

Visual Basic



(Versão 5)

Curso Básico

Agradecimentos

Embora tenha custado horas e horas de montagem (word) essa apostila não teria sido confeccionada se eu não tivesse encontrado a "Home Page" do **José Carlos Macoratti** – *Visual Basic Banco de dados*.

Excelente trabalho de garimpo deste profissional que com certeza prova que podemos encontrar vários serviços de *boa qualidade e de graça* na Internet.

Por favor se quiserem agradecer visitem este site.

<http://www.geocities.com/SiliconValley/Bay/3994/>

macoratti@riopreto.com.br

Trabalhos de edição e montagem:

Carlos Alberto Nunes Susviela

Santana do Livramento – RS – Brasil

susviela@zaz.com.br

<http://www.geocities.com/SiliconValley/Haven/2828>

Índice:

1) INTRODUÇÃO (Conceitos Básicos)	6
1.1) BANCO DE DADOS.....	6
1.2) TRABALHANDO COM TABELAS	8
1.3 - CRIANDO BANCO DE DADOS, TABELAS, ÍNDICES, RELACIONAMENTOS E CHAVES PRIMÁRIAS. (DATA MANAGER) -	10
1.4. RECORDSET: TABLE, DYNASETS, SNAPSHOTS.(DEFINIÇÃO, PRINCIPAIS MÉTODOS E EVENTOS RELACIONADOS)	14
1.5. DATA CONTROL. (DEFINIÇÃO, PRINCIPAIS MÉTODOS E EVENTOS RELACIONADOS).....	18
1.6. EVENTOS: UM NOVO ENFOQUE (EVENTOS: CONCEITOS, EXEMPLOS .).....	21
1.7. O PODER DA SQL (A LINGUAGEM SQL : INTRODUÇÃO.)	22
1.8. USANDO EDIT, ADDNEW E UPDATE...(EDITANDO, INCLUINDO E ATUALIZANDO...)	23
1.9. VISUAL BASIC - DATAS E O ANO 2000.(COMO O VB INTERPRETA O ANO COM DOIS DÍGITOS.(DD/MM/YY)	24
1.10. SQL : CRIANDO CONSULTAS PARAMETRIZADAS NO VISUAL BASIC.	26
2) TRABALHANDO COM SQL (Conceitos Básicos)	29
2.1 Iniciando com a SQL.....	29
2.2 Criando tabelas via SQL.....	30
2.3 Índices e Tabelas - Criar, Alterar e Excluir.	31
2.4 Testando As Instruções Sql	32
3) TRABALHANDO COM REDES (Conceitos Básicos)	35
3.1 Introdução.....	35
3.2 Acesso aos Dados	35
3.3 Bloqueio de Registros.....	37
4) CONTROL DATA (Projeto Comentado)	39
4.1 Acesso a Base de Dados - Data Control.....	39
1) Definição do banco de dados e tabelas.....	39
2) Desenhar a interface da aplicação com o usuário.....	40
3) Implementar o código da aplicação.....	42
1-Código para botão incluir dados:	43
2-Código do botão gravar dados:	43
3-Código do botão Cancelar a inclusão de dados:	44
4-Código para o botão Excluir dados:	44
5-Código para o botão Localizar dados:	45
6-Código do evento Reposition:	45
7-Código do evento Validate:	45
8-Código para tratamento de erros	46
5) D.A.O (Projeto comentado)	47
Estrutura da tabela.	47
Interface com o usuário	47

Codigo da Aplicacao.	49
6) SQL.....	56
Estrutura da tabela.	56
Interface com o usuário57	57
Codigo da Aplicacao59	59
7) CRYSTAL REPORTS (Projeto Comentado).....	65
Introdução:Gerando os seus relatórios com o Crystal Reports.....65	65
Criando um novo relatório.....65	65
Agrupando e ordenando registros.....67	67
Inserindo títulos e Legendas.67	67
Formatação de campos, campos especiais e desenho de linhas e caixas.67	67
Trabalhando com fórmulas.68	68
Determinando o estilo e inserindo uma figura em seu relatório.69	69
Imprimindo o relatório a partir de sua aplicação no Visual Basic.70	70
Relatório com dados de várias tabelas e Seleção de registros71	71
8) ACESSO A BASE DE DADOS PADRÃO XBASE(DBASE/CLIPPER)	81
Acesso utilizando o Controle de Dados Vinculados. (Data Control)81	81
Acesso utilizando a DAO. (Data Access Object)85	85
9) SETUP WIZARD.....	92
10) DBGRID.....	99
Introdução.....99	99
Utilização e Configuração.99	99
Controle Financeiro - Definição de tabelas.102	102
Controle Financeiro - Interface com o usuário.104	104
Controle Financeiro - Código do projeto.....108	108
DbGrid - Dicas e Truques.....113	113
11) DBLIST.....	114
12) REDES (Projeto comentado)	119
- <u>Acesso Exclusivo/Compartilhado</u>119	119
- <u>Acesso as tabelas - dbDenyRead, dbDenyWrite e dbReadOnly</u>121	121
13) ERROS.....	126
Introdução.....126	126
Interceptando erros.126	126
Identificando o Tipo de Erro.127	127
Finalizando um tratamento de erros.130	130
Principais erros relacionados a Banco de dados.131	131
14) OLE.....	133
Introdução.....133	133
Instrução do WordBasic	135

Instrução do VBA 97	135
Usando Automação OLE com o Word - Exemplo Prático.	135
Usando Automação OLE com o Excel - Exemplo Prático.	141
15) GRÁFICOS NO VB	143
Introdução.	143
Utilização e Configuração.	144
16) OPÇÕES DE BUSCA RÁPIDA (Dicas)	150
17) HOT SPOTS NO VISUAL BASIC (Dicas)	152
18) FORMULÁRIOS COM SUBFORMULÁRIOS NO VB ? (Dicas)	154
19) CRIANDO UM SCREEN SAVER NO VB (Dicas)	158
20) ROTINA PARA VALIDAR CPF/CGC NO VISUAL BASIC (Dicas)	160
21) DBGRID COM LISTBOX - PERSONALIZE O SEU DBGRID: CAIXA DE LISTAGEM, TOTALIZANDO POR COLUNAS E CONTADOR DE REGISTROS. (Dicas)	163
<u>Personalizando e incrementando o DBGrid.</u>	163
Interface Visual do Projeto.	163
<u>Código do Projeto.</u>	164

1) INTRODUÇÃO (Conceitos Básicos)

1.1) BANCO DE DADOS

Podemos entender por banco de dados qualquer sistema que reuna e mantenha organizada uma série de informações relacionadas a um determinado assunto em uma determinada ordem.

A lista telefônica é um exemplo, nela percebemos que todos os dados referentes a uma pessoa estão na mesma linha, a isso chamamos registros..

O tipo ou categoria da informação (nome, telefone, etc.) sobre uma pessoa está separada em colunas, as quais chamamos campos..

Um Sistema Gerenciador de banco de dados relacionais(SGBDR) é usado para armazenar as informações de uma forma que permita às pessoas examiná-las de diversas maneiras.

O Gerenciador relacional de bancos de dados do Visual Basic e do Access é o Microsoft Jet, ele pertence a uma categoria diferente dos Gerenciadores tradicionais, como o Dbase e o Paradox, pois possui características em comum com os banco de dados cliente-servidor. Tais características comuns são:

- *Todas as tabelas, índices, consultas, relatórios e código são armazenados num único arquivo .MDB*
- *Os campos de data suportam informação de hora.*
- *Admite o armazenamento permanente de comandos **SQL***
- *é possível forçar a integridade referencial entre as tabelas.*
- *Os campos suportam valores nulos (**Null**)*

No Dbase/Clipper banco de dados significa um arquivo que contém a estrutura de dados(campos) e os dados (Arquivo padrão DBF). Para o padrão MDB este conjunto de dados e sua estrutura denomina-se Tabela.

Portanto aquilo que o Dbase/Clipper considera um banco de dados o Access e o Visual Basic considera como uma Tabela .

Para o Access e o Visual Basic todos os componentes do sistema estão em um único arquivo com extensão MDB, a este "pacote" consideramos o banco de dados.

Logo quando abrimos um arquivo MDB temos acesso a todos os componentes do sistema : tabelas, consultas, macros, relatórios, etc. A esses componentes chamamos objetos do sistema e em resumo podemos descrevê-los a seguir:

Tabelas	Onde armazenamos as informações que queremos tratar.
Consultas	Filtram as informações das tabelas e permitem sua visualização. Geralmente são comandos SQL.
Formulários	São janelas destinadas à edição e visualização dos dados.
Relatórios	Organizam os dados de tabelas e consultas de uma maneira que possam ser impressos
Macros	Rotinas que automatizam determinadas tarefas sem necessidade de programação. (Utilizadas no Access)
Módulos	Armazenam instruções e comandos da linguagem Access Basic/VBA e permitem melhorar e expandir os recursos do sistema.

Obs:

Embora o Visual Basic utilize arquivos padrão MDB; Formulários, Relatórios e Módulos são tratados de forma diferente pelo próprio Visual Basic e, nativamente, o Visual Basic não utiliza Macros.

Além disso no Access e Visual Basic podemos utilizar outros arquivos além dos arquivos MDB; como arquivos DBF do Dbase/Clipper, arquivos do Paradox, do Btrieve, etc.

Os recursos de definição de dados do mecanismo Jet permitem a criação, a modificação e a exclusão de tabelas, índices e consultas. O Jet também aceita a validação de dados em nível de campo e registro. A integridade de dados tem suporte sob a forma de chaves primárias e integridade referencial entre tabelas.

Para manipulação de dados, o Jet admite o uso da SQL e de objetos de acesso aos dados. Esses objetos permitem ao programador manipular informações contidas no banco de dados, através da definição das propriedades dos objetos e pela execução dos métodos associados aos objetos. A tabela abaixo relaciona esses objetos e descreve resumidamente suas funções:

<i>Objeto</i>	<i>Descrição</i>
DBEngine	O objeto que referencia o mecanismo de bancos de dados do Microsoft Jet
Workspace	Um área na qual o usuário pode trabalhar com os bancos de dados
Database	Uma coleção de informações organizadas em tabelas, juntamente com informações a respeito de índices e relações sobre as tabelas
TableDef	Uma definição da estrutura física de uma tabela de dados
QueryDef	Uma consulta armazenada de SQL das informações contidas no banco de dados.
Recordset	Uma coleção de registros de informações sobre um único tópico
Field	Uma única unidade de informações em um banco de dados
Index	Uma lista ordenada de registros em um recordset, baseada em um campo chave definido
Relation	Informações armazenadas a respeito do relacionamento entre duas tabelas

1.2) TRABALHANDO COM TABELAS

As tabelas são o coração dos bancos de dados, e, uma das tarefas fundamentais que você deverá fazer será organizar os dados em tabelas. Para criar uma tabela no Visual Basic você pode usar o Data Manager, mas, uma maneira mais fácil, se você tiver o Access, será utilizar os seus recursos para este fim. Aqui descreveremos como criar uma tabela no Access sem a ajuda de um assistente. Basicamente você irá dar nome aos campos (colunas), e definir o tipo de dados para estes campos. Isto depende da informação que deseja armazenar no campo; se for armazenar um dado que não fará parte de cálculos, um nome por exemplo, o tipo pode ser **texto**, se for armazenar valores numéricos que farão parte de cálculos o tipo será **numérico(inteiro, simples...)**, se for armazenar datas o tipo será **data** e assim por diante.

O tamanho do campo define a quantidade de informação relacionada ao item que você pode armazenar, assim : para um campo do tipo texto com tamanho 30 você poderá armazenar textos com no máximo 30 caracteres, um campo numérico de tamanho 2 armazena números com até dois dígitos.

Vejamos como exemplo a estrutura de uma tabela que chamaremos Clientes. Como o nome indica ela deverá guardar informações sobre os clientes, decidimos que tais informações serão: Código, Nome, Endereço, Cep e idade:

Exemplo de Tabela Clientes

Nome do Campo	Tipo do Dado armazenado	Comprimento para o campo
Codigo	Numerico	3
Nome	Texto	40
Endereco	Texto	40
Cep	Texto	8
Idade	Numerico	3

Embora não exista nenhuma regra absoluta para os dados que devem ser colocados em cada tabela, damos abaixo diretrizes gerais para um projeto de banco de dados eficiente:

- Determine um tópico para cada tabela e certifique-se de que todos os dados contidos na tabela estão relacionados com o tópico.
- Se uma série de registros em uma tabela apresenta campos intencionalmente deixados em branco, divida a tabela em duas tabelas similares.
- Se as informações se repetem em vários registros, desloque essas informações para outra tabela e defina um relacionamento entre elas.
- Campos repetidos indicam a necessidade de uma tabela secundária.
- Use tabelas de pesquisa para reduzir o volume de dados e aumentar a precisão da entrada de dados.
- Não armazene informações em uma tabela se elas puderem ser calculadas a partir dos dados contidos em outras tabelas.

Para cada linha que você adiciona a sua tabela você tem um registro. Feito isto é importante você definir os **relacionamentos** entre as tabelas de seu banco de dados. Por exemplo se você criou duas tabelas, uma para os clientes e outra para os pedidos feitos por esses clientes, voce poderá relacionar as duas tabelas por meio de um campo código do cliente comum às duas tabelas.

Criando Tabelas no Access (No Visual Basic use o **Data Manager**)

É possível criar uma tabela com um Assistente de Tabela, mas se ele não oferecer o tipo de tabela desejada, pode-se criar a tabela sem ajuda. Mesmo que não se utilize o Assistente de Tabela pode-se usar o Construtor de Campos para selecionar campos individuais das tabelas de amostra do Assistente de Tabela.

Para criar uma tabela sem um Assistente (Access versão 2.0)

- 1-Na janela Banco de Dados, clique no botão "Tabela" e, então, escolha o botão "Novo".
- 2-Na caixa "Nova Tabela", clique no botão "Nova tabela".

O Microsoft Access exibe a janela Tabela no modo Estrutura, no qual são definidos os campos da tabela.

Para definir campos na tabela

- 1 - Na coluna "Nome do Campo" digite o nome do primeiro campo seguindo as convenções de nomenclatura padrão do Microsoft Access. -Ou- Clique no botão "Construir" na barra de ferramentas e selecione o campo a partir do Construtor de Campos. O tipo de dados e outras propriedades para cada campo já estão definidas, embora possam ser alterados caso desejado.
- 2 - Na coluna "Tipo de Dados", mantenha o padrão (Texto) ou clique na seta e selecione o tipo de dados desejado.
- 3 - Na coluna "Descrição" digite a descrição da informação que este campo irá conter. A descrição é opcional.
- 4 - Se desejar, defina as propriedades do campo na parte inferior da janela.
- 5 - Repita os passos de 1 a 4 para cada campo.

Para salvar a tabela

- 1 - Após terminar de definir os campos clique no botão "Salvar" na barra de ferramentas ou escolha o comando Salvar no menu Arquivo para salvar a estrutura da tabela. O Microsoft Access, então, envia uma mensagem pedindo para nomear a tabela.
- 2 - Digite um nome para a tabela conforme as convenções de nomenclatura do Microsoft Access e, então, escolha OK.

Observações

É recomendável que se designe o campo de (*) **chave primária** em qualquer tabela. Se isto não for feito o Microsoft Access pergunta ao usuário se este deseja que o programa crie uma chave primária quando salvar a tabela pela primeira vez. Adicionalmente à definição das propriedades de campo, pode-se, também, definir as propriedades da tabela. As propriedades da tabela são atributos de toda a tabela, preferencialmente a campos individuais apenas.

Quando uma coluna ou caixa de propriedades é clicada, o Microsoft Access exibe dicas úteis acerca de cada coluna ou propriedade na porção inferior direita da janela.

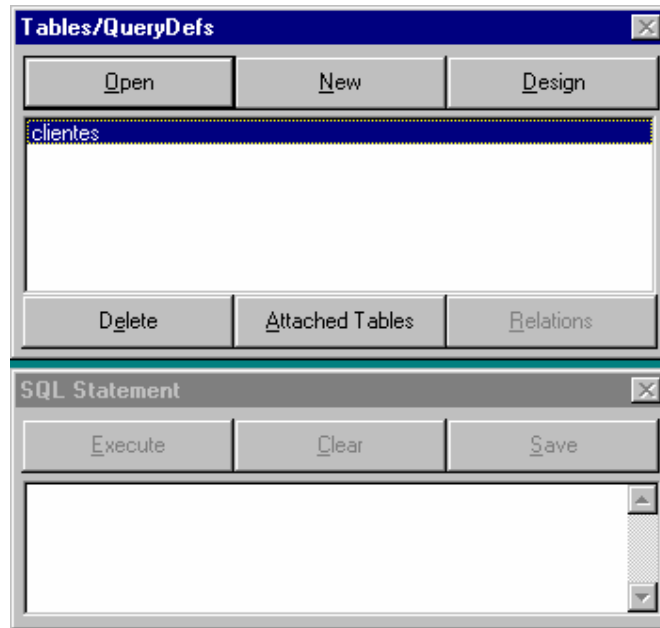
Após criar tabelas, pode-se criar consultas, formulários, relatórios e outros objetos de banco de dados que auxiliam o usuário a manipular seus dados.

(*) - *A chave primária é um campo ou combinação de campos que permite a identificação única de cada registro de uma tabela. Assim como o índice principal para a tabela, ela é utilizada para associar dados entre tabelas. Se uma tabela não inclui um campo de chave primária óbvio, o Microsoft Access pode criar um campo que designa um número único para cada registro.*

1.3 - CRIANDO BANCO DE DADOS, TABELAS, ÍNDICES, RELACIONAMENTOS E CHAVES PRIMÁRIAS. (DATA MANAGER) -

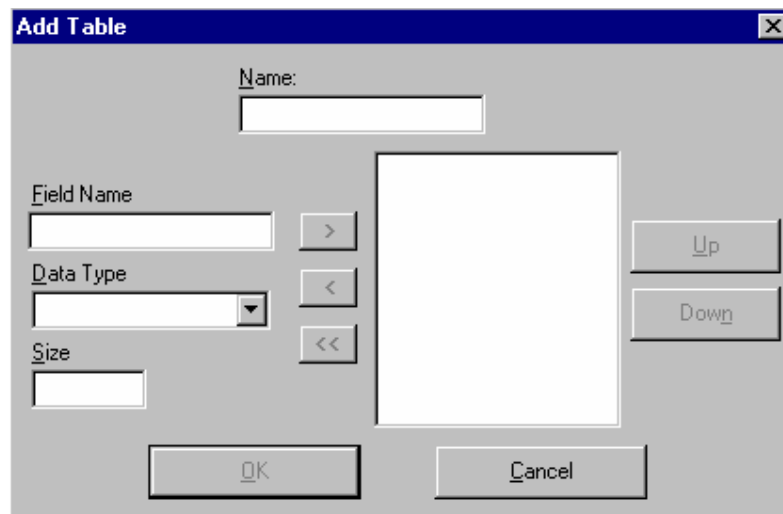
- **Criando um banco de dados.**

Para criar um banco de dados com o Data Manager, execute o data manager e selecione a opção **File/New Database** no menu principal. Um diálogo no padrão **Salvar Como** será aberto solicitando o nome do arquivo; informe o nome do banco de dados que deseja criar, por exemplo, **Controle**, a janela **Tables/QueryDefs** da figura surge, seu banco de dados já esta criado e você esta pronto para criar suas tabelas.



- **Criando uma Tabela com o Data Manager.**

Para criar uma tabela com o Data Manager, proceda da seguinte forma, clique no botão **New** da janela **Tables/QueryDefs** e, no diálogo **Add Table** mostrado abaixo (figura abaixo) siga os passos indica-



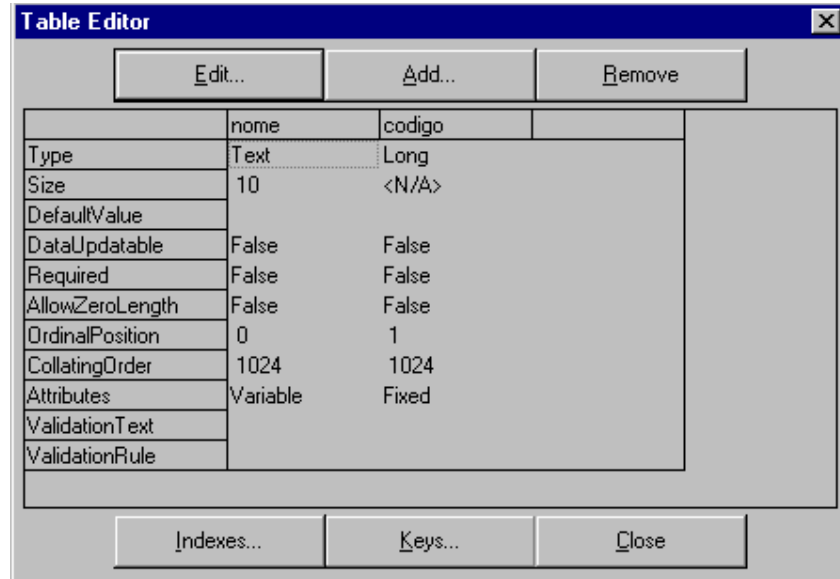
dos:

1. Informe o nome da tabela na caixa de texto Name.
2. Informe o nome do campo na caixa de texto Field Name.
3. Selecione o tipo de dado apropriado na caixa de combinação Data Type.

4. Digite o tamanho do campo, no caso do tipo Text, na caixa de texto Size.
5. Clique no botão > para incluir o campo na tabela.
6. Repita os passos 2 a 5 até que todos os campos da tabela estejam definidos e clique no botão OK para criá-la.

• **Alterando a Estrutura de uma Tabela.**

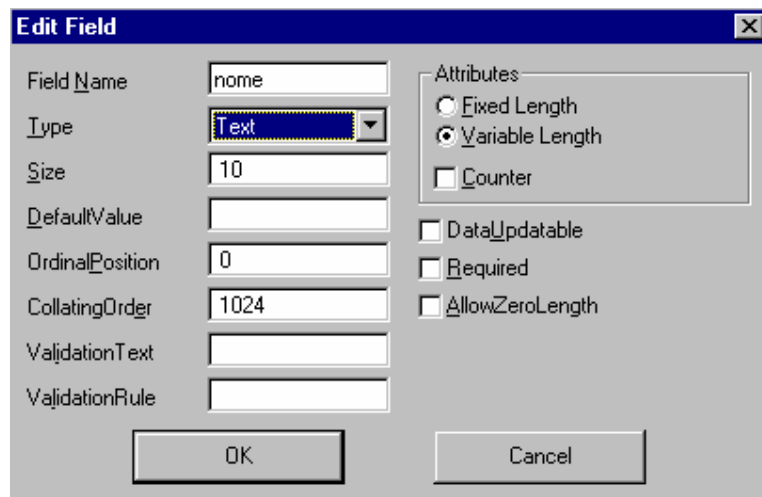
Para definir propriedades não obrigatórias de uma tabela, clique no botão **Design** da janela Tables/QueryDefs. O Data Manager abrirá a janela Table Editor, mostrada na abaixo:



Na janela Table Editor, você pode:

- a) Editar as propriedades de qualquer campo selecionado, neste caso, os dados existentes serão perdidos.
- b) Adicionar e remover campos.
- c) Criar, remover e alterar índices.

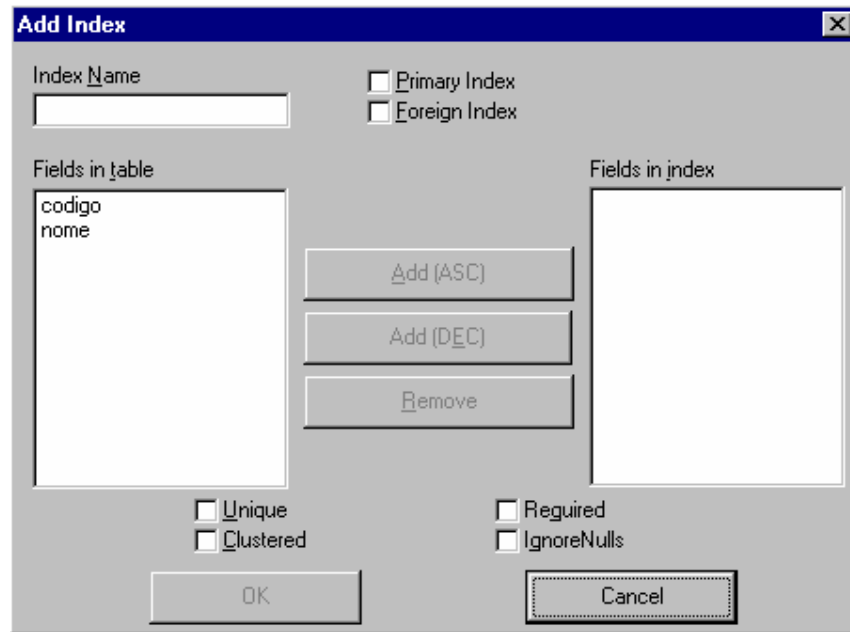
Para a definir as propriedades opcionais dos campos de uma tabela, selecione o campo e clique no botão **Edit** da janela **Table Editor**. O Data Manager abrirá a janela da figura abaixo, agora basta alterar as propriedades desejadas para o campo editado.



Obs: O Data Manager não admite a edição de um campo componente de uma expressão de índice ou de um relacionamento. Nestes casos, é obrigatória a remoção do índice ou do relacionamento.

- **Criando e Editando Índices**

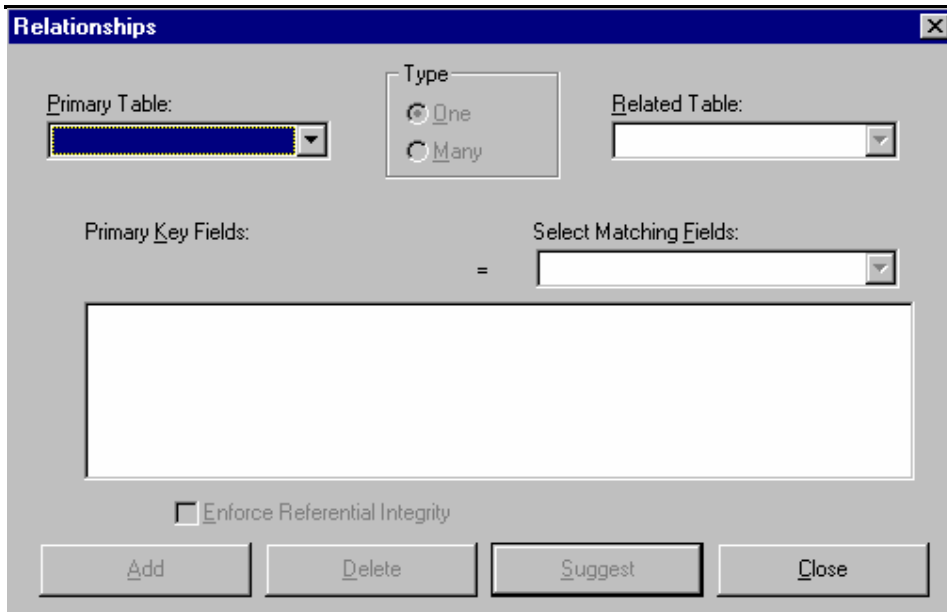
A criação de um índice de uma tabela no Data Manager é simples. Basta selecionar a janela **Table Editor**, clicar no botão **Indexes**, e, seguir os passos indicados:



1. Informar o nome do índice na caixa de texto *Index Name*.
2. Definir as propriedades do índice, marcando as caixas de verificação apropriadas.
3. Selecionar os campos componentes na caixa de listagem apropriada.
4. Selecionar a ordem do índice, se ascendente ou descendente, clicando no botão de comando indicado.

- **Definindo Relacionamentos**

Para visualizar ou definir relacionamentos entre os campos das tabelas de um banco de dados, clique no botão **Relations** na janela Tables/QueryDefse siga os passos indicados:

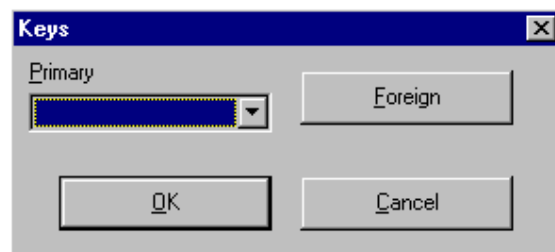


1. Selecione a tabela primária na caixa de combinação **Primary Table**.
2. Selecione a tabela dependente na caixa de combinação **Related Table**.
3. Defina o tipo de relacionamento, selecionando o botão de opção apropriado na caixa de grupo **Type**.
4. Selecione a chave externa na caixa de combinação **Select Matching Fields**.
5. Marque, se for o caso, a caixa de verificação indicando se a integridade referencial dos dados deve ser garantida.

Obs: Para definir um relacionamento, é indispensável que o campo origem seja a chave primária da tabela primária. Além disso o campo definido como chave externa deve ser do mesmo tipo de dado da chave primária.

- **Definindo uma Chave Primária**

Para definir uma chave primária, selecione o botão **Keys**(janela **Table Editor**) e o Data Manager abre o diálogo mostrado na figura abaixo, onde a chave pode ser definida clicando-se e selecionando-a da lista **Primary**.



1.4. RECORDSET: TABLE, DYNASETS, SNAPSHOTS.(DEFINIÇÃO, PRINCIPAIS MÉTODOS E EVENTOS RELACIONADOS)

- **Como abrir um banco de dados.**

A primeira coisa a fazer para armazenar e/ou acessar dados é criar um banco de dados. Se já tiver um banco de dados, você tem de abri-lo antes do seu programa poder usá-lo.

Os objetos de acesso a dados são organizados em uma hierarquia (fig. 1- Hierarquia simplificada).

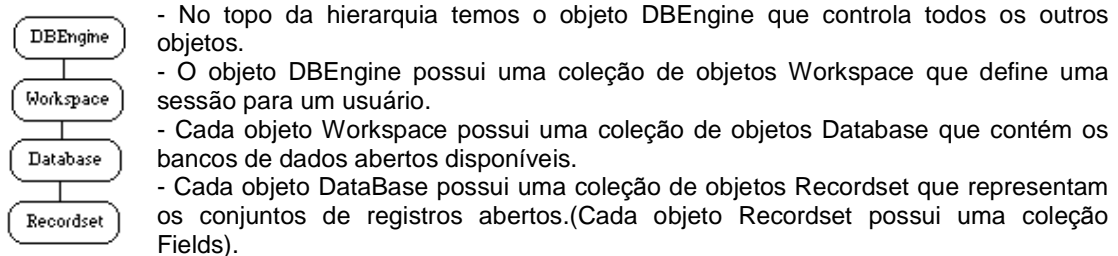


Fig-1

A abertura de um banco de dados é feita com o uso do método `OpenDatabase` do objeto `Workspace`, mostrado no código abaixo :

```
Dim meubd as Database
Set meubd = DBEngine.Workspaces(0).OpenDatabase("C:\meudir\dados.mdb")
```

Para trabalhar com os objetos de acesso a dados você usa as variáveis do objeto para cada tipo de objeto (`Database`, `Recordset`,...). Para designar um objeto a uma variável do objeto utilize a instrução `SET`.

No código acima primeiro definimos uma variável objeto de banco de dados(`meubd`), e a seguir usamos o método **`OpenDatabase`** para designar nosso banco de dados, nome e localização, à variável objeto declarada.

Após abrir o banco de dados você deve definir um **`Recordset`** para ter acesso aos dados contidos no banco de dados.

- **Recordsets**

O novo objeto **`Recordset`** é o recurso mais importante relacionado com os objetos de acesso a dados. Ele ocupa o lugar dos objetos *tabela*, *dynaset* e *snapshot* das versões anteriores. Agora ao invés de abrir uma *tabela* ou criar um *dynaset* você abre um **`Recordset`**. Há três tipos de **`Recordsets`** disponíveis:

- **Tabelas** - são as estruturas físicas que contém os dados em um banco de dados.
- **Dynaset** - são um conjunto de ponteiros para acesso a campos e registros localizados em uma ou mais tabelas.
- **Snapshot**- são uma cópia somente de leitura dos dados de uma ou mais tabelas.

- **Tabelas: uso, vantagens/desvantagens, abertura.**

Como todos os dados de um banco de dados estão armazenados em tabelas, acessar tabelas significa acessar diretamente os dados de um arquivo.

Vantagens do uso de Tabelas

- É a única forma de `Recordset` que aceita o uso de *índices*, tornado desta forma a pesquisa em uma tabela mais rápida do que em um **`dynaset`** ou **`snapshot`**.
- As alterações efetuados pelos usuários ficam imediatamente disponíveis, não sendo necessário dar o refresh para atualizar os dados.

Desvantagens do uso de Tabelas

- Não se pode utilizar filtros para restringir os registros usando determinados critérios.

- O comando **Seek** só encontra o primeiro registro que atende a seus critérios.

Abertura de uma tabela para utilização

Para abrir uma tabela, defina um objeto **Recordset** e use o método **OpenRecordset** para acessar a tabela.

Informe a constante **DbOpenTable** para identificar o tipo de recordset criado. Veja o código abaixo:

```
Dim meubd as Database
Dim minhathl as Recordset
Set meubd = DBEngine.Workspaces(0).OpenDatabase("C:\meudir\dados.mdb")
Set minhathl = meubd.OpenRecordset("Clientes", DbOpenTable)
```

A novidade em relação ao código acima é a definição da variável objeto tipo Recordset minhathl e atribuição a esta variável da tabela *Clientes* do banco de dados *Dados.mdb*. Note que o arquivo de banco de dados é padrão Access.

• Dynasets: uso, vantagens/desvantagens, configuração

Um *dynaset* é um conjunto de ponteiros de registros provenientes de uma ou mais tabelas, e indicam os dados especificados existentes quando da criação do *dynaset*. Os *dynasets* são recordsets atualizáveis e refletem as alterações feitas pelo usuário nas tabelas e no próprio *dynaset*.

Vantagens do uso de Dynasets

- Permitem reunir informações de diversas tabelas, através de instruções **SQL**.
- Permitem utilizar os métodos de busca *Find* e o uso de filtros e ordem de classificação para mudar a visão dos dados.

Desvantagens do uso de Dynasets

- Não admitem a criação de índices e do método *Seek*
- Não reflete inclusões ou exclusões feitas por outros usuários. Para mostrar as alterações é necessário renovar (refresh) o *dynaset*

Configuração de um Dynaset

Para utilizar um *dynaset* você deve definir um objeto recordset (Dim) e depois gerar o *dynaset* com o método *OpenRecordset*. Para selecionar os registros você pode usar uma instrução SQL. Veja exemplo abaixo:

```
Dim meubd as Database
Dim meudyn as Recordset
Set meubd = DBEngine.Workspaces(0).OpenDatabase("C:\meudir\dados.mdb")
Set meudyn = meubd.OpenRecordset("SELECT * FROM Clientes", DbOpenDynaset)
```

Definimos a variável objeto meudyn, abrimos o banco de dados e a seguir criamos um *dynaset* que é composto por todos os registros da tabela Clientes, através da instrução **SQL** que seleciona (SELECT) todos (*) os registros da tabela *Clientes*.

Obs: Também é possível criar um *dynaset* a partir de outro *dynaset*.

• Snapshots: uso, vantagens/desvantagens, configuração

Um **snapshot** é uma cópia dos dados de um recordset em um dado instante. Ele é muito semelhante a um *dynaset* pois pode ser criado a partir das tabelas básicas com as mesmas instruções (SQL, QueryDef,...). A diferença principal é que um **snapshot** não pode ser atualizado.

Vantagens do uso de Snapshots

- Permitem reunir informações de diversas tabelas.
- Permitem utilizar os métodos de busca *Find* e o uso de filtros e ordem de classificação para mudar a visão dos dados.

- Para os **snapshots**, navegação pelos registros e a criação de recordsets pode ser mais rápida do que nos *dynasets*.

Desvantagens do uso de Snapshots

- Um **snapshot** não pode ser atualizado.
- Um **snapshot** não admite a criação de índices.

Configuração de um Snapshot

Para utilizar um *snapshot* você deve definir um objeto recordset (Dim) e depois gerar o *snapshot* com o método *OpenRecordset*. Para selecionar os registros você pode usar uma instrução SQL. Veja exemplo abaixo:

```
Dim meubd as Database
Dim meusnap as Recordset
Set meubd = DBEngine.Workspaces(0).OpenDatabase("C:\meudir\dados.mdb")
Set meusnap = meubd.OpenRecordset("SELECT * FROM Clientes", DbOpenSnapshot)
```

O código é praticamente idêntico ao utilizado para a criação de um *dynaset*.

• Métodos de Movimentação: Movefirst, Movenext,...

Os métodos de movimentação permitem a passagem de um registro para outro no interior dos *recordsets*, e alteram a posição do ponteiro do registro ao passar de um registro ativo para outro registro. Você pode usar os métodos de movimentação sobre quaisquer *recordsets*.

Vejam os métodos de movimentação:

MoveFirst:

Movimenta o ponteiro do registro ativo para o primeiro registro do recordset aberto.

MoveNext:

Movimenta o ponteiro do registro ativo para o registro seguinte. Se não houver registro seguinte, você está no último registro, o flag de final de arquivo **EOF** será ativado.

MovePrevious:

Desloca o ponteiro do registro ativo para o registro anterior no recordset aberto. Se não houver registro anterior, você está no primeiro registro, o flag de início de arquivo **BOF** será ativado.

MoveLast:

Movimenta o ponteiro do registro ativo para o último registro do recordset aberto.

Move n

Desloca o ponteiro de registro *n* registros para frente (*n positivo*) ou para trás (*n negativo*) a partir do registro ativo no recordset aberto. Se o deslocamento levar o ponteiro de registro além dos limites do *recordset* ocorrerá um erro.

• Métodos de Localização: FindFirs, FindNext, ...

Utilizamos os métodos de localização para localizar registros que satisfaçam a critérios determinados. Os métodos de localização somente podem ser utilizados em **Dynasets** e **Snapshots**. Não é possível utilizá-los em objetos **Tabela**.

Vejam os métodos de Localização:

FindFirst:

A partir do início do recordset, localiza o primeiro registro que atende aos critérios definidos.

FindNext:

A partir da posição atual no recordset, localiza o registro seguinte (**próximo**) que satisfaz aos critérios definidos.

FindPrevious:

A partir da posição atual no recordset, localiza o registro seguinte(**anterior**) que satisfaz aos critérios definidos.

FindLast:

A partir do início do recordset, localiza o último registro que satisfaz aos critérios definidos.

Após a executar o método de localização você deve verificar o status da propriedade **NOMATCH**. Se **Nomatch** é verdadeira o método não encontrou o registro desejado, se **Nomatch** é falsa o ponteiro do registro se posiciona no registro encontrado.

Como exemplo suponha que temos um banco de dados **Controle**, no diretório Controle, que contém uma tabela **Clientes**, e que desejamos encontrar o cliente de nome Ana Maria Rosa nesta tabela. Veja o fragmento de código abaixo:

```
Dim bd as Database
Dim rs as Recordset
Dim criterio as string

Set bd = DBEngine.Workspaces(0).OpenDatabase("C:\Controle\Controle.mdb")
Set rs = bd.OpenRecordset("Clientes", DbOpenDynaset)

criterio="nome='Ana Maria Rosa'"

rs.findfirst criterio

if rs.nomatch
    msgbox "Cliente não localizado. "
else
    msgbox rs.Fields("nome") & " localizado no arquivo. "
endif
```

• Uso do método Seek.

O método **Seek** somente pode ser utilizado com o recordset do tipo **Table(Tabela)**, e, embora ele seja o mais rápido dos métodos de localização e posicionamento de registros apresenta as seguintes limitações:

- Só pode ser executado sobre uma tabela; não é possível usá-lo com um *dynaset* ou *snapshot*.
- Só pode ser usado com um índice ativo, e os parâmetros devem coincidir com os campos do índice ativo.
- Um **Seek** só localiza o primeiro registro que satisfaz aos valores de índices especificados.

Para invocar o método **Seek** é necessário fazer a chamada ao método usando os operadores de comparação (<, <=, =, >=, >, <>), lembrando que os valores de chaves que estão sendo comparados devem ter o mesmo tipo de dados dos campos no índice de controle. Veja exemplo a seguir:

```
Dim bd as Database
Dim rs as Recordset

Set bd = DBEngine.Workspaces(0).OpenDatabase "C:\Controle\Controle.mdb")
Set rs = bd.OpenRecordset("Clientes", DbOpenTable)

rs.index = "Nome"

rs.seek "=", "Ana Maria Rosa"

if rs.nomatch
    msgbox "Cliente não localizado. "
else
    msgbox rs.Fields("nome") & " localizado no arquivo. "
endif
```

No código acima definimos um recordset *rs* do tipo *Tabela* e a seguir definimos o índice ativo *rs.index = "Nome"* e invocamos o método **Seek** usando o operador = com nome a ser procurado.

Obs: Note que: O índice já deve ter sido criado, o operador deve estar entre aspas e que deve ser separado por vírgula do valor a ser localizado.

1.5. DATA CONTROL. (DEFINIÇÃO, PRINCIPAIS MÉTODOS E EVENTOS RELACIONADOS)

- **Que é o Controle de dados.**

O *controle de dados* foi criado para proporcionar um meio simples de acesso a um banco de dados. Para usar o controle de dados, siga os passos a seguir:

Selecione o controle na caixa de ferramentas.

Desenhe o controle em seu formulário.

Defina a propriedade **DatabaseName**

Defina a propriedade **Recordsource**

Defina a propriedade **RecordsetType**

A essa altura você criou os vínculos para o banco de dados e o *Recordset* com o qual deseja trabalhar. O controle assume para si as tarefas de:

Abrir o banco de dados e as tabelas ou consultas associadas.

Navegar pelos registros. (Utiliza métodos *MoveFirst*, *MoveLast*, *MoveNext* e *MovePrevious*.)

Carregar os valores diversos campos nos controles utilizados para exibir e editar os dados.

Gravar os dados editados dos controles no *Recordset*

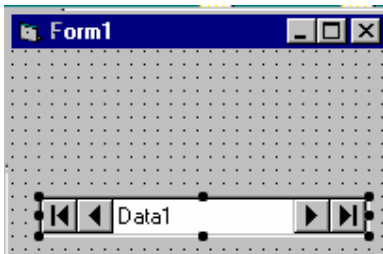
- **Funcionamento e utilização.**

Com base nas propriedades **Name** e **RecordSource** o controle de dados cria um *Recordset* que fornece acesso aos seus dados.



De início selecione o objeto controle de dados na *Toolbox* do Visual Basic e acrescente-o ao seu formulário. (Fig.1)

Fig-1



Defina a seguir as propriedades Name - Nome do Controle (O padrão é *data1*) e Caption - Especifica o nome que aparece no controle de dados. (O padrão é *Data1*) e dimensione o controle de dados. (Fig.2)

Fig-2

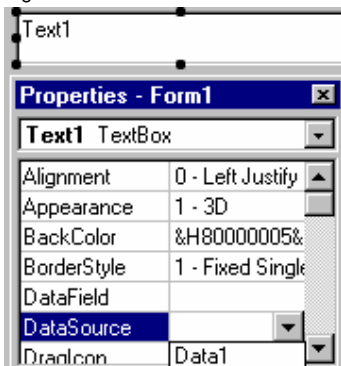


Fig-3

A seguir acrescente ao seu formulário um quadro de texto (*TextBox*) para cada campo que desejar acessar a partir do *Recordsource*, anexe cada quadro de texto ao objeto de controle de dados (*DataSource*) e especifique o campo que cada quadro de texto deverá exibir (*DataField*). (fig3.)

Definição de Propriedades.

Agora você deve associar um banco de dados e um recordset a um controle de dados definindo as propriedades do controle. As propriedades necessárias a isso são :

DatabaseName - Para bancos de dados do Access, representa o nome do arquivo do banco de dados, inclusive o nome completo do caminho. (Para banco de dados diferentes do Access é necessário informar o nome do caminho até o subdiretório dos dados. Ex-Para padrão Dbase : DatabaseName=c:\dir)

RecordSource - Especifica as informações que deseja obter no banco de dados - uma tabela, uma instrução SQL .

RecordsetType - Se deseja criar um *2-snapshot* ou um acessar uma *0-table* altere esta propriedade, pois o padrão é *1-dynaset*.

Connect - Necessária quando o banco de dados não for do Access. Ex - Para banco de dados padrão Dbase - Connect=dBASEIII.

É possível optar por ativar essas propriedades durante a execução do seu aplicativo. Vejamos um exemplo:

Você acrescentou um controle dados a seu formulário e definiu a propriedade *Name* como **Data1**, e quer ter acesso no arquivo *TESTE.MDB* (arquivo access) aos dados da tabela clientes, vejamos o código abaixo:

```
data1.DatabaseName="C:\TESTE\TESTE.MDB" 'path/nome do arquivo
data1.RecordSource="Clientes"           'fonte dos dados(tabela clientes)
data1.recordsetType=0                   'tipo de recordset(0-tabela)
data1.Refresh                          'implementa as alterações
```

Control Data: Vantagens/Limitações.

Vantagens Controle de Dados

A principal vantagem do controle de dados é a necessidade de menor trabalho de programação para desenvolver um aplicativo de acesso de dados. Você não precisa fornecer código de programa para abrir ou criar um banco de dados ou um recordset, para se movimentar pelos registros ou para editar registros.

