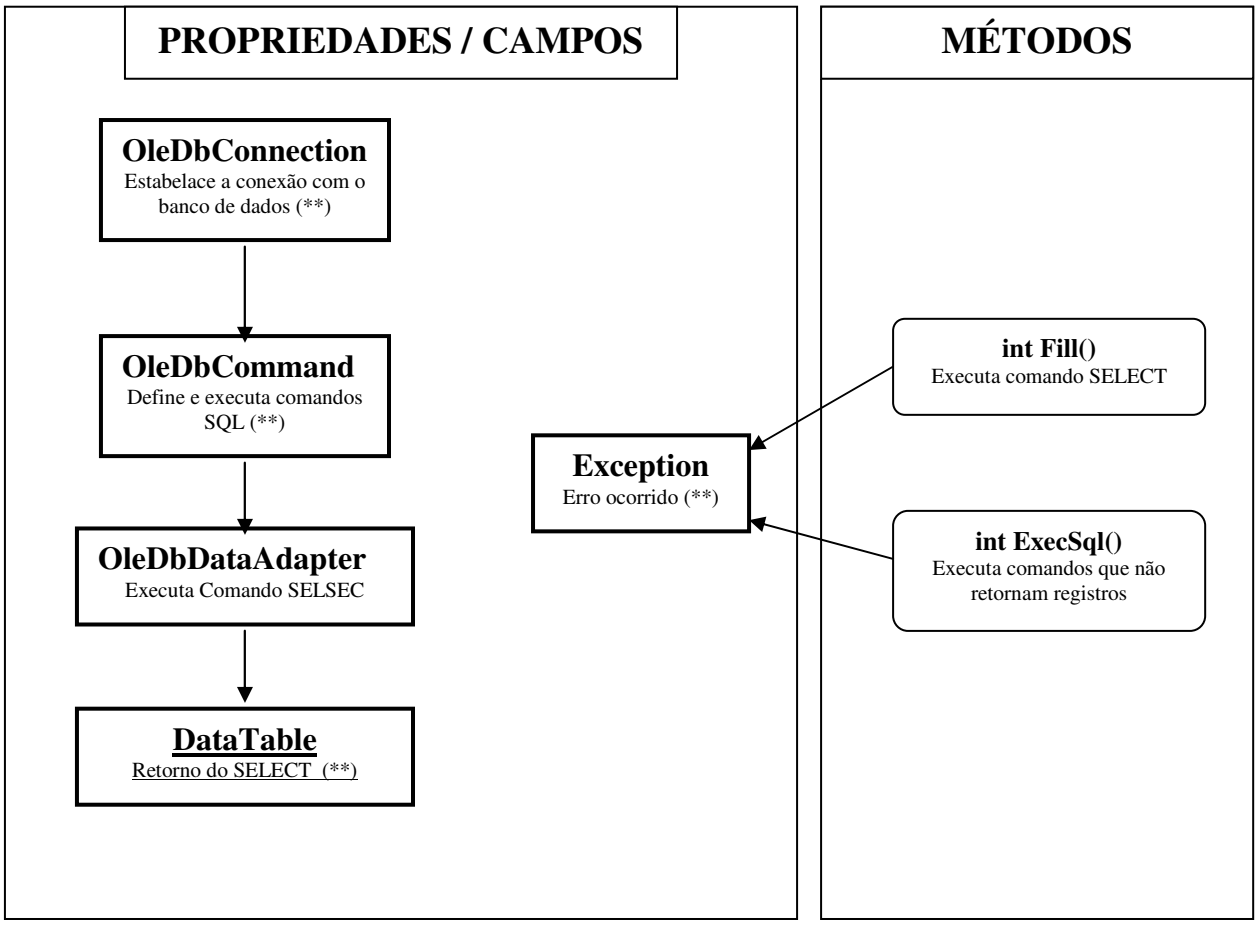
Classe OleDbQuery - Etapa 1

Finalidade: Encapsular as classes para execução de instruções SQL:



PROPRIEDADES:

OleDbConnection Connection: Estabelecer a conexão com o banco de dados e criar o objeto Command contendo o comando que será executado.

```
private OleDbConnection _connection;
   /// <summary>
   /// Define os parâmetros de conexão com o banco de dados
   /// </summary>
   public OleDbConnection Connection
   {
       get { return _connection; }
       set
       {
           _connection = value;
           // cria o objeto Command já associado ao Connection
           _command = _connection.CreateCommand();
       }
   }
```

OleDbCommand Command: Armazena e executa a instrução SQL.

```
private OleDbCommand _command;
   /// <summary>
   /// Instrução SQL que será executada
   /// </summary>
   public OleDbCommand Command
   {
       get { return _command; }
       set { _command = value; }
   }
```

DataTable Table: Armazena o resultado de instrução SELECT

```
  private DataTable _table;
   /// <summary>
   /// Armazena o retorno de instrução SELECT
   /// </summary>
   public DataTable Table
   {
       get { return _table; }
       set { _table = value; }
   }
```

Exception Error: Armazena o objeto Exception ocorrido na tentativa de executar o comando ou null se não ocorrer erro.

```
private Exception _error;
/// <summary>
/// Armazena o erro gerado pelo comando
/// </summary>
public Exception Error
{
    get { return _error; }
    set { _error = value; }
}
```

CONSTRUTORES:

```
/*
 * OleDbQuery qry = new OleDbQuery();
 * qry.Connection = Conexoes.GetConnection();
 */
public OleDbQuery()
{
    // cria o DataTable para armazenar SELECT
    _table = new DataTable();
}
/*
 * OleDbQuery qry = new OleDbQuery(Conexoes.GetConnection());
 */
public OleDbQuery(OleDbConnection conn)
{
    // executa o método SET da propriedade Connection que vai
    // criar o objeto Command
    Connection = conn;
    // cria o DataTable para armazenar SELECT
    _table = new DataTable();
}
```

MÉTODOS:

int Fill(): Executa instrução SELECT e coloca o resultado na propriedade Table. Retorna a quantidade de linhas do SELECT ou -1 caso dê erro.

