

# Desarrollo de aplicaciones de bases de datos

Escuela Universitaria de Informática  
Semestre 3B

---

Tema V: Aspectos tecnológicos de las aplicaciones de Bases de  
Datos

# Índice

---

1. Introducción
2. Arquitecturas del SGBD multiusuarios
3. El estándar *Open Database Connectivity* (ODBC)
4. Tecnología Web y SGBDs

# 1. Introducción

---

- Una aplicación se construye en una arquitectura software
- Aspectos tecnológicos:
  - conexión entre la aplicación y el SGBD: teleproceso, servicio de ficheros, cliente/servidor, web
  - soporte de red: protocolo, tolerancia, ...
  - arquitectura del sistema de gestión de bases de datos: distribuidas, replicadas, heterogéneas
  - sistemas abiertos: evitar la dependencia de los servidores, desarrollo de aplicaciones transportables

## 2. Arquitecturas de SGBD multiusuarios

---

Arquitecturas más comunes usadas en la implementación de SGBD multiusuarios:

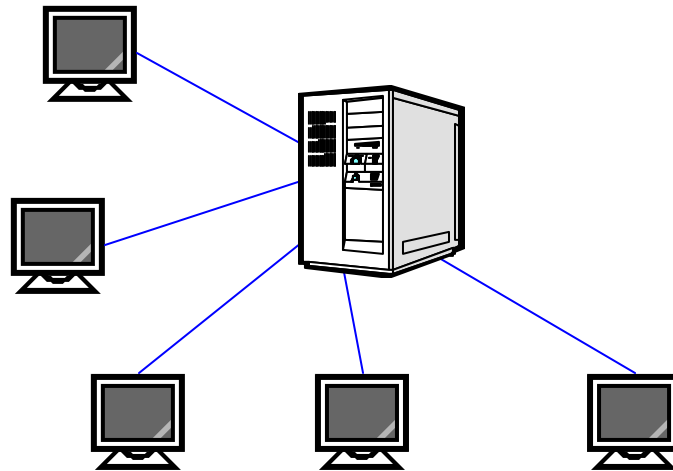
- Teleproceso:
- Entorno de servicio de ficheros
- Cliente/Servidor

## 2. Arquitecturas de SGBD multiusuarios

---

### Teleproceso:

- Arquitectura tradicional
- Una única CPU y un número de terminales



## 2. Arquitecturas de SGBD multiusuarios

---

### Teleproceso:

- Todos los procesos son realizados por un único ordenador
- Los terminales son “tontos”, incapaces de hacer ninguna función
- Los terminales envían señales vía el subsistema de comunicaciones del SO al programa de aplicación del usuario
- El programa de aplicación realiza llamadas al SGBD

## 2. Arquitecturas de SGBD multiusuarios

---

### Teleproceso: problemas

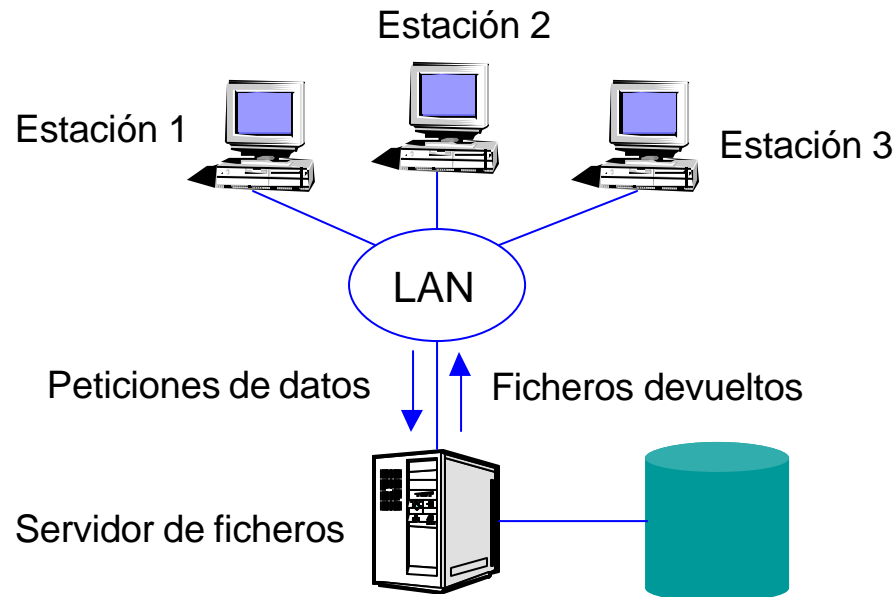
- Todo el trabajo lo realiza el único ordenador (mainframe):
  - los programas de aplicación
  - SGBD (procesamiento de datos)
  - Gestión de terminales, formateo de información, control de teclado, ratón, etc. (interacción con el usuario)
- Con el incremento de necesidades de interfaces más complejas la carga de trabajo para el ordenador es brutal

## 2. Arquitecturas de SGBD multiusuarios

---

### Entorno de servicio de ficheros:

- Arquitectura distribuida en un conjunto de estaciones en una LAN
- El servidor de ficheros mantiene los ficheros que requieren el SGBD y las aplicaciones
- Las aplicaciones y el SGBD de sobremesa se ejecutan en cada estación de trabajo





## 2. Arquitecturas de SGBD multiusuarios

---

### Entorno de servicio de ficheros: tareas de las estaciones

- La aplicación
- SGBD de sobremesa
- Sistema operativo con LAN

### Entorno de servicio de ficheros: tareas del servidor

- Sistema operativo con LAN
- Gestión de los ficheros

## 2. Arquitecturas de SGBD multiusuarios

---

### Entorno de servicio de ficheros:

- El servidor de ficheros actúa como un dispositivo de disco duro compartido
- El SGBD en cada estación solicita al servidor de ficheros todos los ficheros donde se encuentren los datos que necesita
- El servidor de fichero sólo sirve ficheros, no es capaz de evaluar consultas

## 2. Arquitecturas de SGBD multiusuarios

---

### Entorno de servicio de ficheros: problemas

- Hay un gran volumen de tráfico de red, ya que el servidor de ficheros ha de proporcionar ficheros enteros a los SGBD
- Es necesario un SGBD en cada estación
- Si el volumen de datos es grande y el número de usuarios es grande el servidor de ficheros se convierte en un cuello de botella
- Habitualmente no está bien equipado para la gestión de la concurrencia, recuperación y control de la integridad

