Desarrollo de aplicaciones de BD

Bases de datos II: Tema 2

Estructura del tema

1. Introducción.
2. Desarrollo de aplicaciones con SQL en lenguajes de propósito general.
3. Tecnologías web y acceso a bases de datos desde la web
4. Herramientas específicas de los SGBD.

1. Introducción

- La mayor parte de usuarios de una base de datos no usa un lenguaje de consultas como SQL, sino:
  - Formularios
  - Interfaces gráficos
  - Generadores de informes
  - Herramientas de análisis de datos
- Muchos interfaces son interfaces web.
- Los usuarios no trabajan directamente con la base de datos, sino con aplicaciones.

1. Introducción

- Una aplicación es un programa informático que realiza una tarea.
  - Una aplicación de BD es un programa que utiliza los datos de un sistema de gestión de bases datos.
- La mayoría de las aplicaciones de base de datos presentan datos de forma útil o facilitan la introducción o actualización de datos en la BD.

1. Introducción

- Las aplicaciones de bases de datos trabajarán interactuando con el SGBD
- Hay varias formas de desarrollar aplicaciones de BD:
  - desde lenguajes de alto nivel
  - hasta herramientas específicas para este propósito.

2. Desarrollo de aplicaciones con lenguajes de propósito general

- Las consultas (e instrucciones) SQL pueden ser llamadas desde dentro de un programa escrito en un lenguaje de propósito general (lenguaje ***host***)
  - Las sentencias SQL pueden emplear variables “anfitrionas” (incluyendo variables especiales para devolver status)
  - Deberán incluirse sentencias específicas para conectar con la base de datos adecuada

2. Desarrollo de aplicaciones con lenguajes de propósito general

- Como lenguaje de propósito general nos referimos a lenguajes tipo Pascal, C, C++, Java, (o otros no específicos de BD) ...
- Posibilidades para atacar una BD desde uno de estos lenguajes:
  - SQL embebido en uno de estos lenguajes
  - ODBC
  - JDBC
  - otros...

2. Desarrollo de aplicaciones con lenguajes de propósito general

Adaptación entre SQL declarativo y otros lenguajes

- Las consultas SQL, como lenguaje declarativo, pueden devolver una relación
  - La relación es un conjunto de registros, sin límite a priori en el número de ellos
  - Los lenguajes procedimentales no contienen habitualmente una estructura de datos apropiada para ello
- SQL proporciona un mecanismo de adaptación conocido como ***cursor*** .

2. Desarrollo de aplicaciones con lenguajes de propósito general

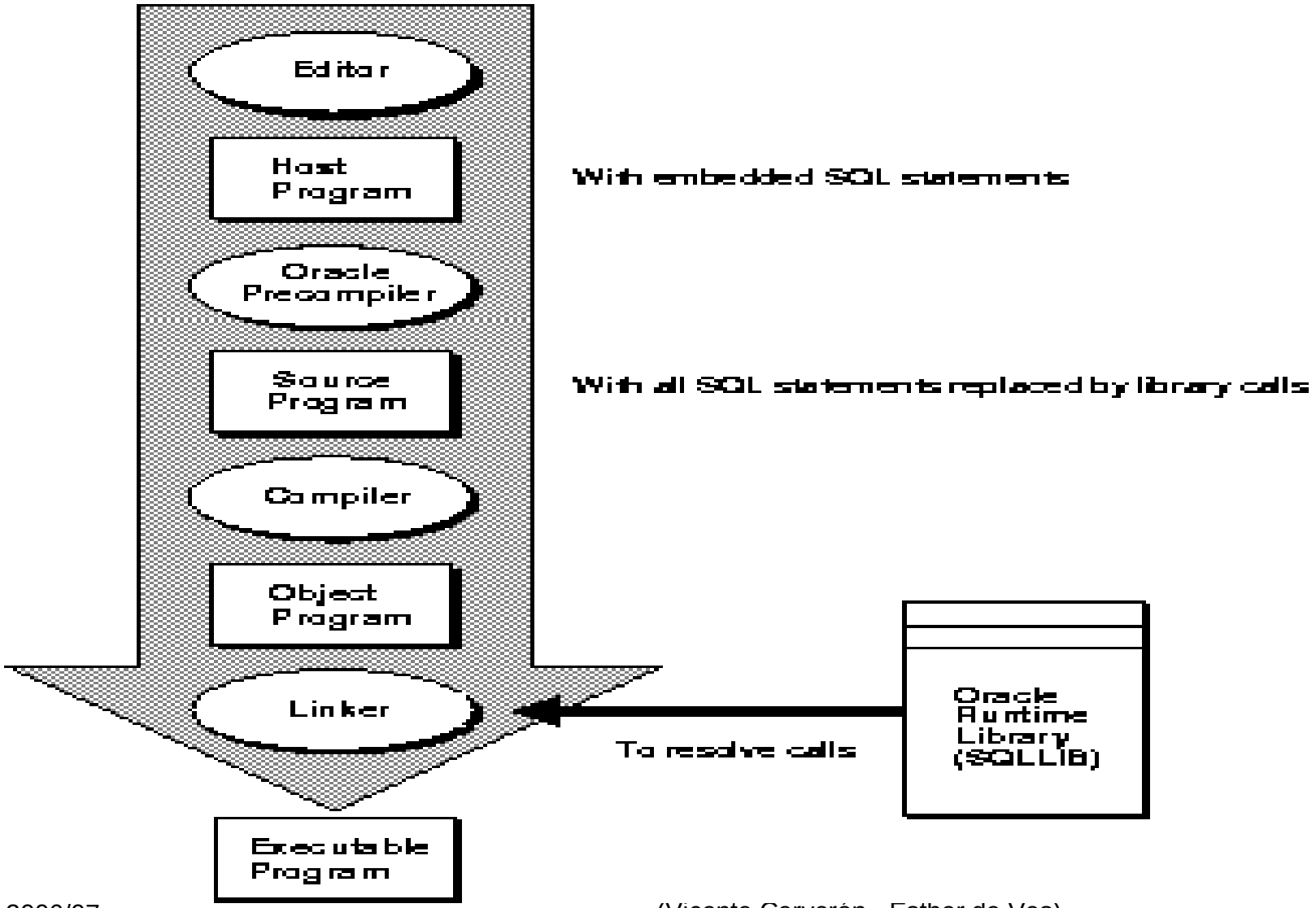
Veremos

- SQL embebido
- ODBC
- JBDC

2.1 SQL embebido

- Está pensado para intercalar sentencias SQL en un lenguaje de alto nivel.
- Se necesita un pre-compilador *(en Oracle Pro*C)*
  - Traduce las sentencias SQL convirtiéndolas en llamadas a unas librerías específicas.
  - Genera un programa fuente modificado que se puede compilar, enlazar y ejecutar.
  - Permite incluir sentencias SQL y bloques PL/SQL.
    - *Oracle dispone de preprocesador para Cobol, C y C++.*