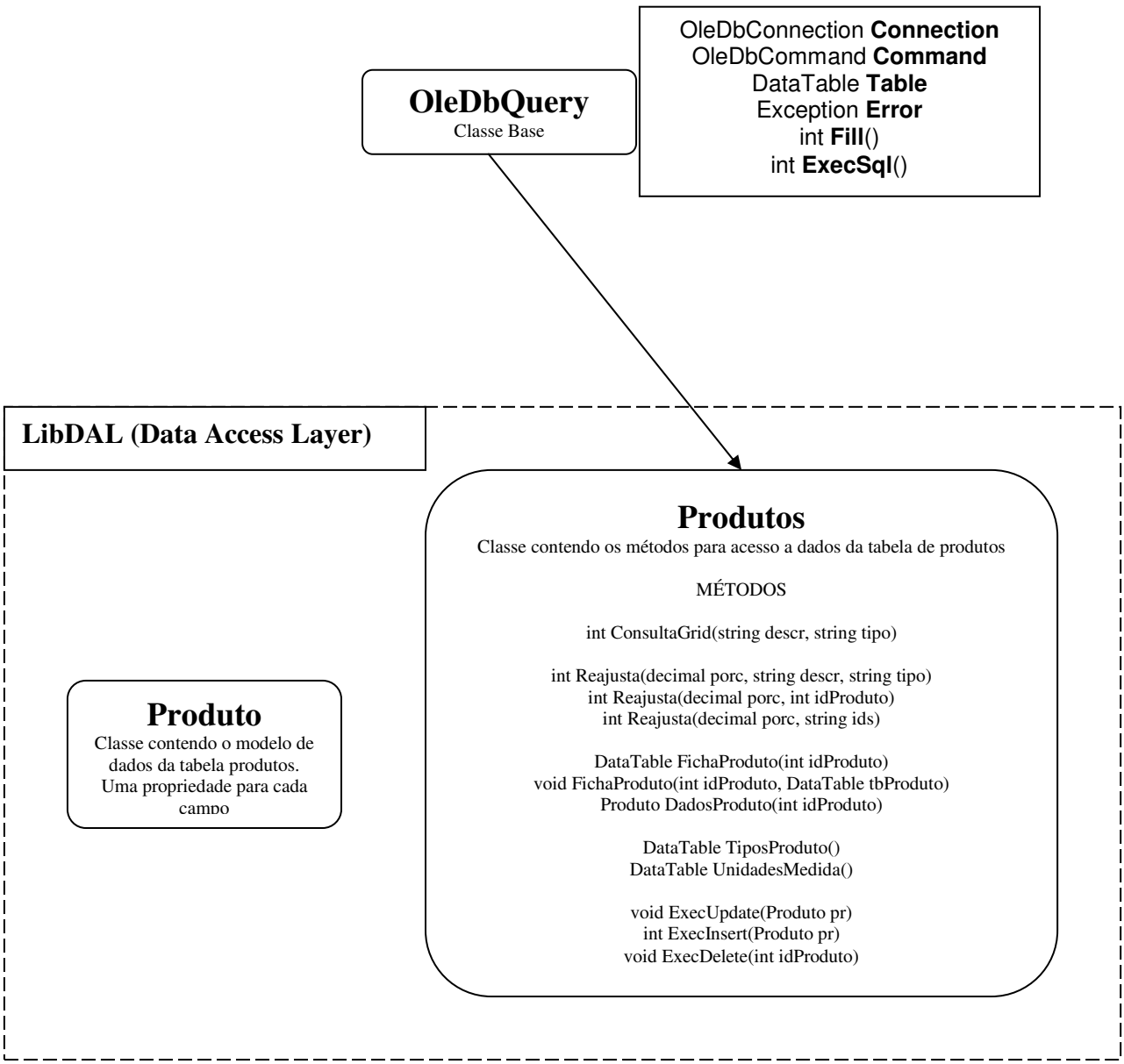
```
/// <summary>
/// Executa instrução SELECT
/// </summary>
/// <returns>Quantidade de linhas retornadas</returns>
public int Fill()
{
    // se não for comando SELECT, gerar erro
    if (!_command.CommandText.ToUpper().Trim().StartsWith("SELECT"))
    {
        throw new Exception("Comando não é SELECT");
    }
    try
    {
        OleDbDataAdapter da = new OleDbDataAdapter(_command);
        _table.Clear();
        da.Fill(_table);
        _error = null;
        return _table.Rows.Count;
    }
    catch (Exception ex)
    {
        _error = ex;
        return -1;
    }
}
```

int ExecSql(): Executa instruções que não retornam registros. Retorna a quantidade de registros afetados pelo comando.

```
/// <summary>
/// Executa instrução que não devolve registros
/// </summary>
/// <returns>Quantidade de linhas afetadas</returns>
public int ExecSql()
{
    // sinaliza se a conexão está aberta ou fechada
    bool connected = _connection.State == ConnectionState.Open;
    try
    {
        // se não estiver conectado, conectar
        if (!connected) _connection.Open();
        int linhas = _command.ExecuteNonQuery();
        _error = null;
        return linhas;
    }
    catch (Exception ex)
    {
        _error = ex;
        return -1;
    }
    finally
    {
        // se não estava conectado no início, desconectar
        if (!connected) _connection.Close();
    }
}
```

Classe Produtos - Etapa 2

Finalidade: Conter todas as instruções de acesso a dados relativas ao gerenciamento da tabela PRODUTOS.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.OleDb;
using LibQuery;
using System.Data;

