

## Parte final del Tema VII de BSDT Curso 2007-08

### ÍNDICE DE CONTENIDO

#### **Tema VII, Apartados VII.3 y VII.4**

(Continuación del Capítulo 4 de Ozsu&Valduriez)

<b>Tema VII, Apartados VII.3 y VII.4</b>	<b>1</b>
<b>ARQUITECTURAS CLIENTE / SERVIDOR Y WEB. BASES DE DATOS WEB</b>	<b>1</b>
VII.3.- Arquitectura funcional del SGBD: Cliente/Servidor (2 capas) y Arquitectura Web (3 capas).	1
VII.3.1.- Arquitectura Cliente-Servidor (dos capas).	1
VII.3.1.1.- Funcionalidad de la Arquitectura Cliente-Servidor.	2
VII.3.2.- Arquitectura Web (tres capas).	4
VII.3.2.1.- Internet y la World Wide Web.	5
VII.3.2.2.- Servidores de la capa intermedia en la Arquitectura Web.	6
VII.3.2.2.1.- Servidor Web: URLs y Protocolo HTTP.	6
VII.3.2.2.2.- La seguridad en Internet.	8
VII.3.2.2.3.- Gateways, CGI y Servidor de Aplicaciones.	9
VII.4.- Sobre la Estructuración de los Datos y Consultas a Datos Intensivos y heterogéneos en la Web.	11
VII.4.1.- Sobre la estructuración de los datos y consultas a datos intensivos y heterogéneos en la Web.	12
Referencias Bibliográficas.ojo, actualizar	15

### Índice de Figuras

Fig. 7.1. Arquitectura Cliente-Servidor de un SGBD.	2
Fig.7.2. Detalle de la capa Cliente de un SGBD con arquitectura C/S.	3
Fig. 7.3. Arquitectura Web Abreviada (a tres capas) de un SGBD.	5
Fig. 7.4. Detalle de la capa 2 de la Arquitectura Web de un SGBD.	7
Fig. 7.5. Detalle del Servidor de Aplicaciones en la Arquitectura Web de un SGBD.	10
Fig. 7.6. Ejemplo de Objeto OEM para la asignatura BSDT 5403.	13
Fig. 7.7. Clasificación de los Sistemas para consultar datos heterogéneos.	14
Fig. 7.8. Arquitectura Mediator-Wrapper.	14



## **Tema VII, Apartados VII.3 y VII.4**

### **ARQUITECTURAS CLIENTE / SERVIDOR Y WEB. BASES DE DATOS WEB**

#### **VII.3.- Arquitectura funcional del SGBD: Cliente/Servidor (2 capas) y Arquitectura Web (3 capas).**

El SGBD, una de las variedades más complejas del software actual, precisa utilizar el término *arquitectura* con diversas acepciones y puntos de vista. La arquitectura del SGBD se describe ahora bajo el punto de vista de sus formas operativas (funcionales), como un ente que realiza las aplicaciones del usuario.

La arquitectura ANSI/X3/SPARC, descrita en el Tema I, explicó cómo se organiza la información de una BD. En el Tema VI (cap. 8 de [Cost99]) se ha descrito cómo se organizan varios SGBDs para la interoperabilidad (generalmente distribuida), allí vimos la arquitectura C/S con funcionalidad varios-a-varios (varios Clientes y varios -no muchos- Servidores). Finalmente, en el Tema XI (cap. 9 de [Cost99]) atenderemos a otro tipo de arquitectura llamada *arquitectura revisitada*, centrada en la descripción de los aspectos más internos del software que poseen los motores o Servidores de bases de datos.

La arquitectura C/S surge a finales de los años 80's, como una extensión del estándar RDA *Remote database Access* [ISO89] y tuvo gran difusión en la tecnología de bases de datos en la década de los 90's, marcando importantes pautas en la concepción arquitectural de los SGBD, separando nítidamente las funciones del Servidor y las de los Clientes. La arquitectura para los SGBD es algo diferente a la organización del software genérico en Cliente/Servidor (entendido sólo mirando a los procesos que invocan y a los que sirven), ya que la interacción en los SGBD está más especializada.

Como sabemos, la distribución es ingrediente común de los Sistemas de Información actuales. Ahora veremos la arquitectura del SGBD como un ente individual (sin cooperar con otros SGBD) para entender los bloques que posibilitan su funcionamiento. Y no entraremos en detalles de cómo opera el motor o Servidor de Bases de Datos, descrito en Cap. 9 de [Cost99]. Nos bastará ahora con entender que dicho motor contiene el núcleo de las funciones de un SGBD y ha de ejecutarse en un ordenador dotado de recursos adecuados para la gestión de datos masivos.

Veremos cómo se organiza el SGBD en capas para -con ellas- llevar a cabo, en la capa Cliente, determinadas funciones de los aplicativos de usuario. Cada Cliente realiza una especie de pre-procesamiento de las aplicaciones que han de terminar ejecutándose en el Servidor de Bases de Datos. Además de esto, la funcionalidad de la capa del Cliente consiste en implementar funciones de usuario con interfaces amigables y ofrecer alta productividad para dichas funciones de usuario (entornos ofimáticos con: e-mail, procesadores de texto, hojas de cálculo, etc.).

Estudiaremos dos tipos de arquitecturas. La primera describe la arquitectura a dos capas, llamada también arquitectura Cliente/Servidor del SGBD. La segunda es la arquitectura a tres capas, dirigida hacia la World Wide Web y añade una nueva capa conocida como Servidor de Aplicaciones que incluye al Servidor Web (o Servidor HTTP).

#### **VII.3.1.- Arquitectura Cliente-Servidor (dos capas).**

En el Tema VI (cap. 8 de [Cost99]) vimos el paradigma Cliente/Servidor como el más simple del amplio espectro de la interoperabilidad de los SGBDs distribuidos. Un SGBD con arquitectura Cliente-Servidor funciona separando limpiamente el papel que juega el Servidor del que desempeña el Cliente. Esta arquitectura, pensada en la distribución, organiza la ejecución de aplicaciones entre diversas

máquinas -conectadas por una red (generalmente LAN)- que suele alcanzar a distintas dependencias del edificio(s) donde ella opera, como muestra la figura 7.1.

En la tecnología relacional, la integración de esta arquitectura se basa principalmente en el protocolo Net (por ejemplo, Oracle lo llama SQL\*Net). Usa un solo motor SGBD y todas las máquinas son de tipo Cliente salvo la que es Servidora (varios Clientes y un Servidor).

Realmente no es preciso que Cliente y Servidor se ubiquen en distintas máquinas, pues el paradigma de esta arquitectura se fundamenta en construir el software con independencia de dónde residan los procesos. Sin embargo, en la gestión de datos, normalmente los procesos del Servidor se ubican en una sola máquina y las máquinas Clientes se esparcen por la red de la institución o empresa para la que se instala este tipo de arquitectura.

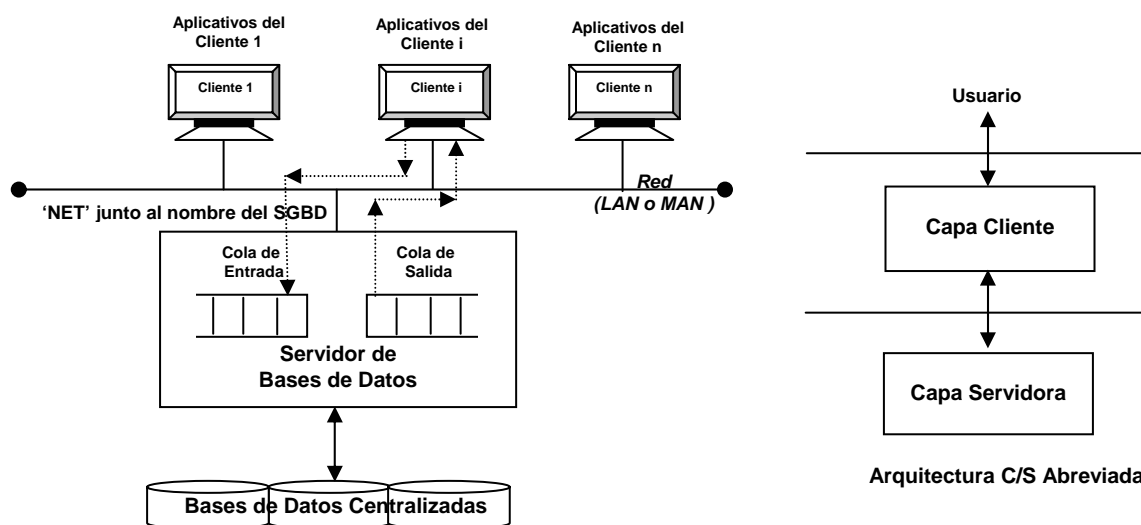


Fig. 7.1. Arquitectura Cliente-Servidor de un SGBD.

