14

Working with Databases

C# Programming: From Problem Analysis to Program Design
4th Edition

Chapter Objectives

- Be introduced to technologies used for accessing databases
- Become familiar with the ADO.NET classes
- Write program statements that use the DataReader class to retrieve database data
- Access and update databases using the DataSet and DataAdapter classes

Chapter Objectives (continued)

- Be introduced to SQL query statements
- Retrieve data using Language-Integrated Query (LINQ) expressions
- Use the visual development tools to connect to data sources, populate DataSet objects, build queries, and develop data-bound applications

Database Access

- As data needs increase, text files become less viable options
- Databases store information in records, fields, and tables
- Database - collection of records stored in a computer in a systematic way, so that a computer program can consult it to answer questions

Database Management System

- Database management system (DBMS): computer programs used to manage and query databases
- Example DBMSs include SQL server, Oracle, and Access
  - Many DBMSs store data in tabular format
    - Data in tables are related through common data field keys

Database Management Systems

- Typically use a query language to program database access
  - Structured query language (SQL)
- ActiveX Data Objects (ADO.NET): .NET data access technology for accessing data in databases

ADO.NET

- Includes number of classes that can be used to retrieve, manipulate, and update data in databases
- Can work with databases in a disconnect manner
  - Database table(s) can be retrieved to a temporary file
- To retrieve data, you must first connect to the database
- ADO.NET uses a feature called data providers to connect, execute commands, and retrieve results from a database

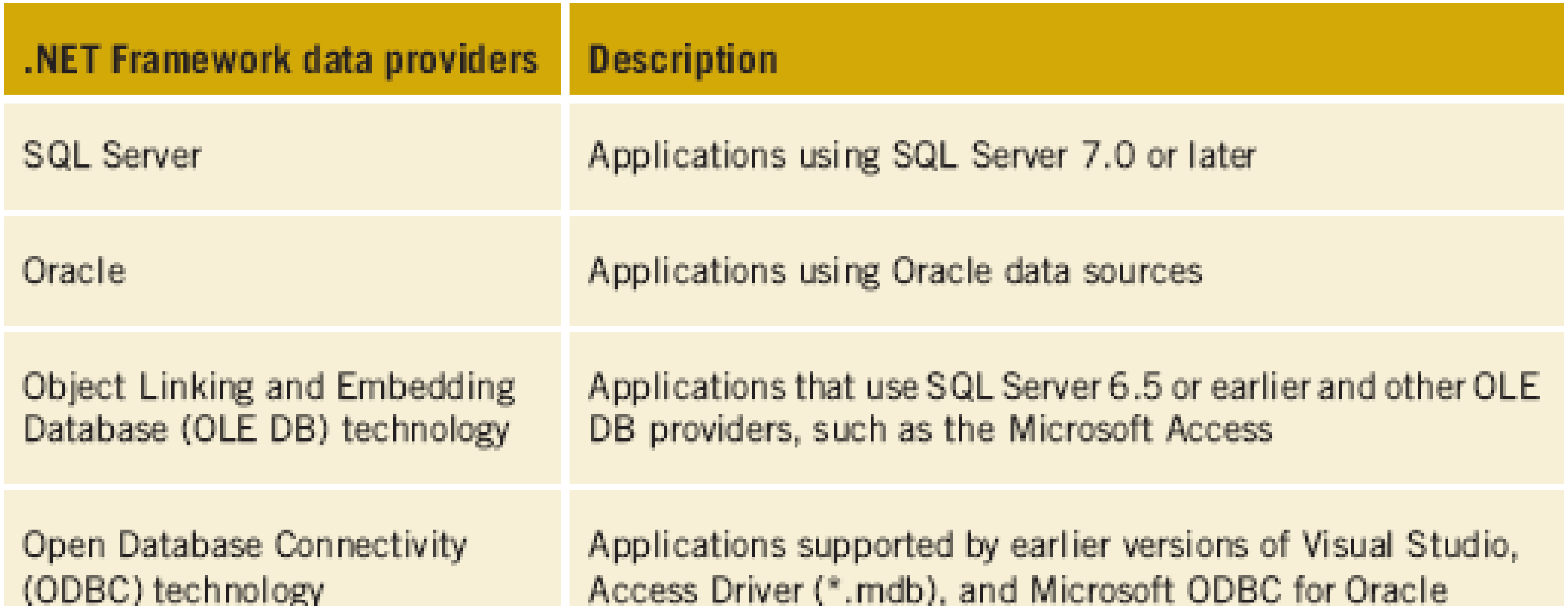
Data Providers

- ADO.NET architecture encapsulates the details of differing database structures
  - Providing common sets of functionality—connecting to a database, executing commands, and retrieving results
- Data provider is a set of classes that understands how to communicate with specific database management system

Data Providers (continued)

- Microsoft SQL Server
  - Applications using SQL Server 7.0 or later
- Oracle
  - Applications using Oracle data sources
- Object Linking and Embedding Database (OLE DB)
  - Applications that use Microsoft Access databases
- Open Database Connectivity (ODBC)
  - Applications supported by earlier versions of Visual Studio

Data Providers (continued)



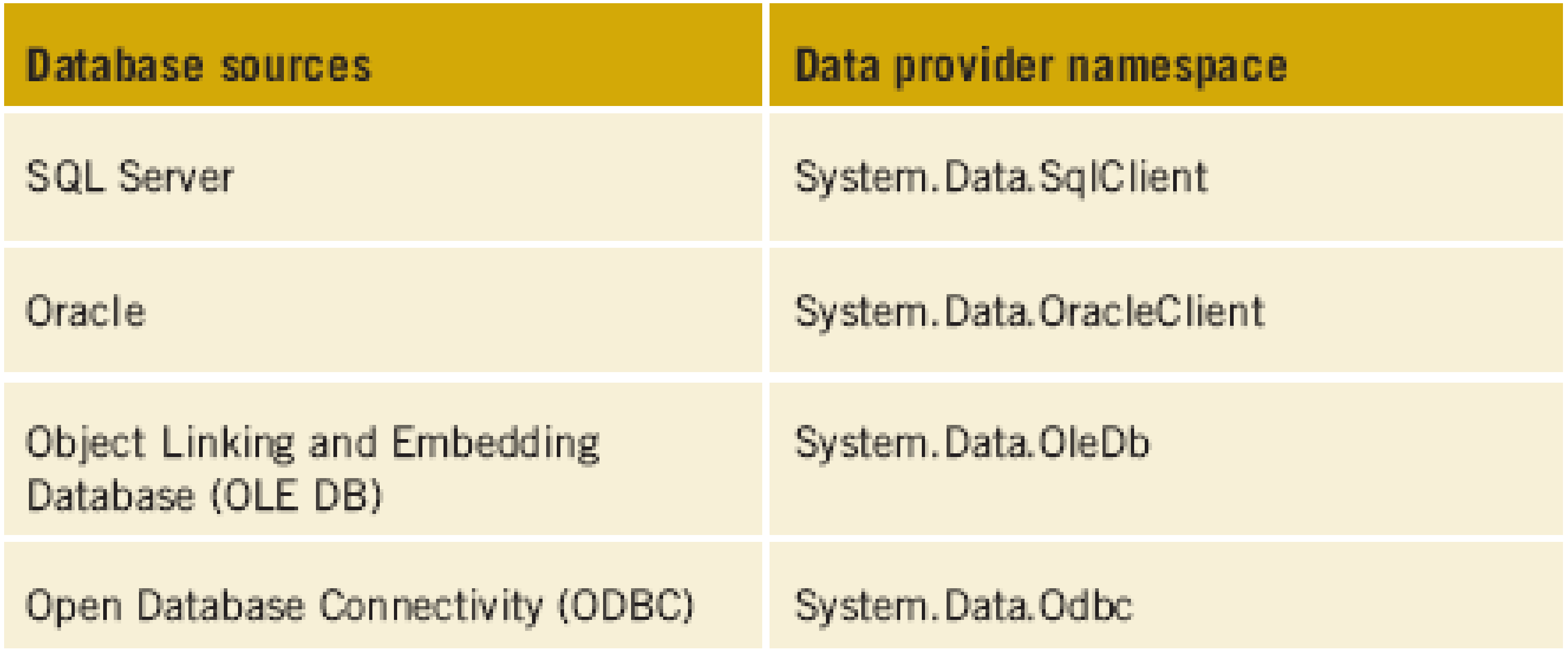
.NET Framework data providers	Description
SQL Server	Applications using SQL Server 7.0 or later
Oracle	Applications using Oracle data sources
Object Linking and Embedding Database (OLE DB) technology	Applications that use SQL Server 6.5 or earlier and other OLE DB providers, such as the Microsoft Access
Open Database Connectivity (ODBC) technology	Applications supported by earlier versions of Visual Studio, Access Driver (*.mdb), and Microsoft ODBC for Oracle

Table 14-1 ADO.NET data providers

Data Providers (continued)

- Classes are encapsulated into a different namespace by provider
- Four core classes make up each data provider namespace
  - Connection
  - Command
  - DataReader
  - DataAdapter

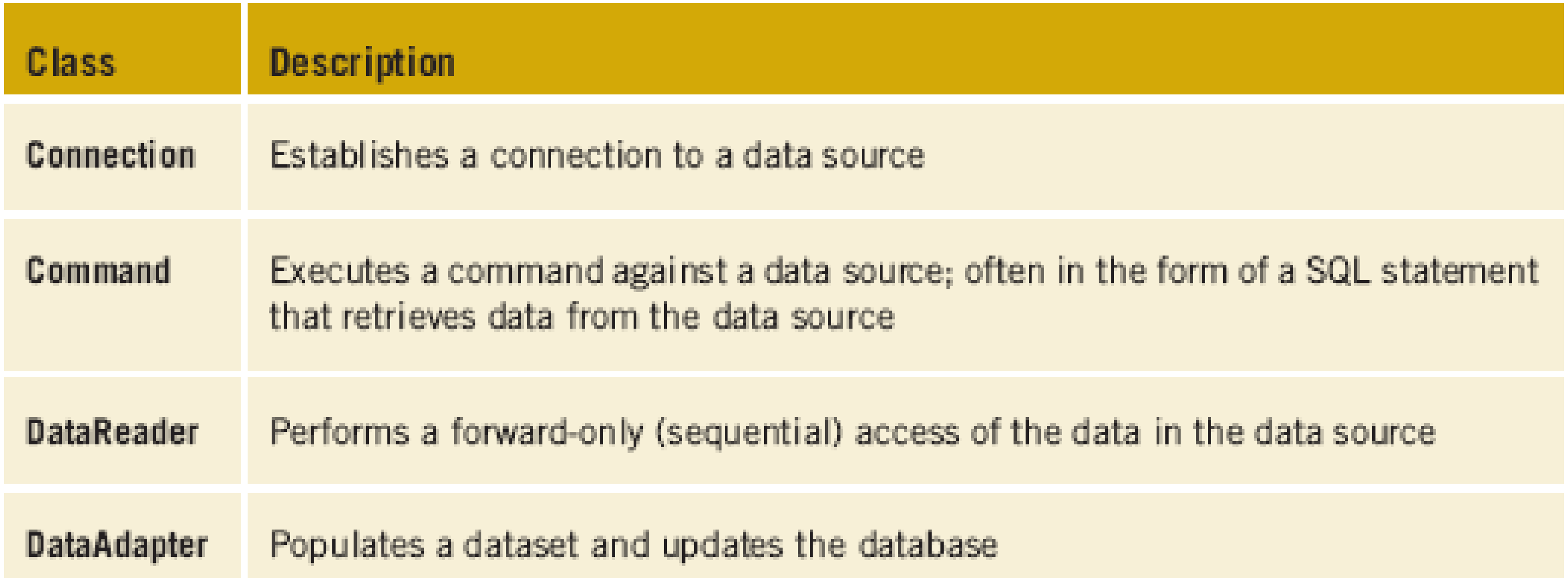
Data Providers (continued)



Database sources	Data provider namespace
SQL Server	System.Data.SqlClient
Oracle	System.Data.OracleClient
Object Linking and Embedding Database (OLE DB)	System.Data.OleDb
Open Database Connectivity (ODBC)	System.Data.Odbc

Table 14-1 ADO.NET data provider namespaces

Data Providers (continued)



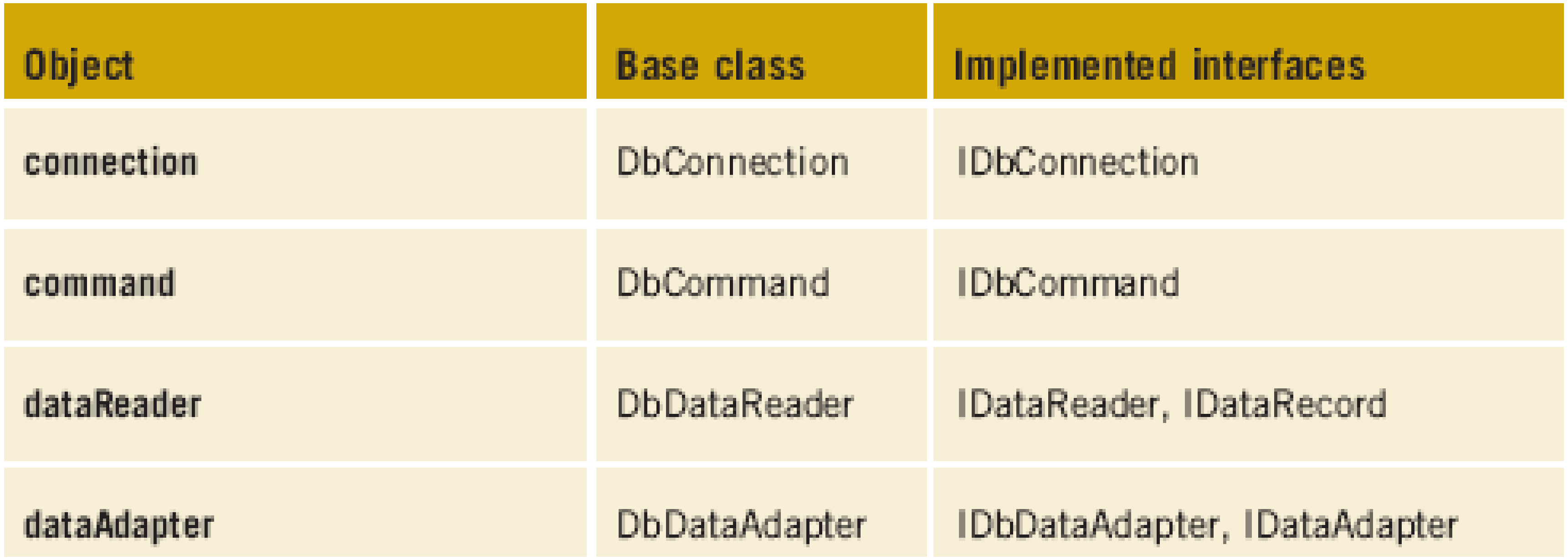
Class	Description
Connection	Establishes a connection to a data source
Command	Executes a command against a data source; often in the form of a SQL statement that retrieves data from the data source
DataReader	Performs a forward-only (sequential) access of the data in the data source
DataAdapter	Populates a dataset and updates the database

Table 14-3 Core classes that make up ADO.NET data providers

Data Providers (continued)

- Third-party vendors provide ADO.NET data providers for their vendor-specific databases
- Four core classes offer common functionality, primarily due to interfaces implemented by each of the core's base classes
  - Implement an interface means to sign a contract indicating it will supply definitions for all of the abstract methods declared in the interface
  - Each provider must provide implementation details for the methods that are exposed in the interface

Data Providers (continued)



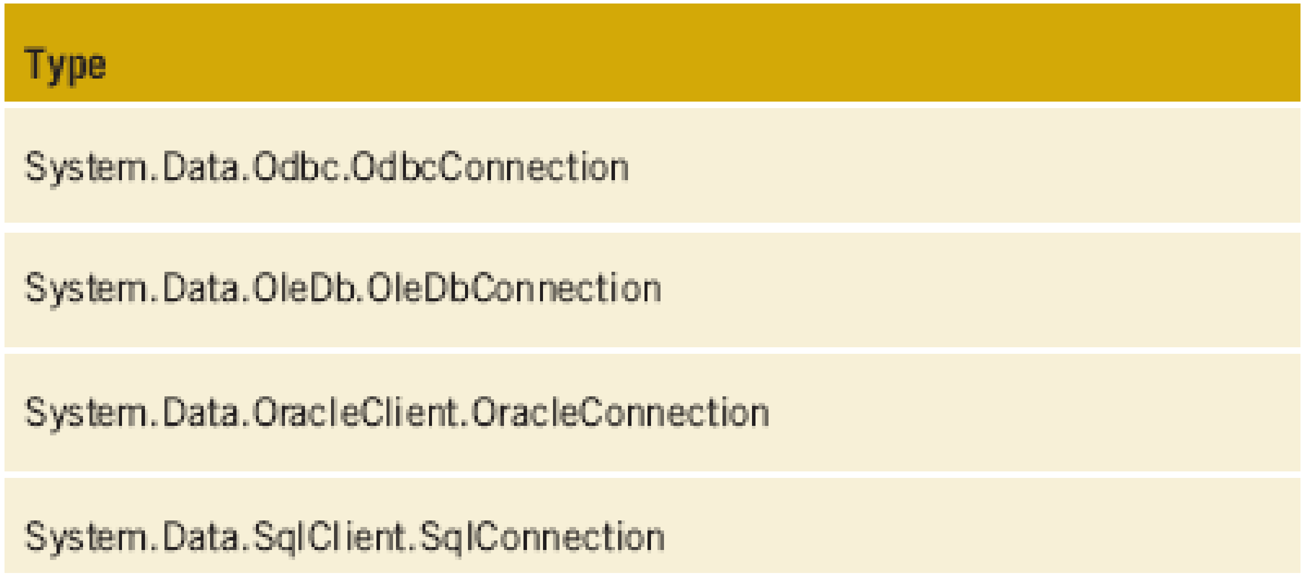
Object	Base class	Implemented interfaces
connection	DbConnection	IDbConnection
command	DbCommand	IDbCommand
dataReader	DbDataReader	IDataReader, IDataRecord
dataAdapter	DbDataAdapter	IDbDataAdapter, IDataAdapter

Table 14-4 Interfaces implemented by the Core ADO.NET objects

- Base classes shown in Table 14-4 are all abstract

Data Providers (continued)

Table 14-5 Derived classes of DbConnection



Type
System.Data.Odbc.OdbcConnection
System.Data.OleDb.OleDbConnection
System.Data.OracleClient.OracleConnection
System.Data.SqlClient.SqlConnection

- OdbcConnection must override and provide implementation details for Close( ), BeginDbTransaction( ), ChangeDatabase( ), CreateDbCommand( ), and OpenStateChange( )

Data Providers (continued)

- Additional namespaces used with ADO.NET classes to access databases include:
  - System.Data.Common
    - These classes are shared by all of the data providers
  - System.Data
    - These classes enables you to build components that use data from multiple data sources

Connecting to the Database (Microsoft Access DBMS)

- Add using directive

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

- Instantiate an object of connection class
  - Send connection string that includes the actual database provider and the data source (name of the database)

Enclose in try… catch block

```
string  sConnection;
sConnection = "Provider=Microsoft.ACE.OLEDB.12.0;" +
              "Data Source=member.accdb";
OleDbConnection  dbConn;
dbConn = new OleDbConnection(sConnection);
dbConn.Open();
```

Retrieving Data from the Database

- One way to retrieve records programmatically: issue an SQL query
- Object of OleDbCommand class used to hold SQL

```
string sql;
sql = "Select * From memberTable Order By LastName Asc, "
        + "FirstName Asc;";        // Note the two semicolons
OleDbCommand dbCmd = new OleDbCommand();
dbCmd.CommandText = sql;    // set command  SQL string
dbCmd.Connection = dbConn; // dbConn is connection object
```

SQL Queries

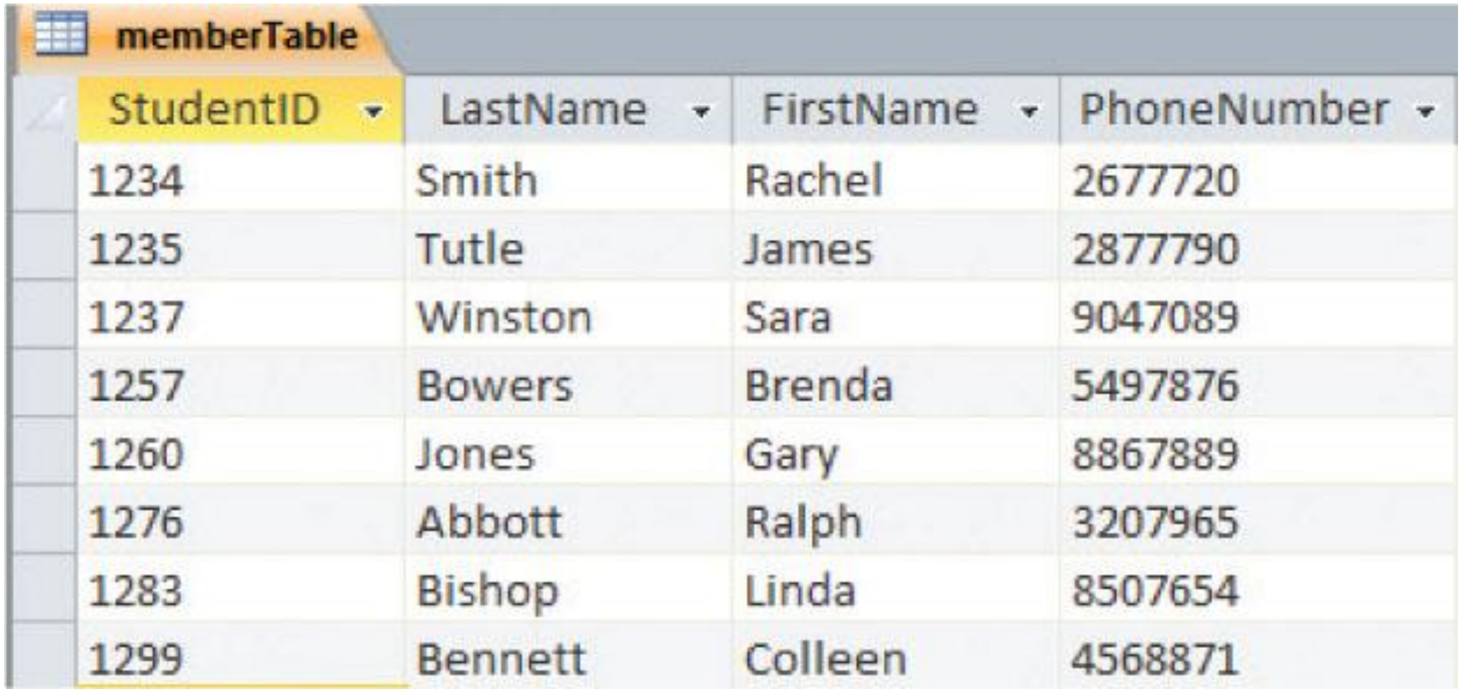
- SQL: universal language used with many database products including SQL Server and Microsoft Access
- Queries can be written to SELECT, INSERT, UPDATE, and DELETE data in database tables
- Can use the SELECT statement to retrieve results from multiple tables by joining them using a common field

SQL Queries (continued)

- Select * From memberTable Order By LastName Asc, FirstName Asc;
  - Asterisk (*) selects all fields (columns) in database
    - Can replace * by field name(s)
  - Asc (ascending) returns in ascending order by LastName; duplicate last names ordered by first name
  - Retrieves all rows (records)
    - Where clause can be added to selectively identify rows

Retrieving Data from the Database

```
Select StudentID, FirstName, LastName, PhoneNumber
From memberTable;
```



memberTable

StudentID	LastName	FirstName	PhoneNumber
1234	Smith	Rachel	2677720
1235	Tutle	James	2877790
1237	Winston	Sara	9047089
1257	Bowers	Brenda	5497876
1260	Jones	Gary	8867889
1276	Abbott	Ralph	3207965
1283	Bishop	Linda	8507654
1299	Bennett	Colleen	4568871

Figure 14-1 Access database table

Retrieving Data from the Database (continued)

- To retrieve a single row or just some of the rows from the table, you add a WHERE clause

```
SELECT PhoneNumber FROM memberTable
WHERE FirstName = 'Gary' AND LastName = 'Jones';
```

