

# 10

---

## Databases



### 10.1 An Introduction to Databases 458

◆ Accessing a Database Table ◆ Binding to Additional Tables ◆ Browsing a Connected Database ◆ Querying a Table with LINQ ◆ Primary and Foreign Keys ◆ The Join of Two Tables

### 10.2 Editing and Designing Databases 479

◆ A Program to Edit the Cities Table ◆ Designing the Form for the Table-Editing Program ◆ Writing the Table-Editing Program ◆ Principles of Database Design

**Summary 486**

**Programming Projects 487**



VideoNote  
Introduction to  
databases

## 10.1 An Introduction to Databases

The management of databases is one of the most important uses of computers today. Airlines use databases to handle nearly 1.5 billion passenger reservations per year. The 7,500 hospitals in the United States utilize databases to document the care of over 30 million patients per year. Banks in the United States employ databases to monitor 1.6 billion credit cards. Although databases vary considerably in size and complexity, most of them adhere to the fundamental principles of design discussed in this chapter. That is, they are composed of a collection of inter-related tables.

A **table** is a rectangular array of data. Table 10.1 provides information about large cities. Each column of the table is called a **field**. (The third column gives the 2010 population in millions and the fourth column the projected 2015 population in millions.) The names of the fields are *name*, *country*, *pop2010*, and *pop2015*. Each row, called a **record**, contains the same type of information as every other row. Also, the pieces of information in each row are related because they all apply to a specific city. Table 10.2, Countries, has three fields and ten records.

**TABLE 10.1** Cities.

name	country	pop2010	pop2015
Bombay	India	20.1	22.0
Buenos Aires	Argentina	13.1	13.4
Calcutta	India	15.6	17.0
Delhi	India	17.0	18.7
Dhaka	Bangladesh	14.8	17.0
Mexico City	Mexico	19.5	20.2
New York	USA	19.4	20.0
Sao Paulo	Brazil	19.6	20.1
Shanghai	China	15.8	17.2
Tokyo	Japan	36.1	36.4

**Note:** The population figures are for “urban agglomerations”—that is, contiguous densely populated urban areas.

**TABLE 10.2** Countries.

name	pop2010	monetaryUnit
Argentina	41.9	peso
Bangladesh	152.6	raka
Brazil	195.2	real
China	1379.7	yuan
India	1196.8	rupee
Indonesia	258.5	rupiah
Japan	129.0	yen
Mexico	117.4	peso
Pakistan	184.2	rupee
USA	310.1	dollar

A **relational database** contains a collection of one or more (usually related) tables that has been created with **database-management** software. Microsoft Access is one of the best known database-management products. Some other prominent ones are Oracle, SQL Server,

and MySQL. VB 2010 can interact with a database that has been created with any of these products.

The databases needed for the exercises in this textbook are contained in the materials downloaded from the companion website. They are in the folder `Programs\Ch10\Databases`. The database files were created with Microsoft Access 2007 and have the extension *accdb* (an abbreviation for ACCess DataBase). For instance, *Megacities.accdb* is the database file containing the two tables presented above. When the tables were created, each field was given a name and a data type. In the *Cities* table, the fields *pop2010* and *pop2015* were given data type Double and the other fields a data type compatible with the String data type.

### ■ Accessing a Database Table

Before a program can access a table from a database, a connection must be established. The following steps provide one way to connect to the *Megacities* database and bind to the *Cities* table.

1. Start a new program.
2. Add a BindingSource control to the Form Designer. (The BindingSource control can be found in the *All Windows Forms* or *Data* groups of the Toolbox.) After you double-click on the control in the Toolbox, a control named BindingSource1 appears in the component tray at the bottom of the Form Designer.
3. Go to the Properties window for BindingSource1 and click on the down-arrow at the right side of the DataSource property's Settings box. The panel in Fig. 10.1 appears.

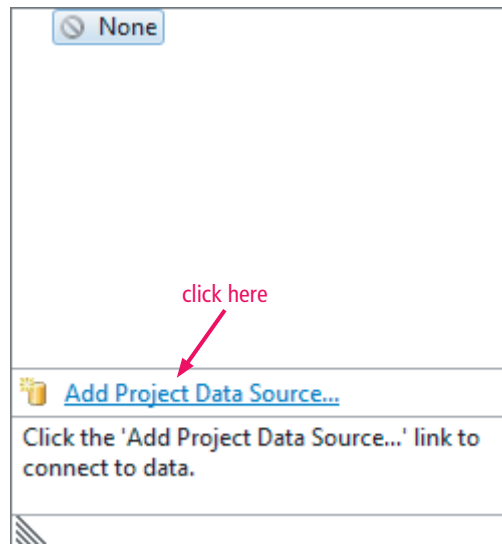


FIGURE 10.1 Panel produced by Step 3.

4. Click on *Add Project Data Source*. The Data Source Configuration Wizard in Fig. 10.2 on the next page appears and asks you to “Choose a Data Source Type”.
5. Select the Database icon in the Data Source Configuration Wizard and click on the *Next* button. The Wizard now asks you to “Choose a Database Model”.
6. Select the Dataset icon and click on the *Next* button. The Wizard now asks you to “Choose Your Data Connection”.
7. Click on the *New Connection* button. An Add Connection window similar to the one in Fig. 10.3 appears.

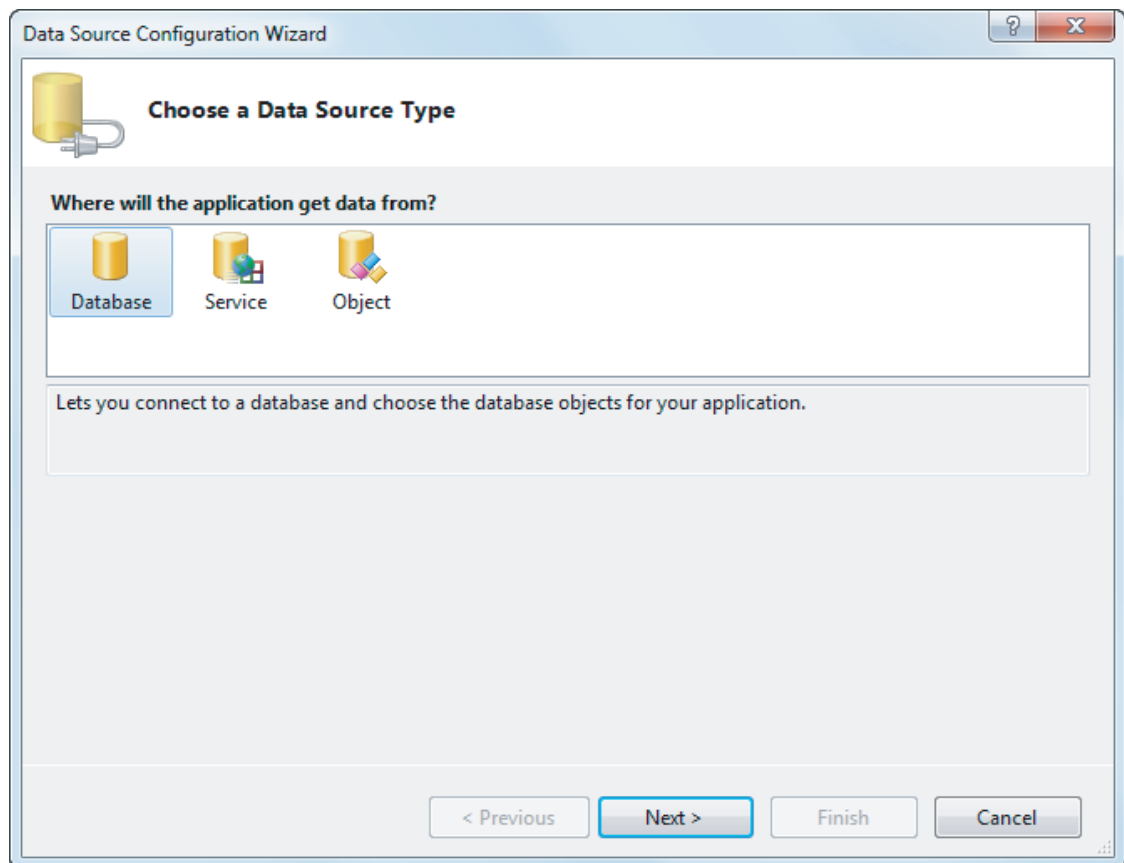


FIGURE 10.2 Window produced by Step 4.

