

# Sistema Datacar - Parte III

## Trabalhando com Relações Mestre/Detalhe

por *Guinther de Bitencourt Pauli e Fernando Sarturi Prass*

**Utilize a tecnologia dbExpress para acessar tabelas com relacionamentos Mestre/Detalhe em banco de dados SQL.**

No primeiro artigo (edição de Junho) apresentamos o sistema Datacar, modelamos as tabelas, criamos o banco de dados, construímos o formulário principal e um formulário base de cadastro. Na segunda parte (edição de Julho), adicionamos funcionalidades ao formulário de cadastro, construímos o DataModule (com os componentes de acesso a dados dbExpress) e criamos os primeiros formulários de cadastros.

Os formulários criados, Marcas e Modelos, eram bastante simples. A maior complexidade estava no formulário de Modelos, onde havia uma procura para informar a marca do veículo. Neste terceiro artigo mostraremos técnicas para o desenvolvimento de formulários que possuem relacionamentos do tipo Mestre/Detalhe (ou *Master/Detail*) usando a linguagem SQL.

### Relacionamentos Mestre/Detalhe

Relacionamentos Mestre/Detalhe são relacionamentos do tipo **um para muitos**, ou seja, são relacionamentos onde **um registro** (chamado *mestre*) possui ligação a outros **n registros** de uma outra tabela (chamados de *detalhe*). Um exemplo disto é a relação existente entre as tabelas CLIENTES e VEICULO\_ATUAL, onde um cliente pode possuir diversos veículos, ou ainda a relação existente entre VEICULO\_ATUAL e ACESSORIOS\_VEICULO.

### Acessando a tabela de Clientes

A maneira de acessar corretamente um registro de uma tabela já foi largamente discutida no segundo artigo. Neste artigo descreveremos novamente passo a passo como é feito este acesso apenas para a tabela de clientes, para as demais tabelas mostraremos apenas o código da instrução SELECT, bastando ao leitor seguir os passos mostrados no artigo anterior (e aqui lembrados) para fazer o acesso.

Abra o DataModule (DM), acrescente um SQLDataSet (paleta dbExpress) e configure suas propriedades Name para dst\_clientes e SqlConnection para SQLConnect. Abra o editor da propriedade CommandText e digite a seguinte instrução SQL:

```
select CL.*, CD.NOME_CIDADE
from CLIENTES CL
inner join CIDADES CD on CD.COD_CIDADE = CL.COD_CIDADE
where COD_CLIENTE = :COD_CLIENTE
```

A instrução **CL.\*** informa ao servidor que queremos que ele nos traga todos os campos da tabela CLIENTES (representada por **CL**).

Abra a propriedade Params de dst\_clientes selecione o parâmetro no editor, configurando sua propriedade DataType para ftInteger. Adicione ao DM um componente DataSetProvider (paleta DataAccess) e configure sua propriedade Name para dsp\_clientes, e aponte sua propriedade DataSet para dst\_clientes.

Acrescente também um ClientDataSet (paleta DataAccess) no DM, definindo sua propriedade Name para cds\_clientes e ProviderName para dsp\_clientes. Dê um clique de direita sobre este componente e escolha a opção FetchParams. Dê um duplo clique sobre o mesmo e no editor de campos aperte CTRL+F, para que todos os campos sejam adicionados.

Configure a propriedade Required do campo COD\_CLIENTE como False (o valor deste campo é dado pelo sistema) e como True para campos NOME\_CLIENTE, ENDERECO e NOME\_CIDADE. Configure também a propriedade EditMask dos campos do tipo data. Coloque também uma máscara de entrada para o campo CEP (00.000-999;0;\_) e CPF (000.000.000-00;0;\_).

