FATEC SOROCABA – **Curso**: ADS **Disciplina**: LP2

Professora: Denilce Veloso – Aula 10

SUMÁRIO

INTRODUÇÃO A BANCO DE DADOS

Dado x Informação x Conhecimento

Banco de dados - Segundo Korth(2010), um banco de dados "é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico", ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados. Exemplos clássicos: lista de alunos, catálogo de filmes ou sistema de controle de RH de uma empresa.

Dado - conjunto de símbolos "arranjados" a fim de representar a informação fora da mente humana.

Elemento de Dado - subconjunto de símbolos que compõem um dado com significado específico, mas não representa a informação completa. Exemplo: *O número de alunos matriculados na disciplina informática no primeiro semestre é 40.*

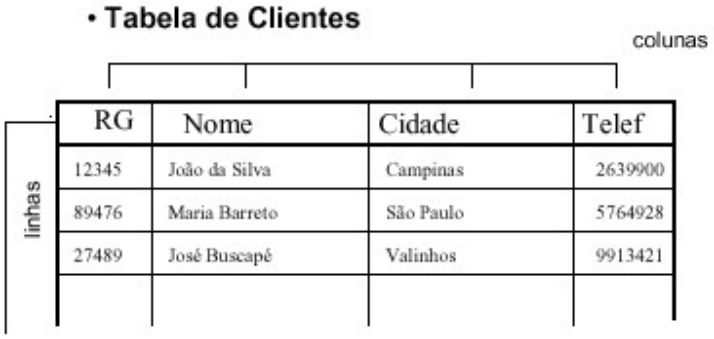
Quais são os elementos de dados?

- Disciplina : informática

FATEC SOROCABA – **Curso**: ADS **Disciplina**: LP2
Professora: Denilce Veloso – Aula 10

- Período : primeiro semestre
- Matriculados : 40

Um Banco de Dados criado para armazenar os dados fisicamente pode ser dividido em: Tabelas, Visões (*views*) e Índices.
Os dados são armazenados em linhas (registros) e colunas (campos) → TABELAS. Os dados de uma tabela normalmente descrevem um assunto tal como aluno, vendas etc.



RG	Nome	Cidade	Telef
12345	João da Silva	Campinas	2639900
89476	Maria Barreto	São Paulo	5764928
27489	José Buscapé	Valinhos	9913421

Um Banco de Dados representará sempre aspectos do Mundo Real. Assim sendo uma Base de Dados (ou Banco de Dados, ou ainda BD) é uma fonte de onde poderemos extrair uma vasta gama de informações derivadas, que possui um nível de interação com eventos como o Mundo Real que representa.
Por exemplo: Do Banco de Dados de uma biblioteca poderia extrair relatórios de aluno que estão em débito.

Sistema de Gerenciamento de Bancos de Dados (SGBD): é um software com recursos específicos para facilitar a manipulação das informações dos bancos de dados e o desenvolvimento de programas aplicativos. Ex. Oracle, DB2, SQLServer, MySQL etc.
Atenção: Paradox, Dbase, Access são Gerenciadores de Arquivos.

SGBD (Sistema Gerenciador Banco de Dados Relacional) x GA (Gerenciador de Arquivos)

ADO.NET

O que é ADO.Net (Definições e Benefícios)

