CHAPTER 8: DATABASE APPLICATION DEVELOPMENT

Modern Database Management
11th Edition

Jeffrey A. Hoffer, V. Ramesh, Heikki Topi

OBJECTIVES

- Define terms
- Explain three components of client/server systems: presentation, processing, and storage
- Distinguish between two-tier and three-tier architectures
- Describe how to connect to databases in 2-tier systems using VB.NET and Java
- Describe key components and information flow in Web applications
- Describe how to connect to databases in 3-tier applications using JSP, PHP, and ASP .NET
- Explain the purpose of XML
- See how XQuery can be used to query XML documents
- Explain how XML fosters Web services and SOAs

CLIENT/SERVER ARCHITECTURES

- Networked computing model
- Processes distributed between clients and servers
- Client–Workstation (usually a PC) that requests and uses a service
- Server–Computer (PC/mini/mainframe) that provides a service
- For DBMS, server is a database server

APPLICATION LOGIC IN C/S SYSTEMS

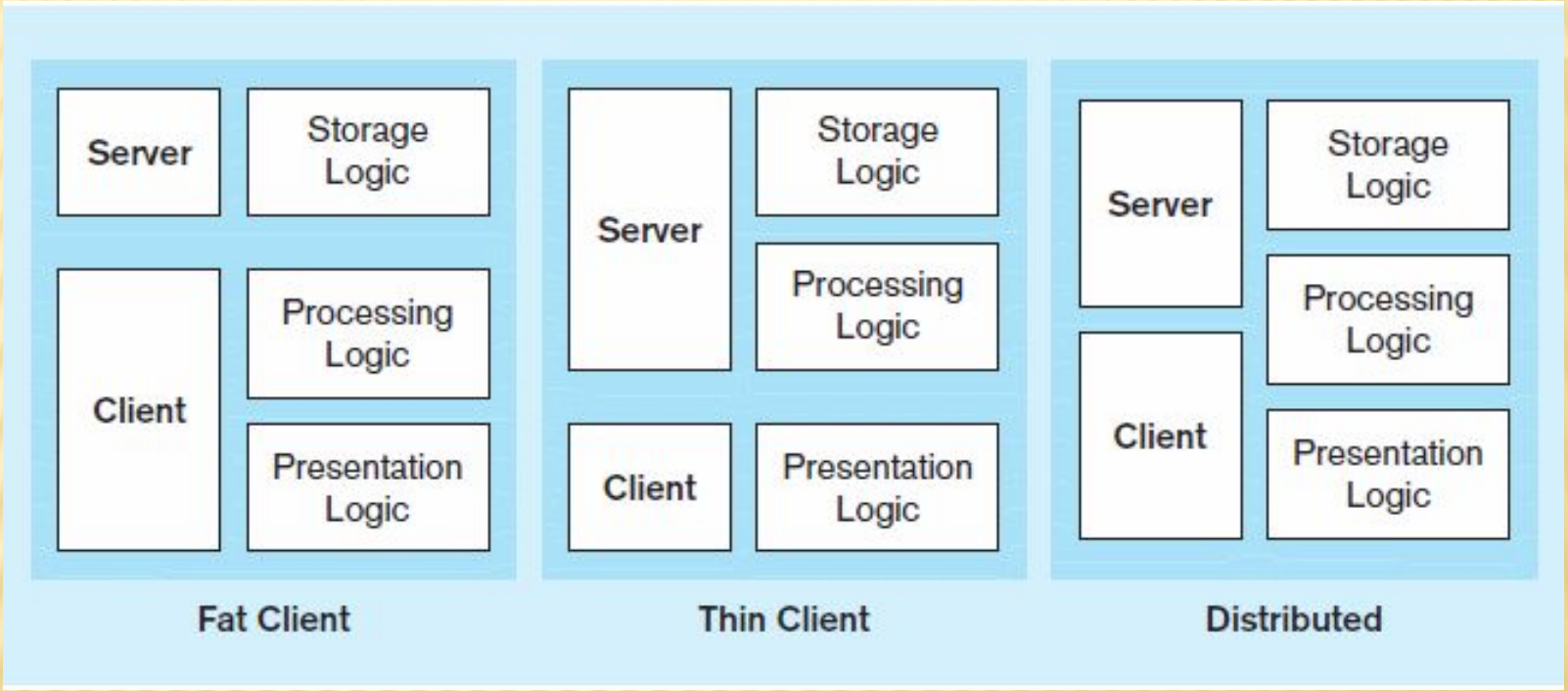
Presentation Logic ▪ Input–keyboard/mouse ▪ Output–monitor/printer	**GUI Interface**
Processing Logic ▪ I/O processing ▪ Business rules ▪ Data management	**Procedures, functions, programs**
Storage Logic ▪ Data storage/retrieval	**DBMS activities**

APPLICATION PARTITIONING

- Placing portions of the application code in different locations (client vs. server) after it is written
- Advantages
  - Improved performance
  - Improved interoperability
  - Balanced workloads

