



BASES DE DATOS AVANZADAS

Tema 6

Bases de Datos Distribuidas

Univ. Cantabria – Fac. de Ciencias

Francisco Ruiz



Objetivos

- Dar a conocer las principales **arquitecturas de los sistemas de BD**.
- Presentar las principales características y diferencias de las arquitecturas **centralizadas** frente a las **cliente/servidor**.
- Dar una visión de las arquitecturas **distribuidas**.
- Introducir las **arquitecturas** para bases de datos **Web**.



Contenido

- Introducción
 - Paradigmas Arquitecturales en BD
- Arquitecturas Cliente/Servidor
- Bases de Datos Distribuidas
 - Clases
 - Sistemas de Gestión de BDD
 - Reglas de Date
 - Tipos de Servidores
 - Independencia Local
 - Fragmentación y Localización de Datos
- Arquitecturas para la Web
 - Transparencia para las Aplicaciones
 - Interoperabilidad
 - Cooperación
 - Replicación
- Arquitecturas para la Web
 - Arquitectura de Tres Capas Web-SGBD
 - Integración Web-SGBD



Bibliografía

- Básica
 - Connolly y Begg (2005): Sistemas de Bases de Datos.
 - Cap. 22.
- Complementaria
 - Elmasri y Navathe (2007): Fundamentos de Sistemas de Bases de Datos.
 - Cap. 25.



Paradigmas Arquitecturales de Sistemas de BD

- **Arquitectura Centralizada**: los datos y las aplicaciones están en una única máquina.
- **Arquitectura Cliente-Servidor**: separación del servidor de BD del cliente.
- **BD Distribuida**: varios servidores de BD usados por la misma aplicación.
- **BD Paralelas**: varias unidades de almacenamiento de datos y procesadores operan en paralelo para incrementar el rendimiento.
- **BD Replicadas**: datos lógicamente representando la misma información están almacenados físicamente en diferentes servidores
- **Almacenes de Datos**: servidores especializados en la gestión de datos orientados al soporte a la decisión.

- Las nuevas arquitecturas de **BD para la Web** son variantes del paradigma general cliente-servidor.



Distintas Funcionalidades

- Sistemas **OLTP** (On-Line Transaction Processing):
 - Gestión optimizada y fiable de **transacciones** en **servidores de BD**, especializados para soportar cientos o miles de transacciones por segundo.
- Sistemas **OLAP** (On-Line Analytical Processing):
 - **Análisis de datos**, operando sobre **servidores de almacenes de datos** (data warehouse), especializados en la gestión de datos para el soporte a la toma de decisiones.



Propiedades Deseables

- En los **Sistema de BD** actuales, caracterizados por la necesidad de interacción con otros sistemas, son altamente deseables las siguientes propiedades:
 - **Portabilidad**: posibilidad de transportar aplicaciones de un entorno a otro.
 - Facilitada por **lenguajes estándares**:
 - SQL.
 - **Interoperabilidad**: habilidad para interactuar entre sistemas heterogéneos.
 - Facilitada por **protocolos estándares de acceso a datos**:
 - ODBC (Open Database Connectivity)
 - DTP (X-Open Distributed Transaction Processing)



Arquitecturas Cliente-Servidor

- **Cliente-Servidor**: modelo general de **interacción entre procesos software** donde los procesos que interactúan están divididos en:
 - **Clientes** (requieren servicios), y
 - **Servidores** (ofrecen los servicios).
- Requiere una definición precisa de un **interfaz de servicios**, que especifica la lista de servicios ofrecidos por el servidor.
- El proceso **cliente** desempeña un papel **activo** mientras que el proceso **servidor** es **reactivo**.
- Habitualmente, un proceso cliente solicita de forma secuencial unos pocos servicios de uno o varios servidores, mientras que un proceso servidor responde a multitud de peticiones de muchos procesos clientes.

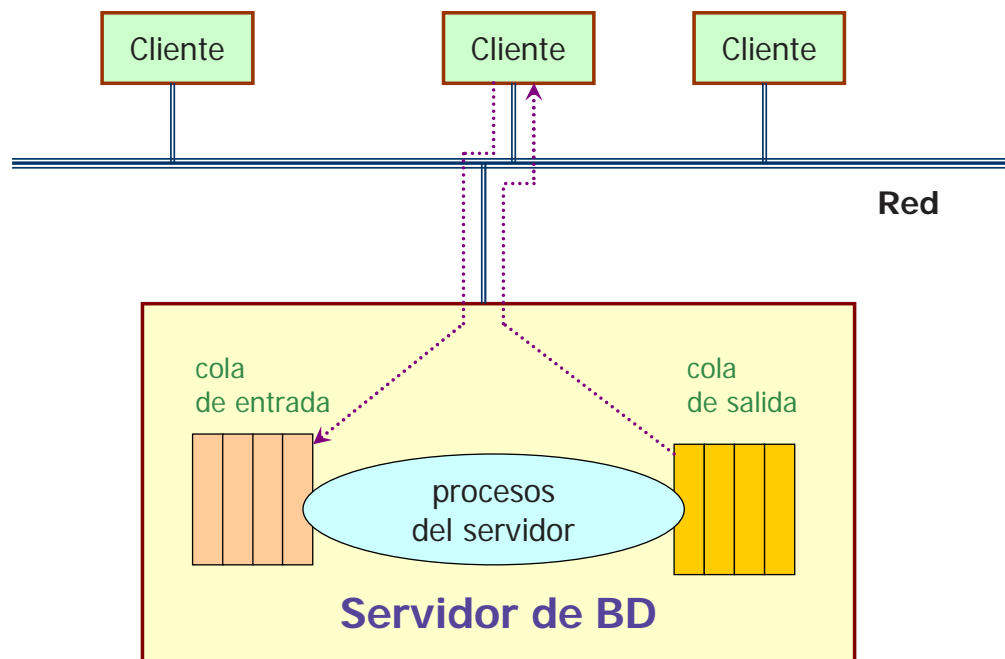


Arquitecturas Cliente-Servidor

- **Ventajas** de su uso en BD:
 - Las **funciones** del cliente y del servidor están **bien identificadas**.
 - El **uso de máquinas diferentes** para el cliente y para el servidor es particularmente **conveniente** en el ámbito de las BD:
 - **Máquina cliente**: orientada a la interacción con el usuario y al soporte a las herramientas de productividad (email, textos, hojas de cálculo, ...)
 - **Máquina servidor**: Debe tener una gran memoria principal (para soportar la gestión de buffers) y una alta capacidad de disco (para almacenar la BD completa).
 - **SQL** ofrece un paradigma de programación **ideal** para la identificación del "**interfaz de servicio**":
 - Las consultas SQL son formuladas por el cliente y enviadas al servidor
 - Los resultados de las consultas son calculados por el servidor y devueltos al cliente
 - La estandarización, portabilidad e interoperabilidad de SQL permite construir aplicaciones cliente que interactúan con diferentes servidores.



