Data Bases and ADO.NET

Relational Databases

- Most data handling today done with relational databases
  - Logical representations of data that allow relationships among data to be considered without concern for the physical structure of the data
  - Composed of tables (like spreadsheets)
  - Lots of proprietary formats
  - Some database sources:
    - Microsoft SQL Server
    - Access
    - Oracle
    - Sybase
  - Microsoft ADO.NET can handle data from multiple locations (servers) stored in different formats

ADO.NET

- Based on Microsoft’s ActiveX Data Objects
  - Data stored and transferred in Extensible Markup Language (XML)
  - Allows simple access to database data in many formats
    - Easy-to-use classes represent tables, columns, rows inside relational databases
    - Introduces DataSet class representing a set of data from related tables encapsulated as a single unit preserving the integrity of the relationships between them
  - Basic types of database connections:
    - SQLClient for SQL Server
    - OleDb for all other database formats
      - Can be used to obtain/update data from sources such as Access, Oracle, Sybase, DB2, etc.
    - Many others supported

Database Terminology

- Each database file can hold multiple tables
- A table:
  - Each row represents data for one item
    - Called a record
  - Each column used to store a different data element
    - Elements represented in columns are called fields

Last Name	First Name	Phone
Smith	John	777-1111
Jones	Mary	777-2222

Records

Fields

Database Terminology, continued

- Primary Key Field
  - Used to identify a record in a table
  - A field that contains unique data not duplicated in other records in the table
    - e.g., social security number for employees
- Current Record
  - Anytime a table is open, one record is considered to be the current record
    - As we move from record to record in a table the current record changes

Queries

- A query retrieves information from a database
- SQL (Structured Query Language) is the standard for expressing queries
  - We won’t need to be experts in using it since Visual Studio .NET provides a “Query Builder” tool to construct SQL queries

XML Data

- Industry standard for storing and transferring data
  - Specs at: www.w3.org/XML
- Most database formats store data in binary
  - Cannot be accessed by other systems or pass through firewalls
- Data stored in XML is text
  - Identified by tags similar to HTML tags
    - Not predefined as in HTML
    - We can define our own XML tags to indicate their content
      - So very flexible for describing any kind of data
- Use of XML allows programs to communicate even though they are written in different languages and run on different hardware

Overview of XML

- Machine-Readable and Human-Readable Data
- Defines the Data Content and Structure
- Allows Developer to Define his/her Own Tags and Attributes

```
<employee>
  <name>Jake</name>
  <salary>25000</salary>
  <region>Ohio</region>
</employee>
```

XML Schemas

- A schema describes fields, data types, and any constraints on the data
- Defines the structure of an XML document
- A schema is expressed in XML as well
- Use of schemas permits strong typing and data validation

Using ADO.NET

- Data from a database can be displayed on a Windows Form or a Web Form
- Add controls to the form and bind the data to the controls
  - – Controls can be what we've already seen:
    - label, text box, list box, combo box, etc.
  - – Or special controls designed just for data:
    - DataGridView
- ADO.NET classes are in the <u>System.Data</u> namespace

Reading Database Data with a DataReader

- A simple way to go – like network & file I/O
- Connected model
- Create and open a DataConnection
  - Establishes a link to the data source, which is a specific database file and server
- Then create a Database Command associated with the connection that specifies the data to be accessed
  - This is an SQL query
- Execute the command
- Use a DataReader to read the data
- Display the data

Creating a Connection

- ADO.NET provides several types of Connection objects
- Two important ones:
  - SqlConnection
    - Only for connecting to a Microsoft SQLServer database
  - OleDbConnection
    - For connecting to other database systems such as Access
- Can set up a data connection by constructing a Connection object
  - Connection string specifies details
- Or use Visual Studio's "Server Explorer" to set one up
  - Start it with "View" | "Server Explorer"

An Example: Manual Coding to Read the contents of a Database Table

- Windows Form Example: DataReadingForm
  - Reads and displays data from a small Access database: rnrbooks.mdb
    - Contains two tables:
      - “Books” with the following fields:
        - » ISBN, Title, Author, Publisher, and other fields
      - “Subjects” with the following fields:
        - » SubjectCode, Subject

