

# Lazarus

## 06 - Acesso a Banco de Dados no Lazarus

---

Marcos Roberto Ribeiro



Instituto Federal Minas Gerais - Campus Bambuí

2018

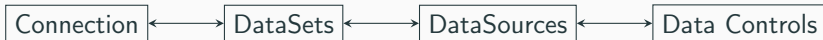
- É possível trabalhar com bancos de dados através do Lazarus de várias maneiras. No entanto, a forma mais comum é através de componentes de acesso a banco de dados;
- O Lazarus oferece por padrão a paleta de componentes *SQLdb*, capaz de acessar diversos tipos de banco de dados;
- Além dos componentes *SQLdb*, podemos instalar componentes de terceiros com a *ZeosLib*<sup>1</sup>.

---

<sup>1</sup><http://sf.net/projects/zeoslib>

# Arquitetura de Componentes para Acesso a Banco de Dados

- O acesso a bancos de dados via componentes no Lazarus normalmente obedece à seguinte arquitetura:



**Connection:** Componente não visual que faz a conexão com o banco de dados;

**Transaction:** Componente não visual que cuida do controle de transação (pode estar incluído no *Connection*);

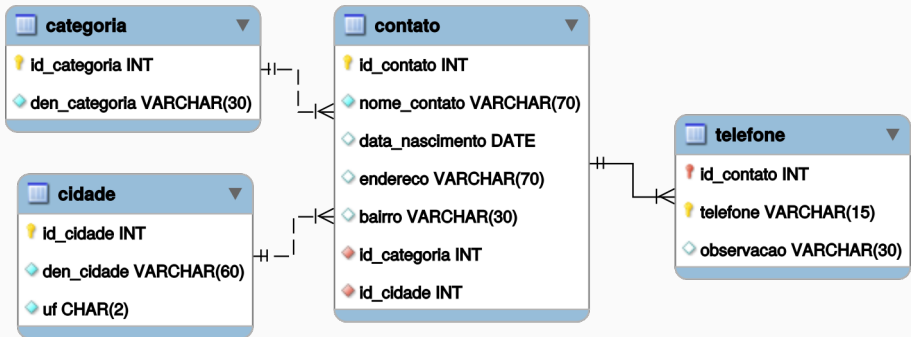
**DataSet:** São componentes não visuais que manipulam uma determinada parte do banco de dados, ou seja, delimitam as tabelas e registros a serem manipulados;

**DataSource:** Componente não visual que interliga um componente *DataSet* com componentes *Data Controls*. Disponível na paleta *Data Access*;

**Data Controls:** São componentes visuais que permitem a exibição e edição dos dados. Os componentes padrões estão na paleta *Data Controls*.

# Banco de Dados

- No decorrer desta aula vamos usar o seguinte banco de dados:



# Criação do Banco de Dados no SQLite I

```
-- Tabela categoria
CREATE TABLE IF NOT EXISTS categoria (
    id_categoria INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    den_categoria VARCHAR(30) NOT NULL
);

-- Tabela cidade
CREATE TABLE IF NOT EXISTS cidade (
    id_cidade INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    den_cidade VARCHAR(60) NOT NULL,
    uf CHAR(2) NOT NULL
);
```

## Criação do Banco de Dados no SQLite II

```
-- Tabela contato
CREATE TABLE IF NOT EXISTS contato (
    id_contato INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    nome_contato VARCHAR(70) NOT NULL,
    data_nascimento DATE,
    endereco VARCHAR(70),
    bairro VARCHAR(30),
    id_categoria INTEGER NOT NULL,
    id_cidade INTEGER NOT NULL,
    CONSTRAINT fk_contato_categoria
        FOREIGN KEY (id_categoria) REFERENCES categoria (id_categoria),
    CONSTRAINT fk_contato_cidade
        FOREIGN KEY (id_cidade) REFERENCES cidade (id_cidade)
);
```

# Criação do Banco de Dados no SQLite III

```
-- Tabela telefone
CREATE TABLE IF NOT EXISTS telefone (
  id_contato INTEGER NOT NULL,
  telefone VARCHAR(15) NOT NULL,
  observacao VARCHAR(30),
  PRIMARY KEY (telefone, id_contato),
  CONSTRAINT fk_telefone_contato
    FOREIGN KEY (id_contato) REFERENCES contato (id_contato)
);
```

```
-- Inserção de Dados
INSERT INTO categoria(den_categoria) VALUES
('Pessoal'),
('Profissional');
```

