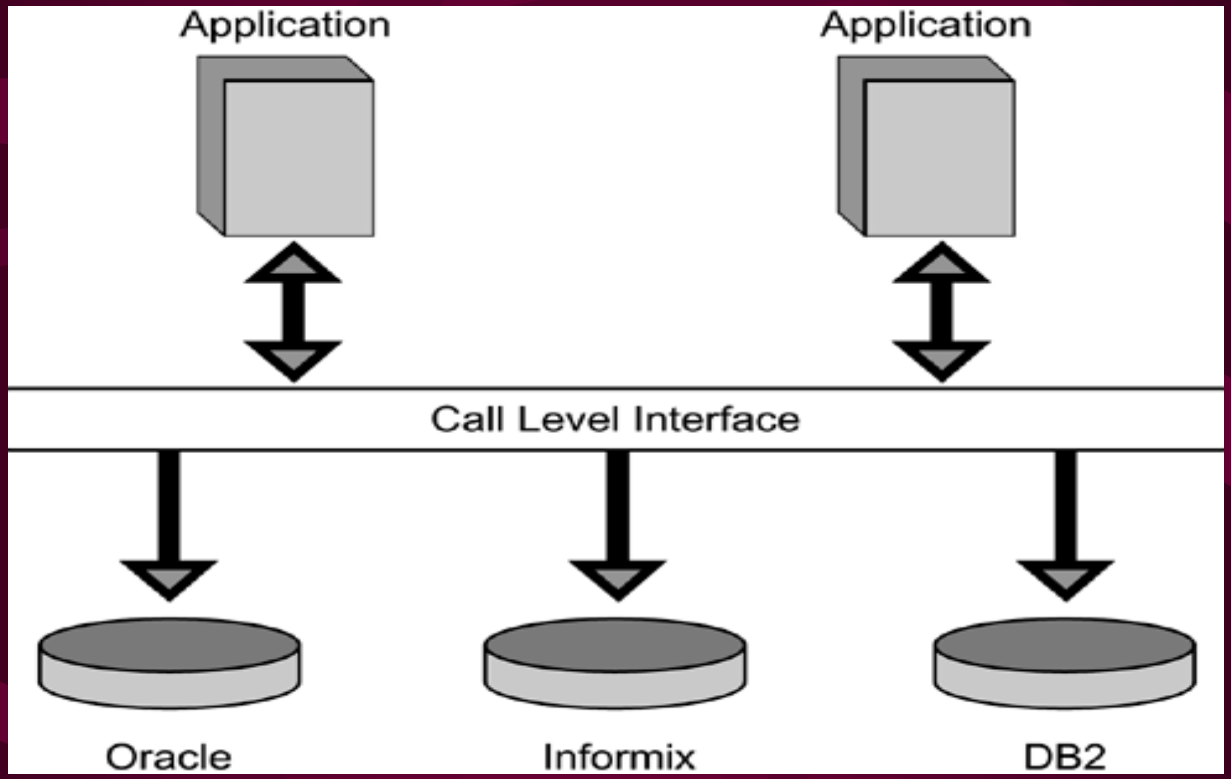
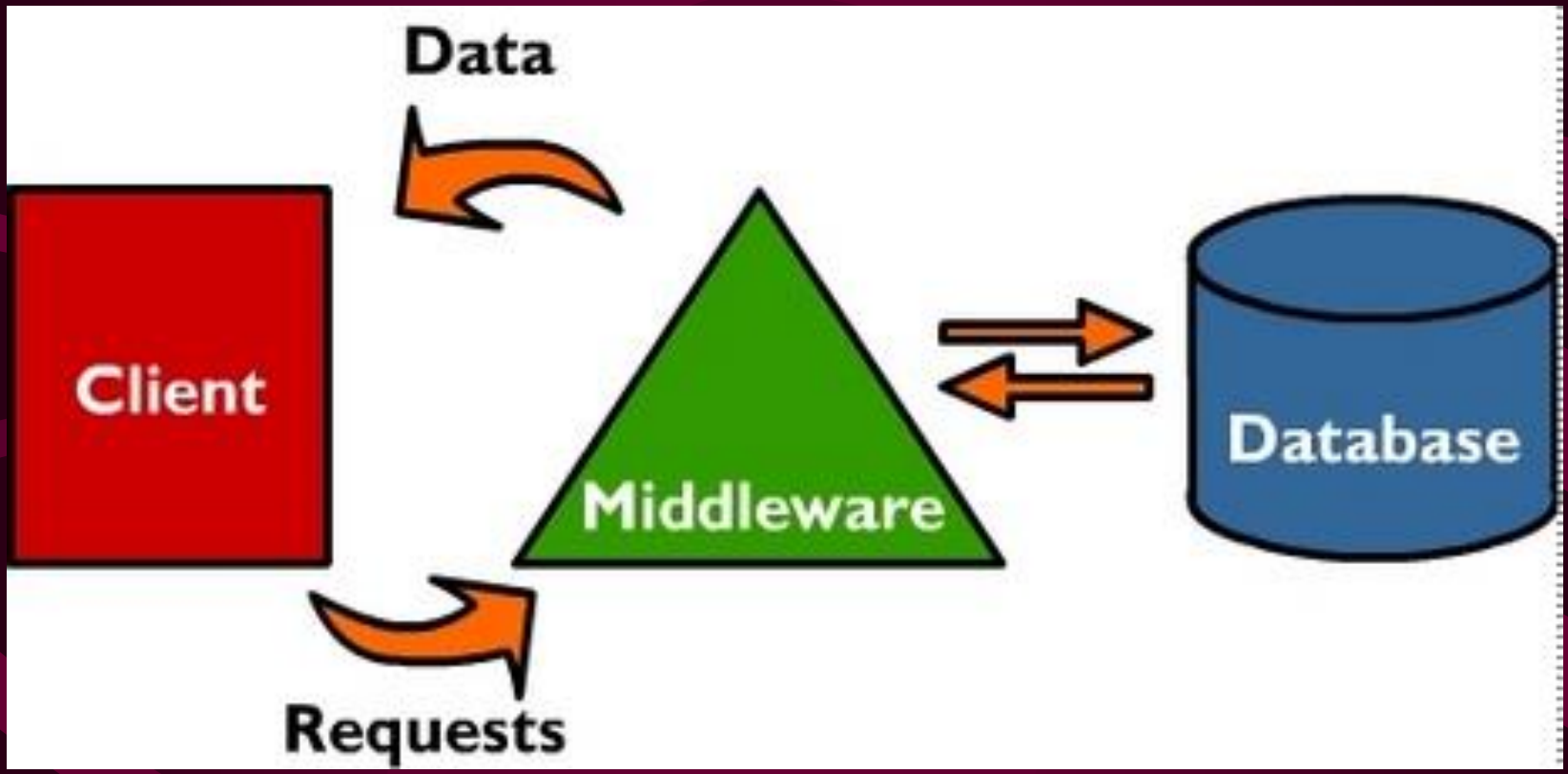
Data Base Connectivity & Web Applications



Database Connectivity

- Mechanisms through which applications connect and communicate with data repositories
  - Database middleware: provides an interface between the application program and the database
  - Data repository: data management application used to store data generated by an application program
  - Universal Data Access (UDA): collection of technologies used to access any type of data source and manage the data through a common interface
    - ODBC, OLE-DB, and ADO.NET form the backbone of MicroSoft's UDA architecture

Middleware



Native SQL Connectivity

- Connection interface provided by database and/or language vendors, which is unique to each vendor
  - Interfaces are optimized for particular vendor's language and/or DBMS
- Oracle provides SQL*Net so client or server applications can directly access an Oracle database

Native SQL Connectivity (con't)

- PHP provides a native interface to MySQL database so one can make direct SQL calls from within a PHP web aplication

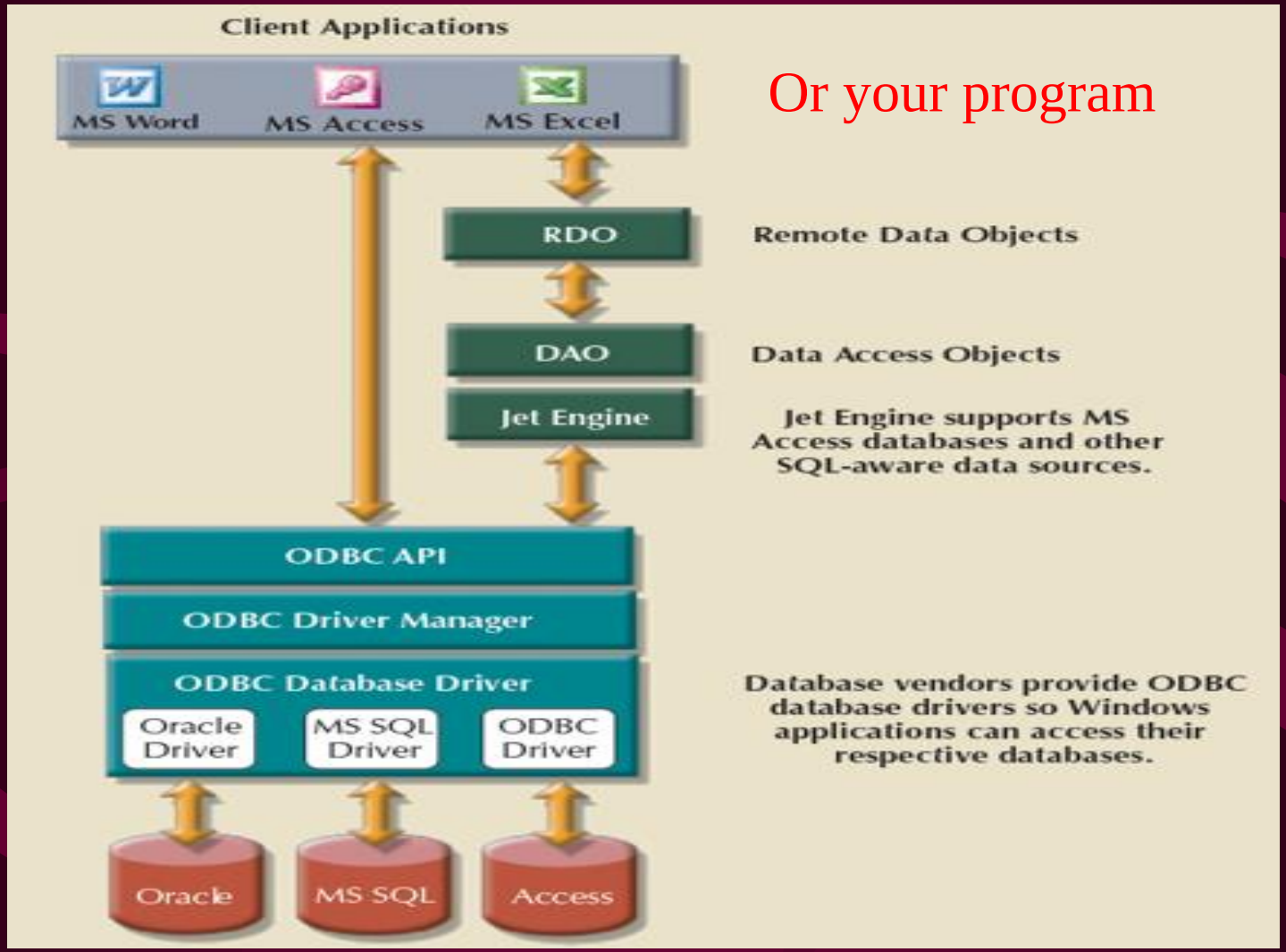
ODBC, DAO, and RDO

- Open Database Connectivity (ODBC): Microsoft's implementation of a superset of SQL Access Group Call Level Interface (CLI) standard for database access
  - Widely supported database connectivity interface
  - Allows Windows application to access relational data sources by using SQL via standard application programming interface (API)
- Data Access Objects (DAO): object-oriented API used to access desktop databases such as MS Access and FileMaker Pro
  - Provides an optimized interface that expose functionality of Jet data engine to programmers
- Remote Data Objects (RDO): higher-level object-oriented application interface used to access remote database servers
    - Optimized to deal with server-based databases
- Dynamic-link libraries (DLLs): implements ODBC, DAO, and RDO as shared code that is dynamically linked to the Windows operating environment

ODBC, DAO, and RDO (con't)

- Components of ODBC architecture
  - High-level ODBC API through which application programs access ODBC functionality
  - Driver manager that is in charge of managing all database connections
  - ODBC driver that communicates directly to DBMS

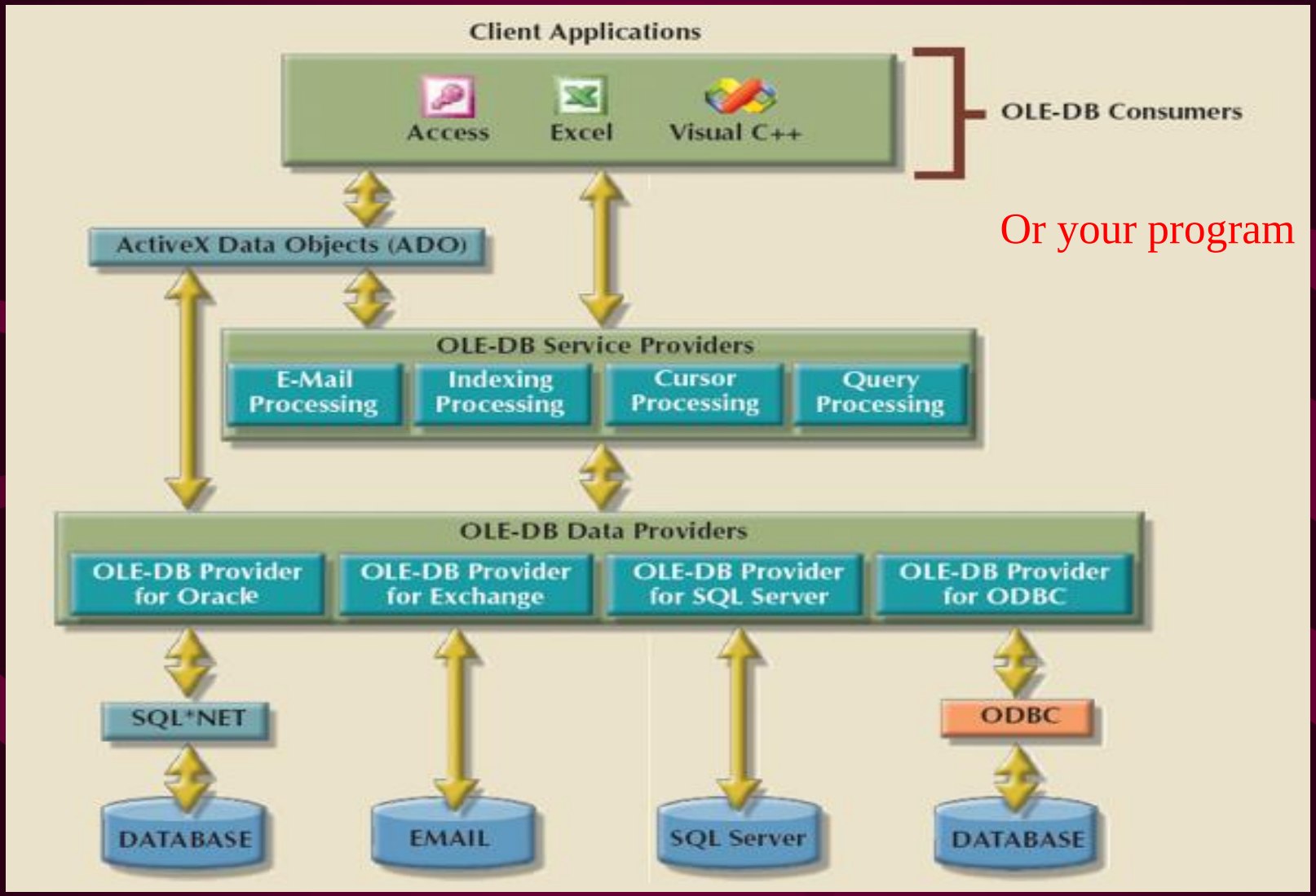
ODBC, DAO, and RDO (con't)



OLE-DB

- Object Linking and Embedding for Database (OLE-DB)
  - Database middleware that adds object-oriented functionality for access to relational and nonrelational data
  - Series of COM objects provides low-level database connectivity for applications
  - Types of objects based on functionality
    - Consumers: applications or processes
    - Providers: data or service
  - Does not provide support for scripting languages
- ActiveX Data Objects (ADO) provide:
  - High-level application-oriented interface to interact with OLE-DB, DAO, and RDO
  - Unified interface to access data from any programming language that uses the underlying OLE-DB objects