2.1 SQL embebido



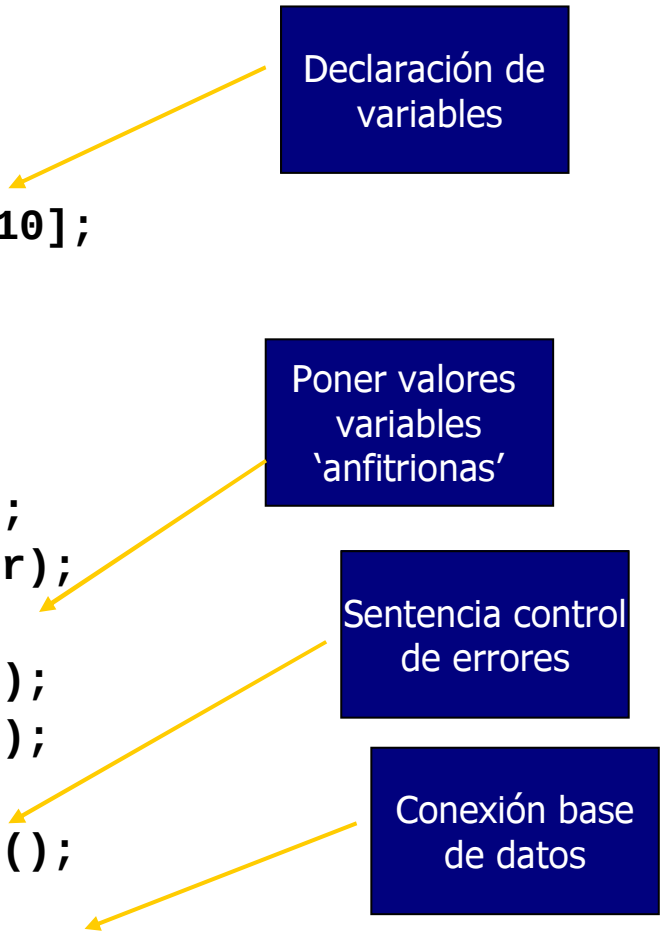
2.1 SQL embebido

- Construcciones del lenguaje
  - Conectar a una base de datos
    - EXEC SQL CONNECT
  - Declarar variables
    - EXEC SQL BEGIN/END DECLARE SECTION
  - Ejecutar sentencias SQL
    - EXEC SQL *Sentencia*

2.1 SQL embebido. Ejemplo

- Un programa ejemplo en C
  - Se conecta a la BD Oracle: scott/tiger
  - Realiza una consulta (que devuelve únicamente un registro).
  - Muestra el resultado por pantalla.
    - Contiene una función para controlar los errores devueltos por Oracle.

```
#include <iostream>
#include <string.h>

main()
{
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR nom_usuario[30],contraseña[10];
    VARCHAR vnombre[20],vapellido[30];
    float vsalario;
EXEC SQL END DECLARE SECTION;

strcpy((char *)nom_usuario.arr,"scott");
nom_usuario.len = strlen(nom_usuario.arr);

strcpy((char *)contrasenya.arr, "tiger");
contrasenya.len=strlen(constrasenya.arr);

EXEC SQL WHENEVER SQLERROR DO sql_error();

EXEC SQL CONNECT :nom_usuario IDENTIFIED BY :contrasenya
```



```
EXEC SQL SELECT nombre, apellido, salario INTO :vnombre,
:vapellido, :vsalario FROM EMPLEADO WHERE salario =
(select max(salario) from empleado);


vnombre.arr[vnombre.len]='\0';
vapellido.arr[vapellido.len]='\0';


cout << "Nombre, apellidos, salario:\n";
 cout << vnombre.arr << ' ' << vapellido.arr << ' ' <<
     vsalario;
 }


int sql_error()
{
    EXEC_SQL WHENEVER SQLERROR CONTINUE;
    cout << "Error detectado" << endl;
}
```

Realizar consulta

Imprime resultados

Tratar errores

2.1 SQL embebido

- Información más completa sobre SQL embebido en

http://informatica.uv.es/guia/asignatu/BD2/sqlembebido.pdf

- Notas:
  - se puede utilizar tanto en Linux como en Windows.
  - Se puede integrar dentro de herramientas como Visual C++ o Builder C++.

2.2 ODBC
2.3 JDBC

- En lugar de modificar los compiladores o emplear precompiladores, pueden utilizarse nuevas librerías con llamadas a la base de datos
  (API = *Application Programming Interface*)

- Mientras que las aplicaciones en SQL embebido son independientes del SGBD en cuanto al código fuente (deben ser recompiladas para cada uno), las aplicaciones con ODBC o JDBC son independientes del SGBD en cuanto al ejecutable.

- ODBC y JDBC pretenden estandarización y portabilidad

2.2 **ODBC**

- ODBC (*Open Database Connectivity*) permite acceder a las BD (a través de los SGBD) desde cualquier aplicación
  - Cada SGBD tiene su propia forma de recibir consultas SQL y su propio modo de devolver el resultado.
- La idea es que se pueda escribir código independiente de cuál sea el SGBD.
- ODBC inserta una capa intermedia llamada **manejador de Bases de Datos**, entre la aplicación y el SGBD
  - el propósito de esta capa es traducir las consultas de datos de la aplicación en órdenes que el SGBD entienda.
  - tanto la aplicación como el SGBD deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir sentencias ODBC y el SGBD debe ser capaz de responder a ellas.