# Criação do Banco de Dados no SQLite IV

```
INSERT INTO cidade(den_cidade, uf) VALUES
('Bambuí', 'MG'),
('Medeiros', 'MG'),
('Tapiraí', 'MG'),
('Córrego Danta', 'MG');
```

```
INSERT INTO contato(nome_contato, data_nascimento, endereco, bairro,
↳ id_categoria, id_cidade) VALUES
('José', '1990-01-02', NULL, NULL, 1, 1),
('Maria', '1992-03-02', 'Rua dos Bobos, 0', NULL, 2, 2),
('João', '1989-09-25', NULL, NULL, 1, 3);
```

```
INSERT INTO telefone(id_contato, telefone, observacao) VALUES
(1, '1111-1111', 'Residencial'),
(1, '2222-1111', 'Celular'),
(2, '3333-3333', NULL),
(2, '4444-4444', NULL),
(2, '5555-5555', NULL);
```



# Módulo de Dados

- Quando trabalhamos com bancos de dados precisamos de unidades de código específicas para alguns componentes de banco de dados;
- Os módulos de dados ou **Data Modules** são **units** para a inserção de componentes não visuais, mais precisamente para acomodar os componentes *Connection*, *DataSets* e *DataSources*;
- A utilização de módulos de dados é fundamental para organizar melhor os componentes de banco de dados. Podemos enxergar um *Data Module* como uma espécie de “repositório” onde os componentes *Connection*, *DataSets* e *DataSources* são inseridos e referenciados pelos formulários;

## Observação

Para que os formulários possam ter acesso aos componentes do *Data Module* devemos incluir seu arquivo na declaração **uses** do formulário. Por exemplo, se o arquivo do *Data Module* chama-se *dmprincipal* então os formulários devem conter *uses dmprincipal* abaixo da declaração **implementation**.

## O Pacote de Componentes Zeos

- O pacote de componentes de banco de dados Zeos está entre os pacotes mais populares e estáveis desenvolvidos por terceiros;
- Além do Lazarus este pacote está disponível para diversos outros IDE;
- Os principais componentes do pacote são:
  - ZConnection:** Conexão com banco de dados;
  - ZReadOnlyQuery:** Execução de instruções SQL de consulta;
  - ZQuery:** Execução de instruções SQL;
  - ZTable:** Acesso direto a tabelas;
  - ZUpdateSQL:** Modificação de dados;
  - ZSequence:** Permite trabalhar com campos auto-incremento.

## O Componente ZConnection

- O componente *ZConnection* é responsável tanto pela conexão quanto pelo controle de transações com o banco de dados;
- Outra característica interessante é que o *ZConnection* é capaz de se conectar com diversos SGBD diferentes;
- As principais propriedades deste componente são:
  - HostName:** Nome de rede ou IP do servidor de banco de dados;
  - Username:** Usuário para conexão;
  - Password:** Senha para conexão;
  - Database:** Nome do banco de dados a ser conectado;
  - Connected:** Indica se o componente esta conectado (*True*) ou não (*False*) com o banco de dados;
  - DesignConnection:** Permite conectar apenas em tempo de edição;
  - Protocol:** Define o tipo de banco de dados a se conectar.

## O Componente ZConnection

- Os principais métodos do componente ZConnection são:
  - Connect():** Conecta-se ao banco de dados;
  - Disconnect():** Finaliza a conexão com o banco de dados;
  - Commit():** Efetiva a transação no servidor de banco de dados;
  - Rollback():** Cancela a transação no servidor de banco de dados.

## Exemplo com ZConnection

- Para entendermos melhor vamos criar um projeto no Lazarus, acrescentar um *Data Module* e inserir um componente *ZConnection* no mesmo;
- As principais propriedades do *Data Module* e do componente *ZConnection* são exibidas abaixo no formato *Lazarus Form (LFM)*:

```
object DataPrincipal: TDataPrincipal
  object ZConPrincipal: TZConnection
    Protocol = 'sqlite-3'
    Database = '...agenda.db'
    Connected = True
  end
end
```

### Observação

No Windows é preciso adicionar o caminho das bibliotecas DLL de conexão com o banco de dados na variável de ambiente PATH (Painel de Controle > Sistema e Segurança > Sistema > Configurações Avançadas do Sistema > Variáveis de Ambiente)

# Os Componentes ZQuery e ZUpdateSQL

- Quando trabalhamos com SGBD é interessante usar os componentes *ZQuery* e *ZUpdateSQL* para reduzir o tráfego de dados pela rede;
- O componente *ZQuery* é responsável por recuperar os dados através de uma consulta SQL e o componente *ZUpdateSQL* cuida das alterações realizadas sobre estes dados;
- As principais propriedades do *ZQuery* são:
  - Connection:** Componente de conexão com o banco de dados;
  - SQL:** Consulta SQL;
  - Active:** Ativa / Desativa os dados do componente;
  - Sequence:** Componente de sequencia;
  - SequenceField:** Campo auto-incremento a ser controlado pelo componente de sequencia;
  - UpdateObject:** Componente *ZUpdateSQL* com as instruções de modificação dos dados;
- As principais propriedades do *ZUpdateSQL* são:
  - DeleteSQL:** Instrução SQL de remoção;
  - InsertSQL:** Instrução SQL de inserção;
  - ModifySQL:** Instrução SQL de alteração.