Para configurar as instruções de INSERT, UPDATE e DELETE, dê um duplo clique em dst\_clientes, e no editor de campos aperte CTRL+F. Selecione então todos os TFields, exceto COD\_CLIENTE, e configure a propriedade ProviderFlags.pfInWhere para False. Selecione apenas o campo NOME\_CIDADE e configure a

propriedade `ProviderFlags.pfInUpdate` para `False` (este campo não pertence à tabela `CLIENTES`, logo não deve estar na instrução de atualização). Selecione agora apenas `COD_CLIENTE` e configure sua propriedade `ProviderFlags.pfInKey` para `True`. Finalmente, configure a propriedade `UpdateMode` de `dsp_clientes` para `upWhereKeyOnly` e insira o comando abaixo no evento **BeforePost** de `cds_clientes` para atribuir a data padrão e gerar o valor da chave primária:

```
if cds_clientes.state=dsInsert then
  cds_clientes.DATA_CADASTRO.Value:=Date;
Incrementa('CLIENTES', cds_clientesCOD_CLIENTE);
```

Pronto, agora já é possível incluir, alterar e excluir um cliente, porém falta ainda a procura. Adicione um `SQLClientDataSet` (paleta `dbExpress`) ao DM e altere o seu nome para `cds_procura_cliente`. Configure a propriedade `DBConnection` para `SQLConnect` e em `CommandText` digite:

```
select COD_CLIENTE, NOME_CLIENTE, TELEFONE
from CLIENTES
where NOME_CLIENTE like :NOME_CLIENTE
order by NOME_CLIENTE
```

Na propriedade `Params` configure o parâmetro com o valor `ftString` para `DataType`. Dê um duplo clique no `cds_procura_cliente` e no editor de campos (`Field Editor`) adicione todos os `TFields` configurando suas propriedades `DisplayLabel` adequadamente.

## Formulário de Clientes

Abra o formulário `FrmFichaClientes`. Clique no menu `File | Use Unit` escolha `UDM`. No `DataSource` (`DtSrc`) aponte sua propriedade `DataSet` para `DM.cds_clientes`. Agora basta adicionar ao formulário `Labels` e `DBEdits` em número suficiente para exibir todos os campos da tabela `CLIENTES`. Codifique o botão de procura deste formulário da mesma maneira que foi feita nos demais cadastros.

Observe que colocamos um controle `RadioGroup` para entrada dos campos `Sexo` e `Estado Civil`. Na sua propriedade `Items` é colocado os valores por extenso e na propriedade `Values` os valores que irão para o banco de dados (neste caso uma única letra já que o campo no banco é um char de tamanho 1).

Configure a propriedade `ReadOnly` para `True` dos `DBEdits` de `DATA_CADASTRO` e também `COD_CIDADE`, para que o usuário não possa alterar estes valores diretamente. Faça isso em todos os controles que usam chaves-externas, para que o usuário não possa digitar um valor inconsistente para o campo, e sim buscar um nome através da procura.

Lembre-se também de colocar dois botões, um para fazer a procura por Cidade (veja o código do botão abaixo) e outro para carregar a foto do cliente usando um `OpenDialog` (da mesma maneira que foi feito no formulário de Modelos).

```
(*procura a cidade do cliente*)
FrmProcurar:=TFrmProcurar.create(self,dm.cds_procura_cidade);
try
  if FrmProcurar.ShowModal=mrOk then
    begin
      if DtSrc.State=dsbrowse then
        DM.cds_clientes.Edit;
        DM.cds_clientesCOD_CIDADE.AsInteger:=DM.cds_procura_cidadeCOD_CIDADE.AsInteger;
        DM.cds_clientesNOME_CIDADE.AsString:=DM.cds_procura_cidadeNOME_CIDADE.AsString;
      end;
    finally
      DM.cds_procura_cidade.close;
      FrmProcurar.Free;
    end;
```

A ficha de cadastro de clientes terá três paletas adicionais: `Veículos Atuais` para listar os veículos pertencentes ao cliente (tabela `VEICULO_ATUAL`), `Veículos Adquiridos` para listar os veículos que foram comprados pelo cliente na revenda (tabela `VEICULO_ADQUIRIDO`) e `Veículos Pretendidos` para listar os modelos de veículos que o cliente pretende comprar (tabela `VEICULO_PRETENDIDO`).