Vínculo direto com os dados. Você não precisa invocar os métodos *Edit* e *Update* para modificar os dados contidos no banco de dados. As alterações aparecem no banco de dados tão logo são digitadas.

Os vários controles vinculados(*DbList*, *DbCombo*, *DbGrid*, ...) oferecem um meio fácil de realizar tarefas que via código seriam complexas.

As limitações do Controle de dados

Não apresentam função para alterar ou excluir registros.

É mais difícil a implementação do processamento de transações.

Um número excessivo de controles vinculados em um formulário torna a sua carga mais lenta.

Adição, Alteração, inclusão e validação de dados

Para contornar a ausência de funções para excluir e alterar registros podemos acrescentar as mesmas à tela de entrada de dados usando comandos de programa atribuídos a um botão de comando ou a eventos do controle de dados. Vejamos um exemplo de código que poderia ser atribuído a botões de comando de um formulário de entrada de dados:

1-Botão para inclusão de dados:

```
data1.Recordset.AddNew
```

2-Botão para exclusão de dados:

```
data1.Recordset.Delete
data1.Recordset.MoveLast
```

Observe que não foi necessário utilizar o comando **Update**, pois as atualizações são feitas automaticamente pelo controle de dados quando do deslocamento para um novo registro ou ao fechar o formulário através dos eventos *Validate* e *Reposition*.

Validate

Este evento ocorre sempre antes de um registro se tornar o registro corrente ou antes da execução do método *Update* ou antes da exclusão de um registro ou do fechamento do recordset/formulário; Você pode associar a este evento o código para validação de seus dados, como no exemplo abaixo:

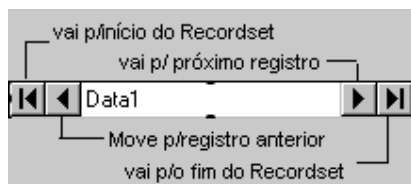
```
Private Sub data1_Validate(Action As Integer, Save As Integer)
If Save Then
    Select Case MsgBox("Salvar Alterações ?", vbQuestion + vbYesNo)
        Case vbYes '6
            If nome.Text = Empty Then
                MsgBox "Informe o nome do Cliente !", , "Gravar Clientes"
                nome.SetFocus
                Save = False
                Action = vbDataActionCancel
                Exit Sub
            ElseIf endereco.Text = Empty Then
                MsgBox "Informe endereço do cliente !", , "Gravar Clientes"
                endereco.SetFocus
                Save = False                '-não salva
                Action = vbDataActionCancel '-cancela
                Exit Sub
            End If
        Case vbNo '7
            Save = False
        End Select
End If
End Sub
```

Outro evento que pode ser utilizado para validar dados é o evento *Reposition*:

Evento Reposition

Ocorre sempre após o controle posicionar o registro corrente e após a carga do recordset pelo controle Data, neste momento o primeiro registro do recordset torna-se o registro corrente e o evento *Reposition* é gerado.

Movendo-se em um recordset com o Controle Data



Quando você move o registro corrente do recordset, através dos botões de comando disponíveis, temos as seguintes ações sendo executadas:

Todos os controles vinculados informam ao *Controle Data* as alterações ocorridas no seu conteúdo.

Controle Data gera um evento *Validate*, indicando a ação tomada.

Controle Data salva os dados no banco de dados, ou se for o caso gera um evento *Error*

Controle se move para o registro solicitado e gera um evento *Reposition* informando a todos os controles vinculados os novos valores dos campos

Considerações finais.

Você pode utilizar o controle de dados sozinho no caso de aplicativos em que o usuário tenha acesso somente para leitura ou quando não existe a necessidade de inclusão/exclusão de registros. O Controle de dados é ótimo para prototipação de um aplicativo, pois você pode desenvolver rapidamente os formulários.

Para um sistema multiusuário de uso intenso o controle de dados pode não ser uma boa escolha, pois pode aumentar a quantidade de conflitos de bloqueio de registros.

Provavelmente a melhor solução seja o uso combinado dos controle de dados e de código de programa.

1.6. EVENTOS: UM NOVO ENFOQUE (EVENTOS: CONCEITOS, EXEMPLOS.)

Quem já utilizou como ferramenta de programação uma linguagem procedural como o Clipper deverá se lembrar que o código do programa era um fluxo contínuo onde havia chamadas a outros programas e onde o usuário deveria obedecer a uma ordem pré-determinada pela aplicação; como exemplo temos que em uma tela de entrada de dados para Nome do Cliente, Endereço, Cep e idade se o usuário quisesse entrar o número do Cep, deveria passar obrigatoriamente pelos campos anteriores.

Os programas que rodam sob o windows não se comportam dessa maneira. Você constrói o seu código em torno de **eventos**, e, eventos são determinados pela ação do usuário e podem ser : pressionar um botão, adicionar, alterar, excluir um registro, selecionar uma opção, etc.

Então, os programas e funções são acionados por eventos que são reconhecidos pelos objetos do sistema que respondem aos eventos à medida que eles forem ocorrendo, executando as rotinas programadas. Existem várias categorias de eventos, entre elas podemos citar:

- 1-Eventos de Janela (Ao Abrir, Form_load, etc.)
- 2-Eventos de Dados (Ao Alterar, etc.)
- 3-Eventos de foco (Ao Receber Foco, Getfocus, etc.)
- 4-Eventos de teclado (Ao Pressionar Tecla, Keypress, etc.)
- 5-Eventos de mouse (Ao Clicar, Click, etc.)
- 6-Eventos de impressão (Ao Imprimir, etc.)
- 7-Eventos de erros (OnError)

Vejamos alguns eventos:

Evento Click

Ocorre quando o usuário pressiona e libera um botão do mouse sobre um **objeto**. O evento Click também pode ocorrer quando a definição da propriedade Value de um controle é alterada. Normalmente, você anexa uma **macro** ou **procedimento de evento Click** a um botão de comando para executar comandos ou ações compatíveis.

Evento Change

Ocorre quando o conteúdo de uma **caixa de texto** ou porção de texto de uma **caixa de combinação** é alterado. Exemplos deste evento incluem o fornecimento de caracteres diretamente na caixa de texto ou caixa de combinação, ou a alteração da definição da propriedade **Text** de um controle usando-se uma **macro** ou o Access Basic.

Durante a execução de uma macro ou um evento de procedimento, quando ocorrer um evento Change, você pode coordenar a exibição de dados entre os controles. Poderá também exibir os dados em um formulário de um controle, e os resultados em um outro controle.

Evento Load

Ocorre quando um formulário é aberto e os registros são exibidos. O evento Load pode ocorrer quando uma aplicação se inicia.

Eventos Open, Close

Open - para os formulários, o evento Open ocorre quando um formulário é aberto, mas antes que o primeiro registro seja apresentado. Para os relatórios, o evento ocorre antes que um relatório seja visualizado ou impresso.

Close - ocorre quando um formulário ou relatório é fechado e removido da tela.

O evento **Open** ocorre antes do evento **Load**, que por sua vez é disparado quando um formulário é aberto e seus registros exibidos.

O evento **Close** ocorre depois do evento **Unload**, que é disparado depois que o formulário é fechado, mas antes que ele seja removido da tela.

Os formulários, relatórios, botões de comando, caixa de listagem, caixa de verificação, botão de opção, grupo de opção e demais objetos possuem propriedades, métodos e eventos associados que podemos utilizar para executar rotinas apropriadas ao contexto. Além disso eventos podem disparar outros eventos.

1.7. O PODER DA SQL (A LINGUAGEM SQL : INTRODUÇÃO.)

A linguagem **SQL** (Structured Query Language) é uma linguagem de alto nível para manipulação de dados dentro do modelo relacional. Seu objetivo é fornecer uma interface de alto nível ao usuário. É uma linguagem não procedural, e, não cabe ao usuário definir como o gerenciador de banco de dados executará uma tarefa, mas somente o ele que deve fazer.

Uma instrução SQL consiste em três partes:

As declarações de parâmetros

A instrução manipulativa

As declarações de opções

Para termos uma idéia do seu poder, imagine que temos que atualizar o campo valor em 10% de uma tabela com diversos registros. Na abordagem procedural teríamos os seguintes passos a seguir:

Abrir a tabela.

posicionar o ponteiro no início da tabela.

atualizar o campo valor em 10%.

mover o ponteiro para o próximo registro.

continuar a atualização do campo valor até o final da tabela.

O código poderia ter o seguinte aspecto:

```
Dim db as database
Dim tabela as recordset
set db=workspaces(0).opendatabase("c:\base.mdb")
set tabela=db.openrecordset("tabela")
While1 not tabela.eof
tabela.edit
tabela.valor=tabela.valor*1.10
tabela .update
tabela.movenext
Wend
tabela.close
```

Agora utilizando uma instrução SQL teríamos o seguinte trecho de código:

```
Dim db as Database
Set db=Workspaces(0).opendatabase("c:\base.mdb")
db.execute "UPDATE tabela SET valor=valor*1.10"
db.close
```

Observe a utilização da instrução SQL **UPDATE**, bem mais simples, não é ?

Então se você não está utilizando a SQL, estará trabalhando muito e seu código sofrendo as consequências.

Vejam na tabela abaixo um resumo das cláusulas manipulativas e suas finalidades:

Instrução	Função
SELECT	Obtém um grupo de registros e insere os registros em um dynaset ou em uma tabela

UPDATE	Define os valores dos campos de uma tabela em uma atualização
TRANSFORM	Cria uma tabela de resumo, utilizando o conteúdo de um campo como cabeçalho de cada coluna
DELETE FROM	Remove registros de uma tabela
INSERT INTO	Acrescenta um grupo de registros a uma tabela.

Vejamos alguns exemplos da instrução SELECT:

1) Seleciona os campos "Primeiro nome" e "Sobrenome" de todos os registros da tabela Empregados.

SELECT [Primeiro nome], [Sobrenome] FROM Empregados

2) Seleciona todos os campos da tabela Empregados. Note o uso parâmetro (*) indicando todos os campos da tabela indicada.

SELECT Empregados.* FROM Empregados

3) Conta o número de registros que têm uma entrada no campo "Código postal" e coloca o título Contagem no topo da coluna.

SELECT Count([Código postal]) AS Contagem FROM Clientes

4) Seleciona os campos "Primeiro nome" e "Sobrenome" de cada registro cujo sobrenome seja Pereira.

SELECT [Primeiro nome], [Sobrenome] FROM Empregados WHERE [Sobrenome] = 'Pereira'

5) Seleciona os campos "Primeiro nome" e "Sobrenome" para Empregados cujos sobrenomes começam pela letra S.

SELECT [Primeiro nome], [Sobrenome] FROM Empregados WHERE [Sobrenome] Like 'S'

FROM - Indica as tabelas utilizadas como fonte de dados

WHERE- Especifica as condições que os registros devem satisfazer para compor o subconjunto de dados.

1.8. USANDO EDIT, ADDNEW E UPDATE...(EDITANDO, INCLUINDO E ATUALIZANDO...)

Ao utilizar o método Edit o Jet copia o registro atual de um Recordset do tipo dynaset ou Tabela para o Buffer de Cópia para posterior edição.

A Sintaxe usada é a seguinte: **recordset.Edit**

O recordset correspondente representa o nome do objeto Recordset atualizável que contém o registro que você quer editar.

Observações

Logo que você usa o método **Edit**, as mudanças feitas nos campos do registro atual são copiadas no buffer de cópia. Depois que você faz as mudanças desejadas no registro, use o método de Atualização **Update** para salvar as alterações.

Precaução

- Se você edita um registro e então executa qualquer operação movendo-se para outro registro sem usar o método **Update** primeiro, suas alterações serão perdidas sem aviso.

- Além disso, se você fecha o recordset ou termina o procedimento que declara o recordset ou

o objeto de Banco de dados relacionado, seu registro editado é descartado sem aviso algum.

Quando a propriedade **LockEdits** do Recordset for Verdadeira(**bloqueio pessimista**) em um ambiente de multiusuário, o registro permanece bloqueado desde o momento de utilização do método **Edit** até que a atualização for completada.

Se a propriedade **LockEdits** for Falsa (**bloqueio otimista**), o registro é bloqueado e comparado com o registro pre-editado logo antes de sua atualização no banco de dados.

Se o registro mudou desde que você usou o método **Edit**, a operação de Atualização falha gerando o **erro(3197)**.

Nota

O bloqueio Otimista sempre é usado em formatos de banco de dados externos, como ODBC e instável ISAM.(Dbase,Fox,...)

O registro atual permanece atual depois que você usou o **Edit**.

Para usar o **Edit**, deve haver um registro atual. Se não há nenhum registro atual ou se o recordset não se refere a um Recordset do tipo dynaset ou tipo Tabela um erro acontece.

Ao Usar o método **Edit** o Jet gera um erro nas seguintes condições:

Não há nenhum registro atual.

O banco de dados ou recordset é somente de leitura.

Nenhum campo no registro é atualizável.

O banco de dados ou recordset foram abertos para uso exclusivo por outro usuário.

Outro usuário bloqueou a página que contém seu registro.

Precaução

Se você usa qualquer método de edição da **DAO (AddNew, Edit, Delete, ou Update)** dentro de um loop que editará mais de um registro, o Jet Engine versão 3.0 pode tratar o loop inteiro como uma transação implícita.

Significa que você necessariamente não poderá garantir que cada atualização seja escrita imediatamente no banco de dados.

Isto pode afetar o tratamento de erros, bloqueio de páginas, e a visibilidade imediata das mudanças para outros usuários.

Para garantir que as alterações sejam escritas imediatamente no banco de dados, inclua cada bloco - **Edit. . . Update** - dentro de uma transação explícita.

Para mais informação sobre transações, veja o **BeginTrans, CommitTrans, Rollback**.

1.9. VISUAL BASIC - DATAS E O ANO 2000.(COMO O VB INTERPRETA O ANO COM DOIS DÍGITOS.(DD/MM/YY)

"**Time is on my side...**", este é um refrão de uma música tocada pelos Rolling Stones, que significa mais o menos o seguinte "O tempo esta do meu lado". O que significa isto? alguém pode pensar ao ler estas linhas.

Calma, isto é apenas uma chamada para alertar que os desenvolvedores de aplicações não podem cantar o refrão sem pensar que na verdade **"Time is against me..."**, i.e, o tempo está contra mim. Sabe porque ? Por que nos aproximamos do século XXI e o **"Bug do Milênio"** está cada vez mais perto.

Vamos procurar entender a razão do problema e de como tratá-lo para evitar frustrações futuras. Iremos nos ater apenas aos seguintes aplicativos da Microsoft utilizados em aplicações desktop: **Visual Basic e Access.**

Um Pouco de história

Para todas as versões do Visual Basic ate a versão 3.0, datas informadas com dois dígitos (dd/mm/yy), são consideradas como pertencentes ao século XX, i.e, 1900.

12/01/09 -----> 12/01/1909
01/01/00 -----> 01/01/1900
30/02/40 -----> 30/02/1940.

O código que implementa este tratamento está, neste caso, embutido no run-time de cada versão, e não depende da versão do sistema operacional utilizado nem do formato da data usada.

Com o surgimento da **Automação OLE** e do **VBA (Visual Basic for Applications)** o código responsável pela conversão dos dois dígitos usados em datas (dd/mm/yy) para quatro dígitos (dd/mm/yyyy) passou a ser responsabilidade das livrarias de AUTOMAÇÃO OLE. Então a VBA não mais implementa este código, apenas chama a livraria responsável pela conversão.

A partir da versão 4.0 do Visual Basic foi definida um novo comportamento para as datas informadas com dois dígitos no ano (dd/mm/yy). Elas são convertidas para o século utilizado na data do sistema. Assim se a data do sistema é o presente século (1900) **01/01/00** será interpretada como 01/01/1900, porém se a data do sistema for o século XXI (2000), a data será interpretada como 01/01/2000.

Esta regra foi implementada na livraria de Automação OLE usada pelo Visual Basic 4.0 e pelo VBA.

Acontece que a Microsoft alterou a interpretação dada aos dois dígitos para o ano a partir da versão 2.20.4049 da biblioteca **OLEAUT32.DLL**. Para as aplicações que usam esta biblioteca datas com dois dígitos para o ano cujo dígitos sejam de 00 a 29, são interpretadas como sendo do século XXI e as demais como sendo do século XX.

12/01/09 -----> 12/01/2009
01/01/29 -----> 01/01/2029
30/02/40 -----> 30/02/1940.
30/02/72 -----> 30/02/1972.

Tentando Arrumar a Casa.

Meu Deus, que confusão !!! Vamos tentar por as coisas nos seus lugares.

Aplicativo	Regra
Aplicações de 16 bits: Access 2.0, VB 3.0	Todos os anos com dois dígitos são automaticamente considerados como do século 20
VB 4.0 16 bits	Todos os anos com dois dígitos são do século corrente (definido pelo relógio do sistema).
Access 95 (com a nova biblioteca), VB 4.0 32 bits, VB5, Access 97	Todos os anos com dois dígitos entre 00 e 29 são do século XXI e os demais do século XX

Já deu para perceber que é importante saber com qual versão de biblioteca OLEAUT32.DLL seu sistema esta trabalhando. Para isso faça o seguinte:

1-abra o Explorer e no diretório System;

2-localize o arquivo **OLEAUT32.DLL**;
 3-Clique com o botão direito do mouse sobre o arquivo;
 4-Selezione propriedades;
 5-Selezione a ficha Versão e clique em Versão do Produto;
 6-Se o número da versão for igual ou superior a 2.20.4049 a nova regra esta sendo usada.

Criando a sua própria interpretação. Você pode querer implementar as suas próprias regras, como por exemplo: - datas com dois dígitos no ano entre 00 e 49 serão interpretadas como do século XXI (2000 e 2049), - e datas entre 50 e 99 como do século XX.(1950 e 1999). Para isto basta criar uma procedure com o seguinte código:

```
Private Sub cmdConvertDate_Click()
    Dim strYear As String
    Dim intSlash As Integer

    If IsDate(txtDate) or txtDate = "2/29/00" Then
        'procura primeiro separador
        intSlash = InStr(txtDate, "/")
        If intSlash > 0 Then
            'procura segundo separador
            intSlash = InStr(intSlash + 1, txtDate, "/")
            If intSlash > 0 Then
                'Extrai ano da data
                strYear = Mid(txtDate, intSlash + 1)
                If Len(strYear) = 2 Then
                    If CInt(strYear) < 50 Then
                        ' Menor que 50: ano = 20XX.
                        strYear = "20" & strYear
                    Else
                        ' Maior que 50: ano = 19XX.
                        strYear = "19" & strYear
                    End If
                End If
                MsgBox "Data Informada: " & txtDate
                MsgBox "ANO (Sua Regra): " & strYear
                MsgBox "ANO (Regra do VB): " & Year(txtDate)
            Else
                MsgBox "Data no formato não esperado!"
            End If
        Else
            MsgBox "Data no formato não esperado!"
        End If
    Else
        MsgBox "Data inválida !"
    End If
    ' Limpando a data no texto
    txtDate.Text = Left(txtDate.Text, intSlash) & strYear
End Sub
```

Bem, espero não ter criado mais confusão com as datas.

(Referência: Article ID:Q162718 - Microsoft)

1.10. SQL : CRIANDO CONSULTAS PARAMETRIZADAS NO VISUAL BASIC.

O Microsoft Access nos dá a possibilidade de criar e armazenar consultas dentro do arquivo de banco de dados. A vantagem em armazenar as consultas no banco de dados é que para executar uma instrução SQL o JET verifica erros de sintaxe, depois ele tenta otimizar a instrução e a seguir executa a instrução, tudo isto leva tempo e a coisa piora se você estiver executando as mesmas instruções SQL diversas vezes, pois o JET terá que refazer tudo novamente.

Ao criar e armazenar definições de consultas através de instruções SQL o Access analisa a instrução e a otimiza só então ele armazena a instrução original e a otimizada, quando for executar a instrução o trabalho estará feito e o tempo de execução será diminuído.

Cada consulta armazenada em um banco de dados Access é um objeto **QueryDef**, o conjunto de objetos QueryDef compõem a **coleção QueryDefs** do objeto **Database**. Então uma **QueryDef** é uma consulta SQL pré-compilada e pré-otimizada.

Para criar uma **QueryDef** usamos o método **CreateQuery** do objeto **Database** ou a criamos diretamente usando o próprio Microsoft Access.

O **CreateQuery** necessita de dois parâmetros: o nome da QueryDef e a instrução SQL que o cria.

Vejamos com o criar uma consulta SQL e armazená-la no banco de dados:

Para criar uma consulta chamada **Lista_Alunos**, que lista todos os alunos por ordem alfabética da tabela **tblalunos** e armazená-la no banco de dados **Escola.mdb**, fazemos o seguinte:

```
Dim db as Database
Dim qd as QueryDef
set db=DbEngine.Workspaces(0).OpenDatabase(app.path & "\escola.mdb")
set qd=db.CreateQueryDef("Lista_alunos", "SELECT * FROM tblalunos ORDER
BY nome")
```

Uma vez criada a **QueryDef** podemos executá-la e manipulá-la como um recordset:

```
Dim db as Database
Dim dyn as Recordset
set db=DbEngine.Workspaces(0).OpenDatabase(app.path & "\escola.mdb")
set dyn=db.QueryDefs("lista_alunos").OpenRecordset(dbOpenDynaset)
```

Ou eliminá-la :

```
db.QueryDefs.Delete "Lista_alunos"
```

Para tirar maior proveito das **QueryDefs** costuma-se criá-las de forma que aceitem parâmetros, para isto usamos a palavra-chave **PARAMETERS** para passar um ou mais parâmetros para as consultas SQL.

Suponha que no caso do exemplo anterior gostaríamos de listar todos os alunos de uma determinada série, para isso criamos a consulta e passamos o parâmetro da série desejada:

```
Dim db as Database
Dim qd as QueryDef
Dim dyn as Recordset
Dim Sql as String
set db=DbEngine.Workspaces(0).OpenDatabase(app.path & "\escola.mdb")
sql= "PARAMETERS pSerie String SELECT * FROM tblalunos WHERE serie = pSe-
rie "
sql=sql & " ORDER BY nome "
set qd=db.CreateQueryDef("Lista_alunos", Sql )
qd.Parameters("pSerie")= "1"
set dyn= qd.OpenRecordset()
```

Métodos de QueryDef

Os métodos usados pelas QueryDefs são: **Execute** e **OpenRecordset**.

O método **Execute** é usado para executar comandos de ação como **DELETE** ou **UPDATE** que não retornam um conjunto de registros, é usado também para criar tabelas e índices via instrução SQL.

O método **OpenRecordset** é usado para retornar um conjunto de registros como resultado da consulta.

Assim para deletar todos os alunos inativos, fazemos:

```
Dim db as Database
Dim qd as QueryDef
Dim dyn as Recordset
Dim Sql as String
set db=DbEngine.Workspaces(0).OpenDatabase(app.path & "\escola.mdb")
sql= "DELETE * FROM tblalunos WHERE ativo=False "
set qd=db.CreateQueryDef("Apaga_alunos", Sql )
```

qd.Execute

As **QueryDef** possuem as seguintes propriedades:

MaxRecords - fixa um limite que um QueryDef pode retornar

Name - Armazena e define o nome de um consulta

RecordsAffected - armazena o número de registros afetados por um comando Execute.

SQL - Retorna o comando SQL que originou a consulta.

2) TRABALHANDO COM SQL (Conceitos Básicos)

2.1 Iniciando com a SQL.

A **SQL - Structured Query Language** (Linguagem de Consulta Estruturada) praticamente surgiu com a terceira geração de banco de dados, os **RDBs-Relational Databases**, ou seja, banco de dados relacionais.

A **SQL** é uma linguagem padrão para o gerenciamento de banco de dados, e não é nem estruturada (Structured) e não está limitada somente a consultas (Queries) em banco de dados.

Na verdade podemos dizer que **SQL** é uma linguagem para definir e manipular bancos de dados relacionais e praticamente todos os produtos de bancos de dados relacionais que estão no mercado suportam a **SQL**.

Infelizmente ainda não existe uma padronização a nível da **SQL**; embora a portabilidade seja grande, dependendo do produto que se está utilizando (Oracle, Sybase, Informix, etc.) haverá diferenças na sintaxe das declarações. Atualmente o padrão **SQL-92** é o mais utilizado.

No nosso caso usaremos as declarações **SQL** utilizadas no **Microsoft Access**, pois estaremos trabalhando a nível de desktop.

Ao utilizar a **SQL** em seu código você terá muitos benefícios;

- Primeiro irá ganhar tempo pois a quantidade de linhas de código necessárias para realizar uma tarefa via **SQL** é menor que o método procedural via DAO.
- Seus projetos também ficarão mais rápidos, pois geralmente a **SQL** é mais rápida que o método procedural via DAO.
- A portabilidade sua aplicação será maior visto que a **SQL** é aceita pelos principais bancos de dados relacionais (Oracle, Sybase, etc.).

Creio que estes motivos já são suficientes para você dar uma olhada no que a **SQL** pode fazer por você e por seu código.

Com a **SQL** podemos realizar as seguintes tarefas:

- Criação de Bases de Dados (Ver Banco de Dados).
- Segurança de acesso aos Dados.
- Recuperar informações e Integridade de transações.
- Manipulação e controle de bases de dados.

Se você tem uma cópia do **Access**, sua incursão no mundo **SQL** será facilitada, pois o **Access** praticamente escreve em **SQL** para você, depois você só precisa copiar e colar as instruções no seu código em Visual Basic. Mas vale a pena aprender pelo menos as instruções mais importantes em **SQL**. Com isso você já conseguirá usar muito do seu potencial.

Talvez o maior problema para os que já utilizavam uma linguagem procedural (Basic, Clipper, Fortran, Cobol, etc), ao lidar com a **SQL** seja o seguinte: Você tem que mudar a forma de pensar ao programar com **SQL**.

Estas linguagens ditas da terceira geração, são caracterizadas por comandos que dizem ao computador exatamente o que fazer em estruturas sequenciais executadas passo a passo. São chamadas de linguagens procedurais.

A SQL, por outro lado, é caracterizada por ser uma linguagem declarativa, ou seja, ela diz ao computador o que quer que ele faça, sem se preocupar de que forma o trabalho será realizado, o que importa é o resultado.

A SQL é composta de subconjuntos de comandos para executar diferentes tarefas. Assim podemos dizer que a SQL suporta :

- Uma linguagem de definição de dados (DDL)
- Uma linguagem de manipulação de dados (DML)
- Uma linguagem de segurança de dados (DCL)

A **DDL** permite criar e modificar e excluir a estrutura de uma tabela e seus índices; seus principais comandos são:

CREATE	- Cria tabelas, campos e índices num banco de dados.
DROP	- Remove tabelas e índices de um banco de dados.
ALTER	- Altera a estrutura de uma tabela de um banco de dados.

A **DML** permite manipular os dados (Inserir, Excluir e Atualizar) bem como executar consultas através da recuperação de subconjuntos de dados para posterior tratamento. seus principais comandos são:

SELECT	- Seleciona um conjunto de registros de uma ou mais tabelas usando um critério específico.
INSERT	- Adiciona dados a uma tabela.
UPDATE	- Atualiza os dados de uma tabela segundo critérios específicos.
DELETE	- Remove registros de uma tabela.

A **DCL** permite a implementação da segurança interna do Banco de dados. Seus comandos principais são **GRANT** e **REVOKE**, mas como não é suportada pela plataforma Access, não iremos utilizá-la.

2.2 Criando tabelas via SQL.

Vamos criar via SQL as tabelas utilizadas na primeira seção de **Desvendando o Crystal Reports** (ver item 7 geral) Só para lembrar a estrutura das tabelas é dada a seguir:

TblAlunos	TblCursos	TblNotas	TblProfessor
codaluno nome endereço telefone nascimento nomepai nomemae observacao Periodo serie numero	codcurso nomecurso codprofessor	codaluno codcurso nota ano bimestre Observacao	codprofessor nome endereço telefone

Obs : **Codaluno** , **Codcurso**, **CodProfessor** são chaves primárias de suas respectivas tabelas. Para exercitar iremos definir os relacionamentos entre as tabelas da seguinte forma (figura 1.0 abaixo):

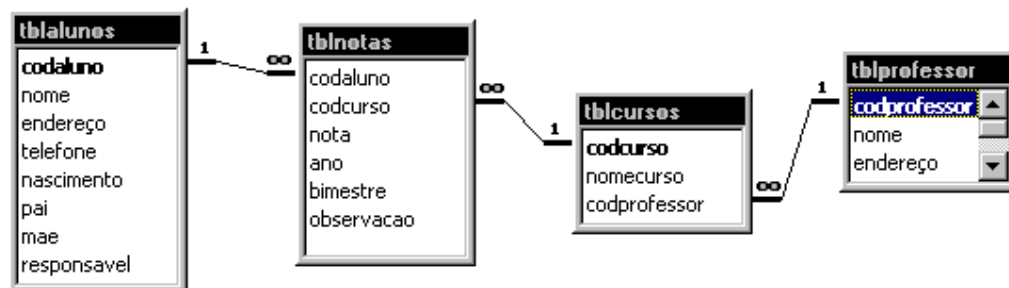


figura 1.0

A tabela **tblprofessor** pode ser definida como a seguir (Access SQL) :

```
CREATE TABLE tblprofessor
(codprofessor INTEGER CONSTRAINT primarykey PRIMARY KEY,
nome TEXT (50),
endereco TEXT (50),
telefone TEXT (15) );
```

A instrução **CREATE TABLE** irá criar a tabela **tblprofessor**, com as definições da lista de campos entre parênteses, separados um dos outros por vírgulas. Cada descrição de campo possui duas partes: o nome do campo e o tipo de dados os quais são separados por um espaço entre si.

A cláusula **CONSTRAINT** é utilizada para definir a chave primária **codprofessor**.

Para criar a tabela no VB execute a instrução **CREATE TABLE** passando-a como parâmetro do método Execute do objeto Database. Ex: db.Execute ("CREATE TABLE...")

Os nomes dos tipos de dados utilizados pelo JET não é igual aos nomes exigidos pelas instruções SQL. Veja na tabela a abaixo a correspondência entre ambos:

Tipos de Dados SQL	Tipos de dados do JET
BIT	YES/NO
BYTE	NUMERIC - BYTE
COUNTER	COUNTER - Contador
CURRENCY	CURRENCY - Moeda
DATETIME	DATE/TIME
SINGLE	NUMERIC - SINGLE
DOUBLE	NUMERIC - DOUBLE
SHORT	NUMERIC - INTEGER
LONG	NUMERIC - LONG
LONGTEXT	MEMO
LONGBINARY	OLE OBJECTS
TEXT	TEXT

A tabela **TblCursos** será criada pela instrução:

```
CREATE TABLE tblcursos
(codcurso INTEGER CONSTRAINT primarykey PRIMARY KEY,
nomecurso TEXT (15),
codprofessor INTEGER CONSTRAINT tblprofessorFK REFERENCES tblprofessor);
```

A cláusula **CONSTRAINT** é utilizada para definir uma chave primária e uma chave externa.

Note que existe uma relação de um para muitos entre a tabela **TblProfessor** e a tabela **TblCursos**, sendo que a coluna codprofessor da tabela **TblCursos**, é uma chave estrangeira (Foreign Key - FK)

2.3 Índices e Tabelas - Criar, Alterar e Excluir.

EXCLUINDO TABELAS

Para excluir uma tabela via SQL usamos a instrução **DROP TABLE nome da tabela**

Assim para excluir a tabela **tblalunos** criada anteriormente fazemos:

```
DROP TABLE tblalunos
```

A tabela e todo o seu conteúdo é excluída.

ALTERANDO TABELAS

Para alterar uma tabela , adicionando ou excluindo um campo da tabela, usamos a instrução: **ALTER TABLE**

1- Na inclusão de um campo temos que especificar o NOME, TIPO e TAMANHO do campo, e usar a cláusula **ADD COLUMN**

Assim para incluir o campo CIDADE com 50 caracteres na tabela **tblalunos** fazemos:

```
ALTER TABLE tblalunos ADD COLUMN cidade TEXT (50)
```

2- Na exclusão de um campo basta especificar o nome do campo e usar a cláusula **DROP COLUMN**

O código abaixo exclui o campo NOME da tabela **tblalunos**

```
ALTER TABLE tblalunos DROP COLUMN nome
```

Note que não será possível excluir um campo empregado em um índice ou em uma relação.

CRIANDO ÍNDICES

Podemos criar um índice de um único campo ou um índice de vários campos com a instrução: **CREATE INDEX**.

Na criação do índice devemos informar o nome do índice o nome da tabela e pelo menos um campo a ser incluído como índice. Assim se quisermos criar um índice pela data de nascimento na tabela **tblalunos** fazemos:

```
CREATE INDEX nascimento ON tblalunos(nascimento)
```

A instrução abaixo cria um índice chamado **series**, usando os campos serie e periodo da tabela **tblalunos**, sendo que serie será usado na ordem ASCENDENTE e periodo na ordem DESCENDENTE.

```
CREATE INDEX series ON tblalunos(serie ASC, periodo DESC)
```

EXCLUINDO ÍNDICES

Para excluir um índice de uma tabela utilize a instrução **DROP INDEX** Devemos informar o nome do índice e o nome da tabela. Assim para excluir o índice **serie** criando anteriormente fazemos:

```
DROP INDEX series ON tblalunos
```

2.4 Testando As Instruções Sql

Para criar e testar todas essas instruções SQL você pode codificá-las direto no Visual Basic e executar o código para ver o resultado, para isto voce deverá usar o método **Execute** (ver a seguir) do objeto **DataBase**(Veja artigo), mas você pode usar os seguintes meios:

- 1-) O aplicativo VISDATA que acompanha o Visual Basic
- 2-) O Microsoft Access
- 3-) O Microsoft Query (para usuários do Excel ou Office)

Eu o aconselho a evitar a codificação quando estiver testando as instruções, pois tanto o VISDATA como o Access são mais fáceis de usar e lhe dão o resultado de imediato.

SQL - Criando Consultas Parametrizadas

O microsoft Access nos dá a possibilidade de criar e armazenar consultas dentro do arquivo de banco de dados. A vantagem em armazenar as consultas no banco de dados é que para executar uma instrução SQL o JET verifica erros de sintaxe, depois ele tenta otimizar a instrução e a seguir executa a instrução, tudo isto leva tempo e a coisa piora se você estiver executando as mesmas instruções SQL diversas vezes, pois o JET terá que refazer tudo novamente.

Ao criar e armazenar definições de consultas através de instruções SQL o Access analisa a instrução e a otimiza só então ele armazena a instrução original e a otimizada, quando for executar a instrução o trabalho estará feito e o tempo de execução será diminuído.

Cada consulta armazenada em um banco de dados Access é um objeto **QueryDef**, o conjunto de objetos QueryDef compõem a **coleção QueryDefs** do objeto **Database**. Então uma **QueryDef** é uma consulta SQL pré-compilada e pré-otimizada.

Para criar uma **QueryDef** usamos o método **CreateQuery** do objeto DataBase ou a criamos diretamente usando o próprio Microsoft Access.

O **CreateQuery** necessita de dois parâmetros: o nome da QueryDef e a instrução SQL que o cria. Vejamos como criar uma consulta SQL e armazená-la no banco de dados:

Para criar uma consulta chamada **Lista_Alunos**, que lista todos os alunos por ordem alfabética da tabela **tblalunos** e armazená-la no banco de dados **Escola.mdb**, fazemos o seguinte:

```
Dim db as Database
Dim qd as QueryDef
set db=DbEngine.workspaces(0).OpenDatabase(app.path & "\escola.mdb")
set qd=db.CreateQueryDef("Lista_alunos", "SELECT * FROM tblalunos ORDER BY nome")
```

Uma vez criada a **QueryDef** podemos executá-la e manipulá-la como um recordset:

```
Dim db as Database
Dim dyn as Recordset
set db=DbEngine.Workspaces(0).OpenDatabase(app.path & "\escola.mdb")
set dyn=db.QueryDefs("lista_alunos").OpenRecordset(db.OpenDynaset)
```

Ou eliminá-la :

```
db.QueryDefs.Delete "Lista_alunos"
```

Para tirar maior proveito das **QueryDefs** costuma-se criá-las de forma que aceitem parâmetros, para isto usamos a palavra-chave **PARAMETERS** para passar um ou mais parâmetros para as consultas SQL.

Suponha que no caso do exemplo anterior gostaríamos de listar todos os alunos de uma determinada série, para isso criamos a consulta e passamos o parâmetro da série desejada :

```
Dim db as Database
Dim qd as QueryDef
Dim dyn as Recordset
Dim Sql as String
set db=DbEngine.workspaces(0).OpenDatabase(app.path & "\escola.mdb")
sql= "PARAMETERS pSerie String SELECT * FROM tblalunos WHERE serie = pSerie "
sql=sql & " ORDER BY nome "
set qd=db.CreateQueryDef("Lista_alunos", Sql )
qd.Parameters("pSerie")= "1"
set dyn= qd.OpenRecordset()
```

Métodos de QueryDef

Os métodos usados pelas QueryDefs são: **Execute** e **OpenRecordset**.

O método **Execute** é usado para executar comandos de ação como **DELETE** ou **UPDATE** que não retornam um conjunto de registros, é usado também para criar tabelas e índices via instrução SQL.

O método **OpenRecordset** é usado para retornar um conjunto de registros como resultado da consulta.

Assim para deletar todos os alunos inativos, fazemos:

```
Dim db as Database
Dim qd as QueryDef
Dim dyn as Recordset
Dim Sql as String
set db=DbEngineworkspaces(0).OpenDatabase(app.path & "\escola.mdb")
sql= "DELETE * FROM tblalunos WHERE ativo=False "
set qd=db.CreateQueryDef("Apaga_alunos", Sql )
qd.Execute
```

As **QueryDef** possuem as seguintes propriedades:

- **MaxRecords** - fixa um limite que um QueryDef pode retornar
- **Name** - Armazena e define o nome de um consulta
- **RecordsAffected** - armazena o número de registros afetados por um comando Execute.
- **SQL** - Retorna o comando SQL que originou a consulta.

3) TRABALHANDO COM REDES (Conceitos Básicos)

3.1 Introdução

Mesmo que você não desenvolva sistemas para um ambiente de rede, precisa saber alguns conceitos relativos ao assunto.

Naturalmente o Visual Basic fornece **acesso compartilhado** ao banco de dados do sistema e vários usuários poderão acessar os dados ao mesmo tempo; o VB faz obloqueio e o desbloqueio das páginas dos dados quando necessário, automaticamente.

Se você quiser que os dados de sua aplicação sejam acessados por um único usuário, este é o **modo exclusivo**, terá que informar isto ao VB.

Para abrir um arquivo de dados no VB usamos o método **OpenDatabase**, vejamos então sua sintaxe:

```
Set database = workspace.OpenDatabase(dbname[, exclusive[, read-only[, source]])
```

Observe o argumento opcional **exclusive**, ele pode assumir dois valores:

- *True* - Abre o arquivo no modo exclusivo, ou seja, não compartilhado.
- *False* - Abre o arquivo no modo compartilhado.

Se você omitir este argumento o arquivo é aberto no modo compartilhado.

O argumento **read-only** também pode assumir dois valores:

- *True* - Abre o arquivo no modo **read-only**, isto é, somente leitura.
- *False* - Abre o arquivo no modo **read/write**, isto é, leitura e escrita.

Se voce omitir este argumento o arquivo é aberto no modo leitura e escrita.

Portanto se você usa a seguinte sintaxe para abrir um banco de dados do seu sistema:

```
dim db as databaseset db = dbengine.workspaces(0).OpenDatabase(app.path & "\nome_arquivo")
```

O seu arquivo estará sendo aberto no modo compartilhado. Até aqui tudo bem !

3.2 Acesso aos Dados

O limite mais restritivo que podemos impor a um banco de dados é sua abertura em **modo exclusivo**. Isto impedirá que qualquer usuário tenha acesso ao banco de dados enquanto ele estiver sendo utilizado. (Isto inclui todas as suas tabelas e consultas.)

Vejamos a sintaxe utilizada para este caso:

```
dim db as databaseset db = dbengine.workspaces(0).OpenDatabase(app.path & "\nome_arquivo", True)
```

A tentativa de abrir o arquivo em uso irá gerar uma mensagem de erro pelo VB.

Quer algumas boas razões para usar o **acesso exclusivo** ?

Use-o em operações que afetem todo o banco de dados tais como:

- Compactação do banco de dados
- Atualização de tabelas inteiras-Alteração da estrutura do banco de dados(inclusão de tabelas, índices, etc...)

Se você não quiser ser tão radical, pode restringir o acesso à tabela em uso pelo seu sistema.

Para fazer isto utilizamos as opções do método **OpenRecordSet** para negar acesso de leitura e/ou gravação às tabelas, vamos a sua sintaxe:

```
Set variable = database.OpenRecordset(source[, type[, options]])
```

Vejamos as opções mais usadas:

dbDenyRead

Impede os usuários de examinar os dados contidos na tabela, até você fechá-la. (Disponível para recordsets do tipo tabela) Você usará esta opção quando for necessário atualizar informações na tabela inteira.

Ex:

```
Dim db as database,rs as recordset
Set rs=db.openrecordset ("nome_tabela", DbOpenTable, dbDenyRead)
```

dbDenyWrite

Os usuários podem visualizar os dados mas não podem atualizar as informações da tabela, nem incluir novos registros, até você fechá-la. (Válido para Tabelas, Dynasets e Snapshot) Você pode usar esta opção quando estiver incluindo novos registros na tabela e não houvesse modificações nos registros existentes.

Ex:

```
Dim db as database,rs as recordset
Set rs=db.openrecordset( "nome_tabela", DbOpenTable, dbDenyWrite)
```

dbReadOnly

Permite somente visualizar os dados da tabela.

Ex:

```
Dim db as database,rs as recordset
set rs=db.openrecordset ("nome_tabela", DbOpenTable, dbReadOnly)
```

Você pode usar esta opção em tabelas de pesquisa.

Uma outra maneira de restringir o acesso a apenas leitura, é utilizar **Snapshots**. Eles sempre são de somente leitura. A vantagem dos **Snapshots** é que eles ficam armazenados na memória.

Ex:

```
Dim db as database,rs as recordset
set rs=db.openrecordset ("nome_tabela", DbOpenSnapshot)
```

dbAppendOnly

Permite somente a inclusão de novos registros (Válido somente para dynaset)

Ex:

```
Dim db as database,rs as recordset set
rs=db.openrecordset("nome_tabela", DbOpenDynaset, dbAppendOnly)
```

Se você tentar abrir qualquer tabela com as restrições acima mencionadas e um outro usuário já estiver com ela aberta ocorrerá uma mensagem de erro.

Dai conclui-se que o código de tratamento de erros deverá ser utilizado em todos os lugares onde o método **OpenRecordset** for utilizado, pois neste caso o método falhará e você tem que estar preparado para interceptar o erro gerado.

Que tal provarmos tudo isso ? Então vamos lá...

Vejá na seção **REDES** os seguintes exemplos:

- Acesso Exclusivo/Compartilhado.

- Acesso as tabelas - dbDenyRead, dbDenyWrite e dbReadOnly

3.3 Bloqueio de Registros

Se sua aplicação for rodar em um ambiente multiusuário mais cedo ou mais tarde alguém tentará atualizar um registro que está bloqueado por outro usuário ou outro usuário tentará modificar um registro que você está atualizando. E aí...?

Antes de fazer um pedido para editar um registro já existente ou incluir um novo registro, o Jet irá verificar se nenhum outro usuário está tentando modificar o registro ou acrescentando um registro novo, para isso o Jet utiliza a técnica de bloqueio de registros.

Mas atenção, o jet não aceita o bloqueio de registros, na verdade o que ocorre é o bloqueio da página ou páginas contendo o registro. Como uma página contém 2K (2048 bytes), o bloqueio será efetuado não somente no registro desejado mas também nos adjacentes, a não ser que o registro tenha exatamente o tamanho de 2048 bytes.

O jet trabalha então com dois métodos de bloqueios : *o pessimista* e *o otimista*. Vejamos cada um deles:

1-Bloqueio pessimista(O padrão usado pelo VB)

Executa o bloqueio da página que contém o registro no momento que sua aplicação executar o método **Edit**

O registro permanecerá bloqueado até ser executado o método **Update** e os dados forem gravados no arquivo.

Perceba que enquanto você estiver editando o registro outros usuários não poderão alterá-lo. Se alguém tentar bloquear a página que contém o registro o Jet retornará um erro.

A desvantagem do bloqueio pessimista é que se você bloquear o registro para edição por um período longo, ninguém mais conseguiria editar o mesmo registro e os outros registros da mesma página além de afetar o desempenho do sistema.

2-Bloqueio otimista

Executa o bloqueio da página contendo um determinado registro somente quando for invocado o método **Update**.

O bloqueio é imediatamente liberado quando se completa a operação de atualização.

A vantagem é que a página é bloqueada por um curto período de tempo.

A desvantagem é a possibilidade de outro usuário alterar os dados contidos no registro entre os instantes em que são empregados os métodos **Edit** e **Update**, se isto ocorrer o Jet retorna um erro.

Com isto em mente você deve levar em conta algumas considerações ao utilizar o bloqueio de registros. Vejamos:

1- O que fazer caso não seja possível bloquear um registro.

2-Como evitar que o usuário mantenha um registro bloqueado por muito tempo.

3-Determinar qual tipo de bloqueio usar: *pessimista* ou *otimista*

No primeiro caso temos os conflitos de bloqueio, ou seja, se um usuário bloqueou um registro e outro usuário requisitou um bloqueio no mesmo registro ocorrerá um erro.

