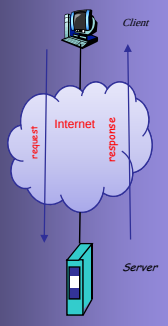


INTERNET PROTOCOLS AND CLIENT-SERVER PROGRAMMING SWE344

Fall Semester 2008-2009 (081)

Module 13: Accessing Databases

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives

- Briefly explain the basics of databases and SQL
- Learn the basics of ADO.NET
- Learn how to access a database from C# program using ADO.NET

Introduction

- A database is an integrated collection of data
- A database management system (DBMS)
  - A software that manages the information in the database
  - It provides mechanisms for storing and organizing data in a way that is consistent with the database's format (schema)
  - It allows storage and access to database without knowledge of internal representation
  - Examples: MS SQL Server, Oracle, MS Access, MySQL, DB2
- Database can be structured in various ways; the most popular form is the *relational database model*
  - A collection of related data organized into tables
  - Use Structured Query Language (SQL) to perform queries (search) and manipulate data
  - Programming languages need an interface to interact with relational databases

Introduction …

- Relational Database Model
  - Logical representation of related data:
    - Relationships can be considered without concern for physical structure of data
  - Composed of tables
    - Rows called records
    - Columns called fields
    - Primary key: field that contains unique data
      - Each record can be identified by at least one distinct value
    - New sets made from queries called result sets

record/row

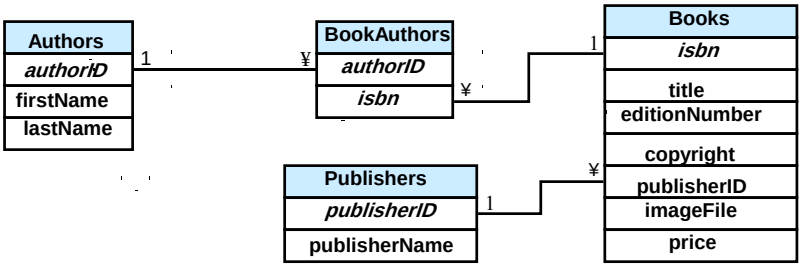
number	name	department	salary	location
23603	Jones	413	1100	New Jersey
24568	Kerwin	413	2000	New Jersey
34589	Larson	642	1800	Los Angeles
35761	Myers	611	1400	Orlando
47132	Neumann	413	9000	New Jersey
78321	Stephens	611	8500	Orlando

primary key

field/column

Introduction …

- Books Database Example



SQL Review

- SQL stands for Structured Query Language and pronounced sequel
- A standard language for accessing relational databases
- There are many SQL statements (classified into DML and DDL); some commonly used are

SQL keyword	Description
SELECT	Selects (retrieves) fields from one or more tables.
FROM	Specifies tables from which to get fields or delete records. Required in every **SELECT** and **DELETE** statement.
WHERE	Specifies criteria that determine the rows to be retrieved.
INNER JOIN	Joins records from multiple tables to produce a single set of records.
GROUP BY	Specifies criteria for grouping records.
ORDER BY	Specifies criteria for ordering records.
INSERT	Inserts data into a specified table.
UPDATE	Updates data in a specified table.
DELETE	Deletes data from a specified table.

SQL Review …

- Joining Data from Tables `Authors`, `BookAuthors`, `Books` and `Publishers`

```
SELECT Books.title, Books.isbn, Authors.firstName,
       Authors.lastName, Books.copyright,
       Publishers.publisherName
FROM
       (Publishers INNER JOIN Books
          ON Publisher.publisherID = Books.publisherID)
       INNER JOIN
       (Authors INNER JOIN BookAuthors
          ON Authors.authorID = BookAuthors.authorID)
       ON Books.isbn = BookAuthors.isbn
ORDER BY Books.title
```

