

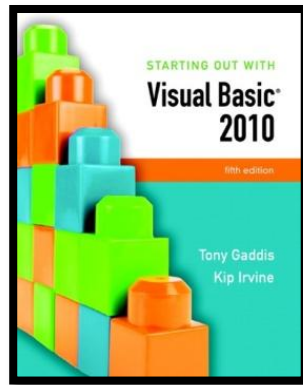


STARTING OUT WITH

Visual Basic® 2010

fifth edition

Tony Gaddis
Kip Irvine



Chapter 10

Working with Databases

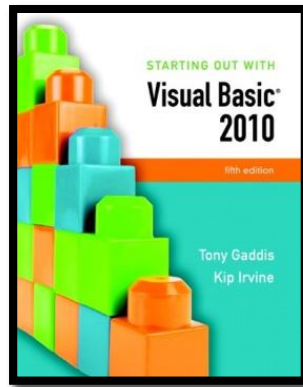
Addison Wesley
is an imprint of



© 2011 Pearson Addison-Wesley. All rights reserved.

Introduction

- In this chapter you will learn:
 - Basic database concepts
 - How to write Visual Basic applications that interact with databases
 - How to use a DataGridView control and display the data in a database
 - How to sort and update database data
 - To create an application that displays database data in list boxes, text boxes, labels, and combo boxes



Section 10.1

DATABASE MANAGEMENT SYSTEMS

Visual Basic applications use database management systems to make large amounts of data available to programs.

Addison Wesley
is an imprint of



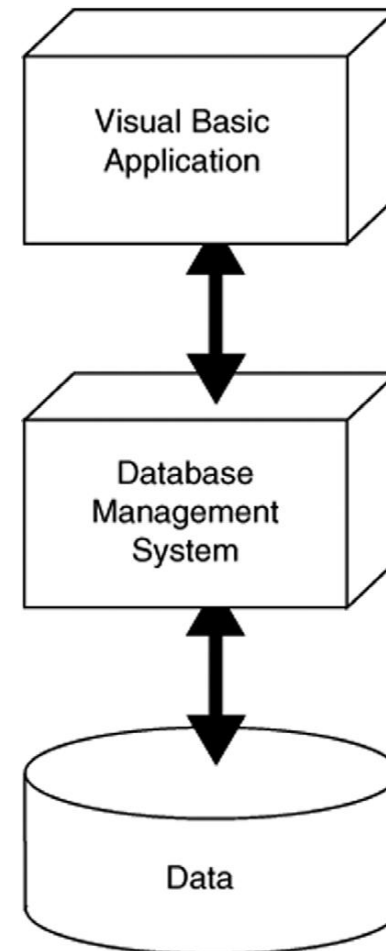
© 2011 Pearson Addison-Wesley. All rights reserved.

Visual Basic and Database Management Systems

- Simple text files as shown in chapter 9 are:
 - Fine for small amounts of data
 - But impractical for large amounts of data
- Businesses must maintain huge amounts of data
 - A **database management system (DBMS)** is the typical solution to the data needs of business
 - Designed to store, retrieve, and manipulate data
- Visual Basic can communicate with a DBMS
 - Tells DBMS what data to retrieve or manipulate

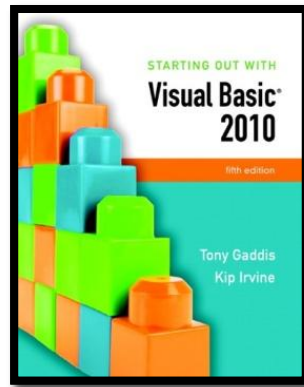
Layered Approach to Using a DBMS

- Applications that work with a DBMS use a layered approach
 - VB application is topmost layer
 - VB sends instructions to next layer, the DBMS
 - DBMS works directly with data
- Programmer need not understand the physical structure of the data
 - Just need to know how to interact with the database



Visual Basic Supports Many DBMS's

- Visual Basic can interact with many DBMS's
 - Microsoft SQL Server
 - Oracle
 - DB2
 - MySQL
- Microsoft SQL Server Express used in this chapter, which is installed with Visual Basic



Section 10.2

DATABASE CONCEPTS

A database is a collection of one or more tables, each containing data related to a particular topic.

Addison Wesley
is an imprint of



© 2011 Pearson Addison-Wesley. All rights reserved.

Terminology

- A **Database** is a collection of interrelated tables
- A **Table** is a logical grouping of related data
 - People, places, or things
 - For example, employees or departments
 - Organized into rows and columns
- A **Field** is an individual piece of data pertaining to an item, an employee name for instance
- A **Record** is the complete data about a single item such as all information about an employee
 - A record is a row of a table
- A **database schema** is the design of tables, columns, and relationships between tables in a database

Database Table

- Each table has a **primary key** or **composite key**
 - Uniquely identifies that row of the table
 - **Emp_Id** is the primary key in this example
- Columns are also called **fields** or *attributes*
- Each column has a particular data type

Emp_Id	First_Name	Last_Name	Department
001234	Ignacio	Fleta	Accounting
002000	Christian	Martin	Computer Support
002122	Orville	Gibson	Human Resources
003400	Ben	Smith	Accounting
003780	Allison	Chong	Computer Support