Como o VB não fornece uma maneira de verificar se um registro está bloqueado, você tem que estar preparado para os erros de bloqueio implementando um código de tratamento de erros em todos os lugares onde irá acrescentar ou atualizar os registros.

Dentre os inúmeros erros que o Jet retorna em ambiente multiusuário existem 3 questões muito importantes e delas você deve se proteger, vejamos :

Erro 3167

O registro foi deletado

Lembre-se que as modificações(alterações,exclusões,etc..) feitas em um registro somente serão refetidas em um dynaset quando o VB reler a página que contém o registro.

Neste caso o erro ocorre quando você tentar executar o método **Edit** em um registro que já foi deletado por outro usuário desde a última vez que você leu a página.

Erro 3197

Os dados foram modificados; a operação foi interrompida. Ocorre quando você tenta editar um registro que sofreu modificações por outro usuário desde a última vez que você leu a página

Erro 3260

Impossível atualizar; atualmente bloqueado pelo usuário x na máquina y. Ocorre quando você tenta bloquear uma página que já está bloqueada por outro usuário.

Quanto ao método de bloqueio a utilizar, na maior parte dos aplicativos de banco de dados, o *bloqueio otimista* é a melhor opção, mas existirão situações em que você deverá usar o *bloqueio pessimista*.

Para estabelecer o método de bloqueio de registro, defina a propriedade **LockEdits** da tabela ou do dynaset com que está trabalhando. Assim:

1-Produzir bloqueio Pessimista - LockEdits = True

```
Dim rs as Recordset  
rs.LockEdits = True
```

2-Produzir bloqueio Otimista - LockEdits = False

```
Dim rs as Recordset  
rs.LockEdits = False
```

Como dissemos a liberação do bloqueio é automática, mas em aplicativos com intensa entrada de dados você pode precisar fazer pausas momentâneas no processamento do aplicativo para permitir ao Jet realizar o seu trabalho de manutenção; para isto utilizar o método **Idle**, da seguinte forma:

DbEngine.Idle dbFreeLocks

4) CONTROL DATA (Projeto Comentado)

Projeto inteiramente comentado utilizando os controles vinculados, seus métodos, eventos e propriedades. Aprenda a Criar Tabelas, Índices, Chaves Primárias, Incluir, Alterar, Excluir, Validar e Localizar registros. Construa uma aplicação com poucas linhas de código. Considerações sobre o uso do Data Control e comparação com a DAO e SQL.

4.1 Acesso a Base de Dados - Data Control

Você quer implantar um cadastro de Clientes em sua empresa, e, deseja armazenar as seguintes informações: Código, Nome, Endereço, Cidade, Cep, UF e Telefone para cada cliente.

Feita a análise você chegou as seguintes conclusões:

- A informação código do cliente deve ser única para cada cliente, ou seja, não poderá haver duplicidade no código quando da importação dos dados.
- As informações referentes a código, nome, endereço, cidade, cep e uf são obrigatórias.
- sistema deve permitir incluir, excluir e localizar dados nos arquivos.
- sistema será desenvolvido em Visual Basic e usará o Controle de dados Vinculados para gerenciar as informações

Podemos agora dividir o projeto em três fases distintas :

1) Definição do banco de dados e tabelas.

Definindo banco de dados, tabelas, campos e registros.

Só para lembrar, você quer implantar um cadastro de Clientes em sua empresa e deseja armazenar as informações: Código, Nome, Endereço, cidade, cep, UF e Telefone para cada cliente. Então...

Podemos dizer que cada uma dessas informações serão os campos de seu arquivo de dados e, juntos, para cada cliente, irão compor o que chamamos de um registro de seu arquivo.

Logo você terá um arquivo com 7 campos e tantos registros quantos forem os clientes que voce cadastrar.

Você deve armazenar o seu arquivo em um banco de dados, e dar um nome para ambos: para o arquivo e para o banco de dados.

Bem, vamos chamar o seu arquivo de tabela e dar a ela o nome de Clientes, e ao seu banco de dados dar o nome de Controle. Teremos então um banco de dados chamado Controle que contém uma tabela chamada Clientes que conterà as informações de todos os clientes cadastrados.

Obs: Os usuários do Clipper/Dbase, utilizam arquivos padrão dbf, e podem ficar confusos com esse conceito de banco de dados, pois o que acabamos de chamar de tabelas o Clipper/Dbase chama de banco de dados.

Nativamente o Visual Basic usa o mesmo tipo de banco de dados que o Microsoft Access, ou seja, os arquivos MDB, por isso você pode criar o seu banco de dados usando tanto o Visual Basic como o Microsoft Access.

1-Crie o seu banco de dados utilizando o Data Manager ou use o Microsoft Access e grave-o com o nome de **CONTROLE.MDB**.

2-Crie uma tabela com o nome de Clientes com a seguinte estrutura:

Nome do Campo	Tipo de Dados	Tamanho
CODIGO	LONG INTEGER(*)	03
NOME	CHARACTER	30
ENDereco	CHARACTER	30
CIDADE	CHARACTER	30
CEP	CHARACTER	08
UF	CHARACTER	02
TELEFONE	CHARACTER	11

(*)Como cada cliente deve ter um código único garantindo assim a exata identificação do mesmo, defina o tipo de dados para o campo código como LONG INTEGER, e ative o atributo Counter e a opção Required, dessa forma criamos um campo do tipo contador que o próprio sistema irá controlar a cada inclusão/exclusão.

3-Defina um índice para o campo código ativando as seguintes opções: Unique, Primary index, Required.

Isto assegura que teremos uma chave única para cada cliente cadastrado e nos fornece um índice que agiliza no processo de localização das informações.

2) Desenhar a interface da aplicação com o usuário

Desenhando a interface com o usuário

A tela principal de nossa aplicação deverá ter o seguinte aspecto:

figura 1.0

Podemos observar que no formulário Clientes temos:

- 1) Sete etiquetas(labels) e sete caixas de texto (Textbox)
- 2) Um objeto controle de dados(Data Control)
- 3) Cinco botões de comandos (CommandButton)

Vamos montar o formulário Clientes conforme descrito abaixo:

- 1) Inicie um novo projeto no Visual Basic.Grave o formulário Form1 como Clientes.
- 2) Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo :

Tabela 1.0 - Objetos e propriedades do formulário Clientes

Objeto	Propriedade	Configuração
Form	Name	Clientes
	Caption	"Cadastro de Clientes"
Data	Name	dtaCli

	Caption	"Total de clientes:"
	Connect	Access
	Databasename	"C:\CONTROLE\CONTROLE.MDB"(*)
	RecordSetType	0-Table
	RecordSource	"Clientes"
label	Name	label1
	Caption	"Código"
	Autosize	True
label	Name	label2
	Caption	"Nome"
	Autosize	True
label	Name	label3
	Caption	"Endereço"
	Autosize	True
label	Name	label4
	Caption	"Cidade"
	Autosize	True
label	Name	label5
	Caption	"UF"
	Autosize	True
label	Name	label6
	Caption	"Cep"
	Autosize	True
label	Name	label7
	Caption	"Telefone"
	Autosize	True
TextBox	Name	codigo
	DataField	"Codigo"
	DataSource	dtaCli
TextBox	Name	nome
	DataField	"Nome"
	DataSource	dtaCli
TextBox	Name	endereco
	DataField	"Endereco"
	DataSource	dtaCli
TextBox	Name	cidade
	DataField	"cidade"
	DataSource	dtaCli
TextBox	Name	uf
	DataField	"UF"
	DataSource	dtaCli
TextBox	Name	cep
	DataField	"Cep"
	DataSource	dtaCli
TextBox	Name	telefone
	DataField	"Telefone"
	DataSource	dtaCli
CommandButton	Name	incluir

	Caption	"&Incluir"
CommandButton	Name Caption	excluir "&Excluir"
CommandButton	Name Caption	localizar "&Localizar"
CommandButton	Name Caption	gravar "&Gravar"
CommandButton	Name Caption	sair "&Sair"

(*) O arquivo Controle.mdb deve estar no diretório CONTROLE.

3) Implementar o código da aplicação

Código Da Aplicação

Objeto Controle de Dados(Data Control)

Lembre-se que você está usando um banco de dados do Microsoft Access, então:
Para o objeto Data você deve configurar as propriedades como abaixo:

- 1- A propriedade Connect deve estar como: Access.
- 2- A propriedade Name do controle deve ser: DtaCli.
- 3- O DatabaseName é o nome do seu arquivo de banco de dados, no caso: Controle.mdb
- 4- O RecordSource poderá ser uma tabela ou uma instrução SQL, no seu caso, é a tabela Clientes do banco de dados Controle.mdb.
- 5- A propriedade RecordSetType deve ser definida como 0-Table.

Por isso criamos o índice com o campo código, para utilizar com o método Seek do Recordset Tabela. Se recordSetType fosse definido como *1-Dynaset* ou *2-Snapshot* não poderíamos usar o método **Seek** e **sim o FindFirst, FindNext, FindLast**, etc.

Obs. Você poderia ter incluído o código no evento Load do seu formulário configurando as propriedades discutidas acima da seguinte forma:

```
Sub Form_load
    dtacli.Datasource="C:\Controle\Controle.mdb"
    dtacli.Recordsource="Clientes"
    dtacli.RecordSetType= 0
End Sub
```

Caixas de Texto(TextBox)

Agora para cada Textbox você deve configurar as propriedades como segue:

- 1- A propriedade DataSource deve receber o nome do objeto Data: DtaCli
- 2- A propriedade DataField de cada TextBox deve ser vinculada ao respectivo campo da tabela Clientes. (Selecione na lista de campos).

Feito isto o Controle de dados, DtaCli fornecerá o vínculo entre o seu formulário e o Banco de dados através da propriedade DataField das Caixas de Texto(TextBox) e também as ferramentas para a navegação através do banco de dados.

Para movimentar-se pelos registros da tabela Clientes basta clicar nos botões do objeto DtaCli.

Código associado a cada botão de commando

Dica: - A palavra Recordset pode representar um objeto e uma propriedade.

Desta forma você pode atribuir um Recordset a uma variável objeto, da seguinte forma:

```
Set meusdados = Data1.Recordset
```

A seguir você pode se referir ao seu Recordset usando a variável objeto.(Não esqueça de defini-la previamente.)

No exemplo a seguir não usaremos esta notação.

Para inserir as linhas de código basta clicar duas vezes no botão correspondente do seu formulário.

1-Código para botão incluir dados:

```
Private Sub Incluir_Click()
    dtacli.Recordset.AddNew      'insere informações no buffer de cópia
                                preparando
                                para incluir um novo registro no recordset
    excluir.Enabled = False      'desabilita o botão excluir
    incluir.Enabled = False      'desabilita o botão incluir
    localizar.Enabled = False    'desabilita o botão localizar
    sair.Caption = "&Cancelar"    'Muda o nome do botão Sair para Cancelar
    gravar.Enabled = True        'Habilita o botão gravar
    nome.SetFocus                'põe o foco na caixa de texto nome
End Sub
```

Após clicar no botão Incluir, voce pode digitar os dados referentes a cada cliente e a seguir clicar no botão Gravar para gravar as informações na sua tabela Clientes ou cancelar o processo de inclusão.

2-Código do botão gravar dados:

```
Private Sub gravar_Click()
    If nome.Text = Empty Then
        MsgBox "Informe o nome do Cliente.", vbExclamation, "Gravar Clientes"
        nome.SetFocus
        Exit Sub
    End If
    If endereco.Text = Empty Then
        MsgBox "Informe o endereco do cliente.", vbExclamation, "Gravar Clientes"
        endereco.SetFocus
        Exit Sub
    End If
    If cidade.Text = Empty Then
        MsgBox "Informe a cidade do cliente.", vbExclamation, "Gravar Clientes"
        cidade.SetFocus
        Exit Sub
    End If
    If uf.Text = Empty Then
        MsgBox "Informe a UF do cliente.", vbExclamation, "Gravar Clientes"
        uf.SetFocus
        Exit Sub
    End If
    If cep.Text = Empty Then
        MsgBox "Informe o Cep do cliente.", vbExclamation, "Gravar Clientes"
        cep.SetFocus
        Exit Sub
    End If
    dtacli.UpdateRecord
    dtacli.Recordset.Bookmark = dtactl.Recordset.LastModified
End Sub
```

```
sair(4).Caption = "&Sair"
localizar.Enabled = True
End Sub
```

Na rotina de gravação verificamos se os dados que são obrigatórios, no nosso caso só o telefone não é, foram informados; então nome, endereço, cidade, uf, cep não podem ser vazios(empty).

Se qualquer deles não for informado é exibida a mensagem respectiva(msgbox), o foco retorna ao campo para preenchimento(setfocus) e a rotina de gravação é abandonada.(Exit Sub)

Se todos os campos obrigatórios foram informados o sistema acrescenta fisicamente o registro na tabela, através do método UpdateRecord. A utilização deste método, ao invés do método Update, tem a vantagem de não disparar os eventos Validate e Reposition do data control

A utilização da propriedade Bookmark do recordset apenas posiciona o ponteiro de registro no último registro que sofreu modificações.

A seguir restauramos o nome do botão Sair e habilitamos o botão localizar.

3-Código do botão Cancelar a inclusão de dados:

```
Private Sub Sair_Click()
    If sair.Caption = "&Cancelar" Then
        dtaccli.Recordset.CancelUpdate
        dtaccli.Recordset.MoveLast
        incluir.Enabled = True
        excluir.Enabled = True
        localizar.Enabled = True
        sair.Caption = "&Sair"
    Else
        If MsgBox("Quer sair do sistema ?", vbYesNo, "Sair do Sistema") = vbYes Then
            End
        Else
            Exit Sub
        End If
    End If
End Sub
```

Neste código verificamos se o nome do botão está como Cancelar, em caso positivo, cancelamos qualquer atualização pendente através do método CancelUpdate.

Em seguida movemos o ponteiro de registros para o último registro.

Finalmente habilitamos os botões Incluir, Excluir e Localizar e mudamos o nome do botão para Sair.

Agora, se o nome do botão não for Cancelar solicitamos a confirmação para sair do Sistema(End) ou abandonar a rotina(Exit Sub) e retornar ao sistema.

4-Código para o botão Excluir dados:

```
Private Sub excluir_Click()
    If MsgBox("Confirma Exclusão do cliente -> " & dtactl.Recordset![codigo], vb-
Question + vbYesNo, "Excluir Clientes") = vbYes Then
        dtaccli.Recordset.Delete
        dtaccli.Refresh
    End If
End Sub
```

Na rotina de exclusão, solicitamos a confirmação da exclusão, e, em caso afirmativo, usamos o método delete, que remove o registro do recordset e estabelece um valor nulo para o ponteiro de registro.

A seguir atualizamos o objeto data control (Refresh).

5-Código para o botão Localizar dados:

```

Private Sub localizar_Click()
    Dim criterio As long
    Dim marcador As variant

    marcador = dtaccli.Recordset.Bookmark
    dtaccli.Recordset.Index = "codigo" 'recordset é do tipo Table !!!

    criterio = InputBox$("Codigo do cliente a localizar: ", "Localizar Clientes")

    If criterio <> Empty Then
        dtaccli.Recordset.Seek "=", criterio
        If dtaccli.Recordset.NoMatch Then
            MsgBox "Cliente não localizado ! ", vbExclamation, "Localizar Clientes"
            dtaccli.Recordset.Bookmark = marcador
        End If
    Else
        dtaccli.Recordset.Bookmark = marcador
    End If
End Sub

```

Para a rotina de localização definimos a posição atual do registro ativo usando a propriedade bookmark

Definimos o índice ativo (codigo que foi criando anteriormente Solicitamos o código do cliente, armazenando o valor na variável critério.

Se (If) criterio for diferente (<>) de vazio (empty), então um valor foi fornecido e podemos iniciar a localização usando o método seek. Se não for informado nenhum valor para a variável criterio retornamos o ponteiro de registro para a posição anterior ao inicio da localização.

Se o valor não for encontrado (Nomatch=True) o sistema emite uma mensagem (msgbox), informando ao usuário e posiciona o ponteiro de registro na posição anterior a busca.

6-Código do evento Reposition:

```

Private Sub dtactl_Reposition() 'primeiro evento a ocorrer
    If dtaccli.Recordset.RecordCount < > 0 Then
        incluir(0).Enabled = True
        excluir(1).Enabled = True
        dtaccli.Caption = "Total de Clientes: " & dtactl.Recordset.RecordCount
    Else
        excluir(1).Enabled = False
        incluir(0).Enabled = True
        gravar(2).Enabled = False
        dtaccli.Caption = "O arquivo esta vazio"
    End If
End Sub

```

O evento Reposition ocorre quando o Data control move-se de um registro para outro. No código acima cada vez que o o evento Reposition é disparado atualizamos a propriedade Caption do controle de dados atribuindo a mesma o número total de registro do arquivo via propriedade RecordCount. Além disso habilitamos os botões incluir e excluir se houver registros no arquivo e desabilitamos os botões excluir e gravar se o arquivo estiver vazio.

7-Código do evento Validate:

```

Private Sub dtactl_Validate(Action As Integer, Save As Integer)
    If Save Then
        Select Case MsgBox("Deseja Salvar Alterações ?", vbQuestion + vbYesNo)
            Case vbYes '6
                If nome.Text = Empty Then
                    MsgBox "Nome do Cliente deve ser informado !",,, "Gravar Clientes"
                    nome.SetFocus
                    Save = False 'nao salva os dados
                    Action = vbDataActionCancel 'Cancela acao
                    Exit Sub
                ElseIf endereco.Text = Empty Then

```

```

        MsgBox "Endereco do Cliente deve ser informado !",, "Gravar Clientes"
        endereco.SetFocus
        Save = False
        Action = vbDataActionCancel
        Exit Sub
    ElseIf cidade.Text = Empty Then
        MsgBox "Por favor, informe a cidade do Cliente !",, "Gravar Clientes"
        cidade.SetFocus
        Save = False
        Action = vbDataActionCancel
        Exit Sub
    ElseIf cep.Text = Empty Then
        MsgBox "Cep do Cliente deve ser informado !",, "Gravar Clientes"
        cep.SetFocus
        Save = False
        Action = vbDataActionCancel
        Exit Sub
    ElseIf uf.Text = Empty Then
        MsgBox "Por favor, informe a UF do Cliente !",, "Gravar Clientes"
        uf.SetFocus
        Save = False
        Action = vbDataActionCancel
        Exit Sub
    End If
    Case vbNo '7
        Save = False
    End Select
End If
End Sub

```

O evento Validate é acionado quando o Data Control está para mover-se de um registro para um novo. Nesse meio tempo você pode cancelar ou não as mudanças feitas no registro. A sub rotina do evento Validate possui dois argumentos:

Action Descreve o evento que causou o evento Validate

Save Assume dois valores

True se qualquer dado for modificado

False se não houver modificação

Utilizamos o evento para validar os dados, cancelar a ação(vbDataActionCancel) e mudar o valor de Save quando necessário.

8-Código para tratamento de erros

```

Private Sub dtactl1_error(dataerr As Integer, response As Integer)
    'trata erros quando nenhum código esta sendo executado

    Select Case dataerr

        Case 3044 'caminho invalido
            MsgBox "O caminho informado não é valido, verifique !"
            End 'uma maneira muito rude de interromper sua aplicacao, mas...

        Case 3024 'nao achou banco de dados
            MsgBox "O arquivo definido não foi encontrado !"
            End

        Case Else
            'MsgBox "Erro em : " & Error$(dataerr) 'para mostrar o erro.
            response = vbDataErrcontinue 'ou vbdataErrdisplay

    End Select

End Sub

```

End Sub

A rotina acima é utilizado para interceptar erros que ocorrem quando nenhum código está sendo executado.

Estes erros podem ocorrer quando a carga do formulário ainda não estiver completa.

5) D.A.O (Projeto comentado)

Data Access Objet - Utilize os objetos de acesso a dados para desenvolver aplicações profissionais. Aprenda a criar Dynasets, Snapshots, Tables, incluir, alterar, excluir e a manipular os dados desses objetos via D.A.O. Projeto explicado passo a passo.

Estrutura da tabela.

Definicao da estrutura da tabela.

Vamos definir uma tabela com o nome de fornecedores que estará armazenada no banco de dados **Controle.mdb** e que possuirá a seguinte estrutura:

nome do campo	Tipo de Dados	Tamanho do Campo
nome	Character	30
cgc	Character	18
endereço	Character	30
cep	Character	09
uf	Character	02
ddd	Character	04
fone	Character	10
ramal	Character	04
fax	Character	10
contato	Character	20
produto	Character	20

1-Os campos Nome, CGC e Endereço não podem ser Nulos, ou seja são de preenchimento obrigatório.

2-Defina um índice para o campo nome desativando as opções: Unique, Primary index. e ativando a opção Requerid

Interface com o usuário

Desenhando a interface com o usuário.

Temos abaixo (figura 1.0) a tela principal de nossa aplicação:

Fig-1

Para montar o formulário acima descrito observe os seguintes passos:

1-Inicie um novo projeto no Visual Basic.Grave o formulário Form1 como Fornecedor.

2-Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo :

Tabela 1.0 - Objetos e propriedades do formulário Fornecedor

Objeto	Propriedade	Configuração
Form	Name Caption	Fornecedores "Cadastro de Fornecedor"
TextBox	Name Maxlength	Nome 30
MaskedTextBox	Name Mask PromptInclude PromptChar	CGC ##.###.###/###-## False " "
TextBox	Name Maxlength	Endereco 30
MaskedTextBox	Name Mask PromptInclude PromptChar	Cep #####-### False " "
TextBox	Name MaxLength	UF 2
TextBox	Name MaxLength	DDD 4
MaskedTextBox	Name Mask PromptInclude PromptChar	Fone ####-##-## False " "
TextBox	Name MaxLength	Ramal 4
MaskedTextBox	Name Mask PromptInclude PromptChar	Fax ####-##-## False " "
TextBox	Name MaxLength	Contato 20
TextBox	Name MaxLength	Produto 20
Frame	Caption Name	" " Frame1
CommandButton	Name Caption	Inclui "&Inclui"
CommandButton	Name Caption	Alterar "&Alterar"
CommandButton	Name	Excluir

	Caption	"&Exclui "
CommandButton	Name Caption	Grava "&Grava "
CommandButton	Name Caption	Cancela "&Cancela "
Frame	Caption Name	"Telefone/Contato/Produto" Frame2
(*) CommandButton	Name Caption	Command1(0) " < "
CommandButton	Name Caption	Command1(1) " < "
CommandButton	Name Caption	Command1(2) " > "
CommandButton	Name Caption	Command1(3) " > "
(**) Label	Caption AutoSize	** **

(*)Constituem um "control array" - Controles com o mesmo nome e do mesmo tipo, dotados de um índice identificador.

(**)Todos os controles Label possuem a propriedade AutoSize=True e Caption sendo igual ao nome do respectivo controle TextBox,MaskEdbox ou CommandButton.

Codigo da Aplicacao.

Codificando a sua aplicação.

Para inserir as linhas de código basta clicar duas vezes no controle correspondente do formulário.

1-Código da seção *General Declarations* do formulário

```
Private base As Database
Private tabela As Recordset
Private atualiza As Integer
```

Define as variáveis que serão visíveis em todo o formulário.

2-Código do evento Load do formulário.

```
Private Sub Form_Load()
    Dim dbname As String

    On Error GoTo loadererror

    dbname = "\controle.mdb"
    Set base = DBEngine.Workspaces(0).OpenDatabase(app.path & dbname)
    Set tabela = base.OpenRecordset("fornecedores", dbOpenTable)

    If tabela.RecordCount > 0 Then
        mostra_reg
    Else
        MsgBox "O arquivo está vazio ... ", vbExclamation
    End If
End Sub
```

```

        altera.Enabled = False
        exclui.Enabled = False
        grava.Enabled = False
        cancela.Enabled = False
    End If
    Exit Sub
loadererror:
    MsgBox Err.Description, vbCritical
End

End Sub

```

3-Código associado aos botões de comando para movimentar os registros.

```

Private Sub Command1_Click(Index As Integer)
    Const MOVE_FIRST = 0
    Const MOVE_PREVIOUS = 1
    Const MOVE_NEXT = 2
    Const MOVE_LAST = 3

    If (tabela.EditMode = dbEditAdd) Or _
        (tabela.EditMode = dbEditInProgress) Then
        cancela_Click
        Exit Sub
    End If
    Select Case Index
        Case MOVE_FIRST
            tabela.MoveFirst
        Case MOVE_PREVIOUS
            tabela.MovePrevious
            If tabela.BOF Then tabela.MoveFirst
        Case MOVE_NEXT
            tabela.MoveNext
            If tabela.EOF Then tabela.MoveLast
        Case MOVE_LAST
            tabela.MoveLast
    End Select
    mostra_reg
End Sub

```

4-Código associado ao botão incluir dados.

```

Private Sub inclui_Click()
    tabela.AddNew
    limpa_reg
    inclui.Enabled = False
    altera.Enabled = False
    grava.Enabled = True
    cancela.Enabled = True
    exclui.Enabled = False
    nome.SetFocus
End Sub

```

5-Código associado ao botão excluir dados.

```

Private Sub exclui_Click()
    If MsgBox("Confirma Exclusao ", vbYesNo, tabela![nome]) = vbYes Then
        tabela.Delete
        If Not tabela.EOF Then
            tabela.MoveNext
        ElseIf Not tabela.BOF Then
            tabela.MovePrevious
        End If
        mostra_reg
    End Sub

```

```
End If  
End Sub
```

6-Código associado ao botão Alterar dados.

```
Private Sub altera_Click()  
    tabela.Edit  
    altera.Enabled = False  
    grava.Enabled = True  
    cancela.Enabled = True  
    exclui.Enabled = False  
    inclui.Enabled = False  
    nome.SetFocus  
End Sub
```

7-Código associado ao botão Gravar dados.

```
Private Sub grava_Click()  
If (tabela.EditMode = dbEditAdd) Or _  
    (tabela.EditMode = dbEditInProgress) Then  
    atualiza = True  
    grava_reg  
    If atualiza Then  
        tabela.Update  
        inclui.Enabled = True  
        exclui.Enabled = True  
        altera.Enabled = True  
        grava.Enabled = True  
        cancela.Enabled = True  
    End If  
End If  
End Sub
```

8-Código associado ao botão Cancelar.

```
Private Sub cancela_Click()  
    Dim marca As Variant  
    marca = tabela.Bookmark  
    If (tabela.EditMode = dbEditAdd) Or _  
        (tabela.EditMode = dbEditInProgress) Then  
        tabela.CancelUpdate  
        tabela.Bookmark = marca  
        mostra_reg  
    End If  
    inclui.Enabled = True  
    exclui.Enabled = True  
    altera.Enabled = True  
    grava.Enabled = True  
    cancela.Enabled = True  
End Sub
```

9-Procedimento de evento para gravar os registros.

```
Public Sub grava_reg()  
    If nome = Empty Then  
        MsgBox "O nome é obrigatorio ! "  
        nome.SetFocus  
        atualiza = False  
        Exit Sub  
    End If  
    If cgc = Empty Then  
        MsgBox "O CGC tambem é obrigatorio ! "  
        cgc.SetFocus  
        atualiza = False
```

```

Exit Sub
End If
If endereco = Empty Then
    MsgBox "O endereco é obrigatorio "
    endereco.SetFocus
    atualiza = False
Exit Sub
End If
tabela![nome] = nome
tabela![cgc] = cgc
tabela![endereco] = endereco
tabela![cep] = IIf(IsNull(cep), "", cep)
tabela![uf] = IIf(IsNull(uf), "", uf)
tabela![ddd] = IIf(IsNull(ddd), "", ddd)
tabela![fone] = IIf(IsNull(fone), "", fone)
tabela![ramal] = IIf(IsNull(ramal), "", ramal)
tabela![fax] = IIf(IsNull(fax), "", fax)
tabela![contato] = IIf(IsNull(contato), "", contato)
tabela![produto] = IIf(IsNull(produto), "", produto)
End Sub

```

Dica: Poderíamos usar a seguinte notação abaixo para diminuir o código:

Ao invés de -> tabela![cep] = IIf(IsNull(cep), "", cep)

Fazemos -> tabela![cep] = "" & cep

ou -> tabela![valor_numérico] = 0 & [valor_numerico]

isto também evitaria a mensagem de erro para campos com **Null**.

10-Procedimento de Evento para mostrar os registros.

```

Public Sub mostra_reg()
    If Not IsNull(tabela![nome]) Then
        nome = tabela![nome]
    Else
        nome = ""
    End If
    If Not IsNull(tabela![cgc]) Then
        cgc = tabela![cgc]
    Else
        cgc = ""
    End If
    If Not IsNull(tabela![endereco]) Then
        endereco = tabela![endereco]
    Else
        endereco = ""
    End If
    If Not IsNull(tabela![cep]) Then
        cep = tabela![cep]
    Else
        cep = ""
    End If
    If Not IsNull(tabela![uf]) Then
        uf = tabela![uf]
    Else
        uf = ""
    End If
    If Not IsNull(tabela![ddd]) Then
        ddd = tabela![ddd]
    Else
        ddd = ""
    End If
End Sub

```

```

If Not IsNull(tabela![fone]) Then
    fone = tabela![fone]
Else
    fone = ""
End If
If Not IsNull(tabela![ramal]) Then
    ramal = tabela![ramal]
Else
    ramal = ""
End If
If Not IsNull(tabela![fax]) Then
    fax = tabela![fax]
Else
    fax = ""
End If
If Not IsNull(tabela![contato]) Then
    contato = tabela![contato]
Else
    contato = ""
End If
If Not IsNull(tabela![produto]) Then
    produto = tabela![produto]
Else
    produto = ""
End If

End Sub

```

11-Procedimento de Evento para limpar os controles .

```

Public Sub limpa_reg()
    nome = ""
    cgc = ""
    endereco = ""
    cep = ""
    uf = ""
    ddd = ""
    fone = ""
    ramal = ""
    fax = ""
    contato = ""
    produto = ""
End Sub

```

Dica: Se tivéssemos utilizado um 'control array' poderíamos ter usado um laço For/Next para diminuir o código.

```

Ex:   for x=0 to 5
        text1(x).text=""
    next

```

Ou, de forma mais elegante, poderíamos criar uma rotina genérica:

```

Public Sub LimpaControles(tela as Form)
    Dim i as integer

    For i=0 to tela.controls-1
        if TypeOf tela.Controls(i) is TextBox then
            tela.Controls(i).text=""
        endif
    Next

End Sub

```

11-Rotina associada a caixa de texto vinculada ao campo Ramal .

```
Private Sub ramal_KeyPress(KeyAscii As Integer)
    If KeyAscii < 48 Or KeyAscii > 57 Then KeyAscii = 0
End Sub
```

12-Rotina associada a caixa de texto vinculada ao campo UF .

```
Private Sub uf_KeyPress(KeyAscii As Integer)
    KeyAscii = Asc(UCase(Chr(KeyAscii)))
End Sub
```

13-Código associado a opção localizar do menu .

```
Private Sub mnulocaliza_Click()
    Dim marca As Variant
    Dim busca As String

    marca = tabela.Bookmark
    tabela.Index = "nome"

    busca = InputBox("Informe o nome do fornecedor : ", "Localiza")

    If busca = Empty Then
        Exit Sub
    Else
        tabela.Seek "=", busca
    End If

    If Not tabela.NoMatch Then
        mostra_reg
    Else
        MsgBox "Fornecedor não localizado ", vbExclamation, "Localiza"
        tabela.Bookmark = marca
    End If
End Sub
```

14-Código associado a opção Sair do menu.

```
Private Sub mnusair_Click()
    End
End Sub
```

15-Função para Validar o CGC.

Podemos implementar nosso sistema com uma função que valide o número do CGC do Cliente.

A função para validação pode ser colocada no evento **Lostfocus** do controle MaskedTextBox CGC chamando a função Calculacgc e passando como parâmetro o número do CGC digitado da seguinte forma:

```
Public Function ValidaCGC(CGC as string) as Boolean

if len(cgc) < > 14 then
    validacgc = False
    Exit function
Endif

if calculacgc(left(cgc,12)) <> mid(cgc,13,1) then
    validacgc=False
    Exit Function
Endif
```

```

if calculacgc(left(cgc,13)) <> mid(cgc,14,1) then
    validacgc=False
    Exit Function
Endif

validacgc=True

End Function

```

A função que faz o calculo do dígito verificador é a seguinte:

```

Public Function CalculaCGC(Numero as string) as string

dim i as integer
dim prod as integer
dim mult as integer
dim digito as integer

if not isnumeric(numero) then
    calculacgc=""
    Exit function
Endif

mult=2
for i=len(numero) to 1 step - 1
    prod=prod+ val(mid(numero),i,1)) * mult
    mult = iif(mult=9 , 2, mult+1)
next

digito= 11 - int(prod mod 11)
digito= iif(digito=10 or digito=11 , 0 , digito)

calculacgc=trim(str(digito))

End Function

```

6) SQL

Strutured Query Language - Otimize sua aplicação usando a SQL em seu código e agilize o acesso a seus dados. Aprenda a manipular as principais instruções da linguagem SQL de uma forma prática. Veja nesta aplicação como incluir, atualizar, excluir dados de suas tabelas usando a SQL. Deixa a SQL fazer todo o serviço.

Utilizando a SQL:

Estrutura da tabela.

Definicao da estrutura da tabela.

Vamos definir uma tabela com o nome de agenda que estará armazenada no banco de dados Controle.mdb e que possuirá a seguinte estrutura:

nome do campo	Tipo de Dados	Tamanho do Campo
sobrenome	Character	30
nome	Character	30
telefone	Character	13
nascimento	date	--
endereço	Character	30
cep	Character	10
uf	Character	02
e_mail	Character	80
país	Character	20
obs	Character	100
foto	Character	50

1-Os campos Sobrenome, nome e nascimento não podem ser Nulos, ou seja são de preenchimento obrigatório.

2-Defina um índice para o campo sobrenome desativando as opções: Unique, Primary index. e ativando a opção Requerid

3-Você deve criar e armazenar no banco de dados Controle.mdb as seguintes consultas SQL: (Para isso utilize o Microsoft Access ou o Data Manager)

1-Consulta Inclusão :

```
INSERT INTO agenda ( sobrenome, nome, telefone, nascimento, endereço, cep,
uf, e_mail, país, obs, foto ) SELECT parsobrenome AS Expr1, parnome AS Expr2,
partelefone AS Expr3, parnascimento AS Expr4, parendereço AS Expr5,
parcep AS Expr6, paruf AS Expr7, pare_mail AS Expr8, parpaís AS Expr9,
parobs AS Expr10, parfoto AS Expr11;
```

2-Consulta Atualização:

```
UPDATE Agenda SET Agenda.sobrenome = parsobrenome, Agenda.nome = parnome,
Agenda.telefone = partelefone, Agenda.nascimento = parnascimento,
Agenda.endereço = parendereço, Agenda.cep = parcep, Agenda.uf = paruf,
Agenda.e_mail = pare_mail, Agenda.país = parpaís, Agenda.obs = parobs,
Agenda.foto = parfoto WHERE ((Agenda.sobrenome=[parsobrenome]));
```

3-Consulta Exclusão:

```
DELETE * FROM agenda WHERE sobrenome=parsobrenome;
```


Interface com o usuário

Desenhando a interface com o usuário.

Temos abaixo (figura 1.0) a tela principal de nossa aplicação em tempo de projeto:

Observe o uso dos
Controles:

- Image
- CommomDialog
- CrystalReport
- ComboBox
- Picture

(Figura 1.0)

Para montar o formulário acima descrito observe os seguintes passos:

- 1-Inicie um novo projeto no Visual Basic.Grave o formulário Form1 como Agenda.
- 2-Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo:

Tabela 1.0 - Objetos e propriedades do formulário Agenda

Objeto	Propriedade	Configuração
Form	Name Caption	Agenda "Agenda"
TextBox	Name Maxlength	Sobrenome 30
textBox	Name Maxlength	Nome 30
TextBox	Name Maxlength	Telefone 15
TextBox	Name	Nascimento
TextBox	Name MaxLength	Endereco 30
TextBox	Name MaxLength	Cep 10

ComboBox	Name	cmbestados
TextBox	Name MaxLength	e_mail 80
ComboBox	Name	cmbpaíses
TextBox	Name MaxLength	foto 80
TextBox	Name Multiline	obs true
CommandButton	Name Caption	Incluir "&Incluir"
CommandButton	Name Caption	Alterar "&Alterar"
CommandButton	Name Caption	Excluir "&Excluir"
CommandButton	Name Caption	Imprime "Im&prime"
CommandButton(#)	Name Caption	Gravar "&Gravar"
CommandButton(#)	Name Caption	Cancelar "&Cancelar"
CommonDialog	Name	CommonDialog1
CrystalReport	Name	CrystalReport1
Picture	Name Visible BackColor	Picture1 True azul
Picture	Name Visible BackColor	Picture2 False azul
Image	Name Stretch	Image1 True
CommandButton	Name Caption	foto Inclui foto
CommandButton	Name Caption	primeiro "<"
CommandButton	Name Caption	anterior "<"
CommandButton	Name Caption	proximo ">"
CommandButton	Name Caption	ultimo "> "
(**)Label	Caption AutoSize	** **

(**) Todos os controles Label possuem a propriedade *AutoSize=True* e *Caption* sendo igual ao nome do respectivo controle TextBox, MaskEdbox ou CommandButton.

(#) Estes botões de comando são colocados no controle *picture2* e os demais no controle *picture1*.

Codigo da Aplicacao

Codificando a sua aplicação.

Para inserir as linhas de código basta clicar duas vezes no controle correspondente do formulário.

1-Código da seção *General Declarations* do formulário

```
Option Explicit
Public banco As Database
Public area As Workspace
Public tabela As Recordset
Public consulta As QueryDef
Public atualiza As Boolean
Public alterar As Boolean
Public fotos As String
Public marca As Variant
```

Define as variáveis que serão visíveis em todo o formulário.

2-Código do evento *Load* do formulário.

```
Private Sub Form_Load()
    Set area = DBEngine.Workspaces(0)
    Set banco = area.OpenDatabase("C:\vb16\CONTROLE.MDB")

    Set tabela = banco.OpenRecordset("Agenda", dbOpenTable)
    tabela.Index = "sobrenome"

    enche_combo cmbestados, "estados", "uf"
    enche_combo cmbpaíses, "países", "país"

    If tabela.RecordCount = 0 Then
        MsgBox "O arquivo esta vazio ... vamos começar a trabalhar..."
    Else
        tabela.MoveFirst
    End If

    mostra_reg
End Sub
```

3-Código da procedure para mostrar os registros.

```
Public Sub mostra_reg()
    Dim nomefoto As String
    If Not IsNull(tabela![sobrenome]) Then
        sobrenome = tabela![sobrenome]
    Else
        sobrenome = ""
    End If
    If Not IsNull(tabela![nome]) Then
        nome = tabela![nome]
    Else
        nome = ""
    End If
```

```

If Not IsNull(tabela![endereco]) Then
    endereco = tabela![endereco]
Else
    endereco = ""
End If
If Not IsNull(tabela![cep]) Then
    cep = tabela![cep]
Else
    cep = ""
End If
If Not IsNull(tabela![uf]) Then
    cmbestados.Text = tabela![uf]
Else
    cmbestados.Text = ""
End If
If Not IsNull(tabela![nascimento]) Then
    nascimento = tabela![nascimento]
Else
    nascimento = ""
End If
If Not IsNull(tabela![telefone]) Then
    telefone = tabela![telefone]
Else
    telefone = ""
End If
If Not IsNull(tabela![pais]) Then
    cmbpaises.Text = tabela![pais]
Else
    cmbpaises.Text = ""
End If
If Not IsNull(tabela![e_mail]) Then
    e_mail = tabela![e_mail]
Else
    e_mail = ""
End If
If Not IsNull(tabela![obs]) Then
    obs = tabela![obs]
Else
    obs = ""
End If
If Not IsNull(tabela![foto]) Then
    nomefoto = Trim(tabela![foto])
    Imagem1.Picture = LoadPicture(nomefoto)
Else
    nomefoto = ""
    Imagem1.Picture = LoadPicture("")
End If
Label11.Caption = "Reg.: " & tabela.RecordCount
End Sub

```

4-Código da procedure para preencher as caixas de combinação.

```

Private Sub enche_combo(combo As Control, data As String, campo As String)
    Dim arqtemp As Recordset

    combo.Clear
    Set arqtemp = banco.OpenRecordset(data, dbOpenSnapshot)
    Do Until arqtemp.EOF
        combo.AddItem arqtemp(campo)
        arqtemp.MoveNext
    Loop
    arqtemp.Close
    combo.ListIndex = 0
End Sub

```

5-Código da procedure para limpar as caixas de texto.

```

Sub limpa_reg(janela As Form)
    Dim i As Integer
    For i = 0 To janela.Controls.Count - 1
        If TypeOf janela.Controls(i) Is TextBox Then
            janela.Controls(i).Text = ""
        End If
    Next i
End Sub

```

6-Código associado ao botão Incluir.

```

Private Sub inclui_Click()
    alterar = False
    marca = tabela.Bookmark
    limpa_reg frmagenda
    foto.Visible = True
    Picture1.Visible = False
    Picture2.Visible = True
    Image1.Picture = LoadPicture("")
    sobrenome.SetFocus
End Sub

```

7-Código associado ao botão Alterar.

```

Private Sub altera_Click()
    alterar = True
    sobrenome.Enabled = False
    nome.SetFocus
    foto.Visible = True
    Picture2.Visible = True
    Picture1.Visible = False
End Sub

```

8-Código associado ao botão Excluir.

```

Private Sub exclui_Click()
    If MsgBox("Confirma Exclusao ", vbYesNo, tabela![nome]) = vbYes Then
        tabela.Delete
        If Not tabela.EOF Then
            tabela.MoveNext
        ElseIf Not tabela.BOF Then
            tabela.MovePrevious
        End If
        mostra_reg
    End If
End Sub

```

9-Código associado ao botão Gravar.

```

Private Sub grava_Click()
    atualiza = True
    If alterar Then
        grava_reg 2
        alterar = False
        sobrenome.Enabled = True
    Else
        grava_reg 1
    End If
    If atualiza Then
        foto.Visible = False
        Picture1.Visible = True
    End If
End Sub

```

```

        Picture2.Visible = False
    End If
End Sub

```

10-Código associado ao botão Cancelar.

```

Private Sub cancela_Click()
    tabela.Bookmark = marca
    mostra_reg
    foto.Visible = False
    Picture1.Visible = True
    Picture2.Visible = False
    sobrenome.Enabled = True
End Sub

```

11-Código associado ao botão Imprime.

```

Private Sub imprime_Click()
    CrystalReport1.Destination = 0
    CrystalReport1.ReportFileName = "c:\vb16\agenda.rpt"
    CrystalReport1.Action = 1
End Sub

```

12-Código associado ao botão Inclui Foto.

```

Private Sub foto_Click()
    Dim filter As String
    fotos = ""
    filter = "Arquivos BMP (*.BMP)|*.bmp|Todos os arqs.*"
    CommonDialog1.filter = filter
    CommonDialog1.DefaultExt = "BMP"
    CommonDialog1.ShowOpen
    fotos = CommonDialog1.filename
    If Not fotos = Empty Then
        Image1.Picture = LoadPicture(fotos)
    Else
        Image1.Picture = LoadPicture("")
    End If
End Sub

```

13-Código associado ao botão primeiro(<).

```

Private Sub primeiro_Click()
    tabela.MoveFirst
    mostra_reg
End Sub

```

14-Código associado ao botão anterior(<).

```

Private Sub anterior_click()
    tabela.MovePrevious
    If tabela.BOF Then
        tabela.MoveFirst
    End If
    mostra_reg
End Sub

```

15-Código associado ao botão proximo(>).

```

Private Sub proximo_click()
    tabela.MoveNext
    If tabela.EOF Then
        tabela.MoveLast
    End If

```

```

    mostra_reg
End Sub

```

16-Código associado ao botão ultimo(>|).

```

Private Sub ultimo_click()
    tabela.MoveLast
    mostra_reg
End Sub

```

16-Código da procedure para gravar os dados nos arquivos.

```

Public Sub grava_reg(operacao As Integer)
    If sobrenome = Empty Then
        MsgBox "O sobrenome/Apelido é obrigatorio ! "
        sobrenome.SetFocus
        atualiza = False
        Exit Sub
    End If
    If nome = Empty Then
        MsgBox "O nome tambem é obrigatorio ! "
        nome.SetFocus
        atualiza = False
        Exit Sub
    End If
    If endereco = Empty Then
        MsgBox "O endereco é obrigatorio "
        endereco.SetFocus
        atualiza = False
        Exit Sub
    End If
    If nascimento = Empty Then
        MsgBox "A data de nascimento é obrigatória "
        nascimento.SetFocus
        atualiza = False
        Exit Sub
    End If
    If operacao = 1 Then
        Set consulta = banco.QueryDefs("insagenda")
    Else
        Set consulta = banco.QueryDefs("upagenda")
    End If
    area.BeginTrans
    consulta("parsobrenome") = sobrenome
    consulta("parnome") = nome
    consulta("parendereco") = endereco
    consulta("parnascimento") = CDate(Format(nascimento, "d-m-yy"))
    consulta("parcep") = IIf(IsNull(cep), "", cep)
    consulta("paruf") = IIf(IsNull(cmbestados.Text), "", cmbestados.Text)
    consulta("parpais") = IIf(IsNull(cmbpaíses.Text), "", cmbpaíses.Text)
    consulta("partelefone") = IIf(IsNull(telefone), "", telefone)
    consulta("pare_mail") = IIf(IsNull(e_mail), "", e_mail)
    consulta("parobs") = IIf(IsNull(obs), "", obs)
    consulta("parfoto") = IIf(IsNull(fotos), "", fotos)
    consulta.Execute
    area.CommitTrans
End Sub

```

17-Código associado a opção sair do menu.

```

Private Sub mnusair_Click()
    End
End Sub

```

18-Código associado ao evento pressionar tecla da caixa de texto e mail.

```
Private Sub e_mail_KeyPress(KeyAscii As Integer)
    KeyAscii = Asc(LCase(Chr(KeyAscii)))
End Sub
```

19-Código associado ao evento perder o foco da caixa de texto sobrenome.

```
Private Sub sobrenome_LostFocus()
    'If sobrenome = Empty Then
    '    Exit Sub
    'End If
    tabela.Seek "=", sobrenome
    If tabela.NoMatch Then
        Exit Sub
    End If
    mostra_reg
    foto.Visible = False
    Picture2.Visible = False
    Picture1.Visible = True
End Sub
```

20-Código associado ao evento perder o foco da caixa de texto nascimento.

```
Private Sub nascimento_LostFocus()
    If Not IsDate(Format(nascimento, "dd-mm-yy")) Then
        MsgBox "Data invalida, formato deve ser dd-mm-yy - Ex: 02-07-78 "
        nascimento.SetFocus
    Else
        nascimento.Text = Format(nascimento, "dd-mm-yy")
    End If
End Sub
```


7) CRYSTAL REPORTS (Projeto Comentado)

Utilize o Crystal Reports para gerar seus relatórios e incorporá-los a sua aplicação. Veja como gerar relatórios utilizando os recursos do Crystal Reports.(controlar a ordem de impressão dos registros, trabalhar com variáveis fornecidas pelos usuários, selecionar os registros para impressão.)