## 2. Arquitecturas de SGBD multiusuarios

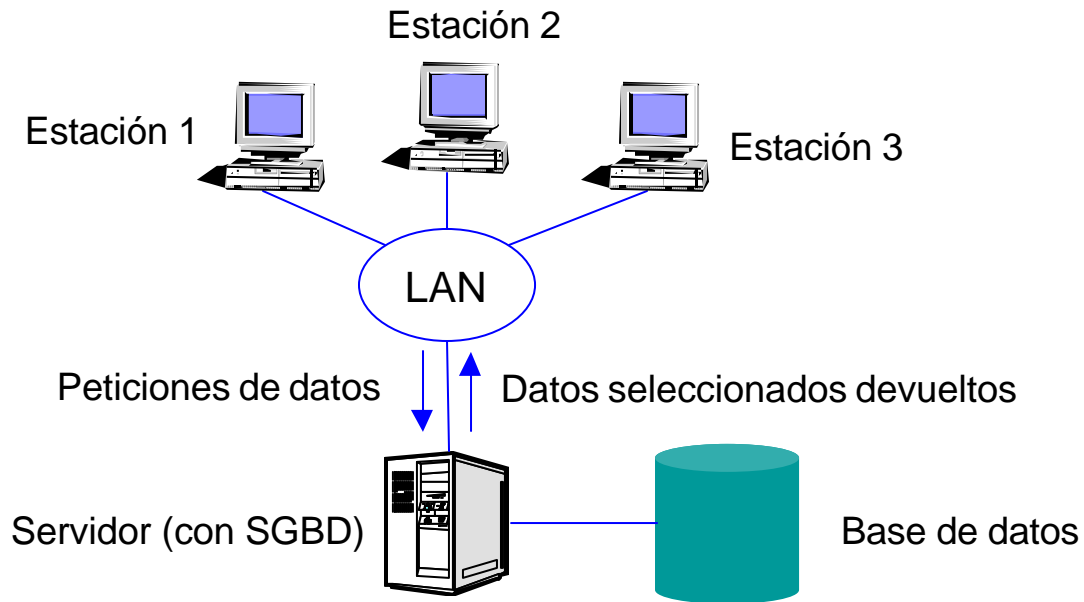
---

### Cliente/Servidor:

- Se desarrolla para solventar los problemas de las dos aproximaciones anteriores
- Separación de tareas: la porción de la aplicación *front-end* en los clientes y la porción *back-end* en el servidor
- Forma de interacción: proceso en un cliente requiere algún servicio (datos usualmente) y un servidor que los proporciona
- El proceso cliente y el servidor residen en el mismo o en diferentes ordenadores (usualmente en ordenadores distintos con un soporte de red para establecer la comunicación)

## 2. Arquitecturas de SGBD multiusuarios

Cliente/Servidor:



## 2. Arquitecturas de SGBD multiusuarios

---

### Cliente/Servidor: tareas del cliente

- Maneja la interfaz de usuario
- Acepta y comprueba la sintaxis de las entradas del usuario
- Procesa la aplicación
- Genera peticiones a la base de datos (en algún DML) y las transmite al servidor
- Espera las respuestas del SGBD y las formatea para el usuario final

## 2. Arquitecturas de SGBD multiusuarios

---

### Cliente/Servidor: tareas del servidor

- Acepta y procesa peticiones de los clientes
- Comprueba las autorizaciones
- Asegura que las restricciones integridad no son violadas
- Realiza el procesamiento de consultas y actualizaciones y transmite la respuesta al cliente
- Mantiene el catálogo del sistema
- Proporciona el acceso concurrente a la base de datos
- Proporciona el control de la recuperación

## 2. Arquitecturas de SGBD multiusuarios

---

### Cliente/Servidor: ventajas

- Proporcionan una división de las tareas más eficiente:
  - cliente: gestión de la interfaz gráfica y de la interacción con el usuario
  - servidor: procesamiento de altas prestaciones de grandes volúmenes con control de la seguridad, integridad y concurrencia
- Permiten el escalado horizontal y vertical de recursos
- Se pueden utilizar clientes de menores prestaciones
- Los usuarios pueden continuar utilizando sus aplicaciones favoritas
- Se permite un acceso más abierto a los datos: SQL
- Se incrementa la seguridad e integridad
- Aumenta el rendimiento (si los clientes y el servidor residen en ordenadores distintos) al realizar procesamiento en paralelo



## 2. Arquitecturas de SGBD multiusuarios

---

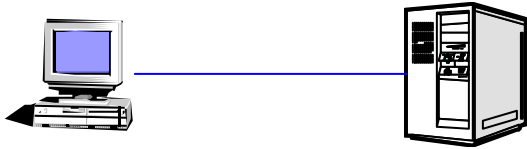
### Cliente/Servidor: ventajas

- Es más sencillo afinar del servidor si su única tarea es realizar el procesamiento de la base de datos
- El coste del hardware se reduce: sólo es necesario que el servidor tenga la potencia de almacenamiento y procesamiento suficiente
- Se reducen los costes de comunicación (sólo peticiones y resultado de las mismas)
- Se incrementa la consistencia: la integridad se maneja en un único lugar (el servidor)
- Esta arquitectura se lleva fácilmente a una arquitectura de sistemas abiertos

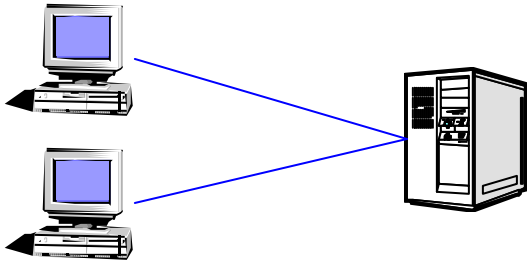
## 2. Arquitecturas de SGBD multiusuarios

---

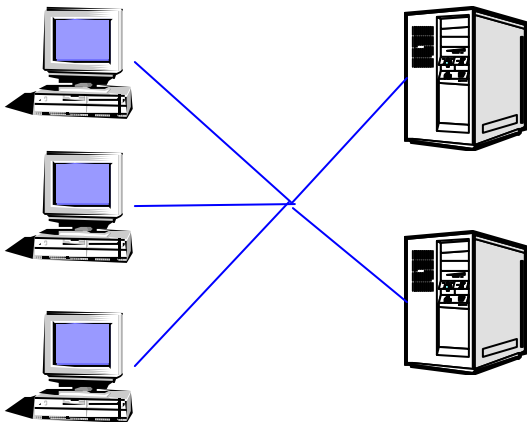
### Cliente/Servidor: variantes de topologías



Cliente único, servidor único



Múltiples clientes, servidor único

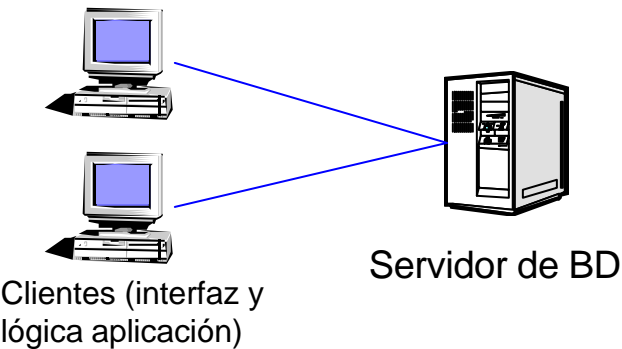


Múltiples clientes, múltiples servidores

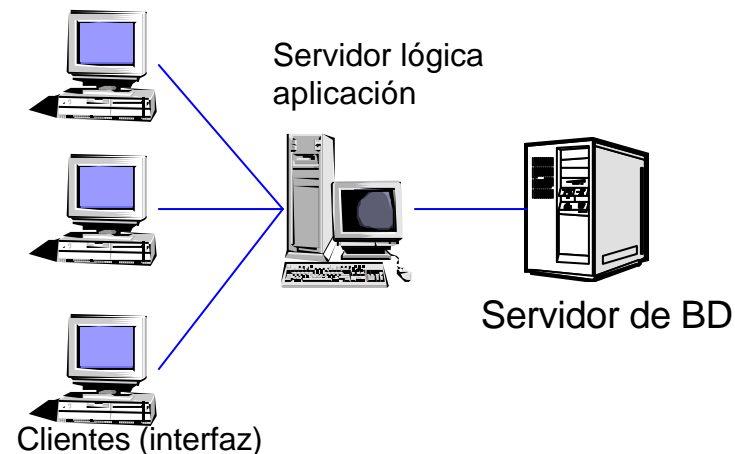
## 2. Arquitecturas de SGBD multiusuarios