Row (Record) →

↑ Column

↙ Field

SQL Server Column Types

SQL type(s)	Usage	Visual Basic Type
bit	True/false values	Boolean
datetime, smalldatetime	Dates and times	Date, DateTime
decimal, money	Financial values in which precision is important	Decimal
float	Real-number values	Double
image	Pictures, Word documents, Excel files, PDF files	Array of Byte
int	Integer values	Integer
nvarchar(<i>n</i>)	Variable-length strings containing 16-bit Unicode characters	String
smallint	Integers between -32,768 and +32,767	Short
text	Strings longer than 8,000 characters	String
varchar(<i>n</i>)	Variable-length strings containing ANSI (8-bit) characters	String

Choosing Column Names

- Define a column for each piece of data
- Allow plenty of space for text fields
- Avoid using spaces in column names
- For the members of an organization:

<u>Column Name</u>	<u>Type</u>	<u>Remarks</u>
Member_ID	int	Primary key
First_Name	varchar(40)	
Last_Name	varchar(40)	
Phone	varchar(30)	
Email	varchar(50)	
Date_Joined	smalldatetime	Date only, no time values
Meeings_Attended	smallint	
Officer	Yes/No	True/False values

Avoiding Redundancy by Using Linked Tables

- Create a department table

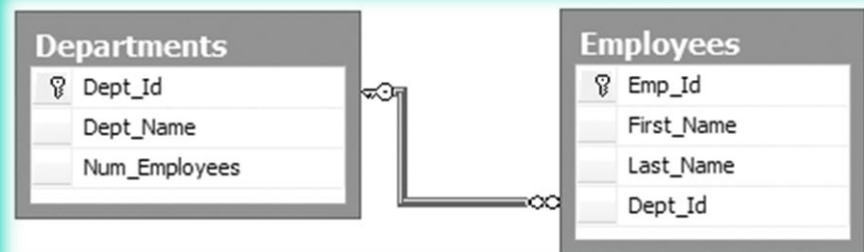
<u>Dept ID</u>	<u>Dept Name</u>	<u>Num Employees</u>
1	Human Resources	10
2	Accounting	5
3	Computer Support	30
4	Research & Development	15

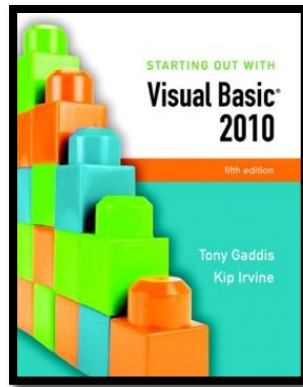
- Reference department table in employee table

<u>ID</u>	<u>First Name</u>	<u>Last Name</u>	<u>Dept ID</u>
001234	Ignacio	Fleta	2
002000	Christian	Martin	3
002122	Orville	Gibson	1
003000	Jose	Ramirez	4
003400	Ben	Smith	2
003780	Allison	Chong	3

One-to-Many Relationship

- Databases are designed around a **relational model**
- A **relation** is a link or relationship that relies on a common field
- The previous changes created a **one-to-many relationship**
 - Every employee has one and only one dept
 - Every department has many employees
 - **DeptID** in *Departments* table is a primary key
 - **DeptID** in *Employees* table is a **foreign key**
- One-to-many relationship exists when primary key of one table is specified as a field of another table





Section 10.3

DATAGRIDVIEW CONTROL

The DataGridView control allows you to display a database table in a grid. The grid can be used at runtime to sort and edit the contents of a table.

Addison Wesley
is an imprint of



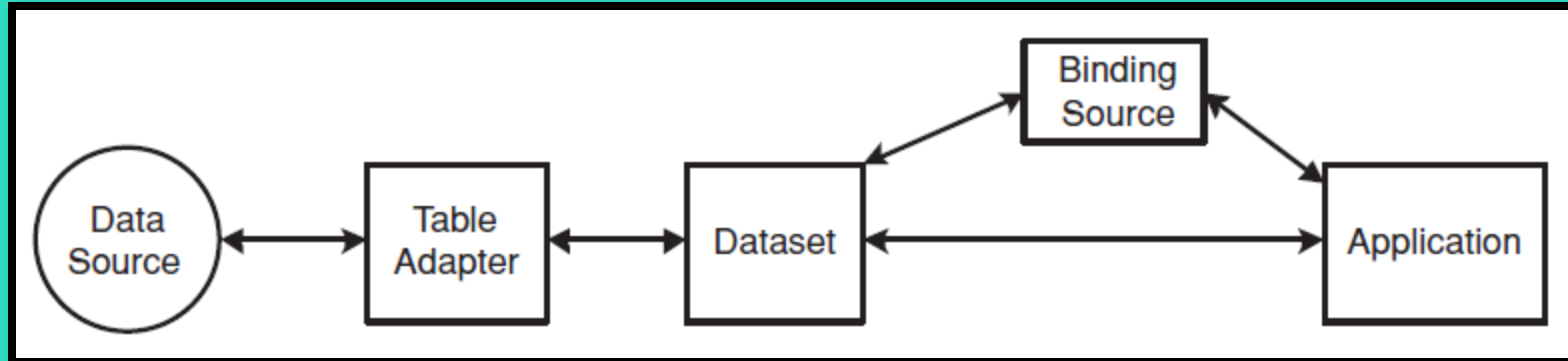
© 2011 Pearson Addison-Wesley. All rights reserved.