La arquitectura C/S de un SGBD balancea la carga de trabajo entre los Clientes y el Servidor, repartiendo las tareas implicadas en la ejecución de las aplicaciones. El SGBD distribuye, por tanto, su software en diversas máquinas cuando opera a nivel de producción industrial y realiza en paralelo la parte Cliente.

### VII.3.1.1.- Funcionalidad de la Arquitectura Cliente-Servidor.

La funcionalidad Cliente-Servidor se basa en un modelo de interacción entre procesos software. Los clientes invocan y solicitan la ejecución de procesos (requests, queries, etc.) que ellos requieren y no poseen. Los procesos solicitados residen en la parte Servidora. La arquitectura genérica C/S se organiza de forma que un Cliente puede solicitar servicios a varios (no muchos) Servidores.

Este epígrafe, dedicado a la arquitectura de un solo SGBD, parte de la hipótesis que el Servidor de Bases de Datos es único en este tipo de arquitectura y los Clientes quedan conectados a él mediante una red (LAN casi siempre) cuya interacción C/S se basa en el protocolo **NET**. Los procesos del lado Cliente se dedican a atender las necesidades del usuario final, ellos reciben las aplicaciones del usuario y las pre-procesan para transformarlas en otras unidades software (**transacciones** de bajo nivel, que estudiaremos en el Tema XI) que envían al Servidor para solicitar su servicio. Los Clientes solicitan a la parte Servidora que lleve cabo la ejecución definitiva de los aplicativos (accesos y/o actualizaciones a los valores de los datos de la BD). Por tanto, el Cliente envía siempre todas sus solicitudes al Servidor, mientras que el Servidor atiende solicitudes de varios Clientes. Las siguientes razones justifican el uso de la arquitectura C/S en bases de datos:

La arquitectura C/S permite una limpia separación de objetivos y trabajos. Por un lado, cada máquina Cliente se dirige a optimizar determinadas aplicaciones del usuario final, conocidas de antemano por quien construye el sistema de información [CoBV93]. El Cliente atiende a un conjunto determinado de aplicaciones (las de un tipo de usuario) y el programador de aplicaciones es responsable de escribir programas (aplicativos) para que cada Cliente responda a las necesidades específicas en cada máquina. Por otro lado, el diseñador y administrador de las bases de datos, trabajará en la capa Servidora para realizar las tareas propias y relativas a diseño y administración: asignación de roles de usuario, permisos y privilegios, definiciones de Vistas, configuración de espacios de tablas, mantenimiento de índices para un óptimo rendimiento funcional, etc. Todo ello será compartido por todos los clientes de una arquitectura dada. En definitiva, los trabajos del Servidor se dirigen a proporcionar óptimos servicios a todos los procesos clientes.

La potencia del ordenador donde se ubica el Servidor depende estrechamente de los servicios que éste tiene que ofrecer, necesita una gran memoria principal para la gestión de múltiples *buffers* y una alta capacidad de disco duro para almacenar completamente la base de datos. Sin embargo, en la máquina Cliente bastaría con un típico ordenador personal que proporcione las herramientas de un entorno ofimático y que albergue el software o procesos de la parte cliente del SGBD. Entre dichas herramientas, casi siempre enmascaradas por interfaces amigables del usuario, tiene que tener instaladas las destinadas a solicitar servicios al Servidor de Bases de Datos y las herramientas de ayuda al desarrollo de las aplicaciones (*reports/forms*), como muestra la figura 7.2.

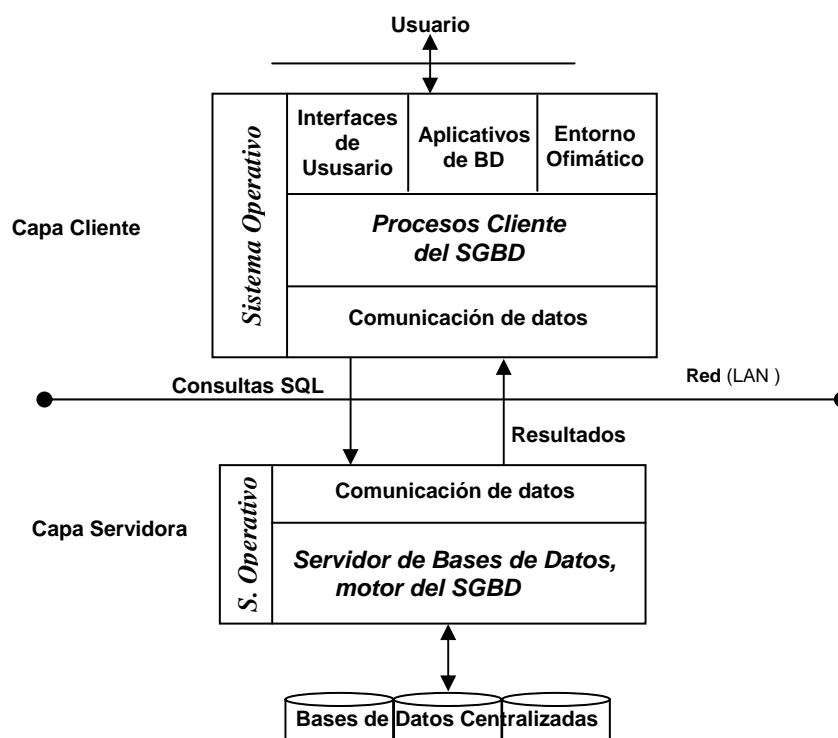


Fig.7.2. Detalle de la capa Cliente de un SGBD con arquitectura C/S.

SQL es ideal para identificar los *servicios de interfaz* de la capa cliente, responsable de interpretar la consulta del usuario, darla formato y presentar los resultados al usuario de forma adecuada. La consulta, escrita en el Cliente, se envía al Servidor para que se procese allí. El Servidor extrae los resultados de la base de datos, los empaqueta y, finalmente, los devuelve al Cliente que hizo la solicitud. De esta forma, por la red circula la *mínima información útil*.

La consulta SQL del usuario admite dos formas de invocación diferentes.

La primera, conocida como *consulta compilada de forma estática* que llega al Servidor sólo una vez, allí se compila y se almacena para ser re-llamada tantas veces cuantas sea preciso. De esta forma, el Servidor almacena el código compilado de la consulta en forma parametrizada; y, en cada invocación del cliente (típicamente se invoca a un procedimiento), basta con pasar los valores de los parámetros que en cada caso corresponda. A menudo, el Servidor que procesa estos procedimientos es multi-hebra (*multi-threaded*) y cada unidad de ejecución del proceso del Servidor, para una transacción dada, es una hebra. Como muestra la figura 7.1, los procesos del Servidor están activos permanentemente para ir recibiendo solicitudes de los Clientes desde la cola de entrada, y dejan en la cola de salida los resultados que el Servidor obtiene para enviárselos a cada respectivo Cliente. El *dispatcher* gestiona las colas del Servidor.

La segunda forma de invocación se conoce como *consulta dinámica*, donde el Cliente envía al Servidor la consulta como un *string* de caracteres, y allí es compilada y ejecutada cada vez.

La idea global de un Sistema de Información con arquitectura C/S es que funcione en el modo de consulta estática (compila una vez y se guarda en el Servidor), para ser re-llamada de continuo, cuantas veces se precise su ejecución en el Servidor. Mientras que el modo de consulta dinámica está pensado para un tipo de consulta esporádica o no frecuente, su rendimiento es más bajo que el de la invocación estática y por tanto con un tiempo de respuesta mayor.

