Databases and LINQ

Now go, write it before them in a table, and note it in a book, that it may be for the time to come for ever and ever.
—Isaiah 30:8

It is a capital mistake to theorize before one has data.
—Arthur Conan Doyle

Objectives
In this chapter you’ll learn:

- The relational database model.
- To use LINQ to retrieve and manipulate data from a database.
- To add data sources to projects.
- To use the Object Relational Designer to create LINQ to SQL classes.
- To use the IDE’s drag-and-drop capabilities to display database tables in applications.
- To use data binding to move data seamlessly between GUI controls and databases.
- To create Master/Detail views that enable you to select a record and display its details.
Self-Review Exercises

18.1 Fill in the blanks in each of the following statements:

a) A table in a relational database consists of _______ and _______ in which values are stored.
ANS: rows, columns.

b) The _______ uniquely identifies each row in a relational database table.
ANS: primary key.

c) A relational database can be manipulated in LINQ to SQL via a(n) _______ object, which contains properties for accessing each table in the database.
ANS: DataContext.

d) The _______ control (presented in this chapter) displays data in rows and columns that correspond to the rows and columns of a data source.
ANS: DataGridView.

e) Merging data from multiple relational database tables is called _______ the data.
ANS: joining.

f) A(n) _______ is a column (or group of columns) in a relational database table that matches the primary-key column (or group of columns) in another table.
ANS: foreign key.

g) A(n) _______ object serves as an intermediary between a data source and its corresponding data-bound GUI control.
ANS: BindingSource.

h) The _______ property of a control specifies where it gets the data it displays.
ANS: DataSource.

i) The _______ clause declares a new temporary variable within a LINQ query.
ANS: Let.

18.2 State whether each of the following is true or false. If false, explain why.

a) Providing the same value for a foreign key in multiple rows causes the DBMS to report an error.
ANS: False. Multiple rows can have the same value for a foreign key. Providing the same value for the primary key in multiple rows causes the DBMS to report an error, because duplicate primary keys would prevent each row from being identified uniquely.

b) Providing a foreign-key value that does not appear as a primary-key value in another table is an error.
ANS: True.

c) The result of a query can be sorted in ascending or descending order.
ANS: True.

d) A BindingNavigator object can extract data from a database.
ANS: False. A BindingNavigator allows users to browse and manipulate data displayed by another GUI control. A DataContext can extract data from a database.

e) LINQ to SQL automatically saves changes made back to the database.
ANS: False. You must call the SubmitChanges method of the DataContext to save the changes made back to the database.

Exercises

18.3 (Display Authors Table Application Modification) Modify the DisplayTable application in Section 21.5 to contain a TextBox and a Button that allow the user to search for specific authors by last name. Include a Label to identify the TextBox. Using the techniques presented in Section 21.9, create a LINQ query that changes the DataSource property of AuthorBindingSource to contain only the specified authors.
Chapter 18  Databases and LINQ

ANS:

1 // Ex. 18.3 Solution: DisplayAuthorsTable.cs
2 // Displaying data from a database table in a DataGridView.
3 using System;
4 using System.Linq;
5 using System.Windows.Forms;
6
7 namespace DisplayTable
8 {
9     public partial class DisplayAuthorsTable : Form
10     {
11         // constructor
12         public DisplayAuthorsTable()
13         {
14             DisplayAuthorsTable();
15         } // end constructor
16
17         // LINQ to SQL data context
18         private BooksDataContext database = new BooksDataContext();
19
20         // load data from database into DataGridView
21         private void DisplayAuthorsTable_Load(object sender, EventArgs e)
22         {
23             // use LINQ to order the data for display
24             authorBindingSource.DataSource =
25             from author in database.Authors
26             orderby author.AuthorID
27             select author;
28         } // end method DisplayAuthorsTable_Load
29
30         // Click event handler for the Save Button in the
31         // BindingNavigator saves the changes made to the data
32         private void authorBindingNavigatorSaveItem_Click(object sender, EventArgs e)
33         {
34             Validate(); // validate input fields
35             authorBindingSource.EndEdit(); // indicate edits are complete
36             database.SubmitChanges(); // write changes to database file
37         } // end method authorBindingNavigatorSaveItem_Click
38
39         // displays only rows that have the specified last name
40         private void findButton_Click(object sender, EventArgs e)
41         {
42             // update DataSource to include only people
43             // with specified last name
44             authorBindingSource.DataSource =
45             from author in database.Authors
46             where author.LastName.StartsWith(findTextBox.Text)
47             orderby author.AuthorID
48             select author;
49         } // end method findButton_Click
50     } // end class DisplayAuthorsTable
51 } // end namespace DisplayTable
18.4 (Display Query Results Application Modification) Modify the Display Query Results application in Section 21.6 to contain a TextBox and a Button that allow the user to perform a search of the book titles in the Titles table of the Books database. Use a Label to identify the TextBox. When the user clicks the Button, the application should execute and display the result of a query that selects all the rows in which the search term entered by the user in the TextBox appears anywhere in the Title column. For example, if the user enters the search term “Visual,” the DataGridView should display the rows for *Simply Visual Basic 2008*, *Visual Basic 2008 How to Program*, *Visual C# 2008 How to Program* and *Visual C++ 2008 How to Program*. If the user enters “Simply,” the DataGridView should display only the row for *Simply Visual Basic 2008*. [Hint: Use the Contains method of the String class.]
// Ex. 18.4 Solution: TitleQueries.cs
// Displaying the result of a user-selected query in a DataGridView.
using System;
using System.Linq;
using System.Windows.Forms;