Connecting to a Database

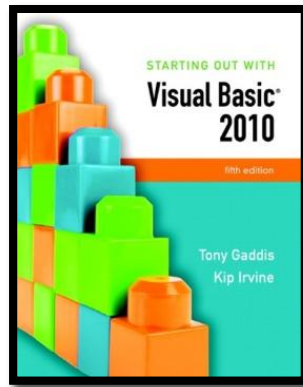
- Visual Basic uses a technique called **Data binding** to link tables to controls on forms
 - Special controls called **components** establish the link
 - A software tool named a **wizard** guides you through the process
- We will use these data-related components:
 - A **Data source** is usually a database
 - Can include text files, Excel spreadsheets, XML data, and Web services
 - A **Binding source** connects data bound controls to a dataset
 - A **Table adapter** pulls data from the database and passes it to your program
 - Uses **Structured Query Language (SQL)** is used to select data, add new rows, delete rows, and modify existing rows
 - A **Dataset** is an in-memory copy of data pulled from database tables

Connecting to a Database

- The flow of data from database to application



- Data travels from data source to application
- Application can view/change dataset contents
- Changes to dataset can be written back to the data source
- Tutorial 10-1 demonstrates how to connect a database table to a **DataGridView control**
- Tutorial 10-2 demonstrates updating and sorting a table



Section 10.4

DATA-BOUND CONTROLS

Some controls can be bound to a dataset. A data-bound control can be used to display and edit the contents of a particular row and column.

Addison Wesley
is an imprint of



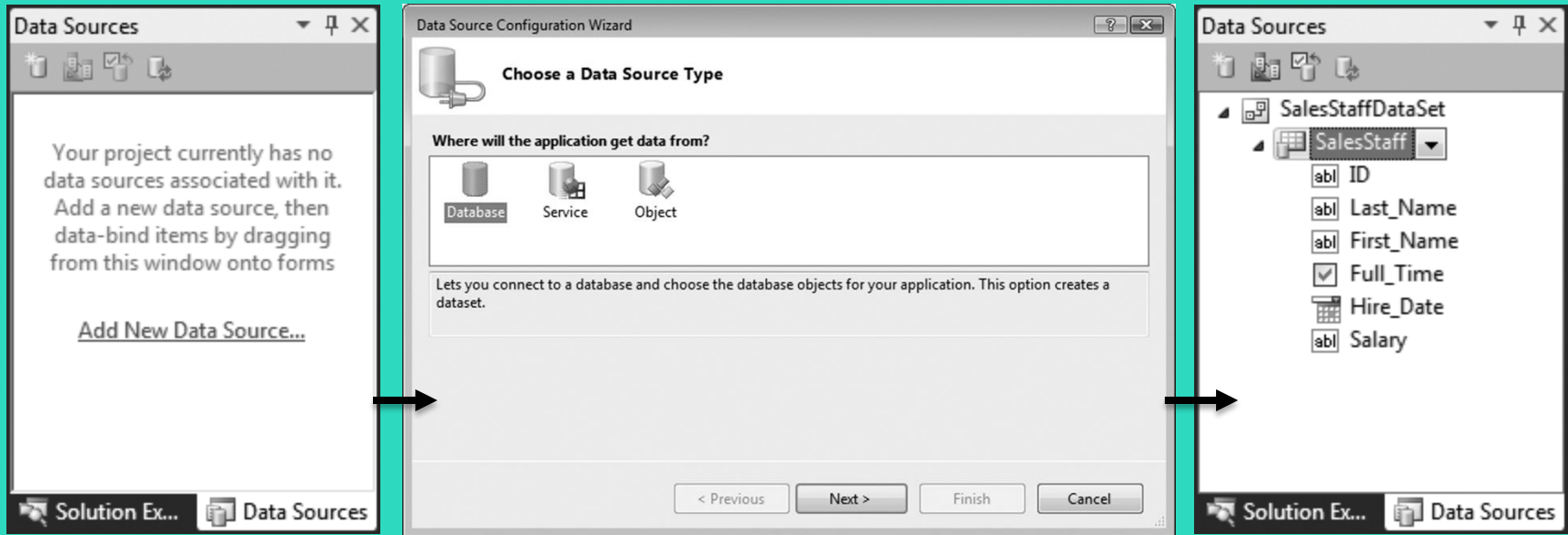
© 2011 Pearson Addison-Wesley. All rights reserved.

Advantages of Data-Bound Controls

- Can bind fields in a data source to controls:
 - Text boxes
 - Labels
 - List boxes
- Contents of **data-bound controls** change automatically when moving from row to row
- *Data-bound controls* also allow the contents of a database field to be changed

Adding a New Data Source

- Open the *Data Sources* window and click the *Add New Data Source* link
- Follow the steps in the *Data Source Configuration Wizard* to create a connection to the database