### VII.3.2.- Arquitectura Web (tres capas).

La espectacular evolución de Internet plantea grandes retos para mejorar su potencialidad y actual forma operativa. La tecnología world-wide-web (www) [BCGP92], [BeCG92], [Bern97] es el substrato de interconexión de la sociedad de la información actual y futura. Su ritmo de expansión y la diversidad de usuarios crecen de forma continua y exponencial. Se conectan a Internet desde las empresas y gobiernos (con redes de alta velocidad) hasta los individuos desde los hogares (con *modems* de baja velocidad). Muchas empresas e instituciones han desarrollado su propia Intranet conectada a Internet. Los sistemas de información así ubicados se conocen como Sistemas de Información Web (en adelante, SIW, en inglés WIS), y acoplados con un SGBD se puede proporcionar acceso rápido e inteligente a grandes cantidades de datos estructurados, como los que existen en una base de datos [FILM98].

La Web empezó siendo una interfaz para el acceso a documentos distribuidos, y hoy es una plataforma para sistemas de información de todo tipo. Se trata de un paradigma que permite difundir y adquirir cualquier tipo de información a través de su arquitectura, configurada en capas que, en el caso de un SGBD con arquitectura web, serían las tres siguientes que muestra la figura 7.3.

**Capa frontal (Front End)**, formada por una herramienta genérica llamada navegador<sup>1</sup> o *browser* (p.ej.: *Netscape Navigator* o *Microsoft Internet Explorer*). Constituye la interfaz de la máquina del usuario para navegar por la Web. El *browser* es la parte del usuario, por tanto, es el lado Cliente (descrito en C/S como 1ª capa) ahora llamado *Cliente delgado, ligero o fino* porque son mínimas las exigencias del ordenador que así funciona. Se puede entender al *browser* como una Interfaz Gráfica de Usuario (GUI) que reside en la máquina cliente o máquina remota del usuario.

**Capa intermedia**, llamada **Servidor HTTP** o **Servidor Web** que, generalmente, contiene además al **Servidor de Aplicaciones**. Se puede entender a esta capa como un conjunto de programas residentes en *sitios Web*. Actualmente, se trata de bibliotecas de servicios (Java) que siguen estándares del W3C (WSDL, SOA, UDDI, etc.)

**Capa dorsal (Back End)**, llamada Servidor de Información o **Servidor de Bases de Datos**, descrito con detalle en el Tema XI (cap. 9 de [Cost99]).

---

<sup>1</sup> Navegador, máquina conectada a Internet, en inglés: browser. Este texto usa indistintamente ambos términos.

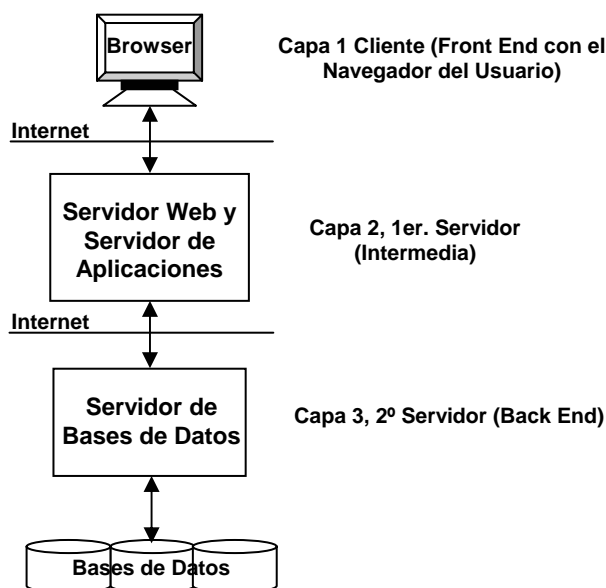


Fig. 7.3. Arquitectura Web Abreviada (a tres capas) de un SGBD.

Casi toda la actual tecnología de bases de datos dispone de un Servidor de Aplicaciones además del Servidor Web como primera máquina servidora (ubicada en la 2ª capa intermedia) y de una segunda máquina servidora donde reside el Servidor de Bases de Datos que pasa a estar en la 3ª y más interna capa operativa de esta arquitectura, como vemos en la figura 7.3.

### VII.3.2.1.- Internet y la World Wide Web.

Internet funciona como una federación de redes comunicadas todas por el mismo conjunto de protocolos, procedentes de la familia TCP/IP (Transmission Control Protocol/Internet Protocol). Un nodo de Internet forma parte de una red de área local (LAN) que se forma conectando nodos (con PCs o estaciones de trabajo) ubicados en un área de pequeño alcance.

Las redes locales se comunican con las demás formando una jerarquía de redes, por ejemplo: la red universitaria, la red de todas las universidades de un país, toda la red nacional, y así sucesivamente. Físicamente, la jerarquía de redes y la estructura de direcciones de los nodos dentro de una red se muestran transparentes (a usuarios y a programas), de tal forma que el usuario puede referenciar<sup>2</sup> a cualquier nodo cuya dirección sea conocida por él. Lógicamente, esta federación de redes se percibe como que cualquier nodo Internet puede conectarse con cualquier otro. Para ello, TCP/IP se organiza en capas como se describió brevemente en el Tema VI.

El protocolo TCP permite que los ordenadores (de cualquier fabricante y con sistemas operativos distintos) se comuniquen entre sí, y resulta apasionante ver cómo se han superado todas las estimaciones en cuanto a su crecimiento. Lo que comenzó a finales de los años 60 como un proyecto de investigación financiado por el gobierno norteamericano sobre redes de conmutación de paquetes, ha llegado a ser desde los 90 la forma de comunicación entre ordenadores más difundida. Se trata de un sistema abierto, donde la definición del protocolo y muchas de sus implementaciones son públicas y gratuitas. Actualmente TCP/IP es la infraestructura de Internet, la red de área global que, como sabemos, recubre el globo interconectando millones de ordenadores.

La Web se inició con la idea de llegar a ser el medio para poder acceder a diversos documentos de varios tipos, producidos para temas diferentes y mantenidos desde multitud de nodos conectados por

<sup>2</sup> Referenciar, en la técnica se entiende como apuntar, señalar o enlazar a algo con otra entidad u objeto.

Internet. El documento recibió el nombre de *hipertexto*<sup>3</sup> porque su naturaleza no es secuencial, está formada por varias partes relacionadas mediante enlaces; de forma que se puede acceder a cada parte directamente, sin la rigidez de tener que seguir una secuencia dada.

La Web se percibe normalmente como una colección de documentos no estructurados. La idea de *hipertexto* se ha generalizado en muchos sentidos. La técnica no estructurada no tiene porqué aplicarse sólo al documento, mas bien puede usarse para relacionar diversos documentos producidos por distintas personas en distintos momentos. Además, los documentos no tienen porqué ser textuales, también pueden tener imágenes, vídeo, voz, etc. y; cuando esto ocurre, entonces hablamos de navegación por los *hipermedia*. Adicionalmente, dichos documentos suelen estar distribuidos entre los nodos de cualquier red de ordenadores; en suma, en Internet. Finalmente, la Web no sólo permite acceso a documentos estáticos, ya que también permite lanzar programas [GoEi01] para que éstos generen dinámicamente documentos nuevos (en páginas dinámicas).

El punto VII.3.2.2 describe los principales componentes tecnológicos que posibilitan la implementación de estas breves ideas sobre la Web. Principalmente son:

1. el protocolo de comunicación HTTP,
2. el concepto URL para referenciar a los servidores que poseen los recursos,
3. el lenguaje HTML para marcar las partes de los documentos, y
4. el avanzado lenguaje XML, extensión de HTML.

Desgraciadamente, la Web está aún lejos de ser un repositorio bien organizado de documentos XML, mas bien hoy es un conglomerado de páginas HTML volátiles cuyas estructuras han de ser extraídas<sup>4</sup> en cada acceso. El gran éxito logrado por la Web ha puesto en evidencia una necesidad urgente de ir más allá de una simple *navegación humana* por entre los documentos. Ahora, es preciso abordar la construcción de nuevos pasos que permitan realizar dicha *navegación automática* a las aplicaciones con el fin de llegar a ofrecer nuevas formas de interoperabilidad entre los servicios Web.

