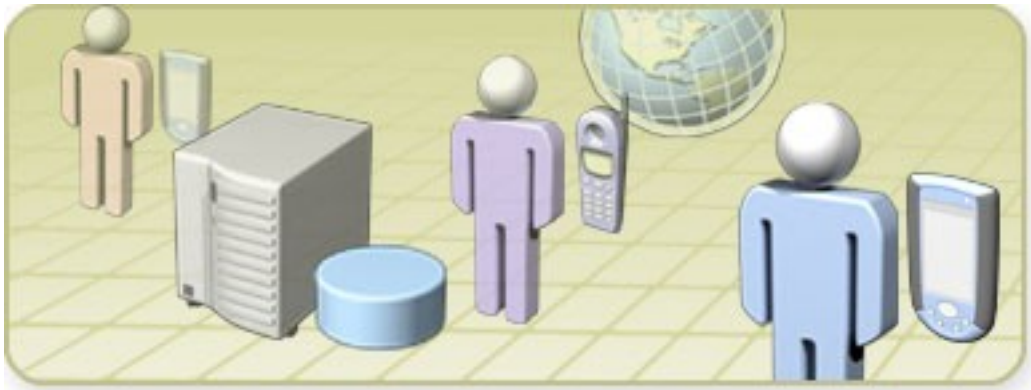
Module 3: Working with Local Data



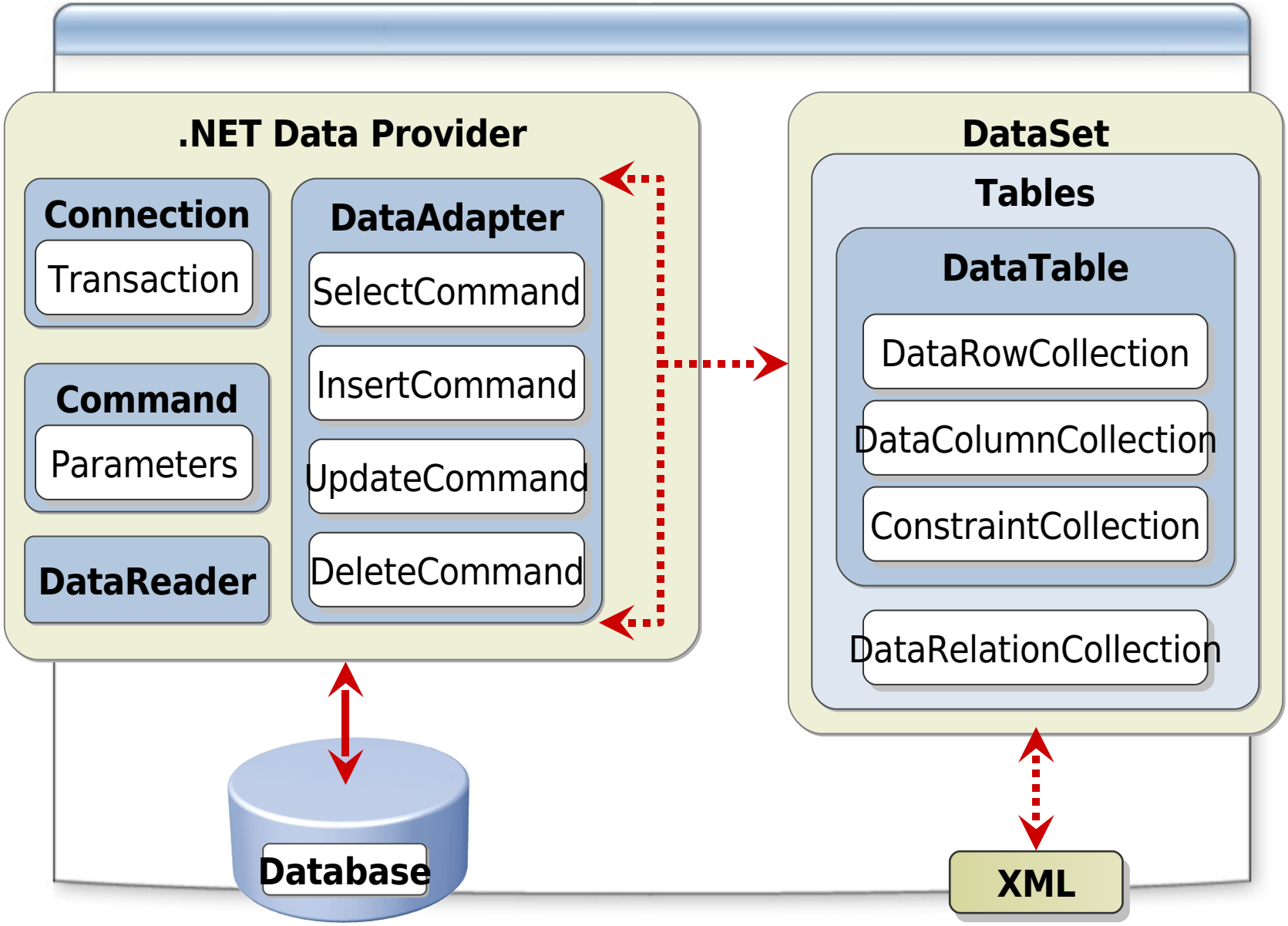
Overview

- Using DataSets
- Using XML
- Using SQL Server CE

Lesson: Using DataSets

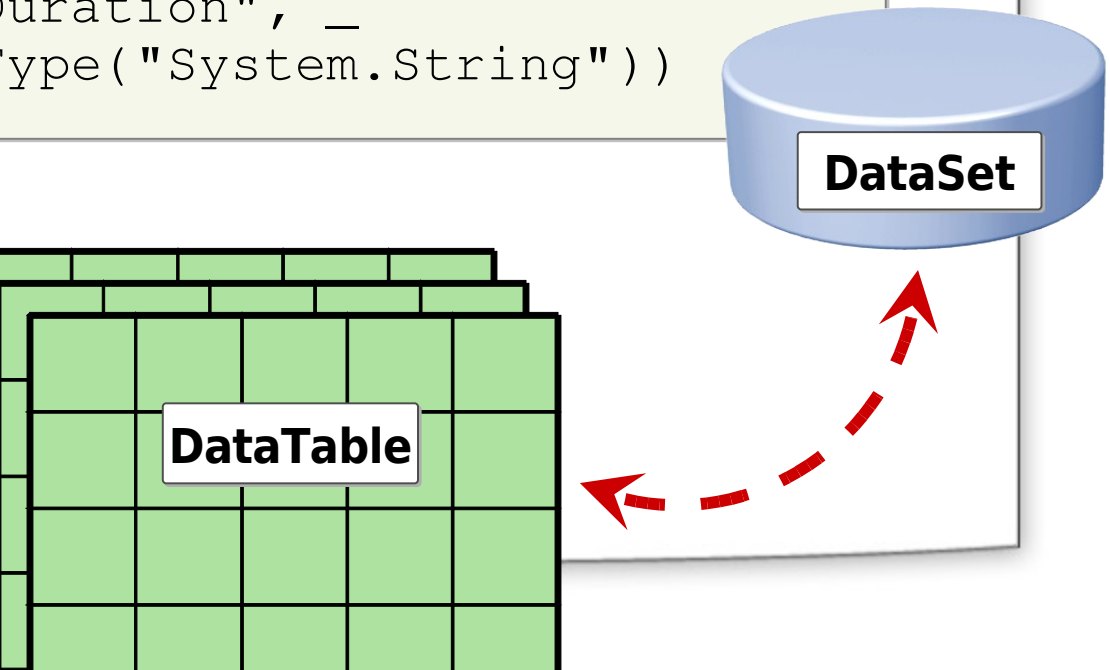
- ADO.NET Model
- Creating a DataSet
- Filling the DataSet
- Persisting the DataSet as an XML File
- Binding to a DataSet
- Using a DataGrid

ADO.NET Model



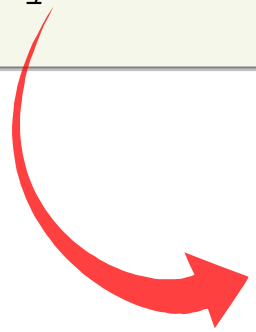
Creating a DataSet

```
Dim myDS As New DataSet("Project")
Dim myDT As DataTable = _
myDS.Tables.Add("Task")
myDT.Columns.Add("Name", _
   System.Type.GetType("System.String"))
myDT.Columns.Add("Start", _
   System.Type.GetType("System.String"))
myDT.Columns.Add("Duration", _
   System.Type.GetType("System.String"))
```