Introdução:Gerando os seus relatórios com o Crystal Reports.

O **Crystal Reports** é o gerador de relatórios do Visual Basic e com ele desenhemos relatórios de nossos aplicativos. Embora possua um objeto chamado **Printer** para imprimir dados, sua utilização além de complexa é trabalhosa pois tudo deve ser codificado. Ao Contrário, o Crystal Reports utiliza uma interface gráfica a partir de onde podemos construir qualquer relatório que necessitamos.

Podemos iniciar o **Crystal Reports** através da opção **Report Designer...** do menu **Add-Ins** ou pelo ícone correspondente na pasta de trabalho do Visual Basic no Windows.

Junto com o Visual Basic 5.0 é distribuída a versão 4.6 do Crystal Reports.

Criando um novo relatório.

Vamos gerar um relatório baseado na tabela agenda que se encontra no banco de dados Controle . Nosso relatório deverá obedecer os seguintes parâmetros:

- 1- Campos a serem impressos : sobrenome , endereço e a data de nascimento.
- 2- O relatório deverá ser ordenado pelo campo sobrenome.
- 3- Deveremos permitir inicialmente a visualização do relatório para posterior impressão.
- 4- O nome do relatório será agenda.rpt.

Selecionando a opção New do menu File teremos a tela da figura 1.0 abaixo:

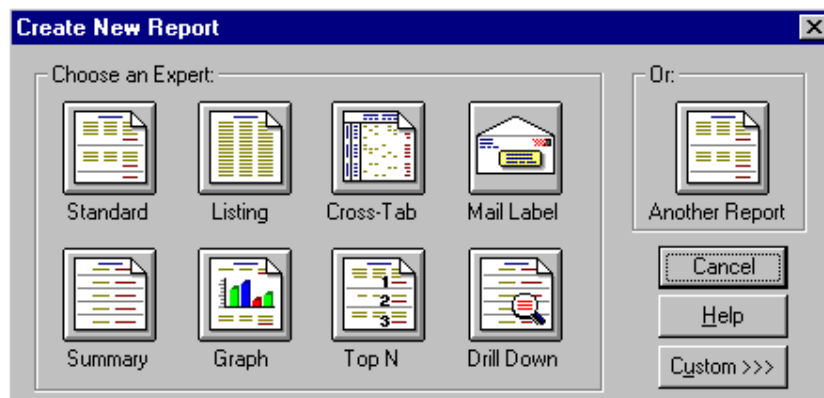


Fig-1

Após selecionar o botão Standard , devemos selecionar a base de dados na opção Data File , para o nosso caso Controle.mdb.

A seguir temos um lista de todas as tabelas e consultas gravadas na base de dados - exclua todos os elementos da lista, exceto a tabela Agenda e clique no botão Next para prosseguir.

Como nosso relatório esta baseado somente na tabela Agenda, o próximo passo Links, pode ser dispensado, portanto clique novamente no botão Next.

Vamos selecionar os campos da tabela que serão impressos no relatório - selecione cada campo e clique no botão Add.

Ao final devemos ter algo parecido com a figura 2.0 abaixo:

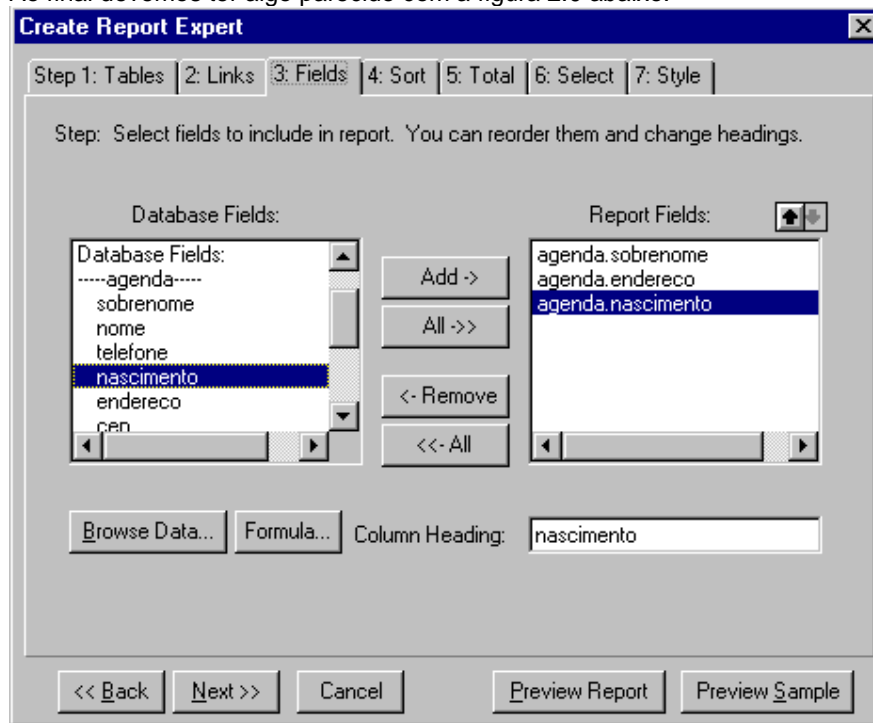


Fig-2

A esta altura o relatório está praticamente terminado, para visualizá-lo clique no botão **Preview Report**. É mostrada a tela da figura 3.0 onde após clicarmos na aba **Design** podemos notar cinco seções:

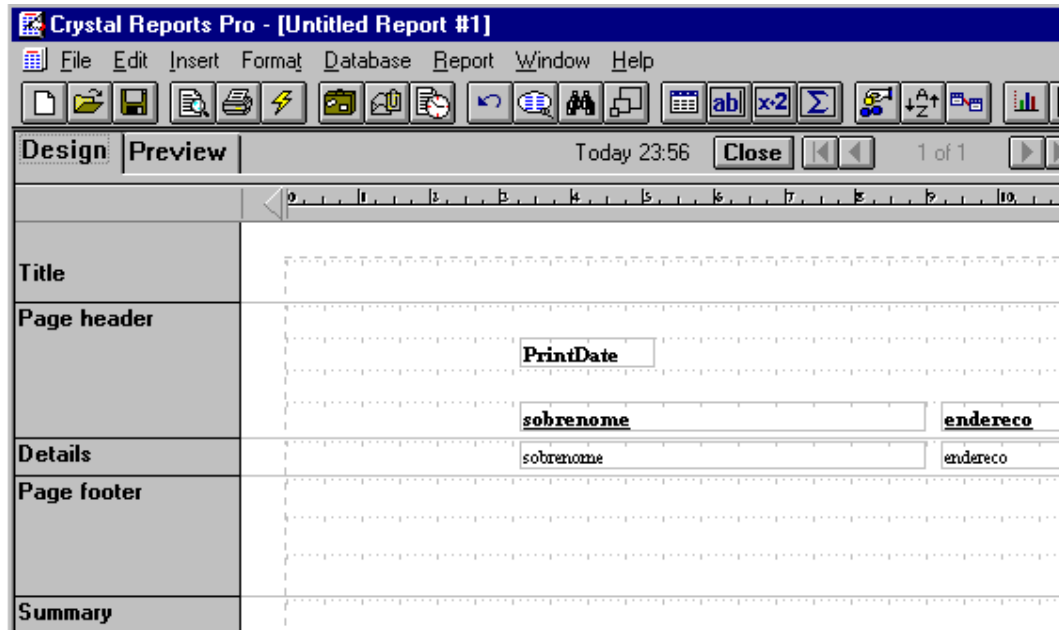


Fig-3

1 - Title

- para o título da aplicação.

- | | |
|------------------------|--|
| 2 - Page Header | - contém os elementos do cabeçalho da página. |
| 3 - Details | - contém os campos de dados a serem impressos. |
| 4 - Page Footer | - refere-se ao rodapé da página. |
| 5 - Summary | - impressão de resumos. |

Agrupando e ordenando registros.

Para agrupar registros, ordenando-os por uma determinada coluna selecione a opção **Group Section** do menu **Insert**. Agrupando pelo campo *sobrenome* em ordem ascendente veremos na tela a figura 4.0:

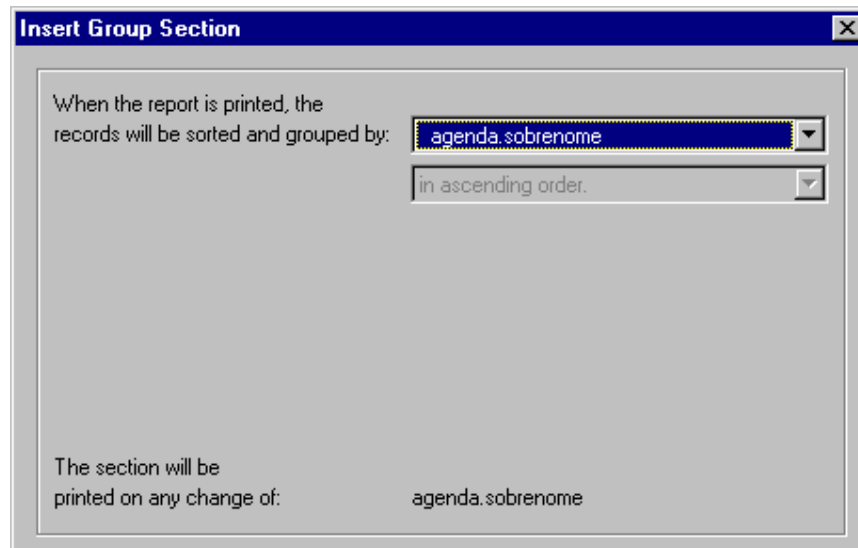


Fig-4

Inserindo títulos e Legendas.

Vamos inserir um título em nosso relatório e uma legenda para o campo *sobrenome*, para isso selecione a opção **Text Field...** do menu **Insert**.

Na caixa de diálogo **Enter Text** digite o título: *Agenda Pessoal* e clique no botão **Accept**. Ao lado do ponteiro do *mouse* acompanha um retângulo que você deverá posicionar no local desejado, ou seja, na seção **Title**.

Para criar a legenda **Nome** para o campo *sobrenome* no cabeçalho de grupo, selecione **Text Field...** novamente e digite **Nome**, clicando em **Accept** e posicionando a legenda mesmo local da legenda *sobrenome*.

Formatação de campos, campos especiais e desenho de linhas e caixas.

Para formatar campos basta selecionar o campo desejado e clicar na opção **Format** do menu ou clicando com o botão direito do *mouse* sobre o campo teremos um menu *pop-up* como na figura 5.0 abaixo:



Fig-5

Pelo menu podemos acessar as opções pertinentes a um determinado campo do relatório. Para o caso selecionamos o título **Agenda Pessoal**, vamos alterar a fonte (**Change Font...**) para 14 e o estilo para negrito.

Para alterar mais de um campo selecione-os mantendo a tecla **Shift** pressionada.

Aproveitando vamos inserir um campo referente a data no canto superior esquerdo.

Selecione a opção **Special Field...** do menu **Insert** escolha a opção **Print Date** e posicione no local indicado.

Clique com o botão direito do mouse sobre o campo nascimento e selecione a opção **Change Format...** escolhendo o formato **DMY** (dia-mês-ano) e clique OK.

Finalmente vamos desenhar um retângulo ao redor do título. Selecione a opção **Box** do Menu **Insert**, note que o ponteiro do *mouse* mudou para um lápis: Desenhe o retângulo ao redor do título mantendo o botão esquerdo do mouse pressionado.

Se quiser colorir o retângulo clique com o botão direito do mouse sobre o mesmo e preencha-o com uma cor de sua escolha.

Através do menu **Insert** podemos desenhar linhas, retângulos, quadrados, além de inserir figuras e até gráficos em nossos relatórios.

Podemos também usar os ícones correspondentes da barra de ferramentas como na figura 6.0 abaixo:



Fig-6

Trabalhando com fórmulas.

Vamos montar uma fórmula para imprimir o número da página no rodapé do relatório.

Para isso usamos o editor de fórmulas do Crystal Reports que pode ser disparado através do ícone



ou da opção **Formula Field...** do menu **Insert**. Após isso você deve informar o nome da fórmula no campo **Formula Name** para o nosso caso informe *"pagina"* e clique no botão OK.

O Editor é mostrado na figura 7.0 :

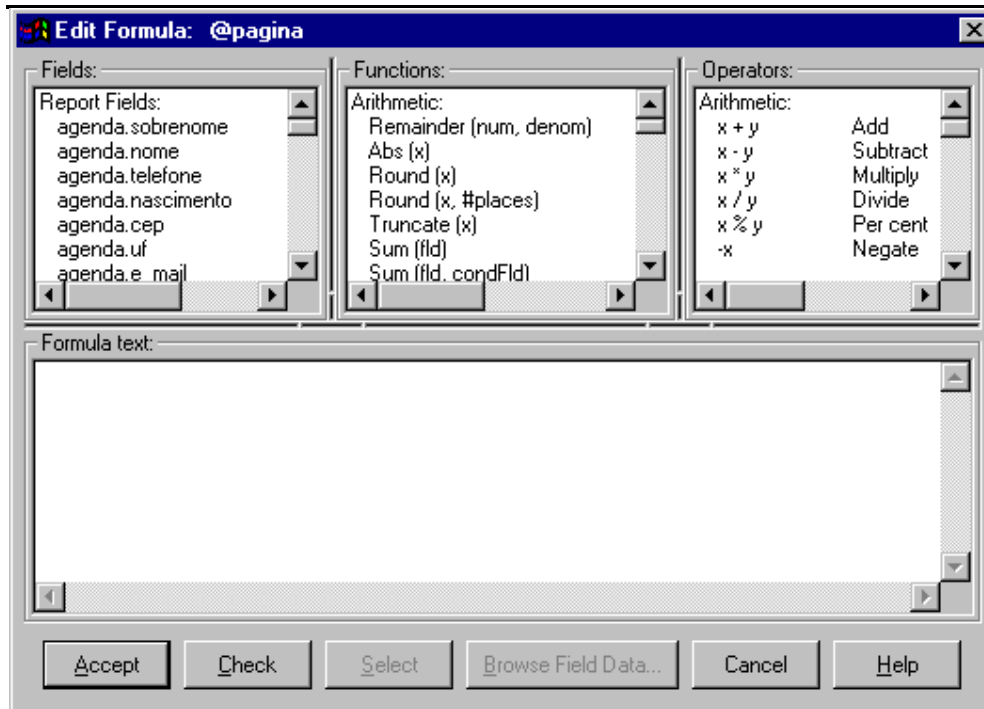


Fig-7

Agora basta digitar a fórmula na caixa **Formula Text**, vamos lá, digite: "Página: " + e, a seguir selecione a função **TrimLeft** na lista **Functions**;, ainda na lista **Functions**, selecione a função **ToText** e finalmente selecione o item **PageNumber** no final da lista **Functions**. Ao final deveremoster o seguinte na caixa **Formula Text**:

```
"Página: " + TrimLeft (ToText (PageNumber, 0 ) )
```

A função **PageNumber** retorna um valor numérico do número da página por isso usamos a função **ToText** para convertê-la em uma string , e a seguir usamos a função **TrimLeft** para removermos os espaços a direita.

Vamos verificar a fórmula clicando no botão **Check** , se tudo estiver correto o Crystal Reports informa com a mensagem **No errors found** indicando que a sintaxe está correta.

Agora basta clicar no botão **Accept** e posicionar a fórmula no canto esquerdo da seção **Page Footer**.

Encerrado o relatório basta salvá-lo através da opção **Save** do menu **File** e informar o nome para o relatório. (Nosso caso informa agenda).

Devemos ressaltar que a linguagem de fórmulas do Crystal Reports é diferente do Visual Basic, assim por exemplo, se usarmos a propriedade **SelectionFormula** do Crystal Reports permite definir condições para a impressão em nosso relatório de forma a imprimir somente os nomes iniciados pela letra "J" teríamos algo como:

```
CrystalReport1.SelectionFormula = "{AGENDA.NOME} >= " & "'" & "J" & "'"
```

Note que a referência a campos da tabela é feita entre chaves ({}).

Determinando o estilo e inserindo uma figura em seu relatório.

Você pode utilizar a guia **Style** para escolher a forma de apresentação do relatório.

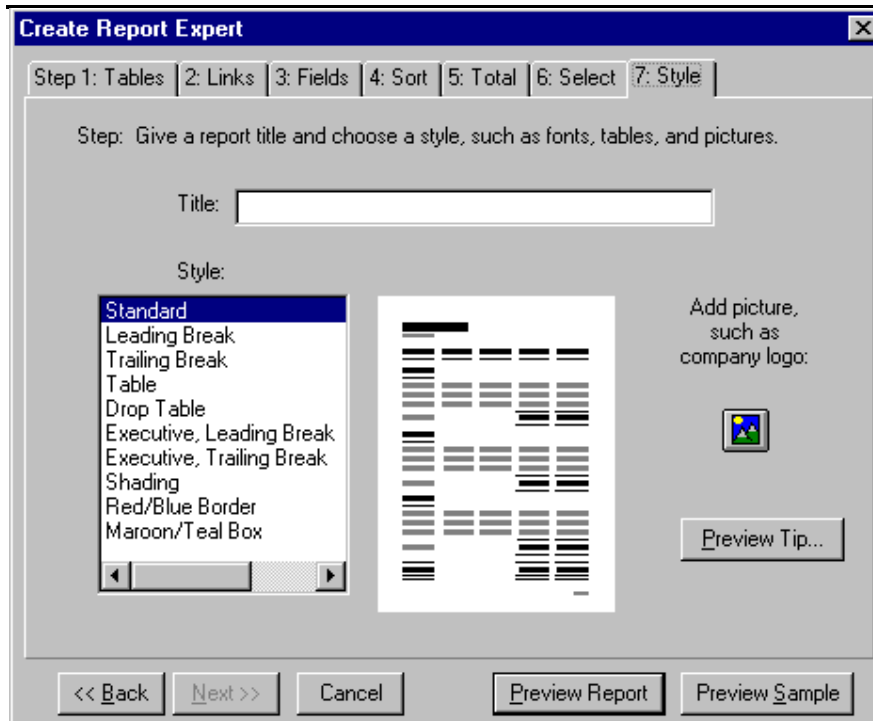


Fig-8

Para isto selecione um dos estilos na caixa de listagem *style* e veja á direita o jeitão do relatório.

Se quiser pode inserir uma figura no relatório clicando no botão **Add Picture...** com o ícone.

A caixa de texto **Title** lhe permite inserir o título para o relatório.

Imprimindo o relatório a partir de sua aplicação no Visual Basic.

Agora que nosso relatório esta pronto vamos associá-lo a nossa aplicação de forma a poder imprimí-lo a partir do Visual Basic.

Para isso devemos ativar o componente do Crystal Reports para a nossa aplicação na opção **Components..** do menu **Project** e a seguir selecionar o controle

Crystal Reports e copiá-lo para nosso formulário.

A seguir vamos definir algumas propriedades para o controle **CrystalReport1**

- | | |
|----------------------------|---|
| 1 - CopiesToPrinter | - Determina o número de cópias do relatório. Informe um (1) |
| 2 - Destination | - Direciona a impressão:
0-na tela 1-na impressora 2-em arquivo. Informe zero (0). |
| 3 - ReportFileName | - Indica a localização do relatório (arquivo .RPT) a ser impresso. |
| 4 - WindowTitle | - Título a janela Preview , informe Agenda . |
| 5 - SortFields | - Configura a ordem de classificação.(Ver abaixo.) |

Finalmente crie um botão de comando no formulário que irá disparar a impressão do relatório com o nome de **Imprime** e a seguir associe o seguinte código ao botão de comando:

```
Private Sub imprime_Click()
    CrystalReport1.Destination = 0
    CrystalReport1.ReportFileName = "c:\Controle\agenda.rpt"
    CrystalReport1.SortFields(0) = "+{Agenda.Sobrenome}"
```

```
CrystalReport1.Action = 1
End Sub
```

A propriedade **Action** definida para 1 dispara a impressão do relatório.

Outra forma de ordenar os registros via código é utilizar a propriedade **sortfields**. Na propriedade **Sortfields**, "{Agenda.Sobrenome}", indica que a ordem de impressão será por campo sobrenome (Agenda.sobrenome) e em ordem ascendente (+).

Relatório com dados de várias tabelas e Seleção de registros

O Crystal Reports é o gerador de relatórios que vem junto com o Visual Basic. Para uma introdução teórica já com um exemplo prático clique no link [Crystal](#).

Nesta seção iremos abordar exemplos práticos abordando as dúvidas mais comuns dos iniciantes. Vamos lá.

Veremos como gerar um relatório que contenha dados oriundos de diversas tabelas, e de como selecionar os registros que queremos imprimir.

Estaremos usando a versão do Crystal Reports que acompanha o Visual Basic 5.0.

Projeto Alunos - Relatório - BOLETIM ESCOLAR

Você trabalha como programador em uma grande escola e recebeu a incumbência de gerar o relatório bimestral para os alunos do 2º grau com os seguintes dados:

- Código do Aluno
- Nome do Aluno
- Data de Emissão do relatório
- Período, Curso, Série, Bimestre
- Nome da Matéria Cursada
- Nota do bimestre para cada aluno/matéria.

Só para lembrar você possui somente o Visual Basic 5.0. E vai ter que usar o Crystal Reports para gerar o relatório.

Ah!, esqueci de avisar, o relatório era para ontem...

- Vamos supor que temos três tabelas:

Dados Disciplinas/Notas	- TblNotas
Cadastro dos alunos	- TblAlunos
Cadastro de Disciplinas	- TblCursos
Cadastro Professores	- TblProfessor

A estrutura das tabelas é dada a seguir:

TblAlunos	TblCursos	TblNotas	TblProfessor
codaluno nome endereco telefone nascimento nomepai nomemae observacao Periodo serie numero	codcurso nomecurso codprofessor	codaluno codcurso nota ano bimestre Observacao	codprofessor nome endereco telefone

Obs : Codaluno, Codcurso, CodProfessor são chaves primárias de suas respectivas tabelas.

Já deu para perceber que os dados dos seus relatórios não estão todos em uma única tabela, e, você vai ter que juntar os dados para obter o relatório que deverá ter o seguinte layout:

Nome Escola		BOLETIM ESCOLAR		pag. n°
		Bimestre		
Nome do Aluno		cod. aluno	data emissão	
Período	Curso	Série	Número	
Nome das disciplinas	Nota	OBSERVAÇÕES		
.				
.				
.				

No espaço reservado para observações devemos informar quando o aluno não atingiu a média mínima para aprovação, nota inferior a 5, e colocar a mensagem:

"NOTA ABAIXO DA NOTA MÍNIMA - ALUNO EM RECUPERAÇÃO".

Além disso deveremos informar a média aritmética das notas do aluno no bimestre.

Bem, vamos ao trabalho, no VB ative o Crystal Reports através da opção **Add Ins->Report Designer...** ou crie um atalho na área de trabalho para chamar o Crystal Reports.

Selecionando a opção **New** do menu **File** teremos a tela da figura 1.0 abaixo:

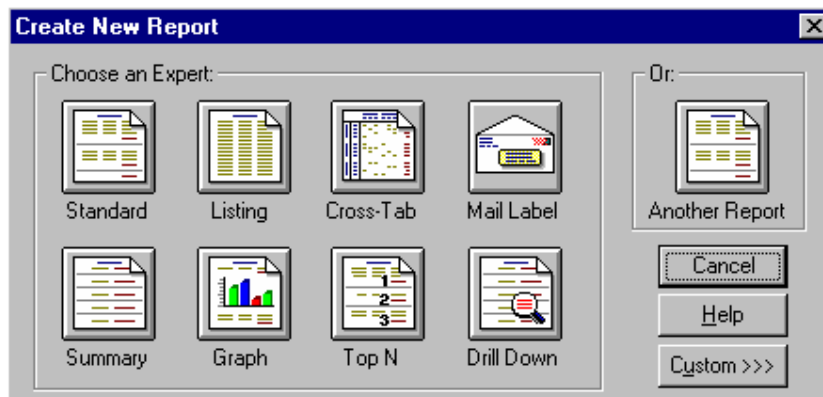


Fig. 1.0



Após clicar no botão **Standard**, devemos selecionar a base de dados na opção Data File, para o nosso caso criamos a base de dados chamada **Escola.mdb**.

Como você pode notar a tabela **Tblprofessor** não contém nenhum campo que iremos usar em nosso relatório, portanto exclua-a da lista e a seguir clique no botão **Next** para prosseguir.

A seguir temos um lista de todas as tabelas e consultas gravadas na base de dados que iremos utilizar (fig 2.0)

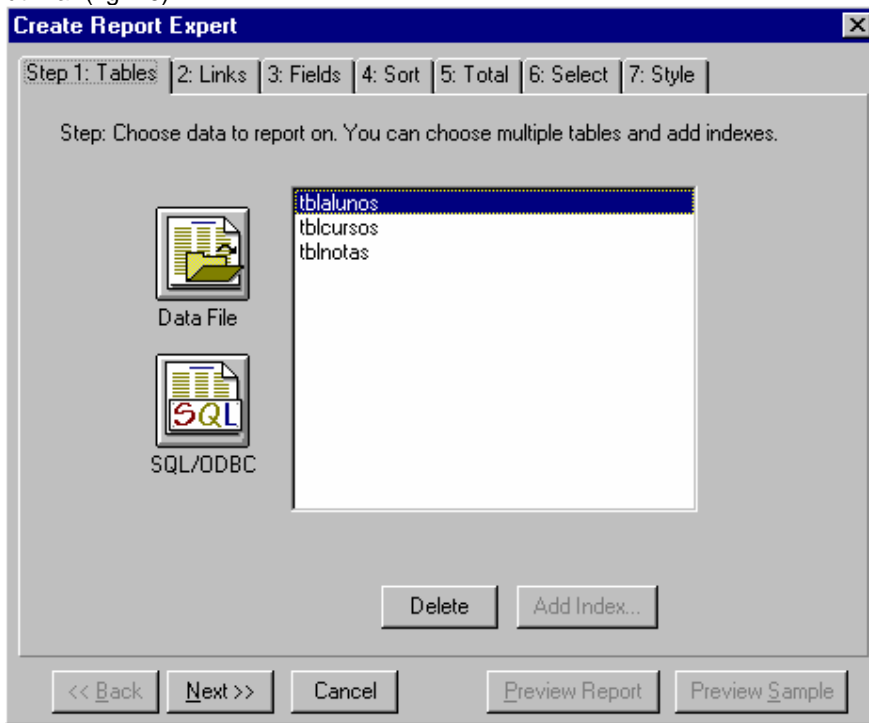


Fig. 2.0

No próximo passo, Links, é que esta o segredo para que os dados das três tabelas sejam incorporados ao nosso relatório e mantenham a correspondência entre os dados das mesmas:

O Crystal reconhece os relacionamentos entre as tabelas automaticamente, mas é possível criá-los nesta etapa, bastando clicar sobre o campo de uma tabela e arrastá-lo até o campo correspondente da outra tabela, após feito isto uma linha unindo os dois campos indica que o relacionamento foi criado.

Nesta etapa podemos também eliminar as associações existentes bem como admitir novas bases de dados ao relatório.

Naturalmente uma relação válida somente será efetivada entre campos indexados (pelo menos um) e do mesmo tipo.

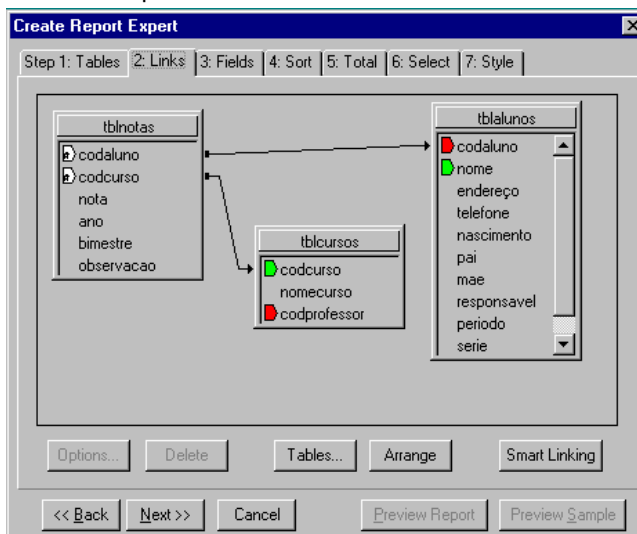


Fig. 3.0

Para verificar o relacionamento clique com o botão direito do mouse sobre uma das linhas e na opção options do menu suspenso.(fig 4.0)

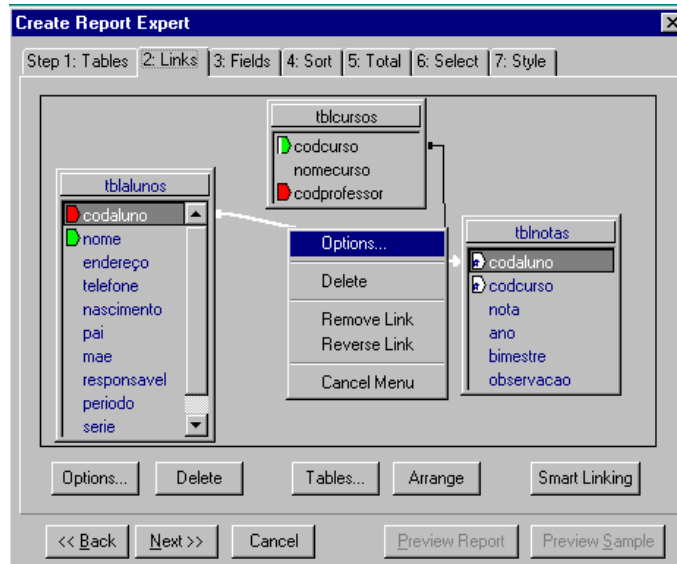


Fig. 4.0

A caixa de diálogo **Link Options** surge mostrando os detalhes dos vínculos (fig 5.0)

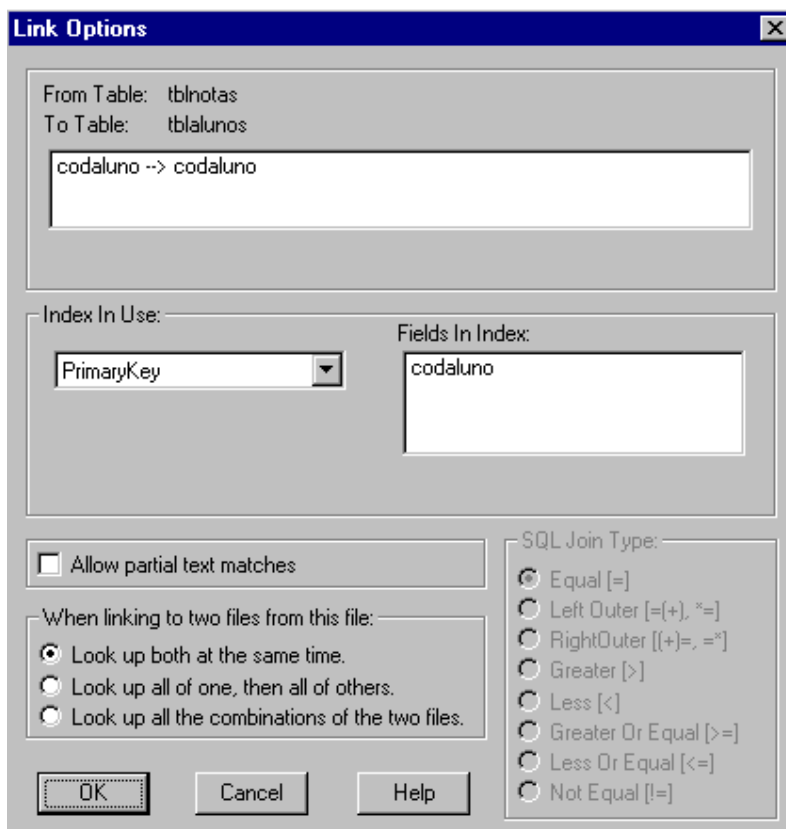


Fig. 5.0

Vamos selecionar os campos da tabela que serão impressos no relatório - selecione cada campo e clique no botão Add. Os campos do nosso relatório serão os seguintes:

Nome do Campo	Tabela de Origem
Codaluno	tblalunos
Nome	tblalunos
periodo	tblalunos
serie	tblalunos
numero	tblalunos
nomecurso	tblcursos
nota	tblnotas
Ano	tblnotas
bimestre	tblnotas

Ao final devemos ter algo parecido com a figura 6.0 abaixo:

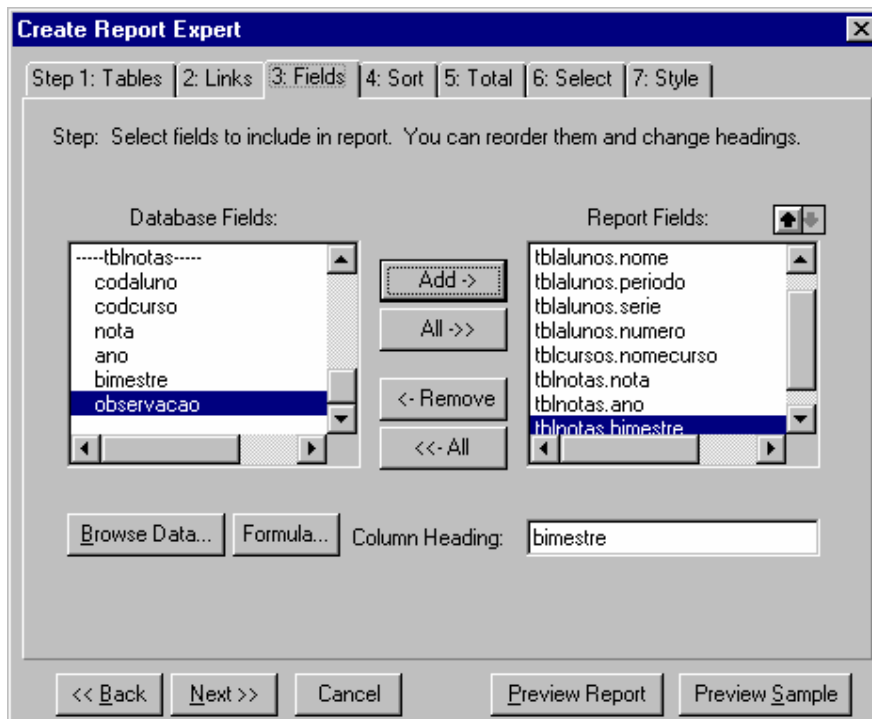


Fig. 6.0

Já podemos clicar no botão **Preview** para podermos visualizarmos nosso relatório(fig. 7.0)

26/07/98

luno	nome	periodo	serie	numero	nomecurso
8.001	MARIO SILVA BUENO	MANHA	1	12	MATEMÁTICA
8.002	ANA ROCHA SILVA	TARDE	2	23	PORTUGUES
8.001	MARIO SILVA BUENO	MANHA	1	12	PORTUGUES
8.003	JOAO FIGUEIREDO	MANHA	2	5	ED. FISICA
8.003	JOAO FIGUEIREDO	MANHA	2	5	FISICA
8.004	VERA LUCIA SOITE	TARDE	2	8	HISTORIA
8.001	MARIO SILVA BUENO	MANHA	1	12	ED. FISICA
8.005	BERNARDO LIMA	MANHA	1	10	MATEMÁTICA
8.005	BERNARDO LIMA	MANHA	1	10	PORTUGUES
8.006	CARLA PEREZ	MANHA	1	4	HISTORIA
8.006	CARLA PEREZ	MANHA	1	4	INGLES
8.001	MARIO SILVA BUENO	MANHA	1	12	INGLES
8.002	ANA ROCHA SILVA	TARDE	2	23	ED. FISICA

Fig. 7.0

Nada animador não é mesmo? Mas iremos ajustá-lo ao nosso lay-out.Clique na aba Design para podermos ajustar o lay-out do relatório.(fig. 8.0)

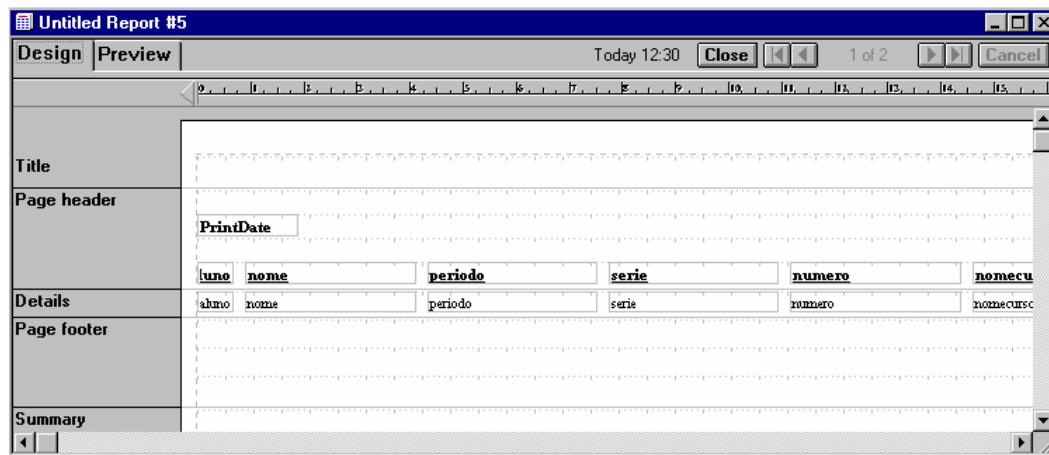


Fig. 8.0

Primeiro iremos inserir um **grupo** em nosso relatório, pois queremos agrupar os alunos por código. Para isso clique no menu **Insert** opção **Group Section**. e selecione a opção **tblalunos.codigo** (fig. 9.0) e clique em OK.

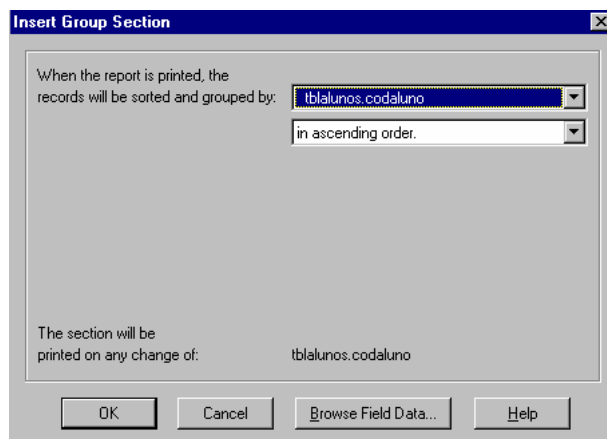


Fig. 9.0

Se você fez tudo certo obterá algo parecido com a figura 10 abaixo:

Fig. 10

Agora é só ajustar os campos conforme o lay-out da figura 11 abaixo:

Fig. 11

Para mover os campos clique sobre os mesmos e arraste-os para a nova posição.

Para inserir um texto selecione a opção do menu **Insert** e a seguir **Text Field**, digitando o texto desejado e posicionando-o no relatório.

Para formatar um campo clique com o botão direito do mouse sobre o campo, o menu da figura 12 surgirá com as várias opções de formatação.

Fig. 12

Iremos formatar a seção que inserimos de forma a obter os dados de cada aluno em páginas distintas, para isso clique com o botão direito do mouse sobre a seção e ative as opções mostradas na figura 13:

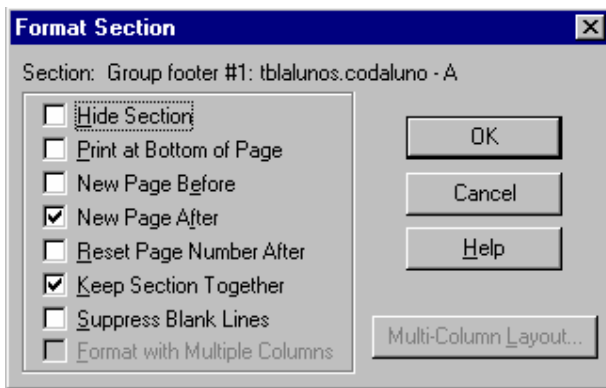


Fig. 13

Vamos agora inserir o campo que calculará a média aritmética das notas dos alunos.

Clique no campo **nota** e a seguir no botão com símbolo de somatória.



A seguir na janela da figura abaixo(figura 14) selecione a opção **average** e posicione o campo abaixo do campo **nota**.

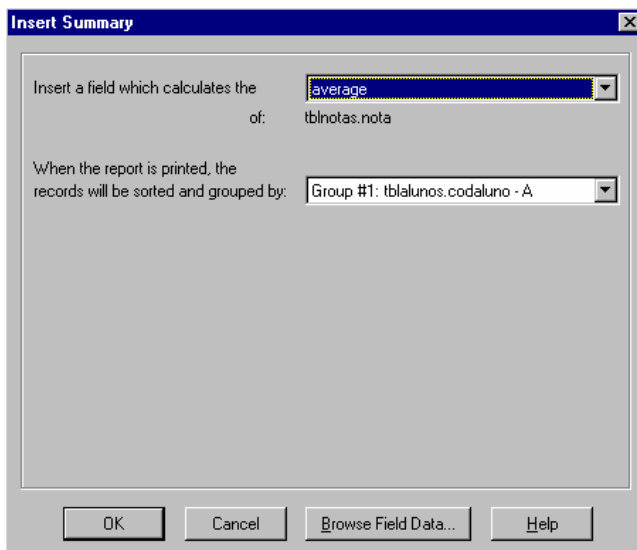


Fig. 14.

Deveremos também inserir uma fórmula que irá mostrar a mensagem no espaço observações quando a nota do aluno para a matéria for abaixo da nota mínima (5)

Para isso clique na opção do menu **Insert** e **Formula Field**, a seguir informe o nome da formula figura 15:

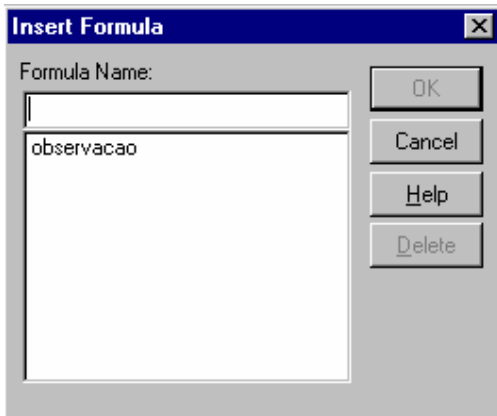


Fig. 15

Logo a seguir a janela da figura 16 será mostrada e iremos montar a fórmula nela mostrada:

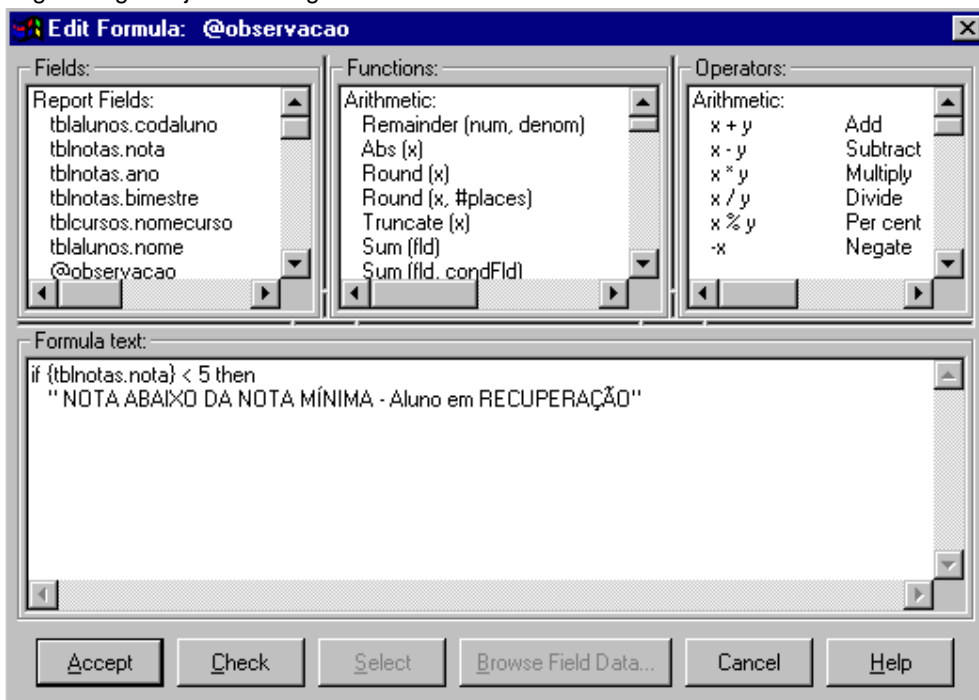


Fig. 16

Observe as janelas **Fields**, **Functions** e **Operators**. Para selecionar um de seus elementos basta clicar duas vezes sobre o mesmo.