Para lograr esto, las fuentes de información de la Web necesitan ser accesibles de forma estructurada y, en este sentido, XML y sus diversas extensiones (en modelo de datos y lenguaje consultivo) son pasos que se vienen dando en esta dirección.

### VII.3.2.2.- Servidores de la capa intermedia en la Arquitectura Web.

La figura 7.4 muestra de nuevo la arquitectura Web (desde la fig. 7.3) destacando la segunda capa donde está la primera máquina Servidora que incluye dos tipos de servicios: el proporcionado por el Servidor Web y el del Servidor de Aplicaciones, descritos a continuación.

#### VII.3.2.2.1.- Servidor Web: URLs y Protocolo HTTP.

La arquitectura Web es una variante de C/S, basada en el protocolo TCP/IP; y, sobre éste, en Web se instala el protocolo HTTP (Hyper Text Transfer Protocol). Con HTTP cualquier cliente *browser* puede solicitar un documento a un servidor Web usando su URL<sup>5</sup> (Uniform Resource Locator). Como indica la figura 7.4, la Web trabaja con **Servidores HTTP**, más conocidos por **Servidores Web**. El protocolo HTTP es de naturaleza muy simple y consta de cuatro fases:

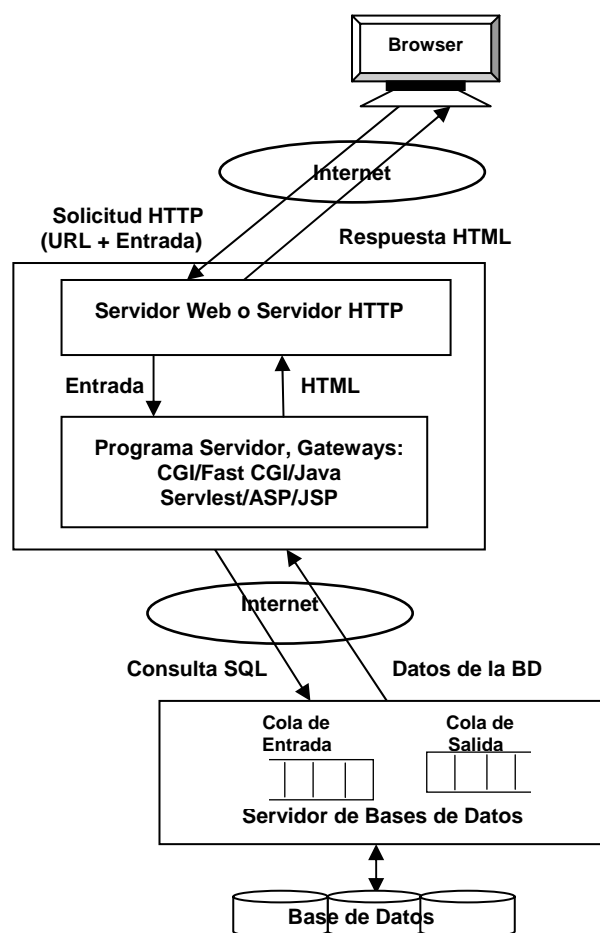
---

<sup>3</sup> Hiper, *prefijo griego que significa "exceso"*.

<sup>4</sup> Extraer, *sacar algo que está incrustado en el documento*.

<sup>5</sup> URL, *es el Localizador de Recursos Uniforme. Su género es masculino, aunque a veces se designe como la dirección donde se encuentra el recurso (en femenino)*.

- a) *Abrir la Conexión*. En ella, el *browser* contacta con el servidor HTTP que se indica en el URL para verificar su disponibilidad y corrección (llamado Solicitud HTTP en la figura 7.4);
- b) *Establecer la Conexión*. Si el servidor está disponible, éste acepta la conexión y envía una confirmación al cliente (no representado en la figura 7.4);
- c) *Enviar Solicitud*. El cliente envía un mensaje al servidor solicitando un servicio y dando el nombre del recurso invocado y los posibles parámetros de la invocación (llamado Entrada en fig. 7.4);
- d) *Recibir Respuesta*. El servidor devuelve al cliente el resultado del servicio requerido (llamado Respuesta HTML en fig. 7.4), y se cierra la conexión por el servidor sin retener información alguna para ser usada en conexiones subsiguientes.



**Fig. 7.4. Detalle de la capa 2 de la Arquitectura Web de un SGBD.**

La Web hace uso de la naturaleza distribuida de los hipertextos y, mediante un esquema C/S, provee un servicio de transferencia de hipertextos basado en el protocolo HTTP, usado desde 1990. Por tanto, HTTP es un protocolo de nivel de aplicación con la agilidad y velocidad necesarias para sostener sistemas de información *hipermedia* distribuidos. Se trata de un protocolo orientado a objetos, de carácter genérico, que se puede usar para muchas tareas, tales como nombrar servidores y sistemas de manipulación de objetos distribuidos, a través de la utilización de las extensiones de sus comandos.

Las principales características de HTTP son su tipado de objetos y la negociación de la representación de los datos, permitiendo que los sistemas se construyan con independencia de los datos que deben transferir.

Una buena parte de los datos accesibles en la Web están almacenados en repositorios estructurados, como ocurre en bases de datos. Otros repositorios, como los ficheros planos (textos, imágenes, audio y

vídeo), se gestionan a menudo mediante aplicaciones *ad hoc*. Aunque, desde el momento en que los datos son accedidos vía Web, la forma física de cómo están organizados estos datos en el repositorio queda siempre oculta para el usuario, quien sólo percibe que unas veces el acceso es ágil e inteligente, y otras es lento, algo rígido y no hábil.

La dirección de cada máquina -nodo de Internet- que funciona en la modalidad C/S es conocida en el alcance donde ella coopera (para un funcionamiento global) por su dirección IP (Internet Protocol, o nombre asociado a ella) que la identifica unívocamente y TCP es el protocolo para controlar las transferencias de información entre máquinas. Sobre TCP/IP, la tecnología Web se fundamenta en saber localizar *sitios Web* desde una máquina conectada a Internet, que -a su vez- también es un *sitio Web*.

El punto de entrada a la Web desde un sitio Web es su página casera (home page), identificada por un URL formado por bloques (de números o nombres asociados a esa dirección) y separados por puntos (por ejemplo: <http://www.etsit.upm.es> es la página casera de nuestra Escuela de Telecomunicación, y <http://www.w3c.org> es el URL del sitio web que es responsable de la definición de estándares Web). Un URL se refiere a otros documentos de otros URLs. Así es como se proporciona la navegación por los *hipertextos* para su localización.

Todo URL tiene la siguiente estructura: **[Protocolo://][Servidor/]Recurso**, y define objetos en una red Internet, que contienen datos sobre:

a) la naturaleza del **Protocolo** que deberá ser de alguno de los tipos disponibles en Internet; por ejemplo HTTP, o FTP (File Transfer Protocol), Telnet (para emulación de terminales) o e-mail para correo electrónico (todos ellos vistos como objetos asociados con alguno de los protocolos o servicios disponibles en Internet: ftp, http, mailto, file, etc.),

b) el nodo **Servidor** de la red, formado **[host]:**, que viene dado generalmente por su nombre simbólico, aunque también se puede escribir la dirección IP donde se encuentra dicho servidor, y

c) el **Recurso** dado por el *path* del directorio seguido del nombre del fichero expresado como [puerto]/[path del fichero] cuyo fichero físico contiene el objeto solicitado.

Por ejemplo, esto es un URL: <http://microsoft.com/download/aspdoc.zip>. Si se omite el puerto se tomará el válido por defecto para el protocolo o servicio utilizado (puerto 80 para servicios Web). Existen URLs absolutos y relativos. Los absolutos especifican un *path* completo (como el del ejemplo anterior) y los relativos especifican un *path* relativo al URL del documento (por ejemplo: *imagenes/dibu1.gif*).

HTTP es un valioso medio para establecer enlaces entre un gran número de nodos independientes. Su contrapartida es que no mantiene ningún contexto entre el cliente y el servidor, lo que puede resultar muy pesado para el usuario a la hora de mantener determinadas sesiones. Si un cliente lanza a un mismo servidor varias solicitudes, el servidor no es capaz de mantener información sobre las solicitudes anteriormente hechas por el mismo cliente ni de los resultados anteriormente enviados a éste. Ello se debe a la gran simplicidad del protocolo, lo que representa serias limitaciones en aquellos casos que requieren una interacción fluida entre máquinas, como las que necesitan las transacciones realizadas en las bases de datos.