OLE-DB (con't)


Client Applications
Access
Excel
Visual C++
OLE-DB Consumers
Or your program
ActiveX Data Objects (ADO)
OLE-DB Service Providers
E-Mail Processing
Indexing Processing
Cursor Processing
Query Processing
OLE-DB Data Providers
OLE-DB Provider for Oracle
OLE-DB Provider for Exchange
OLE-DB Provider for SQL Server
OLE-DB Provider for ODBC
SQL*NET
ODBC
DATABASE
EMAIL
SQL Server
DATABASE

ADO.NET

- Data access component of Microsoft's .NET application development framework
- Microsoft's .NET framework
  - Component-based platform for developing distributed, heterogeneous, interoperable applications
  - Manipulates any type of data using any combination of network, operating system, and programming language
  - Extends and enhances functionality critical for the development of distributed applications
- DataSet: disconnected memory-resident representation of the database
  - Contains tables, columns, rows, relationships and constraints
  - Internally stored in XML format
  - Data in DataSet is made persistent as XML documents

ADO.NET (con't)

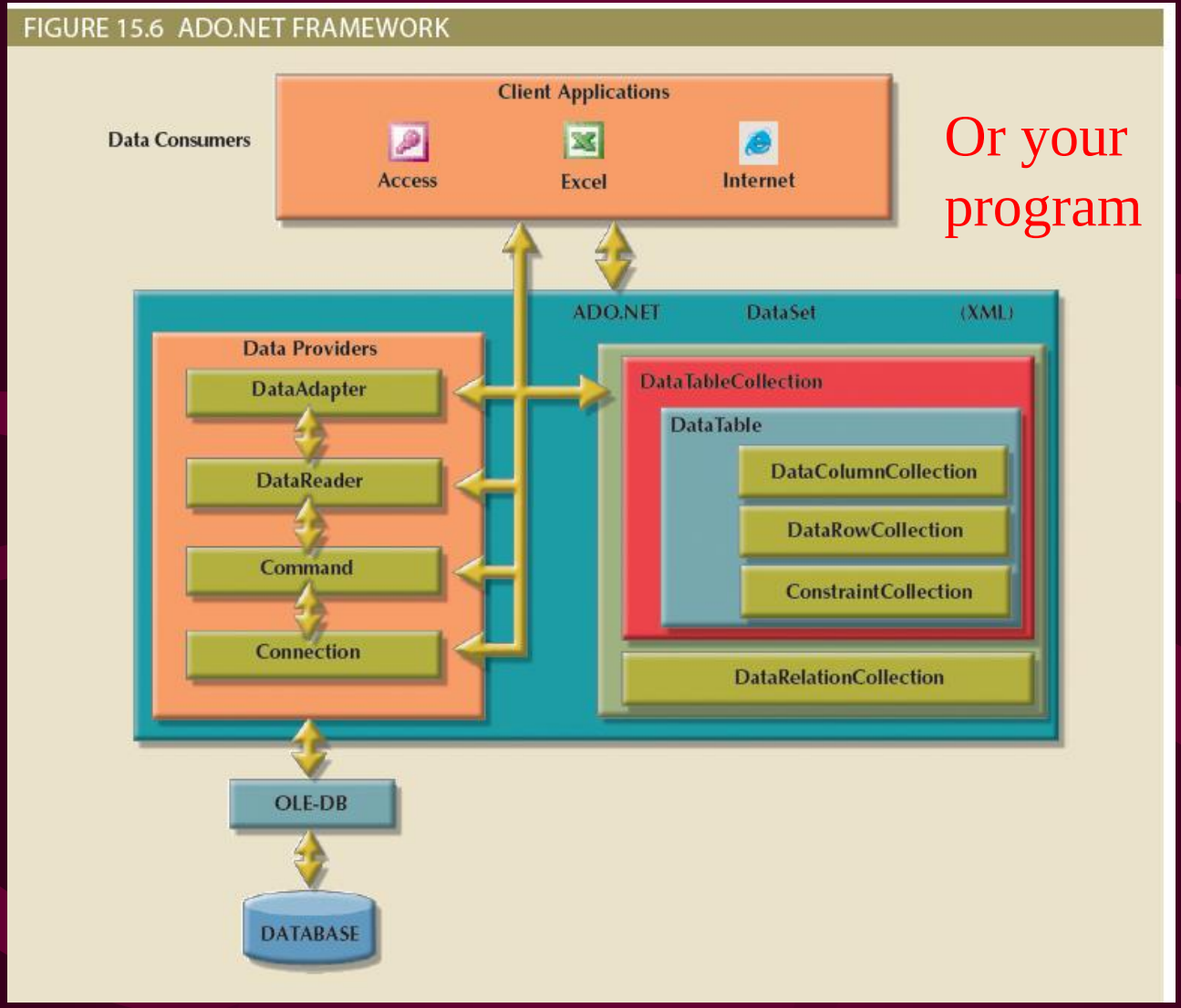
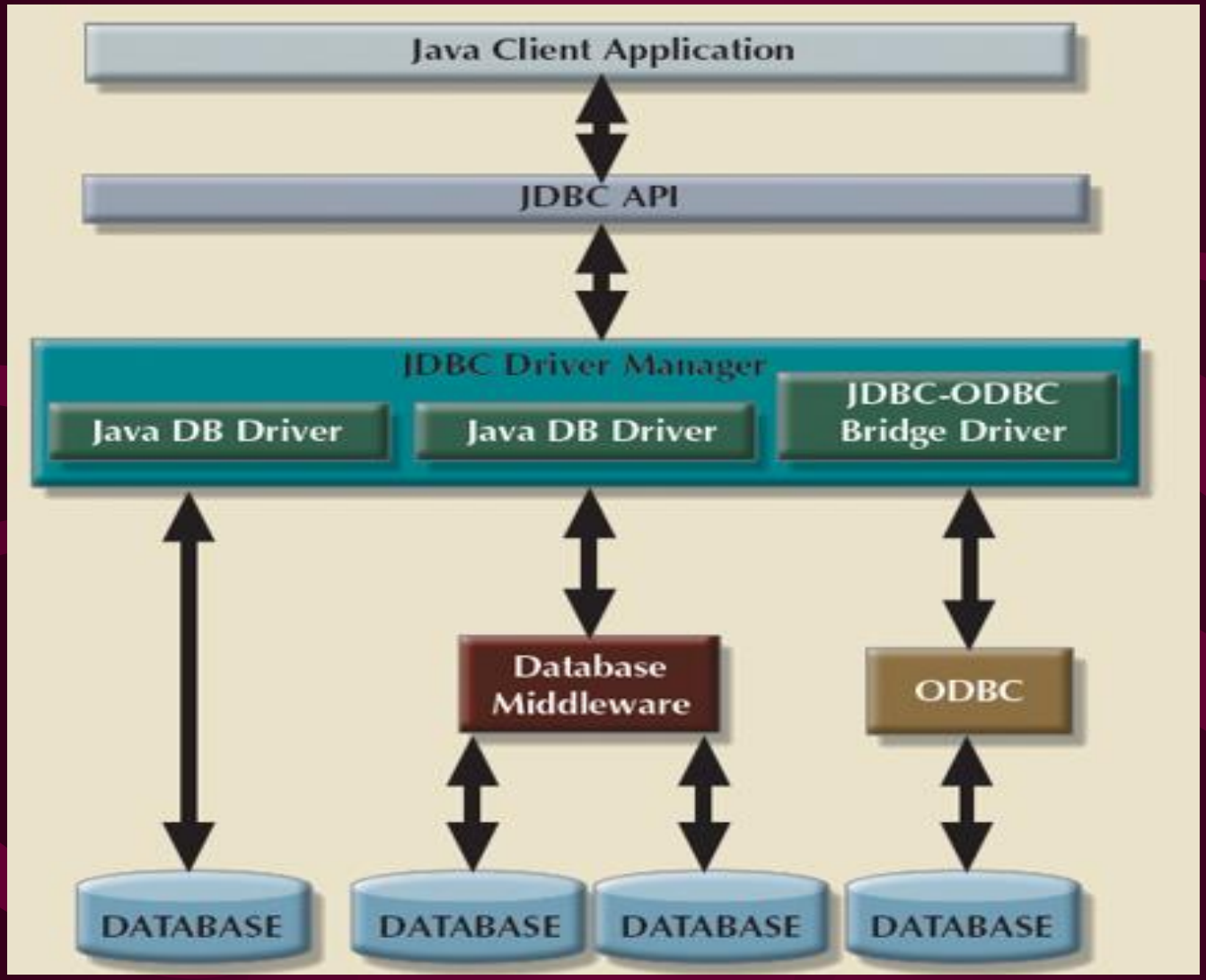


FIGURE 15.6 ADO.NET FRAMEWORK
Data Consumers
Client Applications
Access
Excel
Internet
Or your program
ADO.NET
DataSet
(XML)
Data Providers
DataAdapter
DataReader
Command
Connection
DataTableCollection
DataTable
DataColumnCollection
DataRowCollection
ConstraintCollection
DataRelationCollection
OLE-DB
DATABASE

Java Database Connectivity (JDBC)

- Application programming interface that allows a Java program to interact with a wide range of data sources
- Advantages of JDBC
  - Company can leverage existing technology and personnel training
  - Direct access to database server or access via database middleware
  - Programmers can use their SQL skills to manipulate the data in the company's databases
  - Provides a way to connect to databases through an ODBC driver

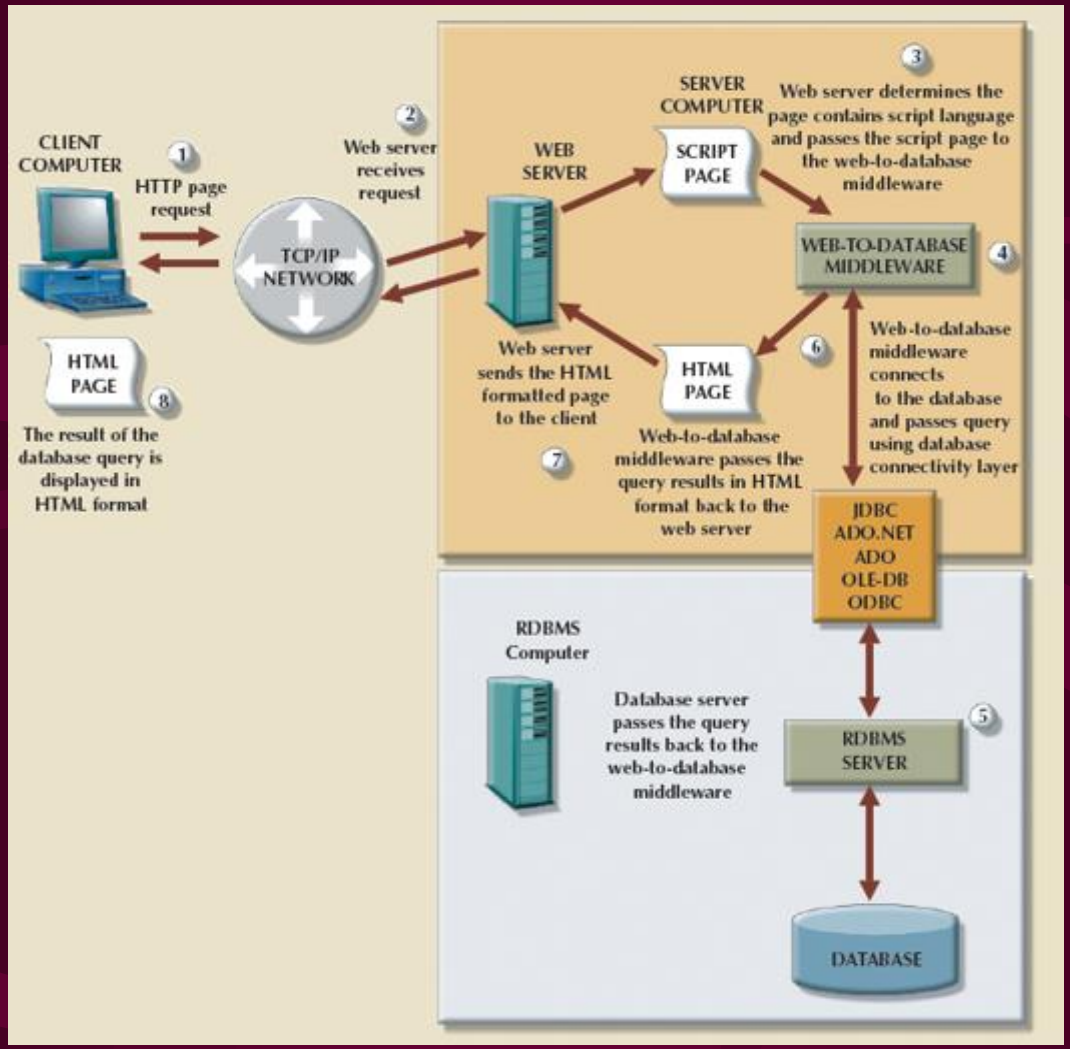
Java Database Connectivity (con't)



Web-to-Database Middleware: Server-Side Extensions

- Web server is the main hub through which Internet services are accessed
  - Server-side extension: program that interacts directly with the web server
    - Provides services to the web server in a way that is totally transparent to the client browser
    - Known as web-to-database middleware

Web-to-Database Middleware: Server-Side Extensions (con't)



Web Server Interfaces

- Currently, there are two well-defined web server interfaces
  - Common Gateway Interface (CGI): uses script files that perform specific functions based on the client's parameters that are passed to the web server
  - Application programming interface (API): implemented as shared code or as dynamic-link libraries; treated as part of the web server program that is dynamically invoked when needed

Web Application Servers

- Middleware application that expands the functionality of web servers by linking them to a wide range of services (also called web servers or information servers)
  - Connects to and query database from web page
  - Presents database data in a webpage using various formats
  - Creates dynamic web search pages
  - Creates webpages to insert, update, and delete data
  - Enforces referential integrity
  - Uses simple and nested queries and program logic to represent business rules

Web Application Servers (con't)

- Web application server features
  - Integrated development environment
  - Security and user authentication
  - Computational languages
  - Automation generation of HTML pages
  - Performance and fault -tolerant features
  - Database access with transaction management capabilities
  - Access to multiple services

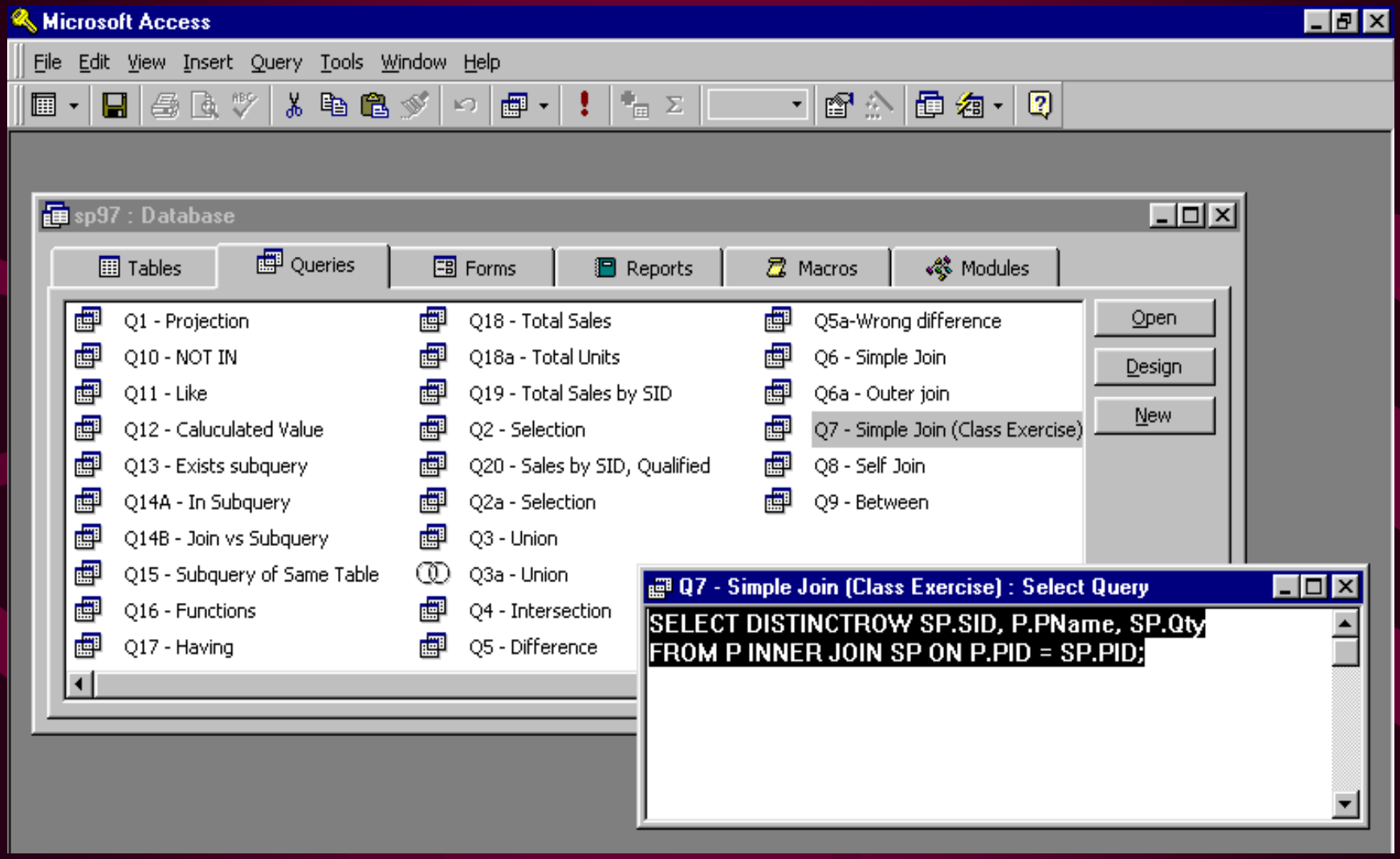
Web Database Options

- Static Publishing - IDC
- Server Side Interfaces
  - CGI
    - Perl & C/C++
  - Information Server API (ISAPI & NSAPI)
    - Active Server Pages (.Net technologies)
    - Java Servlets
    - Java Server Pages (uses Java Servlets)
    - Cold Fusion (uses Java Servlets)
    - PHP
- Direct Socket Communication
  - C/C++
  - Java Applets and Java Servelets

