PRACTICE

- Determine the output displayed when the button is clicked.

```
Dim actor(5,5) As String
Private Sub btnDisplay_Click(...) Handles
btnDisplay.Click
        Dim a As Integer = 2, b As Integer = 3, temp As
Integer
        actor(a, b) = “Bogart”
        temp = a
        a = b
        b = temp
        lstOutput.Items.Add(“1. ” & actor(a, b))
        lstOutput.Items.Add(“2. ” & actor(b, a))
End Sub
```

PRACTICE

- Determine the output displayed when the button is clicked.

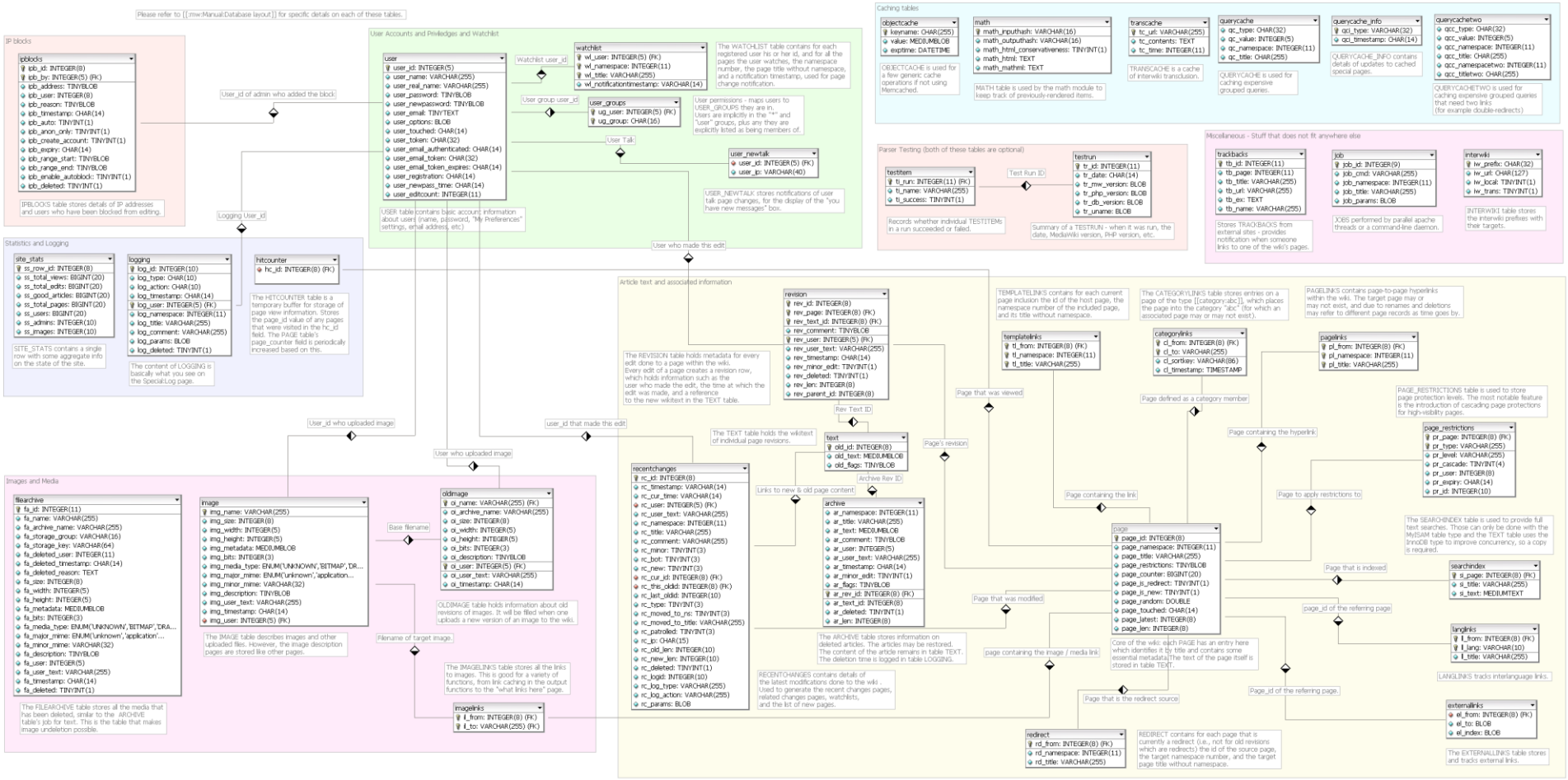
```
Dim a(3,4) As Integer
Private Sub btnDisplay_Click(...) Handles
btnDisplay.Click
        Dim row As String
        For j As Integer = 0 To 3
                row = “”
                For k As Integer = 0 To 4
                        a(j, k) = j + k
                        row &= a(j, k) & “ ”
                Next
                lstOutput.Items.Add(row)
        Next
End Sub
```

Chapter 10 – Database Management

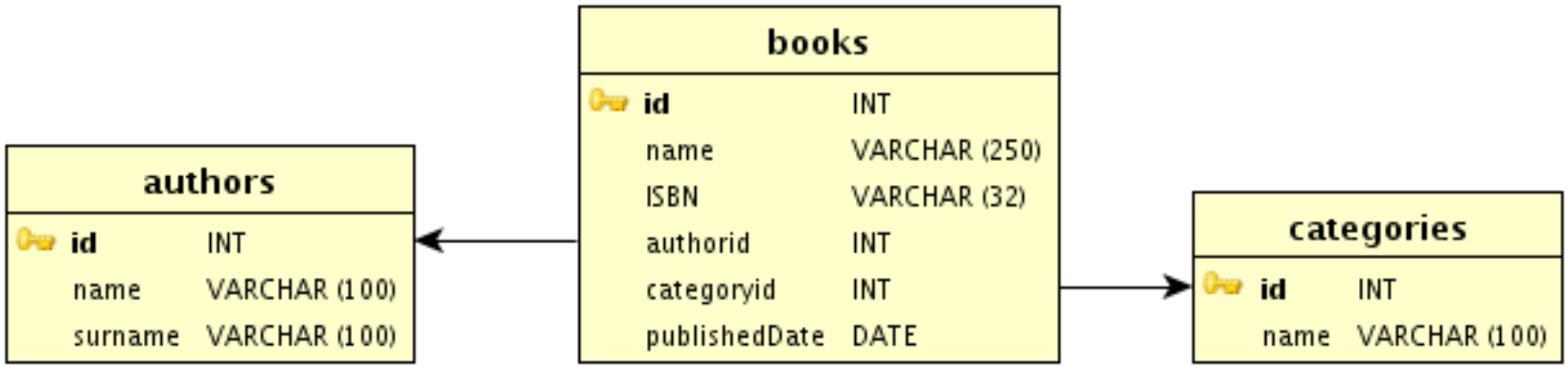
10.1 An Introduction to Databases

10.2 Relational Databases and SQL

Sample DB Schema



Sample DB Schema



Sample Table – Cities Table

city	country	pop2005	pop2015
Bombay	India	18.2	22.6
Calcutta	India	14.3	16.8
Delhi	India	15.1	20.9
Dhaka	Bangladesh	12.4	17.9
Jakarta	Indonesia	13	17.5
Lagos	Nigeria	11	17
Mexico City	Mexico	19	20.6
New York	USA	18.5	19.7
Sao Paulo	Brazil	18.2	20
Tokyo	Japan	35.2	36.2

SAMPLE TABLE – COUNTRIES TABLE

country	pop2005	monetaryUnit
Bangladesh	141.8	taka
Brazil	186.4	real
China	1316.8	yuan
India	1103.4	rupee
Indonesia	222.8	rupiah
Japan	128.1	yen
Mexico	107	peso
Nigeria	131.5	naira
Russia	143.2	ruble
USA	298.2	dollar

What is a database?

