Curso de Delphi

Carga horária - 10 horas em 5 aulas

Prof. Francesco Artur Perrotti

1. Introdução

Programar em Windows sempre foi algo extremamente complicado e acessível apenas a programadores dispostos a investir muito tempo e fosfato na leitura de pilhas de livros, intermináveis testes e análise de programas exemplos que mais confundem do que explicam. Mas porque é tão difícil fazer programas para Windows? Para começar, o Windows usa o conceito de GUI (Graphics User Interface), que embora seja muito familiar para usuários do Unix, é novidade para usuários do DOS. O uso de um sistema GUI implica em aprender vários conceitos que são estranhos ao usuário de um sistema baseado em texto como o DOS. Para complicar um pouco mais, o Windows é um sistema multi-tarefa, e as aplicações são orientadas a eventos, o que implica em aprender um novo estilo de programação. Finalmente, o programador tinha que ter alguma familiaridade com as centenas de funções oferecidas pela API do Windows. Por tudo isso, programação em Windows é um assunto que costuma provocar arrepios nos programadores.

Felizmente as linguagens visuais chegaram para mudar esta situação. Foi só com elas que o Windows consegui cumprir sua promessa de ser um sistema amigável e fácil de usar também para os programadores, que sempre tiveram que pagar a conta da facilidade de uso para o usuário.

Entre as linguagens visuais que surgiram, nenhuma veio tão completa e acabada quanto o Delphi. O Delphi vem com um compilador capaz de gerar código diretamente executável pelo Windows, proporcionando uma velocidade de execução de 5 a 20 vezes maior que as linguagens interpretadas. Além disso, vem também com um gerenciador de banco de dados completo e um gerador de relatórios. O tempo de desenvolvimento de qualquer sistema é reduzido a uma fração do tempo que seria necessário usando outras linguagens e o resultado é sempre muito melhor. É por isso que o Delphi fez tanto sucesso no mundo inteiro.

2. Conceitos básicos

Programação em Windows: Janelas e eventos

Existem muitas diferenças entre a programação em DOS e a programação em Windows. Vejamos a principal: Quando programamos em DOS, nosso programa é responsável pelo fluxo de processamento. Temos que definir claramente não só que instruções, mas também em que ordem devem ser executadas. Em Windows não é bem assim. Nosso programa não controla o fluxo de processamento, ele responde e trata *eventos* que ocorrem no sistema. Existem muitos *eventos* que podem ocorrer, sendo que os principais são aqueles gerados pelo usuário através do mouse e do teclado. A coisa acontece mais ou menos assim: O usuário clica o mouse e o Windows verifica que aplicação estava debaixo do mouse no momento em que foi clicado. Em seguida ele manda uma mensagem para a aplicação informando que ocorreu um clique e as coordenadas do cursor do mouse na tela no momento do clique. A aplicação então responde à mensagem executando uma função de acordo com a posição do mouse na tela. É claro que o Delphi toma conta do serviço mais pesado e facilita muito as coisas para o programador. Detalhes como as coodenadas da tela em que ocorreu o clique, embora estejam disponíveis, dificilmente são necessários nos programas.

Programação Orientada a Objeto (POO)

Embora não seja objetivo deste curso ensinar POO, uma breve introdução é necessária, já que o Delphi é essencialmente orientado a objeto.

De maneira prática, podemos pensar no objeto sendo uma estrutura que agrupa dados e funções para manipular estes dados. Como as funções são sempre "íntimas" dos dados, o sistema todo funciona de maneira mais segura e confiável. Além disso, a POO utiliza conceitos como *encapsulamento* e *herança* que facilitam muito programação e manutenção dos programas.

Neste ponto é oportuno citar que os dados de um objeto costumam ser chamados de *variáveis de instância* e as funçoes de *métodos*. As *variáveis de instância* definem as *propriedades* (as vezes chamada de *atributos*) do objeto e os *métodos* definem seu *comportamento*.

Encapsulamento - Como as variáveis e métodos estão na mesma estrutura, pode-se pensar em variáveis e métodos privados, ou seja, dados e funções que só podem ser manipulados pelas funções que estão dentro da estrutura. Desta maneira é possível formar uma camada protetora nos dados e evitar atribuições desastradas que comprometeriam o funcionamento do programa. Os defensores mais ortodoxos da POO dizem que todos os dados de um objeto deveriam ser privados e o número de funções públicas deve ser o menor possível, mas isso nem sempre é viável ou prático. O Delphi implementa este conceito e oferece dados/funções publicas (*public*) e privadas (*private*). Outra consequência do encapsulamento é que os objetos podem ser "caixas pretas". Não é necessário (teoricamente) conhecer detalhes de funcionamento de um objeto para usá-lo, basta enviar as mensagens apropriadas que ele responde com a ação desejada.

Classes - A classe representa um tipo ou categoria de objetos, o modelo a partir do qual um objeto pode ser construído. É a estrutura propriamente dita, que define os dados e métodos daquela classe de objetos. O objeto em si, é uma instância da classe. Na programação estruturada podemos fazer uma analogia com os tipos e variáveis, onde a classe equivale ao tipo e o objeto à variavel desse tipo.

Herança - É a capacidade que uma classe de objetos tem de herdar variáveis e métodos de outra classe. Esta capacidade permite que o código já escrito seja reutilizado de maneira muito mais eficiente e simples do que na programação estruturada. Um programa orientado a objeto costuma implementar verdadeiras árvores genealógicas de classes, com vários níveis de herança.

Para programar no nível do designer (veja adiante o que significa) não é necessário um conhecimento profundo de POO. Mas é preciso conhecer pelo menos a sintaxe. Todas as aplicações para Windows, precisam de pelo menos uma janela, que no Delphi é chamada de *Form*. Cada *form* (assim como todos os objetos visuais) tem um objeto associado a ele e sua representação visual, que vemos na tela. Todos os componentes que incluímos no form, passam a fazer parte do objeto que define o form. Exemplo: Se colocarmos um botão no form, a classe deste form será modificada para incluir este botão. Os eventos e métodos deste form, também estão na classe. Assim, supondo que o form se chame Form1 (nome default), para por exemplo desativar o botão que incluímos (de nome Buttom1) escreveríamos:

```
Form1.Buttom1.Enabled := false;
```

Note como o Buttom1 faz parte da estrutura que define o form. Mas se quisermos ativar método RePaint (Repintar) do form faríamos:

```
Form1.Repaint;
```

Veja que Repaint, não é uma variável, tecnicamente é uma procedure, mas fazemos referência a ela como parte da estrutura Form1. Pode parecer confuso no início, mas facilita muito a programação.

3. O ambiente do Delphi

O Delphi oferece dois níveis de programação distintos. Existe o nível do que o manual chama de *designer*, que se utiliza dos recursos de programação visual e aproveita componentes prontos, e o nível do *component writer*, que escreve os componentes para o *designer* utilizar nas aplicações. Podemos dizer que o *component writer* programa em um nível mais baixo e o *designer* em um nível mais alto. Para este curso, consideraremos a programação no nível do *designer*.

Quando ativamos o Delphi, a tela inicial é parecida com a figura 1. Na janela superior, temos a barra do menú principal do Delphi, à esquerda a *SpeedBar*, com as opções mais comuns e à direita a paleta de componentes. Estes componentes são a base da programação visual e é onde o *designer* vai buscar recursos para sua aplicação.

Abaixo da *SpeedBar*, está a janela do *Object Inspector*, que permite visualizar e modificar as propriedades e eventos de todos os componentes. É também largamente utilizado pelo *designer*. Abaixo da paleta ficam a janela de código-fonte e as janelas que estão sendo construídas.