### VII.3.2.2.2.- La seguridad en Internet.

Como todo sistema informático de comunicaciones, la Web precisa una serie de herramientas de seguridad asociadas como son:

- La autenticación de solicitudes y de servidores.
- La privacidad de solicitud y respuesta.
- Protección contra abusos de la capacidad del servidor y de la información disponible
- Control de acciones inconscientes sobre la red.

Aunque HTTP intenta abordar algunos de estos problemas, la seguridad se mejora con el protocolo HTTPS, cuya 'S' final significa *Security*.

### VII.3.2.2.3.- Gateways, CGI y Servidor de Aplicaciones.

Los Servidores Web invocan a programas (llamado Entrada en la Figura 7.4) y, en cada invocación, pasan los debidos parámetros, como por ejemplo: los datos de entrada que el usuario ha escrito en un formulario (*form*). Ahora veremos el concepto básico relativo a la forma de acceso a una base de datos.

En la figura 7.4 se observa el flujo de mensajes (unos de petición, otros de respuesta) entre el *browser*, el Servidor Web y el Servidor de Aplicaciones. La primera petición del cliente se envía al Servidor Web y éste, a su vez, envía la solicitud al Servidor de Aplicaciones que es quien invocará al Servidor de Bases de Datos. Cuando éste último servidor realice las acciones necesarias, enviará la respuesta con la información procesada siguiendo el camino anterior, pero ahora recorrido en dirección contraria, hasta que la respuesta llegue al *browser* que lanzó la solicitud [Or9i01].

Describiremos brevemente cómo se establece la comunicación entre el Servidor Web y el Servidor de Aplicaciones. Para ello, el Servidor Web utiliza diferentes tecnologías para obtener y enviar la información que va a ofrecerle el Servidor de Aplicaciones. Dichas tecnologías son las siguientes:

- **CGI** (Common Gateway Interface), mecanismo de comunicación escrito en: Java, C, C++, Perl, etc.,
- **ASP** (Active Server Pages), tecnología de Microsoft,
- **JSP** (Java Server Pages),
- **Java Servlets**, tecnología de Sun,
- **Java Script**, tecnología de Netscape

Un *gateway*<sup>6</sup> es cualquier programa que ha sido llamado por un Servidor Web. El programa puede estar escrito en un lenguaje de alto nivel (*C*, *C++* o *Java*) y compilado, o bien puede estar escrito en un lenguaje interpretado (como *Perl* o *Tcl* -usado para realizar operaciones sobre cadenas de caracteres-), o incluso puede estar en algún lenguaje *script* de los sistemas operativos.

Como su nombre indica, el *gateway* permite establecer una conexión entre el Servidor Web y otro entorno cualquiera que proporcione el servicio requerido.

El Servidor de Aplicaciones es un programa que reside en el 1er. Servidor (2ª capa) de la arquitectura Web y proporciona cierta lógica de negocio para las aplicaciones.

El *gateway* se invoca usando un URL parecido a como se invoca a los ficheros HTML mediante el protocolo HTTP donde, normalmente se inicia con <http://>, después se indica el nombre del servidor, el del directorio y el de un fichero (*path*), y los parámetros se pueden especificar añadidos al URL. El mecanismo de comunicación entre el Servidor Web y los *gateways* se llama **CGI** y sigue los siguientes pasos:

1. El usuario, desde su *browser*, solicita el URL bien enviando un formulario o bien haciendo click con el ratón sobre un ancla. Con ello, se transmiten los parámetros al Servidor Web, cuyas técnicas detalladas escapan al interés de nuestro programa.
2. El Servidor Web lanza el *gateway* (también llamado *programa CGI*) según el protocolo CGI y le transmite los parámetros.

---

<sup>6</sup> Gateway, puerta de paso o pasarela a otro entorno. Usaremos *gateway* por brevedad y uniformidad con la terminología técnica.

3. Con los parámetros recibidos, se ejecuta el *gateway* y, en la arquitectura que nos ocupa, interactuará con el Servidor de Bases de Datos (llamado Consulta SQL en la figura 7.4).

4. Cuando el *gateway* reciba respuesta del Servidor de Bases de Datos (llamado Datos de la BD en la figura 7.4), retornará el resultado al Servidor Web (llamado HTML en la figura 7.4).

5. Finalmente, el Servidor Web transmite el resultado al *browser* (Respuesta HTML en la fig. 7.4).

La figura 7.5 detalla algunas implementaciones utilizadas para el Servidor de Aplicaciones, cuyas principales características se resumen en los siguientes puntos:

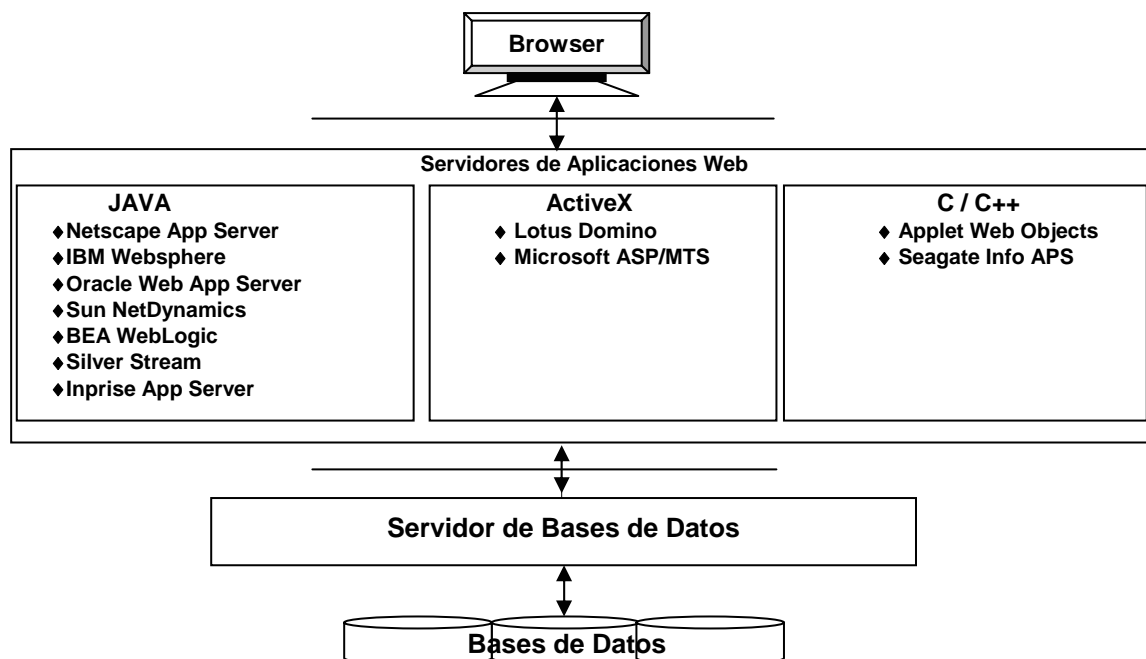


Fig. 7.5. Detalle del Servidor de Aplicaciones en la Arquitectura Web de un SGBD.

**Gestión de Componentes:** Proporcionan herramientas para controlar todos los componentes y servicios de ejecución como la gestión de sesiones, notificaciones síncronas / asíncronas entre clientes, así como la ejecución de servidores de lógica de negocio.

**Tolerancia frente a Fallos:** Capacidad de recuperación frente a fallos, evitando un único punto de ruptura, definiendo políticas de recuperación en caso de fallo de uno o varios objetos.

**Balanceo de Carga:** Posibilidad de enviar las peticiones a diferentes servidores dependiendo de la carga y capacidad de cada uno de ellos.

**Gestión de Transacciones:** Descrito en Tema XI (capítulo 9 de [Cost99]).

**Gestión Centralizada:** Gestión centralizada a través de una consola gráfica para monitorizar los clientes y los distintos servidores o clusters.

