da tabela quando `AutoEdit` estiver como **False**:

- `Edit`
- `Post`
- `Cancel`

O código a seguir é um exemplo utilizando o método `Post`:

```
DataSource1.DataSet.Edit; {Start edit mode}  
DataSource1.DataSet.Fields [3] .AsString := "Hello";  
DataSource1.DataSet.Post;
```

Utilizando o Evento `OnDataChange` de `TDataSource`

O evento `OnDataChange` ocorre sempre que:

- A propriedade `State` do `TDataSource` mudar do estado `dsInactive`
- Ocorrer uma alteração de campo, registro, tabela, query ou layout

Este evento é associado com alterações na exibição de dados, tais como rolar para um novo registro ou ativando `TDataSource`. Este evento é útil para monitorar alterações nos componentes `Data Control`.

Utilizando o Evento `OnUpdateData` de `TDataSource`

O evento `OnUpdateData` ocorre quando:

- O registro atual no `TDataSet` estiver para ser atualizado
- Uma alteração estiver para ser confirmada

Este evento é útil na monitoração de alterações nos dados de uma tabela. Por exemplo, você pode utilizar este evento para criar auditor de alterações nos dados em sua aplicação.

Utilizando o Evento `OnStateChange` de `TDataSource`

O evento `OnStateChange` ocorre sempre que a propriedade `State` de `Data Source` for alterada. Este evento é útil para monitoração de alterações na propriedade `State`. A propriedade `State` pode ter os seguintes valores:

- `dsInactive`
- `dsBrowse`

- dsEdit
- dsInsert
- dsSetKey
- dsCalcFields

Passos para Adicionar um Componente DataSource

Execute os passos a seguir para adicionar um componente DataSource ao form:

| Passo | Ação |
|-------|---|
| 1 | Utilizando a página Data Access da Component Palette, adicione o componente DataSource ao form. |
| 2 | Utilize o Object Inspector para definir a propriedade DataSet ou escreva o código apropriado. |
| 3 | Defina as propriedades AutoEdit e Enabled, ou ambas, apropriadamente. |
| 4 | Escreva event handlers para um ou todos os seguintes eventos: <ul style="list-style-type: none"> • onDataChange event • onUpdateData event • onStateChange event |

Utilizando o Objeto TDataSet

Introdução

Geralmente, quando você manipula um componente Table ou chama um método TTable, você está utilizando um método derivado do objeto TDataSet. TDataSet oferece um grande número de propriedade, métodos e eventos.



Você pode aprender mais sobre TDataSet utilizando o Help Online.

Esta seção discute um pouco sobre propriedades, métodos e eventos do objeto TDataSet.

Propriedades do Objeto TDataSet

A tabela a seguir descreve as propriedades mais importantes do objeto TDataSet:

| Propriedade | Descrição |
|--|---|
| Active | Abre ou fecha um componente DataSet. Abrir um componente DataSet estabelece a conexão entre o componente e o banco de dados. Você pode definir a propriedade Active no Object Inspector durante o design ou diretamente em seu código, como segue: <pre> if DeActivate = true then Table1.Active := True Else Table1.Active := False; </pre> Os métodos Open e Close também definem a propriedade Active de TDataSet. |
| EOF (End of File) BOF (Beginning of File) | Propriedades somente de leitura (read-only) com os seguintes valores: <ul style="list-style-type: none"> • EOF é definido para True quando você tenta mover para além da última linha do dataset • BOF é definido para True quando o componente DataSet é aberto ou quando o ponteiro do DataSet para a linha atual estiver posicionado na primeira linha. |
| Fields | Um array do objeto TField. Você pode definir e ler dados dos campos da linha atual utilizando esta propriedade. Por exemplo: |

| | |
|--|--|
| | <pre>Table1.Fields[0].AsString := 'Hello'; Table1.Fields[1].AsString := 'World';</pre> |
|--|--|

Métodos do Objeto TDataSet

O objeto TDataSet fornece ao componente Table diversos métodos.

Alguns dos métodos mais importantes são mostrados na tabela a seguir:

| Método | Descrição |
|--|--|
| Open Close Refresh | <p>Operam nos datasets, como segue:</p> <ul style="list-style-type: none"> • O método Open é equivalente a definir a propriedade Active para True. • O método Close define a propriedade Active para False • O método Refresh permite ler novamente o dataset do banco de dados . Se você precisa se assegurar que os dados são os mais atuais, contendo quaisquer alterações feitas por outros usuários, utilize o método refresh. Um componente Table, Query, ou StoredProc deve estar aberto com open quando Refresh for chamado. <p>Um exemplo destes métodos é :</p> <pre>Table1.Open; Table1.Close; Table1.Refresh;</pre> |
| First Last Next Prior MoveBy | <p>Permite navegar ou alterar a linha atual do dataset. A seguir, um exemplo utilizando vários destes métodos:</p> <pre>Table1.First While not Table1.EOF do begin {Seu código aqui} Table1.Next; end;</pre> <p>O método MoveBy move um número especificado de linhas. Por exemplo:</p> <ul style="list-style-type: none"> • <code>Table1.MoveBy(3)</code> move 3 linhas para cima. • <code>Table1.MoveBy(-2)</code> move 2 linhas para trás. |
| Insert Edit Delete Append Post Cancel | <p>Permite modificar os dados em uma tabela de banco de dados, como segue:</p> <ul style="list-style-type: none"> • O método Insert permite adicionar uma linha à tabela. Por exemplo: <pre>Table2.Insert; Table2.Fields[0].AsInteger := 20; Table2.Fields[1].AsString := 'News'; Table2.Fields[2].AsString := '5 Horas'; Table2.Post;</pre> <ul style="list-style-type: none"> • O método Post faz com que a operação Insert Update ou Delete ocorra. • O método Cancel faz com que uma operação Insert Delete, Edit ou Append não ocorrida seja cancelada. |
| FieldByName | <p>Fornecer uma maneira de acessar dados de uma coluna especificando no nome da coluna do banco de dados. Como no exemplo a seguir:</p> <pre>s := Table2.FieldByName ('area') .AsString;</pre> |
| SetKey | <p>Procura através dos datasets, como segue:</p> |

| | |
|--|--|
| GotoKey | <ul style="list-style-type: none"> • SetKey alterna a tabela para o modo de pesquisa (search). Enquanto neste modo, a propriedade Fields tem uma utilização especial. • GotoKey inicia a pesquisa por um valor coincidente com o valor encontrado em Fields[n]. Fields[n] contém o valor a ser pesquisado por valores em outras colunas definindo a coluna Fields correspondente. <p>O exemplo a seguir mostra um exemplo da utilização de SetKey e GotoKey:</p> <pre> Table1.SetKey; Table1.Fields[0] .AsString Edit1.Text; Table1.GotoKey; </pre> |
| SetRangeStart SetRangeEnd ApplyRange | <p>Permite ser mais seletivo nos dados que sua aplicação seleciona ou utiliza na tabela. Geralmente uma tabela é grande e você quer selecionar somente uma série de valores da tabela. O método Range permite fazer tal seleção. Exemplo:</p> <pre> Table1.SetRangeStart Table1.Fields[0] .AsString := Edit1.Text; Table1.SetRangeEnd Table.Fields[0] .AsString := Edit2.Text; Table1.ApplyRange; </pre> <p>A primeira chamada à SetRangeStart o coloca no modo range e a propriedade Fields toma um significado especial. Utilize a propriedade Fields para especificar o valor de início para a série. A chamada para SetRangeEnd inicia um modo onde os valores digitados na propriedade Fields são utilizadas como o valor final da série. ApplyRange faz com que o comando seja processado. Um dataset é criado contendo os valores entre os valores de início e final.</p> |
| FreeBookmark GetBookmark GotoBookmark | <p>Permite criar um marcador de linha em uma tabela ou query e depois retornar à esta linha posteriormente. Os métodos Bookmark utilizam um tipo de objeto chamado TBookmark. Por exemplo:</p> <pre> Var Marker : TBookmark; begin Marker := Table2.GetBookmark; Table2.GotoBookmark(Marker); Table2.FreeBookmark(Marker); </pre> <p>O método GetBookmark aloca um marcador para linha da tabela. O método GotoBookmark altera a localização na tabela para a linha indicada pelo Bookmark alocado anteriormente. Utilize o método FreeBookmark para liberar o espaço alocado para o marcador.</p> |

Eventos do Objeto TDataSet

O objeto TDataSet permite responder a um grande número de eventos. A tabela a seguir descreve diversos deles:

| Eventos | Descrição |
|----------------|--|
| OnOpen | Permite construir e controlar o comportamento de aplicações de bancos de dados. Exemplos de utilização seguem: |
| OnClose | <ul style="list-style-type: none"> • Evento BeforePost para validar os campos de um registro antes de inseri-lo ou atualizá-los |

| | |
|---------------------|--|
| OnNewRecord | <ul style="list-style-type: none">• Evento AfterPost para gravar um registro de auditoria quando necessário• Evento OnDelete para gravar código que efetue a deleção em cascata quando apropriado |
| BeforeInsert | |
| AfterInsert | |
| BeforeEdit | |
| AfterEdit | |
| BeforePost | |
| AfterPost | |
| OnCancel | |
| onDelete | |

Utilizando o Objeto TFields

Introdução

O objeto TField, como o objeto TDataSet, não é encontrado na Component Palette. É uma propriedade do objeto TDataSet (e Componente Table). Algumas propriedades no Object Inspector são objetos com seu próprio conjunto de propriedades. TFields é um deles.

Esta seção descreve a propriedade Fields, que é objeto TFields com seu próprio conjunto de propriedades.

Propriedades Fields do Objeto TDataSet

Uma das propriedades do Objeto TDataSet (portanto, os componentes Table, Query, e StoredProc) é a propriedade Fields.

Como discutido anteriormente neste capítulo, a propriedade Fields permite acessar os campos individuais do dataset. A propriedade Fields é um array dos objetos TFields. Este array ou lista é gerada dinamicamente durante a execução (e portanto, não aparece na lista de propriedades do Object Inspector). O array representa cada uma das colunas no componente Table.

Objetos estáticos TFields são visíveis no Object Inspector. Seções posteriores deste capítulo explicam como criar uma lista estática de objetos TField, mas primeiro, você deve entender algumas das propriedades destes objetos.