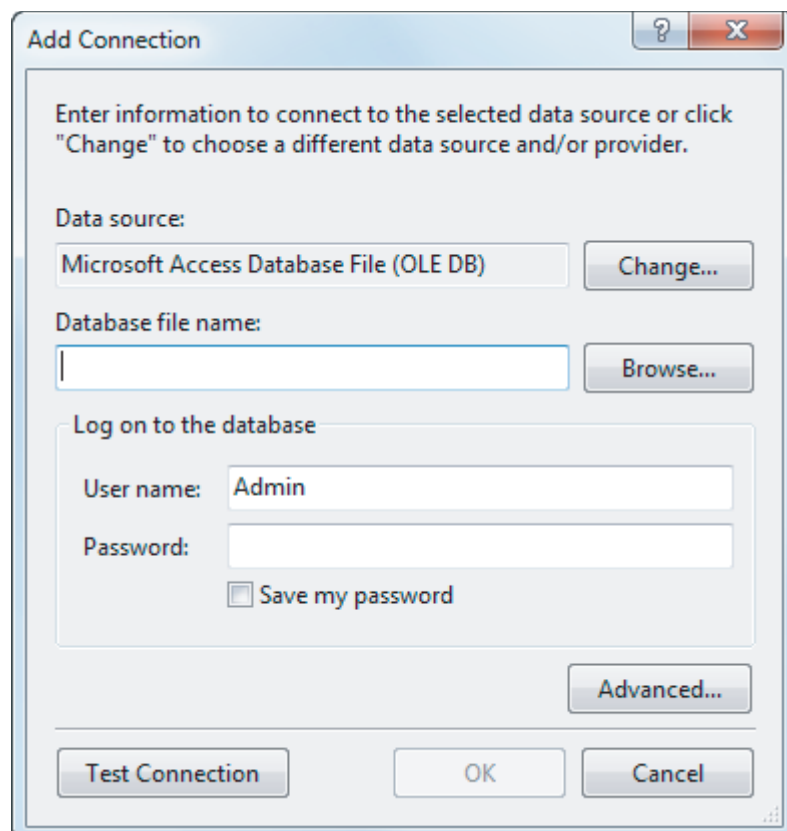


FIGURE 10.3 Window produced by Step 7.

8. If the “Data source:” text box does not say “Microsoft Access Database File (OLE DB)”, click on the *Change* button. A Change Data Source window will appear. Select *Microsoft Access Database File* from the window’s list box and click on the OK button. You will be returned to the Add Connection window shown in Fig. 10.3.
9. Click on the *Browse* button, navigate to and open the Databases folder (a subfolder of Programs\Ch10), double-click on Megacities.accdb, and click on the OK button. The Data Source Configuration Wizard that appeared in Step 6 reappears with the text box now containing Megacities.accdb.
10. Click on the *Next* button. The window in Fig. 10.4 appears, asking whether you would like to place a copy of the file Megacities.accdb into the program.

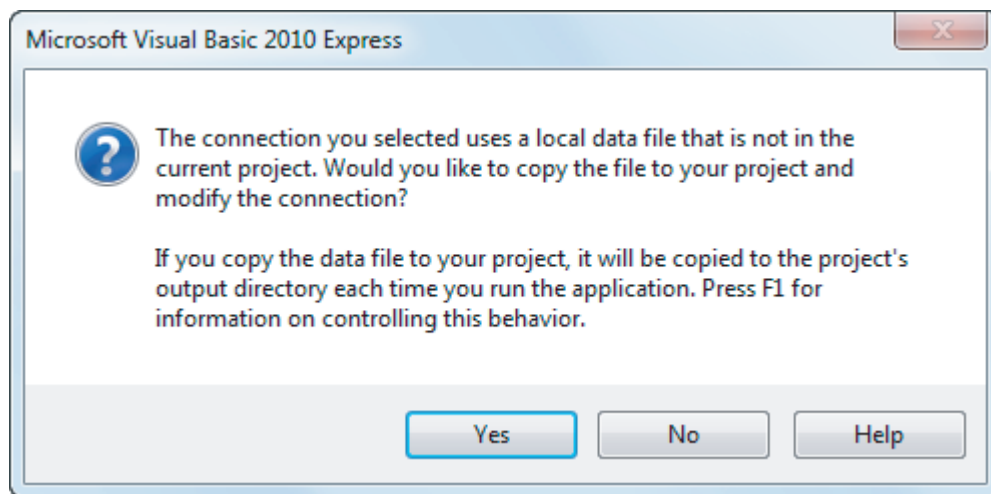


FIGURE 10.4 Window produced by Step 10.

11. Click on the *Yes* button. The Data Source Configuration Wizard will appear and ask whether you would like to Save the Connection String to the Application Configuration File. See Fig. 10.5 on the next page.
12. Make sure that the *Yes* check box is checked and then click on the *Next* button. The Data Source Configuration Wizard in Fig. 10.6 will appear.
13. Check the *Tables* check box and then click on the *Finish* button. The DataSource property of BindingSource1 is now set to MegacitiesDataSet, and a MegacitiesDataSet icon has appeared in the component tray.
14. In the BindingSource1 Properties window, click on the down-arrow at the right side of the DataMember property’s Settings box. A drop-down list containing the tables in the Megacities database will appear.
15. Click on *Cities* in the drop-down list. A CitiesTableAdapter icon will appear in the component tray. Also, a Load event procedure containing one line of executable code is generated in the Code Editor. See Fig. 10.7 on page 463.

The Cities table can now be accessed by the program. You can easily view a list of the fields for both of the tables in the Megacities database. Just bring up the Solution Explorer window and double-click on the file MegacitiesDataSet.xsd to display the page in Fig. 10.8. (**Note:** To close the page, click on the × symbol on its tab.)

