Fundamentos de Bancos de Dados com C#

Michael Schmalz

Novatec

Editor: Rubens Prates
Tradução: Acauan Pereira Fernandes
Revisão técnica: Joel Saade
Revisão gramatical: Giacomo Leone Neto
Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-315-4

Histórico de impressões:

Julho/2012 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
E-mail: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Schmalz, Michael
Fundamentos de bancos de dados com C# / Michael Schmalz ; [tradução Acauan Pereira Fernandes]. — São Paulo : Novatec Editora ; Sebastopol : O'Reilly Media, 2012.

Título original: C# database basics.
ISBN 978-85-7522-315-4

1. C# (Linguagem de computador) I. Título.

12-07773 CDD-005.268

Índices para catálogo sistemático:

1. C# : Linguagem de computador : Processamento de dados 005.268

CAPÍTULO 1

Primeiros passos: formulário com um Datagrid

É hora de mergulharmos no C#. Se você estiver migrando do Visual Basic ou do Microsoft Access, iniciar significa percorrer uma quantidade de coisas que parecem familiares, mas trabalham de forma um pouco diferente.

Instalando o software

Se você ainda não tiver feito isso, pode entrar no website da Microsoft e baixar a versão Express do Visual Studio 2010 para C#. O site está atualmente em *http://www.microsoft.com/visualstudio/en-us*. Na parte inferior da página, você pode ir para Express Product Downloads ou baixar a versão de teste de 90 dias da versão completa. Links mudam o tempo todo, de modo que, se ele não estiver lá quando você procurar, uma pesquisa com um mecanismo de busca irá direcioná-lo ao lugar em que poderá baixá-la. Uma vez instalado e aberto, você verá uma tela semelhante à mostrada na figura 1.1.

Se você quiser a versão Express do SQL Server, ela está disponível na página Express Product Downloads também. Se você não tiver o Microsoft Access instalado no seu computador, precisará dele para trabalhar com os exemplos de dados. Os exemplos que usaremos são intercambiáveis entre as duas plataformas. A principal diferença é a string de conexão que você usará. Embora o SQL Server tenha muitos recursos adicionais, está além do escopo deste livro.

Agora que você tem os programas instalados, está pronto para criar um novo projeto. Para fazer isso, a partir da tela principal do Visual Studio, pode ir até File > New Project, ou simplesmente pressionar Ctrl+Shift+N para mostrar a caixa de diálogo de novo projeto que você vê na figura 1.2.

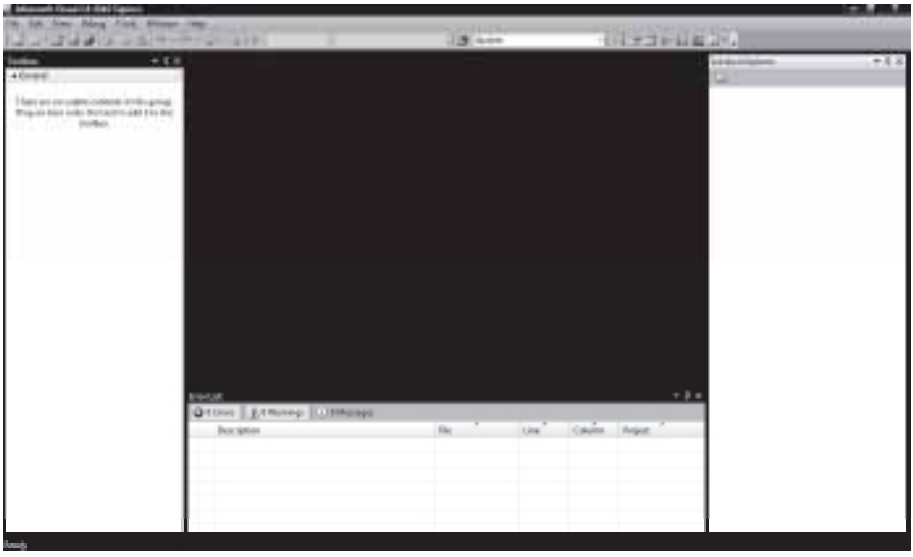

Figura 1.1 – Tela principal do Microsoft Visual C# 2010 Express.

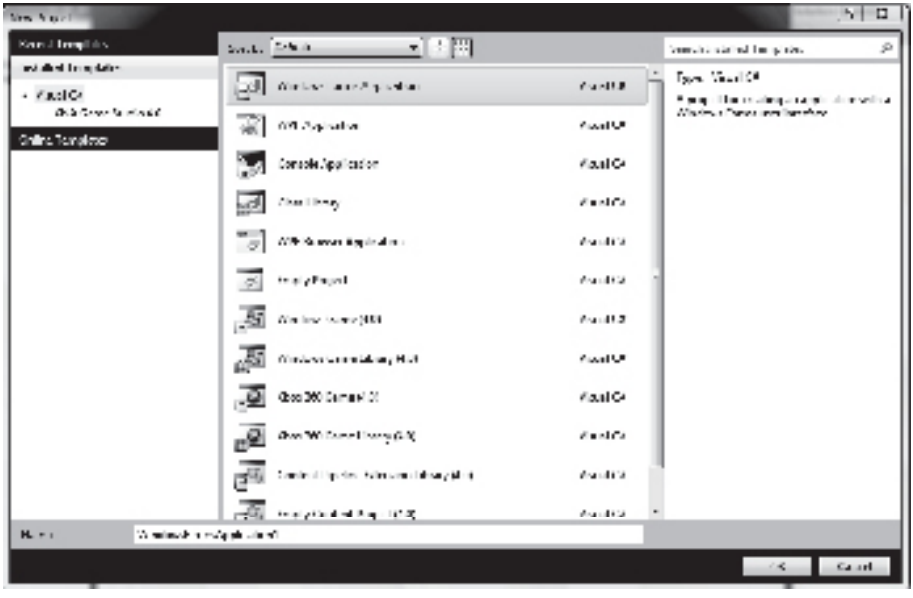

Figura 1.2 – Caixa de diálogo C# New Project Dialog, em que você encontrará o Windows Forms Application.