Join Publishers and Books tables if the publisherID matches

Join Authors and BookAuthors if authorID matches

Join two created tables if titlesISBN matches authorsISBN

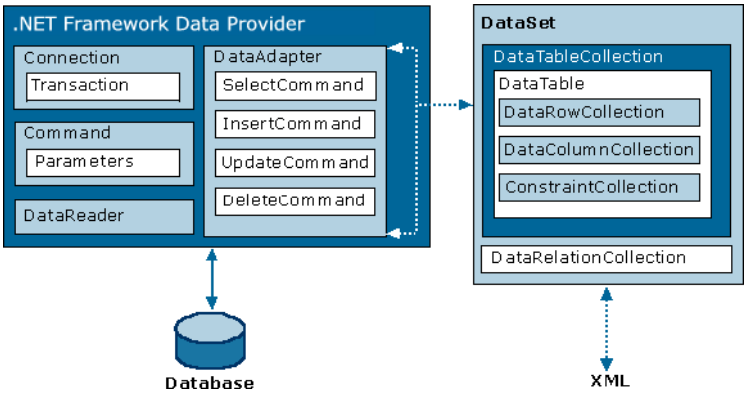
Sort new table by title

Accessing a Database using ADO.NET

- Provides a consistent API for accessing database systems programmatically
- Typically the database will be running on a different computer from the one on which your program runs
- Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.
- When using ADO.NET to access a database
  - Connect to the database
  - Retrieve the information you're interested in
  - Store that information in the memory of the computer on which the C# program is running
  - Manipulate that information locally (e.g. display, add new records, modify records, delete records, …etc)
  - Periodically reconnect to the database to synchronize changes you've made locally with the database

ADO.NET Architecture

- Provide disconnected access to data using multi-tier architecture



Overview of ADO .NET Classes

- ADO.NET defines two sets of classes
  - Generic data classes
    - Objects of the generic data classes are used to store a local copy (in memory) of information retrieved from the database
    - The main generic data class is System.Data.DataSet class that acts as caches to store data from the data source in local memory
  - Managed provider classes
    - The .NET Framework data provider is designed to be lightweight, creating a minimal layer between the data source and your code, increasing performance without sacrificing functionality
    - Provide direct access to the database source
    - Allow you to synchronize your locally stored data with the database source
    - Objects of managed provider classes are used to connect to the database and to read/write information to and from your dataSet object

Namespaces

- System.Data namespace
- System.Data.SqlClient namespace
- System.Data.OracleClient namespace
- System.Data.OleDb namespace
- System.Data.Odbc namespace

Generic Data Classes and Objects

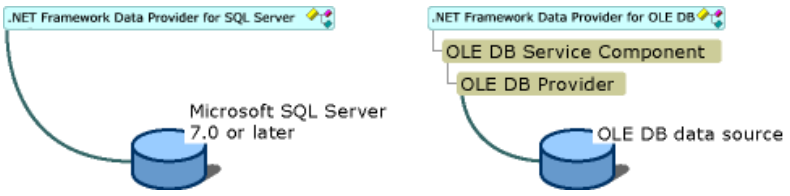
- DataSet class
  - Used to represent a local copy of the information stored in the database source
  - It has the capability to represent database structures such as tables, rows, columns, etc
- DataTable class
  - Used to represent a table
  - You can store multiple DataTable objects in a DataSet
- DataRow class
  - Used to represent a row
  - You can store multiple DataRow objects in a DataTable
- DataColumn class
  - Used to represent a column
  - You can store multiple DataColumn objects in a DataRow
- DataRelation class
  - Used to represent a relationship between two DataTable objects
  - Can be used to model parent-child relationship between two database tables
  - You can store multiple DataRelation objects in a DataSet
- Constraint class
  - Used to represent a database constraint
  - You can store multiple Constraint objects in a DataTable
- DataView class
  - Used to view only specific rows in a DataTable object using a filter

Managed Provider Classes and Objects

- SqlConnection, OleDbConnection, OdbcConnection, OracleConnection
  - SqlConnection used to connect to a SQL Server database,
  - OleDbConnection used to connect to any database that supports OLEDB (Object Linking and Embedding for Database) such as Access or Oracle
  - OdbcConnection used to connect to any database that supports ODBC (Open Database Connectivity); supported by all major databases but is typically slower than the other two options when working with .NET
  - OracleConnection used to connect to Oracle database server
- SqlCommand, OracleCommand, OleDbCommand, OdbcCommand
  - Used to represent a SQL statement that you then execute using one of the appropriate connection classes
- SqlDataReader, OracleDataReader, OleDbDataReader, OdbcDataReader
  - Used to read rows retrieved from a SQL Server, OLEDB-compliant, ODBC-compliant databases (faster than using DataSet)
- SqlDataAdapter, OleDbDataAdapter, OdbcDataAdapter
  - Used to move rows between a DataSet object and a SQL Server, OLEDB-compliant, ODBC-compliant databases (to synchronize locally stored information with the database)
- SqlTransaction, OleDbTransaction, OdbcTransaction
  - Used to represent a database transaction

