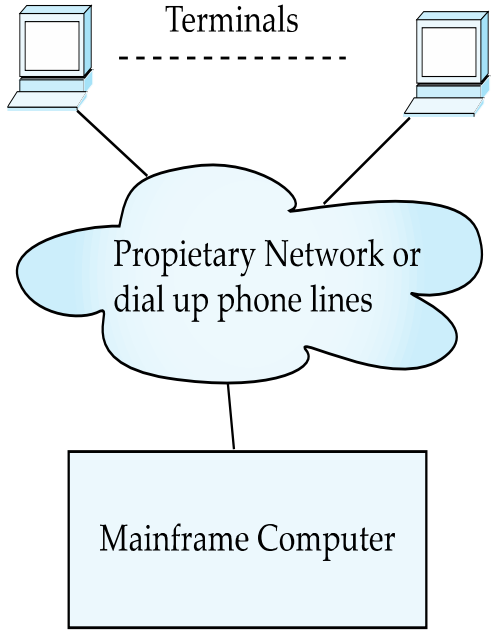
Application Programs and User Interfaces

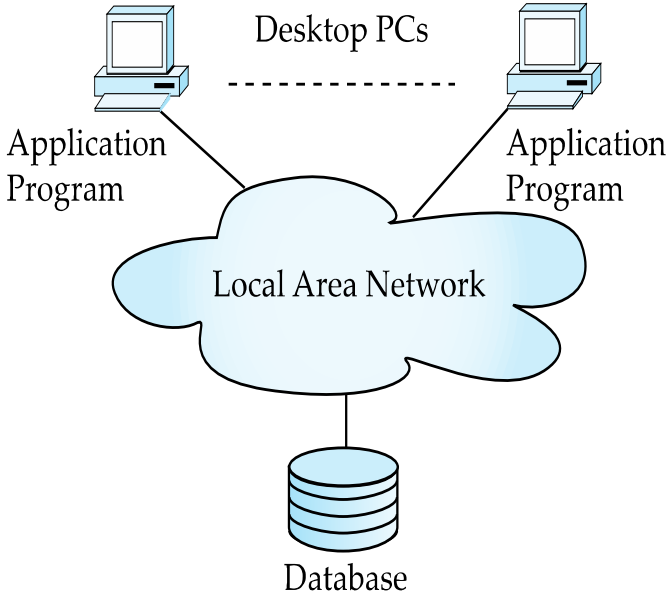
- Most database users do *not* use a query language like SQL
- An application program acts as the intermediary between users and the database
  - Applications split into
    - front-end
    - middle layer
    - backend
- Front-end: user interface
  - Forms
  - Many interfaces are Web-based

Application Architecture Evolution

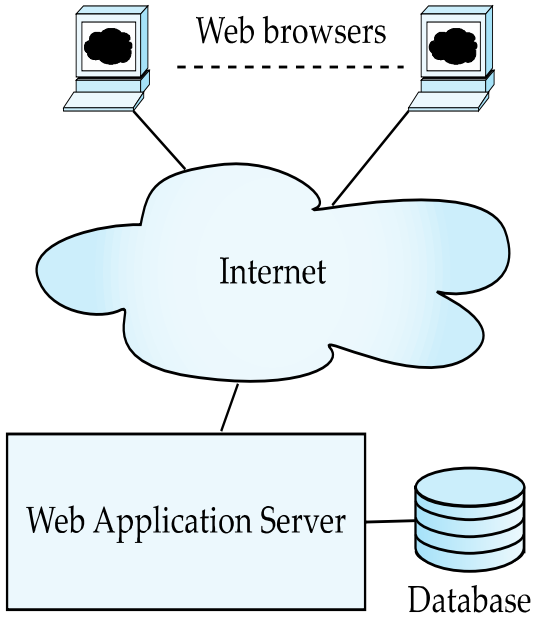
- Three distinct era's of application architecture