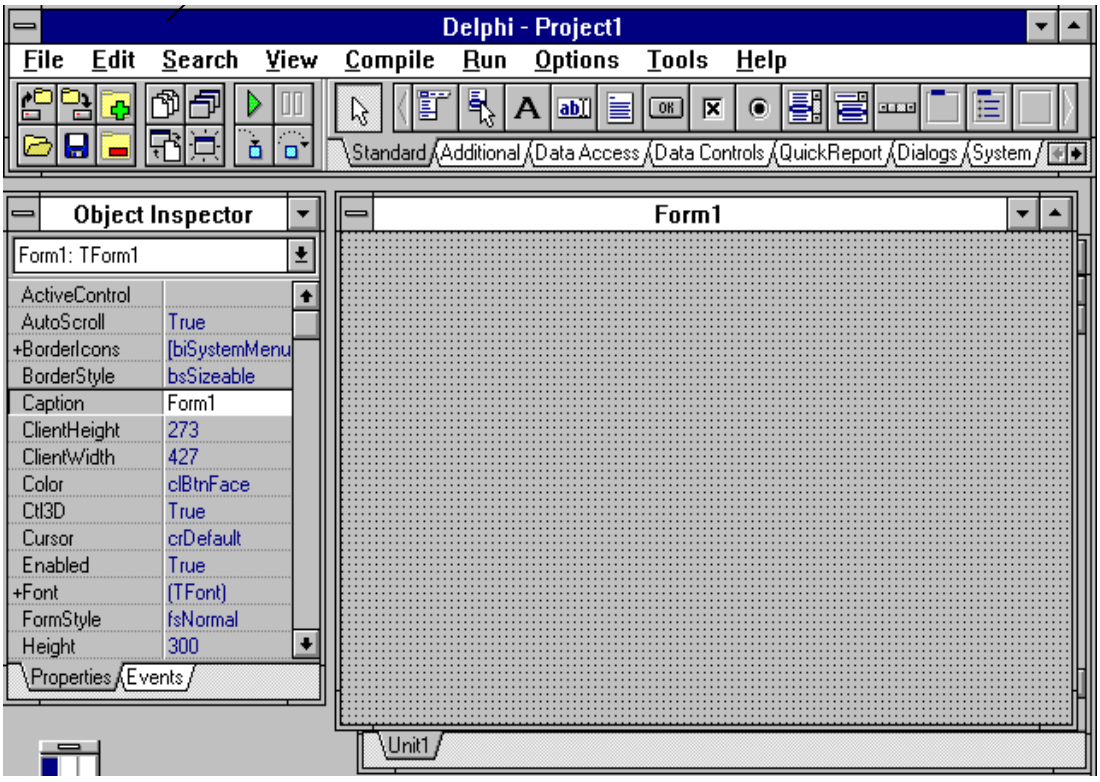

Figura 1 - Tela inicial do Delphi

Primeiro contato

Para iniciarmos poderíamos criar a versão Delphi do famoso "Hello Word", mas vamos partir para alguma coisa um pouco mais interessante e aproveitar para apresentar uma propriedade que pode ser útil no futuro. Como os programas em Windows são orientados a eventos é comum desativar opções de menús e botões, até que o usuário ative as opções que o sistema precisa para inicializar. Neste primeiro programa, vamos criar uma espécie de gangorra eletrônica. Mas antes de começar, vejamos o que está acontecendo no Delphi.

Se a tela está parecida com a figura 1, então não há nenhum projeto selecionado e o Delphi tomou a liberdade de criar um novo projeto para você, chamando-o de Project1. Um projeto é a coleção de arquivos necessários para compilar o sistema. Como toda aplicação precisa de pelo menos um form (Form1), ele também foi criado. Finalmente, todo form tem uma Unit correspondente que é mostrada no Editor de Código (Unit1).

Não há problema em utilizar esse projeto inicial que é oferecido pelo Delphi, mas é uma boa idéia renomear e salvar o projeto o quanto antes. Se você não escolher outro diretório, o projeto será salvo no diretório do Delphi e não é aconselhável salvar todos os seus projetos no mesmo diretório. Portanto vamos começar este programa criando um novo diretório para ele. Após isso, vamos seguir os seguintes passos:

1. Selecione a página *Standard* na paleta de componentes e clique no componente *Buttom*. A seguir clique no form. Um botão deve aparecer no form. Coloque mais dois botões.

2. Alinhe na horizontal os dois primeiros e coloque o terceiro logo abaixo dos dois. Se quizer um alinhamento exato, use a opção *Edit/Align...* do menú. Para marcar mais de um componente, deixe o *Shift* pressionado enquanto clica os componentes. Brinque um pouco com o tamanho dos botões e do form.

3. Clique o primeiro botão para selecioná-lo, a seguir procure no *Object Inspector* a propriedade *Caption*. Mude a string para **"ON".** Para o segundo botão a string é **"OFF"** e para o terceiro **"Close"**. Para o segundo botão, mude a propriedade *Enabled* para *False*.

4. Neste ponto é uma boa idéia renomear e salvar o projeto. Escolha a opção *File/Save project as...*Selecione o diretório criado para o projeto e use o nome **Main.pas** para a unit1 e **Gangorra** para o projeto.

5. Clique a área pontilhada do form e em seguida ache a propriedade *Caption* no *OI*. Mude a string para **"Gangorra".** Aproveite e mude a propriedade *Name* para "**fmMain".**

6. Dê um clique duplo sobre o primeiro botão. A janela de código será ativada já com a função correspondente ao evento OnClick criada e posicionada sob o cursor. Digite o seguinte código:

```
Button1.Caption := ‘OFF’;
Button1.Enabled := False;
Button2.Caption := ‘ON’;
Button2.Enabled := True;
```

7. Selecione a janela do form e faça o mesmo com o segundo botão, mas o código fica invertido:

```
Button2.Caption := ‘OFF’;
Button2.Enabled := False;
Button1.Caption := ‘ON’;
Button1.Enabled := True;
```

8. Finalmente para o terceiro botão o código é simplesmente:

```
Close;
```

9. Já podemos executar o programa pressionando *F9*.

O que está acontecendo: O evento OnClick ocorre sempre que clicamos o mouse sobre um componente. O Delphi se encarrega de determinar qual é o componente que deve responder ao evento e direciona o evento para ele. Neste caso, usamos este evento para ativar/desativar botões manipulando a propriedade Enabled, que determina se o componente está ativo ou não. Quando o componente está inativo, ele não responde aos eventos. Também usamos a propriedade *Caption* para mudar a mensagem que aparece nos botões. Finalmente usamos o método Close do form para fechar a aplicação.

Algumas Propriedades mais utilizadas:

***Name*:** É comum a todos os componentes da paleta. O Delphi nomeia automaticamente todos os componentes que são incluídos no form (inclusive o proprio form). Usa o nome da classe do componente mais um número sequencial. O nome atribuído pelo Delphi pode ser mantido, mas é aconselhável renomear os componentes que serão referidos no programa. Por exemplo, no programa da Gangorra, Button1 e Button2 deveriam ser renomeados, já que é feita referência a eles no código fonte, já para o Button3 não há necessidade por que não há referência a ele. Quando voce renomeia um componente, o Delphi atualiza automaticamente todo o código gerado pelo Delphi, o que inclui o cabeçário da Unit, os eventos do componente e as propriedades de outros componentes que fazem referência ao componente renomeado, **mas não atualiza o código gerado por você.** Exemplo: se renomearmos agora o Button1, o Delphi atualizará o cabeçário da unit, o nome dos eventos de Button1, mas você terá que atualizar as referências que você fez ao Button1 com o novo nome. Aliás esta é uma regra geral no Delphi: ele nunca modifica automaticamente o código gerado pelo programador, mesmo que esteja em comentário.

Caption: Todos os componentes que podem apresentar um rótulo tem esta propriedade. Armazena a string que será mostrada quando o componente for desenhado.

Left e Top: Esquerda e Topo. Armazenam a posição do componente em relação ao form ou painel que o contém. Movendo o componente, estas propriedades se atualizam automaticamente, e alterando estas propriedades, o componente é movido.

Height e Width: Altura e comprimento do componente. Idem acima.

Font: Permite selecionar tamanho e tipo da fonte que será usada para escrever o texto no componente.

Color: Cor do componente. Existe uma lista de cores padrão usadas pelo Windows e pelo Delphi, mas é possível definir qualquer cor através de seus componentes RGB.

TabOrder: Ordem do componente no Form ou painel. Quando há vários componentes selecionáveis no Form ou painel, a tecla Tab permite navegar entre os componentes. Esta propriedade define a ordem em que os componentes são selecionados quando o usuário tecla Tab.

Hint: (Dica) Este é um recurso muito útil e fácil de usar. Permite que apareça um texto de ajuda quando o usuário posiciona o cursor do mouse sobre um componente. Todos os componentes podem ter Hint. Na propriedade Hint, voce deve digitar a frase que deve aparecer. Veja a propriedade abaixo.

ShowHint: Ativa o hint para o componente. Se estiver desligado, o hint não é mostrado.

Alguns Eventos básicos:

OnClick: É gerado cada vez que o botão esquerdo do mouse é pressionado e solto. O evento só ocorre quando o usuário libera o botão. O Delphi já direciona o evento para o componente que está debaixo do cursor do mouse.

OnDblClick: Gerado quando é feito um duplo clique no botão esquerdo.

OnKeyPress: Gerado quando o usuário pressiona (e libera) uma tecla no teclado.

OnEnter: Gerado quando o foco de atenção do Windows "cai" sobre o componente. Exemplo: Suponha uma tela de entrada de dados, onde vários campos devem ser digitados. Quando a tela é apresentada, o foco, ou o cursor de texto, está sobre o primeiro campo. Depois de digitar o campo, o usuário pressiona Tab para *passar* para o campo seguinte. Veja que o que *passa* para o campo seguinte é a atenção da aplicação e do Windows. Essa atenção é chamada de foco, ou ***focus*** em inglês. Este evento é gerado assim que o componente recebe o foco, antes de executar qualquer código do componente.