Filling the DataSet

```
Dim myDR As DataRow = _
    myDS.Tables("Task").NewRow()
myDR("Name") = "Design Code"
myDR("Start") = "2/1/2003"
myDR("Duration") = "2 days"
myDS.Tables("Task").Rows.Add(myDR)
```



Name	Start	Duration
Design UI	1/1/2003	1 day
Design Code	2/1/2003	2 days

Practice: Using DataSets to Access Data

1. Creating and filling a DataSet
2. Adding to a DataSet from a form

Persisting the DataSet as an XML File

- DataSet provides volatile storage
- Use the WriteXml method to save data

```
myDataSet.WriteXml("win\tmp.xml")
```

- Use the ReadXml method to populate data from the file

```
Dim myDataSet As New DataSet()
myDataSet.ReadXml("win\tmp.xml")
```

Practice: Persisting the DataSet as XML

1. Save a DataSet as an XML file
2. Verify the XML file

Binding to a DataSet

- **DataSource property**
  - Binds a control to the data source
  - Provides link from mobile application to DataSet

```
Dim dt As DataTable = _
   tmpDS.Tables("Info")
'Bind to the list box
listBox1.DataSource = dt
'Set column to bind to
listBox1.DisplayMember = "Name"
```

Using a DataGrid

- Provides a user interface for entire tables in a DataSet
- Rich formatting capabilities
- DataGrid is bound to a data source at run time (not design time)

Practice: Binding a Control to a DataSet

1. Binding a control to a DataSet
2. Verifying the binding

Lesson: Using XML

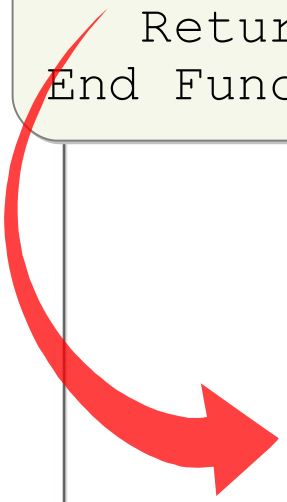
- Supported XML Classes
- Building an XmlDocument
- Reading an XmlDocument

Supported XML Classes

- **XmlTextReader and XmlTextWriter**
  - Forward-only parsers of XML data
  - Better performance because there is no in-memory caching
- **XmlDocument**
  - Data can be read into the object
  - After modification, data can be read from the object back to a stream

Building an XmlDocument

```
Private Function BuildXmlDocument() As XmlDocument
    Dim myXmlDoc As New XmlDocument()
    Dim NewNode As XmlNode
    NewNode = myXmlDoc.CreateElement("Project")
    myXmlDoc.AppendChild(NewNode)
    NewNode = myXmlDoc.CreateElement("Task")
    NewNode.InnerText = "Write Code"
    myXmlDoc.DocumentElement.AppendChild(NewNode)
    Return myXmlDoc
End Function
```



```
<Project>
    <Task> Write Code </Task>
</Project>
```

Reading an XmlDocument

- **XmlDocument is an in-memory DOM tree**
- **Navigate DOM using properties and methods of XmlNode class**
- **Values of nodes can be extracted and**

```
Private Sub DisplayXmlDocument(myXmlDoc As XmlDocument)
   Dim oNodes As XmlNodeList = _
      myXmlDoc.DocumentElement.ChildNodes
   Dim sbXMLDisplay As New StringBuilder()
   Dim TaskNode As XmlNode
   For Each TaskNode In oNodes
      Dim PropertyNode As XmlNode
      For Each PropertyNode In TaskNode
         sbXMLDisplay.Append((PropertyNode.Name + ": " + _
            PropertyNode.InnerText + ControlChars.Lf))
      Next PropertyNode
   Next TaskNode
   MessageBox.Show(sbXMLDisplay.ToString())
End Sub
```

