

CLIENTE/SERVIDOR GUÍA DE SUPERVIVENCIA

2a. edición

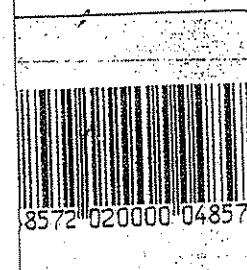
Robert Offali • Dan Harkey • Jeri Edwards

Traducción:

Enrique Mercado González

Revisión Técnica:

Ing. Mario Rodríguez Manzanera



McGRAW-HILL

MÉXICO • BUENOS AIRES • CARACAS • GUATEMALA • LISBOA • MADRID

NUEVA YORK • SAN JUAN • SANTAFE DE BOGOTÁ • SANTIAGO • SÃO PAULO

AUCKLAND • LONDRES • MILÁN • MONTREAL • NUEVA DELHI

SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY • TORONTO

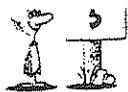
E Y C O

TEC
710
AFRICA
24-2000

Parte 1

Panorama general





ción del sistema estaban integrados a cada componente. Si algo fallaba, ya se sabía a quién recurrir: al ingeniero de sistemas de la compañía.

Si era necesario generar aplicaciones, el proveedor ofrecía la serie "indicada" de metodologías jerarquizadas, las herramientas convenientes y los subsistemas de tiempo de ejecución. Se disponía, por supuesto, de planes quinquenales y grandes arquitecturas para el diseño del crecimiento futuro y su correspondiente presupuestación. Pero lo más importante era que todas las personas integradas a la industria de la computación gozaban de seguridad en el empleo y tenían asegurados una trayectoria profesional y un brillante futuro. Asimismo, los proveedores de macrocomputadoras obtenían jugosas ganancias, como también quienes atendían comercialmente los nichos entre sus órbitas. Así fueron los buenos tiempos anteriores a la revolución de cliente/servidor y sistemas "abiertos".

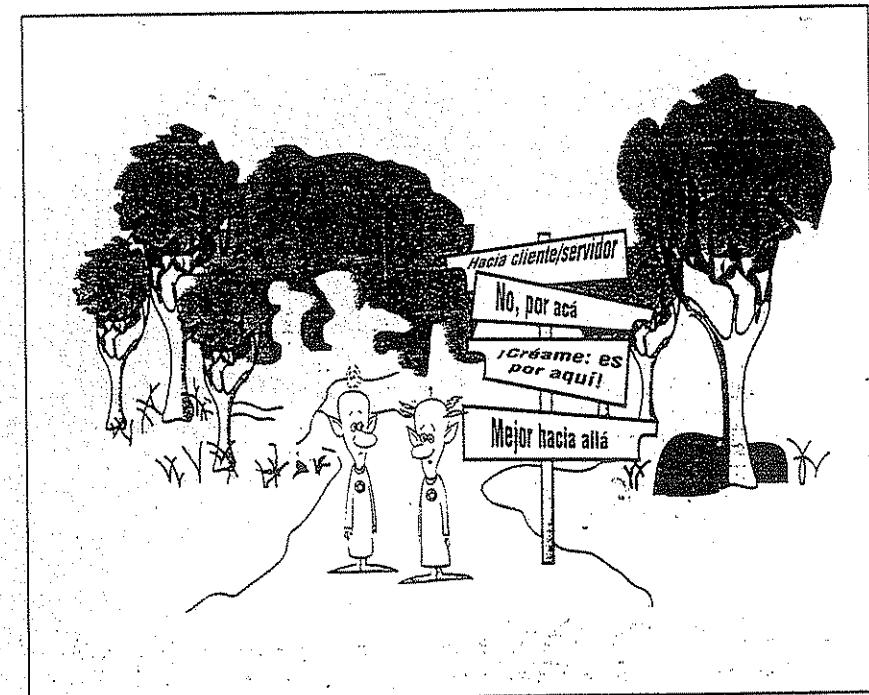
LA VIDA DESPUÉS DE LA REVOLUCIÓN

Pero las cosas ya no son tan sencillas en el nuevo mundo de cliente/servidor y sistemas abiertos. La computación de cliente/servidor es la "plataforma abierta" por excelencia. Cliente/servidor le da a usted la libertad de mezclar e igualar componentes en casi cualquier nivel. Le permite reunir una increíble variedad de combinaciones de clientes y servidores en red. En el mundo de cliente/servidor, todo se vende *a la carta*.

En cada oportunidad se verá usted frente a un menú de opciones digno de un restaurante de comida china: ¿Qué plataforma de servidor? ¿Cuál plataforma de cliente? ¿Qué protocolos de red? ¿Cuál infraestructura de computación distribuida? ¿Cuál servidor de bases de datos? ¿Qué clase de *middleware*? ¿Qué base de administración de sistemas? Y después de seleccionada la primera serie de opciones, tendrá que tomar decisiones aún más complejas en el área de desarrollo de aplicaciones y herramientas de cliente/servidor. Existen al menos cinco grandes tecnologías que se pueden usar para la creación de aplicaciones cliente/servidor: servidores de bases de datos, monitores de TP, *groupware*, objetos distribuidos e Intranets. ¿Cuál es la mejor?

El único que puede tomar las decisiones difíciles en este nuevo orden mundial es usted. Para acertar, debe elegir la plataforma, herramientas, proveedores y base de arquitectura cliente/servidor *correctos*. Debe identificar la ola tecnológica de cliente/servidor correcta y sumarse a ella. Si la ola dominante es la de objetos distribuidos, sería absurdo invertir tiempo y energía en los servidores de bases de datos. Sin embargo, si usted opta prematuramente por la ola de los objetos, corre el riesgo de hacer naufragar a su empresa. De ahí que sea tan importante que esté perfectamente enterado de lo que la tecnología puede hacer por usted en un momento dado. Para determinarlo, debe ser capaz de seleccionar su camino en medio de los lemas comerciales y los ofrecimientos de arquitectura. Sobre todo, debe saber con precisión qué pueden hacer *hoy* por usted los productos ya existentes.

Lo bueno de todo esto es que la tecnología de cliente/servidor es liberadora, flexible y le permite hacer las maravillas que describiremos más adelante en este libro. Lo malo es que usted está abandonado a su suerte. Los proveedores le venderán productos a muy buen precio, pero será usted quien tendrá que descubrir cómo unir las piezas y hacerlas trabajar en conjunto. Si no funcionan, es su problema. Al comprar estos productos no se le harán promesas de ningún tipo.



costosos. ¿Qué pasó entonces con el hermoso ayer? Ya se fue. Ahora los proveedores venden componentes independientes. Todo está desarmado. Incluso el servicio se cotiza por separado. En este nuevo mundo a la carta, el integrador del sistema es usted.

PLAN DE SUPERVIVENCIA

Los autores de este libro llevamos ya algún tiempo recorriendo esta selva, y el resultado es la *Guía de supervivencia básica cliente/servidor*. Esta guía le permitirá sobrevivir, lo que no será fácil. Desafortunadamente, nadie cuenta con un mapa mágico con todos los caminos correctos. Compartiremos con usted nuestros hallazgos, los que, combinados con los suyos, quizás le ayuden a seguir el camino menos riesgoso. Una guía de supervivencia es más que un mapa. Contiene instrucciones para encontrar alimentos, construir refugios, cruzar territorios desconocidos y protegerse de serpientes y escorpiones. Nosotros le ofreceremos todo esto.

Primero explicaremos las partes de cliente y servidor de la ecuación. Después trataremos la diagonal (/) de cliente/servidor: el pegamento que une al cliente con el servidor. Más tarde describiremos las cinco principales tecnologías para el desarrollo de aplicaciones cliente/servidor: servidores de bases de datos, monitores de TP, *groupware*, objetos distribuidos e Internet. Concluiremos ocupándonos de las plataformas y herramientas de administración de sistemas.

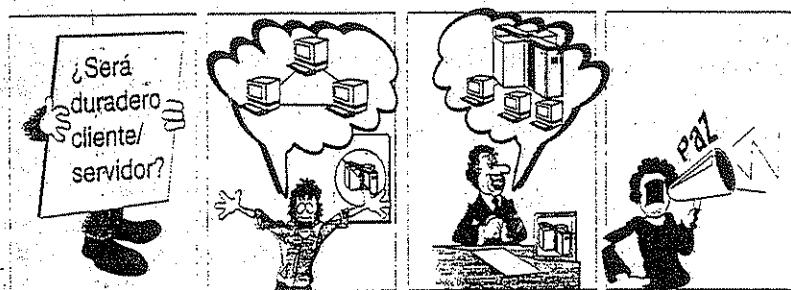
LA ERA DE LA COMPUTACIÓN DE CLIENTE/SERVIDOR

La computación de cliente/servidor es un movimiento irresistible que está cambiando por completo la forma en la que se usan las computadoras. Y aunque se trata de un movimiento relativamente reciente, se halla ya en todo su apogeo y no está dejando sin tocar ni una sola de las facetas de la industria de la computación. Además, Internet es como un maremoto, pues está introduciendo una nueva modalidad de la computación de cliente/servidor: el cliente/servidor *intergaláctico*. Constituye una revolución dentro de la revolución.

¿Qué forma adoptará el desafiante mundo nuevo de la computación de cliente/servidor? ¿Qué efectos tendrá en las empresas —oficinas— de sistemas de información? ¿Qué significa competir en un mercado abierto de computación de cliente/servidor? ¿Qué nuevas oportunidades surgirán para los desarrolladores de software? Intentemos contestar ya estas preguntas.

¿Cuál es la visión real de cliente/servidor?

La computación de cliente/servidor se distingue por contar con ferreos defensores a todo lo largo de la industria de las computadoras. Para las multitudes que sostienen que "las PC lo pueden todo", la computación de cliente/servidor significa el abandono de toda macrocomputadora incapaz de adaptarse al entorno del escritorio y la desaparición de la computación centralizada. Para los fanáticos de las macrocomputadoras, la computación de cliente/servidor significa el surgimiento de una nueva especie de macrocomputadoras en red "renacidas" que harán volver al redil a todas las PC de las empresas. Para quienes se hallan a medio camino entre ambos extremos, cliente/servidor es la "*glasnost* de la computación", lo que en realidad quiere decir una nueva era de coexistencia y apertura en la que todos pueden participar.



Cada una de estas visiones tiene algo de verdad. La computación de cliente/servidor proporciona un entorno abierto y flexible en el que mezclar e igualar es la regla. Las aplicaciones del cliente correrán predominantemente en PC y otras máquinas de escritorio de las redes de área local (LAN: *local area network*). Los exitosos servidores también se sentirán a gusto en LAN y sabrán exactamente cómo comunicarse con sus clientes en PC. Las PC más potentes representan superservidores naturales. Sin embargo, para que las macrocomputadoras tengan éxito como servidores, deberán aprender a convivir en la LAN con las PC en calidad de igual a igual. En este mundo de igualdad, los servidores con base en macrocomputadoras ya no podrán tratar a

las PC como terminales tontas. Tendrán que soportar protocolos de igual a igual, interpretar mensajes de PC, atender a los archivos de sus clientes en PC en sus formatos originales y ofrecerles a las PC datos y servicios en la forma más directa posible. Finalmente, ganará la plataforma de servidores con el mejor costo/desempeño y la mayor cantidad de servicios.

Cliente/servidor y el "nuevo IS"

El desarrollo de aplicaciones cliente/servidor requiere de habilidades híbridas, entre las que están el procesamiento de transacciones, el diseño de bases de datos, experiencia en comunicaciones y conocimientos sobre la interfaz gráfica del usuario. Las aplicaciones más avanzadas requieren conocimientos de objetos distribuidos e Internet. Para el dominio de estas habilidades será necesaria la existencia de nuevos programadores capaces de combinar las mejores, más firmes y confiables ideas con las tradiciones de LAN de PC. ¿De dónde saldrán estos nuevos programadores? Las empresas de sistemas de información (IS: *information systems*) serán capaces de ofrecer soluciones y servicios en este nuevo entorno de computación? ¿O acaso tales servicios tendrán que ser prestados por consultores e integradores de sistemas que hayan dedicado tiempo a la adquisición de esas nuevas habilidades?

La mayoría de las soluciones actuales de cliente/servidor son implementaciones de LAN de PC personalizadas para el grupo que las usa. Todo, desde directorios de LAN hasta requerimientos de seguridad, debe ser adecuadamente configurado a menudo por los propios usuarios. Los departamentos de Sistemas de Información (IS) cuentan con la capacidad no sólo para administrar y desplegar grandes redes, sino también para ofrecer estándares de interoperabilidad. También saben afinar aplicaciones, distribuir tareas y asegurar la integridad de los datos. Los Sistemas de Información (IS) se ocupan tradicionalmente de grandes centros de datos, no de los departamentos de línea que poseen PC y LAN. En su caso, la clave consiste en hacer lo que saben hacer en un entorno de cliente/servidor distribuido en el que comparten capacidad, responsabilidad, habilidades de computación y presupuestos financieros con los gerentes de línea de las empresas (los usuarios finales). En consecuencia, la distribución de la función de Sistemas de Información (IS) es esencial.

La computación de cliente/servidor puede rendir mejores resultados en organizaciones de Sistemas de Información (IS) en dos líneas: un Sistema de Información (IS) de línea para la administración y despliegue de sistemas departamentales y un Sistema de Información (IS) empresarial para la administración de la red global y el establecimiento de los estándares de interoperabilidad. Este tipo de federación alineada no sólo preservará la autonomía departamental, sino que además permitirá que las LAN locales formen parte de la red global de servidores y proveedores múltiples.

La competencia en el mercado de cliente/servidor

Cliente/servidor, el *gran igualador* del ramo de la computación, alienta la apertura y representa un campo de juego nivelado en el que puede participar una amplia variedad de plataformas de cliente y servidor. El entorno de cliente/servidor abierto actúa como catalizador para la "conversión en mercancías" del *hardware* y el *software* del sistema. La PC es un buen ejemplo de mercancía de cómputo; puede obtenerse con múltiples proveedores y se le vende en situacio-



Las características de cliente/servidor aquí descritas permiten la fácil distribución de inteligencia a lo largo de una red. Estas peculiaridades también ofrecen una estructura para el diseño de aplicaciones de acoplamiento holgado basadas en redes.

¿EXISTE UN VERDADERO CLIENTE/SERVIDOR?

A muchos sistemas con muy diferentes arquitecturas se les ha llamado "cliente/servidor". Los proveedores de sistemas suelen emplear el término cliente/servidor como si se aplicara únicamente a sus paquetes específicos. Por ejemplo, los proveedores de servidores de archivos juran haber inventado el término, mientras que en ciertos círculos a los proveedores de servidores de bases de datos se les conoce sencillamente como *los* proveedores de cliente/servidor. Para complicar aún más las cosas, en este libro se añaden los objetos distribuidos, monitores de TP, groupware e Internet a la lista de tecnologías de cliente/servidor. ¿Quién está en lo cierto? ¿Cuál de estas tecnologías es el verdadero cliente/servidor? La respuesta a estas dos preguntas es todo lo que se dijo antes.

La idea de dividir una aplicación a lo largo de líneas de cliente/servidor se ha utilizado en los últimos diez años para crear diversas modalidades de soluciones de software para red de área local (LAN). Por lo general estas soluciones se venden como paquetes comerciales de software, y en muchos casos las distribuyen varios proveedores. Sin embargo, cada una de estas soluciones se distingue por la naturaleza del servicio que ofrece a sus clientes, como se mostrará en las siguientes secciones.

Servidores de archivos

Con un servidor de archivos, el cliente (usualmente una PC) envía solicitudes de registros de archivos al servidor de archivos a través de una red (Figura 2-1). Se trata de una modalidad de servicio de datos muy primitiva, pues se requieren numerosos intercambios de mensajes a través de la red para obtener los datos solicitados. Los servidores de archivos son útiles para compartir archivos por medio de una red. Resultan indispensables para la creación de depósitos compartidos de documentos, imágenes, planos de ingeniería y otros objetos de datos de grandes dimensiones.

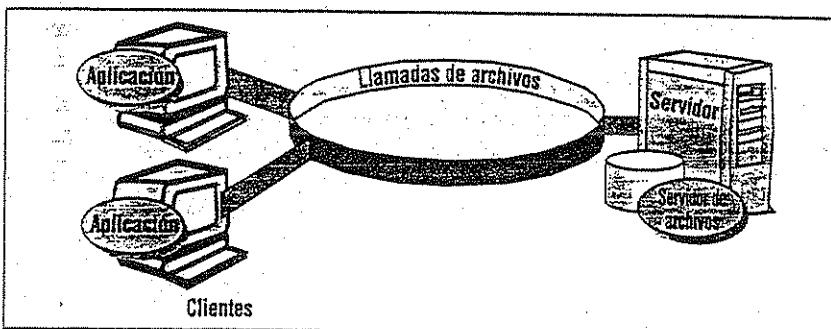


Figura 2-1. Cliente/servidor con servidores de archivos.

Servidores de bases de datos

Con un servidor de bases de datos, el cliente envía solicitudes de SQL en calidad de mensajes al servidor de bases de datos (Figura 2-2). Los resultados de cada orden de SQL son devueltos por medio de la red. El código que procesa la solicitud de SQL y los datos residen en la misma máquina. El servidor hace uso de su propia capacidad de procesamiento para encontrar los datos solicitados, en lugar de hacerle llegar todos los registros al cliente para que éste encuentre sus propios datos, tal como ocurre en el caso del servidor de archivos. El resultado es un uso mucho más eficiente de la capacidad de procesamiento distribuida. En este caso, el código del servidor es ofrecido en un paquete por el proveedor. Sin embargo, es común que deban generarse códigos para la aplicación cliente (o adquirir clientes en paquetes comerciales como Quest y Paradox). Los servidores de bases de datos constituyen el fundamento de los sistemas de apoyo de decisiones que precisan de consultas específicas y reportes flexibles.

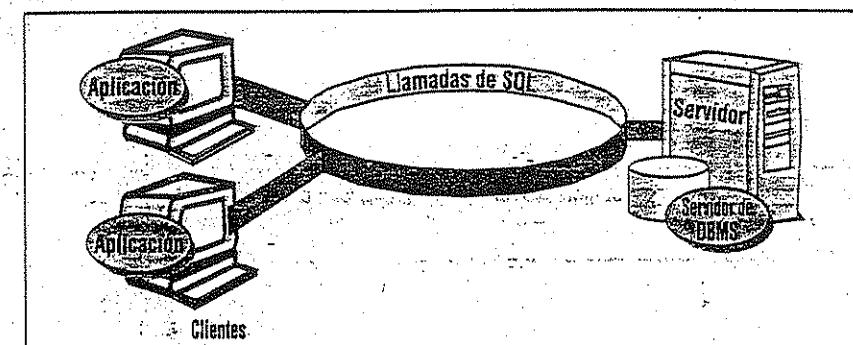


Figura 2-2. Cliente/servidor con servidores de bases de datos.

Servidores de transacciones

Con un servidor de transacciones, el cliente invoca *procedimientos remotos* que residen en el servidor con un mecanismo de bases de datos de SQL (Figura 2-3). Estos procedimientos remotos en el servidor ejecutan un grupo de instrucciones de SQL. El intercambio por la red consiste en un solo mensaje de solicitud/respuesta (a diferencia de lo que ocurre con el servidor de bases de datos, en cuyo caso es necesario un mensaje de solicitud/respuesta para cada instrucción de SQL en una transacción). Los enunciados de SQL acierten o fallan todos como una sola unidad. A estas instrucciones de SQL agrupadas se les conoce como *transacciones*.

Con un servidor de transacciones, usted crea la aplicación cliente/servidor generando el código para los componentes tanto de cliente como de servidor. El componente del cliente suele incluir una interfaz gráfica de usuario (GUI: *graphical user interface*). El componente del servidor consiste por lo general en transacciones de SQL contra una base de datos. A estas aplicaciones se les llama *procesamiento de transacciones en línea* (OLTP: *online transaction processing*). Tienden a ser aplicaciones de misión crítica que requieren de un tiempo de respuesta de 1-3

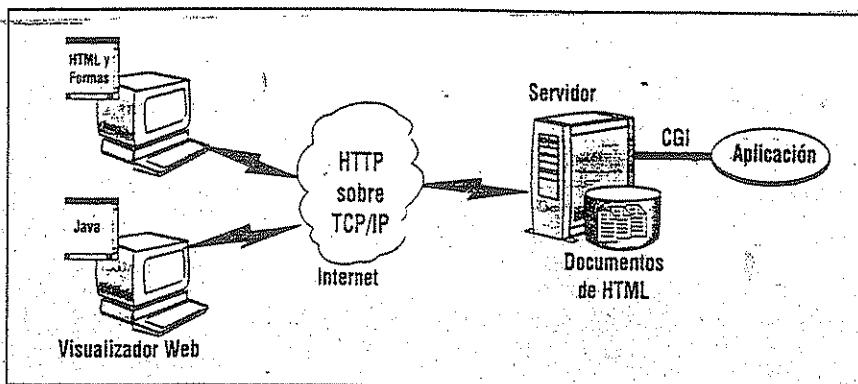


Figura 2-6. Cliente/servidor con servidores Web.

FYI

¿Qué es "middleware"?

Instrucciones

Middleware: 1) batidillo de tecnologías de software; 2) término rimbombante; 3) clave para el desarrollo de aplicaciones cliente/servidor.

Information Week

Middleware es un término vago que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores. Imagínelo como el software que ocupa la parte intermedia del sistema de cliente/servidor. En esta guía nos referiremos al middleware como el componente diagonal (/) de cliente/servidor. Es el enlace que permite que un cliente obtenga un servicio de un servidor. ¿Dónde empieza y dónde termina el middleware? Empieza en el módulo de API de la parte del cliente que se emplea para invocar un servicio y comprende la transmisión de la solicitud por la red y la respuesta resultante. Pero no incluye al software que presta el servicio real; esto pertenece a los dominios del servidor. Tampoco a la interfaz del usuario ni a la lógica de la aplicación, en los dominios del cliente.

Dividiremos al middleware en dos grandes clases:

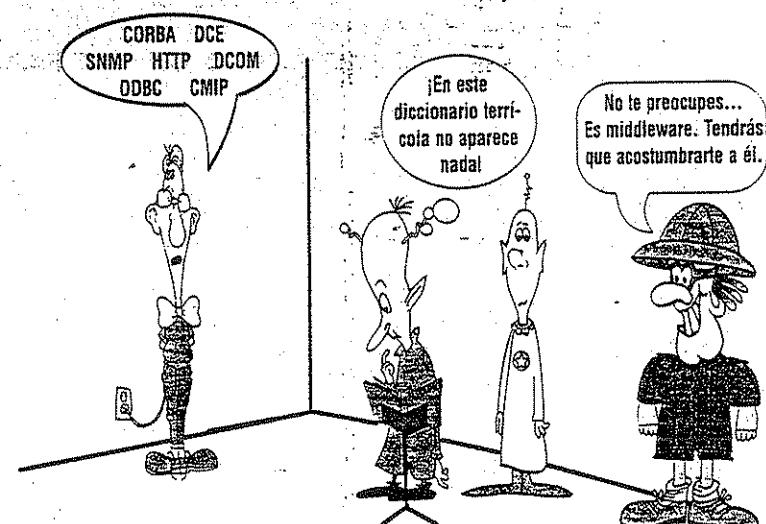
- El *middleware general* es el sustrato de la mayoría de las interacciones de cliente/servidor. Incluye las pilas de comunicación, directorios distribuidos, servicios de autenticación, horario de la red, llamadas a procedimiento remoto y servicios en cola. Esta categoría incluye también las extensiones del sistema operativo de redes, como los servicios distribuidos de archivos e impresión. Entre los productos que pertenecen a la categoría del middleware general están DCE, ONC+, NetWare, Named Pipes, LAN Server, LAN Manager, Vines, TCP/IP, APPC y NetBIOS. También debemos



incluir a los productos de middleware orientado a mensajes (conocido asimismo como MOM: *message-oriented middleware*) de Peerlogic, Covia, Message Express, System Strategies e IBM.

- El *middleware de servicios específicos* es necesario para cumplir un tipo particular de servicio de cliente/servidor. Pertenecen a esta categoría:

- ◆ El middleware para bases de datos, como ODBC, DRDA, EDA/SQL, SAG/CLI y Oracle Glue.
- ◆ El middleware para OLTP, como ATM y /WS de Tuxedo, Transactional RPC de Encina y TxRPC y XATMI de X/Open.
- ◆ El middleware para groupware, como MAPI, VIM, VIC, SMTP y las llamadas de Lotus Notes.
- ◆ El middleware para objetos, como CORBA de OMG y Network OLE (o DCOM) de Microsoft.
- ◆ El middleware para Internet, como HTTP, S-HTTP y SSL.
- ◆ El middleware para la administración de sistemas, como SNMP, CMIP y ORB.



A estas alturas usted pensará que el middleware es creación de personas a las que les encantan las siglas. Hasta donde sabemos, pocas tecnologías cuentan con tantas palabras y siglas extrañas como el middleware de cliente/servidor. Estudiaremos en detalle los estándares de middleware correspondientes a los diferentes tipos de aplicaciones cliente/servidor.

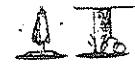
CLIENTE/SERVIDOR INTERGALÁCTICO

Si usted es como la mayoría de la gente —que apenas comienza a sentirse a gusto con su LAN departamental de Éthernet y su servidor de bases de datos local—, quizá no le agradaría oír esto: una segunda revolución cliente/servidor se cierne sobre la industria de las computadoras. Apriete sus cinturones de seguridad, porque esta segunda revolución amenaza con ser tan turbulenta como la que apenas acaba de concluir, en la que cliente/servidor aplicó una sierra gigantesca a las aplicaciones monolíticas basadas en macrocomputadoras y las dividió en componentes de cliente y servidor.

Las aplicaciones cliente/servidor se hallan en un nuevo umbral, producido por 1) el incremento exponencial del ancho de banda de bajo costo en las redes de área amplia, como Internet y CompuServe, y 2) una nueva generación de sistemas operativos para escritorio multihilos y aptos para enlazarse en red, como OS/2 Warp Connect y Windows 95. Este nuevo umbral marca el comienzo de una transición de cliente/servidor de *Ethernet* a cliente/servidor *intergaláctico*, que resultará en la irrelevancia de la proximidad. El centro de gravedad está desplazándose del cliente/servidor departamental basado en LAN en 2 planos de un solo servidor a una modalidad post-carestía de cliente/servidor en la que cada máquina de la "carretera de la información" global puede ser lo mismo cliente que servidor. En la Tabla 2-1 se comparan estos dos eras de la computación de cliente/servidor.

Tabla 2-1. Cliente/servidor Web contra cliente/servidor tradicional

Característica de aplicación	Era de cliente/servidor intergaláctico	Era de cliente/servidor de Ethernet
Número de clientes por aplicación	Millones	Menos de 100
Número de servidores por aplicación	Más de 100 000	1 o 2
Geografía	Global	Basada en campus
Interacciones servidor a servidor	Sí	No
Middleware	ORB sobre Internet	SQL y procedimientos almacenados
Arquitectura cliente/servidor	3 planos (o n planos)	2 planos
Actualizaciones para transacciones	Amplias	Poco frecuentes
Contenido de multimedia	Alto	Bajo
Agentes móviles	Sí	No
Interfaz del cliente	OOUI, documentos compuestos y ubicaciones embarcables	GUI



La visión intergaláctica

La gran interrogante para los próximos diez años es ésta: ¿qué pasaría si las comunicaciones digitales fueran libres? La respuesta es que nuestra manera de aprender, comprar, socializar, hacer negocios y divertirnos sería muy diferente.

**Bill Gates, director
Microsoft
(Enero de 1995)**

En lo que se refiere a las aplicaciones cliente/servidor intergaláctico, todo depende de la imaginación. La perspectiva de un amplio ancho de banda a muy bajo costo ha despertado visiones de una carretera de la información convertida en el centro comercial más grande del mundo. La visión predominante es la de un bazar electrónico de dimensiones planetarias repleto de boutiques, tiendas departamentales, librerías, servicios bursátiles, bancos y agencias de viajes. A la manera del Club Med, este centro comercial emitirá su propia moneda electrónica para facilitar las compras a cualquier hora y transacciones de empresa a empresa. Agentes electrónicos de todo tipo recorrerán la red en busca de oportunidades de negocios y realizarán negociaciones con otros agentes. Todos los días se generarán miles de millones de transacciones comerciales electrónicas. También se generarán, moverán y almacenarán en la red grandes cantidades de datos de multimedia.

Es obvio que lo que acabamos de describir no corresponde a la Internet que hoy conocemos; esa visión va mucho más allá del simple paseo por redes de hipertexto de información con etiquetas de HTML. Hablamos de un índice de transacciones miles de veces superior al actual. Además, estas transacciones serán de más larga vida y más complejas. También los datos sobre los que tales transacciones operarán serán más complejos y más ricos en contenido de multimedia. Así pues, nos referimos a la próxima generación de tecnología de Internet, a la que en este libro llamaremos *Object Web* (véase la Octava parte). Esta tecnología se empleará asimismo en Internets con i minúscula —o intranets— como las LAN, las redes interempresariales y las redes privadas de área amplia.

¿Qué necesitamos?

Para que todo esto ocurra, en el nivel de aplicación cliente/servidor se necesitan tecnologías clave como las siguientes:

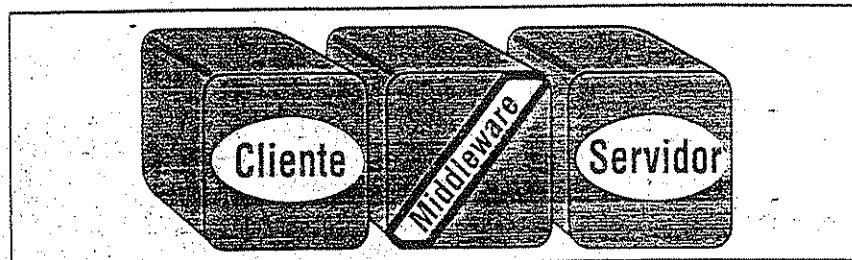
- **Un procesamiento de transacciones más rico.** Además de soportar las venerables transacciones planas, el nuevo entorno precisa de transacciones anidadas que puedan trasladarse a lo largo de servidores múltiples, transacciones de larga vida cuya ejecución se prolongue por amplios períodos al viajar de un servidor a otro y transacciones en cola que puedan emplearse en tratos seguros de empresa a empresa. La mayoría de los nodos de la red deberán estar en condiciones de participar en una transacción asegurada; los nodos de los superservidores manejarán las enormes cargas de transacciones.



CLIENTE/SERVIDOR: UN TAMAÑO PARA TODOS LOS MODELOS

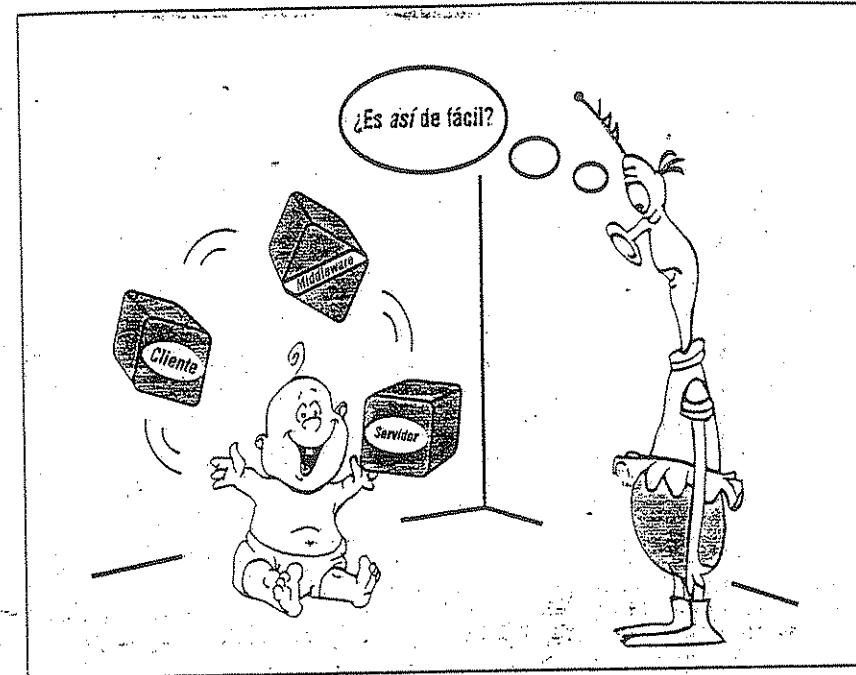
¿Un solo modelo de cliente/servidor puede adaptarse a todos estos tipos de usuarios? Creemos conocer ese modelo. Es engañosamente simple, y opera adecuadamente con las tecnologías de hoy. Es idealmente apto para resolver las necesidades de un mundo de *computación post-carestía*, en el que cliente/servidor se convertirá en el medio por excelencia para la compartición y la colaboración.

El modelo que presentaremos en esta sección es en realidad un juego, que consiste en unir cosas con elementos de construcción. Le demostraremos que es posible satisfacer un amplio espectro de necesidades de cliente/servidor —desde las más insignificantes hasta las intergalácticas— con sólo tres elementos básicos de construcción: un cliente, un servidor y la diagonal (/) que enlaza al cliente con el servidor (véase Figura 3-1). Este juego fascinará a niños de todas las edades. A usted le permitirá identificar algunas de las estructuras más duraderas en el diseño de sistemas de cliente/servidor.



En las siguientes secciones explicaremos (e ilustraremos) la disposición de estos elementos en cuatro situaciones:

- *Cliente/servidor para microempresas y tribus nómadas* es una implementación de elementos en la que el cliente, el software de middleware y la mayoría de los servicios de negocios corren en la misma máquina. Es la implementación que se sugiere para empresas individuales, oficinas domésticas y usuarios móviles con laptops bien dotadas. Se trata de un área de nuevas oportunidades para la tecnología de cliente/servidor.
- *Cliente/servidor para pequeñas empresas y departamentos* es la implementación clásica de elementos de cliente/servidor único de Ethernet. Se le emplea en pequeñas empresas, departamentos y sucursales. Es la modalidad de cliente/servidor que predomina en la actualidad.
- *Cliente/servidor para empresas intergalácticas* es la implementación de elementos de cliente/servidor de servidores múltiples. Los servidores le presentan al cliente una imagen de sistema único. Se les puede dispersar a todo lo largo de la empresa, a pesar de lo cual es posible lograr que parezcan formar parte del escritorio local. Esta implementación satisface las necesidades iniciales de la computación de cliente/servidor intergaláctico.



- *Cliente/servidor para un mundo post-carestía* transforma a todas las máquinas del mundo en cliente y servidor al mismo tiempo. Los agentes personales de cada máquina manejarán todas las negociaciones con otros agentes en cualquier punto del universo. Este sueño está a punto de hacerse realidad.

Encontrará muchas semejanzas entre estas cuatro disposiciones. Esto se debe a que en todas ellas se emplea el mismo tipo de infraestructura de software, middleware y comunicaciones.

Cliente/servidor para microempresas y tribus nómadas

Lo más maravilloso de cliente/servidor es que es sumamente maleable. Es fácil correr la porción de cliente y servidor de una aplicación en el mismo aparato. Los proveedores pueden empaquetar fácilmente versiones de una aplicación cliente/servidor para un solo usuario (véase Figura 3-2). Por ejemplo, una aplicación cliente/servidor para el consultorio de un dentista puede venderse en un paquete destinado a un solo usuario para consultorios de un solo dentista, y en un paquete destinado a múltiples usuarios para consultorios con muchos dentistas. La misma aplicación cliente/servidor cubre ambos casos. La única precaución por tomar sería la de emplear un sistema operativo suficientemente robusto para ejecutar tanto la parte del cliente como la del servidor de la aplicación.