OnExit: Gerado imediatamente antes de o foco deixar o componente.

OnResize: Gerado quando o tamanho do componente é alterado. É usado principalmente em forms e painéis.

OnChange: Gerado quando o valor do componente muda. Só é aplicado em componentes que permitem edição de seu conteúdo.

Alguns métodos úteis

Show: Desenha o componente. Se for uma janela (form) ela é desenhada e ativada.

Close: Fechar. Aplicado geralmente em forms e arquivos. Quando utilizado no form principal, encerra a aplicação.

Repaint: Repintar. Redesenha o componente ou form.

Refresh: Tem o mesmo efeito que o ***Repaint***, mas antes de desenhar, apaga o componente. Quando aplicado em arquivos, faz com que o buffer do arquivo seja recarregado.

Create: Aloca memória para criar um componente ou form, dinâmicamente.

Free: Libera a memória alocada com o ***Create***.

4. Uma aplicação mais sofisticada

A partir de agora, passaremos a desenvolver um aplicação que vai usar os recursos básicos para a maioria das aplicações. Conforme vamos incrementando a aplicação, veremos os principais conceitos da programação em Delphi. A aplicação é um mini sistema de pedidos. O sistema manterá cadastros de clientes e produtos e será capaz de gerar e armazenar os pedidos dos clientes.

Menús

Como primeiro passo, cuidaremos do sistema de menús. O componente do menú é o primeiro na página Standard da paleta de componentes. Coloque o menú no form. O ícone apareceu mas ainda não há nenhuma barra de menú. Para entrar no editor do menú é necessário um duplo clique no ícone no form. Note que quando o editor é ativado o *Object Inspector* passa a mostrar as propriedades e eventos dos itens de menú. Selecione a propriedade *Caption* e digite *Arquivo.* O menú passa a mostrar agora uma opção para Arquivo e um retângulo pontilhado ao lado. Clique no retângulo, selecione a propriedade *Caption* e digite *Cadastros*. Repita a operação e coloque mais um ítem para *Ajuda*.

Neste ponto já temos o que num sistema DOS chamaríamos de menú principal. As opções que aparecem horizontalmente na barra de menu, normalmente ativam sub-menús que oferecem mais opções, e estas eventualmente podem abrir mais sub-menus, num leque que pode crescer bastante, se necessário. Para criar um sub-menu do menú principal, basta clicar na opção desejada e utilizar o retângulo que aparece abaixo da opção. Crie agora os seguintes sub-menús:

Para Arquivo: Sair

Para Cadastro: Clientes, Produtos e Pedidos

Para Ajuda: Sobre o sistema...

Podemos associar letras a opções do menú. Nesse caso a letra aparece sublinhada no menú e o menú principal é ativado quando o usuário tecla ALT + <letra escolhida>. Só é preciso tomar cuidado para escolher letras diferentes para cada opção no mesmo nível. Exemplo: no primeiro nível temos as opções Arquivo, Cadastro e Ajuda. Poderíamos escolher a letra A para Arquivo e a letra C para Cadastro, mas a opção Ajuda não podería estar tambem associada à letra A, o mais aconselhavel seria escolher a letra J. Para selecionar uma letra, basta acrescentar o símbolo & (e comercial) antes da letra na propriedade Caption.

Para criar sub-menús em sub-menús, selecione a opção que será a raiz e pressione CTRL + Seta Direita. Crie as opções *Cadastro* e *Listagem* no cadastro de Clientes e no de Produtos. No cadastro de Pedidos crie três opçoes: Cadastro, Impressão e Listagem. O menú completo fica assim:

- **Arquivo**
  - **Sair**
- **Cadastros**
  - **Clientes**
    - **Cadastro**
    - **Listagem**
  - **Produtos**
    - **Cadastro**
    - **Listagem**
  - **Pedidos**
    - **Cadastro**
    - **Impressão**
    - **Listagem**
- **Ajuda**
  - **Sobre o sistema...**

Antes de fechar definitivamente o editor de menús, temos ainda outra tarefa. Note que a opção Cadastro apararece em 3 sub-menús. Em cada um deles, o ítem de menú para Cadastro deve receber um nome diferente do Delphi, e ele resolve o problema chamando estes ítens de *Cadastro1*, *Cadastro2* e *Cadastro3*. Apenas para deixar o código fonte mais claro e legível, poderíamos renomear estes ítens para *mnCadCli*, *mnCadProd* e *mnCadPed*. O prefixo *mn* indica que é um ítem de menú. É um bom hábito colocar prefixos assim nos nomes dos componentes, porque facilita muito a leitura do código fonte. Para fixar melhor este hábito, renomeie também as opções de Listagem.

Agora já podemos fechar o editor de menús, dando um clique duplo no botão que aparece à esquerda da barra de título da janela. Falta agora criar o código para cada opção do menú. A maioria das opções, deixaremos para codificar mais adiante, de imediato vamos apenas codificar a opção de saída do sistema. Já com o editor de menús fechado, selecione a opção *Arquivo/Sair* no form da aplicação e escreva `Close;`. Nossa aplicação agora já tem como fechar.

Painéis

Embora não pareça, o painel (***Panel***) é um componente extremamente útil e poderoso. Sua aparência é de uma placa metálica e ele funciona como uma sub-janela. Todos os componentes que são colocados sobre um painel, passam a "pertencer" ao painel. Se o painel for movido dentro da janela, os componentes sobre ele se movem junto. Com os painéis é possivel dividir facilmente a janela em regiões e distribuir melhor os componentes. O Delphi também utiliza largamente os painéis de maneira puramente estética, como moldura de grades e imagens.

Em nossa aplicação usaremos um painel para criar uma barra de botões que permita entrar mais rapidamente nos cadastros. Primeiro diminua um pouco a altura da janela. Coloque um painel no form e apague a string que está na propriedade *Caption*. Aumente um pouco o tamanho vertical dele e em seguida, ajuste a propriedade *Align* para *alTop*. O painel agora está configurado para ficar sempre no topo da janela. O tamanho horizontal dele será sempre igual ao espaço interno da janela, mesmo quando a janela muda de tamanho e o tamanho vertical pode ser ajustado à gosto.

A propriedade *Align* tem as seguintes opções de alinhamento:

alTop: alinha no topo da janela (ou painel)

alBotton: alinha embaixo (rodapé)

alLeft: alinha à esquerda

alRight: alinha à direita

alClient: Faz o painel ocupar todo o espaço disponível da janela ou painel

alNone: Não faz nenhum alinhamento.

Neste painel colocaremos 4 botões, tres à direita e um à esquerda, como na figura 2. Mas desta vez usaremos o botão *BitBtn* que está na página *Aditional* na paleta de componentes.

BitBtn

A diferença entre o *BitBtn* e o *Button* é que o primeiro permite incluir um ícone (é chamado de ***glyph***) no botão. Veja como todos os botões que o Delphi usa para cancelar uma operação tem um X vermelho no botão. Esse X é um *glyph.*

Marque todos os botões e altere as propriedades *Heigth* para 60, *Width* para 80 e *Layout* para *blGlyphTop*. Agora desmarque os botões porque as próximas alterações precisam ser feitas individualmente no botões.

No *Caption* do primeiro, escreva *Clientes*, no segundo *Produtos* e no *terceiro Pedidos*. Antes de alterar o *Caption* do quarto, vamos alterar a propriedade *Kind* para *bkCancel*. Veja que o *Caption* dele agora é Cancel e apareceu o X vermelho citado antes. Agora sim podemos mudar *Caption* do quarto botão para *Fechar*.

Dê um clique duplo na prop. Glyph do primeiro botão. Aparece a janela Picture Editor, clique o botão Load. A seguir selecione o diretório C:\Delphi\Images\Buttons. Digite (ou selecione) Picture.Bmp e pressione Ok. Para o segundo botão, selecione o glyph Gears.Bmp e para o terceiro Npadwrit.Bmp.