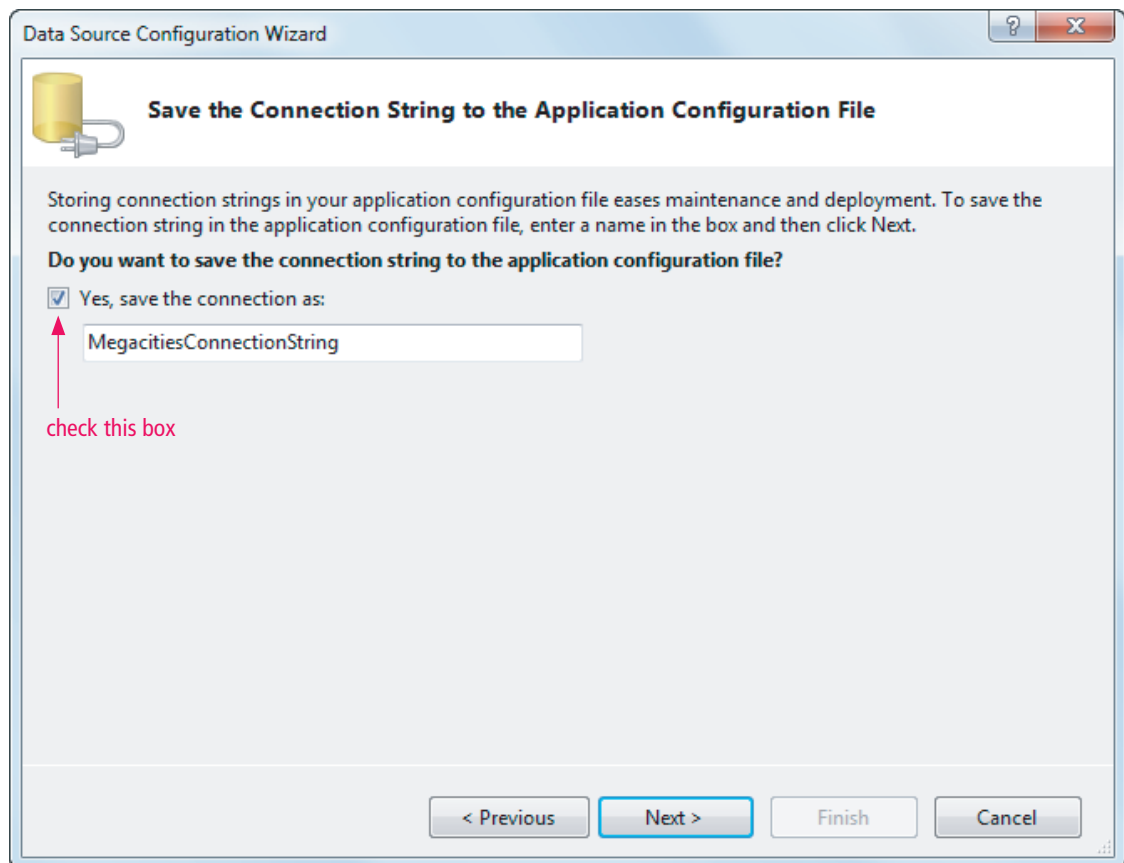


FIGURE 10.5 Window produced by Step 11.

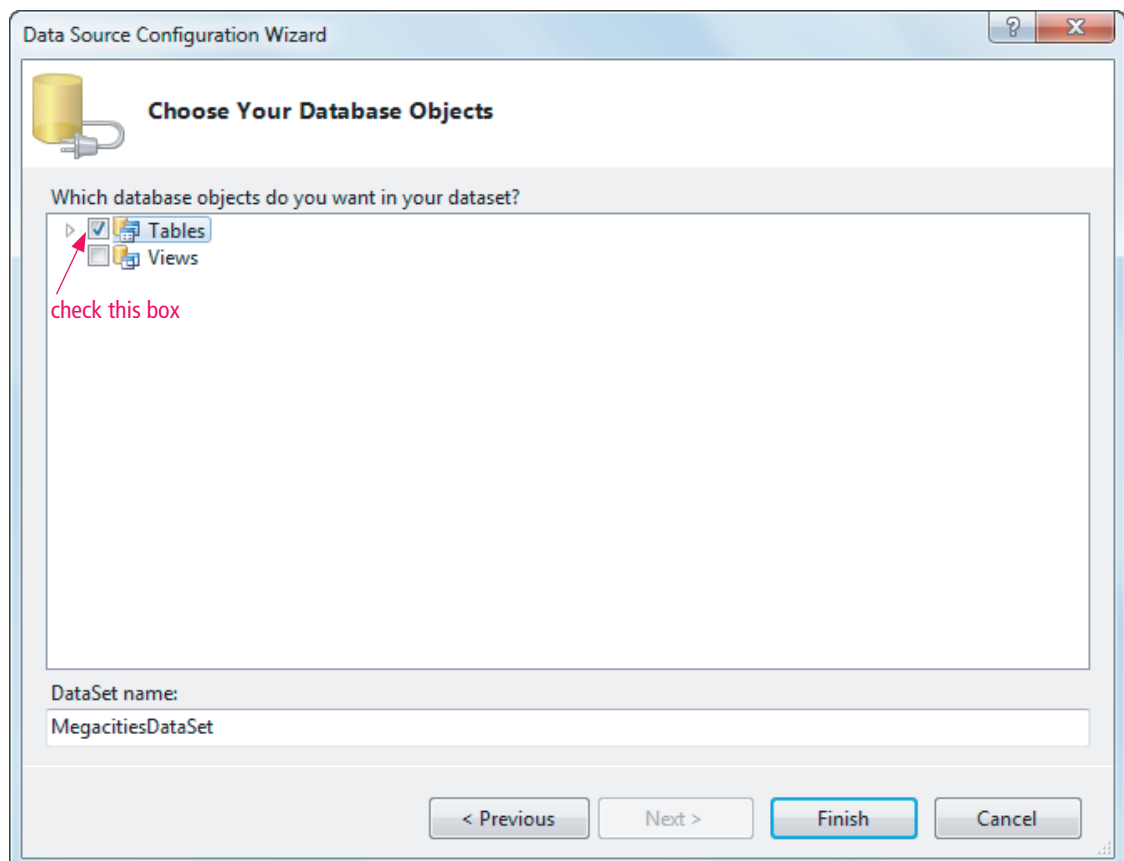


FIGURE 10.6 Window produced by Step 12.

```

Private Sub Form1_Load(...) Handles MyBase.Load
    'TODO: This line of code loads data into the 'MegacitiesDataSet.Cities'
    'table. You can move, or remove it, as needed.
    Me.CitiesTableAdapter.Fill(Me.MegacitiesDataSet.Cities)
End Sub

```

FIGURE 10.7 Code generated by Step 15.

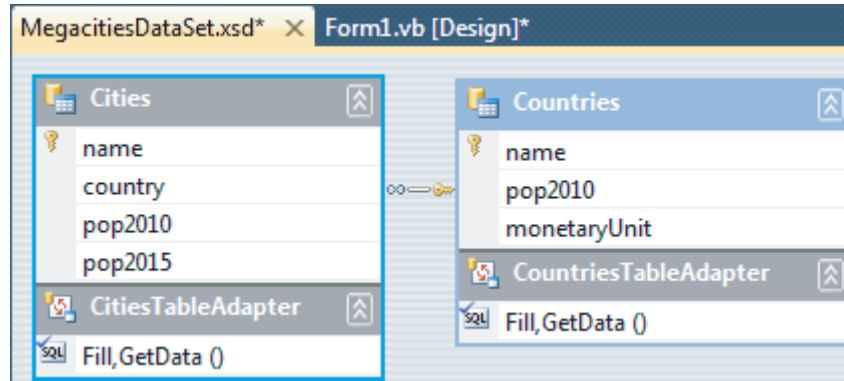


FIGURE 10.8 Tables and fields in the Megacities database.

### ■ Binding to Additional Tables

A program can access many tables through one database connection. The following steps bind the Countries table to the program created in the walkthrough above.

1. Add another BindingSource control to the Form Designer.
2. Set its DataSource property to MegacitiesDataSet. (After you click on the down-arrow, click on the right-pointing triangle (or plus box) to the left of Other Data Sources, click on the right-pointing triangle (or plus box) to the left of Project Data Sources, and then click on MegacitiesDataSet.)
3. Set the DataMember property of the new BindingSource control to Countries. A CountriesTableAdapter icon will appear in the component tray, and another line of code will be added to the Load event procedure.

### ■ Browsing a Connected Database

After a database has been connected to a program, any table from the database can easily be displayed. The following steps display a table from the Megacities database.

1. Right-click on the name of the database (Megacities.accdb) in the Solution Explorer and click on Open in the context menu. With Visual Basic Express the Database Explorer window in Fig. 10.9

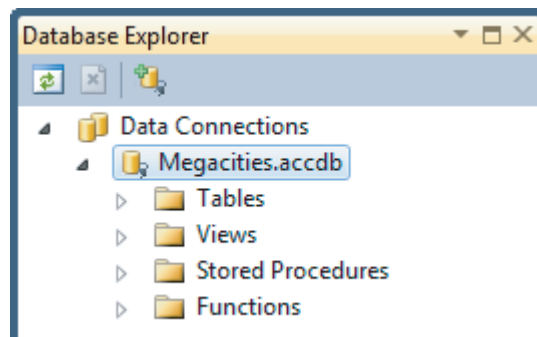


FIGURE 10.9 Database Explorer window.

will appear in the location occupied by the Toolbox. (With Visual Studio, the Server Explorer window will appear.)

