LINGUAGEM DE PROGRAMAÇÃO II

Criação de Aplicativos em Delphi com Banco de Dados Firebird

O objetivo do curso é desenvolver aplicações com Banco de Dados em Delphi. Iremos utilizar o Delphi7 como ambiente de programação, o FireBird como SGBD e como ferramentas de administração de Banco de Dados o IBOConsole.

Alguns Links Interessantes

http://www.comunidade-firebird.org

http://www.firebird.com.br/download.php

http://www.firebase.com.br

http://www.linhadecodigo.com.br

http://www.angelo.com.br

http://www.webmundi.net

http://www.activedelphi.com.br

http://www.delphi.nack.com.br

http://www.delphibr.com.br

http://www.edudelphipage.com.br

http://www.guiadodelphi.com.br

http://www.forumweb.com.br

http://superdownloads.ubbi.com.br

http://www.reddevil.eti.br

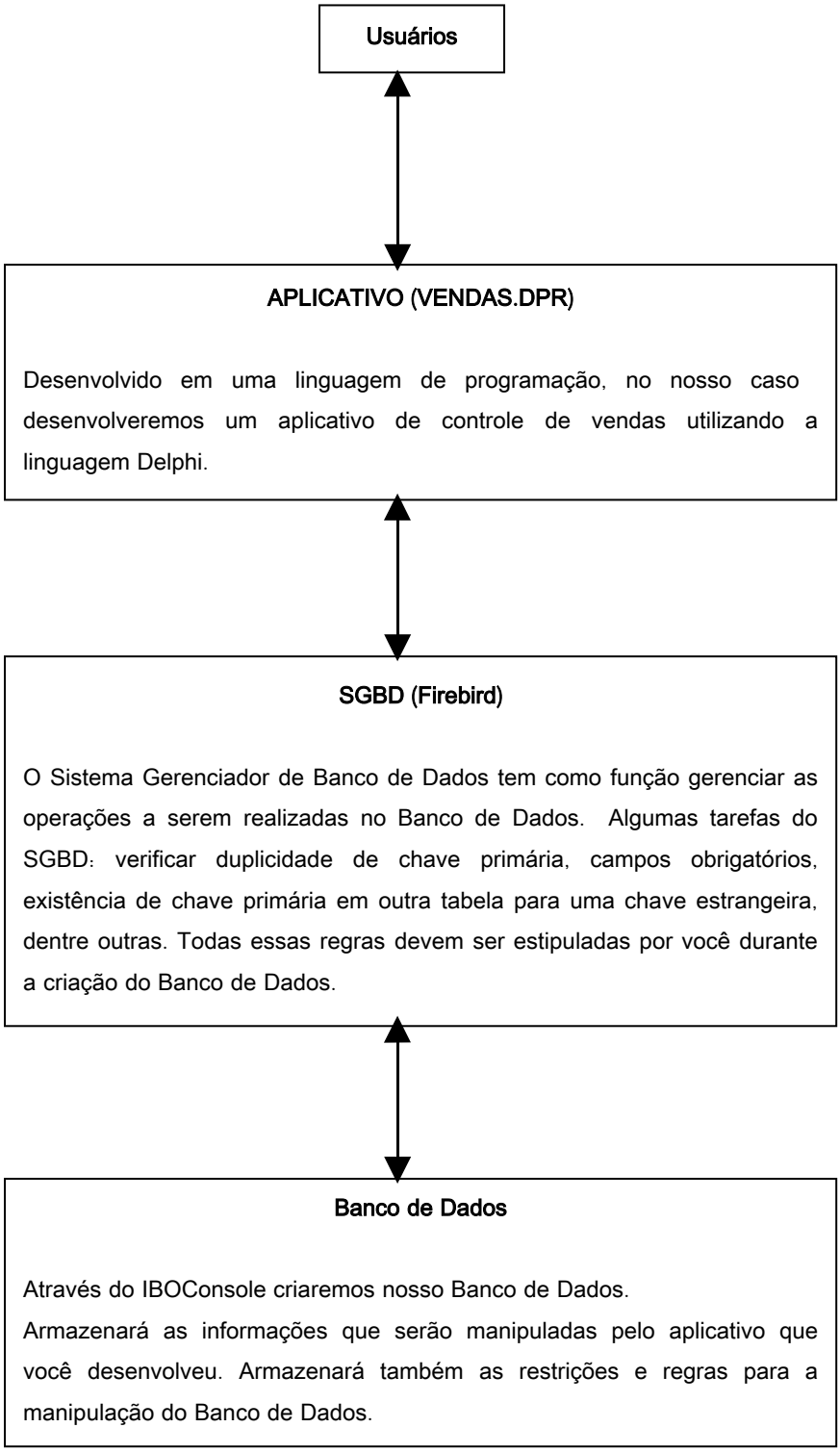
Firebird

A partir do ano de 2000, um grupo de desenvolvedores criou o projeto Firebird. Trata-se de um SGBD baseado na versão 6 do banco de dados relacional Interbase da Borland. Assim como aconteceu com o Linux, o projeto do FIREBIRD está aberto para todos que queiram contribuir para seu aprimoramento, "bugs" tendem a ser mais rapidamente resolvidos e um maior número de aprimoramento tende a ser incorporado em um menor espaço de tempo.

O FIREBIRD está disponível livremente em diversas plataformas: Windows, Linux, SOC UNIX (em teste), Solaris, HP-Ux, Mac-Os.

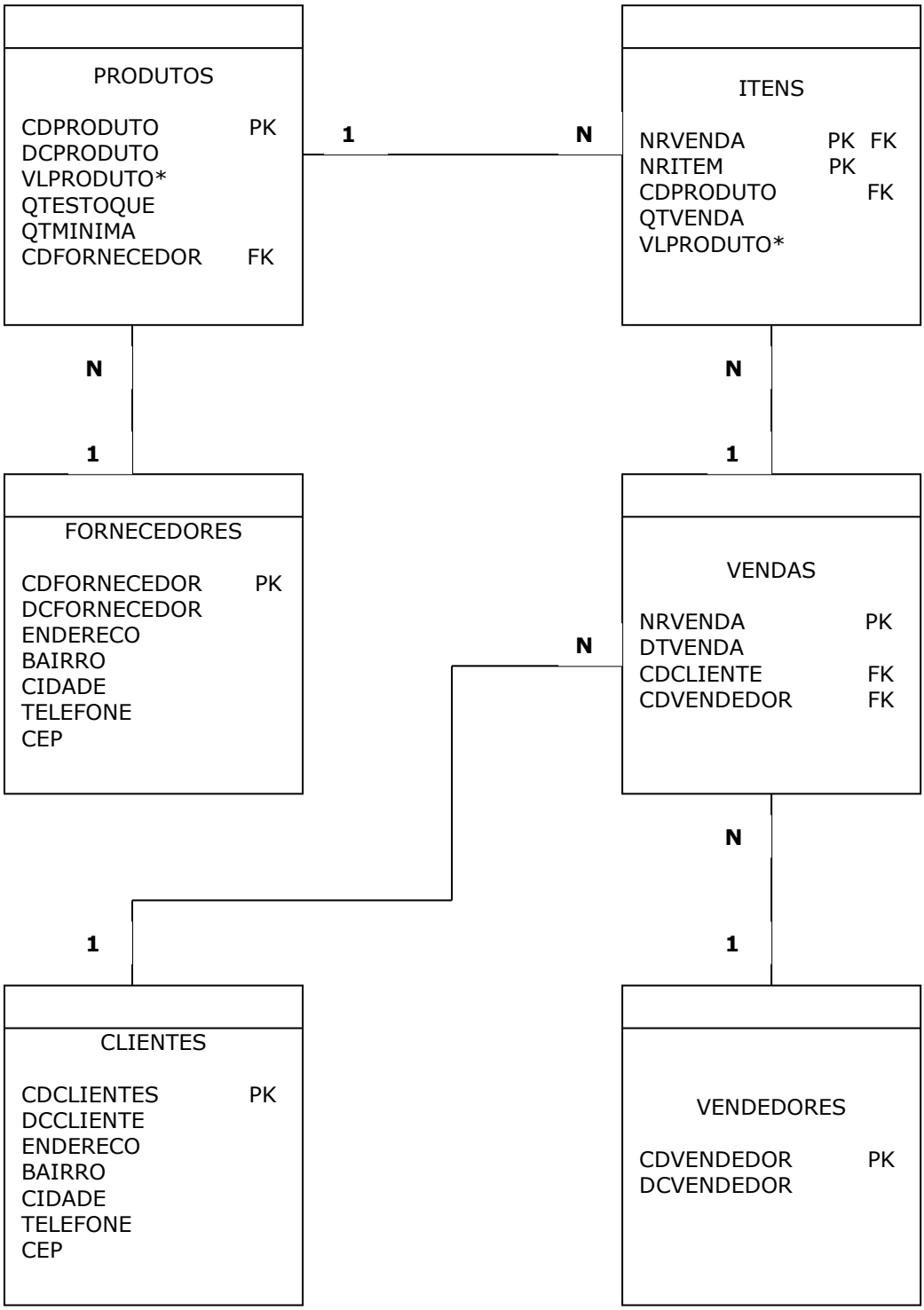
Nota: Em nosso curso iremos desenvolver um aplicativo completo para que o aluno possa relacionar os conceitos de banco de dados e Delphi com uma aplicação real. Trata-se um **SISTEMA DE CONTROLE DE VENDAS** que começaremos a desenvolver agora.

O esquema abaixo visa esclarecer dúvidas referentes ao papel do SGBD, do Banco de Dados, do Aplicativo, do Programador e dos Usuários:



Criando o Modelo de Entidade Relacionamento – MER

Nosso objetivo aqui é construir um modelo que represente as relações entre as entidades do nosso sistema. Obrigatoriamente esta deve ser a primeira etapa no desenvolvimento de um sistema de Banco de Dados.



PK = Primary Key = Chave Primária
FK = Foreign Key = Chave Estrangeira

*pode parecer redundância mas não é, uma vez que poderá haver alteração do valor do produto lançado no momento da venda em relação ao período atual.

Após a construção do MER podemos definir as tabelas:

Tabela: Fornecedores					
Campo	**Descrição**	**Tipo**	**Obs**	**PK**	**FK**
CDFORNECEDOR	Código do Fornecedor	Integer	Not null	X	
DCFORNECEDOR	Descrição do Fornecedor	Varchar(30)			
ENDERECO	Endereço do Fornecedor	Varchar(40)			
BAIRRO	Bairro	Varchar(20)			
CIDADE	Cidade	Varchar(20)			
TELEFONE	Telefone	Varchar(10)			
CEP	Cep	Varchar(8)			

Tabela: Clientes					
Campo	**Descrição**	**Tipo**	**Obs**	**PK**	**FK**
CDCLIENTE	Código do Cliente	Integer	Not null	X	
DCCLIENTE	Descrição do Cliente	Varchar(30)			
ENDERECO	Endereço do Cliente	Varchar(40)			
BAIRRO	Bairro	Varchar(20)			
CIDADE	Cidade	Varchar(20)			
TELEFONE	Telefone	Varchar(10)			
CEP	Cep	Varchar(8)			

Tabela: Vendedores					
Campo	**Descrição**	**Tipo**	**Obs**	**PK**	**FK**
CDVENDEDOR	Código do Vendedor	Integer	Not null	X	
DCVENDEDOR	Descrição do Vendedor	Varchar(30)			

Tabela: Produtos					
Campo	**Descrição**	**Tipo**	**Obs**	**PK**	**FK**
CDPRODUTO	Código do Produto	Integer	Not null	X	
DCPRODUTO	Descrição do Produto	Varchar(30)			
VLPRODUTO	Valor (preço) do Produto	Decimal(16,2)			
QTESTOQUE	Quantidade em estoque do produto	Decimal(16,2)			
QTMINIMA	Quantidade mínima desejável do produto em estoque	Decimal(16,2)			
CDFORNECEDOR	Código do Fornecedor	Integer			X