**Seguridad:** Características de seguridad y gestión de usuarios y grupos.

Los servidores de aplicaciones se organizan principalmente en tres grandes tipos:

**Web Information Servers:** Este tipo de servidores emplean plantillas HTML y Scripts para generar páginas e incorporar valores de las bases de datos en ellas. Estos servidores son *stateless servers*, es decir, no gestionan el estado de los datos ni coordinan las transacciones. Servidores de este tipo son el *Netscape Server*, *Allaire* o *Sybase*.

**Component Servers:** La principal característica de estos servidores es proporcionar acceso a la base de datos y servicios de procesamiento de transacciones a componentes DLL, CORBA, y JavaBeans. Proporcionan el entorno para los componentes del servidor y acceso a los datos y otros servicios a estos componentes. También son *stateless servers*. Ejemplos de este tipo de servidores son *MTS*, *Sybase Jaguar*, y el *IBM Component broker*.

**Active Application Server:** Este tercer tipo de servidores soporta y proporciona un buen entorno para la lógica del servidor expresada en objetos, reglas y componentes. A diferencia de los anteriores, son *stateful servers* y son más propicios para desarrollos basados en *e-commerce* y procesos de toma de decisiones. Los *stateful servers* son los que juegan el rol de coordinadores de transacciones y gestionan el estado de los datos, se estudiarán en el Tema XII junto al protocolo *Two Phase Commit*.

#### **VII.4.- Sobre la Estructuración de los Datos y Consultas a Datos Intensivos y heterogéneos en la Web.**

Todos los conceptos y técnicas del Modelado Conceptual (vistos en el Tema II, Cap. 2 de [Cost99]) y del Diseño de Bases de Datos Relacionales (Tema IV, Capítulo 4 de [Cost99]) son adaptables al caso que nos ocupa. Ahora, la base de datos se concibe para operar en la arquitectura Web, donde el sitio de nuestro interés contendrá un Servidor de Bases de Datos y, por tanto, los accesos a la BD serán percibidos en la Web como un sitio que contiene datos intensivos y estructurados en una BD (o en varias). Los **sitios con datos intensivos** son aquellos cuyo principal propósito es ofrecer grandes cantidades de información a una variedad de usuarios; es decir, sitios que cuentan con un Servidor de Bases de Datos.

La idea general que hoy tenemos de la Web se puede situar en algún punto intermedio de los dos extremos siguientes:

a) Por un lado, se puede percibir la Web como una fuente de información unificada. Pero esto no sería muy veraz, ya que la Web adolece bastante de coherencia en la forma de presentar la *información útil al usuario*, sobre todo en lo concerniente a su calidad en lo sustancial y a su accesibilidad.

b) Por otro lado, la Web se puede considerar como una inmensa colección de páginas independientes (conglomeradas) donde, cada página se percibe como una fuente independiente de información útil que nada tiene que ver con lo que ofrecen las demás (y casi infinitas) páginas. Éste es el punto de vista que adoptan las listas de *bookmarks*<sup>7</sup> para señalar sitios Web, gestionados por el navegador y los buscadores (p. ej. Alta Vista, Yahoo, etc.) ocupados en seleccionar enlaces a páginas simples, cuyas técnicas son propias de la búsqueda documental (Information Retrieval, thesaurus, véase epígrafe I.3.b del capítulo 1 [Cost99]).

Ambos extremos resultan inapropiados para delimitar el alcance del diseño de bases de datos en, y para, la Web. En efecto, el diseño tiene que concentrarse en un determinado Universo del Discurso (el que, en cada caso, resulte necesario) que debe estar sometido al control requerido en los objetivos de cada diseño concreto. Debido a que los sitios Web disfrutaban normalmente de la propiedad enunciada en b), nosotros deberemos entender que los sitios Web son como una extensión a incorporar en la actividad del diseño de bases de datos, como un ingrediente más que se añade al trabajo de diseño de bases de datos que hasta ahora conocemos como clásico.

Este epígrafe se dedica al análisis y estudio de dicho ingrediente nuevo: concentrado en diseñar **sitios Web con datos intensivos** cuyas exigencias son las de ser los mejores servidores utilizando **estructuras muy regulares**. Lo que contrasta con la organización de la información que hoy ofrece Internet, cuya descripción se realiza en todo este epígrafe.

---

<sup>7</sup> Bookmark, lo que se coloca entre las páginas de un libro para marcar un lugar.

#### VII.4.1.- Sobre la estructuración de los datos y consultas a datos intensivos y heterogéneos en la Web.

Aunque la Web comenzó siendo una colección que interconecta documentos no estructurados, hoy cuenta con un gran número de fuentes de datos estructurados en bases de datos (de todo tipo y naturaleza) a las que se accede mediante *gateways*.

Desde la arquitectura C/S, la interfaz de estas *bases de datos gateways* está formada típicamente por múltiples formularios (*forms*) y por multitud de informes (*reports*). Los *forms* son como una plantilla (*template*) donde se actualiza el estado de la BD (altas, bajas y modificaciones) en los campos y ventanas previstos en cada formulario. Los *reports*, por su parte, realizan accesos meramente consultivos para devolver información de la BD, cuya presentación final (valores de datos, barras o tartas estadísticas, etc.) está prevista en cada tipo de informe.

En la Web, el resultado de una consulta a la BD toma forma de documento HTML que está muy estructurado y que puede ser convertido (con un *parser*) a un conjunto de tuplas o a tipos de datos más complejos. Este asunto está hoy bastante bien resuelto, como se describe en el siguiente epígrafe.

Sin embargo, además de lo dicho para las BD operativas en Web, existen otras fuentes de información como el nombre de los servidores, fuentes bibliográficas, amplios sistemas de información (sobre universidades, empresas, etc.) que ofrecen información *online* pero que pueden no estar disponibles en la Web y cuyas interfaces también proporcionan accesos consultivos [TRCH05].

Actualmente, la estructuración de los datos disponibles en Internet se organiza de muy diversas maneras, cuyo rango varía desde los datos sin estructura alguna (texto bruto, mapas, imágenes, etc.) hasta los datos totalmente estructurados como lo están en las bases de datos (jerárquicas, Codasyl, relacionales u orientadas a objetos). Entre estos dos extremos, están los documentos HTML que se han denominado como **semi-estructurados** [Abit97] y se sitúan en algún lugar intermedio de las técnicas de estructuración, y que presentan notables diferencias a las técnicas usadas en las BD.

Los datos semi-estructurados no utilizan el concepto de esquema (nivel intensional en BD) como molde que acoge a múltiples valores de datos (nivel extensional en BD). Al contrario, en el documento HTML (semi-estructurado) cada una de sus partes componentes posee una definición propia y exclusiva. De forma que la organización del documento consiste principalmente en definir los enlaces entre sus partes y albergar el valor del dato (sea un objeto estructurado o valor atómico) en cada parte.

El Modelo de Intercambio de Objetos, en adelante OEM (Object Exchange Model) se ocupa del intercambio de objetos entre fuentes de datos heterogéneas con la idea de incorporar alguna estructura a los datos no estructurados [PaGW95], [BDFS97] y permitir la formulación de consultas y su optimización a datos semi-estructurados [GoWi97]. Se define la semi-estructura como un grafo etiquetado, donde los nodos etiquetados contienen objetos y los arcos son referencias. Un objeto OEM consta de:

- a) una etiqueta nombre de la clase de objetos, opcionalmente puede llevar un oid (object identifier).
- b) un tipo que puede ser atómico (string, entero, etc.) o conjunto (set),
- c) un valor (uno y sólo uno) que puede ser atómico o un conjunto de objetos.

La figura 7.6 muestra un objeto OEM con la etiqueta '*BSDT 5403 Telecomunicación*' cuyos valores es un conjunto de las características docentes de la asignatura donde se usa este libro.