Para criar estas novas paletas, clique com botão direito do mouse sobre o `PageControl` e selecione a opção `New Page`, altere as propriedades `Name` do novo `TabSheet` para `tbsht_veiculo_atual` e `Caption` para “Veículos Atua”. Repita os passos para as outras duas paletas. Para prevenir que o usuário acesse as tabelas relacionadas sem antes escolher um cliente, escreva o seguinte no evento `OnChanging` do `PageControl`:

```
(*só muda se não DataSet não estiver vazio e nem estiver inserindo*)
AllowChange:=(not dm.cds_clientes.IsEmpty) and (not (dm.cds_clientes.state=dsinsert));
```

Seu formulário de Clientes deve estar como mostra a Figura 1:

Figura 1. Cadastrando um Cliente.

## Listando os Veículos do Cliente

Clique na paleta “Veículos Atuais” e adicione um DBGrid. Configure a propriedade Options.dgRowSelect para True e altere a aparência conforme as suas preferências. Acrescente também dois BitBtn, configure as propriedades Name e Caption do primeiro para btn\_incluir\_veiculo\_atual e “Incluir” respectivamente. No segundo botão altere as propriedades Name para btn\_ficha\_veiculo\_atual, Caption para “Ficha” e Enabled para False.

No DM acrescente um SQLClientDataSet (chame de cds\_lista\_veiculo\_atual) para trazer a lista de veículos que o cliente possui. Configure sua propriedade DBConnection e coloque o seguinte CommandText:

```
select M.NOME_MODELO, VA.COD_VEICULO_ATUAL, VA.ANO_FABRICACAO, VA.VALOR_PARA_VENDA, VA.VENDIDO
from VEICULO_ATUAL VA
inner join MODELOS M on M.COD_MODELO = VA.COD_MODELO
where VA.COD_CLIENTE = :COD_CLIENTE
```

Configure o parâmetro COD\_CLIENTE como ftInteger na propriedade Params. Adicione todos os campos no Field Editor, configurando a propriedade DisplayLabel dos campos. Selecione o campo VENDIDO e no evento onGetText digite o comando abaixo, para mostrar na grade “Sim” e “Não”, ao invés de 0 e 1.

```
if not Sender.IsNull then //se o campo nao estiver em branco
case Sender.AsInteger of
0: Text:='Não';
1: Text:='Sim';
end;
```

Volte ao formulário de clientes e adicione um DataSource, configurando seu Name para ds\_lista\_veiculo\_atual e DataSet para cds\_lista\_veiculo\_atual. Aponte o DBGrid para este DataSource. Dê um duplo clique no DBGrid e adicione os campos clicando no botão Add All Fields.

Para trazer os veículos pertencentes ao cliente, digite no evento onShow da tbsht\_veiculo\_atual:

```
DM.cds_lista_veiculo_atual.Close;
DM.cds_lista_veiculo_atual.Params[0].Value:=DM.cds_clientesCOD_CLIENTE.Value;
DM.cds_lista_veiculo_atual.Open;
//habilita o botão somente se o DataSet não estiver vazio
btn_ficha_veiculo_atual.Enabled:=not DM.cds_lista_veiculo_atual.IsEmpty;
```

No btn\_incluir\_veiculo\_atual devemos abrir o cds\_veiculo\_atual, colocá-lo em modo de inserção e mostrar o formulário para que seja possível fazer o cadastro. Para isto, no evento onClick do botão digite:

```
DM.cds_veiculo_atual.Open;
DM.cds_veiculo_atual.Append;
FrmFichaVeiculoAtual:=TFrmFichaVeiculoAtual.Create(self);
try
  FrmFichaVeiculoAtual.ShowModal;
finally
  FrmFichaVeiculoAtual.Free;
end;
tbsht_veiculo_atualShow(self); //atualiza o DBGrid
```

O código para o evento onClick do botão btn\_ficha\_veiculo\_atual é bastante semelhante ao anterior, a única diferença é que neste caso o DataSet deve trazer o registro correspondente a linha que estiver selecionada no DBGrid. Seu código é o seguinte:

```
DM.cds_veiculo_atual.Close;
DM.cds_veiculo_atual.Params[0].Value:=DM.cds_lista_veiculo_atualCOD_VEICULO_ATUAL.Value;
DM.cds_veiculo_atual.Open;
FrmFichaVeiculoAtual:=TFrmFichaVeiculoAtual.Create(self);
try
  FrmFichaVeiculoAtual.ShowModal;
finally
  FrmFichaVeiculoAtual.Free;
end;
tbsht_veiculo_atualShow(self); //atualiza o DBGrid
```