2.2 **ODBC**

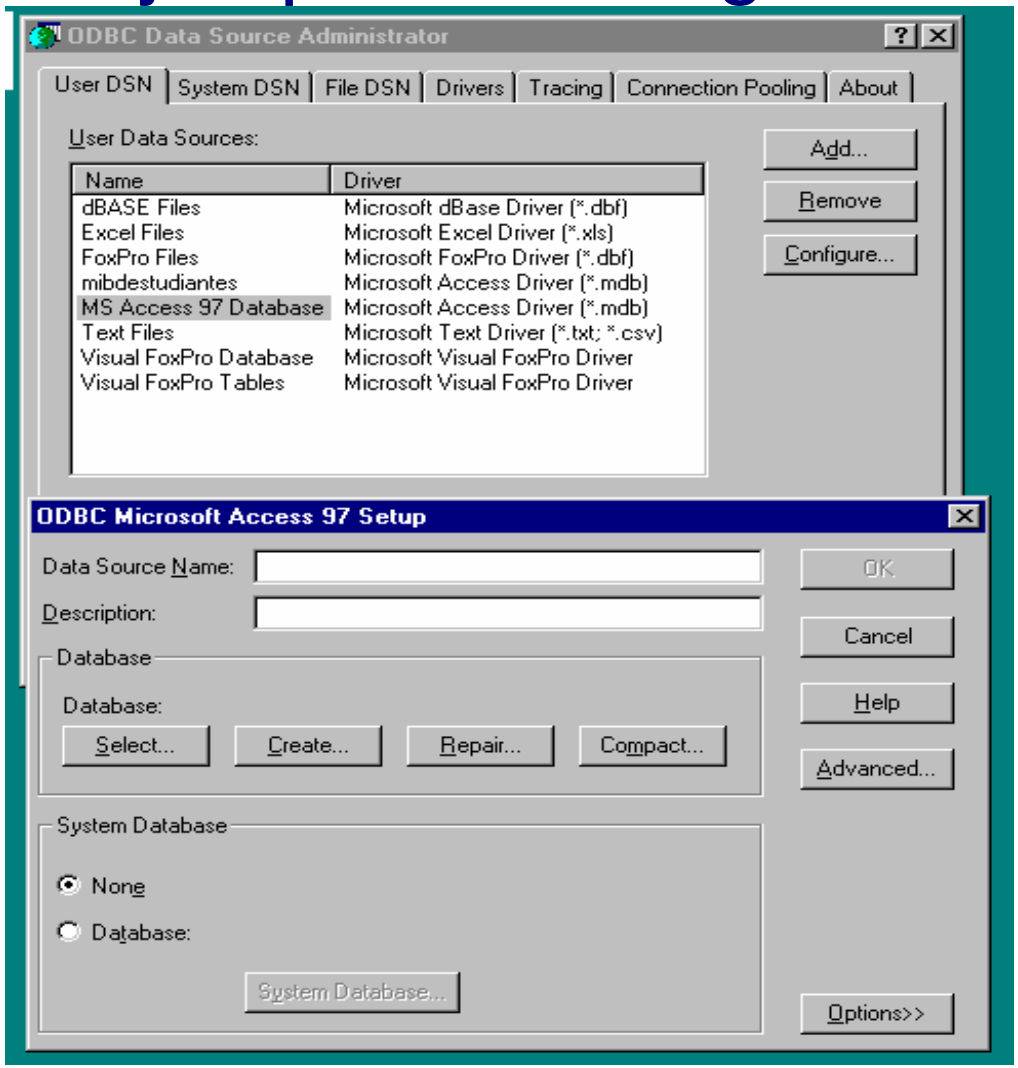
- Permite acceder a bases de datos en diferentes SGBD simultáneamente.
- Cada SGBD necesita su controlador (*ODBC driver*) específico para trasladar las llamadas del ODBC en llamadas específicas del sistema gestor.
- Para conectarse a la Base de Datos se crea una DSN (*Data Source Name*) dentro del ODBC que define los parámetros, ruta y características de la conexión

- *ODBC es un estándar de acceso a Bases de Datos desarrollado por Microsoft*
  - http://www.microsoft.com/data/odbc

2.2 Utilización de ODBC

- Pasos que realiza una aplicación que interacciona con una fuente de datos a través de ODBC:
  - se selecciona una fuente de datos (DSN).
  - se carga el controlador correspondiente.
  - se establece la conexión.
  - cada conexión sólo ve los cambios de las transacciones confirmadas.
  - la aplicación se desconecta de la fuente de datos para terminar la interacción.

2.2 ODBC Ejemplo de configuración



2.2 ODBC Ejemplo sencillo

- Ejemplo en matlab:

```
con1 = database('bdmysql',usu1,password1);
con2 = database('bdoracle',usu2,password2);

consulta='select nombre, apellido from alumnos';
consulta2='select dni, email from socios';

cursor=exec(consulta);
cursor=fetch(cursor)
cursor.data %se muestran los datos recuperados
close(cursor);
close(con1);

cursor2=exec(consulta2);
Cursor2 = fetch(cursor2)
cursor2.data % se muestran los datos de la 2º consulta
close(cursor)..
close (con2)
```

2.2 Ventajas ODBC

- Permite acceder a distintos tipos de SGBD de forma única *(por medio de una API)*.
- No es necesario saber detalles de cómo trabaja cada uno de estos sistemas.
- Biblioteca de funciones fija para distintos sistemas.
- Se puede crear una aplicación que acceda a varias fuentes de datos

2.3 **JDBC**

Java DataBase Connectivity

- ¿Qué es JDBC? Es una biblioteca de clases que permite la conexión con BD que soporten SQL mediante Java.
- El controlador correspondiente es el encargado de interactuar con el sistema.
- ¿Qué ventajas ofrece la utilización de JDBC? La aplicación es independiente de la plataforma y se puede mover la aplicación de un SGBD a otro. Al igual que ODBC, el código Java es independiente del SGBD utilizado.

2.3 JDBC: arquitectura

- **Aplicación** (inicia y termina las conexiones, envía sentencias SQL)
- **Gestor del controlador**
  (carga el *driver* JDBC)
- **Controlador**
  (conecta con la fuente de datos, transmite peticiones y devuelve/traduce resultados y códigos de error)
- **Fuente de datos**
  (el SGBD, procesa peticiones SQL)

2.3 JDBC: tipos de controladores

- Tipo 1 - Puentes: p.e. puente JDBC-ODBC, traducen JDBC a ODBC y se delega en ODBC para la comunicación con el SGBD.
- Tipo 2 – Traducción directa a controlador no-Java: Escrito parcialmente en JAVA y código nativo
- Tipo 3 – Network Bridges: es una biblioteca cliente escrita completamente en Java que utiliza un protocolo independiente de la BD para comunicar las peticiones a un servidor intermedio que las traduce a un protocolo específico del SGBD (el software del servidor intermedio que traduce las peticiones al SGBD concreto).
- Tipo 4 – Traducción directa a controlador Java: Escrito completamente en Java que traduce las peticiones al protocolo específico de cada SGBD.