Tabela: Vendas					
Campo	**Descrição**	**Tipo**	**Obs**	**PK**	**FK**
NRVENDA	Número da venda	Integer	not null	X	
DTVENDA	Data da venda	Date	not null		
CDCLIENTE	Código do cliente	Integer	not null		X
CDVENDEDOR	Código do Vendedor	Integer	not null		X

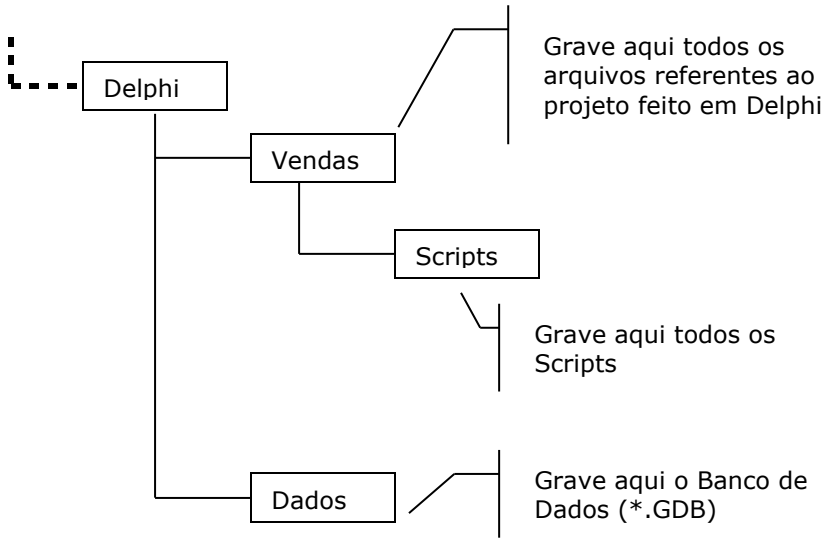
Tabela: Itens					
Campo	**Descrição**	**Tipo**	**Obs**	**PK**	**FK**
NRVENDA	Número da venda	Integer	Not null	X	X
NRITEM	Número do item da venda	Integer	Not null	X	
CDPRODUTO	Código do Produto	Integer			X
QTVENDA	Quantidade Vendida	Decimal(16,2)			
VLPRODUTO	Valor (preço) do Produto	Decimal(16,2)			

Com o MER aprovado pelo usuário podemos criar fisicamente o Banco de Dados. Utilizaremos o IBOConsole como ferramenta para criar e alterar nosso banco de dados. Através do IBOConsole criaremos as tabelas, determinaremos as chaves primárias e estrangeiras, os domínios dos campos e definiremos os relacionamentos entre as tabelas.

Antes de começar a parte prática alguns esclarecimentos:

- o Firebird já está instalado em seu computador e será automaticamente executado quando você ligar o computador, ficando residente na memória do computador.

- Crie a seguinte estrutura subordinada a pasta **ALUNOS** no HD do seu computador (local):

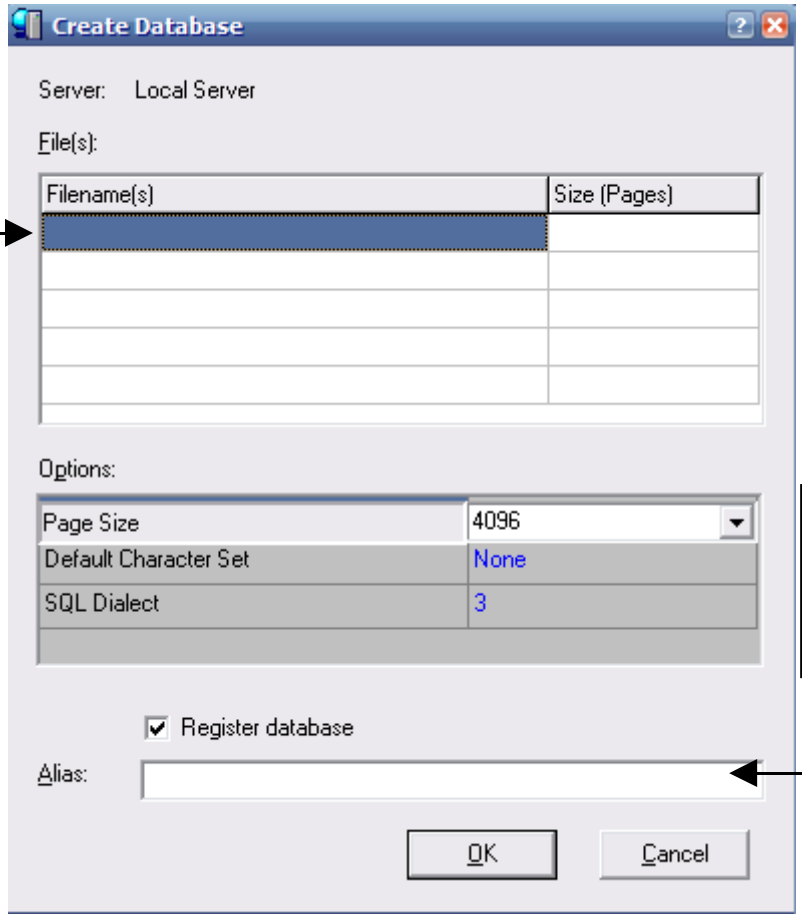


- Uma das formas de **avaliação** utilizada no curso será o desenvolvimento de algumas partes do nosso aplicativo, SISTEMA DE CONTROLE DE VENDAS, pelo **próprio aluno** em um prazo pré-estabelecido. Para evitarmos problemas futuros: **Não esqueça de fazer BACKUP. Reserve os 5 minutos finais de cada aula para realizá-lo.**

Ao executar o IBOConsole deve-se inicialmente dar um clique duplo em Local Server e informar o nome do usuário (SYSDBA) e a senha (masterkey) (repare nas letras maiúsculas e minúsculas).

Para criar um banco de dados selecione as opções *Database*|*Create Database.* Aparecerá a seguinte tela:

Digite o caminho completo e o nome do banco de dados. Por exemplo:
C:\....\DELPHI\DADOS\VENDAS.GDB.



Em Alias digite um apelido para o seu banco de dados. Por exemplo: **VENDAS**

Após a confirmação, o Firebird terá criado o arquivo com extensão GDB no HD.

Criação das Tabelas do Banco de Dados

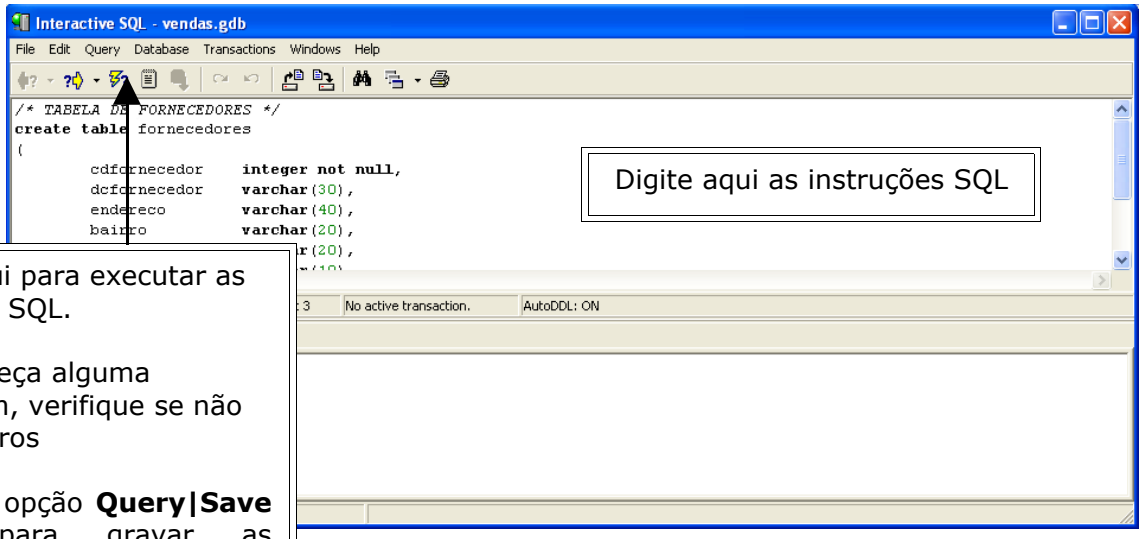
As tabelas devem ser criadas através de instruções SQL. Veja as instruções SQL para criar as tabelas do nosso banco dados:

```
/* TABELA DE FORNECEDORES */
create table fornecedores
   (cdfornecedor     integer not null,
   dcfornecedor      varchar(30),
   endereco          varchar(40),
   bairro            varchar(20),
   cidade            varchar(20),
   telefone          varchar(10),
   cep               varchar(8),
   constraint pkfornecedores primary key(cdfornecedor));



/* TABELA DE CLIENTES */
create table clientes
   (cdcliente  integer not null,
   dccliente   varchar(30),
   endereco    varchar(40),
   bairro      varchar(20),
   cidade      varchar(20),
   telefone    varchar(10),
   cep         varchar(8),
   constraint pkclientes primary key(cdcliente));



/* TABELA DE VENDEDORES */
create table vendedores
   (cdvendedor integer not null,
   dcvendedor  varchar(30),
   constraint pkvendedores primary key(cdvendedor));



/* TABELA DE PRODUTOS */
create table produtos
   (cdproduto    integer not null,
   dcproduto     varchar(30),
   vlproduto     decimal(16,2),
   qtestoque     decimal(16,2),
   qtminima      decimal(16,2),
   cdfornecedor  integer not null,
   constraint pkprodutos primary key(cdproduto),
   constraint fkprodutosfornecedores foreign key(cdfornecedor)
   references fornecedores(cdfornecedor));
```

Repare que relacionamos as tabelas PRODUTOS e FORNECEDORES através da chave estrangeira **fkprodutosfornecedores**, isto significa que na tabela PRODUTOS o banco de dados não irá deixar gravar um código de fornecedor que não esteja cadastrado na tabela FORNECEDORES.

```
/* TABELA DE VENDAS */
create table vendas
   (nrvenda    integer not null,
   dtvenda     date not null,
   cdcliente   integer not null,
   cdvendedor  integer not null,
   constraint pkvendas primary key(nrvenda),
   constraint fkvendasclientes foreign key(cdcliente)
   references clientes(cdcliente),
   constraint fkvendasvendedores foreign key(cdvendedor)
   references vendedores(cdvendedor));

/* TABELA DE ITENS */
create table itens
   (nrvenda    integer not null,
   nritem      integer not null,
   cdproduto   integer not null,
   qtvenda     decimal(16,2) not null,
   vlproduto   decimal(16,2) not null,
   constraint pkitens primary key(nrvenda,nritem),
   constraint fkitensvendas foreign key(nrvenda)
   references vendas(nrvenda),
   constraint fkitensprodutos foreign key(cdproduto)
   references produtos(cdproduto));
```

Para que as tabelas sejam efetivamente criadas, as instruções acima devem ser submetidas ao Firebird. Clique no botão SQL do IBOConsole que aparecerá a seguinte tela:



Clique aqui para executar as instruções SQL.

Caso apareça alguma mensagem, verifique se não contem erros

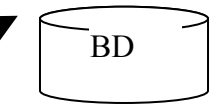
Escolha a opção **Query|Save Script** para gravar as instruções. Grave na Pasta Scripts. Dê nomes adequados, exemplo: fornecedores.sql

Escolha a opção **Load|Save Script** para Abrir instruções anteriormente gravadas.

esenvolver um programa capaz de manipular informações de Dados. Trata-se na verdade de um sistema que controlará as a empresa.

mais a esta parte por ser o nosso objeto de estudo o ***desenvolvimento de aplicativos para banco de dados***. Os conceitos relativos a Banco de Dados já foram vistos em módulos anteriores.

Antes de entrarmos no ambiente de programação do Delphi precisamos entender como funcionam os componentes para banco de dados do Delphi e como é feita a conexão destes componentes ao banco de dados.