---

### Cliente/Servidor: variantes de topologías



Arquitectura cliente/servidor de dos capas



Arquitectura cliente/servidor de tres capas  
(más apropiadas para Internet e intranets)

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Construcción de programas de aplicación: formas

- SQL embebido:
  - en el código fuente de los programas se incluyen instrucciones de SQL puro
  - se proporciona precompiladores (de la casa comercial del SGBD) para tratar las instrucciones SQL incluidas y obtener un programa compilable
- Problemas:
  - Falta de interoperatividad: los programas se han de compilar con el precompilador del SGBD y montar con las librerías del SGBD
  - En caso de instrucciones SQL no estándar aparece la necesidad de versiones para distintos SGBD o código específico para cada SGBD

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Construcción de programas de aplicación: formas

- Interfaz de programación de aplicación (API): conjunto de funciones de librerías para la mayoría de tipos de accesos de base de datos comunes que los programadores necesitan:
  - conectarse a la base de datos
  - ejecutar instrucciones SQL de actualización
  - recuperar filas como resultado de una consulta,
  - ...
- Problemas:
  - Falta de interoperatividad: los programas se han de compilar con el precompilador del SGBD y montar con las librerías del SGBD
  - Versiones para distintos SGBD o código específico para cada SGBD

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Open Database Connectivity (ODBC)

- Propuesta de estándar de Microsoft en 1991
- Tecnología ODBC: proporciona una interfaz común para acceder a bases de datos SQL heterogéneas basada en un SQL restringido, caracterizado por un conjunto mínimo de instrucciones
- Proporciona una alta interoperatividad: una aplicación simple puede acceder a distintos SGBD que soporten SQL a través del mismo código
- Permite al desarrollador construir y distribuir aplicaciones cliente/servidor que no estén orientadas a un SGBD específico
- Se necesitan *drivers* de base de datos para enlazar la aplicación con el SGBD

### 3. El estándar *Open Database Connectivity* (ODBC)

---

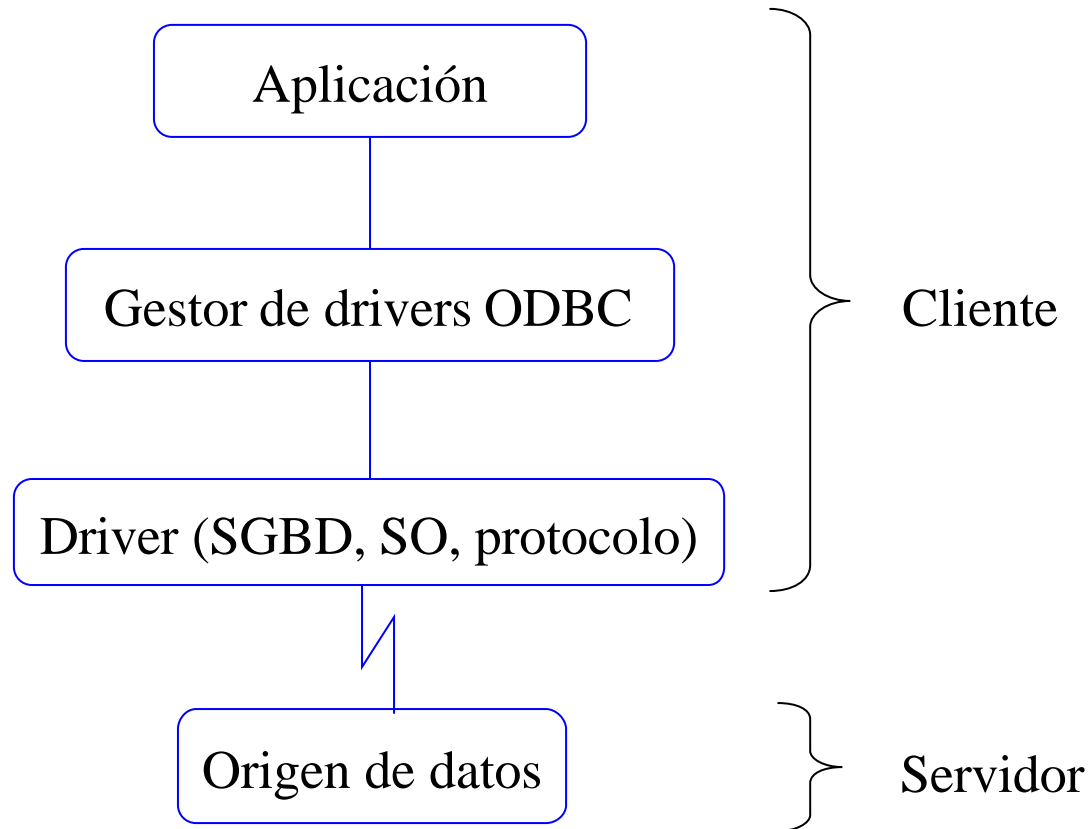
ODBC es el estándar de la industria *de facto*. Razones:

- Las aplicaciones no son esclavas de una casa comercial propietaria de la API
- Las instrucciones SQL pueden ser incluidas explícitamente en el código fuente o construidas en tiempo de ejecución
- La aplicación puede ignorar los protocolos de comunicaciones subyacentes
- Los datos pueden ser enviados y recibidos en un formato que sea adecuado para la aplicación
- ODBC está diseñado en conjunción con los estándares X/Open y ISO CLI
- Actualmente existen *drivers* ODBC para más de 50 SGBD conocidos

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Arquitectura de ODBC:





# 3. El estándar *Open Database Connectivity* (ODBC)

---

## Arquitectura de ODBC: componentes

- **Aplicación:** realiza las llamadas SQL, independientemente del protocolo de comunicaciones, del servidor SGBD y del SO del nodo donde se encuentra el SGBD
- **Gestor de drivers:** es el responsable de cargar los drivers a petición de la aplicación; proporciona algunas funciones para convertir nombres de tipos de datos usados por la aplicación en nombres de tipo usados por el SGBD
- **Drivers:** enmascaran todos los problemas de la heterogeneidad (SGBD, SO y protocolos de red). Son los responsables de llevar a cabo las funciones ODBC. Ejecutan las instrucciones SQL, traduciendo las para adaptarlas a la sintaxis y la semántica de los SGBD específicos. También son responsables de la devolución de los resultados, usando mecanismos de *buffering*
- **Orígenes de Datos:** los SGBD remotos, llevan a cabo las llamadas transmitidas por los clientes.

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Tipos de drivers ODBC: de dos pasos

- reciben las llamadas de la aplicación y procesan directamente su contenido
- simulan la existencia de un servidor SQL
- interpretan las llamadas y atacan un fichero o conjunto de ficheros locales al ordenador donde se ejecuta la aplicación
- ejemplos: drivers de Access, dBase, Paradox, etc.