O DataSet que mostrado na grade deve ser fechado junto com o formulário, para isto acrescente a linha abaixo no evento onClose do formulário (não apague o comando **inherited**).

```
DM.cds_lista_veiculo_atual.Close;
```

## A tabela de Veículo Atual e seu Formulário

Para acessar a tabela VEICULO\_ATUAL proceda da mesma forma que para a tabela CLIENTES. Siga o mesmo padrão de nomenclatura para os componentes, ou seja, dst\_veiculo\_atual para o SQLDataSet, dsp\_veiculo\_atual para o DataSetProvider e cds\_veiculo\_atual para o ClientDataSet. Abaixo está a instrução SQL que traz os dados da tabela VEICULO\_ATUAL (propriedade CommandText do SQLDataSet):

```
select VA.*, M.NOME_MODELO
from VEICULO_ATUAL VA
inner join MODELOS M on M.COD_MODELO = VA.COD_MODELO
where COD_VEICULO_ATUAL = :COD_VEICULO_ATUAL
```

No evento onNewRecord do ClientDataSet (cds\_veiculo\_atual) digite:

```
//seta o proprietário do veiculo como sendo o cliente em memória
cds_veiculo_atualCOD_CLIENTE.Value:=cds_clientesCOD_CLIENTE.Value;
cds_veiculo_atualUNICO_DONO.Value:='0';
cds_veiculo_atualLVENDIDO.Value:='0';
cds_veiculo_atualDATA_CADASTRO.Value:=Date;
```

Quanto à ficha de cadastro do veículo, o leitor pode construí-la seguindo os conhecimentos já apresentados, pois ela não apresenta nenhuma dificuldade que já não tenha sido discutida. Tome apenas alguns cuidados, como o de colocar DBComboBox para escolher o Ano de Fabricação e do Modelo, e RadioGroup para escolher o Combustível. Configure a propriedade Visible do botão de procura como False,

já que a busca é feita na própria grade do formulário de clientes (você não pode excluir um controle que é herdado, apenas ocultá-lo). Observe o formulário final na Figura 2:

The screenshot shows a window titled "Veículo Atual" with a sub-header "GOL 1.6". It has two tabs: "Ficha" (selected) and "Acessórios". The "Ficha" tab contains the following fields:

- Código Veículo: 2
- Ano Fabricação: 2002
- Ano Modelo: 2002
- Nr Portas: 4
- Código Modelo: 3 (with a dropdown arrow) and "GOL 1.6" text
- Quilometragem: 10000
- Cor: Prata
- Placas: XXX-1234
- Data de Cadastro: 08/07/2002
- Combustível: Gasolina (selected)
- Único Dono:  (checked)
- Vendido:  (unchecked)
- OBS: Veículo em ótimo estado de conservação

At the bottom, there are buttons for "Incluir", "Gravar", "Excluir", and "Cancelar". The status bar shows "Alterando" and "Hint do Cadastro".

Figura 2. Lista de veículos do cliente (ao fundo) e a ficha de cadastro do mesmo.

## Adicionando Acessórios ao Veículo

Inclua um novo TabSheet no Form FrmFichaVeiculoAtual configurando as suas propriedades Name para TbSht\_acessorios e Caption para "Acessórios" (observe a Figura 2). Nesta nova paleta, inclua um DBGrid e altere as propriedades Name para DBGrd\_acessorios, Options.DgRowSelect e Options.dgMultiSelect para True (esta última permite a seleção de múltiplos registros).

Inclua um BitBtn e configure suas propriedades Name para btn\_incluir\_acessorio e Caption para "Incluir". Coloque outro BitBtn e configure Name como btn\_excluir\_acessorio e Caption como "Excluir".