SqlConnection

Propridedade	Explicação
Connected	Quando true faz a conexão com o banco de dados
DriverName	Nome do drive de banco de dados
ConnectionName	Nome da conexão. Se não existir, deve ser criada. Propriedades que devem ser p reenchidas quando se cria um conexão: **Database**: *.GDB **Username**: SYSDBA **Password**: masterkey **Obs**: Para se criar uma nova conexão basta dar dois cliques no objeto SqlConnection.
LoginPrompt	Coloque false para não solicitar senha toda vez que o BD for aberto.

SimpleDataSet

Propridedade	Explicação
Active	Quando true faz com que a instrução SQL seja executada
Connection	Escolha um objeto SQLConnection
DataSet.CommandTex t	Armazena a instrução Sql.

DataSource

Propridedade	Explicação
DataSet	Escolha um objeto SimpleDataSet

Objetos de Controle: DBEdit, DBText, DBImage, DBGrid, DBNavigator, ...

Propridedade	Explicação
DataSource	Escolha um objeto DataSource

Criaremos o formulário principal do nosso sistema e um módulo de dados que conterá todos os componentes de acesso a banco de dados.

1. Inicie o Delphi. Salve a unit como UFrmPrincipal e o projeto como **Vendas.dpr** na pasta ***C:\...\DELPHI\VENDAS***

2. Mude o nome do formulário para **FrmPrincipal**. O **Caption** para Sistema de Controle de Vendas.

3. Para que possamos trabalhar com as tabelas do FireBird dentro do Delphi, precisamos utilizar alguns componentes específicos para acesso ao banco. Iremos utilizar os componentes da paleta **dbExpress** e **DataAccess.** Estes componentes podem ser agrupados em DataModules, ou Módulos de Dados, de tal forma que facilite a organização dos componentes.

4. Para criar um Módulo de Dados, vá no menu **File | New | Others | Data Module.** Troque o nome do módulo de dados para **DM.** Salve a unit como **UDM.pas**

5. Insira no formulário **DM** um componente **SQLConnection** (paleta dbExpress), clique com o botão direito do mouse, escolha a opção "edit connection properties", clique em **+** para criar uma conexão ao Banco de Dados e configure-o assim:

   Connection name: DBVENDAS
   Driver Name: Interbase
   Database: C:\.....\DELPHI\DADOS\VENDAS.GDB
   UserName: SYSDBA
   Password: masterkey
   SQLDialect: 3

   Defina a propriedade **LoginPrompt** como **False** para que o sistema não fique perguntando o nome do usuário e a senha de acesso ao banco de dados quando alguma tabela do banco for aberta

   Faça um teste de conexão, dando um clique duplo na propriedade **Connected (não se esqueça que o Banco Firebird deve estar "no ar").** Se o path do banco de dados estiver correto e o nome do usuário e a senha também, a propriedade se alterará para **True,** caso contrário, o Delphi apresentará uma mensagem de erro, e você deve checar as propriedades acima**.**

6. Retorne o parâmetro Connected para False.

1. No Módulo de Dados (formulário DM) já inserimos o componente SqlConnection, agora insira os componentes abaixo e configure-os assim:

   SimpleDataSet (paleta dbExpress)

   Name: SimpleDataSetFornecedores
   Connection: SqlConnection1
   DataSet.CommandText: SELECT * FROM FORNECEDORES
   Active: true

   DataSource(paleta DataAcces)
   Name: DataSourceFornecedores
   DataSet: SimpleDataSetFornecedores

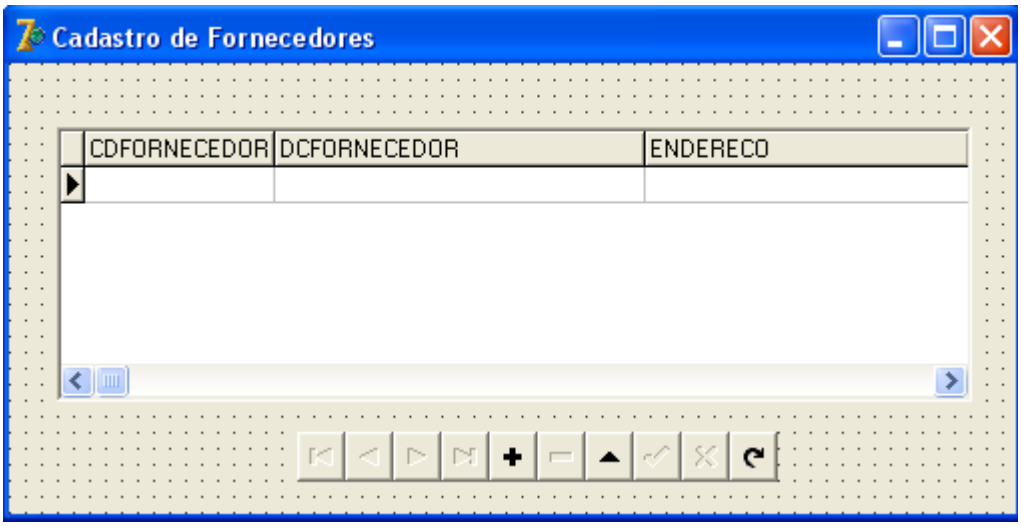
2. Criaremos agora o formulário para Cadastro de Fornecedores. Escolha o menu **File | New | Form**. Altere as seguintes propriedades para este formulário:

   Caption para Cadastro de Fornecedores
   Position para poScreenCenter
   Name para FrmFornecedores
   Salve a Unit como UFrmFornecedores

3. Insira no formulário **FrmFornecedores** um componente **DBGrid** (paleta DataControls). Este componente permite a visualização e edição de vários registros de uma tabela. Para que este DBGrid possa mostrar os registros da tabela Fornecedores, você deve configurar a propriedade **DataSource** com o nome do DataSource correspondente a Fornecedores (que é DataSourceFornecedores).

   - Como este Objeto (DatasourceFornecedores) não está neste formulário e sim em outro (DM), você precisará fazer o seguinte: selecione o formulário de fornecedores e escolha a opção File/Use Unit e escolha qual unit (UDM) está o objeto DataSource desejado.

4. Insira também no formulário **FrmFornecedores** um componente **DBNavigator.** Este componente é responsável pela navegação e edição de registros de uma tabela. Configura sua propriedade **DataSource** com o nome do DataSource correspondente da tabela Fornecedores (que é DataSourceFornecedores). O Formulário ficará com a seguinte aparência:



*Se o componente **SimpleDatasetFornecedores** estiver ativo (propriedade active = true), mesmo em tempo de projeto, os registros da tabela serão visualiados no DBGrid.*

5. Agora vamos criar um menu de opções (Objeto MainMenu) no formulário principal para que o usuário possa visualizar o formulário para cadastrar fornecedores. Crie uma única opção de nome **Cadastro** e com as seguintes sub-opções: Fornecedores, Clientes, Vendedores e Produtos.

6. Visualize o formulário **FrmPrincipal**. Incluir o código **no evento OnClick** da opção **Fornecedores** para que o form **FrmFornecedores** seja mostrado:

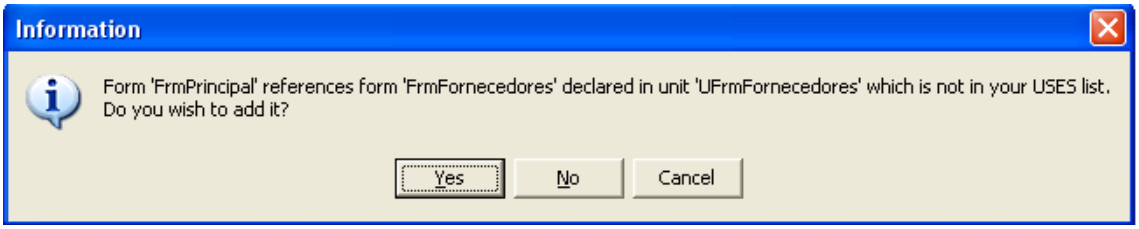
```
FrmFornecedores.Show;
```

7. Para que as inclusões, alterações e exclusões sejam gravados na tabela e não apenas no "buffer" de memória utilize no evento AfterPost e AfterDelete do componente **SimpleDasetFornecedores** a instrução:

```
Dm.SimpleDataSetFornecedores.ApplyUpdates(0);
```

O parâmetro do método ApplyUpdates indica o número máximo de erros que serão permitidos antes de parar a operação de atualização. Se desejar indicar que não há limite para número de erros, use o valor –1 como parâmetro do ApplyUpdates(-1).

8. Salve o Projeto e Execute-o. Ao executar o projeto a mensagem abaixo irá aparecer. Deve-se ao fato do formulário (FrmPrincipal) fazer referência a outro formulário (FrmFornecedores) devido a instrução **`FrmFornecedores.Show`** colocada no evento OnClick da opção Fornecedores do menu. Para que um formulário possa fazer referência a outro, na unit do 1º formulário deve ter uma claúsula Uses referente a unit do 2º formulário. Basta você clicar em **Yes** para que a instrução **Uses FrmFornecedores** seja inserida na Unit FrmPrincipal.



9. Salve o Projeto e tente executar novamente.

AVALIAÇÃO 1: Formulário para Cadastro de Clientes

Com base no cadastro de Fornecedores, faça este sozinho. Lembre-se de nomear os objetos seguindo a padronização utilizada até aqui:

Formulário: FrmClientes
Unit: UFrmClientes
SimpleDataSet: SimpleDataSetClientes
DataSource: DataSourceClientes

AVALIAÇÃO 2: Formulário para Cadastro de Vendedores

Faça este sozinho. Lembre-se de nomear os objetos seguindo a padronização utilizada até aqui:

Formulário: FrmVendedores
Unit: UFrmVendedores
SimpleDataSet: SimpleDataSetVendedores
DataSource: DataSourceVendedores

AVALIAÇÃO 3: Formulário para Cadastro de Produtos

Faça este sozinho. Lembre-se de nomear os objetos seguindo a padronização utilizada até aqui:

Formulário: FrmProdutos
Unit: UFrmProdutos
SimpleDataSet: SimpleDataSetProdutos
DataSource: DataSourceProdutos

Até agora o sistema está funcionando perfeitamente (ou pelo menos deveria), mas com alguns inconvenientes. Iremos no decorrer do desenvolvimento deste sistema eliminar estes inconvenientes implementando algumas facilidades.

A primeira facilidade que iremos implementar será no formulário de cadastro de produtos. Quando o usuário cadastra um produto novo ele precisa informar o respectivo fornecedor através do código do fornecedor (campo CDFORNECEDOR). O problema é que o usuário identificará o fornecedor de um determinado produto mais pelo nome do que pelo código.