- A *database* (*DB*) is a very large, integrated, permanent collection of data.
- Models real-world
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CMPT354).
- Example databases:
  - Customer Transactions
  - Human Genome
  - Online Bookstore
  - . . .

Database Terminology

- A **table** is a rectangular array of data.
- Each column of the table, called a **field**, contains the same type of information.
- Each row, called a **record**, contains all the information about one entry in the database.

Database Management Software (DBMS)

- Used to create databases
- Databases can contain one or more related tables
- Examples of DBMS include Access and Oracle

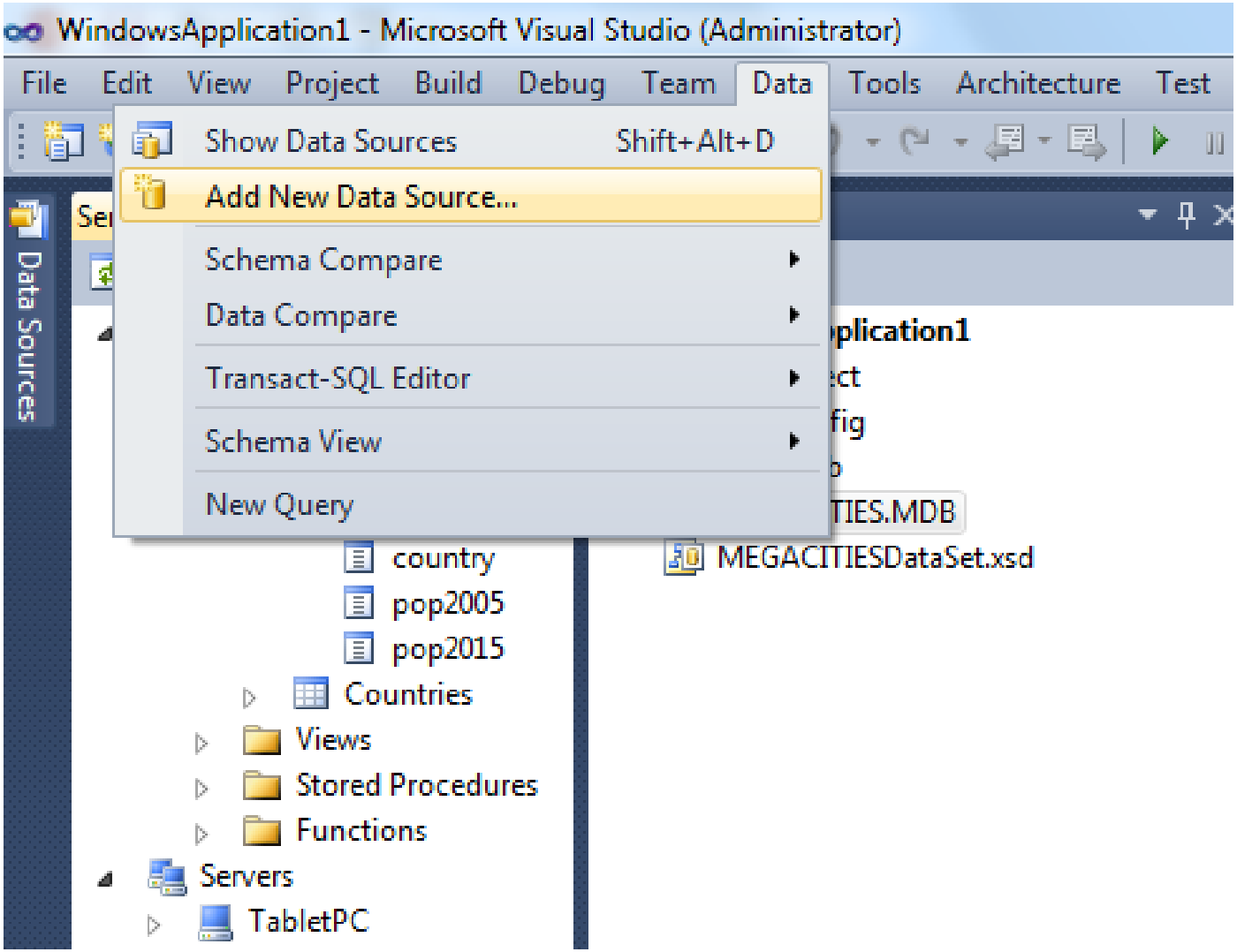
Database Explorer

- A tool provided by Visual Basic Express to examine any database.
- Invoked from the View menu
- Allows you to determine the names of the tables (and their fields) and view the contents of any table.
- Other edition of Visual Basic provide an analogous tool called Server Explorer.

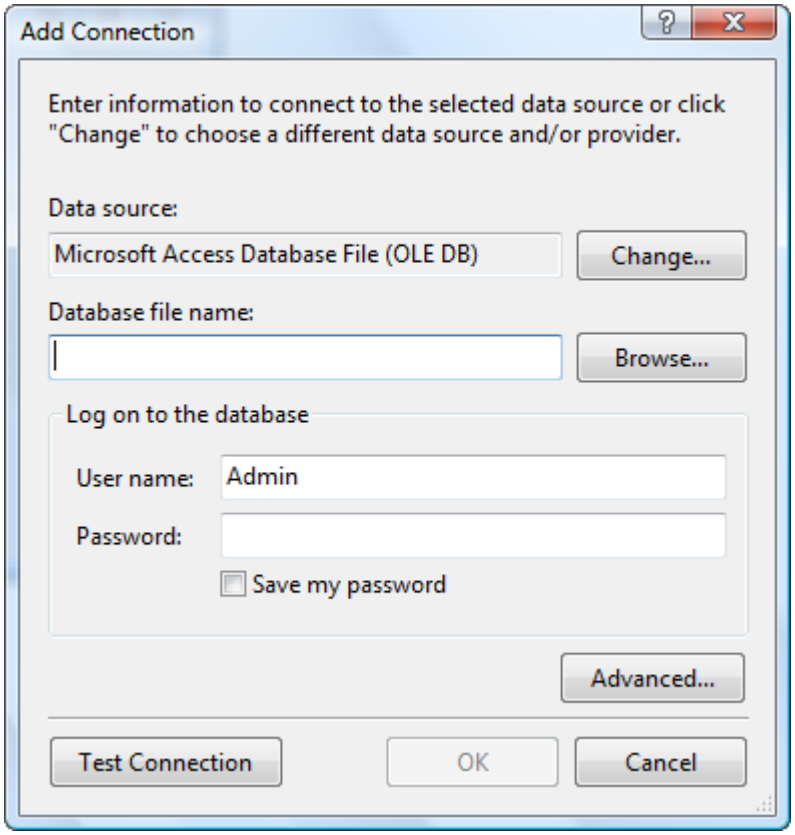
Databases Provided

- The Add Connection dialog box is used by Database Explorer to connect to a database
- The databases used in this book can be found
  - in the folder Programs\Ch10\MajorDatabases
  - website