### 3. El estándar *Open Database Connectivity* (ODBC)

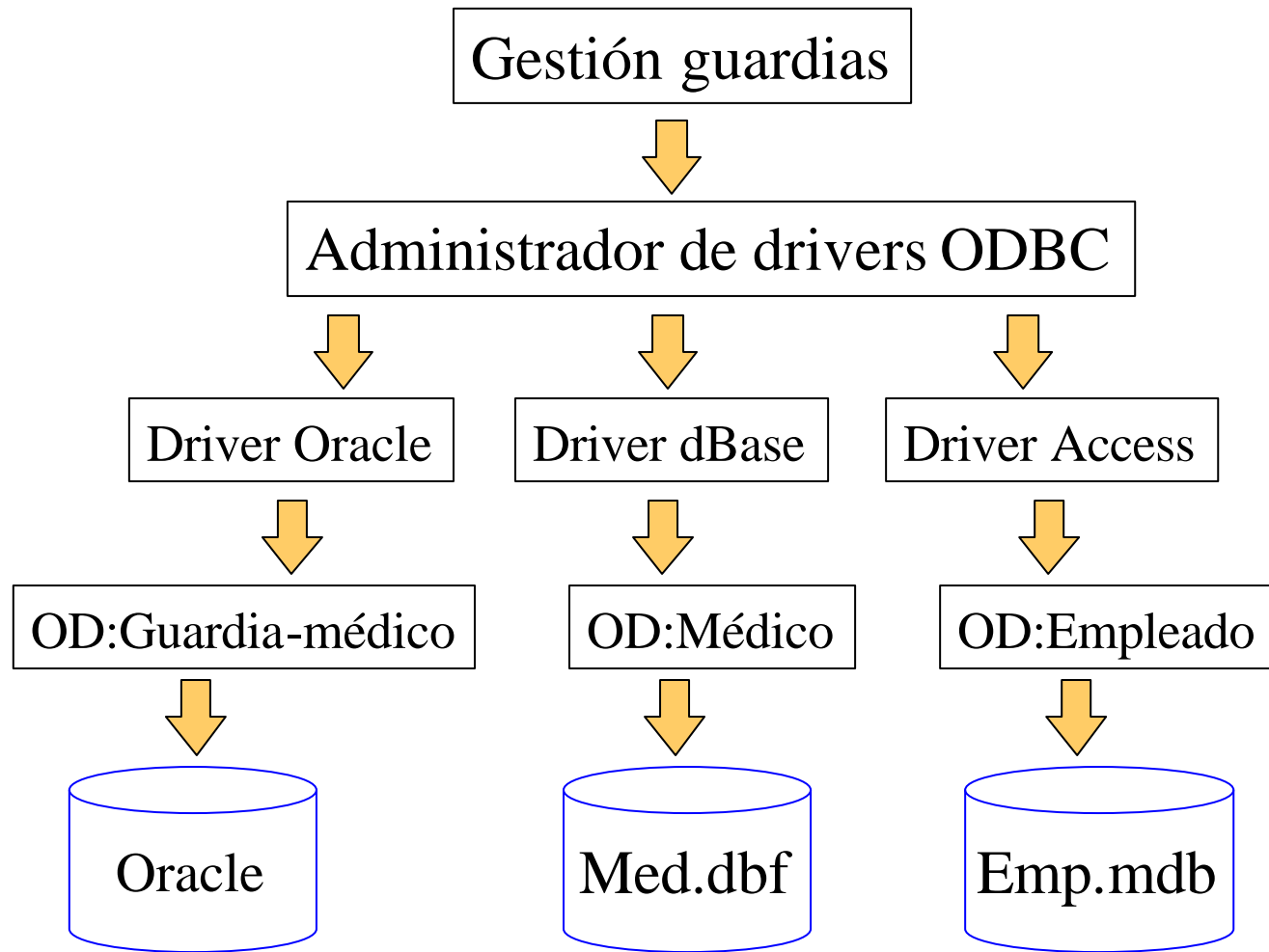
---

#### Tipos de drivers ODBC: de pasos múltiples

- reciben las llamadas de la aplicación y traducen al lenguaje del servidor
- actúan como medio de transporte de las peticiones y de los resultados de las mismas
- pueden constar de dos o más fases
- ejemplos: drivers de Oracle, Sybase, SQL server, ...

### 3. El estándar *Open Database Connectivity* (ODBC)

---



# 3. El estándar *Open Database Connectivity* (ODBC)

---

## Configuración del origen de datos:

### 1. Utilización del administrador ODBC

- Panel de control
- Icono ODBC 32bits o ODBC (16bits)

### 2. Añadir un nuevo driver

- el el caso de que el driver del servidor de base de datos no este instalado
- se ha de tener el software de instalación del fabricante

### 3. Añadir una nuevo origen de datos

- elegir un driver ODBC
- pulsar “Agregar”
- configurar el origen de datos

# 3. El estándar *Open Database Connectivity* (ODBC)

---

## Configuración del origen de datos:

### 4. Configurar el origen de datos

- Panel de control
- Icono ODBC 32bits o ODBC (16bits)
- Especificar el origen de datos: depende del driver ODBC

### 5. Datos solicitados en la configuración

- Driver de un sólo paso (dBase, Paradox, Fox Pro, etc.):
  - nombre del origen de datos
  - el camino de acceso.
- Driver de pasos múltiples:
  - identificación del servidor
  - identificación de la base de datos
  - protocolo de comunicaciones

### 3. El estándar *Open Database Connectivity* (ODBC)

---

Acceso a un origen de datos ODBC (Access): tabla vinculada, utilización del motor Jet

#### 1. Vincular el origen de datos

- Menú Archivo, Obtener datos externos:
  - Importar: copiará los datos a tablas Access
  - Vincular: establece una conexión y opera sobre el origen de datos
- Vincular:
  - Tipo de archivo: ODBC de un sólo paso (*dBase*, *Paradox*, etc.) o ODBC de pasos múltiples (*Bases de datos ODBC*)
  - ODBC de un sólo paso: pide el camino
  - ODBC de pasos múltiples:
    - ◊ muestra los orígenes de datos definidos y seleccionar uno
    - ◊ pedirá usuario y palabra de paso
    - ◊ muestra las tablas

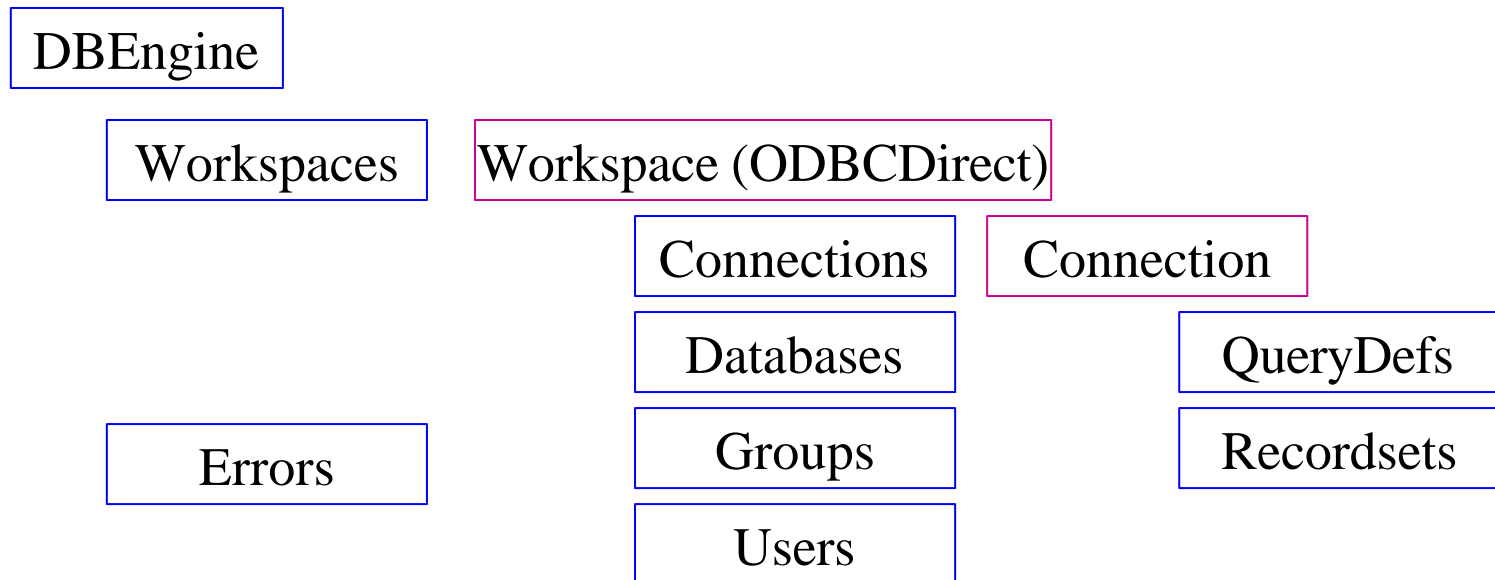
#### 2. Utilizar la tabla vinculada como una tabla Access