O ideal seria o usuário poder **escolher** o fornecedor de um determinado produto pelo nome e não mais pelo código. O problema é que o campo que armazena o nome do fornecedor (DCFORNECEDOR) está na tabela FORNECEDORES e não na tabela PRODUTOS. Para resolver este problema criaremos na tabela PRODUTOS um campo de **Lookup**. Este campo pertencerá a tabela PRODUTOS, mas o seu conteúdo virá da tabela FORNECEDORES. Vamos agora criar um campo de **Lookup** para a tabela PRODUTOS:

1. No módulo de dados dê um duplo clique SimpleDataset correspondente a tabela PRODUTOS (que é SimpleDataSetProdutos). Clique com o botão direito no **Field Editor** e escolha Add All Fields (Adiciona todos os campos). Clique novamente com o botão direito no **Field Editor** e escolha **New Field** (novo campo). Na caixa Field Properties digite **descricaofornecedor** para **Name**, Type escolha string, size digite 30, escolha a opção **Lookup** para **Field Type** e para **Lookup definition** escolha:

   Dataset: SimpleDatasetFornecedores (*tabela de onde vem a descrição do fornecedor)*
   KeyFields: CDFORNECEDOR (*campo de ligação)*
   Lookup Keys: CDFORNECEDOR (*campo de ligação)*
   Result Field: DCFORNECEDOR *(resultado)*

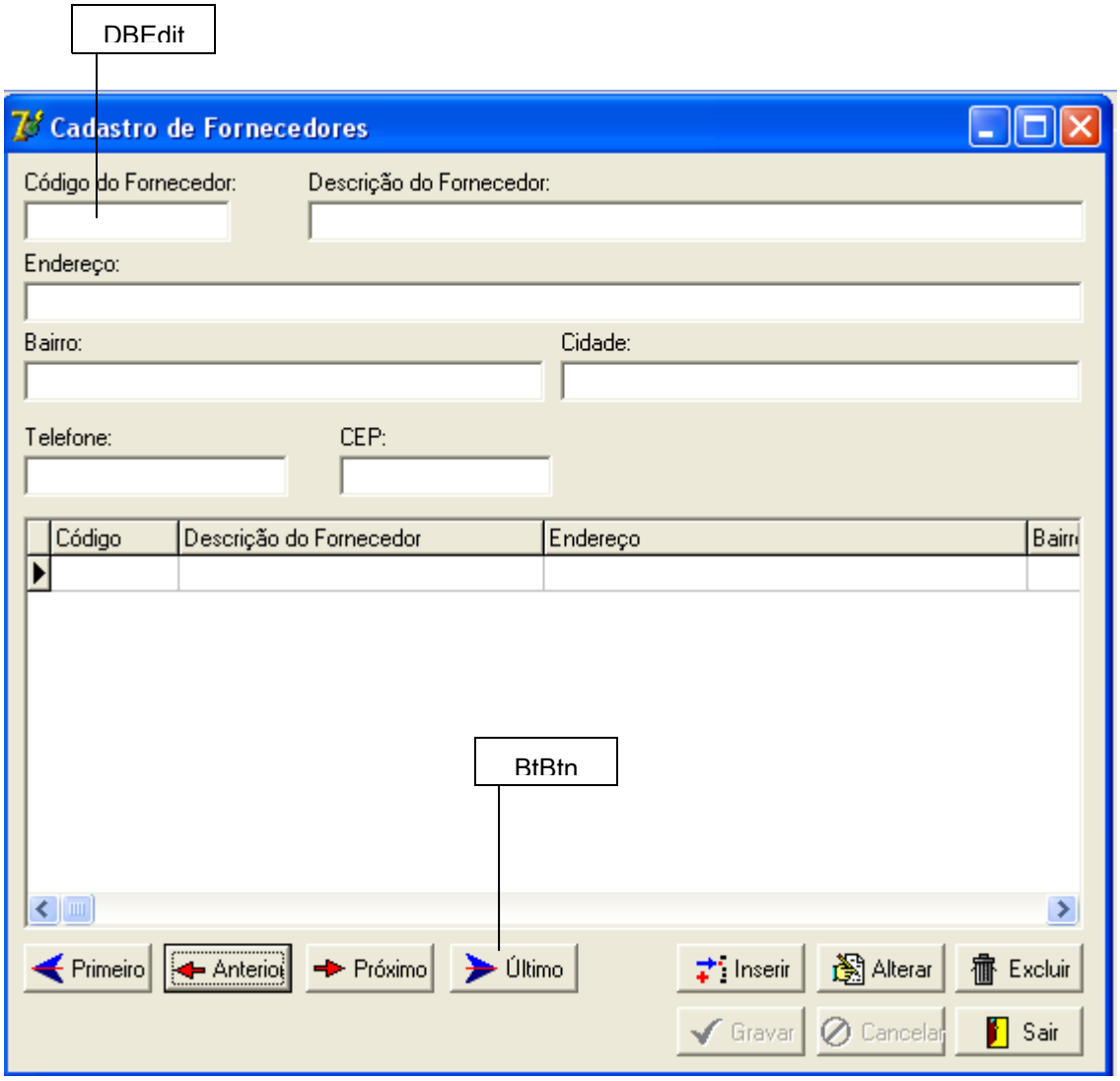
   Clique em OK.

2. Visualize o formulário FrmProdutos. Caso o novo campo não aparece no DBGrid faça o seguinte: dê um clique com o botão direito e escolha ADD. Uma coluna será criada. Escreva então **descricaofornecedor** em sua propriedade FieldName.

3. Salve o Projeto e Execute-o. Agora você não precisa mais digitar o código do Fornecedor, basta ir na nova coluna (descricaofornecedor) clicando na seta para baixo que aparece e escolher a descrição do fornecedor que desejar. Lembre-se que esta descrição do fornecedor que aparece pertence a tabela FORNECEDORES e não a tabela PRODUTOS. Na tabela PRODUTOS ficará gravado somente o campo código do fornecedor.

4. A coluna do DBGrid referente ao código do fornecedor (CDFORNECEDOR) não será mais útil, portanto você pode elimina-la (vá na propriedade Columns do DBGrid). Você não está eliminando o campo CDFORNECEDOR da tabela PRODUTOS e sim sua respectiva coluna destinada a simples visualização/edição no DBGrid.

O objetivo agora é melhorar a aparência e principalmente a funcionalidade do formulário de cadastro de fornecedores. Iremos utilizar novos componentes da paleta Data Controls, botões individuais para as operações de edição e navegação pelos registros da tabela, métodos e propriedades do componente SimpleDataset.



(Nova aparência do formulário de cadastro de fornecedores)

1. Acrescentar os objetos a medida que for sendo solicitado pelas instruções abaixo.

2. O objeto DBEdit (paleta Data Controls) permite editar/visualizar o conteúdo de um campo específico de uma tabela. Utilize as seguintes propriedades:

   Datasource: Indicar o datasource correspondente. Neste caso como estamos tratando de fornecedores o DataSource será o DataSourceFornecedores.
   Datafield: Indicar o nome do campo que se deseja vincular ao DBEdit. Configure cada DBEdit com seu respectivo campo da tabela FORNECEDORES.

3. Mude o título, font, cor, etc das colunas do DBGrid utilize a propriedade Columns.

4. Salve o Projeto e execute-o.

5. Exclua o DbNavigator, pois iremos substitui-lo por botões (BitBtn). Para os botões de edição mude seus nomes (propriedade name) para BtInserir, BtAlterar, BtExcluir, BtGravar e BtCancelar. ***A mudança de nome não é obrigatória mas será de grande ajuda mais tarde.***

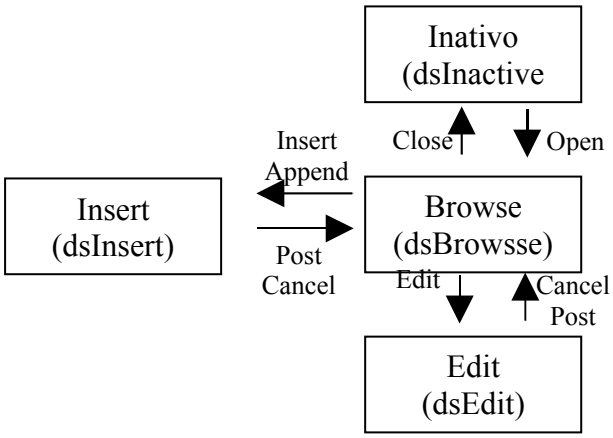
   Antes de continuar veremos alguns métodos e propriedades para objetos dataset (no nosso caso esses métodos devem ser acionados para um dataset do tipo SimpleDataSet).

Métodos:

Edit	Permite editar o data set para alteração dos valores atuais para novos valores.
Append	Cria um novo registro em branco após o último registro do dataset.
Insert	Cria um novo registro em branco após o registro corrente.
Delete	Remove o registro corrente.
Post	Confirma as alterações em um dataset.
Cancel	Cancela as alterações em um dataset.
First	Move o cursor para o primeiro registro do dataset.
Prior	Move o cursor para o registro anterior (se houver).
Next	Move o cursor para o próximo registro (se houver).
Last	Move o cursor para o último registro do dataset.
MoveBy(N)	Move o cursor para frente ou para trás, conforme o valor do parâmetro N (número de registros), que é um valor inteiro, positivo ou negativo.

Propriedades:

BOF : Boolean	Begin Of File. Indica True se o cursor estiver no primeiro registro da tabela.
EOF : Boolean	End Of File. Indica True se o cursor estiver no último registro da tabela.
State	Armazena o "Estado" do dataset. (dsInactive, dsBrowse, dsInsert, dsEdit). O importante é saber que determinadas operações dependem do estado do dataset. Exemplo 1: se o dataset está aberto (State=dsBrowse) e se você acionar o método Post causaria um erro pois o método Post só poder acionado se o dataset estiver em modo de edição (State = dsEdit) ou em modo de inserção (State = dsInsert). Por isso ao gravar um registro (Post), você deve verificar se realmente o registro está no modo de edição ou inserção, para não causar erro em seu programa. Veja a seguir:



(Estados (State) de uma tabela)

6. Para o evento OnClick do BtInserir digite:

```
begin
      Dm.SimpleDataSetFornecedores.append;
      DBEdit1.SetFocus; // Coloca o foco no DBEdit1
end;
```

7. Para o evento OnCLick do BtAlterar digite:

```
begin
     Dm.SimpleDataSetFornecedores.edit;
end;
```

8. Para o evento OnCLick do BtExcluir digite:

```
begin
     if MessageBox(0,'Deseja realmente excluir?','Excluindo …',mb_YesNo)=idYes
then
        Dm.SimpleDataSetFornecedores.delete;
end;
```

9. Para o evento OnCLick do BtGravar digite:

```
begin
     Dm.SimpleDataSetFornecedores.post;
end;
```

10. Para o evento OnCLick do BtCancelar digite:

```
begin
     Dm.SimpleDataSetFornecedores.cancel;
end;
```

11. Salve e execute o projeto. E faça alguns testes: Tente gravar (acionando o botão de gravar) sem ter antes acionado o botão de incluir ou de alterar. Irá ocorrer um erro, pois o método **post** só pode ser executado se a tabela estiver em modo de edição (dsEdit) ou inserção (dsInsert). Devemos então acrescentar no evento OnClick do botão gravar a parte em destaque abaixo. Se não for reconhecida a palavra dsEdit ou dsInsert, acrescente a biblioteca "DB".

```
begin
     if Dm.SimpleDataSetFornecedores.state in [dsedit,dsinsert] then
        Dm.SimpleDataSetFornecedores.post;
end;
```

12. Vamos agora implementar a numeração automática para o código do fornecedor.

    Mas Antes veremos como fazer referência ao conteúdo de um campo de uma tabela. O método FieldByName serve para referenciarmos o conteúdo de um determinado campo de uma tabela. Vejamos abaixo um trecho de programa que exemplifica a utilização do método FieldByname:

```
total:=0;
n:=0;
Dm.SimpleDataSet.first;
while not Dm.SimpleDataSet.eof do
begin
   total:=total + (Dm.SimpleDataSet.fieldbyname('NOTA').asFloat;
   n := n + 1;
   Dm.SimpleDataSet.next;
end;
showmessage('Média: '+formatfloat('0.00',total/n));
```

    Acrescente as instruções em destaque abaixo para o evento OnClick do botão inserir:

```
var
  prox:integer;
begin
  Dm.SimpleDataSetFornecedores.last;
  prox:=Dm.SimpleDataSetFornecedores.fieldbyname('cdfornecedor').
        asInteger +1;
  Dm.SimpleDataSetFornecedores.Append;
  Dm.SimpleDataSetFornecedores.fieldbyname('cdfornecedor').asinteger:=prox;
  Dbedit2.SetFocus;
end;
```

13. Agora vamos sofisticar um pouco mais nosso programa. Vamos fazer com que os botões de edição funcionem em sincronia. Exemplo: se você estiver inserindo ou alterando os botões BtInserir, BtAlterar e BtExcluir devem estar desabilitados (enabled = false) e os botões BtGravar, BtCancelar devem estar habilitados (enabled = true). Entretanto, quando você gravar ou cancelar uma operação de inserção ou alteração os botões BtInserir, BtAlterar e BtExcluir devem ser habilitados novamente. Para tanto crie uma procedure de nome **tratabotoes** conforme abaixo:

```
procedure tratabotoes;
begin
   BtInserir.enabled:=not BtInserir.enabled;
   BtAlterar.enabled:=not BtAlterar.enabled;
   BtExcluir.enabled:=not BtExcluir.enabled;
   BtGravar.enabled:=not BtGravar.enabled;
   BtCancelar.enabled:=not BtCancelar.enabled;
end;
```

Esta procedure deverá ser acionada do evento OnClick do botão BtInserir, BtAlterar, BtGravar e BtCancelar.