- If field has a space, the field name would have to be enclosed in square brackets

```
WHERE (aDate BETWEEN #10/12/2012# AND #10/12/2013#)—Access
WHERE (aDate BETWEEN '10/12/2012' AND '10/12/2013')—SQL Server
```

Retrieving Data from the Database (continued)

- Can use the SELECT statement to retrieve results from multiple tables by joining them using a common field

```
SELECT    memberTable.FirstName, memberTable.LastName,
                departmentTable.major_Name
FROM   memberTable INNER JOIN departmentTable ON
                memberTable.major_ID = departmentTable.major_ID;
```

Retrieving Data from the Database (continued)

- Selectively choose the columns
  - Primary key is column(s) that uniquely identifies row
  - Foreign key is column that refers to a column in another table (used to link the two tables)

```
INSERT INTO memberTable
          (StudentID, FirstName, LastName, PhoneNumber)
VALUES (1123, 'Kathy', 'Weizel', 2345678);

DELETE FROM memberTable WHERE (StudentID = 1299);

UPDATE memberTable  SET LastName = 'Hakim'
WHERE (StudentID = 1234);
```

Processing Data

- Can retrieve one record at a time in memory
  - Process that record before retrieving another
- OR can store the entire result of the query in temporary data structure similar to an array
  - Disconnect from the database
- ADO.NET includes data reader classes (by provider)
  - Used to read rows of data from a database

Retrieving Data Using a Data Reader

- OleDbDataReader and SqlDataReader class
  - READ-ONLY – Forward retrieval (sequential access)
  - Results returned as query executes
    - Sequentially loop through the query results
    - Only one row is stored in memory at a time
    - Useful to accessing data from large database tables
- Declare an object of the OleDbDataReader or and SqlDataReader class
- Call ExecuteReader( ) method

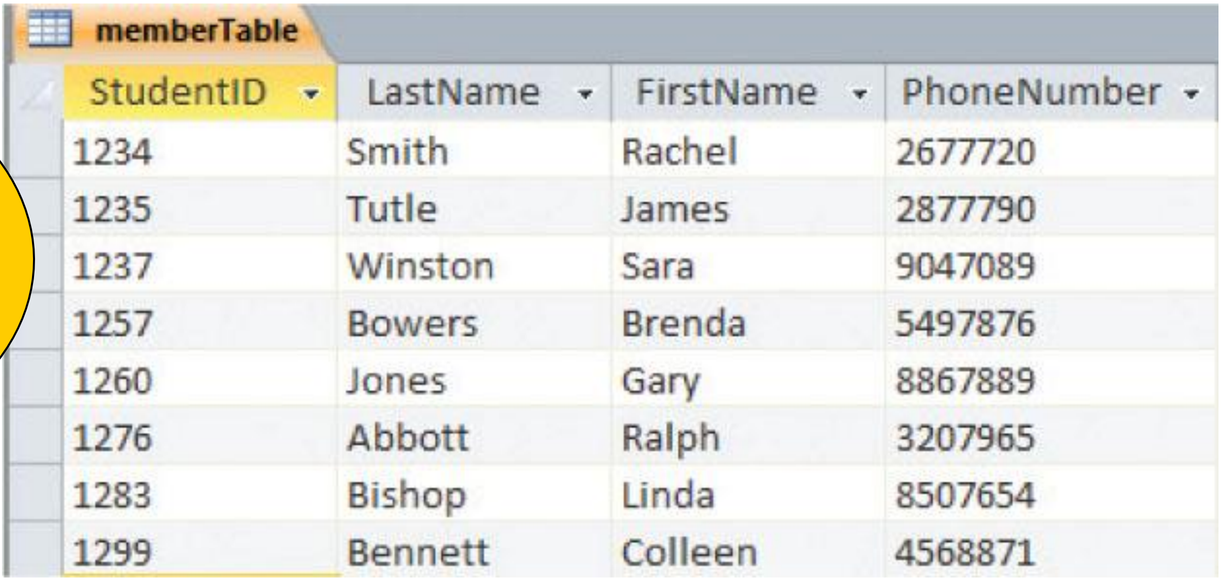
Retrieving Data Using a Data Reader (continued)

- To position the reader object onto the row of the first retrieved query result, use Read( ) method of the OleDbDataReader (or SqlDataReader) class
  - Read( ) also used to advance to the next record
  - Think about what is retrieved as one-dimensional table consisting of the fields from that one row
    - Fields can be referenced using actual ordinal index
    - Fields can also be referenced using the table's field names as indexers to the data reader object

Retrieving Data Using a Data Reader (continued)

- First call to dbReader.Read( ) retrieves first row
  - dbReader[0] refers to 1234
  - dbReader[1] refers to "Smith"
  - dbReader["FirstName"] also refers to "Rachel"

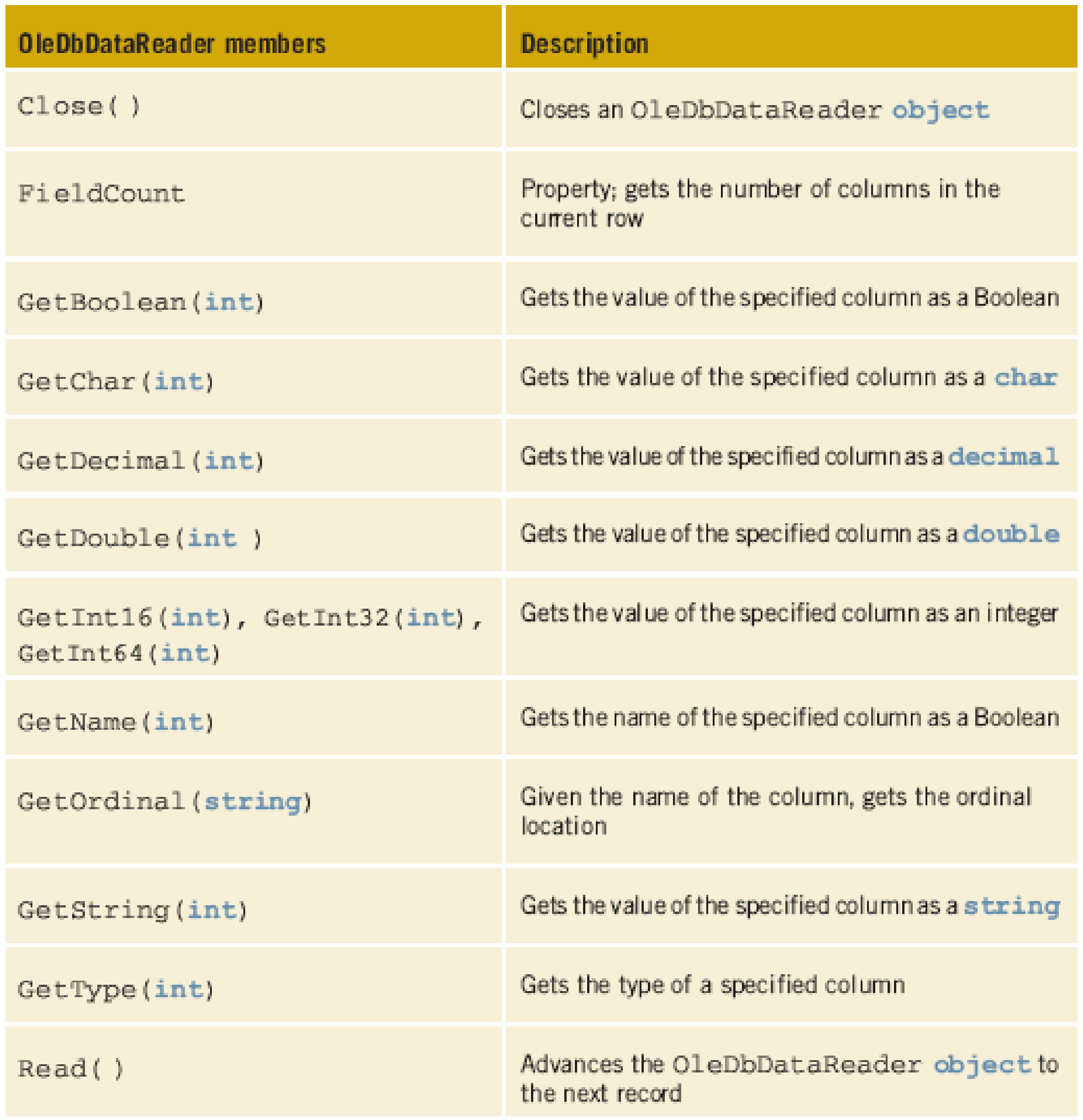
Field name must be enclosed in double quotes when used as indexer


memberTable

StudentID	LastName	FirstName	PhoneNumber
1234	Smith	Rachel	2677720
1235	Tutle	James	2877790
1237	Winston	Sara	9047089
1257	Bowers	Brenda	5497876
1260	Jones	Gary	8867889
1276	Abbott	Ralph	3207965
1283	Bishop	Linda	8507654
1299	Bennett	Colleen	4568871

Figure 14-1 Access database table

Retrieving Data Using a Data Reader (continued)



OleDbDataReader members	Description
`Close( )`	Closes an `OleDbDataReader` `object`
`FieldCount`	Property; gets the number of columns in the current row
`GetBoolean(int)`	Gets the value of the specified column as a Boolean
`GetChar(int)`	Gets the value of the specified column as a `char`
`GetDecimal(int)`	Gets the value of the specified column as a `decimal`
`GetDouble(int )`	Gets the value of the specified column as a `double`
`GetInt16(int), GetInt32(int), GetInt64(int)`	Gets the value of the specified column as an integer
`GetName(int)`	Gets the name of the specified column as a Boolean
`GetOrdinal(string)`	Given the name of the column, gets the ordinal location
`GetString(int)`	Gets the value of the specified column as a `string`
`GetType(int)`	Gets the type of a specified column
`Read( )`	Advances the `OleDbDataReader` `object` to the next record

Table 14-6 OleDbDataReader class members

Retrieving Data Using a Data Reader (continued)

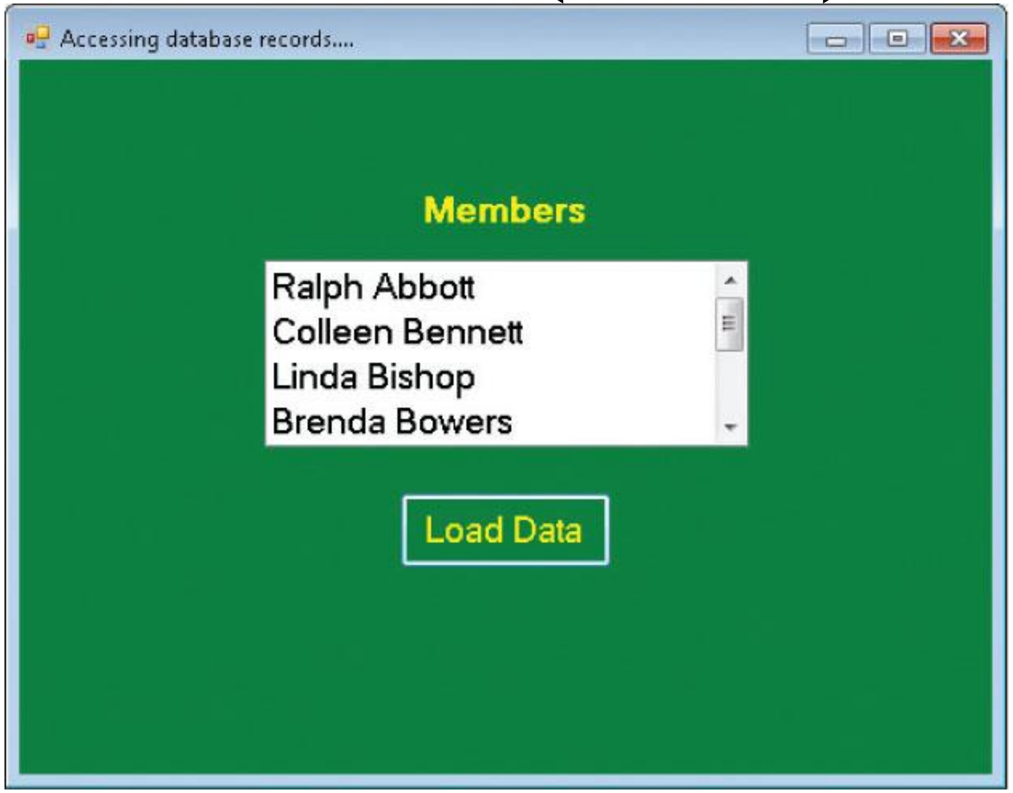
```
Member  aMember;
OleDbDataReader dbReader;
dbReader = dbCmd.ExecuteReader(  ); // dbCmd—OleDbCommand object
while (dbReader.Read(  ))
{    // retrieve records 1-by-1...
     aMember = new Member(dbReader["FirstName"].ToString( ),
               dbReader["LastName"].ToString( ));
     this.listBox1.Items.Add(aMember);
}
dbReader.Close( ); // Close the Reader object
dbConn.Close( ); // Close the Connection object
```

Review DBExample Example

Closing the Connection

- Close connections
  - Often overlooked
  - By doing this, you unlock the database so that other applications can access it
- Can enclose close connection in try. . .catch block
- using statement can be added around the entire block of code accessing the database
  - When added, no longer necessary to call the Close( ) methods

Retrieving Data Using a Data Reader (continued)



Ralph Abbott
Colleen Bennett
Linda Bishop
Brenda Bowers

Figure 14-2 Accessing member.accdb database using the database reader object

Updating Database Data

- Data Reader enables read-only access to database
- Several ways to change or update database
  - Can write Insert, Delete, and Update SQL statements and then execute those queries by calling OleDbCommand.ExecuteNonQuery( ) method
  - Can instantiate objects of dataset and data adapter classes
    - Use data adapter object to populate dataset object
      - Adapter class has Fill( ) and Update( ) methods

Updating Database Data (continued)

- Not required to keep a continuous live connection
  - Can create temporary copy in memory of the records retrieved using a dataset
- Interaction between dataset and actual database is controlled through data adapter
- Each of the different data providers has its own dataset and data adapter objects
  - System.Data.OleDb – Access database

Using Datasets to Process Database Records

- Instantiate a connection object using connection string
- Select records (and fields)

  by executing SQL SELECT
  - SQL statement is packaged in a data command object

See slide 19 – dbCmd set the SQL Select

- Instantiate object of Dataset class (for a table)

```
DataSet memberDS = new DataSet();
```

- Instantiate an object of DataAdapter class

```
OleDbDataAdapter memberDataAdap = new OleDbDataAdapter( );
```

Command Builder Class

- Class that automatically generates SQL for updates
  - Must set the SelectCommand property of the OleDbDataAdapter class

```
private OleDbCommandBuilder cBuilder;
:
cBuilder = new OleDbCommandBuilder(memberDataAdap);
memberDataAdap.SelectCommand = dbCmd;
```

- CommandBuilder object only used for datasets that map to a single database table

Filling the Dataset Using the Data Adapter

- After instantiating objects of data adapter, dataset, and command builder classes
- Using data adapter Fill( ) method to specify name of table to use as the data source

```
memberDataAdap.Fill(memberDS, "memberTable");
```

- To show contents of table, presentation user interface layer is needed
  - Grid control works well

Adding a DataGridView Control to Hold the Dataset

- Place DataGridView control object on Windows Form
  - Structure divided into rows and columns
  - Able to navigate around in data grid
  - Can make changes by editing current records
  - Can insert and delete new records

```
dataGridView1.DataSource = memberDS;
dataGridView1.DataMember = "memberTable";
```

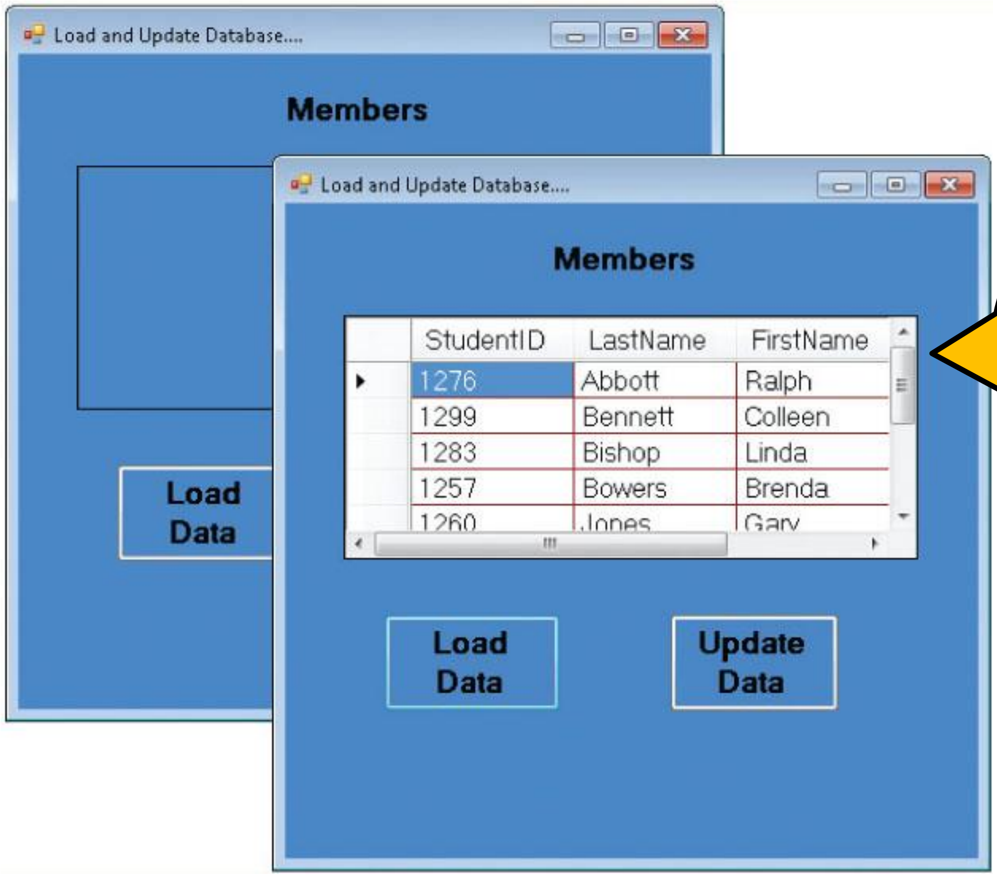
Updating the Database

- Additional SQL statements needed are automatically generated if you instantiate objects of command builder class
- Load the database into a DataGridView object and make changes
- Flush the changes back up to live database using the Update( ) method of DataAdapter class

```
memberDataAdap.Update(memberDS, "memberTable");
```

Review DataSetExample Example

Updating the Database (continued)

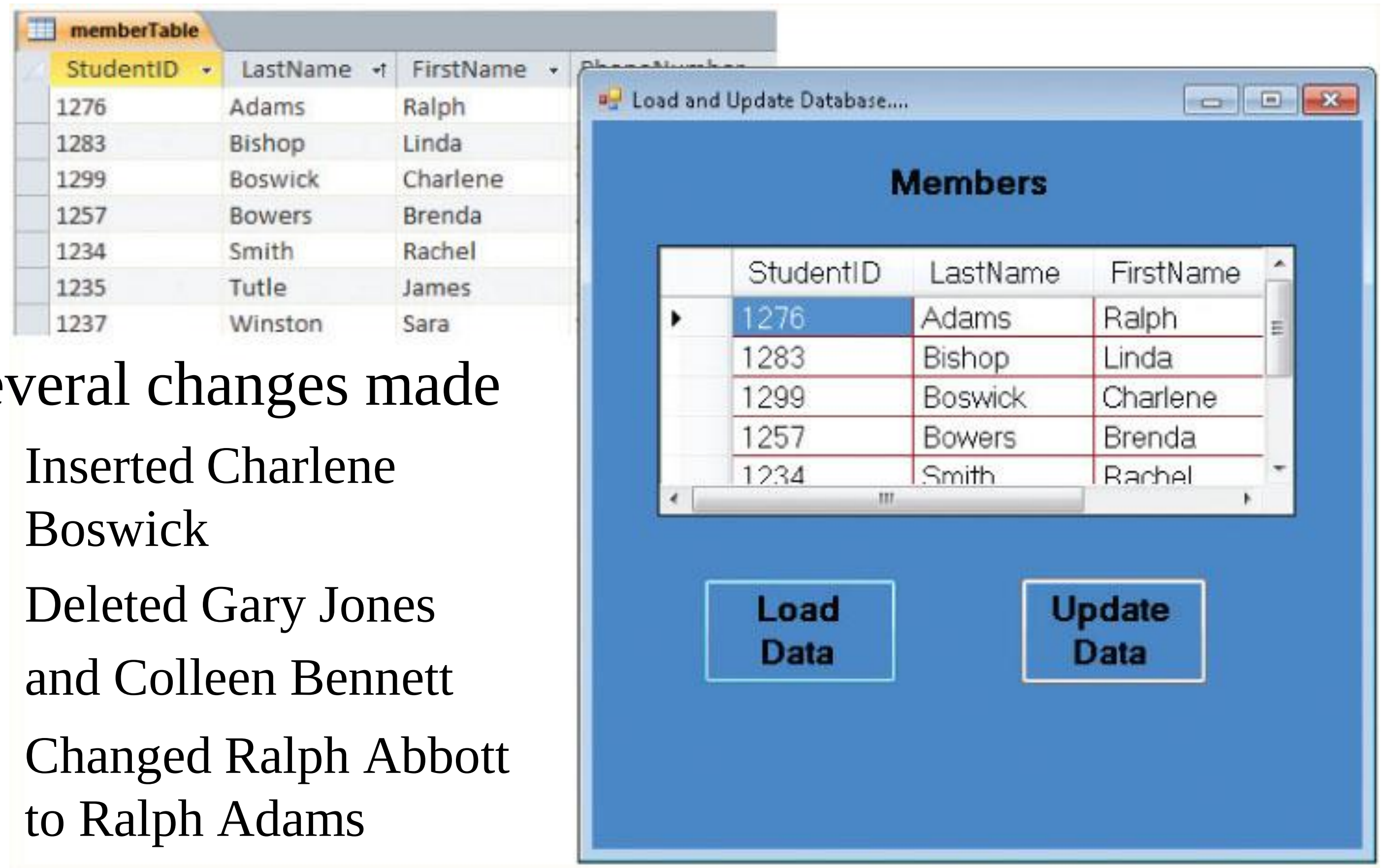


	StudentID	LastName	FirstName
▸	1276	Abbott	Ralph
	1299	Bennett	Colleen
	1283	Bishop	Linda
	1257	Bowers	Brenda
	1260	Jones	Gary

DataGridView object enables you to delete or insert new records (rows)

Figure 14-3 Output from DataSetExample after database is loaded

Updating the Database (continued)



memberTable

StudentID	LastName	FirstName
1276	Adams	Ralph
1283	Bishop	Linda
1299	Boswick	Charlene
1257	Bowers	Brenda
1234	Smith	Rachel
1235	Tutle	James
1237	Winston	Sara

Load and Update Database....