No DM acrescente um SQLDataSet (chame de dst\_acessorios\_veic), um DataSetProvider (dsp\_acessorios\_veic) e um ClienteDataSet (cds\_acessorios\_veic) configurando-os da mesma forma como já foi feito para a tabela CLIENTES, de modo que seja possível incluir e excluir dados na tabela ACESSORIOS\_VEICULO. A instrução SQL a ser usada na propriedade CommandText do SQLDataSet é:

```
select AV.*, A.NOME_ACESSORIO
from ACESSORIOS_VEICULO AV
inner join ACESSORIOS A on AV.COD_ACESSORIO = A.COD_ACESSORIO
where COD_VEICULO_ATUAL = :COD_VEICULO_ATUAL
order by A.NOME_ACESSORIO
```

Note que a instrução apresentada acima mostra o nome de todos os acessórios que determinado veículo possui. Para fazer isto foi preciso fazer um **join** com a tabela ACESSORIOS, portanto tome o cuidado de configurar as propriedades ProvidesFlags.pfInWhere, InKey e pfIUpdate campo NOME\_ACESSORIO no SQLDataSet para False. No evento onShow da TbSht\_acessorios inclua o seguinte código:

```
DM.cds_acessorios_veic.Close;
DM.cds_acessorios_veic.Params[0].Value:=DM.cds_veiculo_atualCOD_VEICULO_ATUAL.Value;
DM.cds_acessorios_veic.Open;
btn_excluir_acessorio.Enabled:=not DM.cds_acessorios_veic.IsEmpty;
```

No evento OnClick do btn\_excluir\_acessorio digite o código abaixo:

```
var i : integer;
begin
  for i:=0 to DBGrd_acessorios.SelectedRows.Count-1 do
    begin
      DM.cds_acessorios_veic.GotoBookmark(Pointer(DBGrd_acessorios.SelectedRows[i]));
      DM.cds_acessorios_veic.Delete; //exclui o registro
    end;
  DM.cds_acessorios_veic.ApplyUpdates(0); //aplica a cache
  btn_excluir_acessorio.Enabled:=(not DM.cds_acessorios_veic.IsEmpty);
end;
```

A propriedade SelectedRows do DBGrid contém a lista de todos os registros que foram selecionados na grade. Uma vez que temos esta lista, basta usar o método GotoBookMark para localizar o referido registro.

Depois de localizado o registro é excluído, porém apenas da memória. Somente após excluir todos os registros selecionados é que a *cache* do DataSet é enviada ao banco, onde é feita a exclusão física do registro. Com isto o sistema fica mais rápido, pois só haverá uma comunicação com o banco de dados.

Já escrevemos o código para mostrar a lista dos acessórios do veículo e também o código que permite excluir um ou mais acessórios simultaneamente, porém falta ainda fazer a inclusão. Para isto precisamos antes construir um formulário que liste todos os acessórios cadastrados.

## Exibindo a lista de Acessórios

No menu selecione File | New | Form. Configure as propriedades Name para FrmListaAcessorios, BorderStyle para fbsSingle (não permiti redimensionar o tamanho do formulário), Position para poScreenCenter (mostra o formulário no centro da tela) e Caption para “Selecione Acessórios”. Acrescente um DBGrid e configure as propriedades Name para DBGrd\_lista\_acessorios, Options.DgRowSelect e Options.dgMultiSelect para True.

No DM adicione um SQLClientDataSet (chame de cds\_lista\_acessorios) com a seguinte instrução SQL:

```
select COD_ACESSORIO, NOME_ACESSORIO
from ACESSORIOS
where COD_ACESSORIO not in
  (select COD_ACESSORIO
   from ACESSORIOS_VEICULO
   where COD_VEICULO_ATUAL = :COD_VEICULO_ATUAL)
order by NOME_ACESSORIO
```

Esta instrução SQL é um pouco diferente das que estávamos utilizando até então, pois é composta por dois comandos de seleção. O primeiro é o comando que realmente seleciona os campos (COD\_ACESSORIO e NOME\_ACESSORIO) na tabela ACESSORIOS. O segundo estabelece a condição da consulta, que é listar todos os acessórios que ainda não estejam (devido à condição **not in**) incluídos para este veículo. Veja o resultado na Figura 3. Não esqueça de configurar o parâmetro na propriedade Params deste componente.