Lembre-se que os botões BtInserir, BtAlterar, BtExcluir devem iniciar habilitados (Enabled=true) e os botões BtGravar e BtCancelar devem iniciar desabilitados (Enabled=false).

14. Salve e execute o projeto para testar.

15. Vamos agora inserir 4 botões (BitBtn) para permitir a navegação pelos registros da tabela. Mude seus nomes (propriedade name) para BtPrimeiro, BtAnterior, BtProximo, BtUltimo.

16. Para o evento OnClick do BtPrimeiro digite:

```
begin
     Dm.SimpleDataSetFornecedores.first;
end;
```

17. Para o evento OnClick do BtAnterior digite:

```
begin
     Dm.SimpleDataSetFornecedores.prior;
end;
```

18. Para o evento OnClick do BtProximo digite:

```
begin
     Dm.SimpleDataSetFornecedores.next;
end;
```

19. Para o evento OnClick do BtUltimo digite:

```
begin
     Dm.SimpleDataSetFornecedores.last;
end;
```

20. Salve e execute o projeto para testar.

21. Melhore a aparência do DBGrid. Clique com o botão direito e escolha Columns Editor e clique novamente com o botão direito e escolha Add All Fields. Selecione cada Coluna e altere a Propriedade Title.

AVALIAÇÃO 4: Melhorando o Cadastro de Clientes

Agora você vai aplicar os conhecimentos adquiridos no cadastro de Fornecedores para melhorar o cadastro de Clientes.

AVALIAÇÃO 5: Melhorando o Cadastro de Vendedores

Agora você vai aplicar os conhecimentos adquiridos no cadastro de Fornecedores para melhorar o cadastro de Vendedores.

AVALIAÇÃO 6: Melhorando o Cadastro de Produtos

Agora você vai aplicar os conhecimentos adquiridos no cadastro de Fornecedores para melhorar o cadastro de Produtos.

1. Agora vamos implementar a consulta através de instruções SQL.

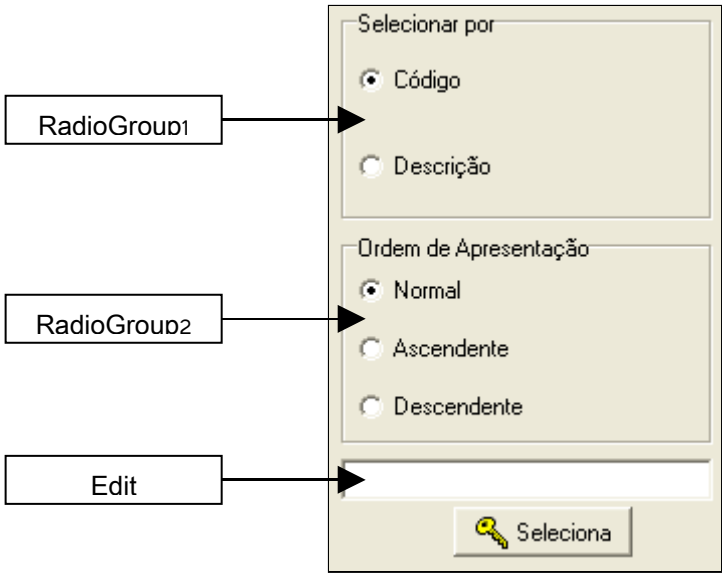
2. Inclua um componente BitBtn (mude o Caption para Seleciona) e inclua para seu evento OnClick as seguintes instruções:

```
Dm.SimpleDataSetFornecedores.close;
Dm.SimpleDatasetFornecedores.DataSet.CommandText:=
                'SELECT * FROM FORNECEDORES WHERE CDFORNECEDOR = 2';
Dm.SimpleDataSetFornecedores.open;
```

3. Salve e execute o programa. Acione o botão Seleciona e verifique se foi selecionado somente o fornecedor com código igual a 2. Se não selecionou nenhum registro provavelmente não existe o código 2.

4. Apesar da instrução SQL funcionar, ela possui uma grande limitação: seleciona somente o fornecedor de código igual a 2. O ideal é permitir ao usuário escolher os critérios e opções desta seleção. A solução é passar as opções de seleção escolhidas pelo usuário como parâmetros para a instrução SQL.

5. Inclua os demais componentes no formulário e configure-os conforme a aparência abaixo. Inclua os comandos abaixo no evento OnClick do botão Seleciona:



```
begin
    Dm.SimpleDataSetFornecedores.close;
    DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT := 'SELECT * FROM FORNECEDORES';

    case radiogroup1.ItemIndex of
        0:  begin
                if edit1.Text <> '' then
                begin
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT :=
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT +
                      ' WHERE CDFORNECEDOR = :Pcdfornecedor';

       Dm.SimpleDataSetFornecedores.DataSet.Params.ParamByName('Pcdfornecedor').Value:=
                      StrToInt(Edit1.Text);
                end;

                if radiogroup2.ItemIndex = 1 then
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT:=
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT +
                      ' ORDER BY CDFORNECEDOR';

                if radiogroup2.ItemIndex = 2 then
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT:=
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT +
                      ' ORDER BY CDFORNECEDOR DESC';
            end;

        1:  begin
                if edit1.Text <> '' then
                begin
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT:=
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT +
                      ' WHERE DCFORNECEDOR CONTAINING :Pdcfornecedor';


       Dm.SimpleDataSetFornecedores.DataSet.Params.ParamByName('Pdcfornecedor').Value:=
                      Edit1.Text;
                end;

                if radiogroup2.ItemIndex = 1 then
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT:=
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT +
                      ' ORDER BY DCFORNECEDOR';

                if radiogroup2.ItemIndex = 2 then
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT:=
                      DM.SIMPLEDATASETFORNECEDORES.DATASET.COMMANDTEXT +
                      ' ORDER BY DCFORNECEDOR DESC';
            end;
    end;

    Dm.SimpleDataSetFornecedores.open;
end;
```

6. Salve e execute o projeto para testar.

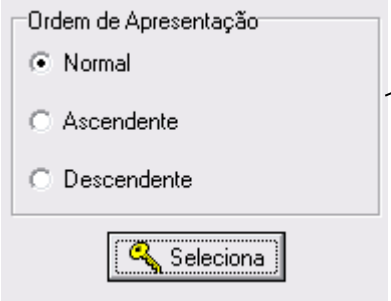
AVALIAÇÃO 7: Consultas com SQL – Clientes

A pesquisa para Clientes deve prever as seguintes opções:

Deve selecionar os registros por aproximação. Se deixar em branco deve selecionar todos os registros.

AVALIAÇÃO 8: Consultas com SQL – Vendedores

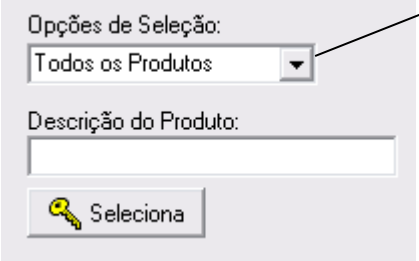
A pesquisa para Vendedores deve prever as seguintes opções:



Deve selecionar os registros ordenando-os pelo nome do vendedor.

AVALIAÇÃO 9: Consultas com SQL – Produtos

A pesquisa para Produtos deve prever as seguintes opções:



Items:
Todos os Produtos
Com Necessidade de Compra

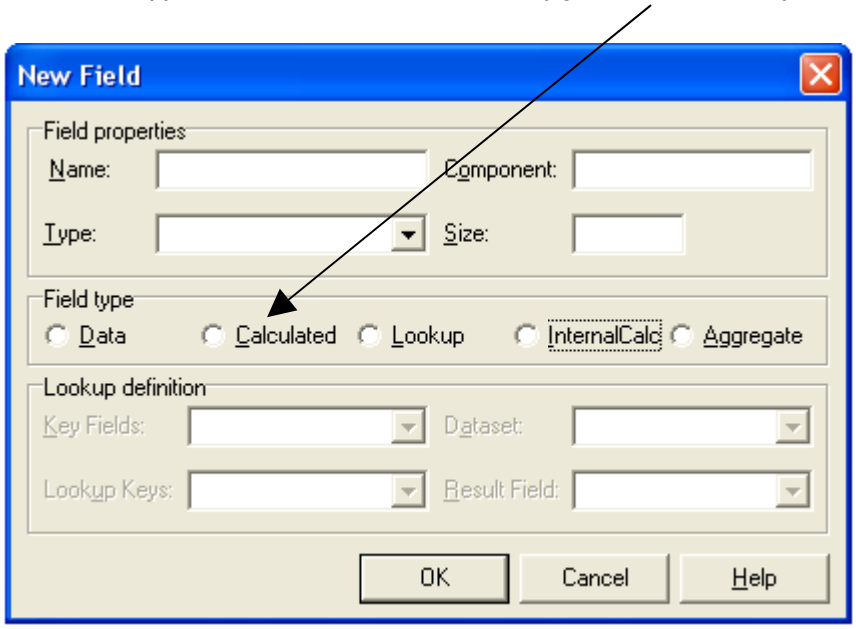
Se você analisar a estrutura da tabela PRODUTOS perceberá dois campos: QTESTOQUE (quantidade em estoque do produto) e QTMINIMA (quantidade mínima que o produto deve ter em estoque). Esses campos servem para nos dar a quantidade a ser adquirida do produto, pois se a QTESTOQUE for menor do que a QTMINIMA é sinal que este produto precisa ser reposto.

SUGESTAO DA QUANTIDADE A SER COMPRADA = QTMINIMA - QTESTOQUE

Porque ao criarmos a tabela de Produtos não foi criado um campo para armazenar esta quantidade a ser comprada? A resposta é simples: como é uma informação que pode ser calculada em tempo de execução, devemos sempre preferir a utilização de **campo calculado**, pois o seu conteúdo só ocupa espaço durante a execução do programa e a possibilidade de cálculo errado é nula. Veremos agora como criar e utilizar um campo calculado:

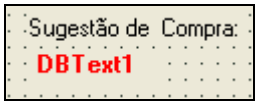
1. No módulo de dados dê um duplo clique no SimpleDataset correspondente a tabela PRODUTOS (que é SimpleDataSetProdutos). Clique com o botão direito no **Field Editor** e escolha Add All Fields.

   Clique novamente com o botão direito no **Field Editor**, escolha **New Field** e aparecerá a janela abaixo. Na caixa Field Properties digite **sugestaocompra** para **Name**, Type escolha float, escolha a opção **Calculated** para **Field Type.**



2. Clique em OK. Pronto, o campo calculado **sugestaocompra** foi criado.

3. Agora precisamos vincular o campo a um objeto de controle para permitir a sua visualização. Você deve acrescentar no formulário FrmProdutos um componente Label e um DBText conforme a aparência abaixo.



Poderia ser usado um DBEdit ao invés do DBText. A diferença é que o DBText não permite a edição do conteúdo do campo. E o que queremos é exatamente isso, não permitir que o usuário edite este campo, só visualize.