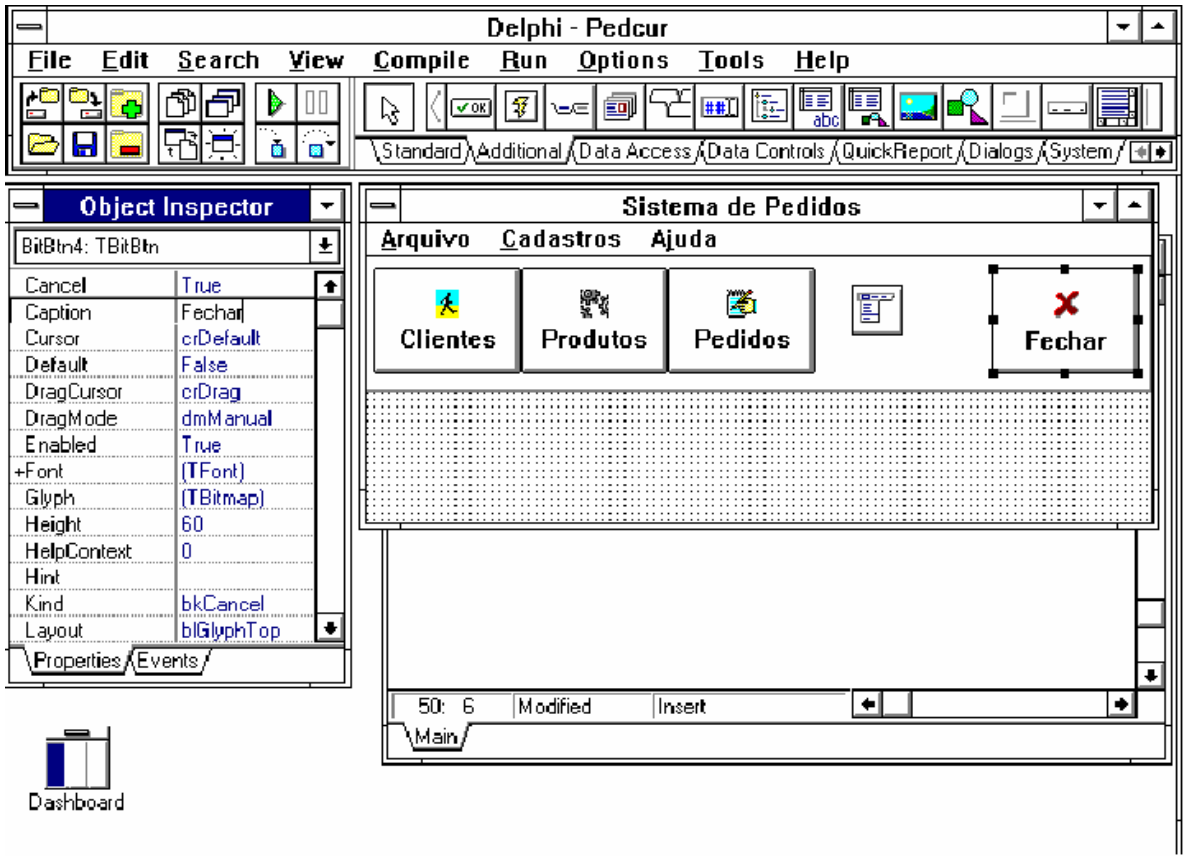

Figura 2 - Janela principal do Sistema de Pedidos

Antes de passar para o próximo tópico é uma boa idéia salvar o projeto. Faça isso antes de continuar.

Usando Templates

O recurso de *Template* que o Delphi oferece permite economizar muito tempo de programação e padronizar melhor o sistema. Qualquer janela pode ser salva como um template e depois reutilizada varias vezes nas construção de novas janelas. Aliás voce pode salvar um projeto inteiro como template e iniciar seus novos projetos a partir do template salvo.

Para experimentar, vamos criar uma janela de About usando um template fornecido pelo Delphi. Selecione o botão *"New Form"* na barra de ferramentas. Escolha a opcao *About Box* na tela de templates e pressione **Ok**. Uma janela de About será criada e já incorporada ao projeto. Modifique o Caption dos *Labels* que já estão na janela para por seus dados e do sistema. Dê um clique duplo no botão **Ok** e escreva Close; no editor de código. No Caption do form da janela escreva "Sistema de Pedidos", mude o nome do form. Agora salve os arquivos da janela pressionando o botão *Save File*. Coloque *About.Pas* no nome da unit.

Agora só falta associar a janela à opção *Ajuda/Sobre o sistema* com a janela que foi criada. Na janela principal clique sobre a opção para criar um evento e abrir o editor de código. Digite:

```
fmAbout.Show;
```

Ainda dentro do editor de código, procure no topo do arquivo a lista de units utilizadas pela janela principal. Acrescente ao final a unit *About*, que acabamos de salvar.

Está pronto, agora é só executar pressionando **F9**.

O que está acontecendo:

5. Usando o Data Base

Boa parte dos sistemas em uso nas empresas faz uso de uma base de dados. O Delphi fornece um gerenciador de banco de dados *completo (Borland Database Engine - BDE*) capaz de atender facilmente grande parte destes sistemas.

Junto com o pacote do Delphi está o *DataBase Desktop*, que facilita muito a vida do programador, permitindo criar e modificar à vontade os arquivos, índices e referências.

Neste ponto, gostaria de ressaltar a importância de planejar o sistema de arquivos que será utilizado antes de começar a criar arquivos no *DataBase*. Um sistema pequeno e didático de 4 arquivos (ou ***Tables***) como o que iremos criar, pode até caber inteiro na nossa cabeça, mas na vida real, os sistemas são sempre maiores e mais complexos. Uma ferramenta muito útil para navegar pelos arquivos sem se perder é o *DER - Diagrama de Entidade-Relacionamento*. Através dele podemos ver as relações entre os arquivos e entender suas dependências. Na página seguinte é mostrado o *DER* do nossa aplicação.

Alias

Um conceito importante que deve ser dominado antes de continuarmos é o de *Alias*. Pode ser considerado um apelido para um diretório. Voce associa um nome a um diretório e passa a se referir ao diretório por esse nome. Por exemplo: suponha que enquanto voce está desenvolvendo o sistema usa o diretorio *C:\Fontes\Sist1\DB* para colocar os arquivos do banco de dados do sistema. Em vez de digitar a todo momento esse diretório quando criar uma referencia a um arquivo, voce pode associar o alias *alSist1* para esse diretório. Mas diminuir a digitação não é o principal benefício do alias. O principal é que depois que a aplicação estiver pronta, certamente será instalada em outro diretório, geralmente em outro micro e tendo um alias, basta modificar o diretório associado para o novo diretorio e nenhuma mudança será necessária no sistema.

DataBase Desktop

Para começar vamos dar uma boa olhada no *DER* da figura 3. Temos o cadastro de clientes que armazena dados sobre os clientes. Da mesma maneira, o cadastro de produtos. Para o cadastro de pedidos, existe um problema: Se o Pedido relacionar diretamente o Cliente com o Produto, teremos uma relação de M para M (Muitos para Muitos), por que um cliente pode pedir muitos produtos e um produto

pode fazer parte de muitos pedidos. Procuramos sempre evitar esse tipo de relação, então incluímos mais uma relação: o Item de Pedido. Relacionando Pedidos com Produtos através de Itens de Pedido, ficamos apenas com relações de 1 para M. Um pedido pode ter muitos itens, mas um item só pode fazer parte de um pedido. Da mesma maneira, um produto pode fazer parte de muitos itens, mas um item só pode ter um produto associado.