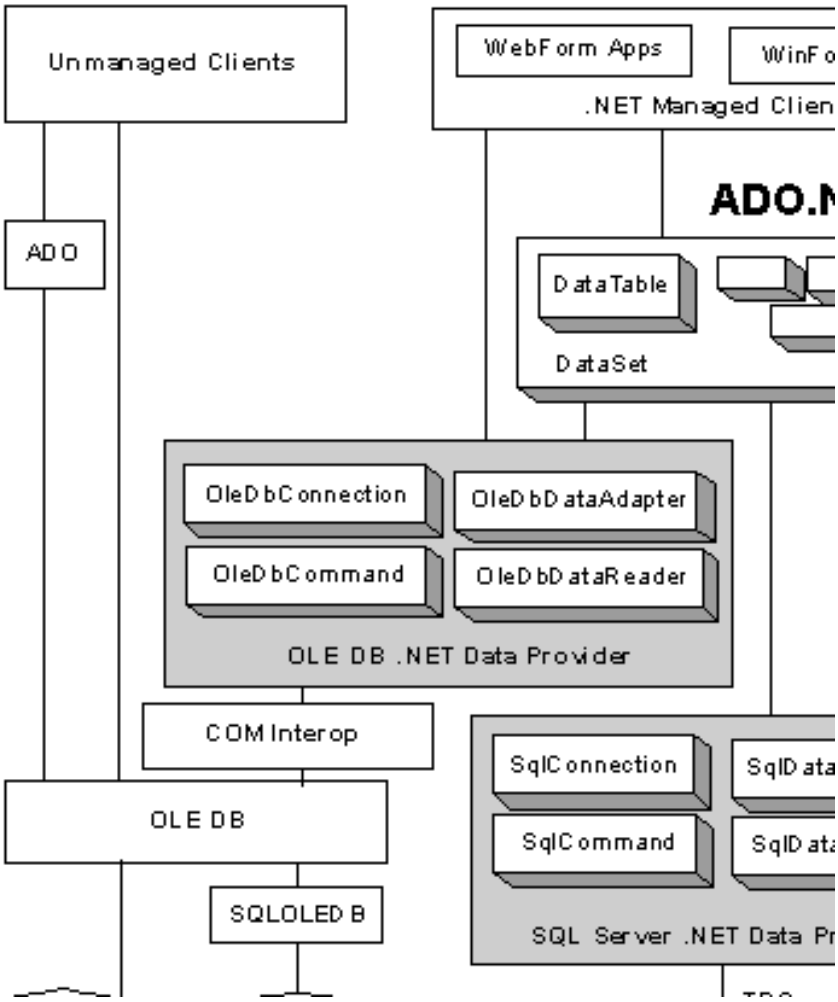
ADO.NET é a nova tecnologia da Microsoft para acessar dados. A Microsoft já vem a muito tempo na linha de frente do movimento de acesso universal a dados, iniciando com a Open Data Base Connectivity (ODBC).

Não há dúvidas sobre a importância de acesso a dados para aplicações modernas, mas o modo como você acessa os dados tem um impacto significativo na performance e escalabilidade da aplicação. Some-se a isto o fato de que as aplicações estão crescendo na sua distribuição, e a passagem de grandes quantidades de dados de um lugar para outro torna-se uma questão ainda mais crítica no design e criação de aplicações.

MODELO DE ACESSO A DADOS

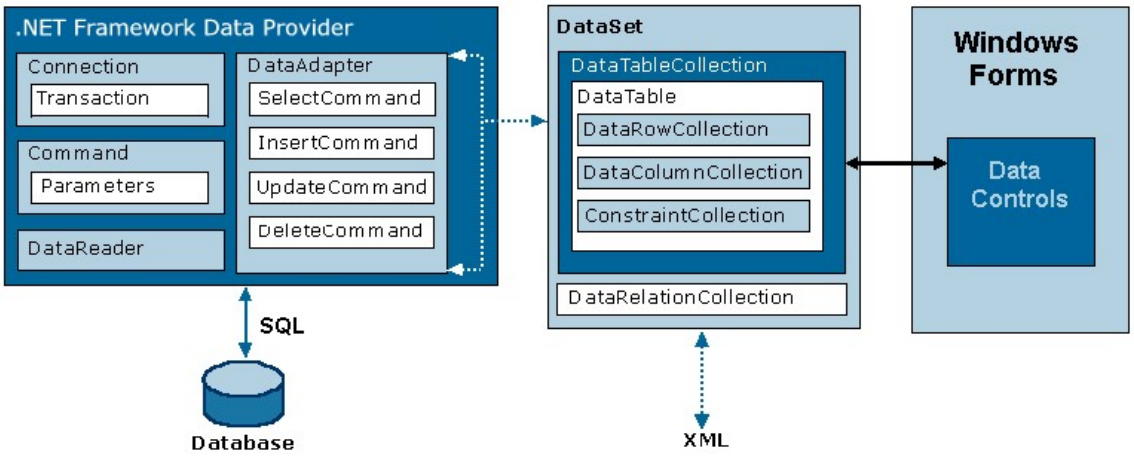
Veja abaixo como o ADO.NET se encaixa no modelo de acesso a dados:

Esquema de integração do ADO.NET

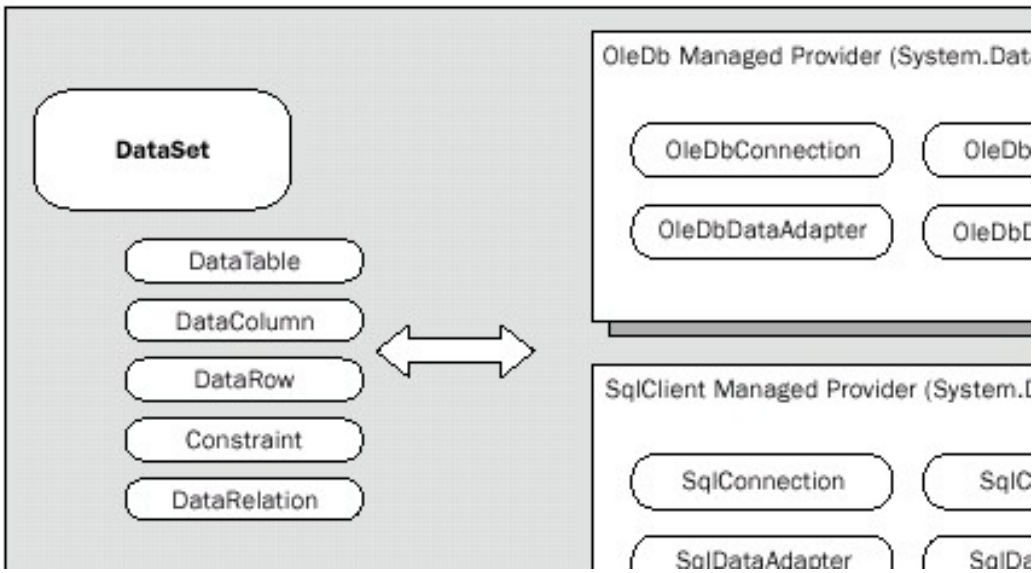


Detalhando mais:

Estrutura do ADO.NET



O **DataSet** proporciona o acesso a *tabelas, linhas , colunas , relacionamentos , constraints* e pode conter diversas tabelas e relacionamentos. A seguir temos um diagrama com uma outra visão da arquitetura **ADO.NET**:



- A direita temos os **Data Providers** os quais um conjunto de funcionalidade para acessar os dados armazenados.
- A direita temos o **DataSet** e seus componentes.

Uma razão a mais para a existência do ADO.NET é o fato de que a Microsoft queria separar o acesso a dados do trabalho com os dados. Isto permite que você separe os objetos que lhe ajudam a trabalhar com dados dos objetos que o ajudam a acessar dados. Também torna bastante simples trabalhar com dados desconectados por padrão, o que pode conduzir a grandes ganhos de escalabilidade.

Além disto, ADO. NET existe por causa da .NET Framework. No passado, ODBC e OLE DB, juntamente com seus modelos de objetos associados, eram independentes de qualquer ferramenta específica. Com a .NET Framework, a Microsoft introduziu o ADO.NET no conjunto padrão de serviços disponível para todas as linguagens que usam a .NET Framework. Agora, o acesso a dados torna-se um dos serviços básicos oferecidos pela infraestrutura de serviços, garantindo uma tecnologia de acesso a banco de dados consistente ao longo de todas as linguagens .NET.
Por ser criado dentro da .NET Framework, isto significa que o ADO.NET leva a vantagem de todos os serviços fornecidos pela Framework, tais como gerenciamento automático de memória, garbage collection, e assim por diante.

Provedores de Dados

ADO.NET disponibiliza classes para a manipulação dos dados. Portanto depende da classe System.Data que contém os seguintes namespaces:

- System.Data
- System.Data.SQL
- System.Data.SqlTypes
- System.Data.SqlClient

Para criar a conexão com o banco de dados o ADO.NET criou provedores de dados, dos quais de dividem em duas classes:

- SQL
- OleDB

Na disciplina LP2 vamos trabalhar com o SQL Server.

A Plataforma.Net criou a classe **SQL** com vários provedores para o acesso de qualquer plataforma, como: SQL Server, Oracle e Access.

Este provedor pertence à classe System.Data.SQLClient e seus principais objetos estão representados na tabela abaixo:

Objetos da classe System.Data.SQL	
Objeto	**Descrição**
SQLConnection	Define a abertura da conexão.
SQLCommand	Define a instrução SQL a ser executada.
SQLDataReader	Define somente para leitura um conjunto de dados.
SQLDataAdapter	Define a conexão a ser usada para preencher um DataSet, e representa um conjunto de comandos de dados.

DataReader - é mais adequada para o processamento de grande volume de registros pois você ganha tempo (começa a processar enquanto o banco de dados ainda está selecionando e entregando os registros) e demanda menos memória pois os registros não precisam ser carregados todos ao mesmo tempo antes de serem processados.

DataAdapter - é mais adequada para o processamento de volume menor de registros e para quando há a intenção de editar os registros e mandar as atualizações para o banco.