El ejemplo del pequeño consultorio dental también procede en el caso de microempresas domésticas y de usuarios móviles. En todos estos casos, la aplicación cliente/servidor decisiva

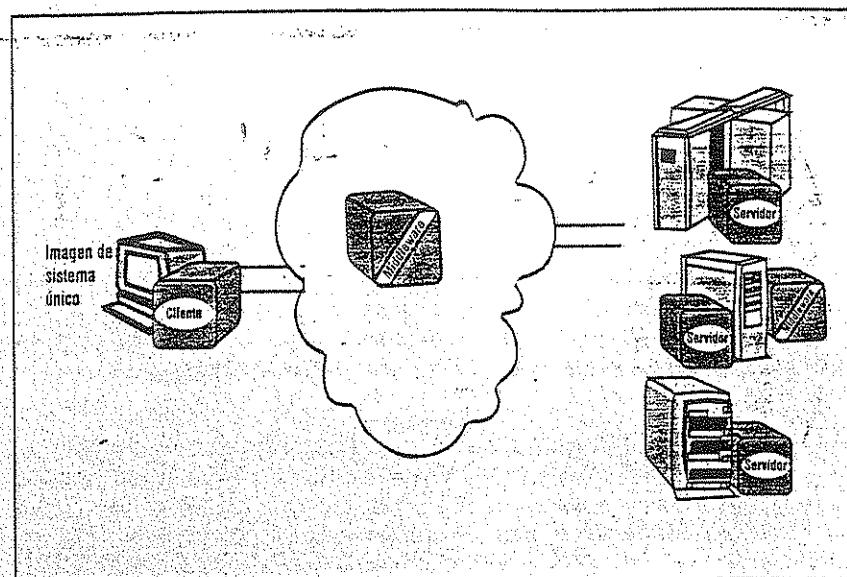


Figura 3-4. Cliente/servidor para empresas intergalácticas.

ción. Puede haber tantas combinaciones de servidores como lo permita el presupuesto. Correctamente empleada, la capacidad de servidores múltiples puede ofrecer una extraordinaria potencia y flexibilidad de cómputo, semejante en muchos casos a la de las macrocomputadoras.

Para explotar al máximo la capacidad de los multiservidores, necesitamos contar con un ancho de banda de alta velocidad y bajo costo y con una enorme cantidad de características de middleware, tales como servicios de directorios de redes, seguridad de red, llamadas a procedimientos remotos y servicios de horario de la red. El middleware produce una apariencia común para todos los servicios de la red, conocida como "imagen de sistema único".

Una buena arquitectura de software para implementaciones de cliente/servidor para empresas intergalácticas se reduce a la creación de "conjuntos" de sistemas a partir de los elementos de construcción modulares. Con un poco de práctica, usted podrá desarrollar habilidades creativas similares a las de los compositores de sinfonías para la articulación de componentes de software que corran en servidores múltiples. Deberá encontrar fórmulas creativas para la distribución del trabajo entre los servidores. Por ejemplo, puede repartir el trabajo mediante el empleo de objetos distribuidos. También tendrá que diseñar sus servidores en tal forma que puedan delegar funciones a otros servidores. Una solicitud compleja puede suponer el funcionamiento conjunto de un grupo de trabajo de servidores para su resolución. Preferentemente, el cliente no debería estar al tanto de esta colaboración tras bastidores. El servidor con el que el cliente haga contacto en primer término debe ocuparse de la orquestación del grupo de trabajo y de la entrega al cliente de los resultados.

El cliente/servidor intergaláctico es la fuerza impulsora tras de estándares de middleware como objetos distribuidos e Internet. Todos estamos en busca de la pieza mágica que hará que el



mundo del multiproveedor distribuido esté tan bien integrado como las macrocomputadoras de proveedor único. Las herramientas para la creación, despliegue y administración de aplicaciones escalables de cliente/servidor están recibiendo cada vez mayor atención. Mucho está por ganarse todavía en el cliente/servidor intergaláctico, puesto que aún nadie ha sido capaz de unir todas las piezas.

Cliente/servidor para un mundo post-carestía

Vamos a jugárnosla en esta sección. En ella investigaremos qué nuevos sistemas podrán crearse sobre una plataforma de cliente/servidor cuando la memoria y el hardware se vuelvan *increíblemente accesibles*. Cada máquina será al mismo tiempo cliente y servidor de plenas funciones (veáse Figura 3-5). Llamamos a este entorno de abundancia *el mundo post-carestía*. Suponemos que el aparato común post-carestía será semejante a una computadora portátil notebook celular de 2000 dólares alimentada por un Pentium de 200 MHz y cargada con 100 MBytes de RAM y 100 GBytes más de espacio en disco. Por supuesto que en esta máquina correrá todo el middleware con el que los proveedores puedan soñar en los próximos años.

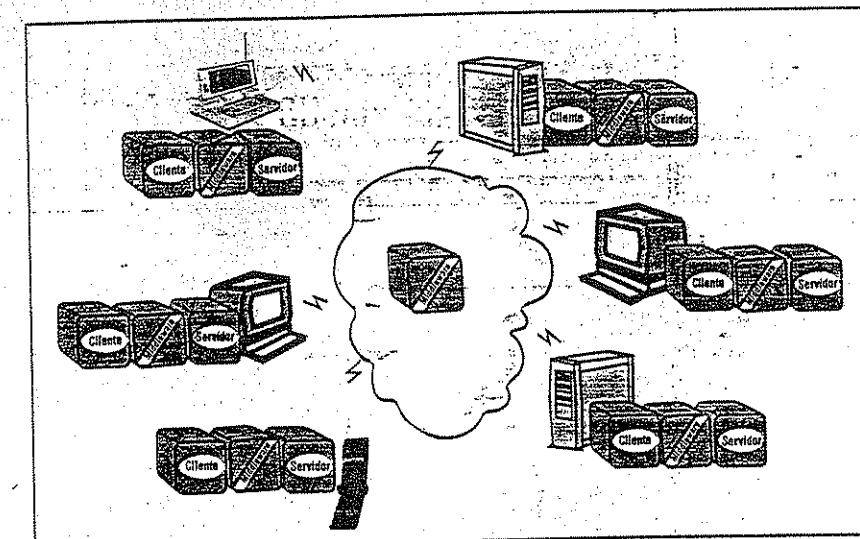


Figura 3-5. Cliente/servidor para un mundo post-carestía.

¿Qué haremos con toda esta capacidad aparte de correr el middleware? ¿Qué sucederá cuando todas las máquinas del mundo se vuelvan servidores y clientes universales? Dado que cada máquina será un servidor de funciones plenas, es de suponer que en ella correrán al menos un servidor de archivos, un servidor de bases de datos, un agente de flujo de trabajo, un monitor de TP y un servidor Web, conectados todos ellos a través de un ORB. Todo esto se añadiría a la totalidad del software y middleware del cliente.

Lo que queremos decir es que es probable que en los próximos años un centenar o más de millones de máquinas ejecuten *casi todas* las modalidades de software de cliente/servidor des-



Middleware servidor a servidor

El middleware no incluye el software que presta el servicio real. Sí incluye, en cambio, el software necesario para coordinar las interacciones interservidores (véase Figura 3-7). La naturaleza de las interacciones servidor a servidor suele ser de cliente/servidor: los servidores son clientes de otros servidores. No obstante, para algunas interacciones servidor a servidor se requiere de middleware de servidor especializado. Por ejemplo, podría precisarse de un protocolo de grabación en dos fases para coordinar una transacción que deba ejecutarse en servidores múltiples. Los servidores de una red de correo emplearán middleware especial de servidor a servidor para el manejo de mensajes del tipo almacenar y enviar. Sin embargo, el software más moderno (incluso en núcleos de sistemas operativos) sigue el paradigma de cliente/servidor.

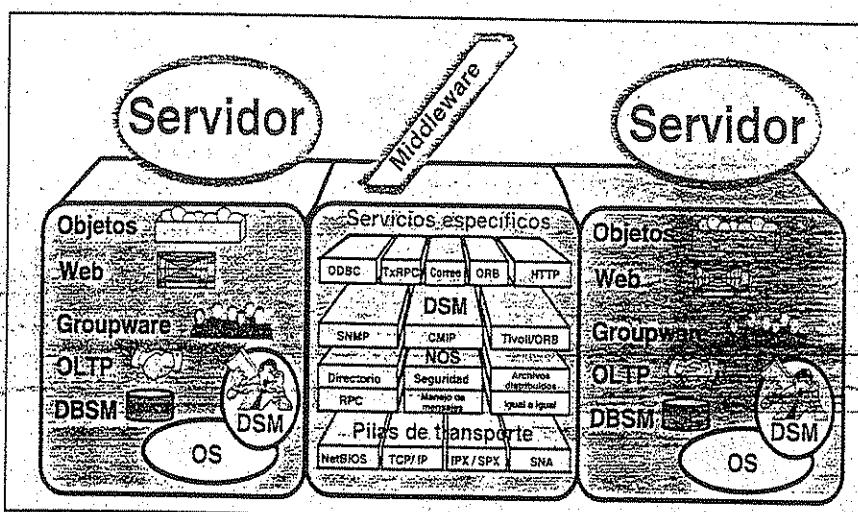


Figura 3-7. Infraestructura del middleware servidor a servidor.

El mapa de cliente/servidor

Los elementos que acabamos de describir nos servirán de mapa en esta guía. Impedirán que usted se pierda en el laberinto de cliente/servidor. En el siguiente capítulo analizaremos el estado de la infraestructura de redes: ¿para cuándo podemos esperar el paraíso del ancho de banda? La *Parte 2* trata de lo que clientes y servidores necesitan de un OS. La *Parte 3* es sobre el NOS intergaláctico. Iniciaremos entonces nuestro ascenso por el espacio de las aplicaciones ocupándonos de las cinco tecnologías competitivas en la lid del cliente/servidor intergaláctico: la *Parte 4* trata de las bases de datos de SQL y las bodegas de datos; la *Parte 5* se dedica a los monitores de TP; la *Parte 6* se ocupa del groupware; la *Parte 7* trata de objetos distribuidos, componentes y documentos compuestos, y la *Parte 8* de Internet y el Web. Ya casi terminamos. La *Parte 9* es sobre cómo administrar estos sistemas; en ella se aborda el tema de la administración de sistemas distribuidos. Finalmente, en la *Parte 10* se propone un modelo para la comprensión de las herramientas de aplicación cliente/servidor y se especula sobre la dirección que sigue todo este movimiento.

Capítulo 4

El camino al paraíso del ancho de banda



Si algo inerte se pone en movimiento, poco a poco cobrará vida.

Lao Tsé

Esperamos haberlo convencido ya, de que muy pronto nuestro planeta se verá invadido por redes ubicuas de cliente/servidor. Mediante el empleo de estas redes podremos comunicarnos más eficazmente con otros seres humanos: clientes, proveedores, jefes, compañeros de trabajo, familiares y amigos. Podremos comunicarnos también con los aparatos que usamos todos los días: automóviles, tanques de gas, televisores e incluso hogares inteligentes.

En este capítulo se ofrece una descripción general de la enorme infraestructura de redes en la que se apoyan estas redes de cliente/servidor. Aunque en realidad este libro trata del software de cliente/servidor, necesitamos desviarnos un poco de nuestro camino para conocer la infraestructura física de las redes en la que se basa ese software. Esta infraestructura es ya muy amplia en la actualidad, y puede interconectar a todas las redes del planeta. Pero para el efectivo despegue del cliente/servidor intergaláctico es necesario que esas redes sean todavía más ubicuas. Deben llegar hasta el centro mismo de nuestros hogares. Tan importante como ello es que sean capaces de ofrecer un abundante ancho de banda de bajo costo. Así pues, ¿en qué estado se

verá estos problemas. Esta industria de rápido crecimiento —con ventas por 3000 millones de dólares en 1995— no cesa de lanzar nuevas características y modelos más robustos. Por ejemplo, los enrutadores más recientes utilizan protocolos especializados para determinar las mejores vías para el tráfico de la red e incluso pueden realizar el equilibrio de cargas entre vías paralelas múltiples.

Pero a pesar de su popularidad, puentes y enrutadores no son la panacea. La segmentación de las LAN con puentes y enrutadores equivale a un remedio casero, no a una solución a largo plazo. Cuando el creciente uso de multimedia se traduzca en un drástico incremento de las cargas de tráfico, los enrutadores se convertirán en remedios *increíblemente costosos*. Su administración se complicará enormemente, aparte de lo cual provocarán demoras inaceptables para flujos de datos de tiempo real, como películas. Creemos que, a la larga, los enrutadores serán remplazados por redes basadas en ATM, de las que hablaremos más adelante en este mismo capítulo. Claro que si usted forma parte del ramo de los enrutadores, podrá considerar a estos conmutadores de ATM de alta velocidad como los nuevos enrutadores de fines de la década de los noventa.

El middleware de pilas de transporte

El fenómeno de los puentes/enrutadores no es el único en haber experimentado grandes progresos. Los sistemas operativos modernos —como Windows 95, OS/2 Warp Connect, NetWare, Windows NT y Unix SVR4— son cada vez más aptos para enfázarse en red. Disponen ahora de características que permiten la fácil conexión con ellos de pilas de comunicaciones de proveedores múltiples y adaptadores para redes. Asimismo, les han facilitado la vida a los programadores, pues ya cuentan con diversas API independientes de las pilas.

A continuación se ofrece un resumen de algunas de las nuevas características de los sistemas operativos que han hecho posible esto:

- **Elsándwich de pilas** proporciona los ganchos para la rápida conexión de pilas de proveedores múltiples con un sistema operativo. Para dar cabida a redes de proveedores múltiples, los sistemas operativos modernos deben soportar protocolos múltiples, redirectores y API. Para conseguirlo eficazmente, deben contar con interfaces perfectamente definidas entre componentes. Por lo general, un sistema operativo moderno dispone las pilas de transporte en una sección intermedia, entre una interfaz independiente del transporte sobre las pilas y una interfaz lógica para los controladores de dispositivos de la red bajo las pilas (véase Figura 4-2).
- **El controlador de red lógico** ofrece una interfaz única para todos los adaptadores de la red. Esta interfaz entre el adaptador de la red y las pilas de transporte es particularmente importante. Lo que menos desean los proveedores de pilas de transporte es tener que generar un controlador para cada posible adaptador de la red. Por supuesto que también los proveedores de adaptadores para redes desean evitarse la necesidad de crear interfaces para cada pila posible. NDIS de Microsoft/3Com y ODI de Novell son los dos estándares de facto más ampliamente aceptados para la interfaz de pilas de protocolos y controladores de dispositivos de adaptadores para redes. Lo són porque aportan un tablero de red lógico que facilita

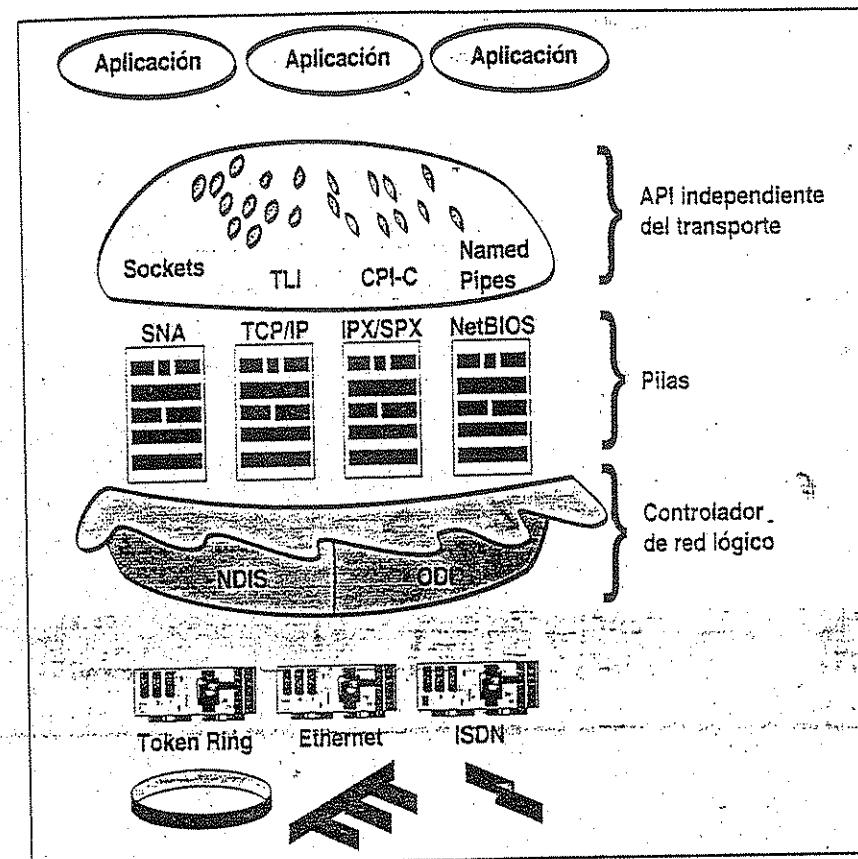


Figura 4-2. El sándwich de pilas.

la interfaz entre diferentes adaptadores de red y pilas de protocolos múltiples (véase Figura 4-2). Los proveedores de pilas de transporte pueden emplear NDIS u ODI como interfaz común para todos los adaptadores de la red. A su vez, los proveedores de adaptadores para redes pueden emplear NDIS u ODI como capa superior para sus controladores de redes. NDIS y ODI se encargan de enviar y recibir datos y administrar la tarjeta de adaptadores.

- Las **API independientes del transporte** se asientan sobre las pilas de transporte y permiten que los desarrolladores conecten sus programas a una sola interfaz que soporta protocolos múltiples. La interfaz *Sockets* es cada vez más la opción preferida en la mayoría de las plataformas de sistemas operativos para la interfaz con pilas de protocolos y proveedores múltiples. Otras opciones son la *interfaz para la capa de transporte* (*TLI: transport layer interface*) que se usa en NetWare y muchas implementaciones de Unix; *CPI-C*, la moderna API igual a igual de SNA, capaz ahora de operar con pilas tanto de SNA como de TCP/IP, y *Named Pipes*, que corre sobre pilas de NetBIOS, IPX/SPX y TCP/IP.



Información

¿Qué es un circuito virtual?

Los circuitos virtuales son semejantes al sistema telefónico. Se les establece cuando dos nodos deben comunicarse, y se les elimina cuando ya no son necesarios. Dado que voz y video circulan en la red en flujos dentro de circuitos virtuales, las demoras son reducidas y constantes. Por ejemplo, la tecnología de conmutación de paquetes Frame Relay basada en WAN se sirve de circuitos virtuales para asignar ancho de banda a solicitud expresa y optimizar el uso del ancho de banda existente.

En contraste, la actual tecnología de LAN asigna ancho de banda por *competencia*. Cada quien rivaliza con todos los demás por conseguir el uso del medio de difusión. Para acceder a la LAN, usted debe esperar una señal (Token Ring) o iniciar la difusión; debe estar preparado para retroceder si detecta una colisión (Ethernet). Los métodos de competencia no pueden garantizar tiempos de respuesta (o demoras) deterministas. Es cuestión de suerte, y las cosas se complican en las horas pico. La situación puede mejorar si se asignan prioridades, pero incluso dentro de los mismos niveles de prioridad se da la competencia.

Algunas de las nuevas tecnologías de LAN isócrona proponen un entorno *híbrido* que asigna cierta cantidad de ancho de banda al tráfico en controversia y cede el resto a circuitos virtuales. Por ejemplo, la FDDI isócrona destina cierto porcentaje del ancho de banda de la red a circuitos virtuales múltiples de 64 Kbit/s y ofrece el resto a datos en competencia normal.

Otro método consiste en emplear commutadores de LAN para eliminar la controversia. Por ejemplo, *Switched Ethernet* le proporciona a cada estación un conducto de uso determinado de 10 Mbit/s cediendo su propio segmento de LAN; cada segmento de cable está conectado directamente a un commutador centralizado de Ethernet, el cual actúa como eje. Este eje cuenta con un bus interno de alta velocidad para conmutar paquetes entre múltiples segmentos de cable al tiempo que ofrece 10 Mbit/s a cada estación. Muchos commutadores de LAN están diseñados para interconectarse vía enlaces ascendentes de alta velocidad como FDDI (100 Mbit/s), Fast Ethernet (100 Mbit/s) y, en situaciones limitadas, ATM (hasta 2.4 Gbit/s). La conmutación de LAN es hoy un gran negocio; IDC calculó que en 1996 generaría ingresos superiores a los 2000 millones de dólares.

El modo de transferencia asíncrona (ATM: asynchronous transfer mode) es la tecnología isócrona más reciente. Asigna ancho de banda de LAN/WAN a solicitud expresa a través de circuitos virtuales. Usa tecnología de conmutación de circuitos de alta velocidad basada en hardware potencialmente capaz de liberar una cantidad impresionante de ancho de banda isócrono a muy bajo costo (a cada nodo se le asigna su propio segmento de LAN de uso determinado en el commutador).



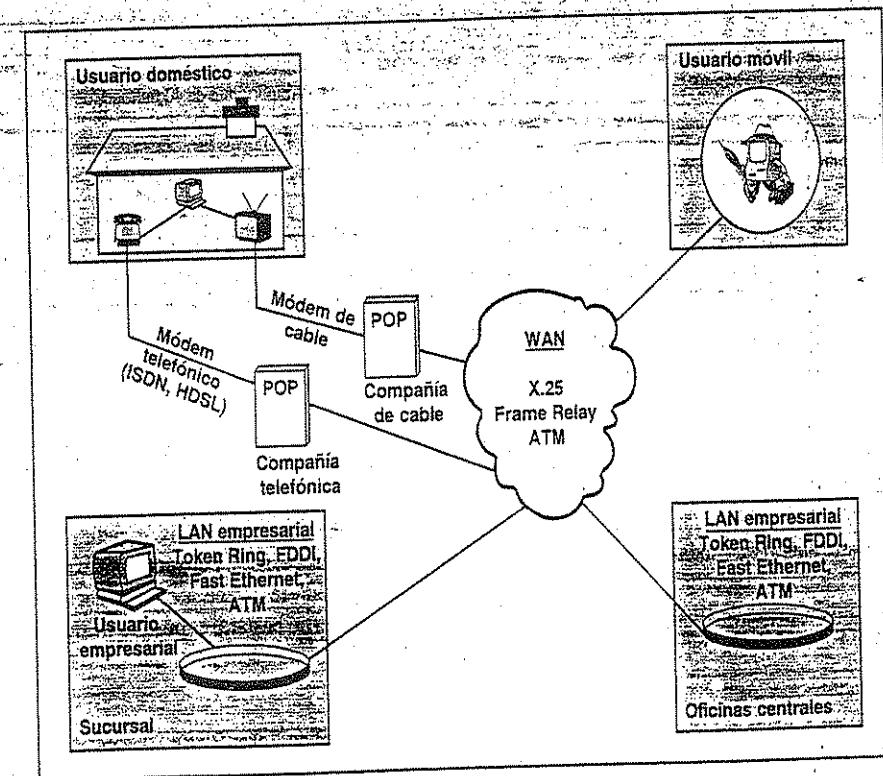
¿Cuánto ancho de banda podemos esperar en realidad?

Eres tan rápido como el más débil de tus eslabones.

Gopi Bala, analista en jefe
Yankee Group
(Mayo de 1996)

La cantidad de ancho de banda de la que disponga dependerá por lo general del lado del muro de protección (firewall) donde se encuentre. Muchos usuarios empresariales están ya en el paraíso del ancho de banda, cuando menos en lo que se refiere a sus LAN. Sin embargo, si usted se halla fuera de las murallas empresariales, es probable que esté en el "infierno del ancho de banda"; tal es el caso de los usuarios domésticos, móviles y empresariales fuera del cenáculo sagrado. Además, incluso el tráfico de las LAN empresariales debe atravesar en ocasiones una WAN lenta —la nube más negra del cielo— para conectarse con otras LAN en lugares distantes (véase Figura 4-4).

Suponiendo la ausencia de competencia, el ancho de banda de que dispone en su trabajo está en función fundamentalmente de la capacidad de sus LAN y WAN y de los enlaces que las unen. El ancho de banda de que dispone en casa está determinado principalmente por el enlace que lo





múltiplos de 51.84 Mbit/s. Lo máximo en la línea es OC48, que especifica una velocidad de línea de 2.4 Gbit/s; ANSI trabaja ahora en el Sonet de 10 Gbit/s. Sonet también especifica una topología de cableado en forma de anillo para el reenrutamiento instantáneo de tráfico en caso de interrupción en el servicio.

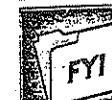
La buena noticia es que todas las grandes portadoras del mundo están desplegando Sonet en sus redes. Además, muchas de ellas ya han comenzado a ofrecer servicios de Sonet a usuarios empresariales y proveedores de servicios de Internet, lo que sin embargo no es barato. Para el año 2000, la mayoría de las WAN públicas se hallarán muy cerca del 100% de Sonet. En Estados Unidos, casi todas las portadoras de larga distancia han instalado fibra óptica para Sonet de costa a costa, y muchas portadoras locales instalan actualmente circuitos de fibra óptica para Sonet en las grandes ciudades. Sprint es quien ha llegado más lejos hasta ahora, con 32 circuitos de Sonet interconectados a nivel nacional. Más del 50% de la red de Sprint ya responde hoy al patrón de Sonet, y llegarán al 100% en 1998. Ahora Sprint introduce troncales de OC12 de Sonet en circuitos de T1 y T3, de menor capacidad, que vende a usuarios individuales; la idea es ofrecer servicios de fibra óptica de bajo costo hasta los niveles inferiores. Pero mientras Sonet invade el planeta, es preciso hacerse la siguiente pregunta: ¿qué correrá en ella? La respuesta debe ser, desde luego, ATM, el cual es capaz de soportar por sí solo velocidades de comunicación de hasta 2.488 Gbit/s.

Las WAN modernas emplean tecnología de conmutación de paquetes para la capa de vinculación de redes físicas de T1, T3 o Sonet. Los conmutadores de paquetes dividen los flujos de datos en paquetes, que luego lanzan a la red. Las cabeceras de dirección sirven para conducir a cada paquete a su destino. Protocolos como TCP/IP o IPX/SPX corren en estas redes de conmutación de paquetes.

En la Tabla 4-4 se comparan las tres principales tecnologías de conmutación por paquetes: *relé de trama* (*frame relay*), *servicios de datos conmutados multimegabit* (*SMDS*; *switched multimegabit data services*) y *modo de transferencia asíncrona* (*ATM*; *asynchronous transfer mode*). Es evidente que ATM es la que ofrece el mejor ancho de banda. Sin embargo, el relé de trama o *frame relay* es una tecnología madura que opera en la infraestructura existente de WAN de T1. SMDS se usa sobre todo en Europa, como precursora de ATM. Tanto el relé de trama como SMDS pueden correr en conmutadores de paquetes de ATM (véase el siguiente recuadro de información).

Tabla 4-4. Red WAN: Alternativas de conmutación de paquetes.

Tecnología de WAN	Velocidad máxima	Aplicaciones	Tamaño del paquete	Despliegue
Frame relay o relé de trama	1-2 Mbit/s (T1/E1)	Datos	Dimensión variable, 4096 bytes máx.	Amplio
SMDS	45 Mbit/s (T3)	Datos	Dimensión variable, 9188 bytes máx. (puede dividirse en células de 53 bytes)	Limitado
ATM	2.4 Gbit/s	Datos, voz y video	Células de 53 bytes	Limitado pero creciente



Información

B-ISDN: relé de trama, SMDS y ATM

La red pública, con una antigüedad de 20 años, se ha visto rebasada por el súbito y drástico incremento de tráfico. Y la situación es cada vez peor.

Joe McGarvey
Inter@ctive Week
(Abril de 1996)

La fibra óptica nos ha hecho pasar, sin transición, de un modesto a un casi infinito ancho de banda.

Nicholas Negroponte, autor
de Being Digital
(Knopf, 1995)

¿Qué es relé de trama?

Relé de trama, también llamado *Frame Relay*, es la tecnología de conmutación de paquetes para WAN más común en la actualidad; la soportan más de 90 portadoras estadounidenses. Hizo su aparición en 1992, como una implementación modernizada de la antigua tecnología de conmutación de paquetes X.25. El relé de trama puede enrutar paquetes de dimensión variable a velocidades de T1 (o E1) en los enrutadores y conmutadores actuales y otro equipo basado en HDLC; para ello basta con una actualización del software. En contraste, X.25 sólo puede dar salida a un máximo de 64 Kbit/s en la misma infraestructura física. El relé de trama logra esa hazaña eludiendo la verificación de errores en cada segmento de la red (los puntos intermedios). En cambio, depende de los puntos terminales para la verificación de errores de extremo a extremo. Su demanda es muy fuerte. Según Vertical Systems Group, se espera que el mercado de relé de trama crezca de ventas por 1600 millones de dólares en 1995 a 5400 millones en 1998. Cabe señalar que el relé de trama es objeto de constantes mejoras; actualmente se hallan en operación las implementaciones de velocidad de T3.

¿Qué es SMDS?

SMDS fue diseñada originalmente para llenar el vacío en los servicios de WAN de alta velocidad hasta que se consiguiera la amplia disponibilidad de ATM. Soporta paquetes de dimensión variable que pueden dividirse en células fijas de tamaño de ATM para facilitar la migración. Hoy puede correr a velocidades de T3 (45 Mbit/s); alcanza este alto rendimiento por no ofrecer soporte a circuitos virtuales: cada paquete se las ve por sí solo. En contraste, tanto ATM como el relé de trama soportan circuitos virtuales. SMDS no ha tenido clamoroso éxito de mercado. De acuerdo con Vertical Systems Group, su mercado fue inferior a los 45 millones de dólares en 1995. En Estados Unidos, sólo MCI ofrece



Estado de la interconexión LAN a WAN

Hoy en día las LAN se conectan a WAN por medio de enrutadores. El enrutador efectúa el enlace con una WAN pública o privada a través de líneas dedicadas especializadas (por lo general T1 o T3). La tecnología de conmutación suele ser relé de trama, X.25 o sencillamente punto a punto (*PTP: point-to-point*). Algunas empresas grandes emplean ya en sus conexiones LAN a WAN ATM sobre Sonet. Otras han recurrido a proveedores de servicios de intranet para remplazar redes privadas por WAN públicas.

A diferencia de los individuos, quizás las empresas requieran de un "conducto amplio" garantizado por parte de sus proveedores de WAN mediante esquemas de reserva de ancho de banda (o *calidad de servicio*).² Además, las empresas pueden permitirse mantener en sus instalaciones sofisticados enrutadores de alta velocidad. Esto significa que la LAN se convierte en un segmento más de la WAN. En consecuencia, la tecnología que describimos en la sección anterior se aplica también a la conexión LAN a WAN. De este modo, la actual conexión LAN a WAN es el relé de trama; la de mañana será ATM. Y si ATM corre en la LAN, incluso dejarán de necesitarse enrutadores independientes. La LAN sencillamente se convertirá en WAN. Ya no habrá desajustes de impedancia.

Estado de la conexión hogar a WAN

¿Cómo llevar la carretera digital hasta el hogar? La mayoría de los usuarios de Internet Web están dolorosamente conscientes de que los hogares se encuentran del lado equivocado en lo que respecta a la pista del ancho de banda. La extensión de la WAN a los hogares implica una solución al desajuste de impedancia (o cuello de botella) provocado por la "última milla" del cableado telefónico que enlaza a hogares y oficinas con los proveedores de servicios de larga distancia. En el caso específico de Estados Unidos, las compañías telefónicas cuentan aún con más de 100 millones de pares de circuitos locales de alambre de cobre. Con todo, las portadoras de larga distancia remplazaron en la última década sus redes principales de alambre de cobre por una red de cables de fibra óptica superveloces, de alto ancho de banda y bajo costo (es decir, Sonet). No obstante, el traslado de estas ventajas a los hogares supone la sustitución de esa última milla de cables locales por fibra óptica, de costo prohibitivo.

Los módems actuales de 28.8 Kbit/s están llevando al límite al sistema telefónico analógico. Pero, aun siendo "tremendamente rápidos", les será imposible ofrecer su publicitado rendimiento si no disponen de las condiciones ideales. Sin embargo, dos tecnologías en pugna de las compañías telefónicas locales, capaces de acelerar los circuitos locales sin necesidad de remplazo del cableado existente, ofrecen perspectivas promisorias: la *red digital de servicios integrados* (ISDN: *integrated services digital network*) y la *línea de suscripción digital de cargo de bits*

(HDSL: *high-bit-rate digital subscriber line*). En caso de que estas tecnologías no funcionaran, se contaría siempre con la opción del *módem para cable* de las compañías de televisión por cable (véase el siguiente recuadro de información).

Pero llegará el momento en que los hogares dispongan de cable de fibra óptica. La fibra óptica es cientos de veces más veloz que incluso el cable coaxial, y más versátil. Las compañías estadounidenses de telefonía local ya están haciendo llegar la fibra óptica más allá de sus oficinas de conmutación, hasta sus clientes residenciales. A la larga, la extenderán indudablemente a todos los hogares. Pacific Bell —que cubre el estado de California en su totalidad— lleva la delantera por lo pronto. Adicionalmente, algunos operadores de cable (como Time Warner) instalan ya fibra óptica en sustitución del cable coaxial. En la Tabla 4-5 se comparan estas diversas opciones (véase también el siguiente recuadro de información).

Tabla 4-5. Conexión hogar a WAN: Comparación de opciones.

Tecnología de conexión	Velocidad	Disposición	Comentarios (cantidades en dólares)
V.32bis	14.4 Kbit/s	Actual	<ul style="list-style-type: none"> ■ Módem de marcación analógico ■ Módem por menos de \$100
V.34	28.8 Kbit/s	Actual	<ul style="list-style-type: none"> ■ Módem de marcación analógico ■ Módem por menos de \$250
BRI de ISDN	128 Kbit/s	Actual	<ul style="list-style-type: none"> ■ Módem de marcación digital ■ \$300-\$1000 por tarjeta ISDN ■ \$100 instalación ■ \$25 al mes + uso
PRI de ISDN	1.544 Mbit/s	Actual	<ul style="list-style-type: none"> ■ Requiere T1 ■ \$500-\$750 al mes
T1	1.544 Mbit/s	Actual	<ul style="list-style-type: none"> ■ Línea dedicada ■ Más de \$700 al mes
HDSL	1.5 Mbit/s	Finales de 1996	<ul style="list-style-type: none"> ■ Emplea cable telefónico existente ■ 2 pares trenzados ■ Módem por menos de \$300 (estimación)
ADSL 3	6 Mbit/s 640 Kbit/s (salida)	1997	<ul style="list-style-type: none"> ■ Ancho de banda asimétrico ■ Un solo par trenzado ■ Módem por menos de \$300 (estimación) ■ \$35-\$100 al mes
Módem para cable	10 Mbit/s (unidireccional)	1997	<ul style="list-style-type: none"> ■ Módem por menos de \$400 ■ Salida telefónica independiente ■ Tarifa de cable ■ Soporte de la compañía de cable
B-ISDN (ATM/Sonet)	100 Mbit/s (y más)	1999	<ul style="list-style-type: none"> ■ Nirvana de ancho de banda ■ Voz, multimedia y datos ■ Requiere fibra óptica

² Los proveedores de servicios de Internet (ISP: *Internet service providers*) también participan en el mercado de ancho de banda garantizado. Por ejemplo, a partir de 1997 BBN Planet prevé ofrecerles a sus clientes empresariales acceso a Internet con "calidad de servicio". Usará para ello un nuevo protocolo de Internet, llamado *protocolo de reserva de recursos* (RSVP: *resource reservation protocol*). Cisco anunció en mayo de 1996 que incorporaría el RSVP en sus enruteadores.