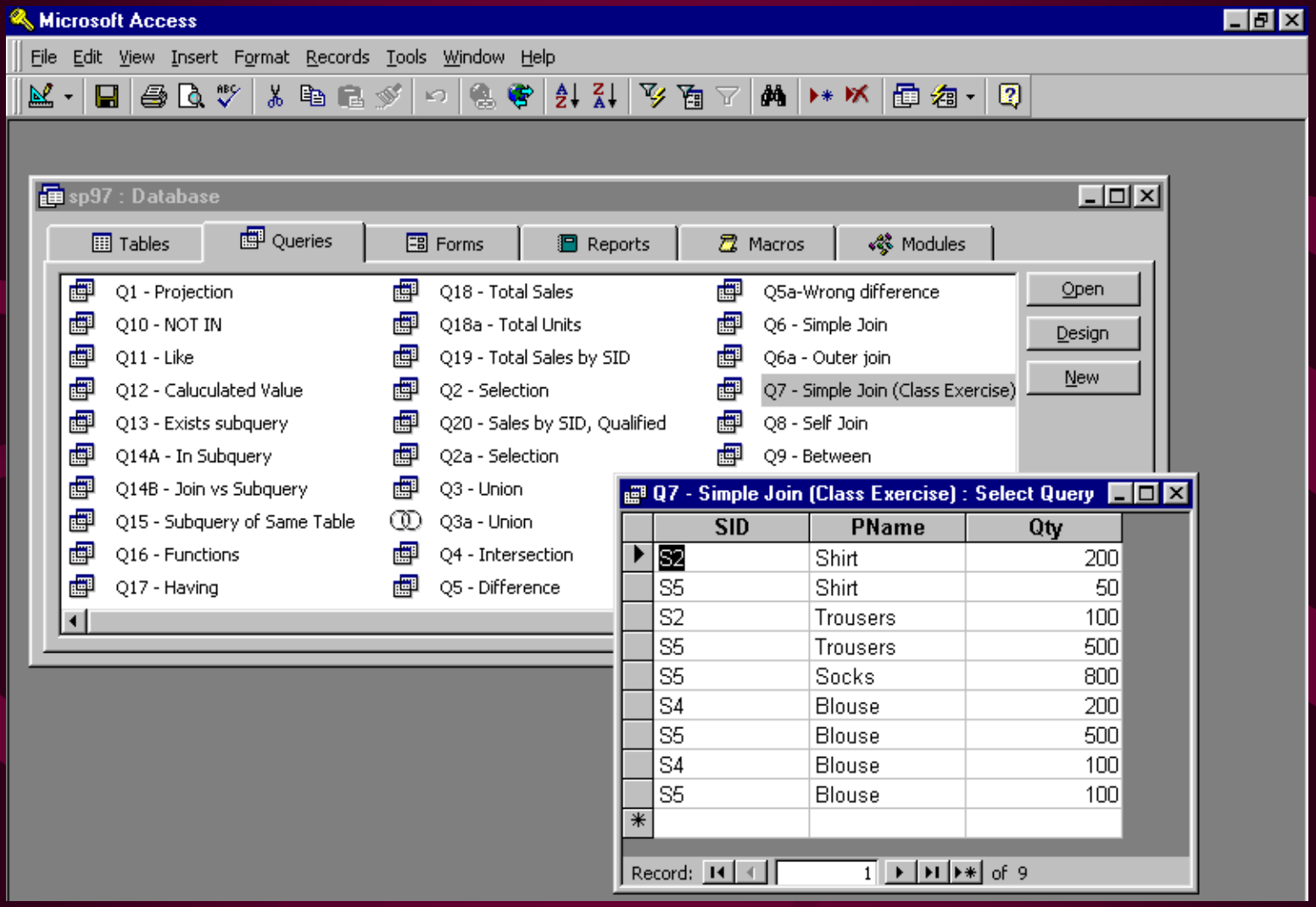
Static vs Active DB Publishing

- Static Publishing - a browser user request a pre-generated HTML page. That page may have been previously generated from a database (ie Access) using IDC or similar
- Active Publishing - a browser user interactively submits information (typically via a “form”) that determines an SQL type query, and the query results are returned to the user typically in an HTML page (or open Java Applet window)

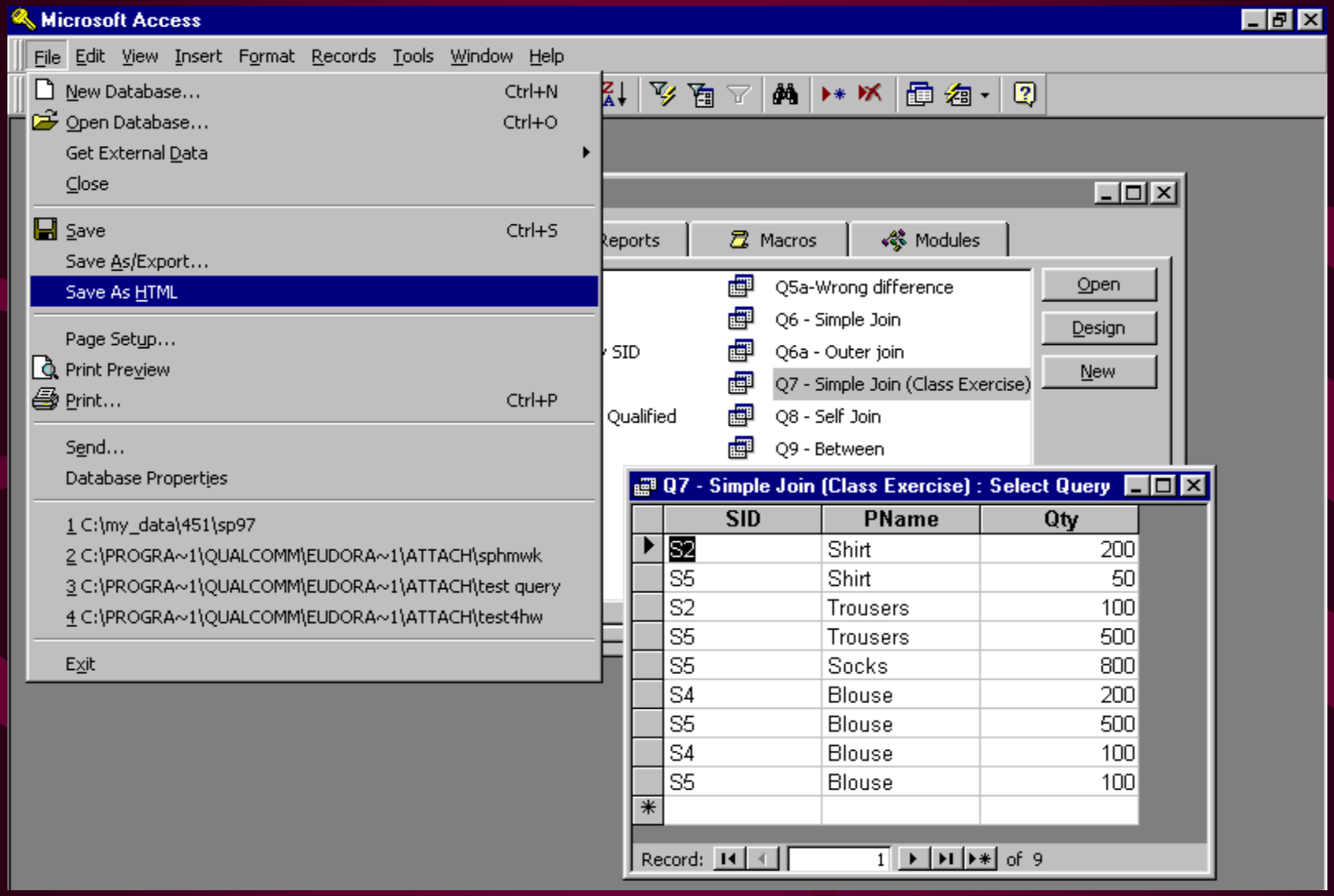
Access Query



Access Query Results



Saving Static Pages



Static HTML Generated

```
<HTML><HEAD><META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Q7 - Simple Join (Class Exercise)</TITLE></HEAD><BODY>
<TABLE BORDER=1 BGCOLOR=#ffffff CELLSPACING=0><FONT FACE="Arial" COLOR=#00000><CAPTION><B>Q7</B></CAPTION><THEAD><TR>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>SID</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>PName</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Qty</FONT></TH></THEAD>
<TBODY><TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S2</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Shirt</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>200</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S5</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Shirt</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>50</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S2</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Trousers</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>100</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S5</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Trousers</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>500</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S5</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Socks</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>800</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S4</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Blouse</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>200</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S5</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Blouse</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>500</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S4</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Blouse</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>100</FONT></TD></TR>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>S5</FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Blouse</FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000>100</FONT></TD></TR>
</TBODY><TFOOT></TFOOT></TABLE></BODY></HTML>
```

Servers Involved

- Information (or Web) Server - server running Microsoft Information Server (IIS) or Apache (open source)
- Uses HTTP (HyperText Transfer Protocol) to communicate with clients; HTTP is a TCP/IP application service (like FTP, SMTP, etc.).
- Database Server - server running Oracle, Sybase, SQLServer, MySQL, or simply containing Access (.mdb or .accdb) files
- Both “servers” may be on one machine

Typical Microsoft Web Server

- Operating System – Windows Server
- Web Server - IIS
- Server Processing
  - ISAPI (direct calls with C++)
  - CGI
  - Active Server Pages
  - Java Server Pages
  - Java Servlets
  - Cold Fusion
- Database Access (ODBC, ADO, OLE/DB, JDBC)
  - SQLServer
  - Access
  - MySQL

Typical Unix Web Server

- Operating System – Unix/Linux
- Web Server - Apache, Netscape, CERN
- Server Processing
  - CGI (Perl, C/C++, Java)
  - NSAPI (direct calls with C++ or Java)
  - Java Server Pages
  - Java Servlets
  - Cold Fusion
- Database (ODBC, JDBC, native)
  - Oracle
  - Informix
  - MySQL

Open Database Connectivity [ODBC]

- Developed in 1990's by X/Open and SQL Access Group committees to provide an open DBMS independent API for relational databases (can also be used for non-relational databases)
- First fully implemented by Microsoft; now implemented by most all DBMS vendors and third party suppliers

ODBC Data Sources

- An ODBC Data Source is the combination of the database, it's associated DBMS, operating system, and network middleware (if any).
- A data source can be an Oracle database, Sybase database, SQLServer database, Access database, MySQL database, Excel spreadsheet, Btrieve file server, etc.

ODBC Components

- Application - makes calls using ODBC API functions to create a connection to a data source, issues dynamic SQL statements, get back result sets, start-commit-rollback transactions, and receive status/error conditions
- Driver Manager - loads appropriate driver and handles communication between the application and the specific data source driver
- DBMS Driver - turns driver manager commands into specific SQL for target database