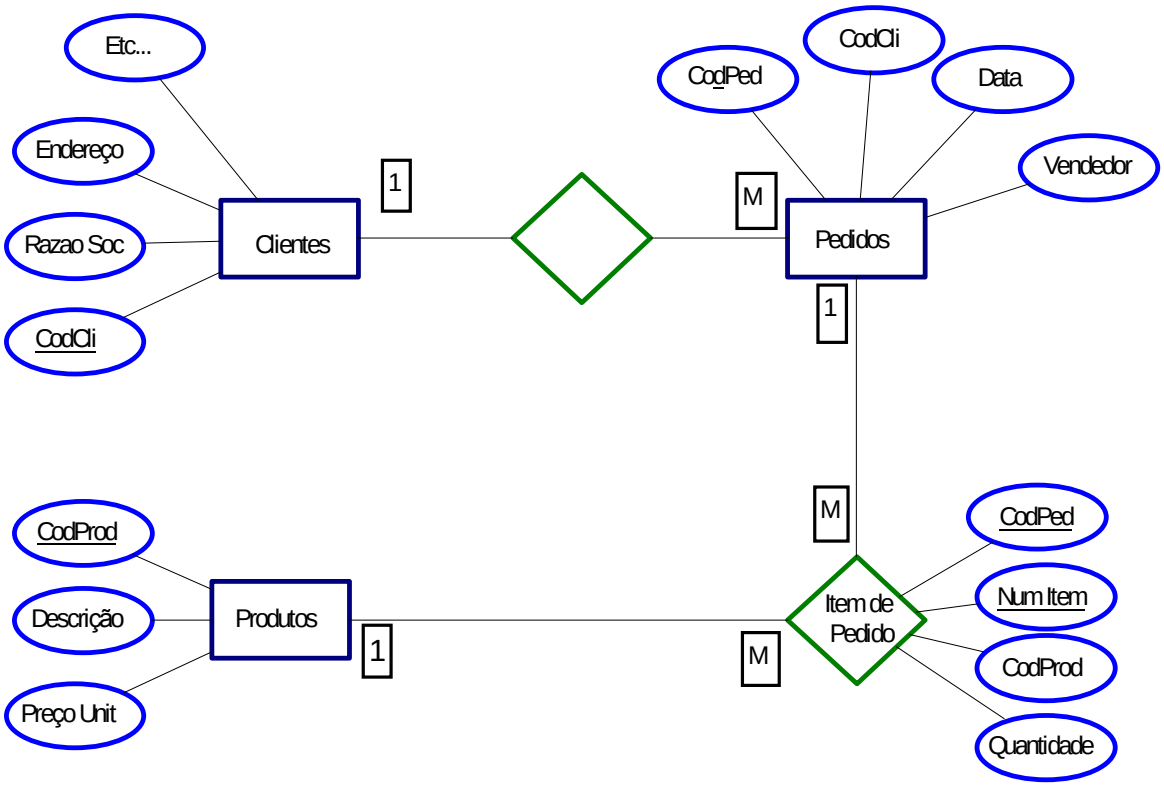

Figura 3 - DER do sistema

Só mais um detalhe antes de ativar o Database Desktop. Cada arquivo de dados na base de dados tem associado a ele vários arquivos necessários para a criação de índices, verificação de referências, validação e etc. Portanto o número total de arquivos na base de dados aumenta rapidamente conforme vamos implementando novos cadastros. Por esse motivo, não é aconselhável deixar a base de dados no mesmo diretório que os programas fontes. Debaixo do diretório que voce criou para o projeto, crie um subdiretório chamado **DB.** É nesse diretório que vamos criar nossa base de dados.

Finalmente estamos prontos para ativar o DataBase Desktop. Depois de ativado, nossa primeira tarefa é criar um aliás para nossa base de dados. Para isso ative a opção *File/Aliases* no menú principal. O DataBase responde mostrando a

janela de diálogo para os aliases. Pressione o botão *New* para criar um novo e em seguida digite *alPedidos* na caixa rotulada com *DataBase Alias* depois coloque o caminho completo do diretório na caixa *Path*. A seguir pressione o botão OK e pronto. Nosso alias está criado.

Outro conceito do DataBase é o *Work Directory* (Diretório de trabalho). Este é o diretório default para o Desktop. Tudo que é feito no Desktop acontece nesse diretório. Clique a opção do menú *File/Working Directory* e na caixa Aliases escolha o alias que acabou de ser criado. Pressione Ok para confirmar.

Para começar a criar os arquivos, selecione a opção *File/New/Table*. Na janela Table Type deixe a opção default (Paradox 5.0) e tecle Ok. Neste ponto o DataBase abre o diálogo de estrutura de arquivo (o Database chama de ***Table*** - Tabela). como aparece na figura 4.

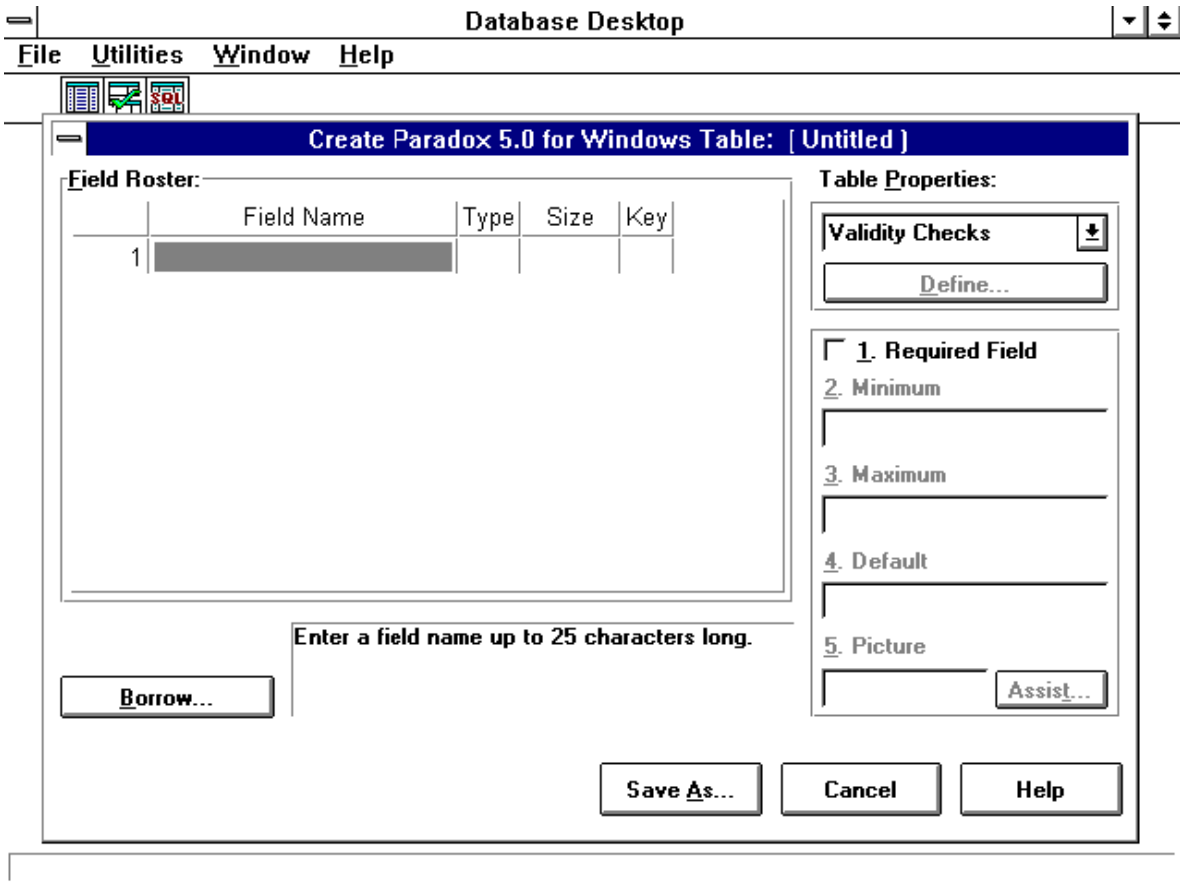

Figura 4. Diálogo para criação de tabelas

Clientes

Vamos começar criando o arquivo de clientes. Use os valores da tabela abaixo para definir os campos do arquivo de clientes:

Field Name	Type	Size	Key
CodCli	A	6	*
RzSocial	A	40	
Endereco	A	40	
Cidade	A	30	
Cep	A	9	
UF	A	2	

Clique no campo CodCli e ative a opção *Required Field*. Isso fará com que a digitação deste campo seja obrigatória e o *BDE* se engarregará de tomar os devidos cuidados sem necessidade de intervenção. Isso é desejável já que CodCli é a chave primária da tabela de Clientes.

Ao terminar pressione o botão *Save As*. O diálogo é fechado e aparece a janela do arquivo criado. Podemos ver os nomes dos campos no topo das colunas, mas a tabela ainda está vazia. Para acrescentar um registro, pressione o *botão Edit Data*, é o botão mais a direita na barra de botões. Clique no campo *CodCli* do registro 1 e comece a digitação. Para mudar de campo tecle **Enter** ou **Tab.** Quando acabar, desligue o modo de edição pressionando novamente o botão *Edit Data*.

Produtos

Para criar a tabela de produtos, use os dados abaixo. Não esqueça de ativar a opção *Required Field* no campo *CodProd*.

Field Name	Type	Size	Key
CodProd	A	6	*
Descricao	A	30	
PrecoUnit	$		

Pedidos

A tabela de pedidos requer alguns cuidados extras, porque depende da tabela de clientes. Todo pedido deve pentencer a um cliente. Depois de criar os campos para esta tabela, selecione na caixa *Table Properties* a opção *Referential Integrity* e a seguir pressione *Define*. Agora voce deve relacionar um campo com uma tabela, no caso, o campo CodCli com a tabela de Clientes. Para isso dê um clique duplo em CodCli na lista à esquerda e outro na tabela de clientes que aparece na lista à direita. Pressione Ok e digite irClientes como nome para a relação. Esta operação torna a tabela de Pedidos oficialmente dependente da tabela de Clientes no ponto de vista gerenciador de banco de dados - BDE e nós podemos deixar esse controle por conta do BDE.

Field Name	Type	Size	Key
CodPed	A	6	*
CodCli	A	6	
DtEmissao	D		
Vendedor	A	12	

Itens de Pedidos

Esta é a tabela mais dependente do sistema. Depende de pedidos e de produtos, então devem ser criadas duas *Referential integrity*: uma relacionando o campo *CodPed* com a tabela de pedidos, e outra relacionando o campo *CodProd* com a tabela de Produtos. Note também que esta tabela tem chave dupla: os dois primeiros campos compõe a chave primária, ou seja, para identificar um único registro é preciso saber os dois campos.