Volte a FrmListaAcessorios, clique no menu File | Use Unit e selecione UDM. Acrescente um DataSource com o nome de ds\_acessorios\_veiculo, aponte a propriedade DataSet para DM.cds\_lista\_acessorios e adicione o campo NOME\_ACESSORIO para que ele seja mostrado na grade. Configure então o DataSource da Grade para ds\_acessorios\_veiculo.

Adicione ao formulário dois BitBtn e configure a propriedade Kind do primeiro para bkOK e a do segundo para bkCancel. Grave a unit como UFormListaAcessorios.

Volte a FrmFichaVeiculoAtual e no evento onClick do btn\_incluir\_acessorio, digite o código:

```
var i : integer;
begin
  DM.cds_lista_acessorios.Params[0].Value:=DM.cds_veiculo_atualCOD_VEICULO_ATUAL.Value;
  DM.cds_lista_acessorios.Open; //lista os acessórios ainda nao selecionados
  FrmListaAcessorios:=TFrmListaAcessorios.Create(self);
  with FrmListaAcessorios do
    if ShowModal = mrOk then //se foi pressionado o botão "OK"
      begin
        //percorre todos os registros que foram selecionados na grade
        for i:=0 to DBGrd_lista_acessorios.SelectedRows.Count-1 do
          begin
            //localiza o registro no DataSet
            DM.cds_lista_acessorios.GotoBookmark(Pointer(DBGrd_lista_acessorios.SelectedRows[i]));
            DM.cds_acessorios_veic.Append; //inicia a inclusão
            DM.cds_acessorios_veicCOD_ACESSORIO.Value:=DM.cds_lista_acessoriosCOD_ACESSORIO.Value;
```

```

DM.cds_acessorios_veicCOD_VEICULO_ATUAL.Value:=DM.cds_veiculo_atualCOD_VEICULO_ATUAL.Value;
DM.cds_acessorios_veic.Post; //grava o registro
end;
DM.cds_acessorios_veic.ApplyUpdates(0); //aplica a cache
end;
FrmListaAcessorios.Free;
DM.cds_lista_acessorios.Close;
TbSht_acessoriosShow(self); //atualiza a lista de acessórios
end;

```

Por último, acrescente o comando abaixo no evento onClose do formulário:

```
DM.cds_acessorios_veic.Close;
```

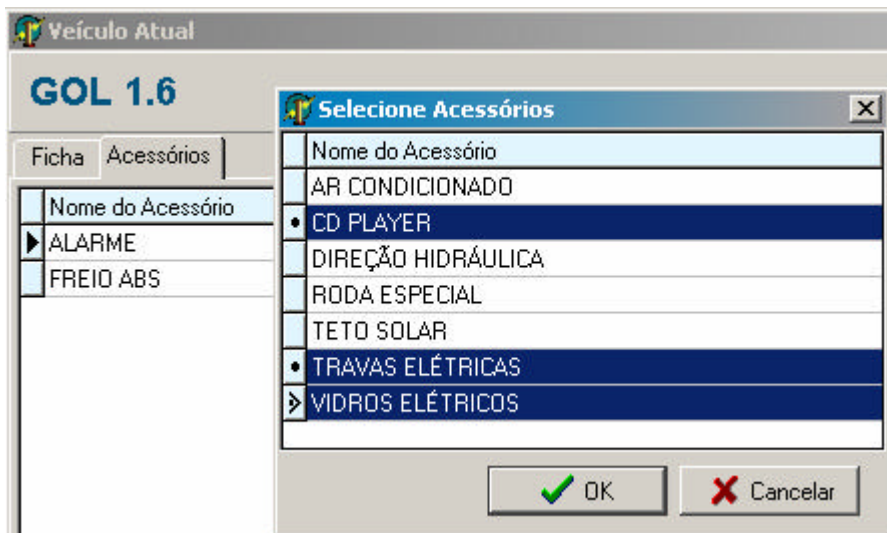


Figura 3. Ficha de cadastro de acessórios do veículo e a listagem de acessórios disponíveis

## Cadastro de Veículo Pretendido

O Cadastro de veículo pretendido é exatamente igual ao cadastro de veículo atual. Acreditamos que o leitor já possui conhecimentos suficientes para construir este formulário (observe-o na Figura 4). Para facilitar um pouco o trabalho, colocamos abaixo a instrução SQL que deve ser usada no componente SQLDataSet.

```

select VP.*, M.NOME_MODELO
from VEICULO_PRETENDIDO VP
inner join MODELOS M on M.COD_MODELO = VP.COD_MODELO
where COD_CLIENTE = :COD_CLIENTE

```

Não esqueça de configurar as ProviderFlags dos campos, tomando o devido cuidado com o campo chave primária e com o campo NOME\_MODELO que não pertence à tabela VEICULO\_PRETENDIDO.