(a) Mainframe Era



(b) Personal Computer Era

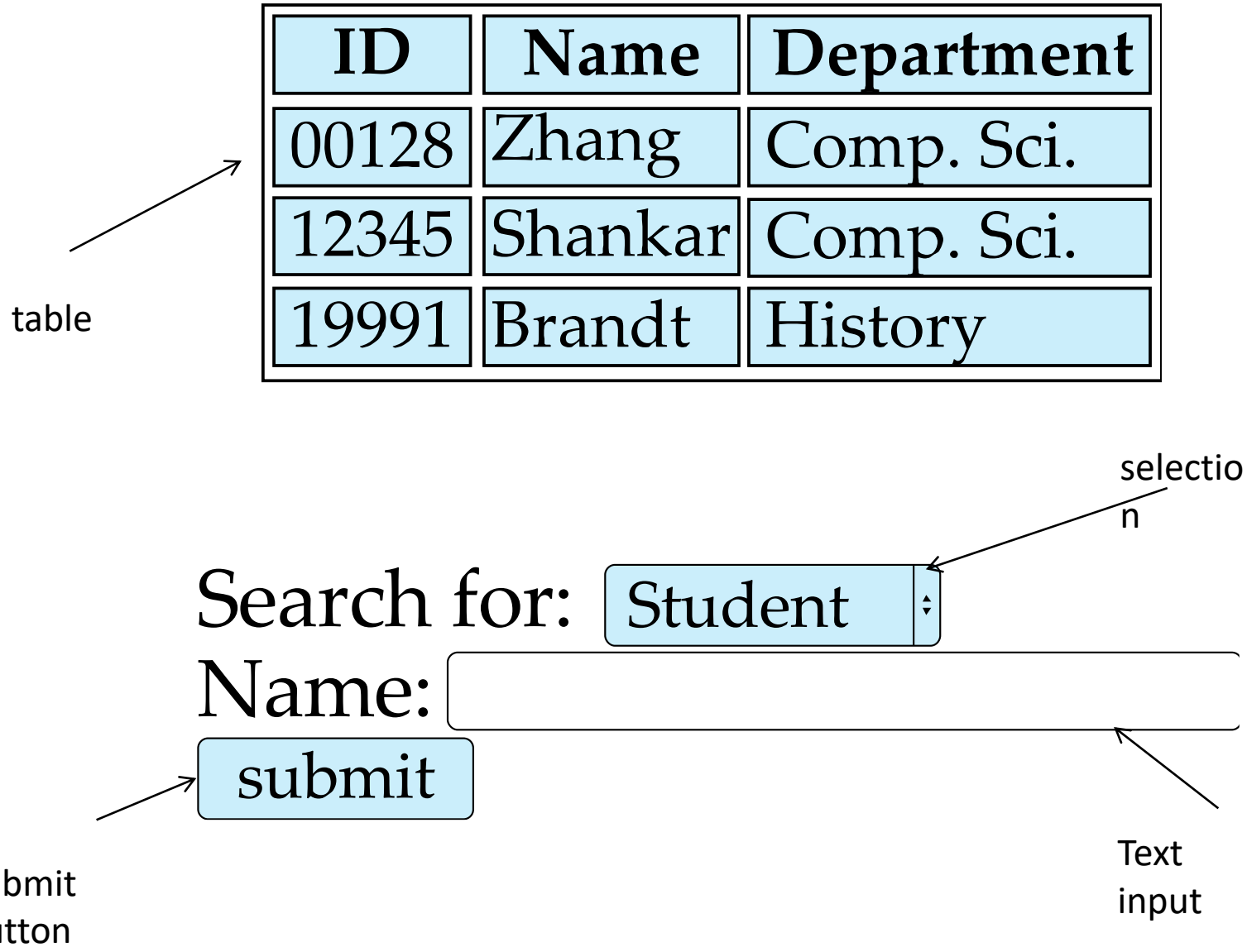


(c) Web era

Web Interface

- Web browsers have become the de-facto standard user interface to databases
  - Enable large numbers of users to access databases from anywhere
  - Avoid the need for downloading/installing specialized code, while providing a good graphical user interface
    - Javascript, Flash and other scripting languages run in browser
  - Examples: banks, airline and rental car reservations, university course registration and grading, an so on.