2.3 Programación con JDBC

- Programación con JDBC:
  - Cargar la clase que representa al controlador.
  - Establecer una conexión a una base de datos.
  - Ejecutar sentencias SQL
  - Obtención de resultados
  - Cerrar la conexión.
- Ejemplo sencillo:

```
DriverManager.registerDriver(new com.mysql.jdbc.Driver());


string url =
   “jdbc:mysql://localhost:3306/estancias?user=usu1&passwor
   d=clave”;
Connection con = DriverManager.getConnection(url);
```

Especifica driver y abre conexion

```
statement stmt = con.createStatement();
resultSet rs = stmt.executeQuery("SELECT a, b, c FROM
   Tabla1");
```

Ejecuta una sentencia SQL

```
while (rs.next()) {
   int x = rs.getInt("a");
   String s = rs.getString("b");
   float f = rs.getFloat("c");
}
stmt.close();
```

Ejecuta una consulta y recoge los resultados

2.3 Programación con JDBC

- Gestión del driver
  - mediante la clase DriverManager (.registerDriver, .getConnection)
- Conexiones
  - cada conexión supone una sesión lógica
- Ejecución de sentencias SQL
  - statement (stmt.executeQuery)
- Obtención de resultados
  - ResultSet , métodos para cursores
    - next, previous, absolute, relative, first, last / get*Datatype* (getInt)

Statement
PreparedStatement
CallableStatement
(proc. almacenados)

2.3 Programación con JDBC

- Aspectos adicionales
  (ver en Ramakrishnan)
  - Gestión de errores y advertencias (6.3.5 / sl. 6.23-24)
  - Acceso a los Metadatos de la BD (6.3.6 / sl. 6.25-26)
  - Procedimientos almacenados (6.5 / sl. 6.31-35)

- Más información *(en inglés)*

http://www-db.stanford.edu/~ullman/fcdb/oracle/or-jdbc.html

3. Acceso a BD desde la web

3.1 Tecnologías web

- Conceptos básicos de la web
- Arquitectura de aplicaciones web
- Interfaces web (formularios)

3.2 Programación de aplicaciones web-BD

3.1 Tecnologías web

- La web es un sistema de información distribuido basado en hipertexto
- Los recursos accesibles en Internet se identifican por un URI (URL) (p.e. **http://informatica.uv.es/iiguia/2000/BD2/index.html**)
  - el protocolo con que acceder al recurso
  - el nombre (o dirección) del host
  - el nombre del recurso
- HTML es un lenguaje que indica como mostrar contenido textual junto al formato, imágenes y enlaces de hipertexto, así como posibilidades de entrada de datos
- HTTP es el protocolo de comunicación entre clientes y servidores web: el cliente envía una petición y el servidor produce una respuesta (una página HTML, [puede ser error])
  - ***HTTP es stateless***

3.1 Tecnologías web

- Arquitectura de aplicaciones en red
  - una aplicación en red intensiva en acceso a datos tiene 3 tipos de funcionalidades
    - Gestión de los datos
    - Lógica de la aplicación
    - Presentación
  - Estas funcionalidades se pueden (o no) separar en capas

Arquitecturas de una capa

- Todas las funcionalidades se combinan en una única capa, usualmente en un *mainframe*
- Los usuarios acceden mediante terminales “tontas” de texto

- *anticuado, (sobre)carga de cómputo en el sistema central*

Arquitecturas cliente-servidor

- División de trabajo con cliente ligero
  - el cliente sólo implementa la interfaz gráfica para presentación
  - el servidor implementa la lógica de la aplicación y la gestión de datos
- División de trabajo con cliente pesado
  - el cliente sólo implementa tanto la interfaz gráfica como la lógica de la aplicación
  - el servidor implementa la gestión de datos

Arquitecturas cliente-servidor

- Inconvenientes de las arquitecturas cliente-servidor, esp. con cliente pesados
  - Los clientes deben instalarse (y reinstalarse)
  - No hay un lugar único donde actualizar la lógica de la aplicación
  - Puede tener problemas de seguridad
    - El servidor debe confiar en los clientes y controlar el control de acceso y la autenticación
  - Es difícil de escalar a muchos clientes, por la gran cantidad de datos a transferir

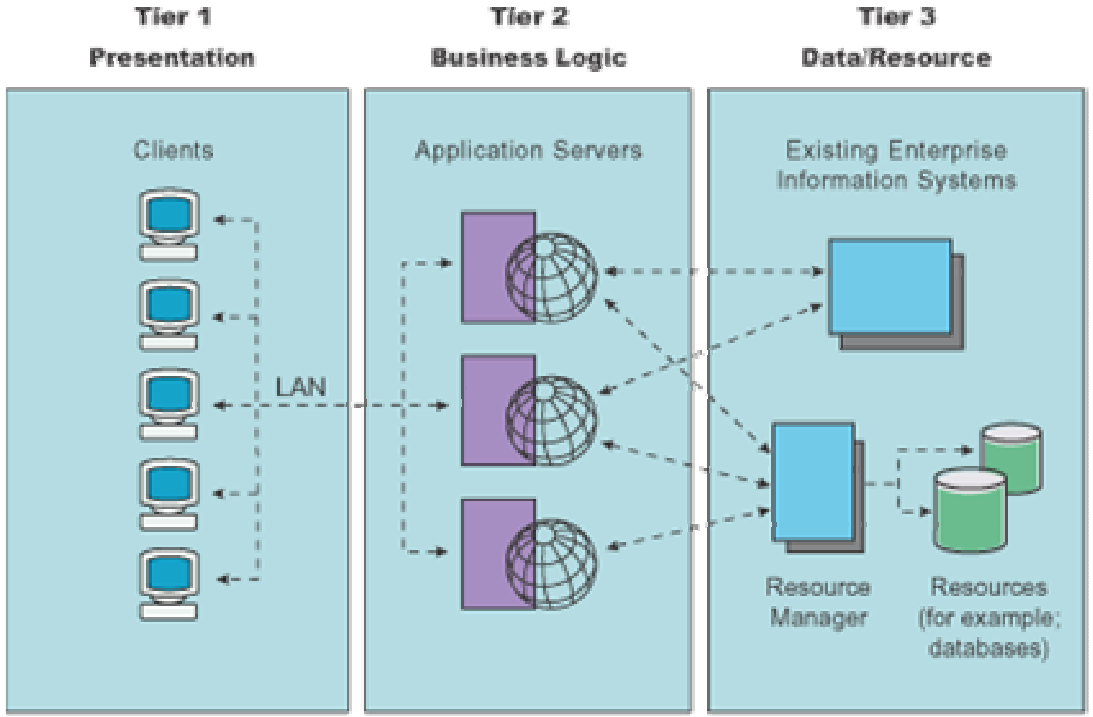
3.1 Tecnologías web

- Arquitectura de tres capas
  - Capa de presentación
    - Interfaz con el usuario (cliente-navegador)
  - Capa intermedia (servidor web, serv. aplicaciones) (también capa de “lógica de la aplicación”)
    - Implementa la lógica de la aplicación (control de flujo)
    - Puede acceder a diferentes fuentes de datos
  - Capa de gestión de datos
    - Uno o más SGBD