Arquitecturas Cliente-Servidor





Arquitecturas Cliente-Servidor

Multi-hebra

- Frecuentemente, el servidor utiliza una **arquitectura multi-hebra (multi-threaded)** para gestionar las peticiones de los clientes.
 - Desde el punto de vista del sistema operativo, se comporta como un único proceso que trabaja de forma dinámica para atender diferentes transacciones.
 - Pero internamente tiene una **unidad de ejecución**, denominada **hebra (thread)**, para atender a cada una de las transacciones.
 - Los **procesos servidores están permanentemente activos** para controlar una cola de entrada de peticiones de los clientes y una cola de salida con los resultados de las consultas.
 - Los servidores gestionan estas colas de forma directa o por medio de otros procesos, llamados **despachadores (dispatchers)**, que distribuyen las peticiones entre los servidores y retornan las respuestas a los clientes adecuados.
 - Algunas veces los despachadores pueden establecer de forma dinámica el número de procesos servidor activos en función del número de peticiones recibidas (igual que las cajas abiertas en un hipermercado).



Arquitecturas Cliente-Servidor

Dos Capas y Tres capas

- Existen dos versiones de Cliente-Servidor:
 - **Arquitectura de dos capas:**
 - El cliente es a la vez el interfaz de usuario y el gestor de la aplicación.
 - Se llama **Thick-Client** (cliente grueso) porque soporta la lógica de la aplicación.
 - **Arquitectura de tres capas:**
 - Incorpora un segundo servidor llamado **servidor de aplicaciones**, responsable de gestionar la lógica de aplicación común a muchos clientes.
 - El cliente se llama **Thin-Client** (delgado) porque sólo es responsable del interfaz con el usuario final (que puede ser implementado usando navegadores web).
 - En esta categoría se incluyen varias propuestas de arquitecturas para aplicaciones web.



Bases de Datos Distribuidas

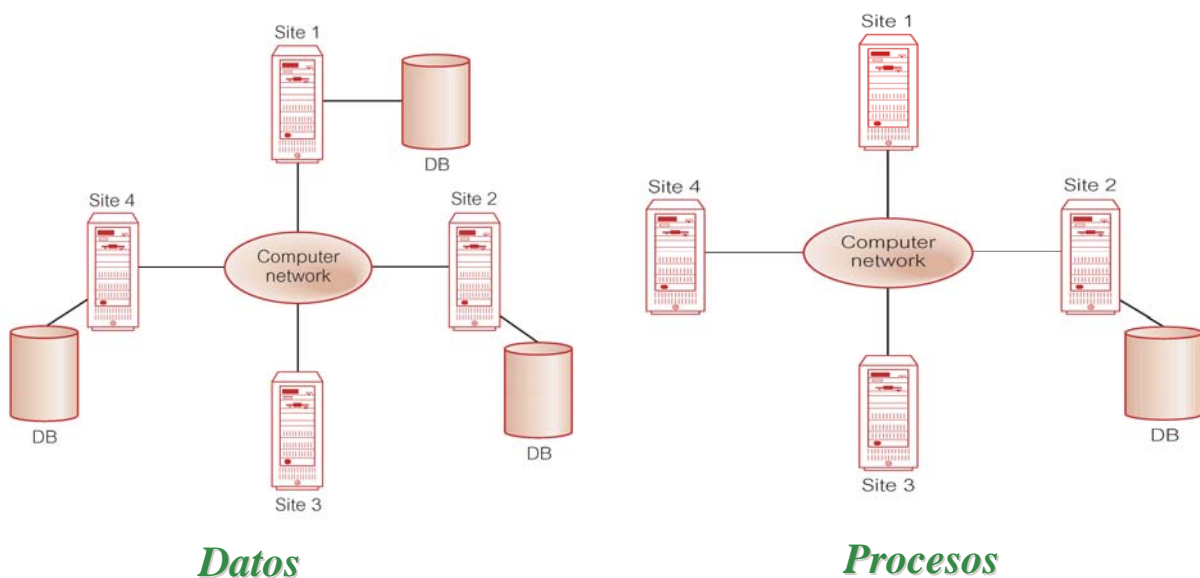
- Una **Base de Datos Distribuida**

- Es una colección de múltiples y lógicamente relacionadas bases de datos sobre una red de ordenadores.
- Un SGBD distribuido se define como el software que permite gestionarla y hacer la distribución transparente a los usuarios.
- Es una BD almacenada en varios ordenadores que se comunican mediante una red de comunicaciones.
- El usuario debe poder usarla como un sistema único.
- Puede procesar todo tipo de peticiones complejas.
- Las peticiones se pueden procesar en el sitio que hizo la petición o en cualquier otro o parcialmente en varios.
- Necesita una gestión de transacciones especial.
- Debe proporcionar optimización de peticiones automáticamente.



Bases de Datos Distribuidas

- **BD Distribuida vs Procesamiento Distribuido**





Clases

- Las podemos clasificar según dos criterios:
 - **Homogéneas o heterogéneas** según todos los servidores utilicen el mismo SGBD o no, y
 - **De área local o de área ancha** según sea la red de comunicaciones que conecta los servidores.

Tipo SGBD	Tipo de Red	
	LAN	WAN
Homogéneo	Gestión de datos y aplicaciones financieras	Gestión de viajes y aplicaciones financieras
Heterogéneo	Sistemas de Información inter-divisionales	Banca integrada y sistemas inter-bancos

Ejemplos de aplicaciones de diversos tipos de BDD



Sistemas de Gestión de BDD

- En un **Sistema de Gestión de BD Distribuidas** (SGBDD) al menos un cliente interactúa con **varios servidores** para la ejecución de una misma aplicación.
- **Ventajas:**
 - Responder mejor a las **necesidades de las aplicaciones**.
 - La mayoría de las empresas son estructuralmente distribuidas y la gestión y control de sus datos también suelen ser distribuidos.
 - Mayor **flexibilidad y modularidad**.
 - Los sistemas distribuidos pueden ser configurados para la adición progresiva y modificación de componentes.
 - Mayor **resistencia a fallos**.
 - Aunque son más vulnerables a los fallos debido a su mayor complejidad estructural, soportan la degradación suave, es decir, responden a los fallos con una reducción del rendimiento pero no con un fallo total.



Bases de Datos Distribuidas

Sistemas de Gestión de BDD

- Un SGBD Distribuidas (**SGBDD**) amplía la funcionalidad de un SGBD normal con:
 - Servicios de Comunicación Extendidos.
 - Diccionario de Datos Extendido.
 - Procesamiento de Consultas Distribuido.
 - Control de Concurrencia Extendido.
 - Servicios de Recuperación Extendidos.