Display of Sample HTML Source



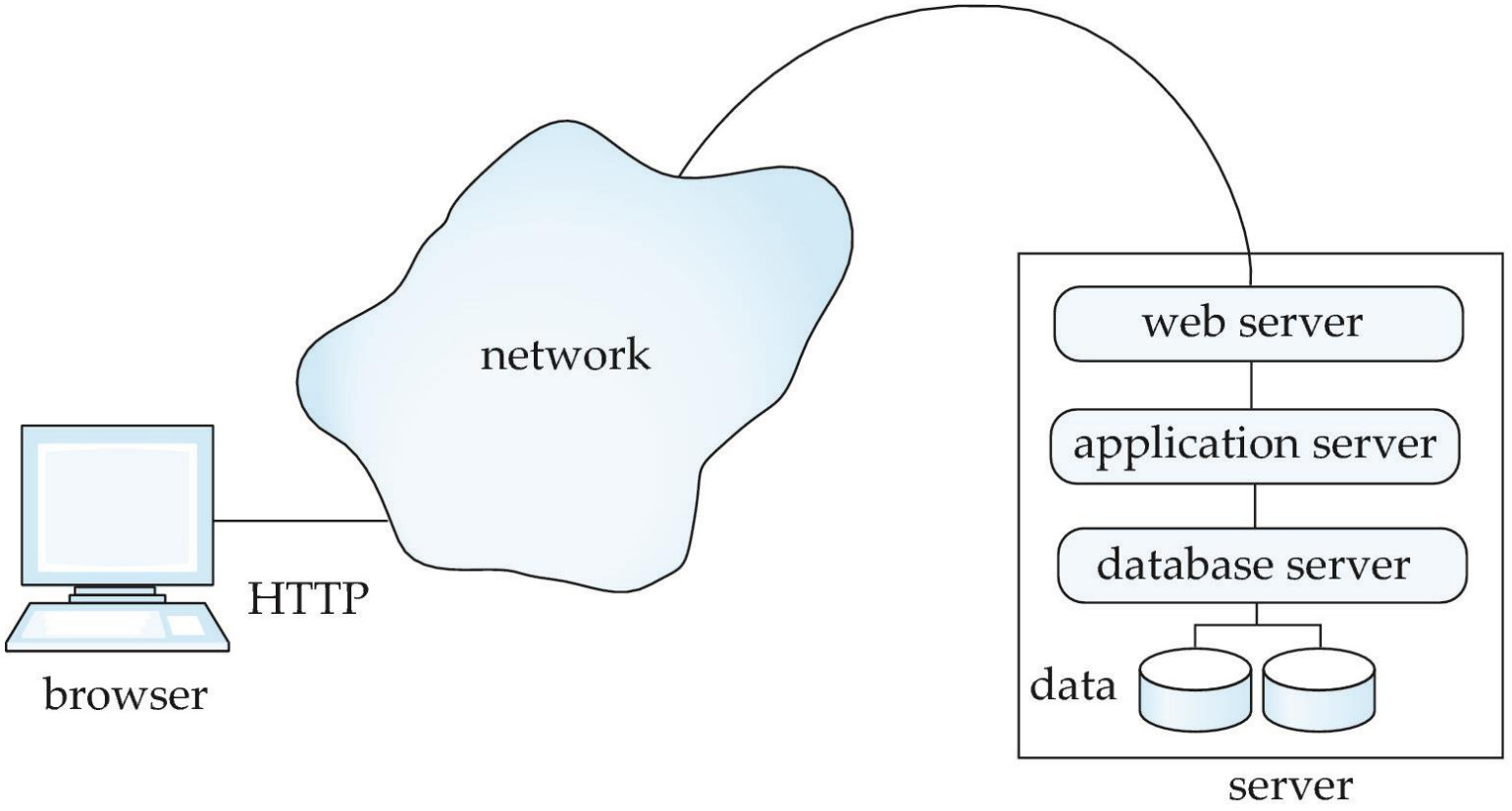
Sample HTML Source Text

```
<html>
<body>
 <table border>
   <tr> <th>ID</th> <th>Name</th> <th>Department</th> </tr>
   <tr> <td>00128</td> <td>Zhang</td> <td>Comp. Sci.</td> </tr>
   ….
 </table>
 <form action="PersonQuery" method=get>
  Search for:
    <select name="persontype">
       <option value="student" selected>Student </option>
       <option value="instructor"> Instructor </option>
    </select> <br>
  Name: <input type=text size=20 name="name">
  <input type=submit value="submit">
 </form>
</body> </html>
```