Estado de la conexión aplicaciones inalámbricas a WAN

La tecnología de comunicaciones inalámbricas totales siempre parece tener que esperar un par de años más.

Angela Hickman, PC-Magazine
(Mayo de 1996)

Es de suponer que las redes inalámbricas habrán de convertirse en la conexión por excelencia con la carretera digital, pues permitirán la comunicación "a cualquier parte en cualquier momento". Dado que la fuerza de trabajo móvil aumenta casi 15% al año, las aplicaciones inalámbricas se están volviendo una necesidad. Pero mientras que los teléfonos celulares ya están en todas partes (hay más de 58 millones en uso), las conexiones para datos inalámbricos ubicuos siguen siendo cosa del futuro. En la Tabla 4-6 se comparan las principales tecnologías inalámbricas de datos. Ninguna de ellas es ideal para altos volúmenes de tráfico. Así, hágase a la idea de conservar sus conexiones alámbricas un año más, hasta que los nuevos *servicios de comunicación personal* (PCS: *personal communication service*) nos proporcionen el ancho de banda que necesitamos en un cielo propicio.

Tabla 4-6. Opciones inalámbricas.

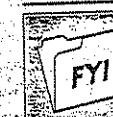
Servicio inalámbrico	Ancho de banda	Tiempo de preparación de llamada	Precio (en dólares)
Comunicación celular	14.4 Kbit/s (o menos)	30 seg	Por minuto (35 centavos por minuto)
Paquete celular (CDPD)	19.2 Kbit/s	5 seg	Por paquete (15 centavos por KByte)
Radio privada en paquete	19.2 Kbit/s	5 seg (o más)	Por paquete (15 centavos por KByte)

Pero, ¿cuál es el estado actual de la aplicación de conexión inalámbrica-WAN? He aquí una breve descripción al día de las opciones disponibles:

- El *celular de commutación de circuitos* consiste sencillamente en un módem para conectar una laptop a un teléfono celular. De este modo es posible conectarse con cualquier teléfono (con un módem) en la red celular nacional. Además, al mismo tiempo puede usarse el teléfono celular para la transmisión de voz. Por lo tanto, todo lo que se necesita para la comunicación inalámbrica es un teléfono celular y un módem celular (alrededor de 400 dólares por los dos). Claro que aparte se debe pagar la tarifa celular por minuto (35 centavos o más), que es donde acaba la diversión.
- Los *datos digitales celulares en paquete* (CDPD: *cellular digital packet data*) permiten la transmisión de paquetes de datos por las porciones sin usar de la red celular existente. CDPD puede identificar dinámicamente los canales de voz abiertos y emplearlos para el tráfico de datos. Para acceder a la red se necesita un módem de CDPD, con un costo de 400 dólares, y realizar pagos por paquete: *Cellular One* cobra alrededor de 15 centavos por KByte. Los módems de CDPD no requieren de un teléfono celular en operación; incluyen sus propios transceptores de radio.

■ Los proveedores de radio privada en paquete ofrecen una opción inalámbrica nacional a la red celular. Los dos principales proveedores de radio privada en paquete de Estados Unidos son RAM Mobile Data y Ardis. RAM dice contar con la capacidad para prestar servicios inalámbricos bidireccionales a más del 94% de la población urbana estadounidense (cubre 260 zonas metropolitanas). Ardis —la red inalámbrica digital de IBM/Motorola— se halla disponible en más de 400 ciudades de Estados Unidos. Por su parte, AT&T invirtió 12 600 millones de dólares en la adquisición de McCaw Cellular Communications para complementar su surtido de comunicaciones inalámbricas de extremo a extremo. AT&T/McCaw se propone ofrecer una sola red inalámbrica basada en PCS que cubra el 80% del territorio de Estados Unidos.

A parte de estos servicios nacionales, existen opciones inalámbricas locales. En el área de la bahía de San Francisco, por ejemplo, Ricochet (<http://www.ricochet.net>) ofrece acceso inalámbrico ilimitado a Internet por una cuota fija de 29.95 dólares al mes. La renta de un módem inalámbrico de 28.8 Kbit/s tiene un costo adicional de 10 dólares al mes. Ricochet formó su propia red de radio privada en paquete mediante la instalación de repetidores digitales en arboletes públicos (un repetidor cada media milla [800 m] en promedio).



Estándares inalámbricos: IEEE 802.11 y NDIS

Información

La panacea inalámbrica ideal, sería decirle a la gente que puede llevar su computadora laptop a cualquier parte y hacerla funcionar como si estuviera conectada a la LAN de su compañía.

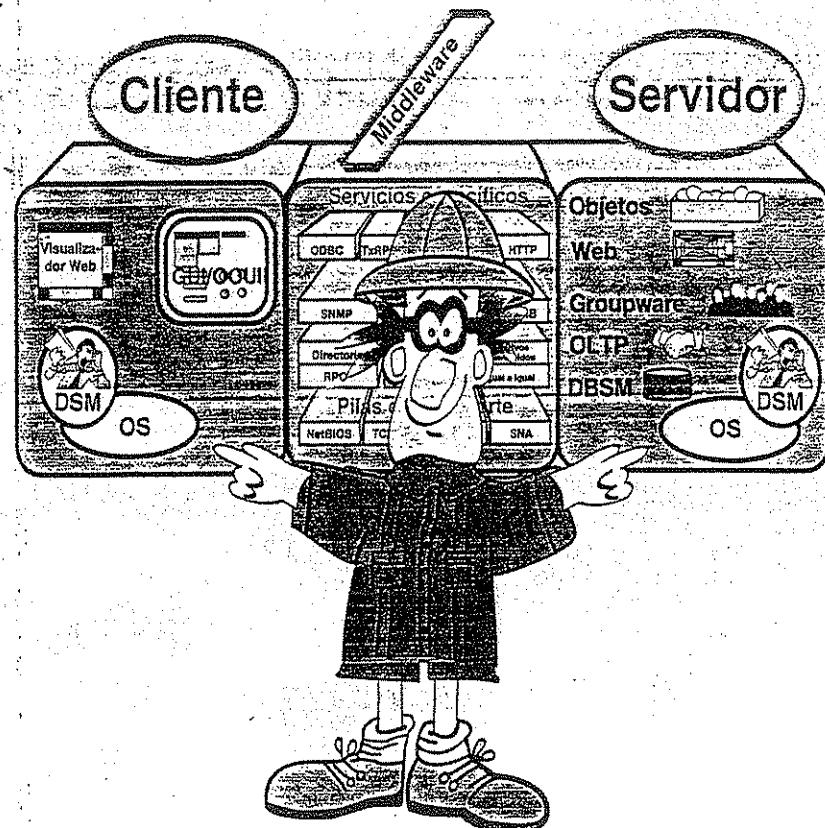
Dan Croft, vicepresidente ejecutivo
Ardis
(Marzo de 1996)

¿Quién se hará cargo del orden en el cielo? Todos los proveedores de redes inalámbricas forjan en la actualidad su propia "pista celestial". Para hacer orden en el caos, la comisión IEEE 802.11 recibió la ingrata encomienda de crear estándares para las redes inalámbricas. Con éstos se pretende cubrir las siguientes áreas: módems inalámbricos, API, dispositivos de PCMCIA, seguridad de datos en el aire, puentes desde LAN inalámbricas-inalámbricas, pautas para impedir interferencia de difusión y un protocolo de acceso a medios para transmisiones de radio e infrarrojos. Luego de tres largos años de gestación, es probable que para el momento en que usted lea este libro ya se haya aprobado el estándar IEEE 802.11. Asimismo, a principios de 1996 ya se hallaba terminado un nuevo estándar NDIS para aplicaciones inalámbricas. Esto significa que las pilas de comunicación existentes podrán acceder transparentemente a cualquier módem inalámbrico, incluidos el celular, Ardis, CDPD, RAM y redes de PCS futuras. Estos estándares volverán a las opciones inalámbricas más accesibles y fáciles de desplegar. □

Introducción a la Segunda parte

Después de la visión general ¿está listo para entrar en acción? Nos aguarda una jornada llena de peligros. ¿Trajo su chaleco antibalas? Más le vale haberlo hecho, porque cruzaremos una zona de guerra. No; no es la guerra de las galaxias, sino la guerra de los sistemas operativos para cliente/servidor, casi tan feroz como aquélla. Le bastará con detenerse en cualquiera de las páginas de esta parte para escuchar el clamor de la batalla. Pero no se preocupe: está en buenas manos.

Esta Segunda parte consiste en una descripción general de las actividades de clientes y servidores en la vida real y de lo que necesitan de sus sistemas operativos. Abordaremos brevemente varios OS y veremos qué les ofrecen a las partes tanto del cliente como del servidor. En lo que respecta a los OS, lo bueno y lo malo no existe; todo se reduce a un prodigioso acto de equilibrio con abundantes áreas sombreadas en gris. Todo es confuso y está en permanente movimiento. Si usted toma una decisión incorrecta o se monta en la ola de cliente/servidor equivocada, sencillamente saldrá lastimado (o tendrá que abordar la siguiente nave espacial de regreso a Marte). Los grandes participantes en cliente/servidor pueden permitirse una visión más profana de los OS y probar todas las plataformas. Si éste es su caso, hágalo. De no ser así, tendrá que ser muy cuidadoso en la elección de sus OS.



Capítulo 5

Ci en te s er vi do re s y s is te m a s o p e r a t i v o s



Examina cada vereda detenida y concienzudamente. Pruébala tantas veces como lo creas necesario. Hazte después una pregunta —después pregúntate a tí mismo—, sólo a tí, una pregunta... ¿Esta vereda tiene corazón? Si lo tiene, es buena; si no lo tiene, no te servirá de nada.

Carlos Castaneda,
Las enseñanzas de Don Juan

Este capítulo comienza con una breve descripción de las actividades de clientes y servidores comunes en la vida real. Después examinaremos lo que cada parte de la ecuación cliente/servidor necesita de un sistema operativo. En este capítulo se le brindarán los conocimientos esenciales, para saber qué buscar en una plataforma para cliente/servidor.

ANATOMÍA DE UN PROGRAMA SERVIDOR

La función de un programa servidor es la de *atender* a clientes múltiples interesados en un recurso compartido en propiedad del servidor. En esta sección describiremos un día en la vida de un servidor común. He aquí lo que realiza un programa de servidor común:



- **Semáforos.** Un sistema operativo debe contar con mecanismos simples de sincronización para impedir la confusión de tareas concurrentes al acceder éstas a recursos compartidos. Conocidos como *semáforos*, estos mecanismos se usan para sincronizar las acciones de tareas independientes del servidor y alertarlas cuando ocurre un evento significativo.
- **Comunicaciones entre procesos (IPC: interprocess communications).** Un sistema operativo debe disponer de mecanismos que permitan que procesos independientes intercambien y comparten datos.
- **Comunicaciones entre procesos locales/remotos.** Un sistema operativo debe permitir el redirecciónamiento transparente de llamadas entre procesos a un proceso remoto a través de una red sin que la aplicación se percate de ello. La extensión de las comunicaciones entre procesos más allá de los límites de la máquina es decisiva para el desarrollo de aplicaciones en las que recursos y procesos puedan circular fácilmente entre diversas máquinas (es decir, que hagan posible el crecimiento y mayor rapidez de los servidores).
- **Hilos.** Éstos son unidades de concurrencia provistas dentro del programa mismo. Los hilos son útiles para la creación de programas servidores muy concurrentes y sujetos a eventos. Cada evento en espera puede ser asignado a un hilo, que bloqueará las acciones hasta que el evento efectivamente ocurra. Mientras tanto, otros hilos pueden emplear productivamente los ciclos del CPU para desempeñar tareas útiles.
- **Protección entre tareas.** El sistema operativo debe proteger a las tareas para evitar que interfieran en sus respectivos recursos. Debe impedirse que una tarea descomponga al sistema en su totalidad. La protección se extiende también al sistema de archivos y las llamadas al sistema operativo.
- **Sistema de archivos de alto desempeño para usuarios múltiples.** El sistema de archivos debe soportar múltiples tareas y contar con candados que protejan la integridad de los datos. Los programas servidores suelen manejar muchos archivos al mismo tiempo. El sistema de archivos debe soportar grandes cantidades de archivos abiertos sin que por ello se deteriore mayormente su desempeño.
- **Administración eficiente de memoria.** El sistema de memoria debe soportar eficientemente programas muy grandes y objetos de datos también muy grandes. Estos programas y objetos de datos deben circular fácilmente al y desde el disco, preferentemente en pequeños bloques granulares.
- **Extensiones de tiempo de ejecución de vinculación dinámica.** Los servicios del sistema operativo deben ser extendibles. Debe disponerse de un mecanismo que permita la ampliación de los servicios en tiempo de ejecución sin necesidad de recompilar el sistema operativo.

Servicios complementarios

Los servicios complementarios provén el software avanzado de sistema que explota el potencial distribuido de las redes, ofrecen acceso flexible a la información compartida y hacen que el sistema sea más fácil de administrar y mantener. Facilitan asimismo la creación de nuevas



aplicaciones servidor a los proveedores independientes de software (*ISV: independent software vendor*) e integradores de sistemas. En la Figura 5-2 aparecen algunos de los servicios complementarios que los programas servidores pueden esperar de su sistema operativo. A continuación nos ocuparemos de estas expectativas, en orden ascendente de acuerdo con la figura. Algunas de ellas son por lo pronto simples deseos, pero llegará el momento en que formen parte de la mayoría de los sistemas operativos.



Figura 5-2. Qué esperan obtener los programas servidores de las extensiones de sus sistemas operativos.



vidores (para crear así un grupo de servidores) o cambiar la máquina servidor en funciones por la generación más reciente de superservidores de PC. Los servidores múltiples eliminan todo límite máximo al crecimiento de la capacidad del servidor. Los servidores ordinarios pueden rendir esa misma capacidad trabajando en todo tipo de combinaciones. Tal como explicaremos en la Tercera parte, extensiones de sistemas operativos de redes como el *entorno de computación distribuida (DCE: distributed computing environment)* y monitores de TP como CICS, Encina y Tuxedo constituyen la base de conducción necesaria para la creación de conjuntos de servidores cooperativos. Con el tiempo, también ORB se desempeñará en este terreno.

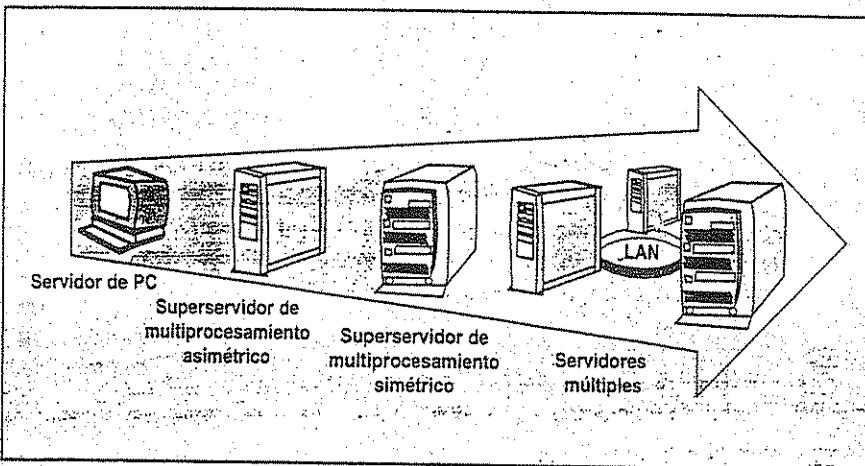


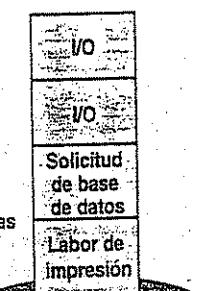
Figura 5-3. Historia de la ampliación del servidor de PC.

SUPERSERVIDORES DE MULTIPROCESAMIENTO

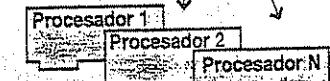
Información

Si usted necesita mayor capacidad de servidor, debe conocer la nueva generación de *superservidores*. Se trata de máquinas totalmente cargadas que incluyen multiprocesadores, matrices de disco de alta velocidad para I/O intensivas y características de tolerancia de fallos. Los sistemas operativos pueden reforzar al hardware del servidor ofreciendo soporte directo a los multiprocesadores. Con la adecuada división de trabajo, los multiprocesadores son capaces de incrementar el rendimiento laboral y la velocidad de aplicaciones servidor. Un servidor con multiprocesador es escalable. Los usuarios pueden obtener un mejor desempeño de sus servidores añadiendo sencillamente más procesadores, no más servidores. El procesamiento múltiple puede ser de dos tipos: asimétrico y totalmente simétrico (véase Figura 5-4).

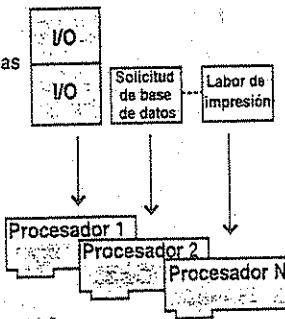
El *multiprocesamiento asimétrico* impone una jerarquía y una división de trabajo entre procesadores. Sólo un procesador específico, el procesador maestro, puede correr el sistema operativo en cualquier momento. El procesador maestro controla (en disposición de



Tareas



Tareas



Multiprocesamiento asimétrico

Figura 5-4. Multiprocesamiento simétrico y asimétrico.

acoplamiento estrecho) a procesadores subordinados, destinados a funciones específicas tales como I/O de disco o I/O de red. Un coprocesador es una modalidad extrema de codependencia; un procesador controla por completo a un procesador subordinado mediante instrucciones entrelazadas de propósito especial. El coprocesador cuenta con un hardware determinado de propósito especial, diferente al del procesador principal. Un ejemplo es un coprocesador gráfico.

El *multiprocesamiento simétrico (SMP: symmetric multiprocessing)* trata como iguales a todos los procesadores. Cualquier procesador puede realizar el trabajo de los demás. Las aplicaciones se dividen en hilos que pueden correr concurrentemente en cualquier procesador disponible. Cualquiera de los procesadores del grupo puede correr el núcleo del sistema operativo y ejecutar hilos generados por el usuario. El multiprocesamiento simétrico eleva el desempeño de la aplicación misma, así como el rendimiento total del sistema del servidor. Lo ideal sería que el sistema operativo soportara el multiprocesamiento simétrico aportando tres funciones básicas: un núcleo reentrant del OS, un generador global de itinerarios que asigne hilos a los procesadores disponibles y estructuras de I/O compartidas. El multiprocesamiento simétrico requiere de hardware de procesador múltiple con alguna modalidad de memoria compartida y cachés de instrucciones locales. Pero requiere sobre todo de nuevas aplicaciones capaces de explotar el paralelismo de hilos múltiples. Las escasas aplicaciones en el mercado que explotan el SMP son los administradores de bases de datos de SQL, como Oracle 7 y Sybase.



Clientes con GUI

Los clientes con GUI simple son aplicaciones en las que las ocasionales solicitudes al servidor resultan de una interacción humana con una GUI. La interfaz de GUI simple es ideal para aplicaciones electrónicas de negocios tipo OLTP con tareas repetitivas y grandes volúmenes. También constituyen excelentes clientes en primer plano (front-end) de servidores de bases de datos. Las aplicaciones cliente con GUI simple son manifestaciones gráficas de los diálogos previamente ejecutados en terminales no inteligentes. Las GUI remplazan los "horrores en pantalla verde" con cuadros de diálogo gráficos, color, barras de menús, recuadros de desplazamiento y ventanas desplegables y de aparición instantánea (véase Figura 5-7). En los cuadros de diálogo, o diálogos para abreviar, de GUI simple se emplea el modelo de objeto/acción, por el cual los usuarios pueden seleccionar objetos y posteriormente las acciones por realizar en los objetos elegidos. La mayoría de los cuadros de diálogo, o diálogos para abreviar, son seriados. Este modelo de interacción del usuario es el que se utiliza predominantemente en aplicaciones como Windows 3.X y OSF Motif. Se le conoce también como modelo gráfico CUA 89.

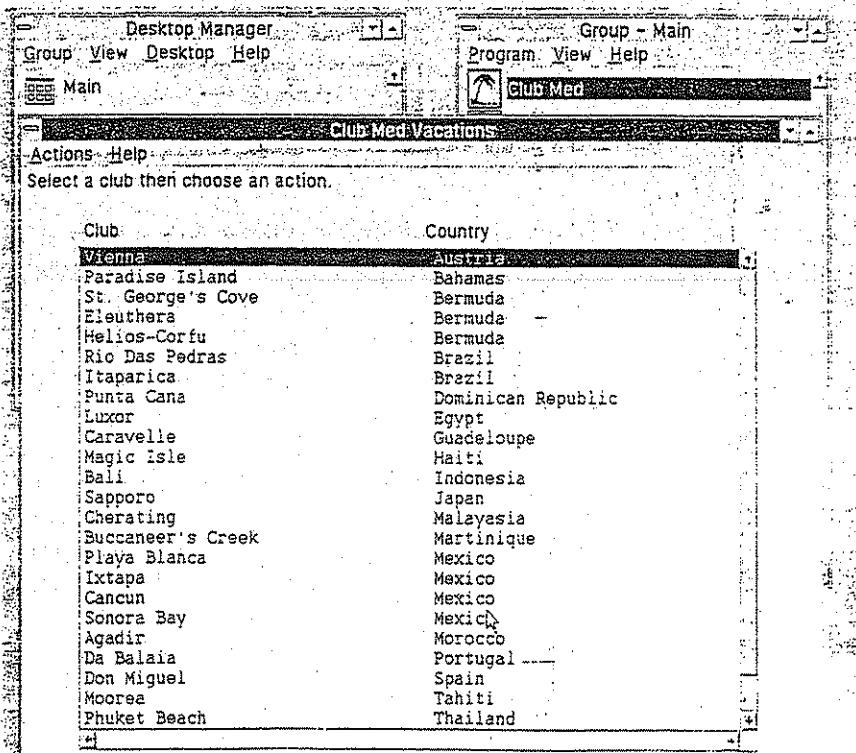


Figura 5-7. Aplicación tangible de GUI de Club Med.

Clientes con interfaz del usuario orientada a objetos (OOUI)

La alegoría de *interfaz del usuario orientada a objetos* (OOUI: object-oriented user interface) se emplea para brindar lo que Bill Gates, director de Microsoft, denomina *información en las yemas de los dedos*. Se trata de una interfaz del usuario orientada a objetos altamente icónica que ofrece acceso inconsútil a información en formatos sumamente visuales. Las OOUI son útiles para empleados que deben trabajar con información y realizar múltiples tareas diversas cuya secuencia es imposible de prever. Como ejemplos pueden citarse las aplicaciones ejecutivas y de apoyo de decisiones, los sistemas de capacitación basados en multimedia, las consolas de administración de sistemas y las estaciones de trabajo de intermediarios bursátiles. El apetito de comunicaciones de las OOUI es insaciable. Sus objetos de escritorio necesitan comunicarse entre ellos y con servidores externos. Las comunicaciones son obligadamente de tiempo real, interactivas y altamente concurrentes.

Son ejemplos de OOUI el OS/2 Workplace Shell, NextStep, OS de Mac y, en cierta medida, Windows 95. Las OOUI actuales proporcionan una alegoría de escritorio visual (que puede ser concebido como una serie de juegos) en el que usted puede reunir objetos y programas afines

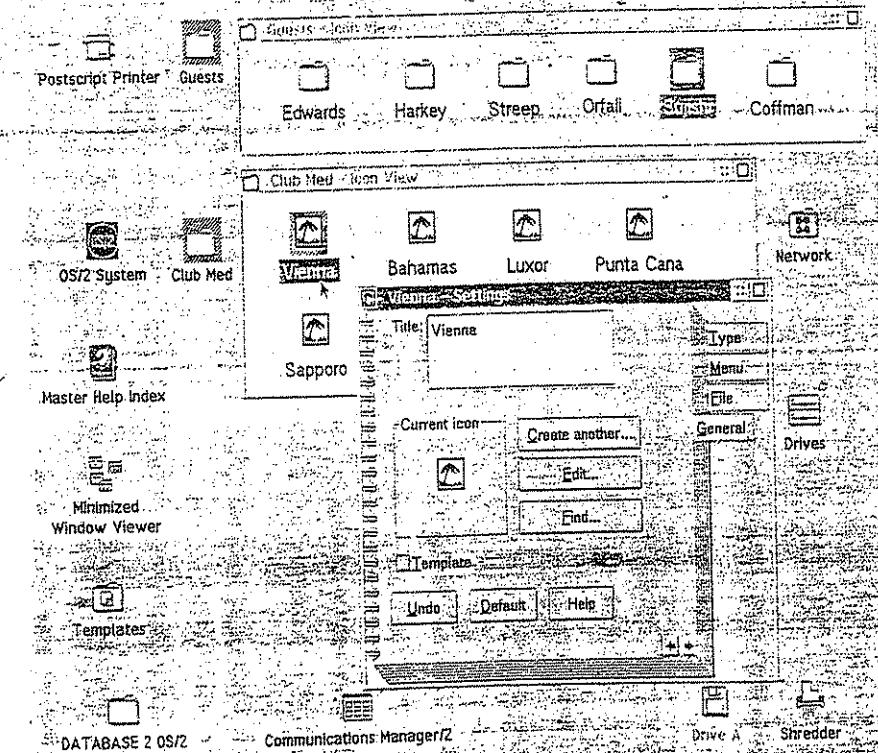


Figura 5-8. Aplicación tangible de OOUI de Club Med.

Tabla 5-1. GUI contra OOUI.

Característica	Interfaz gráfica del usuario (GUI)	Interfaz del usuario orientada a objetos (OOUI)
Estructura de aplicación	Una aplicación gráfica consiste en un ícono, una ventana primaria con una barra de menús y una o más ventanas secundarias. La atención se concentra en la tarea principal. Las tareas subordinadas son soportadas por ventanas secundarias y nuevas. Los usuarios deben seguir la rígida estructura de la tarea (y muchos no llegan más lejos). Una aplicación representa una tarea.	Una aplicación gráfica consiste en un conjunto de objetos cooperativos del usuario. Todo lo que se ve es objeto. Cada objeto es representado por un ícono y tiene al menos una vista. Los objetos pueden reutilizarse en muchas tareas. Los límites de la aplicación son vagos. El usuario define una aplicación al reunir un conjunto de objetos. Éstos pueden provenir de uno o más programas y se integran a los objetos de escritorio provistos por el sistema (como impresoras y trituradoras). Los usuarios pueden innovar y crear sus propios conjuntos de objetos similares a Lego.
Iconos	Los iconos representan una aplicación en ejecución.	Los iconos representan objetos que pueden manipularse directamente.
Puesta en marcha de una aplicación	Los usuarios emprenden aplicaciones antes de elegir un objeto con el cual trabajar.	Los usuarios abren el objeto en el escritorio, con lo que aparece una vista de ventana de aquél.
Ventanas	Los usuarios abren una ventana primaria y después especifican los objetos con los que desean interactuar. La misma ventana sirve para el despliegue de otros objetos.	Una ventana es una vista del interior de un objeto. Entre ventana y objeto existe una relación de identidad.
Menús	Los menús son el método básico para navegar en una aplicación.	Cada objeto posee un menú de contexto. Se navega en una aplicación o entre aplicaciones manipulando directamente los objetos. El escritorio funciona como un gran menú; los íconos representan los objetos que es posible manipular.
Visualización de la aplicación activa	Los iconos representan ventanas reducidas de aplicaciones activas.	Los iconos se amplifican con énfasis en el uso para representar a un objeto activo.
Manipulación directa	Una aplicación puede ofrecer manipulación directa mediante instrucciones específicas.	Los objetos se crean, comunican, mueven y manipulan mediante manipulación de arrastrar y soltar.
Creación de nuevos objetos	Los objetos se crean según las especificaciones de cada aplicación, usualmente por medio de alguna modalidad de mecanismo de copiado u opciones de menú: nuevo o abierto.	Una carpeta de plantillas contiene una de éstas para cada tipo de objetos. Para crear una nueva instancia de un objeto, se arrastra su plantilla donde se desea que resida el nuevo objeto.

Tabla 5-1. GUI contra OOUI (continuación).

Característica	Interfaz gráfica del usuario (GUI)	Interfaz del usuario orientada a objetos (OOUI)
Acciones	Se elige un objeto, y luego una acción de la barra de menús.	A parte de elegir acciones de menús, el usuario puede arrastrar objetos a íconos para realizar operaciones; por ejemplo, arrastrar un archivo al ícono de una impresora.
Contenedores	Recuadros de listas basadas en texto son la modalidad básica de contenido.	Además de recuadros de listas, las OOUI cuentan con objetos contenedores, como carpetas y cuadernos, los cuales pueden contener a su vez otros objetos. Las acciones desarrolladas en objetos contenedores afectan a todos los objetos en su interior.
Atención	La atención se concentra en la tarea principal.	La atención se concentra en los objetos y tareas activos.
¿Quién ejerce el control?	El control se alterna entre el usuario y la aplicación.	Todas las aplicaciones se comportan igual y el usuario actúa como conductor. Puede decirse que el usuario es el programador visual del escritorio.
Ejemplos de productos	Windows 3.X, NT 3.5 y Motif.	OS/2 Workplace Shell, NextStep, OS de Mac, Windows 95 y NT 4.0.

Documentos compuestos: OOUI en esteroides

Las *estructuras de documentos compuestos* —como OLE y OpenDoc— son la mejor y más reciente tecnología de OOUI (véase Figura 5-9). Los documentos compuestos pueden concebirse como OOUI en esteroides. Cada objeto visual en la pantalla es un componente vivo. Algunos componentes son contenedores también, lo que significa que pueden alojar otros componentes. El contenido de todo componente puede editarse “en el mismo punto”, sin importar que esté profundamente incrustado en otros componentes. Un componente es una pieza independiente de software que puede comprarse por separado en el mercado. Los componentes pueden combinarse en series visuales semejantes a aplicaciones.

En el caso de las OOUI actuales, los objetos visuales suelen ser íconos rectangulares con vistas subyacentes. En el de los documentos compuestos, los componentes pueden adoptar cualquier forma, y se les puede mover e incrustar a voluntad, redimensionarlos, ampliar su contenido y reorganizarlos visualmente dentro de un contenedor visual al gusto del usuario. Los componentes comparten automáticamente el menú, portapapeles y paleta del documento. Todo parecería inconsútil; semeja un tapiz visual. Así, los documentos compuestos son una mejor simulación de la realidad que las OOUI cosecha CUA-91. Los documentos compuestos tridimensionales brindarán simulaciones todavía más reales. El documento se convierte de este modo en un mundo virtual poblado por componentes.

En la actualidad se dispone de OLE y OpenDoc en forma de herramientas gratuitas. OLE también forma parte de Windows 95; aporta la base de componentes para los productos de



¿QUÉ NECESITA UN CLIENTE DE UN OS?

Cada uno de los tres tipos de clientes aquí descritos supone un conjunto diferente de requerimientos de sistema operativo. Tales requerimientos se relacionan en la Tabla 5-2. Como se advertirá, todas las aplicaciones cliente precisan de algún mecanismo para comunicar solicitudes de servicio y archivos a un servidor. Las tres categorías de clientes funcionarán mejor en un robusto entorno multitarea. El vigor del entorno del cliente es particularmente importante, ya que a los proveedores de sistemas les resulta imposible probar el software cliente en todas las posibles combinaciones de hardware/software (no se puede determinar qué debe correr la gente en sus PC). Así, es importante usar un sistema operativo capaz de proteger los programas de colisiones e impactos. Ningún programa cliente debe provocar la descompostura del sistema (para remediarla se requeriría de reinicialización).

Los clientes con GUI y OOUI operan mejor con un mecanismo de hilos o equivalente para el manejo de las solicitudes en segundo plano. Al emplear hilos distintos para la interfaz del usuario y el procesamiento de fondo, el programa puede responder a entradas del usuario al tiempo que un hilo diferente maneja la interacción con el servidor. Así es como las GUI evitan el delator ícono de "reloj de arena", señal inequívoca de que el entorno de computación no está

Tabla 5-2. ¿Qué necesita un cliente de un OS?

Requerimientos de un OS	Cliente sin GUI	Cliente con GUI simple	Cliente con OOUI
	Sí multitarea	Sí multitarea	
Mecanismo de solicitud/respuesta (preferiblemente con transparencia local/remota)	Sí	Sí	Sí
Mecanismo de transferencia de archivos para movilizar instantáneas de imagen, texto y base de datos	Sí	Sí	Sí
Multitarea preferente	No	Sí	Deseable
Prioridades de tareas	No	Sí	Deseable
Comunicaciones entre procesos	No	Sí	Deseable
Hilos para comunicaciones de fondo con el servidor y recepción de llamadas de verificación de servidores	No	Sí Sí (a menos que se prefiera el ícono de reloj de arena)	Sí
Vigor del OS, incluidas protección entre tareas y llamadas reentrantes de OS	No	Sí	Deseable
GUI de Windows 3.X (modelo CUA-89)	No	No	Sí
OOUI y documentos compuestos	No	No	No

a la altura de usuarios humanos. Los hilos también contribuyen a que los clientes respondan a las llamadas asíncronas de un servidor (es decir, a que implementen *llamadas de verificación*). Se requiere asimismo de multitarea preferente basada en prioridades para responder a dispositivos de multimedia y crear aplicaciones cliente en las que se desplieguen paralelamente múltiples diálogos.

HÍBRIDOS CLIENTE/SERVIDOR

Otro punto por considerar es que la industria está rebasando ya el modelo puro de cliente/servidor. Esto se debe a que el cliente recibe cada vez más inteligencia (y datos). Los clientes de bases de datos conservan localmente instantáneas de tablas. Los clientes de monitores de TP coordinan transacciones de servidores múltiples. Los clientes de groupware mantienen colas. Los clientes de multimedia realizan registros de entrada y salida de carpetas. En fin, los clientes de objetos distribuidos aceptan solicitudes de objetos de cualquier parte. Estos clientes de la Nueva Era deben ofrecer una función de *servidor ligero*, paso intermedio hacia el cumplimiento de la visión post-carestía de funciones plenas de cliente y servidor en cada máquina. Cabe señalar que aun las PC en Internet más simples deben ser capaces de descargar ubicaciones embarcables, ejecutar applets de Java y recibir llamadas de un servidor. Por ejemplo, el tablero de instrumentos de su automóvil podría recibir de pronto una llamada de un servidor para mostrar la ubicación de usted en un mapa.

Un servidor ligero ofrece un hilo, cola o proceso en segundo plano en la máquina cliente, capaz de aceptar solicitudes en red *no demandadas*, usualmente de un servidor. Por ejemplo, un servidor puede llamar a sus clientes para que sincronicen sus candados en una transacción de larga duración, regeneren una instantánea de base de datos o remitan un documento con multimedia con registro de salida. Una implementación de servidor ligero (en oposición a un servidor completo) no necesita soportar acceso concurrente a recursos compartidos, equilibrio de cargas o comunicaciones de multihilos. A los clientes que brindan funciones de servidor ligero les llamamos *híbridos* (en oposición a los puros).