Se você clicar em Windows Forms Application e digitar `FirstTestApplication` no campo "nome" nessa caixa de diálogo, obterá uma tela como a que vê na figura 1.3. O Solution Explorer estará à direita (se você não o vir, pressione Ctrl+W

e depois a tecla S); ele mostra todos os objetos que estão na sua solução. (Observe que uma solução pode conter vários projetos.) Abaixo dele, você verá a janela Properties, pela qual visualizará e editará propriedades de objetos. À esquerda da tela, você verá a janela Toolbox (talvez veja um número maior ou menor de ferramentas, dependendo do que você instalou). Você pode usar itens da Toolbox arrastando e soltando-os no seu formulário da mesma forma que faria no VB clássico. Na parte inferior da tela, você perceberá a janela Error. Essa janela mostrará a você erros e avisos enquanto escreve o código. Isso pode ser bastante útil enquanto aprende a linguagem. Você não precisa esperar até que compile para descobrir os erros.

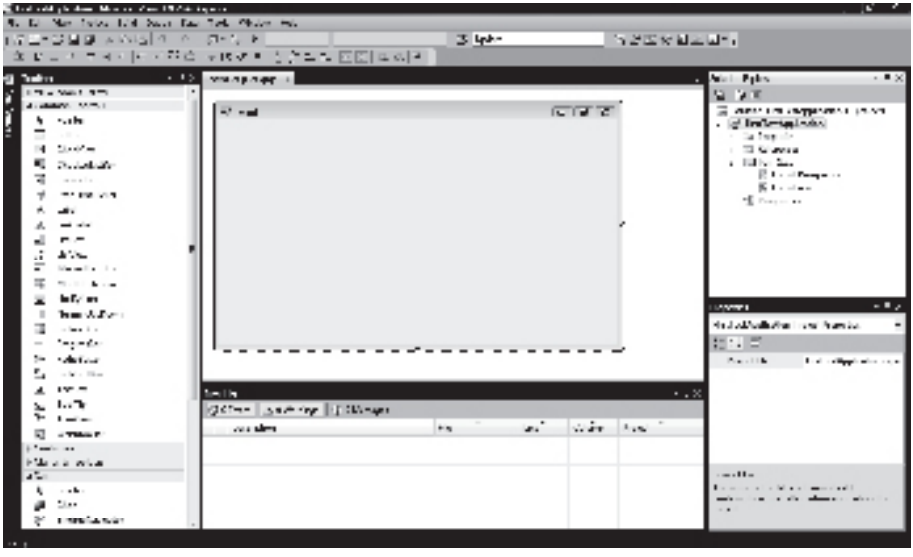

Figura 1.3 – A tela de uma nova aplicação Windows Forms.

Sintaxe básica

A maioria do trabalho que você fará aqui envolve manipulação de objetos, não a criação de objetos complexos, de modo que você não precisa conhecer toda a linguagem C# para iniciar. Há algumas diferenças básicas entre VB6 e C# que é bom conhecê-las desde o início. Essas serão abordadas brevemente aqui e também em maiores detalhes à medida que aparecerem nos exemplos por todo o livro.

Operadores de C#

Estes podem demandar um pouco de tempo para que nos acostumemos a eles. As operações booleanas-padrão que você possa ter usado no VB clássico, às vezes, são as mesmas e, às vezes, são ligeiramente diferentes em C#. Na tabela 1.1, você verá os operadores do VB6 e de C#.

Tabela 1.1 – As diferenças estão nos operadores de igualdade e diferença. Tenha cuidado ao usar o operador = = quando estiver testando igualdade; e o operador = quando estiver atribuindo um valor

Nome do operador	Operador do VB6	Operador de C#
Operador de igualdade	=	==
Operador de diferença	<>	!=
Maior que	>	>
Menor que	<	<
Maior ou igual a	>=	>=
Menor ou igual a	<=	<=

Erros na compilação em virtude do uso de operadores no estilo VB são fáceis de se corrigir quando você conhece o assunto.

Fora os operadores booleanos, há algumas outras pequenas diferenças nos operadores que podem poupar um pouco do seu tempo. A primeira diferença é o operador de incremento. Em VB, você pode já ter feito algo como:

```
X = X + 1
```

Enquanto em C#, você pode usar:

```
X + = 1;
```

Os operadores de incremento de adição, subtração, multiplicação e divisão são `+=`, `-=`, `*=`, e `/=`. Assim, em qualquer lugar onde você usava algo como X = X (operador) Y, pode usar estes como atalho.

Além disso, há alguns outros poucos operadores que podem ajudá-lo em operações sobre muitos dados que não estavam no VB clássico. Por exemplo, se você tiver uma situação em que está tentando processar uma expressão do tipo OR e cada lado dela tiver dados e muito processamento, poderá usar o operador `||`. Fazer isso apenas processará as expressões até que ele retorne "verdadeiro"; assim que uma expressão retornar "verdadeiro", o comando retorna "verdadeiro" e o resto das expressões não é processado. Em operações que

não usam muito processador, você não conseguirá muita economia de tempo com isso. Contudo, quando estiver examinando milhares de linhas de dados de potencialmente milhares de clientes, talvez possa usar esse operador para economizar algum tempo. Essas não são as únicas alterações em operadores, mas são as relevantes para os exemplos deste livro.

Comandos de seleção

As outras alterações que podem demandar um pouco de tempo são os comandos de seleção. No VB clássico, tínhamos `If ... Then ... Else` e `Select ... Case`. Em C#, temos `if ... else` e `switch ... case`. Vamos supor que tenhamos uma variável do tipo inteiro chamada `count` e que estamos tentando processar e uma variável string chamada `reply` na qual queremos colocar uma mensagem. Veja se você consegue encontrar as diferenças, comparadas ao VB, em ambos os comandos:

```
If (count == 0) {
    reply = "The count is 0";
}
else {
    reply = "The count is not 0";
}
switch (count) {
    case 0:
        reply = "The count is 0";
        break;
    default:
        reply="The count is not 0";
        break;
}
```

Observe que, em VB, teríamos que usar a palavra-chave `Then`, que não é usada em C#. Além disso, onde usávamos `Select ... Case` em VB, temos que usar `switch ... case`. E também, em VB, temos uma letra maiúscula no início das palavras-chave, ao passo que em C# estão todas em letras minúsculas. Finalmente, observe as chaves e os pontos e vírgulas que você não usa em VB. Essas diferenças certamente se destacam quanto às aparências, mas, após ter escrito alguns comandos, você as perceberá facilmente.