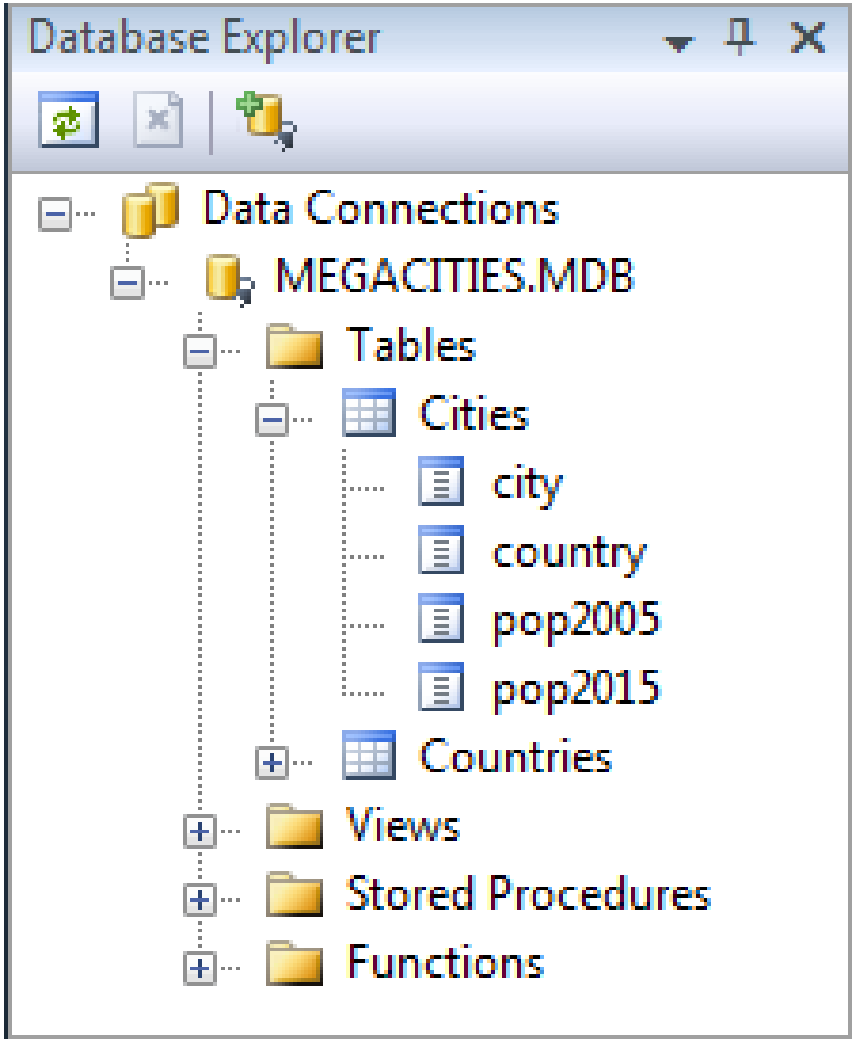
Add Connection Dialog Box



Add Connection Dialog Box



Database Explorer Window after Connection to MEGACITIES.MDB



Cities Table as Displayed by Database Explorer

	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9
	Dhaka	Bangladesh	12.4	17.9
	Jakarta	Indonesia	13	17.5
	Lagos	Nigeria	11	17
	Mexico City	Mexico	19	20.6
	New York	USA	18.5	19.7
	Sao Paulo	Brazil	18.2	20
	Tokyo	Japan	35.2	36.2
✳	*NULL*	*NULL*	*NULL*	*NULL*

Data Table Object

- A DataTable object holds the contents of a table as a rectangular array.
- A data table is similar to a two-dimensional array; it has rows and columns.

DataTable Variable

- The following declares a DataTable variable

```
Dim dt As New DataTable()
```

Connecting with a DataTable

```
Dim dt As New DataTable()
Dim connStr As String = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=MEGACITIES.MDB"
Dim sqlStr As String = "SELECT * FROM Cities"
Dim dataAdapter As New _
         OleDb.OleDbDataAdapter(sqlStr, connStr)
dataAdapter.Fill(dt)
dataAdapter.Dispose()
```


(Boilerplate to be inserted into every program in chapter.)

Properties of the DataTable

- After the six lines of boilerplate code are executed, the number of records in the table is given by

```
dt.Rows.Count
```

- The number of columns in the table is given by

```
dt.Columns.Count
```

- The records are numbered 0 through

```
dt.Rows.Count - 1
```

- The fields are numbered 0 through

```
dt.Columns.Count - 1
```

	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9

More Properties

- The name of the *j*th field is given by

  `dt.Columns(j)`

- The entry in the *j*th field of the *i*th record is

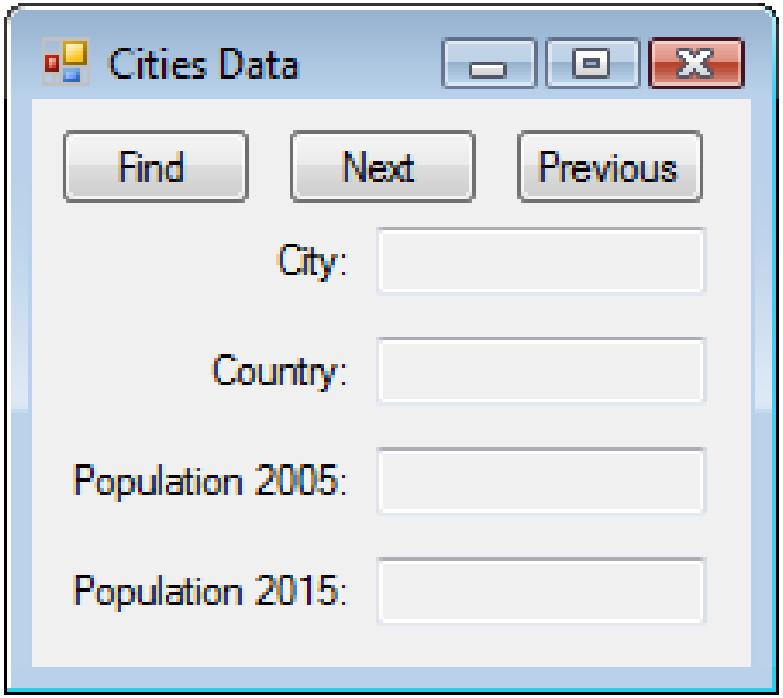
  `dt.Rows(i)(j)`

- The entry in the specified field of the *i*th record is

  `dt.Rows(i)(`*`fieldName`*`)`

	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9

Example 1: Form



Display one record at a time from the Cities table.

Example 1: Partial Code

```
Dim dt As New DataTable()
Dim rowIndex As Integer = 0

Private Sub frmCities_Load(...) Handles _
  MyBase.Load
  (Last five statements of boilerplate)
  UpdateTextBoxes()
End Sub

Sub UpdateTextBoxes()
  'Display contents of row specified by rowIndex
  variable
  txtCity.Text = CStr(dt.Rows(rowIndex)("city"))
  txtCountry.Text =
  CStr(dt.Rows(rowIndex)("country"))
  txtPop2005.Text =
  CStr(dt.Rows(rowIndex)("pop2005"))
  txtPop2015.Text =
  CStr(dt.Rows(rowIndex)("pop2015"))
End Sub
```

	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9
	Dhaka	Bangladesh	12.4	17.9

EXAMPLE 1: PARTIAL CODE CONT.

```
Private Sub btnNext_Click(...) Handles
  btnNext.Click
   'Show the next record if current one is not
   the last
   If (rowIndex < dt.Rows.Count - 1) Then
      rowIndex += 1       'Increase rowIndex by 1
      UpdateTextBoxes()
   End If
End Sub
```

	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9

Example 1: Partial Code cont.

```
Private Sub btnPrevious_Click(...) Handles _

  btnPrevious.Click
   'Show previous record if current one is not
   the first
   If (rowIndex > 0) Then
      rowIndex = rowIndex - 1
      UpdateTextBoxes()
   End If
End Sub
```