Configure o DBText da seguinte forma:

DataSource: DataSourceProdutos
DataField: sugestaocompra(campo calculado)
Font: *colocar vermelho e negrito*

4. Agora precisamos definir como o cálculo deve ser realizado. Selecione o SimpleDataSetProdutos e digite as instruções abaixo para o evento **OnCalcFields**:

```
if DM.SimpleDataSetProdutos.FieldByName('QTESTOQUE').asfloat <
   DM.SimpleDataSetProdutos.FieldByName('QTMINIMA').asfloat then
begin
   DM.SimpleDataSetProdutos.FieldByName('sugestaocompra').asfloat:=
   DM.SimpleDataSetProdutos.FieldByName('QTMINIMA').asfloat -
   DM.SimpleDataSetProdutos.FieldByName('QTESTOQUE').asfloat;
end;
```

5. Para que este novo campo possa aparecer no DBGrid faça o seguinte: dê um clique com o botão direito e escolha ADD. Uma coluna será criada. Escreva então **sugestaocompra** em sua propriedade FieldName.

AVALIAÇÃO 10: Campo Calculado (Valor do Produto em Estoque)

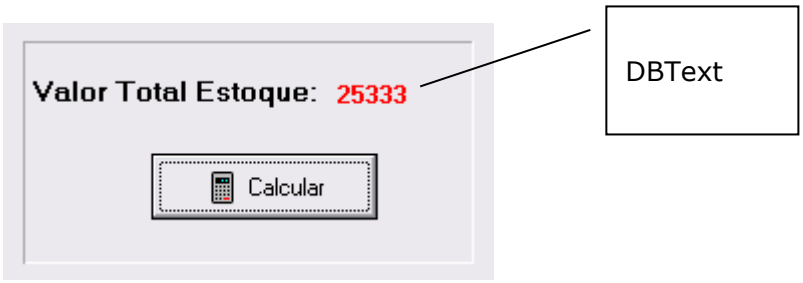
Faça sozinho este campo calculado. Ele deverá armazenar o valor de um determinado produto em estoque, ou seja, a quantidade do produto em estoque multiplicado pelo seu preço unitário.

Sugestão de nome para este campo calculado: **valorestoque.**

Não esqueça de coloca-lo visível tanto no DBGrid quanto no formulário, assim como foi feito para o campo calculado sugestaocompra.

O objetivo aqui é calcular o valor total em estoque, ou seja, acumular o valor de cada produto multiplicado pela sua quantidade em estoque.

1. Acrescente estes objetos no formulário de Cadastro de Produtos:



2. Utilizaremos um SimpleDataSet e um DataSource exclusivo para esta operação. Insira-os no formulário DM e mantenha o nome original.

3. No evento onClick do botão digite:

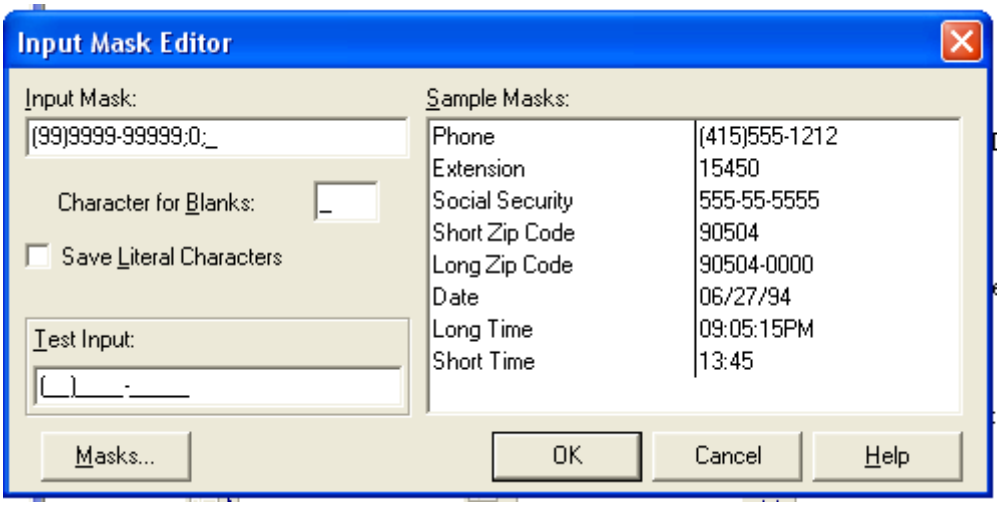
```
Dm.SimpleDataSet1.close;

DBText3.DataSource:=Dm.Datasource1;
DBText3.DataField:='TOTALESTOQUE';
Dm.SimpleDataSet1.DataSet.CommandText:='SELECT SUM(VLPRODUTO * QTESTOQUE)
                                       AS TOTALESTOQUE FROM PRODUTOS';
Dm.SimpleDataSet1.open;
```

6. Salve o Projeto e Execute-o.

Obs.: Tente usar outras instruções como AVG, MIN, MAX e COUNT.

Vamos agora facilitar a digitação do telefone no cadastro de fornecedores através da utilização de máscaras de edição.

1. Selecione o SimpleDataSet correspondente a fornecedores (SimpleDataSetFornecedores), dê um clique com o botão direito, selecione o Fields Editor, selecione o campo desejado (TELEFONE) e selecione a propriedade Edit Mask.



2. Em **Input Mask** coloque: (99)9999-9999. Os parênteses e o hífen não farão parte do conteúdo do campo somente aparecerão durante a edição do campo se você deixar desmarcada a caixa **Save Literal Characters**.

Para campos numéricos não existe a propriedade Edit Mask como você viu acima para campos strings. Para campos numéricos inclua nas propriedades **DisplayFormat** e **EditFormat** a seguinte máscara:

###,##0.00

AVALIAÇÃO 11: Utilizando Máscara de Edição

Agora faça sozinho. Crie máscaras para os seguintes campos: valor do produto, valor em estoque do produto e valor total em estoque da tabela Produtos, Cep da tabela Fornecedor, Telefone e Cep da tabela Clientes.

Estabelecendo Relacionamento Mestre – Detalhe (Registro das Vendas)

Existem inúmeras situações em que aparece um relacionamento do tipo Mestre-Detalhe. Alguns exemplos: Pedidos - Itens do Pedido, Clientes - Pagamentos Efetuados, Nota Fiscal – Itens da Nota Fiscal, dentre outros. Em nosso sistema a tabela mestre é VENDAS e a tabela detalhe é ITENS. Para estabelecer um relacionamento Mestre-Detalhe proceda da seguinte forma:

1. Identifique quem é a tabela Mestre e a tabela Detalhe. No nosso caso Mestre é Vendas e Detalhe é Itens.

2. Insira um SimpleDataSet (SimpleDataSetVendas) e um DataSource (DataSourceVendas) para a tabela Mestre no Módulo de Dados. Configure-os conforme você fez para todas as tabelas até agora. Faça a mesma coisa para a tabela Detalhe (Itens)

3. Agora é que vamos estabelecer o relacionamento entre as duas tabelas. A configuração é feita na tabela Detalhe (SimpleDataSetItens). Basta configurar as propriedades:

   MasterSource: DataSource da tabela mestre (DataSourceVendas)
   MasterField: Campo de ligação entre as duas tabelas (NRVENDAS)
   DataSet.CommandText: SELECT * FROM ITENS WHERE NRVENDA = :NRVENDA

4. Insira um novo formulário para o Registro das Vendas. Altere as seguintes propriedades para este formulário:

   Caption: Registro de Vendas
   Position: poScreenCenter
   Name: FrmVendas
   Salve a Unit como UFrmVendas

5. Insira no formulário FrmVendas um componente DBGrid e um DBNavigator para a tabela Vendas e também para a tabela Itens.

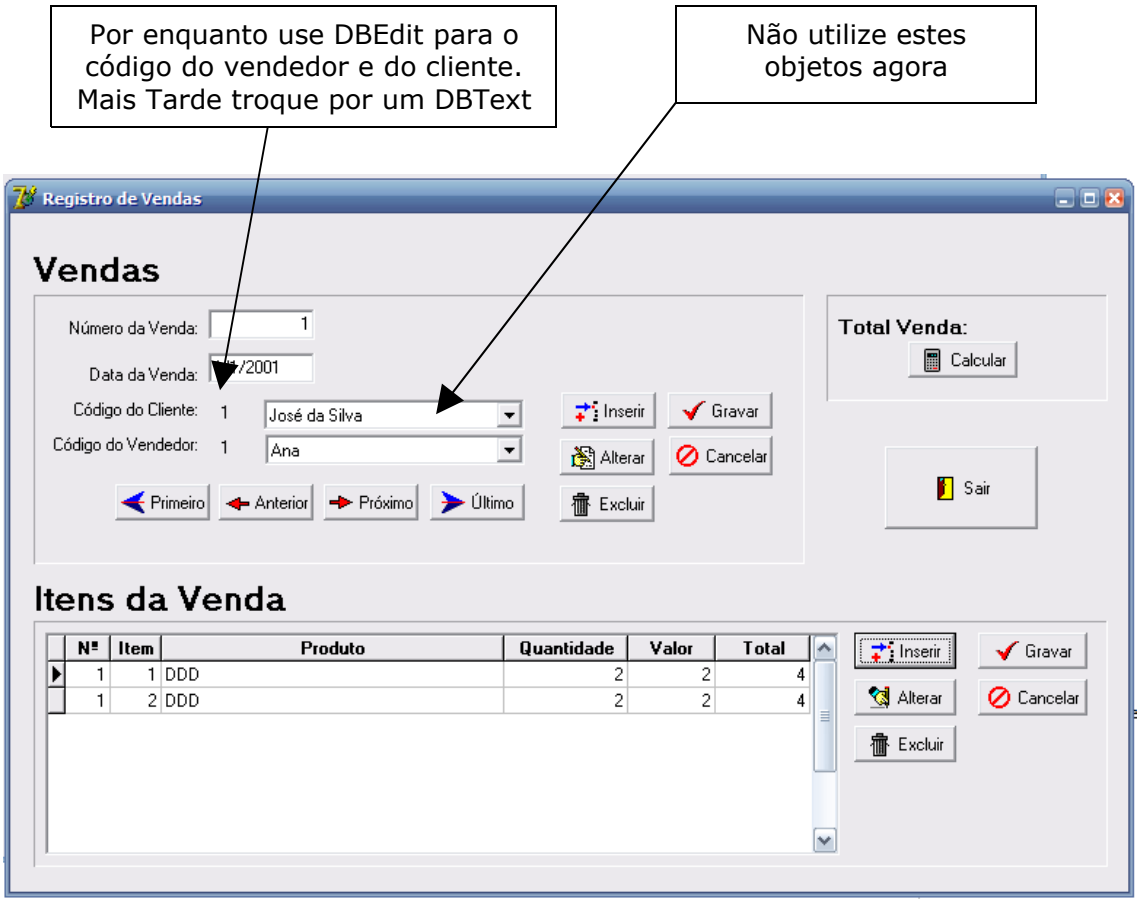
6. Agora vamos criar uma Opção no menu de opções (Objeto MainMenu) no formulário principal para que o usuário possa visualizar o formulário para registrar as vendas. Crie uma opção de nome **Movimentação** e com a sub-opção **Vendas**.

7. Lembre-se do método ApplyUpdates(0) para as duas tabelas.

8. Salve o Projeto e Execute-o. Inclua alguns registros nas duas tabelas.

AVALIAÇÃO 12: Melhorando o Registro das Vendas

1. Alterar o formulário de registro das vendas (FrmVendas) conforme abaixo:



AVALIAÇÃO 13: Valor do Produto Automático

No formulário de registro de vendas, especificamente quando o usuário escolher um determinado produto, fazer com o preço deste produto apareça automaticamente, sem que o usuário precise de digita-lo.

Dica: Utilizar o evento OnChange do campo CDPRODUTO da tabela ITENS (SimpleDataSetItens).

Parei aqui
Utilizando o DBLookupComboBox

Ao executar o programa, você deve ter observado que para registrar uma venda é obrigatório a digitação do código do cliente. Desse jeito o programa estará sujeito a alguns inconvenientes:

a) O usuário deverá saber o código do cliente.
b) O usuário poderá digitar um código inválido.