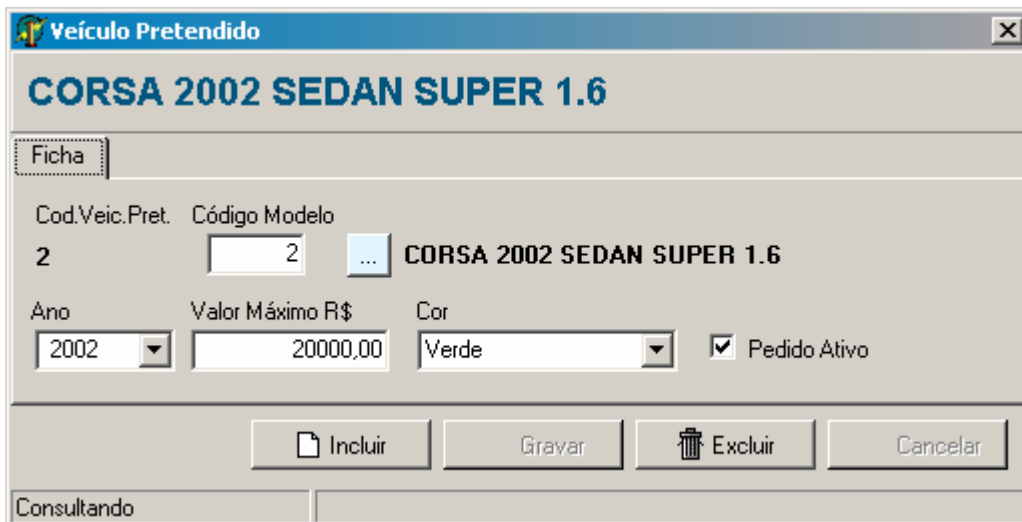


Figura 4. Cadastro de Veículo Pretendido.

Na Figura 5 você pode ter uma visão geral do DataModule até agora:

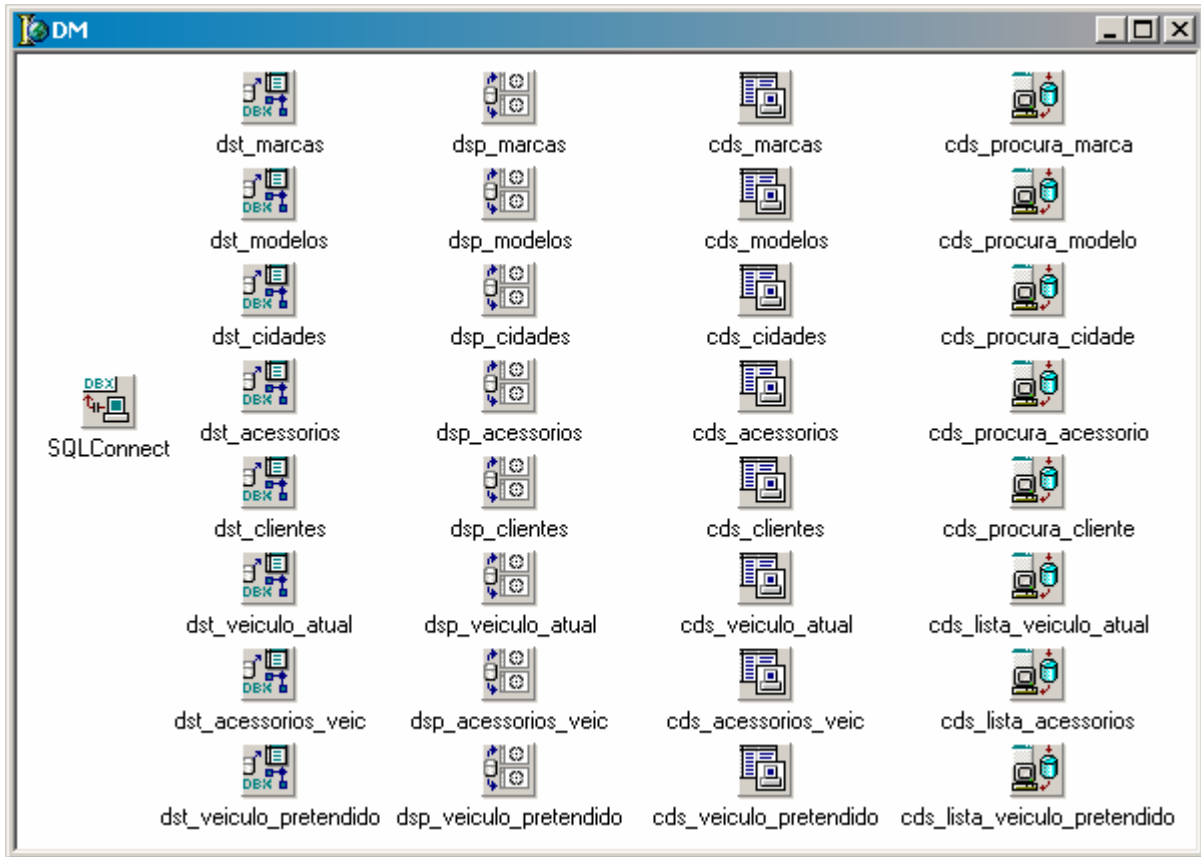


Figura 5. DataModule com um trio de componentes para cada tabela e um SQLClientDataSet auxiliar para consulta.

## Visualizando Mensagens de Erro do Servidor

O tratamento de erros do banco de dados será discutido no próximo artigo. No entanto, vale a pena utilizar a dica abaixo para saber se algo deu errado no momento da gravação dos dados que estão na *cache* do ClientDataSet. Se você gravar os dados e o servidor Interbase recusar por algum motivo, seja violação de chave, problemas de *updates* ou violação de *constraint* (valores inválidos na chave estrangeira), o ClientDataSet não exibirá nenhuma mensagem de erro, como se os dados tivessem sido gravados com sucesso. Fica difícil neste caso localizar a causa do problema. Para visualizar a mensagem de erro, escreva o seguinte código no evento OnReconcileError do ClientDataSet:

```
MessageDlg('Erro SQL, a mensagem nativa do banco é : '+E.Message,MtInformation,[mbok],0);
```

## Conclusão

Neste artigo demos continuidade aos cadastros iniciados no artigo anterior. Aprendemos a trabalhar com relacionamentos do tipo Mestre/Detalhe, cadastros parametrizados, comandos SQL com sub-consultas e também com a seleção de múltiplos registros em grades. No próximo artigo construiremos o cadastro de veículos adquiridos, o que conclui os módulos de cadastros do sistema. Veremos também como fazer validações de dados e tratamento de erros, e também como construir buscas personalizadas. Não deixe de fazer o download do código fonte do sistema, onde também incluímos algumas FAQs para soluções dos problemas comuns. Um grande abraço a todos!

**Fernando Sarturi Prass** é mestrando em Ciência da Computação na UFSC. Desenvolve sistemas para os bancos de dados InterBase/Firebird, DB2 e SQL Server. Pode ser contatado em [prass@inf.ufsc.br](mailto:prass@inf.ufsc.br) ou no site *Planet Delphi* ([www.fprass.hpg.com.br](http://www.fprass.hpg.com.br)).

**Guinther Pauli** é Bacharel em Sistemas de Informação (Unifra, Santa Maria – RS) e desenvolvedor Delphi/Kylix certificado *Master Delphi Programmer* pela Tekmetrics e *Delphi 5 Programmer* pela Brainbench. Trabalha com DataSnap, dbExpress e WebSnap em soluções distribuídas, cliente/servidor e web. Pode ser contatado em [guinther@unifra.br](mailto:guinther@unifra.br).