Bem agora não esqueça de salvar o relatório, chamaremos o nosso de boletim.rpt.

Finalmente você pode clicar na aba **Preview** para ver o resultado final do seu trabalho (figura 17)

The screenshot shows the Crystal Reports Preview window. At the top, there's a title bar with 'Design' and 'Preview' tabs, and a status bar showing 'Today 23:40', 'Close', navigation buttons, '1 of 11', and 'Cancel'. The report content includes the school name 'COLÉGIO EXPERIMENTAL' in large blue letters. Below it, a box contains 'BOLETIM ESCOLAR 1998' and '1 bimestre'. The student's name 'MARIO SILVA BUENO' is listed, along with 'Codigo: 98.001', 'Emitido em: 26/07/98', and 'Número:'. A table follows with columns for 'Disciplina', 'Nota', and 'Observação'. The table lists subjects and their scores: MATEMATICA (5,00), PORTUGUES (7,00), INGLES (9,00), ED. FISICA (5,50), and FISICA (4,00). The average 'Média' is 6,10. A note at the bottom states 'NOTA ABAIXO DA NOTA MÍNIMA - Aluno em RECUPERAÇÃO'.

Disciplina	Nota	Observação
MATEMATICA	5,00	
PORTUGUES	7,00	
INGLES	9,00	
ED. FISICA	5,50	
FISICA	4,00	NOTA ABAIXO DA NOTA MÍNIMA - Aluno em RECUPERAÇÃO
Média :	6,10	

Fig.17

Obs.: Para desenhas as linhas e a caixa em torno do nome BOLETIM usamos os botões da barra de ferramentas do crystal (fig 18).

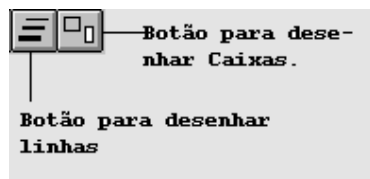


Fig. 18

Muito bem, e como ficaria o código para imprimir o relatório boletim.rpt no Visual Basic ? Que tal algo como:

(Aqui chamamos o nosso controle Crystal Reports de CR1)

```

cr1.WindowState = 2           'janela maximizada
cr1.WindowControlBox = True   'ativa os botões de controle da janela
cr1.ReportFileName = App.Path & "\boletim.rpt" 'caminho nome do relatório
cr1.Destination = 0           'imprime na tela
cr1.Action = 1                 'inicia impressão

```

Obs:

Não esqueça de ao gerar o seu relatório deixar ativa a opção **Verify on Every Print** do menu **Data-base** ;

Desative opção **Save Data with Closed Report** na opção **Database->Set Location...** clique no botão **Same as Report**.

8) ACESSO A BASE DE DADOS PADRÃO XBASE(DBASE/CLIPPER)

Aprenda como acessar arquivos DBF do Dbase/Clipper no Visual Basic . Crie sua aplicação em Visual Basic acessando os dados de seus arquivos DBF do Clipper/Dbase via Controle de dados ou via DAO. Veja exemplo detalhado.

Acesso utilizando o Controle de Dados Vinculados. (Data Control)

- Como acessar uma base de dados padrão XBase(Dbase,Fox,Clipper).

O Visual Basic utiliza o formato do banco de dados do Access, isto significa que os objetos:tabelas, consultas, índices, etc., são armazenados em um único arquivo MDB o qual é considerado como o banco de dados.

No Dbase,FoxPro e Clipper o banco de dados pode ser constituído de muitos arquivos; arquivos de dados(DBF), arquivos de índices(NDX ou NTX) , etc.

Esse formato de banco de dados é considerado um banco de dados externo do tipo ISAM.(Indexed Sequential Access Method)

Ao usar o Controle de Dados para acessar uma base de dados padrão Xbase você deverá fornecer as seguintes informações ao Visual Basic:

- | | | |
|------------------------------|---|---|
| 1 - Propriedade Connect | - | Tipo de arquivo: dBaseIII, DbaseIV, FoxPro 2.X, etc.
Ex: Connect: DbaseIII; |
| 2 - Propriedade Databasename | - | Caminho do diretório dos arquivos do banco de dados.
Ex: Databasename: C:\CONTROLE |
| 3 - Propriedade RecordSource | - | Nome do arquivo DBF que contém os registros que você quer acessar via controle de dados, sem a extensão DBF. EX: Recordsource: Agenda |

Obs:Para arquivos XBase com extensão diferente de DBF informe a extensão separada do nome do arquivo pelo caractere #. Ex:Recordsource: Agenda#dat.

- | | | |
|----------------|--|---|
| 4 - Finalmente | | você deverá informar ao Visual Basic qual DLL utilizar com o banco de dados denominado na propriedade Connect. Para os arquivos do formato Xbase faça uma entrada no arquivo INI da sua aplicação com a seguinte linha: |
|----------------|--|---|

Dbase III=XBS200.DLL

Obs: Se durante a instalação do VB você optou pela instalação dos arquivos ISAM, o Setup colocará automaticamente as entradas no arquivo VB.INI, mas lembre-se que ao distribuir a sua aplicação você deverá instalar um arquivo INI com o mesmo nome do seu arquivo executável no diretório Windows do usuário. (Ou informar o diretório usando a propriedade IniPath). Ex:DBEngine.IniPath=path)

Além disso verifique se o arquivo XBS200.DLL está presente no diretório System do Windows caso contrário você obterá a mensagem de erro: "Can't find installable ISAM".

Exemplo de arquivo INI:

```
[Installable ISAMs]
Dbase III=C:\WINDOWS\SYSTEM\XBS200.DLL
Dbase IV=C:\WINDOWS\SYSTEM\XBS200.DLL
FoxPro 2.0=C:\WINDOWS\SYSTEM\XBS200.DLL
FoxPro 2.5=C:\WINDOWS\SYSTEM\XBS200.DLL
Btrieve=C:\WINDOWS\SYSTEM\BTRV200.DLL
```

Além disso devemos incluir a seção abaixo no arquivo INI:

[Dbase ISAM]

Deleted=On

Isto evita que os registros deletados que ainda não foram removidos do arquivo apareçam nos seus dados. (A eliminação física requer a execução de procedimentos do próprio Dbase/Clipper.)

Finalizando, tenha em mente que o acesso a arquivos externos do tipo ISAM apresentam algumas restrições, tais como:

Não suportam os objetos: QueryDef, Relation, Container e Document

Não suportam os métodos: CompactDatabase , RepairDatabase, CreateQueryDef, OpenQueryDef e DeleteQueryDef

Não suportam a criação de índices.(Acesso via Data control). (Neste caso você pode controlar a ordem na qual os registros serão exibidos utilizando uma instrução SQL com propriedade Recordsource)

Desenhando a interface

Vamos supor que você quer acessar o arquivo agenda.dbf (Arquivo padrão Dbase) que está no diretório teste no drive c.(C:\teste) usando o controle de dados.

O arquivo agenda possui a seguinte estrutura:

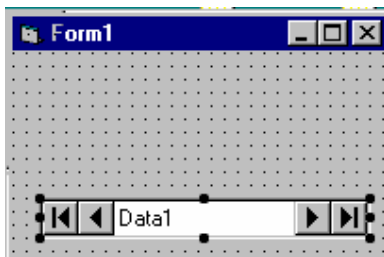
Nome do campo	Tipo	Tamanho
Nome	Character	30
Endereco	Character	30
Telefone	Character	11

Vamos montar o formulário Agenda conforme descrito abaixo:

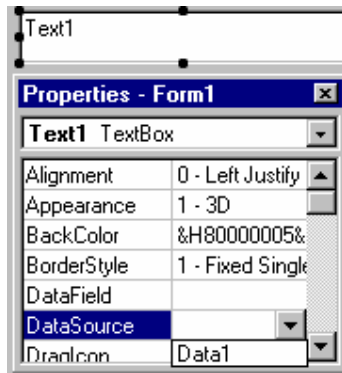
- Inicie um novo projeto no Visual Basic. Grave o formulário Form1 como Agenda, e a seguir insira o controle de dados como indicado abaixo:



De início selecione o objeto controle de dados na *Toolbox* do Visual Basic e acrescente-o ao seu formulário.(Fig.1)

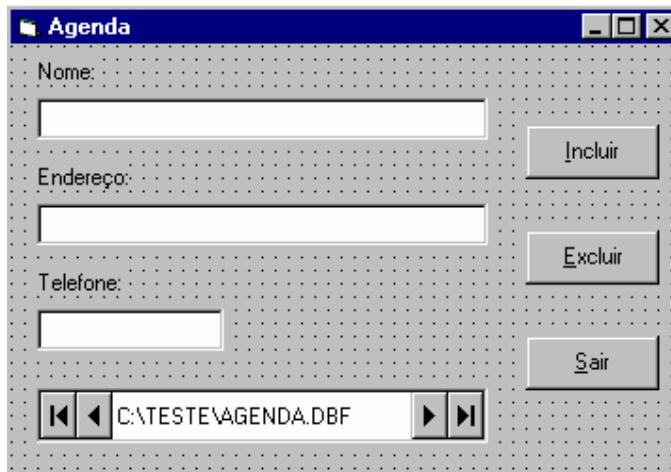


Defina a seguir as propriedades Name - Nome do Controle (O padrão é data1) e Caption - Especifica o nome que aparece no controle de dados.(O padrão é Data1) e dimensione o controle de dados. (Fig.2)



A seguir acrescente ao seu formulário um quadro de texto (TextBox) para cada campo que desejar acessar a partir do Recordsource, anexe cada quadro de texto ao objeto de controle de dados (DataSource) e especifique o campo que cada quadro de texto deverá exibir (DataField). (fig3.)

Ao final deveremos obter o formulário como mostrado na figura 4.0 abaixo:



Agora adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0:

Tabela 1.0 - Objetos e propriedades do formulário Clientes

Objeto	Propriedade	Configuração
Form	Name	Agenda
	Caption	"Agenda"
Data	Name	data1
	Caption	"C:\teste\Agenda.dbf"
	Connect	"dBase III;"
	Databasename	"C:\TESTE" (*)
	RecordSource	"AGENDA"
label	Name	label1
	Caption	"Nome"
	Autosize	True
label	Name	label2
	Caption	"Endereço"
	Autosize	True
label	Name	label3
	Caption	"Telefone"
	Autosize	True

TextBox	Name	TEXT1
	index	0 (**)
	DataField	"Nome"
	DataSource	data1
TextBox	Name	TEXT1
	index	1
	DataField	"Endereco"
	DataSource	data1
TextBox	Name	TEXT1
	index	2
	DataField	"telefone"
	DataSource	data1
CommandButton	Name	COMMAND1
	index	0 (***)
	Caption	"&Incluir"
CommandButton	Name	COMMAND1
	index	1
	Caption	"&Excluir"
CommandButton	Name	COMMAND1
	index	2
	Caption	"&Sair"

(*) Ao informar o diretório no qual os arquivos estão localizados o Visual Basic passa a considerá-lo como o banco de dados onde estão localizados as tabelas, os índices, etc.

(**) As caixas de texto são controladas por um vetor com um índice identificador, possuindo em comum o nome e o tipo.

(***) Os botões de comandos também são um control array como as caixas de texto.

Código do projeto.

Vamos usar o mínimo de código possível, apenas para ilustrar o funcionamento da conexão do motor Jet com o arquivo agenda.dbf.

1 - Código da seção de declarações do formulário

Option Explicit

Private rs As Recordset

2 - Código do evento Activate do Formulário.

```
Private Sub Form_Activate()
    Set rs = Data1.Recordset 'designamos o conjunto de registro
                             'á variável recordset rs .
                             '(isso melhora o desempenho)

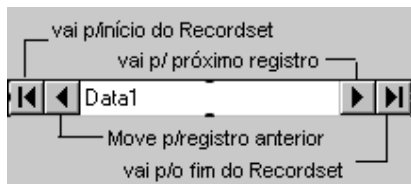
    If rs.RecordCount = 0 Then
        MsgBox "O arquivo vazio , inserir registro .", vbExclamation
        Command1_Click (0) 'botão p/incluir registro
    End If
End Sub
```

Como o controle de dados não pode lidar com um conjunto de registros vazio, se o arquivo estiver vazio ocorrerá uma mensagem de erro. Por isso o evento Activate verifica se a propriedade Recordcount indica que o arquivo está vazio, em caso positivo emitimos uma mensagem ao usuário (msg-box) e "forçamos" a inclusão de um registro no arquivo. (Command1_click(0)).

3 - Código associado aos botões Incluir , Excluir e Sair.

```
Private Sub Command1_Click(Index As Integer)
    Select Case Index
        Case 0 ' botão incluir registro
            rs.AddNew
            Text1(0).SetFocus 'foca caixa de texto vinculada ao nome
        Case 1 'botão excluir registro
            If MsgBox("Confirma exclusão deste registro ? " & rs!nome, _
                vbQuestion + vbYesNo + vbDefaultButton2) = vbYes Then
                rs.Delete 'exclui registro do arquivo
                rs.MoveNext
            If rs.EOF Then
                If rs.BOF Then
                    MsgBox "O arquivo vazio , inserir registro .", vbExclamation
                    Command1_Click (0) 'incluir registro
                Else
                    rs.MoveLast
                End If
            End If
        End If
        Case 2 'botão sair do sistema
            End
    End Select
End Sub
```

Obs: Para movimentar-se pelo arquivo use o controle de dados:



Note que não há necessidade de código para movimentar-se pelos registros.

Acesso utilizando a DAO. (Data Access Object)

Como acessar uma base de dados padrão XBase(Dbase,Fox,Clipper).

Para se conectar a arquivos do Dbase/Clipper usando a DAO você usará o método Openatabase para abrir a base de dados e o método Openrecordset para abrir o conjunto de registros que deseja acessar. Esses são os procedimentos normalmente usados para trabalhar com os arquivos do Access. A diferença esta na sintaxe utilizada, veja a seguir como fazer:

1 - Ao invés de informar o nome do arquivo você deverá informar o caminho do diretório de localização de seus arquivos DBF. (C:\TESTE)

2 - Você deverá informar também qual o tipo de arquivo ao qual você esta se conectando.(DbaseIII, DbaseIV, etc.) Veja abaixo como ficará o código para o nosso caso:

```
Dim db as Database
```

```
Set db=DBEngine.Workspaces(0).OpenDatabase ("C:\TESTE",false,false, "DbaseIII" )
```

3 - Finalmente utilize o método OpenRecordset e informe o nome do arquivo DBF, sem extensão, que você quer acessar. (AGENDA)

```
Dim rs as Recordset
Set rs= db.OpenRecordset("agenda", dbOpenTable )
```

Desenhando a interface

Utilizaremos o mesmo arquivo agenda.dbf que se encontra no diretório c:\teste.

- Inicie um novo projeto no Visual Basic.Grave o formulário Form1 como Agenda2, o qual deverá ter a seguinte aparência:

Fig-5

- Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 2.0:

Tabela 2.0 - Objetos e propriedades do formulário Agenda2

Objeto	Propriedade	Configuração
Form	Name Caption	Agenda "Agenda"
label	Name Caption Autosize	label1 "Nome" True
label	Name Caption Autosize	label2 "Endereço" True
label	Name Caption Autosize	label3 "Telefone" True
TextBox	Name index DataField DataSource	TEXT1 0 (**) "Nome" data1
TextBox	Name	TEXT1

	index	1
	DataField	"Endereco"
	DataSource	data1

TextBox	Name	TEXT1
	index	2
	DataField	"telefone"
	DataSource	data1

CommandButton	Name	COMMAND1
	index	0 (***)
	Caption	"&Incluir"

CommandButton	Name	COMMAND1
	index	1
	Caption	"&Excluir"

CommandButton	Name	COMMAND1
	index	2
	Caption	"&Sair"

CommandButton	Name	COMMAND1
	index	3
	Caption	"&Alterar"

CommandButton	Name	COMMAND1
	index	4
	Caption	"&Gravar"

CommandButton	Name	COMMAND2
	index	0 (****)
	Caption	"<<"

CommandButton	Name	COMMAND2
	index	1
	Caption	"<"

CommandButton	Name	COMMAND2
	index	2
	Caption	">"

CommandButton	Name	COMMAND2
	index	3
	Caption	">>"

(*) Ao informar o diretório no qual os arquivos estão localizados o Visual Basic passa a considerá-lo como o banco de dados onde estão localizados as tabelas, os índices, etc.

(**) As caixas de texto são controladas por um vetor com um índice identificador, possuindo em comum o nome e o tipo.

(***)(****) Os botões de comandos também são um control array como as caixas de texto.

Código do projeto.

Vejamos agora o código associado a cada evento ou botão de comando.

1 - Código da seção declarações gerais do formulário

```
Option Explicit
Private db As Database
Private rs As Recordset
```

2 - Código do evento Activate do formulário

```

Private Sub Form_Activate()
    If rs.EOF And rs.BOF Then
        MsgBox "O arquivo está vazio , vamos inserir um registro .", vbEx-
        clamation, "incluir registro"
        Command1_Click (0)
    End If
    mostra_reg
End Sub

```

3 - Código do evento Load do formulário

```

Private Sub Form_Load()
    'abre base de dados e o arquivo dbf a acessar
    Set db = DBEngine.Workspaces(0).OpenDatabase("C:\TESTE", False, False,
    "Dbase III;")
    Set rs = db.OpenRecordset("agenda", dbOpenTable)
End Sub

```

4 - Código dos botões de comandos de movimentação pelo arquivo

```

Private Sub Command2_Click(Index As Integer)
    Const MOVE_FIRST = 0
    Const MOVE_PREVIOUS = 1
    Const MOVE_NEXT = 2
    Const MOVE_LAST = 3
    Dim marca As Variant

    If (rs.EditMode = dbEditAdd) Or _
    (rs.EditMode = dbEditInProgress) Then
        rs.CancelUpdate
        Exit Sub
    End If
    Select Case Index
        Case MOVE_FIRST
            rs.MoveFirst
        Case MOVE_PREVIOUS
            If Not rs.BOF Then
                rs.MovePrevious
            If rs.BOF Then rs.MoveNext
            End If
        Case MOVE_NEXT
            If Not rs.EOF Then
                rs.MoveNext
            If rs.EOF Then rs.MovePrevious
            End If
        Case MOVE_LAST
            rs.MoveLast
    End Select
    mostra_reg
End Sub

```

5 - Código dos botões de comandos de Incluir, Alterar, Excluir, Gravar e Sair.

```

Private Sub Command1_Click(Index As Integer)
    Select Case Index
        Case 0 'incluir registro
            rs.AddNew

```



```

        limpa_regs
        Text1(0).SetFocus
    Case 1      'excluir registro
        If MsgBox("Confirma exclusão deste registro ? " & Chr(13) &
rs!nome, _
vbQuestion + vbYesNo + vbDefaultButton2) = vbYes Then
            rs.Delete 'exclui registro do arquivo
            rs.MoveNext
            If rs.EOF Then
                If rs.BOF Then
                    MsgBox "O arquivo está vazio , vamos inserir um registro
.", vbExclamation, "agenda"
                    Command1_Click (0)      'incluir registro
                Else
                    rs.MoveLast
                End If
            End If
        End If
    Case 2      'sair do sistema
        End
    Case 3      'alterar
        rs.Edit
        Text1(0).SetFocus
    Case 4      'gravar registros
        grava_regs
    End Select
    mostra_reg
End Sub

```

6 - Código da procedure para mostrar os registros nas caixas de texto.

```

Public Sub mostra_reg()
    Dim i As Integer
    For i = 0 To 2
        Text1(i) = IIf(Not IsNull(rs.Fields(i)), rs.Fields(i), "")
    Next i
    Text1(0).SetFocus
End Sub

```

7 - Código da procedure para gravar os registros no arquivo

```

Public Sub grava_regs()
    Dim i As Integer
    If rs.EditMode = dbEditAdd Or _
rs.EditMode = dbEditInProgress Then
        For i = 0 To 2
            rs.Fields(i) = Text1(i)
        Next i
        rs.Update 'atualiza registro
    End If
End Sub

```

8-Código da procedure para limpar o conteúdo das caixas de texto.

```

Public Sub limpa_regs()
    'limpa as caixas de texto
    Dim i As Integer
    For i = 0 To 2
        Text1(i) = ""
    Next
End Sub

```

Utilizando índices. (DbaseIII / Fox)

A grande vantagem da conexão com base externas via DAO é que você pode utilizar os índices para os formatos de arquivos DbaseIII e FoxPro.

Para isso você deverá fazer o seguinte:

1 - Com o NotePad ou outro editor de texto crie um arquivo com extensão INF com o mesmo nome do arquivo DBF que quer acessar.

2 - No arquivo INF crie uma seção chamada [Dbase III] e relacione os arquivos de índices que deseja utilizar da seguinte maneira:

[Dbase III] NDX1=NOME NDXNDX2=IDADE.NDX	Para o nosso exemplo, o nome do arquivo seria AGENDA.INF e teria o formato: [Dbase III] NDX1=NOME.NDX
---	---

3 - Altere o código do evento Load do formulário para:

```
Private Sub Form_Load()  
    'abre base de dados o arquivo dbf a acessar e ativa o índice NOME  
    Set db = DBEngine.Workspaces(0).OpenDatabase("C:\TESTE", False, False,  
"Dbase III;")  
    Set rs = db.OpenRecordset("agenda", dbOpenTable)  
    rs.index="NOME"  
End Sub
```

Note que o nome do índice é informado sem a extensão NDX.

Nota: O padrão de índices .NTX utilizados pelo Clipper não é suportado pelo Jet. Uma forma de resolver isto é utilizar filtros e ordenações.

Anexando arquivos DBF.

Dependendo da situação será mais produtivo anexar os arquivos DBF a um banco de dados padrão Access. Isto permite ao Visual Basic acessar os arquivos anexados como se fossem tabelas originais do Access. Vejamos como fazer isto:

1 - Se você possuir o Microsoft Access a tarefa é simples.

Abra o seu arquivo MDB.

Selecione a opção Arquivo->Anexar Tabela...

Na janela Anexar selecione o tipo de arquivo a anexar.(Dbase III)

Selecione agora o arquivo DBF a ser anexado e a seguir os índices utilizados.

Se tudo estiver certo o Access informa que o arquivo foi anexado com sucesso.

Na janela banco de dados o arquivo anexado é visualizado como:

♦dB AGENDA

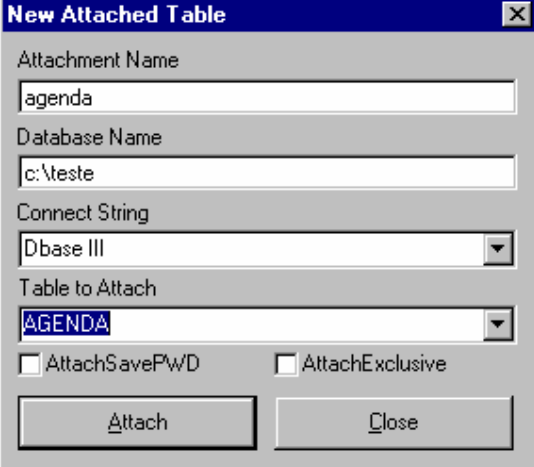
2 - No Visual Basic, usando o Data Manager faça o seguinte:

Abra o Data Manager e selecione a opção File-> Open Database

Selecione o nome do banco de dados da caixa de diálogo Open Database e clique em Open

Clique no botão Attached Tables e escolha o botão New

Na caixa de diálogo Attached tables preencha os campos da seguinte forma:



Attachment Name - Nome que deseja para a tabela a anexar.(Agenda)

Database Name - Diretório dos arquivos DBF a anexar.(C:\teste)

Connect String - Selecione o tipo de arquivo a anexar.(Dbase III)

Table to Attach - Nome do arquivo DBF a anexar.(Agenda.dbf)

Finalmente clique no botão Attach para anexar a tabela ao seu banco de dados.

As restrições para as tabelas anexadas são:

Você não pode forçar a integridade referencial entre as tabelas anexadas e as tabelas nativas

Você só pode abrir as tabelas anexadas como um Dynaset ou Snapshot mas não como Table

Você não pode alterar as propriedades: FieldSize, Validation Rule, DefaultValue e AllowZeroLength.

9) SETUP WIZARD

Veja como gerar os discos de distribuição e o programa de instalação para a sua aplicação usando o **SETUP WIZARD** . Que tal um programa de instalação para o seu sistema em português ?.

- **A questão da distribuição.**

Você acabou de desenvolver uma aplicação em Visual Basic , testou , depurou e, finalmente, está pronto para distribuí-la aos usuários finais.

Geralmente tais usuários não possuem o Visual Basic instalado em suas máquinas. (você deve sempre considerar essa situação como a padrão.)

Você terá então que distribuir com sua aplicação uma série de arquivos DLL, e, se sua aplicação usar controles personalizados, terá também que distribuir os arquivos de controle VBX/OCX que utiliza.

Para isto ou você cria um programa de instalação com o **Setup Wizard** que determina os arquivos que você precisa distribuir ou tenta determinar por si mesmo quais os arquivos precisam ser distribuídos.

Se você acha que pode fazer isso sem ajuda do **Setup Wizard**, e, só para você sentir a complexidade da tarefa, vamos tentar relacionar o que você terá que selecionar para distribuir:

1 - Arquivo executável (EXE) , de banco de dados (MDB) e os arquivos INI.

2 - Os arquivos DLL indispensáveis a qualquer aplicação:
(Considerando a versão 4.0 do Visual Basic 16/32)

- VB40016.DLL ou VB40032.DLL
- VB4EN16.DLL ou VB4EN32.DLL
- OC25.DLL
- OLE2.DLL
- VAEN2.DLL
- VAEN2.OLB

3 - Os arquivos DLL/OCX especificamente exigidos pela sua aplicação:
Ex.: Se você usar o objeto de acesso a dados - DAO e o Crystal Reports:

- DAO2516.DLL ou DAO2532.DLL
- CRYSTL16.OCX ou CRYSTL32.OCX

4 - Os arquivos dependentes de outros arquivos DLL já incluídos:
Ex: Se você usou a DAO (DAO2516.DLL) terá que incluir:

- VAEN2.OLB
- MSAJT200.DLL
- MSJETERR.DLL
- MSJETINT.DLL
- VBAJET.DLL
- VBDB16.DLL

Se você ainda não desistiu, realmente você é corajoso, vá em frente e, boa sorte, você vai precisar.

O **Setup Wizard** é fornecido com o Visual Basic e você poderá usá-lo para criar um programa que instalará sua aplicação na máquina do usuário.

As tarefas básicas que o **Setup Wizard** faz são:

- Construir o arquivo executável (EXE) do seu projeto.
- Criar um programa de instalação para a sua aplicação.
- Determinar os arquivos necessários para a aplicação.
- Compactar os arquivos do programa, copiá-los e dividi-los em discos para distribuição.(CD-ROM, Disco Rígido e Redes).
- Criar um grupo de programas e um ícone no sistema do usuário.

- Gerando o assistente de Instalação e os discos para distribuição.

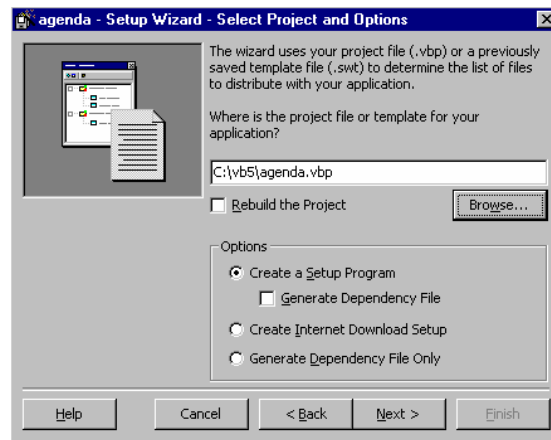
Vamos descrever passo a passo o processo de criação do programa de instalação e da geração dos discos de distribuição para um projeto: **agenda.vbp**.

Este projeto foi desenvolvido no artigo **SQL** e utiliza os controles **CrystalReports**, **CommomDialog**, **Image**, **Picture**, **ComboBox** além de usar a **DAO** e os arquivos de banco de dados padrão **MDB**.

Este projeto não utiliza recursos de **OLE** nem conexões do tipo **ISAM** ou **ODBC**.

Vamos supor que a distribuição será feita em discos de 3 1/2 (1.44), iremos usar o **Setup Wizard** da versão 5.0. Então vamos lá:

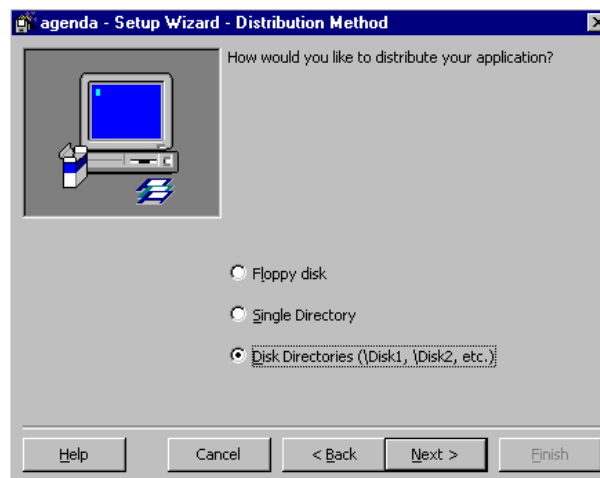
1 - Inicie o **Setup Wizard** no grupo de programas do Visual Basic, ícone **Application Setup**. Ao surgir tela inicial (Introduction) clique no botão **Next** e o **Setup Wizard** mostra a tela 1.0, abaixo:



Tela 01

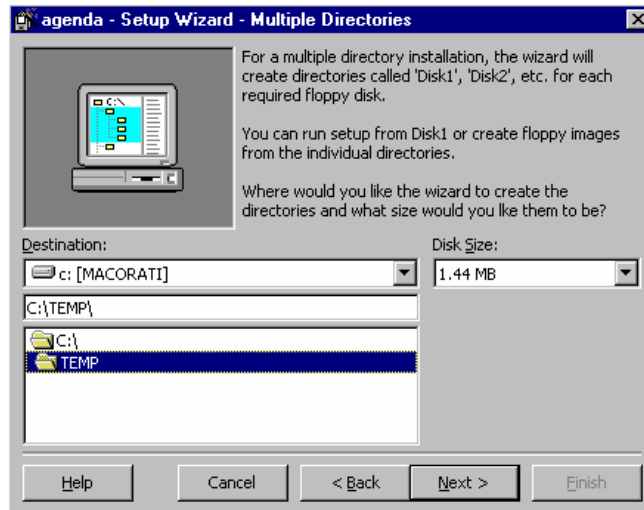
Nesta tela selecionamos a opção **Create a Setup Programa** e a seguir clicamos no botão **Browse...** para selecionar o projeto **agenda.vbp** que está no diretório **C:\VB5**. Vamos clicar em **Next** para ir para o próximo passo.

2 - Na tela 2.0, abaixo, selecionamos a opção **Disk Directories (\Disk1, \Disk2, etc.)** para gerar os arquivos em diretórios (**Disk1**, **Disk2**, etc.) no disco rígido, a seguir clicar no botão **Next** para continuar:



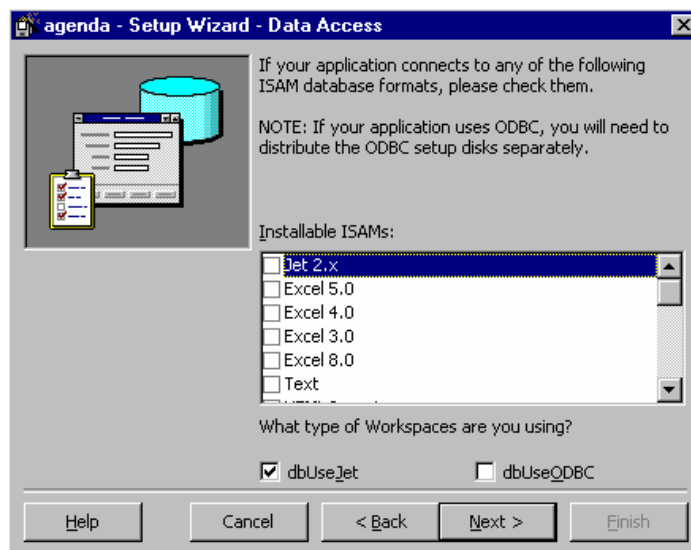
Tela 02

3 - Na tela 3.0, selecionamos o diretório (C:\TEMP) onde os diretórios serão criados e os arquivos copiados e também o tamanho dos discos (1.44) usados para distribuição.



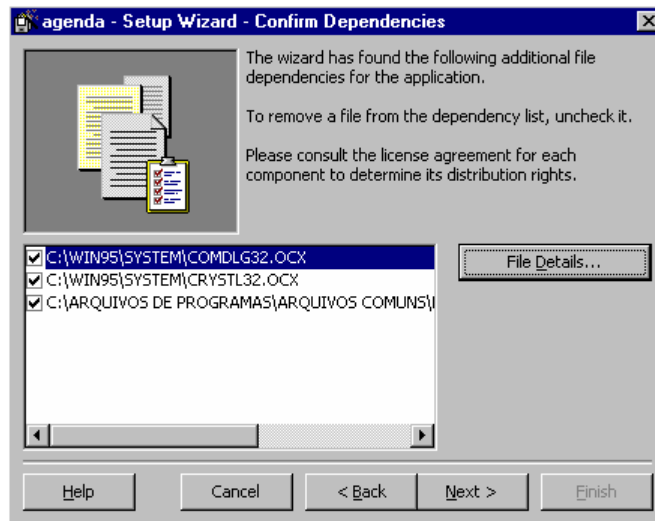
Tela 03

4 - Na tela 4.0, como nossa aplicação não usa drivers ODBC nem conexão via ISAM não selecionamos nada, apenas clicamos em Next para prosseguir.



Tela 04

5 - Ao surgir a próxima tela clicamos em Next pois nossa aplicação não usa controles ActiveX. A tela 5.0 informa os arquivos de controle utilizados por nossa aplicação e a DLL do arquivo DAO usado em nossa aplicação.



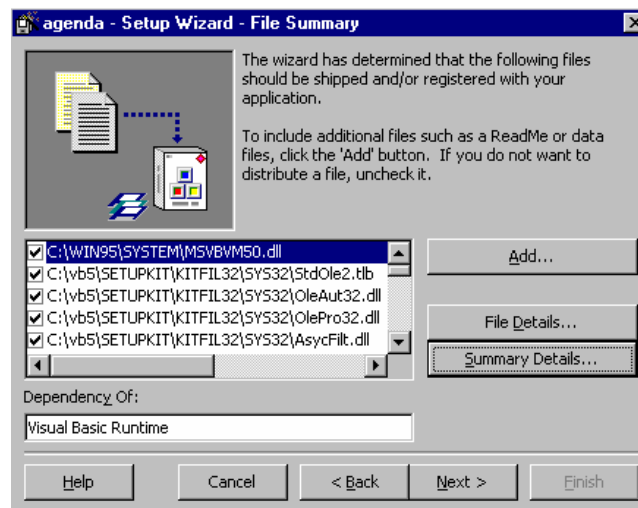
Tela 05

Para ver detalhes dos arquivos selecionados clique no botão **File Details...**

6 - Na tela 6.0, o **Setup Wizard** mostra todos os arquivos que nossa aplicação precisará para funcionar. Você pode remover qualquer arquivo que porventura já exista na máquina do usuário. Para distribuir o arquivo de dados clicamos no botão **Add...** e selecionamos o arquivo controle.mdb.

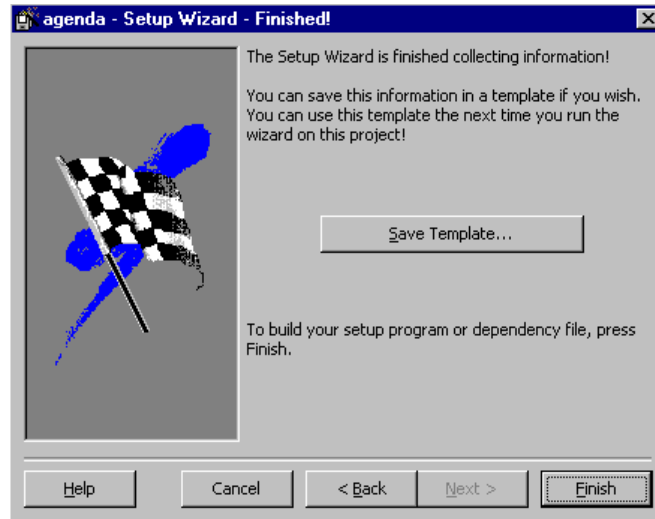
Para ver detalhes sobre cada arquivo selecione-o e clique em **File Details...**

Para ver a dependência de cada arquivo, selecione o arquivo e observe a caixa de texto **Dependency Of:**. Para ver o número de arquivos e a quantidade total de espaço em disco necessária para a instalação de sua aplicação clique em **Summary info...**



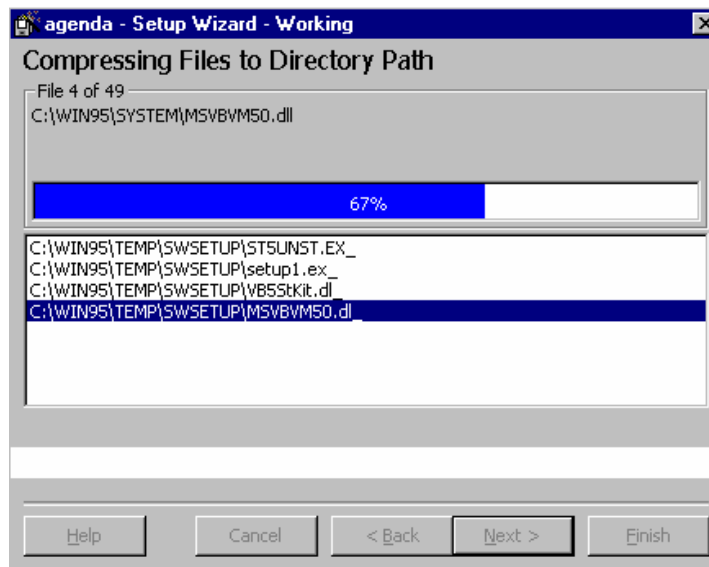
Tela 05

7 - A seguir o **Setup Wizard** mostra a tela 7.0 onde basta clicar no botão **Finish** para iniciar compactação de seus arquivos e criar o arquivo **Setup.lst** contendo lista de todos os arquivos de distribuição. Você pode criar um arquivo modelo (*Template*) com extensão **VBZ** de forma a não ter que repetir todo o processo para recriar os discos ou se quiser fazer alguma modificação.



Tela 07

A compactação de cada arquivo é mostrada na tela 8.0 abaixo:



Tela 08

Quando tudo estiver terminado a tela 9.0 abaixo indica que o processo chegou ao fim.



Tela 09

Basta agora copiar o conteúdo de cada diretório para os discos de instalação e pronto, você já pode distribuir a sua aplicação. Simples, não é !!!

- **Assistente de Instalação em português.**

Infelizmente parece que não esta nos planos da Microsoft em localizar o Visual Basic em português, por isso quando você gerar o assistente de instalação ele estará todo em inglês. Existe porém uma maneira de traduzirmos as mensagens e avisos do Assistente de instalação para a língua portuguesa, vamos explicar como fazer isso passo a passo:

- Gere em um diretório os discos de instalação para qualquer projeto usando o **Setup Wizard**.
- Após a geração verifique no diretório a presença dos arquivos **Setup.exe**, **Setup1.ex_** e **Setup.lst** dentre outros.
- O arquivo **Setup.exe** é um executável e não foi desenvolvido no Visual Basic.
- O arquivo **Setup1.ex_** esta compactado e foi desenvolvido em VB. O projeto **Setup1.vbp** está no diretório *C:\VB\SETUPKIT\SETUP1*
- O arquivo **Setup.lst** é um arquivo de texto com várias seções e contém informações sobre os arquivos que serão utilizadas durante o processo de instalação.
- Lembre-se que para iniciar a instalação você executa primeiro o **Setup.exe**
- O **Setup.exe** copia os arquivos básicos , indicados na seção **[Bootstrap]**, do arquivo **Setup.lst** descompacta e dispara o **Setup1.exe**
- Concluimos então que é através de **Setup1.exe** que poderemos traduzir o nosso assistente de instalação para o português.

Se você pensou em abrir o projeto **Setup1.vbp** no Visual Basic, localizar as mensagens e avisos nos formulários(frm) e nos módulos(bas), traduzí-los e depois salvar o projeto e executá-lo para testes vai ter duas decepções:

- 1 - Você não vai encontrar nenhum aviso ou mensagem quer nos formulários quer nos módulos do projeto **setup1.vbp**.

2 - Você não vai conseguir executar o projeto setup1.vbp no Visual Basic, se tentar obterá a mensagem de erro: Invalid command-line parameters. Unable to continue.

Bem chega de suspense, para facilitar a localização para vários idiomas, todas as mensagens estão contidas no arquivo de recurso(resource files) **Setup1.res** o qual é gerado pela compilação do arquivo editável **setup1.rc** que está no diretório **vb\tools\resource** do CD-ROM do Visual Basic 4 Enterprise. Então vamos lá:

1 - Faça um backup do subdiretório Setup1 por precaução.

2 - Edite o arquivo **Setup1.rc** e faça a tradução das mensagens, salve o arquivo como **Setup1.rc**

3 - Agora compile o arquivo setup1.rc para setup1td.res da seguinte maneira:

```
c:\vb\tools\resource\rc16\rc.exe /r /fo setup1td.res setup1.rc
```

4 - Abra o projeto **Setup1.vbp** e remova o arquivo **setup1.res** e inclua o arquivo **setup1td.res**

5 - Salve o projeto e gere o executável setup1.exe

Pronto, a primeira decepção foi resolvida, agora vamos à segunda: Fazer os testes executando o projeto **Setup1.vbp** diretamente no Visual Basic. Siga o roteiro:

1 - Edite o arquivo **Setup.lst**, do diretório de instalação, com o Notepad.

2 - Localize a linha **Setup=Setup1.exe** e substitua por

Setup=c:\vb\vb16.exe c:\vb\setupkit\setup1\setup1.vbp /cmd

3 - Agora Salve o arquivo **Setup.lst** e execute o arquivo **setup.exe** do diretório aonde foram gerados os arquivos de instalação.

4 - Ao invés de executar setup1.exe é iniciado o Visual Basic e o projeto **Setup1.vbp** é carregado.

5 - Agora é só executar (F5) o projeto e testar o assistente traduzido.

Nota: Para ver qual foi o parâmetro passado pelo comando **/cmd**, Clique em **Tools->Options** e na guia **Advanced**.

10) DBGRID

Mil e uma utilidades. Veja como utilizar o **DBGrid** suas propriedades e como configurá-lo. Tire suas dúvidas através de projeto explicado passo a passo.(Projeto para Controle bancário.)

Introdução

Utilização da Grade Vinculada aos dados: DbGrid.

A Grade Vinculada aos dados - DbGrid oferece um meio para visualizar vários registros ao mesmo tempo. Assemelha-se ao comando **Browse** usado no **Clipper/FoxPro** e ao modo tabela do Access.

O seu primo pobre no VB seria o controle Grid, e, apenas para dimensionar a diferença, o Grid está limitado a 16352 linhas e 5450 colunas no Dbgrid a quantidade de linhas esta condicionada aos recursos do sistema e a 1700 colunas, sem contar que o desempenho do Dbgrid e bem superior ao do Grid.

Enquanto o Grid precisa ser configurado quase que totalmente via código, para usar o Dbgrid basta arrastar o ícone do Controle para o seu formulário e definir a propriedade **DataSource** para identificar o controle de dados que contém os dados que você quer exibir, e pronto, a grade exibe todos os campos dos registros do recordset.

Nota: Para preservar recursos do sistema use uma instrução SQL na propriedade **Recordsource** do controle de dados que o DbGrid irá utilizar.

Utilização e Configuração.

Vejamos passo a passo como utilizar o Dbgrid com o controle de dados vinculados - Data Control.



De início selecione o objeto controle de dados na Toolbox do Visual Basic e acrescente-o ao seu formulário.(Fig.1)

fig1.