## Exemplo com ZQuery e ZUpdateSQL I

- No exemplo anterior realizamos apenas a conexão com o banco de dados, agora vamos acrescentar componentes *DataSets* no *Data Module* para delimitarmos uma parte do banco de dados a ser trabalhada;
- Neste exemplo vamos utilizar os componentes *ZQuery* e *ZUpdateSQL* com as seguintes propriedades<sup>2</sup>:

```
object ZQueryCidade: TZQuery
  Connection = ZConPrincipal
  UpdateObject = ZUpdateCidade
  SQL.Strings = (
    'SELECT * FROM cidade'
  )
  Active = True
end
```

<sup>2</sup>As propriedades *DeleteSQL*, *InsertSQL* e *ModifySQL* do *ZUpdateSQL* podem ser editadas com mais facilidade pelo menu popup *UpdateSQL Editor...*

## Exemplo com ZQuery e ZUpdateSQL II

```
object ZUpdateCidade: TZUpdateSQL
  DeleteSQL.Strings = (
    'DELETE FROM cidade'
    'WHERE cidade.id_cidade = :OLD_id_cidade'
  )
  InsertSQL.Strings = (
    'INSERT INTO cidade(id_cidade, den_cidade, uf)'
    'VALUES (:id_cidade, :den_cidade, :uf)'
  )
  ModifySQL.Strings = (
    'UPDATE cidade SET'
    '  id_cidade = :id_cidade,'
    '  den_cidade = :den_cidade,'
    '  uf = :uf'
    'WHERE cidade.id_cidade = :OLD_id_cidade'
  )
end
```



## Os Componentes ZTable

- O componente *ZTable* deve ser utilizado apenas para manipular tabelas com poucos registros, uma vez que todos os registros da tabela são transferidos entre o servidor de banco de dados e o cliente. Suas principais propriedades são:

**Connection:** Componente de conexão com o banco de dados;

**Active:** Ativa / Desativa os dados do componente;

**Sequence:** Componente de sequencia;

**SequenceField:** Campo auto-incremento a ser controlado pelo componente de sequencia;

**TableName:** Tabela do banco de dados a ser manipulada.

## Exemplo com ZTable

- Vamos incluir um *ZTable* no *Data Module* e configurá-lo para trabalhar com a tabela *categoria* do nosso banco de dados;

```
object ZTableCategoria: TZTable
  Connection = ZConPrincipal
  TableName = 'categoria'
  Active = True
end
```

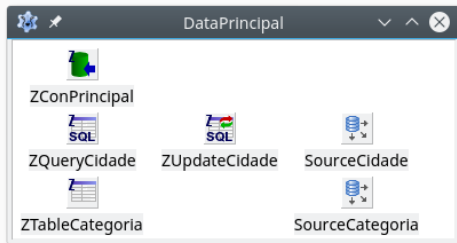
# O DataSource

- Como foi falado anteriormente o componente *DataSource* faz a ligação entre um *DataSet* e os *Data Controls*. Suas principais propriedades são:
  - DataSet:** Componente *DataSet* com os dados desejados, por exemplo um *ZQuery*;
  - AutoEdit:** Faz com que o *DataSet* entre em estado de edição automaticamente. Tal estado ocorre quando desejamos alterar os dados do *DataSet*, é conveniente setar para **False** esta propriedade.
- Continuando nosso exemplo vamos adicionar *DataSources* para as tabelas cidade e categoria em nosso *Data Module*:

```
object SourceCidade: TDataSource
    AutoEdit = False
    DataSet = ZQueryCidade
end
object SourceCategoria: TDataSource
    DataSet = ZTableCategoria
end
```

# Estado Atual do Data Module

- Neste momento o *Data Module* do nosso projeto encontra-se como mostrado na figura abaixo:



- A conexão com o banco de dados foi estabelecida e criamos *DataSets*, porém ainda não visualizamos ou alteramos qualquer informação do banco de dados;
- Agora vamos trabalhar com os controles de dados que devem ser inseridos nos formulários do projeto para que possamos realmente manipular os dados.