namespace DisplayQueryResult
{
    public partial class TitleQueries : Form
    {
        // constructor
        public TitleQueries()
        {
            InitializeComponent();
        } // end constructor

        // LINQ to SQL data context
        private BooksDataContext database = new BooksDataContext();

        // load data from database into DataGridView
        private void TitleQueries_Load(
            object sender, EventArgs e)
        {
            // set the ComboBox to show the default query that
            // selects all books from the Titles table
            queriesComboBox.SelectedIndex = 0;
        } // end class TitleQueries_Load

        // Click event handler for the Save Button in the
        // BindingNavigator saves the changes made to the data
        private void titleBindingNavigatorSaveItem_Click(
            object sender, EventArgs e)
        {
            Validate(); // validate input fields
            titleBindingSource.EndEdit(); // indicate edits are complete
            database.SubmitChanges(); // write changes to database file
            // when saving, return to "all titles" query
            queriesComboBox.SelectedIndex = 0;
        } // end method titleBindingNavigatorSaveItem_Click

        // loads data into titleBindingSource based on user-selected query
        private void queriesComboBox_SelectedIndexChanged(
            object sender, EventArg e)
        {
            // set the data displayed according to what is selected
            switch ( queriesComboBox.SelectedIndex )
            {
            }
        }
    }
}
case 0: // all titles
  // use LINQ to order the books by title
  titleBindingSource.DataSource =
    from title in database.Titles
    orderby title.Title1
    select title;
  break;
case 1: // titles with 2008 copyright
  // use LINQ to get titles with 2008 copyright and sort them by title
  titleBindingSource.DataSource =
    from title in database.Titles
    where title.Copyright == "2008"
    orderby title.Title1
    select title;
  break;
case 2: // titles ending with "How to Program"
  // use LINQ to get titles ending with "How to Program" and sort them by title
  titleBindingSource.DataSource =
    from title in database.Titles
    where title.Title1.EndsWith( "How to Program" )
    orderby title.Title1
    select title;
  break;
18.5 *(Baseball Database Application)* Build an application that executes a query against the Players table of the Baseball database included in the Databases folder with this chapter's examples. Display the table in a DataGridView, and add a TextBox and Button to allow the user to search for a specific player by last name. Use a Label to identify the TextBox. Clicking the Button should execute the appropriate query.

ANS:

```csharp
// Ex. 18.5 Solution: BaseballPlayersForm.cs
using System;

// Displays the Players table of the Baseball database in a DataGridView
// and allows the user to search for players by last name.
using System;
```
namespace BaseballPlayers
{
    public partial class BaseballPlayersForm : Form
    {
        // constructor
        public BaseballPlayersForm()
        {
            InitializeComponent();
        } // end constructor

        // database connection
        private BaseballDataContext database = new BaseballDataContext();

        // load the data from the database when the Form loads
        private void BaseballPlayersForm_Load(object sender, EventArgs e)
        {
            // fill the DataGridView with the player information
            playerBindingSource.DataSource =
            from player in database.Players
            orderby player.PlayerID
            select player;
        } // end method BaseballPlayersForm_Load

        // Click event handler for the Save Button in the
        // BindingNavigator saves the changes made to the data
        private void playerBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            Validate(); // validate input fields
            playerBindingSource.EndEdit(); // indicate edits are complete
            database.SubmitChanges(); // write changes to database file
        } // end method playerBindingNavigatorSaveItem_Click

        // filter to display only players with the selected last name
        private void findButton_Click(object sender, EventArgs e)
        {
            // include only players that have the last name in findTextBox
            playerBindingSource.DataSource =
            from player in database.Players
            where player.LastName.StartsWith(findTextBox.Text)
            orderby player.PlayerID
            select player;
        } // end method findButton_Click
    } // end class BaseballPlayersForm
} // end namespace BaseballPlayers
18.6 (Baseball Database Application Modification) Modify Exercise 18.5 to allow the user to locate players with batting averages in a specific range. Add a minimumTextBox for the minimum batting average (0.000 by default) and a maximumTextBox for the maximum batting average (1.000 by default). Use a Label to identify each TextBox. Add a Button for executing a query that selects rows from the Players table in which the BattingAverage column is greater than or equal to the specified minimum value and less than or equal to the specified maximum value.