A este respecto, la nueva propuesta del estándar SQL-99 [ISOa99], [ISOb99], [ISOc99] y [ISOd99] realiza, entre otros [EiMe00], un importante avance en el SQL/MED que tiene especial importancia para el tema que ahora nos ocupa:

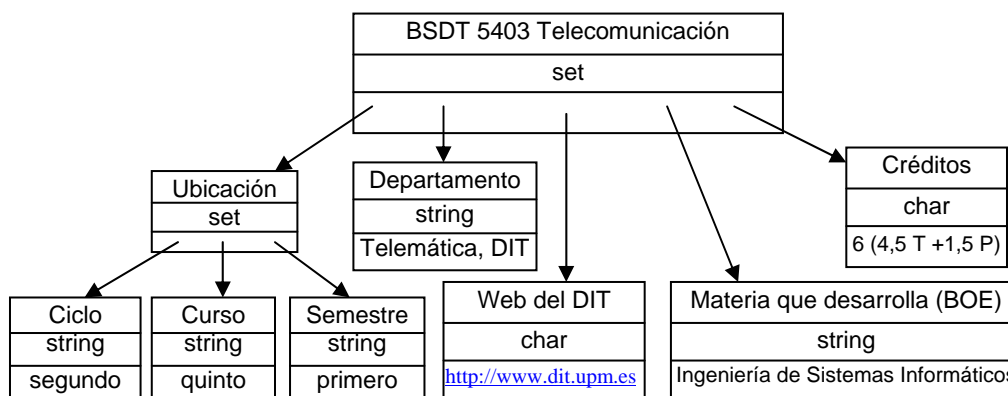


Fig. 7.6. Ejemplo de Objeto OEM para la asignatura BSDT 5403.

**Management of External Data**, abreviado como SQL/MED, se dirige a proporcionar soluciones para permitir que las aplicaciones usen **SQL para acceder a datos no-SQL** (datos en ficheros ordinarios, o en cintas magnéticas, o bases de datos jerárquicas o Codasyl, o incluso datos obtenidos en tiempo real, o datos no almacenados como los que devuelven los sensores). SQL/MED proporciona una API entre un servidor de SQL (base de datos local) y otra entidad llamada '**empaquetador de datos externos**' (*foreign-data wrapper*). Las casas comerciales tienden a establecer una estrecha cooperación en este asunto. Por ejemplo, Oracle tiene *Miracle: Open Transparent Gateway* y proporciona servicios de acceso a datos heterogéneos, Sybase tiene *OmniConnect* para proporcionar accesos a diferentes fuentes de datos y, finalmente, IBM proporciona el *DB2 DataJoiner* para poder acceder a datos tradicionales o no. El documento de SQL/MED puede encontrarse en el directorio */SC32/WG3/Progression\_Documents/FCD/* en el fichero: *fcdi-fcd1-med-1999-11.pdf (.ps o .txt)*.

Los esfuerzos de estandarización de SQL, en su secuencia de promulgaciones parecen no tener un final inmediato. Mientras cambien las necesidades o mientras la tecnología cambie de forma drástica, SQL continuará evolucionando para llegar a ser el *lenguaje intergaláctico de datos* (como dijo Mike Stonebreaker en 2000). Por ello, SQL está llamado a mejorar y seguir creciendo continuamente hasta alcanzar su final. Indiscutiblemente, el tema de la variopinta estructuración de los datos en la Web es un paso más (abierto a la investigación actual) sumado al de la heterogeneidad de las fuentes de datos.

Adicionalmente, el explosivo crecimiento de datos en la Web ha producido el desarrollo de sistemas que aplican el '*estilo de las bases de datos*' para consultar y gestionar los datos Web. Por ejemplo, *WebSQL* [MeMM97], *WebOQL* [ArMe98], *Strudel*, etc. Estos sistemas utilizan algunas técnicas y mecanismos que son exclusivos para la Web. Una referencia recomendada es [FILM98] por cuanto que describe una panorámica de sistemas para la gestión de datos en Web utilizando tecnología de bases de datos.

Finalmente, y como una valiosa panorámica del tema, en [DoDi99] se describe una clasificación de los actuales sistemas para consultar datos heterogéneos (estructurados, semi-estructurados o nada estructurados). La figura 7.7 muestra lo más relevante de esta aportación. En ella, las cajas sin sombra se han estudiado en esta asignatura y las cajas sombreadas se refieren al asunto que ahora nos ocupa.

Para los Sistemas Consultivos basados en el paradigma *Wrapper-Mediator* (figura 7.8) se propone una arquitectura virtual 'al-vuelo'. Su objetivo es permitir consultas distribuidas a múltiples fuentes de datos en Internet [OzVa99] y su enfoque es algo próximo a la arquitectura *Multidatabases* [LiAb87] (Cap. 2 de [OzVa99]).

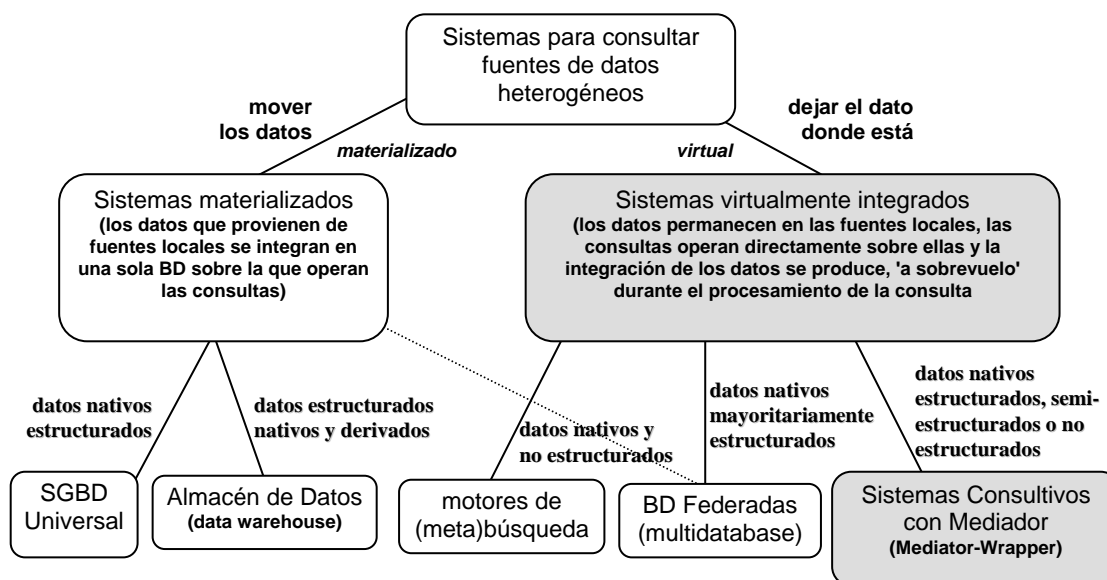


Fig. 7.7. Clasificación de los Sistemas para consultar datos heterogéneos.

Cada *wrapper* exporta al diccionario de datos global información sobre su esquema fuente, sus datos y sus posibilidades consultivas. Con ello, el Mediator centraliza la información recibida de los *wrappers* en una vista unificada de todos los datos disponibles (almacenados en el Diccionario de Datos global), descompone las consultas del usuario en pequeñas consultas (ejecutables por los *wrappers*) toma los resultados parciales y elabora la respuesta a la consulta del usuario web [Wied99].

Como se refleja en la figura 7.7, la arquitectura de la figura 7.8 difiere de la de los almacenes de datos en que en la primera los datos integrados no están materializados, son virtuales.

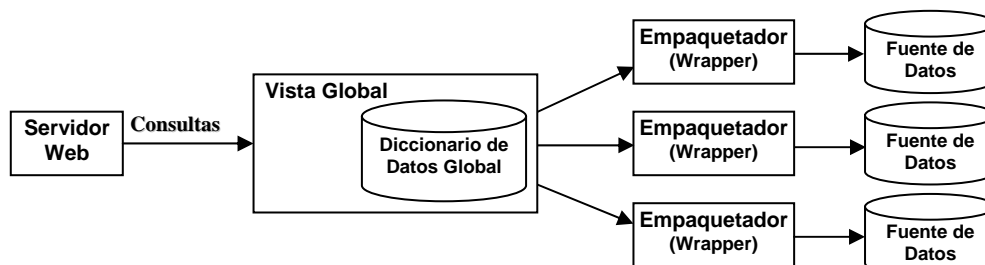


Fig. 7.8. Arquitectura Mediator-Wrapper.

No existe aún consenso sobre cómo puede describir el *wrapper* las posibilidades de su fuente de datos ni sobre cómo trabaja el mediador. La ventaja de este enfoque radica en que cada *wrapper* atiende a un dominio de información específico y particular, sobre el que puede conocer el nivel de estructuración de sus datos fuente, su semántica (si existe), etc.