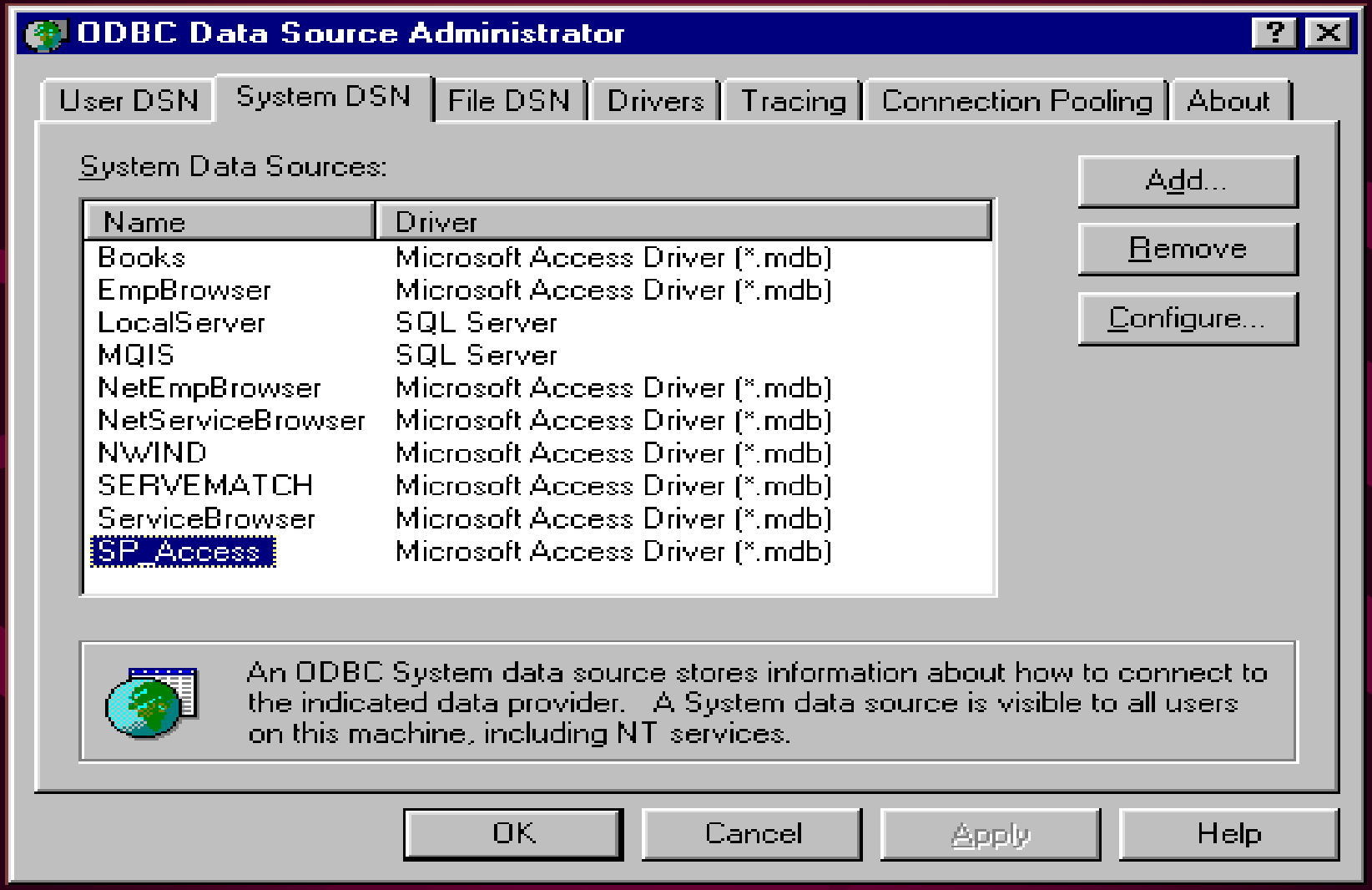
Driver Types

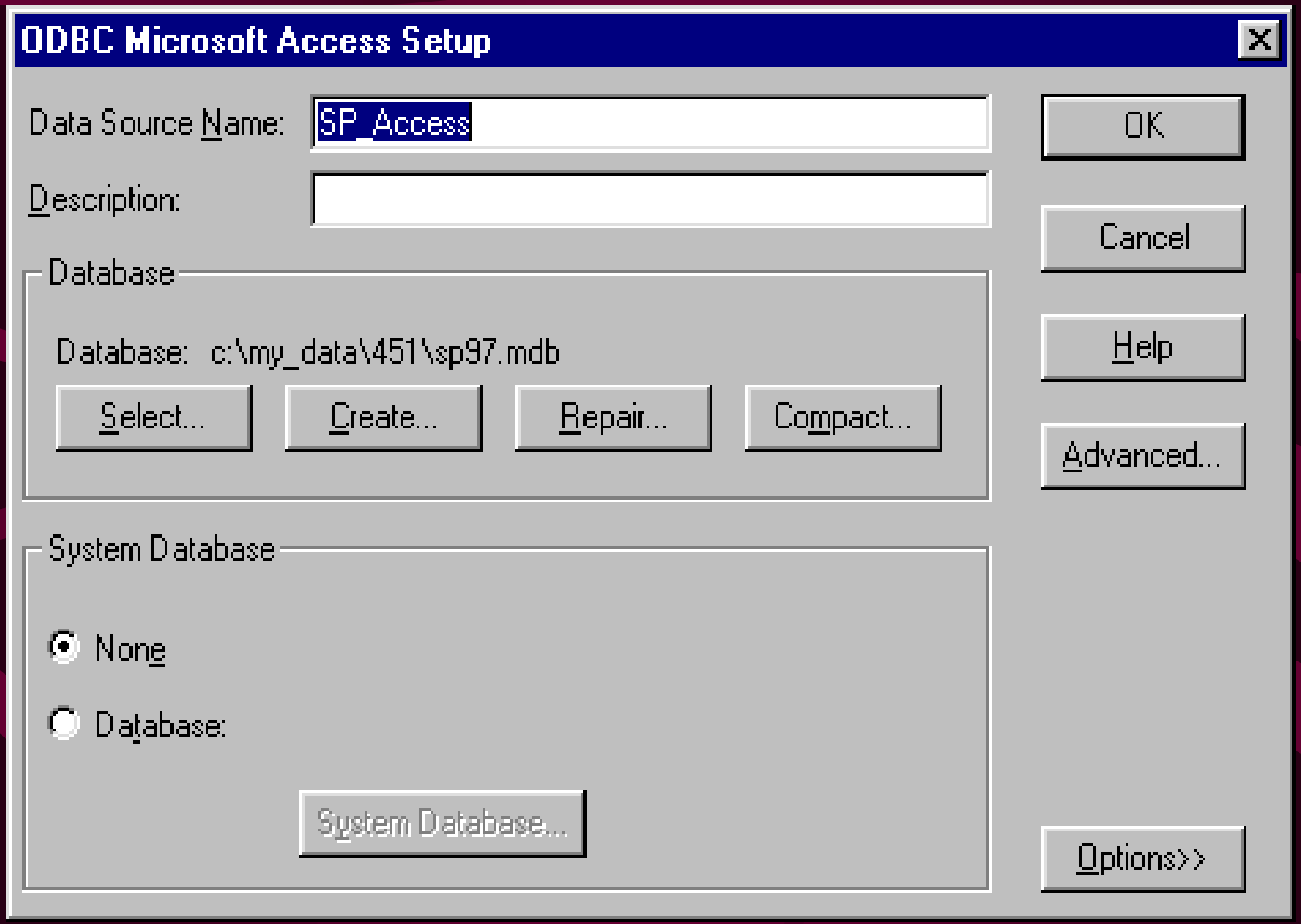
- Single Tier - Database does not handle SQL directly, so the driver on the host translates the SQL into native calls
- Multiple Tier - Driver does not process SQL (although it may format and modify it), but passes SQL directly onto database - multiple servers may be involved

Data Source Types

- File Data Source - A file that is shared among users; users all have same driver and privileges
- System Data Source - A source that is local to a single computer
- User Data Source - only available to user who created it
- Generally you set up a System Data Source on the server on which your CGI/ISAPI programs execute or on the client to a specific database (may be local database, on a “mapped” network drive, or server database)

ODBC Data Source




ODBC Microsoft Access Setup
Data Source Name:
SP_Access
Description:
Database
Database: c:\my_data\451\sp97.mdb
Select...
Create...
Repair...
Compact...
System Database
None
Database:
System Database...
OK
Cancel
Help
Advanced...
Options>>

Internet Database Connector (IDC)

- This is a Microsoft product that can be used to make static snapshot's of database information for publishing on the web
- A server program (Httpodbc.dll) is run which uses an ODBC data source to read a database file (ie Access) and create HTML pages
- IDC needs two files to be created: *.idc to indicate the data source and table, and *.htx to specify the HTML formatting
- Access can generate these files

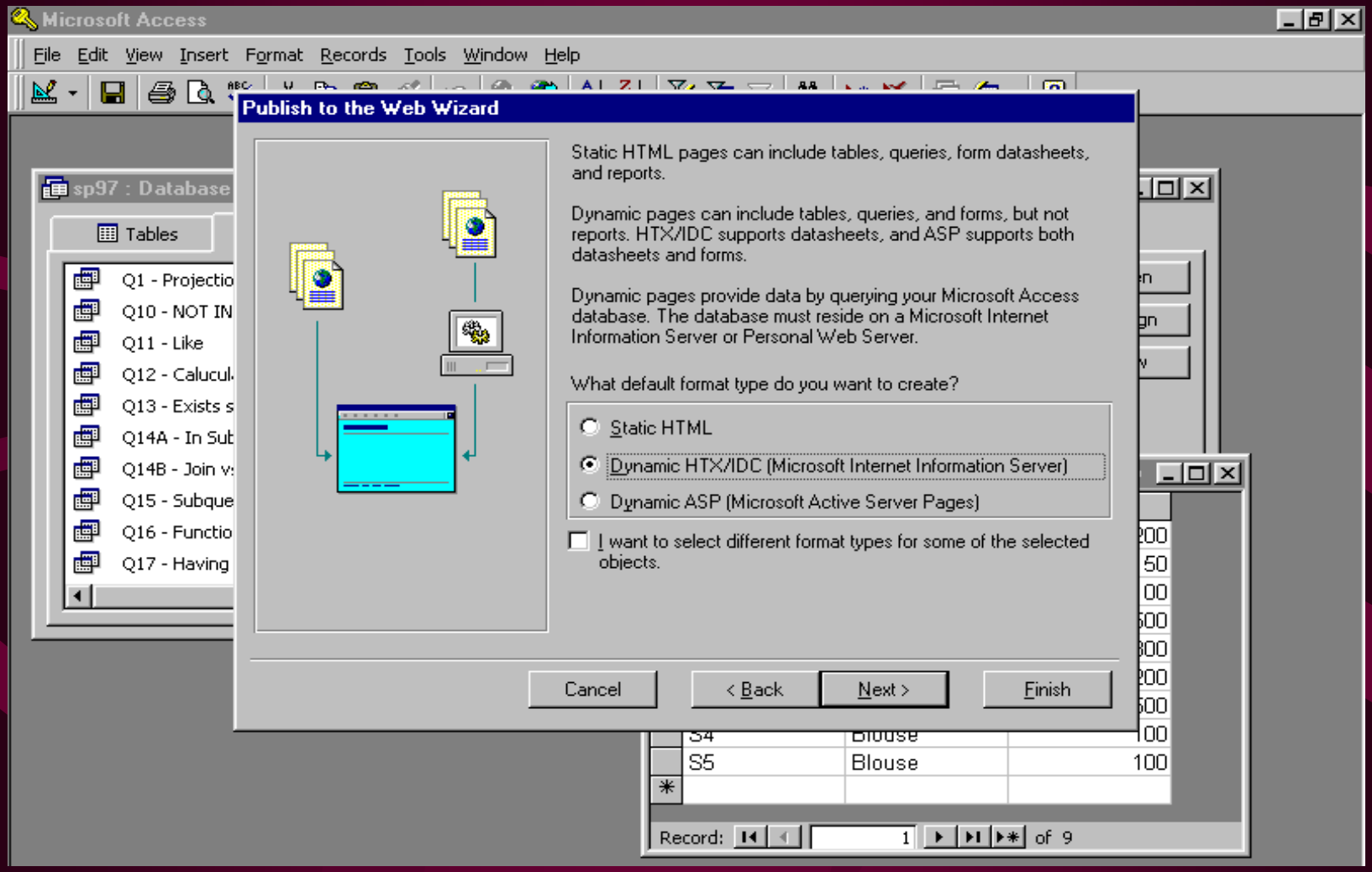
SalesPerson.idc

- Datasource:SP_Access
- Template:SalesPerson.htx
- SQLStatement:SELECT * FROM [S]
- Password:
- Username:

SalesPerson.htx

- Standard HTML except uses placeholders for actual data values:
  - <%SID%> for example each salesperson id
- The is a header part which is the column titles: <THEAD> tag
- And a detail part which is the information for each row: <TBODY>
- Both are generally formatted as tables

Using Access to Make IDC Files



Generated .idc file

- Datasource:SP_Access
- Template:Q7 - Simple Join.htx
- SQLStatement:SELECT DISTINCTROW SP.SID, P.PName, SP.Qty
- +FROM P INNER JOIN SP ON P.PID = SP.PID;

- Password:
- Username:

Generated .htx File

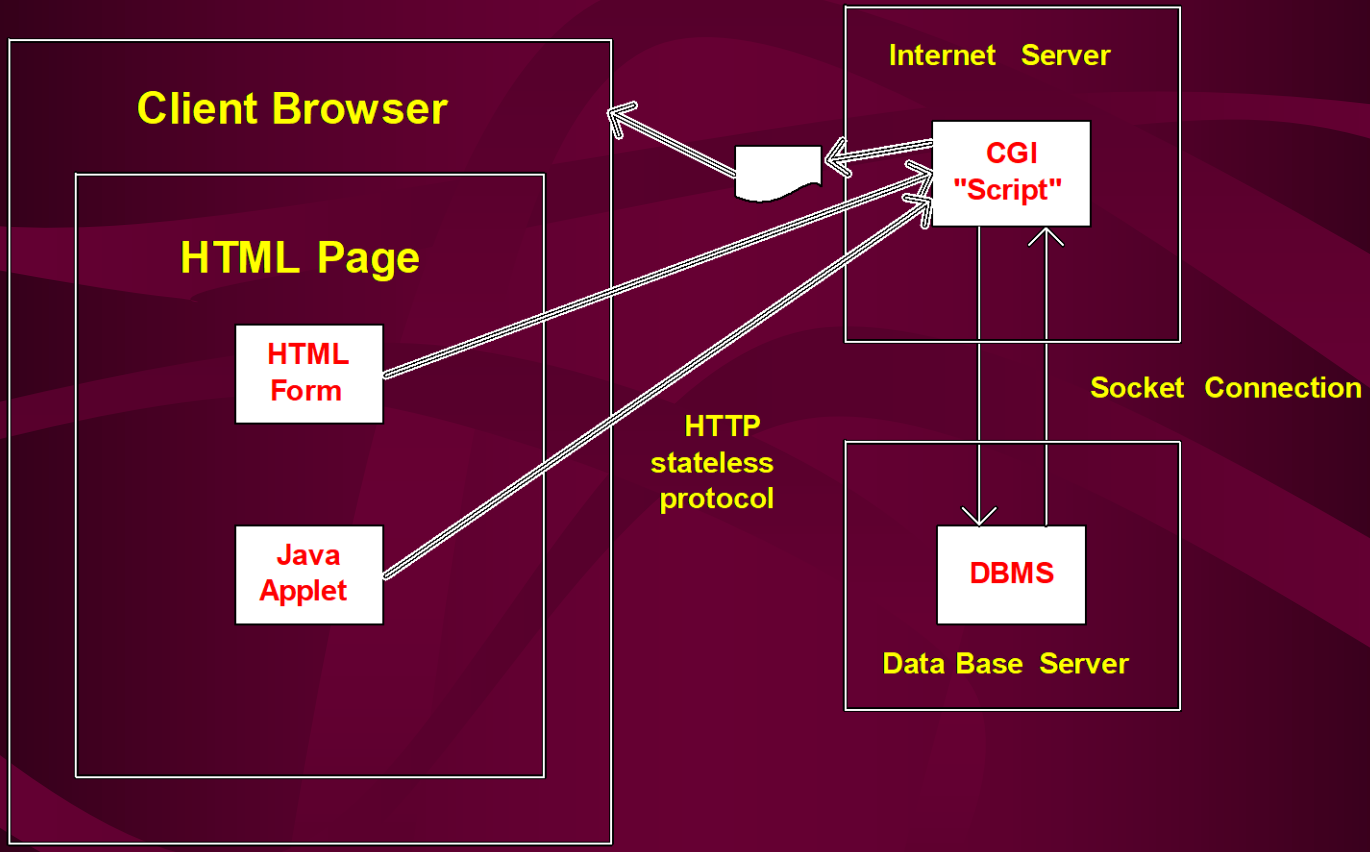
```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Q7 - Simple Join (Class Exercise)</TITLE>
</HEAD>
<BODY>
<TABLE BORDER=1 BGCOLOR=#ffffff CELLSPACING=0><FONT FACE="Arial" COLOR=#000000>
<CAPTION><B>Q7 - Simple Join</B></CAPTION>
<THEAD>
<TR>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>SID</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>PName</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Qty</FONT></TH>
</TR>
</THEAD>
<TBODY>
<%BeginDetail%>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000><%SID%><BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000><%PName%><BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000><%Qty%><BR></FONT></TD>
</TR>
<%EndDetail%>
</TBODY>
<TFOOT></TFOOT>
</TABLE>
</BODY>
</HTML>
```

Note how much smaller this HTML is than prior example

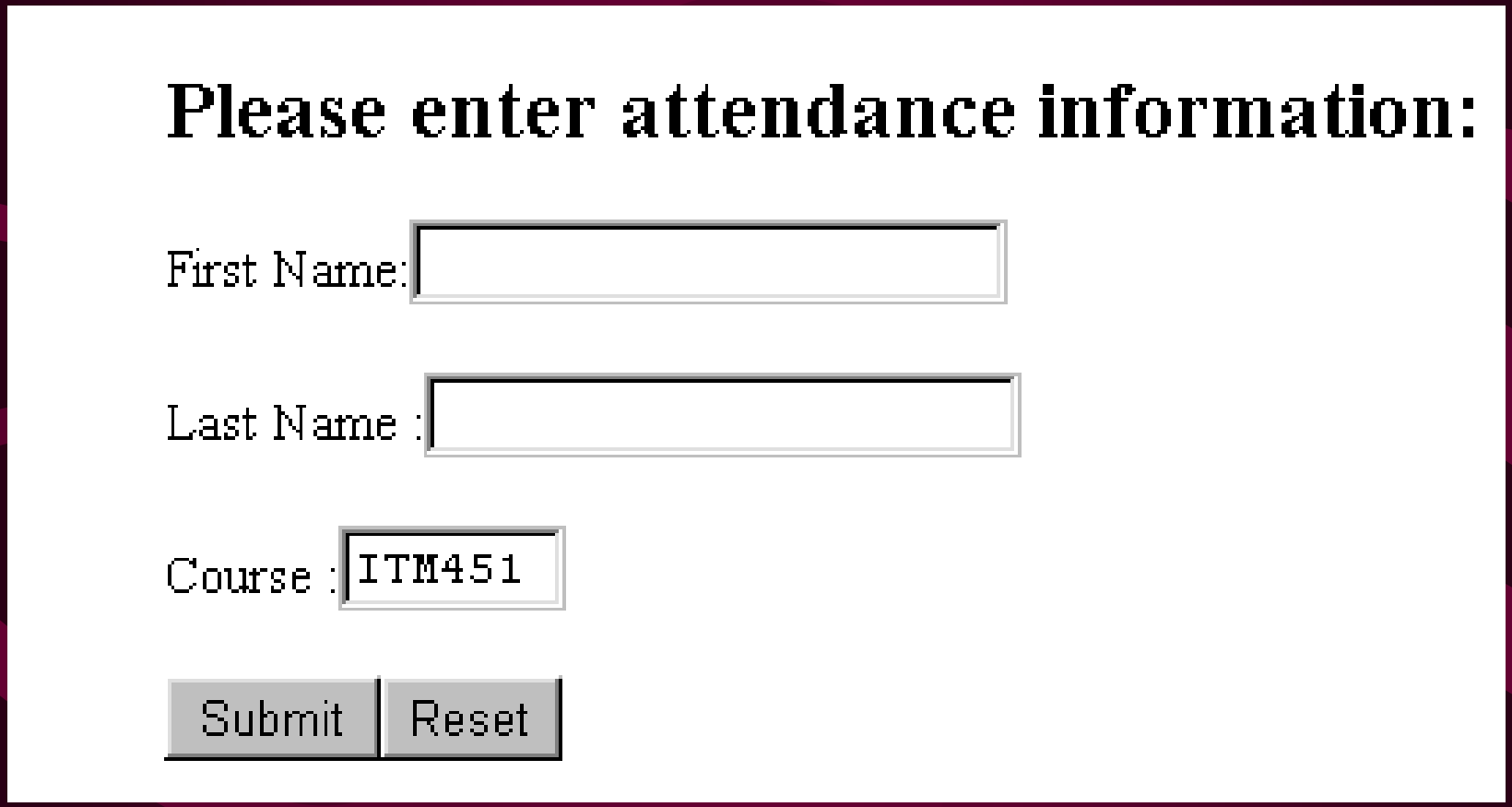
Web Server Interface

- API (Application Program Interface) – Software becomes “part of information server” process
  - ISAPI
    - Microsoft specific
    - Uses *.dll’s
    - Master process started once
  - NSAPI
    - Platform independent
    - Master process started once
- CGI
  - General solution
  - Separate host process (with separate memory areas) for each program executing