Members

StudentID	LastName	FirstName
1276	Adams	Ralph
1283	Bishop	Linda
1299	Boswick	Charlene
1257	Bowers	Brenda
1234	Smith	Rachel

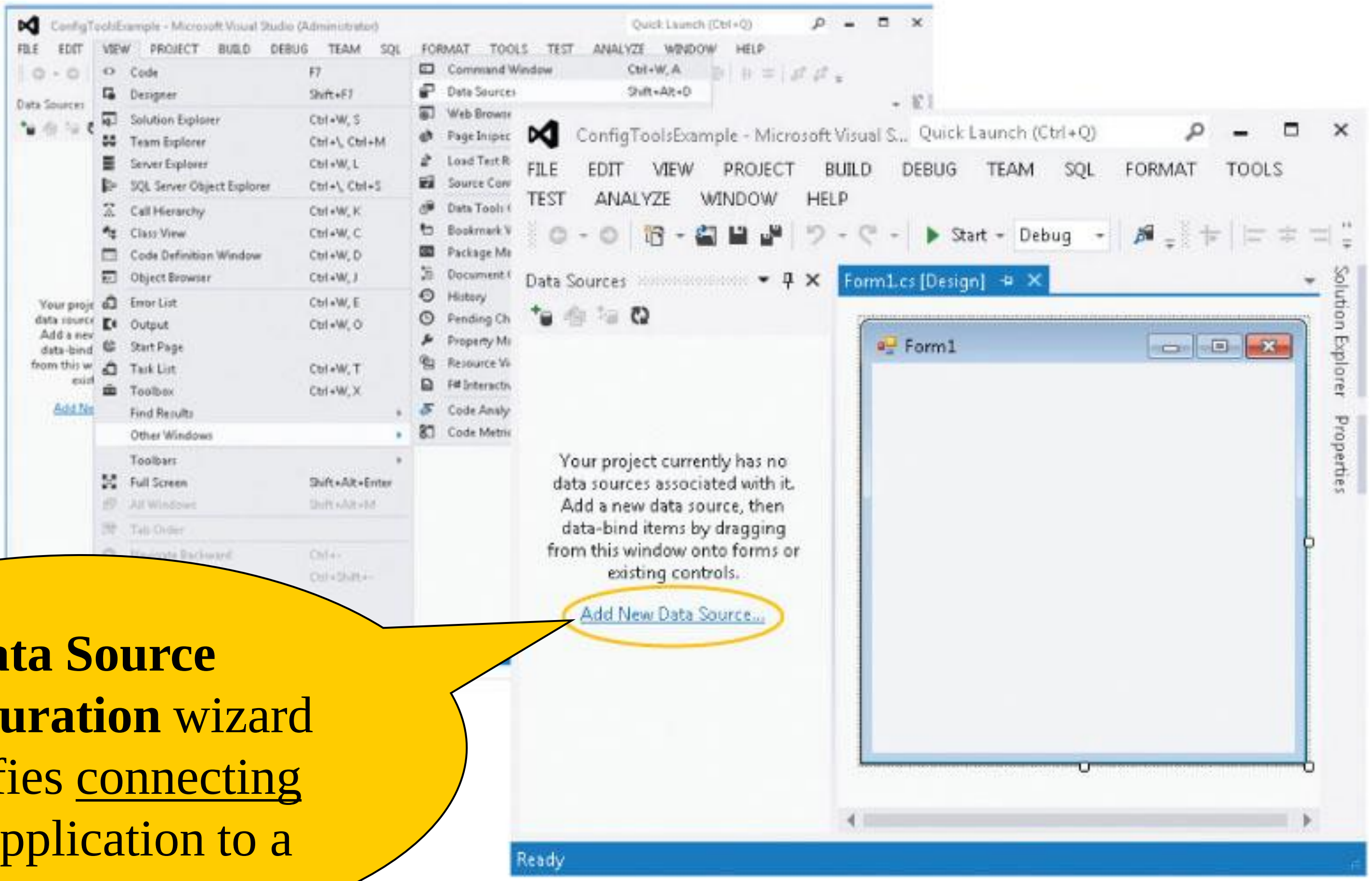
- Several changes made
  - Inserted Charlene Boswick
  - Deleted Gary Jones and Colleen Bennett
  - Changed Ralph Abbott to Ralph Adams

Figure 14-4 Updated database records

Data Source Configuration Tools

- Data configuration tools
  - Makes it easier to develop applications that access data
  - More drag-and-drop development – code is automatically generated
- Wizards that automatically:
  - Generate connection strings
  - Create dataset and table adapter objects
  - Bring data into the application

Data Source Configuration Tools


ConfigToolsExample - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL FORMAT TOOLS TEST ANALYZE WINDOW HELP
Other Windows
Command Window Ctrl+W, A
Data Sources Shift+Alt+D
Data Sources
Form1.cs [Design]
Form1
Start
Debug
Solution Explorer
Properties
Ready

Your project currently has no data sources associated with it. Add a new data source, then data-bind items by dragging from this window onto forms or existing controls.

Add New Data Source...

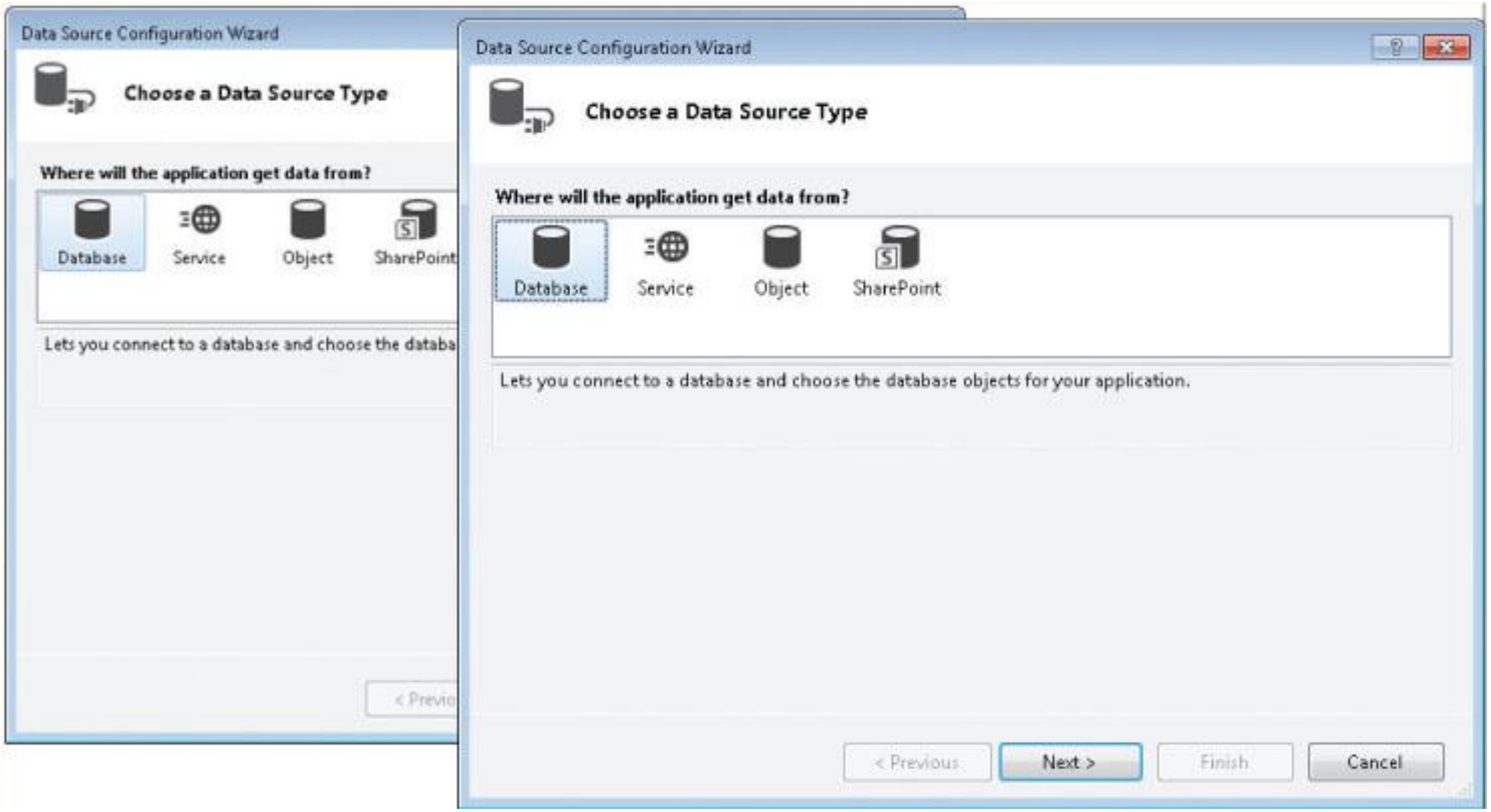
Data Source Configuration wizard simplifies connecting your application to a data source

Figure 14-5 Data Sources window

Add New Data Source

- Add new data source to application
  - Open **Data Sources** window (from **Data** menu)
    - **Data Sources** window visually shows the dataset objects available to the project
      - Datasets represent the in-memory cache of data
      - Datasets mimic the database from which they are based
  - First prompted to choose a data source type

Choose a Data Source Type



Where will the application get data from?

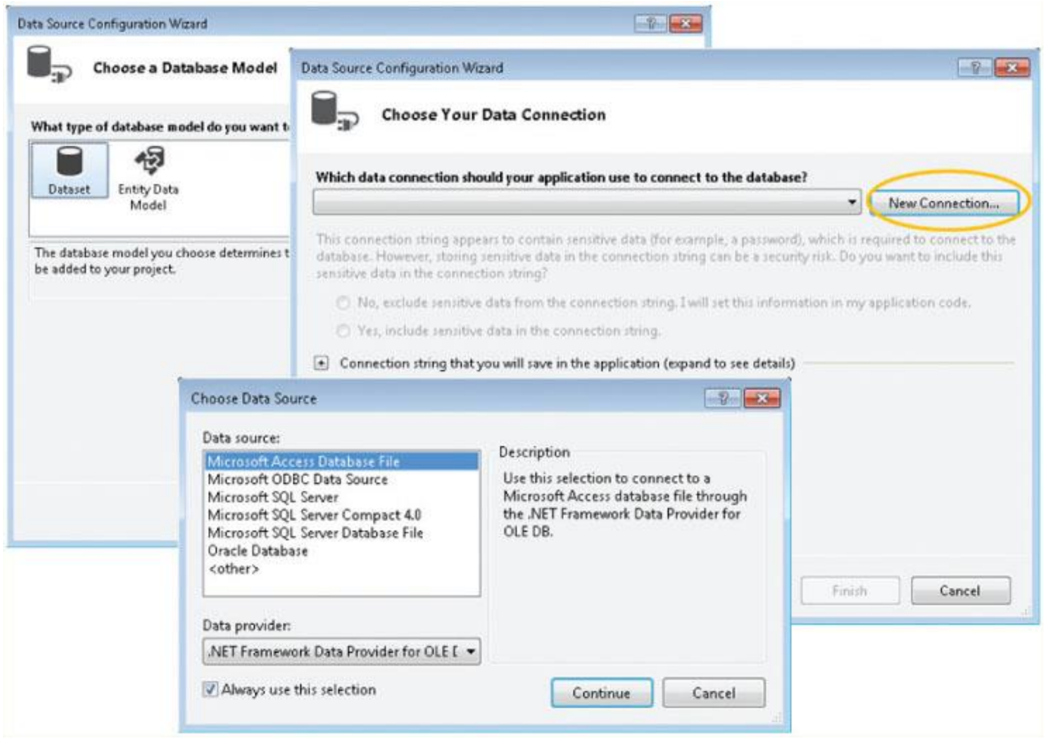
Lets you connect to a database and choose the database objects for your application.

Figure 14-6 Connect to a Database

New Connection

- Connections that are already established (attached) are available from the drop-down list

Follow same steps for SQL Server, Oracle, or Microsoft Access databases


Data Source Configuration Wizard
Choose a Database Model
What type of database model do you want t
Dataset
Entity Data Model
Data Source Configuration Wizard
Choose Your Data Connection
New Connection...
Connection string that you will save in the application (expand to see details)
Finish
Cancel
Choose Data Source
Data source:
Description
Data provider:
Continue
Cancel

The database model you choose determines t be added to your project.

Which data connection should your application use to connect to the database?

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- No, exclude sensitive data from the connection string. I will set this information in my application code.
- Yes, include sensitive data in the connection string.

Data source:
- Microsoft Access Database File
- Microsoft ODBC Data Source
- Microsoft SQL Server
- Microsoft SQL Server Compact 4.0
- Microsoft SQL Server Database File
- Oracle Database
- <other>

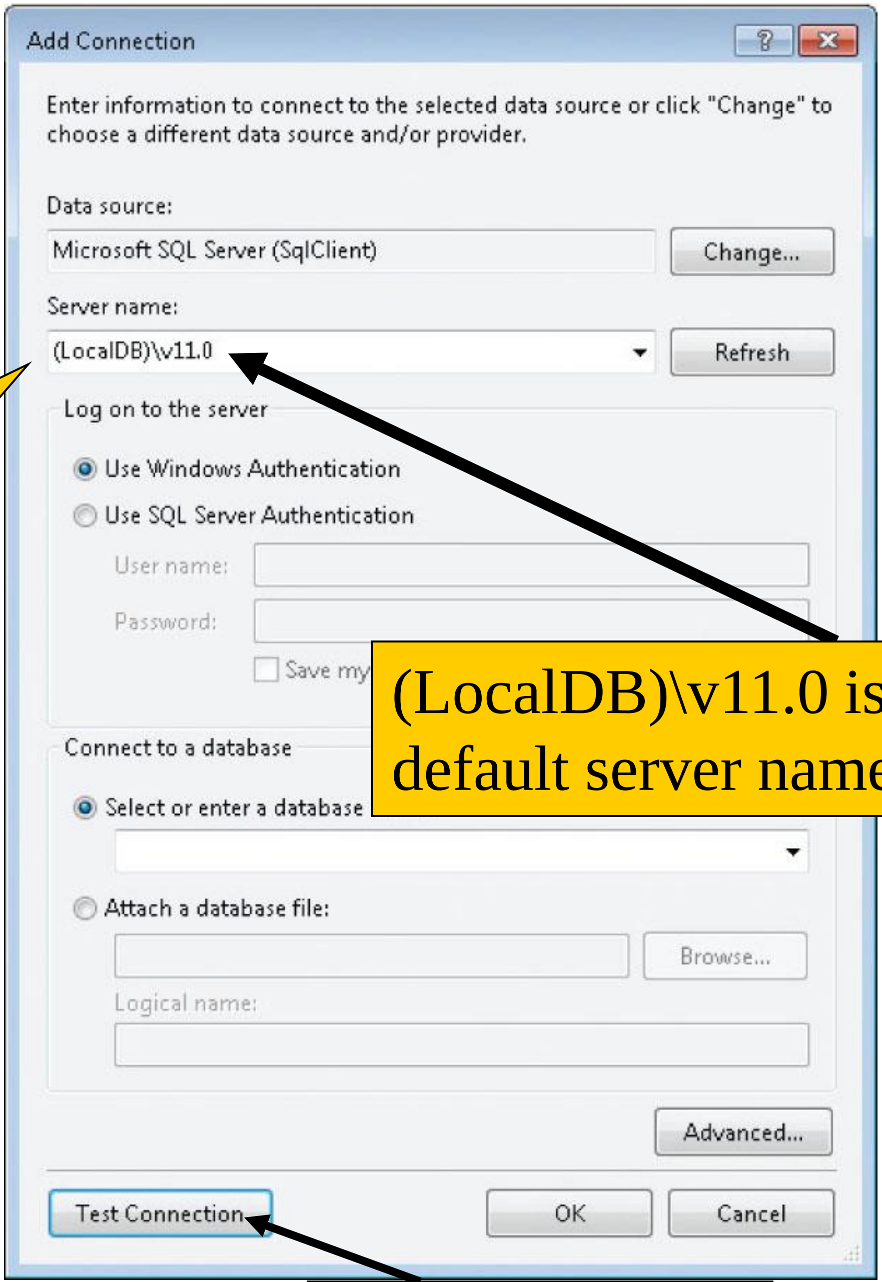
Description: Use this selection to connect to a Microsoft Access database file through the .NET Framework Data Provider for OLE DB.

Data provider: .NET Framework Data Provider for OLE I

[✓] Always use this selection

Figure 14-7 Add a New Connection

Add Connection



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: (LocalDB)\v11.0

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

☐ Save my

Connect to a database

● Select or enter a database

○ Attach a database file:

Logical name:

Refresh button should be pressed after the server name is entered

(LocalDB)\v11.0 is default server name

Test Connection

Figure 14-8 Select the data source

SQL Server Databases

- Create new SQL Server Databases
  - Display **Server Explorer** Window (from View menu)
  - Right-click on **Data Connection**
  - Select **Create new SQL Server database**
- Create new tables
  - Right-mouse click on Tables node
  - Select Add new Table
- Administrative permissions <u>on the local machine</u> needed to create or attach to a SQL Server using Visual Studio

Create SQL Server Database (continued)

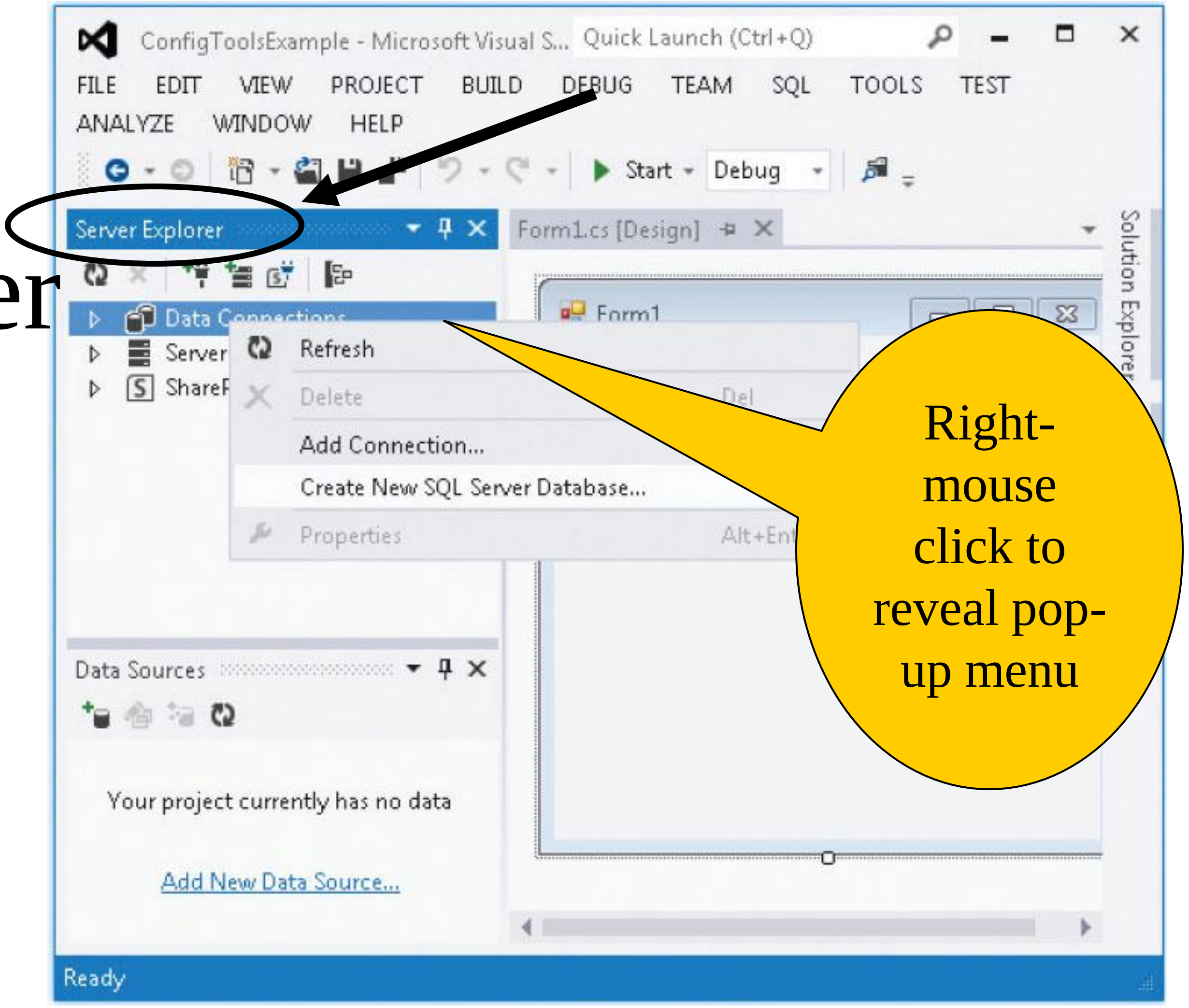


Figure 14-9 Server Explorer window

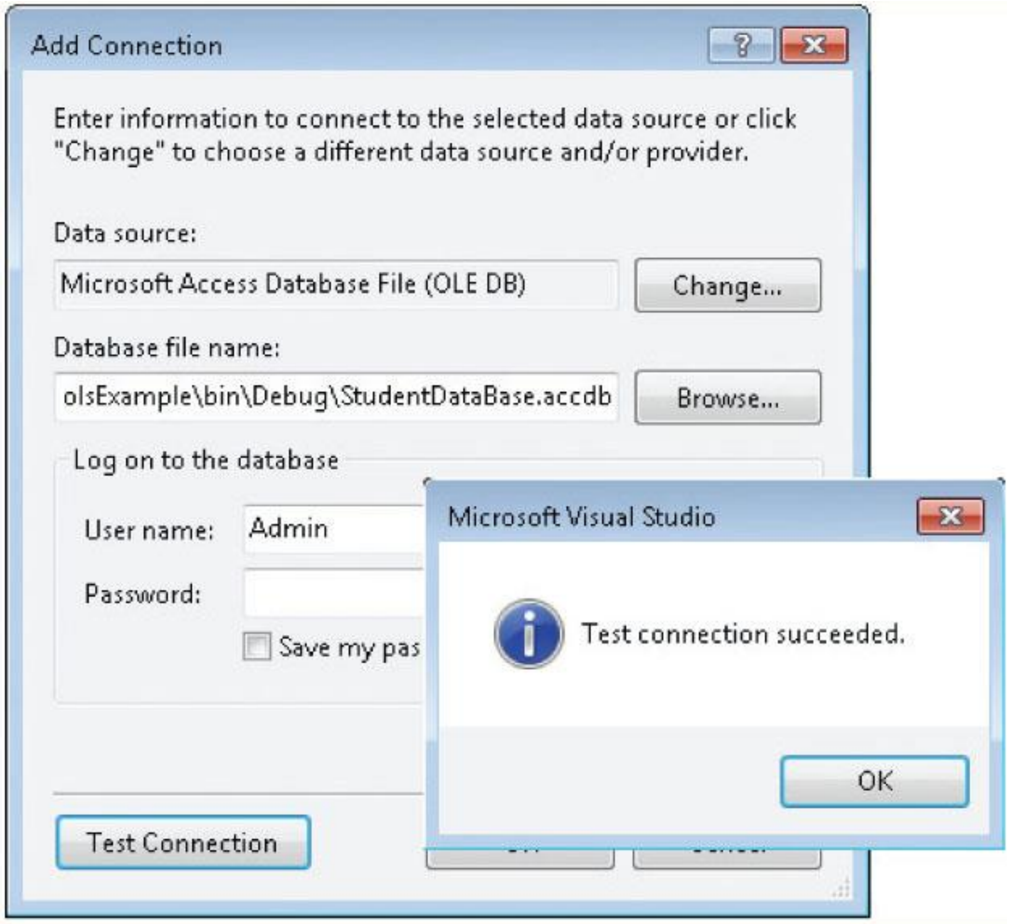
SQL Server Database Tables

- Store Data in Tables
  - Use the Server Explorer window
  - Right-mouse click on a table, select **Show Table Data** to store data
  - Type the data in the table
  - Table saved on exit
- Modify the structure
  - Select **Open Table Definition (**right-mouse click in Server Explorer window)
  - Set primary keys
    - Right-mouse clicking on the key row

Adding a Connection

- Right-click on Server Explorer, select Add Connection option
  - Here you specify the data source, database filename, and test the connection
  - Also add the connection using the Add New Data Source option from the Data menu

Testing the Connection


Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft Access Database File (OLE DB)

Database file name: olsExample\bin\Debug\StudentDataBase.accdb

Log on to the database

User name: Admin

Password:

Save my pas

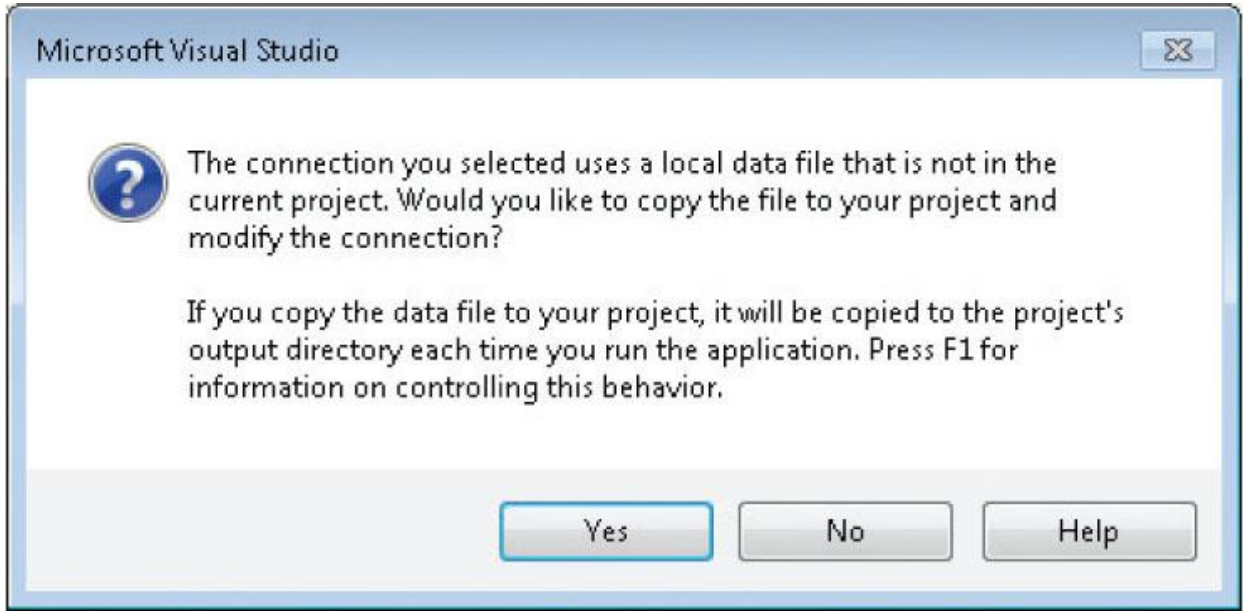
Microsoft Visual Studio

Test connection succeeded.