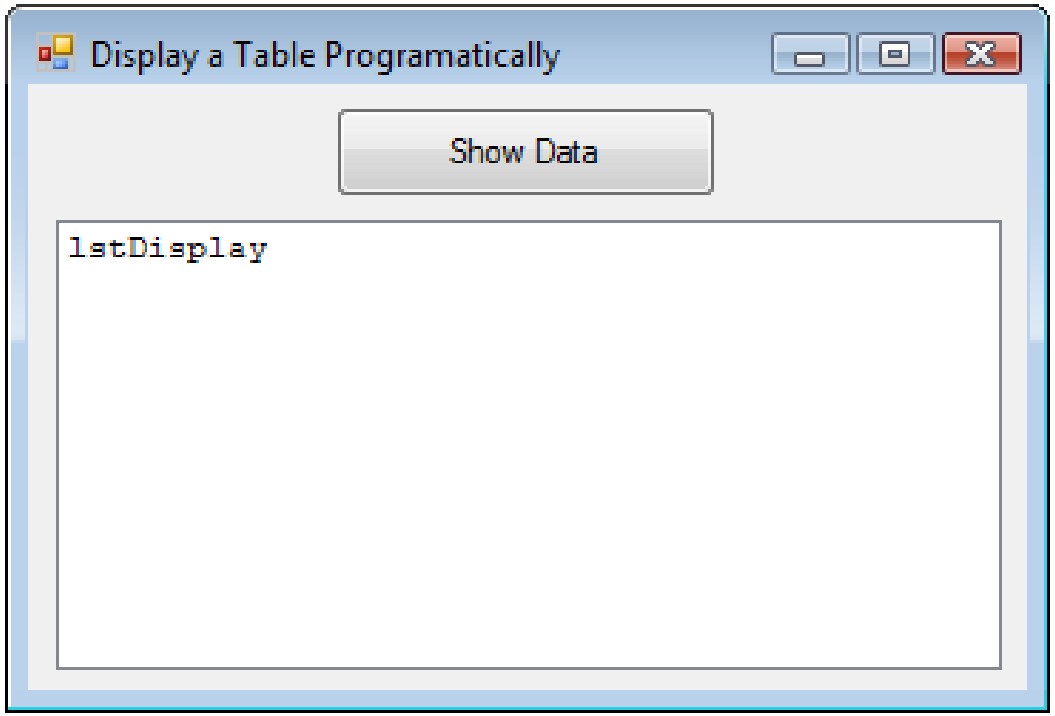
	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9

Example 1: Partial Code cont.

```
Private Sub btnFind_Click(...) Handles btnFind.Click
  Dim cityName As String
  Dim cityFound As Boolean = False
  cityName=InputBox("Enter name of city to search for.")
  For i As Integer = 0 To (dt.Rows.Count - 1)
    If CStr(dt.Rows(i)("city")) = cityName Then
      cityFound = True
      rowIndex = i
      UpdateTextBoxes()
    End If
  Next
  If (Not cityFound) Then
    MessageBox.Show("Cannot find requested city")
  End If
End Sub
```

	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9

Example 2: Form



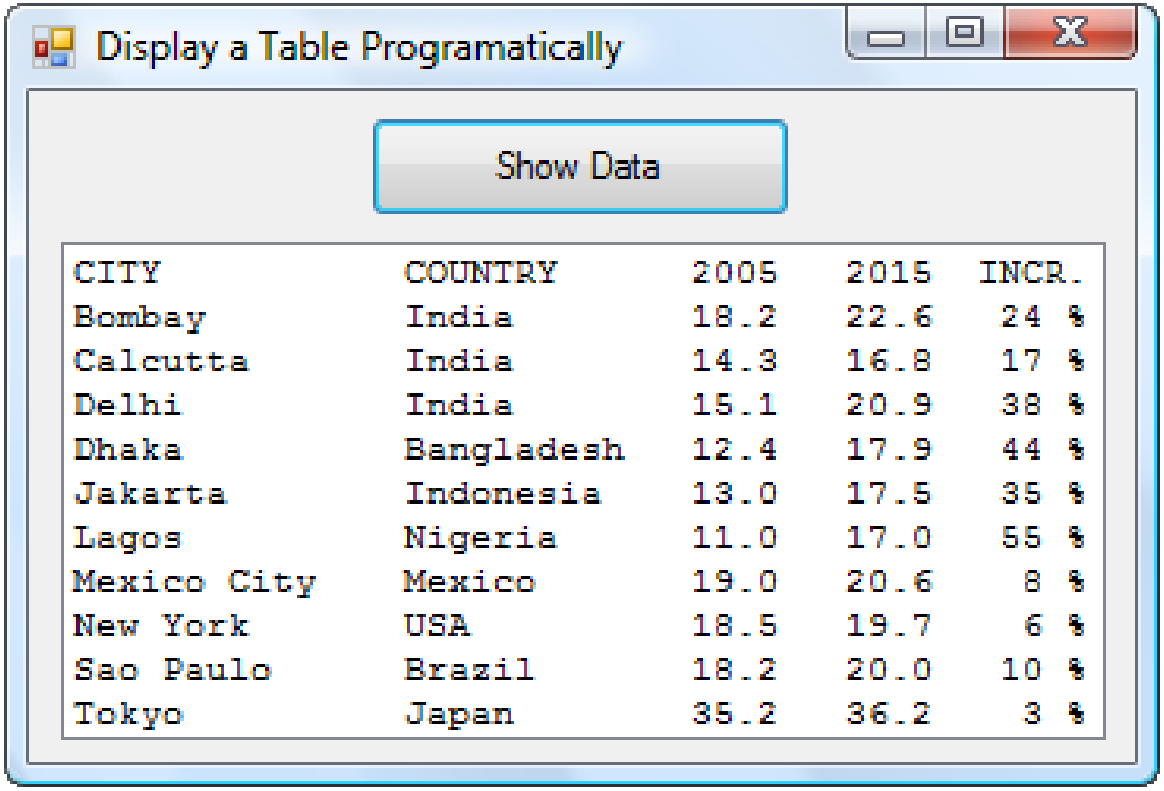
Display Cities table along with percentage growth.

Example 2: Code

```
Private Sub btnShow_Click(...) Handles btnShow.Click
  Dim fmtStr As String="{0,-15}{1,-10}{2,7:N1}{3,7:N1}{4,7:P0}"
  Dim percentIncrease As Double
  (Six statements of boilerplate)
  lstDisplay.Items.Add(String.Format(fmtStr, "CITY", _
                                     "COUNTRY", "2005", "2015", "INCR."))
  For i As Integer = 0 To dt.Rows.Count - 1
    percentIncrease = (CDbl(dt.Rows(i)("pop2015")) - _
      CDbl(dt.Rows(i)("pop2005"))) / CDbl(dt.Rows(i)("pop2005"))
    lstDisplay.Items.Add(String.Format(fmtStr, dt.Rows(i)(0), _
     dt.Rows(i)(1),dt.Rows(i)(2),dt.Rows(i)(3),percentIncrease))
  Next
End Sub
```

	city	country	pop2005	pop2015
▶	Bombay	India	18.2	22.6
	Calcutta	India	14.3	16.8
	Delhi	India	15.1	20.9

EXAMPLE 2: OUTPUT



CITY	COUNTRY	2005	2015	INCR.
Bombay	India	18.2	22.6	24 %
Calcutta	India	14.3	16.8	17 %
Delhi	India	15.1	20.9	38 %
Dhaka	Bangladesh	12.4	17.9	44 %
Jakarta	Indonesia	13.0	17.5	35 %
Lagos	Nigeria	11.0	17.0	55 %
Mexico City	Mexico	19.0	20.6	8 %
New York	USA	18.5	19.7	6 %
Sao Paulo	Brazil	18.2	20.0	10 %
Tokyo	Japan	35.2	36.2	3 %