DB Query/Update via HTTP/CGI



HTML Forms


Please enter attendance information:
First Name:
Last Name :
Course : ITM451
Submit
Reset

HTML Code for Form

- `<FORM METHOD=POST ACTION="attend.cgi">`
- `<H2>Please enter attendance information:</H2>`
- `<P>First Name:<INPUT NAME="first" TYPE=text SIZE="20">`
- `<P>Last Name :<INPUT NAME="last" TYPE=text SIZE="20">`
- `<P>Course :<INPUT NAME="course" TYPE=text SIZE="7" VALUE="ITM451">`
- `<P><INPUT TYPE=submit VALUE="Submit"><INPUT TYPE=reset VALUE="Reset">`
- `</FORM>`

"attend.cgi" is CGI program

CGI Processing

- A program on the internet server that receives HTTP data (URLencoded) from a client (typically via the submit button on an HTML form)
- The program can be
  - C/C++ (fastest execution)
  - Visual Basic (if MS operating system)
  - Perl (Practical Extraction and Report Language) - an interpreted language
- The stream of data from the client is sent in either “GET” format (command line arguments) or in “POST” (stdin) format

PERL

- Practical Extraction and Report Language
- Interpreted language
- Good character/string handling
- Good interface to Unix system functions
- User to create server side programs that respond to HTTP get/post commands, and then interact with a DBMS (or file system) and return HTML pages to browser
- Free and platform independent
- C like syntax, but less powerful

Form Validation

- HTML Forms
  - Server side
    - CGI script validation - host validation including error codes from DBMS
    - API validation - host validation including error codes from DBMS
  - Client Side (Java Script or VBScript) - client side validation and navigation control and browser dependent (also source is typically visible); since JavaScript is neutral, it is normally used
- Java Forms (Applets)
  - Client side validation, navigation control, browser independent, and inheritance

Example Java Script Form Validation

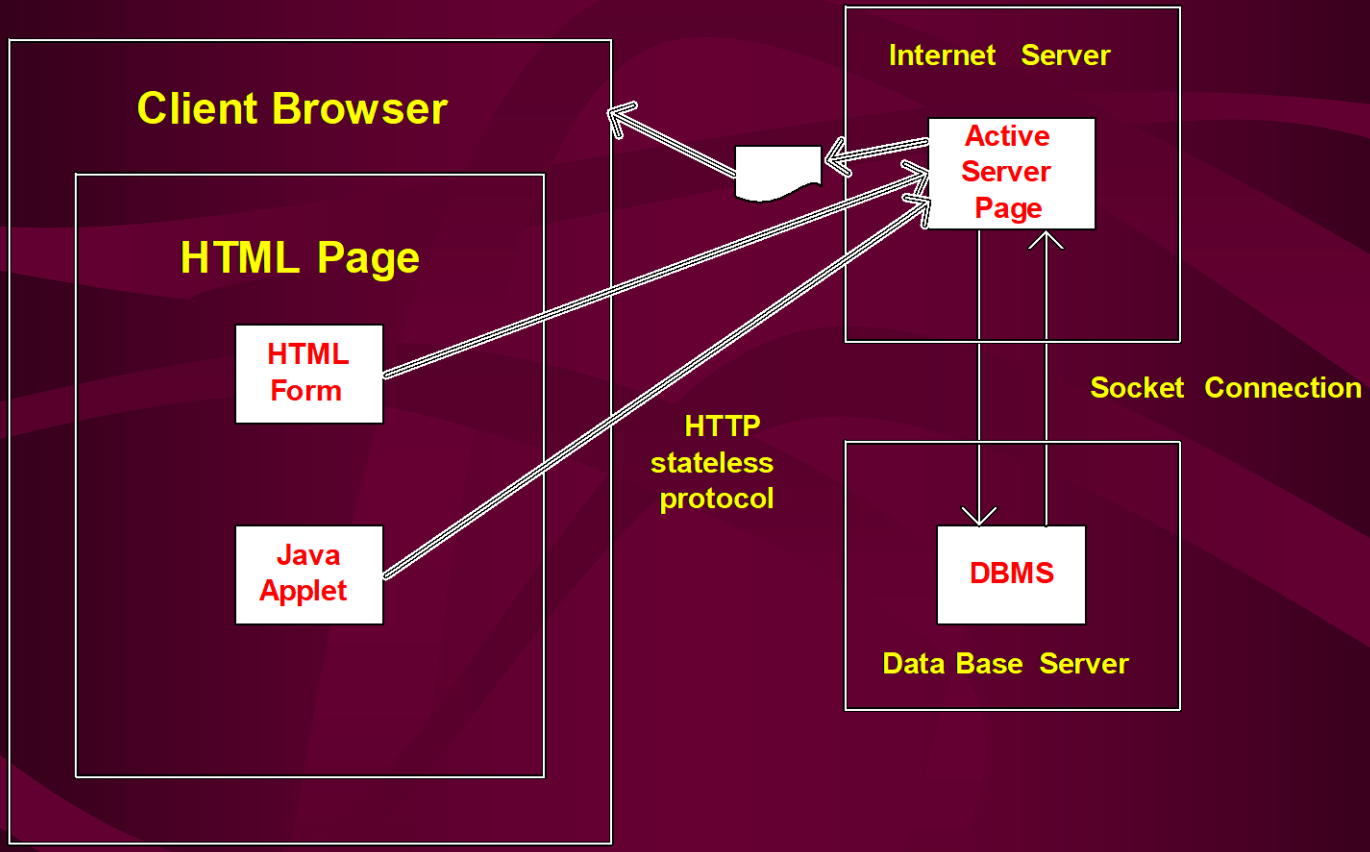
- <HTML><HEAD><TITLE>Some Title</TITLE>
- <SCRIPT LANGUAGE = “JavaScript”>
  - var helpArray = [“Enter your name here”,…];
  - function helpText(msgNum) {
    - myForm.helpBox.value = helpArray[msgNum]; }
  - function validateField (fieldNum) {…}
- </SCRIPT></HEAD><BODY>
- <FORM ID = “myForm”>
- Name: <INPUT TYPE = “text” NAME = “name”
- ONFOCUS = “helpText(0)” ONBLUR = “validateField(0)”>
- …
- <TEXTAREA NAME = “helpBox” STYLE = “position: absolute;
- right: 0; top: 0” ROWS = 4 COLS = 45>…</TEXTAREA>
- </FORM></BODY></HTML>

Active Server Pages (or ActiveX Server Pages)

- ASP is a Microsoft developed technology for sending dynamic Web content - which includes HTML, Dynamic HTML, ActiveX controls, client side scripts, and Java Applets to an internet client
- ASP is like running an Access form (or datasheet) locally except, it occurs over the internet; it is relatively slow and provides minimal validation; also if Access is used as the database (instead of SQL Server, there is no transaction or concurrency control)

- A client sends an HTTP request to an information server (an NT or Windows 2000 server, or a Unix server with special third party software) running an ASP process (a server side ActiveX controll “asp.dll”)
- The server receives the request and directs it the the designated ASP
- The ASP program executes (which often involves interacting with a database) and returns its result to the client (typically in the form of an HTML document)

Active Server Pages



- A number of scripting languages can be used in the ASP
- Typically Visual Basic Script (VBScript) is used and is the default (another scripting language can be specified with the @LANGUAGE tag)

Client Side vs Server Side Scripting

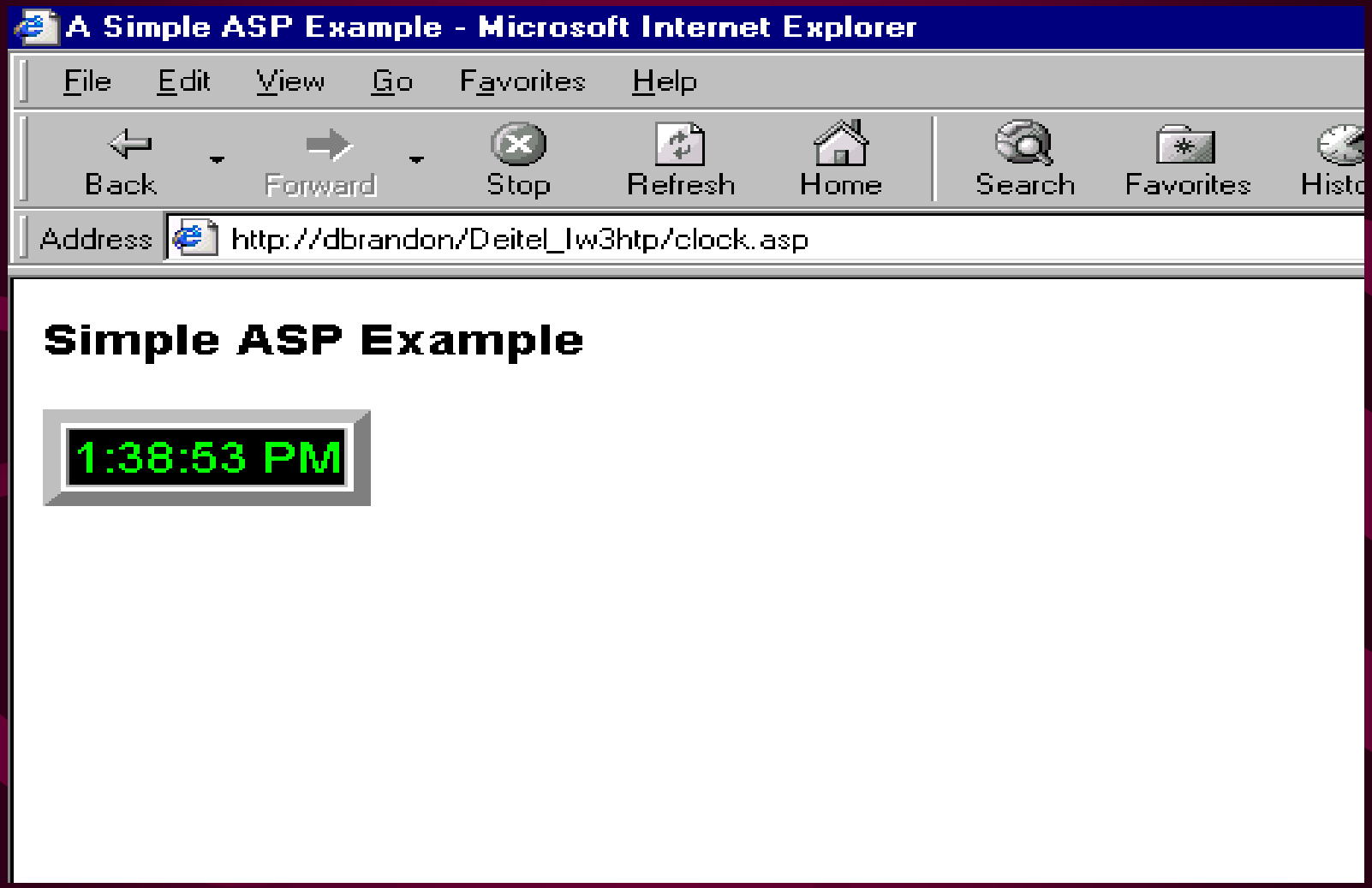
- Server side scripting allows greater flexibility
- Source code is not visible on server side
- Server scripting is implemented on a particular server, and cross platform compatibility is not an issue (VBScript is typically used for Windows servers)
- A typical example is for a client to request to see flights from one place to another; the server script would formulate and send an SQL request to the database (on the same or another server); when it got the results it would format them in and HTML table and send it back to the client

Creating ASP's

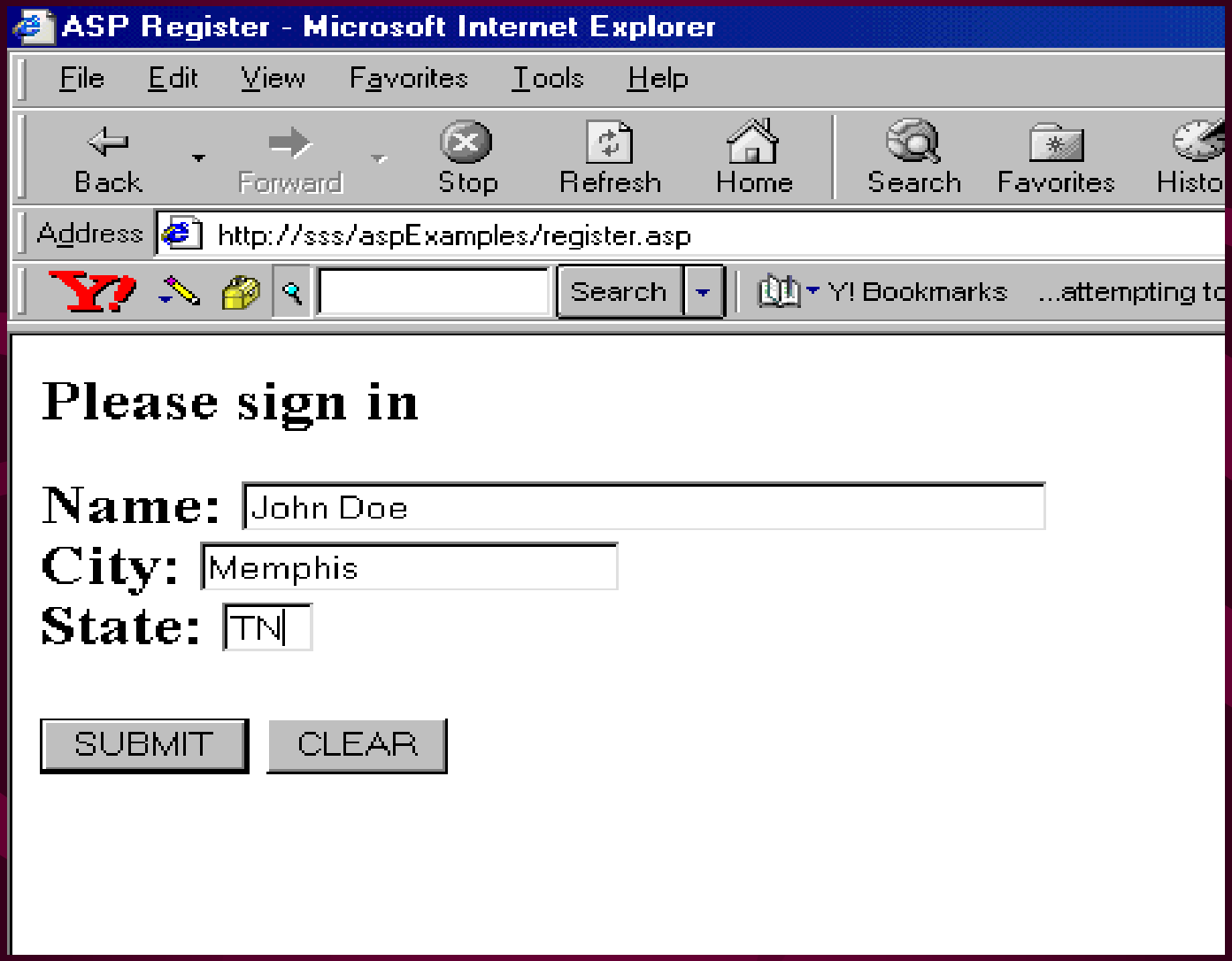
- Microsoft Access
- Microsoft Visual Studio (.Net)
- Case Packages (ie DBApp)
- Manually