Figure 14-10 Locate and test the connection

Local Copy of Your Database

First time you establish a connection to the database for your application

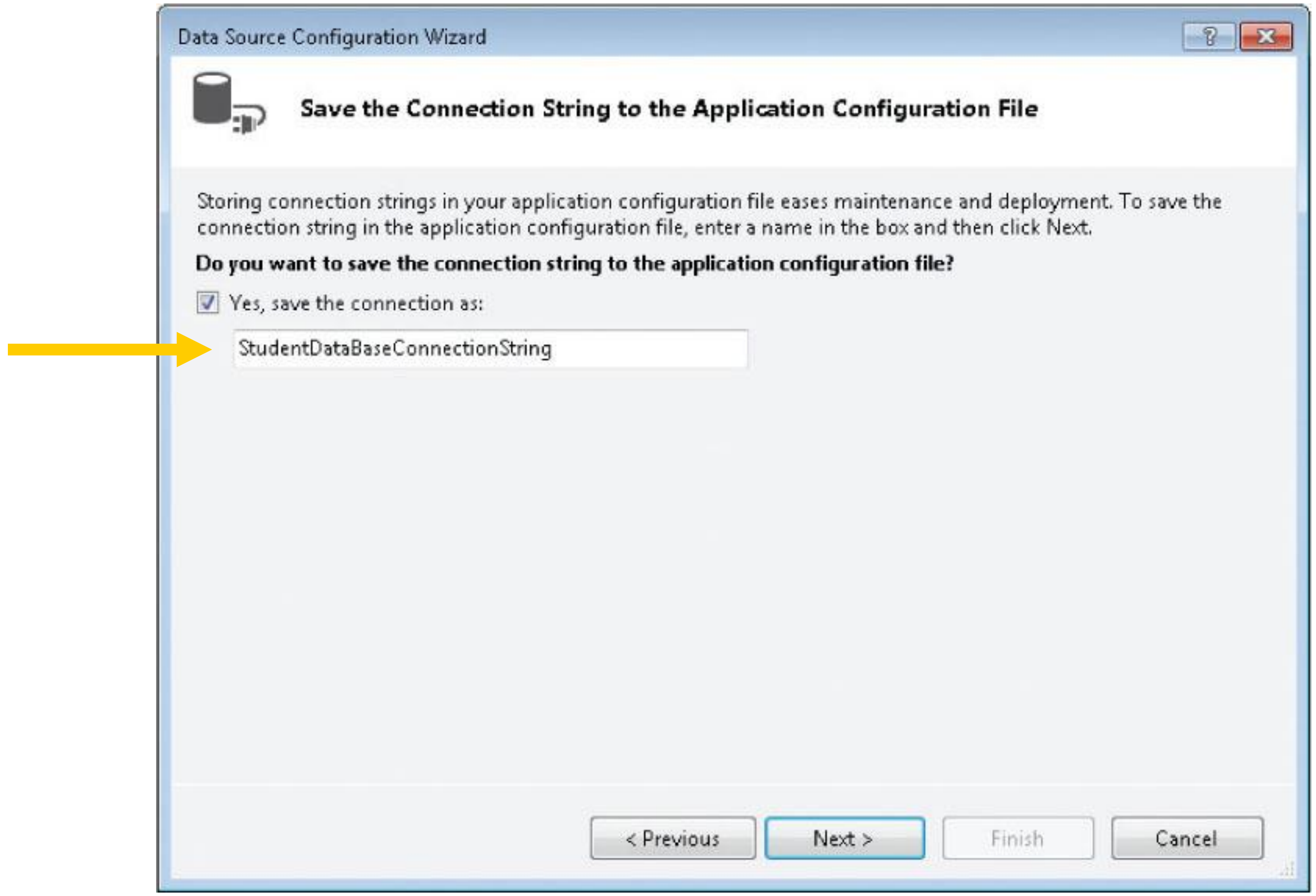

Microsoft Visual Studio

The connection you selected uses a local data file that is not in the current project. Would you like to copy the file to your project and modify the connection?

If you copy the data file to your project, it will be copied to the project's output directory each time you run the application. Press F1 for information on controlling this behavior.

Figure 14-11 Copy database file to your project

Connection String Created


Data Source Configuration Wizard

Save the Connection String to the Application Configuration File

Storing connection strings in your application configuration file eases maintenance and deployment. To save the connection string in the application configuration file, enter a name in the box and then click Next.

Do you want to save the connection string to the application configuration file?

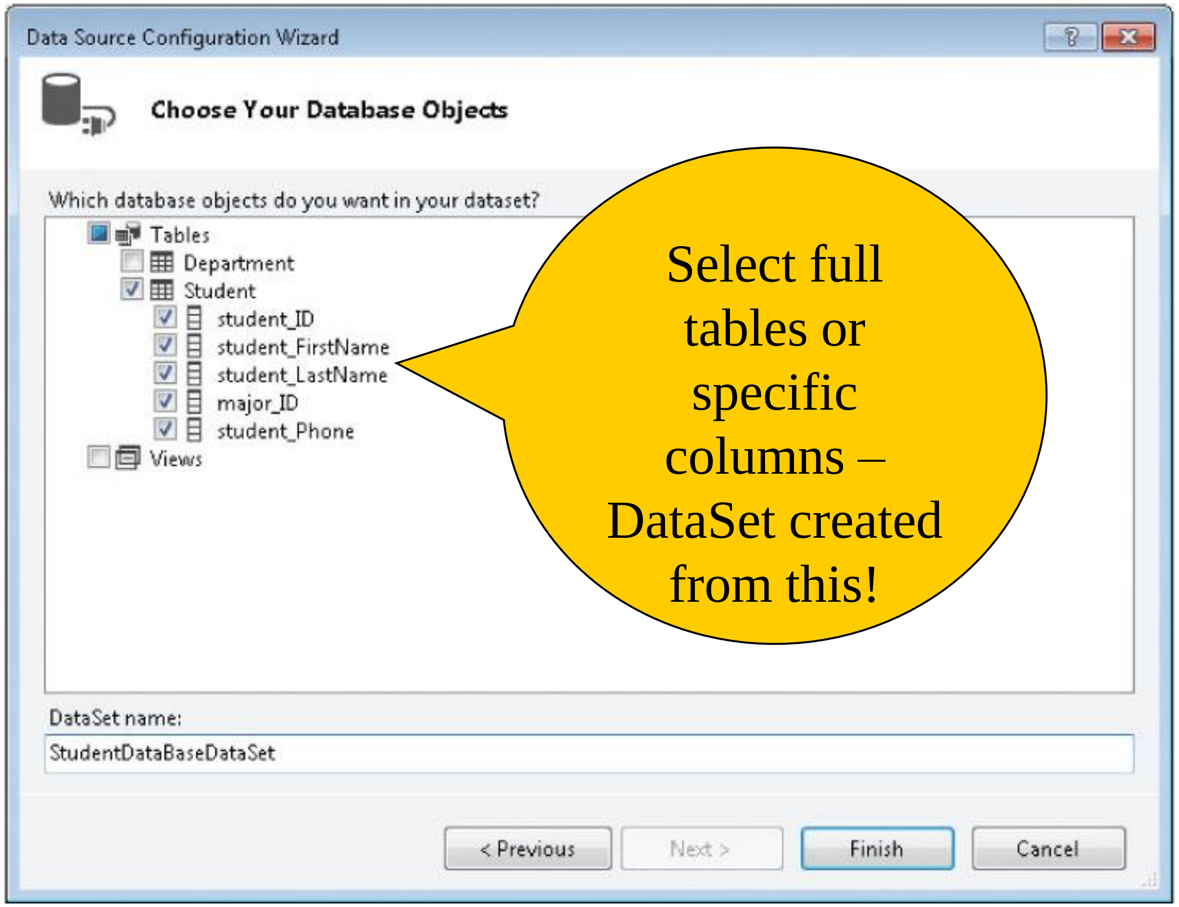
☑ Yes, save the connection as:

StudentDataBaseConnectionString

Figure 14-12 Save connection string

Dataset Object

- Identify database objects that you want to bring into your application
  - Chosen objects become accessible through the dataset object


Data Source Configuration Wizard

Choose Your Database Objects

Which database objects do you want in your dataset?

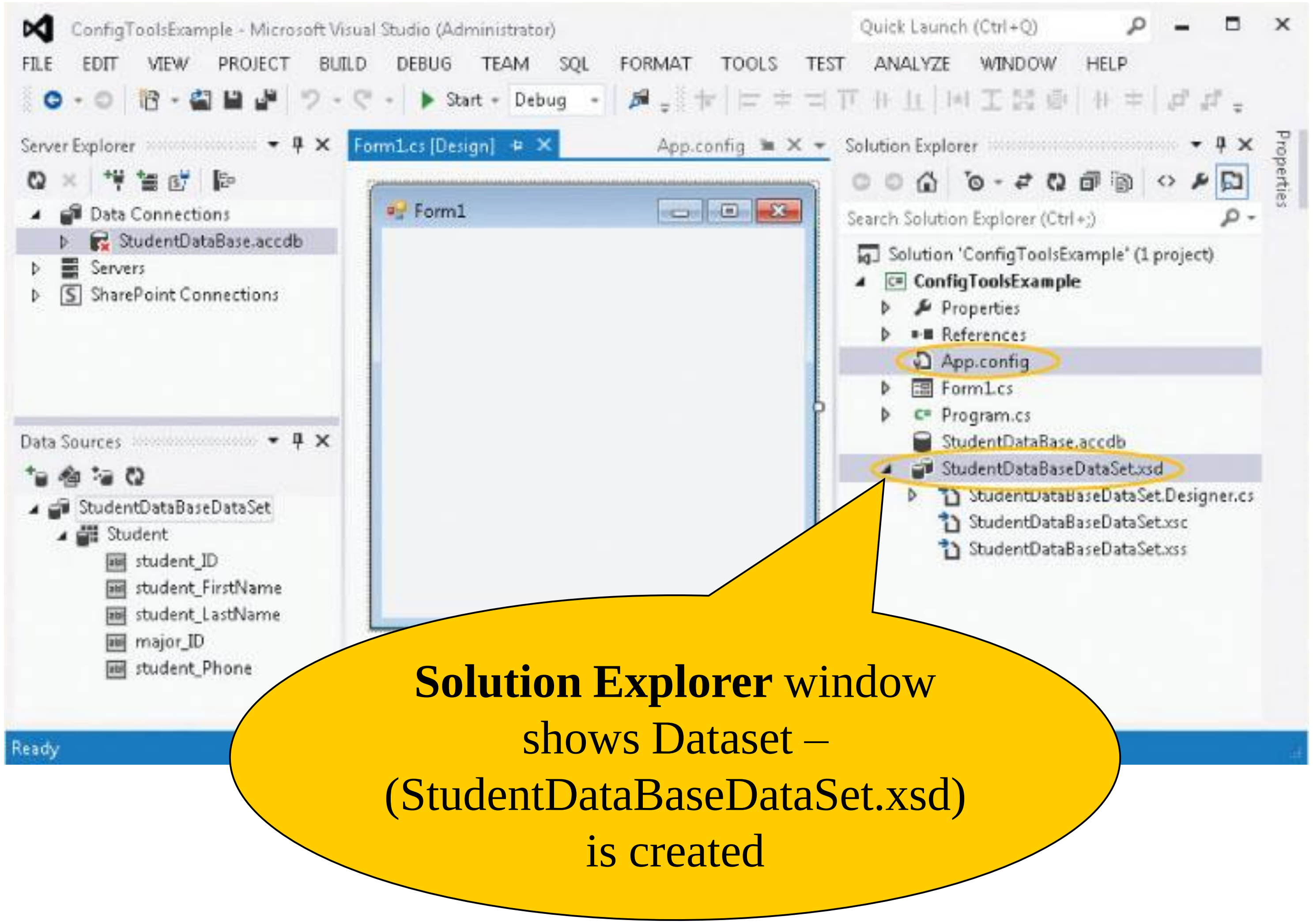
- Tables
  - [ ] Department
  - [✓] Student
    - [✓] student_ID
    - [✓] student_FirstName
    - [✓] student_LastName
    - [✓] major_ID
    - [✓] student_Phone
- [ ] Views

DataSet name: StudentDataBaseDataSet

< Previous
Next >
Finish
Cancel
Select full tables or specific columns – DataSet created from this!

Figure 14-13 Choose dataset objects

Data Sources



Solution Explorer window shows Dataset – (StudentDataBaseDataSet.xsd) is created

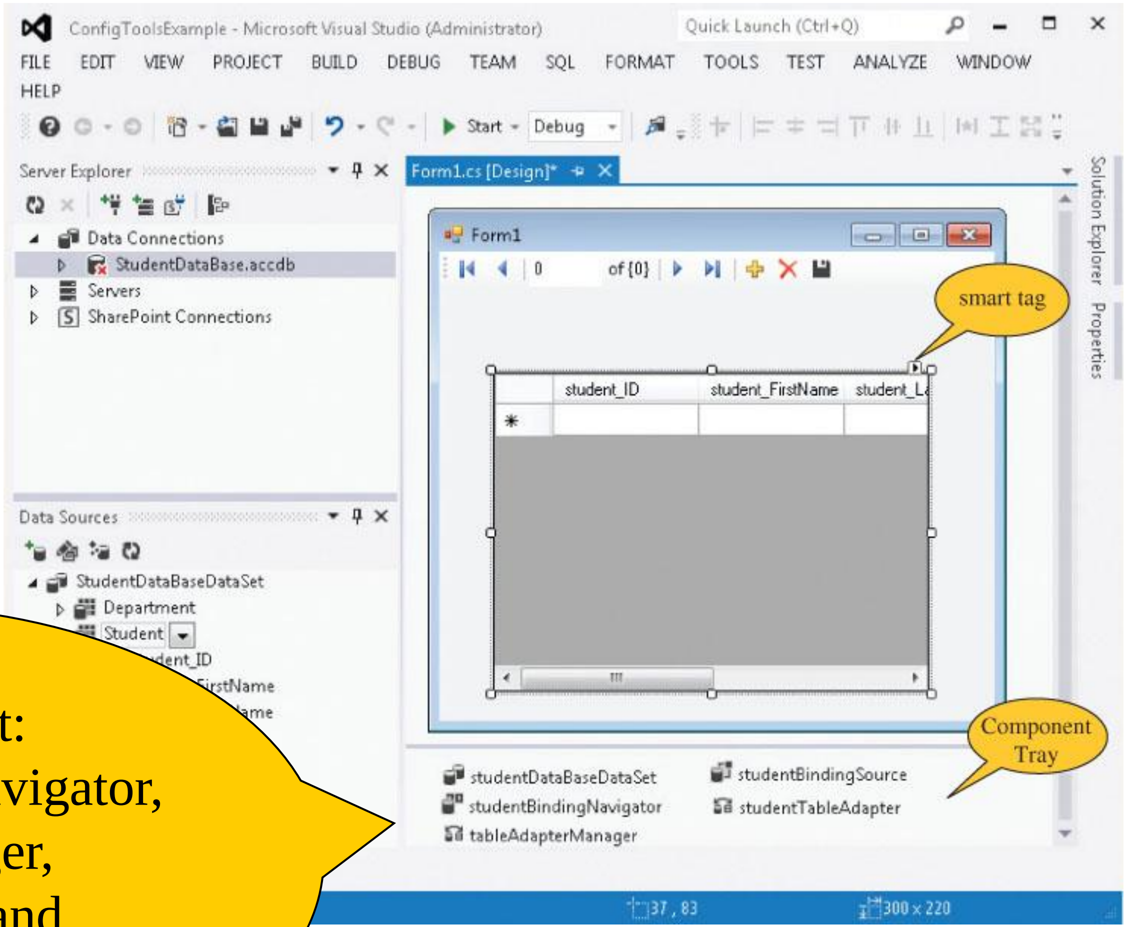
Figure 14-14 Data Sources and Solution Explorer windows

DataGridView Control

- Placeholder control for displaying data on form
  - To instantiate DataGridView control, drag a table from **Data Sources** window to form
- Specify how data is formatted and displayed
  - DataGridView – customizable table that allows you to modify columns, rows, and borders
    - Freeze rows and columns for scrolling purposes
    - Hide rows or columns
    - Provide ToolTips and shortcut menus

Placing DataGridView Control

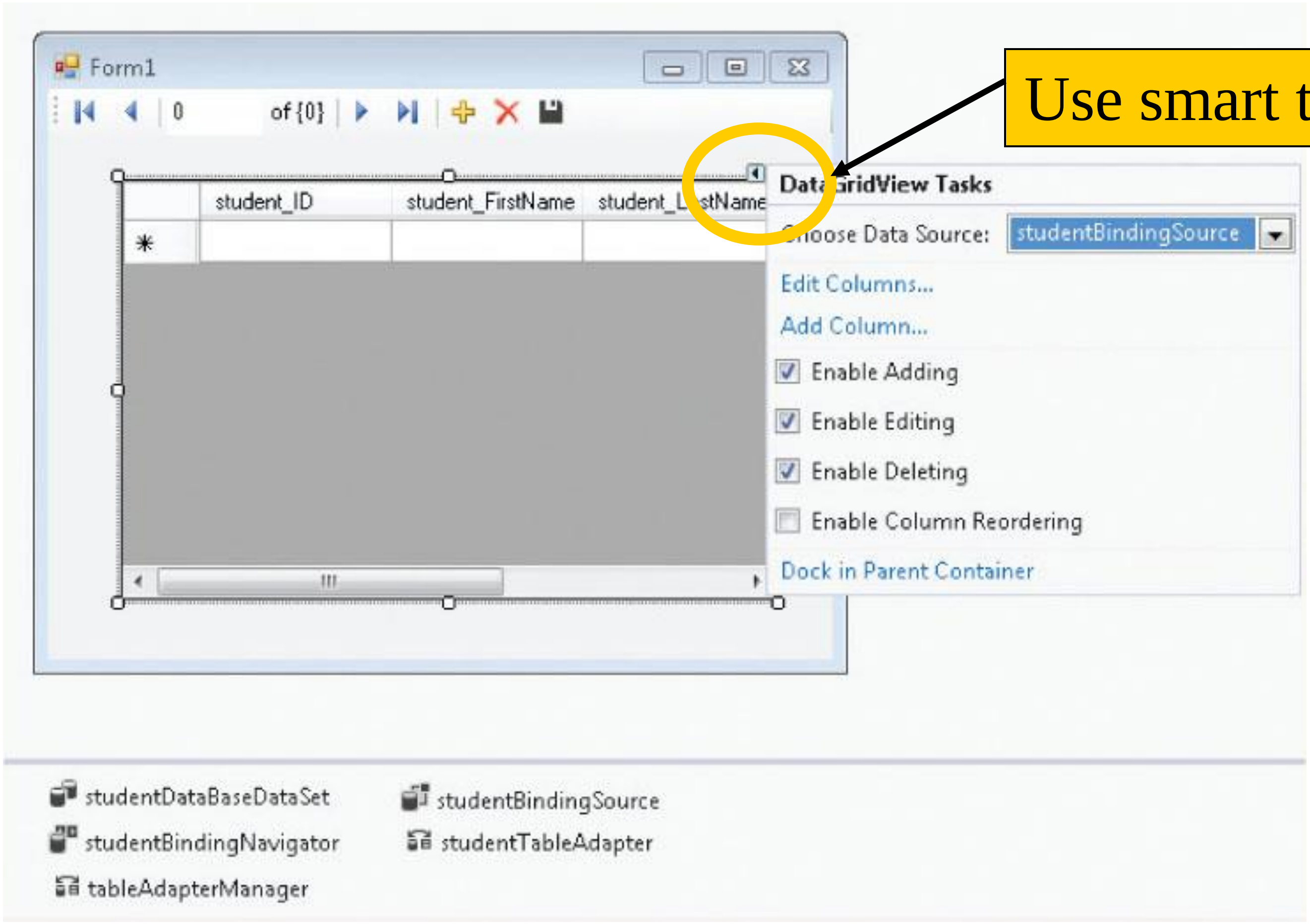
Table dragged from **Data Sources** window to the form; DataGridView Control created



Added benefit:
DataSet, BindingNavigator, AdapterManager, TableAdapter, and BindingSource objects automatically instantiated

Figure 14-15 DataGridView control placed on form

Customize the DataGridView Object



Choose Data Source: studentBindingSource

Edit Columns...

Add Column...