Data Provider for SQL Server vs. OLE DB

- The .NET Framework Data Provider for SQL Server uses its own protocol to communicate with SQL Server
- It is lightweight and performs well because it is optimized to access a SQL Server directly without adding an OLE DB or Open Database Connectivity (ODBC) layer
- The .NET Framework Data Provider for ODBC has a similar architecture to the .NET Framework Data Provider for OLE DB



Example 1: Connecting to SQL Server

- This example demonstrates how to connect to a database, query the database and display the results of the query
  - Connect to the SQL Server Northwind database and perform a SQL SELECT statement to retrieve the CustomerID, CompanyName, ContactName and Address columns for the customer with CustomerID < 'BSBEV' from the Customers table

Example 1…

- To access Microsoft SQL Server, do the following steps
  1. Formulate a string containing the details of the database connection

```
String connString = "server= localhost; database=Northwind; uid=sa;
    pwd=sa";
```

  2. Create a SqlConnection object to connect to the database (passing the connection string to the constructor)

```
SqlConnection mySqlConn = new SqlConnection(connString);
```

  3. Formulate a string containing a SELECT statement to retrieve required information

```
String selectString = "SELECT CustomerID, CompanyName, ContactName,
    Address "+ "FROM Customers "+ "WHERE CustomerID < 'BSBEV' ";
```

Example 1 …

4. Create SqlCommand object and set the CommandText property to the SELECT string

```
SqlCommand mySqlComm = mySqlConn.CreateCommand();
mySqlComm.CommandText = selectString;
```

5. Create SqlDataAdapter object and set its SelectCommand property to the SqlCommand object

```
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter()
mySqlDataAdapter.SelectCommand = mySqlComm;
```

6. Create a DataSet object to store the results

```
DataSet myDataSet = new DataSet();
```

7. Open the database connection using the Open() method of the SqlConnection object

```
mySqlConn.Open();
```


Example 1 …

8. Call Fill() method of the SqlDataAdapter to retrieve the rows from the table and store them locally in a DataTable of the DataSet

```
String dataTableName = "Customers";
mySqlDataAdapter.Fill(myDataSet, dataTableName );
```

9. Get the DataTable object from the DataSet and display the columns for each row using DataRow object

```
DataTable myDataTable =
            myDataSet.Tables[dataTableName];
foreach(DataRow myDataRow in myDataTable.Rows)
{
   // manipulate the myDataRow object
   Console.WriteLine("Customer ID = " + myDataRow["CustomerID"]);
   Console.WriteLine("Contact Name = " + myDataRow["ContactName"]);
}
```

10. Close the database connection using Close() method of SqlConnection object

```
mySqlConn.Close();
```

Connecting to MS Access Database

- To connect to MS Access database, we can use either OleDb or Odbc data providers.
- The following are the corresponding connection strings: use a connection string with the following syntax
- OleDb:

  String conString = " provider= **Microsoft.Jet.OLEDB.4.0**; data source = **C:\\Northwind.mdb** ";

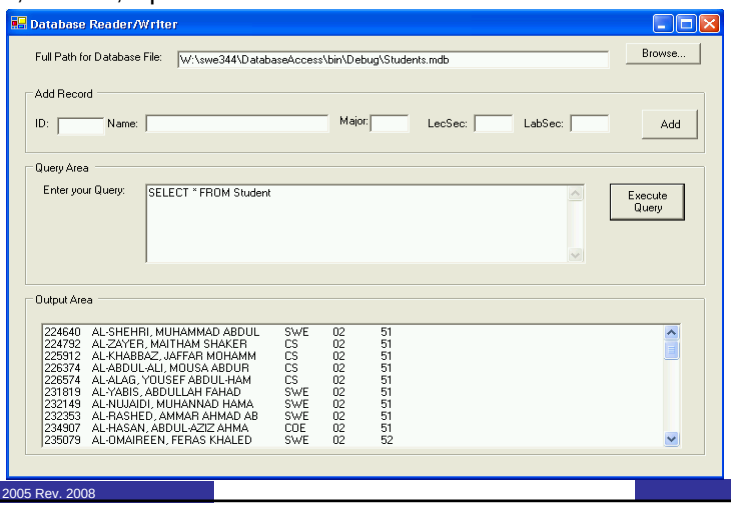
- Odbc:

  String conString = "Driver={Microsoft Access Driver (*.mdb)}; DBQ= C:\\Northwind.mdb";

- Note: Connection String depends on two things; the data base system (SQL, Oracle, etc.) and the data driver (Odbc, Oledb, etc.). This website shows various combinations: http://www.connectionstrings.com/

Example 2

- If you need to read records from a database only one time, it is more efficient to use DataReader instead of DataSet
- The following example allows a database of students to be manipulated – Add, Search, Update or Delete. It uses DataReader.