Para eliminar estes inconvenientes basta usar o componente **DBLookupComboBox** (paleta DataControls). Este componente permite listar em uma caixa de combinação (ComboBox) um campo de outra tabela. As propriedades a serem configuradas são:

DataSource: DataSource correspondente a tabela principal.
DataField: Campo da tabela principal

ListSource: DataSource da outra tabela
ListField: Campo da outra tabela que será listado.
KeyField: Campo de ligação entre as duas tabelas.

Vejamos agora como utilizar um DBLookupComboBox para permitir ao usuário escolher o nome do cliente (DCCLIENTE) ao invés de ter que informar seu código (CDCLIENTE).

1. Visualize o formulário de registro de vendas (FrmVendas).

2. Inclua ao lado do objeto DBEdit correspondente ao campo CDCLIENTE um objeto DBLookupComboBox e configure-o da seguinte forma:

   DataSource: DatasourceVendas
   DataField: CDCLIENTE
   ListSource: DataSourceClientes
   ListField: DCCLIENTE
   KeyField: CDCLIENTE

3. Salve e execute o programa. Agora o usuário não precisa saber o código dos clientes, basta escolher o nome do cliente no objeto DBLookupComboBox.

4. E se o usuário informar um código de cliente inválido no DBEDIT? O programa acusará um erro. Para evitarmos isto basta você substituir o DBEdit vinculado ao campo CDCLIENTE por um DBText.

5. Salve e execute o programa.

AVALIAÇÃO 13: Utilizando o DBLookupComboBox

Agora faça sozinho para o campo Código do Vendedor no formulário de registro de vendas,.

AVALIAÇÃO 14: Campo de Lookup (descrição do produto)

Melhorar ainda mais o formulário de Registro de Vendas (FrmvENDAS). Criar um campo de Lookup com a descrição do produto para a tabela Itens de Venda. Retirar a coluna correspondente ao código do produto do DBGrid e acrescentar este novo campo de lookup.

AVALIAÇÃO 15: Campo calculado (TotalItem)

Acrescentar no DBGrid relativo a Itens da Venda uma coluna com o total do item, ou seja, o valor do produto multiplicado pela quantidade vendida do produto.

AVALIAÇÃO 16: Campo Agregado (TotalVenda)

Acrescentar um DBText para visualizar o valor total das vendas, ou seja, a soma de todos os totais dos itens.

Ultima versão do formulário de Registro de Vendas

Agora vamos aprender a criar relatório usando o Rave Report, que foi incorporado a partir da versão 7 do Delphi. Será o relatório de listagem de fornecedores.

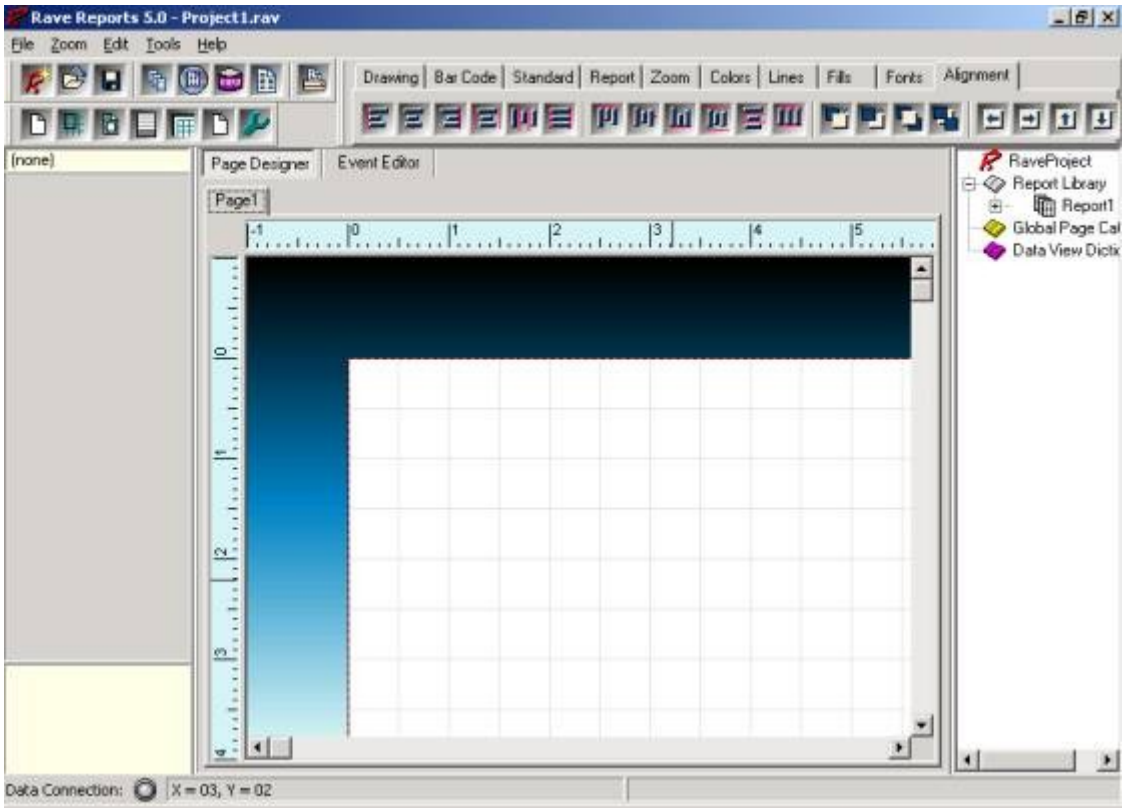
1. Vamos selecionar o módulo de dados **DM** e colocar nele 2 componentes da palheta Rave:

RvDataSetConnection: Exporta um DataSet do projeto Delphi para o projeto Rave. Configurar suas propriedade da seguinte forma:

Name: RvDataSetFornecedores
DataSet: ClientDataSetFornecedores

RvProject: Responsável pela ligação entre o projeto Delphi e o projeto de relatórios Rave. Podemos vincular vários relatórios a este objeto. Configurar sua propriedades **name** da seguinte forma: **RvProjectVendas.**

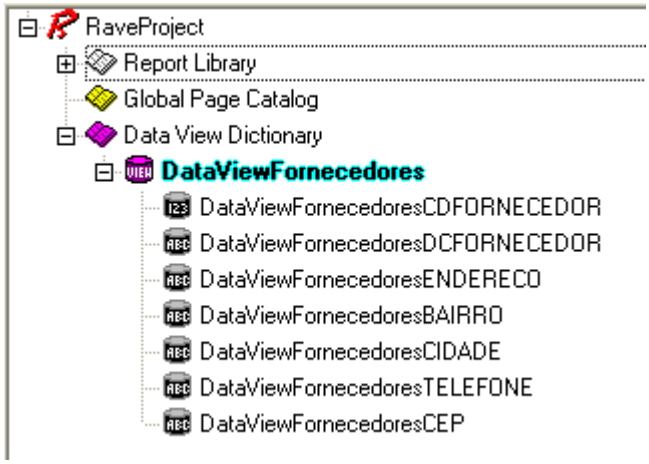
2. Agora dê um duplo clique sobre o componente RvProjectVendas para abrir o Rave Visual Designer.



Ambiente Design do Rave Report

3. Agora dentro do Rave, Vá em **File | New Data Object | Direct Data View** e selecione o RvDataSetFornecedores na área **Active Data Connection.** Os itens que aparecem nessa área são as conexões que colocamos em nosso formulário DM do Projeto em Delphi. No nosso caso só colocamos o objeto RvDataSetFornecedores.

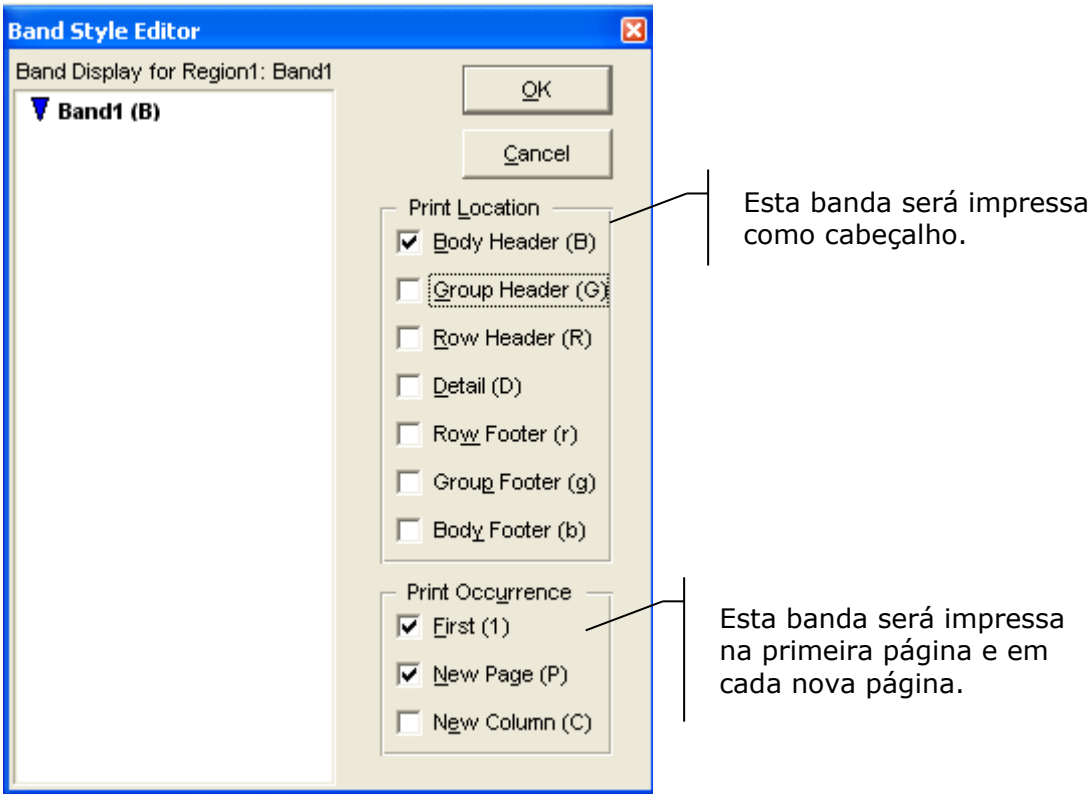
   Isso fará com que seja adicionado mais um item na seção **Data View Dictionary** que se encontra no Tree Panel, nesse local ficará todas as conexões com as tabelas que serão usadas nesse relatório (No nosso caso é só uma tabela). Selecione o item DataView1 e mude o seu nome para DataViewFornecedores através da propriedade Name.



Tree Panel

4. Vá até a guia **Report** e adicione um componente chamado **Region Component**. Todas as bandas (as partes do relatório – cabeçalho, corpo ou detalhe e rodapé) devem estar obrigatoriamente dentro de um Region e esse Region deve ocupar toda a área de impressão.

5. Vamos criar o cabeçalho do relatório. Adicione o componente **Band**, clique na propriedade **name** e mude seu nome de band1 para **Cabecalho.** Clique na propriedade **BandStyle** e marque as opções Body Header, First(1) e New Page(P).



O cabeçalho deverá ficar com a seguinte aparência:

Para colocar os títulos no cabeçalho de página, utilize o componente Text -  -da palheta Standard. Altere as propriedades Text, FontJustify e Font.

Para exibir a data de impressão do relatório, traga para o cabeçalho um componente DataText da palheta Report. Na sua propriedade DataField, clique nos pontinhos...

