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

VideoNote
Editing databases

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.

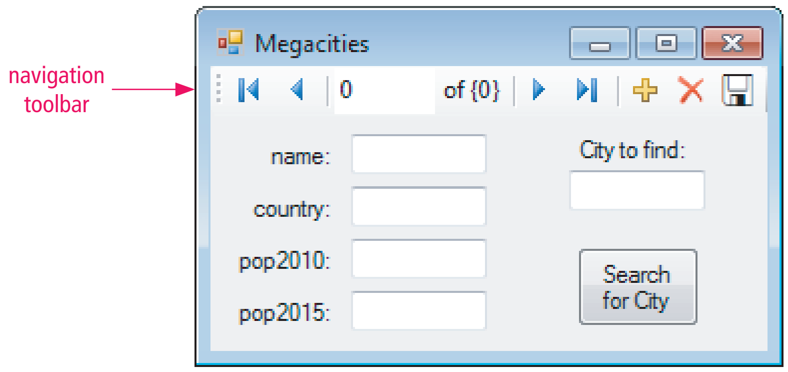


FIGURE 10.20 Form for the editing program.

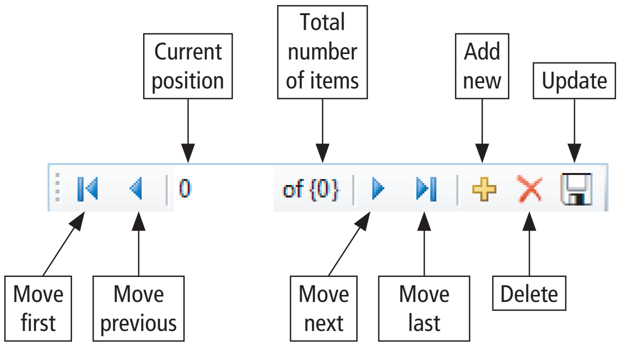
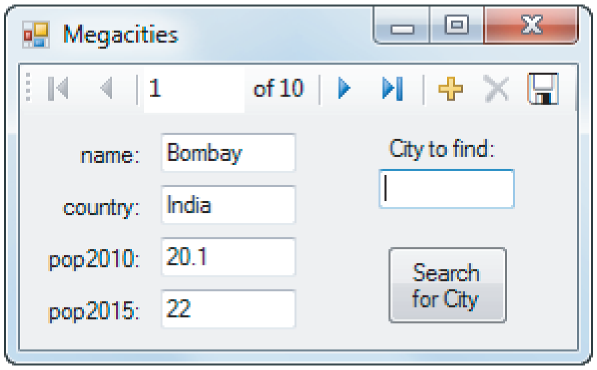


FIGURE 10.21 Navigation toolbar.

Figure 10.22 shows the form just after the program is run. The contents of the first record are shown in the four text boxes arranged vertically on the left side of the form. The *Move first* and *Move previous* buttons are disabled, since there are no records preceding the first record. When you click on the *Move next* button, the second record appears. In general, the four *Move* buttons allow you to navigate to any record of the Cities table. The first record of the table is said to have position 1, the second record position 2, and so on. If you place a number from 1 through 10 in the navigation toolbar's "Current position" box and then press the Enter key, the record having that position number will be displayed. For instance, if you enter 7 into the "Current position" box, the information for New York will be displayed. Another way to obtain a specific record is to type the name of a city into the text box labeled "City to find:" and then click on the *Search for City* button.



name: Bombay
country: India
pop2010: 20.1
pop2015: 22
City to find:

FIGURE 10.22 **Table-editing program at startup.**

To alter the data in the currently displayed record, just replace the value in one or more of the text boxes and then click on one of the *Move* buttons.

To add a new record to the Cities table, click on the *Add new* button ( ). The text boxes for the four fields will become blank. After you type in the data for the new record, and click on one of the *Move* buttons, the new record will be added and the number in the "Total number of items" box will change from 10 to 11.

