

Parte final del Tema VII de BSDT Curso 2006-07

ÍNDICE DE CONTENIDO

Tema VII, Apartados VII.3 y VII.4

(Continuación del Capítulo 4 de Ozsu&Valduriez)

VII.3.- Arquitectura funcional del SGBD: Cliente/Servidor (2 capas) y Arquitectura Web (3 capas).	1
VII.3.1.- Arquitectura Cliente-Servidor (dos capas).	2
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.	6
VII.3.2.2.- Servidores de la capa intermedia en la Arquitectura Web.	7
VII.3.2.2.1.- Servidor Web: URLs y Protocolo HTTP.	8
VII.3.2.2.2.- La seguridad en Internet.	10
VII.3.2.2.3.- Gateways, CGI y Servidor de Aplicaciones.	10
VII.4.- Sobre la Estructuración de los Datos y Consultas a Datos Intensivos y heterogéneos en la Web.	13
VII.4.1.- Sobre la estructuración de los datos y consultas a datos intensivos y heterogéneos en la web.	14
Referencias Bibliográficas.	17

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.	4
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.	8
Fig. 7.5. Detalle del Servidor de Aplicaciones en la Arquitectura Web de un SGBD.	12
Fig. 7.6. Ejemplo de Objeto OEM para la asignatura BSDT 5403.	15
Fig. 7.7. Clasificación de los Sistemas para consultar datos heterogéneos.	16
Fig. 7.8. Arquitectura Mediador-Wrapper.	17

Tema VII, Apartados VII.3 y VII.4**ARQUITECTURAS CLIENTE / SERVIDOR (2 capas) Y WEB (3 capas). BASES DE DATOS WEB****VII.3.- Arquitectura funcional del SGBD: Cliente/Servidor (2 capas) y Arquitectura Web (3 capas).**

Los SGBDs, una de las variedades más complejas del software actual, precisan la utilización del término *arquitectura* con diversas acepciones y varios puntos de vista. Este epígrafe describe la arquitectura del SGBD bajo el punto de vista de sus formas operativas (o de funcionamiento), contemplado ahora como un ente individual que lleva a cabo las aplicaciones del usuario.

La arquitectura ANSI/X3/SPARC, descrita en el capítulo 1, explica cómo se organiza la información de una BD. En el capítulo 8 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 capítulo 9 hemos atendido 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 principios de la década de los 90's, se extendió con gran difusión en la tecnología de bases de datos, y marcó importantes pautas en la concepción arquitectural de los SGBD, separando nítidamente las funciones del Servidor y las funciones 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 se ha descrito, la distribución es el ingrediente más común en los actuales Sistemas de Información. En este capítulo se atiende a la arquitectura de un SGBD, visto como un ente individual (sin cooperar con otros SGBD) para llegar a entender los bloques que posibilitan su funcionamiento. Este punto no entra en detalles sobre cómo opera el motor o Servidor de Bases de Datos (vistos en el capítulo 9 de [Cost99]). Basta ahora con entender que dicho motor contiene el núcleo de las funciones de un SGBD y ha de ejecutarse en un ordenador donde la CPU, la memoria secundaria y la principal han de ser adecuadas para la gestión de datos masivos.

Ahora 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, es como 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 del usuario para ofrecer interfaces amigables a éste 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. En la primera, se describe la arquitectura a dos capas, llamada también arquitectura Cliente/Servidor del SGBD. La segunda describe 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 capítulo 8 vimos el paradigma Cliente-Servidor como el más simple dentro del amplio espectro de la distribución de los SGBDs. Un SGBD cuya arquitectura es de tipo Cliente-Servidor funciona separando limpiamente el papel que juega el Servidor del que desempeña el Cliente. Esta arquitectura, pensada para operar en modo distribuido, organiza la forma de llevar a cabo la ejecución de las aplicaciones entre máquinas distintas que están 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 (Oracle, por ejemplo, lo llama SQL*Net). Usando un solo motor SGBD, todas las máquinas son de tipo Cliente salvo una que es la 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 se ubiquen los procesos. Sin embargo, en la gestión de datos, normalmente los procesos del Servidor se ubican en una sola máquina y las máquinas de los Clientes se esparcen por la red de la institución o empresa para la cual se instala este tipo de arquitectura.