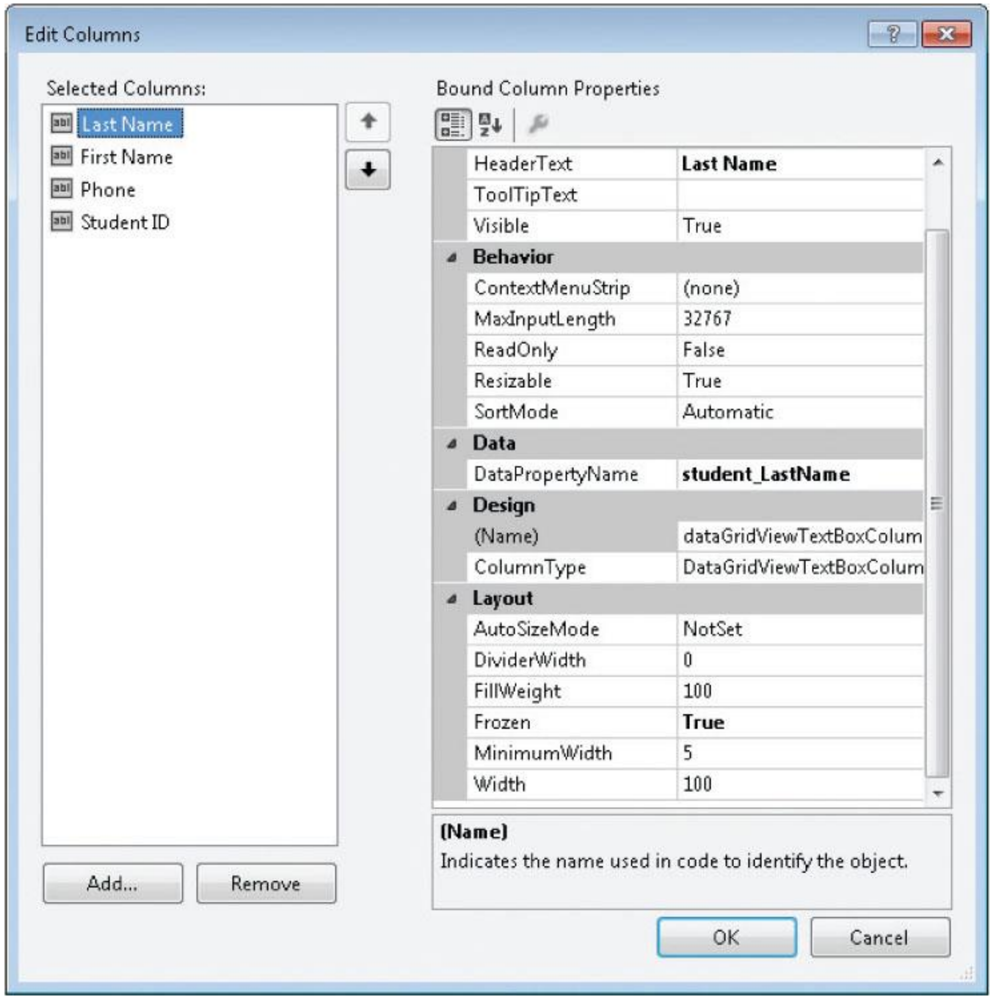
- [✓] Enable Adding
- [✓] Enable Editing
- [✓] Enable Deleting
- [ ] Enable Column Reordering

Dock in Parent Container

studentDataBaseDataSet
studentBindingSource
studentBindingNavigator
studentTableAdapter
tableAdapterManager

Figure 14-16 Customizing the DataGridView control

Editing Columns



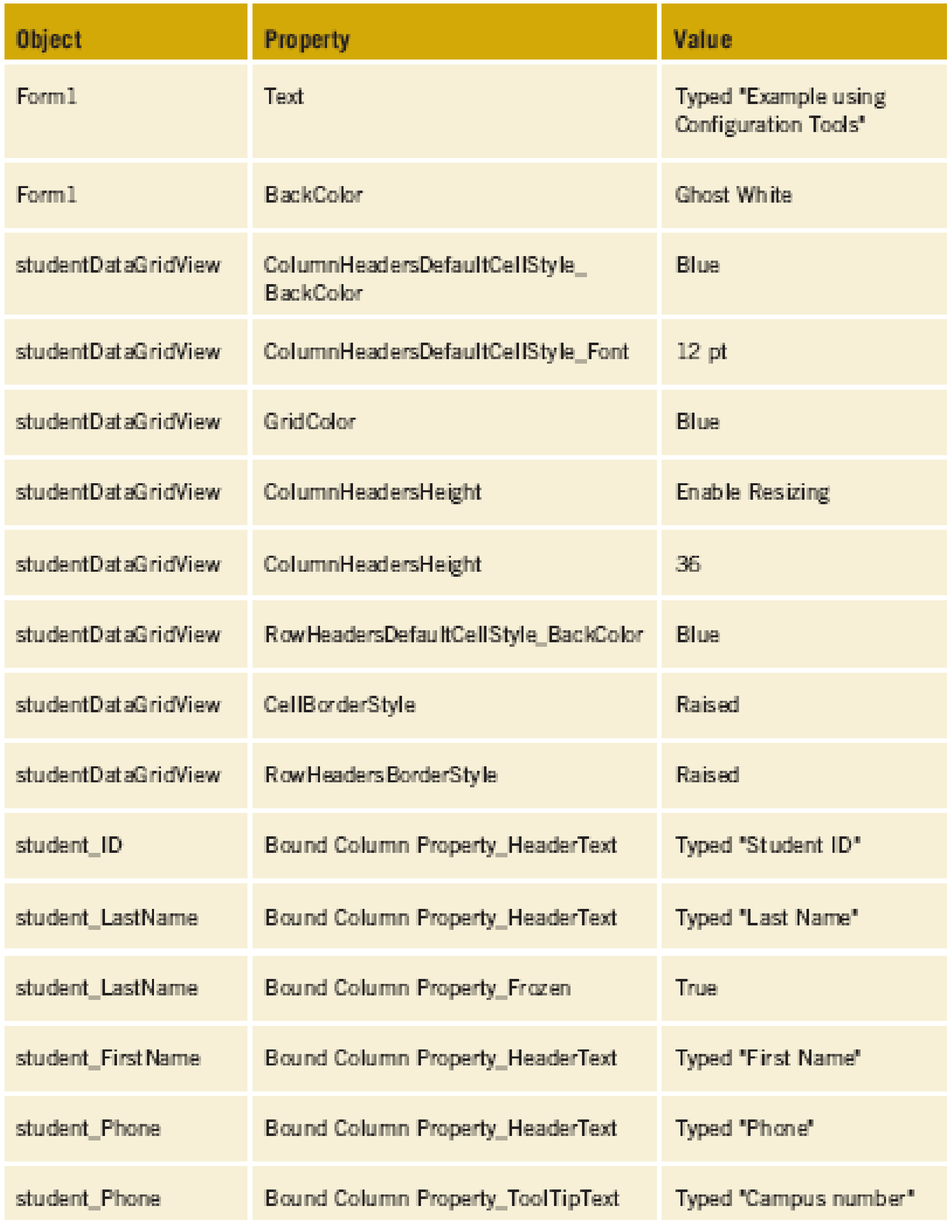
Selected Columns:
- Last Name
- First Name
- Phone
- Student ID

HeaderText	**Last Name**
ToolTipText	
Visible	True
Behavior	
ContextMenuStrip	(none)
MaxInputLength	32767
ReadOnly	False
Resizable	True
SortMode	Automatic
Data	
DataPropertyName	**student_LastName**
Design	
(Name)	dataGridViewTextBoxColum
ColumnType	DataGridViewTextBoxColum
Layout	
AutoSizeMode	NotSet
DividerWidth	0
FillWeight	100
Frozen	**True**
MinimumWidth	5
Width	100

(Name)
Indicates the name used in code to identify the object.

Figure 14-17 Edit DataGridView Columns

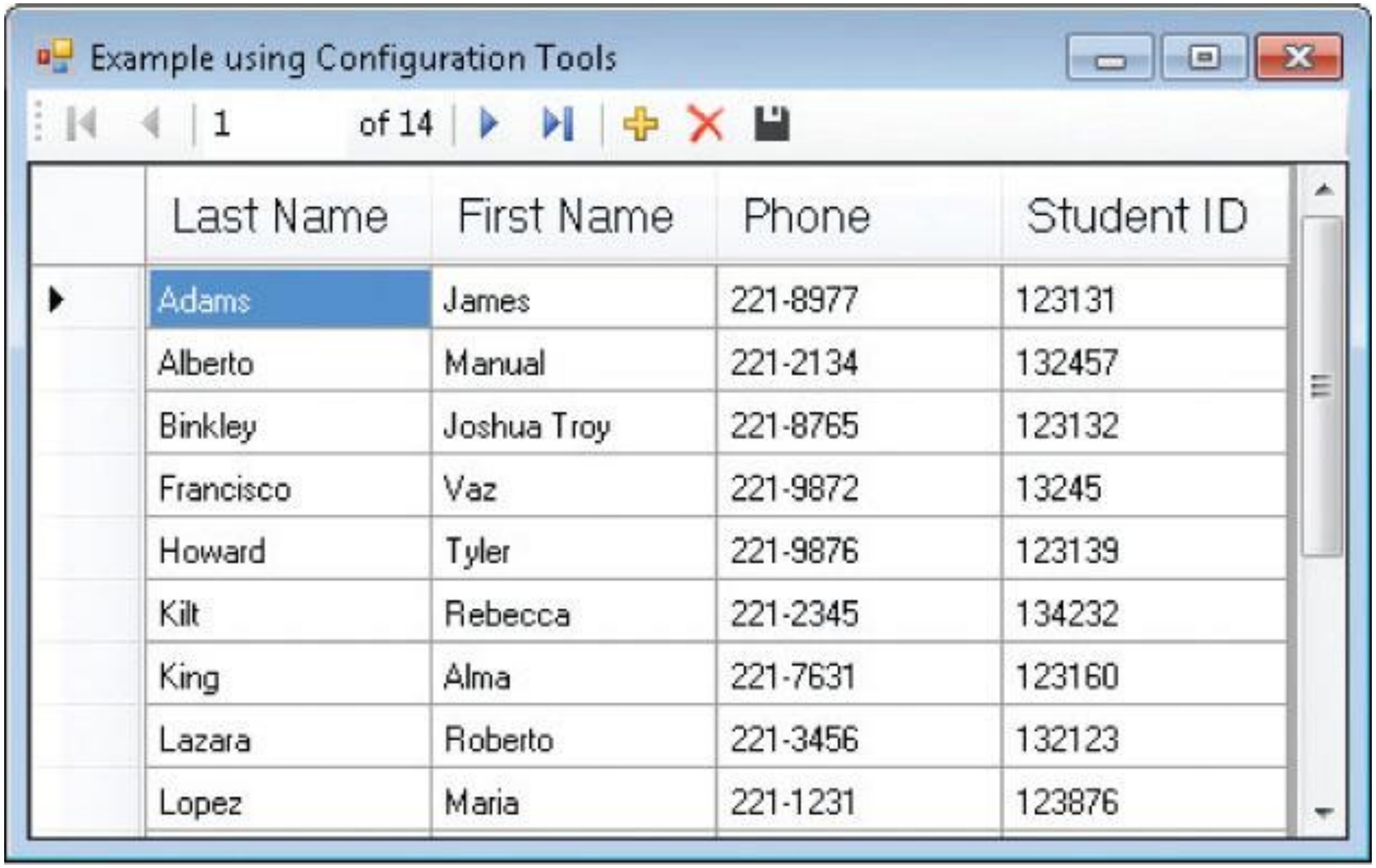
Editing Columns (continued)



Object	Property	Value
Form1	Text	Typed "Example using Configuration Tools"
Form1	BackColor	Ghost White
studentDataGridView	ColumnHeadersDefaultCellStyle_ BackColor	Blue
studentDataGridView	ColumnHeadersDefaultCellStyle_Font	12 pt
studentDataGridView	GridColor	Blue
studentDataGridView	ColumnHeadersHeight	Enable Resizing
studentDataGridView	ColumnHeadersHeight	36
studentDataGridView	RowHeadersDefaultCellStyle_BackColor	Blue
studentDataGridView	CellBorderStyle	Raised
studentDataGridView	RowHeadersBorderStyle	Raised
student_ID	Bound Column Property_HeaderText	Typed "Student ID"
student_LastName	Bound Column Property_HeaderText	Typed "Last Name"
student_LastName	Bound Column Property_Frozen	True
student_FirstName	Bound Column Property_HeaderText	Typed "First Name"
student_Phone	Bound Column Property_HeaderText	Typed "Phone"
student_Phone	Bound Column Property_ToolTipText	Typed "Campus number"

Table 14-7 ConfigToolsExample property values

Editing Columns

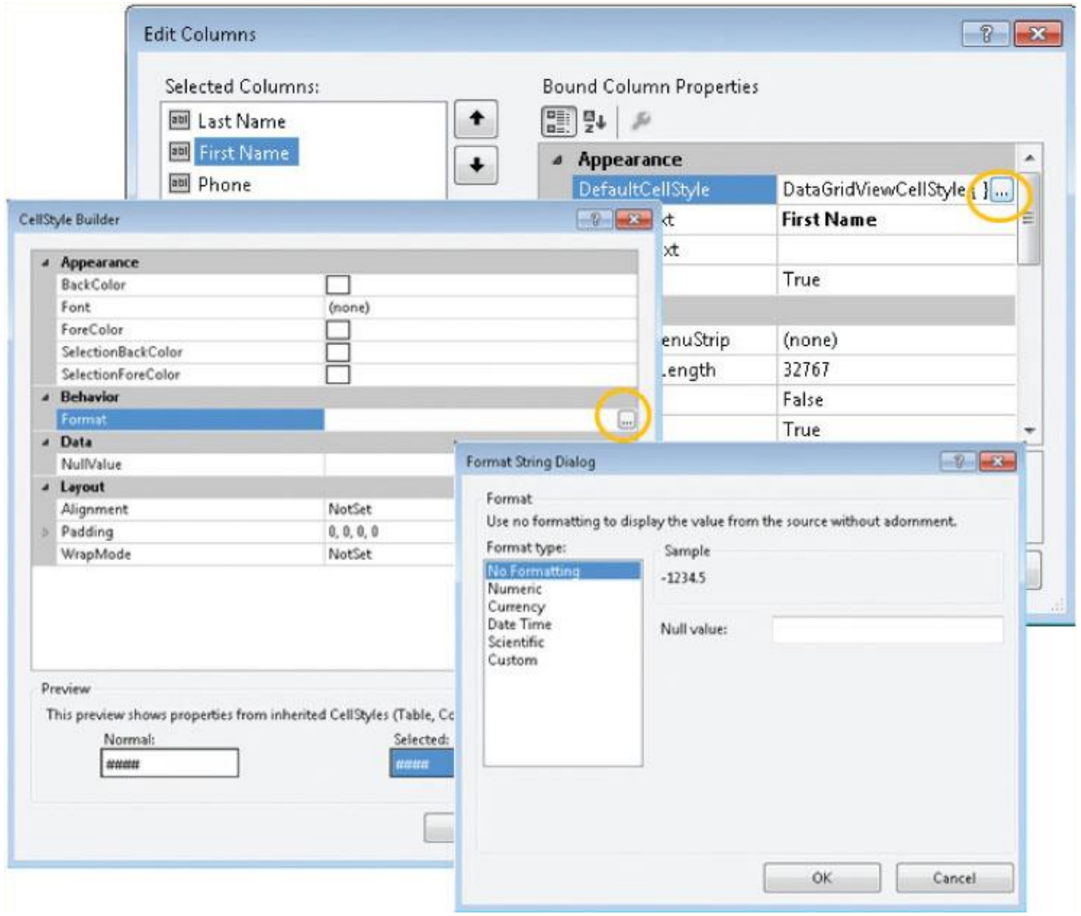


	Last Name	First Name	Phone	Student ID
▶	Adams	James	221-8977	123131
	Alberto	Manual	221-2134	132457
	Binkley	Joshua Troy	221-8765	123132
	Francisco	Vaz	221-9872	13245
	Howard	Tyler	221-9876	123139
	Kilt	Rebecca	221-2345	134232
	King	Alma	221-7631	123160
	Lazara	Roberto	221-3456	132123
	Lopez	Maria	221-1231	123876

Figure 14-18 Example using Configuration Tools output

Formatting DataGridView Cells

Can also define custom formats for a cell



DefaultCellStyle	DataGridViewCellStyle { }
BackColor	
Font	(none)
ForeColor	
SelectionBackColor	
SelectionForeColor	
Format	
NullValue	
Alignment	NotSet
Padding	0, 0, 0, 0
WrapMode	NotSet

Format

Use no formatting to display the value from the source without adornment.

Format type: No Formatting, Numeric, Currency, Date Time, Scientific, Custom

Sample: -1234.5

Null value:

Figure 14-19 Formatting DataGridView cells

BindingNavigator Control

- One of the five objects added to the component tray at the bottom of the form when the table from the Data Sources pane is placed on form
- Provides a standardized way to move through and process the data
- Much functionality is automatically programmed into the tool strip
  - Code was also automatically generated

BindingNavigator Control (continued)

Standardized controls included to move through and process data

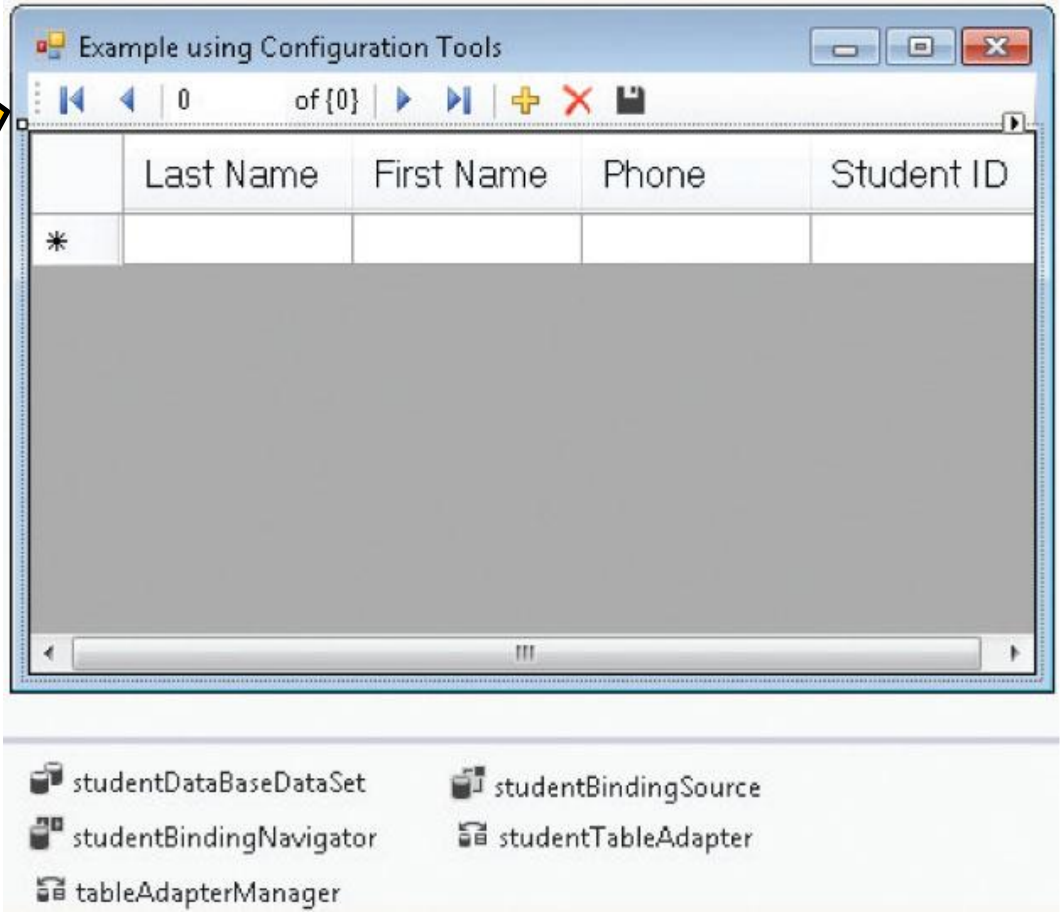


Figure 14-20 BindingNavigator and BindingSource objects

Adding Update Functionality

- Data adapter and dataset used to update data using disconnected architecture
- Data adapters and/or table adapters read data from a database into a dataset
  - Interaction between the dataset and the actual database is controlled through the methods of the data adapter or table adapter objects
- To write changed data from the dataset back to the database – SELECT, INSERT, DELETE, and UPDATE SQL statements used
  - SQL statements set through properties of data adapters and/or table adapters

TableAdapterManager

```
private void studentBindingNavigatorSaveItem_Click (object sender,
                                                     EventArgs e)
{
    this.Validate( );
    this.studentBindingSource.EndEdit( );
    this.tableAdapterManager.UpdateAll
          (this.studentDataBaseDataSet.Student);
}
```

- TableAdapterManager extremely useful when an application pulls data from two or more tables
  - Uses the foreign-key relationships

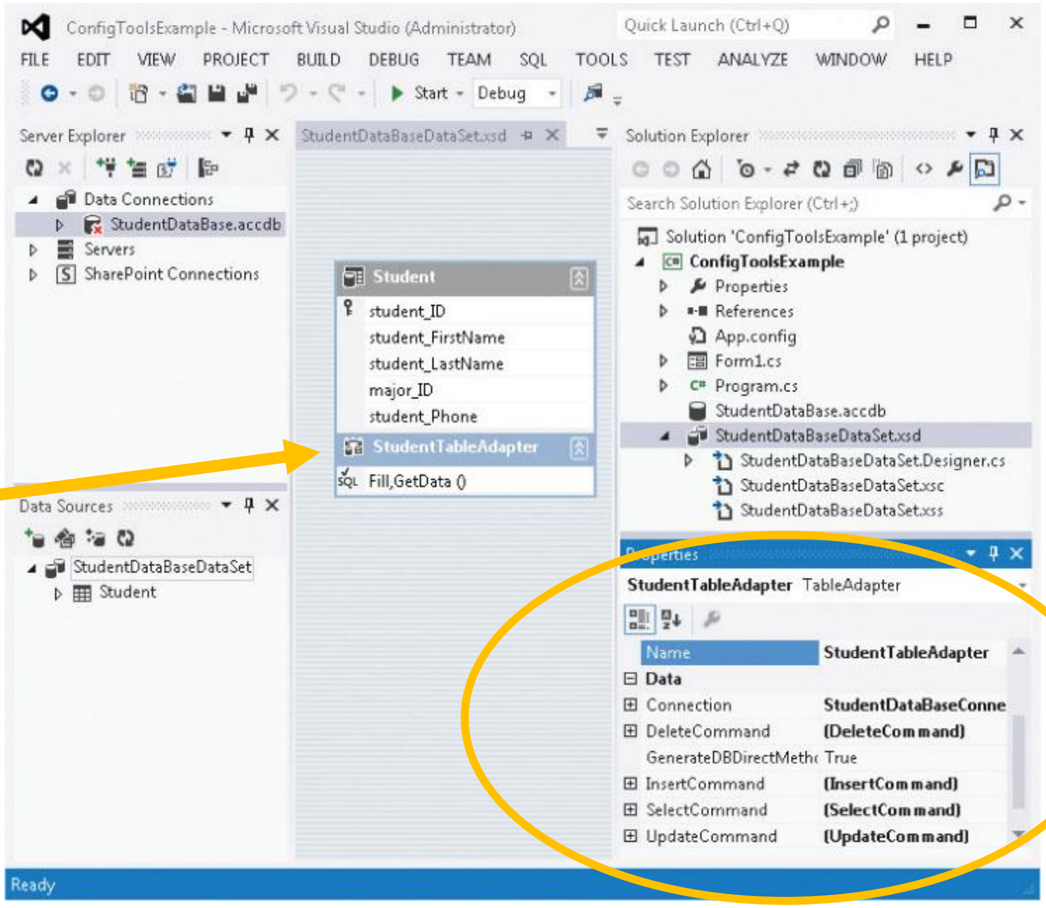
TableAdapters

- Data adapter on steroids
- Update( ) method has to have SQL SELECT, INSERT, DELETE, AND UPDATE commands
- Configure TableAdapter to update data
  - Select TableAdapter object in component tray to view its properties
  - Set the SQL query for the CommandText for SelectCommand, InsertCommand, UpdateCommand, and DeleteCommand properties
- Use the DataSet Designer to view and modify CommandText for these properties

DataSet Designer

- Create and modify data adapters and table adapters (and their SQL statements)
- To start the designer, double-click on dataset in **Solution Explorer** window <u>or</u> right-click the dataset in the **Data Sources** window
- Visual representation of the dataset and table adapter is presented

Dataset Designer (continued)

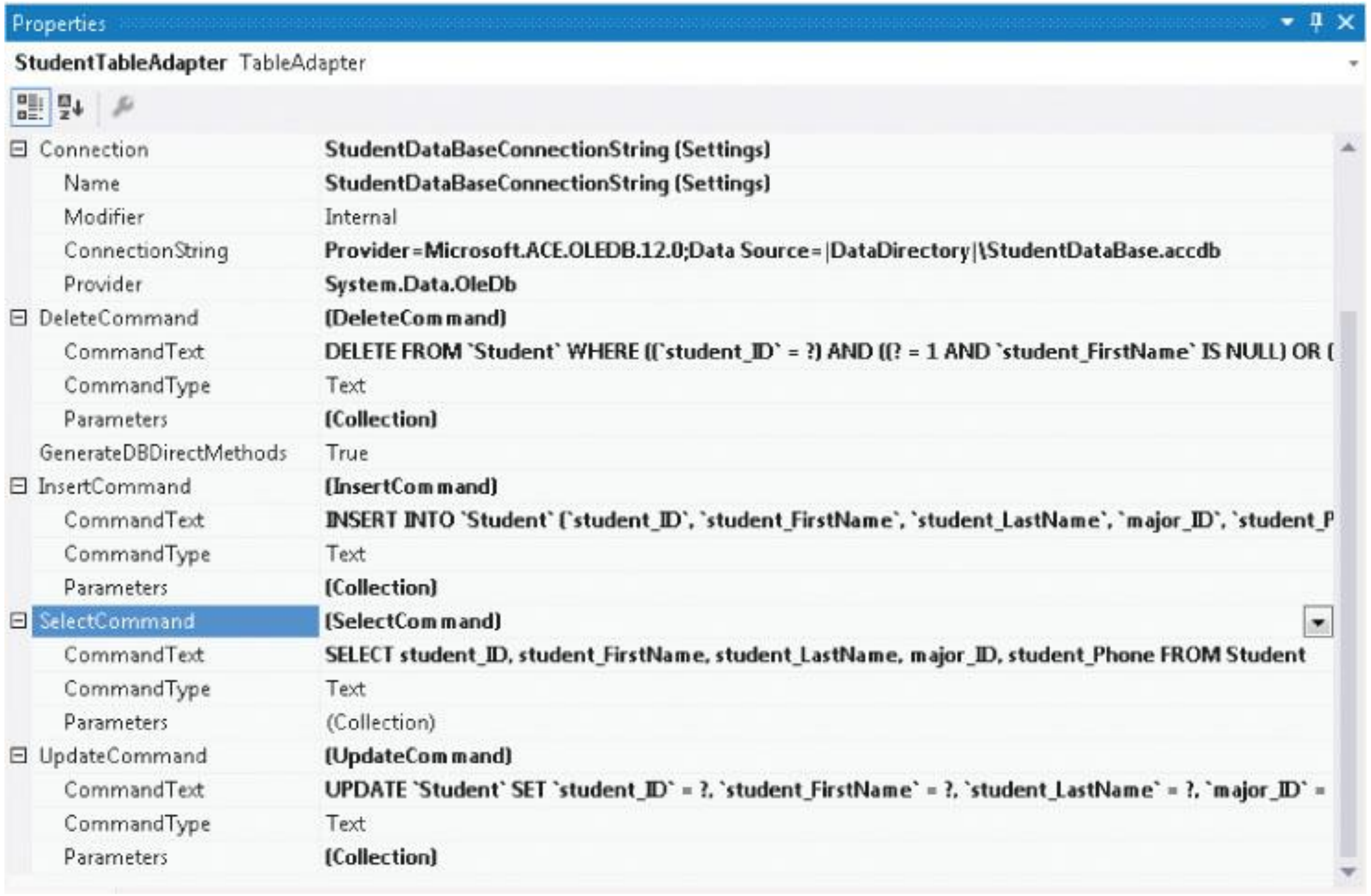

ConfigToolsExample - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Start Debug
Server Explorer
Data Connections
StudentDataBase.accdb
Servers
SharePoint Connections
StudentDataBaseDataSet.xsd
Student
student_ID
student_FirstName
student_LastName
major_ID
student_Phone
StudentTableAdapter
Fill,GetData ()
Data Sources
StudentDataBaseDataSet
Student
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ConfigToolsExample' (1 project)
ConfigToolsExample
Properties
References
App.config
Form1.cs
Program.cs
StudentDataBase.accdb
StudentDataBaseDataSet.xsd
StudentDataBaseDataSet.Designer.cs
StudentDataBaseDataSet.xsc
StudentDataBaseDataSet.xss
Properties
StudentTableAdapter TableAdapter
Ready