ANS:

```csharp
using System;
using System.Linq;

// Ex. 18.6 Solution: BaseballPlayersForm.cs
// Displays the Players table of the Baseball database in a DataGridView
// and allows the user to search for players by name or batting average.
using System;
using System.Linq;
```
using System.Windows.Forms;

namespace BaseballPlayers
{
    public partial class BaseballPlayersForm : Form
    {
        // constructor
        public BaseballPlayersForm()
        {
            InitializeComponent();
        } // end constructor

        // database connection
        private BaseballDataContext database = new BaseballDataContext();

        // load the data from the database when the Form loads
        private void BaseballPlayersForm_Load(object sender, EventArgs e)
        {
            // fill the DataGridView with the player information
            playerBindingSource.DataSource =
                from player in database.Players
                orderby player.PlayerID
                select player;
        } // end method BaseballPlayersForm_Load

        // Click event handler for the Save Button in the
        // BindingNavigator saves the changes made to the data
        private void playerBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            Validate(); // validate input fields
            playerBindingSource.EndEdit(); // indicate edits are complete
            database.SubmitChanges(); // write changes to database file
        } // end method playerBindingNavigatorSaveItem_Click

        // filter to display only players with the selected last name
        private void findButton_Click(object sender, EventArgs e)
        {
            // include only players that have the last name in findTextBox
            playerBindingSource.DataSource =
                from player in database.Players
                where player.LastName.StartsWith(findTextBox.Text)
                orderby player.PlayerID
                select player;
        } // end method findButton_Click

        // filter to display only players with
        // batting averages in the specified range
        private void averageButton_Click(object sender, EventArgs e)
        {
            // convert strings in TextBoxes to decimals
            decimal minimum = Convert.ToDecimal(minimumTextBox.Text);
            decimal maximum = Convert.ToDecimal(maximumTextBox.Text);
        } // end method averageButton_Click
    } // end class
} // end namespace

18.7  (Project: AdventureWorks Sample Database) In this exercise, use Microsoft’s sample AdventureWorks database. There are several versions available, depending on what version of SQL Server you’re using and your operating system. We used the AdventureWorks LT version of the database—a smaller version with fewer tables and less data than the full version. The files for SQL Server 2008 can be downloaded from...
The installer allows you to select which version of the database to install.

Use the AdventureWorks database in an application that runs multiple queries on the database and displays the results. First, it should list customers and their addresses. As this is a large list, limit the number of results to ten. [Hint: Use LINQ’s Take clause at the end of the query to return a limited number of results. The Take clause consists of the Take operator, then an Integer specifying how many rows to take.] Second, if a category has subcategories, the output should show the category with its subcategories indented below it. The queries described here require the AdventureWorks tables Address, Customer, CustomerAddress and ProductCategory.

ANS:

```csharp
using System;
using System.Linq;

namespace AdventureWorks
{
    public class AdventureWorks
    {
        public static void Main(string[] args)
        {
            // create the database connection
            AdventureWorksDataContext database = new AdventureWorksDataContext();

            // use LINQ to retrieve first 10 customer-address pairs
            var customersAndAddresses =
                from customerAndAddress in database.CustomerAddresses
                let customer = customerAndAddress.Customer
                let address = customerAndAddress.Address
                select new
                {
                    Name = customer.FirstName + " " + customer.LastName,
                    Address = string.Format("{0}, {1}, {2} {3}",
                        address.AddressLine1, address.City,
                        address.StateProvince, address.PostalCode )
                };

            // display customers and addresses
            foreach (var element in customersAndAddresses.Take(10))
            {
                Console.WriteLine("{0,-20} {1}",
                    element.Name, element.Address);
            } // end foreach

            Console.WriteLine(); // blank line for readability
        }
    }
}
```
// use LINQ to retrieve categories with subcategories
// include only categories with subcategories
var categories =
    from category in database.ProductCategories
    let children = category.ProductCategories
    where children.Any()
    select new
    {
        category.Name,
        Children = ( from child in children select child.Name )
    };

// display categories and their subcategories
foreach ( var category in categories )
{
    // display category name
    Console.WriteLine( category.Name + ":");

    foreach ( var child in category.Children )
    {
        // display subcategory name
        Console.WriteLine( "\t" + child );
    } // end inner foreach
} // end outer foreach
} // end namespace AdventureWorks