2. Click on the right-pointing triangle (or plus box) to the left of the Tables folder. The folder will open and reveal the names of the two tables.
3. Right-click on Cities, and click on *Retrieve Data* (with VB Express) or *Preview Data* (with Visual Studio) in the context menu. The contents of the Cities table will be displayed in the tabbed page shown in Fig. 10.10.
4. Click on the × symbol on the page's tab to close the page.

	name	country	pop2010	pop2015
▶	Bombay	India	20.1	22
	Buenos Aires	Argentina	13.1	13.4
	Calcutta	India	15.6	17
	Delhi	India	17	18.7
	Dhaka	Bangladesh	14.8	17
	Mexico City	Mexico	19.5	20.2
	New York	USA	19.4	20
	Sao Paulo	Brazil	19.6	20.1
	Shanghai	China	15.8	17.2
	Tokyo	Japan	36.1	36.4
*	NULL	NULL	NULL	NULL

FIGURE 10.10 Contents of the Cities table.



VideoNote  
Querying tables

### ■ Querying a Table with LINQ

Databases are usually quite large and so we rarely want to display an entire table. LINQ can be used to extract information from a data table using similar syntax as used to extract information from an array of records, a CSV text file, or an XML file.

A database table can be thought of a sequence of rows with each row containing several fields. If *line* is a row of a data table, then the elements of the row are identified by the names `line.fieldName1`, `line.fieldName2`, and so on. For instance, if *city* is the first row of the Cities table above, then the value of `city.name` is Bombay, the value of `city.country` is India, the value of `city.pop2010` is 20.1, and the value of `city.pop2015` is 22.0.

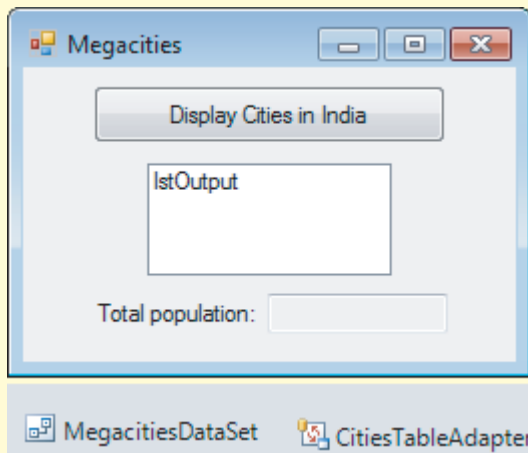


#### Example 1

The following program uses the Cities table of the Megacities database and displays the names of the cities that are located in India. The cities are sorted by their 2010 population in decreasing order. The program also displays the total population of those cities in a text box. `MegacitiesDataSet.Cities` serves as the data source for the LINQ query. **Note:** In the



Order By clause of query1, there was no need to use the CDbl function. The data type Double was given to the *pop2010* field when the database was created, and therefore LINQ knows that *city.pop2010* has type Double.

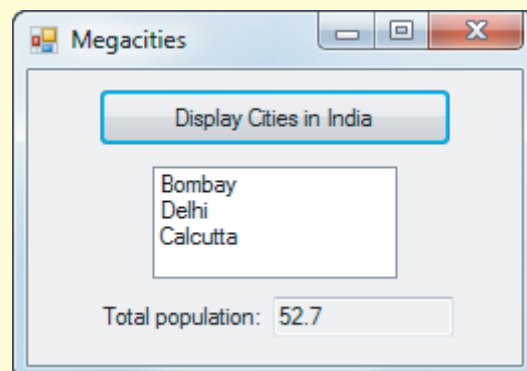


OBJECT	PROPERTY	SETTING
frmCities	Text	Megacities
btnDisplay	Text	Display Cities in India
lstOutput		
lblTotalPop	Text	Total population:
txtTotalPop	ReadOnly	True

```
Private Sub frmCities_Load(...) Handles MyBase.Load
    'code generated automatically when DataMember was set to Cities
    Me.CitiesTableAdapter.Fill(Me.MegacitiesDataSet.Cities)
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query1 = From city In MegacitiesDataSet.Cities
                  Where city.country = "India"
                  Order By city.pop2010 Descending
                  Select city.name
    lstOutput.DataSource = query1.ToList
    lstOutput.SelectedItem = Nothing
    Dim query2 = From city In MegacitiesDataSet.Cities
                  Where city.country = "India"
                  Select city.pop2010
    txtTotalPop.Text = CStr(query2.Sum)
End Sub
```

[Run, and click on the button.]



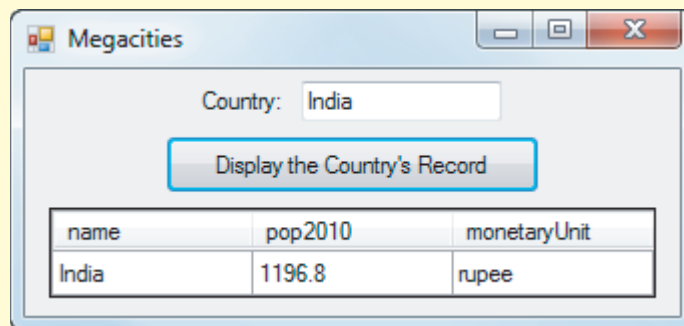
**Example 2**

The following program searches the Countries table for a country requested by the user. Notice that the DataSource method displays “name”, “pop2010”, and “monetaryUnit” in the headers of the table, not “country.name”, “country.pop2010”, and “country.monetaryUnit”. The variable *country* plays a supporting role similar to that of a looping variable in a For Each loop.

```
Private Sub frmCountries_Load(...) Handles MyBase.Load
    'code generated automatically when DataMember was set to Cities
    Me.CountriesTableAdapter.Fill(Me.MegacitiesDataSet.Countries)
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query = From country In MegacitiesDataSet.Countries
                Where country.name = txtName.Text
                Select country.name, country.pop2010, country.monetaryUnit
    If query.Count = 1 Then
        dgvOutput.DataSource = query.ToList
        dgvOutput.CurrentCell = Nothing
    Else
        MessageBox.Show("Country is not in the table.", "Not Found")
    End If
End Sub
```

[Run, enter a country into the text box, and click on the button.]

**Example 3**

The following program uses the Cities table of the Megacities database and displays the cities whose populations are predicted to increase by more than 1 million people from 2010 to 2015. The cities are ordered by their projected population increase, and both the city names and population increases (in millions) are displayed.

```
Private Sub frmCities_Load(...) Handles MyBase.Load
    Me.CitiesTableAdapter.Fill(Me.MegacitiesDataSet.Cities)
End Sub

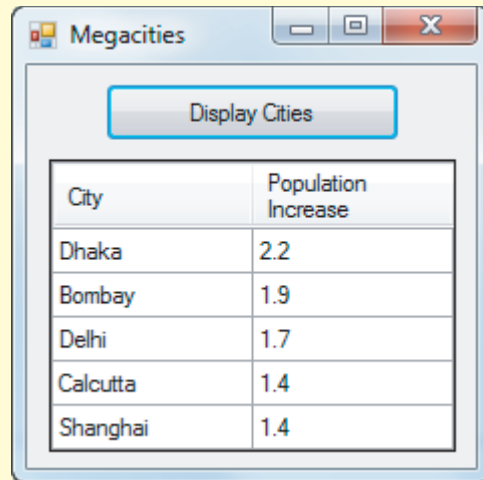
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query = From city In MegacitiesDataSet.Cities
                Let popIncrease = city.pop2015 - city.pop2010
                Let formattedIncr = FormatNumber(popIncrease, 1)
                Where popIncrease > 1
                Order By popIncrease Descending
                Select city.name, formattedIncr
```

```

dgvOutput.DataSource = query.ToList
dgvOutput.CurrentCell = Nothing
dgvOutput.Columns("name").HeaderText = "City"
dgvOutput.Columns("formattedIncr").HeaderText = "Population Increase"
End Sub

```

[Run, and click on the button.]



#### Example 4

The following program displays the cities from the Cities table in a list box sorted by their population in 2010. When the user clicks on one of the cities, its country, population in 2010, and population in 2015 are displayed in text boxes. In the second event procedure, a query is used to search for the desired record of the table. In this case, the query returns a sequence of one value. Since a sequence cannot be assigned to a text box, the First method is used to obtain the desired value. **Note:** Notice that the statement `lstCities.DataSource = query.ToList` in the Load event procedure is not followed by the statement `lstCities.SelectedItem = Nothing`. Had the SelectedItem statement been added, the initial display would show the data about Tokyo without revealing the city that the data referred to.