Propriedades do Objeto TField

Você deve entender em diversos exemplos o uso do método AsString. O objeto TField não faz nenhuma suposição sobre o tipo de dado que ele contém. Ele possui diversos métodos que permitem definir ou recuperar os valores de um campo. Alguns destes métodos são:

- AsBoolean
- AsFlot
- AsInteger
- AsString

O código a seguir mostra alguns exemplos para a utilização de cada um deles:

```
Fields[0].AsString := 'Ísto é uma string';  
FieldByName('Casado').AsBoolean := False;  
SomaDespesas := Fields[5].AsFloat;  
NoPedido := Fields[3].AsInteger;
```


Outras Propriedades do Objeto TField

A tabela a seguir descreve diversas outras propriedades importantes do objeto TField:

| Propriedade | Descrição |
|------------------|---|
| EditMask | Permite definir uma máscara de input para o campo |
| IsNull | Determina se um campo não possui valor. É uma propriedade read-only. |
| Size | Determina o tamanho de um campo. É uma propriedade read-only. |
| Text | Permite definir ou recuperar um valor de string de um campo |
| FieldName | Fornece o nome do campo do banco de dados. É uma propriedade read-only. |

Utilizando o Componente Table

Introdução

O componente Table é um componente TDataSet que comunica com uma tabela de banco de dados através do BDE. A tabela do banco de dados pode ser tanto local ou em um servidor remoto. Esta seção discute:

- O papel do componente Table
- Propriedades, eventos e métodos do componente Table
- Passos para adicionar um componente Table à sua aplicação

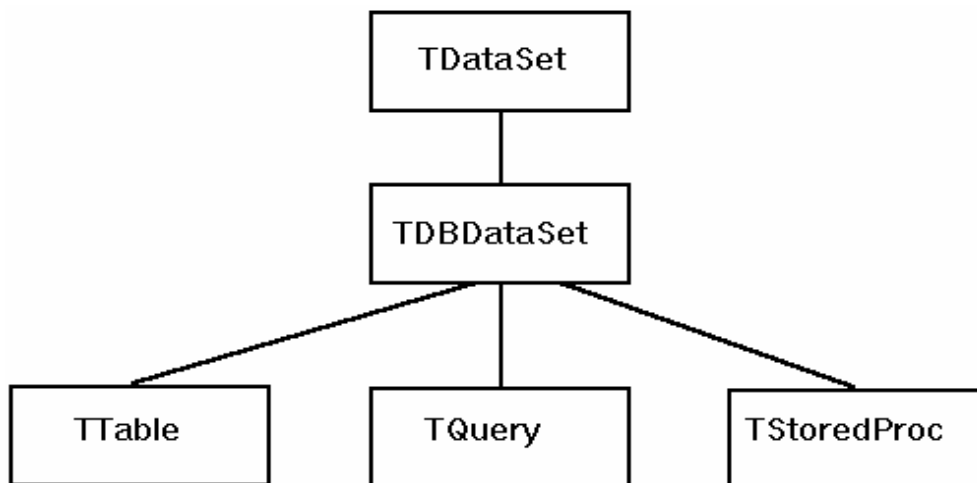
O Papel do Componente Table

O diagrama a seguir mostra que TTable é derivado de TDBDataSet por herança e, por sua vez, TDBDataSet é derivado de TDataSet.

Muito da funcionalidade do componente Table é baseado neste relacionamento.

TTable herda de TDataSet a capacidade de manipular datasets. Ele fornece métodos, tais como Next, First, Last, Edit, SetRange e Insert.

TDbDataSet permite que TTable suporte trabalhar com senhas e outras tarefas associadas com a ligação de sua aplicação com um banco de dados. TTable adiciona métodos e atributos para manipulação de índices, e para armazenar o relacionamento entre esta tabela e outra em sua aplicação .



Propriedades do Componente Table

Como vimos, muito da funcionalidade do componente Table vêm do objeto TDataSet. O componente Table permite utilizar as propriedades, métodos e eventos TDataSet, mas possui diversas propriedades próprias relativas a tabelas de banco de dados. Por exemplo, o componente Table permite que você:

- Especifique índices a serem utilizados
- Crie um cursor “Linkado”

Cursores “Linkados” coordenam dois ou mais componentes DataSet para criar forms master-detail.

A tabela a seguir descreve as propriedades mais importantes do componente Table:

| Propriedade | Descrição |
|---------------------|---|
| DatabaseName | Especifica o seguinte: <ul style="list-style-type: none">• O diretório local de um banco de dados que contenha a tabela a ser visualizada• O alias de um banco de dados remoto |
| TableName | Especifica o nome do banco de dados a ser visualizado |
| Exclusive | Controla o acesso de usuário ao banco de dados. Os valores são: <ul style="list-style-type: none">• True Assegura que nenhum outro usuário acesse ou modifique a tabela enquanto você a mantiver aberta• False Permite que outros usuários acessem ou modifiquem a tabela enquanto você a mantiver aberta. Este é o default. |
| IndexName | Identifica um índice secundário para Table. Você não pode alterar IndexName enquanto a tabela estiver ativa |
| MasterFields | Especifica o nome dos campos ligados ou campos na propriedade MasterFields para criar um cursor “linkado” a uma tabela secundária Para especificar diversos campos, você deve separar os nomes de campos com uma barra vertical () . |
| MasterSource | Especifica o TDataSource de onde TTable obterá os dados para a tabela master |
| ReadOnly | Põe a tabela em modo somente-leitura. Os valores são: <ul style="list-style-type: none">• True Previne o sistema de gravar alterações ao banco de dados onde a tabela reside• False Permite que o sistema grave alterações ao banco de dados onde a tabela reside |



Você não pode alterar a propriedade ReadOnly enquanto a tabela estiver ativa.

Passos para Adicionar um Componente Table

Execute os passos a seguir para adicionar um componente table ao form:

| Passo | Ação |
|-------|---|
| 1 | Utilizando a página Data Access da Component Palette, adicione um componente Table ao form. |
| 2 | No Object Inspector, localize a propriedade DatabaseName e digite o diretório onde o banco de dados reside, ou digite um nome de alias do banco de dados. |
| 3 | Localize a propriedade TableName e digite o nome da tabela ou selecione uma tabela da lista drop-down. |
| 4 | Adicione um componente DataSource e defina o valor da propriedade de DataSet igual ao do componente Table. |

| | |
|---|--|
| 5 | Adicione componentes Data Control e conecte-os ao componente DataSource para exibir dados da tabela do banco de dados. |
|---|--|

Tutorial: Criando uma Aplicação Utilizando Métodos TDataSet do Componente Table

Introdução

Este processo é um tutorial de exemplo. Você construirá uma aplicação de exemplo utilizando uma tabela chamada COUNTRY.

Esta tabela exibe informação sobre países do mundo inteiro. Ao invés de utilizar o DBNavigator, você utilizará botões padrão e métodos DataSet para fornecer a funcionalidade do DBNavigator.

Este tutorial mostra como utilizar:

- Os métodos First, Next, Prior e Last
- Os métodos BOF e EOF
- Os métodos Edit, Insert e Cancel

Estágios do Tutorial

O processo deste tutorial envolve os seguintes estágios:

| Estágio | Processos |
|---------|---|
| 1 | Adicionar e definir propriedades para os componentes TDataSet |
| 2 | Adicionar e definir propriedades para componentes DBGrid e Button |
| 3 | Criar event handlers OnClick para componentes Button |
| 4 | Executar e testar a aplicação |

Passos para o Estágio 1

Execute os passos a seguir para adicionar e definir propriedades para os componentes TDataSet:

| Passo | Ação |
|-------|--|
| 1 | Abra um novo projeto e grave-o. Quando solicitado, grave a unit como UDSEVENT.PAS e o projeto como PDSEVENT.DPR. |
| 2 | Utilizando a página Data Access da Component Palette, adicione o seguinte ao seu form: <ul style="list-style-type: none"> • Um componente Table • Um componente DataSource |
| 3 | Defina as propriedades para os componentes Table e DataSource como segue: |

| Nome do Componente | Propriedade | Valor |
|--------------------|--------------|------------|
| Table1 | DatabaseName | DBDEMOS |
| | TableName | COUNTRY.DB |
| | Active | True |
| DataSource1 | DataSet | Table1 |
| | AutoEdit | False |

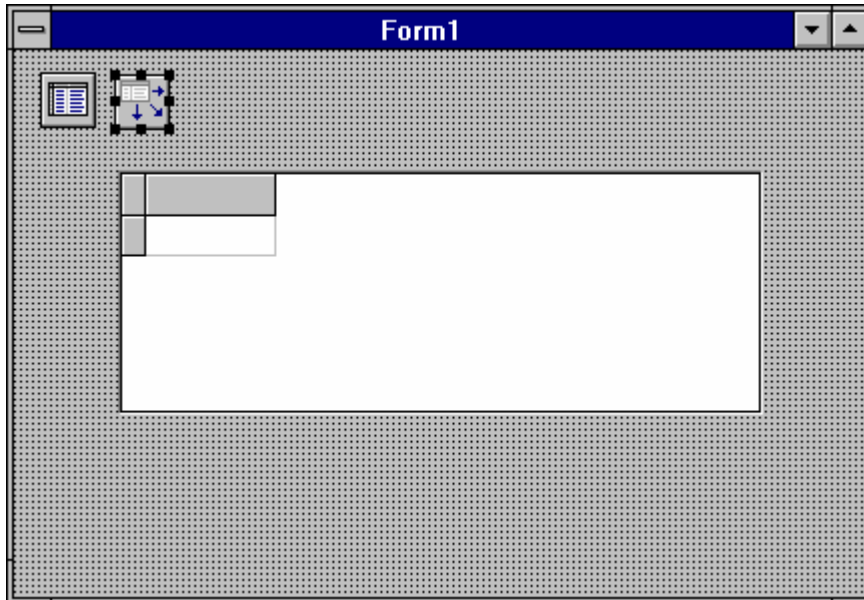
Passos para o Estágio 2

Execute os passos a seguir para adicionar e definir propriedades para os componentes DBGrid e Button:

| Passo | Ação |
|-------|---|
| 1 | Utilizando a página Data Control da Component Palette, adicione um componente DBGrid. |

Introdução ao Delphi

Arranje os componentes para que seu form esteja similar à figura a seguir:



| Passo | Ação |
|-------|--|
| 2 | Utilize a tabela a seguir para definir propriedades do DBGrid para que utilizem o componente DataSource do form: |

| Nome do Componente | Propriedade | Valor |
|--------------------|-------------|-------------|
| DBGrid1 | DataSource | DataSource1 |

| Passo | Ação |
|-------|---|
| 3 | Adicione oito botões ao form, e arranje-os para que seu form esteja similar à próxima figura: |

| Passo | Ação |
|-------|---|
| 4 | Utilize o Object Inspector para definir as seguintes propriedades aos componentes Button: |

| Nome do Componente | Propriedade | Valor |
|--------------------|-------------|-----------|
| Button1 | Caption | Primeiro |
| Button2 | Caption | Último |
| Button3 | Caption | Próximo |
| Button4 | Caption | Anterior |
| Button5 | Caption | Editar |
| Button6 | Caption | Post |
| Button7 | Caption | Cancelarr |

| | | |
|---------|---------|---------|
| Button8 | Caption | Inserir |
|---------|---------|---------|



Passos para o Estágio 3

Execute os passos a seguir para criar event handlers OnClick para os componentes Button:

| Passo | Ação |
|-------|---|
| 1 | Digite o código a seguir para os event handlers OnClick nos botões apropriados. |

| Nome do Componente | Evento | Código |
|--------------------|---------|---|
| Button1 | OnClick | Table1.First; |
| Button2 | OnClick | Table1.Last; |
| Button3 | OnClick | if not Table1.EOF then Table1.Next; |
| Button4 | OnClick | if not Table1.BOF then Table1.Prev; |
| Button5 | OnClick | Table1.Edit; |
| Button6 | OnClick | if Table1.State in [dsEdit, dsInsert] then Table1.Post; |
| Button7 | OnClick | Table1.Cancel; |
| Button8 | OnClick | Table1.Insert; |

Passos para o Estágio 4

Execute os passos a seguir para executar e testar a aplicação:

| Passo | Ação |
|-------|---|
| 1 | Compile e grave a aplicação. Execute e teste cada botão para verificar se o método funciona. |
| 2 | Quando tiver completado o teste, grave e feche o projeto. |

Utilizando o Fields Editor

Introdução

O Fields Editor permite criar uma lista de campo de banco de dados.