FIGURE 8-2 COMMON LOGIC DISTRIBUTIONS

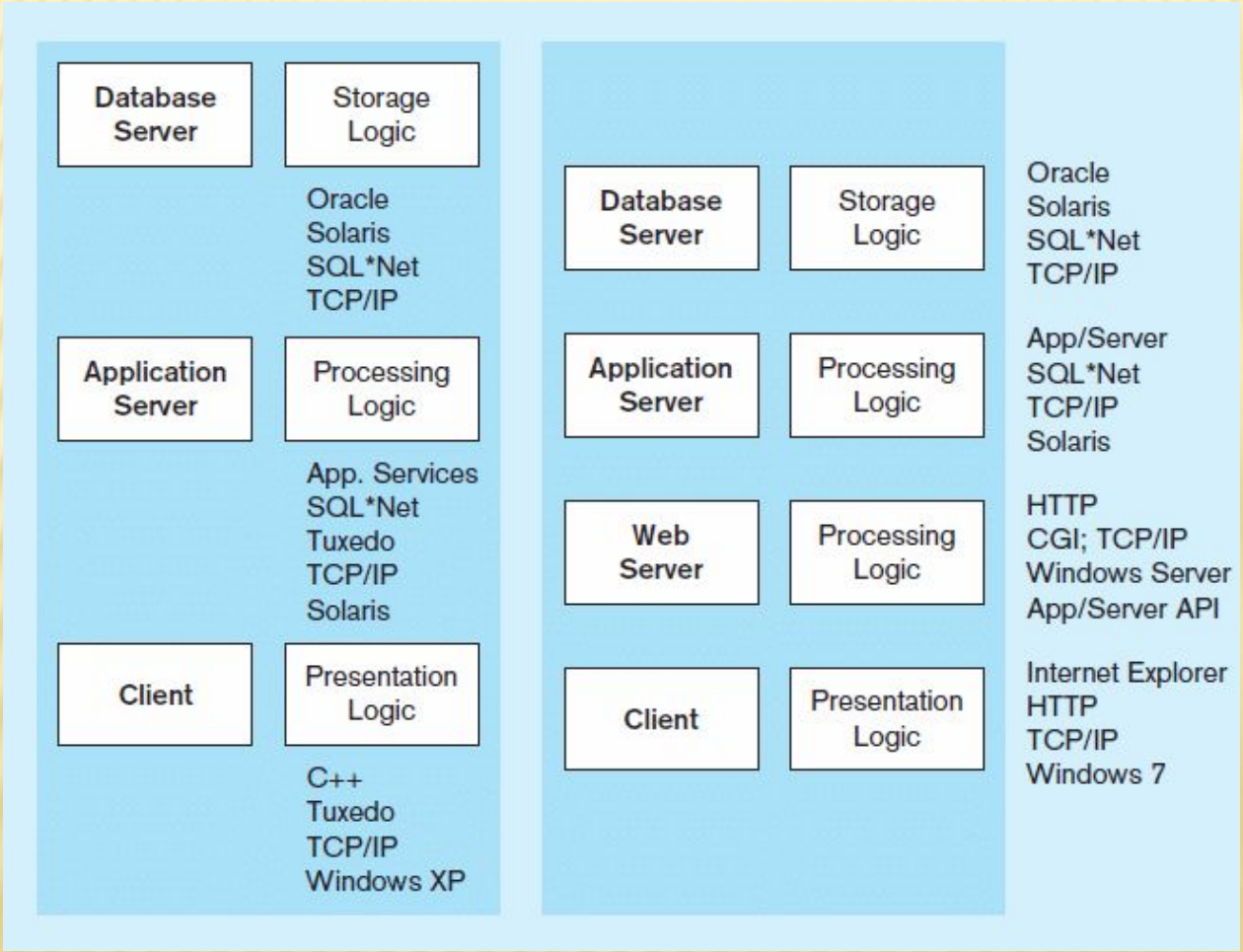
a) Two-tier client-server environments



Processing logic could be at client (fat client), server (thin client), or both (distributed environment).

FIGURE 8-2 COMMON LOGIC DISTRIBUTIONS

b) Three-tier and *n*-tier client-server environments



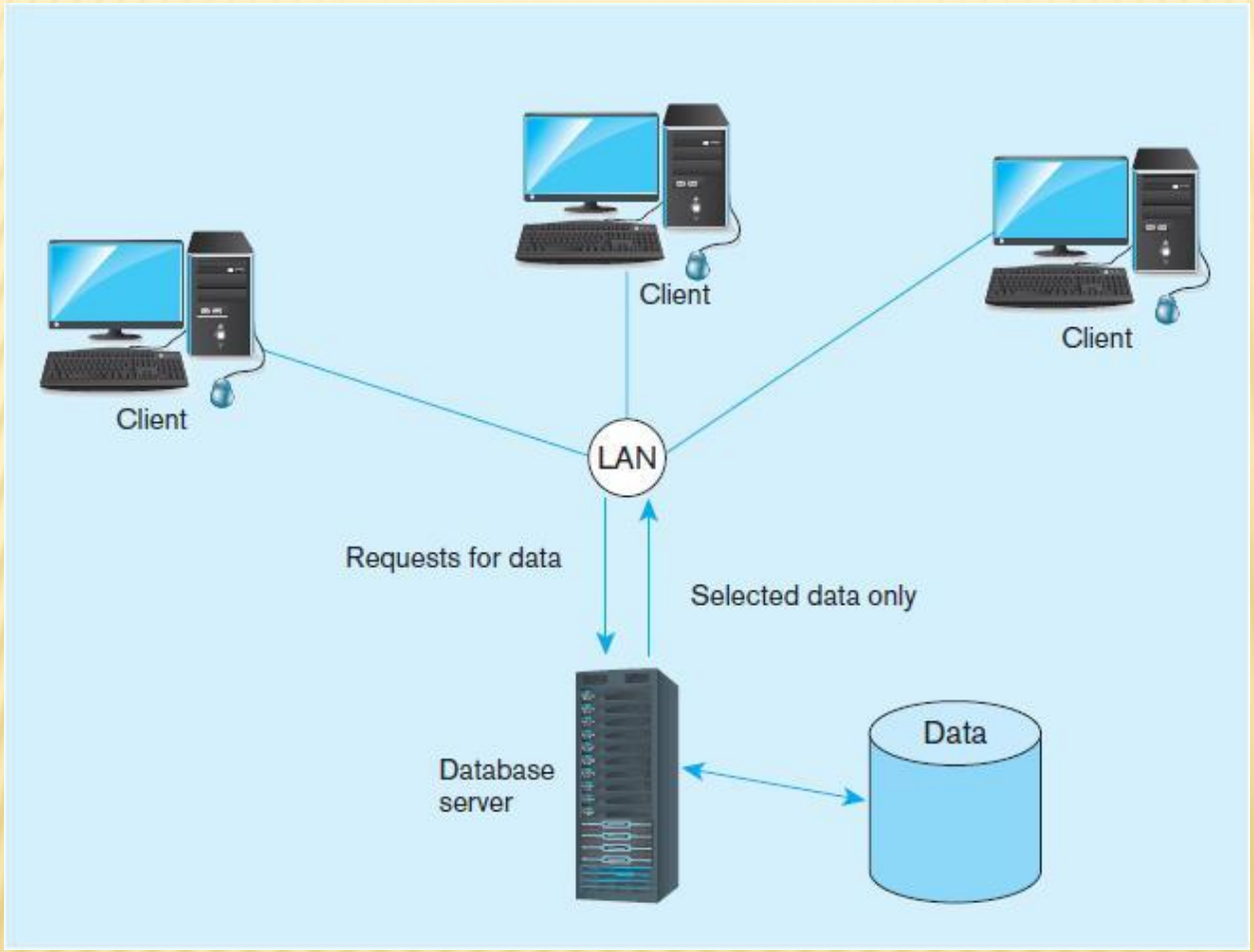
Processing logic will be at application server or Web server.