Há muitas outras diferenças entre as linguagens – destaquei esses exemplos porque são usados com frequência em aplicações que lidam com muitos dados.

Você pode obter uma lista completa de operadores, palavras-chave e comandos no sistema de ajuda do Visual Studio. Além disso, o IntelliSense do Visual Studio é ótimo e pode ajudá-lo muito, além da janela de erros também, especialmente quando você esquece uma chave, um ponto e vírgula ou um comando `include`.

Se você estiver acostumado a trabalhar com o Microsoft Access, talvez esteja mal-acostumado com as coisas que são feitas automaticamente. É muito fácil criar um formulário no Access que permita a você adicionar, atualizar e excluir registros. Além disso, a alteração da fonte de dados para um grid pode, realmente, ser feita com uma linha de código no VBA. Mas, criar a mesma funcionalidade em um aplicativo C# pode requerer um pouco mais de trabalho. Mesmo com toda a sintaxe correta, você precisa saber onde declarar objetos em C#, onde os inicializar, etc. Uma vez entendido onde as coisas precisam ser feitas, tudo ficará muito fácil e você aprenderá rapidamente.

Para este exemplo, mostraremos as telas do Visual Studio 2010 Express, mas o código não mudará se você usar uma versão diferente. Além disso, usaremos o banco de dados Northwind que vem com o Access. O uso do banco de dados Northwind impõe alguns desafios com os quais você irá se deparar ao usar bancos de dados com os quais você não tem controle sobre o esquema. Estss casos serão destacados, e você aprenderá como lidar com eles.

Se você não tiver o Access ou o banco de dados Northwind, poderá baixá-lo do website da Microsoft.

Em primeiro lugar, abra o Visual Studio e clique em File >New Project. Escolha Visual C# e selecione Windows Forms Application. Na parte de baixo dessa caixa de diálogo, digite `EditingDatabaseTest` e depois clique em OK, conforme mostrado na figura 1.4. Feito isso, verá a tela mostrada na figura 1.5.

Para iniciar, recriaremos mais ou menos o que o Access faz automaticamente quando você cria um formulário. Você preencherá um grid com dados, adicionará botões para filtrar os dados e terá um segundo grid que permitirá escolher diferentes tabelas com as quais preencherá o primeiro grid. Além disso, você adicionará o código que vai permitir adicionar, atualizar e excluir linhas de dados. Embora isso pareça bastante simples, verá que há algum planejamento envolvido para realizar essa tarefa.

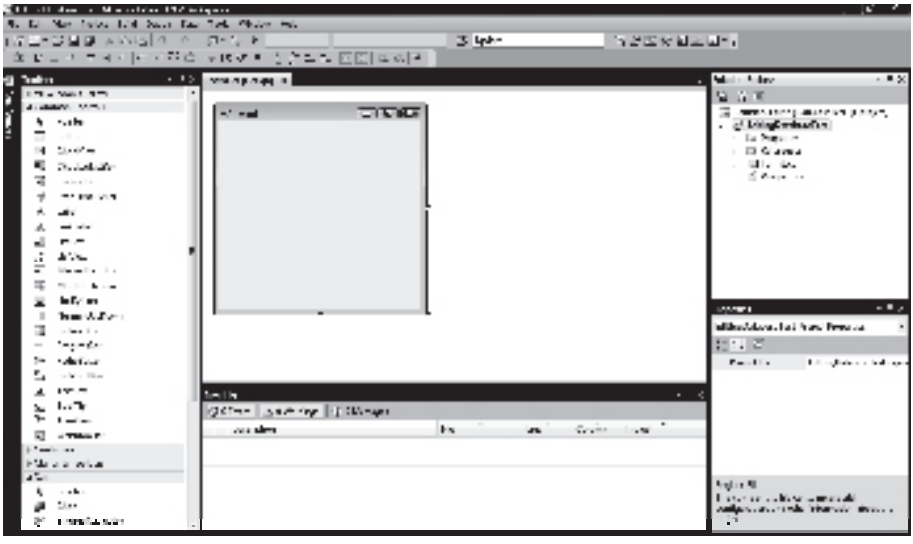

Figura 1.4 – Janela New Project.

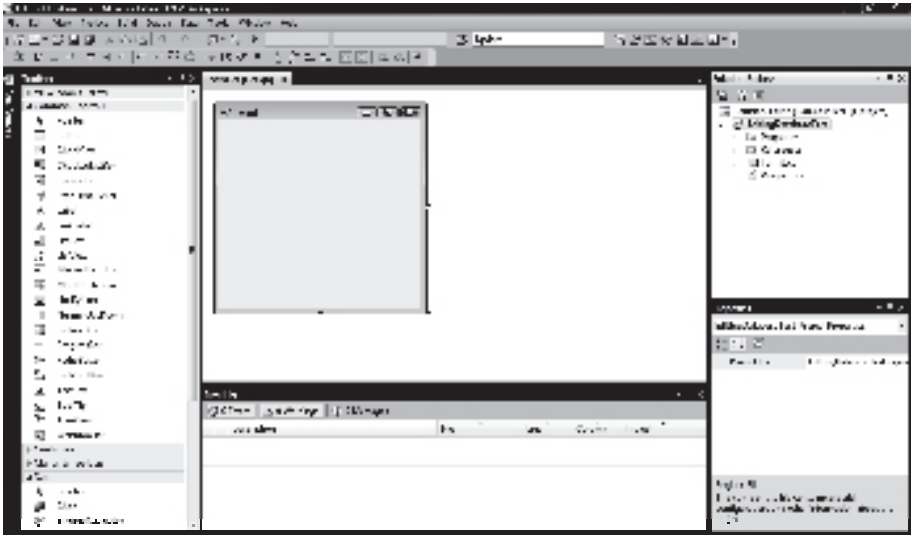

Figura 1.5 – Editando o seu projeto.