- O *DBEdit* é semelhante ao componente *Edit*, porém está relacionado com um campo de um *DataSet*. Suas propriedades primordiais são:
  - DataSource:** Componente de distribuição de dados;
  - DataField:** Campo vinculado ao *DBEdit*;
  - MaxLength:** Tamanho máximo;
  - ReadOnly:** Somente leitura;
- O *DBNavigator* é uma barra com botões de navegação e manipulação dos dados. Suas propriedades mais interessantes são:
  - DataSource:** Componente de distribuição de dados;
  - Hints:** Dicas para cada botão;
  - VisibleButtons:** Botões visíveis;

# Exemplo com DBEdit e DBNavigator



- Neste exemplo vamos criar um formulário que realiza cadastros na tabela *cidade* do nosso banco de dados utilizando os componentes *DBEdit* e *DBNavigator*;
- Lembrando que é necessário acrescentar a *unit dprincipal* ao **FormCadCidade**:

```
...  
implementation  
  
uses dprincipal;  
...
```

- As principais propriedades do formulário e dos componentes são exibidas a na próxima página seguindo o formato *Lazarus Form (LFM)*;

# Exemplo com DBEdit e DBNavigator (Propriedades)

```
object FormCadCidade: TFormCadCidade
  Caption = 'Cadastro de Cidades'
  object Label1: TLabel
    Caption = '&ID'
    FocusControl = DBEditIdCidade
  end
  object Label2: TLabel
    Caption = '&Cidade'
    FocusControl = DBEditDenCidade
  end
  object Label3: TLabel
    Caption = '&Estado'
    FocusControl = DBEditUF
  end
end
```

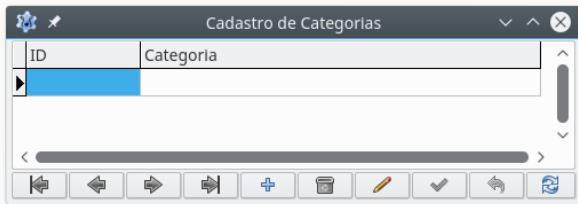
```
object DBNavCidade: TDBNavigator
  Align = alBottom
  DataSource = DataPrincipal.SourceCidade
end
object DBEditIdCidade: TDBEdit
  DataSource = DataPrincipal.SourceCidade
  DataField = 'id_cidade'
  ReadOnly = True
end
object DBEditDenCidade: TDBEdit
  DataSource = DataPrincipal.SourceCidade
  DataField = 'den_cidade'
end
object DBEditUF: TDBEdit
  DataSource = DataPrincipal.SourceCidade
  DataField = 'uf'
end
end
```

## O Componente DBGrid

- O componente *DBGrid* é muito semelhante ao componente *StringGrid*, mas com capacidades de manipulação de dados;
- Suas principais propriedades são:
  - DataSource:** Componente de distribuição de dados;
  - Columns:** Semelhante a propriedade *Columns* do *StringGrid*, mas podemos associar cada coluna a um campo do conjunto de dados.



## Exemplo com DBGrid



- Neste exemplo vamos criar um formulário que realiza cadastros na tabela *categoria* do nosso banco de dados utilizando o componente *DBGrid*;
- Como o nosso programa já possui um formulário de cadastros será necessário incluir um formulário principal com botões para chamar os demais formulários;
- Novamente será necessário acrescentar a *unit dprincipal*;
- As principais propriedades dos componentes e do novo formulário são exibidas na próxima página seguindo o formato *Lazarus Form (LFM)*;

## Exemplo com DBGrid (Propriedades)

```
object FormCadCategoria: TFormCadCategoria
  Caption = 'Cadastro de Categorias'
  Position = poScreenCenter
  object DBNavCategoria: TDBNavigator
    Align = alBottom
    DataSource = DataPrincipal.SourceCategoria
  end
  object DBGridCategoria: TDBGrid
    Align = alClient
    DataSource = DataPrincipal.SourceCategoria
    Columns = <
      item
        ReadOnly = True
        Title.Caption = 'ID'
        FieldName = 'id_categoria'
      end
      item
        Title.Caption = 'Denominação'
        FieldName = 'den_categoria'
      end>
  end
end
```

# Criando um Formulário Principal



- Este formulário de ser o primeiro a ser criado(*Project > Project Options... > Forms*)

# Propriedades do Formulário Principal I

```
object FormPrincipal: TFormPrincipal
  Caption = 'Agenda'
  Position = poScreenCenter
  object ButtonCadCidade: TButton
    Caption = 'C&idades'
    OnClick = ButtonCadCidadeClick
  end
  object ButtonCadCategoria: TButton
    Caption = 'C&ategorias'
    OnClick = ButtonCadCategoriaClick
  end
  object ButtonCadContato: TButton
    Caption = '&Contatos'
    OnClick = ButtonCadContatoClick
  end
  object ButtonConsultas: TButton
    Caption = 'C&onsultas'
    OnClick = ButtonConsultasClick
  end
end
```