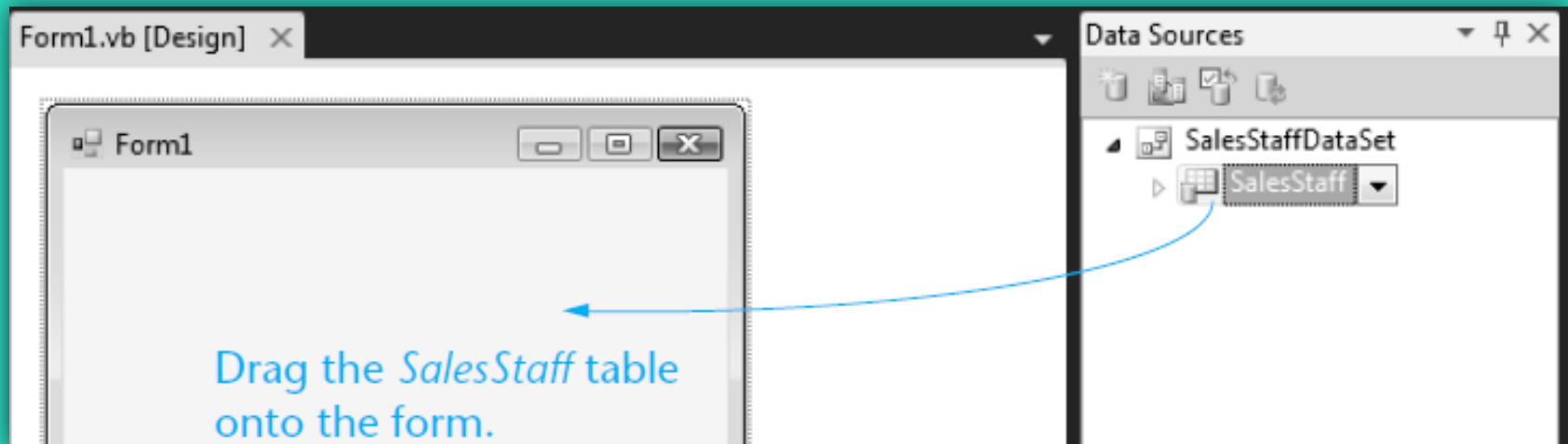


Deleting a Data Source

- Once created, it's almost impossible to rename a data source
- Easier to delete and create a new data source than rename one
- A data source named *Employees* for example would be defined by a file named **Employees.xsd**
- To delete this data source:
 - Select **Employees.xsd** file in *Solution Explorer*
 - Press Delete on the keyboard

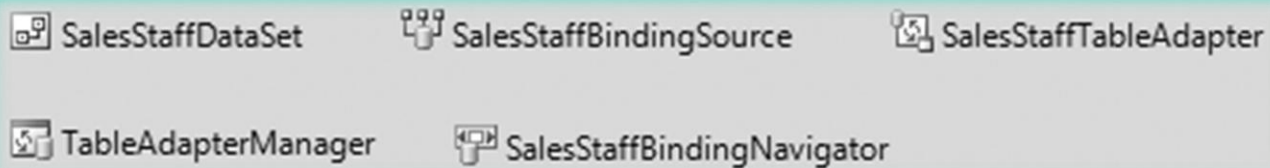
Binding the Data Source to a DataGridView Control

- Drag and drop an existing dataset from the *Data Sources* window to an open area on the form
 - For example:



Binding the Data Source to a DataGridView Control

- At the same time Visual Studio builds a DataGridView on the form, it adds a number of important objects to the form's component tray:



- The **BindingNavigator** creates a *ToolStrip* at the top of the form
- The **DataSet** is an in-memory copy of the table
- The **BindingSource** connects the *DataGridView* to the *DataSet*
- The **TableAdapter** pulls data from the database into the *DataSet*
- The **AdapterManager** is a tool for saving data in related tables

Binding Individual Fields to Controls

- Use the dataset in the *Data Sources* window
 - Select *Details* from the table drop-down list
 - Drag table to an open area of a form
 - Creates a separate control for each field
 - Can also drag columns individually
- Text and numeric fields added as text boxes
- Yes/No fields added as checkboxes
- DateTime fields use **DateTimePicker controls**
- May wish to change some control properties
- Tutorials 10-3 and 10-4 demonstrate binding

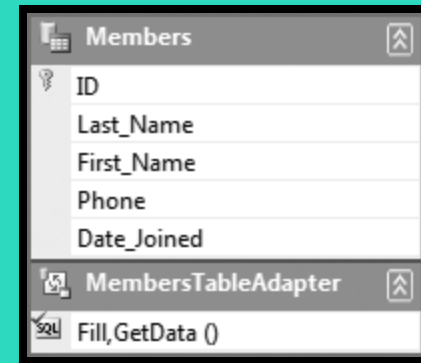
Binding to ListBox and ComboBox Controls

- List and combo boxes are frequently used to supply a list of items for a user to select from
- Such lists are often populated from a table
- Must set two list/combo box properties
 - The **DataSource property** identifies a table within a dataset
 - The **DisplayMember property** identifies the table column to be displayed in the list/combo box
- If table column dragged onto a list/combo box
 - Visual Studio creates the required dataset, table adapter, and binding source components
- Tutorial 10-5 demonstrates binding to a list box

Adding Rows to a Database Table

- A TableAdapter provides an easy way to add a row to a database table
- To find the TableAdapter you must open a data set's **Schema Definition**
- A **schema definition file (.xsd)** was automatically created in Tutorial 10-5 for the Members table Dataset
 - Displays the names and data types of fields in the table

- To edit the schema definition file:
 - Double-click its name in the *Solution Explorer* window
 - An editor window will open



- A **TableAdapter** object was automatically created for the Members DataTable
- Each DataTable has a TableAdapter associated with it

Adding Rows to a Database Table

- A TableAdapter object has an **Insert** method
 - Used to add a new row to the database table
 - Each column is an argument of the method
 - Just provide the values for each argument
 - For example:

```
MembersTableAdapter.Insert(10, "Hasegawa", "Adrian",  
                           "305-999-8888", #5/15/2010#)
```