Dê uma olhada na Toolbox no lado esquerdo da sua tela. (Se ela não estiver lá, clique em View > Toolbox para exibi-la) Observe as seções – você usará controles das seções Common Controls e Data para este exemplo. No formulário, arraste um datagrid da seção Data, uma caixa de texto da seção Common Controls, um combo box da seção Common Controls, dois botões da seção

Common Controls e um segundo datagrid da seção Data. Quando você inserir um datagrid no formulário, verá as caixas de diálogo do tipo popup mostradas na figura 1.6. Para o primeiro datagrid, deixe marcadas as opções para inserir, atualizar e excluir registros. Para o segundo data grid, desmarque essas opções. Em ambos, deixe Choose Data Source como (none). Você pode criar um datasource de projeto e usá-lo aqui, mas vamos iniciar com a programação do datasource para dar a você mais flexibilidade. Você pode colocar esses controles da forma que quiser (pode ver como eu fiz na figura 1.7). Se você pressionou F5 para iniciar o projeto, ele será aberto sem nada funcionar, ainda.

Figura 1.6 – Escolhendo as fontes de dados para o datagrid.

Figura 1.7 – Um layout inicial de formulário.

A seguir, você precisará escrever algum código para que os controles funcionem. Você pode ver o código para um Form pressionando F7, ou clicando com o botão direito do mouse no nome do formulário no Solution Explorer no lado direito da sua tela e selecionando View Code no menu de contexto exibido. Você verá as linhas de código predefinidas mostradas no exemplo 1.1.

Exemplo 1.1 – Código básico para fazer os controles funcionarem

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace EditingDatabaseTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

A primeira coisa que você perceberá no código é a palavra-chave `using`. Essas linhas de código são muito semelhantes à adição de uma referência no VBA. Quando você insere uma diretiva `using`, ela ativa o Intellisense para os objetos, as propriedades e os métodos relacionados àquele namespace.

Observe que você pode e muitas vezes tem de adicionar referências a um projeto C#; só estou descrevendo isso dessa forma para lhe dar um exemplo familiar.

Existe uma diretiva `using` extra que você precisará inserir para este exemplo funcionar. Logo abaixo da linha `using System.Data;`, digite a seguinte linha de código:

```
using System.Data.OleDb;
```

Essa linha de código diz ao C# para usar o .NET Framework Provider for OLE DB. Você usará objetos, propriedades e métodos desse namespace para fazer a

conexão com a fonte de dados. Além disso, precisará que algumas das variáveis e objetos que está usando permaneçam disponíveis enquanto o formulário estiver aberto. Por esse motivo, você precisa declará-los em nível de classe e não nos procedimentos individuais que escreverá. Insira as linhas necessárias para que seu código fique conforme mostrado no exemplo 1.2.

Exemplo 1.2 – Conectando à fonte de dados com OLE DB

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace EditingDatabaseTest
{
    public partial class Form1 : Form
    {
        public string connString;
        public string query;
        public OleDbDataAdapter dAdapter;
        public DataTable dTable;
        public OleDbCommandBuilder cBuilder;
        public DataView myDataView;

        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Você precisa que os objetos e variáveis que estão declarados permaneçam disponíveis porque eles serão necessários para a atualização, ordenação, filtragem e outras operações que irá programar. Se você não os declarar em nível de classe, esses objetos não ficaram disponíveis fora do procedimento em que foram declarados. Após o comando `Initialize Component()`, acrescente as seguintes linhas de código:

```
connString = "Provider=Microsoft.ACE.OLEDB.12.0;
    DataSource=C:\\users\\michael\\documents\\Northwind 2007.accdb";
query = "SELECT * FROM Customers";
dAdapter = new OleDbDataAdapter(query, connString);
dTable = new DataTable();
cBuilder = new OleDbCommandBuilder(dAdapter);
cBuilder.QuotePrefix = "[";
cBuilder.QuoteSuffix = "]";
myDataView = dTable.DefaultView;
```

A string de conexão é muito semelhante à do VBA. Todavia, você deve perceber os caracteres \\ no nome do caminho. Se você usar uma única \, obterá um erro de sequência de escape não reconhecida. A variável query é uma string que define o comando `Select` que você está usando para acessar os dados. `OleDbDataAdapter` é a classe que armazena comandos de dados (data commands) e a conexão que você usará para preencher o `DataTable`. A classe `OleDbCommandBuilder` gera os comandos que harmonizam as alterações que ocorrem em um `DataTable` e no banco de dados conectado.

Uma vez conectado ao banco de dados Northwind, você precisa das propriedades `QuotePrefix` e `QuoteSuffix` definidas com os colchetes. Isso porque o banco de dados Northwind tem espaços nos nomes dos campos. Se você tentar atualizar uma célula do seu datagrid que tenha espaços no nome do campo, sem essas propriedades definidas, obterá um erro. Você sempre pode capturar esse erro, mas seria impossível atualizar tabelas com espaços nos nomes dos campos. Se você não usar essas propriedades e a sua fonte de dados (datasource) não tiver espaços nos nomes dos campos, poderá executar sem erros. Todavia, recomendo sempre usar essas linhas por precaução. A seguir, adicione as linhas de código seguintes para terminar esse primeiro procedimento:

```
dAdapter.Fill(dTable);
BindingSource bndSource = new BindingSource();
bndSource.DataSource = dTable;
this.dataGridView1.DataSource = bndSource;
for (int q = 0; q <= dataGridView1.ColumnCount - 1; q++)
{
    this.comboBox1.Items.Add
        (this.dataGridView1.Columns[q].HeaderText.ToString());
}
```

```
OleDbConnection xyz = new OleDbConnection(connString);
xyz.Open();
DataTable tbl = xyz.GetSchema("Tables");

dataGridView2.DataSource = tbl;
DataView tbl_dv = tbl.DefaultView;
```

Você está executando várias coisas com esse código. Primeiro, está preenchendo o `DataTable` com os dados do adptador de dados (data adapter). A seguir, está criando uma fonte de conexão (binding source) para o formulário. (A classe `BindingSource` faz parte do namespace `System.Windows.Forms`.) Agora você está pronto para configurar a fonte de dados para o datagrid. Assim que você faz isso, os dados que selecionou preencherão o grid.