DataReadingForm Example

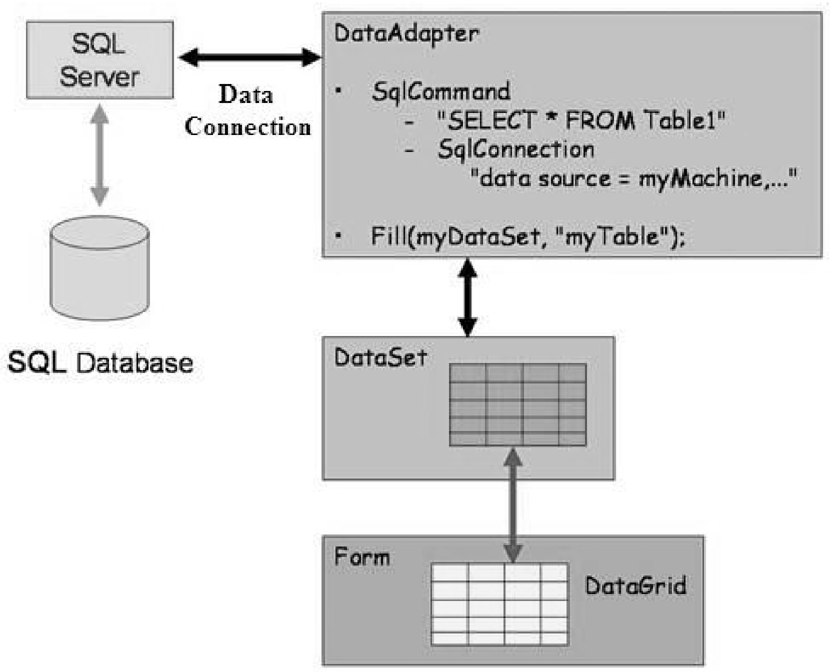
- The important code:

```
OleDbConnection thisConnection = new
        OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
        Source=C:\360\Programs_managedVCSharp\rnrbooks.mdb");
thisConnection.Open();
OleDbCommand thisCommand = thisConnection.CreateCommand();
thisCommand.CommandText = "SELECT Title, Author FROM Books";
OleDbDataReader thisReader = thisCommand.ExecuteReader();  //create reader
while (thisReader.Read())
{   //display DataReader's data rows in a text box called displayTextBox
    displayTextBox.Text += "\r\n" + thisReader["Title"] + thisReader["Author"];
}
thisReader.Close();
thisConnection.Close();
```