Bound Controls

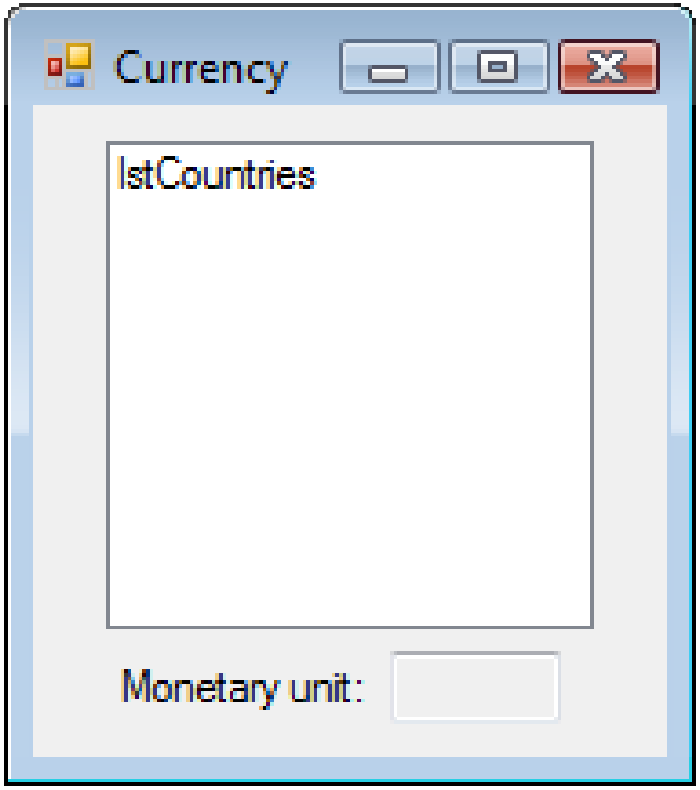
- A data table that is **bound** to a list box can transfer information automatically into the list box.
- The following statement binds a list box to a data table:

```
lstBox.DataSource = dt
```

- The contents of a specified field can be displayed in the list box by:

```
lstBox.DisplayMember = "country"
```

Example 3: Form



Display the list of countries. When the user clicks on a country, its monetary unit should be displayed.

Example 3: Code

```
Dim dt As New DataTable()

Private Sub frmCountries_Load(...) Handles MyBase.Load
   (Last five statements of boilerplate)
   lstCountries.DataSource = dt    'Bind list box
   lstCountries.DisplayMember = "country"
End Sub

Private Sub lstCountries_SelectedIndexChanged(...) _
                  Handles lstCountries.SelectedIndexChanged
   txtMonetaryUnit.Text = _
CStr(dt.Rows(lstCountries.SelectedIndex)("monetaryUnit")
End Sub
```

country	pop2005	monetaryUnit
Bangladesh	141.8	taka
Brazil	186.4	real
China	1316.8	yuan

Example 3: Output

10.2 Relational Databases and SQL

- Primary and Foreign Keys
- SQL
- Four SQL Requests
- The DataGridView Control
- Changing the Contents of a Database
- Calculated Columns with SQL

Primary Keys

- A **primary** key is used to uniquely identify each record.
- Databases of student enrollments in a college usually use a field of Social Security numbers as the primary key.

- What's the primary key used at SFU for students?
- For ICBC drivers licenses?
- Why wouldn't names be a good choice as a primary key?

Primary Key Fields

- Unique identifier of each row in the database table

- Specified when database is created.
- Every record must have an entry in the primary-key field
- Two records cannot have the same entry in the primary-key field
- This pair of requirements is called the **Rule of Entity Integrity**

Two or More Tables

- When a database contains two or more tables, the tables are usually related
- For instance, the two tables Cities and Countries are related by their country field.
- Notice that every entry in Cities.country appears uniquely in Countries.country and Countries.country is a primary key.
- We say that Cities.country is a **foreign key** of Countries.country.

country	pop2005	monetaryUnit
Bangladesh	141.8	taka
Brazil	186.4	real
China	1316.8	yuan
India	1103.4	rupee
Indonesia	222.8	rupiah
Japan	128.1	yen
Mexico	107	peso

city	country	pop2005	pop2015
Bombay	India	18.2	22.6
Calcutta	India	14.3	16.8
Delhi	India	15.1	20.9
Dhaka	Bangladesh	12.4	17.9
Jakarta	Indonesia	13	17.5
Lagos	Nigeria	11	17
Mexico City	Mexico	19	20.6
New York	USA	18.5	19.7
Sao Paulo	Brazil	18.2	20
Tokyo	Japan	35.2	36.2

Foreign Keys

- Foreign keys can be specified when a table is first created. Visual Basic will insist on the **Rule of Referential Integrity.**
- This Rule says that each value in the foreign key must also appear in the primary key of the other table.

JOIN

- A foreign key allows Visual Basic to link (or **join**) together two tables from a relational database
- When the two tables Cities and Countries from MEGACITIES.MDB are joined based on the foreign key Cities.country, the result is the table in the next slide.
- The record for each city is expanded to show its country's population and its monetary unit.

A Join of Two Tables

city	country	pop2005	pop2015
Bombay	India	18.2	22.6
Calcutta	India	14.3	16.8
Delhi	India	15.1	20.9
Dhaka	Bangladesh	12.4	17.9
Jakarta	Indonesia	13	17.5
Lagos	Nigeria	11	17
Mexico City	Mexico	19	20.6
New York	USA	18.5	19.7
Sao Paulo	Brazil	18.2	20
Tokyo	Japan	35.2	36.2

country	pop2005	monetaryUnit
Bangladesh	141.8	taka
Brazil	186.4	real
China	1316.8	yuan
India	1103.4	rupee
Indonesia	222.8	rupiah
Japan	128.1	yen
Mexico	107	peso

Cities. city	Cities. country	Cities. pop2005	Cities. pop2015	Countries. country	Countries. pop2005	Countries. monetaryUnit
Bombay	India	18.2	22.6	India	1103.4	rupee
Calcutta	India	14.3	16.8	India	1103.4	rupee
Delhi	India	15.1	20.9	India	1103.4	rupee
Dhaka	Bangladesh	12.4	17.9	Bangladesh	141.8	taka
Jakarta	Indonesia	13.0	17.5	Indonesia	222.8	rupiah
Lagos	Nigeria	11.0	17.0	Nigeria	131.5	naira
Mexico City	Mexico	19.0	20.6	Mexico	107	peso
New York	USA	18.5	19.7	USA	298.2	dollar
Sao Paulo	Brazil	18.2	20.0	Brazil	186.4	real
Tokyo	Japan	35.2	36.2	Japan	128.1	yen

SQL

- **Structured Query Language** developed for use with relational databases
- Very powerful language
- Allows for the request of specified information from a database
- Allows displaying of information from database in a specific format

Four SQL Requests

- Show the records of a table in a specified order

```
SELECT * FROM Table1 ORDER BY field1 ASC
```