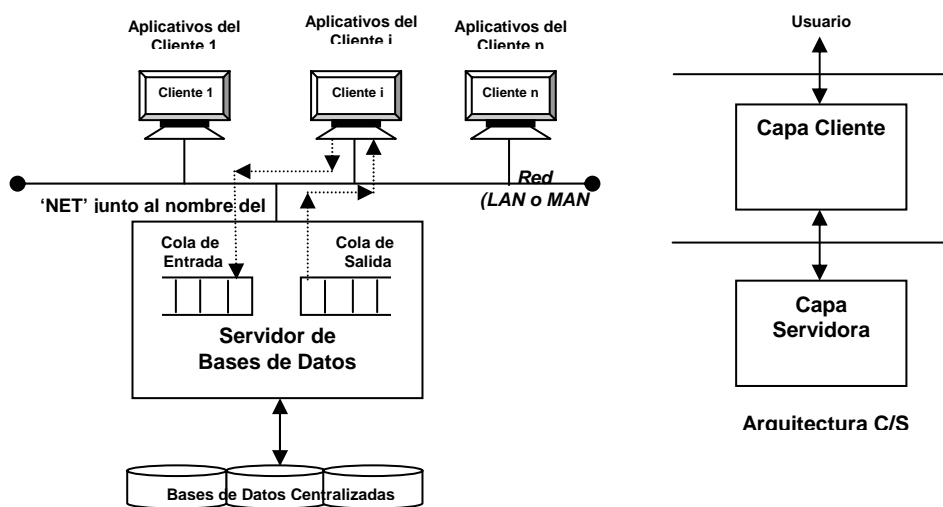


Fig. 7.1. Arquitectura Cliente-Servidor de un SGBD.

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.

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 solicitan y requieren la invocación de procesos (requests, queries, etc.) que ellos no poseen y 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 (Servidor de BDs) es único en este tipo de arquitectura y los Clientes quedan conectados a él mediante una red (LAN casi siempre) cuya forma de 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 preprocesan para transformarlas en otras unidades que envían al Servidor para solicitar sus servicios. Los Clientes son los que 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 las 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. Cada 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 el administrador de las bases de datos, trabajarán en la capa Servidora para realizar las tareas propias y relativas a temas de diseño, asignación de roles de usuario, permisos y privilegios, 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 de sus 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 suele bastar con un típico ordenador personal que proporcione diversas herramientas de un entorno ofimático. Entre dichas herramientas, casi siempre enmascaradas por unas interfaces amigables para el usuario, tiene que tener instaladas las destinadas a solicitar aplicaciones al Servidor de Bases de Datos, como muestra la figura 7.2.
- El lenguaje SQL de los SGBDR es ideal para identificar los *servicios de interfaz* de la capa cliente, que es la capa responsable de interpretar la consulta del usuario, darla formato y presentar los resultados al usuario de forma adecuada. La consulta, escrita en la parte Cliente, se envía al Servidor para que allí sea procesada. El Servidor extrae los resultados de la base de datos, los empaqueta y, finalmente, se lo devuelve al Cliente que hizo la solicitud. De esta forma, por la red circula la *mínima información útil* [Cost88]. La consulta SQL del usuario admite dos formas de invocación diferentes.

La primera, conocida como *consulta compilada de forma estática* donde ésta 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 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 representa 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. La gestión de la colas del Servidor se lleva a cabo por el *dispatcher*.