TWO-TIER DATABASE SERVER ARCHITECTURES

- Client workstation is responsible for
  - Presentation logic
  - Data processing logic
  - Business rules logic
- Server performs all data storage, access, and processing
  - Typically called a ***database server***
    - **DBMS is only on server**

Figure 8-3 Database server architecture (two-tier architecture)

Front-end programs



Back-end functions

CHARACTERISTICS OF TWO-TIER CLIENT/SERVER SYSTEMS

- Departmental in scope (few users)
- Not mission-critical
- Low transaction volumes
- Common programming languages:
  - Java, VB .NET, C#
- Interface database via middleware, APIs

MIDDLEWARE AND APIS

- **Middleware** – software that allows an application to interoperate with other software without requiring user to understand and code low-level operations
- **Application Program Interface (API)** – routines that an application uses to direct the performance of procedures by the computer's operating system
- Common database APIs – ODBC, ADO .NET, JDBC

STEPS FOR USING DATABASES VIA MIDDLEWARE APIS

1. Identify and register a database driver.
2. Open a connection to a database.
3. Execute a query against the database.
4. Process the results of the query.
5. Repeat steps 3–4 as necessary.
6. Close the connection to the

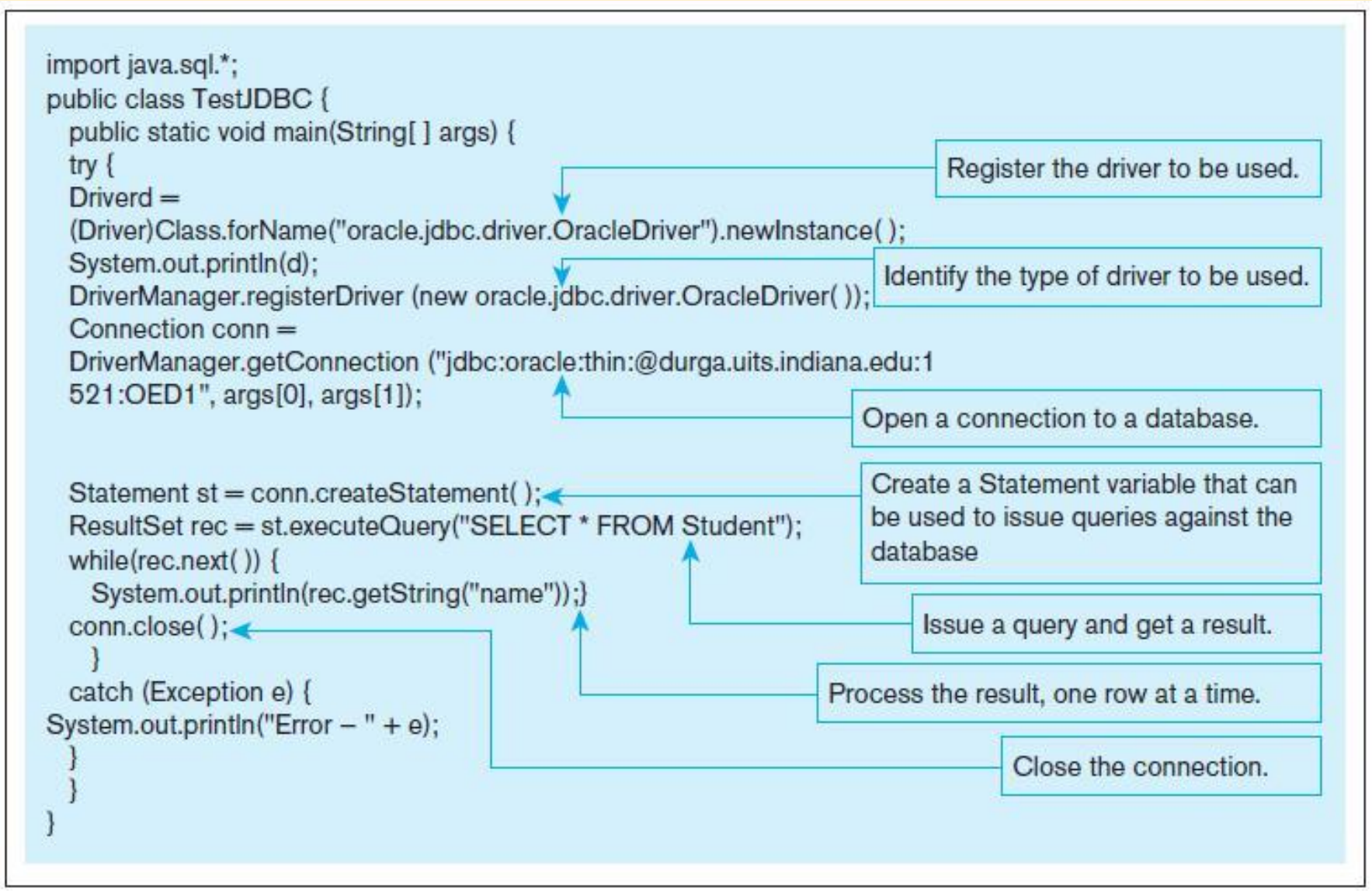


FIGURE 8-5 Database access from a Java program

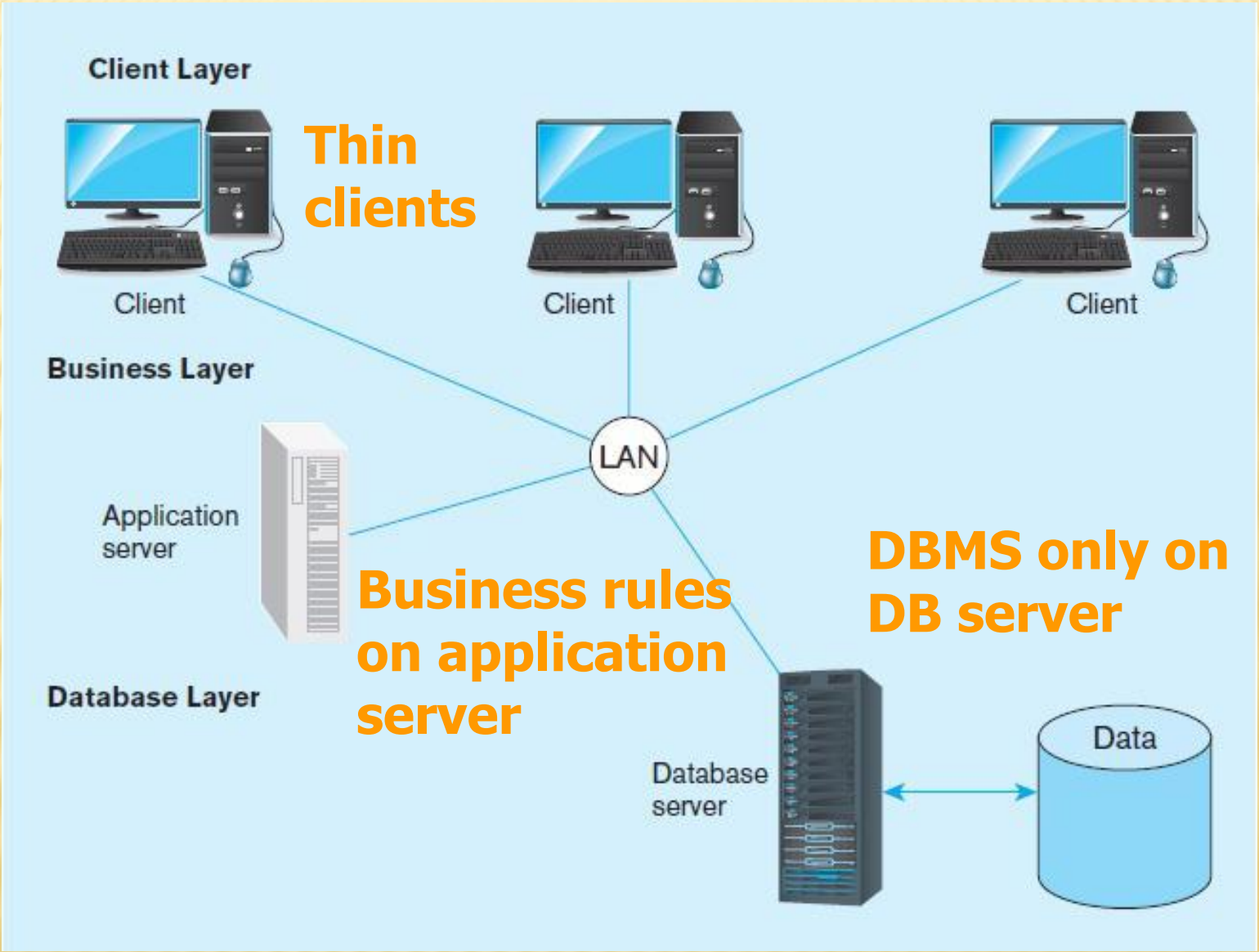
THREE-TIER ARCHITECTURES

Client	**GUI interface (I/O processing)**	***Browser***
Application server	**Business rules**	***Web Server***
Database server	**Data storage**	***DBMS***

Thin Client

- PC just for user interface and a little application processing. Limited or no data storage (sometimes no hard drive)

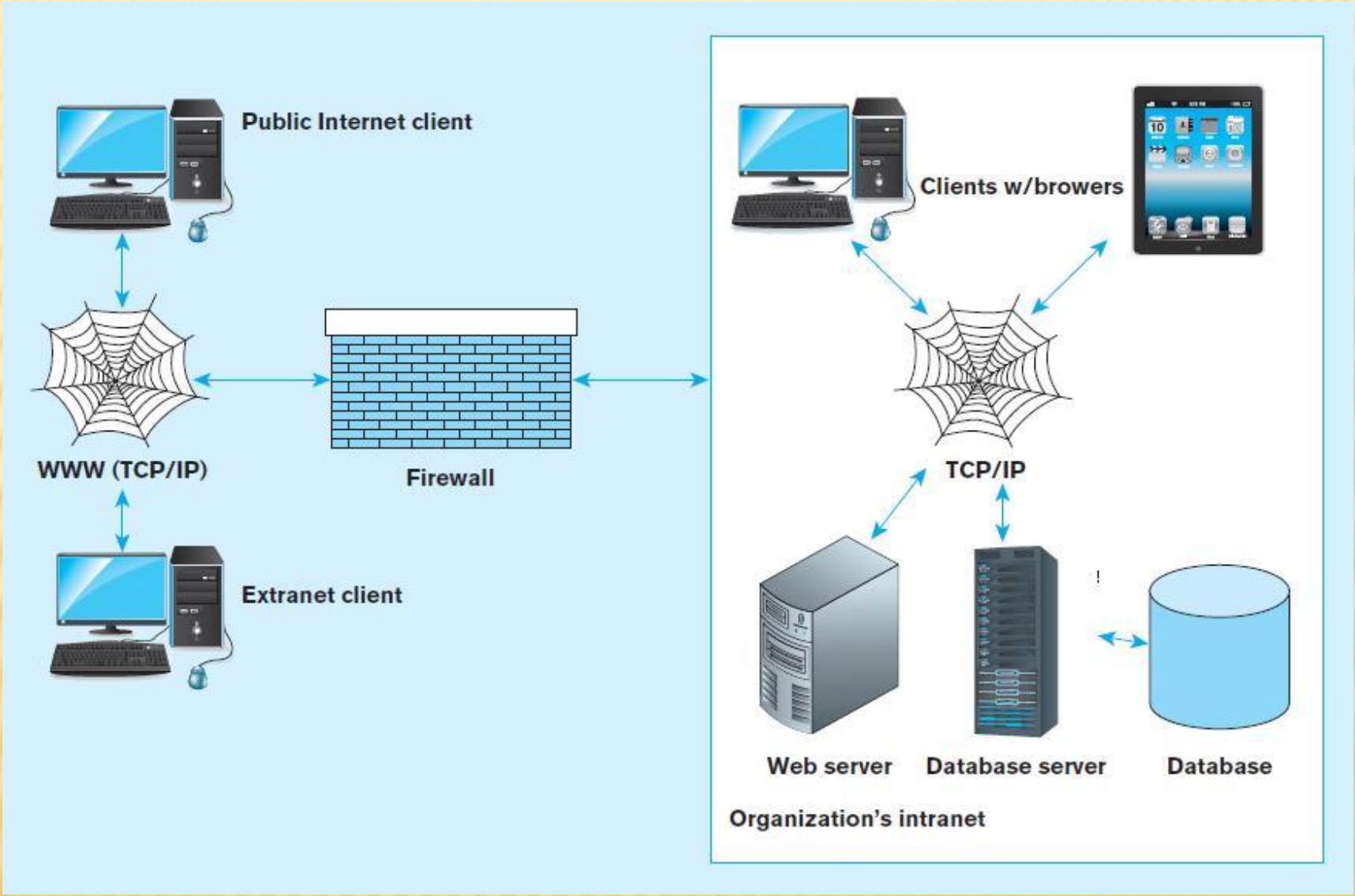
Figure 8-6a Generic three-tier architecture