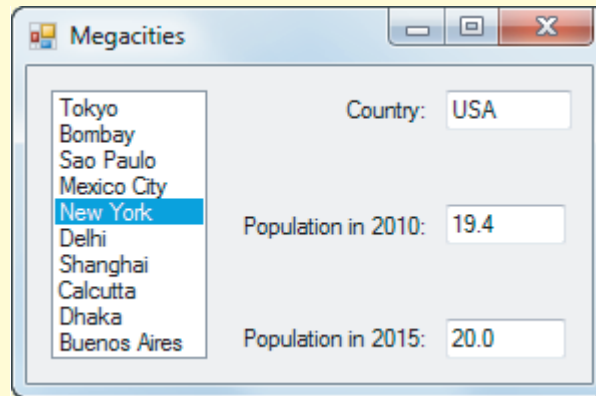
```

Private Sub frmCities_Load() Handles MyBase.Load
    Me.CitiesTableAdapter.Fill(Me.MegacitiesDataSet.Cities)
    Dim query = From city In MegacitiesDataSet.Cities
                Order By city.pop2010 Descending
                Select city.name
    lstCities.DataSource = query.ToList
End Sub

Private Sub lstCities_SelectedIndexChanged(...) Handles _
    lstCities.SelectedIndexChanged
    Dim query = From city In MegacitiesDataSet.Cities
                Where city.name = lstCities.Text
                Select city.country, city.pop2010, city.pop2015
    txtCountry.Text = query.First.country
    txtPop2010.Text = FormatNumber(query.First.pop2010, 1)
    txtPop2015.Text = FormatNumber(query.First.pop2015, 1)
End Sub

```

[Run, and click on one of the cities in the list box.]



### ■ Primary and Foreign Keys

A well-designed table should have a field (or set of fields) that can be used to uniquely identify each record. Such a field (or set of fields) is called a **primary key**. For instance, in the Cities and Countries tables, each *name* field is a primary key. Databases of student enrollments in a college usually use a field of student ID numbers as the primary key. Student names would not be a good choice, because there could easily be two students having the same name.

When a table is created, a field can be specified as a primary key. If so, Visual Basic will insist that every record has an entry in the primary key and that the same entry does not appear in two different records. If the user tries to add a record with no data in the primary key, the error message “Index or primary key cannot contain a Null Value.” will be generated. If the user tries to add a record with the same primary key data as another record, an error message will be displayed: “The changes you requested to the table were not successful because they would create duplicate values in the index, primary key, or relationship. Change the data in the field or fields that contain duplicate data, remove the index, or redefine the index to permit duplicate entries and try again.”

When a database contains two or more tables, they are usually related. For instance, the two tables Cities and Countries are related by their fields that hold names of countries. Let’s refer to these two fields as *Cities.country* and *Countries.name*. Notice that every entry in *Cities.country* appears uniquely in *Countries.name* and that *Countries.name* is a primary key of the Countries table. We say that *Cities.country* can serve as a **foreign key** of *Countries.name*. Foreign keys are usually specified when a database is first created. If so, Visual Basic will enforce the **Rule of Referential Integrity**—namely, that each value in the foreign key must also appear as a value in the primary-key field of the other table.

In the Megacities database, *Cities.name* and *Countries.name* have been specified as primary keys for their respective tables, and *Cities.country* has been specified as a foreign key of *Countries.name*. If the user tries to add to the Cities table a city whose country does not appear in the Countries table, an error message will be displayed: “You cannot add or change a record because a related record is required in table ‘Countries’.” The message will also be generated if the user tries to delete a country from the *Countries.name* field that appears in the *Cities.country* field.

### ■ The Join of Two Tables

A foreign key allows Visual Basic to link (or **join**) two tables from a relational database in a meaningful way. For instance, when the two tables Cities and Countries from the Megacities database are joined based on the foreign key *Cities.country*, the result is Table 10.3. The record

for each city is expanded to show its country's 2010 population and its monetary unit. This joined table is very handy if, say, we want to display a city's currency.

**TABLE 10.3** A join of two tables.

Cities.name	Cities. country	Cities. pop2010	Cities. pop2015	Countries. name	Countries. pop2010	Countries. monetaryUnit
Bombay	India	20.1	22.0	India	1196.8	rupee
Buenos Aires	Argentina	13.1	13.4	Argentina	41.9	peso
Calcutta	India	15.6	17.0	India	1196.8	rupee
Delhi	India	17.0	18.7	India	1196.8	rupee
Dhaka	Bangladesh	14.8	17.0	Bangladesh	152.6	rupee
Mexico City	Mexico	19.5	20.2	Mexico	117.4	peso
New York	USA	19.4	20.0	USA	310.1	dollar
Sao Paulo	Brazil	19.6	20.1	Brazil	195.2	real
Shanghai	China	15.8	17.2	China	1379.7	yuan
Tokyo	Japan	36.1	36.4	Japan	129.0	yen

The query that creates the join above begins as follows:

```
Dim query = From city In MegacitiesDataSet.Cities
             Join country In MegacitiesDataSet.Countries
             On city.country Equals country.name
```

The From clause is standard. The Join clause says that the Countries table should be joined with the Cities table. The On clause identifies the two fields whose values are matched in order to join the tables. The variables *city* and *country* in the first two clauses are looping variables and can have any names we choose. For instance, the query above could have been written

```
Dim query = From town In MegacitiesDataSet.Cities
             Join nation In MegacitiesDataSet.Countries
             On town.country Equals nation.name
```



#### Example 5

The following program displays Table 10.3, the join of the two tables from the Megacities database.

```
Private Sub frmCities_Load(...) Handles MyBase.Load
    Me.CountriesTableAdapter.Fill(Me.MegacitiesDataSet.Countries)
    Me.CitiesTableAdapter.Fill(Me.MegacitiesDataSet.Cities)
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query = From city In MegacitiesDataSet.Cities
                Join country In MegacitiesDataSet.Countries
                On city.country Equals country.name
                Let cityName = city.name
                Let cityPop2010 = FormatNumber(city.pop2010, 1)
                Let cityPop2015 = FormatNumber(city.pop2015, 1)
                Let countryName = country.name
                Let countryPop2010 = FormatNumber(country.pop2010, 1)
                Select cityName, city.country, cityPop2010, cityPop2015,
                    countryName, countryPop2010, country.monetaryUnit
```

```

dgvOutput.DataSource = query.ToList
dgvOutput.CurrentCell = Nothing
End Sub

```

[Run, and click on the button.]

The screenshot shows a window titled "Megacities" with a button labeled "Display the Join of the Two Tables". Below the button is a table with the following data:

cityName	country	cityPop2010	cityPop2015	countryName	countryPop2010	monetaryUnit
Bombay	India	20.1	22.0	India	1,196.8	rupee
Buenos Aires	Argentina	13.1	13.4	Argentina	41.9	peso
Calcutta	India	15.6	17.0	India	1,196.8	rupee
Delhi	India	17.0	18.7	India	1,196.8	rupee
Dhaka	Bangladesh	14.8	17.0	Bangladesh	152.6	raka
Mexico City	Mexico	19.5	20.2	Mexico	117.4	peso
New York	USA	19.4	20.0	USA	310.1	dollar
Sao Paulo	Brazil	19.6	20.1	Brazil	195.2	real
Shanghai	China	15.8	17.2	China	1,379.7	yuan
Tokyo	Japan	36.1	36.4	Japan	129.0	yen



### Example 6

The following program uses the join of the two tables in the Megacities database. When the form is loaded, the Currencies list box is filled with the monetary units from the Countries table in alphabetical order. When the user selects a currency, the cities that use that currency are displayed in the Cities list box in alphabetical order. **Note:** The Cities list box was filled with the Add method rather than the DataSource method, so that the word NONE could be displayed when no cities use the selected currency.