# Código do Formulário Principal

```
uses fcadcidade, fcadcategoria; //, fcadcontato, fconsulta;

...

procedure TFormPrincipal.ButtonCadCidadeClick(Sender: TObject);
begin
    FormCadCidade.ShowModal();
end;

procedure TFormPrincipal.ButtonCadContatoClick(Sender: TObject);
begin
    //FormCadContato.ShowModal();
end;

procedure TFormPrincipal.ButtonConsultasClick(Sender: TObject);
begin
    //FormConsulta.ShowModal();
end;

procedure TFormPrincipal.ButtonCadCategoriaClick(Sender: TObject);
begin
    FormCadCategoria.ShowModal();
end;
```

# Os Componentes DBLookupListBox e DBLookupComboBox

- O Lazarus possui os componentes *DBListBox* e *DBComboBox* semelhantes aos componentes *ListBox* e *ComboBox*, a diferença é que os primeiros podem ser associados a campos de conjunto de dados;
- Além dos componentes *DBListBox* e *DBComboBox* o Lazarus fornece também os componentes *DBLookupListBox* e *DBLookupComboBox*. Estes dois últimos permitem a associação entre campos de duas tabelas e são utilizados para preenchimento de chaves estrangeiras. Suas principais propriedades são:
  - DataSource:** *DataSource* associado a tabela principal;
  - DataField:** Campo a ser modificado na tabela principal;
  - ListSource:** *DataSource* associado a tabela secundária (aquela que está vinculada a chave estrangeira);
  - ListField:** Campo a ser exibido na lista;
  - FieldKey:** Campo que será efetivamente selecionado para preencher o *DataField*;
- É importante salientar que os componentes *DBLookupListBox* e *DBLookupComboBox* não devem ser usados em arquiteturas cliente-servidor uma vez que a propriedade **ListSource** deve estar associada a um *DataSource* de um *DataSets* ativo. Tal *DataSet* pode armazenar em memória uma grande quantidade de dados prejudicando o desempenho do sistema.

## Exemplo com DBLookupComboBox (DataPrincipal) I

```
object ZQueryContato: TZQuery
    Connection = ZConPrincipal
    UpdateObject = ZUpdateContato
    SQL.Strings = (
        'SELECT * FROM contato'
    )
    Active = True
end
object ZUpdateContato: TZUpdateSQL
    DeleteSQL.Strings = (
        'DELETE FROM contato'
        'WHERE contato.id_contato = :OLD_id_contato'
    )
    InsertSQL.Strings = (
        'INSERT INTO contato'
        ' (id_contato, nome_contato, data_nascimento, endereco, bairro,'
        '   id_categoria, id_cidade)'
        'VALUES'
        ' (:id_contato, :nome_contato, :data_nascimento, :endereco, :bairro,'
        '   :id_categoria, :id_cidade)'
```

## Exemplo com DBLookupComboBox (DataPrincipal) II

```
)  
ModifySQL.Strings = (  
    'UPDATE contato SET'  
    '    id_contato = :id_contato,'  
    '    nome_contato = :nome_contato,'  
    '    data_nascimento = :data_nascimento,'  
    '    endereco = :endereco,'  
    '    bairro = :bairro,'  
    '    id_categoria = :id_categoria,'  
    '    id_cidade = :id_cidade'  
    'WHERE contato.id_contato = :OLD_id_contato'  
)  
end  
object SourceContato: TDataSource  
    AutoEdit = False  
    DataSet = ZQueryContato  
end
```



## Exemplo com DBLookupComboBox (FormCadContato)

The image shows a software window titled "Cadastro de Contatos". The window has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The main area of the window is a form with a dotted grid background. The form contains the following fields:

- ID:** A small text input field.
- Nome:** A wide text input field.
- Nascimento:** A dropdown menu currently showing "NULL".
- Categoria:** A text input field followed by a dropdown arrow.
- Endereço:** A wide text input field.
- Bairro:** A wide text input field.
- Cidade:** A text input field followed by a dropdown arrow.

At the bottom of the form is a toolbar with several icons: a left arrow, a right arrow, a double left arrow, a double right arrow, a plus sign, a trash can, a pencil, a checkmark, a refresh/circular arrow, and a save icon.

- O GroupBox foi usado para facilitar o cadastro dos telefones que será feito mais tarde.