THIN CLIENT

- An application where the client (PC) accessing the application primarily provides the user interfaces and some application processing, usually with no or limited local data storage.
- Usually, thin client application is a Web browser and the 3-tier architecture involves a Web application.

Figure 8-7 A database-enabled intranet/Internet environment



WEB APPLICATION COMPONENTS

- Database server – hosts the DBMS
  - e.g. Oracle, SQL Server, Informix, MS Access, MySql
- Web server – receives and responds to browser requests using HTTP protocol
  - e.g. Apache, Internet Information Services (IIS)
- Application server – software building blocks for creating dynamic web sites
  - e.g. MS ASP .NET framework, Java EE, ColdFusion, PHP
- Web browser – client program that sends web requests and receives web pages
  - e.g. Internet Explorer, Firefox, Safari, Google Chrome

LANGUAGES FOR CREATING WEB PAGES

- **Hypertext Markup Language (HTML)**
  - Markup language specifically for Web pages
- **Standard Generalized Markup Language (SGML)**
  - Markup language standard
- **Extensible Markup Language (XML)**
  - Markup language allowing customized tags
- **XHTML**
  - XML-compliant extension of HTML
- **JavaScript/VBScript**
  - Scripting languages that enable interactivity in HTML documents
- **Cascading Style Sheets (CSS)**
  - Control appearance of Web elements in an HML document
- **XSL and XSLT**
  - XMS style sheet and transformation to HTML

Standards and Web conventions established by **World Wide Web Consortium (W3C)**

PROCESSING IN 3-TIER APPLICATIONS

- Static page requests
  - .htm or .html requests handled by the Web server
- Dynamic page requests
  - .jsp, .aspx, and .php requests are routed to the application server
  - Server-side processing by JSP servlet, ASP .NET application, ColdFusion, or PHP
  - Database access via JDBC, ADO .NET, or other database middleware