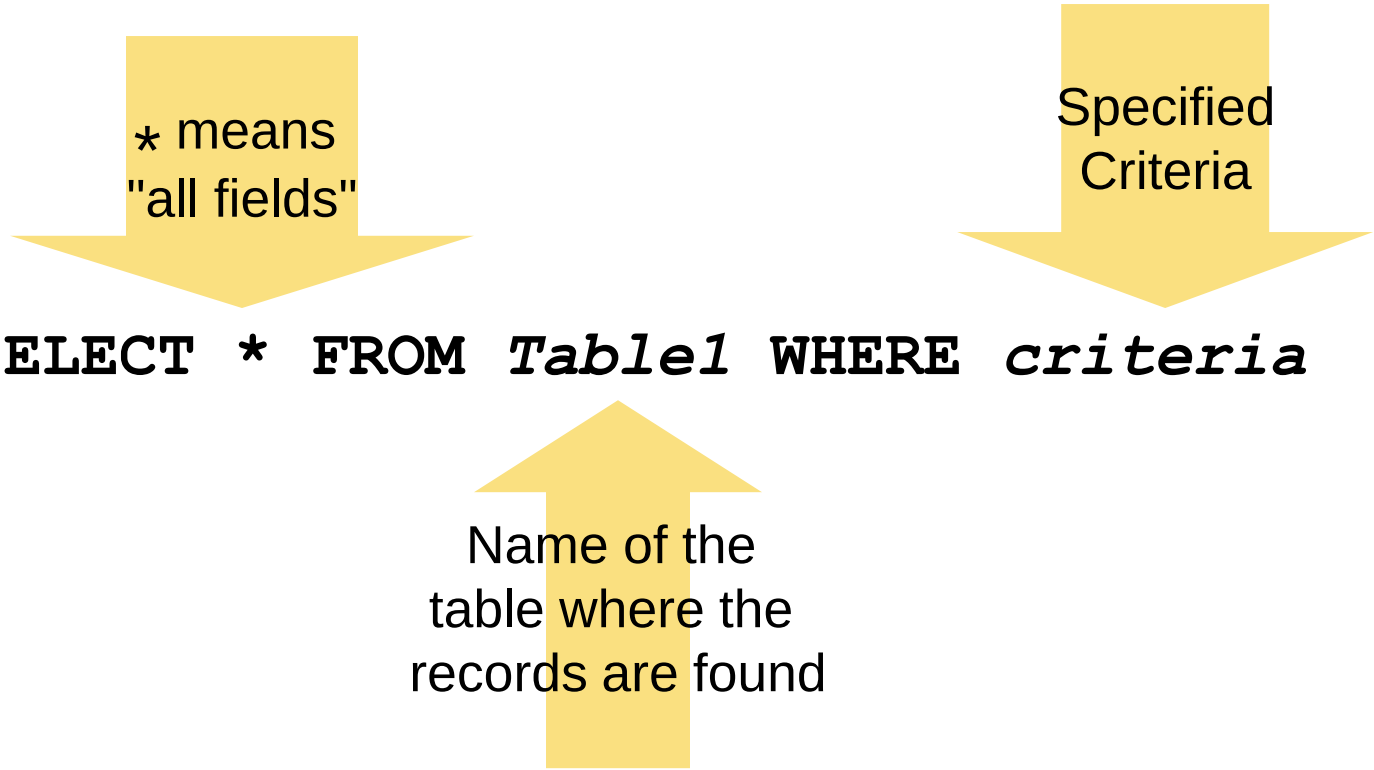
- or

```
SELECT * FROM Table1 ORDER BY field1 DESC
```

* means "all fields"

Specifies ASCending Or DESCending

Show just the records that meet certain criteria



JOIN THE TABLES TOGETHER

- connected by a foreign key, and present the records as in previous requests

```
SELECT *
FROM Table1 INNER JOIN Table2
     ON foreign field = primary field
WHERE criteria
```

Make available just some of the fields

- of either the basic tables or the joined table.

```
SELECT field1, field2, . . .,
  fieldN
FROM Table1
WHERE criteria
```

Criteria Clause

- A string containing a condition of the type used with If blocks.
- Uses the standard operators <, >, and =
- Also can use the operator LIKE.
- LIKE uses the wildcard characters "_" and "%" to compare a string to a pattern.

EXAMPLES USING LIKE

- An underscore character stands for a single character in the same position as the underscore character.
- The pattern “B_d” is matched by “Bid”, “Bud”, and “Bad”.

- A percent sign stands for any number of characters in the same position as the asterisk.
- The pattern “C%r” is matched by “Computer”, “Chair”, and “Car”.

SELECT CLAUSE

- **SELECT *fields***
  FROM *clause*

- *fields* is either * (to indicate all fields) or a sequence of the fields to be available (separated by commas)

- *clause* is either a single table or a join of two tables

JOIN CLAUSE

- A join of two tables is indicated by a clause of the form

table1* INNER JOIN *table2* ON *foreign key of table1=primary key of table2

- Appending **WHERE *criteria*** to the end of the sentence restricts the records to those satisfying *criteria*.

- Appending **ORDER BY *field(s)* ASC (or DESC)** presents the records ordered by the specified field or fields.

General SQL statements

SELECT *www* FROM *xxx* WHERE *yyy*
ORDER BY *zzz*

- **SELECT *www* FROM *xxx*** is always present
- May be accompanied by one or both of **WHERE *yyy*** and **ORDER BY *zzz***.
- The ***xxx*** portion might contain an **INNER JOIN** phrase.

More on SQL statements

- The single quote, rather than the normal double quote, is used to surround strings.
- Fields may be specified with the table they come from by ***`tableName.fieldName`***

Virtual Tables

- SQL statements create a new "virtual" table from existing tables.

```
SELECT city, pop2015 FROM Cities
  WHERE pop2015>=20
```

Results in "virtual" table

city	pop2015
Bombay	22.6
Delhi	20.9
Mexico City	20.6
Sao Paulo	20.0
Tokyo	36.2

Another Virtual Table

```
SELECT * FROM Countries WHERE
  country LIKE 'I%' ORDER BY
  pop2005 ASC
```

- Results in "virtual" table

country	pop2005	monetaryUnit
Indonesia	222.8	rupiah
India	103.4	rupee

Views

- "Virtual" tables don't exist physically.
- For all practical purposes, Visual Basic acts as if they did.
- You may also see a "virtual" table called a **view**.

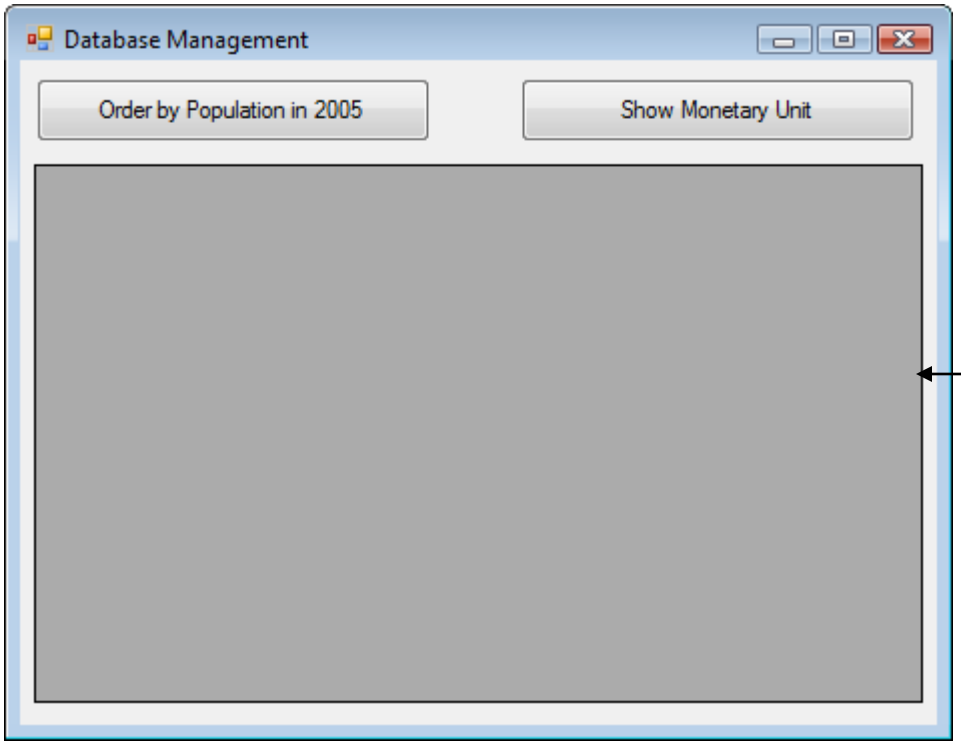
The DataGridView Control

- The DataGridView, displays the values for an entire view in a table format similar to the table displayed by Database Explorer.
- The prefix for the name of a DataGridView control is *dgv*.
- After a data table has been filled, the statement

```
dgvDisplay.DataSource = dt
```

displays the contents of the data table *dt*.