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

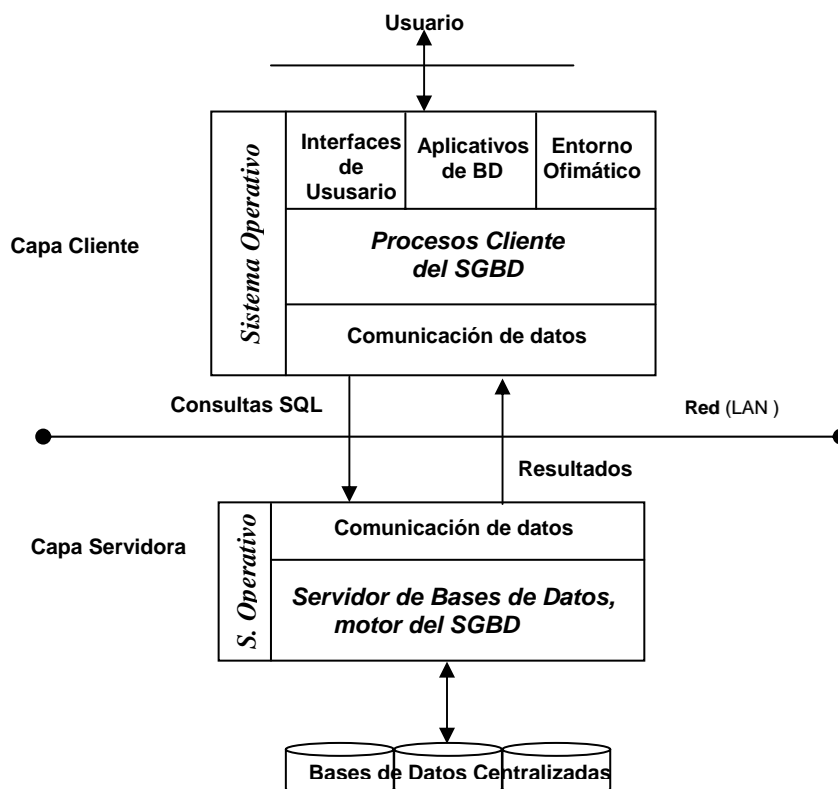


Fig.7.2. Detalle de la capa Cliente de un SGBD con arquitectura C/S.

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

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

La espectacular evolución de Internet plantea grandes retos para mejorar su potencialidad y actuales soluciones. La tecnología world-wide-web (www) es y será el substrato de interconexión de la sociedad

de la información actual y futura. Su ritmo de expansión y la diversidad de usuarios crece 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 este tipo 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 ya conocemos que están y existen en una base de datos.

La web empezó siendo una interfaz para el acceso a documentos distribuidos, y hoy es una plataforma para los sistemas de información de todo tipo. La web es un paradigma que permite difundir y adquirir cualquier tipo de información a través de una arquitectura, configurada en capas que, en el caso de un SGBD con arquitectura web, son las tres siguientes (véase figura 7.3):

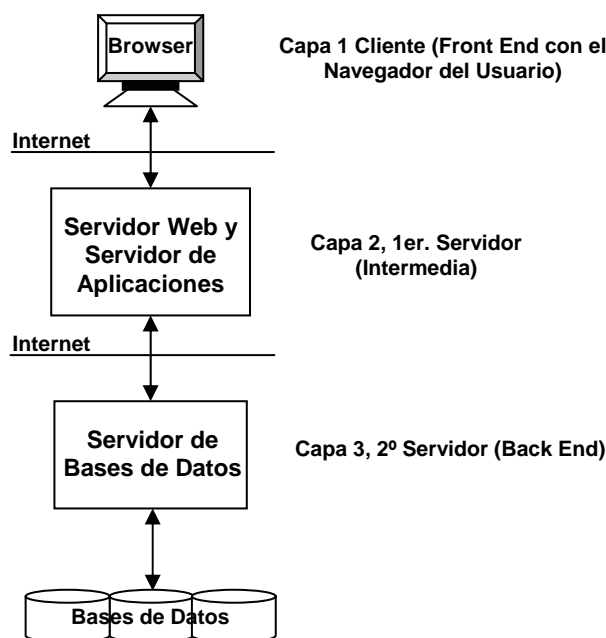


Fig. 7.3. Arquitectura Web Abreviada (a tres capas) de un SGBD.

Capa frontal (Front End), formada por una herramienta genérica llamada navegador¹ o *browser* (como *Netscape Navigator* o *Microsoft Internet Explorer*), que constituye la interfaz de la máquina del usuario para navegar por dicha información. 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.