# Propriedades do FormCadContato I

```
object GroupBoxContato: TGroupBox
  Align = alTop
  object Label1: TLabel
    Caption = '&ID'
    FocusControl = dbeditID
  end
  object Label2: TLabel
    Caption = '&Nome'
    FocusControl = dbeditNome
  end
  object Label7: TLabel
    Caption = 'N&ascimento'
    FocusControl = DBDateNascimento
  end
  object Label3: TLabel
    Caption = '&Categoria'
    FocusControl = dbeditCategoria
  end
  object Label7: TLabel
    Caption = 'En&dereço'
```

## Propriedades do FormCadContato II

```
    FocusControl = dbeditEndereco
end
object Label8: TLabel
    Caption = '&Bairro'
    FocusControl = dbeditBairro
end
object Label9: TLabel
    Caption = '&Cidade'
    FocusControl = dbeditCidade
end
object dbeditID: TDBEdit
    DataSource = DataPrincipal.SourceContato
    DataField = 'id_contato'
end
object dbeditNome: TDBEdit
    DataSource = DataPrincipal.SourceContato
    DataField = 'nome_contato'
end
object DBDateNascimento: TDBDateTimePicker
    DataSource = DataPrincipal.SourceContato
    DataField = 'data_nascimento'
```

# Propriedades do FormCadContato III

```
    DateSeparator = '-'
end
object dbcomboCategoria: TDBLookupComboBox
    DataSource = DataPrincipal.SourceContato
    DataField = 'id_categoria'
    ListSource = DataPrincipal.SourceCategoria
    KeyField = 'id_categoria'
    ListField = 'den_categoria'
    Style = csDropDownList
end
object dbeditCategoria: TDBEdit
    DataSource = DataPrincipal.SourceContato
    DataField = 'id_categoria'
end
object dbeditEndereco: TDBEdit
    DataSource = DataPrincipal.SourceContato
    DataField = 'endereco'
end
object dbeditBairro: TDBEdit
    DataSource = DataPrincipal.SourceContato
    DataField = 'bairro'
```

# Propriedades do FormCadContato IV

```
end
object dbeditCidade: TDBEdit
    DataSource = DataPrincipal.SourceContato
    DataField = 'id_cidade'
end
object dbcomboCidade: TDBLookupComboBox
    DataSource = DataPrincipal.SourceContato
    DataField = 'id_cidade'
    ListSource = DataPrincipal.SourceCidade
    KeyField = 'id_cidade'
    ListField = 'den_cidade'
    Style = csDropDownList
end
end
end
```

# Controle Mestre-Detalhe

- Em algumas situações temos que exibir e manipular dados de duas tabelas ao mesmo tempo em um formulário. Isto acontece quando temos uma tabela principal (mestre) e outra tabela secundária (detalhe) que depende da tabela principal, ou seja, temos um mestre-detahle;
- Em nosso banco de dados, o mestre-detahle acontece entre as tabelas *contato* (mestre) e *telefone* (detalhe). Isto porque um contato pode ter vários telefones e a exibição dos telefones depende do contato atual;
- A implementação de formulários mestre-detahle no Lazarus é feita utilizando dois *DataSets* e dois *DataSources*. Os componentes da tabela mestre são inseridos normalmente, já o *DataSet detalhe* deve ter as seguintes propriedades modificadas:

**MasterSource:** Componente *DataSource* da tabela principal;

**MasterFields:** Campos da tabela principal que filtram os dados na tabela secundária;

**LinkedFields:** Campos da tabela secundária que serão filtrados a partir dos valores dos *MasterFields*.

# Exemplo Mestre-Detalhe (Propriedades do DataPrincipal) I






```
object ZQueryTelefone: TZQuery
  Connection = ZConPrincipal
  UpdateObject = ZUpdateTelefone
  SQL.Strings = (
    'SELECT * FROM telefone'
  )
  MasterFields = 'id_contato'
  MasterSource = DataPrincipal.SourceContato
  LinkedFields = 'id_contato'
  Active = True
end
object SourceTelefone: TDataSource
  DataSet = ZQueryTelefone
end
object ZUpdateTelefone: TZUpdateSQL
  DeleteSQL.Strings = (
    'DELETE FROM telefone'
    'WHERE id_contato = :OLD_id_contato'
    'AND telefone = :OLD_telefone'
  )
end
```

## Exemplo Mestre-Detalhe (Propriedades do DataPrincipal) II

```
InsertSQL.Strings = (  
    'INSERT INTO telefone(telefone, id_contato, observacao)'  
    'VALUES(:telefone, :id_contato, :observacao)'  
)  
ModifySQL.Strings = (  
    'UPDATE telefone SET telefone = :telefone,'  
    '  id_contato = :id_contato,  observacao = :observacao'  
    'WHERE id_contato = :OLD_id_contato'  
    'AND telefone = :OLD_telefone'  
)  
RefreshSQL.Strings = (  
    'SELECT * FROM telefone'  
    'WHERE telefone = :telefone'  
    'AND id_contato = :id_contato'  
)  
end
```





# Exemplo Mestre-Detalhe (Cadastro de Contatos)

  Cadastro de Contatos   

ID:


Nome:











Nascimento:  

Categoria:  


Endereço:



Bairro:







Cidade:  