Cenário Desconectado/ DADOS DESLIGADOS:

DADOS DESLIGADOS

O ADO permitia o uso de dados desligados, mas estava essencialmente preparado para trabalhar com dados ligados. O ADO.Net está preparado de raiz para trabalhar com dados desligados. Esta é a principal mudança conceptual no ADO.Net, obrigando o programador a redefinir a maneira como trabalha com as bases de dados.
Mas o que são dados ligados e dados desligados? As tecnologias de acesso a dados ligados, permitem às estruturas de dados no cliente permanecerem ligadas à base de dados. As tecnologias de acesso com dados desligados, obrigam as estruturas de dados no cliente a ficarem desligadas da base de dados, ou seja, os dados no cliente são carregados da base de dados uma vez, são trabalhados no cliente sem usar os recursos do servidor e são gravados no servidor em bloco.

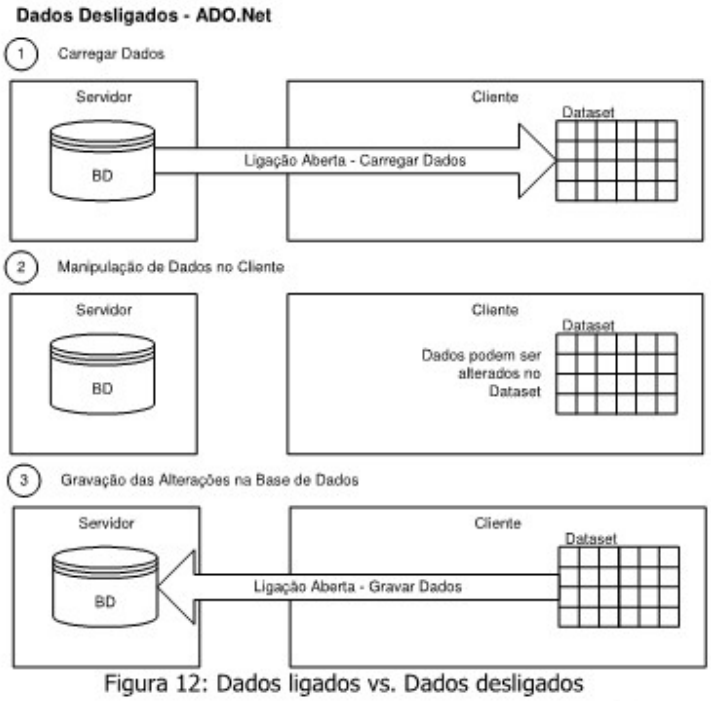


Figura 12: Dados ligados vs. Dados desligados

Num ambiente desconectado, um sub-conjunto de dados pode ser copiado e modificado independentemente e mais tarde as alterações podem ser introduzidas de novo na base de dados

- **Vantagens**
  - Pode-se trabalhar a qualquer altura e pode-se efectuar uma ligação à base de dados apenas quando necessário;
  - Outros utilizadores podem usar os recursos;
  - Este tipo de ambientes aumenta a escalabilidade e desempenho das aplicações;
- **Desvantagens**
  - Os dados nem sempre estão actualizados;

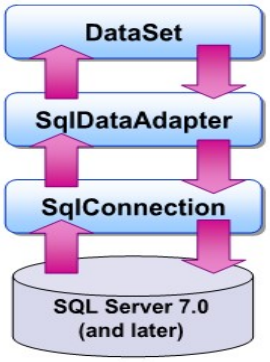


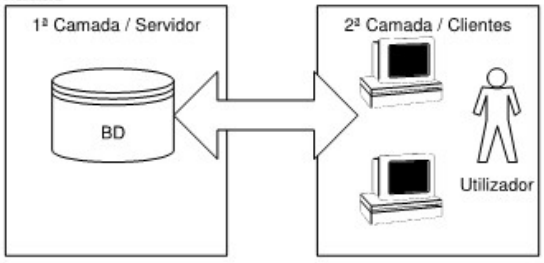
Figura 2 - Cenário Desconectado

Sistemas em várias camadas

Outro factor importante para justificar o uso de dados desligados é serem adequados a sistemas de informação em várias camadas. Os sistemas cliente/servidor são sistemas em 2 camadas: 1ª camada com o servidor de base de dados e a 2ª camada com os clientes.

O sistemas em várias camadas (também chamados de sistemas em 3 camadas) tem pelo menos 3 camadas com: 1ª camada com os dados; 2ª camada com objectos que fazem a ponte entre os dados e os clientes finais e a 3ª camada com os clientes propriamente ditos.