(Ej.: Automatrícula de la UV)

3.1 Tecnologías web

- Arquitectura de tres capas



3.1 Tecnologías web

- Ventajas de la Arquitectura de tres capas
  - Heterogeneidad de los sistemas
    las capas pueden ser modificadas independ.
  - Clientes ligeros: el usuario sólo req. un navegador
  - Acceso integrado a los datos: se puede acceder a varias BD transparentemente para el usuario
  - Escalabilidad: los servidores intermedios pueden replicarse para servir a más clientes
  - Desarrollo de software centralizado

3.1 Tecnologías web

- Interfaces web: Formularios
  - son el modo más común de solicitar y comunicar datos entre el cliente y el servidor
  - los formularios se insertan en una página web
  - componentes de un formulario

<**FORM ACTION**="/cgi-bin/postman" **NAME**="theform" **METHOD**="POST">

  - campos de un formulario
    - INPUT type, name, value *(Ej.: correo.uv.es)*

<**INPUT TYPE**="text" **NAME**="user" **VALUE**="">

3.1 Tecnologías web

- Interfaces web: Formularios
  - métodos de paso de argumentos
    - GET y POST
  - codificación de caracteres introducidos
    - el usuario puede introducir cualquier carácter ASCII, entre ellos caracteres no válidos para un URI
    - los caracteres se convierten según su código
    - un programa en el servidor tendrá que reconvertirlos
  - Un tutorial sobre formularios (y todo HTML) en

http://gias720.dis.ulpgc.es/Gias/Cursos/Tutorial_html/guia_rap.htm#dform

3.2 Aplicaciones web-BD

- Básicamente se tienen dos categorías:
  - con lenguajes de escritura de *scripts.*
  - con lenguajes de programación de propósito general.

- Los programas con *scripts* pueden ser
  - para que se ejecuten en el servidor web
  - para que se ejecuten en el propio navegador
- Los programas con lenguajes de prop.general
  - se ejecutan siempre en el S.O. del servidor web

3.2.1 Lenguajes de escritura de *scripts*

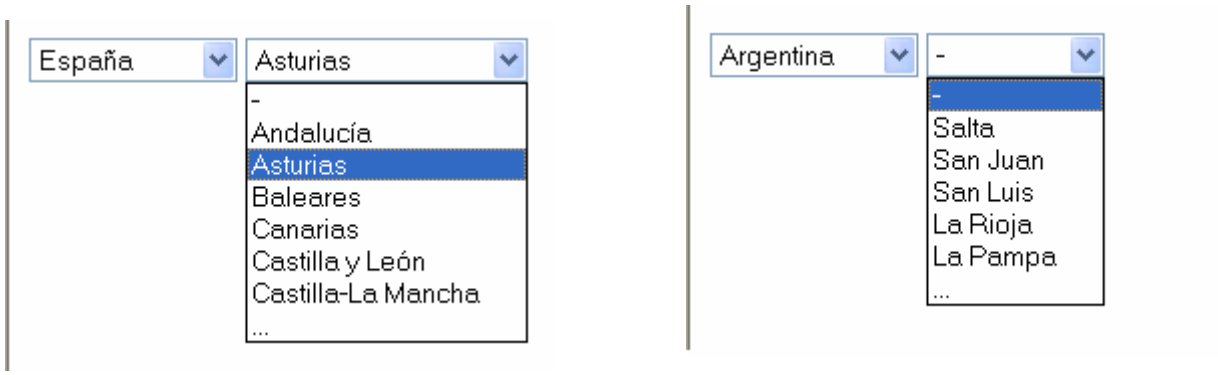
- Se pueden escribir *scripts* que se ejecuten en:
  - el cliente: se ejecutan en las páginas web que se envían al navegador (*host* para este código).
  - el servidor : son ejecutados e interpretados por el propio servidor web quien se encarga de generar el formato comprensible que será enviado al cliente (navegador) que hizo la petición.
    El navegador nunca ve el contenido de los *scripts* del lado del servidor, sólo la salida que éstos producen.

3.2.1 Lenguajes de escritura de scripts

- *Scripts* del lado del cliente:
  - Son pequeños programas que se insertan en la página HTML con la que el servidor web responde a una petición de un navegador (javascript, VBScript)
    - generalmente se incluyen en HTML entre etiquetas <SCRIPT> … </SCRIPT>
  - Su objetivo es añadir ciertas funcionalidades a la capa de presentación
  - Permiten realizar pequeñas tareas: comprobación de datos en un formulario, cambio de idioma dependiendo del cliente.

3.2.1 Lenguajes de escritura de scripts

- *Scripts* del lado del cliente:
  - JavaScript es un lenguaje de *scripts* completo
    - con variables, operadores, sentencias y funciones
  - Ejemplo:
    - cuando se selecciona un país en el menú de la izquierda el menú de la derecha muestra las regiones de ese país



3.2.1 Lenguajes de escritura de *scripts*

- *Scripts* del lado del servidor:
  - cuando un cliente (navegador) solicita al servidor web una URL que no es una página estática sino un *script*, el servidor web utiliza un motor de *scripts* para ello
  - el servidor también recibe los *inputs* de formularios
  - el *script* procesa y debe producir una página web que el servidor devuelve como respuesta.
  - los *scripts* pueden estar incrustados en HTML
  - lenguajes utilizados:
    - ASP
    - PHP
    - JSP

Scripts del lado del servidor

ASP (Active Server Pages) es el lenguaje de *scripts* del lado del servidor creado por Microsoft. ASP permite conexión a BD mediante ODBC.

PHP (PHP: *Hypertext Preprocessor)* es un lenguaje interpretado de alto nivel embebido en páginas HTML. Permite el acceso a las principales BDs a través de funciones específicas de PHP escritas en C para cada SGBD. Permite además acceso a bases de datos mediante ODBC.