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Acceso a un origen de datos ODBC (ODBCDirect)

- No usa el motor Jet
- Está incorporado a la jerarquía de DAO
- Similar al objeto Workspace de Access pero con una colección más, *Connections* de objetos *Connection*





### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Acceso a un origen de datos ODBC (ODBCDirect)

- Se puede programar en VBA haciendo uso de la ODBCDirect
- Más eficiente que usar el ODBC cargando el motor Jet
- Modo de uso:
  - se establece la propiedad *DefaultType* del objeto DBEngine: *dbUseJet* o *dbUseODBC*
  - se crea un área de trabajo con el método *CreateWorkspace* del objeto *DBEngine*
  - se crea una conexión con el método *CreateConnection* del área de trabajo creada en la cual se indica que es una conexión ODBC y cual es el origen
  - se utilizan los métodos del objeto connection creado para crear recordsets

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Acceso a un origen de datos ODBC (ODBCDirect)

- Ejemplo:

```
Dim ws as Workspace, con as Connection, rs as RecordSet
DBEngine.DefaultType = dbUseODBC
Set ws=DBEngine.CreateWorkspace("wrkODBC", "admin", "")
Set con=ws.CreateConnection("mi_conexion", dbDriverComplete, False, _
    "ODBC;DATABASE=;UID=bda;PWD=;DSN=burbuja")
Set rs=con.OpenRecordSet("select count(*) from ciclista",dbOpenSnapShot)
Msgbox rs(0)
rs.Close
con.Close
ws.Close
```

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Propiedades del objeto *Connection*

- *QueryTimeout*: establece o devuelve el número de segundos de espera de la ejecución de la consulta
- *RecordsAffected*:
- *StillExecuting*: indica si la consulta está todavía ejecutándose
- *Transactions*: indica si la conexión permite transacciones
- *Updatable*: indica si los objetos de la conexión son actualizables

### 3. El estándar *Open Database Connectivity* (ODBC)

---

#### Métodos del objeto *Connection*

- *Cancel*: finaliza la consulta que esté en marcha
- *CreateQueryDef*:
- *Execute*:
- *OpenRecordSet*:
- *Updatable*: indica si los objetos de la conexión son actualizables

# 4. Tecnología Web y SGBDs

---

4.1. Introducción a la Internet y la Web

4.2. La Web como una plataforma de aplicaciones de bases de datos

4.3. Aproximaciones para integrar Web y SGBD

4.3.1. *Common Gate Interface* (CGI)

4.3.2. *Server-Side Includes* (SSI)

4.3.3. HTTP Cookies

4.3.4. Extensiones al servidor de Web (NASPI y ISAPI)

4.3.5. Java: JDBC, JSQL, JRB

4.3.6. Lenguajes de *Scripts*: JavaScript, Jscript y VBScript

# 4.1 Introducción a la Internet y la Web

---

## **Internet**

Una colección de ordenadores interconectados en red a lo ancho del mundo

## **Intranet**

Un *sitio* Web o un grupo de sitios de una organización que son accesibles sólo por los miembros de la misma

## **Extranet**

Una intranet que es parcialmente accesible a usuarios externos debidamente autorizados

# 4.1 Introducción a la Internet y la Web

---

## **World Wide Web**

Un sistema con tecnología hipermedia que proporciona una forma de consulta de información mediante hiperenlaces y acceso muy simple

## **HTTP (HyperText Transfer Protocol)**

El protocolo usado para transferir páginas Web a través de Internet

## **HTML (HyperText Markup Language)**

El lenguaje de formateo de documentos utilizado para el diseño de la mayoría de las páginas Web

# 4.1 Introducción a la Internet y la Web

---

## **URL (Uniform Resource Locator)**

Una cadena alfanumérica de caracteres que representa la dirección de un origen en Internet y la forma en la cual se debe acceder a él

## **Página estática**

Un documento HTML almacenado en un fichero. El contenido no cambia a menos que se cambie el fichero

## **Página dinámica**

Un documento HTML que se genera cada vez que se accede a él.  
Características:

- Puede responder a una entrada del usuario
- Puede ser configurado por y para cada usuario



## 4.2. La Web como una plataforma de aplicaciones de BDs

---

### **Requerimientos a una integración Web-SGBD**

- Permita el acceso a información importante de forma segura
- Evite la dependencia de los componentes
- Soporte aproximaciones de arquitecturas abiertas (diferentes servidores de Web, DCOM, CORBA, ...)
- Permita la escalabilidad, crecimiento y cambios en direcciones estratégicas
- Soporte autenticación de sesión y de aplicación
- Existan herramientas que permitan el desarrollo, mantenimiento y puesta en marcha de aplicaciones con sencillez

## 4.2. La Web como una plataforma de aplicaciones de BDs

### Arquitectura de integración Web-SGBD

– Arquitectura tradicional de dos niveles cliente/servidor:

Primer nivel  
*Cliente*



Tareas

- Interfaz de usuario
- Lógica principal del negocio y del procesamiento de datos

Segundo nivel  
*Servidor*



Tareas

- Validación
- Acceso a base de datos



## 4.2. La Web como una plataforma de aplicaciones de BDs

---

### Arquitectura de integración Web-SGBD

- Arquitectura tradicional de dos niveles cliente/servidor: desventajas
  - Los clientes necesitan una potencia considerable
  - Administración del nivel cliente considerable

## 4.2. La Web como una plataforma de aplicaciones de BDs

### Arquitectura de integración Web-SGBD

– Arquitectura tradicional de tres niveles:

Primer nivel  
*Cliente*



Tareas

- Interfaz de usuario

Segundo nivel  
*Servidor de aplicación*



Tareas

- Lógica del negocio
- Lógica del procesamiento de datos

Tercer nivel  
*Servidor de BD*



Tareas

- Validación
- Acceso a base de datos



## 4.2. La Web como una plataforma de aplicaciones de BDs

---

### Arquitectura de integración Web-SGBD

- Arquitectura tradicional de tres niveles: ventajas
  - Los clientes pueden ser menos potentes
  - La centralización de la lógica del negocio conlleva una reducción del mantenimiento de las aplicaciones
  - La modularidad permite la modificación o el cambio de algún nivel sin afectar a los otros
  - La separación de la lógica del negocio y de las funciones de base de datos hace más sencillo el equilibrio del sistema