Sistema em 2 Camadas



Sistema em 3 (ou várias) Camadas

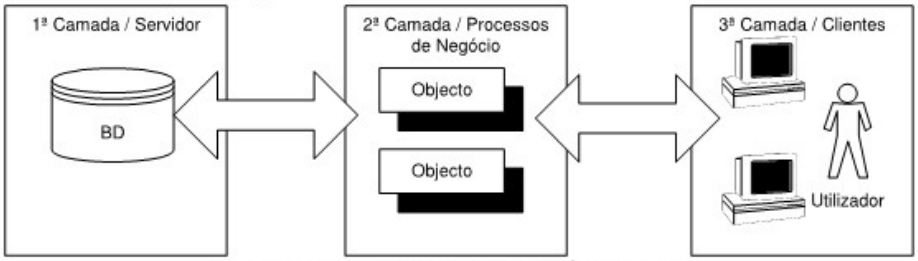


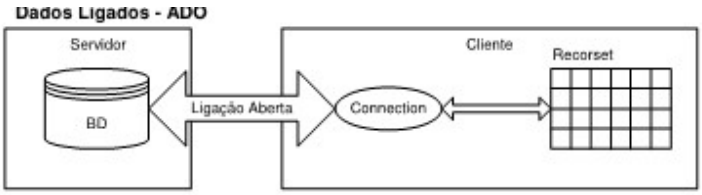
Figura 14: Sistemas de informação em várias camadas

Olhando para a figura anterior é fácil de perceber que os sistemas de informação em várias camadas não são adequados a acesso a dados ligados, pois os clientes finais não estão directamente ligados à base de dados, logo não é possível estabelecer ligações entre a base de dados e os clientes finais. Este modelo de sistema obriga ao acesso desligado aos dados.
Mas porquê é que este modelo (em várias camadas) é actualmente um modelo em voga? A razão é que este modelo é também mais adequado à internet e melhora a escalabilidade do sistema. A necessidade deste tipo de sistemas obriga à manipulação de dados desligados.
Um exemplo de um sistema de informação em 3 camadas são os sites em ASP com acesso a uma base de dados. Vejamos: a base de dados está na 1ª camada; o servidor Web está na 2ª camada (pode estar ou não na mesma máquina que o servidor de base de dados, mas em termos lógicos estão separados) e os browsers estão na 3ª camada e são os clientes deste tipo de sistemas.
Mas a verdade é que este tipo de sistemas não obriga à existência de uma tecnologia de acesso a dados desligados, no entanto, se analisarmos bem um sistema feito em ASP vemos que todos os conceitos de dados desligados são aplicados. A diferença é que é o programador a implementar todos os mecanismos necessários sem ajuda da própria tecnologia. Senão vejamos: os dados são carregados no Web Server que o programador converte em HTML. Quando os dados são carregados, o programador também tem que ter o cuidado de não abrir linhas em excesso (apenas deve carregar as linhas necessárias). Quando os dados são actualizados o que é que acontece? O programador recebe as alterações aos dados nos pedidos HTTP, convertendo essas alterações em comandos Insert, Update, Delete que vai aplicar na base de dados. Os dados no cliente e no servidor estão desligados, mas é o próprio programador que gere todo o processo de sincronização dos dados. As tecnologias de acesso a dados desligados pretendem aliviar o programador desse trabalho.
Os sistemas em várias camadas podem ter outras configurações, tais como: vários servidores Web (aumentando ainda mais a escalabilidade do sistema); outros tipos de servidores na camada dos processos (exclusivamente para processos); vários tipos de clientes (ex: Browser, aplicações windows, PDA's, telefones móveis, etc...)

Transporte de dados em XML.

O XML é actualmente o formato de eleição para representação de dados. O facto de ser um standard permite que várias aplicações possam interagir entre si usando o XML como ponte para transportar dados entre si.

Cenário Conectado:



Um cenário conectado é aquele no qual os utilizadores estão permanentemente ligados a bases de dados

- **Vantagens:**
  - É mais fácil exercer segurança ao nível do ambiente;
  - A concorrência é mais fácil de controlar;
  - Os dados estão mais actualizados que nos outros cenários;
- **Desvantagens**
  - É necessário haver uma ligação constante ao servidor;
  - Escalabilidade;

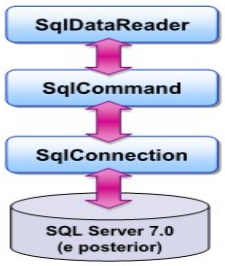


Figura 1- Cenário Conectado

FATEC SOROCABA – **Curso**: ADS **Disciplina**: LP2

Professora: Denilce Veloso – Aula 10

ACESSO AO BANCO DE DADOS VIA COMANDO NO PROGRAMA

(COMO FAREMOS NO NOSSO PROJETO)

ATENÇÃO NOS EXEMPLOS ABAIXO, SERÃO APRESENTADOS OS COMANDOS PARA FAZER UM CADASTRO DE CIDADES.