To remove the currently displayed record from the Cities table, click on the *Delete* button ( ). The record will be deleted and the number in the "Total number of items" box will decrease by 1.

Each time the Cities table is altered in any way, only the copy of the table in memory is actually altered. To alter the database file on the disk, you must click on the *Update* button ( ). **Note 1:** The Solution Explorer actually contains two copies of the file Megacities.accdb. One copy is always displayed near the bottom of the Solution Explorer. A second copy is located in the *bin\Debug* folder. It is this second copy that can be updated by the program. **Note 2:** If the changes you made violate the Rule of Referential Integrity (every value in the foreign key must also appear in the primary key of the other table), the program will crash when the database file is updated. **Note 3**: Changes that violate the Rule of Entity Integrity (no record may have a null entry in its primary key and entries for primary keys must be unique) cause the program to crash as soon as a *Move* button is pressed. Exercises 10 through 12 discuss techniques for addressing the issues in Notes 2 and 3.

The following walkthrough will better acquaint you with the operation of the program.

1. Open and run the program named 10-2-1.
2. Press the *Move next* button in the navigation toolbar twice to reveal the third record, the one for Calcutta.
3. Type "New York" into the "City to find:" text box and then click on the *Search for City* button to display the record for New York.

4. Click on the *Add new* button in the navigation toolbar and then enter the following data into the vertical set of four empty text boxes for the fields: Los Angeles, USA, 12.8, 13.2.
5. Click on one of the *Move* buttons in the navigation toolbar. Notice that the "Total number of items" box now shows that there are 11 records.
6. Terminate and then rerun the program. Notice that the "Total number of items" box shows that there are 10 records. The Megacities database file in the *bin**Debug* folder was not updated because we did not click on the *Update* button before terminating the program.
7. Redo steps 4 and 5, and then click on the *Update* button.
8. Terminate and then rerun the program. Notice that the "Total number of items" box shows that there are 11 records. The Megacities database file in the *bin**Debug* folder was updated.
9. Experiment further with the program to test its other features.

Designing the Form for the Table-Editing Program

The following steps create the navigation toolbar for program 10-2-1, create the four sets of labels and text boxes on the left side of the form, bind the text boxes to the navigation toolbar, and implement the search capability.

1. Start a new program and bind the Cities table as was done in Section 10.1.
2. Add a BindingNavigator control to the form designer. (This 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 BindingNavigator1 appears in the component tray at the bottom of the Form Designer. Also, a navigation toolbar will appear anchored to the top of the form.
3. Go to the Properties window for BindingNavigator1, click on the down-arrow at the right side of the BindingSource property's Settings box, and set the BindingSource property to BindingSource1.
4. Notice that the toolbar is missing the *Update* button. The rightmost item on the toolbar (see Fig. 10.23) is used to add an update button to the navigation toolbar. Click on the small down-arrow on the right side of the rightmost item and then click on *Button* in the drop-down list that appears. A button showing a mountain (🖼) appears.

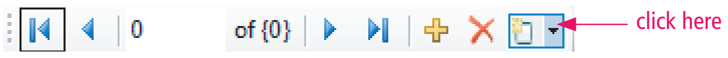


FIGURE 10.23 Original navigation toolbar.

5. Open the Properties window for the mountain button, change its name from ToolStripButton1 to btnUpdate, and set its Text property to Update.
6. Click on the ellipses at the right side of the Image property's Settings box, and then click on the *Import* button in the Select Resource dialogue box that appears. (An Open dialog box appears.)
7. Browse to locate the folder Programs\Ch10, and double-click on the file Disk.bmp to place its picture in the Select Resource window.
8. Click on the *OK* button to change the image on btnUpdate to a diskette (💾).
9. Click on the *Data* menu in the Visual Basic Toolbar, and then click on *Show Data Sources*. A Data Sources window showing the names of the tables in the database will appear in the location occupied by the Solution Explorer window.
10. Double-click on *Cities* in the Data Sources window to obtain the display in Fig. 10.24.