Disconnected ADO.NET Data Access

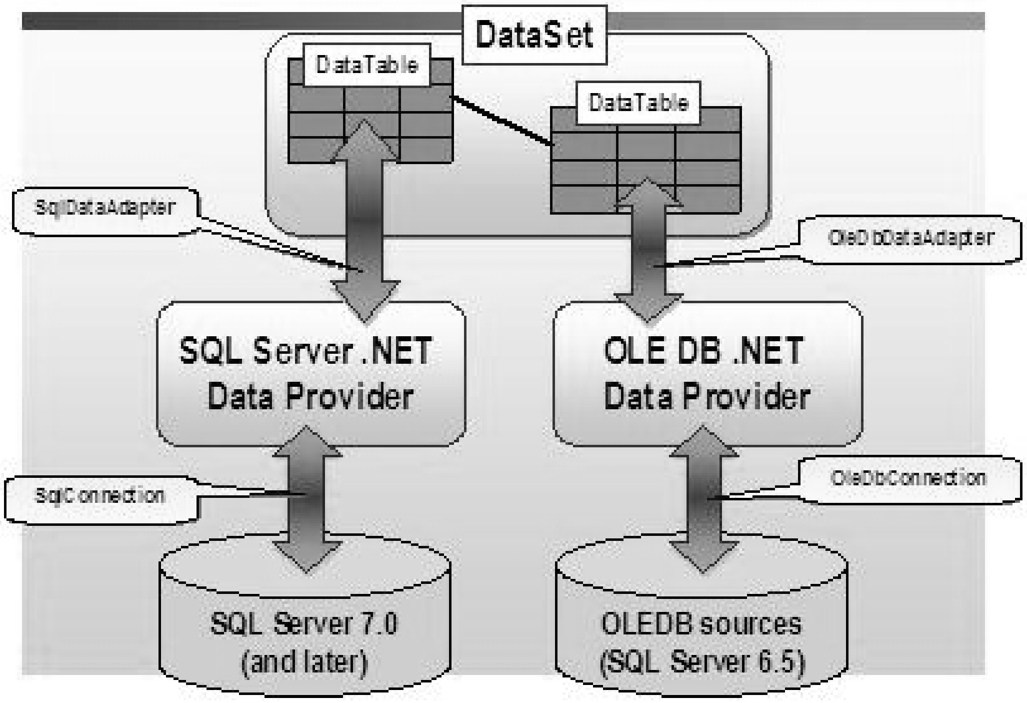
- 1. Set up a Data Connection
  - Establishes a link to the data source
- 2. Set up a DataAdapter
  - Handles retrieving and updating the data
  - Data adapter uses "Command" objects to retrieve/store records from/to the database and can be used to:
- 3. Create a DataSet:
  - A temporary set of data tables stored in the computer's memory
  - ADO.NET datasets are disconnected
    - So data in memory does not keep an active connection to data source
    - Much better: Many more clients can connect and use the data server
  - DataAdapters's Fill(-,-) method gets a data table into the DataSet
    - Uses SQL in a "Command" object to specify data to retrieve/update
- 4.. Add controls on the Windows Form or Web Form
  - Display the data from the DataSet and allow user interaction
- 5. Write C# code to put the data into the controls

Connections, Data Adapters, Datasets



DataAdapter

- SqlCommand
  - "SELECT * FROM Table1"
  - SqlConnection
    "data source = myMachine,..."
- Fill(myDataSet, "myTable");


DataSet
DataTable
DataTable
SqlDataAdapter
OleDbDataAdapter
SQL Server .NET Data Provider
OLE DB .NET Data Provider
SqlConnection
OleDbConnection
SQL Server 7.0 (and later)
OLEDB sources (SQL Server 6.5)

Example using a DataAdapter and a DataSet

- <u>DataReadingWithDataSet</u>
  - Also reads data from the rnrbooks.mdb database
- Also coded manually

Steps to Follow

– Instantiate and Open an OleDbConnection to the DB

  OleDbConnection thisConnection=new OleDbConnection (@"Provider=
    Microsoft.Jet.OLEDB.4.0; Data Source=
    C:\360\Programs_managedVCSharp\rnrbooks.mdb");
  thisConnection.Open( );

  • @-string literal to avoid escape chars: @"c:\x\a.txt" is equivalent to "c:\\x\\a.txt"

– Create an OleDbDataAdapter specifying an SQL SELECT command using the Connection

  OleDbDataAdapter thisAdapter =new OleDbDataAdapter("SELECT ISBN, Title, Author
    FROM Books", thisConnection);

– Instantiate and Fill a DataSet with data from one of the DB tables using the OleDbDataAdapter

  DataSet thisDataSet = new DataSet();
  thisAdapter.Fill(thisDataSet, "Books");

– Index through the rows of the Table to get and display their values of their fields in a multiline text box

  foreach (DataRow r in thisDataSet.Tables["Books"].Rows) //each row in "Books" Table

Finding Items in a Database Table

• Extract a DataTable from the DataSet

  DataTable table = thisDataSet.Tables("Books");

• Set up an array of DataRows to hold the rows in which there's a field matching a search criterion

  DataRows[ ] foundRows;

• Use DataTable's <u>Select(…)</u> method with an appropriate <u>filter</u>

  – Selects one or more records in a DataSet

  foundRows = table.Select(s_query);