Quando um componente DataSet como os componentes Table ou Query é ativado pela primeira vez, uma lista de campo é gerada dinamicamente para dataset baseado nas colunas da tabela ou código SQL. O Fields Editor permite especificar e posteriormente modificar uma lista estática de componentes Field.

Possibilidades do Fields Editor

O Fields Editor permite que você:

- Crie um modelo estático das colunas de uma tabela
- Especificar a ordem das colunas no DataSet
- Especificar o tipo das colunas
- Adicionar a lista estática de campos
- Remover campos da lista
- Modificar propriedades Display de TFields estáticos
- Definir campos calculados
- Definir novos componentes Field baseados nas colunas existentes na tabela

Criando um Modelo Estático de uma Tabela de Banco de Dados

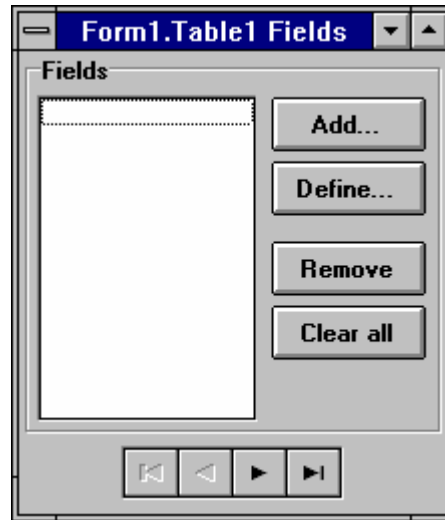
Utilize o Fields Editor quando quiser criar um modelo estático das tabelas do banco de dados. Um modelo estático não é alterado quando modificações são feitas na tabela física no banco de dados.

Quando você adiciona colunas utilizando o Fields Editor, objetos TFields são criados para cada campo adicionado ao DataSet. Após adicionar campos utilizando o Fields Editor você pode visualizar estes campos no Object Inspector. Cada objeto TField possui um conjunto de propriedades, eventos e métodos que você pode utilizar em sua aplicação.

Passos para iniciar o Fields Editor

Execute os passos a seguir par iniciar o Fields Editor:

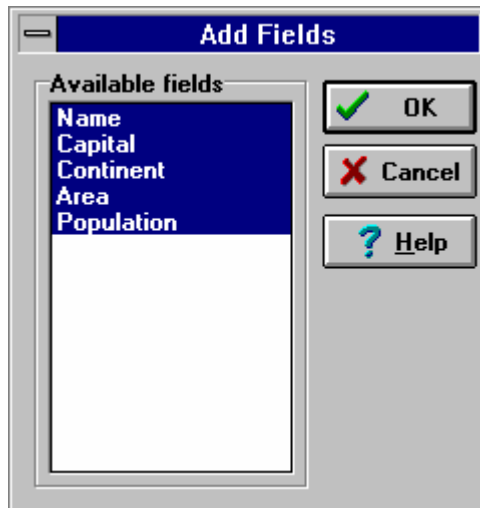
| Passo | Ação |
|-------|--|
| 1 | Adicione um componente Table ou Query ao seu form. |
| 2 | Defina a propriedade DatabaseName do componente Table ou Query. |
| 3 | Execute um dos seguintes: -Defina a propriedade TableName do componente Table -Defina a propriedade SQL do componente Query. |
| 4 | Selecione o componente DataSet no form, e pressione o botão direito do mouse para exibir o SpeedMenu. |
| 5 | No SpeedMenu, selecione Fields editor. A primeira janela do Fields Editor aparecerá, como segue: |



Passos para Criar uma Lista Estática de Campos

Execute os passos a seguir para criar uma lista estática de campos após abrir o Fields Editor:

| Passo | Ação |
|-------|--|
| 1 | No quadro de diálogo Fields Editor, dê um clique em Add. Cada uma das colunas na tabela ou query aparecem selecionadas no quadro de diálogo Add Fields, como segue: |



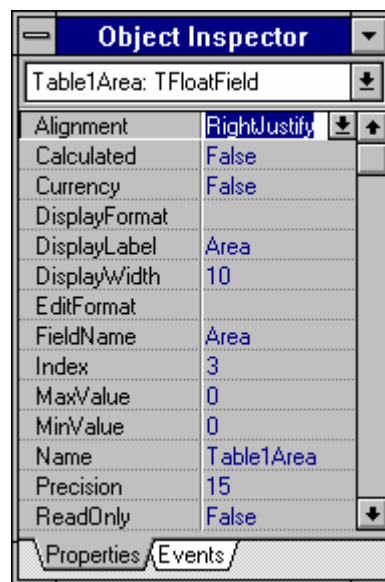
| Passo | Ação |
|-------|---|
| 2 | Selecione os campos que você queira adicionar e dê uma clique em OK. O quadro de diálogo a seguir é exibido: |



| Passo | Ação |
|-------|---|
| 3 | Dê um clique em Add para adicionar campos estáticos, adicionar à lista de campos no dataset. |
| 4 | Dê um clique em Define para criar um novo campo baseado em um campo existente, ou criar um campo calculado. |
| 5 | Dê um clique em Remove para deletar um campo estático da lista de campos no dataset. |

Propriedades do Componente Field

Após utilizar o Fields Editor, quaisquer campos adicionados ao dataset são refletidos no Object Inspector. A figura a seguir mostra o tipo de objeto TStringField (componente Field) e suas propriedades associadas.



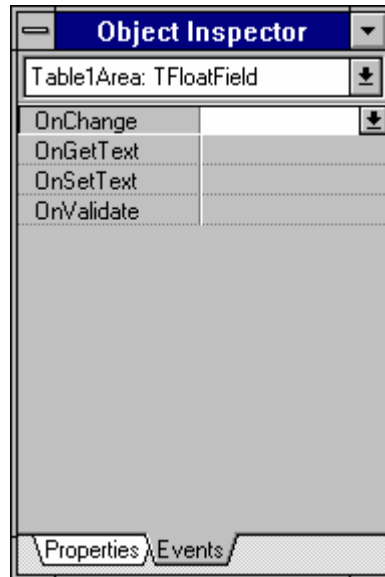
Descendentes de TField

Quando você utiliza o Fields Editor, o Delphi cria objetos estáticos que correspondem aos campos visíveis no Object Inspector. Estes objetos são descendentes do tipo de objeto TField. A tabela a seguir descreve os objetos descendentes TField:

| Descendente | Descrição |
|-----------------------|---|
| TStringField | Dado texto de tamanho fixo, até 255 caracteres |
| TIntegerField | Números inteiros de -2,147,483,648 a 2,147,483,647 |
| TSmallField | Números inteiros de -32768 a 32767 |
| TWordField | Números inteiros de 0 a 65535 |
| TFloatField | Números reais com grandezas absolutas de 1,2x10e-324 a 1.7x10e308 com precisão de 15 a 16 dígitos |
| TCurrencyField | Valores monetários. Números reais com grandezas absolutas de 1.2x10e-324 a 1.7x10e308 com precisão de 15 a 16 dígitos |
| TBCDField | Números reais com número fixo de casas decimais. Com precisão de 18 dígitos. O intervalo dos valores depende do número de casas decimais. |
| TBooleanField | Valor True (verdadeiro) ou False (falso) |
| TDateTimeField | Valor de data e hora |
| TDateField | Valor de data |
| TTimeField | Valor de tempo |
| TBlobField | Campo de dado arbitrário, sem limite de tamanho |
| TBytesField | Campo de dado arbitrário, sem limite de tamanho |
| TVarBytesField | Campo de dado arbitrário de até 65535 caracteres, com tamanho real armazenado nos primeiros dois bytes |
| TMemoField | Texto de tamanho arbitrário |
| TGraphicField | Gráfico de tamanho arbitrário, tal como bitmap |

Eventos do Objeto Tipo TStringField

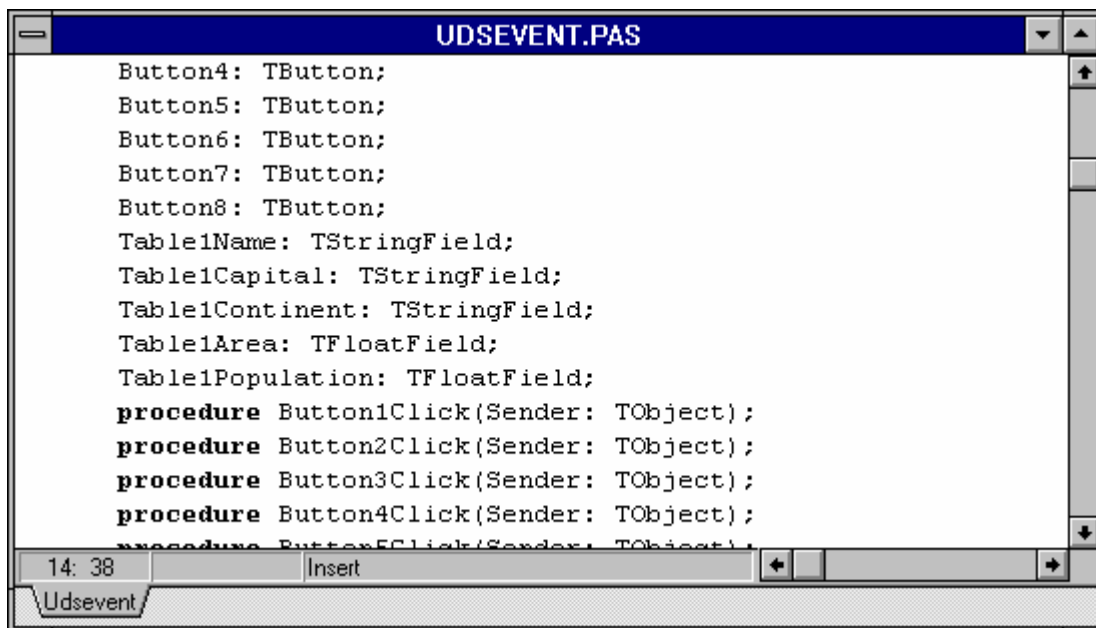
A figura a seguir mostra eventos que um objeto tipo TStringField pode reconhecer:



Campos Estáticos

Quando você utiliza o Fields Editor, são adicionados campos estáticos ao tipo de objeto TForm1 do form . Na figura a seguir, o Fields Editor adicionou os seguintes campos ao TForm 1:

Table1Name, Table1Size, Table1WIGHT, Table1AREA, e Table1BMP:



```
UDSEVENT.PAS
Button4: TButton;
Button5: TButton;
Button6: TButton;
Button7: TButton;
Button8: TButton;
Table1Name: TStringField;
Table1Capital: TStringField;
Table1Continent: TStringField;
Table1Area: TFloatField;
Table1Population: TFloatField;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
```

Utilizando o Componente Database Grid

Introdução

Como você já pode ter observado, o componente DBGrid (databasegrid) fornece uma maneira conveniente de se exibir diversas linhas de dados de um componente Table ou Query. Sua aplicação pode utilizar o DBGrid para inserir, deletar, editar ou exibir dados de um banco de dados. Combinado com o DBNavigator, o DBGrid permite protipar e exibir rapidamente dados do banco de dados.