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

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*.

Capa dorsal (Back End), llamada Servidor de Información o **Servidor de Bases de Datos**, descrito con detalle en el capítulo 8.

Casi toda la actual tecnología de bases de datos ya 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 muestra 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 muestra transparente (a los usuarios y a los programas), de tal forma que el usuario puede referenciar² 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 capítulo 8.

El protocolo TCP permite que ordenadores de todos los tamaños (de cualquier fabricante y con sistemas operativos totalmente 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. Y 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 Internet. El documento recibió el nombre de *hipertexto*³ 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

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

³ Hiper, prefijo griego que significa "exceso".

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, sino que también permite lanzar programas para que éstos generen dinámicamente documentos nuevos (en páginas dinámicas).

El epígrafe 10.3.2 describe los principales componentes tecnológicos que posibilitan la implementación de estas ideas que acaban de esbozarse para 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 extraídas⁴ en cada acceso.

El éxito alcanzado por la web ha puesto en evidencia que existe 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 modelos de datos y lenguajes consultivos) 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 (de la figura 7.3) destacando la segunda capa donde está la primera máquina Servidora en la cual se incluyen dos tipos de servicios: el proporcionado por el Servidor Web y el del Servidor de Aplicaciones, descritos a continuación.

⁴ Extraer, sacar algo que está incrustado en el documento.

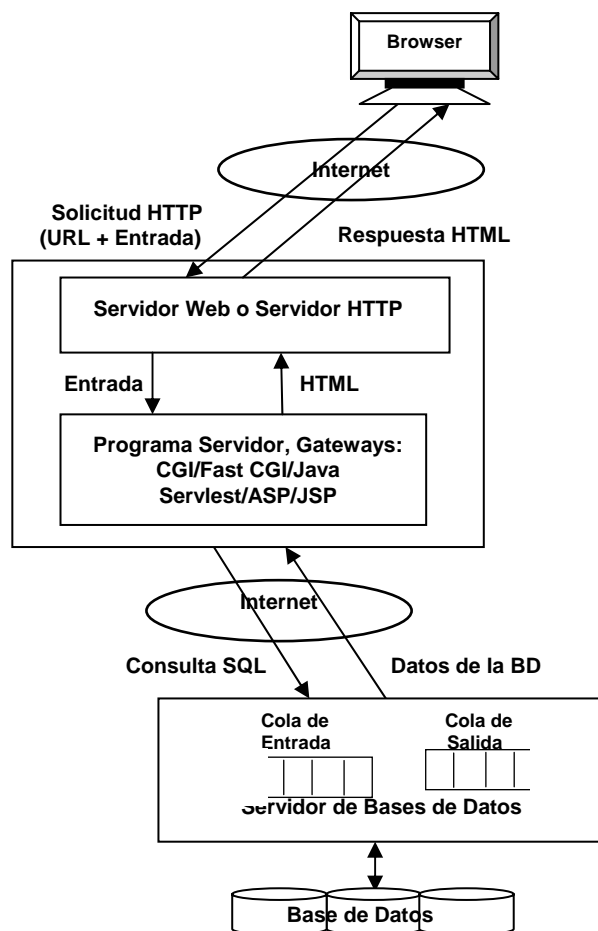


Fig. 7.4. Detalle de la capa 2 de la Arquitectura Web de un SGBD.

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

La arquitectura web es una variante de C/S, ya descrita, que se basa 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 URL⁵ (Uniform Resource Locator).

Como indica la figura 7.4, la web trabaja sobre **Servidores HTTP**, más conocidos como **Servidores Web**. El protocolo HTTP es de naturaleza muy simple y consta de cuatro fases:

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);

⁵ 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).

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 con la solicitud de un servicio dando el nombre del recurso invocado y los posibles parámetros de la invocación (llamado Entrada en la figura 7.4);

d) *Recibir Respuesta*, el servidor devuelve al cliente el resultado del servicio requerido (llamado Respuesta HTML en la figura 7.4), y se cierra la conexión por el servidor sin retener información alguna para ser usada en conexiones subsiguientes.