A próxima parte do código é um laço `for`, usado para preencher a combobox com os nomes dos campos. O código não fará nada com esses dados, mas você poderia usá-lo para definir o campo de ordenação ou executar algumas outras tarefas. Isso está sendo incluído aqui apenas para mostrar a você um exemplo de como iterar pelas colunas de um datagrid.

Concluindo, a seção final desse trecho de código é usada para preencher o segundo datagrid com o esquema da `OleDbConnection`.

Se você pressionar F5 nesse momento, o formulário abrirá e você verá a tela da figura 1.8. Pelo fato de você ter informado no primeiro datagrid que ele poderia adicionar, atualizar e excluir, poderá editar esses campos. Mas você ainda não adicionou um código para harmonizar essas alterações no banco de dados. Assim, você pode editar o campo e tudo será exibido na tela como se tivesse sido alterado; contudo, se você fechar o formulário e o abrir novamente, as alterações não estarão no banco de dados. Além disso, você perceberá que o datagrid de baixo não pode ser editado, porque você desmarcou as caixas. A coisa importante a observar aqui é que essas configurações só têm influência sobre o grid; elas não impactam o banco de dados. Se você procurar por ajuda sobre datagrids em online C#, verá muitas perguntas de pessoas que fizeram alteração no grid, mas não adicionaram o código para aplicar as atualizações – elas não conseguem entender o porquê de os dados não estarem sendo alterados no banco de dados.

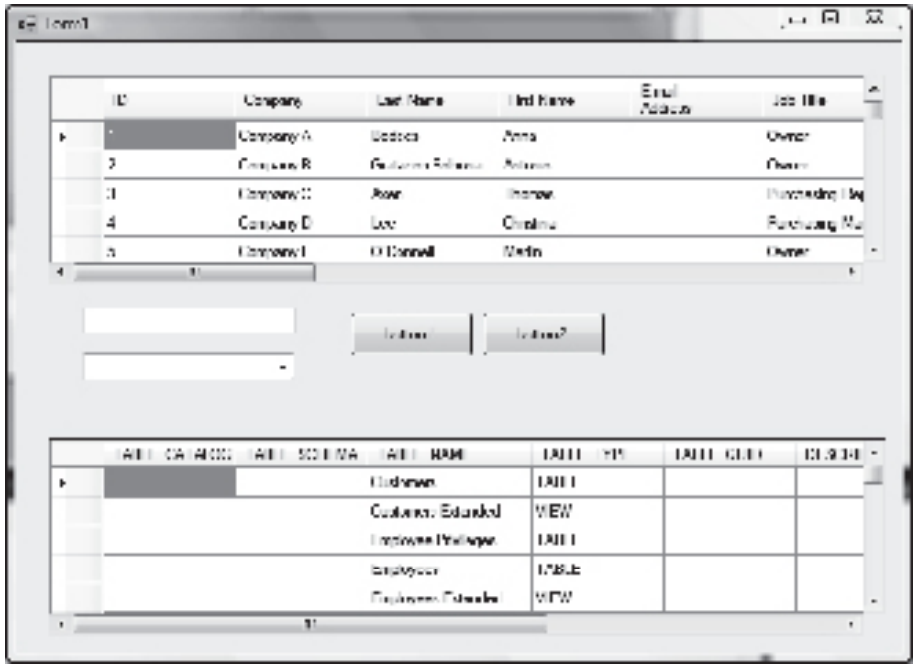

Figura 1.8 – Um datagrid preenchido.

Assim, adicionaremos o código para atualização. Acrescente esse código logo abaixo da chave que termina o procedimento `Form1()`:

```
private void Cell_Update(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        dAdapter.Update(dTable);
        this.textBox1.Text = "Updated " + System.DateTime.Now.ToString();
    }
    catch (OleDbException f)
    {
        this.textBox1.Text = "Not Updated " + f.Source.ToString();
    }
}
```

Assim que você tiver feito isso, precisará configurar o grid para chamar esse procedimento. Alterne para a tela de visualização de projeto (Shift+F7), clique com o botão direito do mouse no primeiro datagrid e selecione `Properties`. Na

janela Properties, clique no ícone que contém um raio para exibir os eventos e encontre o evento chamado `RowValidated`. Nesse evento, selecione `Cell_Update` na caixa drop-down. Ele deve ser o único item disponível na lista, nesse momento.

Quando você está criando um aplicativo, a última coisa que quer é que seus usuários recebam uma janela de depuração ou gerar uma exceção não tratada. Assim, o que fiz nesta seção de código é colocar o código que faz a atualização em um comando `try ... catch`. Você poderia executar essa atualização em uma linha de código: `dAdapter.Update(dTable);`. Entretanto, esse código pode gerar um erro por inúmeros motivos. Por exemplo, você poderia estar atualizando uma tabela que não tivesse uma chave primária definida (isso sempre gerará um erro), ou poderia ter ignorado o passo onde define `QuotePrefix` e `QuoteSuffix` no criador de comandos (command builder) e tenha uma tabela com espaços nos nomes dos campos. Assim, quando isso acontecer, você irá querer que o código trate a exceção adequadamente. Nesse caso, o código tentará executar essa linha e, se funcionar, atualizará a caixa de texto informando ao usuário que foi atualizada. Se ocorrer uma exceção `OleDbException`, ele atualizará a caixa de texto, informando ao usuário que não foi atualizada. No grid também será exibido um "X" vermelho no lado esquerdo da linha que não foi atualizada. Observe que você está apenas capturando uma exceção `OleDbException`. Você pode capturar todas as exceções em vez de definir uma, mas é melhor escrever seções específicas de código para manipular cada tipo de erro que puder ocorrer.