  – Here s_query is a string giving a selection criterion

    • e.g., "title = 'Megatrends' "

• Index through DataRows array and display results

• See <u>DataSelectRow</u> example

Using ADO.NET in Web Forms

- Just use Visual Studio to create a new ASP.NET Web Form
  - “File” | “New” | “Web Site” | “ASP.NET WebSite”
- As usual the .aspx and .aspx.cs files will be in the default IIS Server directory
  - C:\inetpub\wwwroot\project-name
- Can then run the app from a browser on any machine
  - URL:
    - http://machine-domain-name-or-IP-address/directory/app.aspx
- DatabaseWeb.aspx example has same functionality as DataSelectRow example, but it’s now a Web Form
  - Run it from a browser

Changing the Contents of a Database

- SELECT query strings retrieve data
- Other actions to change data in a database:
  - Updating, Adding, Inserting, Deleting records
- All done in the same way:
  - Fill a DataSet with data retrieved from a DataAdapter
    - As in previous examples
  - Modify (change, add, delete) the data in the DataSet
    - Use a CommandBuilder object associated with the DataAdapter
  - After modifications, persist the DataSet changes back to the database by calling da.Update(….)
  - This won’t work without the CommandBuilder object
- See DataUpdate06 for an updating example

Adding a Row

- Again set up a Connection and a DataAdapter
- Create a CommandBuilder object
- Create and Fill a DataSet
- Create a new row with DataSet Table’s NewRow() method

  DataRow dr = thisDataSet.Tables[“Books”].NewRow();
- Give values to all its fields

  dr[“ISBN”] = “New ISBN”;

  dr[“Title]”=“New Title”;

  dr[“Author”]=“New Author”;
- Add the row with the Table’s Rows.Add() method

  thisDataSet.Tables[“Books”].Rows.Add(dr);
  - Row will be added and Rows.Length property will be incremented
- Update DataAdapter to make change permanent

  thisAdapter.Update(thisDataSet, “Books”);
  - Only the changed fields are updated
  - Again, this will fail if there is no CommandBuilder object

Deleting a Row

- After setting up the Connection, DataAdapter, CommandBuilder, and DataSet:
  - If you know the rows, just use retrieve each one and use its Delete() method, for example:

    DataRows[ ] rows = ds.Tables[“Books”].Select(criterion);

    foreach (DataRow r in rows) r.Delete();
  - Or find the row to be deleted:
    - Determine the primary key before filling the data set:

      thisAdapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;

      thisAdapter.Fill(thisDataSet, “Books”);
    - Use DataSet Table’s Rows Find(p-key) method to find the row:

      DataRow foundRow = thisDataSet.Tables[“Books”].Rows.Find(“222-444”);
    - Returns a DataRow if successful, null if not
  - Delete the row using the Delete() method:

    foundRow.Delete();
  - Finally make change permanent with an Update(…):

    thisAdapter.Update(thisDataSet, “Books”);

Executing SQL Commands

- Behind the scenes a CommandBuilder really uses a DataAdapter's Delete, Insert, Select, and Update commands
- After a DataAdapter populates a DataSet, the DataAdapter we can issue any of the following commands:
    - DeleteCommand, InsertCommand, SelectCommand, UpdateCommand

  These are OleDbCommand objects that specify how the data adapter deletes, inserts, selects, and updates data in the database
  - Set their CommandText property to the SQL to be executed in a query:

    thisAdapter.SelectCommand.CommandText = "SELECT ISBN, Title, Author FROM Books WHERE Title = 'Best Book';
  - The DataAdapters's Fill() member then causes its SelectCommand to execute and fill the DataSet with result of the query
  - Then bind the result to a control such as a textbox
    - textBox1.DataBindings.Add(new Binding ("Text", thisDataSet, "Books.ISBN"));
  - It works the same way for the DataAdapter's UpdateCommand, DeleteCommand, and InsertCommand
  - Example: DataSQLSelect2007