Defina a seguir as propriedades:

- **Name** - Nome do Controle (O padrão é data1) e **Caption** - Especifica o nome que aparece no controle de dados.
- **DatabaseName** - Para bancos de dados do Access, representa o nome do arquivo do banco de dados, inclusive o nome completo do caminho. (Para banco de dados diferentes do Access é necessário informar o nome do caminho até o subdiretório dos dados. Ex- Para padrão Dbase : DatabaseName=c:\dir)
- **RecordSource** - Especifica as informações que deseja obter no banco de dados - uma tabela, uma instrução SQL .
- **RecordsetType** - Se deseja criar um 2-snapshot ou um acessar uma 0-table altere esta propriedade, pois o padrão é 1-dynaset.
- **Connect** - Necessária quando o banco de dados não for do Access. Ex - Para banco de dados padrão Dbase - Connect=dBASEIII.

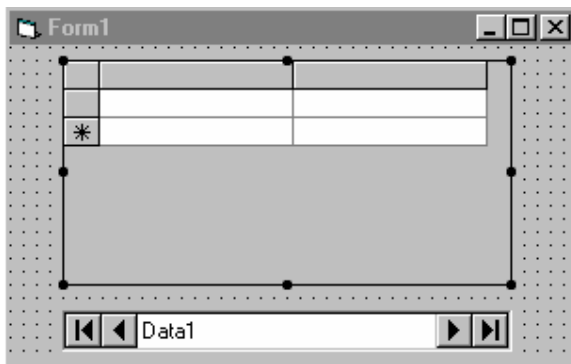


Fig-4

A seguir selecione o ícone do DBGrid na ToolBox e arraste-o até o seu form dimensionando como na fig.4. ao lado.

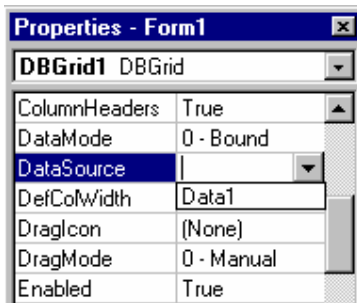


Fig-5

Finalmente defina a propriedade **DataSource** vinculado-a ao controle de dados configurado anteriormente. fig.5.

Para permitir a edição, inclusão e exclusão de registros na grade você deve definir como True as propriedades AllowAddNew, AllowUpdate e AllowDelete. Isto pode ser feito através da folha de propriedades do DBgrid (fig.6), onde podemos também especificar opções avançadas de tratamento do Recordset sem uma linha de código sequer.

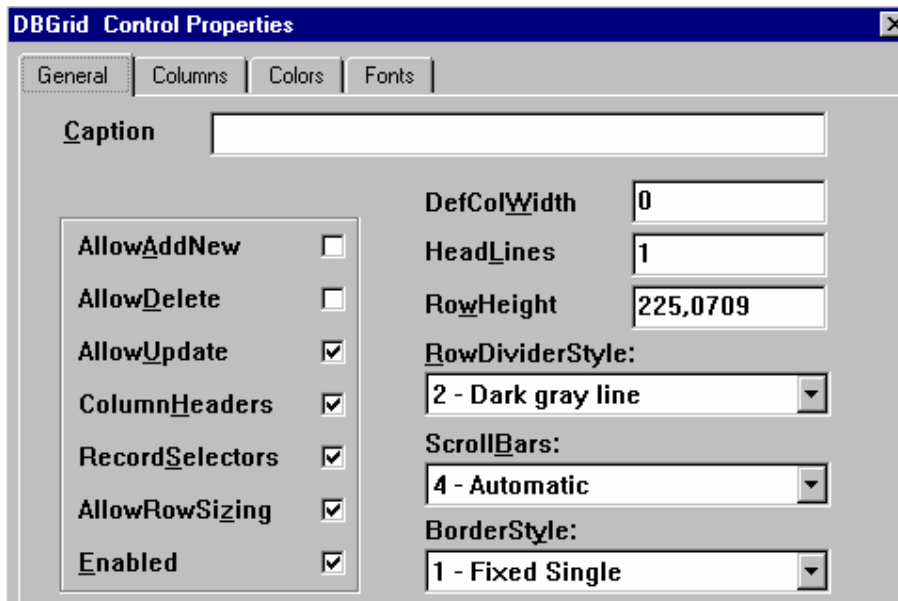


Fig-6

• Projeto - Controle Financeiro.

Chega de conceitos, vamos colocar em prática tudo isto em um projeto simples, um pequeno controle financeiro que você depois pode aperfeiçoar. Então mão a obra:

- objetivo: Controlar a movimentação de contas bancárias.

- Constituição:

- Um formulário MDI para o menu de opções com botões de ícones.
- Um formulário para cadastrar: Clientes, Contas, Transações.
- Um formulário para visualizar a movimentação de determinada conta.
- Um formulário para incluir dados.
- Um formulário para excluir dados.
- Um formulário para alterar dados.
- Um módulo para código utilizado por toda a aplicação.

1 - Vamos iniciar com a criação do menu, simples na verdade, mas que serve para os nossos propósitos. Fica a seu critério incrementá-lo. Vamos lá.

- Inicie o VB e acione a opção Insert->MDI Form para inserir um formulário MDI que conterà os demais formulários do projeto.

- O nosso formulário deverá ter o aspecto mostrado na figura 1.0 abaixo:

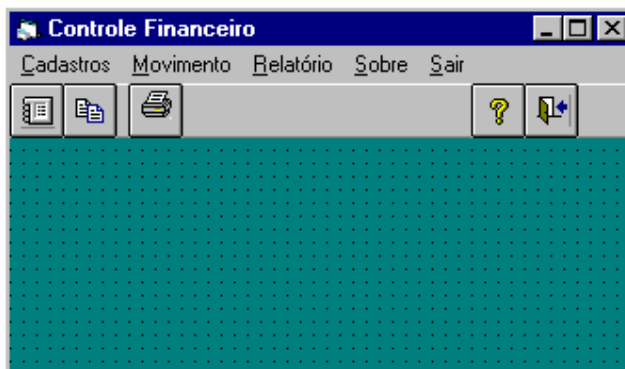


Fig-6

Neste formulário temos os seguintes controles:

- 1 controle SSPanel

- 5 botões SSCommand

Além disso temos um menu feito no Editor de Menus do Visual Basic.

- Insira os controles no formulário e configure suas propriedades como indicado a seguir na tabela 1.0:

Tabela 1.0 - Objetos e propriedades do form frmmenu.

Objeto	Propriedade	Configuração
MDIForm	Name	frmmenu
	Picture	fig 1.0
(*) SSCommand	Name	SSCommand1(0)
	Picture	fig 1.0 - Clientes, Contas, Transações
	BevelWidth	1
SSCommand	Name	SSCommand1(1)
	Picture	fig 1.0 - Gerenciar o movimento
	BevelWidth	1
SSCommand	Name	SSCommand1(2)
	Picture	fig 1.0 - Relatórios do Sistema
	BevelWidth	1
SSCommand	Name	SSCommand1(3)
	Picture	fig 1.0 - Ajuda do Sistema

	BevelWidth	1

SSCommand	Name	SSCommand1(4)
	Picture	fig 1.0 - Sair do Sistema
	BevelWidth	1

SSPanel	Caption	" "
	Name	panel
	Align	1-Align Top
	BevelWidth	1
	BevelInner	0-None
	BevelOuter	2-Raised

(*) Constituem um "control array" - Controles com o mesmo nome e do mesmo tipo, dotados de um índice identificador.

(**) Para usar o Editor de Menus do VB pressione CTRL+E e informe o nome que deseja para o menu na propriedade Caption e na propriedade Name informe o nome de referência do menu.

Para criar subitens clique na seta para direita repita o passo anterior e a seguir no botão Next para ir para o próximo subitem.

Obs.: Para sublinhar a primeira letra do nome do menu insira o caractere & antes da letra. Ex: &Cadastros -> exibe Cadastros.

2 - Agora basta atribuir o código a cada botão de comando, no nosso caso só usaremos o botão que gerencia o movimento e o botão para sair do sistema.

- No botão que gerencia o movimento - Menu Movimento - iremos carregar o formulário frmtrans.

- No botão para sair do Sistema inserimos o código que encerra a aplicação. Veja abaixo:

```
Private Sub SSCommand1_Click(Index As Integer)

    Select Case Index
        Case 0 'botão 1 - Cadastramentos (por sua conta)
        Case 1 'Gerencia movimento
            frmtransa.Show
        Case 2 'Relatórios
        Case 3 'Ajuda
        Case 4 'Sair da aplicação
        end
    End Select

End Sub
```

Controle Financeiro - Definição de tabelas.

Vamos agora definir o banco de dados e as tabelas utilizadas pelo sistema. A princípio o sistema usará as seguintes tabelas:

- BANCO.MDB - será o banco de dados do nosso sistema.
- tblcli - Tabela de cadastro dos clientes.
- tblcontas - Tabela de cadastro das contas a controlar.
- tblcodtrans - Tabela de cadastro dos códigos das transações realizadas.
- tbltrans - Tabela onde será registrada toda a transação realizada.

Vejamos agora a definição dos campos de cada tabela e a criação do banco de dados.

1 - Crie o seu banco de dados utilizando o Data Manager ou use o Microsoft Access e grave-o com o nome de BANCO.MDB.

2 - Crie uma tabela com o nome de tblcli com a seguinte estrutura:

Nome do Campo	Tipo de Dados	Tamanho
COD_CLI	LONG INTEGER (*)	
DES_CLI	CHARACTER	30

(*) Como cada cliente deve ter um código único garantindo assim a exata identificação do mesmo, defina o tipo de dados para o campo código como LONG INTEGER, e ative o atributo Counter e a opção Required.

3 - Crie uma tabela com o nome de tblcontas com a seguinte estrutura:

Nome do Campo	Tipo de Dados	Tamanho
COD_CLI	LONG	
COD_CONTA	LONG	30
DES_CONTA	TEXT	30

- Defina um índice para os campos cod_cli e cod_conta com o nome de cliconta e com as seguintes propriedades: Unique, Primary index, Required.

4 - Crie uma tabela com o nome de tblcodtrans com a seguinte estrutura:

Nome do Campo	Tipo de Dados	Tamanho
COD_TRAN	LONG	
DES_TRAN	TEXT	30

- Defina um índice para o campo COD_TRANS e indique-o como chave primária

5 - Finalmente, Crie uma tabela com o nome de tblcodtrans com a estrutura:

Nome do Campo	Tipo de Dados	Tamanho
COD_LANCA	LONG INTEGER (*)	
COD_TRANS	LONG	
COD_CLI	LONG	
COD_CONTA	LONG	
DATA_TRANS	DATE/TIME	
LANCAMENTO	CURRENCY	
SALDO	CURRENCY	
NU_DCTO	TEXT	15
OBS_TRANS	TEXT	30

(*) Como cada lançamento deve ter um código único garantindo assim sua exata identificação, defina o tipo de dados para o campo cod_lanca LONG INTEGER, e ative o atributo Counter e a opção Required.

- Defina os seguintes índices para esta tabela:

nome do índice	composição	
- PrimaryKey	cod_lanca	índice - chave primária (já definido)
- datatrans	data_trans	índice - datatrans por data de transação

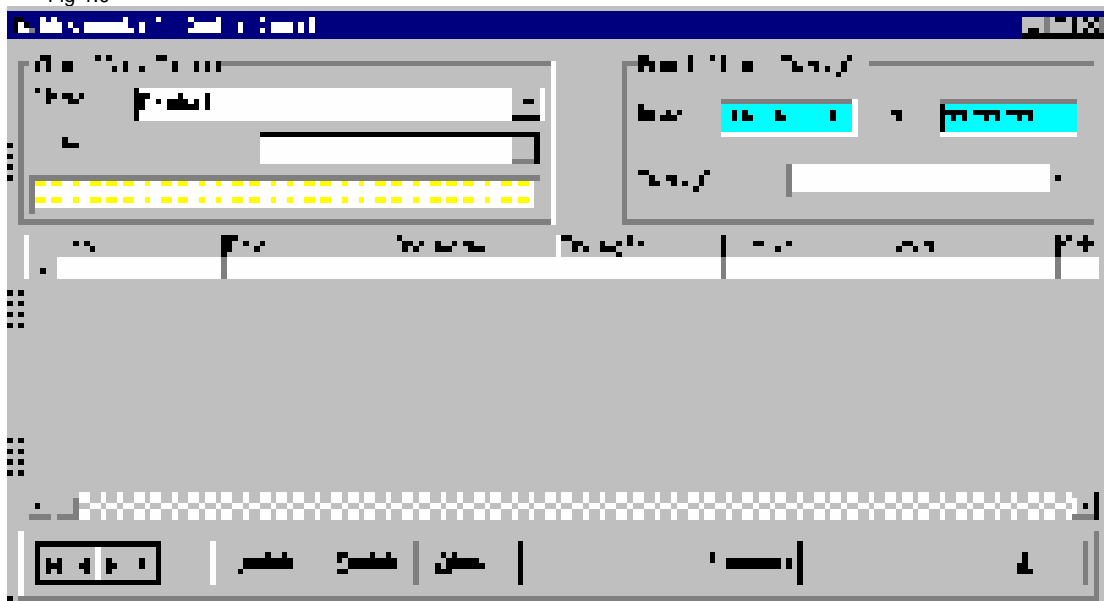
Controle Financeiro - Interface com o usuário.

Vamos agora desenhar a interface com o usuário. Nesta fase iremos construir três formulários que comporão o nosso projeto:

- frmtrans - Formulário que gerencia a movimentação das contas
- frmexcl - Formulário para exclusão de lançamentos
- frmincl - Formulário para incluir lançamentos
- frmalt - Formulário para efetuar alterações nos lançamentos efetuados

1 - O formulário frmtrans é o formulário principal de nossa aplicação, pois através dele iremos gerenciar a movimentação das contas. Ele terá o aspecto da fig1.0 abaixo:

Fig-1.0



Para montar o formulário acima descrito observe os seguintes passos:

- 1 - Inicie um novo projeto no Visual Basic. Grave o formulário Form1 como frmtrans.
- 2 - Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo:

Tabela 1.0 - Objetos e propriedades do formulário frmtrans

Objeto	Propriedade	Configuração
Form	Name	frmtrans
	Caption	"Movimentação Contas Correntes"
	MDIChild	True
Frame	Name	Frame1
	Caption	Cliente / Conta Corrente

Frame	Name Caption	Frame2 Período / Tipo de Transação
ComboBox	Name style	CboCliente 0-Dropdown Combo
ComboBox	Name style	CboConta 0-Dropdown Combo
Label	Name Caption	Label6 " "
MaskedTextBox	Name Mask PromptInclude PromptChar	maskini 99/99/99 False " "
MaskedTextBox	Name Mask PromptInclude PromptChar	maskfim 99/99/99 False " "
ComboBox	Name style	CboTipo 0-Dropdown Combo
(*) CommandButton	Name Caption	command1(0) "&Incluir"
CommandButton	Name Caption	command1(1) "&Excluir"
CommandButton	Name Caption	command1(2) "&Alterar"
CommandButton	Name Caption	command1(3) "&Sair"
CommandButton	Name Caption	command2 "&Processa"
SSPanel	Caption Name	" " SSPanel1
Data	Name	data1
DbGrid	Name Column1 Column2 Column3 Column4 Column5 Column6	DbGrid1 Data Documento Transacao Historico Valor Saldo
(**) Label	Caption AutoSize	** **

(*)Constituem um **"control array"** - Controles com o mesmo nome e do mesmo tipo, dotados de um índice identificador.

(**)Todos os controles **Label** possuem a propriedade **AutoSize=True** e **Caption** sendo igual ao nome do respectivos controle descrito na fig 1.0.

2 - O formulário frmincl é o formulário que será usado para incluir lançamentos.

Tem o seguinte aspecto mostrado na Fig. 2.0 abaixo:

Fig-2.0

Para montar o formulário acima descrito observe os seguintes passos:

- 1 - Inicie um novo projeto no Visual Basic. Grave o formulário Form1 como frmincl.
- 2 - Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo :

Tabela 1.0 - Objetos e propriedades do formulário frmincl

Objeto	Propriedade	Configuração
Form	Name	frmincl
	Caption	"Lançamentos-Inclusão Débito/Crédito
	MDIChild	True
Frame	Name	Frame1
	Caption	Lançamento
OptionButton	Name	Option1
	Caption	Débito
	Value	True
OptionButton	Name	Option2
	Caption	Crédito
	Value	False
Frame	Name	Frame2
	Caption	Movimentação
MaskedTextBox	Name	maskdata
	Mask	99/99/99
	PromptInclude	False
	PromptChar	" "
MaskedTextBox	Name	maskvalor
	format	\$#,##0.00;(\$#,##0.00)
	Mask	" "
	PromptInclude	False
	PromptChar	" "
ComboBox	Name	CboTipo
	style	0-Dropdown Combo
TextBox	Name	txtnudcto
TextBox	Name	txtdescr
SSPanel	Caption	" "

	Name	SSPanel1
(*) CommandButton	Name	command1(0)
	Caption	"&Grava"
CommandButton	Name	command1(1)
	Caption	"&Cancela"
CommandButton	Name	command1(2)
	Caption	"&Sair"

(*) Constituem um **"control array"** - Controles com o mesmo nome e do mesmo tipo, dotados de um índice identificador.

(**) Todos os controles **Label** possuem a propriedade **AutoSize=True** e **Caption** sendo igual ao nome do respectivos controle descrito na fig 2.0.

3 - O formulário frmexcl é o formulário que será usado para excluir lançamentos.

Tem o seguinte aspecto mostrado na fig 3.0 abaixo:

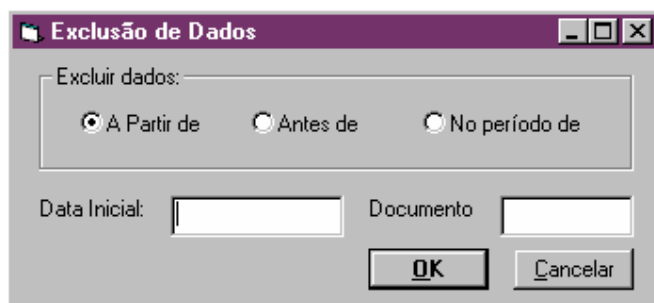


Fig-3.0

Para montar o formulário acima descrito observe os seguintes passos:

1 - Inicie um novo projeto no Visual Basic. Grave o formulário Form1 como frmexcl.

2 - Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo:

Tabela 1.0 - Objetos e propriedades do formulário frmexcl

Objeto	Propriedade	Configuração
Form	Name	frmexcl
	Caption	"Exclusão de Dados"
	MDIChild	True
Frame	Name	Frame1
	Caption	Excluir Dados
OptionButton	Name	Option1
	Caption	A partir de
	Value	True
OptionButton	Name	Option2
	Caption	Antes de
	Value	False
OptionButton	Name	Option2
	Caption	No Período de
	Value	False
TextBox	Name	text2
TextBox	Name	txtdcto
TextBox	Name	text1

	Visible	false
CommandButton	Name	command2
	Caption	"OK"
CommandButton	Name	command1
	Caption	"&Cancela"

Controle Financeiro - Código do projeto.

- Código do formulário de **menu:frmmenu**

Botões de Comando do Menu.

```
Private Sub SSCommand1_Click(Index As Integer)
    select case index
        case 0
        case 1 'frmtransa
            frmtransa.Show
        case 2
        case 3
        case 4 'sair
            end
        end select
End Sub
```

- Código do formulário das **transações:frmtransa**

-Caixa de combinação cliente

```
Private Sub cbocliente_Click()
    If cbocliente.Text < > "" Then
        Dim sql_tmp As String
        sql_tmp = "SELECT * FROM tblcontas WHERE tblcontas.cod_cli=" & cbo-
        cliente.ItemData(cbocliente.ListIndex)
        enche_combo cboconta, sql_tmp, "cod_conta", "cod_cli"
    End If
End Sub
```

- caixa de combinação conta

```
Private Sub cboconta_Click()
    If cboconta.Text < > "" Then
        Dim arqtemp As Recordset
        Set arqtemp = db.OpenRecordset("tblcontas", dbOpenTable)
        arqtemp.Index = "codconta"
        arqtemp.Seek "=", Val(cboconta.Text)
        Label6.Caption = arqtemp("desc_conta")
        arqtemp.Close
        Maskini.SetFocus
    End If
End Sub
```

- Código dos botões Incluir, Excluir, Alterar e Sair

```
Private Sub Command1_Click(Index As Integer)
```

```

Dim msg As String

Set tabela = db.OpenRecordset("tbltrans", dbOpenTable)
tabela.Index = "cod_conta"

Select Case Index
Case 0 'incluir
    alterareg = False
    frmtrans_i.Show 1
Case 1 'excluir
    frmexcl.Show 1
    data2.Refresh
Case 2 'alterar
    if dbgrid1.columns(0) < > "" then
        tabela.Seek "=", DBGrid1.Columns(0)
    else
        msgbox "Selecione um lancamento a alterar ! "
        exit sub
    endif
    alterareg = True
    frmtrans_i.Show 1
Case 3
    Unload Me
    Exit Sub
End Select

atualiza_grid (sql)
Data2.Refresh

End Sub

```

- Código do botão Processar

```

Private Sub Command2_Click()

    Dim msg As String
    Dim x As Integer

    '-----checa seleção de cliente/conta--
    If cbocliente.ListIndex = -1 Then
        msg = "Selecione um cliente !"
        MsgBox msg, vbExclamation
        cbocliente.SetFocus
        Exit Sub
    End If
    If cboconta.ListIndex = -1 Then
        msg = "Selecione uma conta !"
        MsgBox msg, vbExclamation
        cboconta.SetFocus
        Exit Sub
    End If

    '-- atribui cliente/conta as variáveis publicas

    Cliente = cbocliente.ItemData(cbocliente.ListIndex)
    conta = cboconta.Text

    '--habilita botões de comandos--
    For x = 0 To 2
        Command1(x).Enabled = True
    Next

    '-- estabelece consulta SQL -----

```

```

    sql = "SELECT data_trans, cod_trans, lancamento, nu_dcto, saldo,
obs_trans FROM tbltrans"
    sql = sql & " WHERE cod_cli=" & cbocliente-
te.ItemData(cbocliente.ListIndex)
    sql = sql & " AND cod_conta=" & cboconta.Text

    '-----define tipo da transacao para consulta--
    If cbotipo.ListIndex < > -1 Then
        sql = sql & " AND cod_trans=" & cbotipo.ItemData(cbotipo.ListIndex)
    End If

    '---verifica data inicial/final e define período-----
    If Maskini < > " / / " And maskfim < > " / / " Then
        If Not IsDate(Maskini) Then
            msg = "Data inicial invalida"
            Maskini.SetFocus
            MsgBox msg, vbExclamation
            Exit Sub
        ElseIf Not IsDate(maskfim) Then
            msg = "Data final inválida "
            maskfim.SetFocus
            MsgBox msg, vbExclamation
            Exit Sub
        End If
        sql = sql & " AND data_trans >= " & "#" & Format((Maskini),
"mm/dd/yy") & "#"
        sql = sql & " AND data_trans <= " & "#" & Format((maskfim),
"mm/dd/yy") & "#"
    End If

    '-----atualiza RecordSource-----
    sql = sql & " ORDER BY data_trans"
    Data2.RecordSource = sql
    atualiza_grid (sql)
    Data2.Refresh

End Sub

```

- Código de carga do formulário

```

Private Sub Form_Load()
    Dim x As Integer

    '-----define tamanho do form----
    Me.Width = 8295
    Me.Height = 5595

    '-- desabilita botões de comando
    For x = 0 To 2
        Command1(x).Enabled = False
    Next

    '---- abre tabela / e define índice ativo---
    'Set tabela = db.OpenRecordset("tbltrans", dbOpenTable)
    'tabela.Index = "clcnldr"

    '----configura data control---
    Data2.DatabaseName = App.Path & "\joias.mdb"
    Data2.RecordSource = "SELECT * FROM tbltrans WHERE cod_trans = Null"

    '-----formata grid-----
    DBGrid1.Columns(0).Width = 900
    DBGrid1.Columns(1).Width = 950

```

```

DBGrid1.Columns(2).Width = 925
DBGrid1.Columns(3).Width = 2200
DBGrid1.Columns(4).Width = 1300
DBGrid1.Columns(5).Width = 1300
dbgrid1.columns(6).width = 1300

'-----atualiza o grid-----
Data2.Refresh

'----- atualiza combobox cliente -----
enche_combo cbocliente, "tblcli", "des_cli", "cod_cli"
enche_combo cbotipo, "tblcodtrans", "des_tran", "cod_tran"
'-----

End Sub

```

- Código da função que atualiza o Grid

```

Public Sub atualiza_grid(sql As String)
    Dim credito, debito As Currency
    Dim saldo_atu As Currency
    Dim arqtemp As Recordset
    saldo_atu = 0

    '-----
    Set arqtemp = db.OpenRecordset(sql, dbOpenDynaset)
    '-----
    If arqtemp.EOF Then
        MsgBox "Não há dados no arquivo !", vbExclamation
        arqtemp.Close
        Exit Sub
    End If
    arqtemp.MoveFirst
    Do
        credito = 0
        debito = 0
        If Not IsNull(arqtemp![lancamento]) Then
            If arqtemp![lancamento] > 0 Then
                credito = arqtemp![lancamento]
            Else
                debito = arqtemp![lancamento]
            End If
        Else
            credito = 0
            debito = 0
        End If
        saldo_atu = credito + debito + saldo_atu
        arqtemp.Edit
        arqtemp![saldo] = saldo_atu
        arqtemp.Update
        arqtemp.MoveNext
    Loop While Not arqtemp.EOF
    arqtemp.Close
End Sub

```

Código do formulário para **incluir/alterar dados: Frmincl**

Código dos botões Gravar, Cancelar e Sair

```

Private Sub Command1_Click(Index As Integer)
    Select Case Index

```

```

Case 0 'gravar
    grava_recs
    tabela.Update
    If alterareg Then
        Unload Me
        Exit Sub
    End If
    tabela.AddNew
    maskdata.SetFocus
Case 1 'cancelar
    MsgBox "Atenção, as informações não foram gravadas ! "
    If alterareg Then
        Unload Me
        Exit Sub
    End If
    clear_controls
    maskdata.SetFocus
Case 2 'sair
    Unload Me
End Select
End Sub

```

Código de rotina de gravação**Código do botão de opção Antes de**

```

Private Sub Option1_Click()
    If Option1.Value = True Then
        Label2.Visible = False
        Text1.Visible = False
    End If
    Text2.SetFocus
End Sub

```

Código do botão de opção A Partir de

```

Private Sub Option2_Click()
    If Option2.Value = True Then
        Label2.Visible = False
        Text1.Visible = False
    End If
    Text2.SetFocus
End Sub

```

Código do botão de opção No período de

```

Private Sub Option3_Click()
    If Option3.Value = True Then
        Label2.Visible = True
        Text1.Visible = True
    End If
    Text2.SetFocus
End Sub

```

Código do modulo FINANCA.BAS**Código do seção general declarations**

```

Option Explicit

```



```

Public area As Workspace
Public db As Database
Public tabela As Recordset
Public consulta As QueryDef
Public alterareg As Boolean
Public sql As String
Public cliente As Long
Public conta As Long

```

Código de Sub Main

```

Sub main()
    'define área ativa e abre arquivo de dados
    Set area = DBEngine.Workspaces(0)
    Set db = area.OpenDatabase(App.Path & "\banco.mdb")
    'mostra formulário de menu
    frmmenu.Show
End Sub

```

Código da função que preenche as combobox

```

Public Sub enche_combo(combo As Control, data As String, campo As String,
    indice As String)

    '-- cria variável recordset temporária
    Dim arqtemp As Recordset
    '-----limpa combo
    combo.Clear
    '-----abre tabela como Snapshot
    Set arqtemp = db.OpenRecordset(data, dbOpenSnapshot)
    '--inicia loop através da tabela---
    Do Until arqtemp.EOF
        combo.AddItem arqtemp(campo)
        combo.ItemData(combo.NewIndex) = arqtemp(indice)
        arqtemp.MoveNext
    Loop
    '--- fecha recordset e não seleciona nada na combo
    arqtemp.Close
    combo.ListIndex = -1

End Sub

```

DbGrid - Dicas e Truques

1-Tornando uma coluna invisível

```

DbGrid1.columns(i).visible= False

```

11) DBLIST

Tire suas dúvidas de como utilizar o controle **DBLIST**. Veja como configurar as principais propriedades. Acompanhe exemplo comentado passo a passo. Veja exemplo de Pesquisa Dinâmica. Aprenda praticando.

• Introdução.

A Caixa de Listagem Vinculada aos dados - **DBList** tem função idêntica a Caixa de Listagem - List-Box, a qual seja, apresentar ao usuário uma lista de opções. A principal diferença consiste no fato de a **Caixa de Listagem Vinculada aos dados** buscar as informações em um **recordset**, enquanto que a Caixa de Listagem, obtém as informações através de uma série de instruções **AddItem**.

Vejamos como utilizar o controle **DBList** no roteiro a seguir:

1- Em um formulário insira o controle Data Control e configure as propriedades:

- *Name* -> nome do Controle. (O nome padrão é Data1.)
- *DataBasename* -> nome do banco de dados que deseja acessar.
- *RecordSource* -> nome da origem dos dados, geralmente o nome de uma tabela do banco de dados referido na opção anterior ou uma string SQL.
- *RecordSetType* -> Tipo do recordset utilizado (0-Table, 1-Dynaset, 2-Snapshot)

2- A seguir insira o controle **DBList** e atente para as seguintes propriedades:

- *RowSource* -> Nome do controle de dados que contém as informações usadas para preencher a lista.
- *ListField* -> O nome do campo da origem dos dados a ser exibido na lista.

Com essas propriedades definidas o DBList já será povoado. Vejamos as demais:

- *DataSource* -> Nome do controle de dados que contém o recordset de destino para as informações.
- *DataField* -> Nome do campo de destino. (O que será atualizado.)
- *BoundColumn* -> Nome do campo que contém o valor a ser copiado para outra tabela.

• Exemplo - Cadastro de Funcionários.

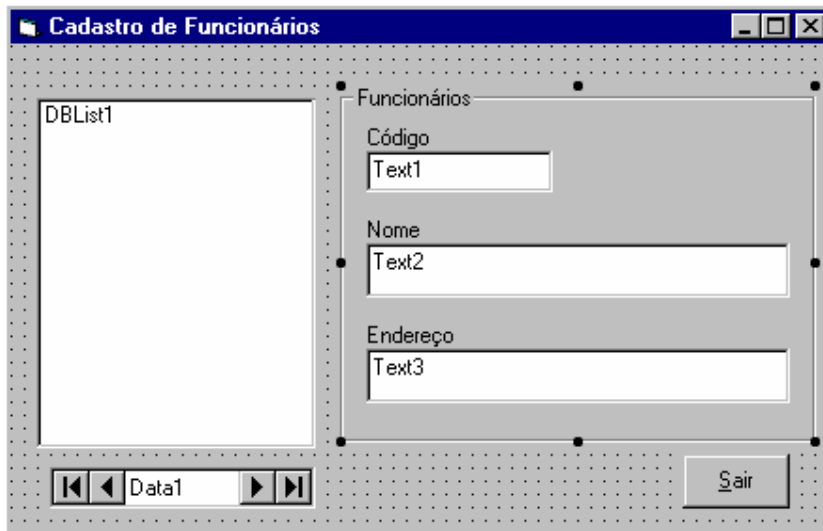
Vamos mostrar um exemplo de utilização típica do Controle **DBList** através de um projeto que tem por objetivo cadastrar os funcionários de uma empresa.

Iremos cadastrar o código, nome, endereço e setor do funcionário, para isso definimos o banco de dados **CADFUN.MDB** e a tabela **tblcad** com os campos indicados da seguinte forma:

Tabela TblCad - CADFUN.MDB

Campo	Tipo	Comp.	Defina um índice para o campo <i>código</i> com nome de código
Codigo	Long	-	Defina um índice para o campo <u>nome</u> com o nome de nome
Nome	Text	40	
Endereco	Text	40	
Setor	Text	12	

Nosso formulário em tempo de projeto terá a aparência da figura 1.0.



Neste formulário temos os seguintes controles:

- 1 data control
- 1 Dblist
- 1 Frame
- 3 Labels
- 3 Textbox
- 1 CommandButton

Fig. 1.0

-Insira os controles no formulário e configure suas propriedades como indicado a seguir na tabela 1.0:

Tabela 1.0 - Objetos e propriedades do form **frmcad**.

Objeto	Propriedade	Configuração
Form	Name	frmcad
	Caption	"Cadastro de Funcionários"
Data	Name	Data1
	Caption	Data1
	Visible	False
DBlist	Name	DBlist1
	RowSource	Data1
	MatchEntry	0 - Basic Matching
CommandButton	Name	Command1
	Caption	&Sair
TextBox	Name	Text1
	Text	"
TextBox	Name	Text2
	Text	"
TextBox	Name	Text3
	Text	"
Label	Name	Label1
	Caption	Código
	AutoSize	True
Label	Name	Label2
	Caption	Nome
	AutoSize	True
Label	Name	Label3
	Caption	Endereço
	AutoSize	True
Frame	Caption	Funcionários
	Name	Frame1

- **Código do Projeto - Cadastro de Funcionários.**

Vejamos agora o código comentado para cada evento do nosso projeto.

- Código da seção General Declarations do form.

```
Dim ws as Workspace
Dim db as DataBase
Dim tblcad as Recordset
```

- Código do evento Form Load que ocorre na carga do formulário.

```
Private Sub Form_Load()

    'define area de trabalho
    Set area = DBEngine(0)
    'abre banco de dados
    Set db = area.OpenDatabase(App.Path & "\cadfun.mdb")
    'abre tabela que será origem dos dados
    Set tblcad = db.OpenRecordset("tblcad", dbOpenTable)
    'define o indice ativo
    tblcad.Index = "nome"

    'configura propriedades do Data Control
    Data1.DatabaseName = App.Path & "\cadfun.mdb"
    Data1.RecordSource = "SELECT nome FROM tblcad ORDER BY nome"
    Data1.Refresh
    Set rsdata = Data1.Recordset

    'posiciona o ponteiro no primeiro registro
    rsdata.MoveFirst

    'configura propriedades do DBList
    DBList1.ListField = "Nome"
    DBList1.BoundText = rsdata!nome

    'carrega as caixas de texto com os valores do registro atual
    load_recs DBList1.BoundText

End Sub
```

- Código da função que carrega os controles com os dados.

```
Public Sub load_recs(strnome As String)
    'busca nome selecionado na tabela
    tblcad.Seek "=", strnome

    'se achar então carrega os controles
    If Not tblcad.NoMatch Then
        Text1 = tblcad("codigo")
        Text2 = tblcad("nome")
        Text3 = tblcad("endereco")
    End If

End Sub
```

- Código do evento Click de DBList.

```
Private Sub DBList1_Click()
    'chama a procedure load_recs passando o parâmetro selecionado de DBlist.
    load_recs DBList1.BoundText
End Sub
```

BoundText é o valor retornado por BoundColumn após cada seleção na lista de opções.

Note que com a propriedade **Match Entry** definida para **0 - Basic Matching** ao pressionarmos uma tecla nos deslocamos para o nome que começa pela letra digitada, e os controles são carregados com os dados automaticamente, tudo isso com pouco código.

- Ao executar a aplicação teremos algo como a tela da figura 2.0 abaixo:

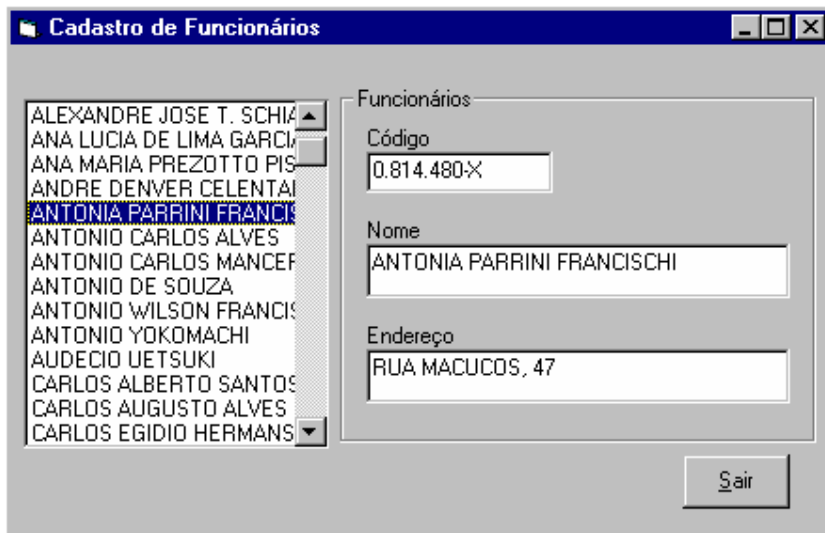


Fig. 2.0

- **Exemplo de busca dinâmica usando Dblist.**

Bem, agora iremos ver mais uma aplicação usando DBLIST. Vamos aproveitar o projeto anterior e incluir mais uma caixa de texto acima do controle DBLIST, conforme figura 3.0 abaixo:



Fig. 3.0

Queremos que a medida que se digita um nome para pesquisa na Caixa de texto a Caixa de listagem seja preenchida com os nomes do banco de dados que coincidem com as letras digitadas, ou seja, se digitarmos 'A' será mostrada na Caixa de Listagem todos os nomes que comecem com **A**, e assim por diante.

Para isso iremos fazer as seguinte alterações:

- Altere a propriedade **RecordsetType** do Controle de dados para **1-Dynaset**
- Acrescente o rótulo - Digite o nome para Pesquisa acima da Caixa de texto text1.

- Defina a propriedade **ListField** da Caixa de Listagem para nome.
- Para as Caixas de Texto defina a propriedade **DataSource** como **DATA1**, e a propriedade **DataField** como nome, endereço e setor respectivamente para text2, text3, e text4.

Agora só falta inserir o código abaixo para cada evento descrito:

- Código da seção General Declarations do form.

```
Option Explicit
Dim DB As Database
```

- Código do evento Form Load que ocorre na carga do formulário.

```
Private Sub Form_Load()
    Set DB = DBEngine.Workspaces(0).OpenDatabase(app.path "\cadfun.mdb")
    Data1.RecordSource = "SELECT nome,endereco,setor FROM tblcad ORDER BY nome"
    Data1.Refresh
End Sub
```

- Código do evento Change da Caixa de Texto text1.

```
Private Sub Text1_Change()
    Dim SQL As String
    Dim criterio As String
    criterio = Chr$(39) & Text1.Text & "*" & Chr(39)
    SQL = "SELECT nome,endereco,setor FROM tblcad WHERE nome LIKE " & criterio
    Data1.RecordSource = SQL
    Data1.Refresh
End Sub
```

Se você fez tudo certo, ao digitar a letra **A** na Caixa de texto text1 o sistema irá preencher a Caixa de listagem (DBLIST1) com todos os nomes que comecem com **A**, conforme figura 4.0 abaixo:

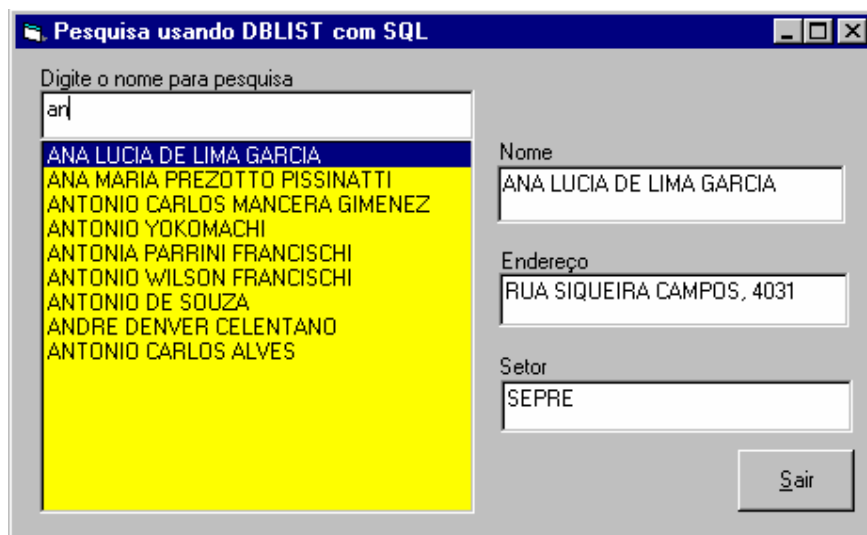


Fig. 4.0

12) REDES (Projeto comentado)

Programas Multiusuários - **Criação de Programas Multiusuários.** Acesso a Dados, esquemas de bloqueio de registros, a segurança do JET, atualização dos dados, erros, conflitos e desempenho. Acompanhe projeto explicativo.

- Primeiro vejamos um exemplo para o caso da abertura do banco de dados nos modos exclusivo e compartilhado :

- [Acesso Exclusivo/Compartilhado.](#)

Exemplo de Projeto - Comportamento Acesso Exclusivo e Compartilhado

Este projeto é bem simples mas serve para analisarmos como o Jet se comporta quando da abertura de arquivos em modo exclusivo e compartilhado em ambiente multiusuário.

Iremos abrir o arquivo **Biblio.mdb** que esta no diretório **c:\teste** em modo exclusivo e ver o como o Jet responde a uma tentativa de abrir o mesmo arquivo por outro usuário primeiro em modo exclusivo e depois em modo compartilhado. Depois inverteremos os modos para ver o resultado.

Você não precisa instalar este aplicativo em uma rede para testar o comportamento da abertura dos arquivos no modo exclusivo e no modo compartilhado.

Para a coisa funcionar basta você abrir duas instâncias do aplicativo e colocar as janelas lado a lado.

Vamos ao Projeto, simples por sinal:

1-Crie um novo projeto denominado **redes1.vbp**. No form1 crie os objetos e as propriedades como listadas na tabela 1.0 e grave o formulário com o nome **redes1.frm**.

Tabela 1.0 - Objetos e propriedades do form redes1.frm.

Objeto	Propriedade	Configuração
Form	Name	redes1.frm
	Caption	Arquivo Fechado
Frame	Name	frame1
	Caption	Modo de Acesso
OptionButton	Name	optexclusivo
	Caption	Modo Exclusivo
	Value	True
OptionButton	Name	optcompartilhado
	Caption	Modo Compartilhado
CommandButton	Name	cmdAbrir
	Caption	"&Abrir Arquivo"
CommandButton	Name	cmdFechar
	Caption	"&Fechar Arquivo"
	Enabled	False
CommandButton	Name	cmdSair
	Caption	"&Sair"
	Cancel	True

2 - Eis código do projeto :

Option Explicit

```
Dim db as Database
Dim dbname as String
```

```
Private Sub CmdAbrir_Click()
```

```
    'inicia o tratamento de erros
    On Error GoTo AbrirError
```

```
    'Localiza e abre o banco de dados conforme as opções do sistema.
    dbname="C:\TESTE\BIBLIO.MDB"
    set db = DbEngine.Workspaces(0).OpenDataBase (dbname, iif(optexclusivo,
True, False))
```

```
    'Mostra na barra do formulário através da propriedade Caption o modo de
abertura do arquivo.
```

```
    Me.Caption = iif(optcompartilhado, "Modo Compartilhado", "Modo Exclusi-
vo")
```

```
    'Alterna os botões para Abrir e/ou Fechar conforme a opção escolhida e
abandona a procedure.
```

```
    cmdFechar.Enabled = True
    cmdAbrir.Enabled = False
```

```
Exit Sub
```

```
    'Inicia o tratamento de erros.
```

```
AbrirError:
```

```
    Dim msg as String
```

```
    if Err= 3356 then
```

```
        'Este erro ocorre quando o usuário tenta abrir um arquivo que
        já está aberto no modo exclusivo ou quando o usuário tenta
        abrir no modo exclusivo um arquivo que já está aberto
        quer no modo exclusivo quer no modo compartilhado.
```

```
        if optcompartilhado then
```

```
            'O usuário tentou abrir o arquivo no modo compartilhado mas
            o mesmo já estava aberto no modo exclusivo.
```

```
            msg = "O arquivo esta aberto no modo exclusivo,"
            msg = msg & " você não pode abri-lo agora, tente mais tarde."
```

```
        else
```

```
            'O usuário tentou abrir o arquivo no modo exclusivo mas
            o mesmo já estava aberto por outro usuário.
```

```
            msg = "O arquivo já está aberto, e você não pode abri-lo "
            msg = msg & " no modo exclusivo agora, tente mais tarde "
```

```
        endif
```

```
    Else
```

```
        'Se ocorrer qualquer outro erro...
```

```
        msg = Err.Description
```

```
    Endif
```

```
    'Mostra a mensagem de erro ao usuário e sai da procedure.
```



```

MsgBox msg, vbExclamation

Exit Sub

End Sub

Private Sub cmdFechar_Click()

'1-Fecha o arquivo
'2-alterna as opções dos botões de abertura e fechamento do arquivo
'3-altera o status da barra do formulário.

db.close

cmdAbrir.enabled = True
cmdFechar.enabled = False

Me.Caption = " Arquivo Fechado "

End Sub

Private Sub cmdSair_Click()
End
End Sub

```

3-Depois de pronto compile o projeto e execute duas instâncias do aplicativo e coloque as janelas lado a lado como a fig 1.0 abaixo:

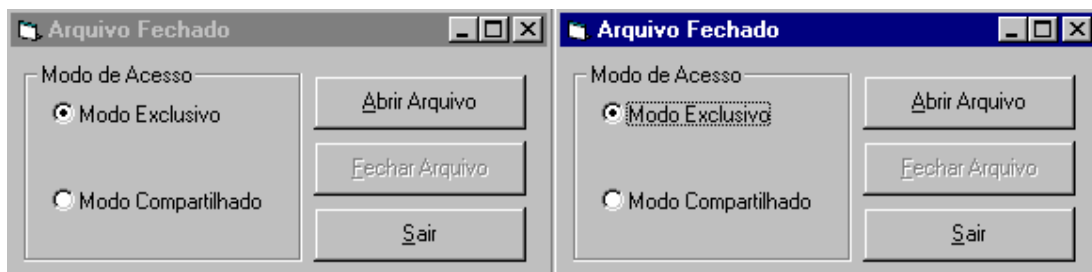


fig 1.0

1 - Primeiro abra o arquivo em modo exclusivo na janela da esquerda.