Figure 8-9 Information flow in a three-tier architecture

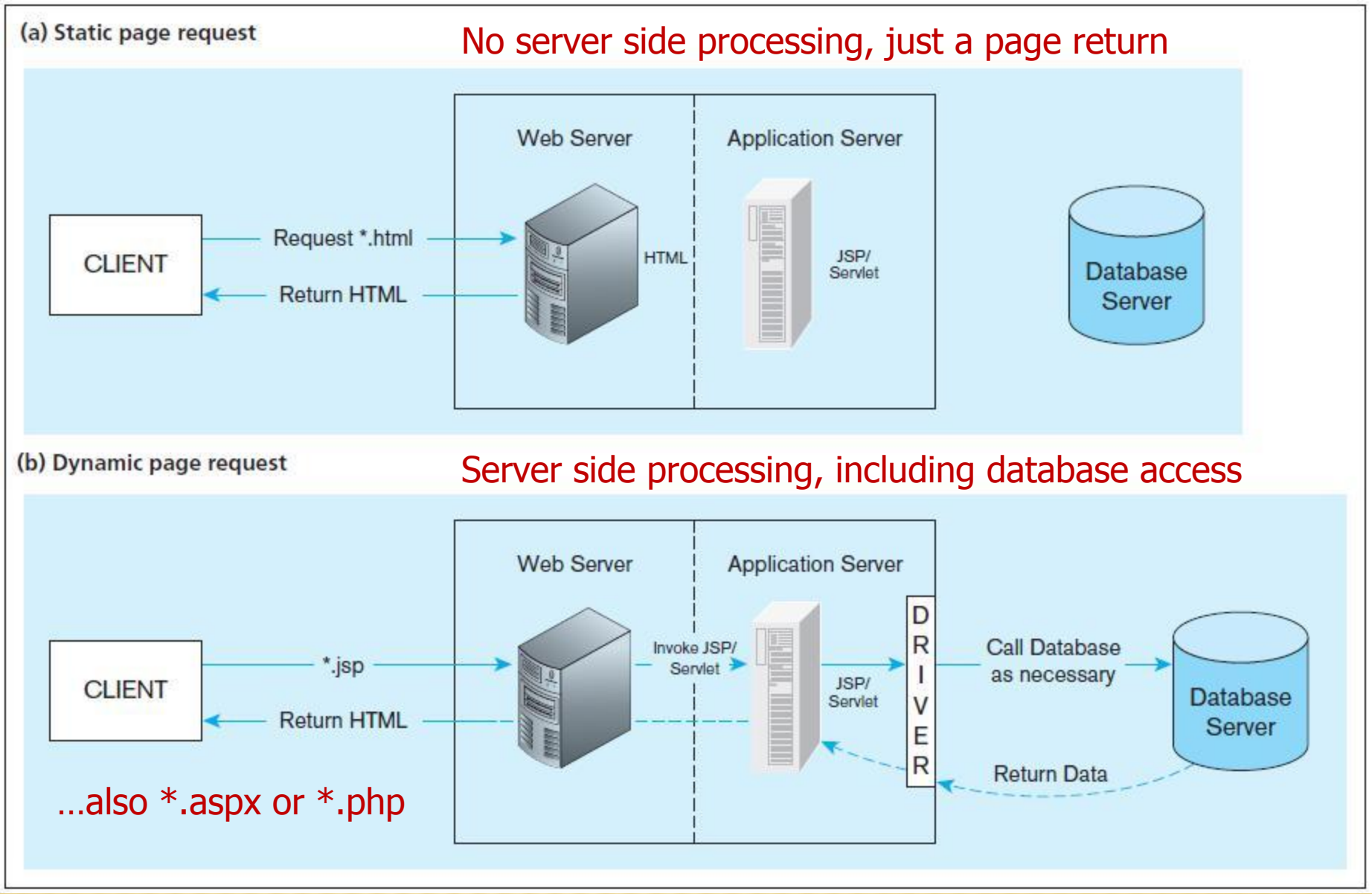


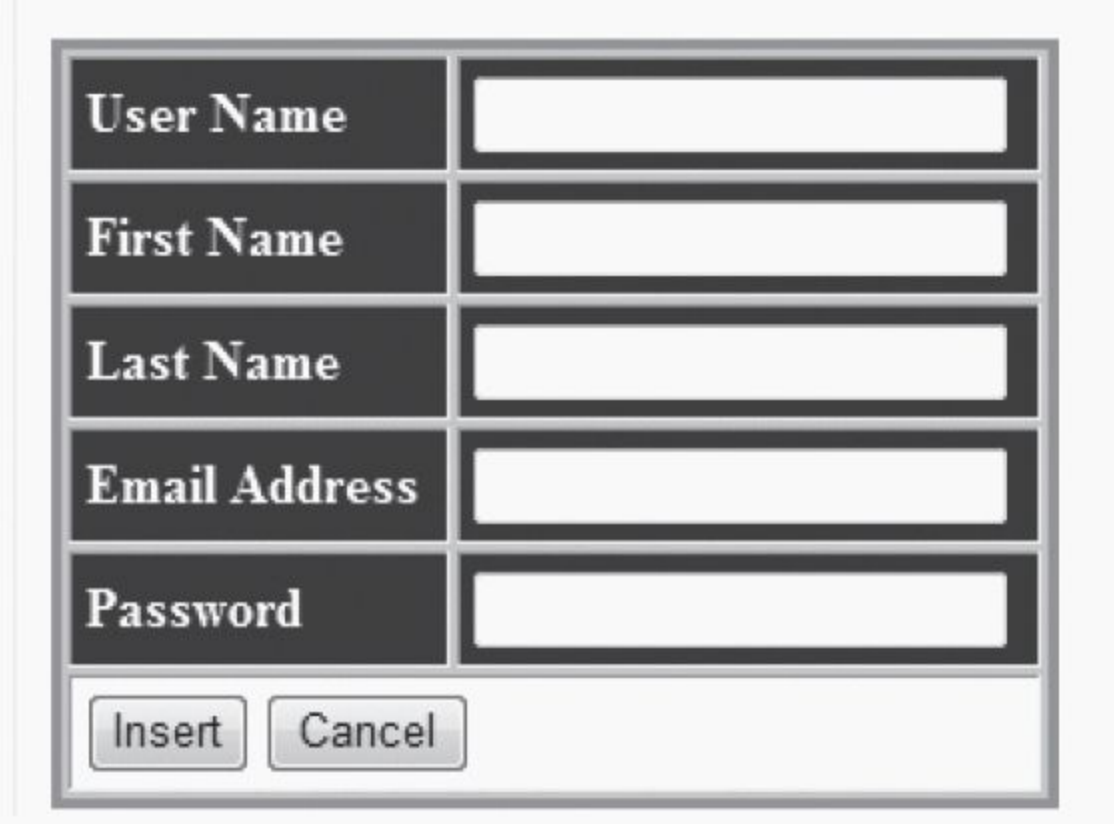
Figure 8-12 A registration page written in ASP .NET

a) Sample ASP .NET code for user registration

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="users.aspx.cs" Inherits="users" %>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Register</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:DetailsView ID="manageUsers" runat="server" DataSourceID="usersDataSource">
        <Fields>
                <asp:BoundField DataField="username" HeaderText="User Name" />
                <asp:BoundField DataField="first_name" HeaderText="First Name" />
                <asp:BoundField DataField="last_name" HeaderText="Last Name" />
                <asp:BoundField DataField="email" HeaderText="Email Address" />
                <asp:BoundField DataField="password" HeaderText="Password" />
                <asp:CommandField ShowInsertButton="True" ButtonType="Button" />
        </Fields>
        </asp:DetailsView>
<asp:SqlDataSource ID="usersDataSource" runat="server"
        ConnectionString="<%$ ConnectionStrings:StudentConnectionString %>"
        InsertCommand="INSERT INTO users(username, first_name, last_name, email, password,
        registration_date) VALUES (@username, @first_name, @last_name, @email, @password, GETDATE())"
        SelectCommand="SELECT [username], [first_name], [last_name], [email], [password] FROM [users]">
</asp:SqlDataSource>
</div>
</form>
</body>
</html>
```


Figure 8-12 A registration page written in ASP .NET

b) Form for the ASP .NET application



CONSIDERATIONS IN 3-TIER APPLICATIONS

- Stored procedures
  - Code logic embedded in DBMS
  - Improve performance, but proprietary
- Transactions
  - Involve many database updates
  - Either all must succeed, or none should occur
- Database connections
  - Maintaining an open connection is resource-intensive
  - Use of connection pooling

BENEFITS OF STORED PROCEDURES

- Performance improves for compiled SQL statements
- Reduced network traffic
- Improved security
- Improved data integrity
- Thinner clients

BENEFITS OF THREE-TIER ARCHITECTURES

- Scalability
- Technological flexibility
- Long-term cost reduction
- Better match of systems to business needs
- Improved customer service
- Competitive advantage
- Reduced risk

CLOUD COMPUTING

- A model for creating ubiquitous, convenient, on-demand access to network services
- Characteristics: on-demand, broad network access, resource pooling, rapid elasticity, measured service
- Types of cloud computing:
  - Infrastructure-as-a-service (IaaS)
  - Platform-as-a-service (PaaS)
  - Software-as-a-service (SaaS)

EXTENSIBLE MARKUP LANGUAGE (XML)

- A text-based markup language (like HTML)
  - Uses elements, tags, attributes
  - Includes document type declarations (DTDs), XML schemas, comments, and entity references
- Revolutionizes the way data are exchanged over the Internet
- Document Structure Declarations (DSD), XML Schema (XSD) and Relax NG replacing DTDs for validating XML document structure
- XSD – language for defining XML databases, recommended by the W3C

SAMPLE XML SCHEMA (XSD)

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema id="salespersonSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Salesperson" type="SalespersonType" />
  <xsd:complexType name="SalespersonType">
      <xsd:sequence>
          <xsd:elementname="SalespersonID"
                  type="xsd:integer"/>
          <xsd:elementname="SalespersonName"
                  type="xsd:string" />
          <xsd:element name="SalespersonTelephone"
                  type="PhoneNumberType">
          <xsd:element name="SalespersonFax"
                  type="PhoneNumber" minOccurs="0" />
          </xsd:element>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="PhoneNumberType">
      <xsd:restriction base="xsd:string">
          <xsd:length value="12" />
          <xsd:pattern value="\d{3}-\d{3}-\d{4}" />
      </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Schema is a record definition, analogous to the Create SQL statement, and therefore provides metadata.

SAMPLE XML DOCUMENT DATA