Using Visual Studio Designer to Set Up Access to the Data Base

- The tasks of setting up the DataConnection, the DataAdapter/DataTable, and the DataSet are automated
- In addition VS facilitates simple navigation through database tables with a BindingNavigator object
- Result is a database application with a LOT of functionality without writing any code

Creating a Data Base Project with Visual Studio 2005

- Start a new VS Windows Application
  - Change Name and Text properties
- Add a Data Source
  - Menu: "Data" | "Show Data Sources"
    - Brings up "Data Sources" Window
  - Click on "Add New Data Source"
  - Select "Database" and click "Next"
  - Click on "New Connection" button
  - In "Add Connection" dialog box:
    - Choose Microsoft Access Database File
    - Browse to directory containing the dbase file and Open it
      - Click "Test Connection" and then "OK"
    - Click "Next" and respond "yes" to question about copying files to your project folder
    - Click "Next" and the database objects in the DB will appear

- From Configuration Page called "Choose Your Database Objects":
  - Expand the "Tables" node to view its tables and the fields in the tables
  - Expand the node and check the fields you want to access
    - (e.g., ISBN, Titles, Author)
  - Click on "Finish"

Using the Data Source in the App

- Menu: "Data" | "Show Data Sources"
  - Brings up a "Data Sources" Window
- Add Data-Bound Controls to the form
  - Expand the Books node in Data Sources
  - Drag each field node over to the form
    - Visual Studio will create data-bound text boxes with appropriate labels on the form
      - Other data-bound controls could be chosen
        - » Click down-arrow next to the data field in Data Sources window
    - Also creates a <u>Binding Navigator</u> tool bar underneath the form's title bar
      - Permits adding, deleting, saving, and navigation through database
    - Also in area below the form a <u>DataSet</u>, a <u>BindingSource</u>, and a <u>TableAdapter</u> objects are created
      - TableAdapter is a single-table version of a DataAdapter
- Run the application
  - Lots of new toolbar functionality without writing any code!!

Adding a DataGridView Control to Form

- Displays all the records in the Database table in a spreadsheet-like format
- Very easy to use VS Designer to add the control:
  - Just drag the desired table from the Data Sources window
  - Resize resulting DataGridView control on the form
  - Run the program
    - DataGridView control is already connected to the database
    - If you click on any row in the grid the data in the other controls change to match the selected row
    - No code needs to be added – Visual Studio generated all the needed code

Using ADO.NET with Web Forms

- Because of client/server/client round trips and stateless nature of web pages, all controls must be explicitly bound
- Set DataBindings in form's properties window or in code
- Simple Data Binding
  - Connects one control to one data element
    - Use to display a field value in controls that display one item (e.g., listbox)
  - Do at design time using control's property window, or in code:
    textBox1.DataBindings.Add("Text", dsBooks1, "Books.Author");
- Also, in a web app with a listbox, each time user makes a selection from the list, a postback occurs
  - After postback, the Web page redisplays and the Page_Load event occurs
    - Logic in Page_Load event handler must be modified or the dataset for the list elements will be recreated
    - Use the fact that a page's IsPostBack property is set to false the first time a page displays and true every time after that
- For list controls AutoPostBack property must be set to true for SelectedIndexChanged event handler to execute on the server

Some Code for Web Forms

```
private void Page_Load(object sender, System.EventArgs e)
{
    if (!IsPostBack)
    {
        daTitles.Fill(dsTitles1);
        titlesDropDownList.DataBind();
    }
}
```

Making ADO.NET Projects Portable

- When moving DB projects from one computer to another, connection information must be changed
- Database must be available on new computer
  - Or ConnectionString must specify where it is
- Easiest to put database file in the project’s bin directory and change the DataSource in the ConnectionString in the Form_Load event handler:

```
Private void Form1_LOAD(object sender, System.EventArgs e)
{
    conRnR.ConnectionString =
      “Provider=Microsoft.Jet.OLEDB.4.0;DataSource=rnrBookd.mdb”;
    daTitles.Fill(dsTitles1);
}
```

- DataSource can be another machine/file