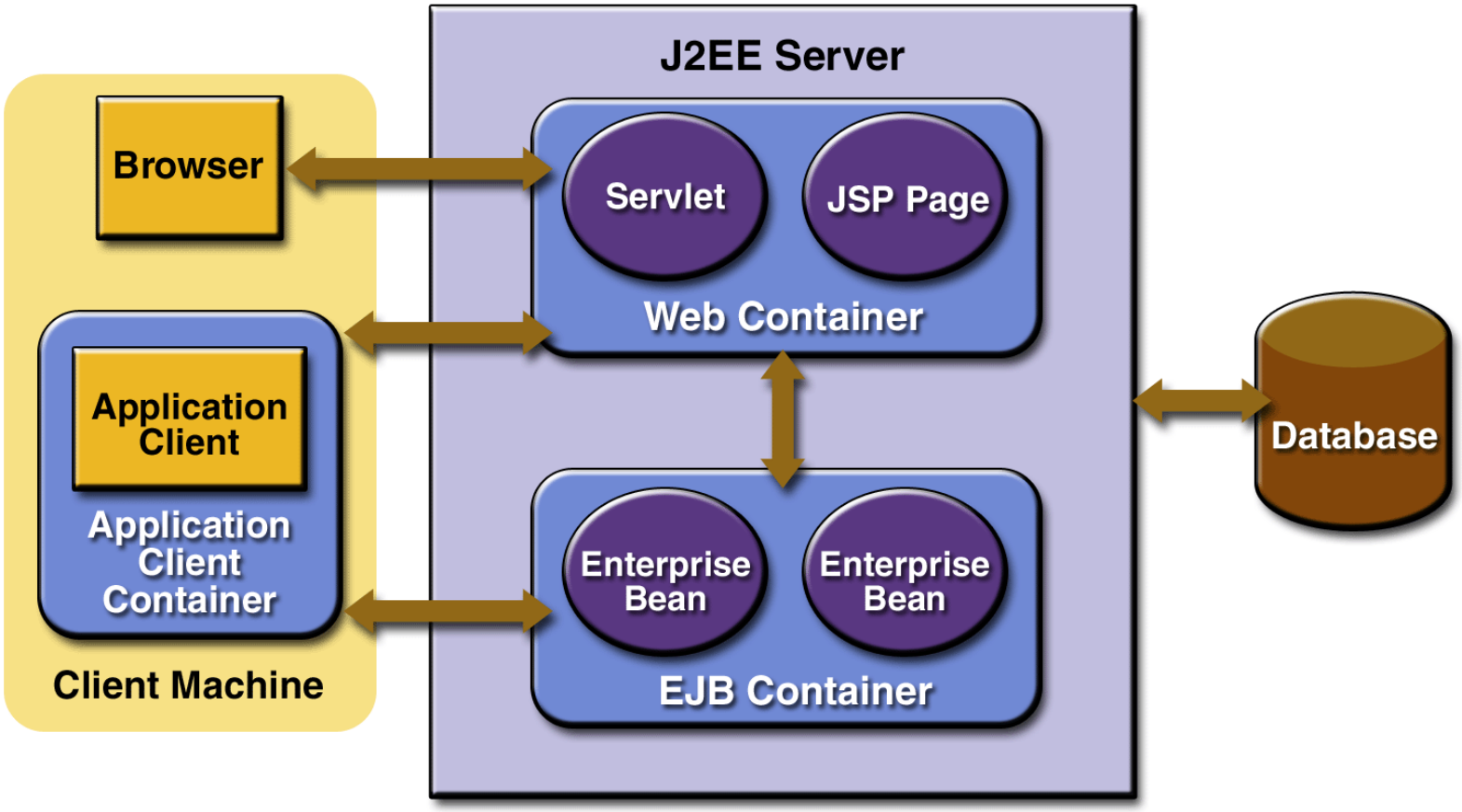
Web Servers

- A Web server can easily serve as a front end to a variety of information services.
- The document name in a URL may identify an executable program, that, when run, generates a HTML document.
  - HTTP server executes the program, and sends back the HTML document that is generated.
  - The Web client can pass extra arguments
- To install a new service on the Web, one simply needs to create and install an executable that provides that service.

Three-Layer Web Architecture



J2EE Applications



http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Oveview3.html

Servlets

- Java Servlet specification defines an API for communication between the Web/application server and application program running in the server
  - E.g., methods to get parameter values from Web forms, and to send HTML text back to client
- Application program (also called a servlet) is loaded into the server
  - Each request spawns a new thread in the server
    - thread is closed once the request is serviced
  - Programmer creates a class that inherits from HttpServlet
    - And overrides methods doGet, doPost, …
  - Mapping from servlet name to the servlet class is done in a file web.xml
    - Done automatically by most IDEs when you create a Servlet using the IDE

Example Servlet Code

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PersonQueryServlet extends HttpServlet {
  public void doGet (HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HEAD><TITLE> Query Result</TITLE></HEAD>");
    out.println("<BODY>");
      ….. BODY OF SERVLET (next slide) …
    out.println("</BODY>");
    out.close();
  }
}
```

Example Servlet Code

```
String persontype = request.getParameter("persontype");
String number = request.getParameter("name");
if(persontype.equals("student")) {
    ... code to find students with the specified name ...
    ... using JDBC to communicate with the database ..
    out.println("<table BORDER COLS=3>");
    out.println(" <tr> <td>ID</td> <td>Name: </td>" + " <td>Department</td> </tr>");
    for(... each result ...){
        ... retrieve ID, name and dept name
        ... into variables ID, name and deptname