Até agora, você tem visto exemplos utilizando o componente DBGrid.

Este tópico oferece sugestões para:

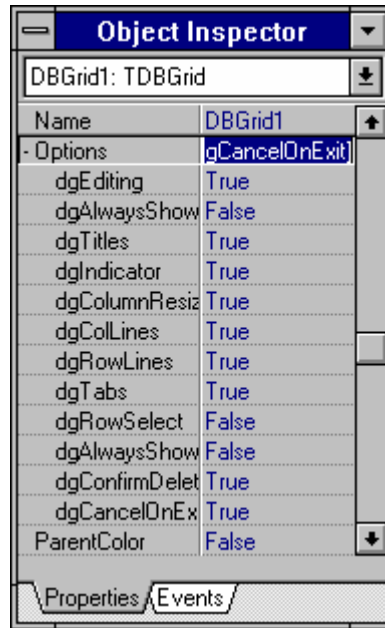
- Definir a propriedade Options do Componente DBGrid
- Definir características de exibição de campo para o componente DBGrid



Uma lista completa das propriedades, métodos e eventos do DBGrid encontram-se no Help Online.

Propriedades de Options do DBGrid

A figura a seguir mostra um conjunto de propriedades que compõem a propriedade Options do componente DBGrid. O conjunto aparece quando você clica o sinal (+) na frente da propriedade Options.



Descrição da Propriedade Options

Você pode alterar a aparência e comportamento de uma grade alterando valores da propriedade Options, utilizando o Object ou escrevendo código. A tabela a seguir descreve as definições de Options do componente DBGrid:

| Propriedade Options | Descrição Quando Definido para True... |
|-----------------------|--|
| dgEditing | O usuário pode editar dados na grade. Quando a propriedade ReadOnly do DataSet for True e dgEditing também for True, os usuários podem utilizar a tecla Insert para inserir uma linha em branco, ou pressionar a tecla de seta para baixo quando posicionado no final da grade para adicionar uma linha em branco, embora não possam digitar texto na nova linha. |
| dgTitles | Os títulos das colunas são visíveis. |
| dgIndicator | Um pequeno ponteiro fica visível para indicar a coluna atual. |
| dgColumnResize | As colunas podem ser reajustadas. |
| dgCollLines | As linhas entre as colunas ficam visíveis |
| dgRowLines | As linhas entre as linhas ficam visíveis. |
| dgTabs | Os usuários podem pressionar a tecla Tab e Shift+ Tab para se moverem entre as colunas da grade. |

Propriedade Options como Tipo Set

A propriedade Options do componente DBGrid é um tipo set. Você pode modificar a propriedade Options utilizando operadores set. As linhas a seguir são comandos Object Pascal válidos:

```
DBGrid1.Options := DBGrid1.Options + [dgTitles];
DBGrid1.Options := DBGrid1.Options - [dgTitles, dgRowLines];
```

Definindo Características de Exibição de Campo para o Componente DBGrid

Geralmente no desenvolvimento de uma aplicação, você precisa controlar o comportamento de campos no componente DBGrid. O comportamento default do DBGrid é determinar dinamicamente o tamanho do campo e permitir ao usuário o uso do mouse para reajustar o tamanho do campo. A chave para obter o controle das características de exibição do DBGrid ou outro componente data-aware é criar uma lista estática de componentes Field. Uma vez criados componentes para cada um dos campos no dataset, você pode definir o seguinte:

- Tamanho de exibição
- Formato de exibição
- Máscara de Edição
- Rótulos de exibição

Passos para Definir Tamanho de Exibição

Execute os passos a seguir para criar campos de exibição de tamanho fixo no componente DBGrid:

| Passo | Ação |
|-------|---|
| 1 | Abra o Fields Editor para o componente TDataSet que será exibido no componente DBGrid. |
| 2 | Adicione cada um dos campos de banco de dados que você queira no dataset. Este passo cria componentes TFields estáticos para os campos a serem exibidos no DBGrid. |
| 3 | Localize o componente DBGrid no Object Inspector, e defina a opção <code>dgColumnReSize</code> para False. Como alternativa, você pode escrever comandos Object Pascal em sua aplicação para alterar esta propriedade. |
| 4 | Altere o tamanho da coluna exibida de uma destas maneiras: -Utilize o mouse para arrastar e reajustar o tamanho das colunas no DBGrid -Defina a propriedade <code>DisplayWidth</code> para cada um dos componentes Field que o Fields Editor adicionou. |

Definindo a Propriedade DisplayLabel

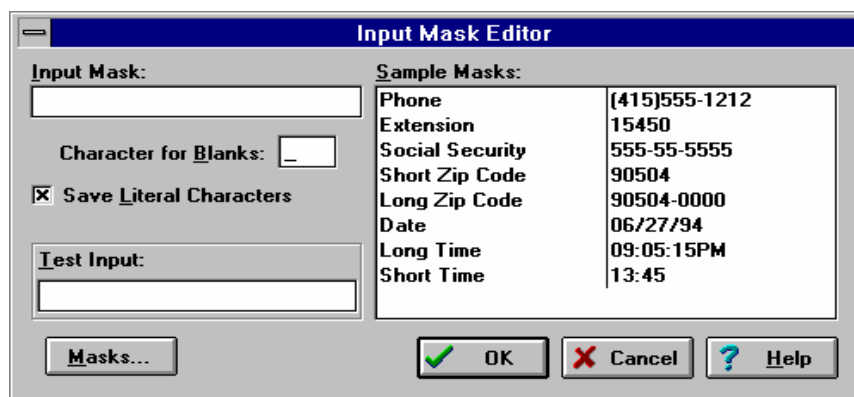
Uma vez que você utilizou o Fields Editor para gerar um conjunto de objetos TField para o dataset, você pode utilizar o Object Inspector para definir a propriedade `DisplayLabel` do componente Field. O componente `DisplayLabel` do componente Field.

Definindo Propriedade DisplayMask

Campos, Float, Integer e Date possuem uma propriedade `DisplayMask`. Você pode utilizar esta propriedade para formatar a exibição em um DBGrid ou outro componente Data Control. Por exemplo, o formato de exibição `mm-dd-yy` pode ser utilizado para exibir um campo data.

Definindo a Propriedade EditMask

Os componentes Field possuem uma propriedade `EditMask` que você pode definir sempre que digitar dados em um DBGrid ou outro componente Data Control. Para definir uma propriedade `EditMask`, localize o componente Field no Object Inspector, e clique a propriedade `EditMask`. O quadro de diálogo Input Mask Editor é exibido, como segue:



Para testar sua máscara de edição, digite um valor no campo **Test Input**.

Utilizando o Componente Query

Introdução

Até agora, este capítulo cobriu as seguintes informações:

- As capacidades do objeto DataSet e suas propriedades, métodos e eventos
- O Fields Editor, que permite definir objetos correspondentes aos campos no dataset

O componente Query, como Table, é derivado do objeto TDataSet. Desta forma, tudo sobre TDataSet aplica-se ao componente Query.

Esta seção explica o seguinte:

- O componente Query
- Propriedades, métodos e eventos importantes do componente
- Uma aplicação de exemplo utilizando o componente Query

Componente Query

O componente Query permite utilizar comandos SQL para executar o seguinte:

- Especificar ou criar datasets que possam ser exibidos
- Inserir linhas
- Editar e atualizar colunas
- Deletar linhas

O componente Query gerencia a comunicação com o BDE e serve como interface entre o BDE e os componentes DataSource (TDataSource) em seus forms

O Componente Query Relacionado com o Componente DataSource

Como com o componente Table, um componente DataSource é anexado ao componente Query para gerenciar a comunicação entre o componente Data Control e o componente Query. Entre os componentes Data Control incluem DBGrid, DBEdit, e DBLooup.

Uma aplicação típica possui um componente DataSource para cada componente Query.

Propriedade do Componente Query

A tabela a seguir descreve diversas propriedades importantes do componente Query:

| Propriedade | Descrição |
|---------------------|--|
| Active | Abre ou fecha uma query. Os valores são: <ul style="list-style-type: none"> • True Abre uma query, o que faz com que o comando SQL seja executado, como no exemplo: <i>{Abre a query}</i> <i>Query1.Active := True;</i> • False Fecha uma query, como segue: <i>{Fecha a query}</i> <i>Query1.Active := False;</i> |
| DatabaseName | Identifica o alias do banco de dados ou o drive e diretório de um banco de dados local. A propriedade DatabaseName pode ser definida somente quando a query não estiver ativa, como no exemplo a seguir: <i>{Fecha o DBDataSet}</i> <i>Query1.Active := False;</i> <i>Query1.DatabaseName := 'Demos';</i> <i>Query1.Active := True;</i> |
| Fields | Suportam os campos no componente Query. É uma propriedade somente durante execução |

| | |
|--|--|
| | e é utilizada para examinar ou modificar um determinado campo, como no exemplo a seguir: <i>Query1.Fields[3].AsString := 'ÓK';</i> |
| DataSource | Fornecer valores para queries parametrizadas. Uma query parametrizada é uma onde um ou mais valores na condição de seleção não é conhecida |
| Params | Guardam os parâmetros para uma query parametrizada. Uma query parametrizada envolve um ou mais valores na condição de seleção que não são conhecidas até a execução, como no exemplo a seguir: <i>Select * from Orders Where CustNo = : SomeNo</i> Esta é uma propriedade de somente-leitura, durante a execução. Consulte o Help Online para maiores informações sobre queries parametrizadas. |
| SQL | Guarda o texto do comando de query SQL |
| EOF (End of File) BOF (Beginning of File) | Propriedade somente-leitura com valores a seguir: <ul style="list-style-type: none"> • EOF é True quando você tenta mover para além da última linha do dataset. • BOF é True quando o componente DataSet é aberto, ou quando o ponteiro do TDataSet da linha atual estiver na primeira linha. |