The screenshot shows a window titled "Megacities" with two list boxes. The left list box is labeled "Currencies" and contains the text "lstCurrencies". The right list box is labeled "Cities" and contains the text "lstCities".

OBJECT	PROPERTY	SETTING
frmCities	Text	Megacities
lblCurrencies	Text	Currencies
lstCurrencies		
lblCities	Text	Cities
lstCities		

```

Private Sub frmCities_Load(...) Handles MyBase.Load
    Me.CountriesTableAdapter.Fill(Me.MegacitiesDataSet.Countries)
    Me.CitiesTableAdapter.Fill(Me.MegacitiesDataSet.Cities)
    Dim query = From country In MegacitiesDataSet.Countries
                Order By country.monetaryUnit Ascending
                Select country.monetaryUnit
                Distinct
    lstCurrencies.DataSource = query.ToList
End Sub

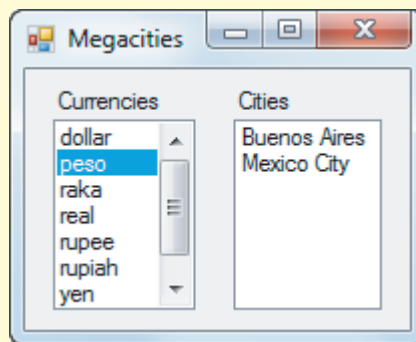
```

```

Private Sub lstCurrencies_SelectedIndexChanged(...) Handles _
    lstCurrencies.SelectedIndexChanged
    Dim query = From city In MegacitiesDataSet.Cities
                Join country In MegacitiesDataSet.Countries
                On city.country Equals country.name
                Where country.monetaryUnit = lstCurrencies.Text
                Order By city.name Ascending
                Select city.name
    lstCities.Items.Clear()
    If query.Count > 0 Then
        For Each city As String In query
            lstCities.Items.Add(city)
        Next
    Else
        lstCities.Items.Add("NONE")
    End If
End Sub

```

[Run, and click on a currency in the Currencies list box.]



### Comments

1. A database resides on a disk, and a DataSet resides in memory. A table adapter serves as a conduit to allow bidirectional data transfer between the two. A binding source is used to simplify attaching form controls to data sources.
2. The requirement that no record may have a null entry in a primary key and that entries for primary keys be unique is called the **Rule of Entity Integrity**.
3. The Join of two tables is a virtual construct. It exists only in memory.

### Practice Problems 10.1

1. Consider the query in the Load event procedure of Example 6. What would happen if the LINQ operator Distinct were omitted?
2. Consider the query in Example 5. Why can't the Select clause be written as follows?

```

Select city.name, city.country, cityPop2010, cityPop2015,
       country.name, countryPop2010, country.monetaryUnit

```

## EXERCISES 10.1

Figure 10.11 contains the outputs produced by the event procedures in Exercises 1 through 6.

cityName	monetaryUnit
Dhaka	raka
Buenos Aires	peso

(a)

cityName	country
Shanghai	China
Calcutta	India

(b)

name	country
Calcutta	India
Dhaka	Bangladesh

(c)

name	monetaryUnit
Indonesia	rupiah
USA	dollar

(d)

name	country
Dhaka	Bangladesh
Sao Paulo	Brazil

(e)

name	monetaryUnit
Japan	yen
China	yuan

(f)

FIGURE 10.11 Outputs for Exercises 1 through 6.

In Exercises 1 through 6, identify the DataGridView in Fig. 10.11 that is the output of the event procedure. Assume that each program has the same Load event procedure as Example 5.

1. Private Sub btnDisplay\_Click(...) Handles btnDisplay.Click  
Dim query = From city In MegacitiesDataSet.Cities  
Where city.country.StartsWith("B")  
Select city.name, city.country  
dgvOutput.DataSource = query.ToList  
dgvOutput.CurrentCell = Nothing  
End Sub
2. Private Sub btnDisplay\_Click(...) Handles btnDisplay.Click  
Dim query = From city In MegacitiesDataSet.Cities  
Where city.pop2015 = 17.0  
Select city.name, city.country  
dgvOutput.DataSource = query.ToList  
dgvOutput.CurrentCell = Nothing  
End Sub
3. Private Sub btnDisplay\_Click(...) Handles btnDisplay.Click  
Dim query = From country In MegacitiesDataSet.Countries  
Where (country.pop2010 > 250) And (country.pop2010 < 300)  
Select country.name, country.monetaryUnit  
dgvOutput.DataSource = query.ToList  
dgvOutput.CurrentCell = Nothing  
Sub
4. Private Sub btnDisplay\_Click(...) Handles btnDisplay.Click  
Dim query = From country In MegacitiesDataSet.Countries  
Where country.monetaryUnit.EndsWith("n")  
Order By country.pop2010 Ascending  
Select country.name, country.monetaryUnit  
dgvOutput.DataSource = query.ToList  
dgvOutput.CurrentCell = Nothing  
End Sub



5. Private Sub btnDisplay\_Click(...) Handles btnDisplay.Click
- ```

    Dim query = From city In MegacitiesDataSet.Cities
                Join country In MegacitiesDataSet.Countries
                On city.country Equals country.name
                Let cityName = city.name
                Where country.pop2010 > (75 * city.pop2010)
                Order By city.country Ascending
                Select cityName, city.country

    dgvOutput.DataSource = query.ToList
    dgvOutput.CurrentCell = Nothing
End Sub

```
6. Private Sub btnDisplay\_Click(...) Handles btnDisplay.Click
- ```

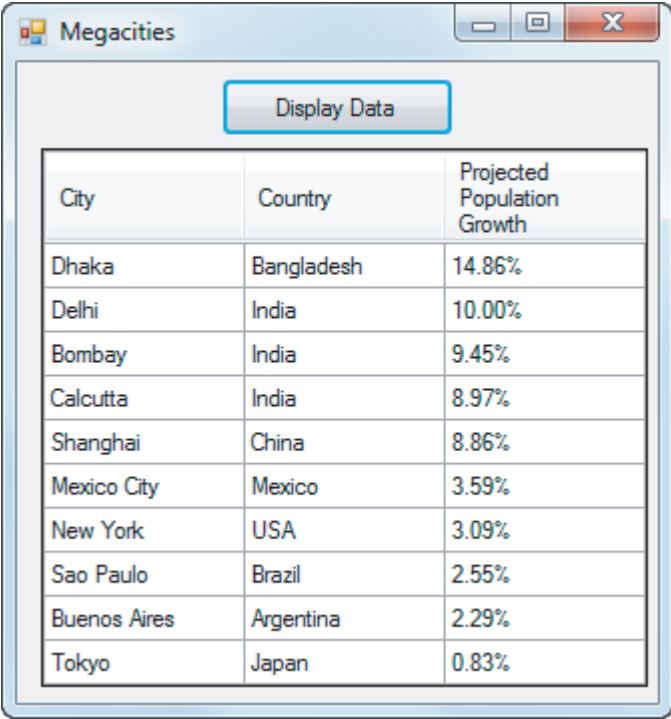
    Dim query = From city In MegacitiesDataSet.Cities
                Join country In MegacitiesDataSet.Countries
                On city.country Equals country.name
                Let cityName = city.name
                Where country.name.Length > 8
                Order By city.pop2010 Descending
                Select cityName, country.monetaryUnit

    dgvOutput.DataSource = query.ToList
    dgvOutput.CurrentCell = Nothing
End Sub

```

Exercises 7 through 16 require the Megacities database.

7. Write a program that displays (in a DataGridView control) the names of all the cities, their countries, and the projected percentage growth of their populations from 2010 to 2015. Records should be sorted in descending order by their projected population growth. See Fig. 10.12.



City	Country	Projected Population Growth
Dhaka	Bangladesh	14.86%
Delhi	India	10.00%
Bombay	India	9.45%
Calcutta	India	8.97%
Shanghai	China	8.86%
Mexico City	Mexico	3.59%
New York	USA	3.09%
Sao Paulo	Brazil	2.55%
Buenos Aires	Argentina	2.29%
Tokyo	Japan	0.83%

FIGURE 10.12 Output of Exercise 7.