Name	StudentTableAdapter
⊟ Data	
⊞ Connection	StudentDataBaseConne
⊞ DeleteCommand	(DeleteCommand)
GenerateDBDirectMeth	True
⊞ InsertCommand	(InsertCommand)
⊞ SelectCommand	(SelectCommand)
⊞ UpdateCommand	(UpdateCommand)

TableAdapter object

Figure 14-21 Dataset Designer opened

Reviewing the TableAdapter's Command Properties



Properties

StudentTableAdapter TableAdapter

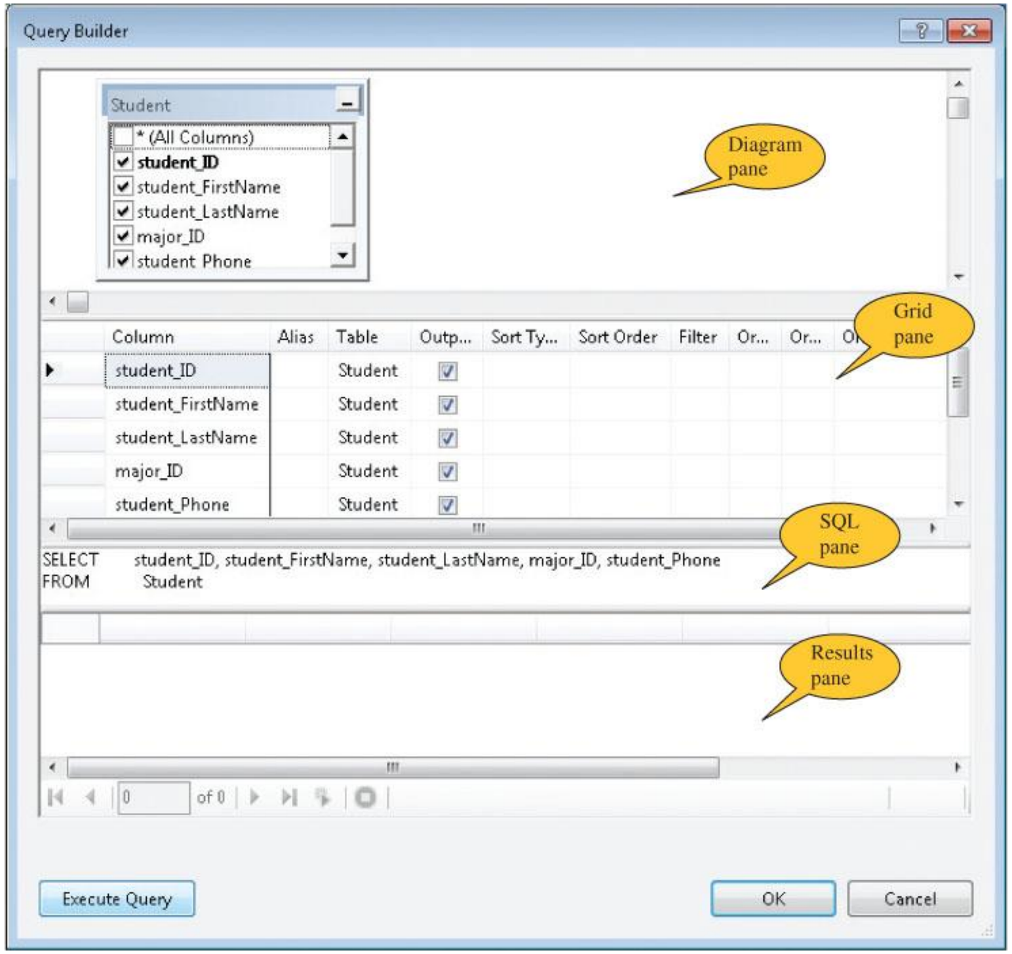
⊟ Connection	StudentDataBaseConnectionString (Settings)
Name	StudentDataBaseConnectionString (Settings)
Modifier	Internal
ConnectionString	Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\StudentDataBase.accdb
Provider	System.Data.OleDb
⊟ DeleteCommand	(DeleteCommand)
CommandText	DELETE FROM \`Student\` WHERE ((\`student_ID\` = ?) AND ((? = 1 AND \`student_FirstName\` IS NULL) OR (
CommandType	Text
Parameters	(Collection)
GenerateDBDirectMethods	True
⊟ InsertCommand	(InsertCommand)
CommandText	INSERT INTO \`Student\` (\`student_ID\`, \`student_FirstName\`, \`student_LastName\`, \`major_ID\`, \`student_P
CommandType	Text
Parameters	(Collection)
⊟ SelectCommand	(SelectCommand)
CommandText	SELECT student_ID, student_FirstName, student_LastName, major_ID, student_Phone FROM Student
CommandType	Text
Parameters	(Collection)
⊟ UpdateCommand	(UpdateCommand)
CommandText	UPDATE \`Student\` SET \`student_ID\` = ?, \`student_FirstName\` = ?, \`student_LastName\` = ?, \`major_ID\` =
CommandType	Text
Parameters	(Collection)

Figure 14-22 Updating the SelectCommand

Query Builder (continued)

- CommandText property holds the SQL statement
- Open the **Query Builder** by clicking the CommandText value box (**. . .)**
  - First prompted to select the table
  - Can type the SQL statement into the **SQL pane** or use the **Diagram pane** to select columns you want to update
  - **Grid pane** in the center can be used to filter and enter parameterized expressions
  - **Results pane** can be used for testing query (Located at bottom of the Query Builder)

Query Builder (continued)



Column	Alias	Table	Outp...	Sort Ty...	Sort Order	Filter	Or...	Or...	Or...
student_ID		Student	✓						
student_FirstName		Student	✓						
student_LastName		Student	✓						
major_ID		Student	✓						
student_Phone		Student	✓						

```
SELECT     student_ID, student_FirstName, student_LastName, major_ID, student_Phone
FROM        Student
```

Figure 14-23 Identify the Table for the Update

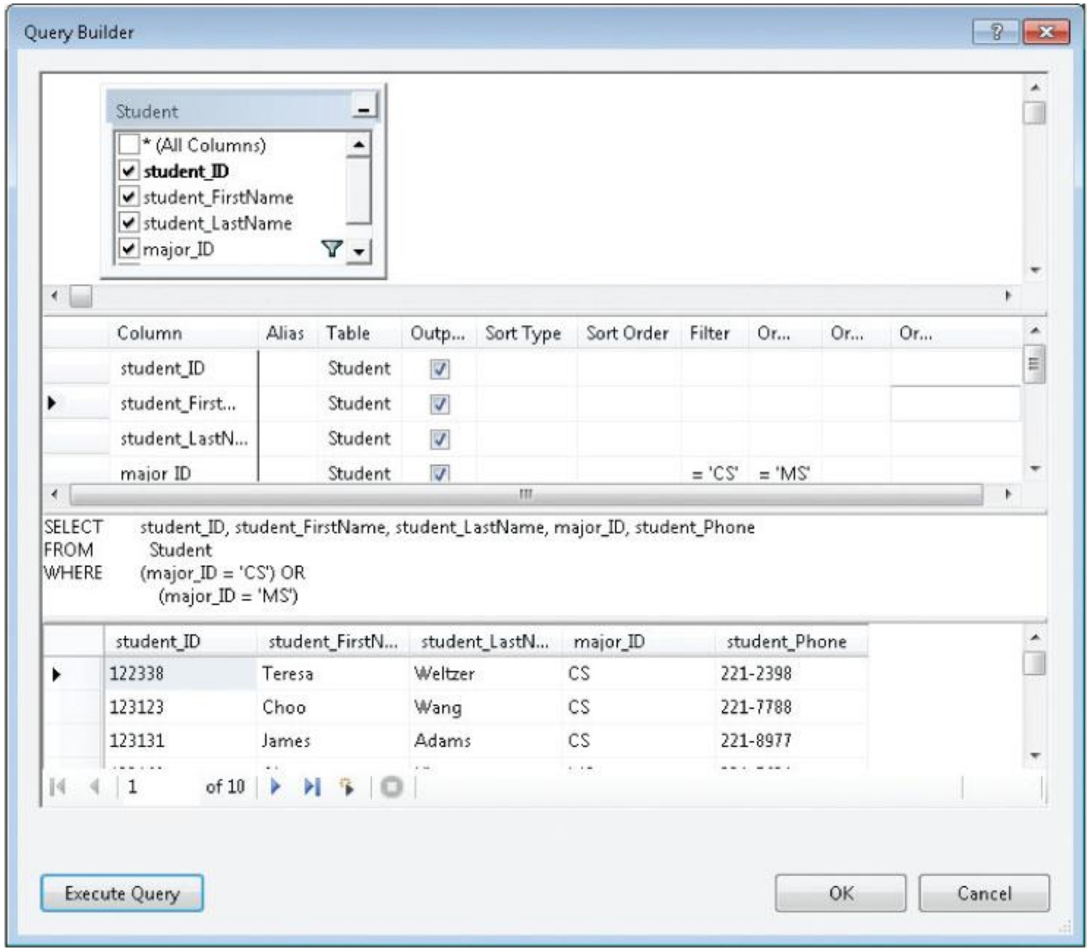
Parameters

- Parameters
  - Values provided at run time
- Special Symbol indicates insertion point
  - SQL Server – (@) is placed in front of an identifier
    - Example

```
DELETE FROM Student
    WHERE (student_ID = @student_ID)
```

  - Access – a question mark symbol (?) is used
    - No identifier can follow the ? symbol with Access
- OLE DB and ODBC Data Providers do not support named parameters

Query Builder



Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
student_ID		Student	✓						
student_First...		Student	✓						
student_LastN...		Student	✓						
major_ID		Student	✓			= 'CS'	= 'MS'		

```
SELECT      student_ID, student_FirstName, student_LastName, major_ID, student_Phone
FROM         Student
WHERE      (major_ID = 'CS') OR
               (major_ID = 'MS')
```

student_ID	student_FirstN...	student_LastN...	major_ID	student_Phone
122338	Teresa	Weltzer	CS	221-2398
123123	Choo	Wang	CS	221-7788
123131	James	Adams	CS	221-8977

Figure 14-24 CommandText property value for UpdateCommand

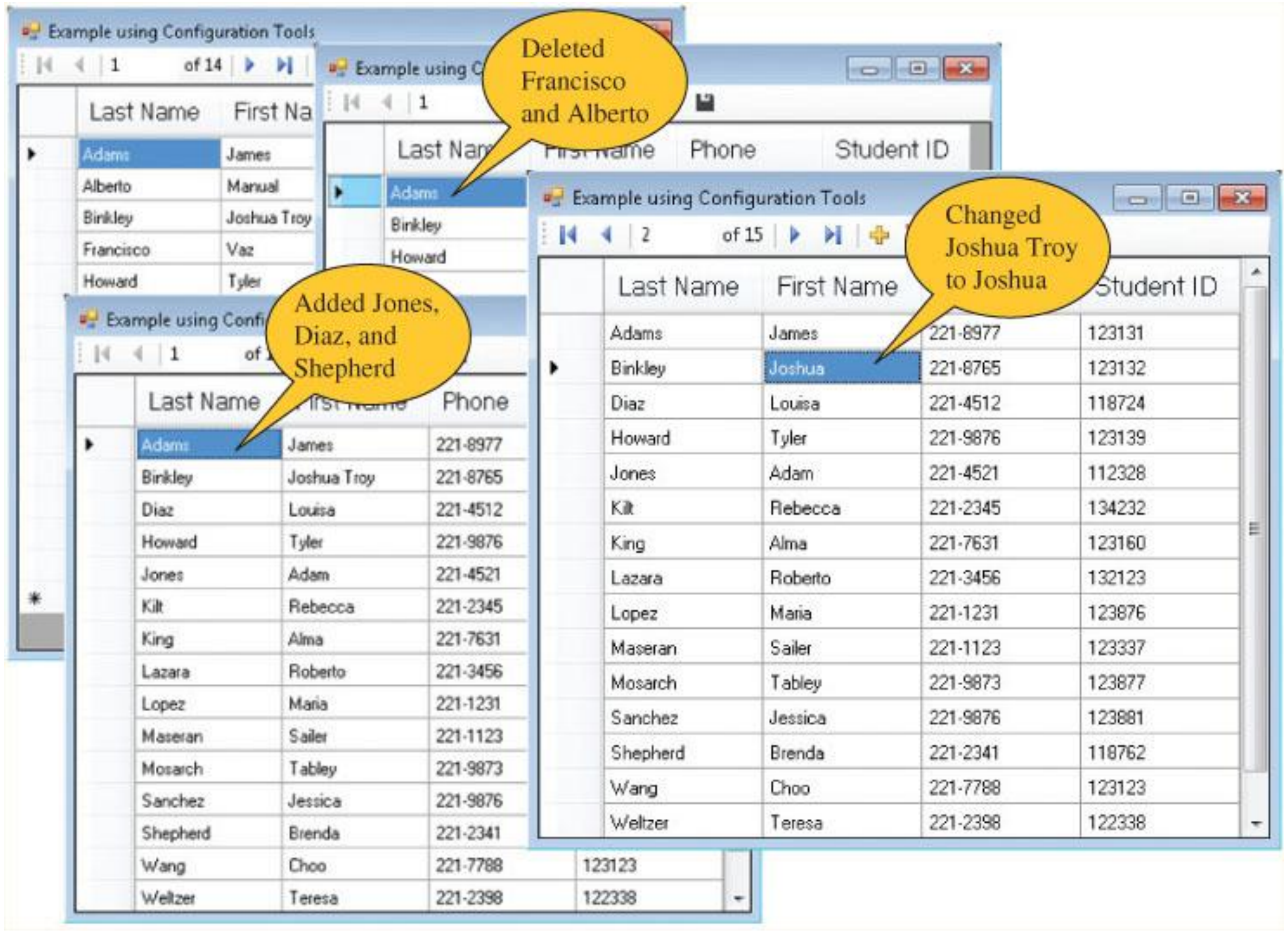
Query Builder (continued)

```
SELECT student_ID, student_FirstName, student_LastName,
                    major_ID, student_Phone
FROM Student WHERE (major_ID = 'CS') OR (major_ID = 'MS')
```

- Inside the Query Builder, test SQL statements by selecting the Execute Query button
  - If there are parameterized values, a dialog box is displayed requesting values for the arguments

Review ConfigToolsExample Example

Query Builder (continued)


Example using Configuration Tools
1 of 14
Deleted Francisco and Alberto
Added Jones, Diaz, and Shepherd
Changed Joshua Troy to Joshua
2 of 15

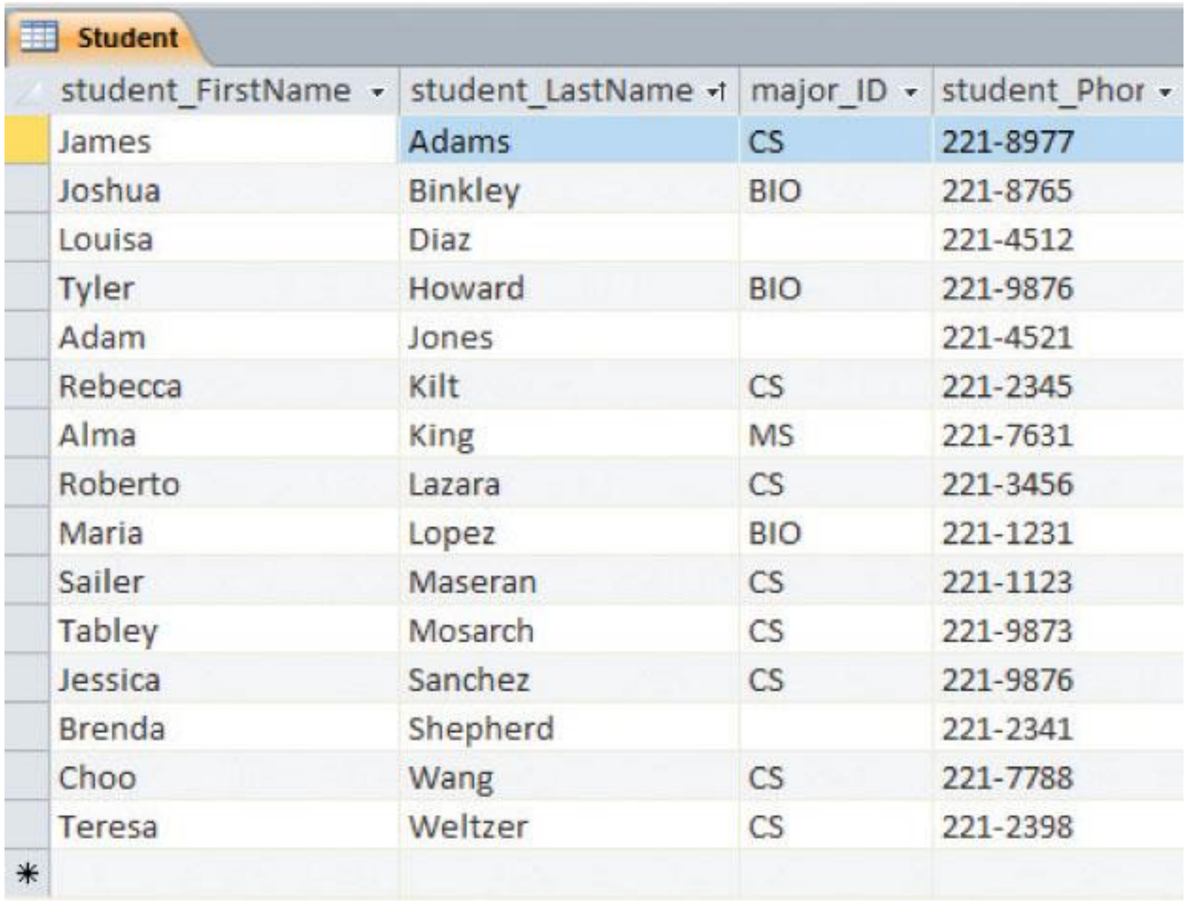
Last Name	First Na
Adams	James
Alberto	Manual
Binkley	Joshua Troy
Francisco	Vaz
Howard	Tyler

Last Name	First Name	Phone	
Adams	James	221-8977	
Binkley	Joshua Troy	221-8765	
Diaz	Louisa	221-4512	
Howard	Tyler	221-9876	
Jones	Adam	221-4521	
Kilt	Rebecca	221-2345	
King	Alma	221-7631	
Lazara	Roberto	221-3456	
Lopez	Maria	221-1231	
Maseran	Sailer	221-1123	
Mosarch	Tabley	221-9873	
Sanchez	Jessica	221-9876	
Shepherd	Brenda	221-2341	
Wang	Choo	221-7788	123123
Weltzer	Teresa	221-2398	122338

Last Name	First Name		Student ID
Adams	James	221-8977	123131
Binkley	Joshua	221-8765	123132
Diaz	Louisa	221-4512	118724
Howard	Tyler	221-9876	123139
Jones	Adam	221-4521	112328
Kilt	Rebecca	221-2345	134232
King	Alma	221-7631	123160
Lazara	Roberto	221-3456	132123
Lopez	Maria	221-1231	123876
Maseran	Sailer	221-1123	123337
Mosarch	Tabley	221-9873	123877
Sanchez	Jessica	221-9876	123881
Shepherd	Brenda	221-2341	118762
Wang	Choo	221-7788	123123
Weltzer	Teresa	221-2398	122338

Figure 14-25 Example using Configuration Tools final output

Query Builder (continued)



Student

student_FirstName	student_LastName	major_ID	student_Phor
James	Adams	CS	221-8977
Joshua	Binkley	BIO	221-8765
Louisa	Diaz		221-4512
Tyler	Howard	BIO	221-9876
Adam	Jones		221-4521
Rebecca	Kilt	CS	221-2345
Alma	King	MS	221-7631
Roberto	Lazara	CS	221-3456
Maria	Lopez	BIO	221-1231
Sailer	Maseran	CS	221-1123
Tabley	Mosarch	CS	221-9873
Jessica	Sanchez	CS	221-9876
Brenda	Shepherd		221-2341
Choo	Wang	CS	221-7788
Teresa	Weltzer	CS	221-2398

Figure 14-26 StudentDataBase Student table contents (from bin\Debug directory)

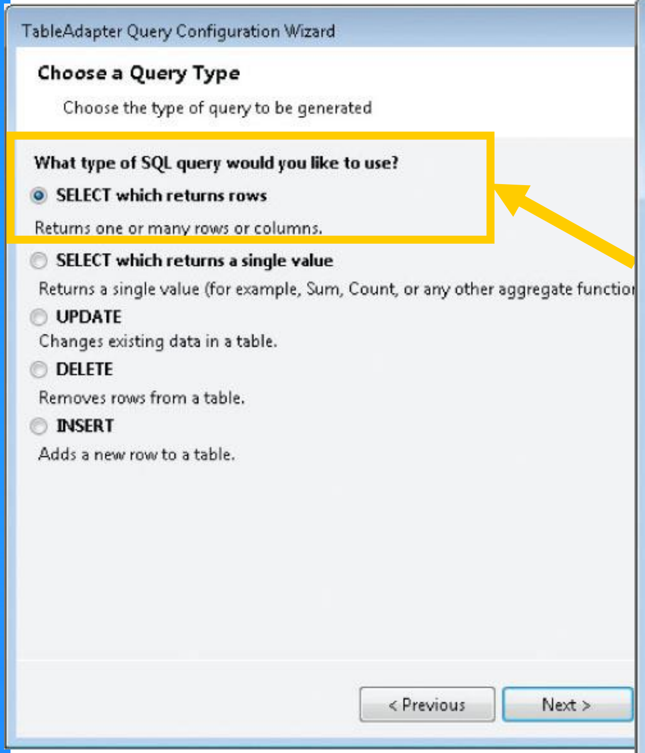
Adding Queries to TableAdapter Objects

- TableAdapters has Fill( ) and Update( ) methods to retrieve and update data in a database
- Other queries can be added as methods, called like regular methods
  - This is the added benefit TableAdapters offer over DataAdapters
  - Use **DataSet Designer** to add the additional queries (methods)
  - Have the option of naming these methods
    - Methods are automatically named FillBy and GetDataBy
  - SQL Select statement generated along with the Fill and Get methods

Adding Queries to TableAdapter Objects (continued)

- Use **DataSet Designer** window to add the additional queries
  - Right-click TableAdapter in the **DataSet Designer** window
  - Select **Add Query** from the pop-up menu
    - TableAdapter Query Configuration tool is displayed
    - Prompt reads *"How should the TableAdapter query access the database?"*
      - Select **Use SQL statement**
      - TableAdapter Query Configuration tool wizard launched

Adding Queries to TableAdapter Objects (continued)


TableAdapter Query Configuration Wizard

Choose a Query Type

Choose the type of query to be generated

What type of SQL query would you like to use?