CONCLUSIÓN

Recorrer los pasos necesarios para la elección de una plataforma de cliente/servidor es un excelente ejercicio mental. Nos ayuda a perfilar nuestra comprensión de los aspectos importantes y las alternativas de plataformas. Las cosas no son blancas o negras en la realidad. En el nivel del cliente/servidor intergaláctico no existen mundos de cómputo herméticamente sellados. Además, muy pocas máquinas hicieron su aparición como sistemas de cliente/servidor en ciernes; la mayoría de las computadoras del mundo, aun las PC, son "sistemas de herencia". Es de esperar que usted tenga que vérselas con combinaciones de plataformas en grandes dosis. Afortunadamente, el middleware cliente/servidor está empezando a convertirse en el *uniformador entre plataformas*. Así, una vez que domine por completo una plataforma, podrá trabajar con la siguiente sin mayor problema. Mientras tanto, la diversidad es la causante de que cliente/servidor ofrezca tantas emociones.

de las principales tendencias en el campo de los OS de cliente y servidor; comprobará usted que nos gustan las emociones fuertes. Más adelante tendremos el gusto de presentarle a los participantes más destacados en esta lid.

TENDENCIAS DE OS DEL CLIENTE

En 10 años, las computadoras de escritorio serán una rareza. La actual capacidad de las estaciones de trabajo habrá pasado a los relojes de pulso y se hallará en todas partes.

Ted Nelson, gurú
(Noviembre de 1995)

En el capítulo anterior nos referimos a lo que los clientes necesitan de sus OS. He aquí algunas tendencias perceptibles en la industria del cliente:

- *El escritorio se fragmenta cada vez más.* El paso a OS de 32 bits está fragmentando el escritorio. A diferencia de sus predecesores—DOS y Windows 3.X—, Windows 95 no es dueño del cliente (véase Figura 6-1). En efecto, Microsoft cuenta con cuatro OS del cliente rivales entre los cuales elegir: Windows 3.X, Windows para Trabajo en Grupo, Windows 95 y NT Workstation, para no hablar de MS-DOS. Aparte de Windows, OS/2 Warp está fuertemente atrincherado en los escritorios empresariales, mientras que el OS de Mac cuenta con seguidores apasionados entre las multitudes en Internet. Así, la diversidad es un hecho. El antiguo tejido homogéneo DOS-Windows está empezando a romperse en jirones en sus costuras.
- *El Web generará una enorme demanda de PC para Java.* Se trata de PC que ejecutan originalmente un OS para Java. Sun ha bautizado como Kona a su nuevo OS para Java. Suponemos que habrá una gran demanda de "PC en red" de 500 dólares que operen con Kona y descarguen applets para el desempeño de tareas. Como explicamos en la Primera

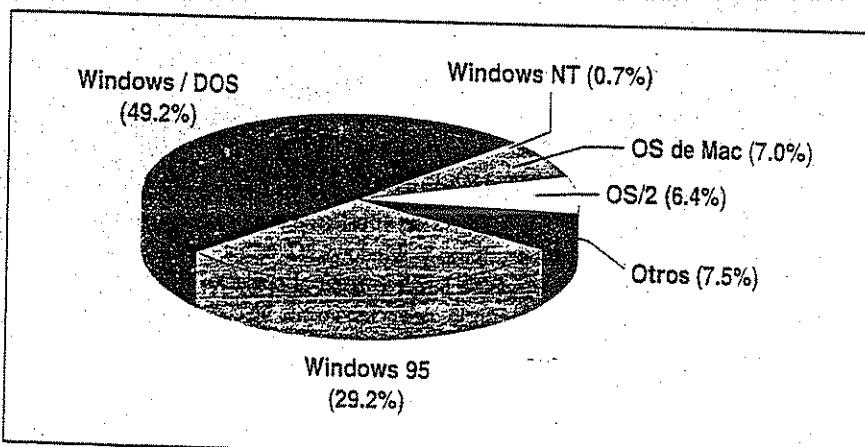


Figura 6-1. Mercado mundial de OS para PC, 1995 (Fuente: IDC).

parte, se dispone ya del ancho de banda suficiente fuera de los muros de protección empresariales para la efectiva descarga de applets a solicitud expresa. De este modo, quizás a usted le baste con Kona. La idea es en este caso la probabilidad de que las PC para Java resulten más fáciles de usar, mantener y mejorar que las PC actuales. Introducirán al Web a millones de usuarios que no podrán permitirse, comprender o enfrentar el costo de poseer un OS en toda forma. Los clientes Web (o máquinas para Java) pueden volverse tan ubicuos como el teléfono. Claro que antes alguien tiene que generar estas applets.

- *Habrá una enorme demanda de PC grandes.* Éstas son PC ordinarias que actúan lo mismo como clientes que como servidores. Estos servidores deberán ofrecerse en paquetes comerciales; nadie podría darse el lujo de un administrador de sistemas con cada servidor.
- *Las ubicaciones embarcables se convertirán en los nuevos escritorios.* Hoy son mayoría quienes se manejan en los terrenos del escritorio de Windows. Pero muy pronto ocuparemos nuestras ubicaciones embarcables favoritas, mundos virtuales que nos conectarán en gran medida con la red. Por ejemplo, habrá ubicaciones para abogados, dentistas y adolescentes de 12 años. Así pues, usted pasará su tiempo en mundos virtuales que lo harán sentirse como en su casa. El escritorio ha dejado de ser una ubicación monolítica única; ahora tendrá usted múltiples ubicaciones para escoger.
- *Habrá clientes incrustados por todas partes.* Ésta es la visión de Novell de "la red de miles de millones de nodos para el año 2000". Para alcanzar esa cifra, en inyectores de combustible, fotocopiadoras, despachadoras de alimentos, refrigeradores, cajas registradoras, dispositivos telefónicos, cajas bancarias automáticas y camionetas pick-up se instalarán millones de pequeños nodos en red. Sun calcula que en 1999 un hogar común contará con 100 microcontroladores. Estos nodos requieren de un OS de huella diminuta capaz de ejecutar también alguna modalidad de middleware cliente/servidor. De nueva cuenta, puede ser que un OS para Java—complementado con un ORBlet de CORBA—sea todo lo que necesitemos en este entorno incrustado. Novell también está trabajando en un OS llamado *Nested NetWare*.

En síntesis, el cliente se está transformando. El OS para escritorio no será el centro absoluto del universo del cliente. Los servidores—vía visualizadores Web y ubicaciones embarcables—ejercerán mayor influencia en cuanto a lo que habrá de desplegarse en el espacio visual del cliente. Los nuevos clientes—como las PC en Internet y los aparatos eléctricos inteligentes—requerirán de OS aptos para red con huellas sumamente pequeñas; Kona parece responder a este perfil. En definitiva, el cliente deberá ser también un servidor personal en plenas facultades. ¿Un solo OS del cliente podrá hacerlo todo? Nosotros no apostaríamos a eso.

OS DEL CLIENTE: CONOZCA A LOS PARTICIPANTES

Como puede verse en la Figura 6-2, la mayoría de las plataformas del cliente de hoy pertenecen a Microsoft. La única competencia en el escritorio es la representada por OS/2 Warp Connect y el OS de Mac. El OS para Java puede convertirse en un tiempo en un competidor formidable, especialmente en el sector de PC en Internet y clientes incrustados. Aunque ubicuos, DOS y Windows 3.X no son plataformas para el cliente del todo aceptables. Sus pequeños y deficientes cerebros de computadora son demasiado débiles como para ofrecernos lo que se necesita para



Windows 95

Windows 95 puede concebirse como Windows 3.X en esteroides. Es un OS de escritorio de entrada con una interfaz de usuario semejante a OGUI, conexión y manejo y cómodas características de autodescubrimiento de hardware. Cuenta también con muchas características de red útiles, como asociabilidad en red, editor de registro remoto y un agente de SNMP integrado. Es apto para red. Contiene una pila mínima de TCP/IP, NetBEUI, IPX/SPX y PPP.

¿Sus inconvenientes? Primero, la OGUI de Windows 95 es inconsistente. Combina los paradigmas de OGUI y GUI, lo que puede dar lugar a grandes confusiones. Segundo, sigue basándose en DOS; es este mismo con una cara bonita. Esto quiere decir que aplicaciones de 16 bits no cuentan con protección contra colisiones, y disponen sólo de multitarea limitada (no preferente). Tercero, Windows 95 no parece suficientemente robusto para el mercado de clientes empresariales. Únicamente el 10% de las empresas se han mudado a él, contra el 30% de los consumidores. Todo indica que las empresas están optando por OS/2 o NT Workstation; muchas aguardan la aparición de NT 4.0, con su nueva interfaz de usuario de Windows 95.

Windows NT Workstation

Windows NT Workstation es un robusto OS del cliente de 32 bits. Soporta multitarea preferente, multihilos, protección de memoria y un sistema de archivos para transacciones. Es apto para red; soporta TCP/IP, NetBEUI, IPX/SPX, PPP y AppleTalk. Sus límites de sistema máximos son escasos. Ofrece también seguridad de nivel C2. NT 4.0 —ahora en prueba beta— soportará la nueva OGUI de Windows 95 en sustitución de la GUI de Windows. Será asimismo la primera plataforma de Microsoft en soportar Network OLE (e DCOM).

¿Sus inconvenientes? En primer lugar, NT es un devorador de recursos. Requiere de un mínimo de 16 MBytes de RAM y 512 MBytes en disco. En segundo, en comparación con Windows 95 y OS/2 Warp, su soporte para laptops es muy deficiente; dispone de soporte de PCMCIA y administración de capacidad limitados. En tercero, NT ofrece una pobre emulación de DOS y aplicaciones de Windows de 16 bits; no soporta controladores de dispositivos virtuales (VxD: *virtual device driver*). Además, no soporta conexión y manejo, lo que dificulta su configuración. Comparado con Windows —e incluso con OS/2—, su soporte de controladores de dispositivos es limitado.

Finalmente, NT Workstation es una plataforma de cliente de alto costo. Su precio es de 250 dólares, contra 100 (o menos) de Windows 95. Debido a todas estas limitaciones, no alcanza el mismo nivel de soporte de ISV del resto de las plataformas de Windows. En consecuencia, son pocas las aplicaciones que corren en él. Sólo se vendieron 480 000 copias en 1995, cifra menor para los estándares del cliente. Claro que esta situación puede cambiar por completo con el lanzamiento de NT 4.0.



Juzgado de divorcios



OS de Mac

Con 10-15 millones de usuarios, Mac es uno de los principales participantes en el ámbito del escritorio. De igual forma, los usuarios de Mac tienen una desproporcionada presencia en el Web. Tal vez representen casi el 20% de la población de clientes del Web. Para Apple, el Web es una importante oportunidad en software y hardware. La idea es extender al Web la facilidad de uso de Mac. Macintosh puede convertirse entonces en la plataforma de cliente ideal para Internet e intranets. En cuanto proveedor de hardware, Apple se halla en buena posición para enfrentarse a PC en Internet con *Pippin*, una PC para el Web de bajo costo basada en la Power Mac.

El arma secreta de Apple para imponerse en la esfera del cliente se llama Cyberdog, el cual es un juego de componentes de OpenDoc para Internet. El acceso al Web estará totalmente integrado al resto del OS. Apple cuenta con la experiencia visual inconsútil de OpenDoc para ofrecer una interfaz de usuario irresistible; Mac parecerá una galería de videojuego. La gente comprará computadoras Mac sólo para disfrutar de la mejor experiencia de multimedia en Internet. Tal como se indicará más adelante, más de 500 ISV desarrollan ya partes de OpenDoc para la Mac. Así, OpenDoc puede conducir a un renacimiento de la programación visual de Mac. Quizá contribuya con ello a que Mac recupere su puesto de principal de plataforma visual en la industria.

¿Cuáles son sus inconvenientes? Primeramente, el OS de Mac no es una plataforma de servidor muy buena; no es siquiera una buena plataforma de cliente avanzada. Su facilidad de amplia-

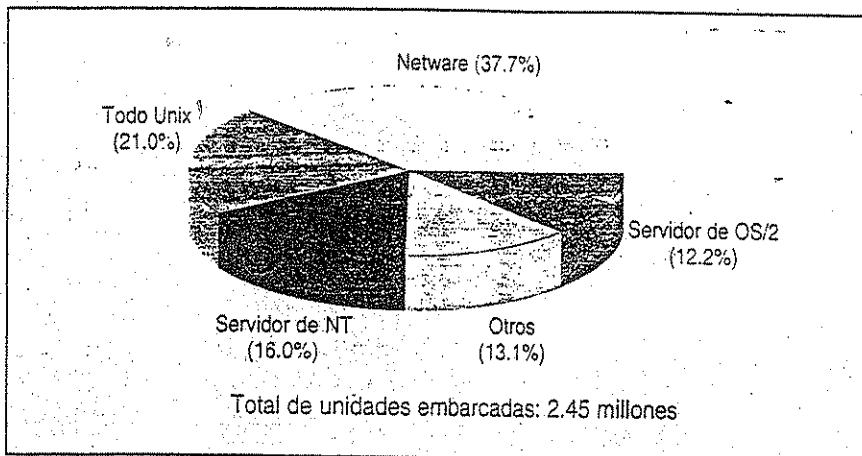
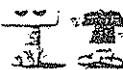


Figura 6-5. Embarques mundiales de OS en 1995 (Fuente: IDC).

etc. Ya pasó la época en la que sólo los "sumos sacerdotes" de la computación podían manipular servidores. Un moderno servidor de OS debe ofrecerle el mismo ámbito tangible de fácil uso de un escritorio ordinario.

OS DEL SERVIDOR: CONOZCA A LOS PARTICIPANTES

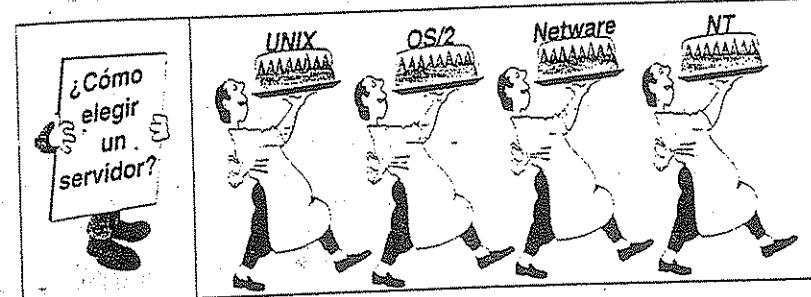
Así sea el diablo o el mismo Dios, tienes que servir a alguien.

Bob Dylan

Como se deduce de la sección anterior, la competencia en el mercado de servidores es intensa. Para volver a la Figura 6-2, en ella se observa cómo se han dividido el terreno las diferentes plataformas de servidor. Los participantes en el extremo bajo a medio del mercado del servidor son NetWare 4.1, OS/2 Warp Server, NT Server 4.0 y las diversas versiones de Unix sobre Intel, como SCO y Solaris. El extremo superior pertenece a los grupos de Unix y macro o superminicomputadoras capaces de fungir como servidores de PC. Los mayores competidores en el extremo más alto son los proveedores de macrocomputadoras con RISC que pueden ofrecer masivamente computación paralela, facilidad de ampliación y/o tolerancia de fallas (como, por ejemplo, Tandem, Pyramid, Stratus, IBM y Sequent). Las macrocomputadoras paralelas IBM basadas en MVS —con motores de transacciones, administración de sistemas basada en empresas y grandes bases de datos— son también importantes participantes en este campo.

NetWare 4.1

Por amplio margen, la mayor base instalada de servidores pertenece a NetWare de Novell. NetWare es un servidor de archivos sumamente veloz, efectivo y bien cimentado que soporta



inconsútilmente a clientes OS/2, Mac y Windows. El producto incluye también un servicio de directorio global semejante a X.500 y las herramientas para usarlo. NetWare es una plataforma del servidor bien administrada; el directorio global ofrece un solo punto de administración de la red. NetWare 4.1 SMP es similar a NetWare 4.1, salvo que en su caso un segundo núcleo maneja y ejecuta hilos entre procesadores.

En la actualidad, más de 4000 aplicaciones corren sobre NetWare, entre ellas los grandes DBMS. NetWare cuenta con una enorme infraestructura de soporte y ventas, compuesta por 200 000 ingenieros NetWare certificados (CNE: Certified NetWare Engineer) y 20 000 socios distribuidores en todo el mundo. Sus versiones futuras brindarán seguridad de nivel C2, así como soporte integrado para Internet, incluido un entorno de servidor de Java.

¿Cuáles son sus inconvenientes? El principal problema es que NetWare no es eficiente como servidor de aplicaciones. Para convertirse en proveedor de servidores de aplicaciones de propósito general, Novell debe encontrar la manera de abrir su plataforma para servidores. Con el fin de atacar este problema, lanzó los módulos cargables de NetWare (NLM: NetWare Loadable Modules) en NetWare 3.1, espacios para nombres especiales reservados en el servidor para permitirles a los programadores ofrecer nuevos servicios de sistema (o escribir/generar nuevas aplicaciones). Los módulos NLM para el servidor que se crean han sido cargados por NetWare para administrar esos espacios para nombres. Así, los NLM del usuario pasan a formar parte del núcleo del sistema operativo de NetWare. Novell proporciona herramientas y un entorno de programación para el desarrollo de NLM.

Pero a pesar de haber abierto NetWare al desarrollo de aplicaciones, los NLM se han convertido en el talón de Aquiles de este sistema operativo. Han reaccionado lentamente a los desarrolladores de terceros que se emplean en la generación de aplicaciones para sistemas operativos convencionales del servidor como Unix, NT y OS/2. Los NLM representan tantos problemas que resulta muy difícil (si no imposible) usar a NetWare como plataforma de servidor de aplicaciones de propósito general. Las tres principales causas de ello son: 1) limitada protección de memoria; 2) carencia de administración de memoria, pues NetWare no cuenta con memoria virtual, y 3) falta de soporte para multitarea preferente. Según Richard Finkelstein, la condición no preferente de NetWare descarga demasiadas responsabilidades en el desarrollador de aplicaciones, pues el sistema precisa de aplicaciones libres de errores, lo cual es "prácticamente imposible dada la complejidad del software de que se trata".

Finalmente, a lo largo de 1996 IBM introdujo una familia de productos servidores integrados bajo la denominación de código *Eagle*. Se trata de servidores Eagle de bases de datos, Lotus Notes, intercambio en Internet, monitor de TP, servicios de objetos de CORBA y seguridad y directorio de DCE. La idea de IBM es volver más accesibles las soluciones de cliente/servidor ofreciendo servidores en paquetes comerciales con características comunes de instalación, mantenimiento y soporte. Todos los servidores Eagle correrán en OS/2 y se beneficiarán de su entorno de fácil uso para el usuario. También correrán en AIX y NT.

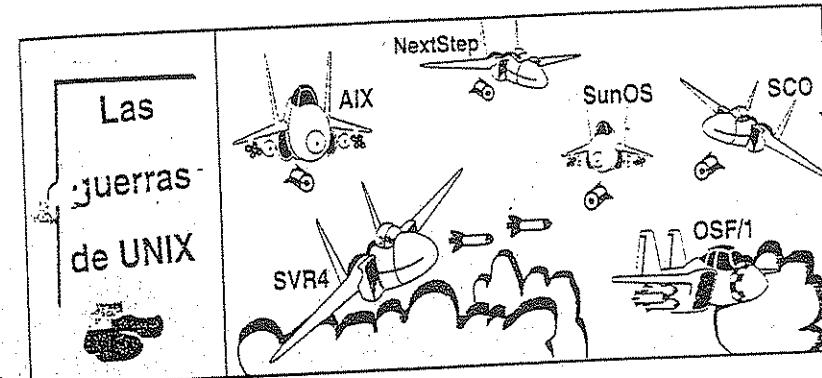
¿Cuáles son las desventajas de OS/2? Primera, OS/2 es una plataforma de servidor que trabaja exclusivamente con Intel: todo indica que OS/2 para Power PC murió al nacer. Cabe señalar sin embargo que, en este caso, la exclusividad de Intel puede representar un beneficio. Significa que OS/2 no incurre en la carga extra de una capa de portabilidad. Así, puede aprovechar plenamente el hardware de Intel y los buses de SMP y convertirse en el OS más veloz para los ubicuos servidores de Intel, los cuales representan el 80% de la totalidad de los servidores. Salvo por lo que se refiere al extremo más alto del mercado de servidores, operar únicamente con Intel no causa daño alguno.

La segunda desventaja de OS/2 son sus límites de sistema. Por ejemplo, sólo soporta archivos de hasta 2 GBytes. Las divisiones del disco se limitan a 512 GBytes. Estos límites pueden parecer razonables por lo pronto, pero nos causarán problemas cuando pasemos a multimedia. Finalmente, muchas de las funciones avanzadas de servidor de OS/2 siguen siendo cuestión del futuro. Entre ellas están la seguridad de C2, el soporte de Unicode (para la internacionalización), el soporte de megagrupos, los sistemas de archivos para transacciones y los archivos de correspondencia de memoria. Muchas de estas características estarán presentes en el inminente OS/2 *Merlin Server*.

Unix

Unix es un sistema operativo maduro y rico en funciones que puede ampliarse desde el escritorio hasta la supercomputadora. Se trata nada menos que del crisol de la industria de las computadoras. Su estrecha relación con universidades lo convierte en un gran incubador de nuevas ideas. La mayoría de estas ideas aparecen inicialmente en el mercado comercial como extensiones y variantes de Unix. Por su parte, la línea principal de Unix avanza con mucha mayor cautela. Unix puede jactarse de contar con todo un ejército de programadores, integradores y técnicos calificados.

La industria de servidores de Unix se desarrolló a partir de la reducción de aplicaciones para macrocomputadoras. En los últimos 15 años, Unix se desprendió exitosamente de las macrocomputadoras. Se volvió entonces una alternativa de "macrocomputadoras para la gente común". Sin embargo, en el entorno de LAN de PC se le tiene por "servidor para ricos". Pero a pesar de que, además, se ve acompañado ahora por opciones de PC de bajo costo como NT, OS/2 y NetWare, lo cierto es que sigue adelante, con un mercado de 30 000 millones de dólares que crece anualmente a una tasa del 25%. El más reciente sector de alto crecimiento para los servidores de Unix es Internet. Los estándares de Unix se han convertido en estándares de Internet, como en el caso del correo, FTP, TCP/IP y el servicio de nombramiento de dominios. En cierto sentido, Internet no es otra cosa que el aparador de la tecnología distribuida de Unix.



Los proveedores de Unix se han hallado desde siempre al frente del middleware cliente/servidor. Ahora introducen objetos de Java y CORBA, nuevamente mediante el recurso de incorporarlos a la infraestructura de Internet. Unix también es una opción adecuada de servidores de bases de datos, especialmente por su facilidad de ampliación. Finalmente, el mundo de Unix trabaja hoy en día en un estándar de 64 bits, con el que superaría a sus competidores: NT, OS/2 y NetWare.

Sus inconvenientes? El gran problema con Unix ha sido siempre cuál Unix escoger. Haciendo cuentas, ha habido más de 45 variantes de Unix en el mercado. Dado que Unix es un sistema operativo independiente del hardware, en principio debería ser posible correr una aplicación en cualquier máquina capaz de soportar Unix, desde una PC hasta una supercomputadora Cray. Esta capacidad de ampliación del servidor es muy atractiva, pero irreal. Dos factores impiden su práctica en la realidad:

- **Falta de compatibilidad binaria.** El mundo de Unix es diferente al de PC, en el que el software se presenta en paquetes comerciales de diskettes de bajo costo capaces de correr en cualquier PC o equivalente que opere con MS-DOS, NT, Windows 95 u OS/2. A diferencia de estos sistemas operativos, desarrollados por una compañía y comercializados sobre muchos tipos de plataformas de hardware, las diversas versiones de Unix varían grandemente. Aún se carece de un estándar binario de soporte amplio. Como mínimo, entonces, las aplicaciones de Unix deben recompilarse para ser trasladadas de una plataforma a otra. Esto significa muchos gastos y dolores de cabeza para los proveedores de software.

- **Diferencias funcionales entre distintas versiones de Unix.** Siempre habrá diferencias entre los productos de Unix: lo que los distribuidores desean es vender, para lo cual se requiere de diferencias funcionales que impidan permanentes guerras de precios imposibles de ganar. Y aunque Spec 1170 de X/Open —que soporta más de 1170 API del núcleo elegidas entre las 50 más importantes aplicaciones de Unix— servirá para relacionar entre sí las distintas versiones de Unix, sólo definirá las funciones de menor común denominador. Para obtener de una plataforma lo que usted pagó por ella, es probable que en este caso deba emplear sus extensiones y sacrificar la portabilidad.

La amenaza representada por NT está obligando a los proveedores de Unix a buscar algún tipo de unificación. SCO se ha convertido en el nuevo guardián del estándar de Unix, pero Sun sigue

Introducción a la Tercera parte

Los sistemas de proveedores múltiples no se prestan a la cooperación. Una de las intenciones más articuladas de la industria es incrementar la capacidad de cooperación de los sistemas.

Hal Lorin, autor de
Doing IT Right
(Prentice-Hall, 1996)

¡Felicitaciones! ¡Sobrevivió a las guerras de OS! ¿No le parecieron emocionantes? Los terrícolas no hacen otra cosa que pelear. Parece que esto está en su naturaleza. Pero ahora cambiaremos de rumbo y lo conduciremos por terrenos menos escabrosos. En lugar de combatir entre sí, esta vez los proveedores de cliente/servidor intentarán ocultar sus diferencias tras la fachada conocida como "imagen de sistema único".

No, nos malinterprete. Los tipos a los que conoció en la Segunda parte están lejos de envainar la espada. Lo que pretenden es lograr sencillamente que los sistemas de todos los demás se parezcan al suyo. Si es imposible deshacerse de los demás sistemas, hay que encontrar la manera de desaparecerlos (o de "volverlos transparentes", como debería decirse correctamente). ¿En qué consiste este truco de desaparición? En desprender capa tras capa de middleware hasta que todo sea la misma cosa. ¡Así de sencillo!

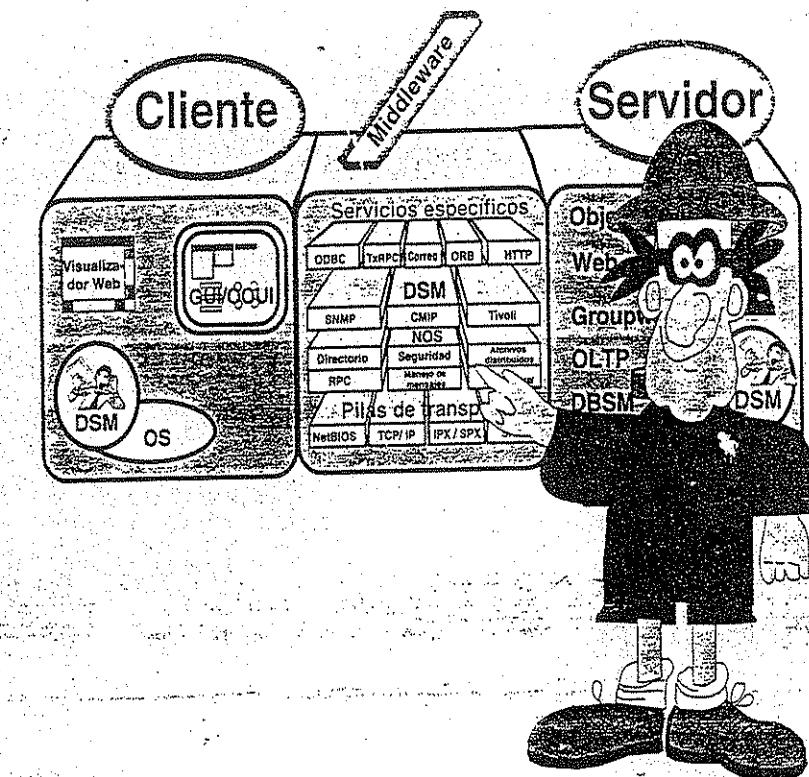
La Tercera parte trata del middleware base con el que se crea la "ilusión de sistema único". Comenzaremos por una breve exposición sobre el middleware de *sistema operativo para redes* (*NOS: network operating system*). Conocerá el "baúl de trucos" que usa el NOS para forjar ilusiones que opacarían al mismísimo Gran Houdini, uno de nuestros mejores magos, quien pasó por la Tierra no hace mucho tiempo. En la época de Houdini no había NOS.

Después de estudiar los NOS, nos ocuparemos del middleware de *pilas*, las cuales cuentan con su propio repertorio de trucos. Al fin y al cabo, nada es lo que parece. Todo se desarma, rearma y vuelve a empaquetarse para que "parezca funcionar" con todo lo demás. Pero usted se enterará de la verdad, por supuesto.

Descubrirá muy pronto que cada distribuidor querrá venderle un juego diferente de productos de middleware. Pero, lamentablemente, el middleware no ha logrado que su acto de desaparición vuelva transparentes a sus productos. Esto se debe a que el middleware es en sí mismo un negocio muy lucrativo. Y las cosas no desaparecen cuando se puede obtener dinero de ellas. Así pues, tenemos que pagar por experimentar la placentera sensación de que este mundo heterogéneo es un sistema felizmente integrado. ¿Acaso en la Tierra no se paga por recibir ilusiones? Piense nada más en la industria cinematográfica.

Finalmente, le presentaremos algunos de los productos que necesitará para crear las ilusiones que satisfagan su fantasía (o necesidades reales). Podrá crear casi cualquier clase de fachada si tiene el dinero para hacerlo. Otra importante razón de esta revisión de productos es que gracias a ella se enterará de aquello de lo que realmente dispone. Como todos los magos, los proveedores de middleware suelen olvidar qué es real y qué no lo es. Así, resulta esencial que nos

Introducción a la Tercera parte



introduzcamos tras bastidores y veamos qué es lo que realmente hay ahí, al menos en lo que se refiere a productos. Esperamos que disfrute el show.

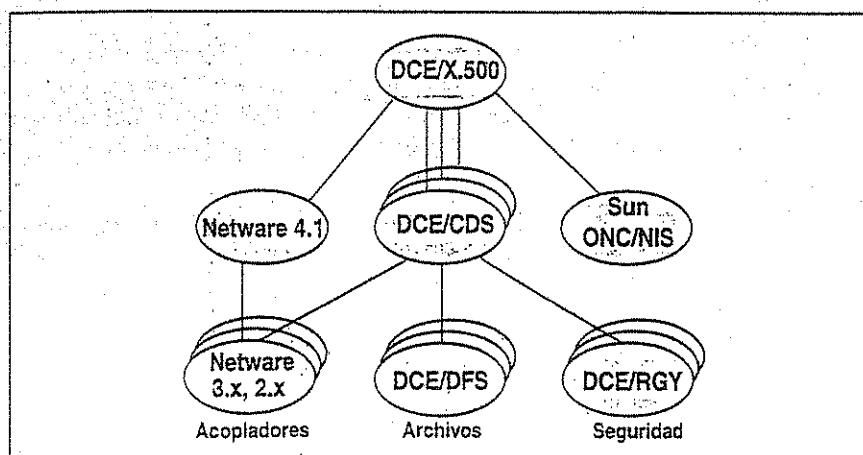
MIDDLEWARE DE NOS: LA ILUSIÓN TRANSPARENTE

Tras haber surgido como un conjunto de estaciones de trabajo independientes, capaces de comunicarse a través de un sistema de archivos compartidos, los NOS se han convertido en entornos reales de computación distribuida por intermedio de los cuales la red se vuelve *transparente* para los usuarios.

¿Qué significa transparencia?

Transparencia significa hacerle creer a la gente que el sistema de cliente/servidor no lleva costuras. Significa en realidad ocultarles a los usuarios, e incluso a los programadores de aplicaciones, tanto la red como sus servidores. He aquí algunos de los tipos de transparencias que se esperan del NOS como parte de su "acto de desaparición de la red":

- **Transparencia de ubicación:** Usted debe desconocer la ubicación de un recurso. Los usuarios no deben verse obligados a incluir la información de ubicación en el nombre del recurso. Por ejemplo, en \Máquina\directorio\archivo se revela el nombre de la máquina servidora. Ésta es una violación a la transparencia.
- **Transparencia de espacio para nombres:** Usted debe estar en posibilidad de emplear las mismas convenciones de nombramiento (y espacio para nombres) para localizar cualquier recurso en la red. Este universo es un solo e inmenso árbol (véase Figura 7-1), el cual incluye todo tipo de recursos de los productos de cualquier proveedor.
- **Transparencia de acceso:** Usted debe disponer de la facilidad de dar una sola contraseña (o autenticación) que funcione para todos los servidores y todos los servicios de la red.
- **Transparencia de duplicación:** A usted no debe interesarle cuántas copias existan de un recurso. Por ejemplo, si un directorio de nombres aparece sombreado en muchas máquinas,



es al NOS al que le corresponde sincronizar las actualizaciones y ocuparse de todos los asuntos de seguridad.

- **Transparencia de acceso local/remoto:** Usted debe estar en condiciones de trabajar con cualquier recurso de la red como si éste se encontrara en la máquina local. El NOS debe manejar los controles de acceso y proporcionar servicios de directorio.
- **Transparencia de tiempo distribuido:** Usted no debe percibir diferencias de horario entre servidores. El NOS debe sincronizar los relojes de todos los servidores.
- **Transparencia de fallas:** Usted debe estar protegido contra fallas de la red. El NOS debe manejar los reprocesamientos y reconexiones de sesión. También debe ofrecer ciertos niveles de redundancia de servicios para la tolerancia de fallas.
- **Transparencia de administración:** Usted sólo debe vérselas con una interfaz de administración de sistema único. El NOS debe estar integrado a los servicios de administración locales.

El desafío para el middleware de NOS es cómo brindar este alto nivel de transparencia *sin sacrificar la autonomía del OS local*.

NOS: Extensión del alcance del OS local

Una de las funciones de un NOS es volver transparente para una aplicación la ubicación física de los recursos (a lo largo de una red). Los primeros NOS fueron diseñados para virtualizar los recursos de archivos e impresoras y redireccionarlos a servidores de archivos e impresoras basados en LAN. Estos NOS disponían de agentes en las máquinas locales —los *solicitantes*— que interceptaban las llamadas a dispositivos y las *redireccionaban* a los servidores de la LAN. La única forma en que una aplicación (o usuario) podía advertir la diferencia entre un recurso local o remoto era a partir de un nombre de vía, que incluía el nombre de la máquina. Sin embargo, podía echarse mano de alias para ocultarles a los usuarios incluso los nombres de las vías. De este modo, el NOS extiende transparentemente el soporte de dispositivos del OS local a lo largo de la red. Prácticamente todo lo que puede hacerse en un OS local también se puede hacer en forma remota y transparente. El NOS permite que aplicaciones generadas para el OS local se conviertan en aplicaciones en red sin modificar una sola línea de código. La mayoría de los NOS permiten que clientes que operan con OS diferentes (como DOS, Mac y Unix) comparten archivos y otros dispositivos. Por ejemplo, un cliente Mac ve archivos DOS en formato Mac.

Una nueva generación de servidores de archivos de redes introducirá presumiblemente aún mayor transparencia en los sistemas de archivos. Por ejemplo, el *servicio distribuido de archivos (DFS: distributed file service)* de DCE ofrece un sistema de archivos de imagen única que puede distribuirse entre un grupo de servidores de archivos (véase Figura 7-2). El esquema de nombramiento de archivos de DFS es independiente de ubicaciones. Cada archivo posee un identificador único, consistente a todo lo largo de la red. Los archivos emplean el espacio para nombres global de DCE como el resto de los recursos de la red. Además, el sistema de archivos está integrado con los mecanismos de seguridad de DCE.