La web hace uso de la naturaleza distribuida de los hipertextos y, mediante un esquema cliente-servidor, proporciona 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 puede ser usado 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 del protocolo 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 sobre 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) el **Protocolo** 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 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 el valioso medio por el que se establecen 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 redundar en una alta pesadez 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 donde se requiere 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 de comunicaciones informático, la web precisa una serie de herramientas de seguridad asociadas como son:

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

Aunque HTTP intenta abordar algunos de estos problemas, la seguridad se ve muy mejorada en 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 información pedida 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.

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) es el mecanismo de comunicación y suele estar 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*⁶ 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 en curso se pueden especificar añadiéndoles 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 este libro.
2. El Servidor Web lanza el *gateway* (también llamado *programa CGI*) según el protocolo CGI y le transmite los parámetros.
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).

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

4. Cuando el *gateway* recibe 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 este resultado al *browser* (llamado Respuesta HTML en la figura 7.4).

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

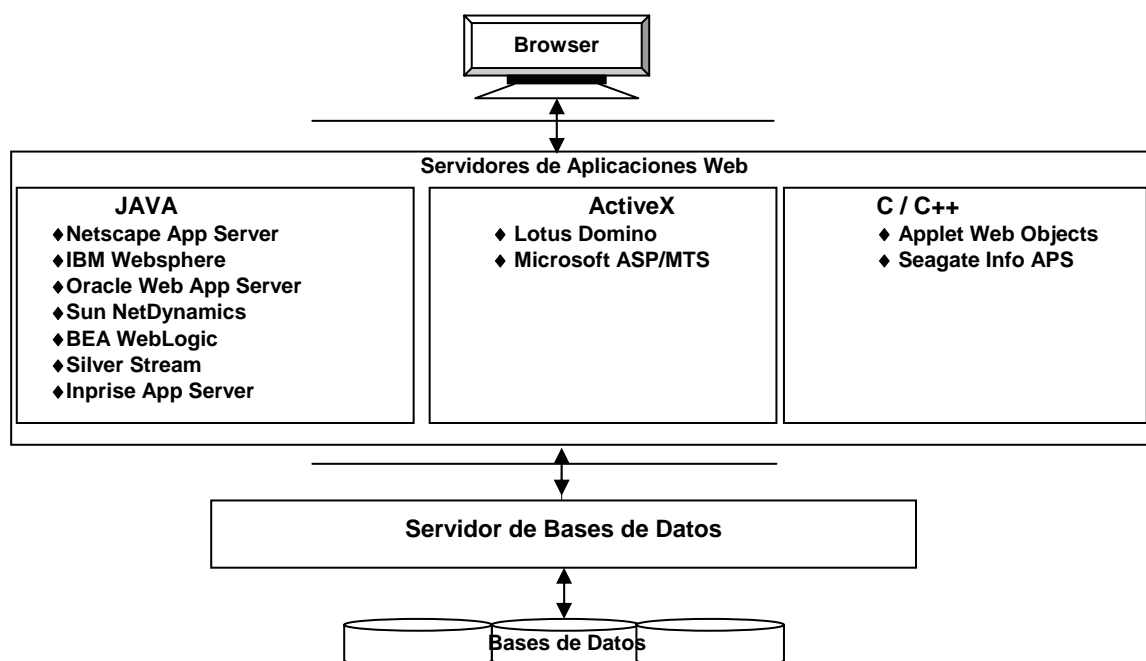


Fig. 7.5. Detalle del Servidor de Aplicaciones en la Arquitectura Web de un SGBD.

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 el 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 capítulo 9 [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 la de 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 soportan y proporcionan 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 el desarrollo basado en *e-commerce* y procesos de toma de decisiones. Los *stateful servers* son aquellos que juegan el rol de coordinadores de transacciones y gestionan el estado de los datos (se estudiarán 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 (descrito en capítulo 2 [Cost99]) y del Diseño de Bases de Datos Relacionales (descrito en capítulo 4 [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, podríamos percibir que toda la web es como una fuente de información unificada. Esto no sería útil ya que la web adolece bastante de coherencia en la forma de presentar la *información útil al usuario*, sobretodo 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*⁷ para señalar sitios web, gestionados por el navegador y los buscadores (como Alta Vista, Yahoo, etc.) ocupados en seleccionar enlaces a páginas simples, cuyas técnicas son las 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 que se exija en los objetivos de

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

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 nuevo ingrediente: 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.