Identity Columns

- Some database tables have an **identity column**
 - Assigned a unique number by the database
 - Occurs automatically for identity columns
 - No need to manually supply a value for this column
- Payments table uses an identity column
 - Omit ID column value
 - Only supply Member_Id, Payment_Date, and Amount
PaymentsTableAdapter.Insert(5, #5/15/2010#, 50D)
 - Tutorial 10-6 shows you how to insert new rows into the *Payments* table of the *Karate* database

Using Loops with DataTables

- A **For Each** statement can be used to iterate over the **rows** collection of a table
- Usually, it is best to create a strongly typed row that matches the type of rows in the table
- For example:
 - Total the *Amount* column of **PaymentsDataSet** dataset

```
Dim row As PaymentsDataSet.PaymentsRow
```

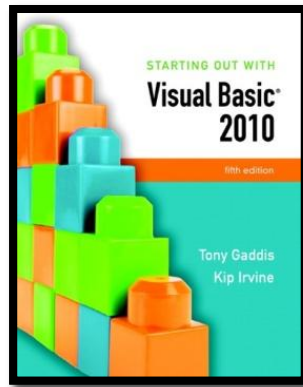
```
Dim decTotal As Decimal = 0
```

```
For Each row In Me.PaymentsDataSet.Payments.Rows
```

```
    decTotal += row.Amount
```

```
Next
```

- Tutorial 10-7 shows how to add a total to the *Karate* student payments application



Section 10.5

STRUCTURED QUERY LANGUAGE (SQL)

SQL, which stands for Structured Query Language, is a standard language for working with database management systems.

Addison Wesley
is an imprint of



© 2011 Pearson Addison-Wesley. All rights reserved.

Introduction

- SQL stands for *Structured Query Language*
 - A standard language for working with database management systems
 - Standardized by the American National Standards Institute (ANSI)
 - The language of choice for interacting with database management systems
- Consists of a limited set of keywords
 - Keywords construct statements called **database queries**
 - Queries are submitted to the DBMS
 - In response to queries, the DBMS carries out operations on its data

SELECT Statement

- The **SELECT statement** retrieves data from a database
 - Used to select rows, columns, and tables
 - The most basic format for a single table is:

SELECT *ColumnList*
FROM *Table*

- ***ColumnList*** must contain table column names separated by commas
- The following statement selects the *ID* and *Salary* columns from the *SalesStaff* table:

SELECT ID, Salary
FROM SalesStaff

SQL Statements and Style

- There is no required formatting or capitalization of SQL statements
 - The following queries are equivalent:

SELECT ID, Salary FROM SalesStaff

select ID, Salary from SalesStaff

Select id, salary from salesstaff

SeLeCt Id, SaLaRy FrOm SaLeSsTaFf

- As a matter of style and readability
- You should try to use consistent capitalization

SELECT Statement

- Field names that contain embedded spaces must be surrounded by square brackets
 - For example:

```
SELECT [Last Name], [First Name]  
FROM Employees
```

- The * character in the column list selects all the columns from a table
 - For example:

```
SELECT *  
FROM SalesStaff
```

Aliases for Column Names

- Column names can be renamed using the **AS** keyword
 - The new column name is called an *alias*
 - For example:

```
SELECT Last_Name, Hire_Date AS Date_Hired  
FROM SalesStaff
```

- Renaming columns is useful for two reasons:
 1. You can hide the real column names from users for security purposes
 2. You can rename database columns to make user friendly column headings in reports

Creating Alias Columns from Other Columns

- A query can create a new column from other existing columns
 - For example:

```
SELECT Last_Name + ', ' + First_Name AS Full_Name  
FROM Members
```
 - When strings occur in queries they must be surrounded by apostrophes
 - The + operator concatenates multiple strings into a single string

Calculated Columns

- You can create new columns from calculated column values

- For example, the following query:

```
SELECT employeeld, hoursWorked * hourlyRate AS payAmount  
FROM PayRoll
```

- Multiplies the values of two columns
 - *hoursWorked* and *hourlyRate*
 - Displays the result as a new column (alias)
 - *payAmount*

Setting the Row Order with **ORDER BY**

- SQL Select has an optional **ORDER BY** clause that affects the order in which rows appear

ORDER BY Last_Name, First_Name

- Displays rows in order by last name, then first
- Sort in descending order (high to low) using **DESC**

ORDER BY Last_Name DESC

- **ORDER BY** clause appears after **FROM** clause

SELECT First_Name, Last_Name, Date_Joined

FROM Members

ORDER BY Last_Name, First_Name

- Lists all members by last name, then first

Selecting Rows with the **WHERE** Clause

- SQL Select has an optional **WHERE clause** that can be used to select (or filter) certain rows

```
WHERE Last_Name = 'Gomez'
```

- Displays only rows where last name is Gomez
- Must be a defined column (in table or created)
- This example selects based on a created field

```
SELECT Last_Name, hrsWorked * Rate AS payAmount  
FROM Payroll  
WHERE payAmount > 1000  
ORDER BY Last_Name
```

- Selects those being paid more than \$1,000

Relational Operators

- SQL **WHERE** clause uses relational operators like an **If** statement

<u>Operator</u>	<u>Meaning</u>
=	equal to
<>	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
BETWEEN	between two values (inclusive)
LIKE	similar to (match using wildcard)