## 4.2. La Web como una plataforma de aplicaciones de BDs

---

### **Ventajas de la integración Web-SGBD**

- Ventajas de la tecnología de Bases de Datos
- Simplicidad
- Independencia de la plataforma
- Interfaz de usuario gráfica (GUI)
- Estandarización
- Soporte en toda plataforma
- Acceso en la red transparente
- Desarrollo escalable
- Innovación

## 4.2. La Web como una plataforma de aplicaciones de BDs

---

### **Desventajas de la integración Web-SGBD**

- Confianza y robustez
- Seguridad
- Coste
- Escalabilidad
- Funcionalidad limitada del HTML
- Sin soporte de estado
- Anchura de banda: comunicaciones
- Rendimiento: ejecución interpretada
- Herramientas de desarrollo inmaduras

## 4.3. Aproximaciones para integrar Web y SGBD

---

### Aproximaciones

- CGI (Common Gate Inteface)
- SSI (Server-Side Includes)
- HTTP cookies
- Extensiones al servidor de Web (NASPI y ISAPI)
- Java: JDBC, JSQL, JRB
- Lenguajes de *Scripts*: JavaScript, Jscript y VBScript



## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### **Common Gate Interface (CGI)**

Una especificación para transferir información entre un servidor Web y un programa CGI

## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### **Servidor de Web: funcionamiento normal**

- petición de un navegador
- búsqueda de la petición (fichero)
- envío de la información contenida con alguna cabecera de identificación (MIME): permite al navegador la identificación del tipo de información

### **Servidor de Web: ejecución de programas**

- el navegador envía una petición de URL que apunta a un programa (*script*)
- ejecuta el *script*
- devuelve al navegador el resultado de la ejecución como si fuera un fichero

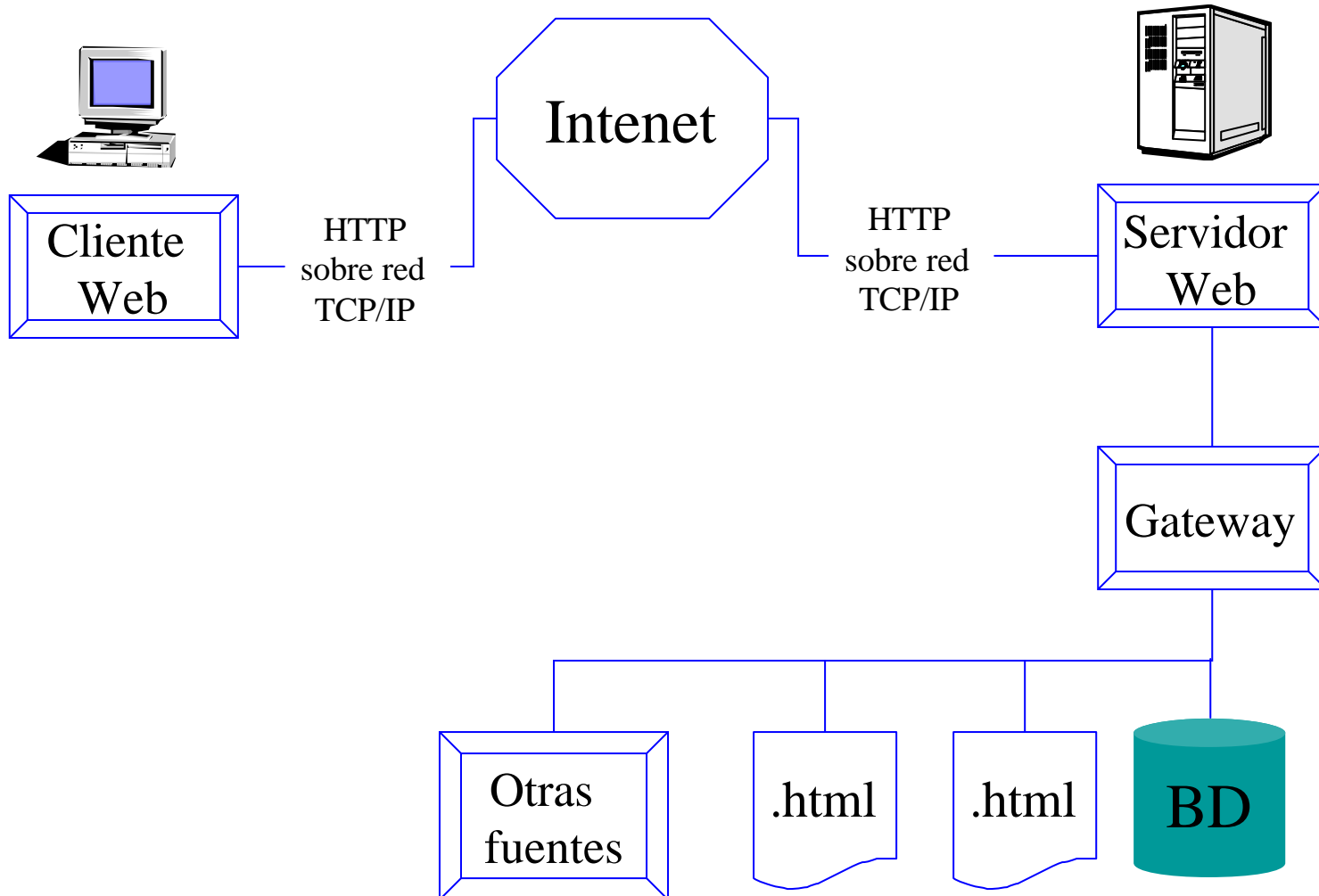
## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### **Common Gate Interface (CGI)**

- Define cómo los scripts deben comunicarse con servidores Web
- Un script CGI es un script diseñado para aceptar y devolver datos que son conformes con una especificación CGI

## 4.3. Aproximaciones para integrar Web y SGBD: CGI



## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### **Secuencia de operaciones en la llamada a un CGI**

1. Usuario llama al script CGI pulsando un vínculo o un botón (también se puede invocar en la carga del documento HTML)
2. Navegador contacta con el servidor solicitando el permiso de ejecutar el script
3. El servidor comprueba la configuración y la seguridad (existe el script y el cliente tiene permiso de ejecución)
4. El servidor prepara las variables de entorno y ejecuta el script
5. El script lee las variables de entorno y STDIN
6. El script envía la cabeceras MIME y la salida al STDOUT y termina
7. El servidor envía los datos en el STDOUT al navegador y cierra la conexión
8. El navegador visualiza la información enviada por el servidor

## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### **Implementación del script CGI**

- Casi en cualquier lenguaje de programación
- Capacidad de leer y escribir en las variables de entorno del sistema operativo
- Perl, C, Forth, ... (UNIX)
- DOS batch, Visual B, C/C++, Delphi, NT Perl, ... (Windows)

## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### **Métodos para el paso de información del navegador al script**

- pasando los parámetros en la línea de comando
- pasando variables de entorno al script CGI
- pasando datos al script CGI vía el entrada estándar
- usando información de caminos con datos añadidos

### **Tipos de resultados que devuelve el script CGI al navegador**

- documento HTML (html)
- documento texto plano (txt)
- imagen (gif)
- datos multimedia ( jpeg, png, ps, avi, mov, mpeg, ...)

## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### Ventajas del CGI

- Estándar de *facto* para crear interfaces entre servidores Web y aplicaciones externas
- Actualmente es el método más usado
- Simplicidad
- Independencia del lenguaje de implementación
- Independencia del servidor Web
- Amplia aceptación



## 4.3. Aproximaciones para integrar Web y SGBD: CGI

---

### Desventajas del CGI

- Comunicación entre el cliente (navegador) y el SGBD a través de un script CGI requiere que el servidor Web actúe como puente para la conversión de los datos que el script obtiene de la base de datos a HTML para el navegador (cuello de botella, sobrecarga)
- Falta de eficiencia y de soporte de transacciones: uso de CGI hereda la ausencia de estado del protocolo HTTP: para cada requerimiento una conexión abierta y cerrada
- Validación de datos (ausencia de estado): vuelta atrás
- Tratamiento de volúmenes grandes de información
- Arquitectura de ejecución (por cada script CGI usado un proceso en marcha): sobrecarga del servidor, concurrencia, accesos a los mismos datos, etc.
- Seguridad: scripts CGI pueden llamar a procesos en el servidor ...

## 4.3. Aproximaciones para integrar Web y SGBD: SSI

---

### Server-Side Includes (SSI)

- Servidores que realizan un análisis del fichero que se va a mandar y ejecutan comandos incluidos en él
- No existe un estándar (se suele seguir NCSA)
- Los comandos SSI se embeben en comentarios HTML
- `<!-- #comando SSI -->`
- Seguridad: mismos riesgos que CGI

## 4.3. Aproximaciones para integrar Web y SGBD: Cookies

---

### HTTP cookies

- Forma de conseguir que los scripts CGI sean más interactivos
- *Cookies*: pequeños ficheros de texto que se almacenan en el cliente
- Forma de trabajo:
  1. Script CGI crea la *cookie*
  2. El servidor Web se la envía al navegador
  3. El navegador la almacena en el disco duro del cliente
  - ...
  4. Vuelve a visitar el sitio
  5. Usa un CGI que requiere una *cookie*
  6. El navegador comprueba si la tiene, y en caso afirmativo la envía

## 4.3. Aproximaciones para integrar Web y SGBD: Cookies

---

### HTTP cookies

- Se usan para mantener información de registro de conexión, de compras, etc.
- Formato de una *cookies*:

Set-Cookie: NAME= *VALUE*; expires= *DATE*; path= *PATH*  
[domain = *DOMAIN-NAME*; secure]

## 4.3. Aproximaciones para integrar Web y SGBD: NASPI y ISAPI

---

### *non-CGI Gateways*

- Aproximación CGI presenta inconvenientes en el uso de recursos compartidos y en el rendimiento
- API que proporcionan algunos servidores: Netscape Server API (NSAPI) y Microsoft Internet Information Server API (ISAPI)
- Evita la creación de procesos externos para cada script CGI, crea una interfaz entre el servidor y las aplicaciones cliente usando enlaces dinámicos o objetos compartidos
- Los script se cargan como parte del servidor y pueden tener acceso a toda su funcionalidad
- Sólo se carga una copia de cada script que puede ser compartida por diferentes peticiones al servidor

## 4.3. Aproximaciones para integrar Web y SGBD: NASPI y ISAP

---

### *non-CGI Gateways: ventajas*

- Mejor rendimiento
- Menos consumo de recursos
- Flexibilidad del formato de la información recibida y servida
- Aumento de la seguridad

## 4.3. Aproximaciones para integrar Web y SGBD: NASPI y ISAP

---

### *non-CGI Gateways: problemas*

- Exige programadores expertos en el servidor y en técnicas de programación con sincronización, protocolos, manejo de excepciones, ...
- Dañar la robustez del servidor Web
- Portabilidad: el API es propietaria de una empresa

## 4.3. Aproximaciones para integrar Web y SGBD: JAVA

---

### JAVA

- Lenguaje sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, independiente de la arquitectura, portable, con alto rendimiento y dinámico
- Máquina Virtual Java (JVM): *escribe una vez, ejecuta donde sea*
- Compilado: *en instrucciones bytecode*
- JVM: interpreta y ejecuta en la plataforma y SO cliente
- Acceso a SGBD desde Java: JDBC, JSQL y JRB



## 4.3. Aproximaciones para integrar Web y SGBD: JAVA

---

### JDBC

- Aproximación más madura y emergente para el acceso a SGBD desde Java
- Filosofía similar a la tecnología ODBC
- JDBC: define una API de acceso a SGBD de tipo SQL
- Aproximación actual: SQL embebido para Java
  - instrucciones SQL se pasan como cadenas a los métodos Java
  - preprocesador de SQL embebido permite trabajar directamente con Java
  - variables Java se usan para pasar y recibir valores