O outro item a ser observado é a variável `dAdapter`. Se você declarar essa variável no procedimento `Form1()`, tudo correrá bem quando o aplicativo começar a ser executado, mas ocasionará um erro ao escrever a seção de atualização do código porque a variável `dAdapter` estará fora de contexto.

Adicionando filtragem

A próxima coisa que você programará aqui é a funcionalidade de filtragem. Volte para a visualização de projeto, no formulário e altere o texto de botões para Set Filter e Clear Filter. Volte para a janela de código para adicionarmos os procedimentos para essa funcionalidade.

Há várias formas para você poder adicionar a funcionalidade de filtros. O que você fará aqui é basicamente a funcionalidade de filtro por seleção do Access, mas deixaremos como default o uso do campo inteiro. Você pode usar caracteres curinga, mas, por enquanto, focaremos o básico. Digite o seguinte código abaixo do procedimento `update`:

```
private void filter_click(object sender, EventArgs e)
{
    string mystr;
    if (myDataView.RowFilter == "")
    {
        mystr = "[" + dataGridView1.CurrentCell.OwningColumn.HeaderText.ToString() + "]";
        mystr += " = '" + dataGridView1.CurrentCell.Value.ToString() + "'";
        myDataView.RowFilter = mystr;
    }
    else
    {
        mystr = myDataView.RowFilter + " and ";
        mystr += "[" + dataGridView1.CurrentCell.OwningColumn.HeaderText.ToString() + "]";
        mystr += " = '" + dataGridView1.CurrentCell.Value.ToString() + "'";
        myDataView.RowFilter = mystr;
    }
}
```

Algumas coisas são importantes aqui. Em primeiro lugar, há uma linha de código que verifica se o grid já está filtrado. Se estiver, clicar no botão de filtragem novamente adiciona ao filtro. Se o filtro estiver vazio, o código apenas configura o filtro. Em segundo lugar, já que não estamos usando a classe `OleDbCommandBuilder` aqui, os colchetes não serão adicionados aos nossos nomes de colunas automaticamente. Assim, você só precisa adicionar os colchetes antes e depois do nome da coluna. Finalmente, você deve examinar todas as propriedades e métodos disponíveis na `CurrentCell`. Nesse caso, você está referenciando a propriedade `OwningColumn` da célula e a propriedade `HeaderText` dessa coluna; a propriedade `HeaderText` é igual ao nome do campo da tabela. Além disso, como no outro procedimento, o objeto ao qual estamos referenciando (`myDataView` nesse caso) é declarado em nível de classe, de modo que está disponível para todos os procedimentos do formulário.

A seguir, você vai querer escrever esse código para ser executado quando o botão Set Filter for clicado. Assim, volte para a visualização de projeto e clique com o botão direito do mouse no primeiro botão (você já deve ter configurado a propriedade `Text` para Set Filter). No menu de contexto exibido, clique em `Properties` e na janela Properties, clique no ícone que contém um raio para exibir os eventos. Encontre o evento click e, na caixa drop-down, clique em `filter_click`.

Volte para a visualização do código e adicione as seguintes linhas de código abaixo do procedimento `filter_click`:

```
private void clear_filter(object sender, EventArgs e)
{
    myDataView.RowFilter = "";
}
```

Volte para a visualização de projeto e configure o evento click para o segundo botão como `clear_filter`, da mesma forma que você fez para o primeiro botão. Talvez perceba que apenas `clear_filter` e `filter_click` estão disponíveis quando você tem outro evento já programado para a atualização de linhas. Isso ocorre porque o procedimento `Cell_Update` é específico do `DataGridViewCellEventArgs`, de forma que só aparecerá para eventos de datagrid.

Assim que você tiver feito isso, pressione F5 e, quando o formulário for aberto, clique na primeira célula sob Job Title, que deve exibir "owner". A seguir, clique no botão Set Filter. Você verá um formulário como o mostrado na figura 1.9.

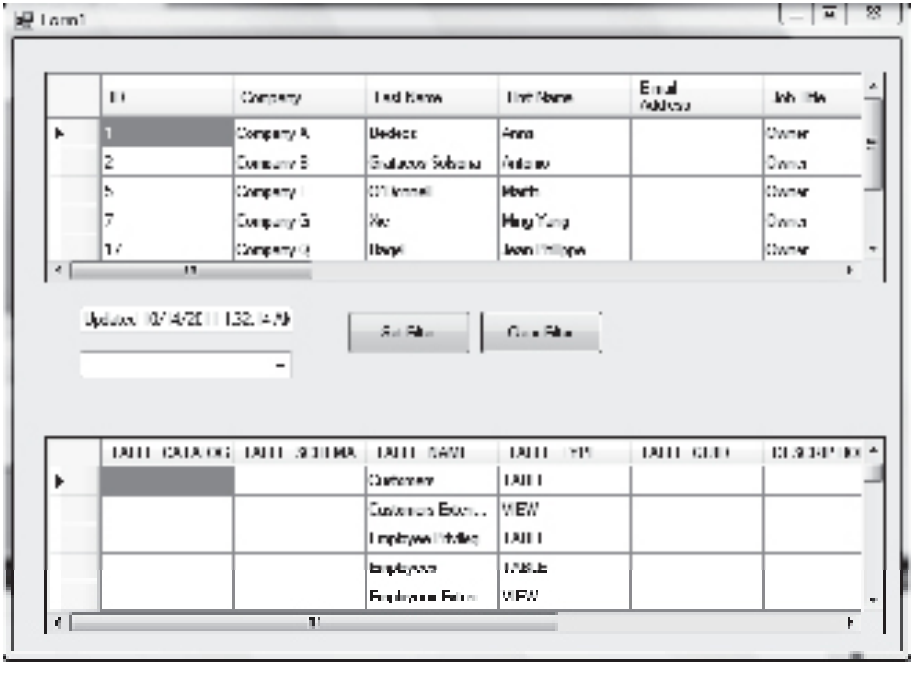

Figura 1.9 – Configurando um filtro.