Métodos do Componente Query

A tabela a seguir descreve alguns métodos do componente Query:

| Método | Descrição |
|----------------|--|
| ExecSQL | Executa comando SQL atribuído à propriedade SQL se o comando não retornar dados. Quando estiver inserindo, atualizando ou deletando dados, você deve utilizar este método. Se estiver executando um comando de seleção, utilize o método Open. A seguir um exemplo da utilização do método ExecSQL: <i>Query1.Close; Query1.Clear; Query1.SQL.Add ('Delete emp where empno = 1010'); Query1.ExecSQL;</i> |
| Open | Abre o componente Query. É equivalente a definir a propriedade Active para True. A seguir um exemplo utilizando o método Open: <i>Query1.Open;</i> |
| Close | Fecha o componente Query fazendo com que quaisquer atualizações pendentes sejam efetuadas no banco de dados. Chamar Close é equivalente a definir a propriedade Active para False. A seguir mostramos um exemplo utilizando o método Close: <i>Query1.Close;</i> |
| Prepare | Traduz a propriedade SQL para criar a propriedade Text para Submeter ao servidor. O método Prepare também envia a requisição ao servidor para propósito de otimização, embora nenhum valor parametrizado esteja incluído. A requisição inteira com parâmetros não é submetida até que o método Open ou ExecSQL sejam chamados. Se você não chamar Prepare explicitamente, o Delphi chama Prepare implicitamente quando utilizar o comando em ExecSQL. A seguir um exemplo utilizando o método Prepare: <i>Query1.Close; Query1.SQL := 'Delete emp where empno = : empno'; Query1.Prepare;</i> |

Métodos TQuery Herdados de TDataSet

O objeto TDataSet fornece ao componente Query uma grande variedade de métodos. Alguns dos mais importantes são mostrados na tabela a seguir:

| Método | Descrição |
|--|---|
| First Last Next Prior MoveBy | <p>Permite navegar ou alterar a linha atual do dataset. A seguir um exemplo utilizando diversos destes métodos:</p> <pre> Query1.First While not Query1.EOF do begin {Seu código aqui} Table1.Next; End; </pre> <p>O método MoveBy move um número determinado de linhas. Por exemplo:</p> <ul style="list-style-type: none"> • Query1.Moveby(3) move 3 linhas para cima. • Query1.Moveby (-2)move 2 linhas para trás |
| Insert Edit Delete Append Post Cancel | <p>Permite modificar o conjunto resultante de uma query. O método Insert permite adicionar linhas à tabela, como no exemplo a seguir:</p> <pre> Query2.Insert; Query2.Fields [0] .AsString := 20; Query2.Fields [1] .AsString := 'News'; Query2.Fields [2] .AsString := '5 horas'; Query2.Post; </pre> <ul style="list-style-type: none"> • O método Post faz com que as operações Insert, Update, ou Delete ocorram. • O método Cancel faz com que um Insert, Delete, Edit ou Append não completado seja cancelado. |
| SetKey GotoKey | <p>Pesquisa através dos datasets, como segue:</p> <ul style="list-style-type: none"> • O método SetKey alterna o dataset para o modo de procura. Neste modo, a propriedade Fields tem uso especial. • O método GotoKey inicia uma procura por um valor que coincida com o valor encontrado em Fields[n]. Fields[n] contém o valor que você está procurando e que ocorre na primeira coluna da tabela ou dataset. Você pode procurar por valores em outras colunas definindo a coluna correspondente. <p>O exemplo a seguir mostra a utilização dos métodos SetKey e GotoKey:</p> <pre> Query1.SetKey; Query1.Fields [0] .AsString := Edit1.Text; Query1.GotoKey; </pre> |
| FreeBookmark GeTBookMark GotoBookmark | <p>Permite criar um marcador em uma linha na tabela ou query e retornar posteriormente para esta linha, como segue:</p> <ul style="list-style-type: none"> • O método FreeBookmark libera espaço alocado para o marcador • O método GeTBookMark aloca um marcador para a linha atual da tabela. • O método GotoBookmark altera a localização na tabela para a linha indicada por um marcador alocado previamente . <p>Os métodos Bookmark utilizam o objeto tipo TBookMark, como no exemplo a seguir:</p> <pre> Var Marker : TBookMark; begin Marker := Query2.GeTBookMark; Query2.GotoBookmark (Marker); Query2.FreeBookmark (Marker); </pre> |

Eventos do Componente Query Derivados de TDataSet

O componente Query responde aos eventos herdados do objeto TDataSet. A tabela a seguir descreve estes eventos:

| Eventos | Descrição |
|--|---|
| OnOpen OnClose OnNewRecord BeforeInsert AfterInsert BeforeInsert AfterEdit BeforePost AfterPost OnCancel OnDelete | Permite construir e controlar o comportamento da aplicação de banco de dados. Por exemplo: <ul style="list-style-type: none">• O event handler BeforePost valida os campos de um registro antes de inserir ou atualizar os dados• O evento AfterPost é útil para gravar um registro de auditoria quando necessário.• O evento OnDelete é útil para escrever código que efetue a deleção em cascata quando apropriado. |

Tutorial: Criando uma aplicação Utilizando o Componente Query

Introdução

Esta seção fornece um aprendizado na utilização do componente Query. O componente Query possui muitas características avançadas. Entretanto, este exemplo focaliza as características básicas deste componente. Este tutorial demonstra como utilizar um componente Query para criar um dataset, e métodos TDataSet para executar operações no dataset.

Estágios do Tutorial

Este tutorial envolve os seguintes estágios:

| Estágio | Processo |
|---------|---|
| 1 | Construir uma aplicação e exibir os dados utilizando um comando SQL |
| 2 | Modificar uma aplicação para consultar um banco de dados, baseado no campo CustID |

Passos para o Estágio 1

Execute os passos a seguir para construir uma aplicação de clientes e exibir dados do cliente utilizando um comando SQL:

| Passo | Ação |
|-------|--|
| 1 | Abra um novo projeto e grave-o. Quando solicitado, grave a unit como EX7SQL.PAS e o projeto como EX7P.DPR. |
| 2 | Utilizando a página Data Access page da Component Palette, adicione os seguintes ao seu form: <ul style="list-style-type: none">• Um componente Query• Um componente DataSource |
| 3 | Defina as propriedades dos componentes Query e DataSource, como segue: |

| Nome do Componente | Propriedade | Valor |
|--------------------|--------------|---------|
| Query1 | DatabaseName | DBDEMOS |
| | RequestLive | True |
| DataSource1 | DataSet | Query1 |

| Passo | Ação |
|----------|--|
| 4 | Utilizando a página Data Controls da Component Palette, adicione os seguintes ao seu form: <ul style="list-style-type: none"> • Um componente DBGrid • Um componente DBNavigator |
| 5 | Utilize a tabela a seguir para definir propriedades dos componentes DBNavigator e DBGrid para que utilizem o componente DataSource do form: |

| Nome do Componente | Propriedades | Valor |
|--------------------|--------------|-------------|
| DBGrid1 | DataSource | DataSource1 |
| DBNavigator | DataSource | DataSource1 |

| | |
|----------|--|
| 6 | <p>Crie um event handler OnActivate para o form utilizando o Object Inspector. Digite o código abaixo no handler OnActivate:</p> <pre>Query1.SQL.Add ('Select * from customer'); Query1.Open;</pre> <p>Este código adiciona uma instrução SQL à propriedade SQL do componente Query. Você pode definir esta instrução SQL como uma propriedade utilizando o Object Inspector.</p> <p>Após adicionar a instrução SQL, seu event handler deve ser:</p> <pre>Query1.SQL.Add('Select * From Customer'); Query1.Open;</pre> |
| 7 | Quando sua aplicação for compilada satisfatoriamente. grave-a. Depois, execute sua aplicação e teste-a utilizando o Navigator para atualizar e inserir linhas no dataset. |
| 8 | Quando tiver terminado, feche a aplicação. |

Passos para o Estágio 2

Execute os passos a seguir para modificar uma aplicação para consultar uma base de dados, baseado em uma identificação do cliente:

| Passo | Ação |
|----------|---|
| 1 | Utilizando a página Standard do Component Palette, adicione os seguintes ao form: <ul style="list-style-type: none"> • Dois componentes botão • Um componente Edit • Um componente Label |
| 2 | Utilize a informação da tabela a seguir para definir propriedades destes componentes: |

| Nome do Componente | Propriedade | Valor |
|--------------------|-------------|------------|
| Button1 | Caption | Query |
| Button2 | Caption | Exec Query |
| Edit1 | Text | (Empty) |
| | Visible | False |
| Label | Caption | IdCliente |
| | Visible | False |

| Passo | Ação |
|----------|---|
| 3 | Adicione o seguinte event para o evento OnClick do Button1: <pre>Edit1.Visible := True; Label1.Visible := True;</pre> |
| 4 | Adicione o seguinte event handler para o evento OnClick do Button2 <pre>Query1.Close; Query1.SQL.Clear;</pre> |

```
Query.SQL.Add  
('Select * from customer where '+'CustNo = '+' Edit1.Text);  
Query.Open;  
Edit1.Visible := False;  
Label1.Visible := False;
```

| Passo | Ação |
|-------|--|
| 5 | Compile e grave sua aplicação. Execute e teste-a através do seguinte: <ul style="list-style-type: none">• Selecione um número de cliente da lista de clientes e dê um clique em Query.• Digite o número e dê um clique em Exec Query para exibir um único cliente no grid. |

Utilizando o Visual Query Builder

Introdução

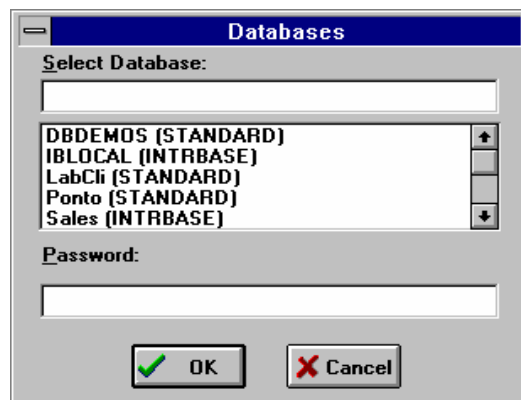
O Visual Builder (VQB) é uma ferramenta para construção de queries baseadas em SQL. Com esta ferramenta, você pode construir queries complexas com pouco ou nenhum conhecimento de SQL. O VQB permite construir estas queries complexas iniciando com um query simples, permitindo executar a query e fornecer ferramentas para refiná-la. Você constrói queries incrementalmente adicionando expressões, tabelas, campos e relacionamentos até obter os resultados desejados.