```
<?xml version="1.0" encoding="utf-8" ?>
<Salesperson xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="salespersonSchema.xsd">
    <SalespersonID>1</SalespersonID>
    <SalespersonName>Doug Henny</SalespersonName>
    <SalespersonTelephone>813-444-5555</SalespersonTelephone>
</Salesperson>
```

This XML data conforms to the XML schema of the previous slide, and involves elements and attributes defined in the schema.

This is analogous to a record in a database.

ANOTHER SAMPLE XML DOCUMENT

```
<?xml version = "1.0"/>
<furniturecompany>
    <product ID="1">
        <description>End Table</description>
        <finish>Cherry</finish>
        <standard price>175.00</standard price>
        <line>1</line>
    </product>
</furniturecompany>
```

STORING XML DOCUMENTS

- Storing as files introduces the same file processing problems stated in Ch 1
- Four common options:
  - Store XML data in a relational database by shredding the XML document
  - Store entire XML document in a large field (BLOB or CLOB)
  - Store the XML document using special XML columns
  - Store the XML document using a native XML database (non-relational)

RETRIEVING XML DOCUMENTS

- XPath – One of a set of XML technologies supporting XQuery development, locating data in XML documents
- XQuery – An XML transformation language that allows applications to query both relational databases and XML data
- Sample XQuery expression:

```
for $p in doc("PVFC.xml")/furniture company/product
where $p/standardprice>300.00
order by $p/description
return $p/description
```


DISPLAYING XML DATA

- Extensible Stylesheet Language Transformation (XSLT) – A language used to transform complex XML documents and also used to create HTML pages from XML documents

- XSLT can translate a single XML document into both standard HTML and WAP/WML for cell phones without the necessity for two different pages

FIGURE 8-15B – XSLT CODE

```
<?xml version = "1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
        <body>
        <h2>Product Listing</h2>
        <table border="1">
        <tr bgcolor="orange">
                <th>Description</th>
                <th>Finish</th>
                <th>Price</th>
        </tr>
        <xsl:for-each select="furniturecompany/product">
        <tr>
            <td><xsl:value-of select="description"/></td>
            <td><xsl:value-of select="finish"/></td>
            <td><xsl:value-of select="price"/></td>
        </tr>
        </xsl:for-each>
        </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

```
<furniture company>
    <product ID="1">
        <description>End Table</description>
        <finish>Cherry</finish>
        <standard price>175.00</standard price>
        <line>1</line>
    </product>
    <product ID="2">
        <description>Coffee Table</description>
        <finish>Natural Ash</finish>
        <standard price>200.00</standard price>
        <line>2</line>
    </product>
</furniture company>
```

Extracted from Figures 8-15a and 8-15c

When applied to the above XML data, the XSLT code from Figure 8-15b produces the display on the right.