Antes de começar a testar os comandos declare no início da Class do form principal:

```
  public partial class frmPrincipal : Form
    {
        public static SqlConnection conexao;
```

Também não se esqueça de declarar os namespaces para poder utilizar os comandos de banco de dados do SQLServer:

```
using System.Data.Sql;
using System.Data.SqlClient;
```

Connection

i - Conectando com um Banco de Dados

No evento load do formulário vamos abrir a conexão com o banco de dados.O comando try.. catch Server para verificar se houve erro na abertura.

```
        private void Form1_Load(object sender, EventArgs e)
        {
            try
            {
// aqui a conexão vai depende da sua máquina da escola ou particular
                conexao = new SqlConnection("Data
Source=PROFDENILCE\\SQLEXPRESS;Initial Catalog=LP2;Integrated Security=True");
                conexao.Open();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Erro de banco de dados =/" + ex.Message);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Outros Erros =/" + ex.Message);
            }
        }
```

ii - Exemplo de Cadastro da Tabela Cidade

iii - Criação da Classe Cidade

Inicialmente cria-se a classe CIDADE com todos os atributos e métodos necessários.

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Data;

namespace PCidade
{
    class Cidade
    {

        private int idcidade;
        private string nomecidade;
        private string ufcidade;

        public int Idcidade
        {
            get
            {
                return idcidade;
            }
            set
            {
                idcidade = value;
            }
        }


        public string Nomecidade
        {
            get
            {
                return nomecidade;
            }
            set
            {
                nomecidade = value;
            }
        }



        public string Ufcidade
        {
            get
            {
                return ufcidade;
            }
            set
            {
                ufcidade = value;
            }
        }
```

```
public DataTable Listar()
        {
            SqlDataAdapter daCidade;

            DataTable dtCidade = new DataTable();

            try
            {
                daCidade = new SqlDataAdapter("SELECT * FROM TBcidade",
frmPrincipal.conexao);
                daCidade.Fill(dtCidade);
                daCidade.FillSchema(dtCidade, SchemaType.Source);
            }
            catch (Exception ex)
            {
                throw ex;
            }
            return dtCidade;
        }



        public int Salvar()
        {
            int nReg = 0;


            try
            {
                SqlCommand mycommand;

                mycommand = new SqlCommand("INSERT INTO TBCIDADE VALUES
(@nome_cidade,@uf_cidade)", frmPrincipal.conexao);

                mycommand.Parameters.Add(new SqlParameter("@nome_cidade",
SqlDbType.VarChar)); ;
                mycommand.Parameters.Add(new SqlParameter("@uf_cidade",
SqlDbType.VarChar));

                mycommand.Parameters["@nome_cidade"].Value = nomecidade;
                mycommand.Parameters["@uf_cidade"].Value = ufcidade;

                nReg = mycommand.ExecuteNonQuery();

            }
            catch (Exception ex)
            {
                throw ex;
            }

            return nReg;
        }



        public int Alterar()
        {
```

```
            int nReg = 0;


            try
            {
                SqlCommand mycommand;

                mycommand = new SqlCommand("UPDATE TBCIDADE SET nome_cidade =
@nome_cidade ,uf_cidade = @uf_cidade WHERE id_cidade = @id_cidade",
frmPrincipal.conexao);

                mycommand.Parameters.Add(new SqlParameter("@id_cidade",
SqlDbType.Int));
                mycommand.Parameters.Add(new SqlParameter("@nome_cidade",
SqlDbType.VarChar));
                mycommand.Parameters.Add(new SqlParameter("@uf_cidade",
SqlDbType.Char));

                mycommand.Parameters["@id_cidade"].Value = idcidade;
                mycommand.Parameters["@nome_cidade"].Value = nomecidade;
                mycommand.Parameters["@uf_cidade"].Value = ufcidade;

                nReg = mycommand.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                throw ex;
            }
            return nReg;
        }


        public int Excluir()
        {
            int nReg = 0;

            try
            {
                SqlCommand mycommand;

                mycommand = new SqlCommand("DELETE FROM TBCIDADE WHERE
id_cidade=@id_cidade", frmPrincipal.conexao);
                mycommand.Parameters.Add(new SqlParameter("@id_cidade",
SqlDbType.Int));
                mycommand.Parameters["@id_cidade"].Value =
Convert.ToInt16(idcidade);

                nReg = mycommand.ExecuteNonQuery();
            }

            catch (Exception ex)
            {
                throw ex;
            }

            return nReg;
        }
    }
```

```
}
```

iv - Criação do Form

v - Evento Load (carrega dados no grid)

' atenção: criar essas variáveis como GLOBAIS