Ativando o Visual Query Builder

Para ativar o Visual Query Builder, você deve utilizar um componente Query. Uma vez inserido um componente Query em seu form, você pode ativar o Visual Query Builder selecionando o componente Query e dando um clique com o botão direito do mouse. O SpeedMenu do Componente aparecerá. Então, selecione **Query Builder**.

Selecionando um Alias de banco de dados

O item de menu **Query Builder** exibe o quadro de diálogo Databases, exibido na figura a seguir. Este quadro de diálogo permite selecionar um banco de dados e logar-se nele. O logon pode ser em um servidor de banco de dados local ou remoto durante o design.

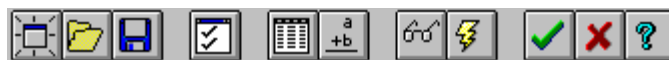


Janela do Visual Query Builder

Após logar-se ao banco de dados, a janela do Visual Query Builder é apresentada. O quadro de diálogo Add sobrepõe-se.

Toolbar do Visual Query Builder

A Toolbar do Visual Query Builder aparece para selecionar as operações a serem executadas. A Toolbar aparece, como segue:



Descrição da Toolbar do Visual Query Builder

A tabela a seguir descreve os botões da Toolbar do Visual Query Builder respectivamente:

| Botão | Descrição |
|-------------------|---|
| New | Inicia uma nova query |
| Open | Abre um arquivo de query |
| Save As | Grava uma query em um arquivo |
| Options | Exibe o quadro de diálogo Options, que permite definir diversas opções de query. Por exemplo, você pode definir uma opção para remover linhas duplicadas. |
| Table | Exibe o quadro de diálogo Add Table, que permite adicionar tabelas à instrução SQL |
| Expression | Exibe o quadro de diálogo Expression, que permite criar expressões SQL, por exemplo, upper (Nome) ou Sum (Custo_Item) |
| SQL | Exibe a janela SQL Statement, que contém, a instrução SQL atual. |
| Run | Executa a instrução SQL atual e exibe os resultados |
| OK | Define a propriedade SQL do componente Query para a instrução SQL atual no Visual Query Builder |
| Cancel | Sai do Visual Query Builder sem definir a propriedade SQL do componente Query |
| Help | Exibe o Help online do Visual Query Builder |

Quadro de Diálogo Add Table

Quando você abre o Visual Query Builder, o quadro de diálogo Add Table aparece. Este quadro permite adicionar tabelas à query. Você adiciona tabela à query quando inicia o processo de construção da query ou quando quer modificar uma query existente.

O quadro de diálogo Add Table lista os nomes de todas as tabelas no banco de dados atual. Se você quiser incluir tabelas de sistema, dê um clique no check box **Include System Tables**.

Passos para Adicionar Tabelas à Query

Execute os passos a seguir para adicionar uma ou mais tabelas ao espaço de trabalho de Visual Query Builder a serem incluídas na query:

| Passo | Ação |
|----------|--|
| 1 | Insira um componente Query ao form, e dê um clique com o botão direito do mouse sobre o componente para exibir o SpeedMenu. |
| 2 | Selecione Query Builder para exibir a janela do Visual Query Builder. |
| 3 | Se o quadro de diálogo Add Table não aparecer na frente da janela do Visual Query Builder, dê um clique sobre o botão Table na Toolbar para exibi-lo. |
| 4 | Selecione o nome da tabela da lista de tabelas exibida no quadro de diálogo Add Table, e dê um clique em Add. A tabela aparece no espaço de trabalho da janela do Visual Query Builder. |
| 5 | Repita o passo 4 até que todas as tabelas sejam adicionadas à query, e dê um clique em Close. |

Adicionando Colunas à Query

Para adicionar uma coluna de uma das tabelas à query, você pode efetuar um dos seguintes:

- Selecionar o nome da coluna e arrastar a coluna e soltá-la na grade da query, na parte inferior da janela.
- Dê um duplo-clique no nome da coluna para inserí-la na grade da query.

Especificando uma Condição de Join

Geralmente você precisará combinar informações de diversas tabelas. Por exemplo, você pode querer consultar o banco de dados para encontrar todos os pedidos, o nome dos clientes, a data de pedido e o fornecedor de um dos itens. Esta query envolverá colunas das tabelas CUSTOMER, VENDORS, ITEMS e ORDERS. Para construir uma query deste tipo e complexidade, você precisará especificar como as diversas tabelas deverão ser combinadas. Com o Visual Query Builder você especifica as colunas a serem combinadas arrastando o nome de uma coluna e soltando-a sobre o nome da coluna onde a combinação será feita. Quando completar esta operação, uma linha será desenhada no espaço de trabalho da query, ligando as colunas das duas tabelas.

Revisando e Editando o Critério de Combinação

Você pode revisar e editar o critério de combinação dando um duplo-clique sobre a linha indicando a combinação no espaço de trabalho da query. O quadro de diálogo Join aparecerá.

Especificando Critério de Seleção de uma Query

Para especificar um critério de seleção de uma query, utilize a linha Criteria da grade da query.



Se a linha Criteria não estiver visível, utilize a barra de rolagem.

A linha Criteria permite qualquer expressão válida de query dentro da cláusula WHERE de uma instrução SQL. Expressões válidas incluem os seguintes operadores:

| Operador | Significado |
|----------|---|
| = | Igual |
| > | Maior que |
| < | Menor que |
| != | Não igual |
| like | Comparação de string de caracteres com separação de padrão |
| between | Não menor que o valor inicial e não maior que o valor final |
| in | Encontrado em uma lista |

Expressões digitadas na linha Criteria são condições AND. Por exemplo, se você tiver uma tabela com um nome de coluna e quiser encontrar todas as entradas na coluna nome que iniciem com C, você digitaria nome like 'C. /.' na linha Criteria. O efeito seria equivalente a adicionar AND nome LIKE 'C. /.' à cláusula WHERE da instrução SQL. Condições OR São digitadas como expressões na linha OR (abaixo da linha Criteria) da grade da query.

Ordenando Resultados da Query

Você pode ordenar resultados da query na ordem ascendente ou descendente em uma determinada coluna. Para especificar a ordem de ordenação, posicione o ponteiro do mouse sobre a coluna, na linha Sort da grade

da query. Dê um clique com o botão direito do mouse e selecione. Ascending ou Descending no menu pop-up. Você pode especificar até oito colunas para ordenação.

Agrupando Resultados de Query

Você pode agrupar resultados de query utilizando a linha Option da grade da query. Para especificar uma opção para uma coluna, posicione o ponteiro do mouse sobre a coluna na linha Option. Dê um clique com o botão direito do mouse para exibir o SpeedMenu Option. As opções são funções de agrupamento: Show, Average, Count, Maximum, Minimum, Sum, Group (1), e Group (2).

Especificando Condições de Agrupamento

Você pode especificar condições de agrupamento utilizando a linha Group Condition da grade da query. Condições de agrupamento especificam operações para agregação de colunas. Por exemplo, Você pode querer linhas de dados onde a coluna Average Cost, seja maior que zero.. A linha Group Condition é equivalente a adicionar uma expressão com a cláusula HAVING em uma instrução SELECT do SQL agrupado (cláusula GROUP BY).

Definindo Expressões de Query

O Visual Query Builder permite definir expressões como parte de uma query. Expressões podem ser cálculos sobre valores de dados numéricos ou expressões de strings, tais como concatenação ou uma substring.

Para definir uma expressão, dê um clique no botão Expression na Toolbar. O quadro de diálogo Expression aparecerá

O quadro de diálogo Expression permite desenvolver expressões SQL. Você pode especificar operadores aritméticos, tais como:

- + (sinal “mais” ,de adição)
- - (sinal”menos” , de subtração)
- * (asterisco, para multiplicação)
- / (barra, para divisão)

Você pode incluir nomes de colunas e agregar expressões, tais como avg, count, min,max,e sum. Ou você pode editar expressões manualmente ou construir uma, utilizando o quadro de edição Expression.

Definindo Opções para Queries

O quadro de diálogo Options é utilizado para especificar opções para instruções SQL. Você pode especificar que registros duplicados sejam removidos . Esta opção tem o efeito de utilizar a instrução SQL DISTINCT.

Para especificar uma opção, dê um clique no botão Options na Toolbar. O quadro de diálogo Options aparece.

Exibindo Resultados de Query

Você pode executar a query que o Visual Query Builder gera. O resultado da query é exibido na janela Result Window. Esta janela permite verificar se as colunas da query, critério de seleção, agrupamento e ordenação foram especificados corretamente pela query.

Para executar a query, dê um clique no botão Run da Toolbar. Uma Result Window aparecerá.

Exibindo a Instrução SQL para Query

Você pode checar instruções SQL para uma query visualizando a janela SQL Statement. A janela exibe a instrução SQL SELECT associada com a query atual. Conforme adicionar ou alterar colunas de query, critério ou ordenação, a SQL Window é atualizada automaticamente. Visualizar as instruções SQL fornece um feedback imediato sobre a construção da query e o auxilia no aprendizado da sintaxe SQL.

Para checar as instruções, dê um clique no botão SQL na Toolbar. A janela SQL statement aparecerá.

Definindo e Utilizando Campos Calculados

Introdução

Um campo calculado é um campo que exibe valores gerados pelo programa. Por exemplo, você pode ter uma tabela de banco de dados contendo informações sobre objetos retangulares. Cada linha possui uma coluna de altura e largura, mas você precisa exibir a área dos objetos retangulares. A solução é calcular a área através das alturas e larguras.

O Delphi permite utilizar campos calculados em suas aplicações.

Implementar um campo calculado envolve o seguinte processo:

| Estágio | Processo |
|---------|--|
| 1 | Definir o campo calculado |
| 2 | Escrever código para o campo calculado |

Passos para o Estágio 1

Campos calculados são definidos com a assistência do Fields Editor.

O Fields Editor trabalha com os componentes Table e Query do TDataSet. Quando você define um campo calculado para um dataset, o Fields Editor adiciona um novo objeto campo ao dataset.

Execute os passos abaixo para definir um campo calculado

| Passo | Ação |
|-------|--|
| 1 | Selecione o componente Query ou Table (derivado TDataSet), e inicialize o Fields Editor. Para TTable, este passo assume que as propriedades DatabaseName e TableName estão definidas. Para TQuery, este passo assume que as propriedades DatabaseName e SQL estão definidas. |
| 2 | Adicione campos para exibição ou cálculo no quadro de diálogo Fields Editor. |
| 3 | Dê um clique no botão Define no Fields Editor. O quadro de diálogo Define aparecerá. |
| 4 | Digite um nome para o campo calculado na caixa de texto Field Name. Conforme digitar o nome, ele é exibido concatenado ao nome do componente no campo Component name. Você pode alterar este nome se desejar. |
| 5 | Selecione uma entrada da caixa de lista Field type. |
| 6 | Habilite Calculated. |
| 7 | Dê um clique em OK para completar a definição do campo calculado. |

Passos para o Estágio 2

Uma vez definido o campo, você fornece valores para o campo escrevendo código para ele.