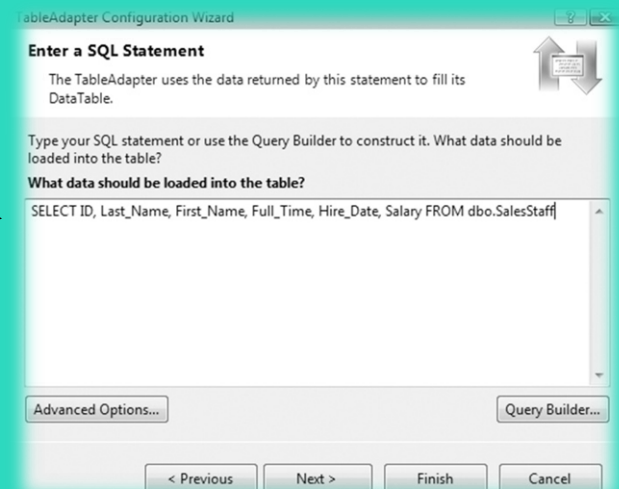
- Example of **BETWEEN** operator:
WHERE (Hire_Date BETWEEN '1/1/1992' AND '12/31/1999')
- Example of **LIKE** operator with % sign as wildcard:
WHERE Last_Name LIKE 'A%'

Compound Expressions

- SQL uses **AND**, **OR**, and **NOT** to create compound expressions
- Select all employees hired after 1/1/1990 *and* with a salary is greater than \$40,000
`WHERE (Hire_Date > '1/1/1990') AND (Salary > 40000)`
- Select all employees hired after 1/1/1990 *or* with a salary is greater than \$40,000
`WHERE (Hire_Date > '1/1/1990') OR (Salary > 40000)`
- Select employee names not beginning with A
`WHERE Last_Name NOT Like 'A%'`

Modifying the Query in a Data Source

- Dataset schema file contains an SQL query
 - Created as part of schema file
 - Named Fill, GetData() by default
- Right-click title bar of *TableAdapter* in schema
 - Click *Configure* from pop-up
 - Use *Configuration Wizard* to change simple queries
 - *Query Builder* often used for complex queries



Query Builder

- Visual Studio tool to work with SQL queries
- Consists of four sections called *panes*
 - The **Diagram pane** displays tables
 - The **Grid pane (Criteria pane)** displays query in spreadsheet form
 - The **SQL pane** shows actual SQL created
 - The **Results pane** shows data returned by query

Example Query Builder Window

The screenshot shows a 'Query Builder' window with the following components:

- Diagram pane:** A tree view showing the 'SalesStaff' table with selected columns: ID, Last_Name, First_Name, and Full_Time.
- Grid pane:** A table with columns: Column, Alias, Table, Outp..., Sort Type, Sort Order, Filter, and Or... It contains two rows for 'SalesStaff' with columns 'ID' and 'Last_Name' checked in the 'Outp...' column.
- SQL pane:** A text area containing the SQL query:

```
SELECT ID, Last_Name, First_Name, Full_Time, Hire_Date, Salary  
FROM SalesStaff
```
- Results pane:** A table displaying the query results with columns: ID, Last_Name, First_Name, Full_Time, Hire_Date, and Salary. It shows three rows of data.

Annotations with blue arrows point to these four panes from the right side of the window.

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...
ID		SalesStaff	<input checked="" type="checkbox"/>				
Last_Name		SalesStaff	<input checked="" type="checkbox"/>				

ID	Last_Name	First_Name	Full_Time	Hire_Date	Salary
104	Adams	Adrian	True	1/1/2010 12:00:...	35007
114	Franklin	Fay	True	8/22/2005 12:00:...	56001
115	Franklin	Adiel	False	3/20/2010 12:00:...	41000

Adding a Query to a TableAdapter

- Can add a new query as well as changing an existing one
 - Right-click the TableAdapter icon in *component* tray
 - Select *Add Query*
 - The *Search Criteria Builder* window appears
- Add **WHERE** clause to the **SELECT** statement
 - Select the *New query name* to enter a name for query
- Query made available from *ToolStrip* control
- Tutorial 10-8 shows how to filter rows in the *SalesStaff* table

Example *Search Criteria Builder* Window

Search Criteria Builder

Choose an existing query or enter a new query below. A ToolStrip will be added to the form to run the query. To edit an existing query or use stored procedures use the Configure command on the TableAdapter in the DataSet Designer.

Select data source table:
SalesStaffDataSet.SalesStaff

Select a parameterized query to load data:

New query name: FillBy

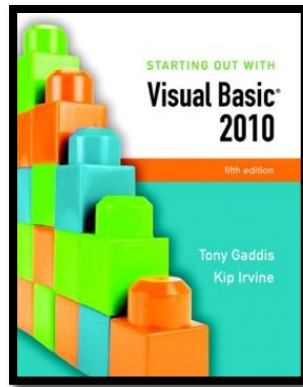
Existing query name:

Query Text:
SELECT ID, Last_Name, First_Name, Full_Time, Hire_Date, Salary FROM dbo.SalesStaff

Sample: SELECT ColumnName1, ColumnName2 FROM TableName
WHERE ColumnName1 = @ParameterName

Query Builder...

OK Cancel



Section 10.6

FOCUS ON PROBLEM SOLVING: *KARATE SCHOOL MANAGEMENT APPLICATION*

Develop the *Karate School Management* Application

Addison Wesley
is an imprint of



© 2011 Pearson Addison-Wesley. All rights reserved.