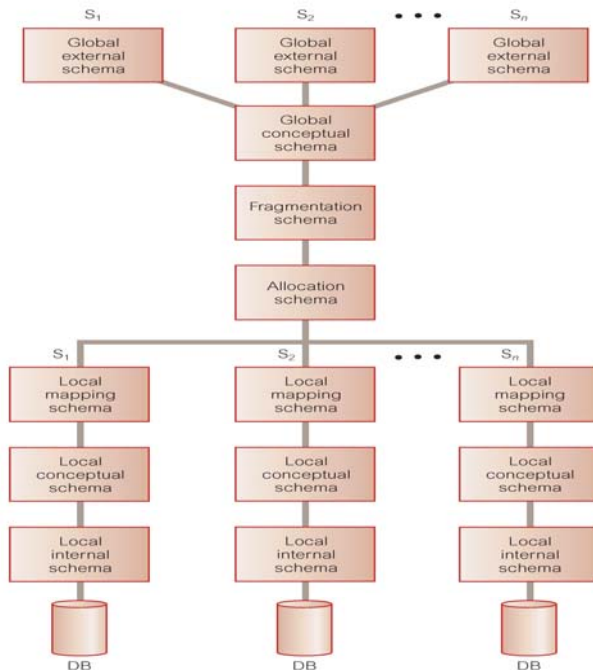
Bases de Datos Distribuidas

Sistemas de Gestión de BDD

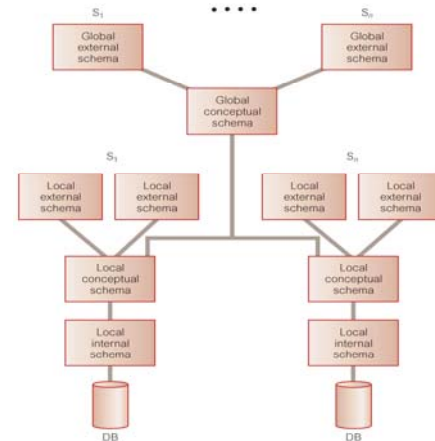
- No hay un estándar para la **arquitectura de SGBDD**.
- Los **tipos de elementos** de una arquitectura de BDD son:
 - Conjunto de Esquemas Externos Globales (GES).
 - Un Esquema Conceptual Global (GCS).
 - Esquema de Fragmentación.
 - Esquema de Localización.
 - Conjunto de esquemas para cada SGBD local (según estándar ANSI/SPARC).
 - depending on levels of transparency supported.



Bases de Datos Distribuidas Sistemas de Gestión de BDD



- El Esquema Conceptual Global (GCS) es la unión de todos los esquemas conceptuales locales.
- Algunos tipos de elementos pueden no aparecer



Multi-BD Federadas

Francisco Ruiz - BDA

6.19



Bases de Datos Distribuidas Reglas de Date

- Principio fundamental:
 - *Para el usuario un sistema distribuido (SD) debe funcionar igual que si no fuera distribuido.*
- 1. **Autonomía local:** los sitios de un SD deben ser autónomos en el mayor grado posible.
- 2. **No dependencia de un sitio central:** Todos los sitios deben ser tratados como iguales.
- 3. **Operación continua:** El SD debe aumentar la **confiabilidad** (probabilidad de que esté listo en un momento dado) y la **fiabilidad** (probabilidad de que esté listo en un periodo largo de tiempo). Los apagados planeados nunca deberían ser necesarios.
- 4. **Independencia de localización:** para el usuario la **localización física** de los datos debe ser transparente.
- 5. **Independencia de fragmentación:** los usuarios no necesitan conocer los **fragmentos físicos** en que está dividida cada colección lógica de datos.
- 6. **Independencia de replicación:** a nivel lógico los usuarios no necesitan tener en cuenta si los datos tienen **réplicas** o no.

Francisco Ruiz - BDA

6.20



Reglas de Date

7. **Procesamiento de consultas distribuidas:** el SD debe disponer de mecanismos para **optimizar las consultas** y en el especial para reducir la carga de tráfico necesaria.
8. **Gestión de transacciones distribuidas:** el SD debe disponer de mecanismos (protocolos) adecuados para el **control de concurrencia** y la **recuperación** de transacciones distribuidas.
9. **Independencia del hardware:** poder ejecutar el mismo SGBD en sitios con diferentes plataformas hardware.
10. **Independencia del sistema operativo:** poder ejecutar el mismo SGBD en sitios con diferentes sistemas operativos.
11. **Independencia de la red:** el SD debe poder operar con diferentes redes de comunicaciones.
12. **Independencia del SGBD:** Debe permitirse la **heterogeneidad**, es decir, que cada sitio puede funcionar con un SGBD diferente, incluso basado en un modelo de datos diferente, siempre y cuando compartan un **interfaz común**.



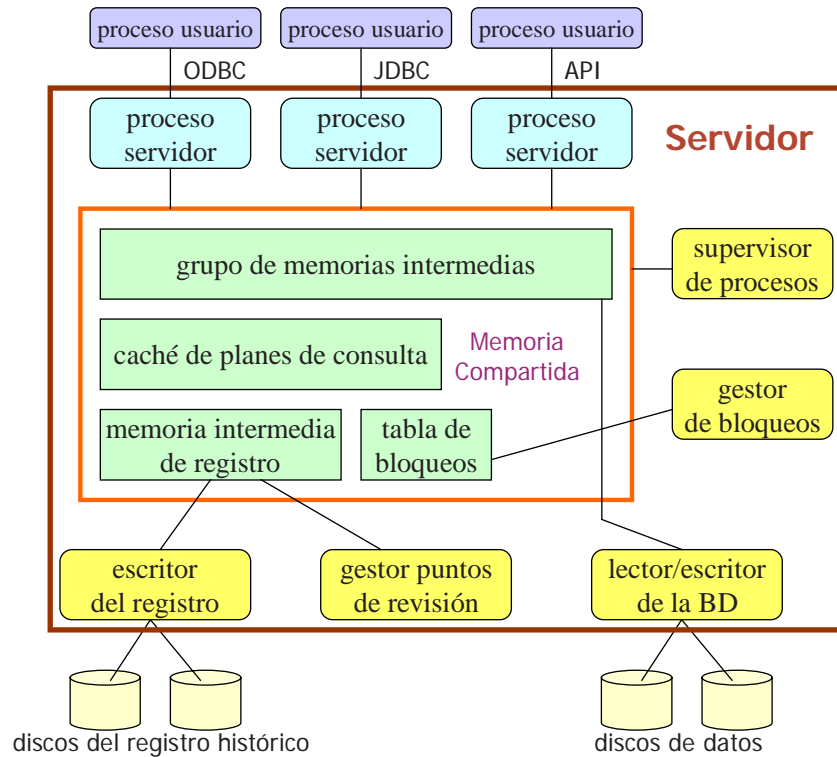
Tipos de Servidores

- **Servidores de Datos:**
 - Los clientes interactúan con los servidores realizando **peticiones de lectura o escritura de datos** en unidades de datos de diversa granularidad (archivos, páginas, tablas, registros, objetos, etc.).
- **Servidores de Transacciones:**
 - También llamados **servidores de consultas**.
 - Proporcionan una interfaz para que los clientes envíen **peticiones para realizar una acción (transacción)** que el servidor ejecutará y devolverá los resultados al cliente.
 - Este tipo es mucho más potente y mucho más utilizado que el anterior.