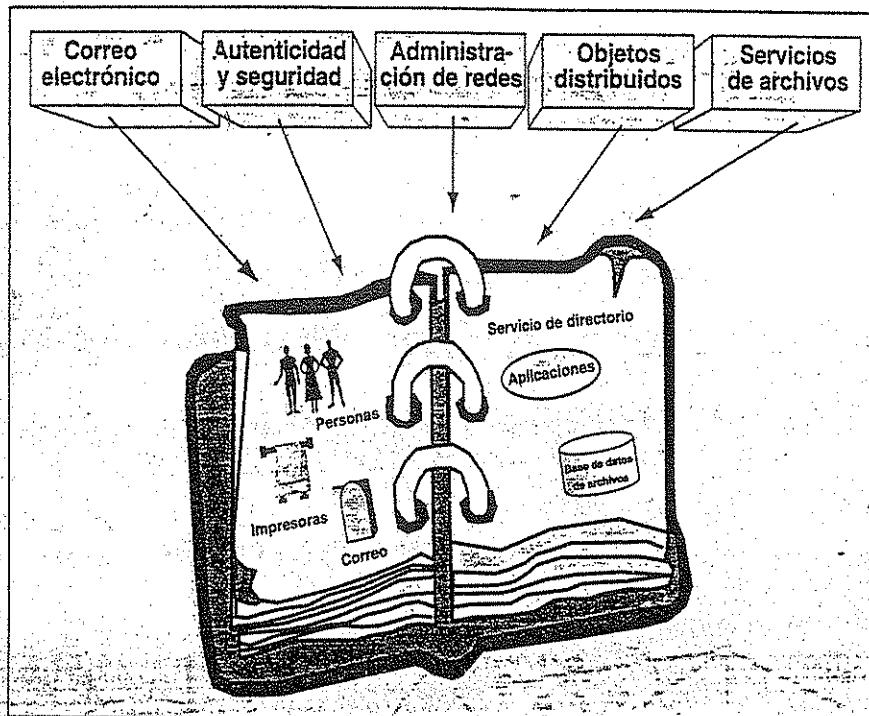


Figura 7-3. Directorios globales: registro de los recursos del NOS.

jerárquicos como en un sistema de archivos (véase Figura 7-4). En cada nombre hay un *componente global* y un *componente local*. El componente global es el nombre con que se conoce al directorio local en el nivel intergaláctico. El componente global administra una federación de directorios locales de acoplamiento holgado. Usted puede nombrar al componente local de acuerdo con convenciones locales. Además, un agente de gateway puede residir en cada directorio local y enviar consultas de nombres no locales a un directorio (o servicio de nombramiento) global.

¿Cómo se duplican los directorios? Por lo general, un directorio mantiene una copia maestra y duplicados sombra exclusivos para lectura. Para la regeneración de los duplicados se emplean dos tipos de esquemas de sincronización:

- La *duplicación inmediata* provoca que toda actualización de la copia maestra sea inmediatamente sombreada en todos los duplicados.
- La *actualización periódica* provoca una propagación periódica (una vez al día, por ejemplo) de todos los cambios realizados en la copia maestra a todos los duplicados.

En síntesis, la nueva generación de servicios de directorio del NOS usa parte de la tecnología de bases de datos y objetos distribuidos más avanzada para el rastreo de los recursos del sistema distribuido. La tecnología es tan flexible que maneja y registra las toscas entidades actuales de

red —como impresoras, usuarios, programas y servidores— y las entidades de grano más fino que comienzan a aparecer —como los objetos distribuidos.

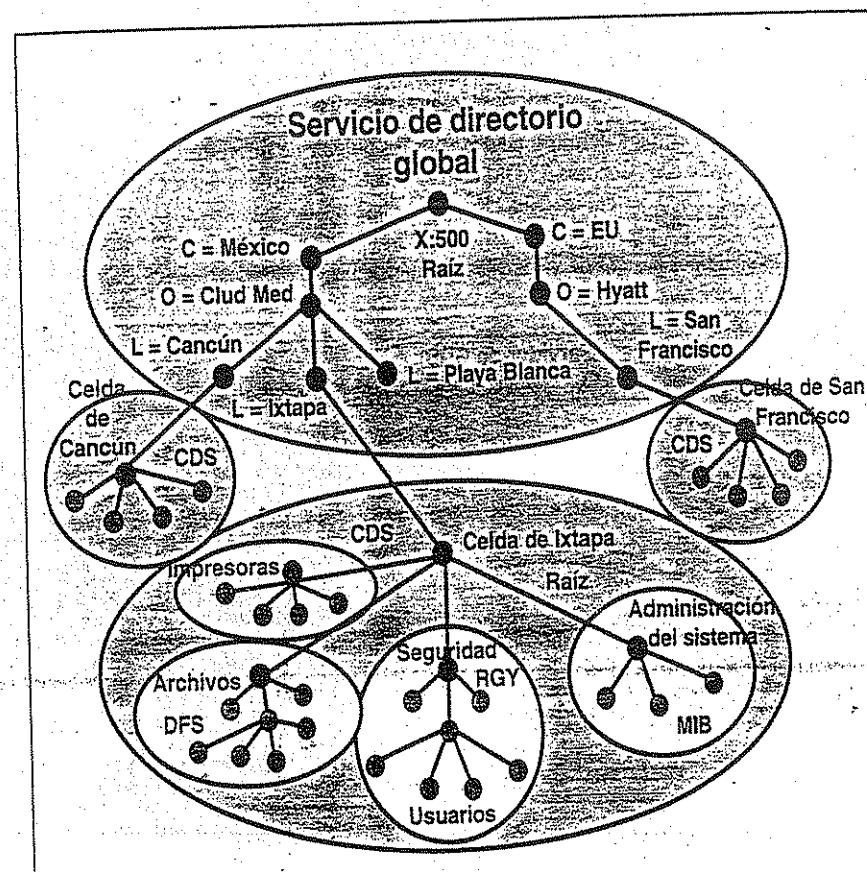


Figura 7-4. Directorios en federación: creación de un espacio para nombres unificado.



información

El estándar de directorio global X.500

X.500 es el único existente capaz de convertirse en el esperanto del mundo de los directorios.

Tim Sloane, Aberdeen Group

Cuidado! Este recuadro está repleto de siglas.

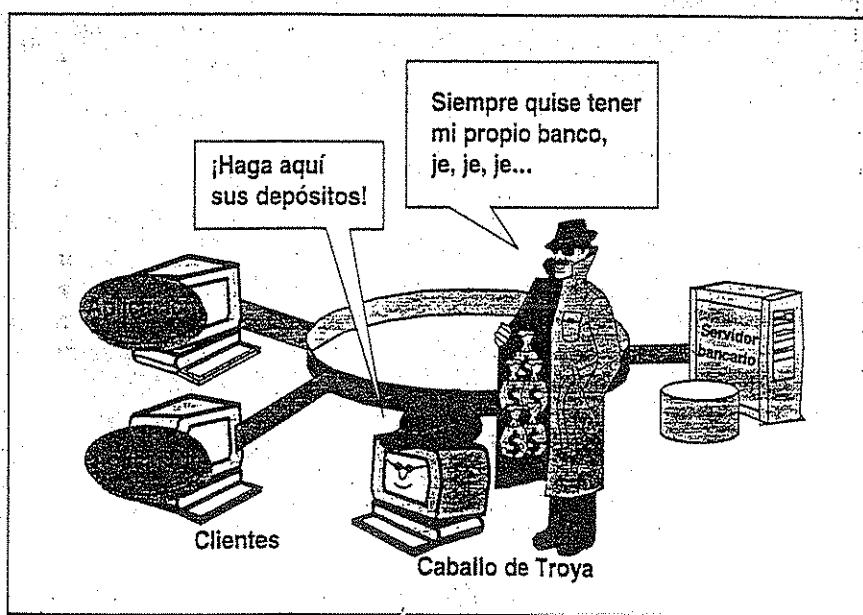


cada aplicación individual de cada servidor de la empresa? Veamos qué ofrecen los NOS al respecto.

¿Puede obtenerse seguridad de nivel C2 en la LAN?

C2 es un estándar de seguridad del gobierno estadounidense para sistemas operativos; implica la autenticación de usuarios y aplicaciones para que puedan tener acceso a los recursos de cualquier sistema operativo. Para obtener la certificación C2 en una red, todos los clientes deben disponer de un identificador de usuario autenticado, todos los recursos deben estar protegidos por listas de control de acceso, debe contarse con rastros para auditoría y debe impedirse la cesión de derechos de acceso a otros usuarios que reutilizan los mismos artículos. Conozcamos los mecanismos de seguridad que puede proporcionar un NOS moderno para cumplir (o incumplir también) la seguridad de nivel C2 en la red.

■ **Autenticación:** *¿Es usted quien dice ser?* En sistemas de tiempo compartido, la autenticación se realiza mediante contraseñas del OS. Pero en los NOS hace falta más. Cualquier intruso con una PC y equipo de registro de redes podría hacerse de una contraseña y reutilizarla. ¿Por qué entonces no codificar la contraseña? ¡Por favor! ¿Quién se encargaría de las claves secretas y todo lo que eso supone? Afortunadamente, los NOS tienen la respuesta: Kerberos. Kerberos es el tercero autorizado que permite que dos procesos se demuestren entre sí que son quienes dicen ser. Es como si dos espías se encontraran en una esquina y murmuraran las palabras mágicas que establecen una relación "confiable". Ambas partes obtienen por separado de Kerberos esas palabras mágicas (véase el siguiente recuadro de información).



FYI

Información

Kerberos: "No se puede confiar en nadie"

En el proyecto Athena del MIT se adoptó la postura de que es casi *imposible* garantizar la seguridad de cada una de las estaciones de trabajo de una red. Se supuso en cambio que un "personaje" podía presentarse en la LAN en cualquier momento, y se decidió obtener protección contra él. El resultado fue una fortaleza para software llamada Kerberos, que ofrece mayor nivel de seguridad que las contraseñas y listas de control de acceso tradicionales. Kerberos autentica automáticamente a todos los usuarios de todas las aplicaciones. Su protocolo, especialmente con las adiciones introducidas por el DCE de OSF, cumple los requisitos de autenticación de C2. Permite que los servidores confíen en sus clientes (en su mayoría PC) y viceversa. No olvide que es posible insertar caballos de Troya en la parte del servidor, así que también los servidores deben demostrar su identidad. □

■ **Autorización:** *¿Se le ha permitido usar este recurso?* Una vez autenticados los clientes, las aplicaciones servidor son responsables de verificar cuáles operaciones se les ha permitido realizar a los clientes con la información a la que pretenden acceder (por ejemplo, un servidor de nóminas puede controlar el acceso a información sobre salarios según el individuo de que se trate). Los servidores emplean *listas de control de acceso* (ACL: *access control list*) para controlar el acceso de los usuarios. Las ACL pueden asociarse con cualquier recurso de cómputo. Contienen la lista de nombres (y nombres de grupos) y tipo de operaciones que les está permitido efectuar en cada recurso. Los servicios de administración de NetWare, por ejemplo, permiten que los administradores de la red añadan nuevos usuarios a grupos sin tener que especificar derechos primarios de acceso. Los NOS cumplen fácilmente los requerimientos de ACL de C2.

■ **Rastros de auditoría:** *Usted, ¿dónde ha estado?* Los servicios de auditoría permiten que los administradores de redes vigilen las actividades de los usuarios, incluidos sus intentos de acceso y los servidores o archivos utilizados. Forman parte del arsenal que necesitan los administradores de redes para detectar intrusos en sus organizaciones. Por ejemplo, pueden vigilar todas las actividades de la red relacionadas con una estación de trabajo (o usuario) cliente sospechoso. Saber de la existencia de rastros de auditoría suele desalentar la indebida manipulación de servidores por parte de miembros de la red con el uso de su propio registro de acceso, aunque cabe la posibilidad de que la realicen con un registro ajeno. La mayoría de los NOS soportan rastros de auditoría, lo que debería ser motivo de satisfacción para los acreditadores de C2.

En síntesis, todo indica que la seguridad C2 de las LAN está al alcance de los NOS modernos.



plea una cifra de codificación basada en DES. En cuanto a llaves, usa un método de llave privada compartida específica para cada sesión.

- El *método de llave pública* se sirve de dos llaves: una *pública* y una *privada*. La llave pública puede aparecer en directorios y está a la vista de todo el mundo. Usted codifica su mensaje con su llave privada, y el receptor emplea la llave pública de usted para decodificarlo. Además, cualquier persona puede enviarle un mensaje cifrado con la llave pública de usted (que decodificaría con su llave privada). De este modo, se ignora incluso quién es el emisor. También en este caso el secreto es condición indispensable para el adecuado funcionamiento de su llave secreta (véase Figura 7-7).

RSA es un algoritmo de llave pública inventado por el MIT; sus siglas son las iniciales de sus tres inventores. Se le considera el mejor algoritmo de llave pública y su uso es prioritariamente de autenticación, aunque también de codificación de mensajes muy breves. Su problema es la excesiva lentitud para codificar mensajes largos, para lo cual requiere de DES. Si la codificación se realiza en software, DES es unas 100 veces más veloz que RSA. En hardware es de 1000 a 10 000 veces más veloz, dependiendo de la implementación (Fuente: Stang y Moore, Network Security Secrets, IDC Books, 1993).

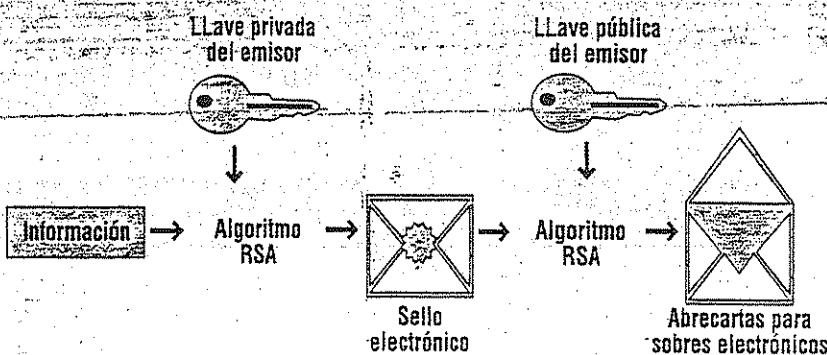


Figura 7-7. Llaves públicas: el sobre digital sellado.

¿A qué se debe entonces la popularidad de RSA? A que permite codificar mensajes sin el previo intercambio de secretos (o señales) y ofrece una señal electrónica infalsificable. Sólo el propietario conoce la llave privada; en este caso no hay "tercero autorizado" como Kerberos. Una señal de RSA puede tener validez legal siempre y cuando carezca de huellas digitales. Si se pierde control de la llave privada, el único responsable es el propietario. Uno de los problemas de RSA es que la parte de llave privada debe entregarse a cada nodo de la red sin quebrantos para la seguridad. Se necesitaría algo semejante a Kerberos para hacerlo. Una opción es el empleo de una *autoridad de certificación* (CA: certificate authority) que autentique la llave pública. La CA firma digitalmente la llave pública (tema sobre el que abundaremos en la Octava parte). □

Capítulo 8

RPC, manejo de mensajes de igual a igual

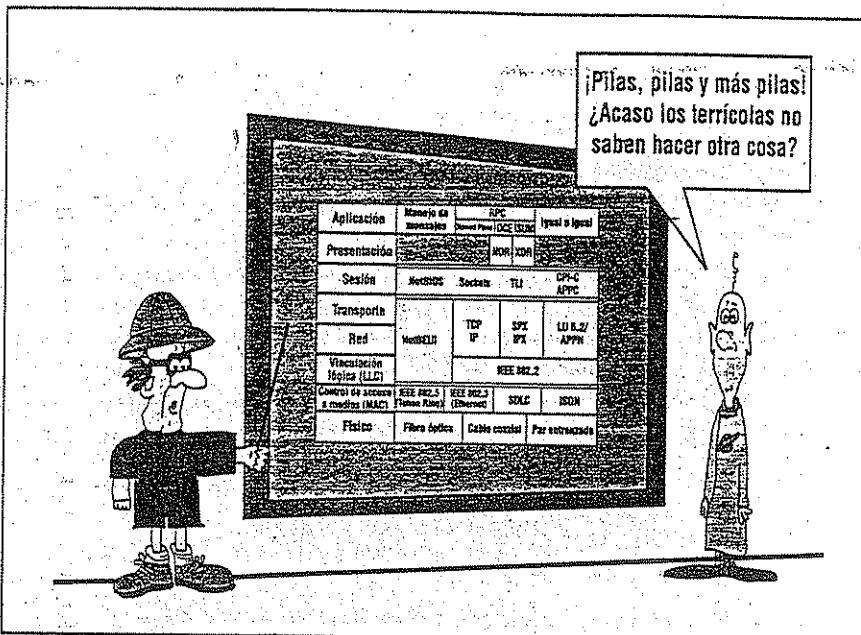


Tocad así las dulces flautas.

Keats

Las aplicaciones cliente/servidor se distribuyen a lo largo de espacios para direcciones, aparatos físicos, redes y sistemas operativos. ¿Cómo se comunican entre sí clientes y servidores? ¿Cómo se sincronizan solicitudes y respuestas? ¿Cómo se manejan representaciones de datos disímiles en diferentes computadoras? ¿Qué ocurre si una de las partes no está disponible? Adivinó usted: el NOS moderno asume muchas de estas responsabilidades. Forman parte de su territorio. El propósito del NOS es volver transparente la computación distribuida. Esto significa que debe crear un entorno que oculte el fastidio de tratar con protocolos, redes y pilas de comunicaciones.

Todos los NOS ofrecen interfaces de *igual a igual* que permiten que las aplicaciones se comuniquen mediante una semántica de emisión/recepción semejante a la telegráfica. Casi todos ellos cuentan con alguna modalidad de *middleware de llamada a procedimiento remoto* (RPC: *remote procedure call*), la cual oculta el "alambre" para dar la apariencia de que todos los servidores de la red son una función por llamar. Un modelo alternativo —las colas de mensajes, o *middleware orientado a mensajes* (MOM: *message-oriented middleware*)— está ganando adeptos. Resulta que el manejo de mensajes es increíblemente útil en situaciones en las que no se desea que clientes y servidores estén estrechamente sincronizados. Los NOS actuales no incluyen MOM entre sus características. Sin embargo, compañías especializadas en este campo ofrecen pro-



en el sentido de que no le oculta por completo al programador la red subyacente. Por ejemplo, la interfaz revelará interrupciones de transmisión, condiciones de carrera y errores de red, que dejará en manos del programador. Los protocolos de igual a igual comenzaron como API de pilas específicas. Pero como se explicó en la Primera parte, la mayoría de estas API soportan ahora múltiples pilas. Así, su asociación con una pila en particular ya es sólo de interés histórico. He aquí una breve descripción de los principales protocolos de igual a igual y sus pilas asociadas.

Sockets

Los sockets aparecieron en 1981 como la interfaz genérica de Unix BSD 4.2 que ofrecería comunicaciones de Unix a Unix en redes. En 1985, el OS de Sun introdujo NFS y RPC sobre sockets. En 1986, AT&T lanzó la *interfaz de capa de transporte (TLI: transport layer interface)*, con funcionalidad similar a la de los sockets pero con mayor independencia de la red. Unix SVR4 integra tanto sockets como TLI. Los sockets son por ahora mucho más prevalentes que TLI. Ambos son muy parecidos desde la perspectiva de un programador. TLI es simplemente una versión más sencilla de los sockets. En teoría, una aplicación generada para TLI es independiente de las pilas. Debería correr en IPX/SPX o TCP/IP con muy pocas modificaciones. La API de TLI se compone de alrededor de 25 llamadas de API.

Prácticamente todos los sistemas operativos soportan sockets. La API de sockets de Windows, conocida coloquialmente como *WinSock*, es una especificación de proveedor múltiple que estandariza el uso de TCP/IP en Windows. La API WinSock se basa en la interfaz de sockets de

Berkeley. En el sistema BSD de Unix, los sockets forman parte del núcleo y ofrecen un servicio de IPC tanto independiente como en red. Los sistemas no BSD de Unix, MS-DOS, Windows, OS de Mac y OS/2 cuentan con sockets en forma de bibliotecas. Puede decirse entonces que los sockets constituyen el actual estándar portátil *de facto* para los proveedores de aplicaciones de redes en redes TCP/IP.

Los tres tipos más comunes de sockets son los de flujo, de datagramas y en bruto. Los sockets de flujo y de datagramas sirven de interfaz con los protocolos TCP y UDP, mientras que los sockets en bruto son interfaz con el protocolo IP. El tipo de socket se especifica al momento de su creación. En teoría, la interfaz del socket puede ampliarse, y es posible definir nuevos tipos de sockets para disponer de servicios adicionales. Una dirección de socket en la Internet TCP/IP se compone de dos partes: una dirección Internet (*IP_address*) y un número de puerto (véase Figura 8-2). ¿Qué es una dirección Internet? ¿Qué es un número de puerto?

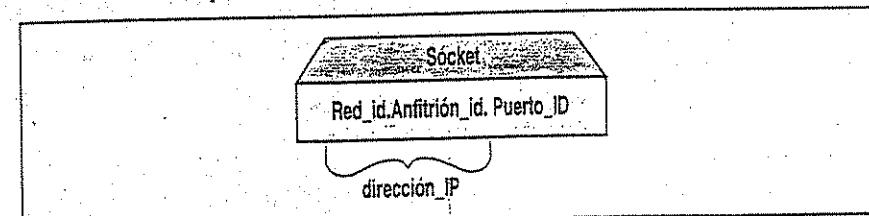


Figura 8-2. Socket = dirección Internet (IP) + dirección de puerto.

Una *dirección Internet* es un número de 32 bits, usualmente representado por cuatro números decimales separados por puntos, los cuales deben ser únicos para cada tarjeta de interfaz de la red TCP/IP dentro de un dominio AF_INET administrado. Un *anfitrión* de TCP/IP (es decir, una máquina en red) puede tener tantas direcciones Internet como interfaces de red.

Un *puerto* es un punto de entrada a una aplicación que reside en un anfitrión. Está representado por un número entero de 16 bits. Los puertos se usan comúnmente para definir los puntos de entrada de servicios provistos por aplicaciones servidores. Los programas comerciales de servidores más importantes —como los DBMS de Oracle y Sybase— tienen sus propios puertos, los cuales son *reconocidos*.



Información

Datagramas contra sesiones

Los protocolos orientados a conexiones —también conocidos como *protocolos basados en sesiones, circuitos virtuales o intercambios secuenciales de paquetes*— ofrecen un servicio de conexión bidireccional confiable en una sesión. A cada paquete de información intercambiado en una sesión se le asigna un número de secuencia único mediante el cual se le rastrea y reconoce individualmente. Los paquetes duplicados son detectados y descartados por los servicios de sesión.



con IPX/SPX como con TCP/IP.¹ IBM y Microsoft emplean a NetBIOS como interfaz tanto con TCP/IP como con NetBEUI. Así pues, sea cauteloso, especialmente en la lectura de nuestros libros; se nos ha dicho que empleamos NetBIOS y NetBEUI indistintamente.

NetBEUI ofrece potentes servicios de datagramas y orientados a conexiones, lo mismo que un dinámico servicio de nombramiento basado en protocolos de descubrimiento. Su principal defecto es que carece de capa de red. Otro, su carencia de seguridad. El mecanismo de difusión, usado para el "descubrimiento" dinámico de nombres, puede ser una desventaja en una red insegura, en la que no es buena idea exponer nombres. La difusión de nombres también causa tráfico indeseable en la red. Afortunadamente, la mayoría de los puentes y enrutadores tienen manera de filtrar los paquetes de descubrimiento e impedir su propagación a otras redes.

Los servicios de NetBIOS son provistos mediante un conjunto de comandos, especificados en una estructura llamada *bloque de control de red* (NCB: *network control block*). Esta estructura también contiene los parámetros asociados con el comando y los campos en los que NetBIOS devolverá información al programa. Un comando puede ser emitido ya sea en modo de espera o no espera. En el modo de *espera*, el hilo de solicitud es bloqueado hasta que el comando se completa. En el modo de *no espera*, el control es devuelto al hilo de llamada lo más pronto posible, por lo general antes de que se complete el comando. Una vez que esto ocurre, la DLL de NetBIOS coloca un código de retorno en el NCB.

Named Pipes

Named Pipes ofrece comunicaciones bidireccionales altamente confiables entre clientes y un servidor. Brinda una API de programación semejante a archivos que abstrae un intercambio de datos bidireccional basado en sesión. Con Named Pipes, los procesos pueden intercambiar datos como si generaran o leyeren un archivo secuencial. Es especialmente conveniente para implementar programas de servidor que requieren de conductos muchos a uno. Una aplicación servidor puede instalar un conducto donde su extremo receptor intercambie datos con varios procesos del cliente. Named Pipes se encarga de todos los asuntos de creación de itinerarios y sincronización.

Uno de los más importantes beneficios de Named Pipes —al menos para los programadores de cliente/servidor en Windows, OS/2 y NT— es que forma parte del servicio base de comunicaciones entre procesos. La interfaz de Named Pipes es idéntica independientemente de que los procesos sean ejecutados en una máquina individual o se distribuyan en una red. Named Pipes corre en pilas NetBIOS, IPX/SPX y TCP/IP. Se le incluye entre las características de enlace en red de Windows NT, Windows para Trabajo en grupo, Windows 95 y Warp Server. El soporte de Unix para Named Pipes es provisto por LAN Manager/X.

¹ Quizá la confusión se originó por la denominación NetBEUI de Microsoft, la cual significa interfaz del usuario extendida NetBIOS (*NetBEUI: NetBIOS extended user interface*).



La nueva SNA: APPC, APPN y CPI-C

IBM está convirtiendo la *arquitectura de redes de sistemas* (SNA: *system network architecture*) en un verdadero sistema operativo distribuido capaz de soportar servicios de directorio entre redes, acceso de red transparente a recursos (como servidores, aplicaciones, pantallas, impresoras y datos), flujos de datos comunes y administración de red integrada. La *red de igual a igual avanzada* (APPN: *advanced peer-to-peer network*) es la infraestructura de red responsable de esta "verdadera distribución". APPN crea una red de SNA sin la jerarquía centrada en macrocomputadoras de las configuraciones tradicionales de SNA. La macrocomputadora es simplemente uno más de los nodos de la red. APPN permite que aplicaciones LU 6.2 de SNA, que usan API de APPC o CPI-C, se beneficien plenamente de redes de iguales. Asimismo, simplifica enormemente la configuración de SNA, ofrece mayor disponibilidad a través de enruteamiento dinámico, facilita el mantenimiento de redes de SNA y satisface los requerimientos de flexibilidad de las redes modernas.

La *interfaz común de programación para comunicaciones* (CPI-C: *common programming interface for communications*) se instala sobre APPC y oculta sus complejidades e irregularidades (véase Figura 8-3). Cada producto que soporta APPC tiene una API ligeramente diferente. CPI-C resuelve el problema. Remitirse a la API de CPI-C le permite exportar sus programas a todas las plataformas de SNA. La API de CPI-C consiste en alrededor de 40 llamadas; APPC se compone de más de 60. La mayoría de estas llamadas se refieren a configuración y servicios.

El consorcio X/Open obtuvo de IBM la licencia de la interfaz de CPI-C, como lo han hecho también otras compañías (Novell y Apple entre ellas). Además de estar presente en *todas* las plataformas de IBM, también Insession (para Tandem Computers), System Strategies (para Unix), Rabbit Software (para DOS) y DCA (para DOS y OS/2) cuentan con API de CPI-C.

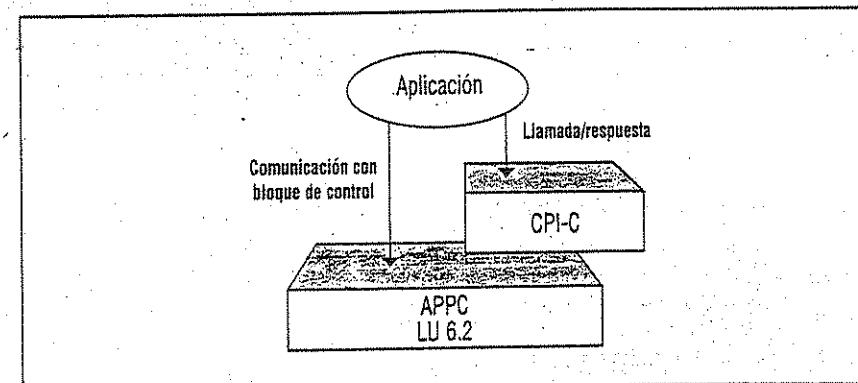
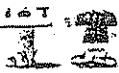


Figura 8-3. CPI-C sobre APPC.



- **¿Cómo maneja RPC la seguridad?** Los NOS modernos —como DCE— facilitan la incorporación automática de sus características de seguridad en la RPC. Todo lo que usted tiene que especificar es el nivel de seguridad requerido (autenticación, codificación, etc.); luego, la RPC y la característica de seguridad cooperarán para hacerlo posible.
- **¿Cómo encuentra el cliente a su servidor?** A la asociación de un cliente con un servidor se le conoce como acoplamiento. La información de acoplamiento puede ser de codificación dura en el cliente (algunos servicios, por ejemplo, son realizados por servidores con direcciones *conocidas*). O bien, un cliente puede encontrar a su servidor consultando un archivo de configuración o un parámetro del entorno. Un cliente también puede hallar a su servidor en tiempo de ejecución a través de los servicios de directorio de la red. Desde luego que los servidores deben anunciar sus servicios en el directorio. Al proceso de empleo del directorio para encontrar a un servidor en tiempo de ejecución se le llama *acoplamiento dinámico*. La manera más fácil de hallar un servidor es permitir que la RPC lo haga por usted. Esto se llama *acoplamiento automático*, lo que significa que el talón del cliente de RPC localizará a un servidor en una lista de servidores que soporte la interfaz.
- **¿Cómo se maneja la representación de datos entre sistemas?** El problema en este caso es que diferentes CPU representan estructuras de datos en forma distinta (*big-endian* contra *little-endian*, por ejemplo). ¿Cómo se logra la transparencia de datos en el nivel de RPC? Para mantener independencia de la máquina, la RPC debe proporcionar cierto nivel de traducción de formatos de datos entre sistemas. Por ejemplo, la RPC de Sun requiere que los clientes conviertan sus datos a un formato canónico neutral mediante las API de *representación de datos externos* (XDR: *external data representation*).

En contraste, el servicio de *representación de datos de red* (NDR: *network data representation*) de DCE es multicanónico, lo que quiere decir que soporta múltiples representaciones de formatos de datos. El cliente elige uno de estos formatos (su propia representación de datos nativa en la mayoría de los casos), etiqueta los datos con el formato elegido y delega en el servidor la transformación de los datos a un formato que éste comprenda. En otras palabras, *el servidor es el que se encarga del asunto*. DCE parte del supuesto de que en la mayoría de los casos el cliente y el servidor emplearán la misma representación de datos, de manera que carece de sentido incurrir en la carga de la traducción. Sun supone a su vez que las MIP del cliente son de bajo costo, así que deja en el cliente la responsabilidad de realizar la traducción, lo que le facilita las cosas al servidor. Con Sun, todos los clientes son iguales para el servidor: es *el cliente el que se encarga del asunto*.

En la Figura 8-5 se muestra la manera en que el mecanismo de RPC reúne todos los elementos. El escenario corresponde a una aplicación simple de reserva de asientos. El servidor de asientos es el primero en ponerse en marcha, anuncia su ubicación y servicio en el directorio de la red y emprende su ciclo permanente de recepción y atención de solicitudes. Un cliente de boletos conserva en su caché la ubicación del servidor. Cuando un cliente está listo para comprar un boleto para un concierto de Madonna, se emite una RPC para reservar un asiento. Adviértase que los talones del cliente y el servidor cooperan para que tal cosa ocurra. Cuesta mucho trabajo hacer una reserva para un concierto de Madonna. Las RPC eliminan algunas de esas dificultades.

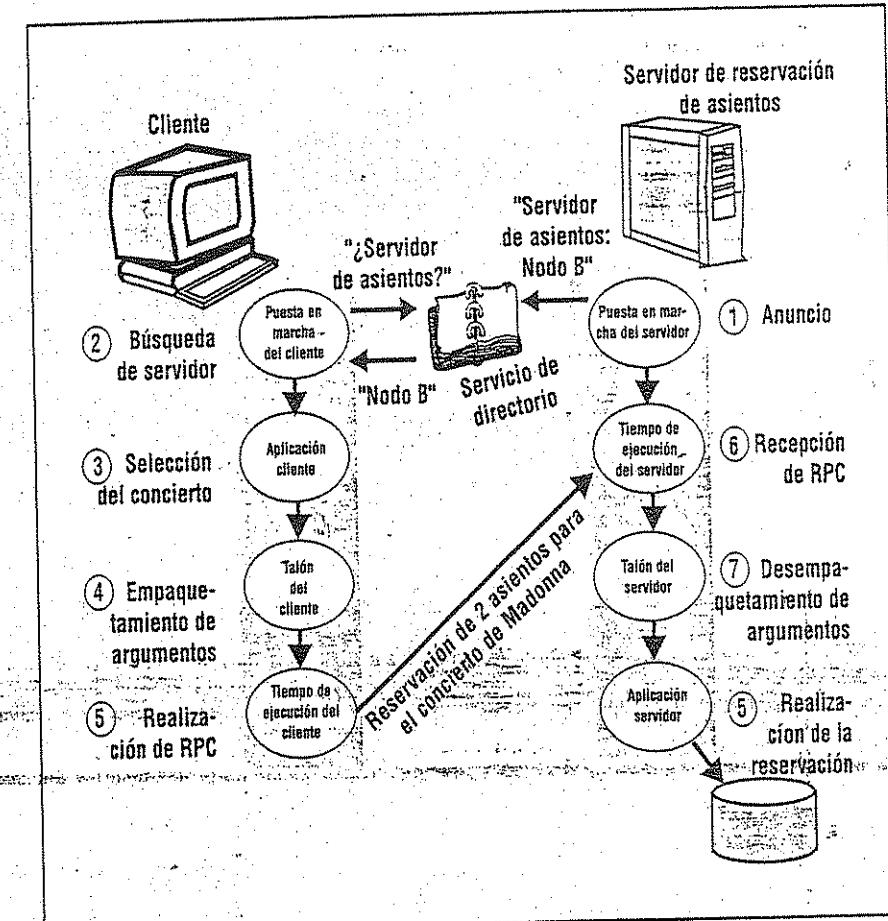


Figura 8-5. Obtención de un asiento para un concierto de Madonna mediante RPC.

MANEJO DE MENSAJES Y COLAS: EL MIDDLEWARE DE MOM

Todo papá (DAD) necesita una mamá (MOM).

The Message-Oriented Middleware (MOM) Consortium

"Todo papá (DAD) necesita una mamá (MOM)" es el lema extraoficial del MOM Consortium. En este contexto, DAD significa *desarrollo de aplicaciones distribuidas* (*distributed application development*) y MOM, *middleware orientado a mensajes* (*message-oriented middleware*).



La mayoría de los productos de manejo de mensajes de MOM ofrecen un conjunto de API simple que corre en múltiples plataformas de sistemas operativos. Casi todos ellos brindan también colas de mensajes *persistentes* (registrados en el disco) y *no persistentes* (en memoria). Los mensajes persistentes son más lentos, pero pueden recuperarse en caso de fallas de energía eléctrica tras la reiniciación de un sistema. En ambos casos, los mensajes pueden ser ya sea copiados o eliminados de una cola. Una cola de mensajes puede ser *local* para la máquina o *remota*. Los administradores de sistemas pueden especificar por lo general el número de mensajes que una cola está en condiciones de alojar y el tamaño máximo de los mensajes.