Karate School Manager Startup Form

- Menu Selections:
 - File
 - Exit
 - Membership
 - List All
 - Find member
 - Add new member
 - Payments
 - All members
 - One member



All Members Form



The screenshot shows a web browser window with the title 'All Members'. Below the title bar is a 'File' menu. The main content is a table with six columns: ID, First_Name, Last_Name, Phone, and Date_Joined. The table contains eight rows of data. The first row is highlighted with a dark background. The table is displayed within a window that has standard window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

	ID	First_Name	Last_Name	Phone	Date_Joined
▶	1	Keoki	Kahumanu	111-2222	2/20/2002
	2	Anne	Chong	232-3333	2/20/2010
	3	Elaine	Hasegawa	313-3455	2/20/2004
	4	Brian	Kahane	646-9387	5/20/2008
	5	Aldo	Gonzalez	123-2345	6/6/2009
	6	Jascha	Kousevitzky	414-2345	2/20/2010
	7	Moses	Taliafea	545-2323	5/20/2005
	8	Rafael	Concepcion	602-3312	5/20/2007

Find Member by Last Name Form

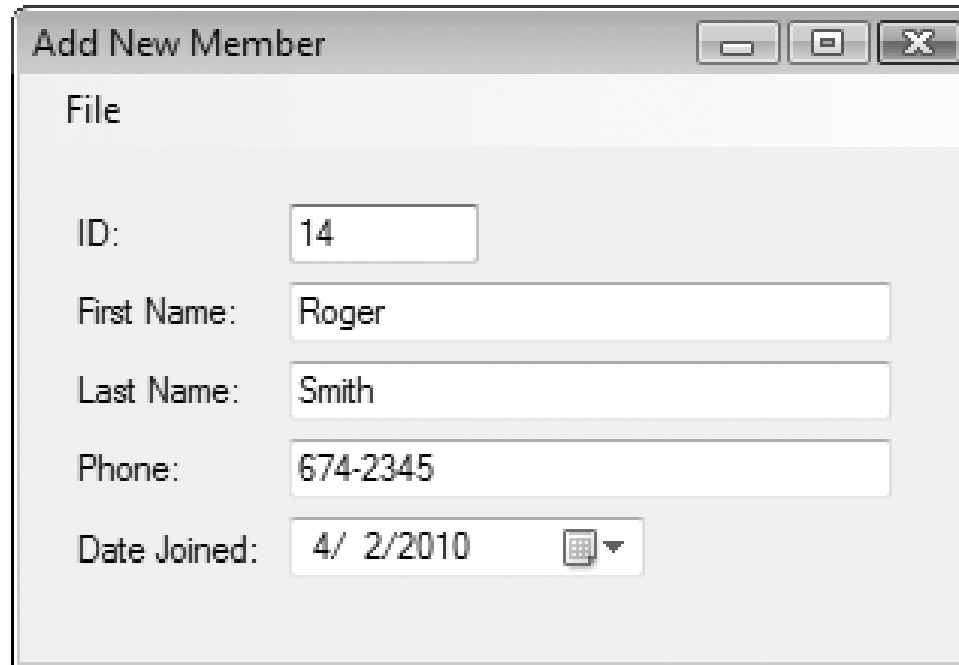
Find Member by Last Name

File


Enter a partial last name:

ID	Last_Name	First_Name	Phone	Date_Joined
1	Kahumanu	Keoki	111-2222	2/20/2002
4	Kahane	Brian	646-9387	5/20/2008