8. Write a program that shows (in a list box) the names of all the countries from the Cities table. When the user clicks on one of the countries, the program should display (in a text box) the name of its most populous city in 2010.
9. Write a program that displays (in a list box) the names of the cities in the Cities table whose populations are projected to exceed 20 million by the year 2015. The cities should be ordered by their projected population in descending order.
10. Write a program that displays (in a DataGridView control) the names of the cities in the Cities table whose 2010 populations are between 13 and 19 million. The countries and their 2010 populations also should be displayed, and the records should be ordered alphabetically by the names of the countries.
11. Write a program to find and display (in a DataGridView control) the entire record for the city in the Cities table that will experience the greatest percentage growth from 2010 to 2015. **Note:** The percentage growth is  $(\text{pop2015} - \text{pop2010}) / \text{pop2010}$ .
12. Write a program that displays (in a DataGridView control) the names of all the cities, their countries, and for each city the percentage of its country's population that lived in that city in 2010. Records should be sorted in descending order by the percentages.
13. Write a program that displays the cities in a list box. When the user clicks on one of the cities, the program should display (in a text box) the percentage of its country's population that lived in that city in 2010.
14. Write a program that displays the cities in a list box. When the user clicks on one of the cities, the program should display (in a text box) the city's currency.
15. Write a program that creates a CSV text file containing the contents of the Cities table. Run the program, and compare the size of the text file with the size of the file Megacities.accdb.
16. Write a program that creates an XML file containing the contents of the Cities table.

The database UN.accdb has the single table Nations that contains data for the 192 member countries of the United Nations. The fields for the table are *name*, *continent*, *population*, and *area*. (Population is given in millions and area in square miles.) Use the United Nations database in Exercises 17 through 19. Some records in the table are

Canada	North America	32.9	3855000
France	Europe	63.5	211209
New Zealand	Australia/Oceania	4.18	103738
Nigeria	Africa	146.5	356669
Pakistan	Asia	164	310403
Peru	South America	27.9	496226

17. Write a program that displays the names of the continents from the Nations table in a list box. When the user clicks on a continent's name, the countries in that continent should be displayed in two other list boxes. One list box should display the countries in descending order by their population, and the other should display the countries in descending order by their area. See Fig. 10.13.
18. Write a program that displays the names of the continents from the Nations table in a list box. When the user clicks on a continent's name, the countries in that continent should be displayed (in a DataGridView) along with their population densities. The records should be in ascending order by their population densities.



FIGURE 10.13 Sample output of Exercise 17.

19. Write a program to find and display (in a DataGridView control) two entire records, where the first record gives the data for the country with the largest population, and the second the data for the country with the smallest population.

The database Exchrates.acddb has the single table Rates that gives the exchange rates (in terms of American dollars) for 45 currencies of major countries in December, 2009. Figure 10.14 shows the first eight records in the database in a DataGridView control. The dollarRate column gives the number of units of the currency that can be purchased for one American dollar. For instance, one American dollar purchases 1.05875 Canadian dollars. Use the Exchrates database in Exercises 20 through 22.

country	monetaryUnit	dollarRate
America	Dollar	1
Argentina	Peso	3.81072
Australia	Dollar	1.10248
Brazil	Real	1.76002
Canada	Dollar	1.05875
Chile	Peso	499.978
China	Yuan	6.83236
Colombia	Peso	1994.5

FIGURE 10.14 Exchange rates.

20. Write a program that shows the names of the countries in a list box. When the user clicks on one of the names, the monetary unit and the exchange rate should be displayed.
21. Write a program that displays the names of the countries in a list box in ascending order determined by the number of units that can be purchased by one American dollar. When the user clicks on one of the names, the monetary unit and exchange rate should be displayed.
22. Write a program containing two list boxes as shown in Fig. 10.15 on the next page. When the user selects two countries, enters an amount of money, and clicks on the button, the program should convert the amount from one currency to the other.

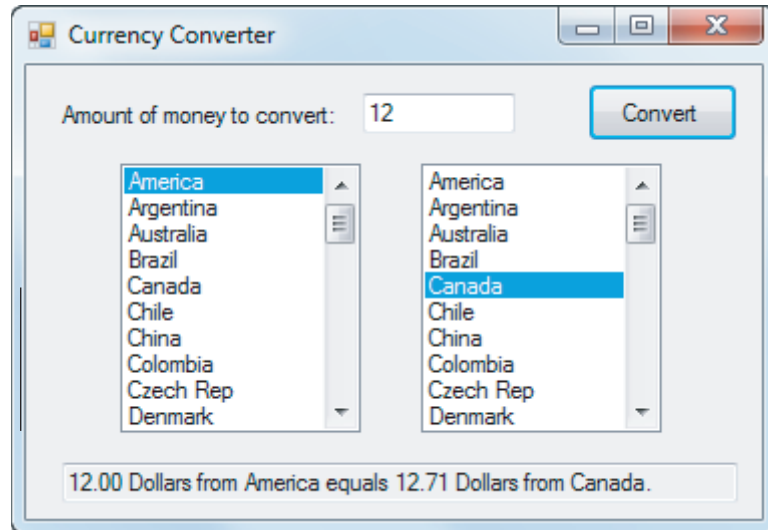


FIGURE 10.15 A possible output for Exercise 22.

The database *Baseball.accdb* has the two tables *Players* and *Teams*. The fields for the *Players* table are *name*, *team*, *atBats*, and *hits*. The fields for the *Teams* table are *name*, *location*, *league*, *stadium*, *atBats*, and *hits*. The database has been filled with information from the 2009 baseball season for the major league. The *Players* table lists all major-league players with at least 350 at bats during the season. The *Teams* table lists all major-league teams. Use the *Baseball* database in Exercises 23 through 38. Here are three sample records from each table:

Players

Aaron Hill	Blue Jays	682	195
Ichiro Suzuki	Mariners	639	225
Derek Jeter	Yankees	634	212

Teams

Cubs	Chicago	National	Wrigley Field	5486	1398
Nationals	Washington D.C.	National	National Park	5493	1432
Red Sox	Boston	American	Fenway Park	5543	1495

23. Write a program that shows all the teams from the *Teams* table in a list box. When the user clicks on one of the teams, the program should display the team's home stadium in a text box.
24. Write a program to display in a list box the player (or players) in the *Players* table with the highest batting average.
25. Write a program to display in a list box the player (or players) in the *Players* table with the most hits.
26. Write a program to display in a *DataGridView* control the names of all the teams, their home stadiums, and the teams' batting averages. Records should be sorted in ascending order by the batting averages.
27. Write a program that shows all the teams from the *Teams* table in a list box. When the user clicks on one of the teams, the program should display in a *DataGridView* control the names of all the players in the *Players* table from that team, along with their batting averages. The players should be listed in descending order of their batting averages. See Fig. 10.16.
28. Write a program that shows all the players' batting averages above .300 in a list box. When the user clicks on one of the batting averages, the program should display in a *DataGridView* control the names of all the players in the *Players* table with that batting average,

Name	Batting Average
Albert Pujols	0.327
Skip Schumaker	0.303
Yadier Molina	0.293
Brendan Ryan	0.292
Ryan Ludwick	0.265

FIGURE 10.16 Sample output of Exercise 27.

along with their teams. The players should be listed in alphabetical order of their last names.