/*
 * DAL: Data Access Layer (Camada de acesso a dados)
 *
 * Cada classe desta biblioteca possui os comandos SQL para manipular
 * uma tabela do banco de dados
 *
 */
namespace LibDAL
{
    // modelo de dados da tabela PRODUTOS (estrutura da tabela)
    public class Produto
    {
        private int _ID_PRODUTO;

        public int ID_PRODUTO
        {
            get { return _ID_PRODUTO; }
            set { _ID_PRODUTO = value; }
        }

        private string _COD_PRODUTO;

        public string COD_PRODUTO
        {
            get { return _COD_PRODUTO; }
            set { _COD_PRODUTO = value; }
        }

        private string _DESCRICAO;

        public string DESCRICAO
        {
            get { return _DESCRICAO; }
            set
            {
                if (value != "") _DESCRICAO = value;
                else throw new Exception("Descrição não pode ficar vazia");
            }
        }

        private short _COD_UNIDADE;

        public short COD_UNIDADE
        {
            get { return _COD_UNIDADE; }
            set { _COD_UNIDADE = value; }
        }

        private short _COD_TIPO;

        public short COD_TIPO
        {
            get { return _COD_TIPO; }
            set { _COD_TIPO = value; }
        }

        private decimal _PRECO_CUSTO;
```

```
    public decimal PRECO_CUSTO
    {
        get { return _PRECO_CUSTO; }
        set { _PRECO_CUSTO = value; }
    }

    private decimal _PRECO_VENDA;

    public decimal PRECO_VENDA
    {
        get { return _PRECO_VENDA; }
        set
        {
            if (value >= 0) _PRECO_VENDA = value;
            else throw new Exception("Preço não pode ser negativo");
        }
    }

    private int _QTD_ESTIMADA;

    public int QTD_ESTIMADA
    {
        get { return _QTD_ESTIMADA; }
        set { _QTD_ESTIMADA = value; }
    }

    private int _QTD_REAL;

    public int QTD_REAL
    {
        get { return _QTD_REAL; }
        set { _QTD_REAL = value; }
    }

    private int _QTD_MINIMA;

    public int QTD_MINIMA
    {
        get { return _QTD_MINIMA; }
        set { _QTD_MINIMA = value; }
    }

    private string _CLAS_FISC;

    public string CLAS_FISC
    {
        get { return _CLAS_FISC; }
        set { _CLAS_FISC = value; }
    }

    private int _IPI;

    public int IPI
    {
        get { return _IPI; }
        set { _IPI = value; }
    }

    private decimal _PESO_LIQ;

    public decimal PESO_LIQ
    {
        get { return _PESO_LIQ; }
        set { _PESO_LIQ = value; }
    }
}
```

```
/// <summary>
/// Comandos necessários para manipular a tabela PRODUTOS
/// </summary>
public class Produtos: OleDbQuery
{
    // construtor
    public Produtos(OleDbConnection conn)
    {
        // propriedade Connection herdada de OleDbQuery
        Connection = conn;
    }

    /// <summary>
    /// Executa o SELECT que alimenta o grid da tela de produtos
    /// </summary>
    /// <param name="descricao">
    /// Descrição do produto que queremos consultar
    /// </param>
    /// <param name="tipo">
    /// Categoria do produto que queremos consultar
    /// </param>
    /// <returns>Quantidade de linhas retornadas</returns>
    public int ConsultaGrid(string descricao, string tipo)
    {
        // propriedade Command herdada de OleDbquery
        Command.CommandText =
                 @"SELECT PR.ID_PRODUTO, PR.COD_PRODUTO, PR.DESCRICAO,
                         T.TIPO, U.UNIDADE, PR.PRECO_VENDA, PR.QTD_REAL,
                         PR.QTD_MINIMA
                   FROM PRODUTOS PR
                   JOIN TIPOPRODUTO T ON PR.COD_TIPO = T.COD_TIPO
                   JOIN UNIDADES U ON PR.COD_UNIDADE = U.COD_UNIDADE
                   WHERE DESCRICAO LIKE ? AND TIPO LIKE ?
                   ORDER BY DESCRICAO";
        // passar os parâmetros na mesma ordem em que aparecem dentro do
        // comando SQL
        Command.Parameters.Clear();
        Command.Parameters.AddWithValue("descricao", "%" + descricao + "%");
        Command.Parameters.AddWithValue("tipo", tipo + "%");
        // método Fill() herdado de OleDbQuery
        return Fill();
    }