Simple ASP for a Clock

```
<% @LANGUAGE=VBScript %><% Option Explicit %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>A Simple ASP Example</TITLE>
<META HTTP-EQUIV="REFRESH" CONTENT="60; URL=CLOCK.ASP">
</HEAD><BODY>
<FONT FACE=ARIAL SIZE = 4><STRONG>Simple ASP Example<STRONG>
</FONT><P>
    <TABLE BORDER="6">
    <TR><TD BGCOLOR="#000000">
        <FONT FACE=Arial COLOR="#00FF00" SIZE=4>
            <% =Time() %>            <% … %> indicates script
        </FONT>
    </TD></TR>
    </TABLE>
</BODY></HTML>
```

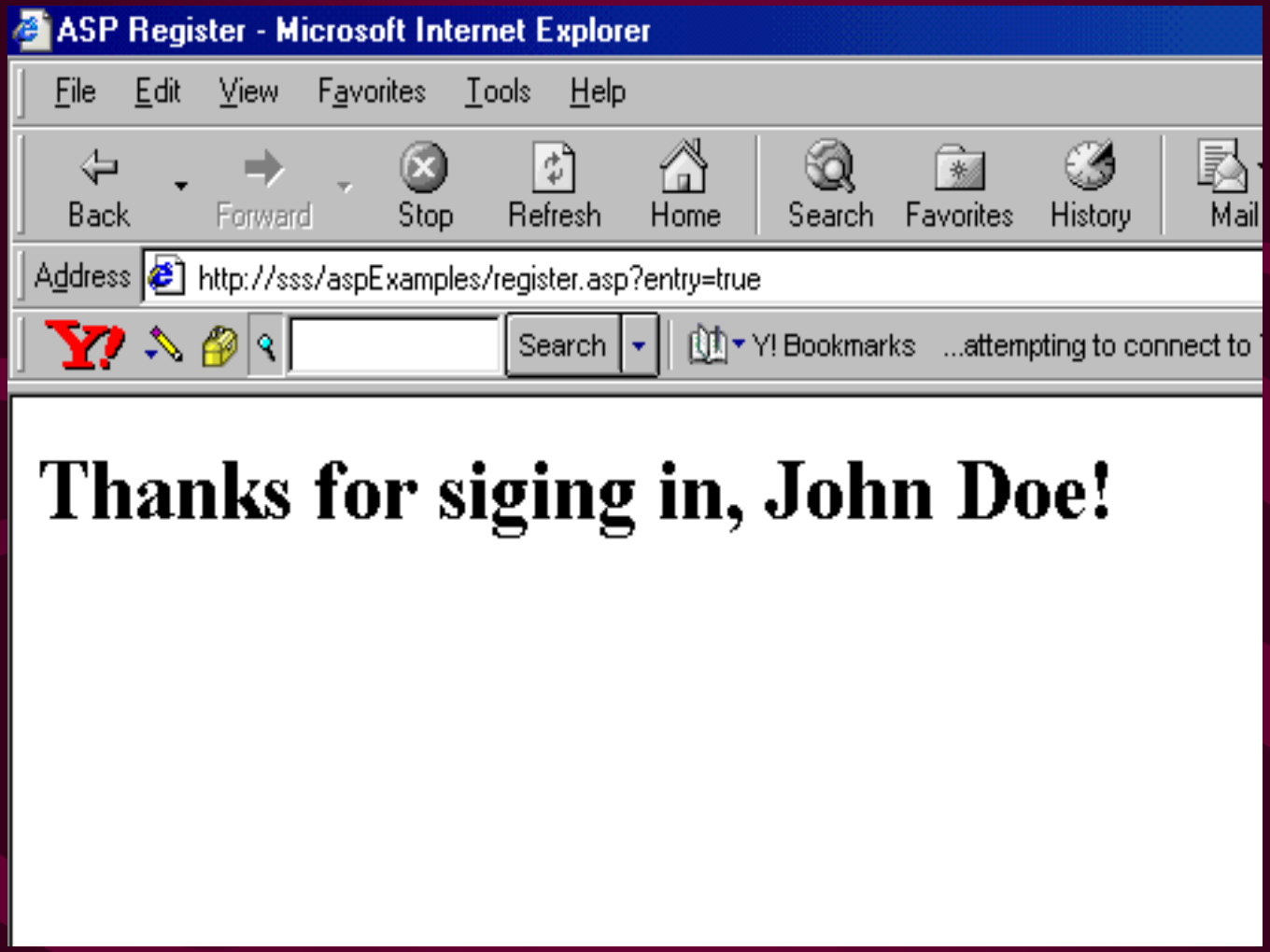

A Simple ASP Example - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites
Address http://dbrandon/Deitel_Iw3htp/clock.asp
Simple ASP Example
1:38:53 PM

Form Processing with ASP

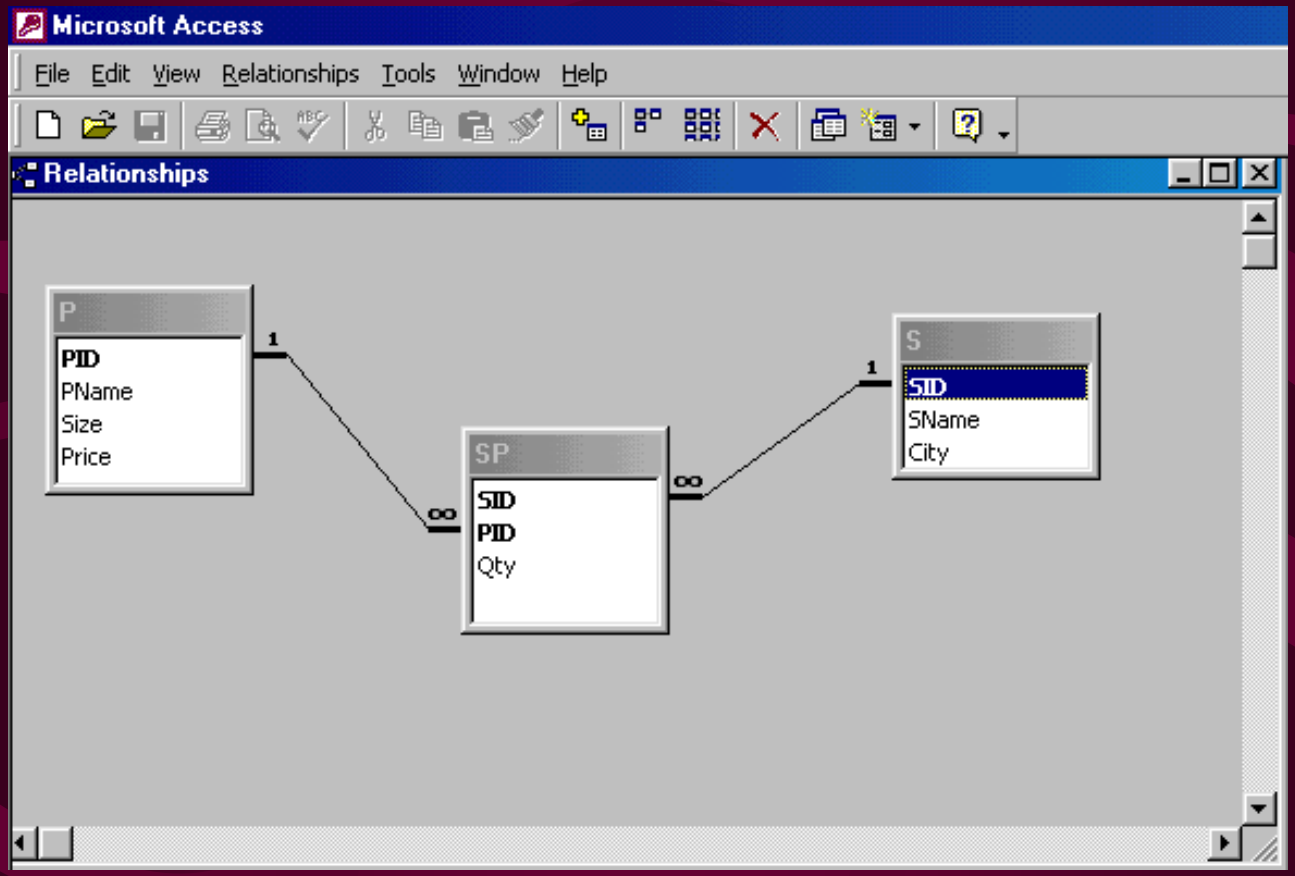


Form Generation & Processing with ASP
[register.asp]

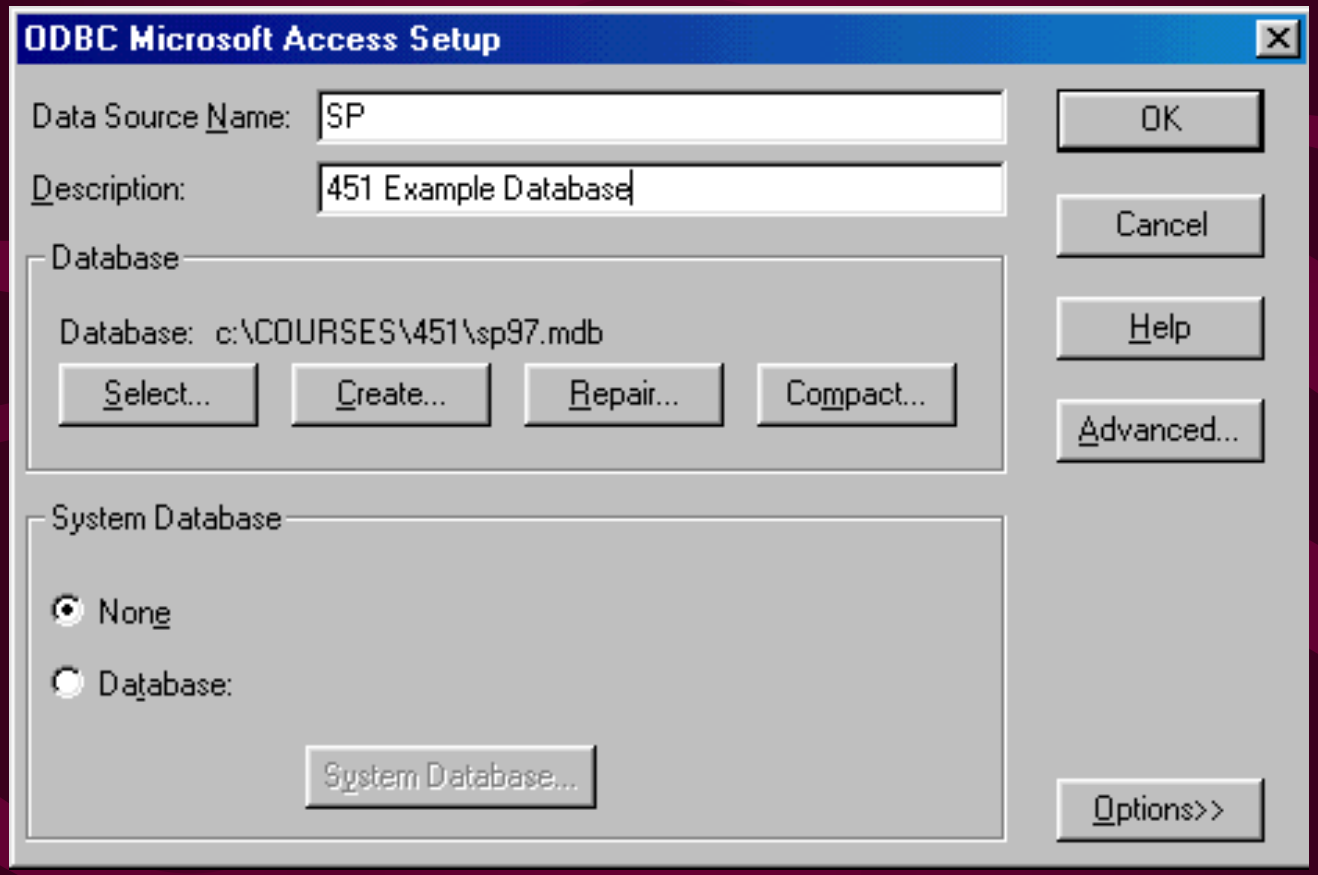
```
<% @LANGUAGE = VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP Register</TITLE>
<BODY>
<H1>
<%
   If Request( "entry" ) = "true" Then
      Call Response.Write( "Thanks for siging in, " )
      Call Response.Write( Request( "name" ) & "!" )
   Else
%>
</H1>
<H2>Please sign in<BR>
<FORM ACTION = "register.asp?entry=true" METHOD = "POST">
Name:  <INPUT TYPE = "text" FACE = "Arial"    SIZE = "40" NAME = "name"><BR>
City:  <INPUT TYPE = "text" FACE = "Arial" SIZE = "20" NAME = "city"><BR>
State:  <INPUT TYPE = "text" FACE = "Arial" SIZE = "3" NAME = "state"><BR><BR>
<INPUT TYPE = "submit" VALUE = "SUBMIT">
<INPUT TYPE = "reset" VALUE = "CLEAR">
<%
      End If
%>
</BODY></HTML>
```


ASP Register - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address http://sss/aspExamples/register.asp?entry=true
Search Y! Bookmarks ...attempting to connect to
Thanks for siging in, John Doe!