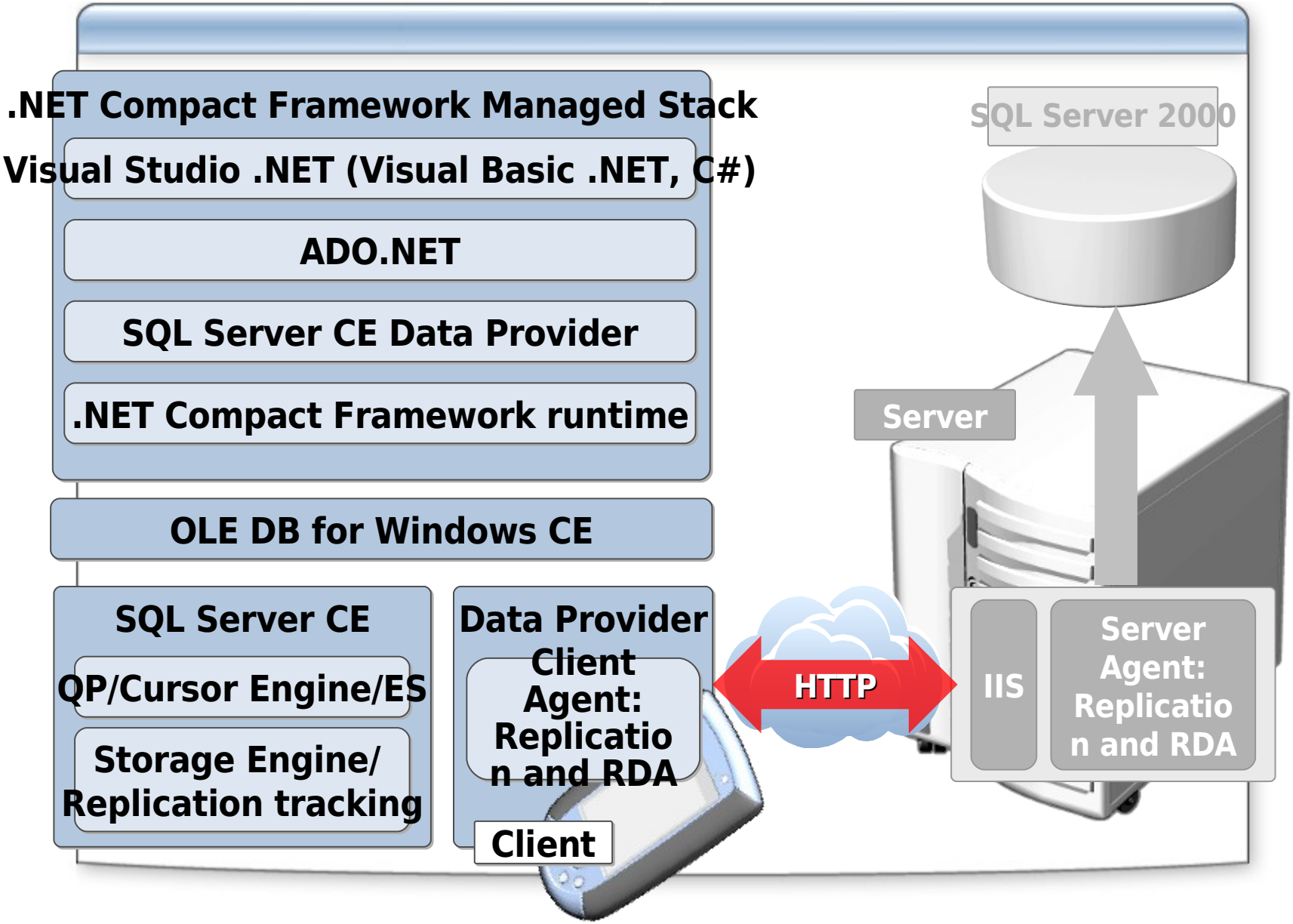
Practice: Adding an Element to an XmlDocument

1. Add an element to an XmlDocument
2. Verify that the element is added

Lesson: Using SQL Server CE

- SQL Server CE Storage Architecture
- Working with SQL Server CE
- Using SQL Server CE Query Analyzer
- Using a SQL Server CE Data Connector
- Filling a DataSet from SQL Server CE
- Using Parameterized Queries
- Reading Data
- Updating SQL Server CE from the DataSet