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 informe.

En web, el resultado de una consulta a una BD toma la forma de un 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. Se puede decir que este asunto está hoy bastante bien resuelto y será tratado 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.

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 tal manera, 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:

- una etiqueta que es el nombre de la clase de objetos, opcionalmente puede llevar un oid (object identifier).
- un tipo que puede ser atómico (string, entero, etc.) o conjunto,
- 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.

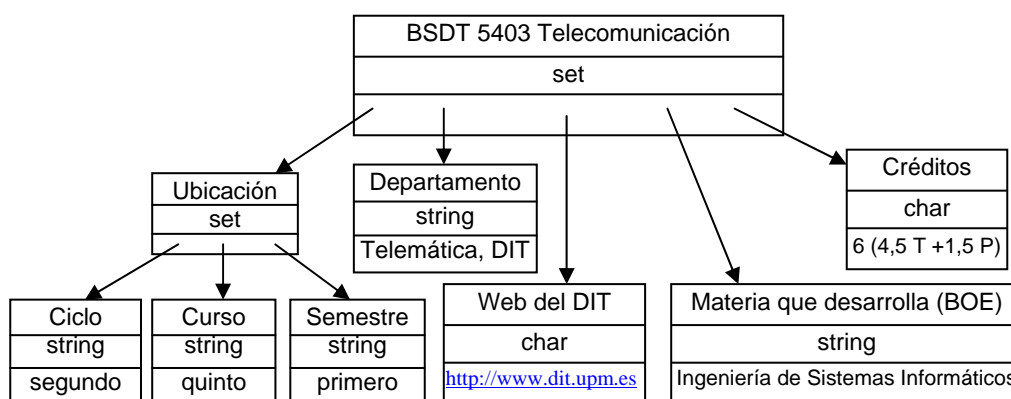


Fig. 7.6. Ejemplo de Objeto OEM para la asignatura BSDT 5403.

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:

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/* dentro del fichero: *fcdi-fcd1-med-1999-11.pdf* (o *.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). Por ello, SQL está llamado a mejorar y seguir creciendo probablemente de forma continua hasta alcanzar su final. Indiscutiblemente, el tema de la variopinta estructuración de los datos en la web es un paso más (tema abierto a la investigación actual) que se añade al de la heterogeneidad de diferentes 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* [FFKL97], etc. Estos sistemas utilizan alguna 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 por tratarse de 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 sombreadas se refieren al asunto que ahora nos ocupa.



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

Por otro lado, la figura 7.8 muestra la arquitectura propuesta para los Sistemas Consultivos con Mediador y Empaquetador. Su objetivo es permitir consultas distribuidas a múltiples fuentes de datos en Internet [OzVa99] y su enfoque está muy cerca a la arquitectura de las *Multidatabases* descrita en el capítulo 2.

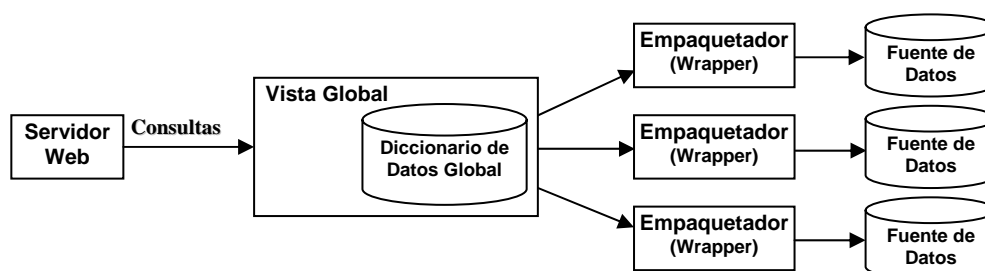


Fig. 7.8. Arquitectura Mediator-Wrapper.