## 4.3. Aproximaciones para integrar Web y SGBD: JAVA

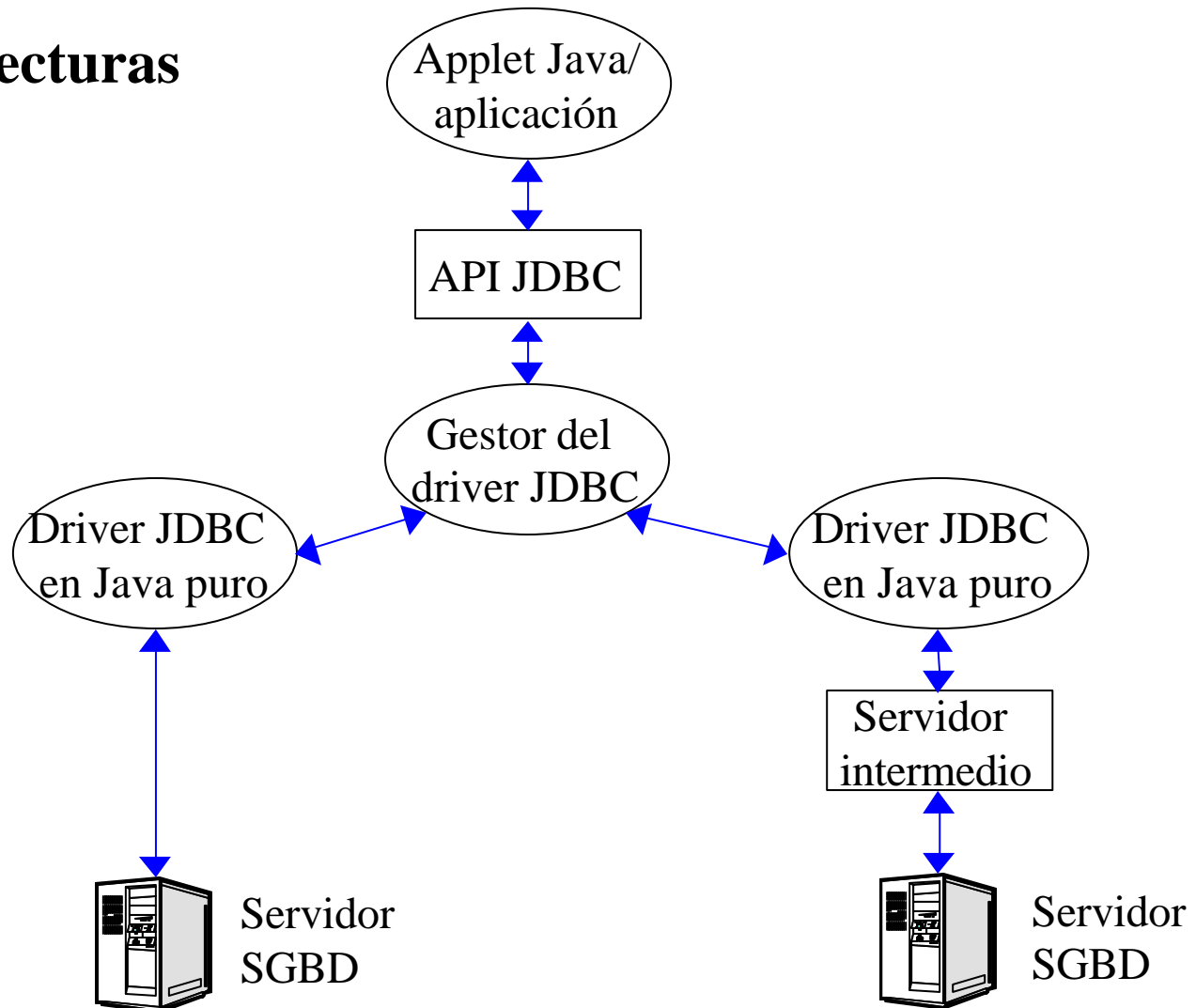
---

### JDBC: arquitecturas

- *driver* en Java puro con conexión directa al SGBD: convierte las llamadas JDBC al protocolo de red usado por el SGBD
- *driver* en Java puro con uso de un servidor intermedio: convierte las llamadas JDBC al protocolo del servidor intermedio. Este traduce éstas al protocolo de red del SGBD correspondiente. El servidor intermedio proporciona conectividad a diferentes SGBD

## 4.3. Aproximaciones para integrar Web y SGBD: JAVA

### JDBC: arquitecturas



## 4.3. Aproximaciones para integrar Web y SGBD: JAVA

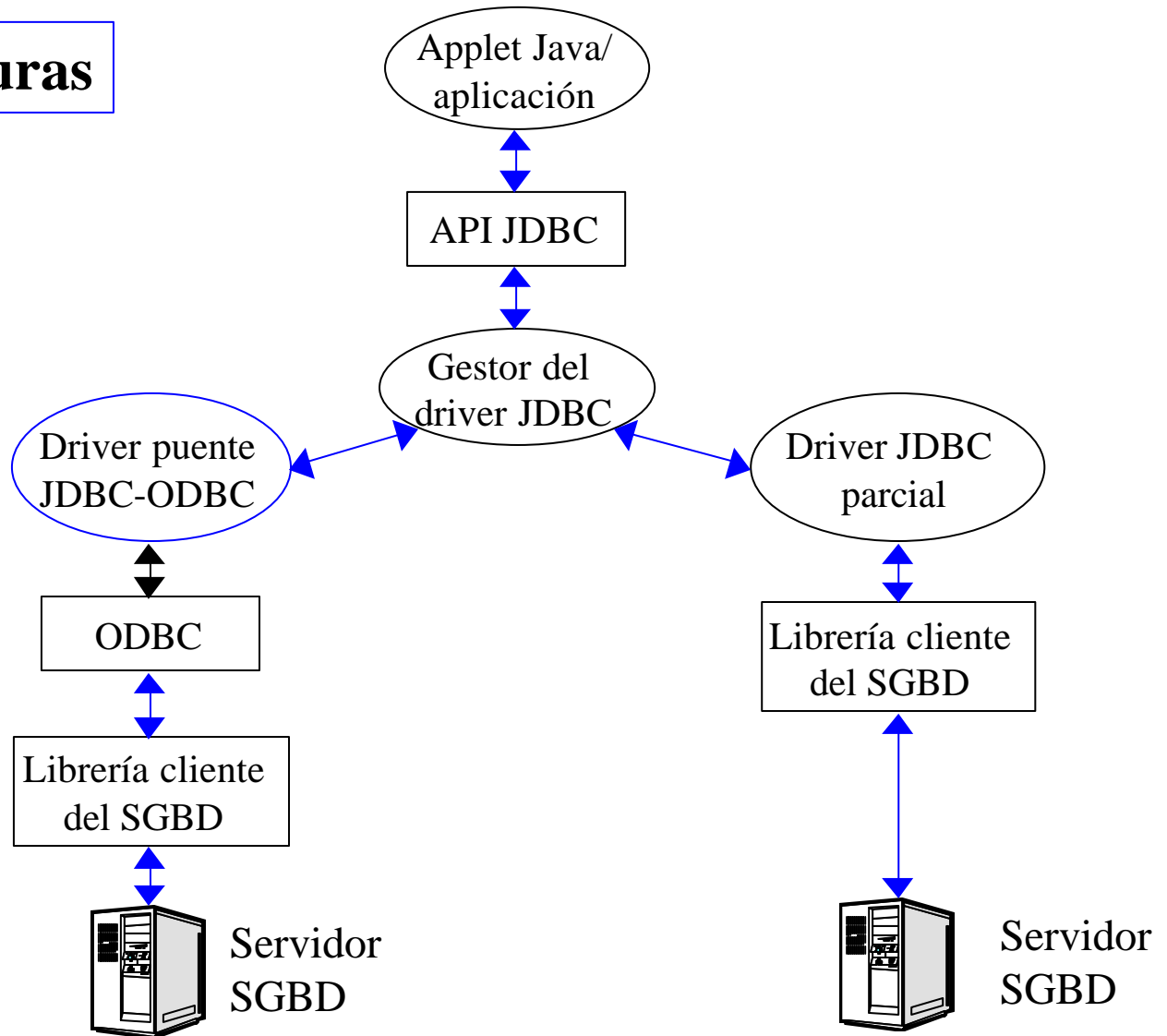
---

### **JDBC: arquitecturas**

- *driver* JDBC parcial: convierte las llamadas JDBC en llamadas al API cliente del SGBD (requiere instalación de software cliente)
- puente JDBC-ODBC: proporciona un acceso JDBC a través de los drivers ODBC (requiere instalar los drivers ODBC y software cliente)

## 4.3. Aproximaciones para integrar Web y SGBD: JDBC

### JDBC: arquitecturas



## 4.3. Aproximaciones para integrar Web y SGBD: JAVA

---

### JSQL

- Aproximación que usa Java con SQL embebido (Oracle, IBM y Tandem)
- JSQL es el conjunto de cláusulas que extienden Java para incluir constructores SQL
- Un traductor JSQL (precompilador) transforma las cláusulas JSQL en código Java estándar que accede a la base de datos a través de una interfaz de llamada

## 4.3. Aproximaciones para integrar Web y SGBD: JAVA

---

### **JRB (Java Relational Binding)**

- Producto intermedio que establece un puente entre Java y los sistemas relacionales
- Proporciona persistencia ortogonal a Java, mediante un proceso de tres etapas:
  - creación de la base de datos
  - programa de importación
  - API JRB

### **Lenguajes de scripts: JavaScript, Jscript y VisualScript**

- Aumentan la potencia y expresividad del HTML
- Son interpretados
- Pueden ser ejecutados en el navegador cliente o en el servidor
- JavaScript y Jscript (Microsoft) son una simplificación Java
- VisualScript proviene del Visual Basic
- Existen un conjunto de instrucciones cuando se ejecuta los script en el servidor que dotan de funcionalidad de manipulación de base de datos
  - Conexión y desconexión de una base de datos
  - Iniciar, confirmar y anular una transacción SQL
  - Mostrar el resultado de una consulta SQL
  - Crear cursores actualizables para consultar, insertar, borrar y modificar datos
  - ...