        out.println("<tr> <td>" + ID + "</td>" + "<td>" + name + "</td>" + "<td>" + deptname
              + "</td></tr>");
    };
    out.println("</table>");
}
else {
    ... as above, but for instructors ...
}
```

Servlet Sessions

- Servlet API supports handling of sessions
  - Using cookie
- Store/retrieve attribute value pairs for a particular session
  - session.setAttribute(“userid”, userid)
  - session.getAttribute(“userid”)

Servlet Support

- Servlets run inside application servers such as
  - Apache Tomcat, Glassfish, Jboss
  - IBM WebSphere and Oracle Application Servers
- Application servers support
  - deployment and monitoring of servlets

Server-Side Scripting

- Server-side scripting simplifies the task of connecting a database to the Web
  - Define an HTML document with embedded executable code/SQL queries.
  - Input values from HTML forms can be used directly in the embedded code/SQL queries.
  - When the document is requested, the Web server executes the embedded code/SQL queries to generate the actual HTML document.
- Numerous server-side scripting languages
  - JSP, PHP
  - General purpose scripting languages: VBScript, Perl, Python

Java Server Pages (JSP)

- A JSP page with embedded Java code

```
<html>
<head> <title> Hello </title> </head>
<body>
<% if (request.getParameter("name") == null)
{ out.println("Hello World"); }
else { out.println("Hello, " + request.getParameter("name")); }
%>
</body>
</html>
```

- JSP is compiled into Java + Servlets
- JSP allows new tags to be defined, in tag libraries
  - such tags are like library functions, can are used for example to build rich user interfaces such as paginated display of large datasets

Client Side Scripting

- Browsers can fetch certain scripts (client-side scripts) along with documents, and execute them in "safe mode" at the client site
  - Javascript
  - Adobe Flash and Shockwave for animation/games
- Client-side scripts/programs allow documents to be active
  - ensure that values entered by users satisfy some correctness checks
  - Executing programs at the client site speeds up interaction by avoiding many round trips to server

A complete view of DB application