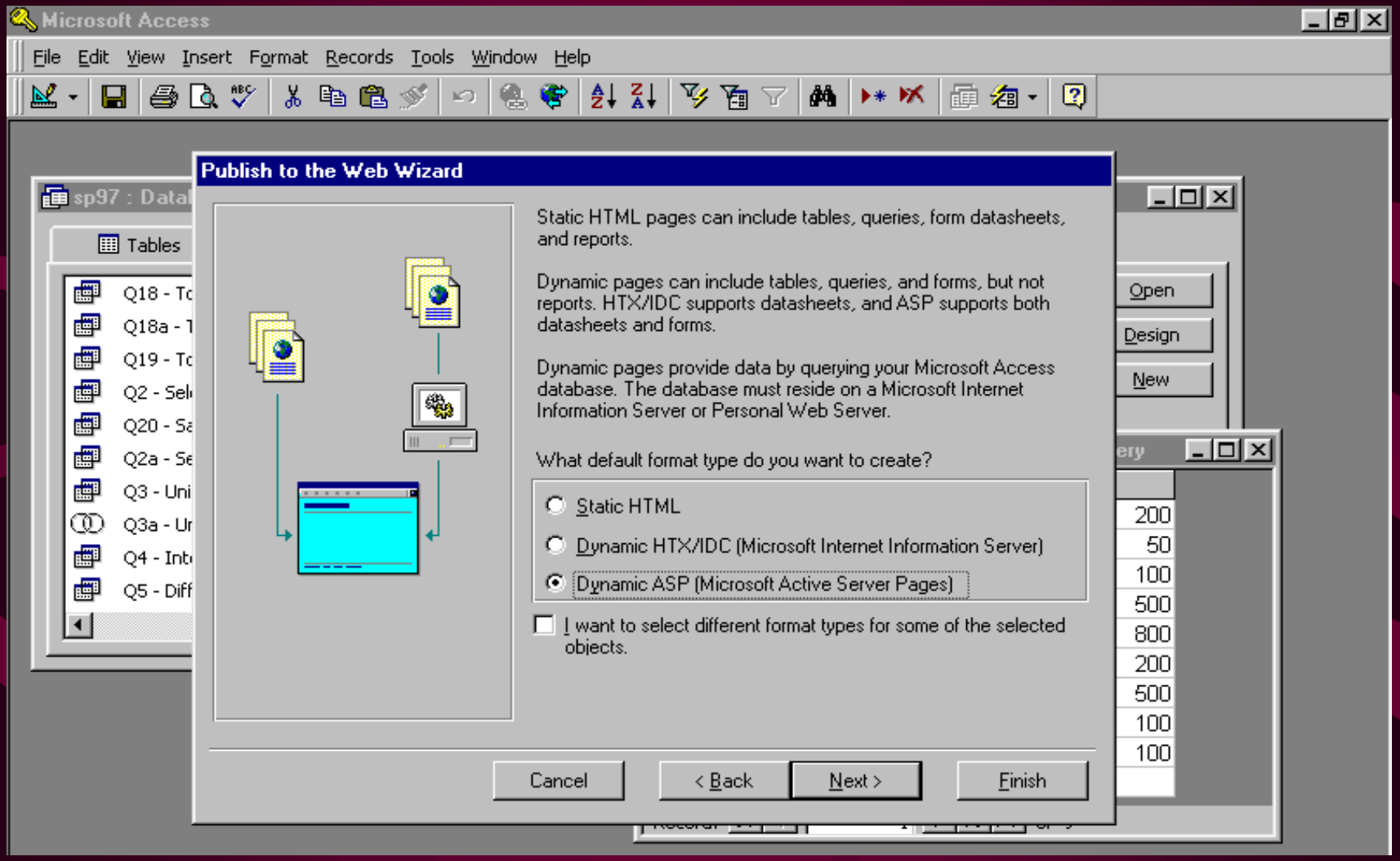
Access Database Example



ODBC System DSN



Using Access to Generate ASP



Generated ASP

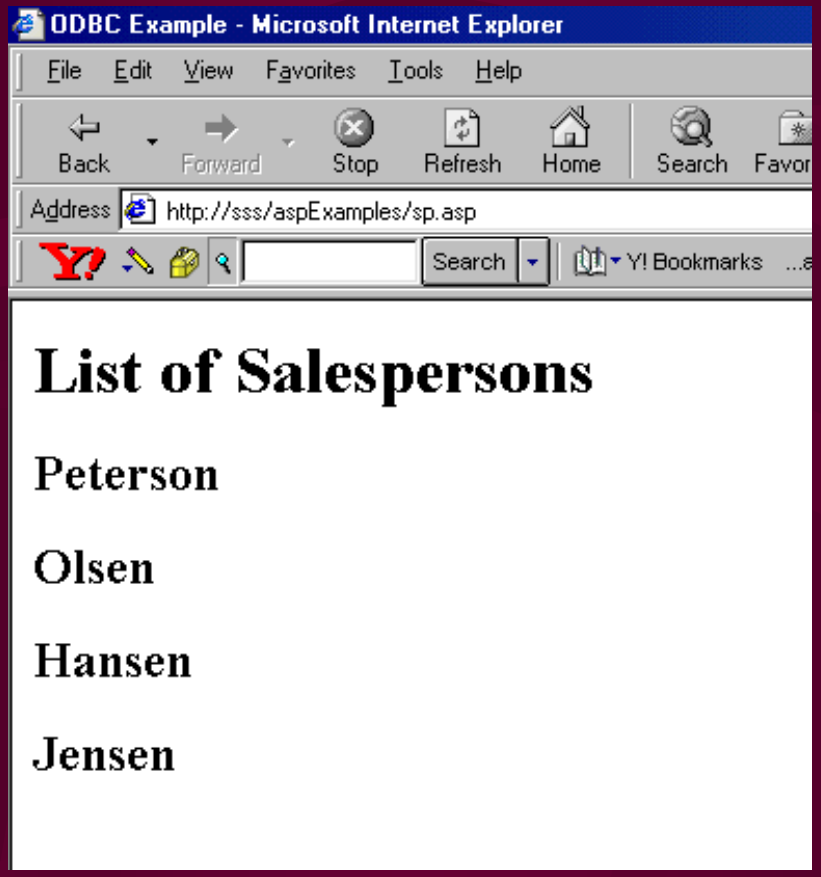
```
<HTML><HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Q7 - Simple Join (Class Exercise)</TITLE></HEAD><BODY>
<%
Param = Request.QueryString("Param")
Data = Request.QueryString("Data")
%>
<%
If IsObject(Session("SP_Access_conn")) Then
    Set conn = Session("SP_Access_conn")
Else
    Set conn = Server.CreateObject("ADODB.Connection")
    conn.open "SP_Access","",""
    Set Session("SP_Access_conn") = conn
End If
%>
<%
    sql = "SELECT DISTINCTROW SP.SID, P.PName, SP.Qty  FROM P INNER JOIN SP ON P.PID = SP.PID
"
    If cstr(Param) <> "" And cstr(Data) <> "" Then
        sql = sql & " WHERE [" & cstr(Param) & "] = " & cstr(Data)
    End If
    Set rs = Server.CreateObject("ADODB.Recordset")
    rs.Open sql, conn, 3, 3
%>
```

```
<TABLE BORDER=1 BGCOLOR=#ffffff CELLSPACING=0><FONT FACE="Arial" COLOR=#000000><CAPTION><B>Q7 - Simple Join (Class Exercise)</B></CAPTION>
<THEAD><TR>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>SID</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>PName</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial" COLOR=#000000>Qty</FONT></TH></TR>
</THEAD><TBODY>
<%On Error Resume Next
rs.MoveFirst
do while Not rs.eof %>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000><%=Server.HTMLEncode(rs.Fields("SID").Value)%><BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial" COLOR=#000000><%=Server.HTMLEncode(rs.Fields("PName").Value)%><BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial" COLOR=#000000><%=Server.HTMLEncode(rs.Fields("Qty").Value)%><BR></FONT></TD>
</TR>
<%rs.MoveNextloop%>
</TBODY><TFOOT></TFOOT></TABLE></BODY></HTML>
```

Manually Written ASP

```
<% @LANGUAGE = VBScript %>
<% Option Explicit %>
<HTML>
<HEAD><TITLE>ODBC Example</TITLE></HEAD>
<% Dim connection, query, data
   Set connection = Server.CreateObject( "ADODB.Connection" )
   Call connection.Open( "SP" )
   ' Create the SQL query
   query = "SELECT * FROM S"
   ' Create the recordset
   Set data = Server.CreateObject( "ADODB.Recordset" )
   Call data.Open( query, connection )
   ' If an error occurs, ignore it
   On Error Resume Next
%>
<BODY>
<H1>List of Salespersons</H1>
<%While Not data.EOF%>
      <H2><% =data( "SName" ) %></H2>
     <%Call data.MoveNext()
Wend%>
</BODY></HTML>
```

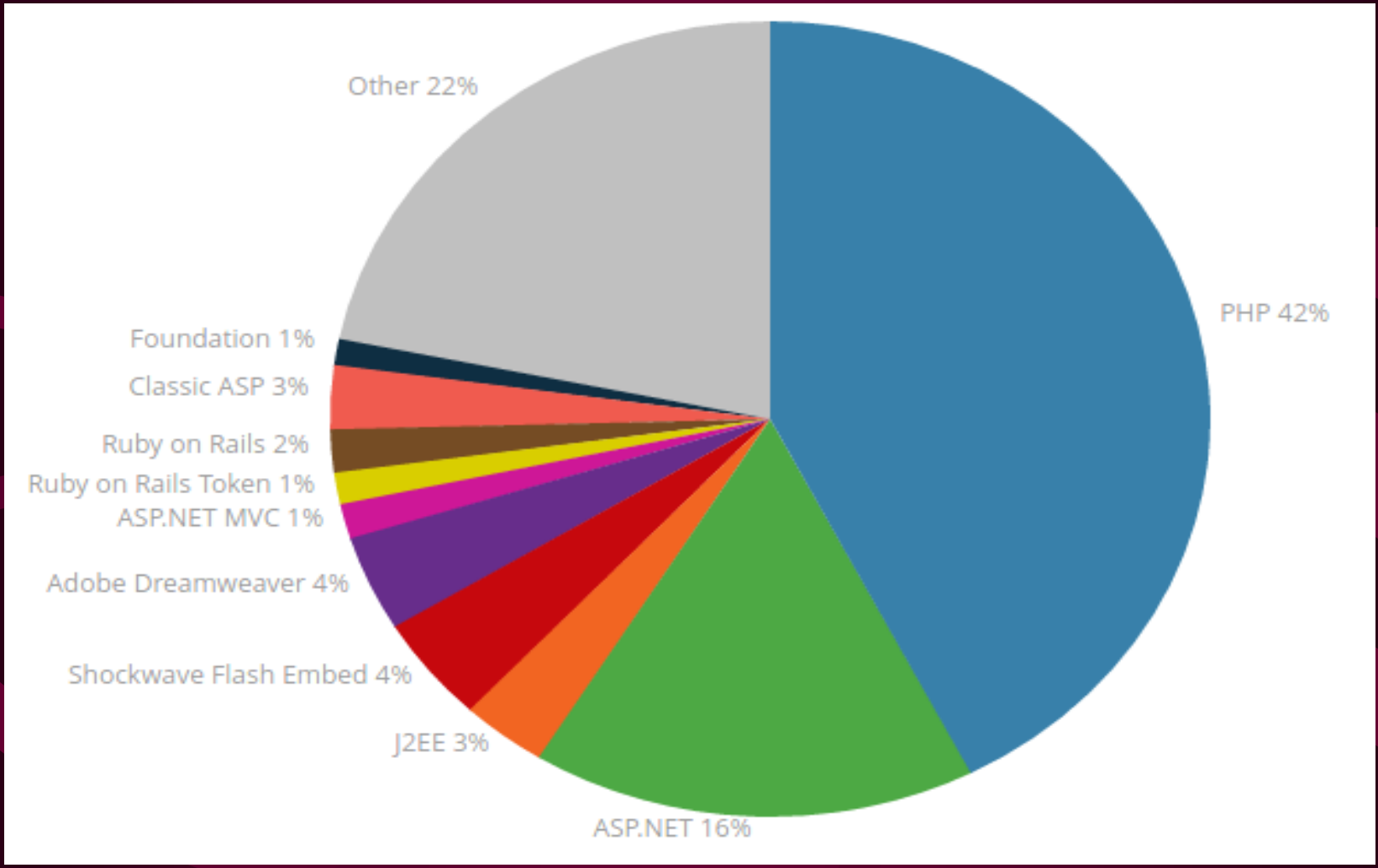
Running ASP


ODBC Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favor
Address http://sss/aspExamples/sp.asp
Search Y! Bookmarks
List of Salespersons
Peterson
Olsen
Hansen
Jensen

Case Study

Internet Database Interaction Using Modern Web Server Technology

Web Server Application Languages



Case Study Form

HTML for Form

```
<HTML>
<HEAD>
  <TITLE>Add a New Person</TITLE>
</HEAD>
<BODY>
  <FORM ACTION="changeme.cgi">
    <TABLE>
      <TR>
        <TD>First Name:</TD>
        <TD><INPUT TYPE="text" NAME="firstname"></TD>
      </TR>
      <TR>
        <TD>Last Name:</TD>
        <TD><INPUT TYPE="text" NAME="lastname"></TD>
      </TR>
      <TR>
        <TD>Zip Code:</TD>
        <TD><INPUT TYPE="text" NAME="zipcode"></TD>
      </TR>
      <TR>
        <TD COLSPAN="2" ALIGN="RIGHT">
        <INPUT TYPE="submit" VALUE="Add Person">
        </TD>
      </TR>
    </TABLE>
  </FORM>
</BODY>
</HTML>
```

HTML Response page

I added Brian Jepson to the database. Here is the complete list of people:

Name	Zip Code
Brian Jepson	02881

SQL

- Add row to PEOPLE table:
  - INSERT INTO PEOPLE VALUES (firstname,, lastname, zipcode)
- Retrieve all data for all rows in PEOPLE table:
  - SELECT * FROM PEOPLE

SavePerson.cgi (Perl)

```
#!/usr/bin/perl -w

use strict; # restrict unsafe constructs
use DBI;    # import the DBI module

use CGI qw(:standard *table); # import the CGI functions

# Move form data into local variables.
#
my $firstname = param("firstname");
my $lastname  = param("lastname");
my $zipcode   = param("zipcode");

# Print the standard text/html header before you do
# anything that might raise an error.
#
print header();

# Connect to the local MySQL server.
#
my $user   = "bjepson";
my $passwd = "password";
my $dsn    = "DBI:mysql:database=bjepson";
my $dbh = DBI->connect($dsn, $user, $passwd);
if (!$dbh) { die "Error connecting: $DBI::errstr.\n" };

# Insert the new person.
#
my $sql = "INSERT INTO PEOPLE VALUES
          ('$firstname', '$lastname', '$zipcode')";

if (!$dbh->do($sql)) {
  print "Warning: SQL statement:
        <PRE>$sql</PRE>
        failed!!!<BR>";
  exit;
}
```

```
# Display the HTML document.
#
print start_html("New Person: $firstname $lastname");
print "I added $firstname $lastname to the database.
       Here is the complete list of people:";

# Retrieve the people with a SELECT statement.
#
my $stmt = $dbh->prepare("SELECT * FROM PEOPLE");

if (!$stmt->execute()) {
  print "<BR>Warning: SELECT statement failed!!!<BR>";
  exit;
}
# Fetch all the rows and display them in a table.
#
print start_table({-border => 1});
while (my @row = $stmt->fetchrow_array) {

  my ($first, $last, $zip) = @row;
  print "<TR>
          <TD>$first $last</TD><TD>$zip</TD>
         </TR>";

}
$stmt->finish;
print end_table;

print end_html;
$dbh->disconnect;
```

SavePerson.asp (Active Server Pages)

```
<%@ page language="C#" contentType="text/html" %>
<%@ import namespace= "System.Data" %>
<%@ import namespace= "System.Data.OleDb" %>

<%

 /*
  * Move the form data into local variables.
  */
 string firstname = Request["firstname"];
 string lastname  = Request["lastname"];
 string zipcode   = Request["zipcode"];

 /*
  * Connect to the database server.
  */
 string conn_str =
   "provider=SQLOLEDB;server=(local)\\NetSDK;uid=sa;";
 OleDbConnection conn = new OleDbConnection(conn_str);
 conn.Open();

 /*
  * Create an SQL statement to insert the new person.
  */
 string sql_template =
   "INSERT INTO PEOPLE VALUES ('{0}', '{1}', '{2}')";
 string sql = String.Format(sql_template,
                            firstname,
                            lastname,
                            zipcode);
 /*
  * Send the SQL to the database.
  */
 OleDbCommand cmd = new OleDbCommand(sql, conn);
 cmd.ExecuteNonQuery();

%>
```

```
<HTML>
<HEAD>
  <TITLE>
    New Person: <%= firstname + " " + lastname %>
  </TITLE>
</HEAD>

<BODY>
  I added <%= firstname + " " + lastname %> to the
  database.  Here is the complete list of people:

  <%

    /* Fetch all the customers and bind to the
     * HTML table.
     */
    sql = "SELECT firstname + ' ' + lastname " +
          "AS fullname, zipcode FROM PEOPLE";
    OleDbDataAdapter da =
      new OleDbDataAdapter(sql, conn);

    DataSet ds = new DataSet();
    da.Fill(ds, "PEOPLE");
    Table1.DataSource =
      ds.Tables["PEOPLE"].DefaultView;
    Table1.DataBind();

  %>

  <!-- Insert an ASP.NET DataGrid component. -->
  <asp:DataGrid id="Table1"
                AutoGenerateColumns="false"
                runat="server">
    <Columns>
      <asp:BoundColumn
          HeaderText="Name"
          DataField="fullname"/>
      <asp:BoundColumn
          HeaderText="Zip Code"
          DataField="zipcode"/>
    </Columns>

  </asp:DataGrid>

</BODY>
</HTML>
```

SavePerson.JSP (Java Server Pages)

```
<%@ page language="java"
contentType="text/html" %>
<%@ page import = "java.sql.*" %>
<%
/*
* Move the form data into local
variables.
*/
String firstname =
request.getParameter("firstname");
String lastname =
request.getParameter("lastname");
String zipcode =
request.getParameter("zipcode");
/*
* Load the JDBC driver for
PostgreSQL.
*/
Class.forName("org.postgresql.Driver");
/*
* Connect to the database server.
*/
String connection_string =
"jdbc:postgresql:testdb";
String username = "bjepson";
String password = "password";
Connection conn = DriverManager.
getConnection(connection_string,
username, password);
Statement stmt = conn.
createStatement();
/*
* Create an SQL statement to insert
the new person.
*/
String sql = "INSERT INTO PEOPLE " +
"VALUES('" + firstname + "', " +
"'" + lastname + "', " +
"'" + zipcode + "') ";
/*
* Send the SQL to the database.
*/
stmt.executeUpdate(sql);
%>
```

```
<HTML>
<HEAD>
<TITLE>
New Person: <%= firstname + " " +
lastname %>
</TITLE>
</HEAD>
<BODY>
I added <%= firstname + " " +
lastname %> to the database. Here is
the complete list of people:
<TABLE BORDER>
<TH>Name</TH><TH>Zip Code</TH>
<%
/*
* Fetch each row in the table,
and display it as an HTML table row.
*/
ResultSet rs =
stmt.executeQuery("SELECT *
FROM PEOPLE");
while(rs.next()) {
firstname = rs.getString(1);
lastname = rs.getString(2);
zipcode = rs.getString(3);
%>
<TR>
<TD><%= firstname + " " +
lastname %></TD>
<TD><%= zipcode %></TD>
</TR>
<% } /* end of the while loop */ %>
</TABLE>
</BODY>
</HTML>
```