Bases de Datos Distribuidas Tipos de Servidores

Estructura de memoria y procesos de un servidor de transacciones



Bases de Datos Distribuidas Tipos de Servidores

- **Procesos servidor:** son procesos que reciben consultas del usuario, las ejecutan, y devuelven los resultados.
- **Proceso gestor de bloqueos:** se encarga de la concesión de bloqueos, liberación de bloqueos y detección de interbloqueos.
- **Proceso escritor de BD:** hay uno o más procesos que vuelcan al disco los bloques de memoria intermedia modificados de forma continua.
- **Proceso escritor del registro:** genera entradas del registro en el almacenamiento estable a partir de la memoria intermedia del registro.
- **Proceso punto de revisión:** este proceso realiza periódicamente puntos de revisión.
- **Proceso monitor de proceso:** observa a otros procesos y, si cualquiera de ellos falla, realiza acciones de recuperación para el proceso, tales como cancelar cualquier transacción que estuviera ejecutando el proceso fallido, y reinicia el proceso.



Tipos de Servidores

Todos los procesos de la base de datos pueden acceder a los datos de la memoria compartida, que contiene:

- Grupo de memorias intermedias
- Tabla de bloqueos
- Memoria intermedia del registro, que contiene las entradas del registro que esperan a ser volcadas en el almacenamiento estable
- Planes de consulta en caché, que se pueden reutilizar si se envía de nuevo la misma consulta



Independencia Local

- En una BDD cada servidor tiene su propia capacidad para gestionar aplicaciones de forma independiente.
- La **independencia de cada servidor** es un objetivo importante en los SGBDD.
 - Una BDD no debe implicar maximizar la interacción y la necesidad de transmitir datos por la red de comunicaciones.
 - Al contrario, la planificación de la distribución y localización de los datos debe hacerse buscando que el mayor número de aplicaciones puedan operar de forma independiente en un único servidor.



Fragmentación y Localización de Datos

- La **Fragmentación de Datos** es una técnica para organización de datos que permite la distribución y el procesamiento eficientes.
 - Sólo es aplicable si la distribución de los datos sigue un patrón bien conocido, que es tomado en cuenta durante el diseño de la BDD.
 - Dada una colección de elementos de datos R , su fragmentación consiste en establecer **fragmentos** R_i aplicando a R diversas operaciones algebraicas.
 - Hay dos **tipos de fragmentación**:
 - **Horizontal**: cada fragmento R_i incluye un subconjunto de los elementos de datos de R .
 - **Vertical**: cada fragmento incluye todos los elementos de R pero sólo algunas de sus propiedades (atributos).
 - Es **correcta** si es
 - **Completa**: cada elemento de datos de R debe estar presente en uno de sus fragmentos R_i .
 - **Restaurable**: el contenido de R puede ser regenerado a partir de sus fragmentos.



Fragmentación y Localización de Datos

- Cada **fragmento** R_i corresponde a un **archivo físico diferente** que está localizado en un **servidor diferente** =>
 - La colección de elementos de datos R sólo existe virtualmente pero no físicamente.
 - Los fragmentos existen físicamente.
 - El **esquema de localización** (allocation schema) describe los vínculos (mapping) entre las colecciones completas o los fragmentos y los servidores, es decir, entre la descripción lógica de los datos y su descripción física en archivos de servidores.
 - Esta vinculación puede ser:
 - **Redundante**, si al menos un fragmento o colección está localizado en más de un servidor.
 - **No Redundante**, en caso contrario.



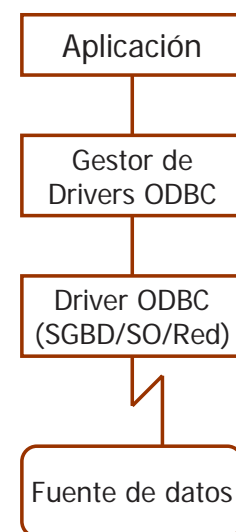
Transparencia para las Aplicaciones

- Una **BDD debe ser transparente** para las aplicaciones:
 - Las aplicaciones deben utilizarla como si fuera no distribuida, sin preocuparse por la fragmentación o localización de los datos.
- De mayor a menor, los **niveles de transparencia** que puede haber son:
 - **De fragmentación:** el programador no necesita saber si la BD es distribuida o está fragmentada.
 - **De localización:** el programador debe conocer los fragmentos que existen pero no necesita indicar su localización.
 - **De lenguaje:** el programador debe conocer los fragmentos y sus localizaciones, pero interactúa con todos los servidores con el mismo lenguaje.
 - **Ausencia de transparencia:** cada servidor utiliza un lenguaje o dialecto particular.



Interoperabilidad

- Es la principal dificultad en el desarrollo de **aplicaciones heterogéneas para BDD**, es decir, cuando los servidores utilizan **más de un SGBD diferente**.
 - Significa capacidad para la interacción entre diferentes SGBDs.
 - Requiere la disponibilidad de funciones para la adaptación y conversión, de forma que sea posible el intercambio de información entre los diversos SGBDs.
 - Es posible gracias a la existencia de **estándares adecuados** (tema 9):
 - ODBC
 - JDBC
 - ...





Cooperación

- Se plantea al intentar **utilizar sistemas de BD pre-existentes**.
- Es la capacidad de las aplicaciones de un determinado sistema de utilizar los servicios provistos por las aplicaciones de otro sistema, incluso de organizaciones diferentes.
- La **cooperación es necesaria en BD** porque:
 - La integración de BD es bastante difícil. Los objetivos de integración o estandarización ambiciosos siempre han fracasado.
 - El modelo ideal de una BD altamente integrada que puede ser consultada de forma transparente y eficiente es imposible de desarrollar y gestionar.