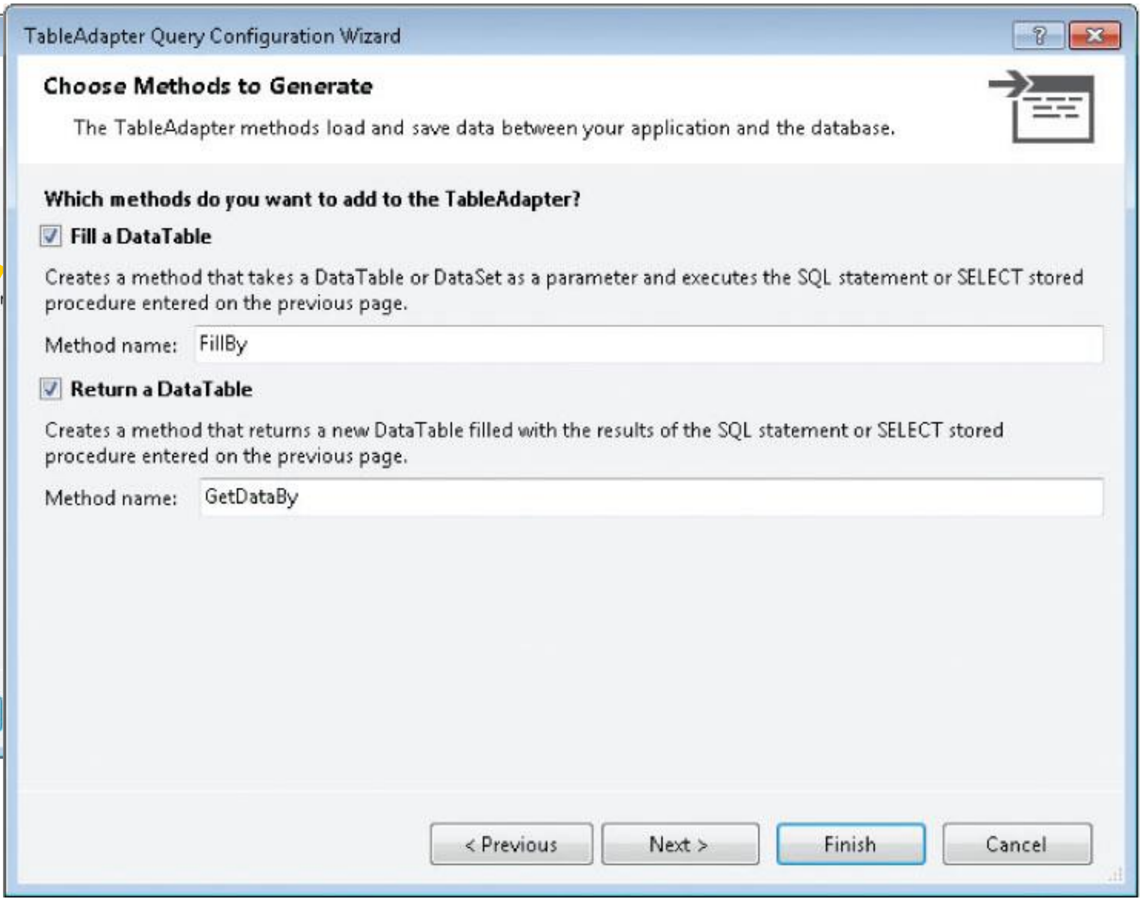
[✓ SELECT which returns rows]
Returns one or many rows or columns.

SELECT which returns a single value
Returns a single value (for example, Sum, Count, or any other aggregate functio

UPDATE
Changes existing data in a table.

DELETE
Removes rows from a table.

INSERT
Adds a new row to a table.


TableAdapter Query Configuration Wizard

Choose Methods to Generate

The TableAdapter methods load and save data between your application and the database.

Which methods do you want to add to the TableAdapter?

✓ **Fill a DataTable**

Creates a method that takes a DataTable or DataSet as a parameter and executes the SQL statement or SELECT stored procedure entered on the previous page.

Method name: FillBy

✓ **Return a DataTable**

Creates a method that returns a new DataTable filled with the results of the SQL statement or SELECT stored procedure entered on the previous page.

Method name: GetDataBy

Figure 14-27 Multiple Queries with the TableAdapter

Figure 14-28 Naming the new query methods

Adding Queries to TableAdapter Objects (continued)

- To simply return values for display make selection "SELECT which returns rows"
- To retrieve rows based on input values, like user's last name, you could add a parameterized query using the WHERE clause

```
SELECT student_ID, student_FirstName, student_LastName,
                student_Phone
FROM Student WHERE (student_LastName = ?)
```

Add a Button and Textbox for the New Queries

- Buttons to execute the new TableAdapter queries can be added to the navigational tool strip
- Click on the navigational tool strip to the right of the Save button; a new button appears
  - ToolStripButton
    - Button enables you to add additional controls
  - Could also add text box for user input
    - ToolStripTextBox

Add a Button and Textbox for the New Queries (continued)

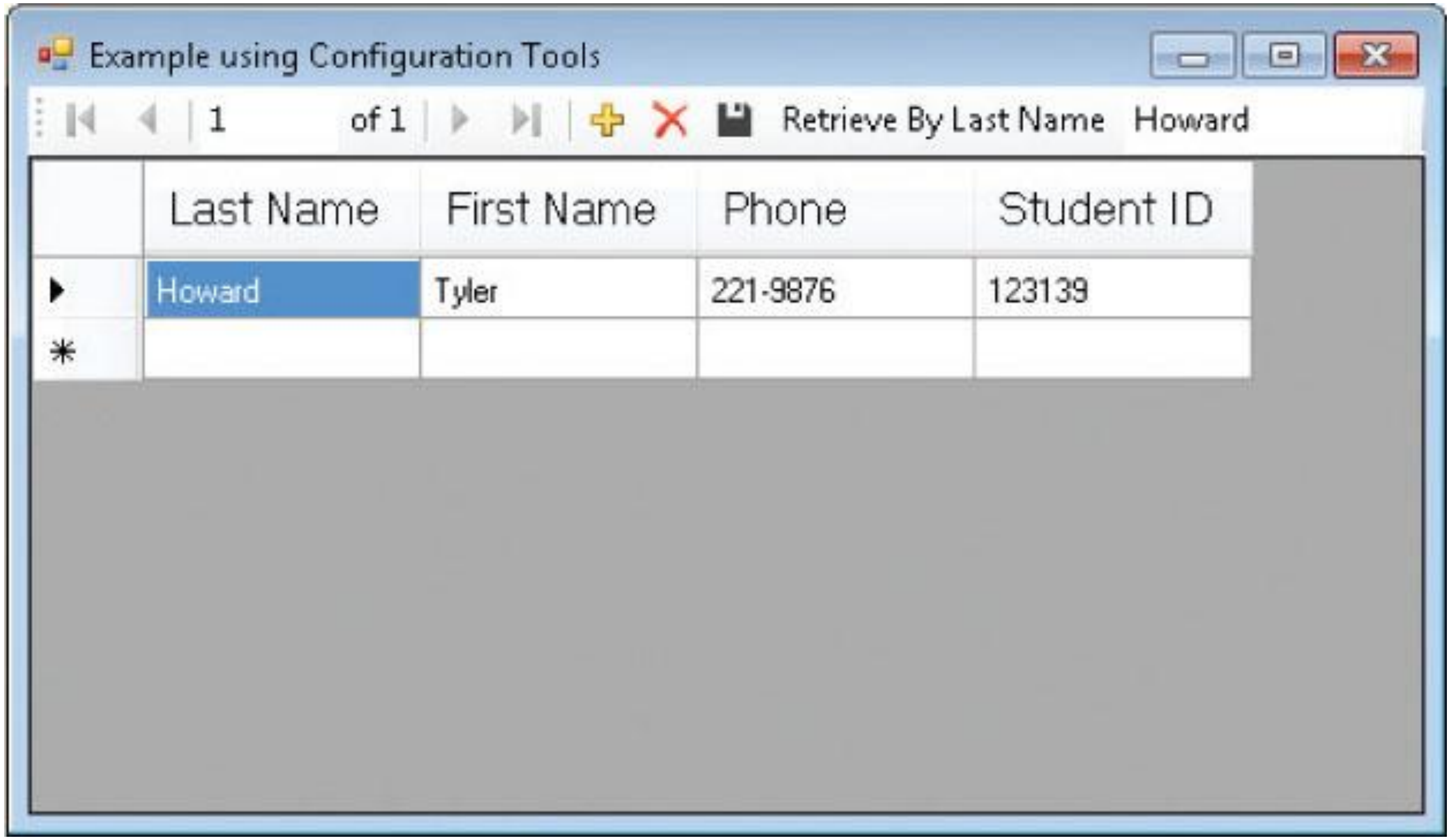
- Double-click on ToolStripButton button to create event-handler method

```
 private void btnRetrieve_Click( object sender, EventArgs e )
  {
            studentTableAdapter.FillByLastName
                  (studentDataBaseDataSet.Student, txbxLastName.Text);
  }
```

- Value entered in text box retrieved and used as a parameter to the query's SQL statement

Review ConfigToolsExampleWithQuery Example

Adding Queries to TableAdapter Objects (continued)


Example using Configuration Tools
1 of 1
Retrieve By Last Name Howard

	Last Name	First Name	Phone	Student ID
▶	Howard	Tyler	221-9876	123139
*				

Figure 14-29 TableAdapter's Query

Connecting Multiple Tables

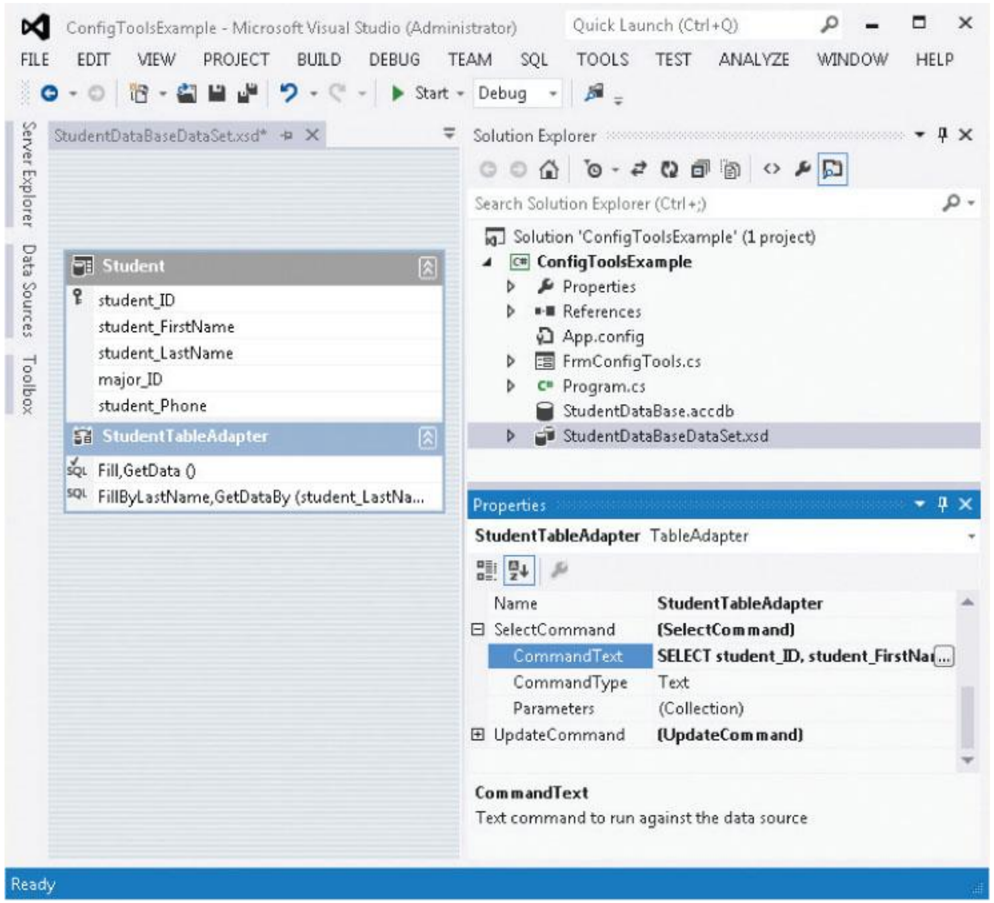
- Best to select all of the tables that you will need originally when you create the dataset object
  - Without regenerating the dataset, several options
    - Use **Query Builder** and add INNER JOIN to SELECT statement for the TableAdapter's SelectCommand
      - Use the graphical capabilities of the tool on **Diagram Pane**, or you can type the SQL statement into SQL pane
    - Use the **DataSet Designer**
      - Double-click on the dataset file
        - » **DataSet Designer** opens the DataSet and TableAdapter objects graphically displayed as a single unit

Use the DataSet Designer to Connect Multiple Tables

- Change the TableAdapter CommandText for the SelectCommand so when the Fill( ) method is called, dataset is populated with results from both tables
- Call the TableAdapter's Fill( ) method in the page load event handler

```
this.studentTableAdapter.Fill(this.studentDataBaseDataSet.Student);
```

Use the DataSet Designer (continued)

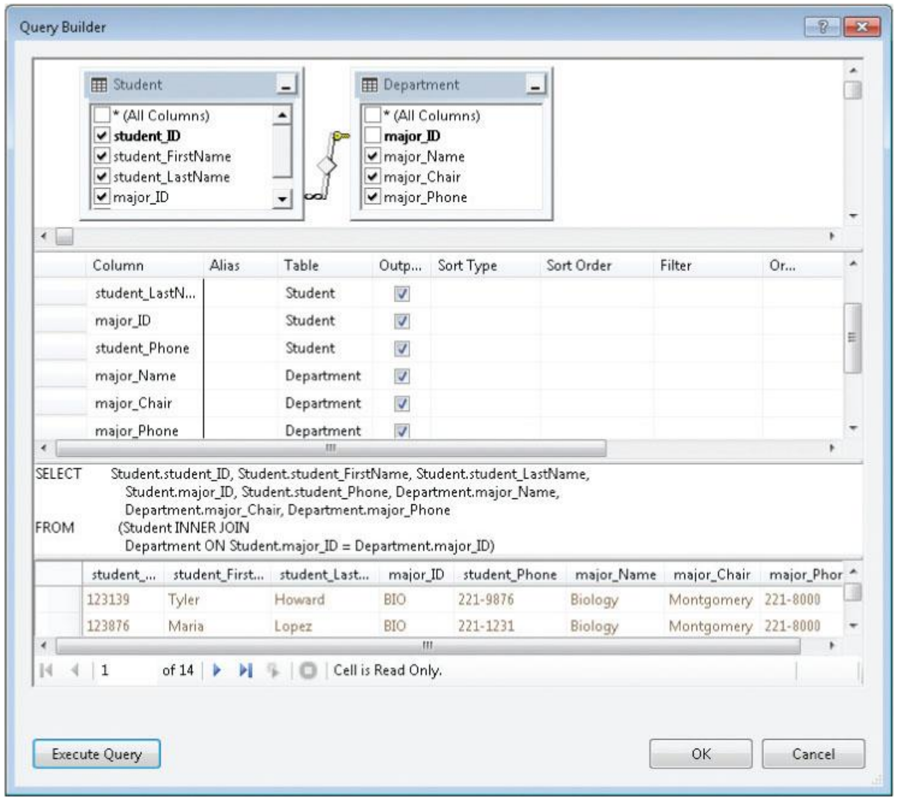

ConfigToolsExample - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Start Debug
StudentDataBaseDataSet.xsd*
Server Explorer
Data Sources
Toolbox
Student
student_ID
student_FirstName
student_LastName
major_ID
student_Phone
StudentTableAdapter
Fill,GetData ()
FillByLastName,GetDataBy (student_LastNa...
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ConfigToolsExample' (1 project)
ConfigToolsExample
Properties
References
App.config
FrmConfigTools.cs
Program.cs
StudentDataBase.accdb
StudentDataBaseDataSet.xsd
Properties
StudentTableAdapter TableAdapter
Ready

Name	StudentTableAdapter
⊟ SelectCommand	(SelectCommand)
CommandText	SELECT student_ID, student_FirstNa
CommandType	Text
Parameters	(Collection)
⊞ UpdateCommand	(UpdateCommand)

CommandText
Text command to run against the data source

Figure 14-30 Revise the CommandText for the SelectCommand

Modify the SelectCommand Using the Query Builder



Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...
student_LastN...		Student	✓				
major_ID		Student	✓				
student_Phone		Student	✓				
major_Name		Department	✓				
major_Chair		Department	✓				
major_Phone		Department	✓				

```
SELECT     Student.student_ID, Student.student_FirstName, Student.student_LastName,
             Student.major_ID, Student.student_Phone, Department.major_Name,
             Department.major_Chair, Department.major_Phone
FROM       (Student INNER JOIN
             Department ON Student.major_ID = Department.major_ID)
```

student_...	student_First...	student_Last...	major_ID	student_Phone	major_Name	major_Chair	major_Phor
123139	Tyler	Howard	BIO	221-9876	Biology	Montgomery	221-8000
123876	Maria	Lopez	BIO	221-1231	Biology	Montgomery	221-8000

1 of 14 Cell is Read Only.

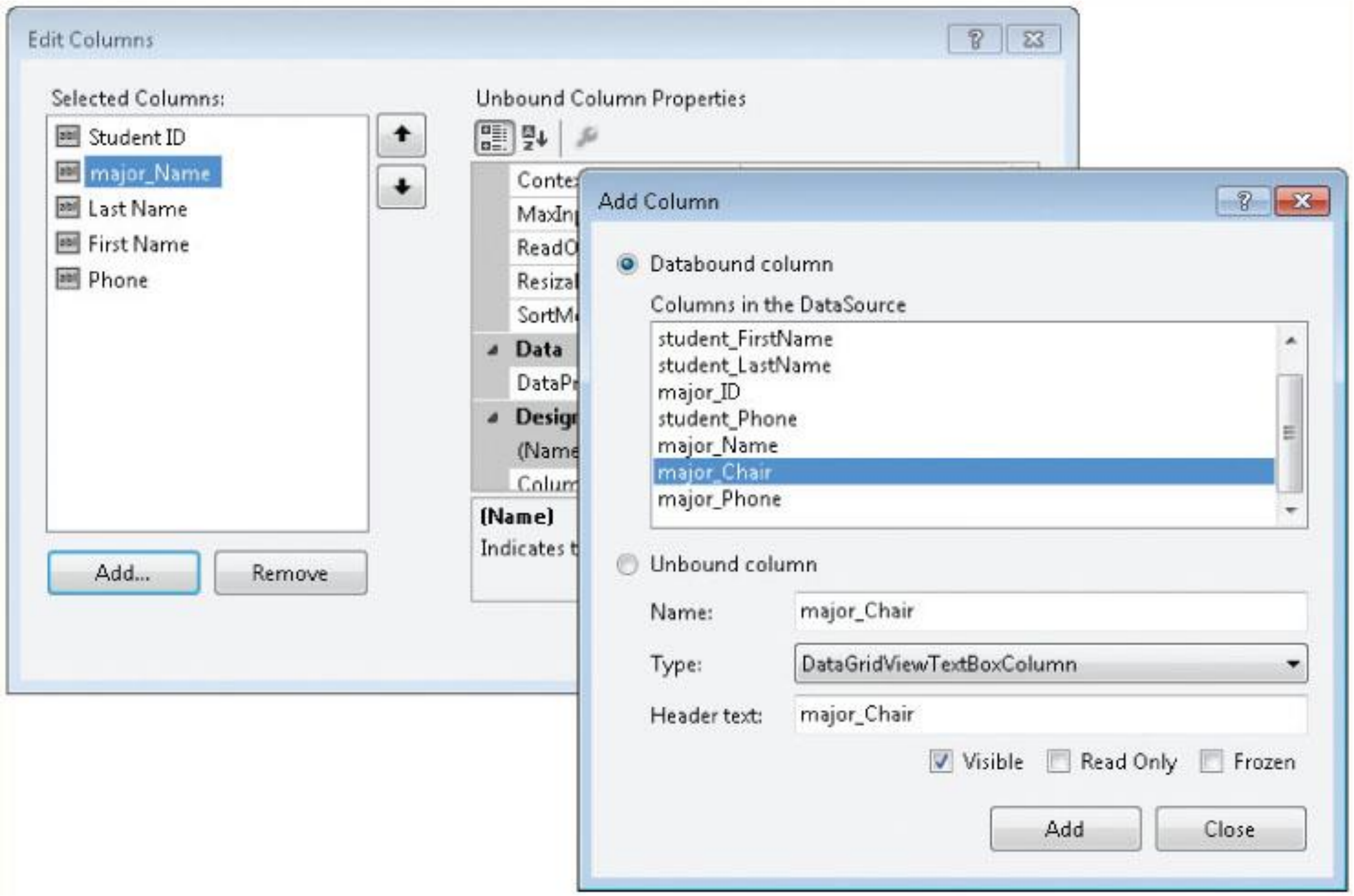
Figure 14-31 Use the Query Builder to modify the SelectCommand CommandText

Modify the SelectCommand to Connect Multiple Tables Using the Query Builder

```
SELECT     student_ID, student_FirstName, student_LastName, major_ID,
    student_Phone, major_Name, major_Chair, major_Phone
FROM   Student
    INNER JOIN Department ON Student.major_ID = Department.major_ID
```

- Once the relationship is established between the tables, add columns from the second table to the data grid
  - Do this by selecting the data grid's smart tag in the form design mode

Updating the Windows Form



Edit Columns

Selected Columns:

- Student ID
- major_Name
- Last Name
- First Name
- Phone

Unbound Column Properties

Add Column

[✓ Databound column]

Columns in the DataSource

- student_FirstName
- student_LastName
- major_ID
- student_Phone
- major_Name
- major_Chair
- major_Phone

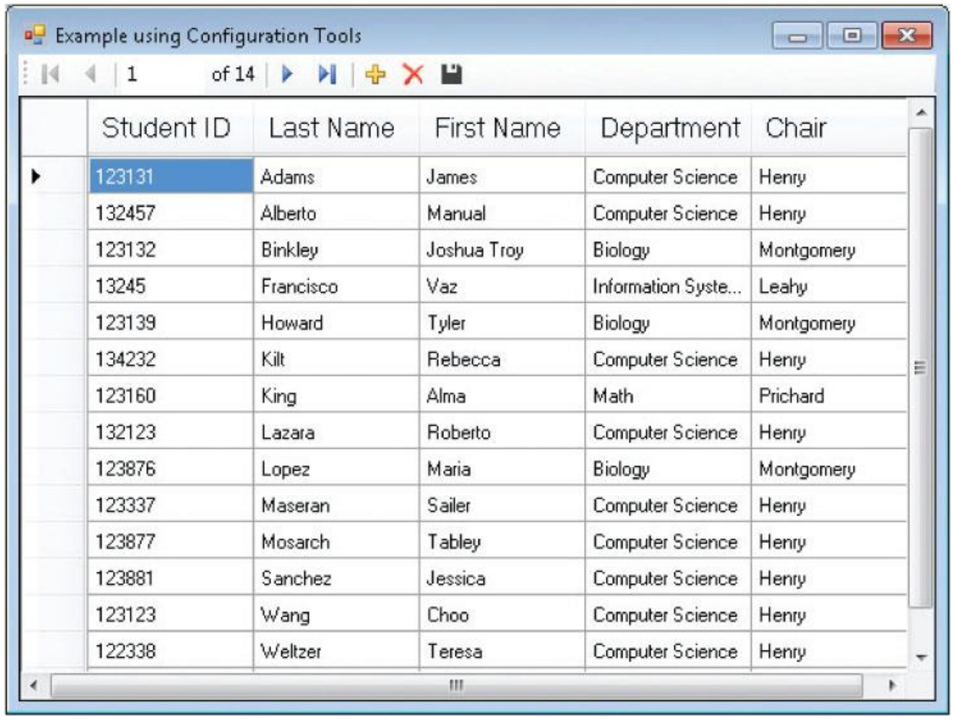
Unbound column

Name:	major_Chair
Type:	DataGridViewTextBoxColumn
Header text:	major_Chair

[✓ Visible] Read Only Frozen

Figure 14-32 Adding fields from the second table

Updating the Windows Form (continued)