Los productos de manejo de mensajes proporcionan en su mayoría un nivel mínimo de tolerancia de fallas bajo la forma de colas persistentes. Algunos de ellos cuentan con alguna modalidad de *protección para transacciones*, lo que hace posible que la cola participe en un protocolo de sincronización de grabación en dos fases. Asimismo, algunos pueden incluso redirigir mensajes a colas alternas en caso de una falla en la red.

MOM CONTRA RPC

Comparar los paradigmas de RPC y manejo de mensajes equivale a comparar una llamada telefónica con el intercambio de cartas o faxes para hacer negocios (véase Figura 8-9). La interacción a través de una llamada telefónica es inmediata; ambas partes se comunican directamente para resolver sus asuntos. Al término de la conversación telefónica concluye una unidad de trabajo. En contraste con ello, hacer negocios a través del correo permite periodizar el trabajo, priorizarlo y llevarlo a cabo cuando se está listo para emprenderlo. Usted, y no el tintineante teléfono, controla el flujo de trabajo. Por otro lado, para la parte del cliente puede resultar frustrante no recibir retroalimentación inmediata.

En la Tabla 8-1 se comparan las arquitecturas de manejo de mensajes y RPC. El primero es, por supuesto, más flexible, de acoplamiento más holgado y más tolerante al tiempo que RPC. Sin embargo, sólo aplaza temporalmente las cosas y puede generar su propio nivel de complicaciones. En la analogía telefónica (RPC), usted lleva a su término el trabajo en cuanto lo recibe; no tiene que ocuparse de montones de cartas (o faxes) recibidas. A sus clientes les satisface obtener servicio inmediato. Cuando cierra su negocio al final del día, ha cumplido con la totalidad de sus actividades. En la analogía del correo, se corre el riesgo de que las cartas empiecen a apilarse, y de que los clientes revisen una y otra vez sus buzones a la espera de la respuesta. Quizá le facilitemos las cosas al servidor a expensas del cliente. Por otra parte, el manejo de mensajes exime a los clientes de la necesidad de sincronizarse con sus servidores; esto puede ser muy liberador para usuarios móviles y domésticos.

El manejo de mensajes alienta un modelo de comunicaciones sujeto a eventos. Usted puede enviar múltiples solicitudes a múltiples servidores, y después aceptar las respuestas conforme llegan. Su aplicación no bloquea la espera de respuestas. En cambio, éstas son tratadas como eventos. Sencillamente ocurren. Por supuesto que usted debe estar preparado para enfrentarlas al momento en que ocurran. Las RPC pueden imitar este tipo de comportamiento asíncrono, de acoplamiento holgado y sujeto a eventos mediante hilos.

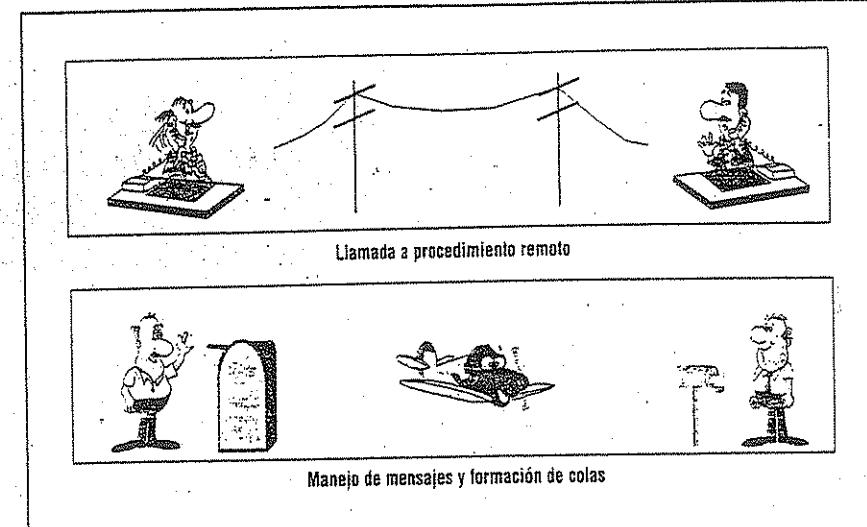


Figura 8-9. MOM contra RPC: ¿prefiere el correo o el teléfono?

Tabla 8-1. Comparación entre MOM y RPC.

Característica	MOM: Manejo de mensajes y formación de colas	Llamada a procedimiento remoto (RPC)
Alegoría	Semejante al correo.	Semejante al teléfono.
Relación temporal de cliente/servidor	Asíncrona. Clientes y servidores pueden operar a diferentes tiempos y velocidades.	Síncrona. Clientes y servidores deben operar concurrentemente. Los servidores deben marchar al paso de los clientes.
Secuencia de cliente/servidor	Sin secuencia fija.	Los servidores deben ponerse en marcha primero para que los clientes puedan comunicarse con ellos.
Estilo	Colas.	Llamada-respuesta.
El socio debe estar disponible	No.	Sí.
Equilibrio de cargas	Puede emplearse una sola cola para implementar atención de acuerdo con el orden de recepción o con una política basada en prioridades.	Se requiere de un monitor de TP independiente.
Soporte de transacciones	Sí (algunos productos). La cola de mensajes puede participar en la sincronización de grabaciones.	No. Se requiere de una RPC para transacciones.
Filtración de mensajes	Sí.	No.

Capítulo 9

NOS: Presentación de participantes



Elegir un NOS puede parecer cuestión de suerte y misticismo, como cuando se entra a un casino. El recién llegado a NOS debe aprender las reglas de cada producto para redes y descubrir qué proveedor requiere de pruebas previas, cuál juego es el que probablemente ofrecerá los mejores resultados y cuál se adapta mejor a su situación.

PC Magazine

Quizá le sorprenda saberlo, pero en la segunda parte ya conoció usted a casi todos los participantes en el campo del NOS. Esto se debe a que los sistemas operativos del servidor ya han asumido la mayoría de las funciones que los NOS independientes solían ofrecer. Los OS del servidor se han convertido en NOS. Los NOS más importantes son *NetWare 4.1* de Novell, *NT Server 4.0* de Microsoft y *OS/2 Warp Server* de IBM. Adicionalmente, el mundo de Unix tiene dos NOS entre los cuales elegir: *ONC+* de Sun y *DCE* de OSF. Curiosamente, ninguno de estos NOS ofrece funciones de MOM. Así, tiene que seguir comprándolas "a la carta" con distribuidores de MOM.

En este capítulo nos ocuparemos primeramente de la tecnología y tendencias de mercado del NOS. Despues le presentaremos brevemente a los principales participantes. Finalmente, trataremos con cierto detalle el *Entorno de Computación Distribuida* (*DCE: Distributed Computing Environment*) de OSF. El DCE es importante porque representa la solución de NOS más com-

DCE: EL NOS POSMODERNO

DCE es un gran sistema diseñado para resolver grandes problemas.

Sandy Rockowitz, Minaret Software
(Abril de 1996)

El *Entorno de Computación Distribuida* (DCE: *Distributed Computing Environment*) de la Open Software Foundation (OSF) —y ahora de X/Open— crea un entorno abierto de NOS que comprende múltiples arquitecturas, protocolos y sistemas operativos. El consorcio de estándares de X/Open está incluyendo las especificaciones de DCE en la versión 4 de su *Guía de portabilidad de X/Open (XPG: X/Open Portability Guide)*. La importancia de DCE es que casi todos los proveedores de software planean soportarlo de una forma u otra. DCE ofrece tecnologías distribuidas clave, como una llamada a procedimiento remoto, un servicio de nombramiento distribuido, un servicio de sincronización de tiempo, un sistema de archivos distribuidos, un servicio de seguridad de redes y un paquete de hilos (véase Figura 9-2). Estas seis importantes tecnologías serán incorporadas prácticamente a todos los sistemas operativos. En las secciones siguientes se presenta un breve resumen de los principales componentes de DCE, incluidos sus orígenes.

DCE permite que un cliente interopere con uno o más procesos del servidor en otras plataformas de computación, aun cuando sean de proveedores diferentes con distintos sistemas operativos. Además, cuenta con un método integrado de seguridad, nombramiento y comunicaciones entre procesos. Todas estas piezas son útiles para crear un coherente entorno heterogéneo de cliente/servidor. DCE es el mejor ejemplo arquitectónico de NOS posmoderno intergaláctico. Su intención es convertirse en la "madre de todos los NOS".

RPC DE DCE

La RPC de DCE provino originalmente de HP; se trata de una adaptación de la RPC de Apollo. DCE cuenta con un *lenguaje de definición de interfaces (IDL: interface definition language)* y un compilador que facilitan la creación de RPC. El compilador de IDL crea talones de código C portátiles para las partes tanto de cliente como de servidor de una aplicación. Los talones se compilan y vinculan con la biblioteca de tiempo de ejecución de RPC, responsable de la búsqueda de servidores en un sistema distribuido, la realización de intercambios de mensajes, el empaquetamiento y desempaquetamiento de parámetros de mensajes y el procesamiento de todos los errores que puedan ocurrir.

La característica más sugerente de la RPC de DCE es que puede integrarse a los servicios de seguridad y nombramiento de DCE. Esta integración hace posible autenticar cada llamada a procedimiento y localizar dinámicamente servidores en tiempo de ejecución. Los servidores pueden atender concurrentemente RPC mediante el empleo de hilos. El mecanismo de RPC ofrece independencia de protocolos y redes.

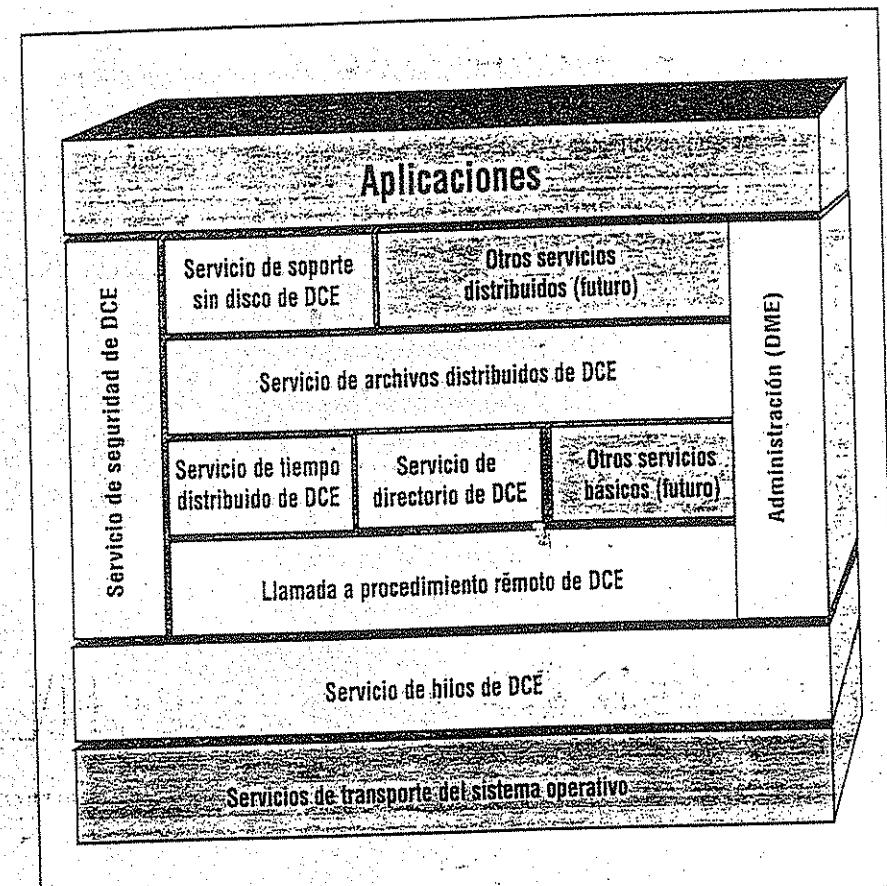


Figura 9-2. Componentes de DCE.

DCE: SERVICIOS DE NOMBRAMIENTO DISTRIBUIDO

OSF adoptó los servicios de nombramiento distribuido para DCE del producto *DECdns* de Digital y los servicios *DIR-X X.500* de Siemens. Los servicios de nombramiento de DCE permiten que recursos tales como programas, servidores, archivos, discos o colas de impresoras sean identificados por nombres orientados al usuario en una base de datos distribuida de propósito especial que describa los objetos de interés. Los nombres de los objetos son independientes de su ubicación en la red.

DCE divide el entorno distribuido en unidades (o dominios) administrativas llamadas *celdas*. Una celda de DCE es una combinación de estaciones de trabajo del cliente y el servidor. El dominio de la celda es definido por el cliente. Una celda suele consistir en el conjunto de máquinas utilizado por uno o más grupos que trabajan en tareas relacionadas. El tamaño



DCE: SERVICIOS DE SEGURIDAD DISTRIBUIDA

Kerbero: Perro de tres cabezas que resguardaba las puertas del hades.

Mitología griega

OSF adoptó el sistema de autenticación Kerberos del MIT y lo complementó con algunas características de seguridad de HP. Protocolo que habría sido del gusto de Maquiavelo, Kerberos se basa en la desconfianza mutua total. Sus inventores del MIT lo llamaron así por el monstruo mitológico de tres cabezas que protegía las puertas del hades (sí, el infierno). ¿A causa de qué alguien quería quebrantar las puertas del infierno? Dejemos que sean los administradores de seguridad quienes respondan esta pregunta. Kerberos posee en efecto tres "cabezas", las cuales residen en el mismo servidor de seguridad: el *servidor de autenticación*, la *base de datos de seguridad* y el *servidor de privilegios*. El tricefálico monstruo del MIT les concede a los administradores de redes la extrema seguridad que necesitan para evitar que "piratas intrusos" se cuelen por las puertas de su red.

Los servicios de seguridad de redes de DCE proporcionan autenticación, autorización y administración de cuentas del usuario. La *autenticación* comprueba que un cliente, por lo general un usuario o programa, es quien dice ser. Esta comprobación se efectúa mediante la capacidad de comunicaciones seguras provista por la RPC y el mecanismo de boletoaje de Kerberos. Cada máquina del DCE debe contar con un agente de seguridad.

El *servidor de seguridad* de DCE es un servidor físicamente protegido que almacena información de seguridad como nombres y contraseñas asociadas. Cada celda de DCE debe disponer de un servidor de seguridad, por lo general una máquina especial. Una celda puede contar con servidores de seguridad duplicados para efectos de respaldo. La *facilidad de acceso* de DCE permite que los usuarios (llamados en DCE *principales* o *unidades autorizadas*) establezcan su



identidad autenticándose mediante una contraseña. El sistema de seguridad de DCE nunca envía por la red una contraseña en "texto simple".

El agente de seguridad colabora con la RPC autenticada para garantizar acceso protegido a todos los servicios de DCE, no sólo a la autenticación. El mecanismo de RPC oculta al usuario la complejidad del sistema de seguridad: Obtiene los boletos, ofrece codificación en caso necesario y realiza sumas de verificación autenticadas si la política así lo requiere. En forma oculta, tanto el cliente como el servidor de RPC deben autenticarse mutuamente mediante el intercambio de boletos (secretos) a través de un tercero autorizado (el servidor de Kerberos). Cada parte confía en que el servidor de Kerberos identificará a la otra en la red. Esto se llama *codificación de llave secreta de tercero autorizado*. Además, en la información se incluye un registro de hora para que la vigencia de un boleto expire en un periodo relativamente corto, medido en horas.

La *autorización* es posterior a la autenticación; determina si el cliente autenticado tiene permiso de acceder a un recurso. DCE soporta la autorización mediante *listas de control de acceso* (ACL: *access control list*). Cada implementación de DCE que emplea ACL debe implementar un administrador de ACL que controle el acceso a los servicios y recursos que administra (DCE cuenta con un código de muestra en el que se indica cómo crear un administrador de ACL). La *integridad de los datos* es provista por DCE con sumas de verificación de datos criptográficas para determinar si un mensaje fue corrompido o alterado al pasar por la red. Adicionalmente, la *privacidad de los datos* puede asegurarse codificando aquellos que son transferidos por la red.



información

DCE y listas de control de acceso

Los NOS modernos como DCE de OSF cuentan con una serie de API que les permiten a los servidores crear y administrar sus ACL. El NOS de DCE también ofrece ganchos para que los clientes presenten a las aplicaciones servidores sus credenciales de autorización. En DCE éstas se llaman *certificados de atributos de privilegio* (PAC: *privilege attribute certificate*), boletos emitidos por el servidor de seguridad que el cliente debe presentar al servidor. Los PAC contienen información de autorización específica del cliente, como su grupo designado. DCE dispone de una serie de API del servidor capaces de leer la información contenida en los PAC y de compararla con la información en las ACL. □

En síntesis, DCE resuelve los problemas asociados con la autenticación de usuarios en redes distribuidas. Las contraseñas nunca se envían en "texto simple". La base de datos de seguridad (llamada *registro* en DCE) puede ser propagada a través de servidores autorizados. Además de

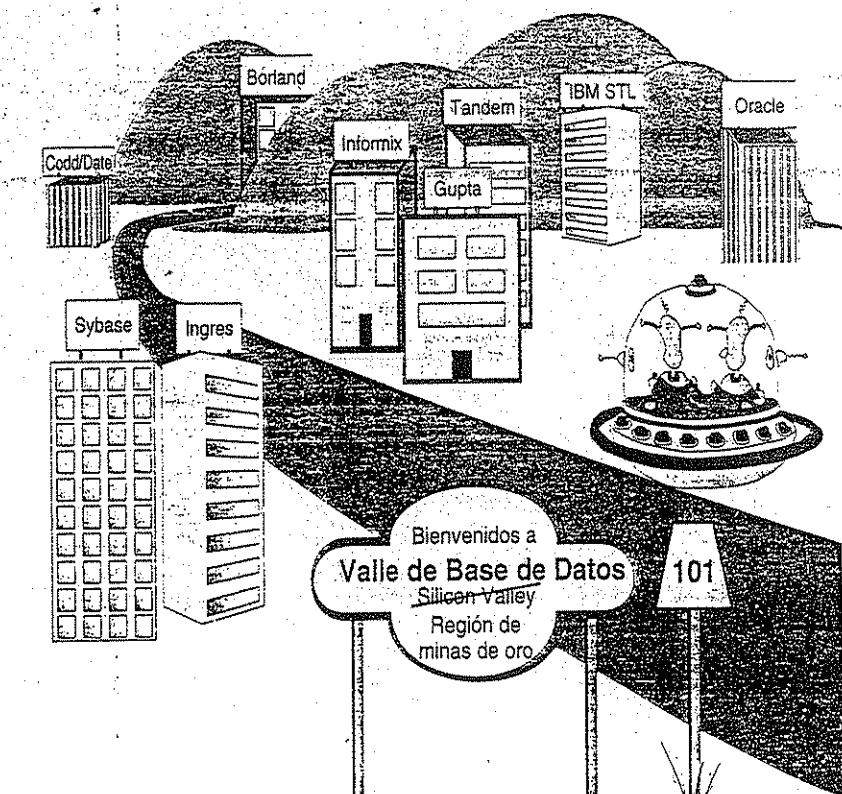


CONCLUSIÓN

Los NOS intergalácticos están abriendo nuevas y espléndidas posibilidades a los desarrolladores de aplicaciones cliente/servidor. No hay límites a las combinaciones de sistemas que pueden formarse sobre estos sustratos de NOS distribuido. Pero lo mejor aún está por llegar. Los objetos distribuidos son el NOS por antonomasia. Extiende el NOS distribuido a todo lo largo de la esfera de las aplicaciones, y después lo unirán a Internet (tema sobre el que abundaremos en las séptima y octava partes). Triunfarán en el juego del desarrollo de software quienes aprendan a utilizar la enorme capacidad del NOS y la empleen en la creación de nuevos e interesantes paquetes de software.

Parte 4

Servidores de bases de datos de SQL



Capítulo 10

Servidores de bases de datos de SQL



La mayor parte del actual software de cliente/servidor pertenece al área de bases de datos, y en ella radica el mayor reto para cualquier empresa.

Richard Finkelstein, presidente
Performance Computing

Este capítulo trata de las bases de datos de SQL desde la perspectiva de cliente/servidor. Los servidores de SQL son el modelo dominante para la creación de aplicaciones cliente/servidor. Los proveedores de servicios de SQL —como Oracle, Sybase, Informix, Ingres y Gupta— se han vuelto famosos en todas partes. ¿Por qué es tan común SQL desde la perspectiva de la conectividad cliente/servidor? ¿Las bases de datos relacionales podrán dar la batalla contra modelos más recientes de computación de cliente/servidor, entre ellos las bases de datos de objetos, los corredores de solicitudes de objetos, el Web y groupware? ¿Son realmente necesarios los monitores de TP o tenemos suficiente con los procedimientos almacenados provistos por los distribuidores de bases de datos? En este capítulo le ofreceremos una visión condensada del estado de cosas en la computación de cliente/servidor centrada en bases de datos. Con ello sentaremos las bases que nos permitirán responder estas preguntas en secciones posteriores de este libro.

Nuestro plan para este capítulo es abordar primero la magia de SQL y el modelo relacional desde la perspectiva de cliente/servidor. Expondremos lo referente a los estándares, incluidos SQL-89,



■ **SQL contribuye a la protección de los datos en entornos de red de usuarios múltiples.** Lo hace gracias a que cuenta con excelentes características de confiabilidad, como validación de datos, integridad de referencias, retrocesos (anulación de transacciones), candados automáticos y detección y resolución de bloqueo en entornos de LAN multiusuarios. También se encarga de la seguridad y control de acceso a objetos de bases de datos.

SQL ofrece un buen número de ventajas a los creadores de sistemas, ya que sirve lo mismo para definir una base de datos que para manipularla. El lenguaje SQL facilita la especificación precisa de requerimientos de productos. Esto favorece las comunicaciones entre clientes, desarrolladores y administradores de bases de datos (DBA: *database administrator*).

Las normas ISO: SQL-89, SQL-92 y SQL3

Aunque a partir de 1979 han aparecido numerosas implementaciones comerciales de SQL, no se contó con un estándar oficial al respecto hasta 1986, año en que el Instituto Estadounidense de Normas Nacionales (ANSI: *American National Standards Institute*) y la Organización de Normas Internacionales (ISO: *International Standards Organization*) emitieron uno en forma conjunta. Modificado en 1989 para introducir en él integridad de referencia (y restricciones de verificación), el estándar de 1986 se conoce hoy como *SQL-89*, o SQL de ANSI. A fines de 1989 se añadió a SQL-89 un agregado exclusivo de ANSI para SQL incrustado (*Embedded SQL*).

SQL-89

El estándar SQL-89 fue una "intersección" de las implementaciones de SQL de entonces, lo que facilitó su adopción por los productos existentes. SQL-89 era un SQL "diluido", de manera que el término "observante de SQL" prácticamente perdió significado. Los proveedores (Gupta, Oracle y XDB, por ejemplo) tendrían que añadir por lo general a su lista de verificación de acatamientos los referidos a DB2. Aun así, esto tampoco significaba gran cosa, al menos en términos de la creación de un SQL unificado.

SQL-92

El SQL-92 de ISO (también llamado SQL2), ratificado a fines de 1992, tiene una extensión más de cinco veces mayor al estándar original SQL-89. En SQL-92 se estandarizan muchas de las características anteriormente confiadas a la discreción del implementador (es decir, las lagunas), de modo que consiste en esencia en una superserie de SQL-89. C. J. Date estima que lograr que las bases de datos relacionales actuales se adecuen a los estándares de SQL-92 implicará un enorme esfuerzo de implementación. Para resolver este problema, la propia ISO ha sugerido un avance por etapas con tres niveles de cumplimiento: inicial, intermedio y total. Para facilitar la identificación de la etapa por la que se atraviesa, el estándar SQL-92 introduce el concepto de *banderillero*, programa que examina el código fuente y "marca con banderas" todas las instrucciones SQL que no respondan a SQL-92.



¿Qué es lo nuevo en SQL-92?

Curiosamente, en el estándar no aparece una sola vez el término "relación". Además, el término "base de datos" sólo se usa informalmente (pues se le sustituyó formalmente por "datos de SQL"). ¿Es éste un estándar de bases de datos relacionales?

C. J. Date
(Mayo de 1993)

Esta sección contiene un breve resumen de las novedades incluidas en SQL-92 dirigido a lectores con conocimientos sobre SQL y el anterior estándar SQL-89. Si usted no está familiarizado con SQL, no lea este recuadro sin antes concluir la lectura de este capítulo.

El estándar SQL-89 anterior fundamenta el lenguaje de definición de datos (DDL: *data definition language*) de SQL para la creación de tablas, índices, vistas y restricciones de integridad de referencias. Fundamenta asimismo los privilegios de seguridad CONCESSION/REVOCACION (*GRANT/REVOKE*). El lenguaje de manipulación de datos (DML: *data manipulation language*) de SQL-89 está integrado por los comandos SELECT, INSERT, UPDATE y DELETE. COMMIT y ROLLBACK se emplean en la administración de transacciones. Un mecanismo de cursor permite la navegación por cada fila. La adición del SQL incrustado de SQL-89 define el mecanismo para la incrustación de instrucciones SQL en FORTRAN, COBOL, PL/I y Pascal.

El "nuevo" estándar SQL-92 se adhiere a todas las características del SQL-89 y añade las siguientes:

- **Agentes de SQL:** se les define como programas o usuarios interactivos que producen instrucciones SQL. En el estándar anterior, las instrucciones SQL se asociaban con identificadores de autorización (un concepto ambiguo).
- **Conexiones de cliente/servidor de SQL:** antes de ejecutar cualquier operación de bases de datos, un agente de SQL debe solicitar al código de cliente de SQL para conectarse con CONNECT a con algún servidor de SQL. Una conexión establece una sesión de SQL. SQL-92 aprueba las conexiones (o sesiones) concurrentes, pero sólo una de ellas puede estar activa en un momento dado. Los agentes pueden conmutar explícitamente entre conexiones mediante el comando SET CONNECTION.
- **Más controles de transacción granulares:** mediante el empleo del comando SET TRANSACTION podemos especificar una transacción como exclusiva para lectura (*read-only*) o como de lectura/escritura (*read/write*). Una transacción exclusiva para lectura no puede modificar el estado de la base de datos. Además, podemos fijar el *nivel de aislamiento* (es decir, el nivel de protección automática de candados) para que una transacción dada sea de lectura sin grabación (*read-uncommitted*), lectura con grabación (*read-committed*), lectura y repetición (*read-repeable*) o serializable.

lación (*undo*), repetición (*redo*) y grabación (*commit*). También la especificación de PSM está mereciendo trato de vía rápida. No se sorprenda de que se haya convertido en una norma ISO a fines de 1996. Abordaremos el tema de los procedimientos almacenados en una sección posterior de este mismo capítulo.

- En la Quinta parte, *SQL/Acoplamientos*, se define la mecánica para la combinación de SQL con otros lenguajes a través de precompiladores y SQL incrustado. Hablaremos de estos asuntos en el siguiente capítulo.
- En la Sexta parte, *SQL/Transacciones*, se define el modo de participación de las bases de datos de SQL en transacciones globales. La comisión encargada de SQL3 está acelerando la adopción de SQL/Transacciones por medio del cumplimiento del estándar XA de X/Open de ISO, con apenas unas cuantas modificaciones. Trataremos de XA en la Quinta parte.
- En la Séptima parte, *SQL/Temporal*, se define el modo de empleo de datos de series de tiempo por las bases de datos de SQL. La idea es que éstas modelen el tiempo, para que sea posible hacer consultas con el tiempo como variable. Un ejemplo de consulta sería: "¿Cuáles fueron las ventas de Q4 en 1990?" La comisión de SQL3 se ocuparía de estandarizar los trabajos realizados por Richard Snodgrass, investigador de bases de datos temporales de la Universidad de Arizona. Por desgracia, los participantes de Inglaterra no estuvieron de acuerdo. Así, no retenga la respiración a la espera de un estándar.

El borrador de SQL3 también contiene sugerencias de mejoras a SQL, como cursos persistentes (o "retenidos") que permanezcan abiertos después de una grabación, nuevos tipos de uniones, vistas temporales, privilegios de columnas específicas y una mejor definición del modo de actualización de vistas. Aborda asimismo temas peculiares, como puntos de sincronización sobre sesiones, subtablas y supertablas y ejecución asíncrona de instrucciones SQL.

SQL3 podría incluir de igual forma especificaciones de SQL para multimedia, llamado *SQL/MM*. El grupo de ISO comisionado para analizar las implicaciones de datos de "texto íntegro" para SQL amplió su cometido para incluir también el tema, más general, de los datos de multimedia, incluidos texto íntegro, audio digitalizado, videoclips, datos espaciales y sísmicos y otras modalidades de estructuras de datos de la realidad. SQL/MM empleará tipos abstractos de datos para definir las operaciones soportadas por cada tipo de objetos de multimedia. A diferencia de los BLOB actuales, los tipos abstractos de datos ofrecen métodos para la manipulación de cada uno de los tipos de datos de multimedia. Proporcionar el almacenamiento es la parte sencilla; lo difícil es proveer los métodos y campos de datos específicos de multimedia que nos permitan hacer cosas significativas con esos BLOB (como rotarlos o jugar con ellos). Tal como lo expresó Jim Melton, uno de los principales estrategas de SQL3, "BLOB y objetos son animales muy diferentes".

SQL3 añade muchas nuevas características a un ya profuso estándar SQL-92. ISO calcula que SQL3 será ratificado en su totalidad en julio de 1998 (y que podría disponerse de un estándar preliminar en febrero de 1997). Sólo para que se dé una idea del asunto, la conversión en estándar del primer borrador de SQL-92, el cual apareció en el curso de 1989, tardó tres años. Sin embargo, es importante contar con información del momento sobre lo que se propone para el estándar SQL3 a fin de deducir la dirección que está siguiendo SQL (véase Figura 10-1).

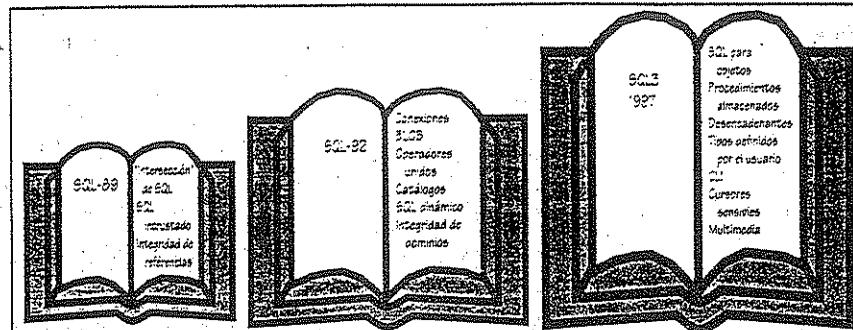


Figura 10-1. Evolución de la especificación de SQL.

¿QUÉ HACE UN SERVIDOR DE BASES DE DATOS?

La diferenciación de productos de DBMS ha dado como resultado que ningún DBMS sea el mejor en todas las categorías de funciones.

Meta Group
(Noviembre de 1995)

En una arquitectura cliente/servidor centrada en bases de datos, por lo general una aplicación cliente solicita datos y servicios relacionados con datos (como clasificación y filtración) a un servidor de bases de datos. El servidor de bases de datos, también llamado mecanismo de SQL, responde a las solicitudes del cliente y ofrece acceso protegido a datos compartidos. Con una sola instrucción SQL, una aplicación cliente puede recuperar y modificar un conjunto de registros de la base de datos del servidor. El mecanismo de bases de datos de SQL puede filtrar los conjuntos de resultados de la consulta, lo que deriva en considerables ahorros de comunicación de datos.

Un servidor de SQL administra el control y ejecución de comandos de SQL. Proporciona las vistas lógica y física de los datos y genera planes de acceso optimizado para la ejecución de los comandos de SQL. Además, la mayoría de los servidores de bases de datos proporcionan características y utilerías de administración del servidor que facilitan el manejo de los datos. Un servidor de bases de datos también mantiene tablas dinámicas de catálogos que contienen información sobre los objetos de SQL alojados en él.

Puesto que un servidor de SQL permite que múltiples aplicaciones accedan a la misma base de datos al mismo tiempo, debe brindar un entorno que proteja a la base de datos contra una variedad de posibles amenazas internas y externas. El servidor administra los aspectos de recuperación, concurrencia, seguridad y consistencia de una base de datos. Esto supone controlar la ejecución de una transacción y anular sus efectos en caso de fallas. Implica asimismo la obtención y entrega de candados durante la ejecución de una transacción y la protección de los objetos de la base de datos contra acceso no autorizado.

La mayoría de los servidores de SQL ofrecen por lo menos funcionalidad de nivel SQL-92. La mayoría incluye también algunas características de SQL-92. Sólo unos cuantos ofrecen versiones propietarias de procedimientos almacenados, desencadenantes y reglas de SQL3. Algunos

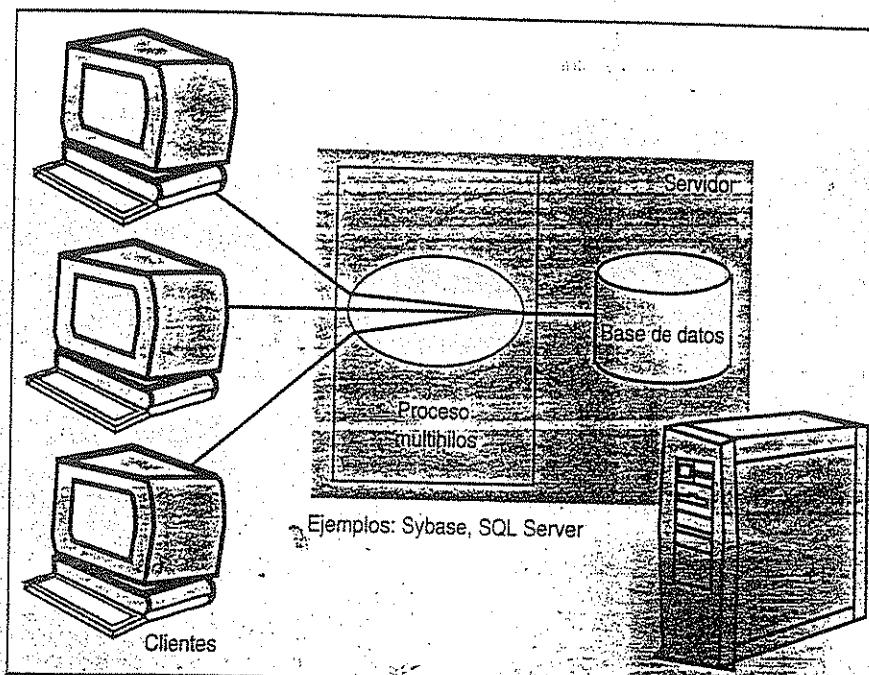


Figura 10-3. Arquitectura del servidor de base de datos de multihilos.

servidor compartidos y reutilizables que extraen el trabajo de la ejecución y colocan la respuesta en una cola de salida. La ventaja de esta arquitectura es que proporciona un entorno protegido para la ejecución de las tareas del usuario sin asignar a cada uno de éstos un proceso permanente. Las desventajas son los estados latentes de la cola. Aunque aparentemente se trata de una arquitectura aceptable, su equilibrio de cargas no es tan bueno como el provisto por un monitor de TP. De hecho, las colas pueden interponerse en el camino de los propios algoritmos de creación de itinerarios del monitor de TP. El primer servidor de base de datos en implementar esta arquitectura es Oracle7. De acuerdo con Rich Finkelstein, puede esperarse desde un 20% de mejoras hasta un 20% de deterioros en desempeño entre Oracle V6 y Oracle7.