```
namespace PCidade
{
    public partial class frmCidade : Form
    {
        private BindingSource bnCidade = new BindingSource();
        private bool bInclusao = false;
        private DataSet dsCidade = new DataSet();
```

```
private void frmCidade_Load(object sender, EventArgs e)
        {

            try
            {
                Cidade Cid = new Cidade();
                dsCidade.Tables.Add(Cid.Listar());
                bnCidade.DataSource = dsCidade.Tables["TBCidade"];
                dgvCidade.DataSource = bnCidade;
                bnvCidade.BindingSource = bnCidade;

                txtId.DataBindings.Add("TEXT", bnCidade, "id_cidade");
                txtNomeCidade.DataBindings.Add("TEXT", bnCidade, "nome_cidade");
                cbxEstado.DataBindings.Add("SelectedItem", bnCidade, "uf_cidade");
// AJUSTAR DROPDOWNSTYLE PARA DropDownList PARA NAO DEIXAR INCLUIR
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

vi - Evento Adicionar novo item

```
  private void btnNovoRegistro_Click(object sender, EventArgs e)
        {
            if (TabControl1.SelectedIndex == 0)
            {
                TabControl1.SelectTab(1);
            }
            bnCidade.AddNew();
            txtNomeCidade.Enabled = true;
            cbxEstado.Enabled = true;
            cbxEstado.SelectedIndex = 0;
            txtNomeCidade.Focus();
            btnSalvar.Enabled = true;
            btnAlterar.Enabled = false;
```

```
            btnNovoRegistro.Enabled = false;
            btnExcluir.Enabled = false;
            btnCancelar.Enabled = true;
            bInclusao = true; ;
        }
```

vii - Evento Excluir item

```
      private void btnExcluir_Click(object sender, EventArgs e)
        {
            if (TabControl1.SelectedIndex == 0)
            {
                TabControl1.SelectTab(1);
            }

            if (MessageBox.Show("Confirma exclusão?", "Yes or No",
MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2)
== DialogResult.Yes)
            {
                Cidade RegCid = new Cidade();

                RegCid.Idcidade = Convert.ToInt16(txtId.Text);
                RegCid.Nomecidade = txtNomeCidade.Text;
                RegCid.Ufcidade = cbxEstado.SelectedItem.ToString();

                if (RegCid.Excluir() > 0)
                {
                    MessageBox.Show("Cidade excluída com sucesso!");
                    Cidade R = new Cidade();
                    dsCidade.Tables.Clear();
                    dsCidade.Tables.Add(R.Listar());
                    bnCidade.DataSource = dsCidade.Tables["TBCidade"];
                }
                else
                {
                    MessageBox.Show("Erro ao excluir cidade!");
                }
            }
        }
```

viii - Evento Alterar item

```
       private void btnAlterar_Click(object sender, EventArgs e)
        {
            if (TabControl1.SelectedIndex == 0)
            {
                TabControl1.SelectTab(1);
            }

            txtNomeCidade.Enabled = true;
            cbxEstado.Enabled = true;
            txtNomeCidade.Focus();
            btnSalvar.Enabled = true;
            btnAlterar.Enabled = false;
            btnNovoRegistro.Enabled = false;
```

```
            btnExcluir.Enabled = false;
            btnCancelar.Enabled = true;
            bInclusao = false;
        }
```

ix - Evento Gravar

```
private void btnSalvar_Click(object sender, EventArgs e)
        {
            // validar os dados
            if (txtNomeCidade.Text == "")
            {
                MessageBox.Show("Cidade inválida!");
            }
           else
            {
                Cidade RegCid = new Cidade();

                RegCid.Idcidade = Convert.ToInt16(txtId.Text);
                RegCid.Nomecidade = txtNomeCidade.Text;
                RegCid.Ufcidade = cbxEstado.SelectedItem.ToString();

                if (bInclusao)
                {
                    if (RegCid.Salvar() > 0)
                    {
                        MessageBox.Show("Cidade adicionada com sucesso!");

                        btnSalvar.Enabled = false;
                        txtId.Enabled = false;
                        txtNomeCidade.Enabled = false;
                        cbxEstado.Enabled = false;
                        btnSalvar.Enabled = false;
                        btnAlterar.Enabled = true;
                        btnNovoRegistro.Enabled = true;
                        btnExcluir.Enabled = true;
                        btnCancelar.Enabled = false;

                        bInclusao = false;

                        // recarrega o grid
                        dsCidade.Tables.Clear();
                        dsCidade.Tables.Add(RegCid.Listar());
                        bnCidade.DataSource = dsCidade.Tables["TBCidade"];
                    }
                    else
                    {
                        MessageBox.Show("Erro ao gravar cidade!");
                    }
                }
                else
                {
                    if (RegCid.Alterar() > 0)
                    {
                        MessageBox.Show("Cidade alterada com sucesso!");
```