JSP (Java Server Pages) es un lenguaje de *scripts* del lado del servidor. Utiliza para la parte dinámica de la página el lenguaje Java. Puede utilizar todas las funcionalidades que ofrece Java para acceder a BDs.

Ejemplo de código ASP

```
<HTML>
<HEAD>
<TITLE>Consulta en una tabla</TITLE>
</HEAD>
<BODY>
<h1><div align="center">Lectura de la
tabla</div></h1>
<br>
<br>
<%
'instanciar el objeto Connection
Set Conn =
Server.CreateObject("ADODB.Connection")

'Se abre la base de datos (nombre dado ODBC)
Conn.Open "Mifuentededatos"

'Ahora creamos la sentencia SQL que nos
'servira para hablar a la BD
sSQL="Select * From Clientes Order By
nombre"

'Ejecutamos la orden
set RS = Conn.Execute(sSQL)
```

conexión

Consulta

```
'Mostramos los registros%>
<table align="center">
<tr>
<th>Nombre</th>
<th>Teléfono</th>
</tr>
<%
Do While Not RS.Eof
%>
<tr>
<td><%=RS("nombre")%></td>
<td><%=RS("telefono")%></td>
</tr>
<%
RS.MoveNext
Loop

'Cerramos la conexion
Conn.Close
%>
</table>
</BODY>
</HTML>
```

Imprimir datos

Ejemplo de código PHP

```
<HTML>
<HEAD>
<TITLE>consulta.php</TITLE>
</HEAD>
<BODY>
<h1><div
align="center">Consultar
Tabla</div></h1>
<br>
<br>

<?
//Conexion con BD

mysql_connect("localhost",“usu
",“elpassword");

//Ejecutamos la sentencia SQL
$result=mysql_db_query(“miejem
plo","select * from
clientes");
?>
```

conexión

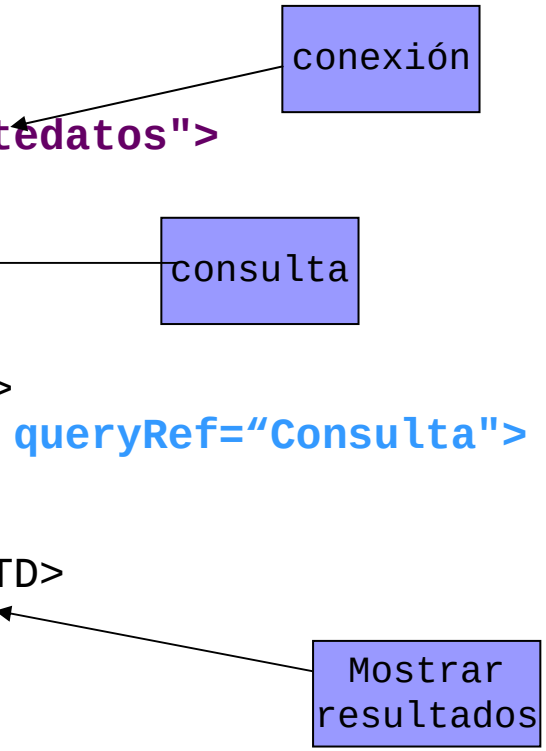
consulta

```
<table align="center">
<tr>
<th>Nombre</th>
<th>Teléfono</th>
</tr>

<?
//Mostramos los registros
while
($row=mysql_fetch_array($result))
{
echo
'<tr><td>'.$row["nombre"].'</td>';
echo
'<td>'.$row["telefono"].'</td></tr>';
}
mysql_free_result($result)
?>

</table>
</BODY>
</HTML>
```

Imprimir resultados

Ejemplo de código JSP

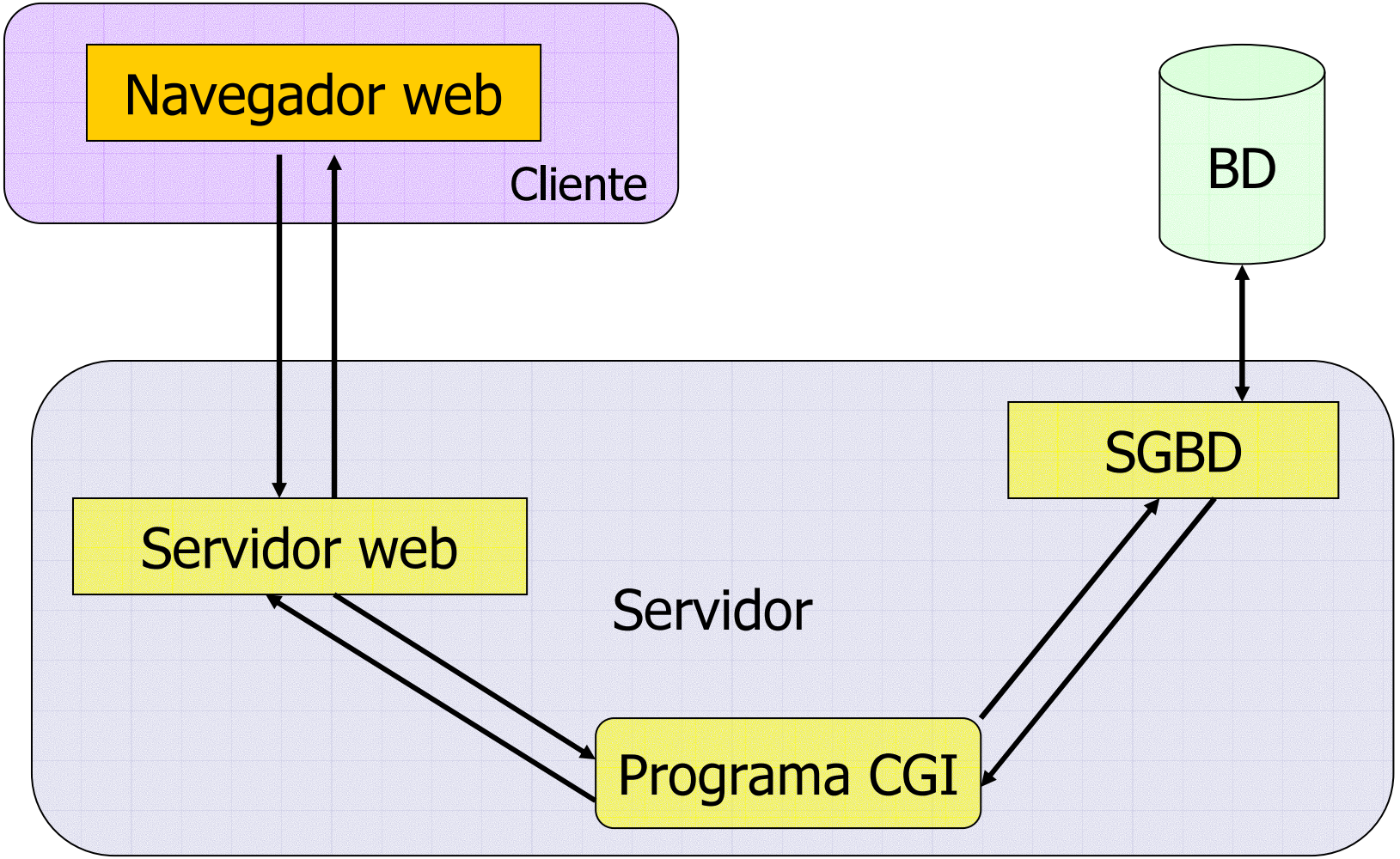
```
<%@ taglib prefix="blx" uri="/blx.tld" %>
<HTML>
<BODY>
<P>
<blx:sqlConnection jndiName="mifuentedatos">
<blx:sqlQuery id="Consulta">
     SELECT dni,nombre FROM socios
</blx:sqlQuery>
<TABLE>
<TR><TD>id </TD><TD>Nombre</TD></TR>
<blx:sqlExecuteQuery resultSet="rs" queryRef="Consulta">
<TR>
<TD><%= rs.getInt("id") %></TD>
<TD><%= rs.getString("Nombre") %></TD>
</TR>
</blx:sqlExecuteQuery>
</TABLE>
</blx:sqlConnection>
</BODY>
</HTML>
```