Se você clicar no botão Clear Filter, ele removerá o filtro. Essa funcionalidade é razoavelmente simples, mas você pode ver como programá-la é um pouco complexo. Não faria sentido passar por tudo isso se o que quiséssemos fosse apenas editar uma tabela estática. Se você quisesse fazer isso, poderia criar um datasource de projeto, que estabelecesse o código para permitir atualizações, edições, exclusões etc. Então, o que estou tentando mostrar aqui é como você pode selecionar uma tabela diferente e preencher o primeiro datagrid.

Sua próxima tarefa é adicionar um outro botão ao formulário e chamá-lo de Change Source. Adicione o seguinte código abaixo do último procedimento que você escreveu:

```
private void change_data_source(object sender, EventArgs e)
{
    string tbl_str = dataGridView2.CurrentRow.Cells[2].Value.ToString();
    query = "SELECT * FROM [" + tbl_str + "]";
    dAdapter = new OleDbDataAdapter(query, connString);
    dTable = new DataTable();
    cBuilder = new OleDbCommandBuilder(dAdapter);
    cBuilder.QuotePrefix = "[";
    cBuilder.QuoteSuffix = "]";
    myDataView = dTable.DefaultView;
    dAdapter.Fill(dTable);
    BindingSource bSource = new BindingSource();
    bSource.DataSource = dTable;
    this.dataGridView1.DataSource = bSource;
    for (int q = 0; q <= dataGridView1.ColumnCount - 1; q++)
    {
        this.comboBox1.Items.Add(this.dataGridView1.Columns[q].HeaderText.ToString());
    }
}
```

Esse código é basicamente o mesmo que o inicial, exceto que estamos definindo o nome da tabela igual à terceira coluna do grid. Observe que as colunas do grid são baseadas em 0, assim a terceira coluna possui índice 2. Uma vez que você tiver feito isto, volte para a visualização de projeto e configure o evento click para `change_data_source`. Seu formulário final deve se parecer como o mostrado na figura 1.10.

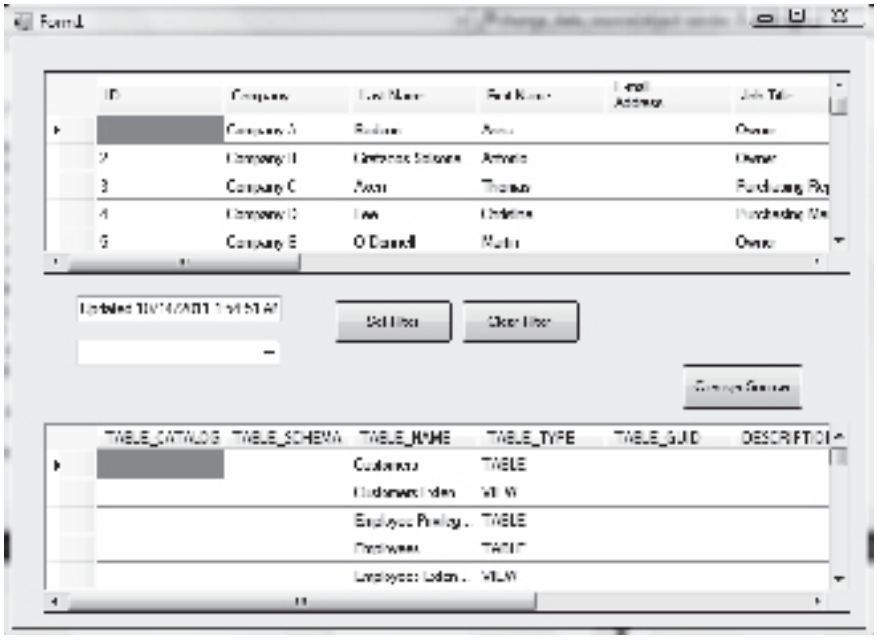

Figura 1.10 – Formulário com o botão Change Source.

Algumas considerações adicionais

Você deve estar ciente de alguns erros que verá no datagrid, especialmente com o banco de dados Northwind. Se tentar adicionar registros a algumas tabelas, verá um ponto de exclamação vermelho à esquerda da linha e, se passar o mouse por cima dela, verá "An `INSERT INTO` query cannot contain a multi-valued field" ("Uma consulta `INSERT INTO` não pode conter um campo multivalorado"). Isso acontece porque algumas das tabelas do banco de dados Northwind aproveitam o recurso único do Access de armazenar mais de um valor em um campo (por exemplo, múltiplos exemplos de uma lista). Já que você não poderá inserir registros no banco de dados se a sua tabela tiver um campo como esse, eu evitaria isso se você estiver planejando atualizar fora do Access.

Vejamos uma situação em que você possa atualizar. No segundo datagrid, clique na célula que diz Invoices (você precisará rolar as linhas para baixo) e depois clique no botão Change Source. Os dados do primeiro datagrid mudarão para mostrar a tabela Invoices. A seguir, role até o fim e tente adicionar uma nova linha. Use o valor 125 como Order ID e 7/1/2011 como Invoice Date e depois coloque zeros nas colunas com números. A seguir, desloque para baixo até a próxima linha ou clique fora da linha que você está tentando adicionar.

Quando você fizer isso, verá um ponto de exclamação vermelho. Quando passar o mouse sobre ele, será informado de que precisa de um registro relacionado na tabela Orders. Assim, altere Order ID para 58 (que existe na tabela Orders) e depois clique fora da linha. Você verá que a atualização funciona. A seguir, clique em Inventory Transactions (logo acima de Invoices no grid de baixo) e clique no botão Change Source. Após tal ação, volte para Invoices e clique novamente no botão Change Source. Se você rolar para baixo, verá as linhas de dados que adicionou e que o banco de dados adicionou a chave primária automaticamente.

Agora, você pode tentar excluir essa linha. Clique no espaço logo à esquerda da primeira coluna dessa linha que você adicionou. Isso destacará a linha. Agora pressione a tecla Delete. Isso exclui a linha do banco de dados.