Example 1: Form



Example 1: Code

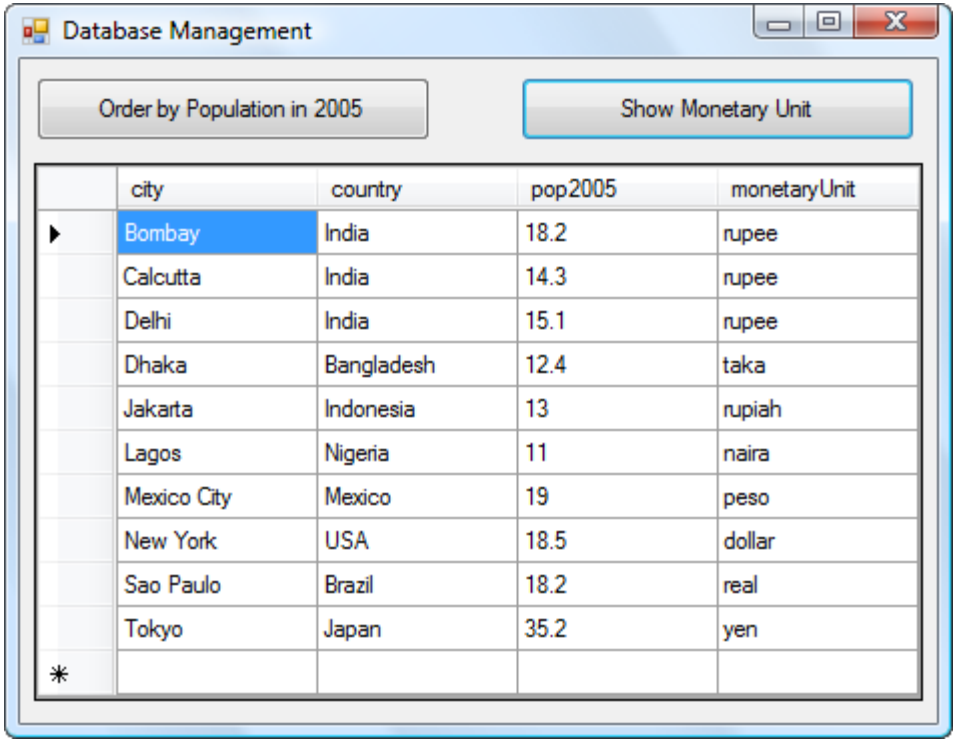
```
Private Sub frmCities_Load(...) Handles MyBase.Load
  UpdateGrid("Select * From Cities")
End Sub

Private Sub btnOrderbyPop_Click(...) Handles
  btnOrderbyPop.Click
  UpdateGrid("Select * From Cities Order By pop2005 ASC")
End Sub

Private Sub btnShowMonUnit_Click(...) _
                                    Handles
  btnShowMonUnit.Click
  UpdateGrid("SELECT city, Cities.country, " & _
             "Cities.pop1995, monetaryUnit " & _
             "FROM Cities INNER JOIN Countries " & _
             "ON Cities.country=Countries.country " & _
             "ORDER BY city ASC")
End Sub
```

Example 1: Code continued

```
Sub UpdateGrid(ByVal sqlStr As String)
  Dim dt As New DataTable()
  Dim connStr As String ="Provider=Microsoft.Jet.OLEDB.4.0;" & _
                         "Data Source = MEGACITIES.MDB"
  Dim dataAdapter As New OleDb.OleDbDataAdapter(sqlStr, connStr)
  dataAdapter.Fill(dt)
  dataAdapter.Dispose()
  dgvDisplay.DataSource = dt
End Sub
```

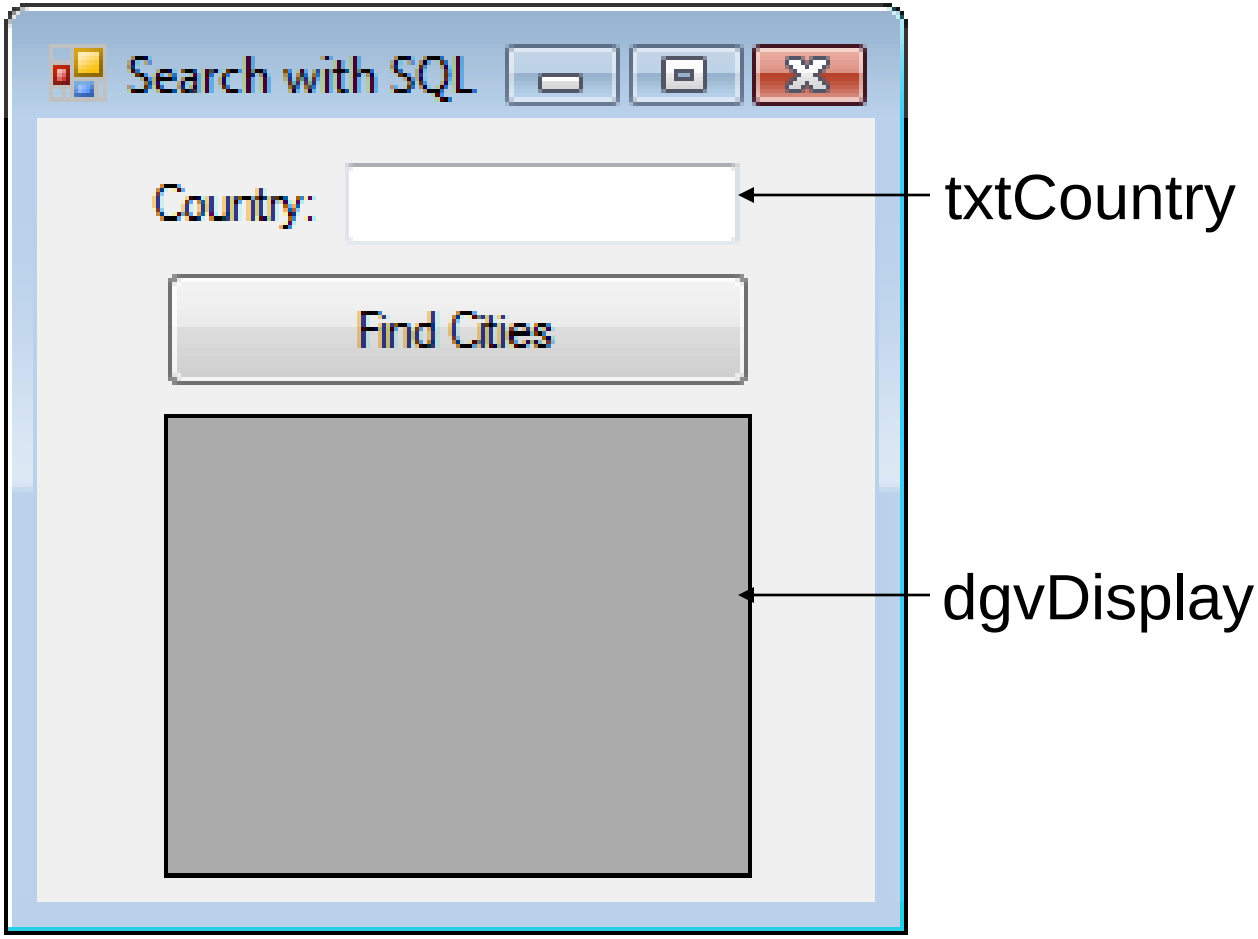
EXAMPLE 1: OUTPUT

Click on the “Show Monetary Unit” button.