Telefones

Telefone	Observação
	

# Propriedades Cadastro de Contatos I

```
object GroupBoxTelefone: TGroupBox
    Align = alBottom
    Caption = '&Telefones'
    object DBNavTelefone: TDBNavigator
        Align = alRight
        DataSource = DataPrincipal.SourceTelefone
        Direction = nbdVertical
        VisibleButtons = [nbInsert, nbDelete, nbEdit, nbPost, nbCancel,
                        nbRefresh]
    end
    object DBGridTelefone: TDBGrid
        Align = alClient
        DataSource = DataPrincipal.SourceTelefone
        Columns = <
            item
                Visible = False
                FieldName = 'id_contato'
```

## Propriedades Cadastro de Contatos II

```
end
item
    Title.Caption = 'Telefone'
    Width = 150
    FieldName = 'telefone'
end
item
    Title.Caption = 'Observação'
    Width = 300
    FieldName = 'observacao'
end>
end
end
```

- Atualize o código do formulário principal para exibir o cadastro de contatos

## Consultas (DataPrincipal)

- O componente ideal para a realização de consultas é o *ZReadOnlyQuery*. Vamos acrescentar este componente e um *DataSource* ao módulo de dados do projeto anterior. As propriedades a serem modificadas são:

```
object ZRQueryConsulta: TZReadOnlyQuery
    Connection = ZConPrincipal
end
object SourceConsulta: TDataSource
    DataSet = ZRQueryConsulta
end
```

# Formulário de Consulta I

- Vamos criar também o seguinte formulário

The image shows a web application window titled "Consulta". Inside the window, there is a section titled "Opções de Consulta". This section contains three rows of input fields, each followed by a "Consultar" button:

- Row 1: "Nome:" followed by a text input field and a "Consultar" button.
- Row 2: "Nascimento:" followed by a date input field containing "2017-07-04" and a dropdown arrow, then "a.", then another date input field containing "2017-07-04" and a dropdown arrow, and finally a "Consultar" button.
- Row 3: "Cidade:" followed by a text input field and a "Consultar" button.

Below the "Opções de Consulta" section, there is a large empty rectangular area, likely intended for displaying search results. The window has a standard title bar with a gear icon, a pencil icon, and window control buttons (minimize, maximize, close).

## Formulário de Consulta II

```
object FormConsulta: TFormConsulta
  Caption = 'Consulta'
  Position = poScreenCenter
  object GroupBoxOpcoes: TGroupBox
    Align = alTop
    Caption = '&Opções de Consulta'
    object Label1: TLabel
      Caption = '&Nome'
      FocusControl = EditNome
    end
    object EditNome: TEdit
      Text = ''
    end
    object Label2: TLabel
      Caption = 'N&ascimento'
      FocusControl = DateInicio
    end
    object EditCidade: TEdit
      Text = ''
    end
  end
```

## Formulário de Consulta III

```
object Label3: TLabel
    Caption = '&Cidade'
    FocusControl = EditCidade
end
object DateInicio: TDateTimePicker
    DateDisplayOrder = ddoYMD
    DateSeparator = '-'
end
object Label4: TLabel
    Caption = 'a'
end
object DateFim: TDateTimePicker
    DateDisplayOrder = ddoYMD
    DateSeparator = '-'
end
object ButtonConsultarNome: TButton
    Caption = 'Consultar'
end
object ButtonConsultarNascimento: TButton
    Caption = 'Consultar'
end
```

## Formulário de Consulta IV

```
object ButtonConsultarCidade: TButton
    Caption = 'Consultar'
end
end
object DBGridConsulta: TDBGrid
    Align = alClient
    ReadOnly = True
    AutoFillColumns = True
end
end
```



## Consulta por Data de Nascimento

```
procedure TFormConsulta.ButtonConsultarNascimentoClick(Sender: TObject);
begin
    with DataPrincipal.ZRQueryConsulta do begin
        Close();           // Fecha uma possível consulta aberta
        SQL.Clear();       // Limpa se houver qualquer instrução SQL
        SQL.Add('SELECT co.id_contato AS ID,');
        SQL.Add('        co.nome_contato AS Nome,');
        SQL.Add('        co.data_nascimento AS Nascimento,');
        SQL.Add('        ci.den_cidade AS Cidade');
        SQL.Add('FROM contato AS co,');
        SQL.Add('        cidade AS ci');
        SQL.Add('WHERE co.id_cidade = ci.id_cidade');
        SQL.Add('AND co.data_nascimento BETWEEN :inicio AND :fim');
        ParamByName('inicio').AsDate:=DateInicio.Date;
        ParamByName('fim').AsDate:=DateFim.Date;
        Open();
    end;
end;
```