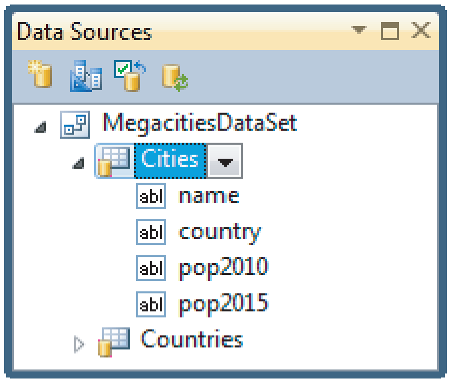


FIGURE 10.24 Data Sources window.

11. Click on *name* and drag it onto the form. Both a label and text box for the field *name* appear on the form. The label will be named NameLabel and the text box will be named NameTextBox.
12. Repeat Step 11 for each of the other three fields. The left side of the form has now been created and the four text boxes have been bound to the navigation toolbar.
13. Create the remaining controls in Fig 10.20 in the usual way.

Writing the Table-Editing Program

We have seen that each record of the Cities table is given a position number from 1 through 10 by the navigation toolbar. For programming purposes, each record has an **index number** that ranges from 0 through 9. That is, the record for Bombay has index number 0, the record for Buenos Aires has index number 1, and so on. BindingSource controls have a Position property and a Find method that return index numbers.

Assume the program is running. Let's refer to the record whose fields are displayed as the **current record**. At any time, the value of

```
BindingSource1.Position
```

is the index of the current record. Pay particular attention to the fact that the Position property gives the index, not the position!

If *n* is a nonnegative integer, then the statement

```
BindingSource1.Position = n
```

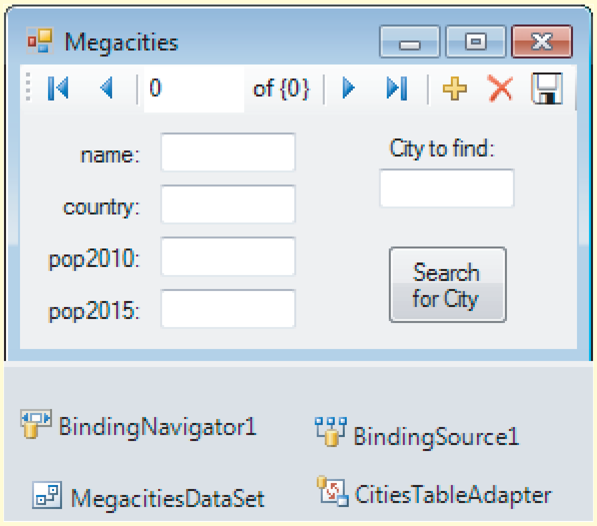
makes the record of index *n* the current record.

If *strVar* is a string variable whose value is a city in the Cities table, then the value of

```
BindingSource1.Find("name", strVar)
```

is the index of the record for that city. If the value of *strVar* is not a city in the table, then the Find method returns the value 0. ***Note:*** The Find method returns the same value, 0, both for Bombay and for a city not present in the Cities table.

Example 1 The following program edits the Cities table. Assume that the Cities table has been bound, and the navigation toolbar, text boxes, and *Search for City* button have been created as described above. The second statement inside the Load event procedure is optional. It prevents the city Bombay from being highlighted when the program is first run. In the btnSearch_Click event procedure, the two statements that refer to *currentCity* are optional. If they are omitted, the program will display the record for the first city (in this case Bombay) after an unsuccessful search. The remaining statements are self-explanatory.