Este tipo de arquitecturas virtuales se *instancian* 'al vuelo' (on-the-fly) y están basadas en estándares del W3c Consortium, como J2EE o 'web-centric' (<http://www.w3.org>). Dichas arquitecturas utilizan la invocación a Servicios Web (SOA, WSDL, UDDI, etc.) y/o Servicios Grid y se implementan mediante bibliotecas de Java [CRCF05], [RoCC06].

Finalmente, por tratarse de un enfoque de futuro y por venir del autor a quien debemos la Web, merece la pena señalar una de sus reciente perspectivas investigadoras. Literalmente, en [Bern05], Tim Berners-Lee dice lo siguiente:

*The key property of the WWW is its universality: One must be able to access it whatever the hardware device, software platform, and network one is using, and despite the*

*disabilities one might have, and whether one is in a “developed” or “developing” country; it must support information of any language, culture, quality, medium, and field without discrimination so that a hypertext link can go anywhere; it must support information intended for people, and that intended for machine processing. The Web architecture incorporates various choices which support these axes of universality.*

*Currently the architecture and the principles are being exploited in the recent Mobile Web initiative in W3C to promote content which can be accessed optimally from conventional computers and mobile devices. New exciting areas arise every few months as possible Semantic Web flagship applications. As new areas burst forth, the fundamental principles remain important and are extended and adjusted. At the same time, the principles of openness and consensus among international stakeholders which the WWW Consortium (W3C) employs for new technology are adjusted, but ever-important.*

Como conclusión final hay que resaltar que el acceso o consulta a múltiples sitios Web con datos heterogéneos y diferentes niveles de estructuración, como hoy se precisa en la Web, es un problema complejo abierto a la investigación y cuya solución inteligente y eficaz requiere de nuevos conceptos y técnicas sobre los que aún queda mucho camino por andar.

### **Referencias Bibliográficas.**

- [Abit97] Abiteboul S., *Querying Semi-Structured Data*, in *Proc. 6th Int. Conf. On Database Theory*, Vol. 1186 of Lecture Notes in Computer Science, Springer\_Verlag, January pp. 1-18, 1997.
- [ArMe98] Arocena G. and Mendelzon A., *WebOQL: Restructuring Documents, Databases and Webs*, Proc. of 14<sup>th</sup> Int. Conf. on Data Engineering (ICDE98), Florida, 1998.
- [BCGP92] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, Bernd Pollermann: *World-Wide Web: The Information Universe*. Electronic Networking: Research, Applications and Policy 1(2): 74-82 (1992)
- [BeCG92] Tim Berners-Lee, Robert Cailliau, Jean-François Groff: *The World-Wide Web*. Computer Networks and ISDN Systems 25(4-5): 454-459 (1992)
- [Bern97] Tim Berners-Lee: *World-Wide Computer*. Commun. ACM 40(2): 57-58 (1997)
- [Bern05] Tim Berners-Lee: *WWW at 15 years: looking forward*. WWW 2005: 1
- [BDFS97] Buneman P., Davidson S., Fernandez M. and Suciu D., *Adding Structure to Unstructured Data*, in *Proc. 6<sup>th</sup> Int. Conf. on Database Theory*, Vol. 1186 of Lecture Notes in Computer Science, Springer-Verlag, January pp. 336-350, 1997.
- [CoBV93] Carmen Costilla, M. J. Bas, J. Villamor: *SIRIO: A Distributed Information System over a Heterogeneous Computer Network*. SIGMOD Record, 22(1): 28-33, 1993.
- [Cost99] C.Costilla, *Sistemas de Bases de Datos. Conceptos, Técnicas y Lenguajes*, ISBN: 84-7402-271-1, 464 páginas, Servicio de Publicaciones, ETSI Telecomunicación, 1999.
- [CRCF05] Calleja A, Rodríguez MJ, Costilla C and Fernández R., *A Grid Semantic Approach for a Digital Archive Integrated Architecture*, in *The Third Int. Conf. on Information Technology and Applications (ICITA'05)*, Session: Agent, Datamining and Ontologies, ADO'05, Sydney, ISBN: 0-7695-2316-1, IEEE Computer Society, USA, pp. 227-232, July 2005.
- [CRPC05] Costilla C, Rodríguez MJ, Palacios JP, Cremades J, Calleja A and Fernández R, *A Contribution to Web Digital Archive Integration from the Parliamentary Management System 'SIAP'*, in 'Frontiers in Artificial Intelligence and Applications, Data Bases and Information Systems'. Selected papers from the Sixth Int. Baltic Conference DB&IS'2004, Vol. 118, Barzdins J. and Caplinskas A. (eds), ISBN: 1-58603-485-5, IOS Press, The Netherlands, pp. 273-287, 2005.
- [DoDi99] Domenig R. and Dittrich K.R., *An Overview and Classification of Mediated Query Systems*, in *ACM SIGMOD RECORD*, Vol. 28, N. 3, pp. 63-72, September, 1999.
- [EiMe00] Eisenberg A. and Melton J., *SQL Standardization: The Next Steps*, in *ACM SIGMOD Record*, 29(1), 2000.

- [FILM98] Florescu D., Levy A. and Medelzon A., *Database Techniques for the World Wide Web: A Survey*, SIGMOD RECORD, September, 1998.
- [GoEi01] Goodman, D., Eich, B., *JavaScript Bible*”, 4<sup>th</sup> Edition, Hungry Minds Inc., 2001.
- [GoWi97] Goldman R. and Widom J., *DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases*, in *Proc. 23<sup>rd</sup> Int. Conf. on Very Large Data Bases*, September, pp. 436-445, 1997.
- [ISO89] *Information Systems -Open Systems - Remote Database Access*. ISO/JTC 1/SC 21, 1989.
- [ISOa99] ISO/IEC 9075-1:1999, *Information Technology - Database language - SQL - Part 1: Framework (SQL/Framework)*, 1999.
- [ISOb99] ISO/IEC 9075-2:1999, *Information Technology - Database language - SQL - Part 2: Foundation (SQL/Foundation)*, 1999.
- [ISOc99] ISO/IEC 9075-3:1999, *Information Technology - Database language - SQL - Part 3: Call Level Interface (SQL/CLI)*, 1999.
- [ISOd99] ISO/IEC 9075-4:1999, *Information Technology - Database language - SQL - Part 4: Persistent Stored Modules (SQL/PSM)*, 1999.
- [LiAb87] W. Litwin and A. Abdellatif, *An overview of the Multidatabases Manipulation Language - MDL*, Proc. IEEE, Vol. 75, N° 5, pp. 621-631, May 1987.
- [MeMM97] Mendelzon A., Mihaila G. and Milo T., *Querying the World Wide Web*, J. of Digital Libraries, Vol. 1, N. 1, 1997.
- [Or9i01] *ORACLE 9i, SQL Reference*, disponible en: <http://technet.oracle.com/doc>, June 2001
- [OzVa99] Özsu T. and Valduriez P., *Principles of Distributed Database Systems*, 2nd edition, Prentice Hall, 1999.
- [PaGW95] Papakonstantinou Y., Garcia-Molina H. and Widom J., *Object Exchange across Heterogeneous Information Sources*, in *Proc. 11th Int. Conf. On Data Engineering*, March pp. 251-260, 1995.
- [RoCC06] Ramses Rodríguez, Carmen Costilla, Antonio Calleja: *A Topic-based approach to express dynamic capabilities of Semantic WS-Resources*, Int. Conf. on Internet and Web Applications and Services, AICIT/ICIW 2006, track: Web Services-based Systems and Applications (WEBSA 2006), IEEE Computer Society, USA, ISBN: 0-7695-2522-9, On page(s): 181- 181, Digital Object Identifier: 10.1109/AICT-ICIW.2006.39, 2006.
- [TRCH05] Philippe Thiran, Tore Risch, Carmen Costilla, Jean Henrard, Thomas Kabisch, Johan Petrini, Willem-Jan van den Heuvel, Jean-Luc Hainaut: *Report on the workshop on wrapper techniques for legacy data systems*. SIGMOD Record, 34(3): 85-86, 2005.
- [Wied99] Gio Wiederhold: *Meditation to Deal with Heterogeneous Data Sources*. INTEROP 1999: 1-16