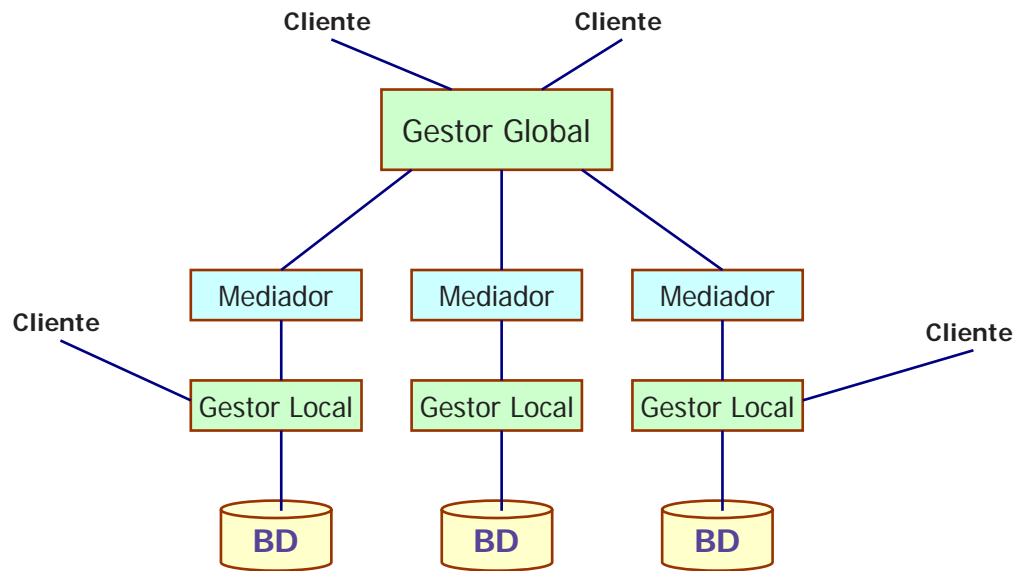
Cooperación

- Hay de dos clases:
 - **Centrada en los procesos**: los sistemas ofrecen servicios a los otros sistemas por medio de intercambio de mensajes, información o documentos, o el disparo de actividades, sin hacer visibles "sus datos" a los demás sistemas.
 - **Centrada en los datos**: *hacer accesibles desde localizaciones remotas, por medio de algún acuerdo de cooperación, datos distribuidos, heterogéneos y autónomos.*
 - Características:
 - **Nivel de transparencia**, grado de enmascaramiento de la distribución y heterogeneidad de los datos)
 - **Complejidad de las operaciones distribuidas**, grado de coordinación necesario para realizar operaciones entre BD's coordinadas.
 - **Nivel de actualización**, indica si los datos accesibles están actualizados o no (réplicas que se actualizan periódicamente).
 - Según se consideren estos tres criterios, existen **tres arquitecturas para la cooperación de BD**.