## Consulta por Nome

```
procedure TFormConsulta.ButtonConsultarNomeClick(Sender: TObject);
begin
    with DataPrincipal.ZRQueryConsulta do begin
        Close();
        SQL.Clear();
        SQL.Add('SELECT co.id_contato AS ID,');
        SQL.Add('        co.nome_contato AS Nome,');
        SQL.Add('        co.data_nascimento AS Nascimento,');
        SQL.Add('        ca.den_categoria AS Categoria');
        SQL.Add('FROM contato AS co,');
        SQL.Add('        categoria AS ca');
        SQL.Add('WHERE co.id_categoria = ca.id_categoria');
        SQL.Add('AND co.nome_contato LIKE ' + QuotedStr('%' + EditNome.Text + '%'));
        Open();
    end;
end;
```

- Precisamos usar função *QuotedStr* para colocar um string entre aspas, pois o delimitador de string é uma aspa.
- Não usamos o método *ParamByName*, pois o mesmo não reconhece parâmetros entre aspas;

## Código para o Botão ButtonConsultarCidade

```
procedure TFormConsulta.ButtonConsultarCidadeClick(Sender: TObject);
begin
    with DataPrincipal.ZRQueryConsulta do begin
        Close();
        SQL.Clear();
        SQL.Add('SELECT co.id_contato AS ID,');
        SQL.Add('        co.nome_contato AS Nome,');
        SQL.Add('        co.data_nascimento AS Nascimento,');
        SQL.Add('        ci.den_cidade AS Cidade');
        SQL.Add('FROM contato AS co,');
        SQL.Add('        cidade AS ci');
        SQL.Add('WHERE co.id_cidade = ci.id_cidade');
        SQL.Add('AND ci.den_cidade LIKE ' + QuotedStr('%' + EditCidade.Text +
        ↪ '%'));
        Open();
    end;
end;
```

- Localização automática do banco de dados
- Conexão automática com o banco de dados
- Código das consultas no módulo de dados

# Localização Automática do Banco de Dados I

- Primeiro é preciso modificar o componente ZConPrincipal

```
object ZConPrincipal: TZConnection
  Connected = False
  Database = ''
end
```

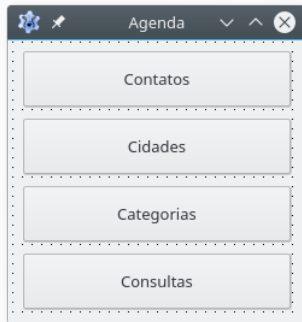
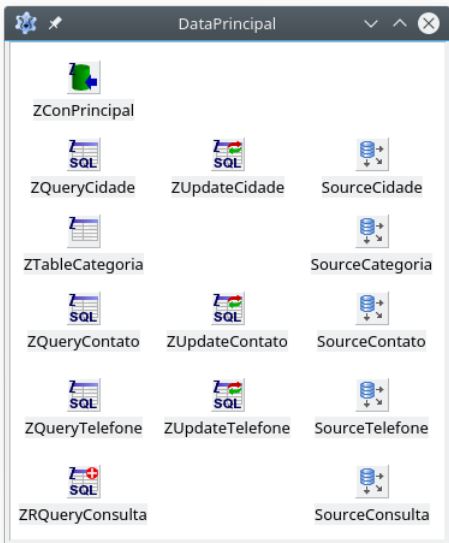
- Em seguida, devemos colocar o DataPrincipal para ser criado primeiro formulário a ser criado
- Por fim, acrescentamos o código para conexão no evento de criação do DataPrincipal

# Localização Automática do Banco de Dados II



## Código de Conexão

```
...
uses
    Classes, Forms...
...
procedure TDataPrincipal.DataModuleCreate(Sender: TObject);
var
    DBDir, DBName: String;
begin
    DBDir := ExtractFilePath(Application.ExeName);
    DBName := DBDir + DirectorySeparator + 'agenda.db';
    ZConPrincipal.Database := DBName;
    ZConPrincipal.Connect();
    ZTableCategoria.Open();
    ZQueryCidade.Open();
    ZQueryContato.Open();
    ZQueryTelefone.Open();
end;
```

# Exemplo Completo com DataModule













# Exemplo Completo com DataModule

  Cadastro de Categorias

ID	Categoria

< >



         

  Cadastro de Cidades

ID






Cidade

UF





# Exemplo Completo com DataModule

  Cadastro de Contatos   

ID:


Nome:











Nascimento:  

Categoria:  


Endereço:










Bairro:

Cidade:  

Telefones

Telefone	Observação
	

- Lazarus Database Tutorial. Lazarus Wiki. Disponível em [http://wiki.lazarus.freepascal.org/Lazarus\\_Database\\_Tutorial](http://wiki.lazarus.freepascal.org/Lazarus_Database_Tutorial);
- Zeos Quick Start Guide. Disponível em <http://zeos.firmos.at/kb.php?mode=article&k=6>.