- Feito isto tente a mesma coisa na janela da direita
- Você receberá uma mensagem de erro informando que o banco de dados já está aberto

2 - Agora abra o arquivo no modo compartilhado na janela da esquerda.

- Abra o arquivo no mesmo modo na janela da direita
- Observe que agora você tem o arquivo aberto nas duas instâncias do aplicativo.
- Feche o arquivo e tente abri-lo no modo exclusivo na janela da direita.
- O Jet retornará uma mensagem de erro.

- A seguir um exemplo que analisa o comportamento do Jet na abertura das tabelas nas opções **db-DenyRead**, **dbDenyWrite** e **dbReadOnly** :

- [Acesso as tabelas - dbDenyRead, dbDenyWrite e dbReadOnly](#)

Acesso a Tabelas usando as opções dbDenyRead, dbDenyWrite e dbReadOnly.

Como sabemos o VB abre as tabelas para acesso compartilhado e faz o bloqueio de registros durante a atualização dos mesmos.

Se você necessitar bloquear alguns dados dentro de um conjunto de registros poderá utilizar as opções **dbDenyRead**, **dbDenyWrite** e **dbReadOnly** dependendo do tipo do conjunto de registros que estiver acessando. (Ver tabela 1.0 abaixo)

Restrição	Constante	RecordSet
Proíbe a outros usuários ver os dados	dbDenyRead	Table
Proíbe a outros usuários modificar ou acrescentar dados	dbDenyWrite	Table, Snapshot Dynaset
Permite acesso somente de Leitura aos Dados	dbReadOnly	Table, Dynaset

Tabela 1.0

Ao usar estas opções para acessar uma tabela você deve estar preparado para lidar com as possíveis mensagens de erro que o Jet retornará caso outro usuário tentar abrir a mesma tabela com as mesmas opções que você utilizou.

Uma forma de reduzir o número de erros que você receberá do Jet, quando precisar de acesso somente para leitura, é abrir a tabela como um **Snapshot**.

Iremos usar o arquivo **Biblio.mdb** que esta no diretório **c:\teste** e abrir a tabela **Publishers** com as opções mencionadas e ver o comportamento do Jet.

Para a coisa funcionar basta você abrir duas instâncias do aplicativo e colocar as janelas lado a lado.

Início do Projeto :

1-Crie um novo projeto denominado **redes2.vbp**. No form1 crie os objetos e as propriedades como listadas na tabela 1.0 e grave o formulário com o nome **redes2.frm**.

Tabela 1.0 - Objetos e propriedades do form redes2.frm.

Objeto	Propriedade	Configuração
Form	Name	redes2.frm
	Caption	"Banco de Dados - Biblio.mdb"
	ControlBox	False
Frame	Name	frame1
	Caption	"Abertura da Tabela Publishers Modo"
optionbutton	Name	option1(0)
	Caption	"Não Permite Acesso aos Dados - dbDenyRead"
optionbutton	Name	option1(1)
	Caption	"Não permite modificar Dados - dbDenyWrite"
optionbutton	Name	option1(2)
	Caption	"Permite Acesso só de leitura aos Dados - dbReadOnly"
CommandButton	Name	command1(2)
	Caption	"&Incluir Registro"
	Enabled	False

CommandButton	Name	command1(1)
	Caption	"&Fechar Tabela"
	Enabled	False
CommandButton	Name	command1(3)
	Caption	"&Sair"
	Cancel	True
CommandButton	Name	command1(0)
	Caption	"&Abrir a Tabela"

2-Código do Projeto:

Option Explicit

```
Dim db as Database
Dim rs as Recordset
Dim dbname as string
```

Private Sub Form_load()

```
'Localiza e abre o arquivo Biblio.mdb no modo compartilhado.
dbname="C:\TESTE\BIBLIO.MDB"
set db = DbEngine.Workspaces(0).OpenDataBase(dbname)
```

End Sub

Private Sub Command1_Click()

```
Dim msg As String
```

```
Select Case Index
```

```
Case 0 ' processar abertura da tabela com opção escolhida
```

```
On Error GoTo abrir_erro
```

```
If Option1(0).Value = True Then
    Set rs = db.OpenRecordset("publishers", dbOpenTable, dbDenyRead)
End If
If Option1(1).Value = True Then
    Set rs = db.OpenRecordset("publishers", dbOpenTable, dbDenyWrite)
End If
If Option1(2).Value = True Then
    Set rs = db.OpenRecordset("publishers", dbOpenTable, dbReadOnly)
End If
```

```
rs.Index = "Primarykey"
DBEngine.Idle dbFreeLocks
```

```
Command1(0).Enabled = False
Command1(1).Enabled = True
Command1(2).Enabled = True
```

```
Exit Sub
```

abrir_erro:

```
Select Case Err
```

```
Case 3261
```

```
msg = "Não posso abrir a tabela agora "
```

```
Case Else
```

```

        msg = Err.Description
    End Select

    MsgBox msg, vbExclamation

Case 1 ' fechar tabela

    rs.Close
    Command1(0).Enabled = True
    Command1(1).Enabled = False
    Command1(2).Enabled = False

Case 2 'incluir registro na tabela

    On Error GoTo inclui_erro
    Dim chave As Long

    rs.MoveLast
    chave = rs("pubid") + 1

    rs.AddNew

    rs("pubid") = chave
    rs("name") = "!!! TESTE DE INCLUSÃO !!!"

    rs.Update

    MsgBox "Um registro foi incluido na tabela Publishers"

Exit Sub
inclui_erro:
    Select Case Err
        Case 3027
            msg = "A tabela esta aberta apenas para leitura você não po-
de modificá-la"

            Case 3260
                msg = "O registro esta atualmente bloqueado, tente mais tar-
de. "

            Case Else
                msg = Err.Description
    End Select
    MsgBox msg, vbExclamation

Case 3 'sair da aplicação
    End
End Select

End Sub

```

3-Depois de pronto compile o projeto e execute duas instâncias do aplicativo e coloque as janelas lado a lado como a fig 1.0 abaixo:

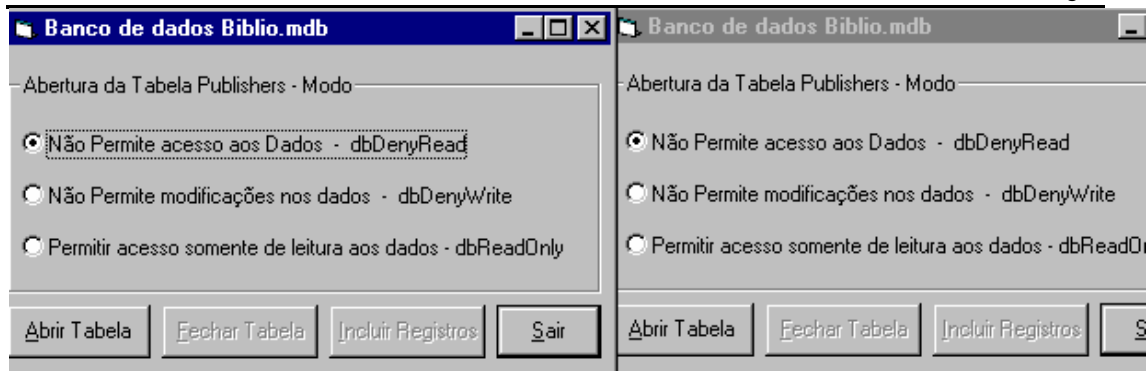


fig 1.0

1 - Primeiro na janela da esquerda selecione a opção com dbDenyRead.

- Abra a Tabela e clique em incluir registros. Até aqui tudo bem !

- Agora tente abrir a tabela do lado direito com qualquer das opções - DbDenyRead, dbDenyWrite e dbReadOnly.

- Provavelmente você receberá uma mensagem de que não pode abrir a tabela nestes modos.

2 - Agora feche tabela da janela da esquerda desative a opção dbDenyRead e ative a opção dbDenyWrite.

- Tente abrir a tabela na janela do lado direito com as duas primeiras opções : dbDenyRead e dbDenyWrite. Você será informado que não pode abrir a tabela nestes modos.

- Agora tente abrir a tabela do lado direito com a opção de somente Leitura dbReadOnly, desta vez você conseguira abrir a tabela.

- Tente incluir um registro clicando em incluir registro e uma mensagem o informará de que a tabela esta aberta no modo somente leitura e você não pode incluir registros.

Nota : Da mesma forma você pode usar as opções descritas acima para criar um **dynaset**. Ex: Para criar um dynaset com a opção **dbDenyWrite**

```
Set rs = db. OpenRecordset("SELECT * FROM Tabela", DbOpenDynaset, dbDenyWrite)
```

13) ERROS

Tratamento. - Como interceptar erros, a instrução **On Error**, o **Objeto Err**, Determinando o tipo de erro, localizando os erros, como encerrar o tratamento de erros, os principais erros relacionados a banco de dados e redes, informações recheadas de exemplos para você não ter dúvidas.

Introdução.

Você terminou de desenvolver aquele super projeto e esta ansioso para mostrar para o seu chefe. Naturalmente você depurou o seu código e testou tudo antes de gerar os discos de instalação, afinal o seu emprego está em jogo.

A instalação foi um sucesso, basta agora carregar a aplicação, então você clica no ícone da aplicação e surge na tela a fatídica mensagem:

Data error event hit in D:\PROJETO1\TESTES\CLIENTES.MDB isn't a valid Path...", imediatamente você percebe que esqueceu de mudar o caminho para a abertura do banco de dados, e agora... todos os olhos se voltam para você e... bem aí você acorda suando frio, ainda bem que foi tudo um sonho ... ou um pesadelo.

Veja bem, quando ocorre um erro durante a execução de um programa Visual Basic, o controle vai para uma lógica de tratamento de erros que consiste em exibir uma mensagem descrevendo a causa do erro e encerrando o programa a seguir.

Este é o procedimento padrão adotado pelo Visual Basic durante a execução de um programa compilado; não o leve a mal, ele apenas esta sendo gentil solicitando a sua intervenção.

Você pode, e deve, intervir neste processo construindo uma lógica de interceptação e tratamento de erros no seu código, que irá interceptar os possíveis erros e tratá-los de uma forma mais elegante que a adotada pelo procedimento padrão do Visual Basic.

Tudo isto se torna mais enfático quando se trabalha com banco de dados no VB, pois neste caso, teremos muitas condições de erro durante a execução da aplicação.

Lembre-se, não existem nem aplicações nem programadores nem usuários perfeitos.

Interceptando erros.

Para interceptar um erro no Visual Basic você usa a instrução **On Error**.

Quando uma instrução **On Error** está em vigor e ocorrer um erro o VB vai executar as ações determinadas pela instrução.

Para estar em vigor a instrução **On Error** deve ser executada antes da ocorrência de um erro na mesma função ou sub-rotina onde ocorreu o erro ou em uma função ou sub-rotina que tenha chamado a função ou sub-rotina onde o erro tiver ocorrido.

A instrução On Error tem a seguinte sintaxe:

```
On Error goto label
```

Traduzindo-a teremos:

Quando ocorrer um erro, transfira a execução para a linha após a etiqueta.

A etiqueta é um identificador alfanumérico que termina com dois pontos (:).
(Ex.: Trataerro:)

Ela tem que começar na primeira coluna e tem que estar na mesma função ou sub-rotina da instrução **On Error**, sendo exclusiva dentro de um módulo.

Após a etiqueta você fornece o código que vai tratar o erro ocorrido.

Vejam os exemplos:

```
Public Sub Exemplo()  
    On error Goto Trata_Erro -> Inicia tratamento de erro  
    Print 1/0                 -> Ocorreu um erro, a execução é  
                               desviada para a o código que vem após  
a etiqueta  
    Exit sub                 Trata_Erro  
  
Trata_Erro:  
msgbox "Não existe divisão por zero !!!!"  
    -> Exibe a mensagem e encerra a aplicação.  
End Sub
```

Obs.: Note que antes da etiqueta existe a instrução **Exit Sub**. Sem ela o código após a etiqueta seria sempre executado, mesmo não havendo ocorrência de erros.

Identificando o Tipo de Erro.

Cada erro gerado no Visual Basic esta associado a um número e tem uma descrição.

Para identificar um erro ocorrido, informando o seu número, a mensagem de erro, de onde o erro procede, etc. usamos o Objeto **Err**.

O objeto **Err** foi introduzido na versão 4.0 do VB e incorpora a funcionalidade da instrução Err, da função Err, da instrução e Função Error e da função Error\$.

Utilizaremos então as propriedades do objeto **Err** para identificar o tipo de erro e fazer o seu tratamento, se for o caso.

As principais propriedades do objeto Err que utilizaremos são:

Number	Fornece Numero do erro gerado
Description	Fornece a descrição do erro.
Source	Identifica o nome do objeto que gerou o erro

Quando ocorre um erro o VB coloca o número do erro na propriedade **Number** do objeto **Err**, conhecendo o número do erro você pode determinar o tipo de erro e tomar a decisão quanto ao seu tratamento. Vejam os exemplos práticos:

Vamos definir uma tabela com o nome de erro que estará armazenada no banco de dados Controle.mdb (já criado anteriormente ver “Definição de bando de dados e Tabelas”) e que possuirá a seguinte estrutura:

nome do campo	Tipo de Dados	Tamanho do Campo
codigo	Long	04
nome	Character	30

- 1 - Defina um índice para o campo **código**
ativando as opções: **Unique**, **Primary index**. e **Requerid**

Desta forma estamos criando uma chave primária para o campo codigo.

Se tentarmos gravar um registro com um número de código que já foi gravado o Jet retornará um erro, o erro de número de 3022, poderemos portanto interceptá-lo e através do tratamento informar ao usuário que o código já foi utilizado

permitindo a seguir ao usuário informar outro código.

Tudo se passa como se estivéssemos fazendo a crítica da entrada de dados do usuário, só que sem utilizar nenhum código a não ser a interceptação e o tratamento do erro.

1- Ao iniciar a aplicação teremos tela da figura 1.0 abaixo

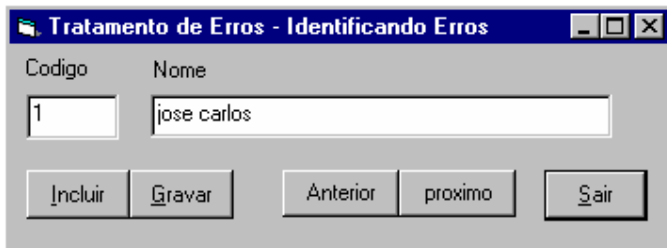


Fig.1.0

2 - Suponhamos que o usuário tente incluir um código que já foi utilizado, ao clicar no botão gravar ocorrerá uma mensagem de erro, pois o Jet não permite a duplicação de uma chave primária.

Prevendo tal ação iniciamos o tratamento de erros e interceptando-o retornamos ao usuário a mensagem da figura 2.0 abaixo:

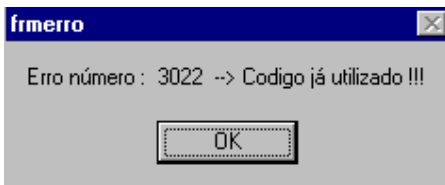


Fig.2.0

Fazemos isto utilizando a instrução Resume Next colocada após a mensagem de aviso ao usuário.

```
Private Sub Command5_Click()
    On Error GoTo erro_mdb 'inicia o tratamento de erros

    ' verifica se esta incluindo ou editando caso contrário não faz nada
    If rs.EditMode = dbEditAdd Or rs.EditMode = dbEditInProgress Then
        grava_regs
        rs.Update    --> linha que irá gerar o erro.
        limpa_regs
        Form1.Caption = "Tratamento de Erros "
        rs.Bookmark = rs.LastModified
        load_regs
    End If
Exit Sub

erro_mdb:

    MsgBox "Erro número : " & Str$(Err.Number) & " --> Código já utilizado !!!"
    Resume Next      'retorna a ação para a linha de código subsequente aquela que
                    'gerou o erro
End Sub
```


- Código do projeto para Tratamento de erros.

```
Option Explicit
```

```
Dim db As Database
Dim rs As Recordset
Dim dbname As String
```

```
Private Sub Command1_Click()
    'Inicia processo inclusão, limpa os controles e 'foca o textbox codigo
    Form1.Caption = "Inclusao !!! "
    rs.AddNew
    limpa_regs
    Text1.SetFocus
End Sub
```

```
Private Sub Command2_Click()
    'encerra aplicação
    End
End Sub
```

```
Private Sub Command3_Click()
    'move para o próximo registro
    rs.MoveNext

    If rs.EOF() Then rs.MovePrevious
    Form1.Caption = "Tratamento de Erros "
    load_regs

End Sub
```

```
Private Sub Command4_Click()
    'move para o registro anterior
    rs.MovePrevious
    If rs.BOF() Then rs.MoveNext
    Form1.Caption = "Tratamento de Erros "
    load_regs
End Sub
```

```
Private Sub Form_Load()
    dbname = App.Path & "\controle.mdb"

    Set db = DBEngine.Workspaces(0).OpenDatabase(dbname)
    Set rs = db.OpenRecordset("erro")

    If rs.RecordCount > 0 Then
        load_regs
    Else
        MsgBox "Arquivo vazio"
        limpa_regs
    End If
End Sub
```

```
Public Sub load_regs()
    'carrega os dados nos controles
    Text1 = "" & rs("codigo")
    Text2 = "" & rs("nome")
End Sub
```

```

Public Sub limpa_regs()
    'limpa os controles
    Text1 = ""
    Text2 = ""
End Sub

Public Sub grava_regs()
    'descarrega o conteúdo dos controles para gravação
    rs("codigo") = Text1.Text
    rs("nome") = Text2.Text
End Sub

```

Finalizando um tratamento de erros.

Após a interceptação do erro você faz o tratamento do mesmo e tem que terminar o tratamento de erros com uma instrução que elimine o erro, caso contrário o próprio tratamento de erros irá gerar um erro.

No caso anterior a instrução que eliminou o erro foi a instrução Resume Next, pois devolveu a ação ao usuário.

Vejamos abaixo as principais instruções utilizadas para eliminar um erro.

Resume Next	Retorna a execução na linha que vem logo após à linha que gerou o erro
Resume	Executa mais uma vez a linha que gerou o erro.
Resume <i>Label</i>	Retorna a execução da linha que vem após a etiqueta citada.
Resume Number	Retorna a execução na linha com o número indicado.
Exit Sub	Sai da sub rotina atual.
Exit Function	Sai da função atual.
Exit Property	Sai da propriedade atual.
On Error	Redefine a lógica de tratamento de erros.
Err.Clear	Elimina o erro sem afetar a execução do programa.
End	Encerra a execução do programa.

Vejamos um exemplo onde simulamos várias situações de erro e utilizamos o tratamento de erros para determinar o número do erro a linha onde ocorreu o erro e que gerou o erro.

Note que utilizamos números de linha em nosso código para ser possível a identificação do número da linha pelo tratamento de erros.

```

Private Sub Form_Load()

    Dim db As Database
    Dim dbName As String
    Dim rs As Recordset
    Dim s As String

    On Error GoTo TrataErro 'ativa o tratamento de erros

    dbName = app.path & "\controle.mdb"
10 Set db = DBEngine.Workspaces(0).OpenDatabase(dbName)

    ' Aqui ocorre um erro, pois a tabela não existe neste banco de dados
20 Set rs = db.OpenRecordset("erros", dbOpenTable)

    ' Aqui temos a abertura correta da tabela erro
30 Set rs = db.OpenRecordset("erro", dbOpenTable)

    ' Aqui ocorre mais um erro, pois não existe o campo endereço na
    ' tabela erro
40 s = rs![endereço]

    ' Aqui temos outro erro pois o código não aceita valores alfanuméricos
50 rs![código] = "XYZ"

```

```

' Encerra a aplicação
60 End

Exit Sub

LoadError:
MsgBox "Erro número #" & Str$(Err.Number) & " na Linha " & Str$(Erl) & " - "_
& Err.Description & " - gerado por " & Err.Source
Resume Next 'devolve a ação para a linha subsequente à que gerou o erro.
End Sub

```

Você deve obter, ao iniciar o projeto, a mensagem da figura 3.0 e a seguir a mensagem de erro da figura 4.0, e assim por diante.

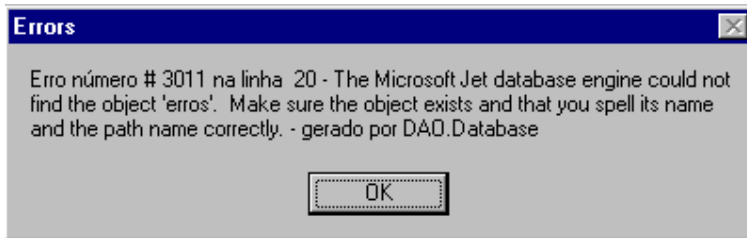


Fig.3.0



Fig.4.0

Observe que a descrição utilizada é fornecida pela propriedade **description** do objeto **Err**.

A instrução que eliminou o erro foi a instrução **Resume Next**, pois devolve a ação para a linha subsequente à que gerou o erro permitindo assim que o aplicação seja executada até o seu final

Principais erros relacionados a Banco de dados.

Naturalmente você deve estar preparado para prever os erros potenciais e planejar o seu tratamento, para isso deves conhecer os principais erros relacionados ao seu ambiente de trabalho.

Relacionar aqui todos os erros tratáveis do Visual Basic seria impossível, mas vamos tentar listar os principais fornecendo o seu número e descrição, vamos lá:

número	Mensagem
Erros genéricos.	
3005	Database name' isn't a valid database name.
3006	Database 'name' is exclusively locked.
3008	Table 'name' is exclusively locked.
3009	Couldn't lock table 'name'; currently in use.
3010	Table 'name' already exists.
3015	'Index name' isn't an index in this table.
3019	Operation invalid without a current index.
3020	Update or CancelUpdate without AddNew or Edit.
3021	No current record.
3022	Duplicate value in index, primary key, or relationship.
	Changes were unsuccessful.

3023 AddNew or Edit already used.
3034 Commit or Rollback without BeginTrans.
3036 Database has reached maximum size.
3037 Can't open any more tables or queries.
3040 Disk I/O error during read.
3044 'Path' isn't a valid path.
3046 Couldn't save; currently locked by another user.

Erros relacionados a bloqueio de registros

3027 Can't update. Database or object is read-only.
3158 Couldn't save record; currently locked by another user.
3167 Record is deleted.
3186 Couldn't save; currently locked by user 'name' on machine 'name'.
3187 Couldn't read; currently locked by user 'name' on machine 'name'.
3188 Couldn't update; currently locked by another session on this machine.
3189 Table 'name' is exclusively locked by user 'name' on machine 'name'.
3197 Data has changed; operation stopped.
3260 Couldn't update; currently locked by user 'name' on machine 'name'.
3261 Table 'name' is exclusively locked by user 'name' on machine 'name'.
The database is opened by user 'name' on machine 'name'.

Erros relacionados a Permissões

3107 Record(s) can't be added; no Insert Data permission on 'name'.
3108 Record(s) can't be edited; no Update Data permission on 'name'.
3109 Record(s) can't be deleted; no Delete Data permission on 'name'.
3110 Couldn't read definitions; no Read Definitions permission for
table or query 'name'.
3111 Couldn't create; no Create permission for table or query 'name'.
3112 Record(s) can't be read; no Read Data permission on 'name'.

14) OLE

Conceitos. - O que é OLE ? Automação OLE ?, DDE - você ainda usa ?, mesclando os dados de suas tabelas com arquivos feitos no WORD, não reivente a roda use os recursos que já estão á sua disposição, VB com Excel, VB com PowerPoint, tudo com exemplos codificados para você ver como se faz.

Introdução.

OLE ! não, não é o grito da torcida incentivando o seu time, nem a platéia a gritar em uma tourada. Mas o que é OLE então ? Esta pergunta não é tão simples de responder, e, se pudéssemos escrever um livro sobre o assunto não o esgotaríamos, então vamos por partes.

No princípio o **OLE 1.0** surgiu para substituir o **DDE (Dynamic Data Exchange)**, devido as suas limitações. Nessa época OLE **Object Linking and Embedding**. Introduzia dois conceitos:

1 - **Linking** - Cria vínculos ou referências aos objetos armazenando no documento principal apenas os dados realmente necessários para exibir, imprimir, etc.

2 - **Embedding** - Incorpora os dados dos objetos ao documento principal.

Neste contexto surgem as conceitos de **objeto vinculado** e do **objeto incorporado**, então vejamos:

Objeto Vinculado - São informações (objeto) criadas em um arquivo (arquivo origem) e inseridas em outro arquivo (arquivo destino).

Embora o **objeto vinculado** não se torne parte do arquivo de destino, existe um vínculo, uma conexão entre os dois arquivos de forma que o **objeto vinculado** no arquivo de destino é automaticamente atualizado quando o arquivo de origem é atualizado.

Objeto Incorporado - São informações (objeto) inseridas em um arquivo (arquivo de destino. Ao ser incorporado o objeto se torna parte do arquivo de destino.

Ao clicar duas vezes no **objeto incorporado**, ele é aberto no programa de origem em que foi criado. Qualquer alteração feita no **objeto incorporado** se refletirá no arquivo de destino.

Como exemplo podemos considerar um documento do Word aonde inserimos uma figura do Paint-Brush.

- Se **vincularmos** a figura ao documento , apenas seus vínculos são gravados junto com o documento.
- Ao contrário se **incorporarmos** a figura ao documento ela é gravada juntamente com o texto.
- Dá para concluir que o tamanho dos arquivos com objetos incorporados é muito maior que aqueles com objetos vinculados.

Outro conceito importante no OLE é o conceito de **Cliente** e **Servidor**.

- **Cliente** - Aplicação cliente é aquele que solicita os dados.
- **Servidor** - Aplicação é aquela que disponibiliza os dados.

No caso do exemplo do Word com a figura do PaintBrush , temos:

- **Cliente** -> Word , pois solicita os dados (a figura).
- **Servidor** -> Paintbrush, pois fornece os dados (a figura).

Os programas MS-Graph, WordArt, etc. são exemplos de programas que atuam somente como servidores, pois só entram em cena quando são requisitados. Excel , Word , PowerPoint, etc. podem atuar como Clientes ou Servidores.

No **OLE 2.0**, tivemos muitos aperfeiçoamentos dentre os quais podemos citar:

- O Drag-and-Drop
- O Uniform Data Transfer
- O Visual Editing
- A automação OLE

Dentre os acima citados o mais importante para nós é a **Automação OLE**.

A **automação OLE** permite que uma aplicação seja controlada por outra aplicação.

É através dela que podemos requisitar a outro aplicativo que execute uma determinada tarefa.

Como exemplo clássico iremos , estando no Visual Basic, solicitar ao Word que crie um novo documento e insira um texto no documento criado.

Isto é possível usando **automação OLE** pois o Word suporta OLE e pode atuar como cliente ou Servidor, fornecendo um objeto que suporta os comandos e funções da linguagem **WordBasic**.

Então mãos a obra:

1 - Inicie o Visual Basic, crie um novo projeto.

2 - No formulário crie um botão de comando com a propriedade Caption definida como **AUTOMAÇÃO OLE**.

3 - Crie um botão de comando com Caption definida como **&Saí** e acrescente o seguinte código ao botão:

```
Private Sub Command1_Click()  
    End  
End Sub
```

4 - Insira o seguinte código ao botão :

```
Private Sub Command2_Click()  
    Dim objword As Object                'declara um variável objeto  
    Set objword = CreateObject("Word.Basic") 'cria objeto e inicia o Word  
    objword.filenew "Normal", 0          'cria novo arquivo  
    objword.Bold                        'define fonte em negrito  
    objword.FontSize 24                 'define tamanho da fonte  
    objword.Insert "AUTOMAÇÃO OLE !!!"   'insere o texto no arquivo  
    objword.appmaximize "", 1  
    objword.FilePrintPreview             'Visualiza a impressão  
    objword.AppActivate "Microsoft Word"  
End Sub
```

Ao executar a aplicação e clicar no botão de comando **AUTOMAÇÃO OLE**, o Word será inicializado , criará um novo arquivo , incluirá o texto **AUTOMAÇÃO OLE** em negrito e com fonte de tamanho 24 mostrando o resultado no visualizador de impressão.

Para retornar ao aplicativo feche o Word.

Observe que o "coração" do sistema está na função **CreateObject("Word.Basic")** pois é ela que cria efetivamente o objeto WordBasic e inicia o Word.

NOTA IMPORTANTE !

Até a versão 7.0 o Word utilizava o WordBasic, a partir da versão 8.0 (MSOFFICE 97) o WordBasic foi substituído pelo VBA, mas mantém o objeto WordBasic apenas por questão de compatibilidade com as versões anteriores.

Como o WordBasic é sensível ao idioma de localização do WORD o exemplo acima provavelmente não funcionará no Word 6.0 em português, mas funcionará no Word 7.0, pois ele converte os comandos do WordBasic para o VBA automaticamente.

No VBA foi introduzida uma estrutura de objetos hierarquizada, e para programar em VBA é necessário conhecer esta estrutura. Apenas para ilustrar a diferença vejamos como alguns comandos do WordBasic ficariam em VBA:

Instrução do WordBasic	Instrução do VBA 97
Bold	Application.Selection.Font.Bold=True
insert "AUTOMAÇÃO OLE"	Selection.TypeText Text:="AUTOMAÇÃO OLE"
FilePrintPreview	ActiveDocument.PrintPreview

Usando Automação OLE com o Word - Exemplo Prático.

Imagine que você possua uma aplicação em VB que controla um banco de dados dos empregados de sua empresa.

Para facilitar vamos utilizar a tabela **empregados** do banco de dados de exemplo que vem com o Access, **NWIND.MDB**.

Você criou um documento padrão no Word, uma carta desejando feliz aniversário aos seus empregados.

Estando no seu aplicativo em VB você deseja que o nome do empregado atual seja inserido em seu documento Word e o mesmo seja impresso.

Vejamos como realizar tal tarefa:

1 - Crie um documento no Word com o nome de ANIVERSARIO.DOC. Veja modelo abaixo:

|

Parabéns

Nesta data desejamo-lhes muitas felicidades e que os seus sonhos se tornem realidade, afinal sonhar não faz mal a ninguém.

Sinceramente

Seus amigos da JCMSOFT LTDA

2 - Vamos Criar um novo projeto no Visual Basic , que será a aplicação que controla nosso cadastro de empregados.

Temos abaixo (figura 1.0) a tela principal de nossa aplicação em tempo de projeto:



Fig.1.0

Para montar o formulário acima descrito observe os seguintes passos:

- 1 - Inicie um novo projeto no Visual Basic. Grave o formulário Form1 como Empregados.
- 2 - Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo :

Tabela 1.0 - Objetos e propriedades do formulário Empregados

Objeto	Propriedade	Configuração
Form	Name	empregados
	Caption	"Automação OLE com Word 7.0"
	StartUPosition	2-CenterScreen
Data	Name	dtaole
	Caption	"Automação OLE com Word 7.0"
	Connect	Access
	Databasename	C:\access\exemplos\Nwind.mdb(*)
	RecordSetType	1-Dynaset
	RecordSource	"Empregados"
label	Name	label1(0)
	Caption	"Sobrenome"
	Autosize	True
label	Name	label1(1)
	Caption	"Nome"
	Autosize	True
label	Name	label1(2)
	Caption	"Endereço"
	Autosize	True
label	Name	label1(3)
	Caption	"Cidade"
	Autosize	True
label	Name	label1(4)
	Caption	"Estado"
	Autosize	True

label	Name	label1(5)
	Caption	"Cep"
	AutoSize	True
TextBox	Name	text1(0)
	DataField	"sobrenome"
	DataSource	dtaole
TextBox	Name	text1(1)
	DataField	"Nome"
	DataSource	dtaole
TextBox	Name	text1(2)
	DataField	"Endereco"
	DataSource	dtaole
TextBox	Name	text1(3)
	DataField	"cidade"
	DataSource	dtaole
TextBox	Name	text1(4)
	DataField	"estado"
	DataSource	dtaole
TextBox	Name	text1(5)
	DataField	"Cep"
	DataSource	dtaole
CommandButton	Name	command1
	Caption	"&Sair"
CommandButton	Name	command2
	Caption	"&Automação OLE com Word 7.0"
OLE	Name	Photo
	Datafield	Foto
	DataSource	dtaole
	SizeMode	0 - Clip

(*)-Esteja certo que o caminho informado aponte para o NWIND.MDB.

Como você pode observar utilizando o controle de dados teremos pouco a codificar.

Veja a seguir o código associado aos botões de comandos:

1 - Código associado ao botão que ativa automação OLE:

```
Private Sub Command2_Click()

    On Error GoTo trataerro 'ativa tratamento de erros
    Dim Word As Object ' Declara uma varial objeto

    Set Word = CreateObject("Word.Basic") 'Cria uma variável objeto Word
    Word.FileOpen ("e:\winword\ANIVERSARIO.DOC") 'Abre o Arquivo do Word
    Word.Bold 'ativa texto em negrito
    Word.FontSize 24 'aumenta tamanho da fonte para 24
    Word.Insert CStr(Form1.Text1(1).Text) 'Insere o conteúdo de
Text1(1)(Nome) de Form1
    Word.Insert " " 'Insere um espaço entre o nome e o sobrenome
    Word.Insert CStr(Form1.Text1(0).Text) 'Insere o conteúdo de
Text1(0)(Sobrenome) de Form1
    Word.appmaximize "", 1
    Word.FilePrintPreview 'Visualiza a impressao
    Word.AppActivate "Microsoft Word"
```

```

Exit Sub

trataerro: 'inicia tratamento de erros
If MsgBox("Ocorreu um erro , você quer continuar ?", 52) = 6 Then
    Resume Next
Else
    Exit Sub
End If
End Sub

```

2 - Código associado ao botão Sair:

```

Private Sub Command1_Click()
    End
End Sub

```

Ao executar sua aplicação você obterá a tela da figura 2.0 abaixo:

Fig.2.0

Agora ao clicar no botão Automação OLE com Word 7.0 o Visual Basic irá ativar o Word e visualizar a carta para o empregado corrente.

Para o primeiro registro iremos obter a carta como a figura 3.0 abaixo

Fig.3.0

Nada mal, não é mesmo ?

Você acredita que podemos usar o Word para imprimir um relatório de uma tabela do nosso banco de dados ? Não ??? bem, então vamos provar...

Nosso objetivo será imprimir um relatório com o Nome, Sobrenome e Endereço da tabela empregados do banco de dados **Nwind**, usando o Word 6.0 via automação OLE. Parece complexo, mas na verdade basta definir o objeto **Wordbasic** como fizemos no exemplo anterior. Para melhorar a formatação iremos imprimir nas células de uma tabela previamente preparada para receber os dados. Mão a obra:

1 - Primeiro vamos criar um novo botão de comando para imprimir o relatório, como abaixo:

Objeto	Propriedade	Configuração
CommandButton	Name Caption	command2 "&Relatório"

2 - A seguir iremos associar ao evento Click do botão o seguinte código:

Código associado ao botão que ativa automação OLE:

```
Private Sub Command2_Click()  
    Call relat  
End Sub
```

Ao executar sua aplicação você obterá a tela da figura 4.0 abaixo:



Fig.4.0

Observe que ao clicar no botão o procedimento **relat** é ativado, então devemos criar uma **procedure** com o nome de **relat**. Para isto clique em **Tools-> Add Procedure** e informe o nome **Relat** clicando a seguir em o **OK** e insira código abaixo :

Código associado ao botão que ativa automação OLE:

```
Public Sub relat()  
  
Dim db As Database  
Dim rs As Recordset  
Dim wordobj As Object  
Dim i As Integer  
Dim texto As String  
Dim ColumnWidths(3) As String
```

```
'ativa tratamento de erros
On Error GoTo erros

'abre os arquivos e cria um Snapshot
Set db = DBEngine.Workspaces(0).OpenDatabase("C:\access\exemplos\Nwind.mdb")
Set rs = db.OpenRecordset("Empregados", dbOpenSnapshot)

'cria objeto wordbasic
Set wordobj = CreateObject("Word.Basic")
With wordobj
    .filenew
    .Bold
    .Insert " AUTOMAÇÃO OLE - WORD 7.0 "
    .centerpara
    .Style "Normal"
End With

'inserir uma tabela para receber os dados
wordobj.tableinserttable NumColumns:=3, NumRows:=2, InitialColWidth:=4"

For i = 0 To 1
    With wordobj
        .TableSelectColumn
        .TableColumnWidth columnWidth:=ColumnWidths(i)
        .NextCell
    End With
Next

'salta uma linha
With wordobj
    .NextCell
    .NextCell
    .NextCell
End With

'Prepara o cabeçalho do Relatório
With wordobj
    .FontSize 14
    texto = "Sobrenome"
    .Insert texto
    .NextCell
    texto = "Nome"
    .Insert texto
    .NextCell
    texto = "Endereço"
    .Insert texto
    .NextCell
End With

'salta uma linha da tabela
With wordobj
    .NextCell
    .NextCell
    .NextCell
End With

wordobj.FontSize 12
wordobj.Style "Normal"

'inserir os dados na tabela
While Not rs.EOF
    With wordobj
        texto = rs("sobrenome")
        .Insert texto
        .NextCell
        texto = rs("nome")
        .Insert texto
        .NextCell
        texto = rs("endereço")
        .Insert texto
        .NextCell
        .Insert " "
    End With
    rs.MoveNext
Wend
```

```

End With
rs.MoveNext
Wend

With wordobj
.appmaximize "", 1
.FilePrintPreview 'Visualiza a impressao
.AppActivate "Microsoft Word"
End With

'libera o objeto
Set wordobj = Nothing

Exit Sub

erros: 'inicia tratamento de erros
If MsgBox("Ocorreu um erro , você quer continuar ?", 52) = 6 Then
Resume Next
Else
Exit Sub
End If
End Sub

```

Devemos obter algo parecido com a figura 5.0 abaixo:

AUTOMAÇÃO OLE - WORD 7.0		
Sobrenome	Nome	Endereço
Daylio	Nancy	507 - 20th Ave. E. Apt. 2A
Fuller	Andrew	908 W. Capital Way
Leverling	Janet	722 Moss Bay Blvd.
Peacock	Margaret	4110 Old Redmond Rd.
Buchanan	Steven	14 Garrett Hill
Suyama	Michael	Coventry House Miner Rd.
King	Robert	Edgeham Hollow Winchester Way
Callahan	Laura	4726 - 11th Ave. N.E.
Dodsworth	Anne	7 Houndstooth Rd.

Fig.4.0

Usando Automação OLE com o Excel - Exemplo Prático.

Bem, agora veremos como é fácil criar uma planilha no Excel 5.0 com os dados de uma tabela usada por meu programa em Visual Basic.

Para começar precisaremos de um objeto 'Planilha' do Excel para receber os dados da tabela. Vamos usar o mesmo banco de dados do exemplo anterior (Automação OLE com Word 7.0).

Iremos acrescentar um novo botão de comando ao formulário e associar o código que irá exportar os dados da tabela empregados para o Excel 5.0.

1 - Primeiro vamos criar um novo botão de comando para imprimir o relatório, como abaixo:

Objeto	Propriedade	Configuração
CommandButton	Name Caption	command3 "&Exportar"

2-A seguir iremos associar ao evento Click do botão o seguinte código:

Código associado ao botão que ativa automação OLE:

```
Private Sub Command2_Click()  
    Call export  
End Sub
```

15) GRÁFICOS NO VB

Aprenda como criar gráficos no Visual Basic 5.0 usando o MSChart. Veja como criar gráficos de barras, torta, pizza, 3D, de linha, etc. Configuração e utilização, tudo com exemplos para esclarecer todas as suas dúvidas.

Introdução.

Para construir gráficos em sua aplicação você pode usar o Crystal Reports, o Excel via automação OLE ou o controle gráfico do VB, o **MSChart**.

O **MSChart** permite apresentar seus dados em forma de gráfico como uma planilha do Excel (linhas e colunas); gráficos tridimensionais, gráficos de torta, barra, pizza, etc.

Antes de iniciar você deve adicionar o controle **Microsoft Chart Control** ao seu projeto da seguinte maneira:

Selecione a opção Components do menu Project, selecione o controle Microsoft Chart Control e clique em OK e adicione o controle ao seu formulário.

A figura 1.0 abaixo mostra a barra de ferramentas e a configuração inicial do **MSChart**:



Fig. 1.0 - Barra de Ferramentas

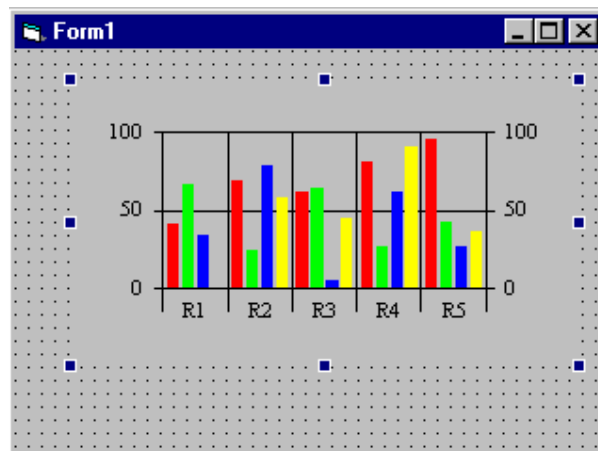
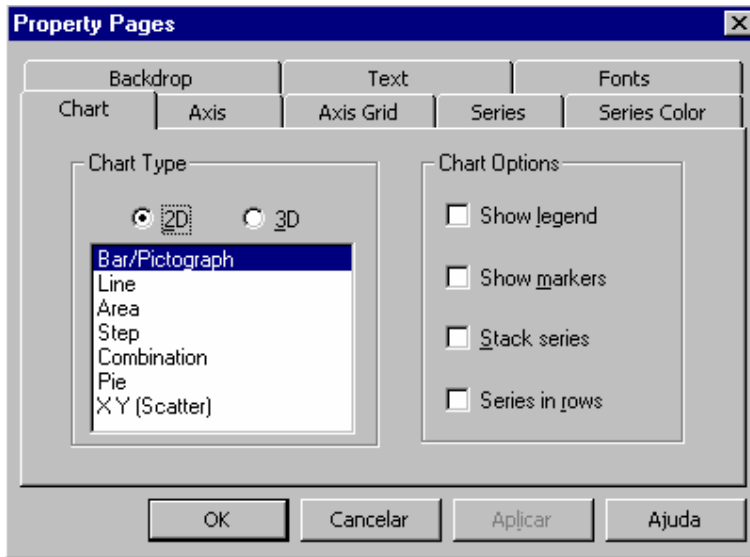


Fig. 2.0 - MSChart no Formulário - Configuração Inicial

Agora basta definir a fonte de dados que alimentará o gráfico, geralmente uma tabela, e fazer a configuração do Controle **MSChart**.

Para abrir um diálogo de propriedades do controle, clique com o botão direito do mouse sobre o mesmo, selecione a opção **Properties** e a janela para configuração surge como mostrada na figura 2.0 abaixo:



Cada aba permite configurar diversas propriedades utilizadas na confecção de um gráfico.

Fig.3.0 - Pág. de Propriedades do MSChart

O **MSChart** não é lá aquela maravilha, mas consegue resolver, eu diria uns 70% dos problemas relacionados a confecção de gráficos no Visual Basic.

Utilização e Configuração.

Vejamos agora as propriedades do **MSChart** e de como configurá-lo.

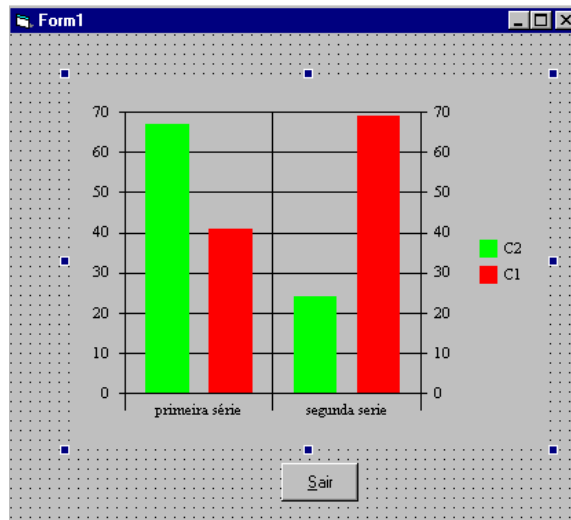


Fig. 4.0

Properties - MSChart1	
MSChart1 MSChart	
Alphabetic	Categorized
Height	3990
HelpContextID	0
Index	
Left	645
MousePointer	0 - VtMousePointerDefault
RandomFill	True
Repaint	True
Row	2
RowCount	2
RowLabel	
RowLabelCount	1
RowLabelIndex	1
SeriesColumn	2
SeriesType	1 - VtChSeriesType2dBar
ShowLegend	True
TabIndex	0
TabStop	True
Tag	
TextLengthType	0 - VtTextLengthTypeVirtue
TitleText	
ToolTipText	
Top	480
Visible	True
WhatsThisHelpID	0

A configuração inicial do controle gráfico apresenta uma série de quatro barras em cinco colunas com legendas R1 a R5. Esta seqüência de dados é definida pela propriedade **RowCount**. A quantidade de colunas é definida pela propriedade **ColumnCount**. Na figura acima alteramos **RowCount** para 2 e **ColumnCount** também para 2 e ainda definimos a propriedade **RowLabel** para cada coluna e a propriedade **ShowLegend** para True.