29. Write a program that contains two radio buttons captioned *American League* and *National League*. When the user clicks on one of them, the program should display in a DataGridView control the names of all the teams in the Teams table from that league, along with their team batting averages. The teams should be listed in descending order of their batting averages.
30. Write a program that uses the Teams table to calculate the total number of hits by teams in the National League.
31. Write a program that uses the Teams table to calculate the overall batting average for players in the American League.
32. Write a program to count the number of players in the Players table who play for a New York team.
33. Write a program to count the number of players in the Players table who play for a National League team.
34. Write a program that contains two radio buttons captioned *American League* and *National League*. When the user clicks on one of the radio buttons, the program should display (in a DataGridView control) the names of all the players from the league who had more than 150 hits, along with their batting averages and home stadiums. The players should be listed in descending order of their batting averages.
35. Write a program that shows all the teams from the Teams table in a list box. When the user clicks on one of the teams, the program should display (in another list box) the names of all the players in the Players table from that team, whose batting average was greater than their team's batting average. The players should be listed in descending order of their batting averages.
36. Write a program to display (in a list box) the player (or players) in the American League with the most hits.
37. Write a program to display the player (or players) in the National League with the highest batting average.
38. Write a program that requests a batting average and a league (American or National) and then displays (in a list box) the names of all the players in the league whose batting average is greater than the given batting average. The players should be listed in descending order by the number of hits they had during the season. The program should not allow the given batting average to be greater than 1 or less than 0.

The database *Movies.accdb* has two tables named *Lines* and *Actors*. The *Lines* table contains famous lines from films that were spoken by the leading male actor. The first field of the table gives the famous line and the second field gives the film. Figure 10.17(a) shows the

first three records of the Lines table. The Actors table contains some names of films and their leading male actors. Figure 10.17(b) shows the first three records of the Actors table. The *film* field in the Lines table is a foreign key to the *film* field in the Actors table. Use the Movies database in Exercises 39 through 44.

famousLine	film	film	maleLead
Rosebud.	Citizen Kane	On the Waterfront	Marlon Brando
We'll always have Paris.	Casablanca	Sudden Impact	Clint Eastwood
I coulda been a contender.	On the Waterfront	Taxi Driver	Robert DeNiro

(a) Lines

(b) Actors

FIGURE 10.17 Some records from the tables in the Movies database.

39. What is the primary key in the Actors table?
40. What is the primary key in the Lines table?
41. Write a program that fills a list box with the names of the films in the Lines table. When the user clicks on the name of a film, the lead male actor in that film should be displayed in a text box. See Fig. 10.18.

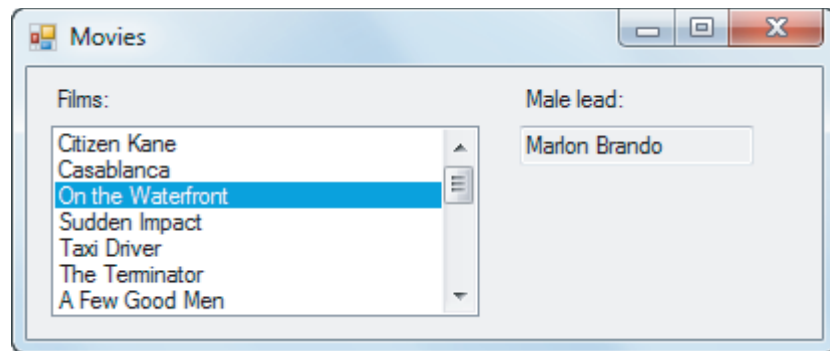


FIGURE 10.18 Sample outcome of Exercise 41.

42. Write a program that displays a DataGridView control containing all the famous lines from the Lines table along with the actors who spoke them and the films. See Fig. 10.19. **Note:** Set the AutoSizeColumnMode property of the DataGridView control to AllCells.

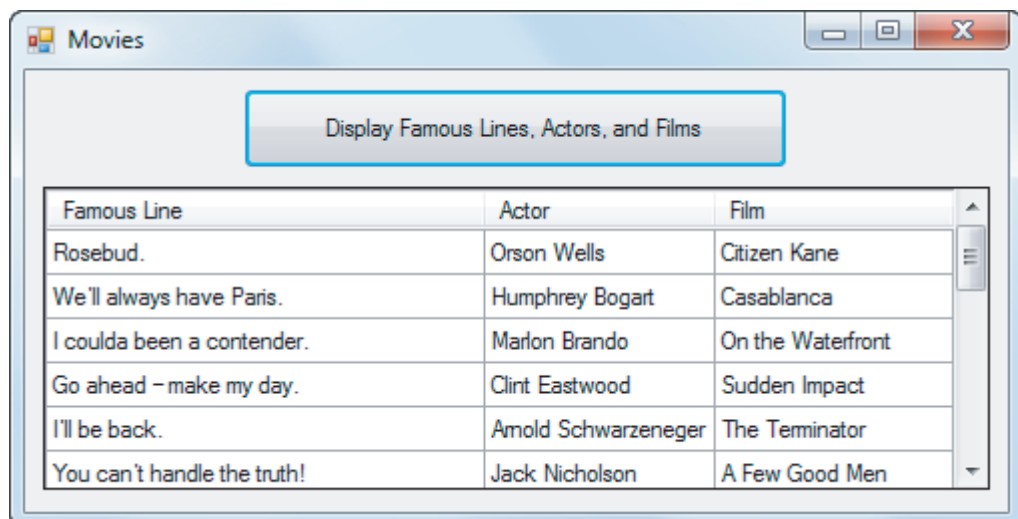


FIGURE 10.19 Outcome of Exercise 42.

43. Write a program that displays the names of the actors from the Actors table in a list box. When the user clicks on an actor's name, the famous lines from the Lines table that he spoke should be displayed in a second list box.
44. Write a program that displays the names of the films from the Actors table in a list box. When the user clicks on a film's name, the famous lines from the Lines table that were spoken in that film should be displayed in a second list box.

#### Solutions to Practice Problems 10.1

1. In the absence of the Distinct operator, both "peso" and "rupee" would appear twice in the list box.
2. The problem here is that the Select clause contains both *city.name* and *country.name*. The query would try to create two fields named *name*.

## 10.2 Editing and Designing Databases

In Section 10.1, we showed how to connect to a database and how to use LINQ queries to manipulate information retrieved from the database. In this section we learn how to alter records, delete records, and add new records to a database table. We end the section with a discussion of good database design.

### A Program to Edit the Cities Table

We begin by examining a program that edits the Cities table of the Megacities database; then we show how to create the program. The program, named 10-2-1, is in the folder Programs\Ch10 that you downloaded from the companion website for this book.

Figure 10.20 shows the form for the program. All of the controls on the form are familiar except for the navigation toolbar docked at the top of the form. Figure 10.21 shows the toolbar and identifies its components.



VideoNote  
Editing  
databases

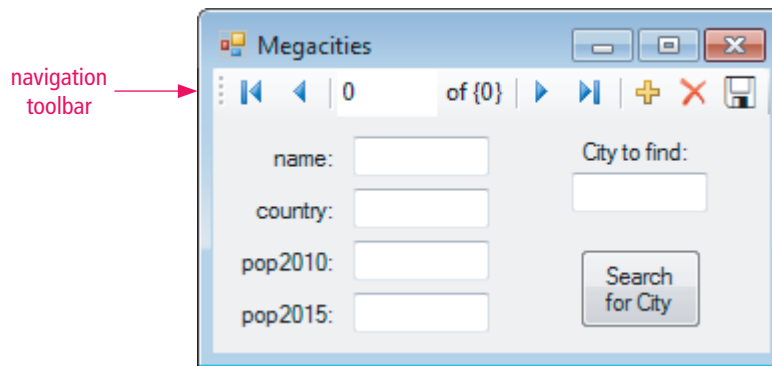


FIGURE 10.20 Form for the editing program.

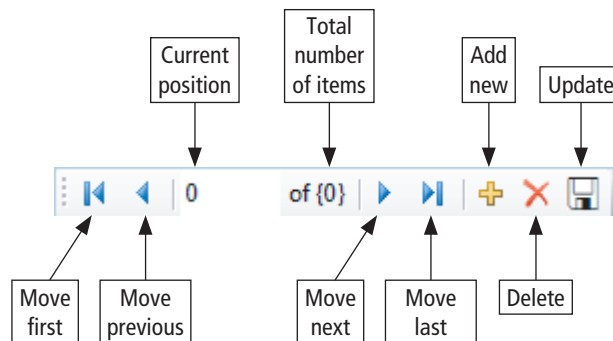


FIGURE 10.21 Navigation toolbar.