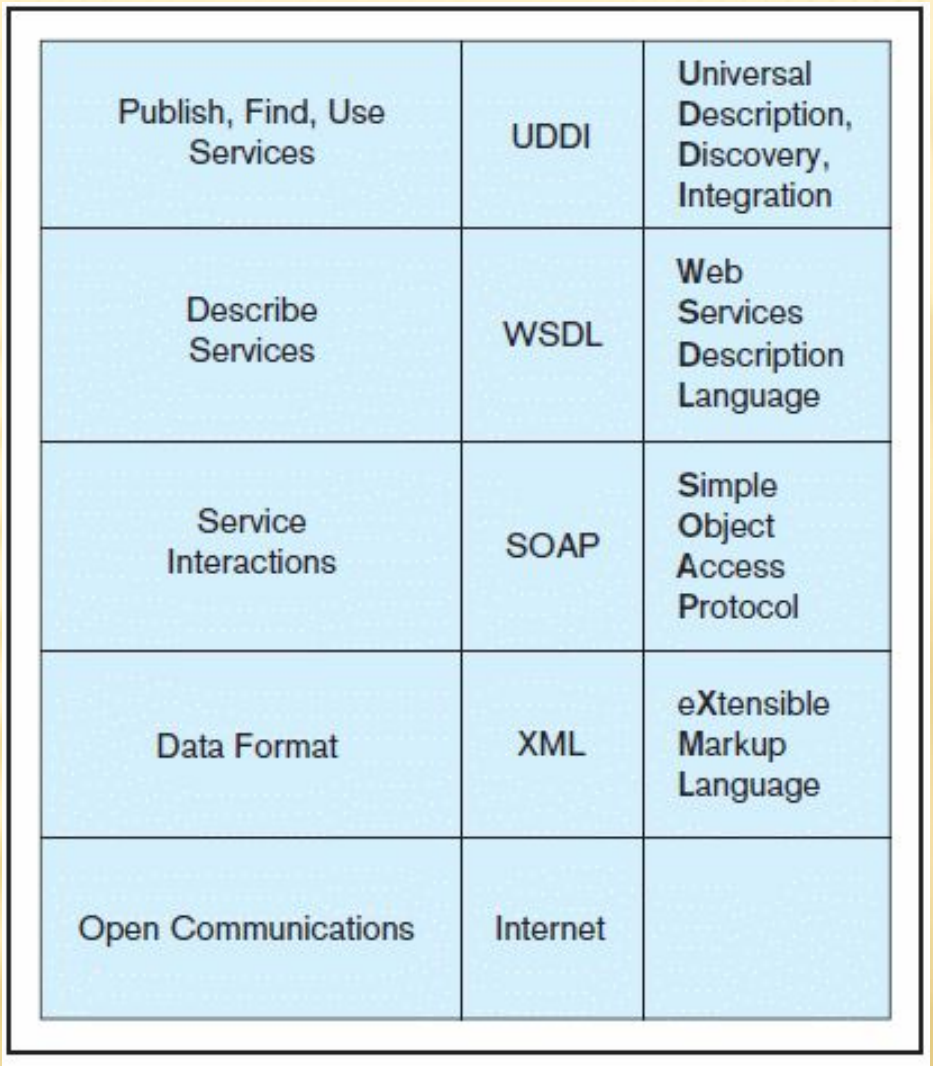
Product Listing

Description	Finish	Price
End Table	Cherry	175.00
Coffee Table	Natural Ash	200.00

XML AND WEB SERVICES

- Web Services – a set of emerging XML-based standards that define protocols for automatic communication between software programs over the Web
- Universal Description, Discovery, and Integration (UDDI) – standard for creating and distributing Web services
- Web Services Description Language (WSDL) – XML-based grammar for describing a Web Service and specifying its public interface
- Simple Object Access Protocol (SOAP) – XML-based communication protocol for sending messages between applications over the Internet

Figure 8-17 Web Services protocol stack



Publish, Find, Use Services	UDDI	Universal Description, Discovery, Integration
Describe Services	WSDL	Web Services Description Language
Service Interactions	SOAP	Simple Object Access Protocol
Data Format	XML	eXtensible Markup Language
Open Communications	Internet	

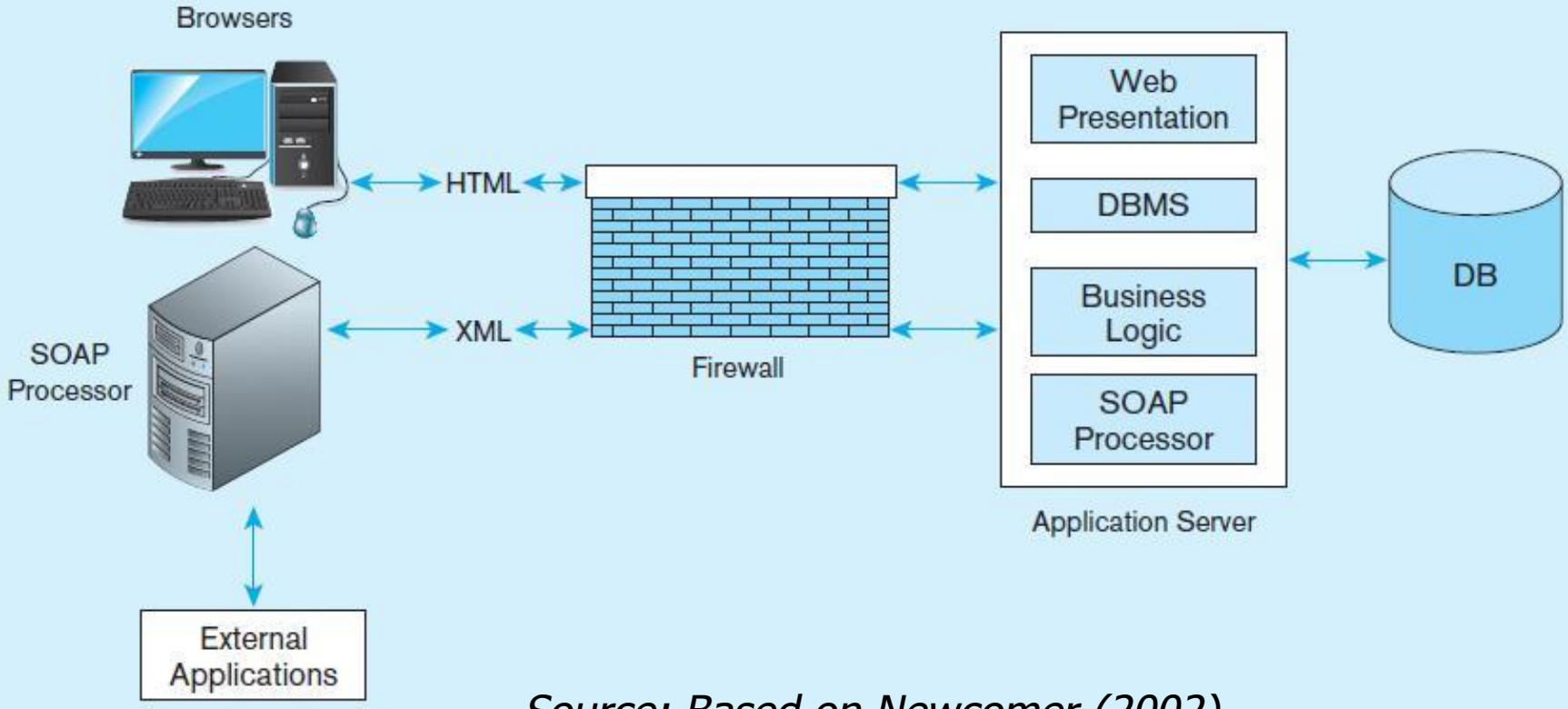
SOAP request sent from customer to supplier

```
<soap:Envelope xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/>
  <soap:Body>
    <getProductDetails xmlns=http://supplier.example.com/ws
      <productID>32879</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

SOAP response sent from supplier to customer

```
<soap:Envelope xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/>
  <soap:Body>
    <getProductDetailsResponse xmlns="suppliers.example.com/ws">
      <getProductDetailsResult>
       <productName>Dining Table</productName>
       <Finish>Natural Ash</Finish>
       <Price>800</Price>
       <inStock>True</inStock>
       </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```


Figure 8-18 Web services deployment



Source: Based on Newcomer (2002).

SERVICE ORIENTED ARCHITECTURE (SOA)

- A collection of services that communicate with each other, usually by passing data or coordinating a business activity
- A new paradigm for IT application development, based mostly on Web services
- Loosely coupled, highly interoperable components
- Leads to flexibility and shorter development time