Orlando Gee          2251 Elliot Avenue, Seattle, Washington 98104
Keith Harris         7943 Walnut Ave, Renton, Washington 98055
Keith Harris         3207 S Grady Way, Renton, Washington 98055
Donna Carreras       12345 Sterling Avenue, Irving, Texas 75061
Janet Gates          800 Interchange Blvd., Austin, Texas 78701
Janet Gates          165 North Main, Austin, Texas 78701
Lucy Harrington      482505 Warm Springs Blvd., Fremont, California 94536
Rosmarie Carroll     39933 Mission Oaks Blvd, Camarillo, California 93010
Dominic Gash         5420 West 22500 South, Salt Lake City, Utah 84101
Kathleen Garza       6388 Lake City Way, Burnaby, British Columbia V5A 3A6

Bikes:
    Mountain Bikes
    Road Bikes
    Touring Bikes
Components:
    Handlebars
    Bottom Brackets
    Brakes
    Chains
    Cranksets
    Derailleurs
    Forks
    Headsets
    Mountain Frames
    Pedals
    Road Frames
Exercises 14

18.8 (Project: AdventureWorks Master/Detail view) Use the Microsoft AdventureWorks database from Exercise 18.7 to create a master/detail view. One master list should be customers, and the other should be products—these should show the details of products the customers purchased, and customers who purchased those products, respectively. Note that there are many customers in the database who did not order any products, and many products that no one ordered. Restrict the drop-down lists so that only customers that have submitted at least one order and products that have been included in at least one order are displayed. The queries in this exercise require the Customer, Product, SalesOrderHeader and SalesOrderDetail tables.

ANS:

```
// Ex. 18.8 Solution: AdventureWorksMasterDetailForm.cs
// A master/detail view using the AdventureWorks database.
using System;
using System.Linq;
using System.Windows.Forms;
namespace AdventureWorksMasterDetail
{
    public partial class AdventureWorksMasterDetailForm : Form
    {
        // constructor
        public AdventureWorksMasterDetailForm()
        {
            InitializeComponent();
        } // end constructor
    } // end namespace
```
Chapter 18  Databases and LINQ

// connection to the database
private AdventureWorksDataContext database =
    new AdventureWorksDataContext();

// this class allows us to show the
// customer's full name in the ComboBox
private class CustomerBinding
{
    public Customer Customer { get; set; } // contained customer
    public string Name { get; set; } // customer's full name
}

// initialize data sources when the form is loaded
private void AdventureWorksMasterDetailForm_Load(
    object sender, EventArgs e )
{
    customerComboBox.DisplayMember = "Name"; // display full name
    // set the customerComboBox's DataSource
    // to the list of customers
    customerComboBox.DataSource =
        from customer in database.Customers
        where customer.SalesOrderHeaders.Any()
        orderby customer.LastName, customer.FirstName
        let name = customer.FirstName + " " + customer.LastName
        select
            new CustomerBinding { Customer = customer, Name = name };;

    // display product's name
    productComboBox.DisplayMember = "Name";
    // set the productComboBox's DataSource to the list of products
    productComboBox.DataSource =
        from product in database.Products
        where product.SalesOrderDetails.Any()
        orderby product.Name
        select product;

    // initially, display no "detail" data
    adventureWorksBindingSource.DataSource = null;
    // set the DataSource of the DataGridView to the BindingSource
    adventureWorksDataGridView.DataSource =
        adventureWorksBindingSource;
} // end method AdventureWorksMasterDetailForm_Load

// display products that the customer purchased
private void customerComboBox_SelectedIndexChanged(
    object sender, EventArgs e )
{
    // get the selected Customer object from the ComboBox
    Customer currentCustomer =
        ( (CustomerBinding) customerComboBox.SelectedItem ).Customer;
Exercises

71  // traverse the order tables to get
72  // products the customer ordered
73  adventureWorksBindingSource.DataSource =
74     from header in currentCustomer.SalesOrderHeaders
75     from detail in header.SalesOrderDetails
76     select detail.Product;
77 } // end method customerComboBox_SelectedIndexChanged
78
79  // display customers that purchased a specific product
80  private void productComboBox_SelectedIndexChanged(
81      object sender, EventArgs e )
82  {
83  // get the selected Product object from the ComboBox
84      Product currentProduct =
85         ( Product ) productComboBox.SelectedItem;
86
87  // traverse order tables to get customers
88  // that ordered this product
89  adventureWorksBindingSource.DataSource =
90     from order in currentProduct.SalesOrderDetails
91     select order.SalesOrderHeader.Customer;
92 } // end method productComboBox_SelectedIndexChanged
93 } // end class AdventureWorksMasterDetailForm
94 } // end namespace AdventureWorksMasterDetail