Você pode perceber a partir desse exemplo que é um pouco mais complicado do que fazer a mesma coisa em Access, assim que você tiver o padrão, será relativamente fácil adicionar um datagrid e alterar a fonte de dados, o filtro etc.

O exemplo 1.3 fornece a listagem completa do exemplo deste capítulo.

Exemplo 1.3 – Juntando todo o código

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace EditingDatabaseTest
{
    public partial class Form1 : Form
    {
        public string connString;
        public string query;
        public OleDbDataAdapter dAdapter;
        public DataTable dTable;
        public OleDbCommandBuilder cBuilder;
        public DataView myDataView;
```

```
public Form1()
{
    InitializeComponent();
    connString = "Provider=Microsoft.ACE.OLEDB.12.0;
    Data Source=C:\\users\\michael\\documents\\Northwind 2007.accdb";
    query = "SELECT * FROM Customers";
    dAdapter = new OleDbDataAdapter(query, connString);
    dTable = new DataTable();
    cBuilder = new OleDbCommandBuilder(dAdapter);
    cBuilder.QuotePrefix = "[";
    cBuilder.QuoteSuffix = "]";
    myDataView = dTable.DefaultView;
    dAdapter.Fill(dTable);
    BindingSource bndSource = new BindingSource();
    bndSource.DataSource = dTable;
    this.dataGridView1.DataSource = bndSource;
    for (int q = 0; q <= dataGridView1.ColumnCount - 1; q++)
    {
        this.comboBox1.Items.Add(this.dataGridView1.Columns[q].HeaderText.ToString());
    }
    OleDbConnection xyz = new OleDbConnection(connString);
    xyz.Open();
    DataTable tbl = xyz.GetSchema("Tables");
    dataGridView2.DataSource = tbl;
    DataView tbl_dv = tbl.DefaultView;
}

private void Cell_Update(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        dAdapter.Update(dTable);
        this.textBox1.Text = "Updated " + System.DateTime.Now.ToString();
    }
    catch (OleDbException f)
    {
        this.textBox1.Text = "Not Updated " + f.Source.ToString();
    }
}
```

```
        private void filter_click(object sender, EventArgs e) {
            string mystr;
            if (myDataView.RowFilter == "")
            {
                mystr = "[" + dataGridView1.CurrentCell.OwningColumn.HeaderText.ToString() + "]";
                mystr += " = '" + dataGridView1.CurrentCell.Value.ToString() + "'";
                myDataView.RowFilter = mystr;
            }
            else
            {
                mystr = myDataView.RowFilter + " and ";
                mystr += "[" + dataGridView1.CurrentCell.OwningColumn.HeaderText.ToString()+ "]";
                mystr += " = '" + dataGridView1.CurrentCell.Value.ToString() + "'";
                myDataView.RowFilter = mystr;
            }
        }

        private void clear_filter(object sender, EventArgs e) {
            myDataView.RowFilter = "";
        }

        private void change_data_source(object sender, EventArgs e) {
            string tbl_str = dataGridView2.CurrentRow.Cells[2].Value.ToString();
            query = "SELECT * FROM [" + tbl_str + "]";
            dAdapter = new OleDbDataAdapter(query, connString);
            dTable = new DataTable();
            cBuilder = new OleDbCommandBuilder(dAdapter);
            cBuilder.QuotePrefix = "[";
            cBuilder.QuoteSuffix = "]";
            myDataView = dTable.DefaultView;
            dAdapter.Fill(dTable);
            BindingSource bSource = new BindingSource();
            bSource.DataSource = dTable;
            this.dataGridView1.DataSource = bSource;
            for (int q = 0; q <= dataGridView1.ColumnCount - 1; q++) {
                this.comboBox1.Items.Add(this.dataGridView1.Columns[q].HeaderText.ToString());
            }
        }
    }
}
```

Antes de passar para o próximo capítulo e conectar-se ao SQL Server, vamos revisar algumas das diferenças entre o acesso a dados no Microsoft Access e em C#. Um dos maiores desafios é configurar os eventos para dispararem no momento certo e declarar as variáveis no lugar correto. Neste exemplo, isso foi feito para você. Mas, quando você está escrevendo a partir do zero, é fácil cometer alguns erros. Você saberá que isso aconteceu no momento em que tentou acessar uma variável que o Visual Studio diz que está fora de contexto. Assim, quando isso ocorrer, você saberá exatamente onde ir.

Os eventos são um pouco mais complicados. Como exemplo, algumas pessoas percorrerão os que estiverem disponíveis no datagrid e poderão escolher um evento como `CellEndEdit` para escrever o código de atualização. Todavia, você acabará com tais erros quando tentar adicionar novas linhas porque faltarão campos obrigatórios no momento de a atualização ser disparada logo após a primeira coluna ser atualizada. Se você procurar em alguns fóruns de tecnologia, verá algumas discussões sobre onde disparar o evento de atualização. Minha opinião é melhor fazer isso após a linha ser validada, já que só irá disparar quando você sair de uma linha. Você também tem a opção de colocar um botão Save e só disparar as atualizações quando este for clicado. A questão é que você tem opções para quando chamar eventos e pode testá-las para ver onde funciona melhor no seu aplicativo específico.

O item de importância final é entender quando você está lidando com um objeto ou controle que está no namespace Windows Forms ou no namespace `System.Data` e quando está no namespace `System.Data.OleDb`. Há momentos em que você pode querer experimentar algo, mas não consegue encontrar o objeto ou método que quer. Quando você se deparar com situações como essa, tudo o que precisa fazer é passar o mouse sobre o nome da classe pelo qual declarou a variável e ela informará o namespace em que essa classe está.

O que há a seguir?

O próximo capítulo conectará o SQL Server. Se você não possui o SQL Server, pode baixar o SQL Server Express. Além disso, você verá alguns outros exemplos em capítulos posteriores sobre databinding sem grid e até mesmo sobre o retorno de dados a partir de um banco de dados em um webservice. Este primeiro capítulo de codificação estabelece a base para tudo o que vier a ser abordado. Se você pretende aproveitar o que vem a seguir, irá querer assegurar-se de entender tudo neste capítulo antes de seguir em frente.