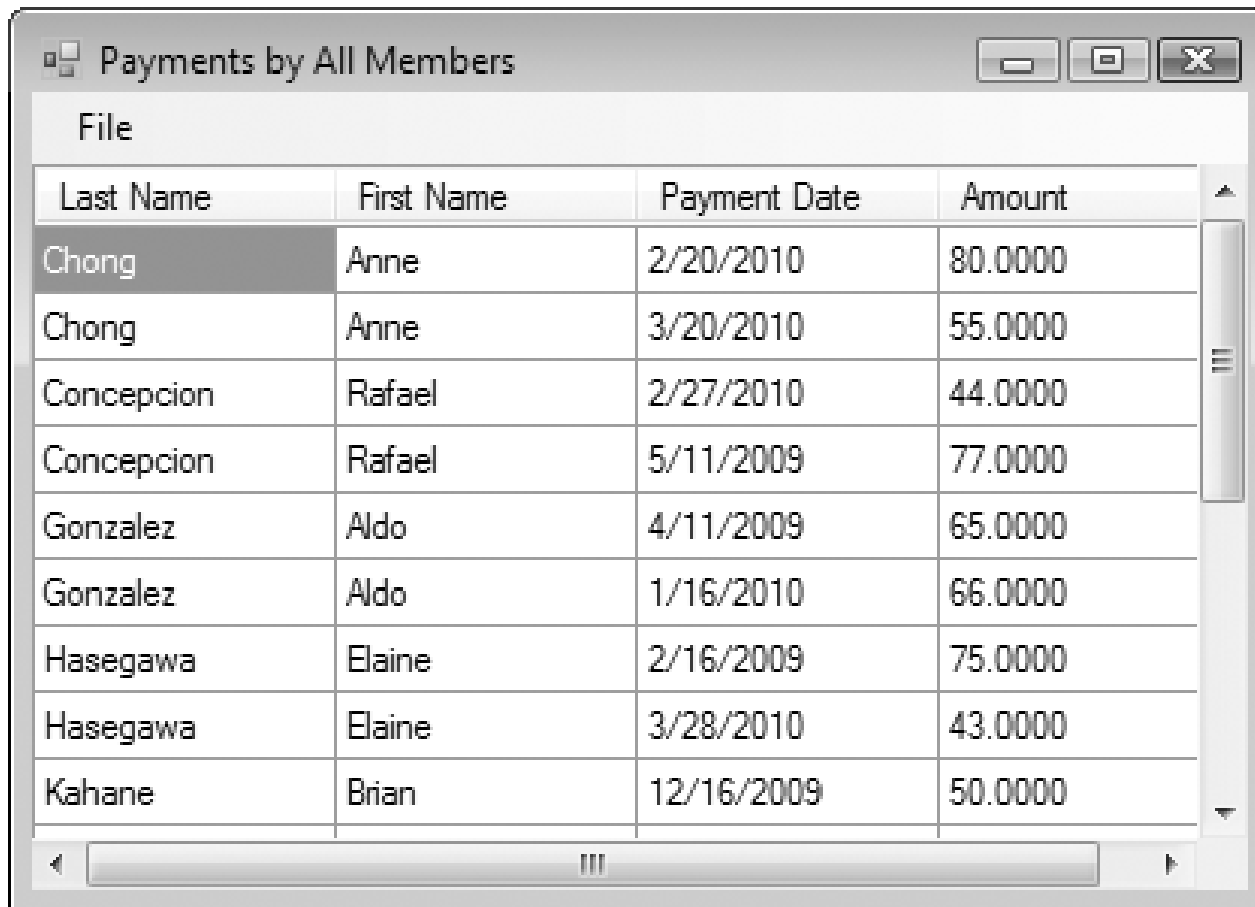
Add New Member Form



A screenshot of a web application window titled "Add New Member". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar is a "File" menu. The main content area contains a form with the following fields:

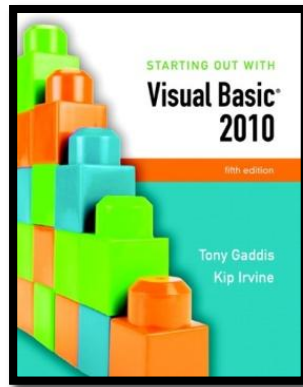
ID:	<input type="text" value="14"/>
First Name:	<input type="text" value="Roger"/>
Last Name:	<input type="text" value="Smith"/>
Phone:	<input type="text" value="674-2345"/>
Date Joined:	<input type="text" value="4/ 2/2010"/> 

Payment Form



The screenshot shows a window titled "Payments by All Members" with a menu bar containing "File". Below the menu bar is a table with four columns: "Last Name", "First Name", "Payment Date", and "Amount". The table contains ten rows of payment data. The first row is highlighted. A scrollbar is visible on the right side of the table, and a horizontal scrollbar is at the bottom.

Last Name	First Name	Payment Date	Amount
Chong	Anne	2/20/2010	80.0000
Chong	Anne	3/20/2010	55.0000
Concepcion	Rafael	2/27/2010	44.0000
Concepcion	Rafael	5/11/2009	77.0000
Gonzalez	Aldo	4/11/2009	65.0000
Gonzalez	Aldo	1/16/2010	66.0000
Hasegawa	Elaine	2/16/2009	75.0000
Hasegawa	Elaine	3/28/2010	43.0000
Kahane	Brian	12/16/2009	50.0000



Section 10.7

INTRODUCTION TO LINQ

LINQ (Language Integrated Query) is a query language that is built into Visual Basic and can be used to query data from many sources other than databases.

Addison Wesley
is an imprint of



© 2011 Pearson Addison-Wesley. All rights reserved.

LINQ

- SQL allows you to query the data in a database.
- **LINQ** allows you to query data from many other sources.
- LINQ is built into Visual Basic.

Using LINQ to Query an Array

- Suppose we have the following array:

Dim intNumbers() As Integer = {4, 104, 2, 102, 1, 101, 3, 103}

- The following statement uses LINQ to query the array for all values greater than 100:

```
From item In intNumbers  
Where item > 100  
Select item
```

Using LINQ to Add Query Results to a ListBox

- We can add the results to a ListBox

' Create an array of integers.

```
Dim intNumbers() As Integer = {4, 104, 2, 102, 1, 101, 3, 103}
```

' Use LINQ to query the array for all numbers

' that are greater than 100.

```
Dim queryResults = From item In intNumbers
```

```
    Where item > 100
```

```
    Select item
```

' Add the query results to the list box.

```
For Each intNum As Integer In queryResults
```

```
    lstResults.Items.Add(intNum)
```

```
Next
```


Sorting the Results of a LINQ Query

- Sort in ascending order:

```
Dim queryResults = From item In intNumbers  
Where item > 100  
Select item  
Order By item
```

- Sort in descending order:

```
Dim queryResults = From item In intNumbers  
Where item > 100  
Select item  
Order By item Descending
```

More About LINQ

- LINQ uses operators that are similar to SQL
- Unlike SQL, LINQ is built into Visual Basic
- Queries are written directly into the program
 - VB compiler checks the syntax of the query
 - Immediately displays LINQ mistakes
- LINQ can be used to query any data that is stored in memory as an object
- An application named LINQ can be found in the Chap10 student sample programs folder