SavePerson.cfm (Cold Fusion)

```
<!-- Import the form data. -->
<cfset firstname=form.firstname>
<cfset lastname=form.lastname>
<cfset zipcode=form.zipcode>

<!-- Insert the new person. -->
<cfquery name="MyUpdate" dataSource="Local">
  INSERT INTO PEOPLE VALUES (
    <cfqueryparam value="#firstname#"  CFSQLType="CF_SQL_CHAR">,
    <cfqueryparam value="#lastname#"   CFSQLType="CF_SQL_CHAR">,
    <cfqueryparam value="#zipcode#"    CFSQLType="CF_SQL_CHAR">  )
</cfquery>

<HTML>
<HEAD>
  <TITLE>
  New Person:  <cfoutput>#firstname# #lastname#</cfoutput>
  </TITLE>
</HEAD>
```

- <BODY>
- I added <cfoutput>#firstname# #lastname#</cfoutput>
- to the database.
- Here is the complete list of people:

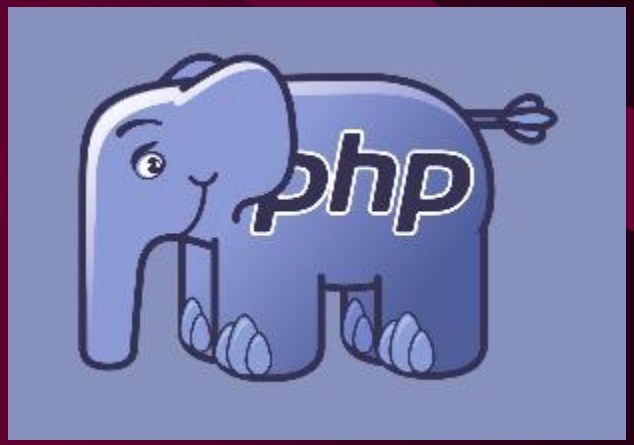
- <!-- Retrieve all the people. -->
- <cfquery name="MyQuery" dataSource="Local">
- SELECT * FROM PEOPLE
- </cfquery>

- <!-- Create a ColdFusion HTML table and associate it with MyQuery -->
- <cftable border HTMLTable query="MyQuery">
- <cfcol header="Name" text="#firstname# #lastname#">
- <cfcol header="Zip" text="#zipcode#">
- </cftable>

- </BODY>
- </HTML>

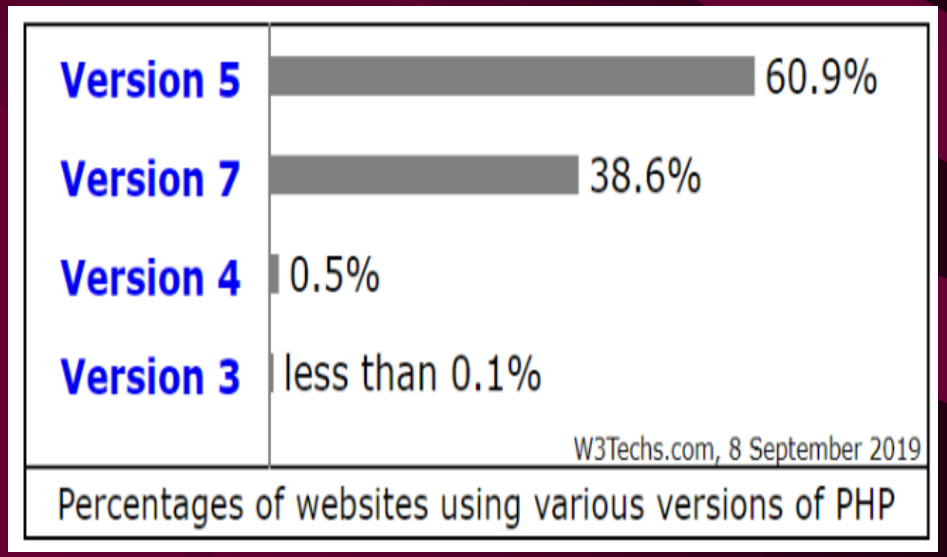
SavePerson.PHP

- <?php
  /*
  * Connect to a MySQL database server.
  */
  $db = mysql_connect(“myhost", "bjepson', "password");
  mysql_select_db("bjepson", $db);
  /*
  * Create an SQL statement to insert the new person.
  */
  $sql = "INSERT INTO PEOPLE VALUES
  ("$firstname", "$lastname", "$zipcode")";
  /*
  * Send the SQL to the database.
  */
  mysql_query($sql, $db);
  ?>



PHP covered in MIS 470

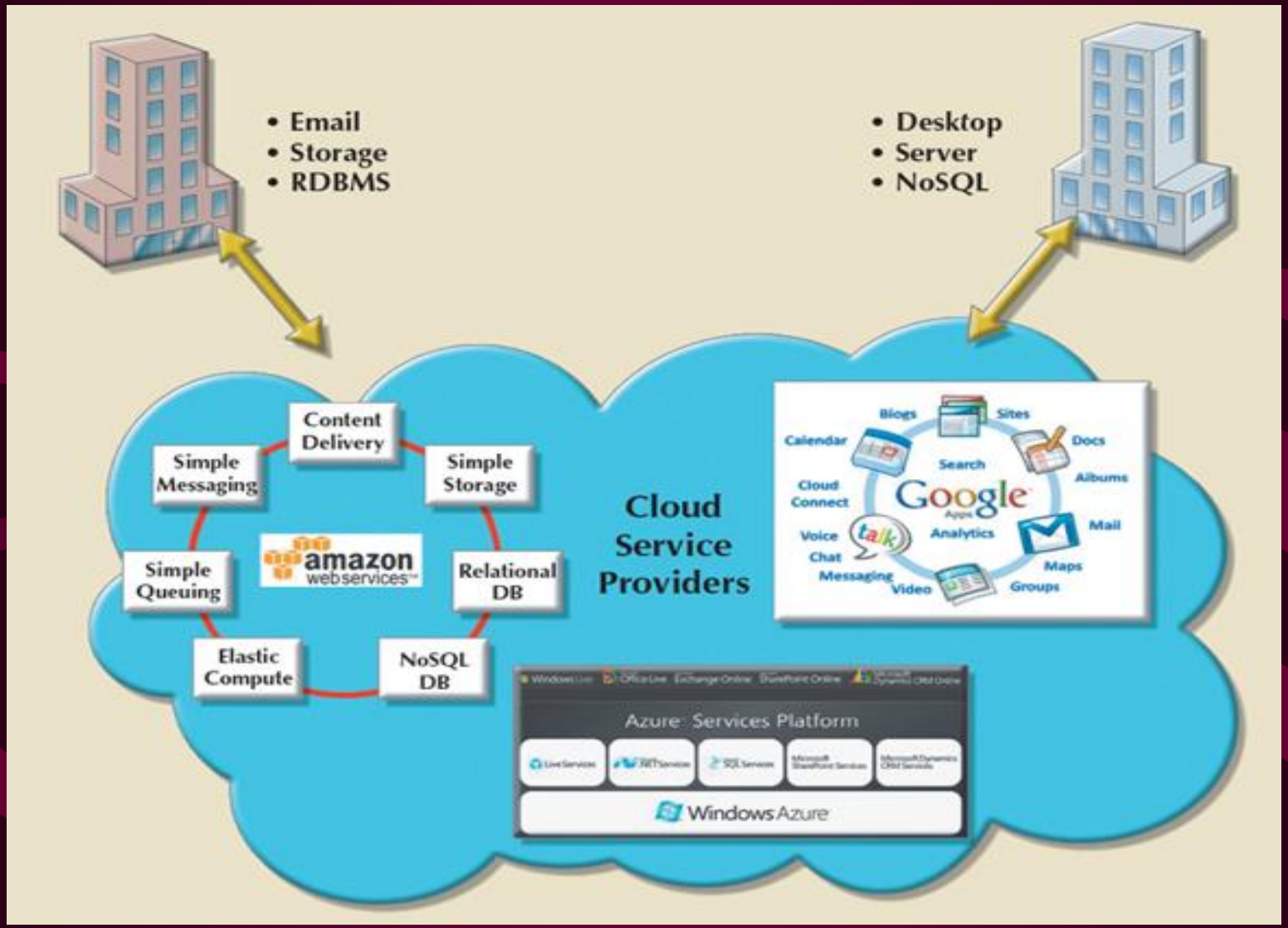
```
<HTML>
<HEAD>
<TITLE>
New Person: <?php echo "$firstname
$lastname"; ?>
</TITLE>
</HEAD>
<BODY>
I added <?php echo "$firstname
$lastname"; ?> to the
database. Here is the complete list of
people:
<TABLE BORDER>
<TH>Name</TH><TH>Zip Code</TH>
<?php
/*
* Fetch each row in the table, and
display it as an HTML table row.
*/
$rs = mysql_query("SELECT * FROM
PEOPLE", $db);
while($row = mysql_fetch_array($rs))
{
$first = $row["firstname"];
$last = $row["lastname"];
$zip = $row["zipcode"];
echo "<TR>
<TD>$first
$last</TD><TD>$zip</TD>
</TR>";
}
?>
</TABLE>
</BODY>
</HTML>
```



Cloud Computing Services

- Computing model that enables access to a shared pool of configurable computer resources
  - Can be rapidly provisioned and released with minimal management effort or service provider interaction
  - Potential to become a game changer; eliminates financial and technological barriers

Cloud Computing Services (con't)



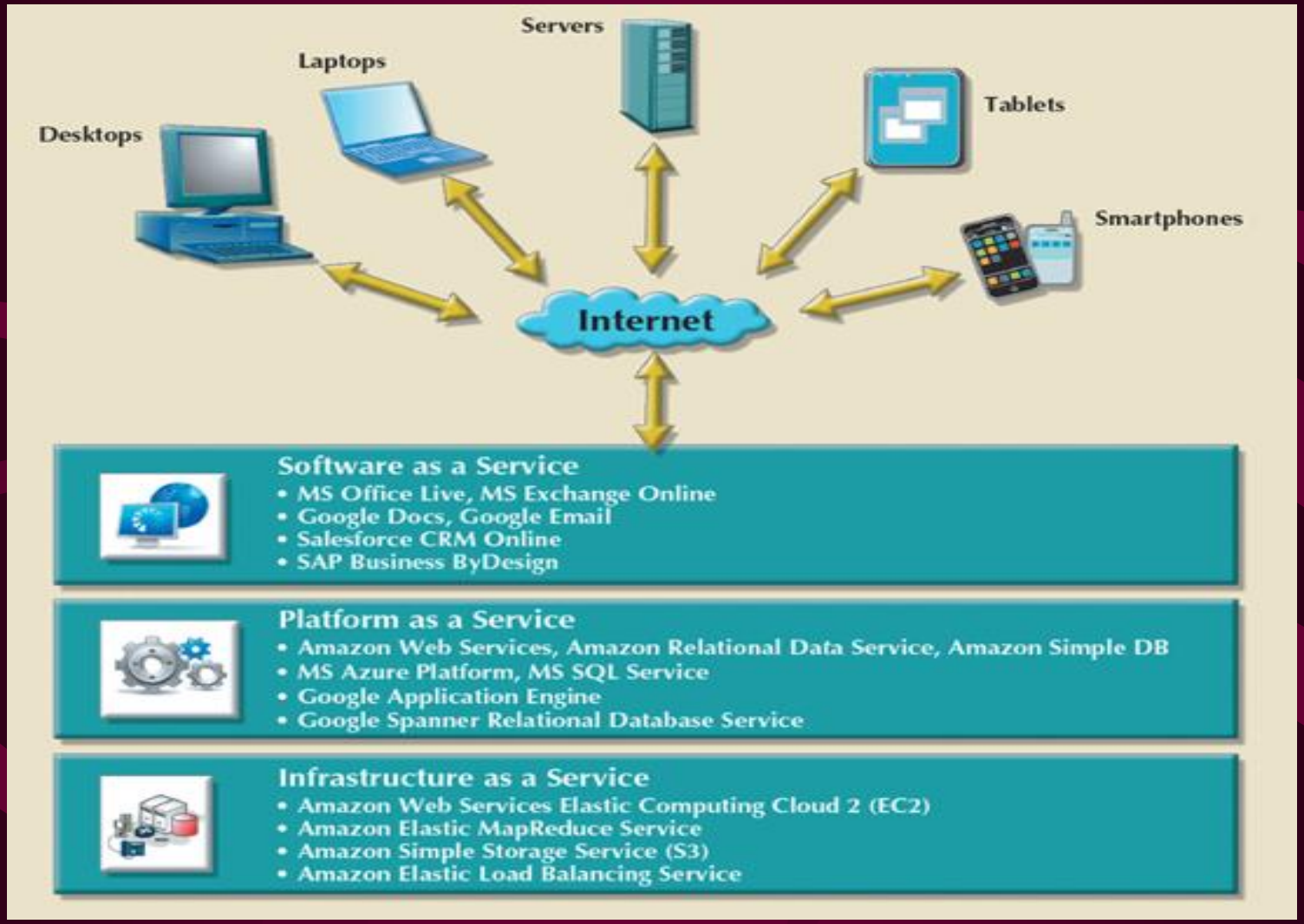
Cloud Implementation Types

- Public cloud
  - Built by a third-party organization to sell cloud services to the general public
- Private cloud
  - Built by an organization for the sole purpose of servicing its own needs
- Hybrid cloud
  - Part public, part private
- Community cloud
  - Built by and for a specific group of organizations that share a common trade

Characteristics of Cloud Services

- Cloud computing services share a set of guiding principles
  - Ubiquitous access via Internet technologies
  - Shared infrastructure
  - Lower costs and variable pricing
  - Flexible and scalable services
  - Dynamic provisioning
  - Service orientation
  - Managed operations

Types of Cloud Services



Cloud Services: Advantages and Disadvantages

Advantage	**Disadvantage**
Low initial cost of entry. Cloud computing has lower costs of entry when compared with the alternative of building in house.	Issues of security, privacy, and compliance. Trusting sensitive company data to external entities is difficult for most data-cautious organizations.
Scalability/elasticity. It is easy to add and remove resources on demand.	Hidden costs of implementation and operation. It is hard to estimate bandwidth and data migration costs.
Support for mobile computing. Cloud computing providers support multiple types of mobile computing devices.	Data migration is a difficult and lengthy process. Migrating large amounts of data to and from the cloud infrastructure can be difficult and time-consuming.
Ubiquitous access. Consumers can access the cloud resources from anywhere at any time, as long as they have Internet access.	Complex licensing schemes. Organizations that implement cloud services are faced with complex licensing schemes and complicated service-level agreements.
High reliability and performance. Cloud providers build solid infrastructures that otherwise are difficult for the average organization to leverage.	Loss of ownership and control. Companies that use cloud services are no longer in complete control of their data. What is the responsibility of the cloud provider if data are breached? Can the vendor use your data without your consent?
Fast provisioning. Resources can be provisioned on demand in a matter of minutes with minimal effort.	Organization culture. End users tend to be resistant to change. Do the savings justify being dependent on a single provider? Will the cloud provider be around in 10 years?
Managed infrastructure. Most cloud implementations are managed by dedicated internal or external staff. This allows the organization’s IT staff to focus on other areas.	Difficult integration with internal IT system. Configuring the cloud services to integrate transparently with internal authentication and other internal services could be a daunting task.

SQL Data Services

- Cloud computing-based data management service
  - Provides relational data management to companies
  - Hosted data management and standard protocols
  - Standard protocols
  - Common programming interface
- Advantages
  - Reliable and scalable at a lower cost than in-house systems
  - High level of failure tolerance
  - Dynamic and automatic load balancing
  - Automated data backup and disaster recovery are included
  - Dynamic creation and allocation of processes and storage

References

- Build Your Own Database Driven Web Site Using PHP & MySQL by Kevin Yank
- Practical PHP 7, MySQL 8, and MariaDB Website Databases: A Simplified Approach to Developing Database-Driven Websites by Adrian W. West and Steve Prettyman
- PHP and MySQL Web Development: PHP MySQL Web Develo _5 (Developer's Library) by Luke Welling and Laura Thomson

Homework

- Textbook Chapter 15
- Review Questions 1 thru 9
- Project Final Report