Cada *wrapper* exporta al diccionario de datos global información sobre su esquema fuente, sus datos y sus posibilidades consultivas. Con ello, el Mediador 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.

La arquitectura de la figura 7.8 difiere de la de los almacenes de datos en que los datos integrados no están materializados, como se refleja en la figura 7.7.

No existe aún consenso sobre cómo puede el wrapper describir 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.

Como conclusión final a este punto 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 que está abierto a la investigación y cuya solución inteligente y eficaz requiere de nuevos conceptos y técnicas sobre los que aún queda mucho por trabajar.

Este tipo de arquitecturas se instancian ‘al vuelo’ (on-the-fly) y están basadas en estándares del w3c World Wide Web 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. Para los interesados, se señala que en algunas publicaciones que figuran en <http://sinbad.dit.upm.es> se pueden encontrar varias referencias recientes que tratan de la investigación realizada actualmente en este campo (aquí no incluidas).

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.
- [ANSI75] ANSI/X3/SPARC, *Study Group on Data Base Management Systems: Interim Report*, FDT (ACM SIGMOD bull.), Vol.7, N.2, 1975.
- [ANSI78] ANSI/X3/SPARC Study Group Database Management Systems, *"Framework Report on Database Management Systems"*, AFIPS Press, 1978; also published in *"The ANSI/X3/SAPARC DBMS Framework"*, Information Systems, Vol.13, N 3, 1978.
- [ANSI85] ANSI/X3/SPARC *Reference Model for the DBMS Standardization*, Database Architecture Framework Task Group of the ANSI/X3/SPARC, in ACM SIGMOD RECORD, Vol.15, N.1, March, 1986.
- [ANSI89] *The SQL Standard*, Propuesta que coincide con el SQL de [ISO87], 1989.