Field Name	Type	Size	Key
CodPed	A	6	*
NumItem	S		*
CodProd	A	6	
Quantidade	N		

Lembrete

Não esqueça de definir pelo menos um registro em cada tabela. Esse registro será útil mais adiante quando criarmos as janelas de cadastro. Quando definir os campos que são chave numa tabela, deixe sempre ligada a opção *Required Field*.

6. Usando o Data Base Form Expert

Depois de terminada a criação de tabelas e já de volta ao Delphi, é hora de criar as janelas para os cadastros do sistema. Para esse tipo de janelas o Delphi oferece uma ferramenta excelente: o *Data Base Form Expert*. Os *Experts* são especializados em criar janelas a partir de uma iteração com o usuário. O usuário responde algumas perguntas e no final do processo uma janela é criada. O *Data Base Form Expert* é especializado em criar janelas para acesso ao banco de dados. Vamos exemplificar criando o cadastro de clientes. ComeÇe pressionando o botão *New form* e no *Browse Gallery* selecione a página *Experts*. O Database é o default então é só clicar Ok.

Neste ponto começam as perguntas que definirão a janela que será criada. Siga a sequencia abaixo para definir a janela de cadastro de clientes.

1a. tela: Serve para selecionar o tipo de form que será criado e o tipo de arquivo que será utilizado. Deixe tudo no default e clique *Next*.

2a. tela: Define o arquivo que será utilizado. No nosso caso é o de Clientes. Antes de selecionar o arquivo, não esqueça de selecionar na caixa *Drive or Alias name* o alias que criamos para este sistema.

3a. tela: Define que campos e em que ordem vão aparecer na janela. Clique o botão **>>** para selecionar todos os campos. Depois voce pode usar a lista à direita para ordena-los.

4a. tela: Estilo da disposição dos campos na janela. Deixe no default e clique *Next*.

5a. tela: Finalmente na ultima tela, desative a opção *Generate a main form*. Se ficar ativada a janela ficará no topo da lista do projeto e será a primeira janela a ser ativada quando o sistema for executado. A seguir clique *Create* para finalizar a criação da janela.

A janela está criada, mas podemos notar que o Expert não tem um senso estético muito apurado. Em geral as janelas criadas assim precisam ser arrumadas movendo e mudando o tamanho de alguns componentes. Mesmo assim o tempo economizado na criação da janela é bem significativo, vale a pena.

Use a mesma sequência de ações para criar a janela de produtos. Para a janela de pedidos, o procedimento mudará um pouco. Veja o tópico seguinte.

Criando um form mestre-detalhe

A relação mestre-detalhe existe quando um arquivo depende de outro. Por exemplo, podemos considerar o arquivo de pedidos mestre do arquivo de ítens de pedido (que é detalhe) um ítem de pedido só pode existir como parte de um pedido, ou seja, depende da existência de um pedido. A mesma coisa acontece quando consideramos o arquivo de clientes e o de pedido, mas agora o de pedidos é detalhe. O arquivo mestre neste caso é o de clientes. Assim como podem haver muitos itens em um pedido mas apenas um pedido associado a cada item, podem haver muitos pedidos para um cliente, mas apenas um cliente em cada pedido.

O BDE é muito bom para tratar casos assim e permite arranjos altamente sofisticados nas telas de entrada/saída de dados sem muito esforço do programador. Para testar esse recurso, vamos criar o form de pedidos de maneira a mostrar seus itens. Siga os passos abaixo:

1a. tela: Selecione a opção *Create a master/detail form* na caixa *Form Options* e clique *Next*.

2a. tela: Escolha do arquivo mestre. Neste caso é o de Pedidos. Não esqueça de selecionar o alias que criamos antes de escolher o arquivo.

3a. tela: Clique o botão **>>** para selecionar todos os campos do arquivo de pedidos.

4a. tela: Estilo da disposição dos campos na janela. Deixe no default e clique *Next*.

5a. tela: Escolha do arquivo detalhe. Selecione o de Ítens de Pedidos. Não esqueça o alias.

6a. tela: Clique o botão **>>** para selecionar todos os campos do arquivo detalhe.

7a. tela: Estilo. Deixe no default e clique *Next*.

8a. tela: Ligação dos campos na relação mestre-detalhe. Na caixa *Detail fields*, clique em *CodPed*. Na caixa *Master fields* clique em também em CodPed. A seguir pressione o botão Add. A relação está agora completa.

9a. tela: Finalmente a ultima tela. Desative a opção *Generate a main form* e seguir clique *Create* para finalizar a criação da janela.

Depois de criada uma janela com o Expert habitue-se a fazer o seguinte:

1. Renomeie o form. O nome default para o form é algo como Form1, Form2, etc. Mais adiante voce terá que se referir a esse form para ativa-lo e será bem mais fácil se cada form tiver um nome um pouco mais significativo. Para este form que acabou de ser criado, o nome *fmPedidos* é bem apropriado. Mas para renomear, é preciso antes selecioná-lo para que suas propriedades apareçam no Object Inspector. Como o Expert cobre o form com painéis é preciso um trabalho adicional para isso. Use a caixa-com-lista logo abaixo da barra de título do Object Inspector para procurar o form na lista de objetos. A seguir mude a propriedade *Name* para *fmPedidos*. Aproveite e mude a propriedade *Caption* para *Cadastro de Pedidos*.

2. Salve a unit. Cada form está definido dentro de uma unit que também é criada junto com o form. O nome default é Unit1, Unit2, Unit3, etc. Esse nome não é significativo e é aconselhável substituí-lo. Clique o botão Save file na caixa de ferramentas para ativar o diálogo Save file as, certifique-se que o diretório é o do projeto e digite Pedidos no nome do arquivo. Em seguida pressione **Ok**.

3. Renomeie *Tables* e *DataSources*. A todo momento voce fará referências às *Tables* e *DataSources* no código do programa, portanto coloque nomes que esclareçam o conteúdo dos arquivos. Use o prefixo **tbl** para *Tables* e **dts** para *DataSources*. Para o form de pedidos use os seguintes nomes:

Table1 > *tblPedidos*
DataSource1 > *dtsPedidos*
Table2 > *tblItens*
DataSource2 > *dtsItens*

Para ter certeza se *Table1* é mesmo a tabela de pedidos, olhe na propriedade *TableName*. Ali deve estar o nome que voce escolheu para o arquivo de pedidos .

4. Ative as tabelas. Depois de remonear as *Tables*, atíve-as mudando a propriedade *Active* para *True*. Isso fará com que os componentes da janela passem a mostrar um registro da tabela.

5. Inclua objetos para os campos. Cada campo nas tabelas tem também propriedades e eventos que podem ser configurados. Para acessar estas propriedades, voce precisa incluir objetos para estes campos no form. Faça isso dando um clique duplo no ícone da *Table,* o que ativa o *Field editor*. A seguir clique o botão **Add**, selecione os campos que quer incluir e pressione **Ok**. Os campos agora tem objetos no form. Para acessar suas propriedades basta clicar no nome do campo que aparece na janela do *Field editor*.

Incrementando o form

A página Data Controls na paleta de componentes oferece vários recursos para mostrar/editar dados que estão no banco de dados. Todos estes componentes tem que estar ligados a uma Table através de um DataSource. Alguns são conectados a um determinado campo do registro, e outros ao registro inteiro. Em todos eles vamos encontrar a propriedade DataSource que tem que ser preenchida pelo programador. Os componentes conectados a campos tem também a propriedade DataField, que também tem que ser preenchida.

Você já está usando alguns destes componentes na sua aplicação. As caixas de edição de campos que aparecem na janela criada pelo Database Expert vem deste conjunto de componentes, assim como a grade para mostrar os Itens no form de Pedidos.

DBEdit

É de longe o componente mais usado para mostrar campos. Oferece uma caixa de edição conectada a um campo no arquivo. Para usá-lo, basta preencher as propriedades DataSource e DataField (nessa ordem).

DBComboBox e DBListBox

Conectados a um campo, permitem associar uma lista de strings para preencher o campo. Use quando existe uma lista de valores determinados que podem ser atribuidos ao campo. Exemplo: Se o campo armazena o Estado (UF) de um endereço, poderíamos por todos os estados do Brasil em uma lista onde o usuário escolheria o estado desejado. Vamos utilizar este componente para fazer uma lista de vendedores no form de Pedidos. Faça o seguinte:

1. Delete a caixa de edição de Vendedores e coloque no lugar um *DBComboBox*.
2. Modifique sua propriedade *DataSource* para *dtsPedido*. Na propriedade *DataField* coloque *Vendedor*.
3. Procure a propriedade *Items* e dê um clique duplo sobre ela, para ativar o editor de listas de strings. Digite alguns nomes e pressione **Ok**. Está pronto.