	Student ID	Last Name	First Name	Department	Chair
▶	123131	Adams	James	Computer Science	Henry
	132457	Alberto	Manual	Computer Science	Henry
	123132	Binkley	Joshua Troy	Biology	Montgomery
	13245	Francisco	Vaz	Information Syste...	Leahy
	123139	Howard	Tyler	Biology	Montgomery
	134232	Kilt	Rebecca	Computer Science	Henry
	123160	King	Alma	Math	Prichard
	132123	Lazara	Roberto	Computer Science	Henry
	123876	Lopez	Maria	Biology	Montgomery
	123337	Maseran	Sailer	Computer Science	Henry
	123877	Mosarch	Tabley	Computer Science	Henry
	123881	Sanchez	Jessica	Computer Science	Henry
	123123	Wang	Choo	Computer Science	Henry
	122338	Weltzer	Teresa	Computer Science	Henry

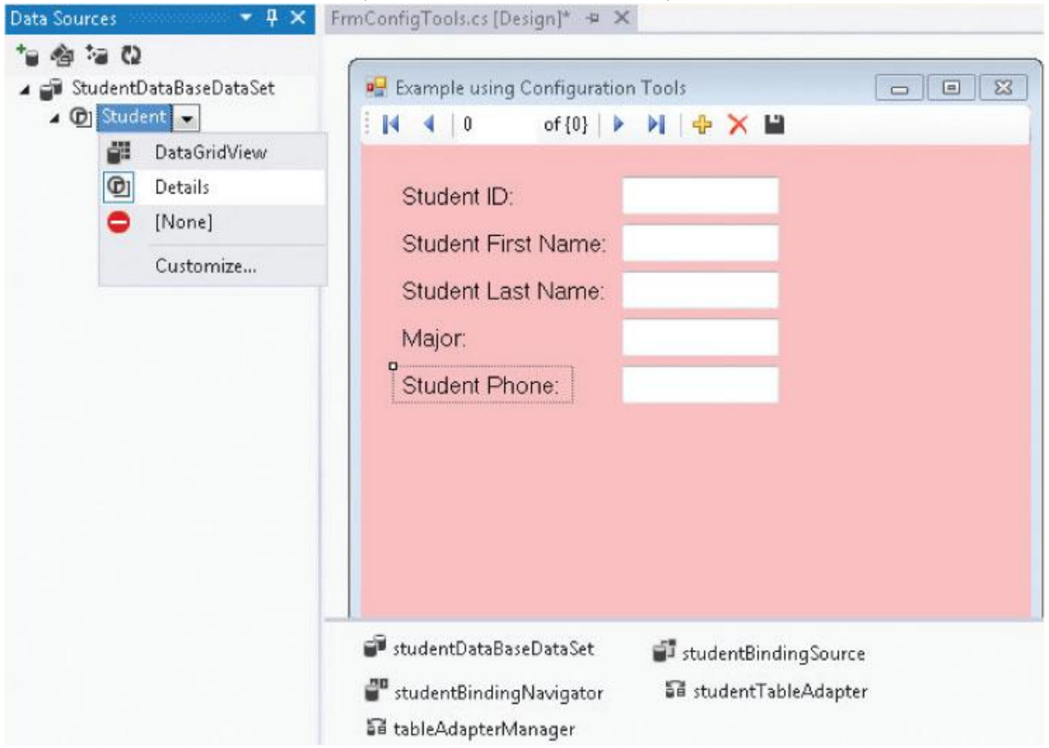
Figure 14-33 Data retrieved from multiple tables

Review ConfigToolsExampleWithMultipleTables Example

Display Data Using Details View

- Instead of displaying data in gridline view, Details view (labels and textboxes) available
- From **Data Sources** window
  - Use pull-down menu and select **Details**
  - Drag the entire table onto the form
    - You get Label and TextBox objects for each column in the dataset
      - Label is the column identifier with spaces replacing underscores
        - » Change its Text property from the **Properties** window

Display Data Using Details View (continued)



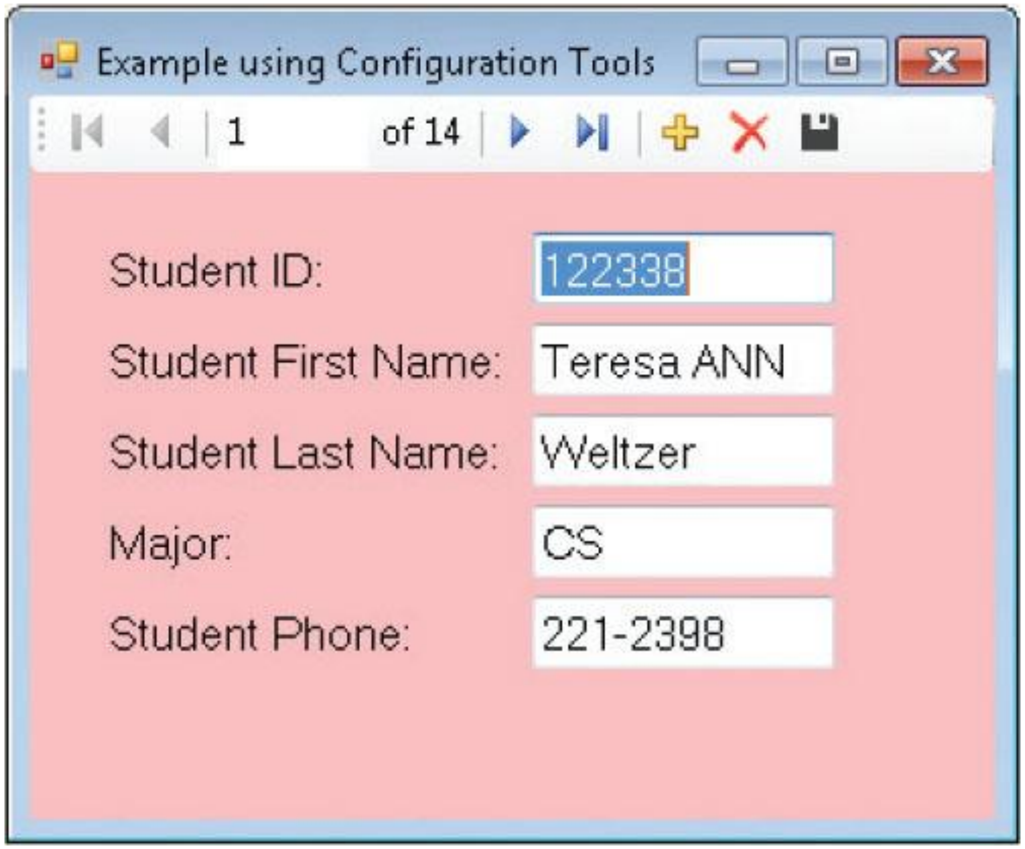
Student ID:
Student First Name:
Student Last Name:
Major:
Student Phone:

Figure 14-34 Details view

Review ConfigToolsExampleDetailView Example

Adding Controls from the Toolbox

- Can drag controls from **Toolbox** as opposed to using **Data Sources** window
  - Set DataSource and DisplayMember properties



Student ID: 122338
Student First Name: Teresa ANN
Student Last Name: Weltzer
Major: CS
Student Phone: 221-2398

Figure 14-35 Output from ConfigToolsExampleDetailView

Modifying the Data-Bound Controls

- Click on individual columns in the Data Sources window to change default-bound control to a ComboBox, Label, LinkLabel, or ListBox
  – Or customize the data-bound control
- If you select controls from Toolbox (as opposed to from Data Sources windows), you have to set DataSource and DisplayMember properties
  – DataSource -> name of the dataset table object
  – DisplayMember -> name the table’s column object

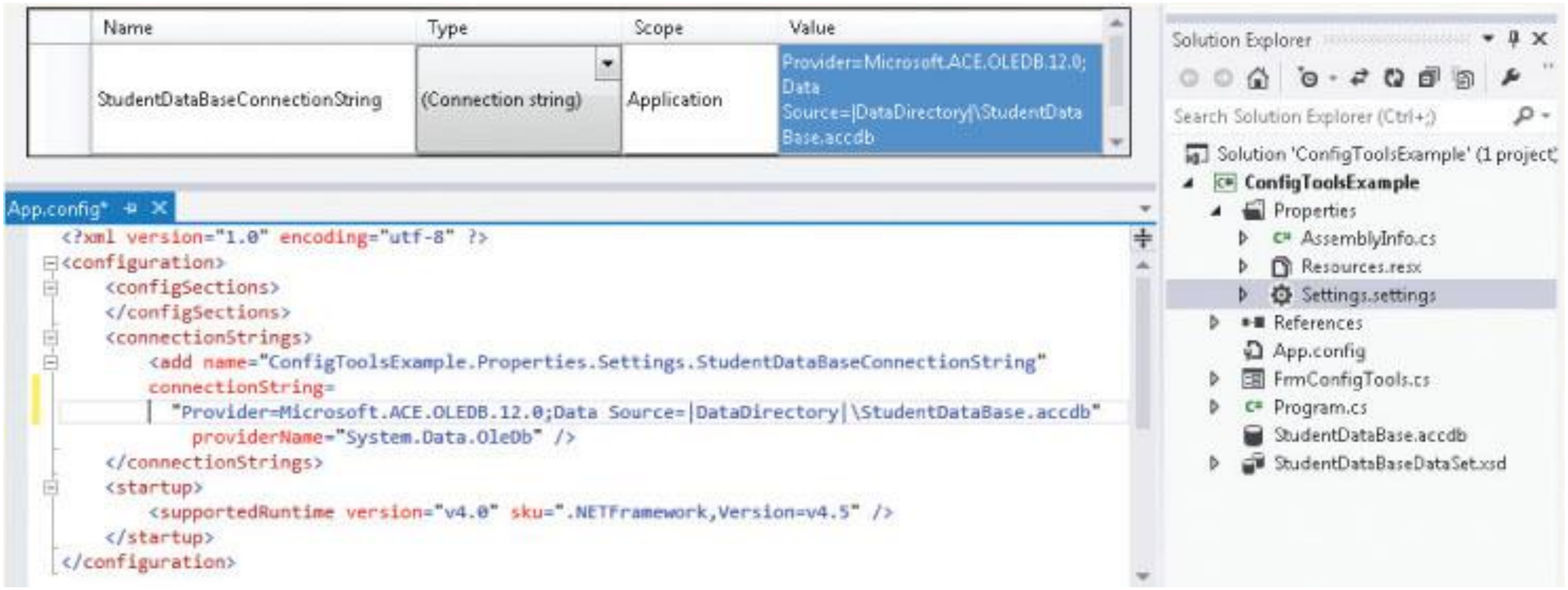
ConfigToolsExample

- No program statements were written for these applications
  - Data Source Configuration tool used to identify the data source tables
  - Data-bound controls placed on the form from the **Data Sources** window
  - DataSet Designer used to create the relation between the tables
  - TableAdapter populated Dataset objects
  - Properties were changed

Modifying Connection Strings

- Several options
  - Change the XML app.config file when the connection string is saved with the application
  - Use the Settings page of the Property Designer to modify the project's settings
    - Access this from Solution Explorer window
      - Settings.settings file

Modifying Connection Strings (continued)



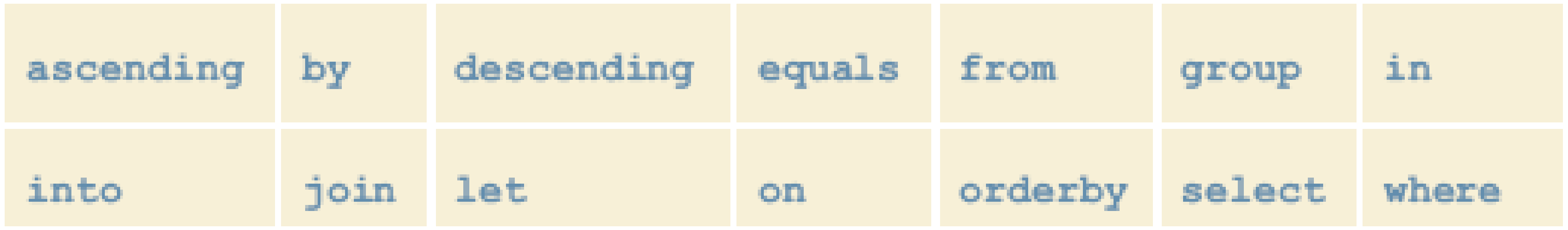
Name	Type	Scope	Value
StudentDataBaseConnectionString	(Connection string)	Application	Provider=Microsoft.ACE.OLEDB.12.0; Data Source=|DataDirectory|\StudentDataBase.accdb

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
    </configSections>
    <connectionStrings>
        <add name="ConfigToolsExample.Properties.Settings.StudentDataBaseConnectionString"
        connectionString=
          "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\StudentDataBase.accdb"
            providerName="System.Data.OleDb" />
    </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
</configuration>
```

Figure 14-36 Modifying Settings.settings and App.config files

Language-Integrated Query (LINQ)

- Standard query operators defined in System.Linq namespace enable select, filter, aggregate, and partition of data
- Used with relational data sources, XML data, and any class that implement IEnumerable interface
  - IEnumerable supports iteration over a collection



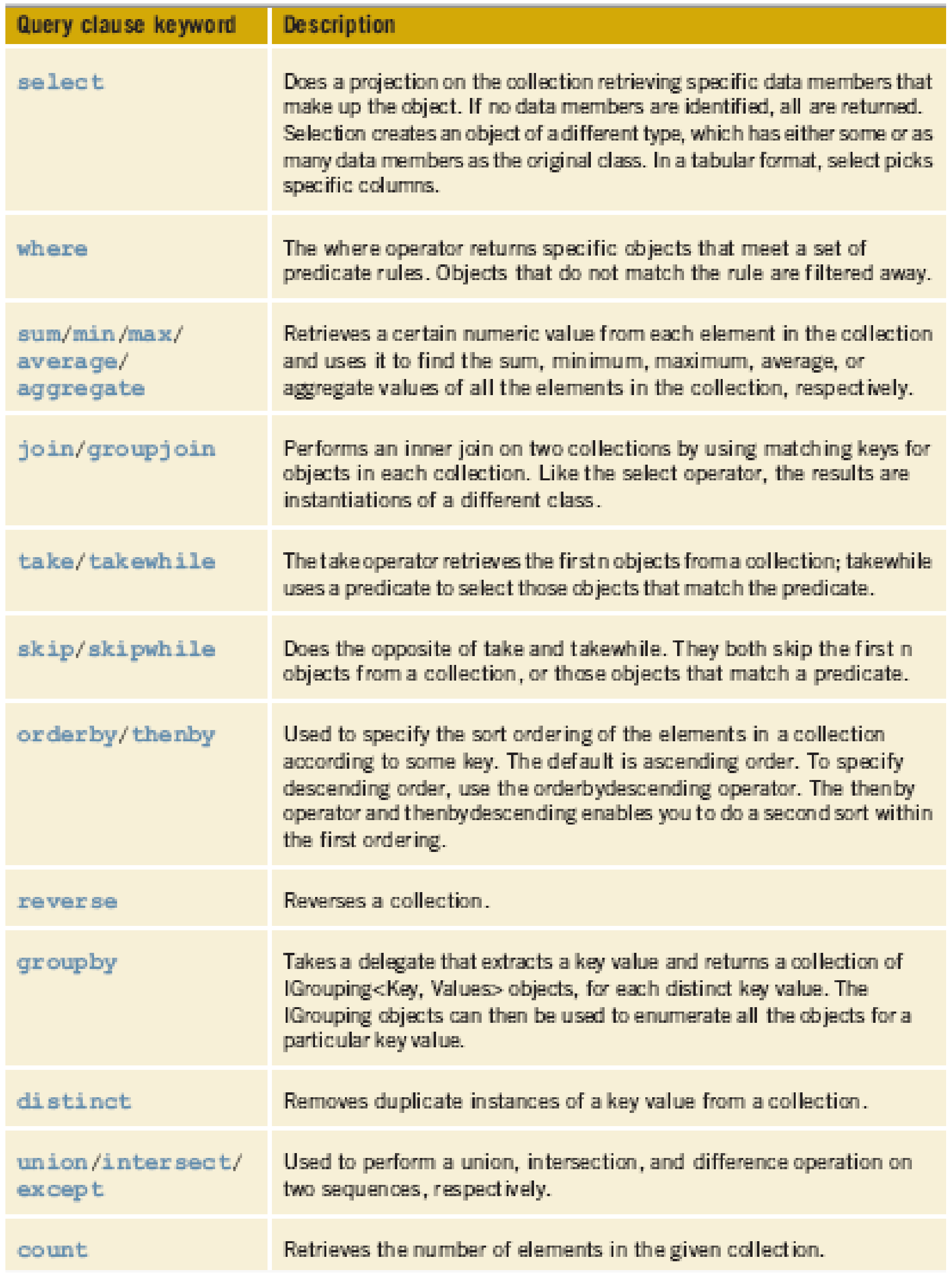
ascending	by	descending	equals	from	group	in
into	join	let	on	orderby	select	where

Table 14-8 Query Contextual Keywords

Query Expressions

- Most query expressions begin with from and end with either select or group clause
  - Each from identifies data source and a range variable
    - Range variable similar to iteration variable with foreach
- Can add where clause to filter or exclude items
- Additional operators can be added to expression

Query Expressions (continued)



Query clause keyword	Description
`select`	Does a projection on the collection retrieving specific data members that make up the object. If no data members are identified, all are returned. Selection creates an object of a different type, which has either some or as many data members as the original class. In a tabular format, select picks specific columns.
`where`	The where operator returns specific objects that meet a set of predicate rules. Objects that do not match the rule are filtered away.
`sum/min/max/ average/ aggregate`	Retrieves a certain numeric value from each element in the collection and uses it to find the sum, minimum, maximum, average, or aggregate values of all the elements in the collection, respectively.
`join/groupjoin`	Performs an inner join on two collections by using matching keys for objects in each collection. Like the select operator, the results are instantiations of a different class.
`take/takewhile`	The take operator retrieves the first n objects from a collection; takewhile uses a predicate to select those objects that match the predicate.
`skip/skipwhile`	Does the opposite of take and takewhile. They both skip the first n objects from a collection, or those objects that match a predicate.
`orderby/thenby`	Used to specify the sort ordering of the elements in a collection according to some key. The default is ascending order. To specify descending order, use the orderbydescending operator. The thenby operator and thenbydescending enables you to do a second sort within the first ordering.
`reverse`	Reverses a collection.
`groupby`	Takes a delegate that extracts a key value and returns a collection of IGrouping<Key, Values> objects, for each distinct key value. The IGrouping objects can then be used to enumerate all the objects for a particular key value.
`distinct`	Removes duplicate instances of a key value from a collection.
`union/intersect/ except`	Used to perform a union, intersection, and difference operation on two sequences, respectively.
`count`	Retrieves the number of elements in the given collection.

Table 14-9 Some of the LINQ query operators

Query Expressions

```
string[ ] nameArray = {"Wong", "Abi","Fredrick",
                       "Davis","Howard","Abbott",
                       "Fang","Erlanger","Halcomb",
                       "George","King","Doyle",
                       "Mitchell","Ralph","Barry"};
IEnumerable<string> queryResult = from aName in nameArray
                       where aName.Length > 5
                       orderby aName descending
                       select aName;
```

Review LinqArrayExample Example

Implicitly Typed Local Variables

- keyword var indicates type will be determined from the expression

```
var queryResult = from aName in nameArray
                  where aName.Length > 5
                  orderby aName descending
                  select aName;
```

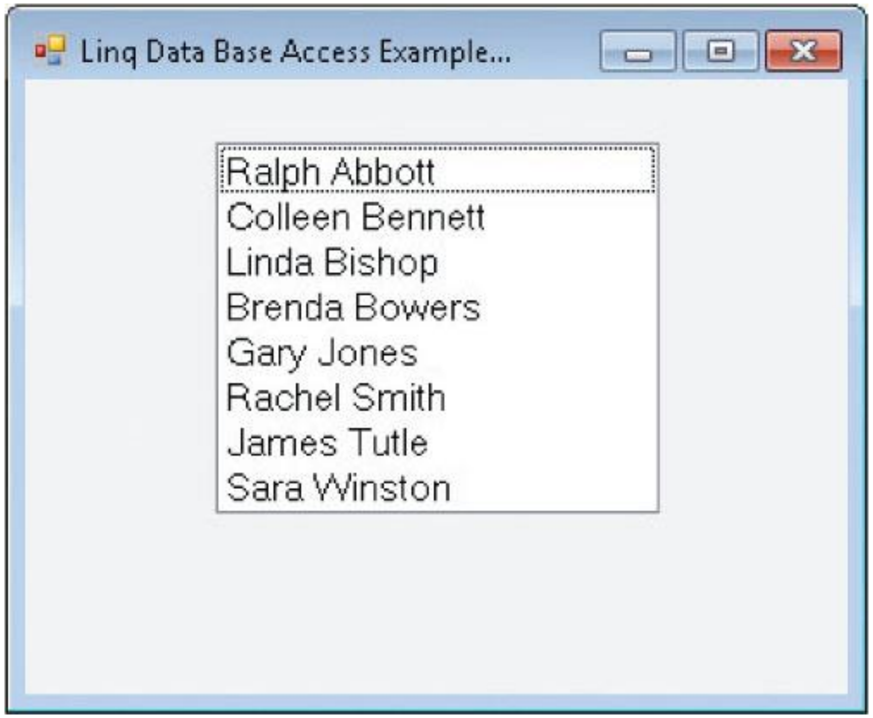
LINQ with Databases

- After connection made to data source, instead of embedding a SQL statement in string argument, include your query expression directly in your C# program

```
this.memberTableAdapter.Fill(this.memberDataSet.MemberTable);
var memberResults = from member in this.memberDataSet.MemberTable
                    where member.LastName.Length > 4
                    orderby member.LastName  select member;
foreach (var aRecord in memberResults)
                    this.lstBxResult.Items.Add(aRecord.FirstName  + " " +
                                                aRecord.LastName);
```

LINQ with Databases (continued)

- Query expression traverses through the table producing a list of items to populate the listbox



```
Ralph Abbott
Colleen Bennett
Linda Bishop
Brenda Bowers
Gary Jones
Rachel Smith
James Tutle
Sara Winston
```

Figure 14-37 LINQ database output

Review LinqDataBaseAccessExample Example

LINQ to SQL

- Used to query SQL Server databases
- Defines a mapping framework
  - Mapping defines classes that correspond to tables in database
- Dlinq is version of LINQ that focuses on querying from relational data sources
- XLinq is the aspect geared toward querying XML data

Coding Standards

- Database tables should be designed to have a primary key
  - Retrieve key as one of fields from your SQL query
- Use uppercase characters for SQL keywords
- Use primary key in the WHERE condition of an UPDATE or DELETE SQL statement to avoid errors
- Avoid using spaces within database names

Resources

Database Tutorials –

http://www.quackit.com/database/tutorial/

Access Tutorials –

http://databases.about.com/od/tutorials/Tutorials.htm

101 LINQ Samples –

http://msdn.microsoft.com/en-us/vcsharp/aa336746.aspx

LINQ to SQL: .NET Language-Integrated Query for Relational Data –

http://msdn.microsoft.com/en-us/library/bb425822.aspx

Chapter Summary

- ActiveX Data Object (ADO.NET) classes can be used to retrieve, manipulate, and update data in databases
- ADO.NET Data Providers
- Connect to the database
  - Connection string
- Programmatically access and update database

Chapter Summary (continued)

- Data reader class – forward read-only retrieval
  - Read( )
- Disconnected architecture
- SQL statements
- DataAdapter and TableAdapter
  - Fill( ) & Update( ) methods
- DataSet

Chapter Summary (continued)

- Configuration tools
  - Use of Server Explorer
    - Creation of New SQL Server database
  - Use of Data Sources window
    - Add connections
  - Use of DataSet Designer
    - Query Builder
- DataGridView Control

Chapter Summary (continued)

- LINQ
- Query Expressions
  - Contextual keywords
- LINQ with databases
- LINQ to SQL
- Implicitly typed local variables
  - var