¿Cuál arquitectura es mejor para cliente/servidor? La elección es difícil. Las arquitecturas de proceso por cliente se desempeñan deficientemente cuando un gran número de usuarios se conecta con una base de datos, pero ofrecen la mejor protección. Las arquitecturas multihilos pueden soportar la realización de transacciones breves de gran cantidad de usuarios, pero no se desempeñan adecuadamente en caso de consultas extensas. Tampoco brindan protección "a prueba de balas". En teoría, las arquitecturas híbridas son muy prometedoras. No obstante, ¿superan a un monitor de TP con un servidor de base de datos de proceso por cliente? Por regla general, a estas arquitecturas no les importa gran cosa que usted realice una simple operación de apoyo de decisiones basado en LAN, pero sí les importa que planee crear un sistema de OLTP a prueba de balas. Si éste es su caso, le sugerimos comprobar cuidadosamente las referencias y perseguir el grado máximo de protección.

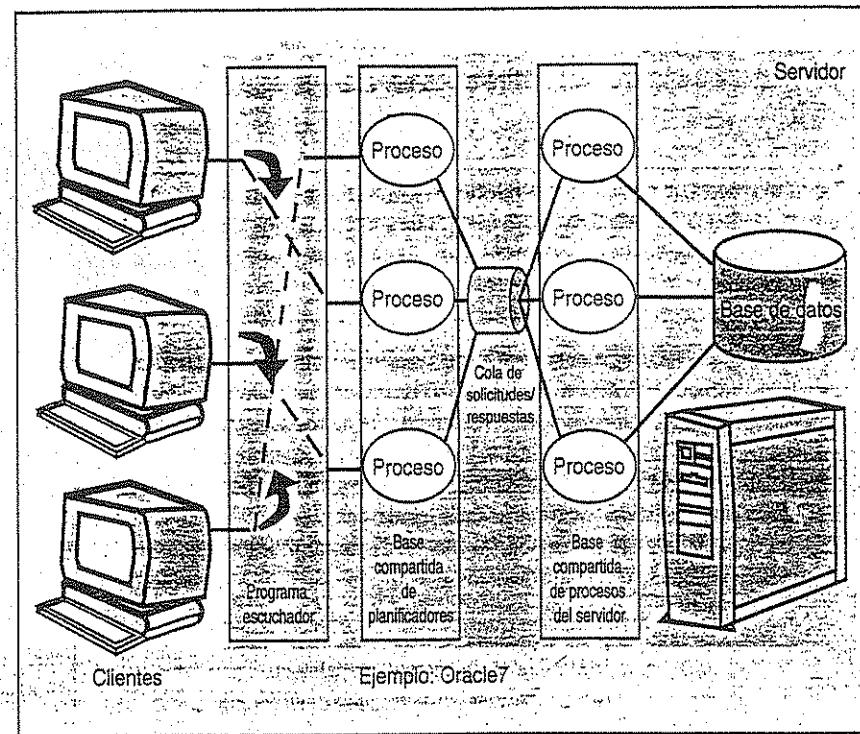


Figura 10-4. Arquitectura del servidor de base de datos híbrida.

PROCEDIMIENTOS ALMACENADOS, DESENCADENANTES Y REGLAS

Las bases de datos relacionales disponen ahora de extensiones de procedimientos integradas, como procedimientos almacenados, desencadenantes y reglas. Estas extensiones son muy útiles, pero absolutamente ajenas al estándar. ¿Por qué entonces los servidores de bases de datos están incursionando en el terreno de los procedimientos? ¿En qué consisten estas extensiones y qué nuevos servicios prestan? Responderemos la primera pregunta con un recuadro de debate. El contenido de esta sección intenta contestar la segunda.

¿Qué es un procedimiento almacenado?

Muchos proveedores de bases de datos ofrecen en la actualidad un mecanismo semejante a RPC para base de datos. Se le conoce como "TP ligero" o "procedimientos almacenados". Un procedimiento almacenado es un conjunto nombrado de instrucciones y lógica de procedimientos de SQL compilado, verificado y almacenado en la base de datos del servidor.



En otras palabras, un procedimiento almacenado es una entidad de SQL semejante a RPC, centrada en base de datos, persistente, compartida y con nombre. Reduce el tráfico en la red, mejora los tiempos de respuesta y ofrece un tipo de servicio de base de datos orientado a objetos apto para aplicaciones de OLTP. Los procedimientos almacenados brindan también mayor *autonomía local*, dado que la modificación remota de tablas sólo puede ocurrir mediante la ejecución local de programas. Si las tablas cambian, no es necesario recompilar todas las aplicaciones remotas. En general, los procedimientos almacenados realizan una mejor distribución de inteligencia que SQL remoto estático o dinámico.



Información

SQL estático y dinámico

Las instrucciones de *SQL estático* se definen en el código de usted y son convertidas en un plan de acceso en tiempo de preparación del programa. La instrucción SQL se conoce antes de que su programa se ponga en marcha. Los objetos de base de datos deben existir cuando se precompilan instrucciones de SQL estático. El SQL estático puede concebirse como una forma compilada del lenguaje SQL. Es una característica que favorece el desempeño.

Las instrucciones de *SQL dinámico* se crean y emiten en tiempo de ejecución. Ofrecen flexibilidad máxima a expensas de la velocidad de ejecución. El SQL dinámico puede concebirse como una forma interpretativa del lenguaje SQL. No es necesario que existan objetos de base de datos al precompilarse instrucciones de SQL dinámico. La compilación de instrucciones de SQL dinámico se realiza en tiempo de ejecución y debe repetirse cada vez que la misma instrucción sea ejecutada de nuevo.

El SQL estático se usa para la generación de programas de transacciones altamente optimizados. El SQL dinámico se usa para la generación de utilerías de programación general de bases de datos y en herramientas frontales o de primer plano de GUI que necesitan crear consultas específicas. □

Procedimientos almacenados contra SQL estático y dinámico

En la Tabla 10-2 se comparan las características funcionales de cliente/servidor de los procedimientos almacenados y otras modalidades de programación de SQL. Como puede verse, los procedimientos almacenados ofrecen muchas ventajas.

Tabla 10-2. Procedimientos almacenados contra SQL estático y dinámico.

Característica	Procedimiento almacenado		SQL remoto
	Estático incrustado	Dinámico	
Función nombrada	Sí	No	No
Función compartida	Sí	No	No
Persistentemente almacenado en el servidor	Sí	Sí	No
Parámetros de entrada/salida	Sí	No	No
Registrado en catálogo	Sí	Sí	No
Lógica de procedimientos	Dentro del objeto	Externa	Externa
Flexibilidad	Baja	Baja	Alta
Nivel de abstracción	Alto	Baja	Bajo
Estándar	No	Sí	Sí
Desempeño	Rápido	Medio	Lento
Propicio para herramientas	No	No	Sí
Propicio para paquetes comerciales de cliente/servidor	Sí (procedimiento de llamada)	No (confusión)	Sí (llamadas de CLI)
Mensajes en red	Una solicitud/respuesta para muchos comandos de SQL	Una solicitud/respuesta por comando de SQL	Una solicitud/respuesta por comando de SQL

¿Entonces, cuál es el problema de los procedimientos almacenados?

Uno de los inconvenientes de los procedimientos almacenados es que ofrecen menos flexibilidad específica que el SQL dinámico remoto. Además, pueden desempeñarse muy deficientemente si sus planes no son regenerados (reacoplados) para aprovechar las estadísticas del optimizador; el SQL dinámico crea un plan nuevo con cada ejecución. Otro inconveniente es que no hay sincronización para transacciones —es decir, grabación en dos fases— entre procedimientos almacenados; cada procedimiento almacenado es una transacción independiente.

Sin embargo, el principal defecto de los procedimientos almacenados es que son totalmente ajenos al estándar. Esto da como resultado varios problemas. No existen dos implementaciones de proveedores iguales. El lenguaje para la descripción de los procedimientos almacenados y su funcionalidad varía de un servidor a otro; los procedimientos almacenados no son exportables entre plataformas de proveedores. Se carece de un medio estándar para transmitir o describir los parámetros. Es difícil que las herramientas de bases de datos creen y administren procedimientos almacenados. Vérselas con los parámetros es muy complicado (no hay un lenguaje de definición de interfaces estándar ni herramientas para el compilador de talones).

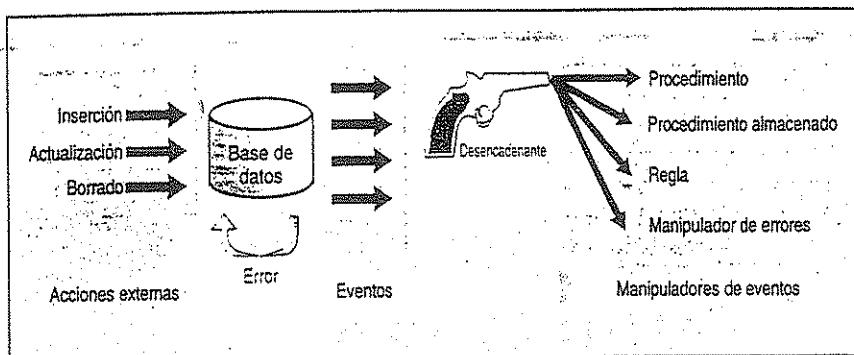


Figura 10-6. Mecánica de los desencadenantes de SQL.

Los desencadenantes y las reglas suelen usarse para el desarrollo de tareas relacionadas con cambios en tablas, como auditoría, búsqueda de umbrales de valores o establecimiento de columnas por omisión. Los desencadenantes o reglas habilitados son ejecutados cada vez que una tabla se actualiza por un comando DELETE, INSERT o UPDATE de SQL. Un desencadenante o regla distintos pueden ser definidos para cada uno de estos comandos, o bien puede definirse un sólo desencadenante para todas las actualizaciones de una tabla.

En general, los desencadenantes pueden llamar a otros desencadenantes o procedimientos almacenados. ¿Qué diferencia entonces a un desencadenante de un procedimiento almacenado? Los desencadenantes son llamados implícitamente por eventos generados por la base de datos, mientras que los procedimientos almacenados son llamados explícitamente por aplicaciones cliente. Las implementaciones de desencadenantes del servidor son extremadamente ajenas al estándar y específicas del proveedor. He aquí algunos ejemplos de diferencias en las implementaciones del proveedor:

- *Sybase* y *SQL Server* sólo soportan un desencadenante por operación INSERT/UPDATE/DELETE.
- *Ingres* soporta múltiples desencadenantes, pero la ejecución de éstos es no determinante.
- *Oracle7* soporta hasta 12 desencadenantes por tabla. Lo hace permitiéndole a usted especificar lo siguiente para cada INSERT/UPDATE/DELETE: un *desencadenante anterior*, que se acciona antes de que sea ejecutada la instrucción SQL, y un *desencadenante posterior*, que se acciona después de ejecutada la instrucción SQL. Además, Oracle permite especificar el número de veces que se activa un desencadenante. Los *desencadenantes de filas* se disparan una vez por cada fila actualizada; los *desencadenantes de instrucción* se disparan una vez para la totalidad de la instrucción SQL, incluso si no se insertan, actualizan ni borran filas. Ambos pueden definirse para ser activados simultáneamente. La implementación

de desencadenantes de Oracle7 se acerca al estándar preliminar SQL3 (aunque no lo cumple por completo).

- *Informix* soporta desencadenantes anteriores y posteriores y más de un desencadenante por operación: usa los números de columnas para determinar la secuencia de activación de desencadenante.
- *DB2 V2.1* —introducida en junio de 1995— soporta ahora desencadenantes en las versiones OS/2, AIX y NT. Se pueden definir múltiples desencadenantes para que sean ejecutados por cada INSERT/UPDATE/DELETE en una tabla. DB2 ejecuta los desencadenantes en el orden en que se crean. Un desencadenante de DB2 consiste en una o más funciones de INSERT, UPDATE y DELETE de SQL. Además, DB2 V2 introdujo un tipo especial de desencadenantes llamado *alerta*, que posee la capacidad de informar a una aplicación externa de un cambio de estado en una base de datos.
- *SQLBase 6.0* soporta ahora desencadenantes anteriores y posteriores. Se pueden ejecutar desencadenantes por fila o por comando. Además, SQLBase 6.0 introdujo un tipo especial de desencadenante llamado *evento*, que ejecuta un procedimiento en un momento especificado o a intervalos periódicos.

Los desencadenantes se generan en extensiones propietarias de procedimientos de SQL. Diferentes implementaciones limitan lo que los desencadenantes pueden hacer. Por ejemplo, con Oracle7 no se puede emitir grabaciones ni retrocesos desde un desencadenante; DB2 no permite generar código de procedimientos en un desencadenante ni llamar a un procedimiento almacenado. Los desencadenantes y reglas también son usados, en forma por demás *ajena al estándar*, por Sybase (antes del Sistema 10) y SQL Server para el cumplimiento de la integridad de referencias. Por ejemplo, un desencadenante asociado con una tabla en particular es invocado cuando los datos de la tabla se modifican o actualizan. Sin embargo, la integridad de referencias efectuada por desencadenantes tiene muchos inconvenientes, de manera que muy pocos servidores de base de datos la emplean (véase el siguiente recuadro de advertencia). En cambio, la mayoría de los servidores implementa el estándar de *integridad de referencias declarativa* (*declarative referential integrity*), definido en SQL-89. En suma, los desencadenantes son extremadamente ajenos al estándar, situación que no cambiará hasta que SQL3 se convierta en estándar.



hacerse por ahora es permitir que una "federación" de servidores de bases de datos de proveedores múltiples, propiedad autónoma y acoplamiento holgado se comunique empleando el método de "mínimo común denominador". La industria llama a esta modalidad *sistemas de bases de datos en federación*.

En este capítulo se analiza el middleware necesario para lograr que clientes y servidores de SQL trabajen en redes de base de datos heterogéneas de proveedores múltiples o, para decirlo llanamente, bases de datos en federación. ¿Qué tan satisfactorio es este middleware en la creación de una "ilusión de base de datos única" en un ámbito en federación?

Para crear la "ilusión de base de datos única", el middleware debe satisfacer a dos grupos de clientes: 1) los desarrolladores de aplicaciones y herramientas frontales, que precisan de una sola API de SQL independiente del OS para acceder a cualquier servidor de bases de datos, y 2) las personas relacionadas con la conectividad de MIS, quienes deben lograr que clientes de escritorio dispares se comuniquen con los servidores de bases de datos "en federación" de sus redes empresariales. El middleware debe resolver asuntos tan difíciles como los siguientes: ¿en qué forma puede emitir un programa cliente llamadas de SQL de proveedores múltiples?; ¿Cómo interoperan los escritorios de base de datos en federación con servidores de bases de datos en federación?; Todo esto puede hacerse transparentemente?

Por fortuna, disponemos ya de middleware para la fusión de sistemas dispares, pero lamentablemente no puede usarse para crear bases de datos en federación en apoyo a la producción. Aun así, constituye un fundamento adecuado para sistemas de apoyo de decisiones y bodegas de datos.

MIDDLEWARE DE SQL: LAS OPCIONES

La verdadera independencia de la base de datos no será posible sin arquitecturas en tres planos.

Meta Group
(Noviembre de 1995)

Con base en nuestra definición anterior, el middleware comienza por la API en la parte del cliente por medio de la cual se invoca un servicio, y comprende la transmisión de la solicitud por la red y la respuesta resultante. El middleware no incluye el software que presta el servicio como tal. Así, las preguntas que debemos responder son: ¿qué API les brindan a los clientes los servidores de bases de datos de SQL?; ¿Cómo se realiza el intercambio de solicitud/respuesta con el servidor? Como lo comprobará más adelante, existen muchas respuestas a estas preguntas.

Peró antes de pasar a respuestas detalladas, formaremos primero una idea común que nos ayude a comprender las soluciones. Empezaremos con el "nirvana" de SQL, los productos de proveedor único integrados. Pasaremos después a los problemas causados en un entorno de SQL en federación de proveedores múltiples. Luego le ofreceremos una breve descripción general de las dos arquitecturas principales para la reducción de discrepancias en las bases de datos en federación. Finalmente, le daremos nuestra humilde opinión de aquello que un nirvana de SQL en federación debería incluir.



Nirvana de SQL: La opción de proveedor único

Si una solución de SQL de proveedor único puede satisfacer todas sus necesidades de datos compartidos, considérese muy afortunado. Basta con leer esta sección y pasar de inmediato al siguiente capítulo. En la Figura 11-1 se muestra lo que ofrece en la actualidad una solución común de middleware de proveedor único:

- Una API de SQL del proveedor-propietario que trabaja con múltiples plataformas del cliente. La mayoría de los proveedores soportan clientes con DOS, Windows y OS/2; muy pocos soportan también Macintosh y algunas variantes de Unix. Casi todas las API de los proveedores soportan SQL-89 con extensiones propietarias. Algunas de ellas emplean SQL incrustado (ESQL: embedded SQL), y otras soportan una interfaz del nivel de llamada (CLI: call-level interface). Abundaremos en este tema más adelante en este mismo capítulo.
- Un controlador de SQL del proveedor-propietario. Éste es un pequeño elemento de tiempo de ejecución del cliente que acepta las llamadas de API, da formato a un mensaje de SQL y maneja los intercambios con el servidor. El formato del mensaje de SQL y el manejo de enlaces se conocen coloquialmente como *formato y protocolos* (FAP: format and protocols). Los FAP de SQL son por lo general definidos por el proveedor.
- Soporte de FAP para múltiples pilas de protocolos. Como resultado de las presiones de los usuarios, la mayoría de los proveedores soportan ahora múltiples pilas de protocolos. Algunos las integran a sus controladores, mientras que otros soportan una interfaz de transporte común (como Sockets o Named Pipes), con lo que se requiere que usted aporte sus propias pilas. En cuanto al servidor, el proveedor suele proporcionar "escuchadores" para las diferentes pilas. Sin embargo, algunos ofrecen sus propios gateways de protocolos internos; por ejemplo, Oracle7 en Unix traduce paquetes de IPX/SPX a TCP/IP en la parte del servidor.

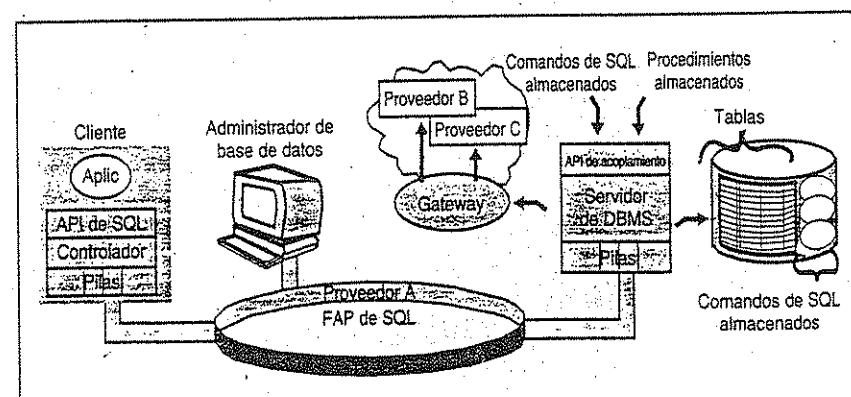


Figura 11-1. Componentes de una propuesta de cliente/servidor de SQL de proveedor único.



Algunos gateways llegan más lejos para "compensar" la pérdida de funciones de los mecanismos de base de datos "extranjeros". Por ejemplo, el gateway de Sybase/MDI ofrece un componente de macrocomputadora llamado *servidor de acceso CICS de DB2 (DB2 CICS Access Server)*, que emplea el monitor de TP de CICS para simular procedimientos almacenados de Sybase en DB2 (aunque, por supuesto, el costo es elevado). Los gateways de Oracle7 e Ingres soportan cierto nivel de grabación en dos fases para la actualización de datos extranjeros, método que sin embargo sigue correspondiendo al de mínimo común denominador.

En nuestra opinión (no se olvide que estamos en un recuadro de debate), si usted ya se aferró a una solución de base de datos de un solo proveedor, adopte el gateway del proveedor, para su comodidad. Tener acceso a bases de datos extranjeras suele ser útil para el apoyo de decisiones ocasional. Resulta además sumamente conveniente, porque usa las mismas API y middleware para acceder a esos datos extranjeros en forma casi transparente. Sin embargo, se halla por completo a merced de su proveedor, lo cual lo ata en exceso. Claro que dado el caótico estado del "middleware de base de datos abierto", aferrarse a una solución de proveedor único quizás no sea tan mala idea (lo que comprenderá mejor cuando termine de leer este capítulo). Por lo que respecta a la mayoría de las grandes empresas, la solución de base de datos de proveedor único no es siquiera digna de atención. Sus requerimientos de administración de base de datos son tan diversos que deben cubrir todo el espectro, desde sistemas de apoyo de decisiones basados en PC hasta datos de producción de OLTP de gran volumen. □

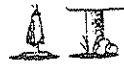
La pesadilla de SQL: La opción de proveedores múltiples

La interoperabilidad entre bases de datos de N proveedores termina por convertirse en N^2 problemas.

Mohsen Al-Ghosein, arquitecto de TP
Microsoft
(Septiembre de 1995)

En la Figura 11-3 se muestra lo que ocurre cuando se incursiona en un mundo de bases de datos de proveedores múltiples. He aquí la corta lista de inconsistencias obvias que inmediatamente enfrentará usted:

- Diferentes API de SQL vuelven una pesadilla generar una serie común de aplicaciones. Aún si la semántica de API común surgiera mágicamente en un apartado posterior de este capítulo, seguiríamos necesitando un medio para manejar la totalidad de las extensiones de SQL propietarias.
- Controladores de bases de datos múltiples consumen precioso espacio de memoria en las máquinas cliente (especialmente en el caso de DOS). ¿Estos controladores pueden usar las mismas pilas de protocolos o tenemos que duplicar pilas? Si se necesitan múltiples



pilas, ¿cómo compartirán el adaptador de LAN? ¿A quién recurrir cuando surge un problema?

- FAP múltiples y no interoperables significa que los protocolos de base de datos de diferentes proveedores sencillamente comparten la LAN, pero no pueden comunicarse entre sí.
- Herramientas de administración múltiple significa que los administradores de base de datos deben familiarizarse con una amplia variedad de estaciones de trabajo de administración, cada una de las cuales posee su propia semántica e interfaces de usuario.

Y eso que por lo pronto hemos dejado fuera algunos de los asuntos más espinosos, como las uniones de bases de datos en federación, las grabaciones en federación y el acceso concurrente a datos en federación.

Las soluciones de middleware en federación se concentran en el acceso de SQL simple a una base de datos en federación, una conexión por vez. Esquemas más ambiciosos, como RDA y DRDA, persiguen la creación de un entorno en federación que iguale la capacidad de una base de datos distribuida de proveedor único. Sin embargo, aún están lejos de cumplir esa meta. Así pues, lo mejor que podemos hacer por ahora es limitarnos a los aspectos relacionados con el empleo, una por una, de instrucciones SQL simples en una base de datos en federación.

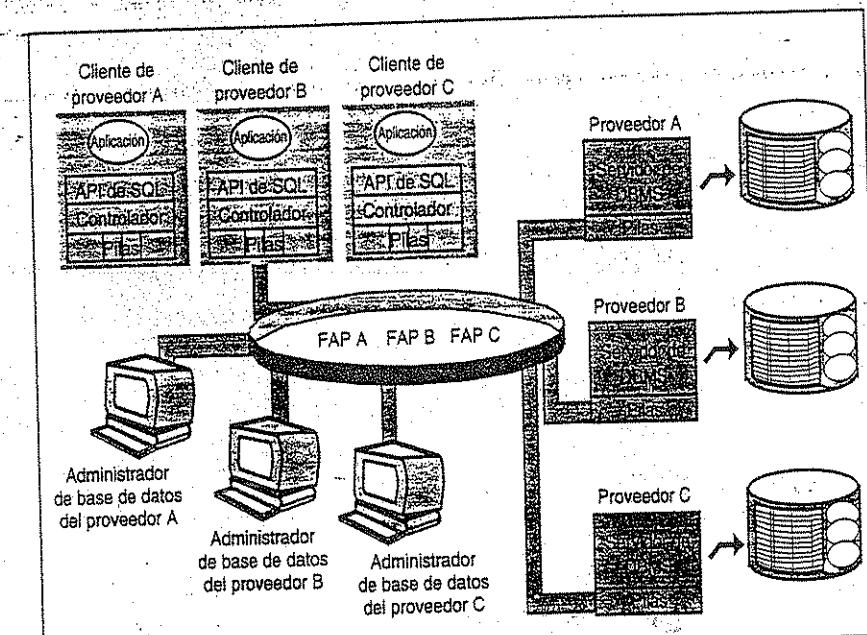
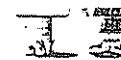


Figura 11-3. La pesadilla de base de datos de proveedores múltiples de SQL.



Solución de middleware #3: Nirvana en federación

Supongamos que ya contamos tanto con el FAP común como con la API común. ¿Qué más se necesita para crear un entorno de SQL en federación que ofrezca el mismo nivel de integridad de la implementación de proveedor único? En la Figura 11-6 se muestra cuál sería la apariencia de este "ideal". Nótese que hemos eliminado los receptores de gateways, lo que eleva el desempeño del servidor, reduce costos y simplifica el mantenimiento. Además, hemos creado una sola interfaz de administración de base de datos.

Para eliminar los receptores de gateways, el FAP común debe ya sea soportar un superconjunto de la totalidad de los dialectos de SQL o tolerar dialectos de SQL nativos (lo que significa que debe permitir aberturas). Por su parte, los proveedores de SQL deben acceder a remplazar sus propios FAP privados por el FAP común.

La facilidad de administración de base de datos será el último bastión del propietario por derribar. Las implementaciones de servidor de base de datos son extraordinariamente variadas como para crear una interfaz común. Incluso si resolvíeramos los aspectos tecnológicos, quedarían por resolver aún delicados aspectos políticos. Por ejemplo, ¿existe un punto único de control de administración en un entorno de base de datos en federación? *DataHub* de IBM, herramienta basada en OS/2, es un ejemplo de herramienta de administración de base de datos en federación (aunque de proveedor único). Los productos de administración de base de datos de terceros —como *EcoTools* de Compuware, *Patrol* de BMC y *DBVision* de Platinum— también comienzan ya a ocuparse de estos asuntos. La buena noticia es que los proveedores de bases de datos

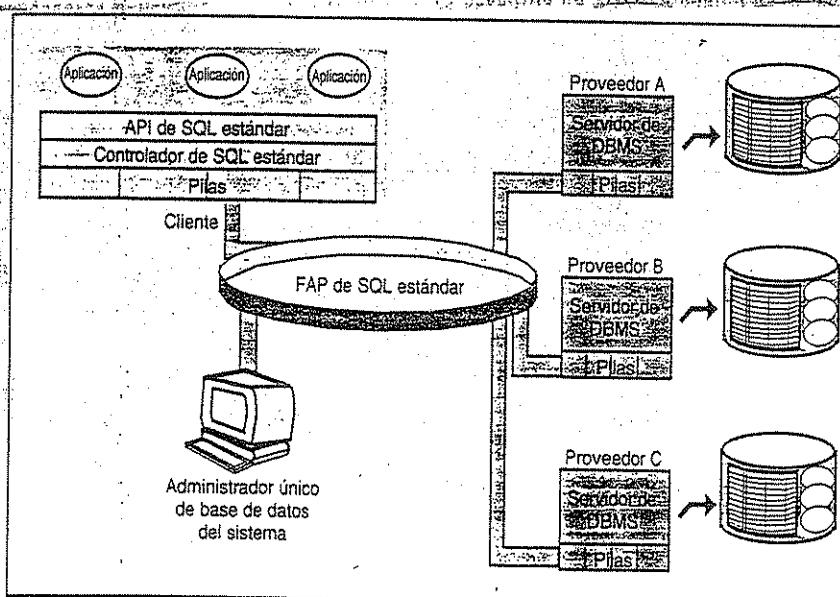


Figura 11-6. Tercera convergencia: Implementaciones directas del FAP común.

en la actualidad cuentan con un estándar de MIB de DBMS que define la configuración y parámetros de control de bases de datos para estaciones de administración de SNMP.

¿EXISTE LA API DE SQL VERDADERA?

¿Cómo se accede a datos de SQL? ¿Una aplicación puede acceder transparentemente a un sistema de bases de datos en federación? ¿Una aplicación construida para una base de datos de SQL puede desplegarse sobre otra? ¿Cuál es el estado de los estándares de acceso a datos de SQL? Como descubrirá en esta extensa sección, las respuestas son un vago sí, no y tal vez.

Los primeros arquitectos de SQL consideraron de gran importancia mantener un lenguaje de SQL neutral. SQL fue creado de hecho como un lenguaje declarativo del más alto nivel, que habría de alejarnos de construcciones de procedimientos de bajo nivel. Recuérdese que se le diseñó como un lenguaje de consulta para el usuario final. Pero para crear aplicaciones en las que se emplee SQL, resultó obvio que era necesario integrar construcciones de SQL a lenguajes de programación ya existentes. Casualmente, las personas para las que SQL es "lo primero" conciben este proceso como la extensión de SQL con capacidades de procedimientos, mientras que los programadores lo conciben como la provisión de una interfaz para servicios de SQL. Dos métodos en pugna prevalecen actualmente para el soporte de SQL con lenguajes de pro-

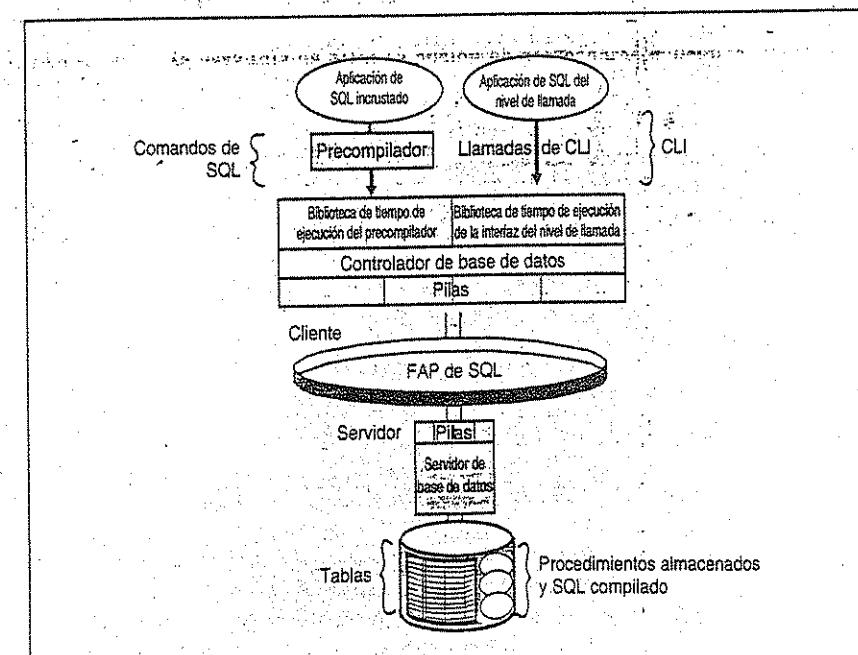


Figura 11-7. Los dos estilos de API de SQL: CLI y ESQL.

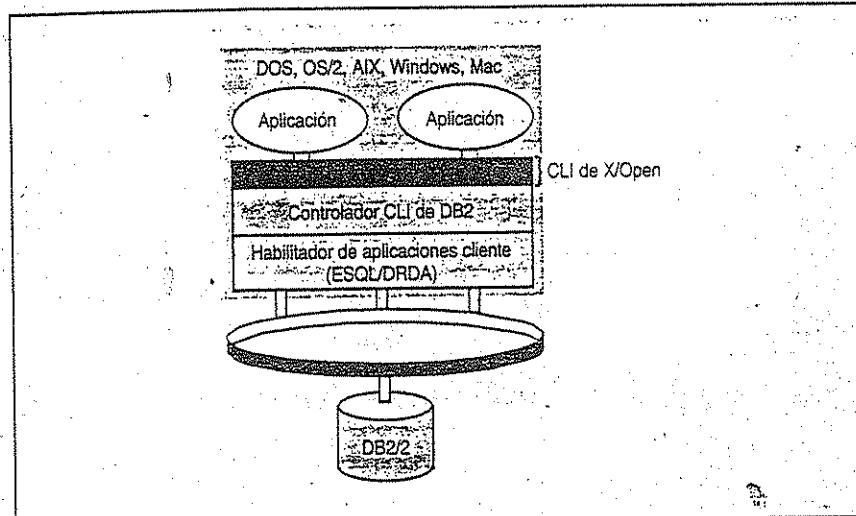


Figura 11-8. Controlador de CLI de X/Open de DB2.

con el estándar de ODBC de Microsoft. Sin embargo, no emplea el administrador de ODBC de Microsoft ni su arquitectura de controladores en planos múltiples (véase la siguiente sección).²

La CLI de Microsoft

ODBC ha iniciado el proceso de comercialización de las bases de datos y a nueve de cada 10 proveedores de DBMS eso no les hace la menor gracia.

Kingsley Idehen, presidente
OpenLink
(Octubre de 1995)

El estándar de API de Windows *conectividad de bases de datos abierta* (*ODBC: open database connectivity*) de Microsoft para SQL es una versión ampliada de la CLI de SAG. En agosto de 1992, Microsoft lanzó la ODBC 1.0 SDK, que habría de convertirse en la respuesta en paquete comercial para el acceso a bases de datos con Windows. Desde entonces, ODBC ha cruzado ya varias plataformas. Visigenic obtuvo de Microsoft la licencia exclusiva para ofrecer ODBC SDK en plataformas diferentes a Windows. Aparte de Visigenic, otros terceros interesados

² La implementación de DB2 de la CLI de X/Open, también introduce nuevas características ausentes en X/Open y ODBC. Por ejemplo, soporta BLOB y ofrece mejor control sobre el aislamiento de transacciones. Así, no pierda de vista las extensiones: volverán a hacerle falta si necesita verdadera exportabilidad.



—como Intersolv (*Q+E Software*) y OpenLink— ofrecen juegos de controladores de ODBC sobre Windows, Windows 95, Windows NT, OS/2, Mac y Unix que operan con una amplia variedad de servidores de bases de datos.

ODBC 1.0 era lenta y pesada; se le limitó a la plataforma de Windows, y carecía de los ejemplos necesarios de documentación y código para desarrolladores educativos. En abril de 1994, Microsoft lanzó la ODBC 2.0 SDK, en la que se habían resuelto ya muchos de los problemas del anterior administrador de controladores. En diciembre del mismo año se dieron a conocer los primeros controladores de 32 bits. ODBC 3.0 —anunciada en febrero de 1995— promete brindar más funciones: se esperaba que la SDK fuera lanzada a fines de 1996.³ Microsoft controla por completo el estándar de ODBC. Así, la pregunta del millón de oro en este caso es: ¿la futura ODBC se alinearán con la SQL3/CLI o se convertirán en un estándar propietario basado en OLE? Según Microsoft, la respuesta es todo lo anterior. Microsoft que soportará SQL3, anunció ya a OLE/DB como el futuro sustituto de ODBC.