    /// <summary>
    /// Reajusta os preços dos produtos filtrados
    /// </summary>
    /// <param name="fator">
    /// Fator de reajuste
    /// </param>
    /// <param name="descr">
    /// Descrição dos produtos diltrados
    /// </param>
    /// <param name="tipo">
    /// Categoria dos produtos filtrados
    /// </param>
    /// <returns>Quantidade de linhas afetadas</returns>
    public int Reajusta(decimal porc, string descr, string tipo)
    {
        Command.CommandText =
             @"UPDATE PRODUTOS SET PRECO_VENDA *= ?
               FROM PRODUTOS PR
               JOIN TIPOPRODUTO T ON PR.COD_TIPO = T.COD_TIPO
               WHERE DESCRICAO LIKE ? AND TIPO LIKE ?";
        Command.Parameters.Clear();
        Command.Parameters.AddWithValue("porc", 1 + porc/100 );
        Command.Parameters.AddWithValue("descricao", "%" + descr + "%");
        Command.Parameters.AddWithValue("tipo", tipo + "%");
```

```
            // método ExecSQL() herdado de OleDbQuery
            return ExecSql();
        }
        /// <summary>
        /// Reajusta os preços dos produtos cujos IDs estejam
        /// contidos na lista
        /// </summary>
        /// <param name="porc">Porcentagem de reajuste</param>
        /// <param name="ids">Lista de IDs de produtos</param>
        /// <returns>Quantidade de linhas afetadas</returns>
        public int Reajusta(decimal porc, string ids)
        {
            Command.CommandText = @"UPDATE PRODUTOS SET PRECO_VENDA *= ?
                                    WHERE ID_PRODUTO IN (" + ids + ")";
            Command.Parameters.Clear();
            Command.Parameters.AddWithValue("porc", 1 + porc / 100);

            return ExecSql();
        }
        /// <summary>
        /// Retorna com os dados de um produto
        /// </summary>
        /// <param name="id">ID do produto</param>
        /// <returns>
        /// Objeto da classe Produto contendo os campos do produto
        /// </returns>
        public Produto FichaProduto(int id)
        {
            OleDbCommand cmd = Connection.CreateCommand();
            cmd.CommandText = @"SELECT * FROM PRODUTOS
                                WHERE ID_PRODUTO = " + id;
            OleDbDataAdapter da = new OleDbDataAdapter(cmd);
            DataTable tb = new DataTable();
            int linhas = da.Fill(tb);

            if (linhas == 0) return null;
            else
                return new Produto
                {
                    ID_PRODUTO = (int)tb.Rows[0]["ID_PRODUTO"],
                    COD_PRODUTO = tb.Rows[0]["COD_PRODUTO"].ToString(),
                    DESCRICAO = tb.Rows[0]["DESCRICAO"].ToString(),
                    COD_TIPO = (short)tb.Rows[0]["COD_TIPO"],
                    COD_UNIDADE = (short)tb.Rows[0]["COD_UNIDADE"],
                    PRECO_CUSTO = (decimal)tb.Rows[0]["PRECO_CUSTO"],
                    PRECO_VENDA = (decimal)tb.Rows[0]["PRECO_VENDA"],
                    QTD_REAL = (int)tb.Rows[0]["QTD_REAL"],
                    QTD_MINIMA = (int)tb.Rows[0]["QTD_MINIMA"],
                    CLAS_FISC = tb.Rows[0]["CLAS_FISC"].ToString(),
                    IPI = (int)tb.Rows[0]["IPI"],
                    PESO_LIQ = (decimal)tb.Rows[0]["PESO_LIQ"],
                };
        }
        /// <summary>
        /// Devolve um DataTable contendo os tipos de produto
        /// </summary>
        /// <returns></returns>
        public DataTable TiposProduto()
        {
//            OleDbCommand cmd = Connection.CreateCommand();
//            cmd.CommandText = @"SELECT * FROM TIPOPRODUTO
//                                ORDER BY TIPO";
//            OleDbDataAdapter da = new OleDbDataAdapter(cmd);
//            DataTable tb = new DataTable();
//            da.Fill(tb);
//            return tb;
            OleDbQuery qry = new OleDbQuery(Connection);
```