OBJECT	PROPERTY	SETTING
frmCities	Text	Megacities
NameLabel	Text	name:
NameTextBox		
CountryLabel	Text	country:
CountryTextBox		
Pop2010Label	Text	pop2010:
Pop2010TextBox		
Pop2015Label	Text	pop2015:
Pop2015TextBox		
lblCity	Text	City to find:
txtCity		
btnSearch	Text	Search for City

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

Private Sub btnUpdate_Click(...) Handles btnUpdate.Click
  'These two lines update the database file in the bin\Debug folder.
  BindingSource1.EndEdit()
  CitiesTableAdapter.Update(MegacitiesDataSet.Cities)
End Sub

Private Sub btnSearch_Click(...) Handles btnSearch.Click
  Dim currentCity As String = NameTextBox.Text
  If txtCity.Text <> "" Then
    BindingSource1.Position = BindingSource1.Find("name", txtCity.Text)
    If NameTextBox.Text <> txtCity.Text Then
      MessageBox.Show("City not found.")
      BindingSource1.Position = BindingSource1.Find("name", currentCity)
    End If
  Else
    MessageBox.Show("You must enter the name of a city.")
  End If
End Sub
```

Principles of Database Design

Good relational database design is more of an art than a science. However, the designer should keep in mind certain fundamental guidelines.

- *Data should often be stored in their smallest parts*.

  For instance, city, state, and zip code are usually best stored in three fields. So doing will allow you to easily sort a mailing by zip code or target a mailing to the residents of a specific city.

- *Avoid redundancy*.

  The process of avoiding redundancy by splitting a table into two or more related tables is called **data normalization**. For instance, the excessive duplication in Table 10.4 can be avoided by replacing the table with the two related tables 10.5(a) and 10.5(b).

TABLE 10.4 A table with redundant data.

Course	Section	Name	Time	Credits	Prerequisites
CS102	1001	Intro to Databases	MWF 8-9	3	CS101
CS102	1002	Intro to Databases	MWF 1-2	3	CS101
CS102	1003	Intro to Databases	MWF 2-3	3	CS101
CS102	1004	Intro to Databases	MWF 3-4	3	CS101
CS105	1001	Visual Basic	MWF 1-2	4	CS200

TABLE 10.5(a)

Course	Section	Time
CS102	1001	MWF 8-9
CS102	1002	MWF 1-2
CS102	1003	MWF 2-3
CS102	1004	MWF 3-4
CS105	1001	MWF 1-2

TABLE 10.5(b)

Course	Name	Credits	Prerequisites
CS102	Intro to Databases	3	CS101
CS105	Visual Basic	4	CS200

- *Avoid tables with intentionally blank entries.*

  Tables with entries that are intentionally left blank use space inefficiently. Table 10.6, which serves as a directory of faculty and students, has an excessive number of blank entries. The table should be split into two tables, each dealing with just one of the groups.

TABLE 10.6 A table with an excessive number of blank entries.

Name	ssn	Classification	Date Hired	Dept	Office Number	gpa	Credits Earned
Sarah Brown	816-34-9012	student				3.7	78
Pat Riley	409-22-1234	faculty	9/1/02	biology	Y-3014		
Joe Russo	690-32-1108	faculty	9/1/05	math	T-2008		
Juan Lopez	509-43-4110	student				3.2	42

- *Strive for table cohesion.*

  Each table should have a basic topic to which all the data in the table are connected.

- *Avoid fields whose values can be calculated from existing fields.*

  For instance, if a table has fields for both the population and area of countries, then there is no need to include a field for the population density.

Comments

1. Since the *name* field of the Cities table is the primary key, each record of the table must have a value assigned to its *name* field. However, the database does not require that values be assigned to the other three fields.
2. In Example 1, the four pairs of labels and text boxes on the left side of the form have the names given to them by Visual Basic when the fields were dragged from the Data Sources window to the Form Designer. You can change these names if you like. For instance, CountryTextBox can be changed to txtCountry. Also, the names of the controls BindingSource1 and BindingNavigator1 can be changed to more meaningful ones, such as CitiesBindingSource and CitiesBindingNavigator.

Practice Problems 10.2

1. Can a record for any city be added to the Cities table?
2. Can a record for any country be added to the Countries table?
3. Can any record in the Cities table be deleted?
4. Can any record in the Countries table be deleted?

EXERCISES 10.2