3.2.2 Aplicaciones Web escritas con lenguajes de propósito general

- CGI (Common Gateway Interface):
  - Esta interfaz fue una de las primeras técnicas empleadas para crear contenido dinámico.
  - El objetivo del interfaz CGI es transmitir los *inputs* de formularios HTML a un programa de aplicación ejecutándose en la capa intermedia (de aplicación)
  - Con CGIs el programa servidor web:
    - pasa las peticiones de los clientes a un programa externo
    - el servidor web envía la salida de este programa al cliente como página estática
    - en una aplicación de bases de datos, el programa CGI interactuará con los SGBD

Acceso a BDs con CGI

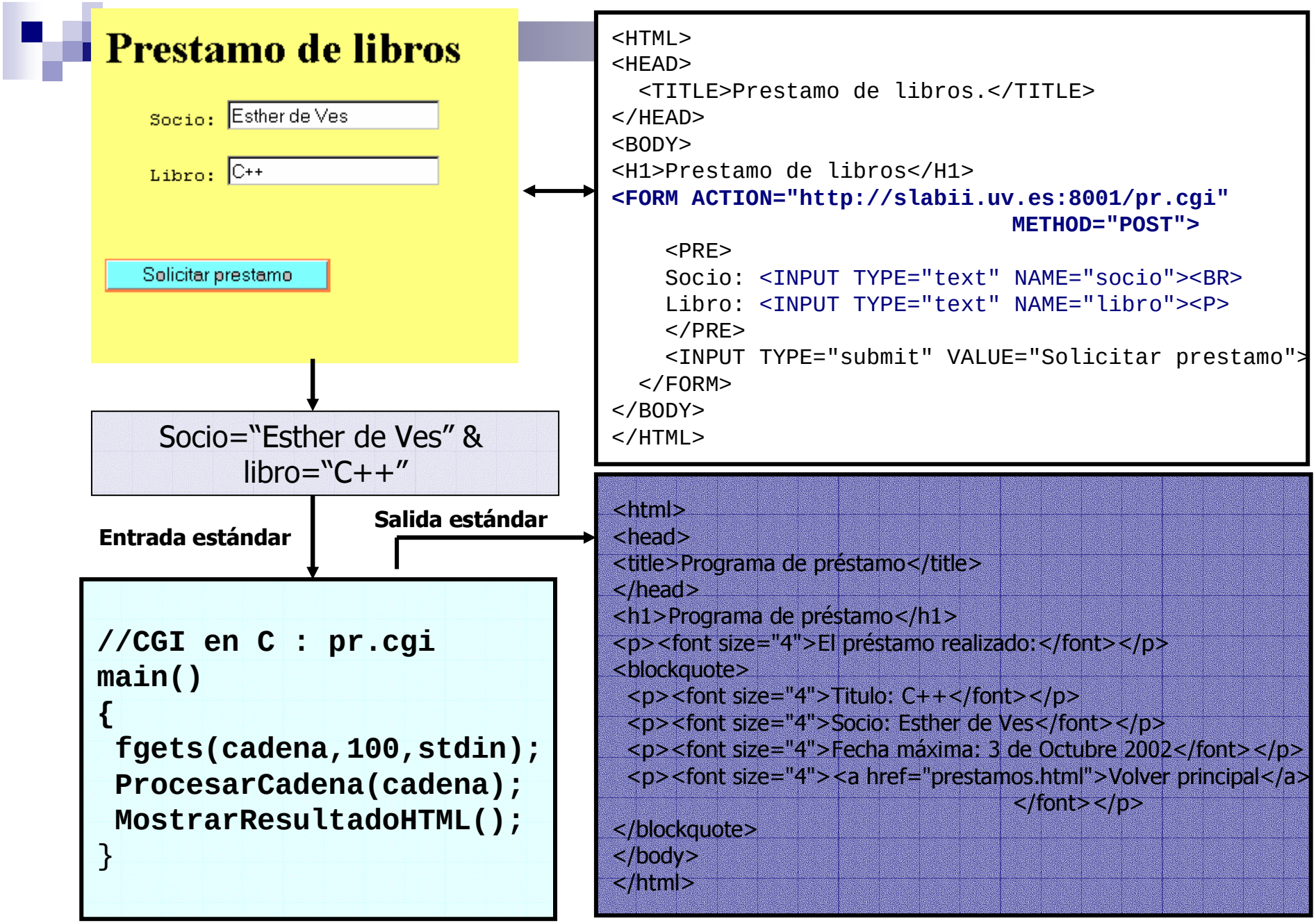


3.2.2 Aplicaciones Web escritas con lenguajes de propósito general

- CGI (Common Gateway Interface):
  - Los programas CGI pueden ser:

Compilados: C, C++, Pascal	Scripts ejecutables: Perl, tcl, sh

  - Desventajas de los CGI
    - se crea un nuevo proceso para cada invocación
    - no se pueden compartir recursos entre prog. de aplicación
    - ambos problemas se resuelven en parte con FastCGI o con servidores de aplicaciones


Prestamo de libros
Socio: Esther de Ves
Libro: C++
Solicitar prestamo
<HTML>
<HEAD>
<TITLE>Prestamo de libros.</TITLE>
</HEAD>
<BODY>
<H1>Prestamo de libros</H1>
<FORM ACTION="http://slabii.uv.es:8001/pr.cgi"
METHOD="POST">
<PRE>
Socio: <INPUT TYPE="text" NAME="socio">
Libro: <INPUT TYPE="text" NAME="libro"><P>
</PRE>
<INPUT TYPE="submit" VALUE="Solicitar prestamo">
</FORM>
</BODY>
</HTML>
Socio="Esther de Ves" &
libro="C++"
Entrada estándar
Salida estándar
//CGI en C : pr.cgi
main()
{
fgets(cadena,100,stdin);
ProcesarCadena(cadena);
MostrarResultadoHTML();
}
<html>
<head>
<title>Programa de préstamo</title>
</head>
<h1>Programa de préstamo</h1>
El préstamo realizado:
<blockquote>
Titulo: C++
Socio: Esther de Ves
Fecha máxima: 3 de Octubre 2002
Volver principal

</blockquote>
</body>
</html>

Detalles del código de un ejemplo CGI

Ver sobre los materiales de la práctica 1

- cgi.cxx: métodos para tratar los *inputs* del CGI
- vuelo.pc.alu: programa que produce la salida HTML

3.2.2 Aplicaciones Web escritas con lenguajes de propósito general

- Java Servlets:
  - Alternativa de Java a los programas CGI tradicionales.
  - Es un programa escrito en Java (por tanto independiente de la plataforma), ejecutándose en la capa intermedia, que responde a las peticiones de uno o más clientes.
  - La conexión a BD se puede realizar a través de JDBC.

Ejemplo parcial Java Servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class ServletDelDesc extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  private static final String USER = "nuria";
  private static final String PASSWORD = "nuria";
  // Conexion a la BD
  Connection Conexión;
….
 // Crear la conexión a la BD
 try {
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Conexion = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:midb",USER,PASSWORD);
         if (eliminar != null){

              EliminarBD(request, out);

            out.println("</body></html>");
            out.flush();
            out.close();
    }
……..
```

3.3 Consideraciones sobre aplicaciones web

HTTP es un protocolo sin estado

- Ventajas
  - facilidad de uso e implementación
  - efectividad para pag. estáticas sin usar mucha memoria
- Inconvenientes
  - el protocolo en sí no recuerda el flujo ni las acciones anteriores
- Hay diversas vías de superar este inconveniente

3.3 Consideraciones sobre aplicaciones web

- Guardar información en las capas “bajas”
  - La información se puede guardar
    - en la base de datos
    - en memoria local de la capa intermedia
  - Tiene inconvenientes en cuanto a sobrecarga en los accesos a la BD o a la memoria
  - Sólo debe usarse esta solución para guardar en BD información que debe ser duradera

3.3 Consideraciones sobre aplicaciones web

- Guardar el estado en el cliente: cookies
  - Consisten en guardar una pequeña información textual en el cliente, que se envía al servidor en cada petición HTTP
  - Son una colección de (nombre, valor)
  - Son percibidas como “peligrosas”
  - Son fáciles de usar pero limitadas a 4 KB
  - Pueden ser deshabilitadas por el cliente
  - Son útiles para información de *login* y como modo de memoria temporal (no permanente)

3.3 Consideraciones sobre aplicaciones web

- Guardar el estado en el cliente
  - campos ocultos
  - información en el *path*
  - ambos mecanismos se pueden usar para solventar la deshabilitación de *cookies*
  - no requieren almacenamiento adicional ya que son “pasados” en cada petición HTTP
    - *ver ejemplo en correo.uv.es*

4.Herramientas específicas de los SGBD

- En general cada sistema dispone de herramientas para el desarrollo de aplicaciones de BD.
- Oracle:
  - Developer 2000: permite al usuario diseñar de forma interactiva programas con consultas y transacciones.
  - Servidor web de Oracle: permite acceder a BD Oracle desde el web.

4 Herramientas específicas del SGBD

- Access dispone de varias características:
  - Asistentes para crear formularios,
  - Asistente para informes,
  - Incluir macros,
  - Programación en Visual Basic.
- MySQL
  - phpMyAdmin permite administrar y consultar la base de datos a través de la web (escrito en PHP)

Bibliografía

- R. Ramaskrishnan, J. Gehrke. Database Management Systems. (3ª edición) McGraw Hill. Capítulos 6 y 7.
- Elmasri y Navathe. Fundamentos de Sistemas de Bases de datos. Addison Wesley. Capítulo 10 y 27.
- Documentacion Oracle 8.0.

Ejercicios:

1. Describe las fases para generar un ejecutable de un programa SQL embebido.
2. Imagina que tienes datos almacenados en dos SGBDs diferentes (Oracle, SQL Server, Access) y tienes que hacer una aplicación que trabaje con todas esta información, ¿Qué podrías utilizar?
3. Ventajas y diferencias entre ODBC y JDBC.
4. ¿Qué es un CGI?
5. ¿Qué utilizarías para realizar una aplicación con acceso a una BD *mysql?*
6. Enumera algunas de las alternativas existentes para realizar una aplicación web que acceda a una BD Oracle

Supongamos que tenemos una BD Oracle instalada en un servidor pokemon.uv.es. El nombre de la BDs es mibd. En esa BD tenemos almacenada información sobre una biblioteca en la que, simplificando, tenemos 3 tablas:

```
Socios(dni, nombre, apellidos,email,telefono)
Libros(idlibro,titulo, autor)
Prestamos(dni,idlibro, fecha)
```

Deseamos crear una pequeña aplicación accesible desde web que nos permita saber si un determinado libro está alquilado, por quien y desde cuando. El servidor web está instalado en una maquina diferente al servidor de datos: slabii.uv.es.

Utilizando los ejemplos sobre cgi y sql embebido, intenta escribir un pequeño programa que permitiría realizar esta tarea. Ten en cuenta que el servidor de datos y el servidor web no son la misma máquina.