```
    qry.Command.CommandText = @"SELECT * FROM TIPOPRODUTO
                                ORDER BY TIPO";
    qry.Fill();
    return qry.Table;
}

public DataTable UnidadesMedida()
{
    OleDbCommand cmd = Connection.CreateCommand();
    cmd.CommandText = @"SELECT * FROM UNIDADES
                        ORDER BY UNIDADE";
    OleDbDataAdapter da = new OleDbDataAdapter(cmd);
    DataTable tb = new DataTable();
    da.Fill(tb);
    return tb;
}

public void ExecUpdate(Produto pr)
{
    OleDbCommand cmd = Connection.CreateCommand();
    cmd.CommandText = "EXEC SP_PRODUTOS_UPDATE " +
        "?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?";
    cmd.Parameters.AddWithValue("id", pr.ID_PRODUTO);
    cmd.Parameters.AddWithValue("cod", pr.COD_PRODUTO);
    cmd.Parameters.AddWithValue("descricao", pr.DESCRICAO);
    cmd.Parameters.AddWithValue("cod_unid", pr.COD_UNIDADE);
    cmd.Parameters.AddWithValue("cod_tipo", pr.COD_TIPO);
    cmd.Parameters.AddWithValue("preco_custo", pr.PRECO_CUSTO);
    cmd.Parameters.AddWithValue("preco_venda", pr.PRECO_VENDA);
    cmd.Parameters.AddWithValue("qtd_est", pr.QTD_ESTIMADA);
    cmd.Parameters.AddWithValue("qtd_real", pr.QTD_REAL);
    cmd.Parameters.AddWithValue("qtd_min", pr.QTD_MINIMA);
    cmd.Parameters.AddWithValue("cf", pr.CLAS_FISC);
    cmd.Parameters.AddWithValue("ipi", pr.IPI);
    cmd.Parameters.AddWithValue("peso_liq", pr.PESO_LIQ);

    bool connected = Connection.State == ConnectionState.Open;
    try
    {
        if (!connected) Connection.Open();
        cmd.ExecuteNonQuery();
        Error = null;
    }
    catch (Exception ex)
    {
        Error = ex;
    }
    finally
    {
        if (!connected) Connection.Close();
    }
}

public int ExecInsert(Produto pr)
{
    OleDbCommand cmd = Connection.CreateCommand();
    cmd.CommandText = "EXEC SP_PRODUTOS_INSERT " +
        "?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?";
    //cmd.Parameters.AddWithValue("id", pr.ID_PRODUTO);
    cmd.Parameters.AddWithValue("cod", pr.COD_PRODUTO);
    cmd.Parameters.AddWithValue("descricao", pr.DESCRICAO);
    cmd.Parameters.AddWithValue("cod_unid", pr.COD_UNIDADE);
    cmd.Parameters.AddWithValue("cod_tipo", pr.COD_TIPO);
    cmd.Parameters.AddWithValue("preco_custo", pr.PRECO_CUSTO);
    cmd.Parameters.AddWithValue("preco_venda", pr.PRECO_VENDA);
    cmd.Parameters.AddWithValue("qtd_est", pr.QTD_ESTIMADA);
    cmd.Parameters.AddWithValue("qtd_real", pr.QTD_REAL);
```