Execute os passos abaixo para implementar um campo calculado:

| Passo | Ação |
|-------|--|
| 1 | Selecione o componente DataSet (TQuery ou TTable) no Object Selector do Object Inspector. |
| 2 | Exiba a página Events do Object Inspector |
| 3 | Dê um duplo-clique sobre a coluna do event handler do evento OnCalcFields para criar e exibir a procedure OnCalcFields no Code Editor. |
| 4 | Digite o código que define o valor e propriedades do campo calculado. |
| 5 | Compile e grave sua aplicação. Execute e teste-a para certificar-se que o campo está sendo calculado corretamente. |

Tutorial: Criando uma Aplicação Utilizando um Campo

Introdução:

A melhor maneira de se aprender a trabalhar com campos calculados é utilizando um exemplo. Nesta seção você exibirá informação sobre países do mundo. As tabelas deste exemplo, Country, contém dois campos chamados Area e Population, entre outros.

Passos para Exibir a População por Kilometro Quadrado

Assumindo que o campo Area esteja em Kilometros quadrados execute os passos a seguir para exibir a população por kilometro quadrado em um form:

| Passo | Ação |
|-------|--|
| 1 | Abra um novo projeto e grave-o. Quando solicitado, grave a unit como EX7BMAIN.PAS e o projeto como EXAMP7B.DPR. |
| 2 | Utilizando a página Data Access da Component Palette, adicione um componente Table e DataSource ao seu form. |
| 3 | Defina as propriedades para os componentes Table e DataSource como segue: |

| Nome do Componente | Propriedade | Valor |
|--------------------|--------------|------------|
| Table1 | DatabaseName | DBDEMOS |
| | TableName | COUNTRY.DB |
| | Active | True |
| DataSource1 | DataSet | Table1 |

| Passo | Ação |
|-------|---|
| 4 | Utilizando a página Data Controls da Component Palette, adicione um componente DBGrid e DBNavigator ao seu form. |
| 5 | Utilize a tabela a seguir para definir as propriedades do DBNavigator e DBGrid para que utilizem o componente DataSource. |

| Nome do Componente | Propriedade | Valor |
|--------------------|-------------|-------------|
| DBGrid1 | DataSource | DataSource1 |
| DBNavigator | DataSource | DataSource1 |

| Passo | Ação |
|-------|---|
| 6 | Inicie o Fields Editor para o Componente Table1. |
| 7 | Adicione todos os campos da tabela ao dataset. |
| 8 | Defina um campo chamado PopArea e selecione FloaTField na list box Field type. |
| 9 | Certifique-se que o check box Calculated está selecionado, e clique em OK. |
| 10 | Saia do Fields Editor. |
| 11 | Utilize o Object Inspector para criar um event handler para o evento OnCalcFields do componente Table. Digite código no event handler para que fique similar ao seguinte: |

| Passo | Ação |
|-------|--|
| 12 | Utilize o Object Inspector para definir as seguintes propriedades do componente Field: |

| Nome do Componente | Propriedade | Valor |
|----------------------|---------------|-------|
| Table1PopArea | DisplayFormat | . # |

| Passo | Ação |
|-------|--|
| 13 | Execute e teste a aplicação |
| 14 | Quando tiver terminado o teste, grave e feche seu projeto. |

Utilizando os Componentes Database Lookup

Introdução

Um database lookup, ou tabela de busca é o processo de encontrar texto descritivo de um valor codificado. Uma situação comum onde tabelas de busca seriam utilizadas, seria quando você está editando o input de usuário e quiser exibir uma informação mais significativa, e não valores codificados.

O Delphi fornece dois componentes para buscar valores em uma tabela de banco de dados, ou editar o input contra um valor em uma tabela. Eles são os seguintes:

- Componente DBLookupList
- Componente DBLookupCombo

O termo lookup em cada nome de componente refere-se às tabelas de busca. O termo lookup tables refere-se às tabelas que contém informação descritiva sobre um valor codificado. Esta seção discute a utilização destes componentes. Eles se encontram na página Data Controls da Component Palette.

Componente DBLookupList

O componente DBLookupList é um componente ListBox data-aware projetado para buscar valores em uma tabela baseado no valor de uma segunda tabela. O DBLookupList contém um conjunto finito de dados, o usuário deve selecionar uma das opções da lista. Um DBLookupList permite exibir um conjunto de opções baseado no valor em outra tabela. O componente DBLookupList difere do componente DBListBox porque permite coordenar o valor selecionado do DBLookupList com a linha corrente de outra tabela do banco de dados.

Componentes DBLookupCombo

O componente DBLookupCombo é um componente ComboBox data-aware similar ao DBLookupList, exceto que um usuário pode selecionar um valor na lista ou digitar um novo valor. Um ComboBox de onde o DBLookupCombo é derivado combina as capacidades de um ListBox com as capacidades de um componente Edit.

Adicionando um Componente Database Lookup a um Form

Quando você adiciona um componente DBLookupList ou DBLookupCombo ao seu form, assumimos que:

- Você possui uma aplicação de banco de dados
- O form na aplicação possui pelo menos um DataSource e um componente derivado de TDataSet sendo utilizado para exibir informações do banco de dados

Adicionar um componente database lookup envolve o seguinte processo:

| Estágio | Processo |
|---------|--|
| 1 | Adicionar o componente database lookup e ligá-lo a um componente DataSource existente e propriedade DataField |
| 2 | Adicionar um novo componente Query ou Table (TDataSet) e DataSource, e utilizar este DataSource para buscar valores codificados no primeiro DataSource |

Passos para o Estágio 1

Execute os passos a seguir para adicionar um componente DBLookupList ou DBLookupCombo a um data source existente:

| Passo | Ação |
|-------|--|
| 1 | Adicione um componente database lookup em seu form. |
| 2 | Defina a propriedade DataSource a um componente DataSource que exista no form e que contenha o valor que você esteja procurando. |
| 3 | Defina a propriedade DataField ao campo que necessite de busca. |

Passos para o Estágio 2

Execute os passos a seguir para anexar o componente DBLookupList ou DBLookupCombo à tabela de busca:

| Passo | Ação |
|-------|--|
| 1 | Adicione um componente TDataSet utilizando um componente Table ou Query que corresponda à tabela de busca. |
| 2 | Defina a propriedade DatabaseName do novo componente TDataSet. |
| 3 | Execute um dos seguintes: <ul style="list-style-type: none"> • Defina a propriedade TableName do novo componente Table. • Entre com um query para a propriedade SQL do componente query. |
| 4 | Adicione um novo componente DataSource, e defina a propriedade DataSet ao novo componente TDataSet. |
| 5 | Defina a propriedade LookupSource de componente DBLookup ao novo componente DataSource. |
| 6 | Defina a propriedade LookupField ao valor chave da tabela de busca. |
| 7 | Defina a propriedade LookupDisplay ao campo que você queira exibir no componente DBLookupList. |

Tutorial: Criando uma Aplicação Utilizando um Componente Database Lookup

Introdução

A melhor maneira de se aprender a utilizar um componente DBLookup é através de um exemplo. Nesta seção, você utilizará um componente DBLookupList para exibir um nome de empresa baseado no campo CustID encontrado na tabela Orders.

Passos do Tutorial

O processo do tutorial envolve os seguintes estágios:

| Estágio | Processo |
|---------|--|
| 1 | Criar um form que exiba uma lista estática dos campos de uma tabela de banco de dados utilizando um componente Table, DataSource, DBGrid e DBNavigator |
| 2 | Adicionar e conectar um componente TDBLookupList à aplicação |
| 3 | Adicionar código para coordenar ações do componente TDBLookupList quando o sistema insere ou atualiza uma linha |

Passos para o Estágio 1

Execute os passos a seguir para criar um form que exiba campos selecionados da tabela Orders:

| Passo | Ação |
|-------|--|
| 1 | Abra um novo projeto e grave-o Quando solicitado, grave a unit como EX7CMAIN.PAS e o projeto como EXAMP7.DPR. |
| 2 | Utilizando a página Data Access da Component Palette, adicione um componente Table e |

Introdução ao Delphi

| | |
|---|--|
| | DataSource ao seu form. |
| 3 | Utilizando a página Data Controls, adicione os seguintes ao seu form: <ul style="list-style-type: none">• Um componente DBGrid• Um componente DBNavigator |
| 4 | Defina as seguintes propriedades para cada componente, como mostrado na tabela: |

| Nome do Componente | Propriedade | Valor |
|--------------------|---------------|-------------|
| Table1 | Database Name | DBEMOS |
| | TableName | ORDERS.DB |
| | Active | True |
| DataSource1 | DataSet | Table1 |
| | AutoEdit | False |
| DBGrid1 | DataSource | DataSource1 |
| DBNavigator | DataSource | DataSource1 |

| Passo | Ação |
|-------|---|
| 5 | Utilize o Fields Editor para adicionar os seguintes campos ao dataset Table1: <ul style="list-style-type: none">• OrderNo• CustNo• SaleDate• ItemsTotal• AmountPaid |
| 6 | Arranje os campos no Fields Editor para que OrderNo seja o primeiro campo e CustNo seja o segundo. |
| 7 | Compile e grave a aplicação. Execute e teste-a. |

Passos para o Estágio 2

Execute os passos a seguir para adicionar um componente DBLookupList à aplicação:

| Passo | Ação |
|-------|--|
| 1 | Utilizando a página Data Controls da Component Palette, adicione um componente DBLookupList ao seu form: |
| 2 | Utilize o Object Inspector para definir as seguintes propriedade do componente DBLookupList: |

| Nome do Componente | Propriedade | Valor |
|----------------------|-------------|-------------|
| DBLookupList1 | DataSource | DataSource1 |
| | DataField | CustNo |

| Passo | Ação |
|-------|--|
| 3 | Adicione um novo componente Table e DataSource ao seu form. Defina as propriedades para cada componente, como mostrado na tabela a seguir: |

| Nome do Componente | Propriedade | Valor |
|--------------------|--------------|-------------|
| Table2 | DatabaseName | DBDEMOS |
| | TableName | CUSTOMER.DB |
| | Active | True |
| DataSource2 | DataSet | Table2 |

| Passo | Ação |
|-------|---|
| 4 | Utilize o Fields Editor para adicionar os seguintes campos ao dataset Table2: <ul style="list-style-type: none"> • CustNo • Company |
| 5 | Conecte o componente DBLookupList ao segundo DataSource utilizando a tabela a seguir: |

| Nome do Componente | Propriedade | Valor |
|--------------------|---------------|-------------|
| DBLookupList1 | LookupSource | DataSource2 |
| | LookupField | CustNo |
| | LookupDisplay | Company |

| Passo | Ação |
|-------|--|
| 6 | Compile e grave a aplicação. Execute e teste-a. O componente DBLookupList destaca a empresa que corresponde ao valor de CustNo na linha do DBGrid. |

Passos para o Estágio 3

Este código de event handler permitirá definir o valor de CustNo durante o modo de edição ou inserção através de um duplo-clique no componente DBLookupList.