O componente *DBListBox* funciona da mesma maneira, só que tem um "visual" diferente.

DBLookUpCombo e DBLookUpList

São componentes poderosos para tratar de referências a outros arquivos. Visualmente é igual ao *DBComboBox*, a diferença é que a lista de strings é fornecida por outro arquivo. Exemplo: poderíamos usa-los para que o usuário pudesse escolher o código do clientes em uma lista. Essa lista vem do arquivo de clientes, de maneira que podemos ficar seguros que todos os códigos na lista existem no arquivo de clientes. Só que isso não é tudo, podemos ir muito além. Que tal se o que o usuário visse na lista fosse o nome e não o código do cliente? Também é possível, e querendo podemos fazer aparecer na lista o nome **e** o código do cliente. O componente sabe que na hora de gravar no arquivo é o código e não o nome do cliente que precisa ser gravado. Então vamos implementar, antes acrescente mais uma *Table* e um *DataSource* (estão na página *Data Access*) para o arquivo de clientes e renomeie-os para *tblClientes* e *dtsClientes*, respectivamente. A seguir coloque um *DBLookUpCombo* no form e altere as seguintes propriedades:

DataSource:	dtsPedidos
DataField:	CodCli
LookupSource:	dtsClientes
LookupField:	CodCli
LookupDisplay:	RzSoc

DBNavigator

Permite navegar pelos registros de um arquivo. Para testar o uso deste componente, vamos incluir um navegador para os Itens de pedido na janela de pedidos. Antes temos que abrir espaço para ele. Clique o painel que está debaixo da grade de Itens, use a borda do painel para seleciona-lo. Mude a propriedade *Align* para *alNone* e afaste-o um pouco. Coloque mais um painel no form, com altura suficiente para caber um Navigator. Neste novo painel, mude a propriedade *Align* para *alTop*. Em seguida clique o painel sob a grade e deixe a propriedade *Align* em *alClient*. Agora temos aonde colocar o *DBNavigator*. Coloque um no novo painel como mostra a figura 5. Para conecta-lo com o arquivo de Itens, bastar setar sua propriedade *DataSource* em *dtsItens*.

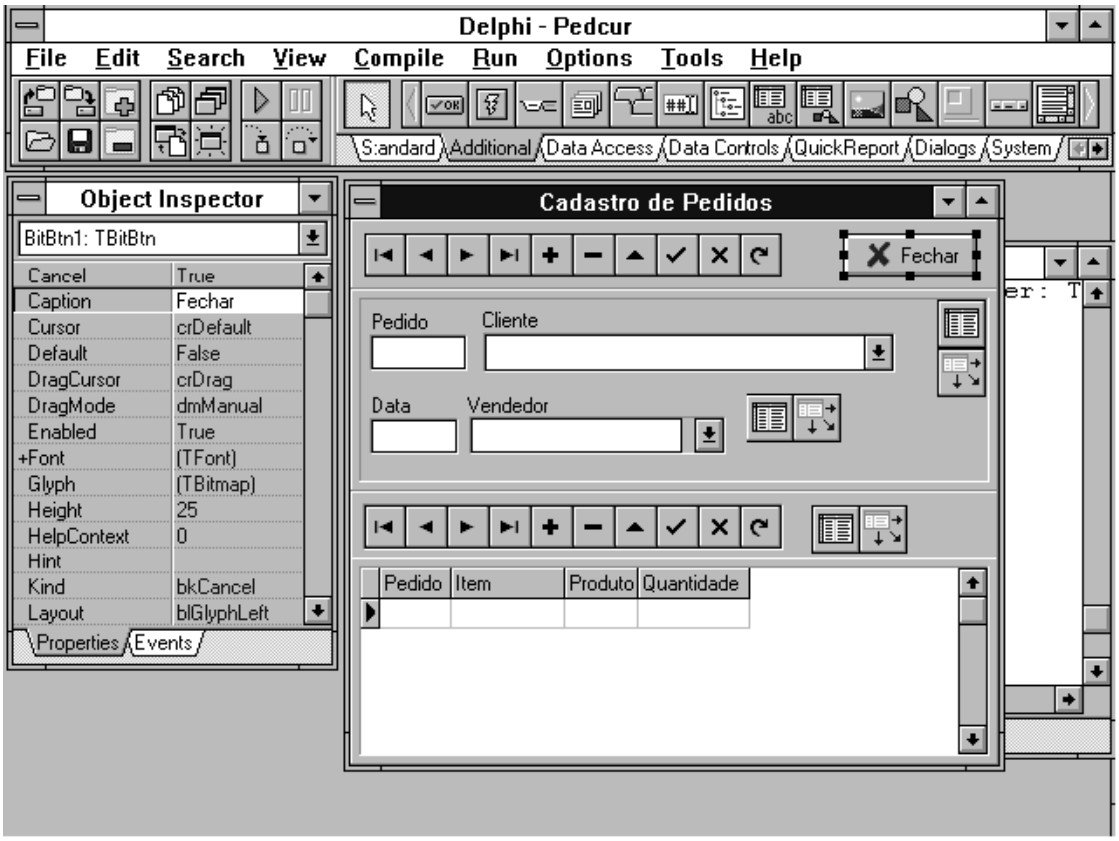

Figura 5 - Cadastro de Pedidos

DBGrid

Este componente permite mostrar/editar os registros de um arquivo na forma de tabela. A vantagem deste tipo de saída de dados é que é possivel ver vários registros ao mesmo tempo. Para testar seu uso, vamos incluir uma grade no form de cadastro de Clientes para mostrar os pedidos do cliente. Ative este form e:

1. Clique no painel onde estão os campos do registro (use a borda do painel) e modifique sua propriedade *Align* para *alTop*. Abra espaço para caber outro painel abaixo do primeiro e neste novo painel coloque a propriedade *Align* em *alClient*. Apenas para que os dois painéis fiquem com o mesmo estilo, coloque a propriedade *BevelInner* em *bvLowered* e a o valor 4 na propriedade *BorderWidth*.

2. Ponha uma *Table* e um *DataSource* no form e mude as seguintes propriedades:

Table

DatabaseName:	alPedidos (O alias que criamos para o sistema)
TableName:	Pedidos.DB (Nome do arquivo de pedidos.
Name:	tblPedidos
Active:	true
MasterSource:	dtsClientes

MasterField: Clique o botãozinho que aparece nesta propriedade para ativar *Field Link Editor*. Na caixa *Available Indexes* selecione *CodCli*. Clique em *CodCli* nas listas *Detail Fields* e *Master Fields*. Pressione o botão *Add* e em seguida o botão **OK**.

DataSource

Name: dstPedidos

Dataset: tblClientes

3. Pegue um DBGrid na página Data Controls e coloque-o no novo painel. Mude a propriedade Align da grade para alClient. Na propriedade DataSource ponha dtsPedidos. Está pronto.

7. O QuickReport

O Delphi 1.0 vem com um gerador de relatórios chamado *Report Smith*. É um gerador muito poderoso mas tem alguns problemas. Primeiro, não é possível gerar relatórios dentro do ambiente do Delphi, é preciso ativar o Report Smith, criar o relatório, salva-lo e depois inclui-lo no projeto. Além disso, no momento que o usuário vai imprimir o relatório, o Report Smith precisa ser carregado, o que implica em instalar o Report Smith no computador do cliente. Em segundo lugar, é um sistema lento e pesado, se o cliente não tem um computador extremamente rápido a carga do Report Smith pode ser bastante demorada.

O *QuickReport* é um gerador de relatórios para o Delphi desenvolvido pela *QSD - Quick Software Development*. Vem na forma de componentes que podem ser instalados na paleta de componentes. Isso permite que o relatório seja criado através de forms e componentes, sem sair do Delphi. É um software bastante poderoso e resolve a maior parte das necessidades de relatórios, além disso é bastante fácil de usar e não exige que seja feita nenhuma instalação adicional no computador do cliente. Fez tanto sucesso entre os usuários do Delphi que a Borland resolveu incluí-lo no pacote do Delphi 2.0. Se voce tem a versão 1.0 do Delphi, então é preciso instalar o pacote. A versão shareware do QuickReport pode ser obtida gratuitamente na Internet acessando a página da QSD ou alguma página especializada em componentes para Delphi (existem muitas, veja na Bibliografia), mas voce está recebendo a última versão (1.0d) no disquete do curso.