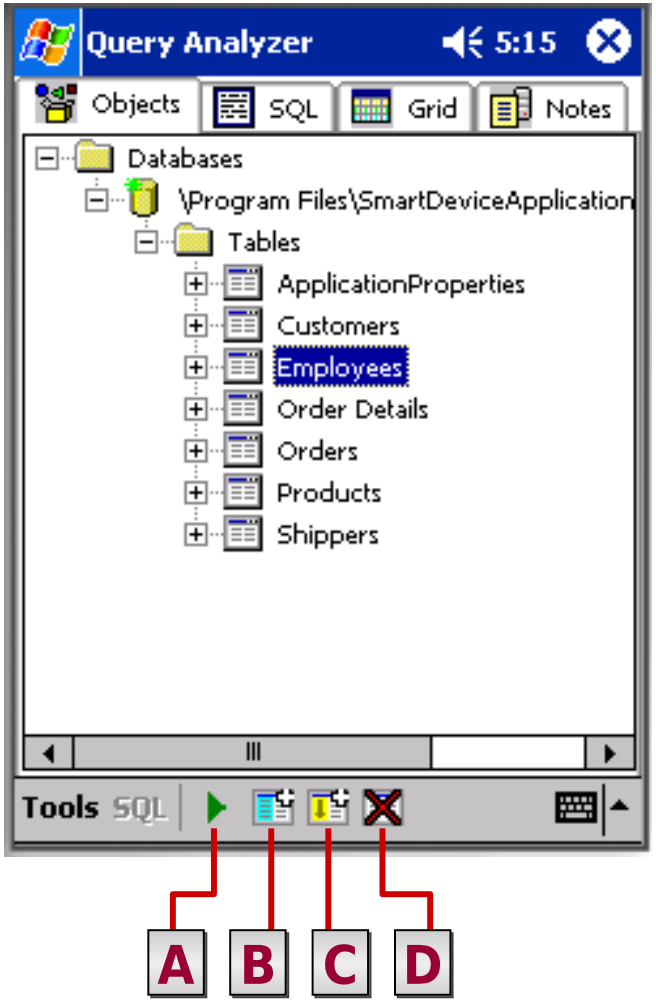
SQL Server CE Storage Architecture



Working with SQL Server CE

- **Available database storage in Pocket PC is limited**
- **SQL Server CE 2.0 Features (see list in Student Notes)**
- **Visual Studio .NET automatically configures development environment for use with SQL Server CE**
  - SQL Server CE 2.0 is included with the installation of Visual Studio .NET
  - Must still configure IIS and Windows CE-based device
- **Installing SQL Server CE on the client device**
  - Add a reference to **System.Data.SqlServerCe**

Using SQL Server CE Query Analyzer



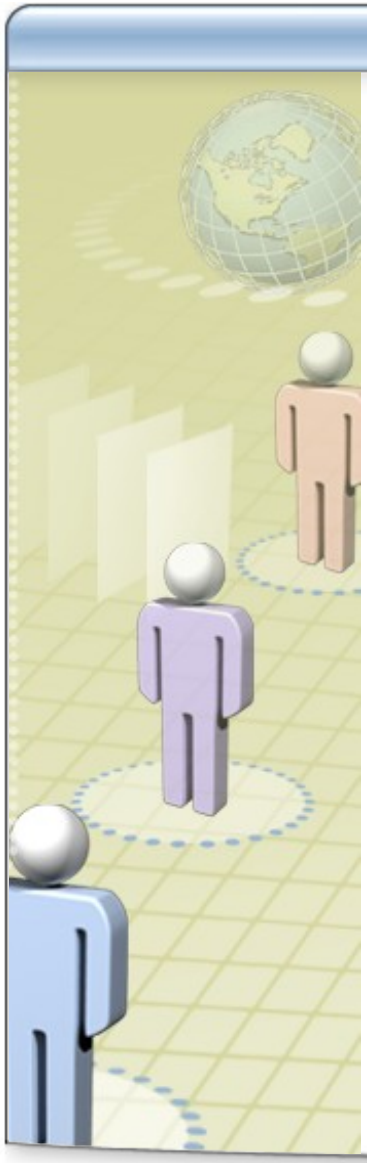
A Tap to execute a SELECT * FROM **Employees** statement