- [Anst98] Anstey D., *ORACLE8 Object-Oriented Design High Performance*, Coriolis, 1988.
- [ArMe98] Arocena G. and Mendelzon A., *WebOQL: Restructuring Documents, Databases and Webs*, Proc. of 14th Int. Conf. on Data Engineering (ICDE98), Florida, 1998.
- [BDFS97] Buneman P., Davidson S., Fernandez M. and Suciu D., *Adding Structure to Unstructured Data*, in Proc. 6th Int. Conf. on Database Theory, Vol. 1186 of Lecture Notes in Computer Science, Springer-Verlag, January pp. 336-350, 1997.
- [BeHG87] *Concurrency Control and Recovery in Database Systems*, P.A.Bernstein, V.Hadzilacos and N.Goodman, Addison Wesley, 1987.
- [Boba96] A.R. Bobak, *Distributed and Multi-Database Systems*, Artech House, 1996.
- [Cast00] Castro, E., *HTML 4 for the World Wide Web (Visual Quickstart Guide)*, 4th Edition, 2000.
- [Catt00] R.G.G. Cattell (de.), *The Object Database Standard: ODMG 3.0*, Morgan Kaufmann, 2000.
- [CePe84] *Distributed Databases. Principles and Systems*, S.Ceri and G.Pelagati, Mc Graw Hill, 1984.
- [CoBV93] C.Costilla, M.J. Bas, J.Villamor, *SIRIO: A Distributed Information System over a Heterogeneous Computer Network*, ACM SIGMOD RECORD, Vol. 22, N° 1, pp. 28-33, March, 1993.
- [Cost88] Costilla C. , *A Contribution to Knowledge Communication in Distributed Knowledge-Based Systems, in Research into Networks and Distributed Applications*, EUTECO'88 (R.Speth, de.), pp. 1153-1162, North-Holland, 1988.
- [Cost90] C.Costilla, *Estado Actual de las Bases de Datos Distribuidas. Evolución desde la Tecnología Relacional*, en Computerworld, N. 400, Junio, 1990.
- [Cost92] C.Costilla, *Avances Recientes y Tendencias Previsibles de las Bases de Datos*, en "Encuentro sobre Bases de Datos en la Administración Pública", De. MAP, ISBN:84-7088-618-5, pp. 119-145, Madrid, Mayo, 1992.
- [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.
- [DAIS88] DAISY Group, *DAISY: Distribution Aspects in Information Systems*, in "Research into Networks and Distributed Applications" (R.Speth, de.), pp. 1029-1049, North-Holland, 1988.
- [Date87] C.J. Date, *A Guide to INGRES*. Addison-Wesley, ver su capítulo 21 "Distributed Database Support", 1987.
- [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, Vol. 29, No. 1, March 2000.
- [FFKL97] Fernandez M., Florescu D., Kang J. and Levy A., *STRUDEL: A Web Site Management System*, in Proc. of the 1997 ACM SIGMOD, Vol. 26, N. 2, pp. 549-552, Arizona, May, 1997.
- [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*, 4th Edition, Hungry Minds Inc., 2001.
- [GoWi97] Goldman R. and Widom J., *DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases*, in Proc. 23rd Int. Conf. on Very Large Data Bases, September, pp. 436-445, 1997.
- [GuPe99] Gulutzan P. and Pelzer T., *SQL-99 Complete, Really. An example-Based Reference Manual of the New Standard*, R&D Books, 1999.
- [Gupt89] A. Gupta (de.), *Integration of Information Systems: Brindging Heterogeneous Databases*, IEEE Press, 1989.
- [IHIS89] DAISY Group, *IHIS: Interoperability of Heterogeneous Information Systems*, A Report from de DAISY Group of the Cost11 ter european action in Teleinformatics. Commission of European Communities, DG XIII, (Tirri, de.), Dpt. Of Computer Science, Univ. Helsinki, Finland, ISBN 951-45-4597-4, March 1989.
- [HsNS93] D.K. Hsiao, E.J. Neuhold and R. Sacks-Davis (eds.), *IFIP Transactions: Interoperable Database Systems (DS-5)*, North-Holland, 1993.
- [IRO94] G.Attali, M.Bas, C.Costilla, P.Fankouser, B.Finance, G.Gardarin and W.Klas, *IRO-DB: Interoperable Relational and Object Databases; General Functionality Specification Document*, Identifier: IRO/SPEC/IBER-UPM/ D1-1.1/CC940730, 30 July, 1994.
- [ISO87] *Information Processing Systems -Database language- SQL*. ISO 8975: 1987.

- [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.
- [JaKS88] S. Jajodia, W.Kim and A. Silberschatz (eds.), *Proceedings: International Symposium on Databases in Parallel Distributed Systems*, IEEE Computer Society Press, 1988.
- [LiAb87] W. Litwin and A. Abdellatif, *An overview of the Multidatabases Manipulation Language - MDL*, Proc. IEEE, Vol. 75, Nº 5, pp. 621-631, May 1987.
- [Mals88] F. Malsall, *Data Communication, Computer Networks and OSI*, Addison- Wesley, 1988.
- [MeMM97] Mendelson A., Mihaila G. and Milo T., *Querying the World Wide Web*, Journal of Digital Libraries, Vol. 1, N. 1, 1997.
- [OMG91] OMG, *The Common Object Request Broker: Architecture and Specification*, OMG Doc. N. 91.12.1, Published by Object Management Group and X/Open, 1991.
- [Or9i01] *ORACLE 9i, SQL Reference*, disponible en: <http://technet.oracle.com/doc>, June 2001
- [OSDT89] *OSDT, Open Systems Data Transfer*, advising in Open Systems Interconnection Standards, Technology and Products, ISSN 0741286X, December, 1989.
- [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.
- [SiZd97] Silberschatz A. and Zdonik S., *Database systems-breaking out of the box*, ACM SIGMOD RECORD Vol. 26, n.3, September 1997.
- [SrCh99] J.Srivastava and P.Y.Chen, *Warehouse Creation - A Potential Roadblock to Data Warehousing*, IEEE Transaction on Knowledge and Data Engineering, Vol. 11, N.1, pp. 118-126, January/February, 1999.