Instalação

Para instalar o QuickReport, siga os seguintes passos:

1. Dentro do diretório do Delphi, crie um subdiretorio chamado *Qrep*, nele descompacte o arquivo *QRep10d.Zip* (use o pkunzip v2.0 ou maior).
2. Ative o Delphi. Se houver um projeto aberto, feche-o com a opção *File/Close Project* do menú.
3. Selecione a opção *Options/Install Components* no menú.
4. No diálogo que apareceu, clique o botão *Add*.
5. No diálogo *Add Module* clique o botão *Browse* e navegue até o diretório Qrep que acabamos de criar. Selecione arquivos do tipo *.DCU e clique no arquivo *QuickRep.Dcu*. Em seguida pressione o botão *Ok*.
6. De volta ao diálogo *Install Components*, clique *Ok*. O Delphi vai agora recompilar sua biblioteca de componentes e instalar o QuickReport na paleta.
7. Se a página *Quick Report* não apareceu na tela, clique na seta à direita da paleta para ver as paginas que não cabem na tela.

Criando relatórios

Agora que voce tem o QuickReport instalado, vamos utiliza-lo para criar uma listagem simples de clientes.

1. Abra o projeto do Sistema de Pedidos.

2. Clique o botão *New Form* na barra de ferramentas e selecione um form em branco.

3. Pegue o componente *QuickReport* (o primeiro) e coloque-o no form. Note que o form ganha barras de rolagem.

4. Coloque um componente *QRBand* (o segundo) no form. A faixa branca que aparece é o que o QuickReport chama de Banda (*Band*). Mude a propriedade *Name* para *bnTitulo.*

5. Coloque um componente *QRSysData* e ponha na banda. Altere as seguintes propriedades:

Alignment:	taCenter
AlignToBand:	True
Data	qrsReportTitle

6. Ponha no form uma *Table* e um *DataSource* para o arquivo de clientes e ajuste as propriedades:

Table

DatabaseName:	alPedidos
TableName:	CLIENTES.DB
Name:	tblClientes

DataSource

DataSet:	tblClientes
Name:	dtsClientes

7. Clique o ícone do *QuickReport* e altere a propriedade *DataSource* para *dtsClientes* e a propriedade *ReportTitle* para *Listagem de Clientes*.

8. Coloque mais uma banda no form e mude seu nome para *bnTitCol*. Altere a propriedade *BandType* para *rbColumnHeader*.

9. Na nova banda, ponha 4 componentes *QRLabel* alinhados lado a lado e mude a propriedade *Caption* para *Código, Razão Social, Cidade* e *UF* respectivamente. Deixe entre eles espaço suficiente para caber os valores dos campos.

10. Coloque outra banda no form e mude seu nome para *bnDetalhe*. Altere a propriedade *BandType* para *rbDetail*.

11. Na banda *bnDetalhe*, ponha 4 componentes *QRDBText* alinhados com os componentes da banda anterior. Em todos eles coloque a propriedade *DataSource* em *dstClientes*. A propriedade *DataField* fica: *CodCli, RzSoc, Cidade* e *UF* respectivamente.

12. Salve a unit com o nome *ListCli.Pas*. Use a opção *File/Save File*. Dê um clique duplo sobre o ícone do QuickReport para ver o resultado.

13. Agora só faltar ligar o form à janela principal. Ative a janela principal do sistema e clique no menú a opção *Cadastro/Clientes/Listagem*. Na janela de código escreva:

```
fmListPed.QuickReport1.Preview;
```

Criando um Relatório Mestre-Detalhe

Um relatório Mestre-Detalhe usa dados de dois ou mais arquivos relacionados. O exemplo que usamos antes é o de Pedidos e Ítens de pedidos, onde o arquivo de pedidos é Mestre e o de Ítens de Pedidos é Detalhe. Mas essa relação existe também entre o arquivo de Clientes e o de Pedidos, só que agora o de Pedidos é Detalhe e o de Clientes é Mestre.

Vamos aproveitar o relatório que acabamos de criar e incrementá-lo para implementar uma relação Mestre-Detalhe. A idéia é fazer o relatório mostrar os pedidos de cada cliente:

1. Coloque uma Table e um DataSource no form para o arquivo de Pedidos. Ajuste suas propriedades como indicado no ítem 2 da página 24.

2. Ponha mais uma banda no form e remomeie para bnSubDetalhe. Ajuste a propriedade BandType para rbSubDetail.

3. Coloque um componente QRDetailLink no form e mude as seguintes propriedades:

DataSource:	dtsPedidos
DetailBand:	bnSubDetalhe
Master:	QuickReport1

4. Na banda bnSubDetalhe coloque 3 componentes QRDBText, ajuste a propriedade DataSource de todos para dtsPedidos. A propriedade DatadField fica: CodPed, Data e Vendedor, respectivamente.

5. Está pronto. Dê um duplo clique no ícone do QuickReport para ver o resultado.

8. Blibliografia

Livros

Manuais do Delphi 1.0

Delphi Segredos e Soluções
Gary Cornnel / Troy Strain
Editora Makron Books

Teach Yourself Database Programming With Delphi in 21 Days.
Nathan and Ori Gurewich
Borland Press

Internet

Quick Soft Development AS (Quick Report)
http://www.qsd.no

Borland Delphi
http://www.borland.com/Product/Lang/Delphi/Delphi.html

Borland Delphi Components Page
http://www.neosoft.com/~startech/delphi

Delphi Super Page
http://sunsite.icm.edu.pl/~robert/delphi

The Delphi Station
http://www.teleport.com/~cwhite/delphi.shtml

The Delphi Hackers' Corner
http://www.it.kth.se/~ao/DHC

Delphi Components
http://www.ucinet.com/~softec/comp1.html

Delphi Freeware Components
http://www.vyatka.su/Peoples/ws/html/delfc.htm

Arquivos no Disquete

No disquete do curso voce está recebendo os seguintes arquivos:

QRep10d.Zip - Versão shareware do QuickReport
QRDoc10d.Zip - Documentação do QuickReport em formato de Word

Quick Soft Development AS

lado a lado e mude a propriedade Caption para Código, Razão Social, Cidade e UF respectivamente. Deixe entre eles espaço suficiente para caber os valores dos campos.

Roteiro rápido para iniciar um projeto em Delphi

1. **Preparação**
   - **Antes de programar planeje: É essencial ter um modelo lógico do sistema antes de começar a programar. O DER é uma ótima ferramenta para mostrar as relações entre os arquivos do sistema.**
   - **Crie um novo diretório para o projeto.**
   - **Ative o Database Desktop e dentro dele:**
     - **Crie um novo aliás para o projeto.**
     - **Ajuste o diretório de trabalho do Database para o novo diretório criado.**
     - **Crie as estruturas dos arquivos.**
     - **Defina as chaves, índices e as integridades referenciais.**
     - **Salve os arquivos no diretório de trabalho**
2. **Etapas iniciais**
   - **Agora você está pronto para começar: Ative o Delphi e**
     - **Você pode usar o form em branco que o Delphi apresenta quando é ativado. Comece pelo menú. Selecione o primeiro componente na seção Standard da paleta de componentes e em seguida clique na área útil do form.**

- **No form, aparece o ícone do menú. Use um clique duplo para ativar a janela de edição do menú e coloque as opções desejadas. Depois feche a janela de edição.**
- **O form agora apresenta a barra de menú. Para definir os comandos que devem ser executados nas opções do menú, basta dar um clique na opção que o Delphi já abre a janela dos fontes dentro da função correta.**
- **Como teste, dê um clique na opção Sair do menú e escreva** Close; **no corpo da função. Sua aplicação agora tem como fechar.**
- **Neste ponto é uma boa idéia salvar o projeto.**

- **Crie um form para cada cadastro do sistema usando o Database Form Expert:**
  - **Clique o ícone "New Form" na barra de ferramentas.**
  - **Escolha a página "Experts".**
  - **Selecione o "Database form" e clique "OK".**
  - **Responda às perguntas do assistente.**
  - **Renomeie e salve o form.**
- **Para linkar o novo form ao form principal:**
  - **Dê um clique na opção do menú que deve ativar o form**
  - **Escreva** NomeForm.Show; **no corpo da função.**
  - **Vá até o topo do arquivo e procure a cláusula** Uses**. No final da lista de units, acrescente o nome da unit do novo form.**

- Obtendo ajuda on-line
- Labels
- Caixas de edição

Aula 2

- Ativando outras Janelas
  - Forms 'modless'
  - Forms modais
  - Diálogos
    - Diálogos do sistema
    - Mensagens ao usuário
- Outros componentes
  - Caixa de checagem
  - Painel de grupo
  - Caixa de RadioButtons
  - Caixa de lista
  - Caixa de lista combinada - ComboBox
  - Memos

    - Imagens
- Criação de um diálogo para busca de arquivo de imagens

Aula 3

- Aplicação prática - Editor de arquivos texto (NotePad)
- Uso do Clipboard
- Tratamento de erros - Exceções
- Economizando recursos - Criação dinâmica de forms