B Tap to add a column to the **Employees** table

C Tap to create an index on the **Employees** table

D Tap to drop the **Employees** table

Demonstration: Using the SQL Server CE Query Analyzer



Using a SQL Server CE Data Connector

- **Connection string to SQL Server requires a database provider**

```
SqlConnString = "Provider=sqloledb;
   Data Source=London;
   Initial Catalog=Northwind"
```

- Connection string to SQL Server CE is similar, but a database provider is not specified

```
SqlCeConnString =
   "Data Source=My Documents\Northwind.sdf"
```

Filling a DataSet from SQL Server CE

- **Establish a connection**
- **Create a data adapter**
- **Call the Fill method**

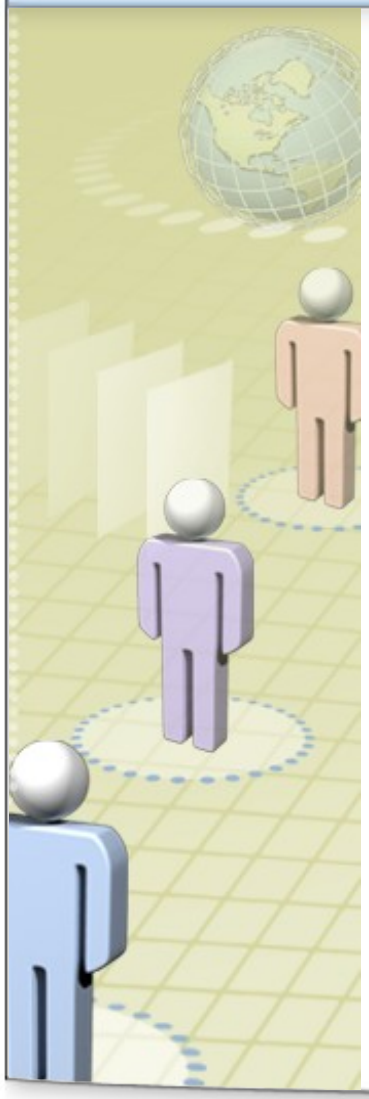
```
Dim myAdapter As New SqlCeDataAdapter()
myAdapter.TableMappings.Add("Table", "Titles")
cn.Open()
Dim myCommand As New SqlCeCommand( _
   "SELECT * FROM Titles", cn)
myCommand.CommandType = CommandType.Text
myAdapter.SelectCommand = myCommand
Dim ds As New DataSet()
myAdapter.Fill(ds)
```

Using Parameterized Queries

- **Parameterized queries**
  - Have built-in input validation
  - Execute quicker and are more secure

```
' Insert data into the table.
SQL = "INSERT INTO Titles (TitleID,TitleName)
    VALUES (?,?)"
cmd.CommandText = SQL
cmd.Parameters.Add("@TitleID",
    System.Data.SqlDbType.NChar, 5)
cmd.Parameters.Add("@TitleName",
     System.Data.SqlDbType.NVarChar, 40)
cmd.Parameters["@TitleID"].Value = "MSCF1"
cmd.Parameters["@TitleName"].Value = "Compact Framework"
cmd.ExecuteNonQuery()
```

Demonstration: Creating a Local Data Store



- Create a SQL Server CE table
- Populate the table

Reading Data

- **The ExecuteReader method runs the SQL or stored procedure and returns a DataReader object**
- **The Read method moves the DataReader to the next record**
- **Read must be called before data access methods are used**

```
Dim reader As _
    System.Data.SqlServerCe.SqlCeDataReader = _
    cmdTxt.ExecuteReader()
While reader.Read()
    MessageBox.Show(reader.GetString(0))
End While
```

Updating SQL Server CE from the DataSet

- Save new or modified DataSet data to SQL Server CE
- Update method updates the SQL Server CE table with changes made in the DataSet

```
myConn.Open()
Dim custDS As New DataSet()
myDataAdapter.Fill(custDS)
'code to modify data in dataset here
myDataAdapter.Update(custDS, myTableName)
myConn.Close()
```

Review

- Using DataSets
- Using XML
- Using SQL Server CE

Lab 3: Working with Local Data

- Exercise 1: Reading an XML File into a DataSet
- Exercise 2: Appending Data to the XML File
- Exercise 3: Saving Data to a SQL Server CE Table