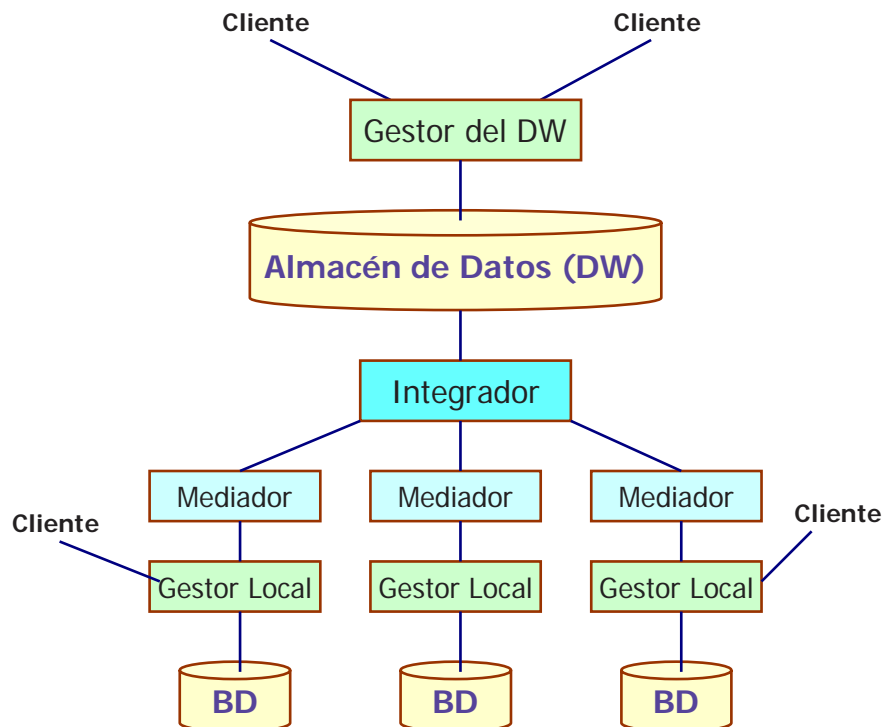
Bases de Datos Distribuidas

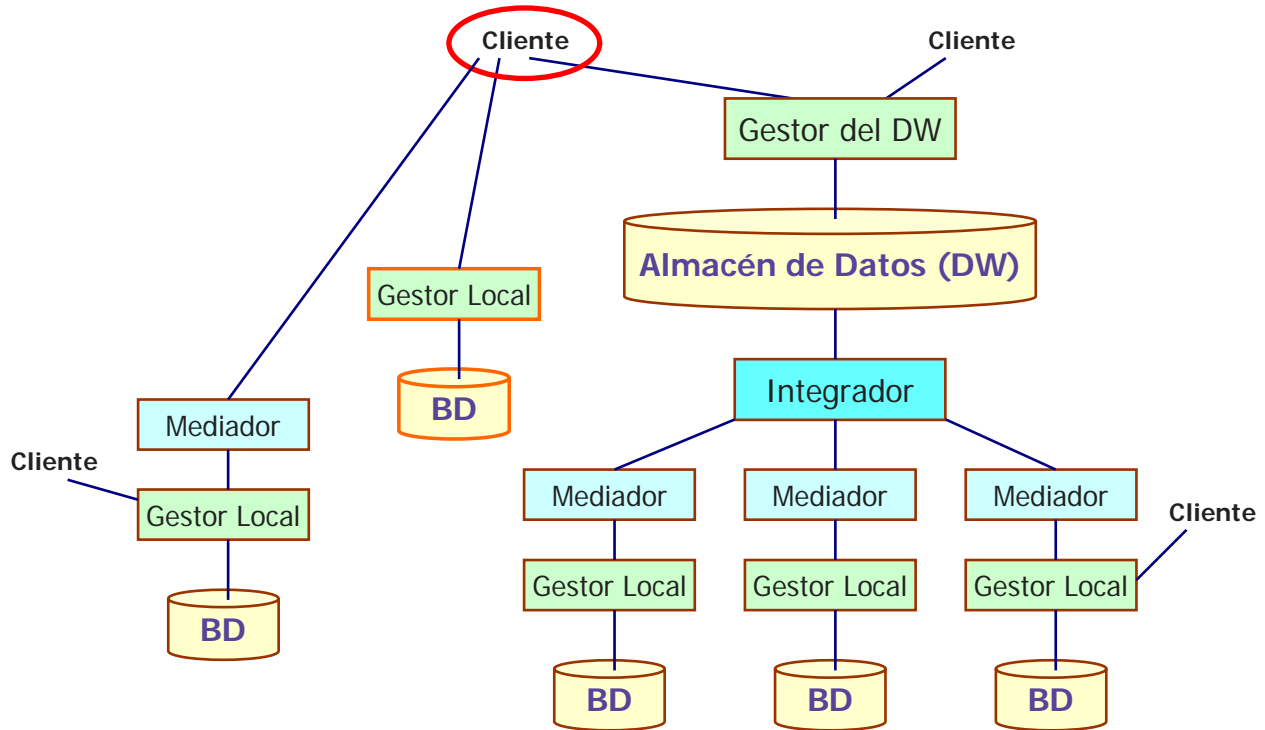
Cooperación – Multi-BD



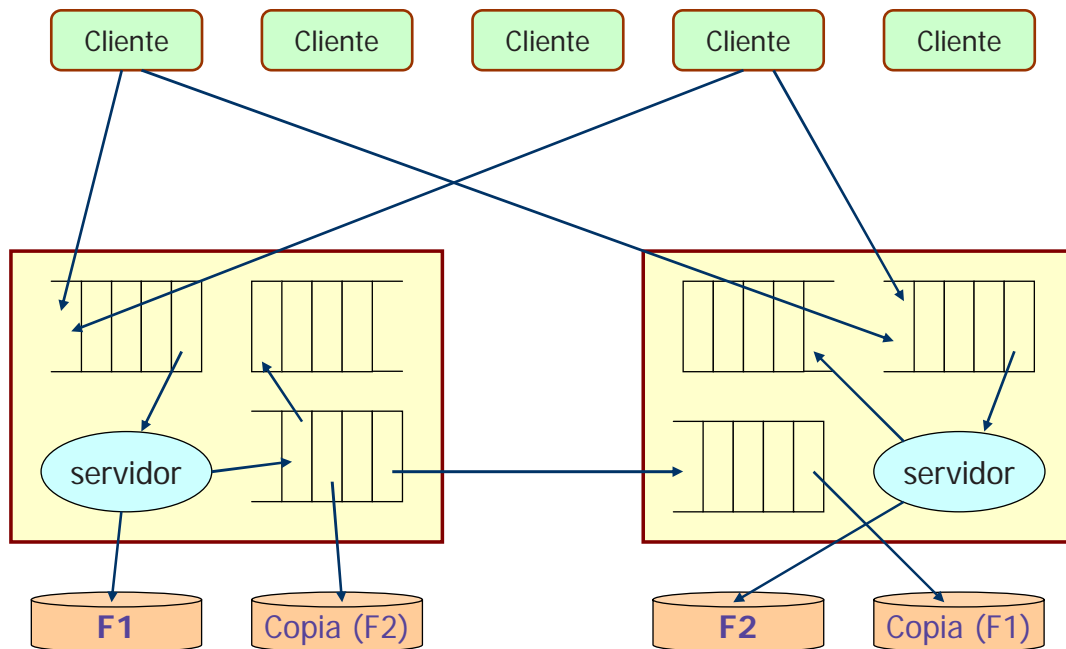
Bases de Datos Distribuidas

Cooperación – Sistemas de Almacenes de Datos





- **Replicación de Datos** es un servicio esencial para muchas aplicaciones distribuidas.
 - Es provisto por productos llamados **replicadores (replicators)**, cuya función principal es **mantener la consistencia entre las copias**. Funcionan de forma transparente a las aplicaciones que se ejecutan en el servidor SGBD.
 - En general, existe **una copia principal y varias copias secundarias**, a las cuales se propagan las modificaciones de forma asíncrona.
 - Existen dos **tipos de propagación** de modificaciones:
 - **Incremental**: la información que se envía desde la copia principal a las secundarias son las variaciones en los datos.
 - **Total**: se envía toda la copia principal completa.
 - **Principal Utilidad**: hacer que el sistema sea **menos sensible a los fallos**, ya que si la copia principal no estuviera disponible se puede seguir usando alguna de las copias secundarias.



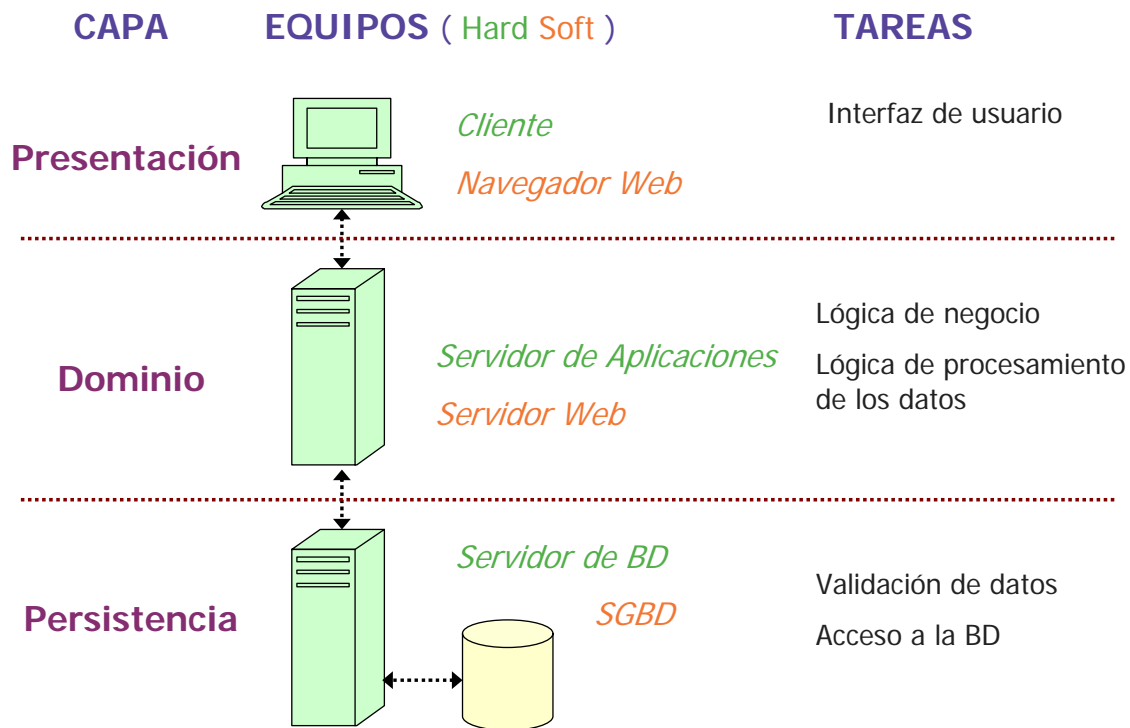
Arquitecturas para la Web

- Los principales requisitos para la **integración** de las tecnologías **Web** y **SGBD**, son:
 - Poder acceder a datos corporativos de forma segura.
 - Disponer de conectividad independiente de los datos y el vendedor, para dar libertad a la hora de elegir SGBD.
 - Poder interactuar con la BD de forma independiente a cualquier navegador o servidor web.
 - Una arquitectura abierta que permita la interoperabilidad con diferentes sistemas y tecnologías: servidores web, DCOM/COM, CORBA, Java/RMI, etc.
 - Conseguir la escalabilidad, crecimiento y cambios a costes aceptables.
 - Soportar transacciones que abarcan múltiples peticiones HTTP.
 - Soportar autenticación basada en sesiones y usuarios.
 - Rendimiento aceptable.