```
        cmd.Parameters.AddWithValue("qtd_min", pr.QTD_MINIMA);
        cmd.Parameters.AddWithValue("cf", pr.CLAS_FISC);
        cmd.Parameters.AddWithValue("ipi", pr.IPI);
        cmd.Parameters.AddWithValue("peso_liq", pr.PESO_LIQ);

        bool connected = Connection.State == ConnectionState.Open;
        try
        {
            if (!connected) Connection.Open();

            int id = Convert.ToInt32(cmd.ExecuteScalar());
            Error = null;
            return id;
        }
        catch (Exception ex)
        {
            Error = ex;
            return -1;
        }
        finally
        {
            if (!connected) Connection.Close();
        }
    }

    public void ExecDelete(int id)
    {
        OleDbCommand cmd = Connection.CreateCommand();
        cmd.CommandText = "EXEC SP_PRODUTOS_DELETE " + id;

        bool connected = Connection.State == ConnectionState.Open;
        try
        {
            if (!connected) Connection.Open();
            cmd.ExecuteNonQuery();
            Error = null;
        }
        catch (Exception ex)
        {
            Error = ex;
        }
        finally
        {
            if (!connected) Connection.Close();
        }
    }
    //------------------------------------- Edição com CommandBuilder
    /// <summary>
    /// Carrega os dados de 1 produto em um DataTable
    /// </summary>
    /// <param name="idProduto">ID do produto que será pesquisado</param>
    /// <param name="tbProduto">DataTable que receberá os dados</param>
    public void FichaProduto(int idProduto, DataTable tbProduto)
    {
        OleDbCommand cmd = Connection.CreateCommand();
        cmd.CommandText = @"SELECT * FROM PRODUTOS
                            WHERE ID_PRODUTO = " + idProduto;
        OleDbDataAdapter da = new OleDbDataAdapter(cmd);
        da.Fill(tbProduto);
    }
    /// <summary>
    /// Grava no banco de dados as alterações feitas no DataTable
    /// </summary>
    /// <param name="tbProduto">
    /// DataTable com os dados que serão gravados
    /// </param>
    public void GravaAlteracao(DataTable tbProduto)
```

```
        {
            // DataAdapter reproduzindo a mesma estrutura de campos
            // de tbProduto
            OleDbDataAdapter da = new OleDbDataAdapter(
                "SELECT * FROM PRODUTOS WHERE ID_PRODUTO = 0",
                Connection);
            // gera no DataAdapter as propriedades DeleteCommand
            // InsertCommand e UpdateCommand
            new OleDbCommandBuilder(da);
            // lê o status de cada linha de tbProduto (só terá uma)
            // e aplica o comando necessário para atualizar a tabela
            try
            {
                da.Update(tbProduto);
            }
            catch (Exception ex)
            {
                Error = ex;
            }
        }
        /// <summary>
        /// Grava inclusão de registro e retorna com o ID_PRODUTO gerado
        /// </summary>
        /// <param name="tbProduto">
        /// DataTable contendo a linha que será inserida
        /// </param>
        /// <returns>ID do produto inserido</returns>
        public int GravaInclusao(DataTable tbProduto)
        {
            OleDbCommand cmd = Connection.CreateCommand();
            cmd.CommandText = "SELECT @@IDENTITY";

            OleDbDataAdapter da = new OleDbDataAdapter(
                "SELECT * FROM PRODUTOS WHERE ID_PRODUTO = 0",
                Connection);
            new OleDbCommandBuilder(da);

            Connection.Open();
            try
            {
                da.Update(tbProduto);
                return Convert.ToInt32(cmd.ExecuteScalar());
            }
            catch (Exception ex)
            {
                Error = ex;
                return -1;
            }
            finally
            {
                Connection.Close();
            }
        }
    }
}
```