Execute os passos a seguir para adicionar um event handler para os eventos de DBLookupList, OnDbClick, ou OnClick:

| Passo | Ação |
|-------|---|
| 1 | Adicione as instruções a seguir no evento OnDbClick do componente DBLookupList: <i>if Table.State in [dsEdit, dsInsert] then</i> <i>Table1.CustNo.Value := Table2.CustNo.Value;</i> |
| 2 | Compile e grave sua aplicação. Execute e teste-a alternando a aplicação entre o modo de edição e inserção e utilizando o DBLookupList para definir o valor de CustNo na tabela Orders. |

Lab: Utilizando Componentes de Banco de Dados

Objetivos

- Este lab reforça a sua habilidade em:
- Adicionar um componente Table
- Adicionar um componente Database Grid
- Adicionar um componente Database Query
- Adicionar componentes Data Access
- Utilizar o Visual Query Builder
- Manipular e coordenar as ações dos componentes Data Access e Data Control

Cenário

Você construirá o início de um sistema de acompanhamento de vendas. Este sistema utilizará uma única grade de banco de dados para executar duas tarefas:

- Exibir informações de clientes e faturas
- Ligar clientes a faturas

No processo de construção desta aplicação, você utilizará e coordenará as ações dos componentes Data Access, Data Control e Standard.

Processo

Utilize o seguinte processo para aplicar o que você aprendeu:

| Estágio | Processo |
|---------|---|
| 1 | Abra um novo projeto e grave-o. Quando solicitado, grave a unit como LAB7MAIN.PAS e o projeto como LAB7.DPR. Altere o Caption do form para que exiba Aplicação de Acompanhamento de Vendas. |
| 2 | Adicione os seguintes componentes ao form: <ul style="list-style-type: none">• DataSource• Table• DBGrid• Button Defina as propriedades a seguir aos componentes, como mostrado na tabela: |

| Nome do Componente | Propriedade | Valor |
|--------------------|--------------|------------------------|
| Table1 | DatabaseName | DBDEMOS |
| | TableName | CUSTOMER.DB |
| | Active | False |
| DataSource1 | DataSet | Table1 |
| DBGrid1 | DataSource | DataSource1 |
| Button1 | Caption | Abrir Tabela &Customer |

| Estágio | Processo |
|---------|--|
| 3 | Adicione event handler OnClick para Button1 para que funcione como um botão de Liga?Desliga da Tabela Customer. Compile e grave sua aplicação. Execute e teste-a. |

```
Procedure TForm1.Button1Click(Sender: TObject);
begin
  if Table1.Active = True then
  begin
    Table1.Close;
    Button1.Caption := 'Abrir Tabela &Customer';
  end
  else
  begin
    Table1.Close;
    Button1.Caption := 'Fechar Tabela &Customer';
  end
end;
```

| Estágio | Processo |
|---------|---|
| 4 | Adicione os seguintes componentes ao form: <ul style="list-style-type: none">• Query• Data Source• Button Defina as seguintes propriedades aos componentes: |

| Nome do Componente | Propriedade | Valor |
|--------------------|---------------|--------------------|
| Query1 | Database Name | DBDEMOS |
| | SQL | select*from orders |
| | Active | False |
| DataSource2 | DataSet | Query1 |

| | | |
|----------------|---------|---------------------|
| Button2 | Caption | Abrir Tabela &Order |
|----------------|---------|---------------------|

| Estágio | Processo |
|----------|--|
| 5 | Adicione o event handler OnClick a seguir para Button2. Compile e grave sua aplicação. Execute e teste-a. Você consegue abrir a tabela Order? |

```

procedure TForm1.Button2Click (Sender : TObject);
begin
  if Query1.Active = True then
  begin
    Query1.Active := False;
    Button2.Caption := 'Abrir Tabela &Order';
  end
  else
  begin
    Query1.Active := True;
    Button2.Caption := 'Fechar Tabela &Order';
  end
end;
    
```

| Estágio | Processo |
|----------|---|
| 6 | Adicione dois componentes RadioButton ao form. Defina as seguintes propriedades aos componentes: |

| Nome do Componente | Propriedade | Valor |
|---------------------|-------------|--------------|
| RadioButton1 | Caption | Ver Clientes |
| | Checked | True |
| RadioButton2 | Caption | Ver Pedidos |

| Passo | Áção |
|----------|--|
| 7 | Adicione o event handler OnClick a seguir para RadioButton1 e RadioButton2 respectivamente: <i>DBGrid1.DataSource := DataSource1;</i> <i>DBGrid1.DataSource := DataSource2;</i> |
| 8 | Compile e teste sua aplicação como segue: <ul style="list-style-type: none"> • Dê um clique nos botões para abrir ambas as tabelas. • Utilize os botões de rádio para alternar entre os conjuntos de dados. • Tudo está funcionando? |
| 9 | Adicione os seguintes componentes complementares: <ul style="list-style-type: none"> • Group • Button • Dois componentes Edit • Dois componentes Label Defina as propriedades a seguir para cada um dos componentes e utilize o código a seguir para event handler OnClick de Button3: |

| Nome do Componente | Propriedade | Valor |
|--------------------|-------------|-------------------|
| GroupBox1 | Caption | (Vazio) |
| Button3 | Caption | &Definir Clientes |
| Edit1 | Text | (Vazio) |
| Edit2 | Text | (Vazio) |
| Label1 | Caption | Início: |
| Label2 | Caption | Final: |

Introdução ao Delphi

```

procedure TForm1.Button3Click(Sender : TObject);
begin
    Table1.SetRangeStart;
    Table1.Fields[0] .AsString := Edit1.Text;
    Table1.SetRangeEnd;
    Table1.Fields[0] .AsString := Edit2.Text;
    Table1.ApplyRange;
end;

```

| Estágio | Processo |
|-----------|--|
| 10 | <p>Compile e grave sua aplicação. Execute e teste-a, como segue:</p> <ul style="list-style-type: none"> • Dê um clique em Abrir Tabela Orders e depois em Ver Pedidos. • Digite valores nos campos Início e Final (por exemplo, 1005 e 1009), e dê um clique em Definir Clientes. • Qual o maior número de pedido exibido?----- • Qual o menor?----- |
| 11 | <p>Adicione os seguintes componentes complementares ao seu form:</p> <ul style="list-style-type: none"> • Query • DataSource • RadioButton <p>Defina as propriedades a seguir para cada componente, como mostrado na tabela.</p> |

| Nome do Componente | Propriedade | Valor |
|---------------------|--------------|-------------------------|
| Query2 | DatabaseName | DBDEMOS |
| DataSource3 | DataSet | Query2 |
| RadioButton3 | Caption | Ver Pedidos de Clientes |

| Estágio | Processo |
|-----------|--|
| 12 | <p>Execute as tarefas a seguir para dar à sua aplicação a habilidade de utilizar o componente DBGrid para exibir todos os seus pedidos agrupados por cliente.</p> <ul style="list-style-type: none"> • Inicialize o Query Builder utilizando o SpeedMenu do Componente Query2. Selecione DBDEMOS na lista. • Adicione as tabelas Customer e Orders. • Crie uma ligação do campo CustNo na tabela Customer com o campo CustNo na tabela Orders. • Adicione os seguintes campos ao resultado da query: Customer.CustNo Customer.Company Orders.OrderNo Orders.AmoundPaid |
| 13 | <p>Defina o critério de ordenação. Ordene o resultado da query na ordem ascendente por Customer Number. Grave seu trabalho e saia do Visual Query Builder.</p> |
| 14 | <p>Utilizando o Object Inspector, localize o componente Query2. Visualize a propriedade SQL e defina a propriedade Active para True.</p> |
| 15 | <p>Adicione o seguinte event handler OnClick para RadioButton3:</p> <pre> procedure TForm1.RadioButton3Click(Sender : TObject); begin DBGrid1.DataSource := DataSource3; end; </pre> |
| 16 | <p>Compile e grave sua aplicação. Execute e teste-a, como segue:</p> <ul style="list-style-type: none"> • Dê um clique em Ver Pedidos de Clientes. |

| | |
|--|--|
| | <ul style="list-style-type: none"> • Dê um clique em Ver Pedidos de Clientes. Sua query foi executada corretamente? |
|--|--|

Resumo do Capítulo

Pontos Chave

Após completar este capítulo, você aprendeu que:

- Para manipular e consultar bancos de dados no Delphi, você deve entender o conceito de um dataset. Um dataset no Delphi é o objeto tipo TDataSet e é uma classe abstrata. Os componentes Query, Table, e StoredProc são chamados de componentes TDataSet porque são derivados do objeto TDataSet.
- O componente DataSource atua como um intermediário entre o componente DataSet (TTable ou TQuery) e o componente Data Control.
- O componente Table é um componente DataSet que se comunica com uma tabela de banco de dados através do BDE.
- O Fields Editor permite criar uma lista de campos de um dataset.
- O componente DBGrid fornece uma maneira conveniente de exibir diversas linhas de dados de um componente Table ou Query. Sua aplicação pode utilizar o DBGrid para inserir, deletar, editar ou exibir dados de um banco de dados.
- O componente Query permite utilizar SQL, gerenciar a comunicação com o BDE, e serve como interface entre o BDE e os componentes DataSource em seus forms.
- O visual Query Builder é uma ferramenta visual para construir queries baseadas em SQL, e permite construir queries complexas com pouco ou nenhum conhecimento de SQL.
- O Fields Editor trabalha com os componentes Table e Query do TDataSet para definir campos calculados. O Fields Editor adiciona um novo objeto campo ao dataset.
- O Delphi fornece os componentes DBLookupList e DBLookupCombo para valores de busca em uma tabela.

Termos e Definições.

A tabela a seguir é uma referência rápida aos termos explicados neste capítulo.

| Termo | Definição |
|----------------------------|--|
| Campo calculado | Um campo que exibe valores gerados pelo programa |
| Classe abstrata | Uma classe de onde você pode derivar outras classes. Entretanto, você não pode criar uma variável desta classe. |
| Componente DataSet | Um componente Table, Query, ou StoredProc, que é derivado de um objeto the TDataSet |
| Database lookup | O processo de encontrar o texto descritivo para um valor codificado |
| Dataset | Um objeto no Delphi que consiste de uma série de registros, cada um contendo qualquer número de campos e um ponteiro para o registro atual |
| Query parametrizada | Uma query onde um ou mais valores na condição de seleção são desconhecidos até o momento da execução |
| Tabelas de busca | Tabelas que contém informação descritiva sobre um valor codificado |