Arquitecturas para la Web

Arquitectura de Tres Capas Web-SGBD



Francisco Ruiz - BDA

6.39



Arquitecturas para la Web

Arquitectura de Tres Capas Web-SGBD

- **Ventajas:**
 - Debidas al uso de SGBD's
 - Simplicidad
 - HTML
 - Independencia de plataforma
 - HTML, Navegador web
 - Interfaz Gráfico de Usuario
 - Navegador web
 - Estandarización
 - HTML y otros
 - Soporte inter-plataforma.
 - Navegador web
 - Transparencia del acceso a red
 - Internet / Intranet
 - Desarrollo escalable
 - Innovación
- **Inconvenientes:**
 - Fiabilidad
 - Internet
 - Seguridad
 - Internet
 - Escalabilidad
 - Sobrecarga de servidores
 - Funcionalidad limitada de HTML
 - Extensiones
 - Inestabilidad
 - Juventud
 - Ancho de banda
 - Internet
 - Rendimiento
 - Clientes interpretados
 - Inmadurez de las herramientas

Francisco Ruiz - BDA

6.40



Arquitectura de Tres Capas Web-SGBD: *Ventajas*

- **Debidas al uso de SGBD's:** muchas de las ventajas de utilizar SGBD vs. Sistemas de ficheros son aplicables para la integración the Web y el SGBD. Por ejemplo, el problema de sincronización de información en laBD y los archivos HTML desaparece, ya que las páginas HTML se generan dinámicamente desde la BD.
- **Simplicidad:** en su forma original el lenguaje HTML era fácil para desarrollos y usuarios finales.
- **Independencia de plataforma:** Un razón importante de crear una versión Web de una aplicación de BD, es que los clientes Web (navegadores) son en su mayoría independientes de la plataforma.
- **Interfaz Gráfico de Usuario (GUI):** Los navegadores Web proveen interfaces comunes, fáciles de usar, que pueden servir para acceder a a diferentes cosas, como la BD. Tener una interfaz común reduce el tiempo y coste de entrenamiento.
- **Estandarización:** HTML es el estándar de facto para todos los navegadores Web. Utilizando HTML, los desarrolladores tienen que aprender un lenguaje fácil, y los usuarios finales utilizan una Interfaz Gráfica de Usuario muy sencilla.



Arquitectura de Tres Capas Web-SGBD: *Ventajas*

- **Soporte inter-plataforma:** Los navegadores Web están disponibles para cualquier tipo de plataforma. Este tipo de soporte inter-plataforma permite a los usuarios en diferentes tipos de plataformas acceder a las BD desde cualquier lugar en el mundo. De esta manera, la información puede estar diseminada con mínimo esfuerzo y tiempo, y sin tener que resolver los problemas de incompatibilidad de diferentes hardware, S.O. y software.
- **Transparencia del acceso a red:** el principal beneficio de la Web es que el acceso a la red es esencialmente transparente al usuario.
- **Desarrollo escalable:** al utilizar una arquitectura de tres capas, se almacena la aplicación en un servidor diferente al del cliente, la Web elimina el tiempo y coste asociado al desarrollo de la aplicación. Permite el manejo de actualizaciones y la gestión de trabajar con múltiples plataformas a lo largo de diferentes oficina.
- **Innovación:** La Web permite a las organizaciones proveer nuevos servicios y así llegar a nuevos clientes a través de aplicaciones globalmente accesibles.



Arquitectura de Tres Capas Web-SGBD: *Desventajas*

- **Fiabilidad:** Internet es actualmente un medio de comunicación poco fiable y lento. Debido a la falta de fiabilidad muchas empresas continúan dependiendo de su Intranet.
- **Seguridad:** la seguridad es un aspecto fundamental para una organización que permite el acceso a sus datos a través de Internet. La autenticación de usuarios y la transmisión de datos segura so aspectos criticos debido a la gran cantidad de usuarios anónimos.
- **Coste:** el coste de sitios Web comerciales suele ser elevado.
- **Escalabilidad:** Sobrecarga de servidores
- **Funcionalidad limitada de HTML:** Aunque HTML es una interfaz común y fácil de usar, su simplicidad hace que algunas aplicaciones de Bd altamente interactivas, no sena fácilmente convertidas en aplicaciones Web. Para agregar funcionalidad extra existen diferentes lenguajes como JavaScript, VBScript, etc .



Arquitectura de Tres Capas Web-SGBD: *Desventajas*

- **Inestabilidad:** La juventud del entorno Web hace que la gestión de las conexiones BD y transacciones de usuario sean difíciles.
- **Ancho de banda:** Una restricción de Internet es el ancho de banda.
- **Rendimiento:** la mayor parte de clientes de bases de datos Web se centran en lenguajes interpretados, haciéndolos tan lentos como los clientes de BD tradicionales.
- **Inmadurez de las herramientas:** las herramientas que permiten crear aplicaciones de BD Web son están en su mayoría inmaduras.

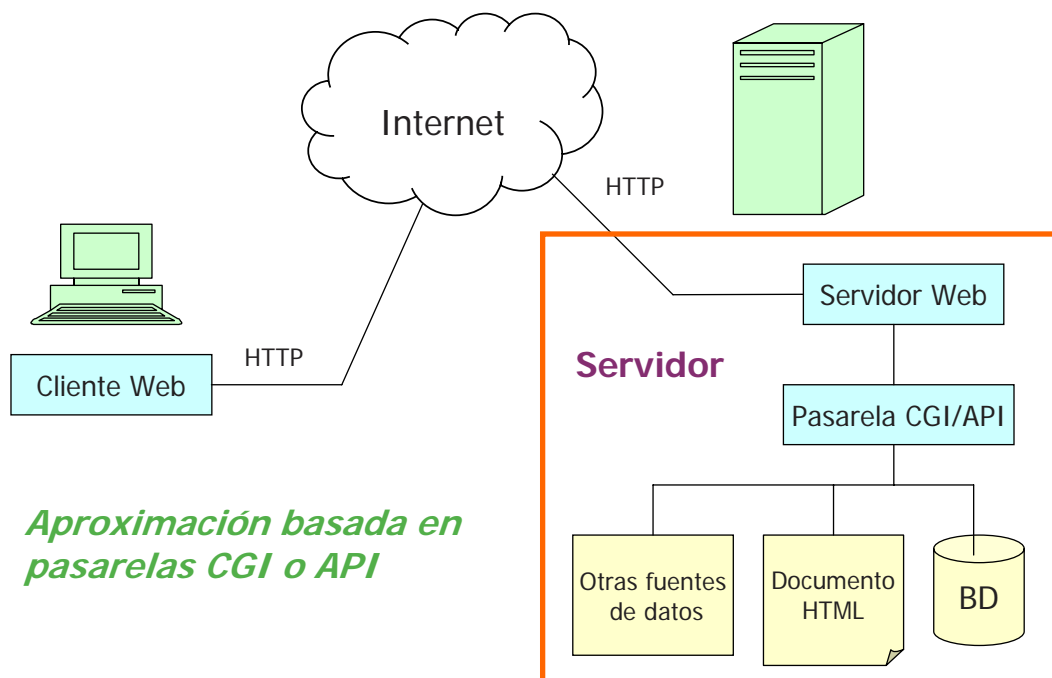


Arquitecturas para la Web Integración Web-SGBD

- Aproximaciones para **integrar la Web y los SGBDs**:
 - **Lenguajes de Script**: órdenes en otro lenguaje incrustadas dentro de código HTML.
 - JavaScript, VBScript, Perl, PHP
 - **CGI (Common Gateway Interface)**: especificación para transferir información entre un servidor web y un programa del mismo equipo.
 - **API (Application Programming Interface)**: son interfaces de programación que permiten añadir nueva funcionalidad a los servidores web superando las limitaciones de CGI.
 - NSAPI (Netscape), ISAPI (Microsoft), ...
 - **Java Servlets**: programas ejecutados en servidores web basados en Java.
 - Ventajas frente a CGI: mejor rendimiento (multi-hebra), portabilidad, extensibilidad, gestión de sesiones más simple, seguridad y fiabilidad mejoradas.