In Exercises 1 through 8, carry out the following tasks on the Cities table with the program from Example 1.

1. Change the name of the city Bombay to Mumbai.
2. Add the following record: Karachi, Pakistan, 13.1, 14.9.
3. Use the *Search for City* button to find the record for Mexico City.
4. Find the record for the city in position 8.
5. Delete the record for Delhi.
6. Add a record that will raise an exception when you try to Update the database file.
7. Add a record that will raise an exception as soon as you move to another record.
8. Add a record with some empty fields.
9. Consider Example 1. Let *strVar* be a string variable. Give a condition that will be True if the value of *strVar* is not the name of a city in the Cities table.
10. Consider Example 1. If you violate the Rule of Referential Integrity, an exception is generated when you click on the *Update* button. Revise the btnUpdate_Click event procedure so that the program will not crash when the principle is violated. ***Note:*** The name of the exception is OleDB.OleDbException.
11. Modify Example 1 so that a Rule of Referential Integrity violation is caught as soon as an improper country is entered into the country text box.
12. Modify Example 1 so that a Rule of Entity Integrity violation is caught as soon as the user leaves an empty NameTextBox or a NameTextBox whose entry duplicates a city from another record.

Exercises 13 through 18 refer to the database Movies.accdb discussed in the exercises for Section 10.1.

13. Write a program that can be used to make changes to the Lines table. (There is no need to include a Search capability.)
14. Write a program that can be used to make changes to the Actors table. (There is no need to include a Search capability.)
15. Use the program created in Exercise 13 to change the famous line "Here's looking at you kid." to the line "Play it Sam." Both lines are from the same film.
16. Use the program created in Exercise 14 to add the following record to the Actors table: Patton, George Scott.
17. Why can't the following record be added to the Lines table: "Houston, we have a problem.", Apollo 13?
18. Why can't the following record be deleted from the Actors table: Casablanca, Humphrey Bogart?

19. Eliminate the redundancy in Table 10.7.

TABLE 10.7 **A table with redundancies.**

name	address	city	state	stateCapital
R. Myers	3 Maple St.	Seattle	Washington	Olympia
T. Murphy	25 Main St.	Seattle	Washington	Olympia
L. Scott	14 Park Ave.	Baltimore	Maryland	Annapolis
B. Jones	106 5th St.	Seattle	Washington	Olympia
W. Smith	29 7th Ave.	Baltimore	Maryland	Annapolis
V. Miller	4 Flower Ave.	Chicago	Illinois	Springfield

20. Eliminate the redundancy in Table 10.8, a table of members of the U.S. House of Representatives. (***Note:*** The value of *numColleges* is the number of colleges in the representative's state.)

TABLE 10.8 **A table with redundancies.**

name	state	party	statePop	numColleges
J. Dingell	Michigan.	Democratic	10.2	97
E. Cantor	Virginia	Republican	7.8	83
J. Moran	Virginia	Democratic	7.8	83
J. Sarbanes	Maryland	Democratic	5.8	97
S. Hoyer	Maryland	Democratic	5.8	97
F. Wolf	Virginia	Republican	7.8	83

21. The database Justices.accdb in the Programs\Ch10\Databases folder was current as of January 1, 2010. If some justices have retired and new justices have been appointed since then, update the database.

Solutions to Practice Problems 10.2

1. No. The city must be from a country appearing in the Countries table.

2. Yes.

3. Yes.

4. No. Only the records for Pakistan and Indonesia can be deleted.

CHAPTER 10 SUMMARY

1. A *table* is a group of data items arranged in a rectangular array, with each row containing the same categories of information. Each row is called a *record*. Each category (column) is called a *field*. A *database* is a collection of one or more tables that are usually related.

2. A *BindingSource control* can be used to bind a table of a database to a program.

3. *Database Explorer* in VB Express and *Server Explorer* in Visual Studio can be used to view the table from a database that has been connected to a program.

4. *LINQ* can be used to set criteria for information retrieval from a table.

5. A sequence of records resulting from the execution of a LINQ query can be displayed in a *DataGridView control*.