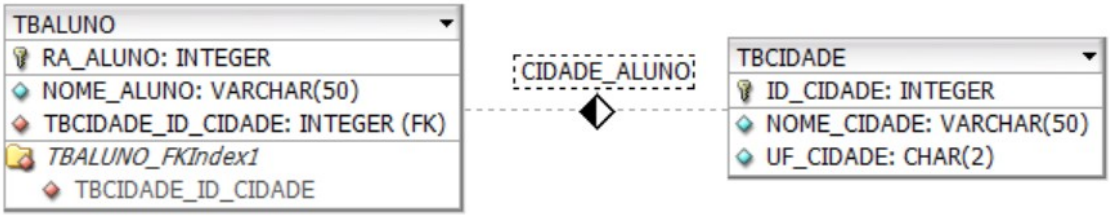
```
                    dsCidade.Tables.Clear();
                    dsCidade.Tables.Add(RegCid.Listar());
                    txtId.Enabled = false;
                    txtNomeCidade.Enabled = false;
                    cbxEstado.Enabled = false;
                    btnSalvar.Enabled = false;
                    btnAlterar.Enabled = true;
                    btnNovoRegistro.Enabled = true;
                    btnExcluir.Enabled = true;
                    btnCancelar.Enabled = false;
                }
                else
                {
                    MessageBox.Show("Erro ao gravar cidade!");
                }

            }
        }

    }
```

x - Evento Cancelar (incluir novo item ou alterar item)

```
private void btnCancelar_Click(object sender, EventArgs e)
    {

        bnCidade.CancelEdit();

        btnSalvar.Enabled = false;
        txtNomeCidade.Enabled = false;
        cbxEstado.Enabled = false;
        btnAlterar.Enabled = true;
        btnNovoRegistro.Enabled = true;
        btnExcluir.Enabled = true;
     }
```

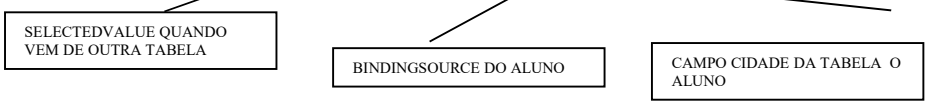
xi – Dicas de como Carregar Dados de Chave Estrangeira

'supondo que houvesse uma tabela de alunos que precisasse de um campo cidade
'carregar o combobox da CIDADE como nomes de todas as cidades no form do aluno ao invés de
'entrar com o ID da Cidade




```
            Cidade Cid = new Cidade();
            dsCidade.Tables.Add(Cid.Listar());

            cbxCidade.DataSource = dsCidade.Tables["TbCidade"];
```

```
                //CAMPO QUE SERÁ MOSTRADO PARA O USUÁRIO

                cbxCidade.DisplayMember = "nome_cidade";

                //CAMPO QUE É A CHAVE DA TABELA CIDADE E QUE LIGA COM A TABELA DE
ALUNO

                cbxCidade.ValueMember = "id_cidade";


                //No momento de linkar os componentes com o Binding Source linkar
também o combox da cidade

                cbxCidade.DataBindings.Add("SelectedValue", bnAluno,
"tbcidade_id_cidade"); // AJUSTAR DROPDOWNSTYLE PARA DropDownList PARA NAO DEIXAR
INCLUIR
```



Referências

Date, C.J. *An introduction to database systems*, Addison-Wesley, 8th edition, 2003. (Tradução: Introdução a Sistemas de Bancos de Dados, Editora Campus, 2004)

Heuser, C.A. *Projeto de Banco de Dados*, 5a. edição, Editora Sagra Luzatto, 2004.

Korth, H.F. and Silberschatz, A. and Sudarshan, S. *Database System Concepts*, 6th. edition, McGraw-Hill, 2010. (Tradução: Conceitos de Banco de Dados, Makron Books.)