Irá aparecer o Data Text Editor. Para que apareça no relatório: **Emitido em quinta-feira, 21 de julho de 2005** faça o seguinte:

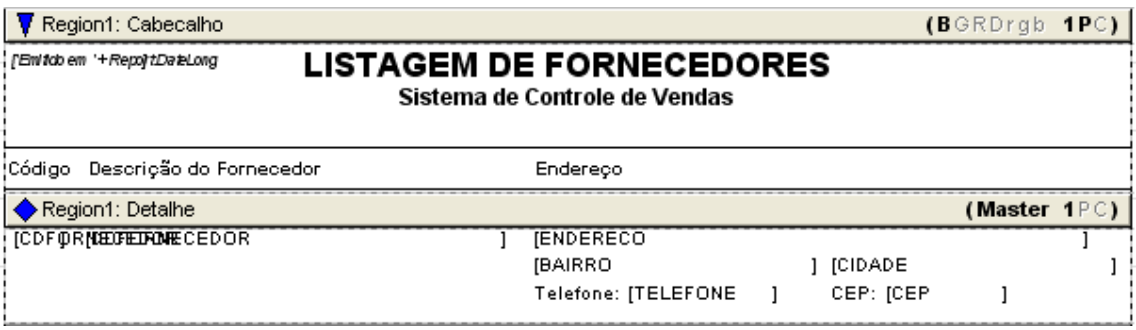
- Em Data Text digite: 'Emitido em '+
- em Report variables selecione DateLong e clique em Insert Report Var

Para colocarmos duas linhas horizontais conforme acima, selecione a palheta Drawing e traga um componente HLine -  . Utilize as propriedades LineWidth, Color etc.. para configurá-la como desejado.

6. Vamos agora criar o corpo do relatório (linhas detalhe). Na guia **Report** adicione o componente **DataBand**, mude sua propriedade name para **Detalhe** e ligue a propriedade **DataView** ao **DataViewFornecedores**.

   Devemos agora adicionar os campos que deverão ser impressos. Para isso, fique pressionando CTRL, clique no campo **DataViewFornecedoresCD_FORNECEDOR** e arraste ele para dentro de **Detalhe**, repita esse processo para os demais campos.

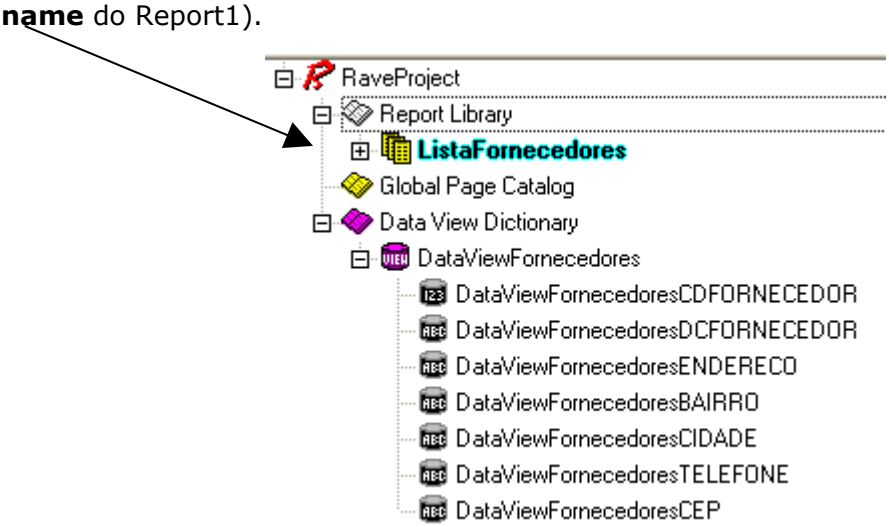
7. Selecione a banda **Cabecalho** e ligue a propriedade **ControllerBand** a banda **Detalhe**. Teremos algo semelhante a figura abaixo:



8. Vamos agora criar o rodapé do relatório, que constará do número da página e o total de páginas do relatório. Coloque um componente DataText (Report) no final da página e fora de Region1. Na sua propriedade DataField coloque:

```
'Página ' + Report.CurrentPage + '/' + Report.TotalPages
```

9. Será necessário diferenciar este relatório dos demais que iremos criar. Para tanto dê o nome a este relatório de **ListaFornecedores** (vá na propriedade **name** do Report1).



10. Para testar o relatório basta teclar F9.

11. Volte para o Delphi, faremos nossa aplicação Delphi iniciar a impressão desse relatório. Selecione o formulário de históricos e adicione um TBitBtn com o seguinte código:

```
DM.RvProjectVendas.ExecuteReport('ListaFornecedores');
```

12. Salve esse projeto como **RelatoriosVendas.rav** (opção **File | Save**) na mesma pasta em que gravamos o projeto em Delphi. Saia do Rave e volte para o Delphi.

13. No componente RvProjectVendas, aponte a propriedade **ProjectFile** para o projeto Rave que acabamos de criar (RelatoriosVendas.rav).

14. Salve o projeto e execute-o.

Resumo:

1. Inserir um objeto de conexão com a tabela (RvDataSetConnection) no Módulo de Dados. Para cada tabela do seu banco de dados que você desejar utilizar em relatórios vincular a um objeto RvDataSetConnection. Configurar as seguintes propriedades:

   Name: o nome do objeto
   DataSet: o nome do objeto do tipo ClientDataSet que está vinculado a tabela.

2. Inserir um objeto que permitirá criar um projeto de relatórios (RvProject). Dando dois cliques neste objeto você entra no Rave Visual Designer. Agora você poderá criar um ou mais relatórios.

3. Para cada relatório novo utilize (File | New Report). Mude o seu nome de Report1 para o nome desejado (vá na propriedade name do Report1).

4. Informar qual o RvDataSetConnection deste novo relatório ((File | New Data Object | Direct Data View e selecione o RvDataSetConnection desejado na área Active Data Connection).

   Isso fará com que seja adicionado mais um item na seção Data View Dictionary que se encontra no Tree Panel. Nesse local ficará todas as conexões com as tabelas que serão usadas nos relatórios. Selecione o novo item DataView1 e mude para o nome desejado.

5. Criar o relatório: Cabeçalho, Corpo e Rodapé.

   Cabeçalho: Componente **Band.**

   Corpo do Relatório: Insira um componente **DataBand** e em sua propriedade **DataView** informe o DataView correspondente a tabela desejada. Inserir um componente DataText para cada campo.

   Rodapé fica fora do região do relatório.

6. Para testar o relatório basta teclar F9.

7. Para fazer nossa aplicação Delphi iniciar a impressão do relatório:

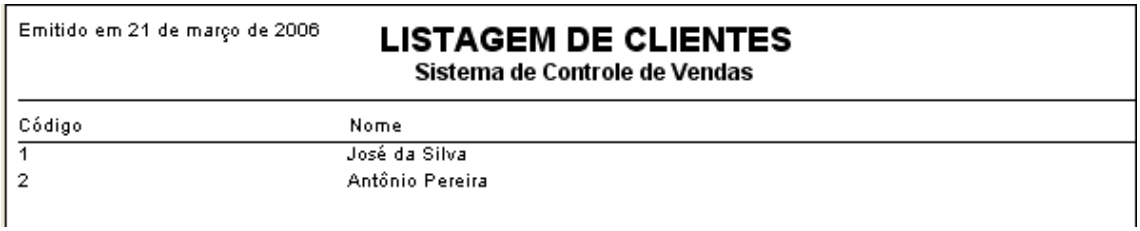
   ```
   RvProject1.ExecuteReport('Report1');
   ```

15. Salve o projeto de relatórios (File | Save).

16. No componente RvProject, aponte a propriedade **ProjectFile** para o projeto de relatórios que você acabou de gravar no item anterior.

AVALIAÇÃO 17: Listagem de Clientes

Faça sozinho um relatório de listagem de Clientes conforme abaixo:



Emitido em 21 de março de 2006

LISTAGEM DE CLIENTES

Sistema de Controle de Vendas

Código	Nome
1	José da Silva
2	Antônio Pereira

AVALIAÇÃO 18: Listagem de Vendedores

Faça sozinho.

AVALIAÇÃO 19: Listagem de Produtos

Faça sozinho.

Falta fazer o relatório de produtos com campos agregados
Falta fazer o relatório com agrupamento.

SCCC 19. PARTE: RELATÓRIO COM QUEBRA DE GRUPO

Objetivo:

- ✓ Criar relatórios onde a listagem dos registros será apresentada agrupada por determinado campo.
- ✓ Revisão de conhecimentos adquiridos

Para este relatório utilizaremos a tabela Lancamentos. Os registros serão listados agrupados por banco.

1. Inserir um novo objeto RvDataSetConnection no formulário DM. Mude seu nome para **RvDataSetLancamentos** e aproveita já configura a propriedade DataSet com **ClientDataSetlancamentos.**

2. Criaremos nosso novo relatório no mesmo projeto de relatório (RvProjectSCCC). Abra o Rave Visual Designer e selecione File - New Report. Irá aparecer um novo relatório de nome Report1, mude o seu nome de Report1 para **ListaLancamentos.**

3. Ainda no Rave Visual Designer crie um novo Data View e vincule ao **RvDataSetLancamentos**. Mude seu nome de DataView1 para **DataViewLancamentos**.

4. Insira um componente Region (Report). Neste componente Region insira os componentes abaixo e altere suas propriedades da seguinte forma:

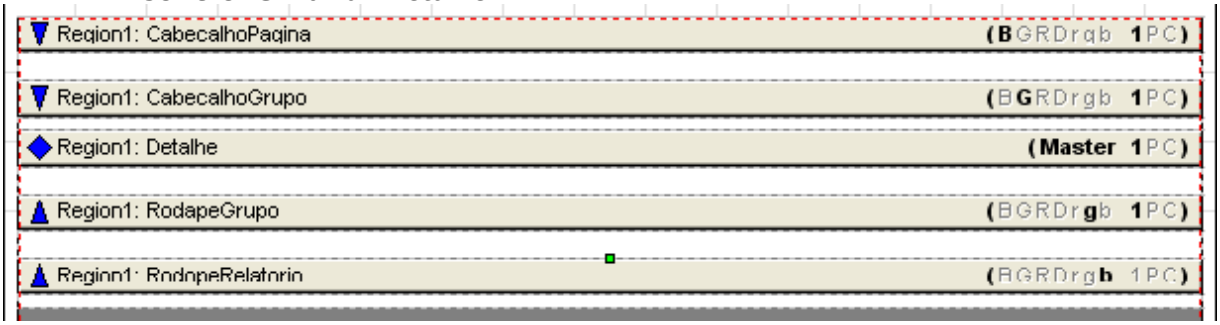
   - 1 componente **Band** (Report)
     Name: CabecalhoPagina
     BandStyle: Selecione BodyHeader e First
     ControllerBand: Detalhe

   - 1 componente **Band** (Report)
     Name: CabecalhoGrupo
     BandStyle: Selecione GroupHeader e First
     ControllerBand: Detalhe
     GroupDataView: DataViewContas
     GroupKey: DS_CONTA
     StartNewPage: true

   - 1 componente **DataBand** (Report)
     Name: Detalhe
     DataView: DataViewLancamentos

   - 1 componente **Band** (Report)
     Name: RodapeGrupo
     BandStyle: Selecione GroupFooter e First
     ControllerBand: Detalhe
     GroupDataView: DataViewContas
     GroupKey: DS_CONTA

- 1 componente **Band** (Report)
  Name: RodapeRelatorio
  BandStyle: Selecione BodyFooter
  ControllerBand: Detalhe



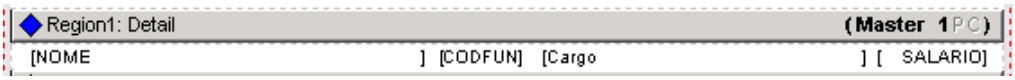
Visão parcial do projeto do relatório

Parei aqui

5. Monte a banda **CabecalhoPagina** da seguinte forma:

6. Monte a banda **CabecalhoGrupo** da seguinte forma:

Para a banda de detalhe, segure a tecla CONTROL e arraste os campos desejados para dentro da banda. Utiliza a propriedade FontJustify para alinhar os campos CODFUN e SALARIO à direita.



Para as bandas GroupFooter e ReportFooter, utilize o componente CalcText -  - da palheta Report para efetuar os cálculos de totalização.

Altere as propriedades: CalcType: ctSum

ControllerBand: Detail
DataView: dvEmp
DataField: SALARIO
DisplayFormat: #,##0.00

FontJustify: pjRight

1. Não esqueça de colocar no formulário correspondente a bancos uma "chamada" ao relatório.