A propriedade ChartType permite obter os seguintes tipos de gráficos:

- Barra - Valores de 0 e 1
- Linha - Valores de 2 e 3
- Áreas - Valores 4, 5, 6 e 7
- Combinação - Valores 8 e 9
- Torta - Valor 14
- XY - Valor 16

Aba Chart

Chart Type - Permite escolher o tipo de gráfico usado

Chart Options

- | | |
|----------------|---|
| Show Legend | - Mostra a legenda no gráfico |
| Show Markers | - Marcas sobre cada coluna do gráfico |
| Stack Series | - Empilhar as colunas |
| Series em rows | - Inverte apresentação, colunas em linhas |

Aba Axis e Axis Grid

Existem diversas propriedades que permitem configurar os eixos de um gráfico como:

Espessura e cor de linhas

Definição da escala - Valor Máximo e Mínimo

Divisão da Escala - Valor Maior e Menor

Aba Series e Series Color

Permitem configurar diversas propriedades relacionadas às séries do gráfico e a cor interior e da borda de cada série

Aba BackDrop

Configura os elementos de fundo e da borda do gráfico. O elemento a ser configurado é selecionado na lista **Property Name**

Aba Text

Através da lista **Property Name** podemos selecionar o título, o rodapé, títulos dos eixos X, Y ou de um segmento eixo Y e definir um nome para cada item, inclusive definindo o alinhamento e a orientação do texto através do grupo **Alignment** e **Orientation**.

Aba Fonts

Permite definir a fonte, estilo, tamanho e efeitos de cor.

Além destas propriedades temos as seguintes:

- | | |
|--------------------|--|
| Row | - especifica a linha corrente |
| RowCount | - Determina o número de seqüência de dados |
| RowLabel | - define o rótulo de dados da linha corrente |
| Data | - Permite a leitura e a atribuição dos valores de dados ao gráfico |
| Column | - Define a coluna ativa. |
| ColumnCount | - Define o número de colunas ativas |
| ColumnLabel | - Define a legenda para a coluna ativa |
| BorderStyle | - Define a borda do gráfico |

ChartData - Permite atribuir valores às seqüências de dados a partir de uma matriz (array) de duas dimensões.

Exemplos de Utilização

Vejam os dois exemplos de utilizando o **MSChart**:

1 - Vamos construir um gráfico cujo objetivo é mostrar a evolução das receitas e despesas mensais. Nosso gráfico deverá ter a seguinte configuração:

Título do Gráfico - Receitas e Despesas - Ano 1998

Título do Eixo X - Meses

Título do Eixo Y - Reais - R\$

Seqüências de Dados - Duas seqüências : Receitas e Despesas

Mostrar Legendas - Azul para Receitas e Vermelha para despesas

gráfico deverá ter borda com sombra.

Número de linhas de seqüências de dados - Três, representando os meses: Agosto, Setembro e Outubro.

O jeito do gráfico depois de pronto deverá ser o da figura 5.0

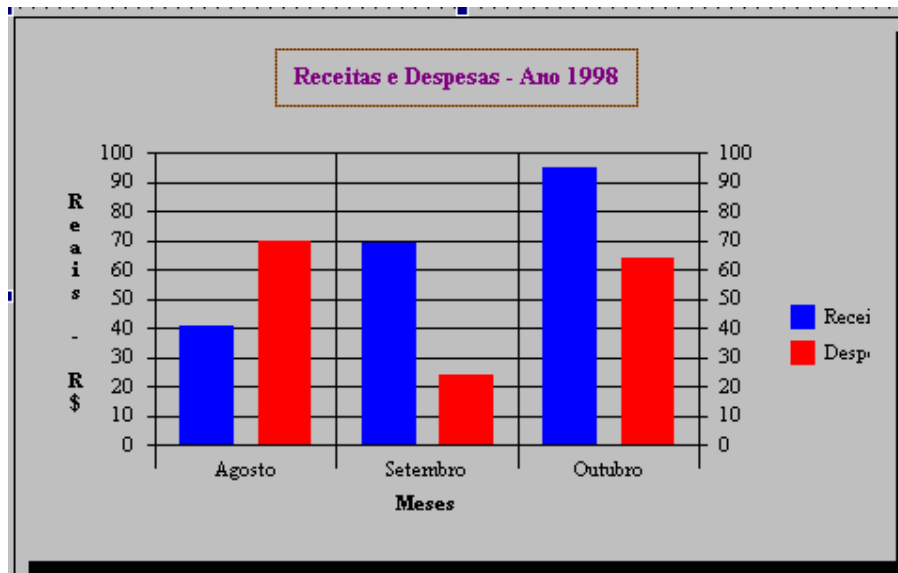


Fig. 5.0

Agora vamos por a mão na massa e montar o gráfico . Iremos fazer isto em tempo de projeto.

Inicie um novo projeto no Visual Basic. Grave o formulário Form1 como **Gráfico**

Selecione no Menu **Project** a opção **Components** e marque **Microsoft Chart Control**

No menu **View** opção **ToolBox** insira o controle OCX (ver fig 1.0) no seu formulário.

Ative a janela de configuração de propriedades (ver figura 2.0), na aba **Text** , **Property Name** Title em Text digite o título do gráfico - Receitas e Despesas - Ano 1998, selecione alinhamento centralizado e orientação de texto na vertical

Como no passo anterior selecione na Aba **Text** em Property Name selecione X axis Title e em Text informe - Meses com alinhamento centralizado e orientação vertical

Ainda como no passo anterior selecione a aba **Text** em Property Name selecione Y axis Title e em Text informe - Reais - R\$, com alinhamento centralizado e orientação na horizontal

Defina a propriedade **ShowLegend** para True

Defina **ColCount** igual a 2 , pois teremos duas séries , Receitas e Despesas

Defina **RowCount** igual a 3 , pois teremos 3 seqüências de dados - Agosto, Setembro e Outubro

Agora faça **Column** igual a 1 e **Row** igual a 1 . Isto define que iremos tratar a primeira seqüência de dados e a primeira série que será a receita:

Defina **RowLabel** como Agosto

Informe na propriedade **ColumnLabel** - Receita

Informe na propriedade **Data** o valor de 40

Mantendo Row igual a 1 faça Column igual a 2 . Pois iremos tratar a série Despesa
 Informe na propriedade **ColumnLabel** - Despesa
 Informe na propriedade **Data** o valor de 50
 Agora faça **Column** igual a 1 e **Row** igual a 2
 Defina **RowLabel** como Setembro
 Informe na propriedade **Data** o valor de 50
 Mantendo Row igual a 2 faça Column igual a 2 .
 Informe na propriedade **Data** o valor de 26
 Agora faça **Column** igual a 1 e **Row** igual a 3.
 Defina **RowLabel** como Outubro
 Informe na propriedade **Data** o valor de 95
 Mantendo **Row** igual a 3 faça **Column** igual a 2 .
 Informe na propriedade **Data** o valor de 26
 Na aba **Axis** selecione Y Axis e em Axis Scale desative Automatic Scaling, definindo a seguir:
 Minimum -> 0
 Maximum -> 100
 Major divisions -> 10
 Minor divisions -> 1
 Na aba **Series Color** defina a series Receita com a cor Azul e Despesa com a cor Vermelha
 Na aba **BackDrop** , Property Name -> Title, define a cor do título e em Border-> Style defina como Single line , Width igual a 1 pt e color como verde

Bem, acho que com isso terminamos o gráfico, agora basta executar o projeto.

Projeto - Alunos.

A esta altura você deve estar pensando - "... Po se toda vez que eu tiver que fazer um gráfico tiver que usar este processo trabalhoso eu desisto. ! ", e você esta certo não tem cabimento , mas existe uma maneira bem mais fácil de fazer isto. Que tal alimentar os dados do seu gráfico diretamente de seus arquivos de dados ?

Vamos mostrar a seguir um projeto simples que indica como fazer isto. Iremos criar um gráfico com os dados existentes em nossa tabela **tblalunos** para obter um gráfico que indique o número de alunos matriculados por série.

Para facilitar criamos uma consulta que calcula a quantidade de alunos por série e a armazenamos no banco de dados com o nome de **SEL_SERIE** , a instrução SQL da consulta é a seguinte:

```
SELECT DISTINCTROW TBLALUNOS.SERIE, Count(TBLALUNOS.CODALUNO) AS ALUNOS
FROM TBLALUNOS
GROUP BY TBLALUNOS.SERIE;
```

O código do projeto é dado abaixo:

```
Private Sub Form_Load()
Set db = DBEngine.Workspaces(0).OpenDatabase("d:\escola\escola.mdb")
End Sub
```

```
Option Explicit
Dim db As Database
Dim rs As Recordset
Dim qry As QueryDef
Dim reg As Integer
Dim i As Integer
```

```
Private Sub grafico_Click()
'O objeto Mschart foi chamado de graf1
```

```
Set qry = db.QueryDefs("sel_series")
Set rs = qry.OpenRecordset
```

```
rs.MoveLast
```

```

rs.MoveFirst

reg = rs.RecordCount

graf1.chartType = 1 'barra em duas dimensões
graf1.ShowLegend = False 'não mostra legenda
graf1.Title = "Relação de Alunos por Turma" 'titulo do gráfico

graf1.ColumnCount = 1 'uma série
graf1.RowCount = reg 'número seqüência de dados
graf1.Visible = True

While Not rs.EOF()

For i = 1 To reg

graf1.Row = i
graf1.RowLabel = rs("SERIE") & " Série "
graf1.Data = rs("alunos")

rs.MoveNext

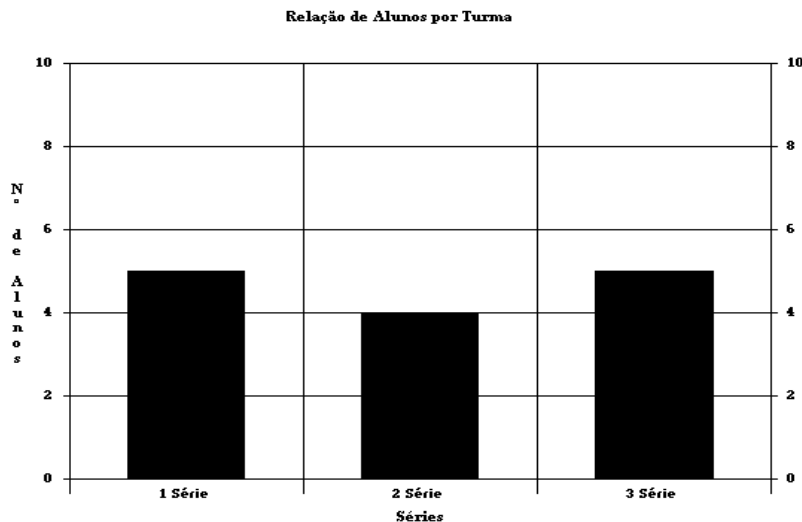
Next

Wend

End Sub

```

O gráfico obtido esta exposto abaixo:



Bem, deu para perceber que se quisermos algo mais requintado teremos que usar produtos de terceiros pois o **Mschart** deixa a desejar mas eu diria que para coisas simples da para quebrar o galho.

Ah! ia esquecendo, **como imprimir o gráfico ???** Insira o controle Picture no seu formulário e deixa-o invisível, e seguir insira um botão de comando com o nome de imprimir e acrescente o seguinte código a ele:

```

Private Sub Command2_Click()

graf1.EditCopy

Picture1.Picture = Clipboard.GetData()
Printer.PaintPicture Picture1.Picture, 0, 0

```

```
Printer.EndDoc  
Picture1.Picture = LoadPicture()  
  
End Sub
```

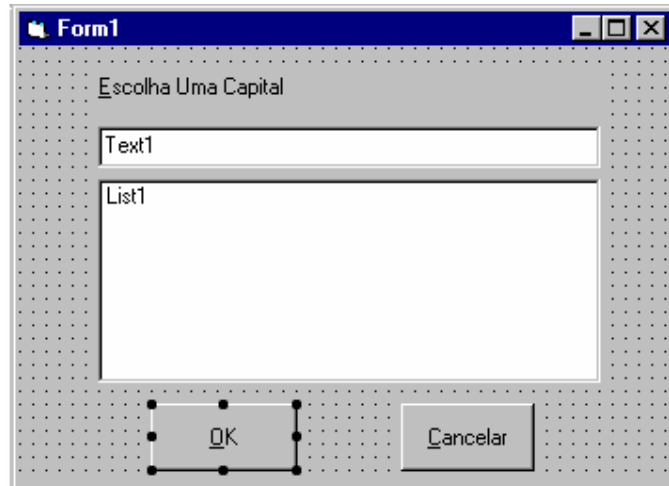
16) OPÇÕES DE BUSCA RÁPIDA (Dicas)

Vamos mostrar como criar uma janela de localização onde à medida que o usuário digita o nome a procurar ,o sistema se antecipe ao que ele esta escrevendo mostrando na tela o item digitado.
Muito difícil !!!

Não, siga o roteiro passo a passo:

Crie um novo formulário (form), desenhe nele uma caixa de texto e, logo abaixo dela uma caixa de listagem.

Adicione dois botões de comando, um Cancelar para encerrar o programa e outro OK para mostrar como se determina o valor do texto selecionado na caixa de listagem.(Veja figura abaixo.)



Código para o botão **Cancelar**

```
Sub Cancelar_Click
    end
End Sub
```

Código para o botão **OK**

```
Sub OK_Click
    msgbox "Você selecionou " + list1.text
End Sub
```

Na rotina do evento **Form_Load** , use o comando **AddItem** para preencher a caixa de listagem com alguns nomes de cidades:

```
Sub Form_Load( )
    list1.AddItem "Salvador"
    list1.AddItem "Fortaleza"
    list1.AddItem "Recife"
    list1.AddItem "São Paulo"
    list1.AddItem "Rio de Janeiro"
    list1.AddItem "Natal"
    list1.AddItem "Vitória"
    list1.AddItem "Porto Alegre"
    list1.AddItem "Manaus"
    list1.AddItem "Porto Velho"
    list1.AddItem "Belém"
End Sub
```

No evento Text1_Change, digite a rotina de busca, ela entra em ação a cada caractere escrito e tenta localizar na caixa de listagem o item iniciado pelo conjunto de letras já digitado.

```
Sub Text1_Change( )
    Search$=UCase$(text1.text)
    Searchlen=len(Search$)
    If Searchlen then
        For i=0 to list1.ListCount-1
            if Ucase$(Left$(List.List(i),Searchlen))=Search$ then
                List1.ListIndex=i
                Exit For
            End if
        Next
    Endif
End Sub
```

A localização do item coincidente é feito pelo laço For/Next que transfere a barra de seleção para o item correspondente (List1.ListIndex=i), saindo do loop com Exit For.

Pronto ! Agora Salve o seu projeto e teste-o. Que tal ? Simples, não ?

Fonte:Revista Exame Informática

17) HOT SPOTS NO VISUAL BASIC (Dicas)

Que tal transformar uma *picture box* do VB em um *Hot Spot* ? HOT SPOTS ???

Um *Hot Spot* nada mais é do que uma figura que possui regiões com eventos distintos para o **clique**. São também chamadas de áreas quentes ou hipergráficos.

O recurso é simples : basta criar dentro da *picture box* um vetor - *array* - de *labels* transparentes. Cada elemento do vetor terá um *clique* diferente.

Vamos ao roteiro passo a passo:

Inicie um novo projeto no Visual Basic - o *Form1* é criado automaticamente. De a ele o nome de **frm-princ**.

Adicione ao *Form1* uma *picture box* com um arquivo *bitmap*(BMP) ou *metafile*(WMF) de sua preferência. Vamos usar o arquivo *COMPUTER.WMF* do diretório VB\METAFILE\BUSINESS. De o nome a *picture* de *PI_Princ*.(Veja Figura).



Adicione um *label* sobre *PI_Princ* com as seguintes propriedades:

```
BackStile = 0      'Transparent
BorderStyle = 1    'Fixe Single
Caption =          'Deixe em branco
Name = LB_Rg
```

Crie um vetor com o controle *LB_Rg* com quatro índices.

Coloque uma *label* sobre o teclado, outra sobre o video, outra sobre o mouse e outra sobre a cpu.

Crie uma *label* na base do formulário **frmprinc** com o nome de **lab_qual** e na sua propriedade *Caption* informe: SISTEMA TESTE.

Crie uma *label* no topo do formulário **frmprinc** com o nome de **lab_nome** e na sua propriedade *Caption* inform: Clique em uma área para ver o nome do componente.

Adice o código seguinte à *label* *LB_Rg*:

```
Sub LB_Rg_Click (Index as Integer)
Dim qual as String
Select Case Index
Case 0
qual = "Teclado"
Case 1
qual = "VÍdeo"
Case 2
qual = "CPU"
```



```

Case 3
qual = "Mouse"
End Select
lab_qual.Caption = " " + qual + " "
End Sub

```

Adicione o seguinte código à *picture* PI_Princ:

```

Sub PI_Princ_Click( )
LB_status.Caption= " "
End Sub

```

10. Deixe a borda dos *labels* visíveis em tempo de desenho ocultando-as somente em tempo de execução. Fazemos isso quando o formulário é carregado:

```

Sub Frm_Load
Dim i as String
For i=0 to 3
    LB_Rg(i).BorderStyle = 0 'none
next i
End Sub

```

Pronto acabamos.

Agora salve e teste o seu programa. Há inúmeras aplicações para esta técnica ,basta você usar a imaginação.

Dica: Para trabalhar com áreas irregulares combine várias *labels* até conseguir a área desejada. Lembre-se que a instrução *Select Case* pode conter cláusulas como : Case 1 to 4,7,10 isto nos dá mais flexibilidade.

Ah! se quiser pode usar outros eventos associados à *label*; que tal *MouseMove* ?
 Fonte:Revista Fórum Access

18) FORMULÁRIOS COM SUBFORMULÁRIOS NO VB ? (Dicas)

Se você utiliza o Visual Basic e conhece a facilidade com o Access cria um formulário com um sub-formulário já deve ter ficado com dor de cotovelo, afinal o VB não possui tal recurso. Será que não possui mesmo ? E se... criarmos algo parecido... vamos lá.

A idéia básica é montar um projeto com os seguintes elementos:

- Um formulário principal - Form1
- Um controle DBGrid - DBGrid1
- Dois controles Data Control - Data1 e Data2
- Um arquivo de dados padrão Access. Usaremos o NWIND.MDB do Access. (Usaremos as tabelas Pedidos e Detalhes do Pedido)
- Um botão de comando para encerrar o programa.
- Duas Caixas de Texto para visualizar os dados
- Duas Labels - etiquetas para identificar as caixas de texto.

1 - Inicie um novo projeto no Visual Basic. Grave o formulário como Form1

2 - Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo:

Tabela 1.0 - Objetos e propriedades do formulário Agenda

Objeto	Propriedade	Configuração
Form	Name	Form1
	Caption	"Formulário/Subformulário"
Data	Name	Data1
	Caption	Tabela -> Pedidos (NWIND.MDB)
	Connect	Access
	DatabaseName	C:\ACCESS2\APLEXEMP\NWIND.MDB
	RecordSetType	1-Dynaset
	RecordSource	Pedidos
Data	Name	Data2
	Caption	Data2
	Connect	Access
	DatabaseName	C:\ACCESS2\APLEXEMP\NWIND.MDB
	RecordSetType	1-Dynaset
	RecordSource	Detalhes do Pedido
	Visible	False
(4) DBGrid	Name	DBGrid1
	Caption	Detalhes dos Pedidos(NWIND.MDB)
	DataSource	Data2
	Column1	Número do Pedido
	Column2	Preço Unitário
	Column3	Quantidade
TextBox	Name	Text1
	Caption	" "
	DataField	Número do Pedido
	DataSource	Data1
TextBox	Name	Text2
	Caption	" "
	DataField	Nome do destinatário
	DataSource	Data1
Label	Name	Label1
	Caption	Número Pedido

	AutoSize	True
Label	Name	Label2
	Caption	Destinatário
	AutoSize	True
CommandButton	Name	Command1
	Caption	&Sair

3 - O seu formulário deverá ter o seguinte jeitão:

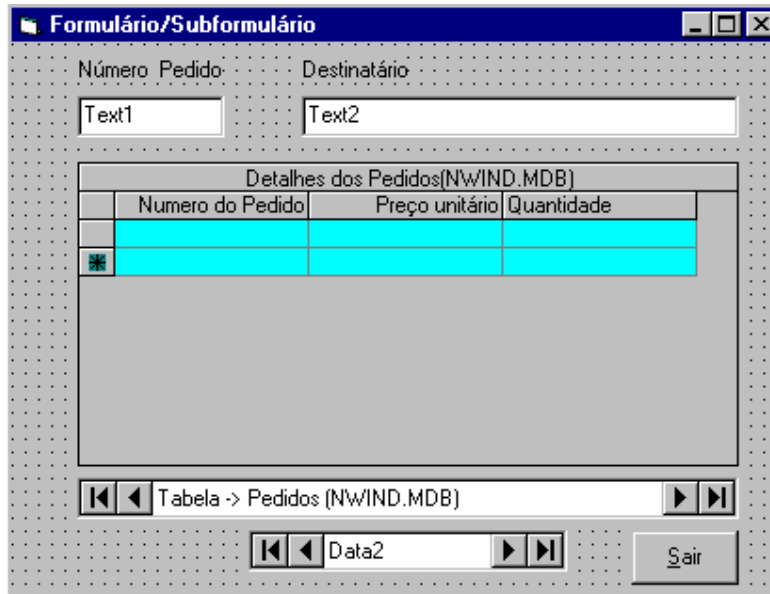


Fig. 1.0

4- Para incluir as colunas no DBGrid faça o seguinte:

- Após configurar os controles data1 e data2 como indicados e ter colocado o controle DBGrid no seu formulário, dimensione-o como na figura 1.0. acima.
- Selecione o controle DBGrid e clique com o botão direito do mouse sobre o mesmo; o menu da figura 2.0 deverá surgir, nele clique na opção **Edit**.
- Clique novamente com o botão direito do mouse sobre o controle DBGrid para obter o menu da figura 3.0.

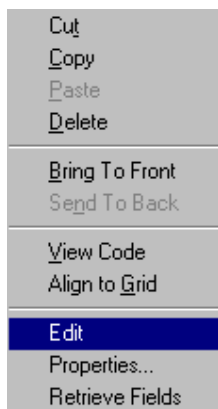


Fig. 2.0

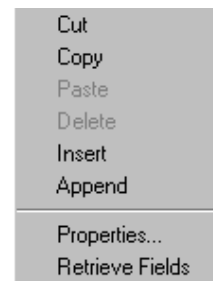


Fig. 3.0

- Nele notamos as opções **Insert**, para inserir colunas em determinada posição e **Append** para inserir após a última coluna.

- Clique em **Insert** para inserir mais uma coluna.
- Agora clique na opção **Properties...** do menu da figura 3.0 . Você deverá obter a tela da figura 4.0, onde na selecionando **Columns** você atribui a cada coluna o campo do **RecordSource** atribuído ao controle **Data2**.

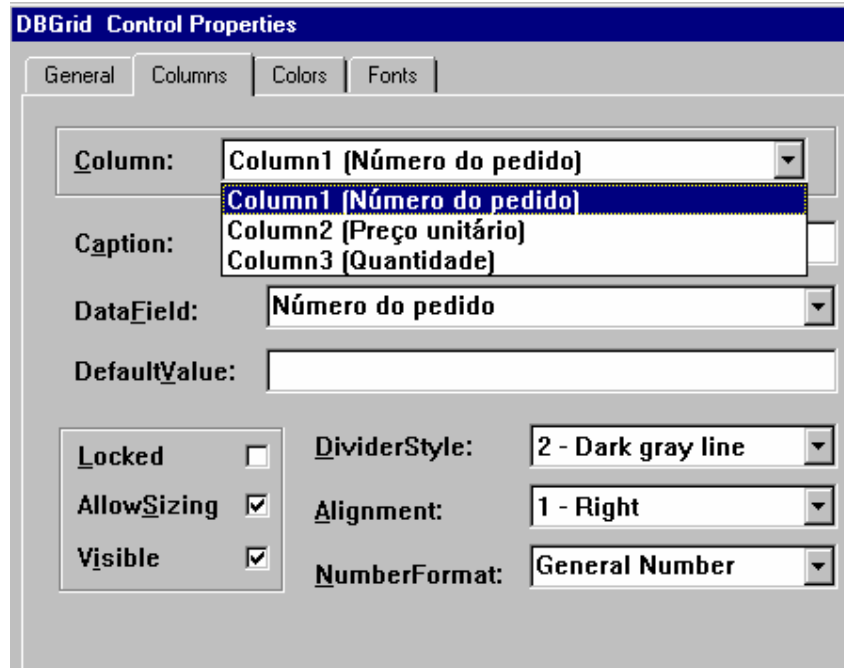


Fig. 4.0

5 - Agora falta somente acrescentar o seguinte código ao evento **reposition** do controle **Data1**:

```
Private Sub Data1_Reposition()
Dim SQL As String
SQL = "SELECT * FROM [detalhes do pedido] WHERE [número do pedido] = " & Text1.Text
Data2.RecordSource = SQL
Data2.Refresh
End Sub
```

6 - E o código para encerrar o programa no botão de comando:

```
Private Sub Command1_Click()
End
End Sub
```

7- Se você fez tudo direitinho ao executar o projeto obterá a tela da figura 5.0, e ao clicar para movimentar-se pelos registros da tabela pedido os registros dos detalhes do pedido serão mostrados simulando um subformulário.

Número Pedido: 10001

Destinatário: Mère Paillarde

Detalhes dos Pedidos(NwIND.MDB)			
	Numero do Pedido	Preço unitário	Quantidade
▶	10001	R\$9,80	30
	10001	R\$12,80	40
	10001	R\$38,50	8
	10001	R\$23,00	15
✖			

Tabela -> Pedidos (NwIND.MDB)

Sair

Fig. 5.0

- E Então ? Com o VB podemos ou não criar formulário / Subformulário ???

19) CRIANDO UM SCREEN SAVER NO VB (Dicas)

Vamos criar um programa no Visual Basic e transformá-lo em um Screen Saver. Pode parecer complexo, mas não é.

Primeiro vamos criar um programa que vá enchendo a tela de pontos (o código foi tirado de um exemplo do próprio Visual Basic), e, a seguir fazê-lo funcionar como um Screen Saver.

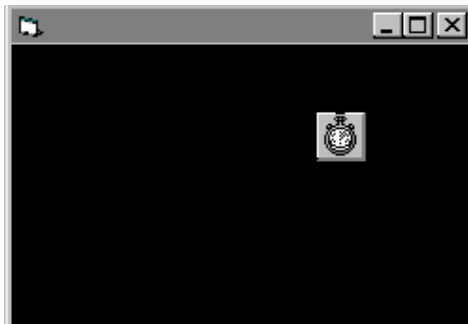
Para isto basta atentar-mos para os seguintes detalhes:

1- Ao criar o programa o form deve ocupar toda a janela e não ter título, nem os botões para maximizar ou minimizar devem estar habilitados.

Valores das propriedades do Form1

Propriedade	Valor
Caption	" "
ControlBox	False
BackColor	&H00000000& (preto)
MaxButton	False
MinButton	False
WindowState	2

2 - Insira o controle Timer no formulário (fig 1.0) e defina as propriedades



Valores das Prop. de Timer1

Propriedade	Valor
Name	Timer1
Enabled	True
Interval	60

Fig. 1.0

3 - A seguir insira o código abaixo no evento timer do temporizador: (Aqui fica a seu critério, use a sua imaginação...)

```
Private Sub Timer1_Timer()
    Dim CX, CY, Msg, XPos, YPos          ' Declare variables.

    ScaleMode = 3                        ' Set ScaleMode to
                                        ' pixels.
    DrawWidth = 5                        ' Set DrawWidth.
    ForeColor = QBColor(4)               ' Set background to red.
    FontSize = 24                        ' Set point size.
    CX = ScaleWidth / 2                  ' Get horizontal center.
    CY = ScaleHeight / 2                 ' Get vertical center.
    Cls                                  ' Clear form.
    Msg = "Bom Dia!"
    CurrentX = CX - TextWidth(Msg) / 2   ' Horizontal position.
    CurrentY = CY - TextHeight(Msg)      ' Vertical position.
    Print Msg ' Print message.
    Do
        Counter = Counter + 1
        XPos = Rnd * ScaleWidth          ' Get horizontal position.
        YPos = Rnd * ScaleHeight          ' Get vertical position.
        PSet (XPos, YPos), QBColor(Rnd * 15) ' Draw confetti.

        DoEvents ' Yield to other
```

```

        If Counter > 2000 Then          'contador para limpar a tela
            Counter = 0
            Form1.Cls
        End If

    Loop                                ' processando

End Sub

```

4 - Como o Screen Saver pode ser ativado mais de uma vez, insira o código a seguir no evento Load do formulário para evitar mais de uma instância de seu aplicativo na memória.

```

Private Sub Form_Load()
    If App.PrevInstance Then
        End
    End If
End Sub

```

5 - Para encerrar o programa utilize os eventos MouseMove, MouseDown e KeyDown, pois quando o usuário pressionar algo ou movimentar o mouse o Screen Saver deve ser encerrado. Insira os códigos como descrito abaixo:

```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    End
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    End
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
    End
End Sub

```

6 - Como o evento MouseMove é ativado quando o form for lido pela primeira vez, insira o código abaixo no evento MouseMove para não fechar o programa na sua ativação.

```

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Sin-
gle, Y As Single)
    Static conta As Integer
    If conta > 3 Then
        End
    Else
        conta = conta + 1
    End If
End Sub

```

7 - Agora ative a opção File->Make EXE File... e informe o nome do seu Screen Saver.

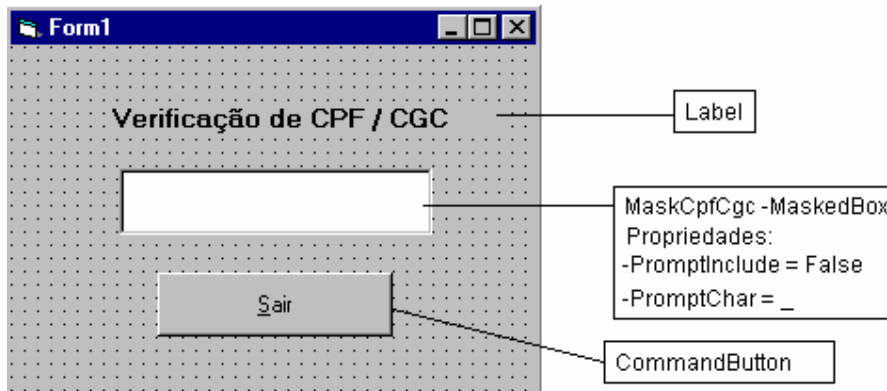
8 - Troque a extensão EXE do arquivo gerado para SCR, e copie o arquivo para o diretório \WINDOWS\SYSTEM

9 - Agora configure a proteção de tela no seu Windows 95 selecionando o nome do seu programa como o protetor de tela.

10 - Se quiser pode incrementar ainda mais este exemplo, mas atenção, eu não testei este exemplo no Windows 3.X, só no Windows 95.

20) ROTINA PARA VALIDAR CPF/CGC NO VISUAL BASIC (Dicas)

Inicie o Visual Basic, Crie um form como abaixo e a seguir insira o código para cada evento do controle **MaskedTextBox**. A seguir insira um módulo com as funções que irão checar o CGC e o CPF e fique a vontade para otimizá-las. Acho que o resto é intuitivo.



Rotinas do formulário Form1

```
Private Sub Maskcpfcgc_LostFocus()
    If Len(Maskcpfcgc.Text) > 0 Then
        Select Case Len(Maskcpfcgc.Text)
            Case Is = 11
                Maskcpfcgc.Mask = "###.###.###-##"
                If Not calculapf(Maskcpfcgc.Text) Then
                    MsgBox "CPF com DV incorreto !!!"
                    Maskcpfcgc = ""
                    Maskcpfcgc.Mask = "#####"
                    Maskcpfcgc.SetFocus
                End If
            Case Is = 14
                Maskcpfcgc.Mask = "##.###.###/####-##"
                If Not ValidaCGC(Maskcpfcgc.Text) Then
                    MsgBox "CGC com DV incorreto !!!"
                    Maskcpfcgc = ""
                    Maskcpfcgc.Mask = "#####"
                    Maskcpfcgc.SetFocus
                End If
        End Select
    End If
End Sub

Private Sub Maskcpfcgc_GotFocus()
    Maskcpfcgc.Mask = "#####"
End Sub

Private Sub Maskcpfcgc_KeyPress(KeyAscii As Integer)
    'se teclar enter envia um TAB
    If KeyAscii = 13 Then
        SendKeys "{TAB}"
        KeyAscii = 0
    End If
End Sub

Private Sub Command1_Click()
    End
End Sub
```


Funções para Validar CPF e CGC**Validar CGC**

```

Public Function CalculaCGC(Número As String) As String

Dim I As Integer
Dim prod As Integer
Dim mult As Integer
Dim digito As Integer

If Not IsNumeric(Número) Then
    CalculaCGC = ""
    Exit Function
End If

mult = 2
For I = Len(Número) To 1 Step -1
    prod = prod + Val(Mid(Número, I, 1)) * mult
    mult = IIf(mult = 9, 2, mult + 1)
Next

digito = 11 - Int(prod Mod 11)
digito = IIf(digito = 10 Or digito = 11, 0, digito)

CalculaCGC = Trim(Str(digito))

End Function

Public Function ValidaCGC(CGC As String) As Boolean
If CalculaCGC(Left(CGC, 12)) <> Mid(CGC, 13, 1) Then
    ValidaCGC = False
    Exit Function
End If

If CalculaCGC(Left(CGC, 13)) <> Mid(CGC, 14, 1) Then
    ValidaCGC = False
    Exit Function
End If

ValidaCGC = True

End Function

```

2- Validar CPF**Function calculacpf(CPF As String) As Boolean**

```

'Esta rotina foi adaptada da revista Fórum Access
On Error GoTo Err_CPF
Dim I As Integer          'utilizada nos FOR... NEXT
Dim strcampo As String    'armazena o CPF que será utilizada para o cálculo
Dim strCaracter As String 'armazena os dígitos do CPF da direita para a esquerda
Dim intNumero As Integer  'armazena o dígito separado para cálculo (uma a uma)
Dim intMais As Integer    'armazena o dígito específico multiplicado pela sua base
Dim lngSoma As Long       'armazena a soma dos dígitos multiplicados pela sua base(intmais)
Dim dblDivisao As Double  'armazena a divisão dos dígitos * base por 11
Dim lngInteiro As Long    'armazena inteiro da divisão
Dim intResto As Integer   'armazena o resto
Dim intDig1 As Integer    'armazena o 1º dígito verificador
Dim intDig2 As Integer    'armazena o 2º dígito verificador
Dim strConf As String     'armazena o dígito verificador

lngSoma = 0

```

```

intNumero = 0
intMais = 0
strcampo = Left(CPF, 9)

'Inicia cálculos do 1º dígito
For I = 2 To 10
    strCaracter = Right(strcampo, I - 1)
    intNumero = Left(strCaracter, 1)
    intMais = intNumero * I
    lngSoma = lngSoma + intMais
Next I
dblDivisao = lngSoma / 11

lngInteiro = Int(dblDivisao) * 11
intResto = lngSoma - lngInteiro
If intResto = 0 Or intResto = 1 Then
    intDig1 = 0
Else
    intDig1 = 11 - intResto
End If

strcampo = strcampo & intDig1 'concatena o CPF com o primeiro dígito verificador
lngSoma = 0
intNumero = 0
intMais = 0
'Inicia cálculos do 2º dígito
For I = 2 To 11
    strCaracter = Right(strcampo, I - 1)
    intNumero = Left(strCaracter, 1)
    intMais = intNumero * I
    lngSoma = lngSoma + intMais
Next I
dblDivisao = lngSoma / 11
lngInteiro = Int(dblDivisao) * 11
intResto = lngSoma - lngInteiro
If intResto = 0 Or intResto = 1 Then
    intDig2 = 0
Else
    intDig2 = 11 - intResto
End If
strConf = intDig1 & intDig2
'Caso o CPF esteja errado dispara a mensagem
If strConf <> Right(CPF, 2) Then
    calculacpf = False
Else
    calculacpf = True
End If
Exit Function

Exit_CPF:
    Exit Function
Err_CPF:
    MsgBox Error$
    Resume Exit_CPF
End Function

```

21) DBGRID COM LISTBOX - PERSONALIZE O SEU DBGRID: CAIXA DE LISTAGEM, TOTALIZANDO POR COLUNAS E CONTADOR DE REGISTROS. (Dicas)

Personalizando e incrementando o DBGrid.

Olhe bem para a figura 1.0 abaixo. Não, não é uma montagem o que você está vendo. Você está realmente vendo um projeto usando o controle Dbgrid juntamente com alguns recursos que muitos pensavam que somente outros controles OCX mais poderosos poderiam usar. Na verdade usamos alguns truques para driblar as limitações do DBGrid, mas que da para fazer você mesmo vai ver com os seus próprios olhos.

As dicas não são de minha autoria, mas adaptadas de alguns exemplos da própria Apex e de outros garimpos pela Web.

Uso do Controle ListBox em um Controle DBGrid.
Ao Clicar na célula a lista é mostrada com as opções.
Você seleciona e atualiza o Grid com a opção selecionada.

Totalizador por Colunas: você altera um valor no Grid e o resultado é atualizado e mostrado no seu projeto.

Contador de Registros dinâmico: Mostra a posição do registro atual, e o total de registros da base.

Aluno	Curso	Valor
José Da Silva	Matemática	R\$218,00
Ana Lima Bueno	Inglês	R\$315,00
Carlos Rodrigues	Agronomia	R\$412,00
Vera Lucia Sá	História	R\$215,00
Tadeu Ingor	Química	R\$482,00
Marcos Ribeiro	Psicologia	R\$502,00
Miriam Silveira	Química	R\$218,00
Janice Sbroggio	Biologia	R\$150,00
Andre Lara Olstrich	Matemática	R\$100,00
Irma Rocha Prado	Arquitetura	R\$650,00
Jaime Strochi	Biologia	R\$315,00
Samanta Lisching	Educação Física	R\$318,00

Total : 4.873,00

Registro : 6 / 14

Interface Visual do Projeto.

Lista

Total :

Data1

A coisa é mais simples do que parece, e, na verdade a quantidade de código é pequena.

Para montar o formulário acima descrito observe os seguintes passos:

1 - Inicie um novo projeto no Visual Basic. Grave o formulário Form1 como frmgrid.

2 - Adicione ao Form1 os objetos e configure as propriedades conforme a tabela 1.0 abaixo :

Tabela 1.0 - Objetos e propriedades do formulário frmgrid

Objeto	Propriedade	Configuração
Form	Name	frmgrid
	Caption	"DBGGrid Personalizado"
ListBox	Name	lista
SSPanel	Caption	" "
	Name	PnlTotal
Data	Name	datal
	DatabaseName	Grid_List
	RecordSource	teste
DbGrid	Caption	"DBGGrid Personalizado"
	Name	grid
	Column0	Aluno
	Column1	Curso
	Column2	Valor
	DataSource	Datal

Obs.: A base de dados (Grid_List) foi criada apenas para o exemplo acima e pode ser substituída a seu critério.

Foi utilizada uma tabela chamada *teste* com a seguinte estrutura:

aluno, text, 50
curso, text, 30
valor, Currency

Código do Projeto.

Declaração de variáveis visíveis em todo o formulário.

```
Option Explicit
Private numero_registros As Variant 'contador de registros
Const colindex_lista = 1 'coluna onde será mostrada a lista de itens
```

Evento Reposition do controle de dados

```
Private Sub Datal_Reposition() 'ao iniciar

'-----atualiza o total no controle SSPanel-----
If pnltotal.Caption = "" Then
    Soma_Colunas Grid, Datal, "valor", pnltotal, "##,##0.00" 'chama rotina de soma
End If

'-----verifica se o contador esta vazio-----
If IsEmpty(numero_registros) Then
    numero_registros = Conta_Registros(Datal) 'conta os registros
End If
With Datal
    If .Recordset.RecordCount Then 'se há registros mostra a posição
        .Caption = " Registro : " & (.Recordset.AbsolutePosition + 1) & " / " & numero_registros
    Else
        .Caption = " O arquivo esta vazio "

```

```

End If
End With
End Sub

```

Evento Validate do controle de dados

```

Private Sub Data1_Validate(Action As Integer, Save As Integer)

    'dependendo da ação incrementa ou decrementa o contador de registros
    Select Case Action
        Case vbDataActionAddNew 'inclusão
            numero_registros = numero_registros + 1
        Case vbDataActionDelete 'exclusão
            numero_registros = numero_registros - 1
    End Select

End Sub

```

Carga do formulário

```

Private Sub Form_Load()
    'formata o grid
    Grid.Columns(0).Width = 2500 'nome
    Grid.Columns(1).Width = 1550 'curso
    Grid.Columns(2).Width = 1000 'valor
    Grid.Columns(0).Alignment = 0 'alinha a esquerda
    Grid.Columns(1).Alignment = 0
    Grid.Columns(2).Alignment = 1 'alinha

    'enche a lista com os itens
    With Lista
        .AddItem "Matemática"
        .AddItem "Agronomia"
        .AddItem "Arquitetura"
        .AddItem "Física"
        .AddItem "Engenharia"
        .AddItem "Inglês"
        .AddItem "Química"
        .AddItem "Biologia"
        .AddItem "Proc. de Dados"
        .AddItem "Psicologia"
        .AddItem "Ed. Física"
    End With

    Data1.Refresh

    'mostra o botão na coluna definida
    Grid.Columns(colindex_lista).Button = True

    'define célula selecionada(curso) como negra
    Grid.MarqueeStyle = dbgHighlightCell

End Sub

```

Procedimento para Somar as colunas

```

Public Sub Soma_Colunas(dbg As DBGrid, dados As Data, nomecampo As String, _
    pnl As SSPanel, num_formato As String)

    Dim soma As Single
    Dim rs As Recordset

    'so para se prevenir
    On Error GoTo trata_erro

```

```

Set rs = dados.Recordset.clone

'rotina para somar a coluna escolhida(valor)
Do Until rs.EOF
    soma = soma + Val(rs(nomecampo))
    rs.MoveNext
Loop
pnl = Format(soma, num_formato)
Exit Sub

trata_erro:
    MsgBox "Ocorreu um erro durante o processamento, verifique ! "

End Sub

```

Evento AfterUpdate do Grid

```

Private Sub Grid_AfterUpdate() 'atualizar após modificar/inserir dados
    Soma_Colunas Grid, Data1, "valor", pnltotal, "##,##0.00"
End Sub

Evento BeforeColEdit
Private Sub Grid_BeforeColEdit(ByVal ColIndex As Integer, ByVal KeyAscii As Integer, Cancel As Integer)
    If ColIndex = colindex_lista Then 'força a seleção da lista
        Cancel = True
        Grid_ButtonClick (ColIndex)
    End If
End Sub

```

Evento ButtonClick do Grid

```

Private Sub Grid_ButtonClick(ByVal ColIndex As Integer)

    Dim coluna As Column

    'mostra a lista abaixo da coluna selecionada
    If ColIndex = colindex_lista Then
        Set coluna = Grid.Columns(ColIndex)

        With Lista
            .Left = Grid.Left + coluna.Left
            .Top = Grid.Top + Grid.RowTop(Grid.Row) + Grid.RowHeight
            .Width = coluna.Width + 15
            .ListIndex = 0
            .Visible = True
            .ZOrder 0
            .SetFocus
        End With

    End If

End Sub

```

Evento Scroll do Grid

```

Private Sub Grid_Scroll(Cancel As Integer)
    'oculta a lista se rolar o grid
    Lista.Visible = False
End Sub

```

Evento DblClick do ListBox

```

Private Sub Lista_DblClick()
    'assume o valor clicado
    Lista_KeyPress vbKeyReturn
End Sub

```

Evento KeyPress do Grid

```
Private Sub Lista_KeyPress(KeyAscii As Integer)
    'verifica a tecla pressionada e dispara ação pertinente
    Select Case KeyAscii
        Case vbKeyReturn
            Grid.Columns(colindex_lista).Text = Lista.Text
            Lista.Visible = False
        Case vbKeyEscape
            Lista.Visible = False
    End Select
End Sub
```

Evento LostFocus do ListBox

```
Private Sub Lista_LostFocus()
    'oculta a lista ao perder foco
    Lista.Visible = False
End Sub
```