	city	country	pop2005	monetaryUnit
▶	Bombay	India	18.2	rupee
	Calcutta	India	14.3	rupee
	Delhi	India	15.1	rupee
	Dhaka	Bangladesh	12.4	taka
	Jakarta	Indonesia	13	rupiah
	Lagos	Nigeria	11	naira
	Mexico City	Mexico	19	peso
	New York	USA	18.5	dollar
	Sao Paulo	Brazil	18.2	real
	Tokyo	Japan	35.2	yen
*				

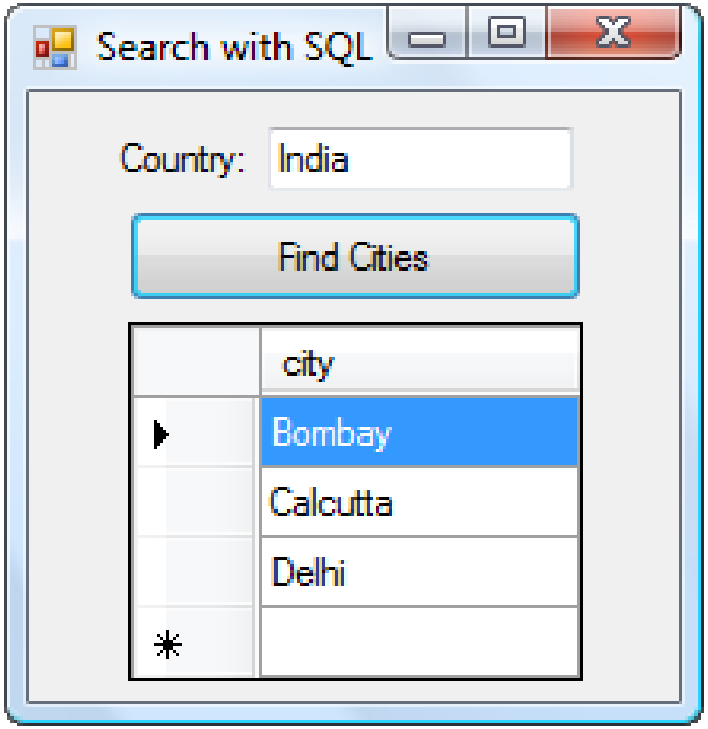
Example 2: Form



Example 2: Code

```
Private Sub btnFindCities_Click(...) _
                            Handles btnFindCities.Click
  UpdateGrid("SELECT city FROM Cities WHERE" & _
             "country = '" & txtCountry.Text & _
             "' ORDER BY city ASC")
End Sub

Sub UpdateGrid(ByVal sqlStr As String)
  (Boilerplate, except for Dim sqlStr statement)
  If dt.Rows.Count = 0 Then
    MessageBox.Show("No cities from that country " & _
                    "in the database")
  Else
    dgvDisplay.DataSource = dt
  End If
End Sub
```

EXAMPLE 2: OUTPUT



Changing the Contents of a Database

- Data grid views can also be used to add, modify, and delete records from a database.
- After a DataAdapter has been created, the statement

```
Dim commandBuilder As New _

OleDbCommandBuilder(dataAdapter)
```

will automatically generate the commands used for the Insert, Update, and Delete operations.

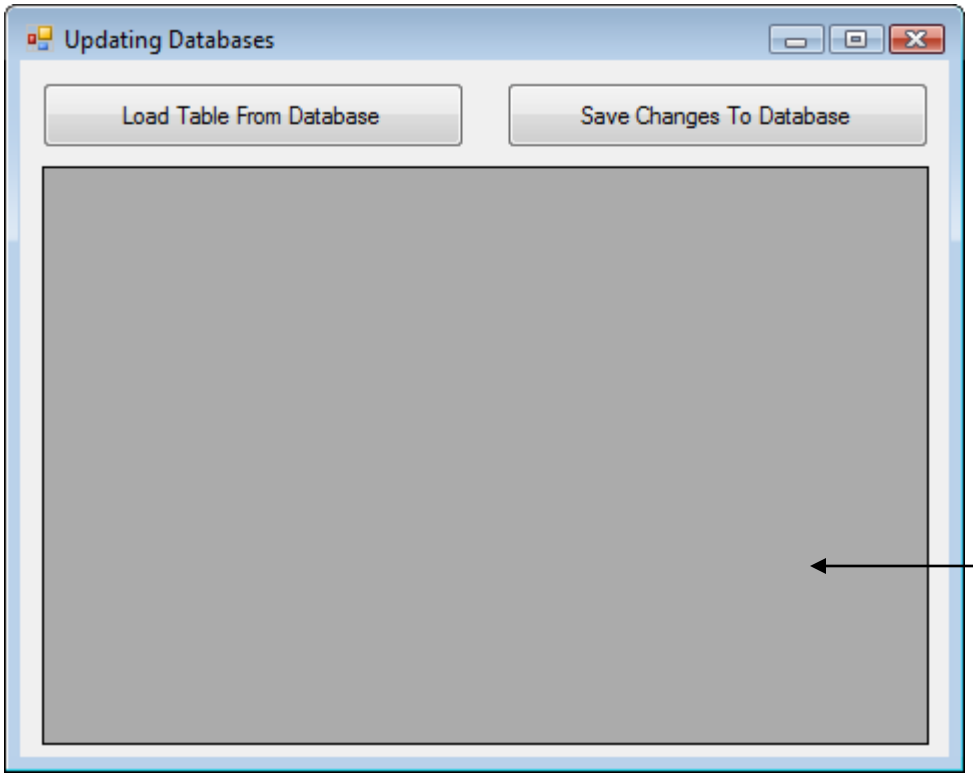
Using the DataAdapter to Change a Database

- If *changes* is an Integer variable, then the statement

```
changes = dataAdapter.Update(dt)
```

will store all of the insertions, updates, and deletions made in the data table to the database and assign the number of records changed to the variable *changes*.

Example 3: Form



Example 3: Partial Code

```
Dim connStr As String =
  "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                          "Data Source=MEGACITIES.MDB"
Dim sqlStr As String = "SELECT * FROM Cities"
Dim dt As New DataTable()
Private Sub btnLoad_Click(...) Handles btnLoad.Click
  dt.Clear()
  Dim dataAdapter As New
  OleDb.OleDbDataAdapter(sqlStr, connStr)
  dataAdapter.Fill(dt)
  dataAdapter.Dispose()
  dgvDisplay.DataSource = dt
End Sub
```

Example 3: Code continued

```
Private Sub btnSave_Click(...) Handles btnSave.Click
  Dim changes As Integer
  Dim dataAdapter As New OleDb.OleDbDataAdapter(sqlStr, connStr)
  Dim commandBuilder As New _
                          OleDb.OleDbCommandBuilder(dataAdapter)
  changes = dataAdapter.Update(dt)
  dataAdapter.Dispose()
  If changes > 0 Then
    MessageBox.Show(changes & " changed rows.")
  Else
    MessageBox.Show ("No changes made.")
  End If
End Sub
```

CALCULATED COLUMNS WITH SQL

In the SQL statement

```
SELECT field1, field2,..., fieldN FROM
  Table1
```

one of the field listings can be an expression involving other fields, followed by a clause of the form "AS *header*". If so, a new column will be created whose values are determined by the expression and having the stated header. For instance, using

```
sqlStr = "SELECT city, Round(pop2015-pop2005, 1)" & _
         "AS popGrowth FROM Cities"
```

to fill the table produces the output shown in the next slide.

Calculated Columns with SQL

	city	popGrowth
	Bombay	4.4
	Calcutta	2.5
	Delhi	5.8
	Dhaka	5.5
	Jakarta	4.5
	Lagos	6
	Mexico City	1.6
	New York	1.2
	Sao Paulo	1.8
	Tokyo	1