Arquitecturas para la Web Integración Web-SGBD





Arquitecturas para la Web Integración Web-SGBD

- Plataformas para Aplicaciones Web:
 - **J2EE (Java 2 Platform Enterprise Edition)**
 - Estándar industrial (Sun, IBM, Oracle, ...) para aplicaciones web robustas, escalables, multiusuario y seguras.
 - La pieza clave son los **Enterprise Java Beans (EJB)**, un estándar para construir componentes Java del lado del servidor. Hay de dos clases:
 - **EJB de sesión:** implementan la lógica de negocio, reglas de negocio y flujos de trabajo. Su vida es la de una sesión y sólo son usados por un cliente a la vez.
 - **EJB de entidad:** encapsulan datos de empresa. Son persistentes y pueden compartirse por varios clientes.
 - Otra pieza importante es **Java Server Pages (JSP)**: lenguaje de script para servidores web basados en Java que permite combinar HTML normal (estático) con **HTML dinámico** (páginas generadas automáticamente a partir de código HTML y código Java).



Arquitecturas para la Web Integración Web-SGBD

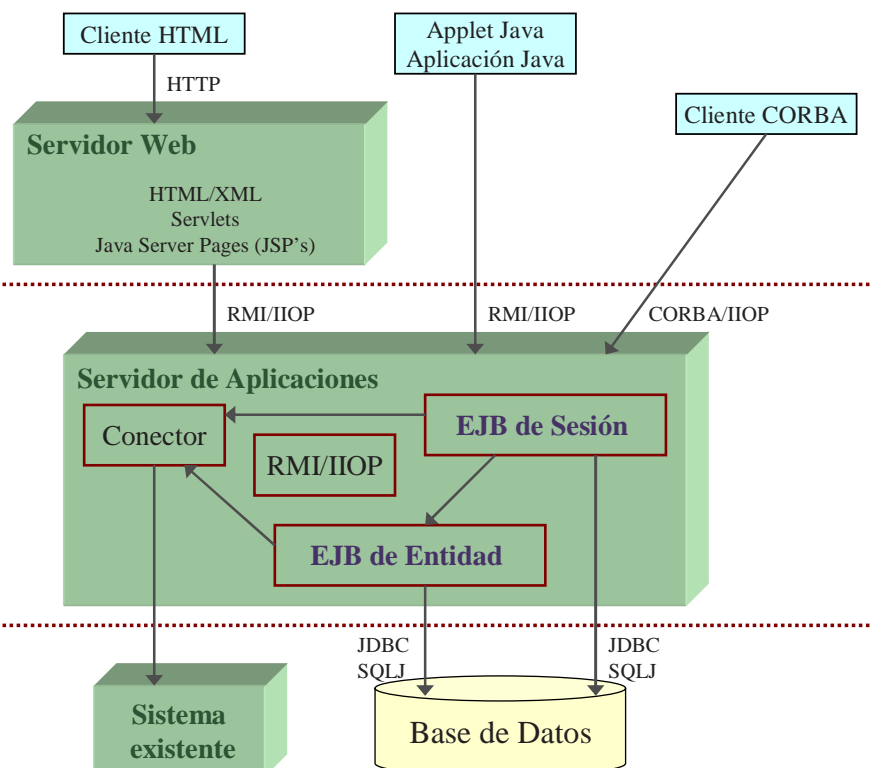
Arquitectura

J2EE

Presentación

Negocio

Datos

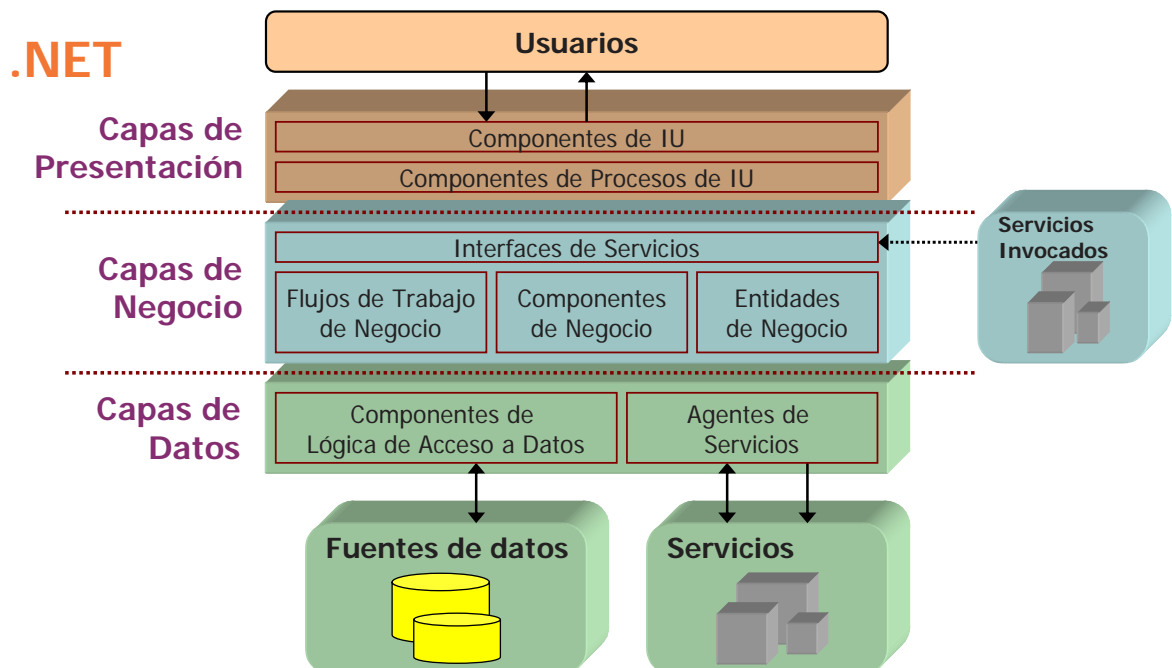




• Plataformas para Aplicaciones Web:

■ Microsoft .NET

- Incluye una infraestructura completa para el desarrollo e implantación de software multi-lenguaje. Está basada en la existencia de una infraestructura común (**.NET Framework**).
- Los principales componentes de .NET relacionados con la interacción Web-SGBD son:
 - **ASP .NET**: conjunto de clases para crear aplicaciones web usando
 - páginas de servidor activas (**active server pages**) que pueden incluir código escrito en cualquier lenguaje .NET; y
 - **servicios web** (con soporte a los estándares SOAP, WSDL y UDDI).
 - **ADO .NET**: clases para acceso a datos almacenados en BD relacionales (SQL), en XML y en otros formatos.
- Propone una arquitectura de aplicaciones web con **n capas** y basada en el nuevo paradigma "**Service-Oriented Computing**".



Aplicaciones web basadas en 'Service-Oriented Architecture' (SOA)