Example 2 …

```
1.  using System.ComponentModel;
2.  using System.Drawing;
3.  using System.Windows.Forms;
4.  using System.Data;
5.  using System.Data.Odbc;

6.  namespace DatabaseAccess
7.  {
8.      public partial class Form1 : Form
9.      {
10.         public Form1()
11.         {
12.             InitializeComponent();
13.         }
14.         private void btnBrowse_Click(object sender, EventArgs e)
15.         {
16.             OpenFileDialog dlg = new OpenFileDialog();
17.             if (dlg.ShowDialog() == DialogResult.OK)
18.             {
19.                 txtDatabase.Text = dlg.FileName;
20.             }
21.         }
```


Example 2 …

```
22. private void btnAdd_Click(object sender, EventArgs e) {
23.   string conString = "Driver={Microsoft Access Driver (*.mdb)};"+
24.                      "DBQ="+ txtDatabase.Text;
25.     //check http://www.connectionstrings.com/ for other databases
26.     OdbcConnection dbConn = new OdbcConnection(conString);
27.     dbConn.Open();
28.     try {
29.         string sqlString = "INSERT INTO Student (ID, Name, Major,
30.           LecSec, LabSec)" + "VALUES ('" + txtID.Text + "', '" +
31.           txtName.Text.ToUpper() + "', '" + txtMajor.Text.ToUpper() +
32.           "','" + txtLec.Text + "','" + txtLab.Text + "')";
33.         OdbcCommand sqlCommand = new OdbcCommand(sqlString, dbConn);
34.         sqlCommand.ExecuteNonQuery();
35.         dbConn.Close();
36.         txtResult.Text += "One record added successfully" + "\r\n";
37.     }
38.     catch (Exception ex)
39.     {
40.         txtResult.Text += "Update Error " + ex.Message + "\r\n";
41.     }
42. }
```

Example 2 …

```
43.  private void btnExecute_Click(object sender, EventArgs e) {
44.      string conString = "Driver={Microsoft Access Driver (*.mdb)};"+
45.                         "DBQ="+ txtDatabase.Text;
46.      OdbcConnection dbConn = new OdbcConnection(conString);
47.      dbConn.Open();
48.      try {
49.          string sqlString = txtQuery.Text;
50.          OdbcCommand sqlCommand = new OdbcCommand(sqlString, dbConn);
51.          if (sqlString.ToLower().StartsWith("select"))
52.          {
53.              OdbcDataReader reader = sqlCommand.ExecuteReader();
54.              txtResult.Text = "";
55.              while (reader.Read())
56.              {
57.                  for (int i = 0; i < reader.FieldCount - 1; i++)
58.                      txtResult.Text += reader.GetString(i) + "\t";
59.                  txtResult.Text +=
60.                      reader.GetString(reader.FieldCount - 1) + "\r\n";
61.              }
62.              reader.Close();
63.          }
```


Example 1: Broadcast Echo System …

```
64.     else {
65.              int count = sqlCommand.ExecuteNonQuery();
66.              txtResult.Text += GetMessage(sqlString, count);
67.          }
68.      }
69.      catch (Exception ex) {
70.          txtResult.Text += "Connection did not succeeed: " +
71.                            ex.Message + ": " + conString + "\r\n";
72.      }
73.  }
74.   string GetMessage(string command, int count) {
75.              string msg = count + " record(s) ";
76.              command = command.ToLower();
77.              if (command.StartsWith("delete"))
78.                  msg += "deleted";
79.              else if (command.StartsWith("insert"))
80.                  msg += "inserted";
81.              else
82.                  msg += "updated";
83.              msg += " successfully";
84.              return msg;
85.   } }}
```

Resources

- MSDN Library
  - http://msdn.microsoft.com/en-us/default.aspx
- Books
  - Richard Blum, C# Network Programming. Sybex 2002.
  - C# How to Program, by Deitel
- Lecture notes of previous offerings of SWE344 and ICS343
- Some other web sites and books; check the course website at
  - http://faculty.kfupm.edu.sa/ics/alfy/files/teaching/swe344/index.htm