ODBC 2.0 define alrededor de 61 llamadas de API, que corresponden a tres niveles de conformidad:

- El núcleo ofrece 23 llamadas base que permiten conexión con una base de datos, ejecución de instrucciones SQL, búsqueda de resultados, grabación y retroceso de transacciones, manejo de excepciones y conclusión de la conexión.
- El nivel 1 ofrece 19 llamadas adicionales, que permiten recuperar información de un catálogo de bases de datos, buscar objetos grandes (BLOB) y asumir funciones específicas del controlador.
- El nivel 2 ofrece otras llamadas de API, que corresponden a tres niveles de conformidad:

Las aplicaciones son responsables de asegurar que un controlador de ODBC soporta un nivel de conformidad (véase el siguiente recuadro de información). Es de hacer notar que la CLI de X/Open incluye el núcleo de ODBC y algunas de las funciones de los niveles 1 y 2. Incluye también descriptores de SQL ausentes en ODBC. No incluye en cambio las funciones de ODBC que soportan aplicaciones de Microsoft, como SQL Server, Access y Excel.

La mayoría de los proveedores de servidores de bases de datos —como Microsoft, IBM, Oracle, Sybase, Tandem, CA/Ingres e Informix— soportan ahora la API de ODBC 2.X aparte de sus API de SQL originales. Incluyen asimismo controladores de ODBC para sus respectivos servidores (véase Figura 11-9). El problema es que ODBC siempre parece ocupar un segundo puesto en relación con las interfaces nativas en las partes del cliente y el servidor. Por ejemplo, Oracle soporta ODBC como API opcional, pero su CLI nativa es la *interfaz del nivel de llamada* (OCI) de Oracle. La familia DB2 de IBM soporta ODBC (con extensiones) y la CLI de X/Open (con extensiones), pero su protocolo nativo es ESQL/DRDA. Sybase soporta ODBC

³ ODBC 3.0 añade 300 páginas a la especificación ya existente. Introduce 20 nuevas llamadas de funciones y soporta Unicode. También brindará más información sobre aproximadamente 40 nuevos conceptos; introducirá un descriptor semejante a SQLDA para SQL dinámico.

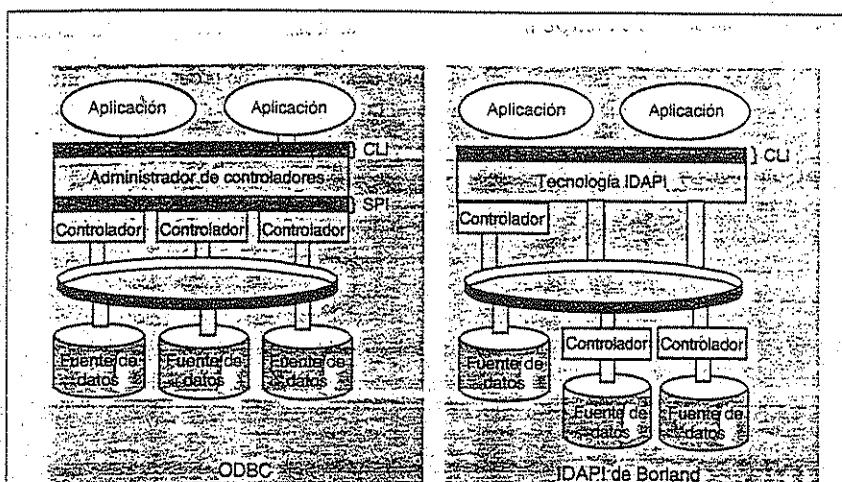


Figura 11-10. Controladores : ODBC contra IDAPI.

Los controladores de ODBC suelen obtenerse con proveedores de bases de datos y herramientas frontales. También se les incluye en algunas aplicaciones cliente/servidor que los requieren. Ciertas compañías de servidores de bases de datos han integrado controladores de ODBC en sus bases de datos sin costo adicional. También es posible adquirir controladores entre el nuevo género de compañías de "middleware para middleware". Por ejemplo, *Q+E Software* de Intersolv ofrece un juego de controladores para más de 35 servidores de bases de datos. A los proveedores de bases de datos no les interesa hacer controladores a menos que se vean obligados a ello. En consecuencia, algunos de ellos —como Oracle, Informix y Gupta— remiten directamente a los clientes necesitados de controladores con Visigenic o Intersolv. No obstante, la compra de controladores a terceros introduce una complicación más en un rizo (*loop*) de por sí complejo. Por su parte, los proveedores de "middleware para middleware" pueden verse convertidos en el único punto de apoyo y contacto para integraciones de bases de datos en federación que emplean CLI. □

CLI contra ESQL

En la Tabla 11-1 se hace una comparación de capacidades entre CLI y ESQL. En general, el surtido actual de CLI de SQL es flexible pero lento. Sólo se le puede emplear en sistemas de apoyo de decisiones. Sistemas de alto desempeño requerirán el uso ya sea de procedimientos almacenados o SQL incrustado estático. Una CLI, como ODBC, puede ser engañosa y permitirle invocar un procedimiento almacenado específico del proveedor mediante aberturas (usted especifica como parámetros el nombre del procedimiento almacenado y el servidor en la llamada EXECUTE SQL API). X/Open especificará en definitiva una llamada a procedimiento al-



macenado de CLI basada en SQL3. Pero no se moleste en esperar. En todo caso, las CLI no dejan de ser demasiado lentas en comparación con el desempeño de las API nativas. Y en tanto los proveedores no implementen nativamente la CLI (véase el siguiente recuadro de debate), las cosas no cambiarán.

Tabla 11-1. La CLI de X/Open contra ESQL de ISO

Característica	Interfaz del nivel de llamada (CLI) de SQL de X/Open	SQL incrustado (ESQL) de SQL-92 de ISO
Requiere conocimiento anticipado de la base de datos destino	No	Sí
Soporta SQL estático	No (futuro)	Sí
Soporta SQL dinámico	Sí	Sí
Soporta procedimientos almacenados	No (futuro)	No (futuro)
Usa el modelo declarativo de SQL	No	Sí
Las aplicaciones deben precompilarse y acoplarse con el servidor de bases de datos	No	Sí
Facilidad de programación	No	Sí
Propiedad a herramientas	Sí	No
Facilidad de depuración	Sí	No
Facilidad de empaquetamiento	Sí	No

¿Cuál CLI?



Debate

Una empresa promedio cuenta en la actualidad con ocho bases de datos, pero yo preferiría contar con un lenguaje de cliente/servidor común que todas pudieran hablar.

David Waller, director, Intersolv
(Octubre de 1995)

Use API nativas (directas) siempre que pueda. Las API estándar como ODBC deben considerarse el último recurso... no el primero.

Richard Finkelstein

al formato FAP, las transporta y después las hace corresponder con las llamadas del servidor apropiadas (y viceversa). El gateway abierto debe proveer (o soportar) una interfaz de SQL estándar (CLI o ESQL). Asimismo, debe ser capaz de localizar servidores remotos y de ofrecer servicios de catálogo sin necesitar un servidor de bases de datos intermedio. Finalmente, debe brindar herramientas para la creación del gateway por parte del servidor.

Describiremos tres arquitecturas (o productos) rivales de gateways comunes: *acceso a datos remotos* (RDA: *remote data access*) de ISO/SAG, *acceso a datos relacionales distribuidos* (DRDA: *distributed relational data access*) de IBM y *EDA/SQL* de IBI, gateway abierto que hoy en día soporta más de 50 plataformas de servidor de bases de datos. Los gateways son un remedio temporal hasta que los proveedores acuerden un FAP común y lo implementen *nativamente* en sus servidores. Así, analizaremos cuál de los FAP en contienda tiene mayores posibilidades de convertirse en ese estándar común.



RDA y DRDA: Mucho más que simples gateways

Información

DRDA y RDA son más que simples protocolos de gateway. Ambos cuentan con arquitecturas de extremo a extremo para la creación de verdaderas bases de datos distribuidas en federación. La mayoría de los gateways simplemente transmite una instrucción SQL a un sistema de bases de datos remoto, tratando por lo general a cada uno de ellos como una transacción independiente. DRDA y RDA persiguen el soporte de transacciones de sitios múltiples (aunque RDA está muy lejos de lograrlo todavía). Algunos gateways manejan la conversión de caracteres, pero carecen de algunas de las peculiaridades sofisticadas de DRDA y RDA para la creación de representaciones de datos comunes. Los gateways suelen enlazar dos ubicaciones; DRDA y RDA están hechos para soportar redes principales de datos (con múltiples puntos de entrada y salida).

EDA/SQL de IBI

Acceso a datos empresariales/SQL (EDA/SQL: *enterprise data access/SQL*), de Information Builders, Inc. (IBI), es una familia de productos de gateway abierto que se sirve de SQL para acceder a más de 50 servidores de bases de datos relacionales y no relacionales, todo un récord en la industria. Además, IBI ha desarrollado, junto con Microsoft, un conductor de ODBC para servidores de gateway de EDA/SQL. EDA/SQL es resultado de 10 años de experiencia de IBI en el desarrollo de código de gateways, fundamentalmente para acceso a consultas exclusivo para lectura. IBI no ofrece servidores de bases de datos; se concentra en el ramo del "pegamiento".

En la Figura 11-11 se muestran los componentes de EDA/SQL. He aquí lo que hacen:

- API/SQL es otra CLI "común" que emplea SQL-89 como lenguaje estándar de acceso a bases de datos. Permite el paso de llamadas de SQL que no reconoce.

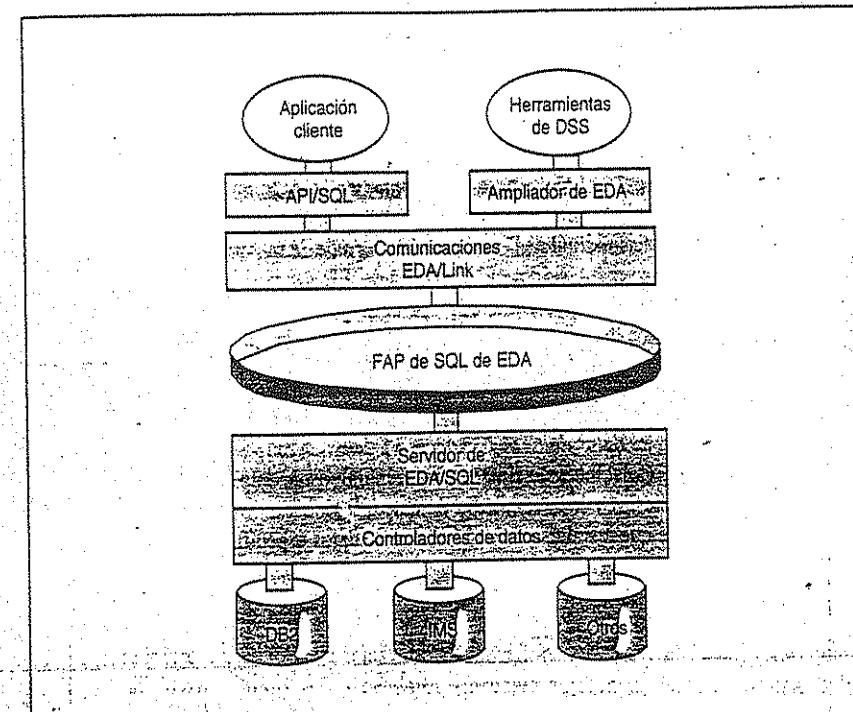


Figura 11-11. Componentes del gateway de EDA/SQL.

Las llamadas pueden ser emitidas asíncronamente, lo que significa que la aplicación cliente no se ve obligada a bloquear la espera para que la llamada se complete. Los clientes pueden consultar el estado de todas las solicitudes pendientes. La API/SQL también ofrece una llamada RPC que puede utilizarse para invocar transacciones de CICS o procedimientos generados por el usuario. Se halla disponible en DOS, Windows, Windows 95, OS/2, OS/400, AIX, Solaris, VAX/VMS, HP-UX, MVS, VM, Wang/VS y Macintosh.

- Los *ampliadores de EDA* son utilerías que permiten la emisión de llamadas de API/SQL desde productos existentes que soporten alguna modalidad de SQL dinámico. Se les puede concebir como "redireccionadores" de llamadas de SQL. Las llamadas son redirigidas a API/SQL, por supuesto, desde donde se les enruta después a través de la red de gateway. Los ampliadores se hallan disponibles para muchas aplicaciones de uso común —como Lotus 1-2-3—, así como para herramientas frontales de bases de datos que operan con bases de datos muy conocidas (relacionales o no relacionales). Muchas de las herramientas del cliente más comunes vienen habilitadas para EDA.
- EDA/Link soporta más de 12 protocolos de comunicaciones —entre ellos NetBIOS, Named Pipes, SNA, TCP/IP y DECnet. Ofrece verificación y autenticación de contraseñas y maneja traducciones de formatos de mensajes. Permite crear perfiles de comunicaciones em-

La meta de DRDA es proporcionar un estándar de interoperabilidad para entornos de bases de datos relacionales heterogéneas plenamente distribuidas. Para lograrlo, DRDA define los protocolos (o FAP) para interacciones de bases de datos cliente a servidor o servidor a servidor. En la terminología de DRDA se denomina al cliente como *solicitante de aplicaciones* (AR: application requester), y al servidor de bases de datos como *servidor de aplicaciones* (AS: application server). El protocolo AR a AS se emplea en las interacciones comunes de cliente/servidor. El protocolo AS a AS sincroniza transacciones dispersas entre múltiples servidores de SQL; también se le usaba para enrutar comandos de SQL de un servidor a otro.

En la Figura 11-12 se observan los cuatro niveles de transacciones de bases de datos definidos por DRDA:

- *Solicitud remota* significa un comando de SQL para una base de datos. Sirve principalmente para emitir consultas entre sistemas disímiles.
- *Unidad de trabajo remota* significa muchos comandos de SQL para una base de datos. Ésta es la transacción más común entre cliente y servidor único. El cliente puede conectarse con un servidor de bases de datos por vez, emitir múltiples comandos de SQL a la base de datos del servidor, emitir una grabación para volver permanente el trabajo y conmutar después a otro servidor de bases de datos para iniciar una unidad de trabajo subsecuente.
- *Unidad de trabajo distribuida* significa muchos comandos de SQL a muchas bases de datos, aunque cada comando está dirigido a una base de datos. Ésta es la transacción común de servidores múltiples. DRDA maneja la sincronización de sitios múltiples, la seguridad y las funciones de integridad de los datos (grabación en dos fases). Localiza sitios de datos remotos y coordina solicitudes (incluida la actualización de datos) en varias ubicaciones en una sola transacción.

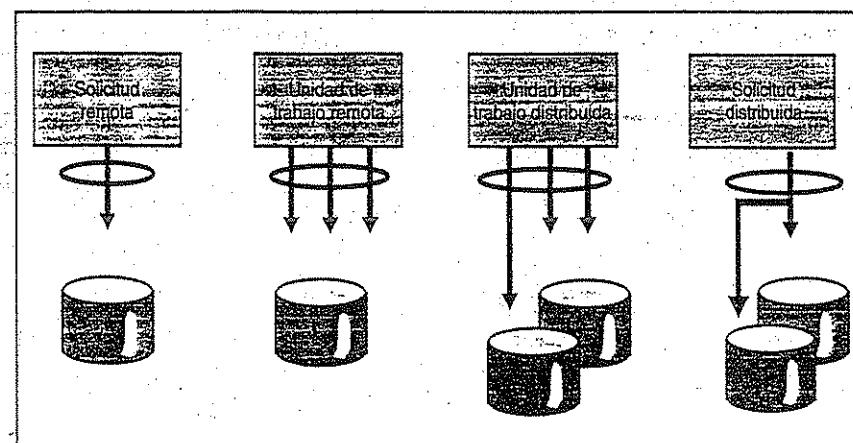


Figura 11-12. Los cuatro tipos de transacciones de bases de datos de DRDA.

- *Solicitud distribuida* significa muchos comandos de SQL a muchas bases de datos, pero en este caso cada comando puede ejecutarse en múltiples bases de datos. Con esta capacidad, DRDA puede atender solicitudes únicas destinadas a muchos sitios, como en el caso de una unión (JOIN) de sitios múltiples. Asimismo, puede distribuir una consulta única entre múltiples servidores para elevar el desempeño por medio del paralelismo.

Características de DRDA

DRDA funciona espléndidamente si todos cumplen con la sintaxis y semántica de SQL del servidor. Pero en un mundo heterogéneo, quizás las organizaciones de IS deban esperar mucho tiempo para ver realizado ese acuerdo.

**David Stodder, editor en jefe
Database Programming and Design
(Octubre de 1995)**

¿Qué tipo de funciones adicionales ofrece DRDA? Se encarga sobre todo de los espinosos asuntos de la red y la exportabilidad del código, tales como:

- *Contenido de los mensajes y protocolo de intercambio de SQL*: DRDA maneja las negociaciones entre clientes y servidores para atributos del servidor soportados. Efectúa traducciones de mensajes sólo en caso necesario. En este ámbito no priva el concepto de formato canónico de mensajes, sino un protocolo según el cual "el receptor se encarga del asunto". Esto significa que si los datos deben convertirse, se hace sólo una vez. Si cliente y servidor usan los mismos formatos, se omite la conversión. DRDA se ocupa de representaciones de datos disímiles, estructuras de catálogo y conversiones de sintaxis de comandos. No etiqueta cada campo en cada una de las filas de un conjunto de resultados de filas múltiples. Por el contrario, crea un solo descriptor para el conjunto de resultados en su totalidad. Todas estas características contribuyen a reducir el tráfico en la red y elevar el desempeño.
- *Independencia de la pila de transporte*: DRDA soporta la interfaz de MPTN, lo que significa que puede correr sobre APPC/APPN o TCP/IP (los dos protocolos actualmente soportados por AnyNet) para comunicaciones de cliente/servidor. DRDA maneja formación de bloques de datos, seguridad, autenticación y enruteamiento del servidor, y genera alertas de fallas tanto de la red como de la base de datos.
- *Preparación de programas de plataformas múltiples*: DRDA soporta en forma oculta preparación de programas de plataformas múltiples. Un programa se crea localmente; su producción puede distribuirse a servidores múltiples por medio de una utilería de BIND remota. El proceso de BIND produce código de SQL ejecutable (llamado paquetes o planes) en los servidores.
- *Soporte de SQL estático o dinámico*: Un cliente de DRDA puede invocar una o más instrucciones SQL en el servidor mediante la identificación de un paquete y de la instrucción al interior de ésta. Aparte de hacerlo con SQL dinámico, los paquetes también permiten



tadas de dinero. La pobre máquina tenía incluso que levantarla cada mañana con un análisis de la situación de su portafolio de inversiones.

¿DÓNDE SE GUARDAN LOS DATOS DE OLTP?

Las empresas modernas viven de datos. La cantidad total de datos en computadoras se duplica en nuestros días cada cinco años. Con la proliferación de tecnología de cliente/servidor (y multimedia), calculamos que en el futuro los datos se duplicarán al menos cada año. ¿Quién crea todos estos datos? La respuesta es: las instituciones modernas en el curso de la realización de sus actividades diarias. Los sistemas de producción computarizada que reúnen y consumen estos datos se llaman sistemas de OLTP, verdaderas fábricas de datos que no dejan de trabajar un solo instante del día.

¿Qué es OLTP?

Las aplicaciones cliente/servidor centradas en bases de datos se dividen en dos categorías: *sistemas de apoyo de decisiones (DSS: decision support systems)* y *procesamiento de transacciones en línea (OLTP: online transaction processing)*. Estas dos categorías de cliente/servidor ofrecen soluciones de negocios absolutamente distintas. Estas diferencias deben conocerse para poder apreciar lo que ofrecen las bodegas de datos.

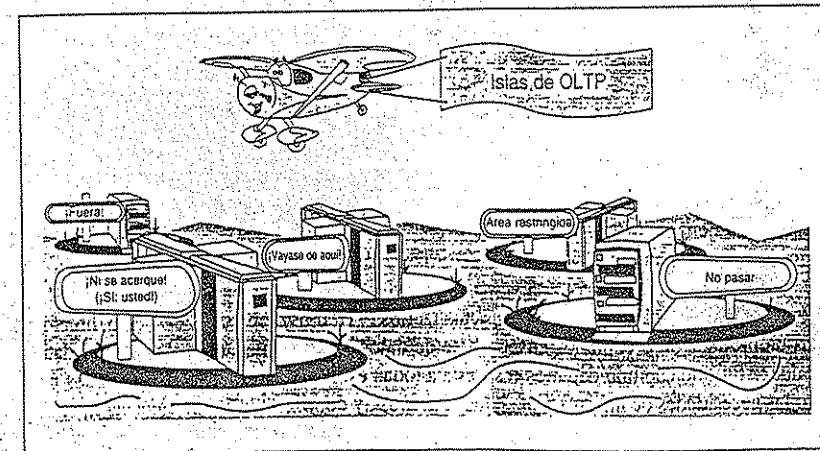
Los sistemas de OLTP sirven para crear aplicaciones en todo género de actividad. Entre ellas pueden citarse los sistemas de reservaciones, punto de venta, sistemas de rastreo, control de inventarios, estaciones de trabajo de intermediarios bursátiles y sistemas de control para fábricas manufactureras. Se trata por lo general de aplicaciones decisivas para el cumplimiento de objetivos que en el 100% de los casos requieren de un tiempo de respuesta de 1-3 segundos. El número de clientes soportados por un sistema de OLTP puede variar considerablemente en un día, mes o año, pero lo que no puede cambiar es el tiempo de respuesta. Las aplicaciones de OLTP también requieren rigurosos controles sobre la seguridad e integridad de la base de datos. La confiabilidad y disponibilidad del sistema general deben ser del más alto nivel. Los datos deben conservarse en forma consistente y correcta.

En los sistemas de OLTP lo común es que el cliente interactúe con un servidor de transacciones, no con un servidor de base de datos. Esta interacción es necesaria para contar con el alto desempeño que estas aplicaciones requieren. Los servidores de transacciones pueden ser de dos tipos: *OLTP ligero*, provisto por procedimientos almacenados, y *OLTP pesado*, provisto por monitores de TP. En ambos casos, el cliente invoca *procedimientos remotos* que residen en un servidor. Estos procedimientos remotos se ejecutan como transacciones con la base de datos del servidor (tema en el que abundaremos en la Quinta parte). Las aplicaciones de OLTP demandan la generación de código tanto para el componente del cliente como para las transacciones del servidor. La carga general de comunicaciones en las aplicaciones de OLTP se mantiene al mínimo. La interacción del cliente con el servidor de transacciones suele limitarse a breves intercambios estructurados. Estos intercambios consisten en una sola solicitud/respuesta, en oposición a los intercambios de mensajes múltiples de SQL.

¿Cliente/servidor está creando nuevas islas de OLTP?

Las aplicaciones de OLTP corrían anteriormente en costosas macrocomputadoras que almacenaban abundantes cantidades de datos, ofrecían interrupciones (tiempo de caída) mínimas y eran el orgullo de las empresas y las tiendas de MIS. Hoy, las aplicaciones de OLTP de mayor calidad —como reservaciones de aerolíneas, bancarias, de mercados accionarios, torres de control de aeropuertos y hospitales— siguen corriendo en costosos superservidores y macrocomputadoras, y estando bajo el control de las tiendas de MIS. Sin embargo, todo departamento con el presupuesto suficiente para comprar unas cuantas PC las enlaza en la actualidad a una LAN de cliente/servidor y contrata los servicios de un programador (o consultor) capaz de crear una aplicación de OLTP *propia*. Ahora también puede disponerse ya de paquetes de software. En otras palabras, la tecnología de cliente/servidor centrado en base de datos ha derribado las barreras de entrada para la creación de sistemas de OLTP privados o departamentales. Estos sistemas les están concediendo a los departamentos autonomía y control total sobre las aplicaciones que crean y los datos que recolectan. En casos extremos, todo un sistema de OLTP puede correr en una base de datos de escritorio de usuario único; todos los datos recolectados pueden mantenerse en privado (es decir, fuera del alcance de las empresas). Bien sabemos lo fácil que es crear sistemas específicos de bases de datos en PC independientes que empleen hojas de cálculo y herramientas simples de bases de datos.

En general, la totalidad de los datos reunidos por un sistema de OLTP es de uso directo en la aplicación y para las personas que los crean. Éstas saben con precisión qué significan esos datos. Saben además cómo usarlos para resolver sus inmediatos problemas de producción de todos los días. La aplicación suele ofrecer una sofisticada interfaz gráfica para la visualización y manipulación de los datos con controles de transacciones. Los miembros de la organización saben cómo están estructurados los datos. Pueden crear complejos reportes integrados y manipular los datos para sus usos de producción.



¿Qué ocurre cuando alguien ajeno al grupo de OLTP necesita los datos de éste? ¿Cómo puede saber qué datos están a su disposición? ¿Dónde puede buscarlos? ¿Cómo puede acceder a ellos? ¿En qué formato los encontrará? Y, ¿qué significan? Lo que menos quieren las personas que trabajan con OLTP es permitir a extraños el acceso a sus preciosos sistemas de producción. Lo común es que estos extraños ni siquiera sepan qué desean, y que por lo tanto emitan largas consultas específicas que entorpezcan la operación del sistema de producción en su conjunto, contaminen a los datos y generen puntos muertos.

Antes, los extraños podían pedirles a sus representantes de MIS que hicieran contacto con sus contrapartes de MIS que controlaban datos de producción para que les indicaran qué datos se hallaban disponibles y cómo obtenerlos. Pero con la proliferación de soluciones de OLTP privadas unipersonales y departamentales, incluso los representantes de MIS ignoran ahora qué datos se encuentran disponibles. Los datos de las empresas se han fragmentado, de manera que hemos vuelto a islas de procesamiento de datos. Ahora los datos pueden estar en cualquier parte: en el cliente que los origina, en el servidor departamental, en uno de muchos servidores en federación o en el servidor de la empresa. La visión integrada ya no existe más.

Uno de los mayores atractivos de cliente/servidor y las PC es la autonomía que otorgan. Lo común es que nos sintamos ajenos a los datos de la empresa como tal y prefiramos el control local de nuestros recursos. En la mayoría de los casos, nuestra recién descubierta libertad nos induce a replegarnos en nuestras pequeñas esferas de producción y a ignorar las necesidades de la comunidad. Hemos creado una dicotomía entre las necesidades personales (o departamentales) y las necesidades de la organización o la comunidad. De igual forma, también hemos creado una dicotomía entre datos de producción y datos de información.

Pero, ¿quién es esos "extraños" a los que pretendemos mantener lejos de nuestro ámbito? Son personas que analizan datos en busca de socios, tendencias y piezas de información que les permitan tomar mejores decisiones. Erigir barreras para la protección de los datos es como erigir barreras comerciales. Si los demás no pueden acceder a nuestros datos y nosotros no podemos acceder a los tuyos, todos salimos perdiendo. De esta manera, bien puede ocurrir que datos sumamente valiosos queden fuera del alcance de aquellos que más los necesitan.

INFORMACIÓN AL INSTANTE

¿Cómo preservar la autonomía local de los sistemas de producción y permitir al mismo tiempo que personas ajenas tengan acceso a ellos? ¿Cómo cerciorarnos de que estas personas no causarán impactos indeseables en los sistemas de producción? ¿Qué datos poner a su disposición? ¿Qué datos mantener en privado en el sistema de producción? ¿A quién asignarle la propiedad de los datos compartidos? ¿Quién puede actualizarlos? Hemos de permitir el acceso directo a datos de producción o copiarlos en otra base de datos? ¿Cómo mantener y regenerar datos extraídos? En este capítulo daremos respuesta a todas estas preguntas. Pero examinemos primero las necesidades de información de los "extraños" y conozcamos las diferencias entre los sistemas de apoyo de decisiones y los sistemas de producción de OLTP.

Buscadores de información

Comencemos por darles un nombre a los "extraños" que desean consumir nuestra información. Estas personas van desde individuos con un apetito insaciable de datos —como el personaje interpretado por Danny De Vito en *Riqueza ajena*— hasta aquellos con apenas ocasionales necesidades de datos, como el estudiante que realiza una investigación para un trabajo escolar. ¿Qué nombre les pondremos? ¿Qué le parece a usted *tomadores de decisiones*? ¿O preferiría *buscadores de información*? Quedémonos con *buscadores de información*, porque bajo esa denominación podemos incluir a los millones de personas que muy pronto adoptarán ese papel con la tecnología de las bodegas de bases de datos. Cualquier individuo con una PC conectada a una bodega de datos podrá hacer lo mismo que Danny De Vito en aquella multirreferida película.

Por supuesto que los primeros en consumir esta tecnología son los hombres y mujeres de negocios que tienen que tomar decisiones estratégicas —precios y análisis de mercado— para las cuales dependen de la disponibilidad de datos precisos y oportunos. La capacidad para tener acceso a información y proceder en consecuencia con toda celeridad será cada vez más importante para el éxito de cualquier compañía (o persona). Los datos en bruto se convierten en información cuando llegan a manos de alguien capaz de situarlos en su debido contexto y utilizarlos. Los datos son el ingrediente básico gracias al cual esto es posible. Existen muchos paralelismos entre la fabricación y distribución de bienes y la fabricación y distribución de información. La toma de decisiones de alto impacto y alto valor implica riesgos. Tomar decisi-

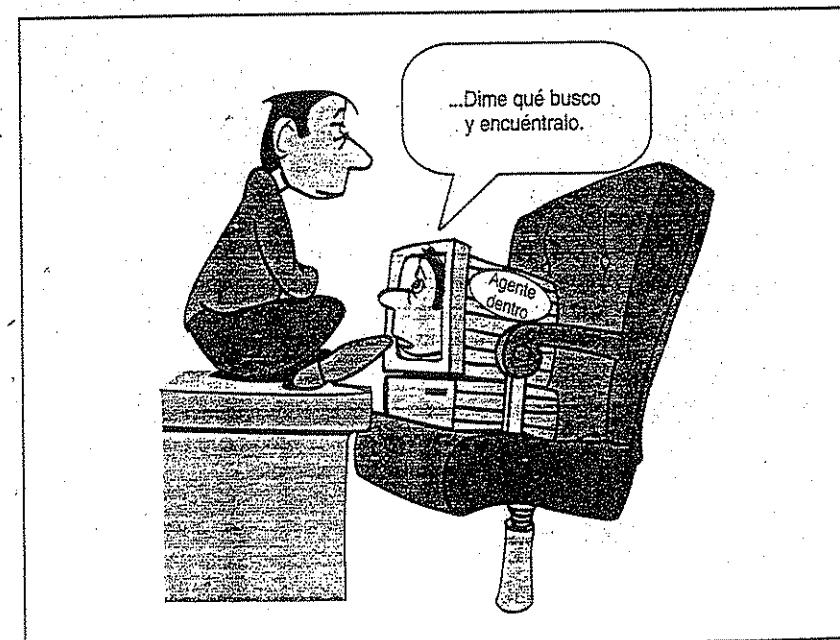




Tabla 12-1. Comparación de las acciones de programación de apoyo de decisiones y OLTP.

	Aplicación cliente/servidor	Servidor	Mensajes
Apoyo de decisiones	Herramienta comercial de apoyo de decisiones con creación de scripts por el usuario final. Manipuladores de eventos y comunicaciones con el servidor prestablecidos.	Servidor comercial de bases de datos. Tablas usualmente definidas por DBA como parte de una bodega de datos.	Consultas y uniones de SQL.
OLTP	Aplicación cliente. La herramienta de GUI diagrama la pantalla, pero los manipuladores de eventos y llamadas a procedimiento remoto requieren de programación de nivel C.	Aplicación cliente. El código de transacciones debe programarse al nivel C o COBOL. La base de datos es comercial.	Optimización de las llamadas de funciones del cliente para desempeño y acceso seguro.

Bases de datos de producción contra información

Cuando una organización puede examinar información durante un periodo prolongado, las tendencias no perceptibles en la información del momento se vuelven evidentes.

W. H. Inmon
(Agosto de 1995)

En la Tabla 12-2 se hace una comparación de requerimientos de bases de datos para el caso de los sistemas de OLTP y apoyo de decisiones. Es importante que conozcamos estas diferencias, pues de este modo comprenderemos lo que las bodegas de datos pueden hacer por los buscadores de información. Los puntos clave de diferenciación son que los datos de apoyo de decisiones deben mantenerse estables en una foto instantánea temporal para efectos de generación de reportes. Por el contrario, las bases de datos de producción reflejan la situación de último minuto de la empresa en tiempo real. A los buscadores de información no suele agradárles que los datos cambien con tanta frecuencia que les sea imposible obtener dos veces la misma respuesta en una fila. Así, las copias de información pueden actualizarse menos a menudo.

Los datos de apoyo de decisiones —o *datos de información*— se reúnen a partir de múltiples fuentes; los datos de producción son recolectados por aplicaciones de OLTP. Normalmente, los datos en bruto extraídos por los sistemas de apoyo de decisiones de bases de datos de producción no se actualizan directamente. No obstante, los buscadores de información tienen la gran exigencia de modelar la base de datos de información a la medida de sus necesidades específicas. A este proceso se le conoce como “mejora de datos derivados”. La base de datos de información puede contener datos derivados en los que se registren cambios a través del tiempo y resúmenes. A los buscadores de información rara vez les interesa un evento pasado específico. Siempre están a la caza de resúmenes y tendencias.

Tabla 12-2. Necesidades de base de datos: OLTP versus DSS.

Característica	Necesidades de base de datos de OLTP	Necesidades de base de datos de apoyo de decisiones
¿Quién la usa?	Trabajadores de producción.	Buscadores de información.
Oportunidad de los datos	Se necesita valor actual de los datos. Los reportes no pueden reelaborarse.	Se necesita fotos instantáneas estables de datos con registro de tiempo. Los intervalos de regeneración en momentos precisos son controlados por el usuario. Los reportes pueden reelaborarse con datos estables.
Frecuencia de acceso a los datos	Continua a lo largo del día de trabajo. Esporádica. Pueden ocurrir picos determinados por las labores.	
Formato de datos	Datos capturados en bruto. Ausencia de datos derivados. Datos de transacciones detallados y sin resumir.	Niveles múltiples de conversiones, filtración, resúmenes, condensación y extracción.
Recolección de datos	De una sola aplicación.	De múltiples fuentes.
¿Fuente de datos conocida?	Sí, generados en su mayoría por una aplicación única.	No, proceden de diferentes bases de datos.
Instantáneas temporales o versiones múltiples	No, datos continuos. Versión única.	Sí, se puede deducir la fecha/hora de la instantánea. Cada una de ellas es una versión, a menos que se le sobrescriba durante la regeneración.
Patrón de acceso a datos	Usuarios múltiples que actualizan la base de datos de producción.	Fundamentalmente acceso de usuario único. Uso intenso ocasional. Por ejemplo, cuando debe elaborarse un reporte.
¿Los datos pueden actualizarse?	El valor presente se actualiza permanentemente.	Exclusivos para lectura, a menos que se posea un duplicado.
Flexibilidad de acceso	Inflexible; el acceso a los datos se realiza vía programas precompilados y procedimientos almacenados.	Muy flexible, vía un generador de consultas, uniones de tablas múltiples y OLAP.
Desempeño	Rápido tiempo de respuesta obligatorio. Tareas altamente automatizadas y repetitivas.	Relativamente lento.
Requerimientos de datos	Claramente determinados. Conocidos antes de la construcción.	Vagos e inestables. Alto grado de trabajo e investigación detectivescos. Datos orientados a temas.
Alcance de información	Finito. Todo lo que se encuentra en la base de datos de producción.	Los datos pueden proceder de cualquier parte.
Cantidad promedio de registros accedidos	Menos de 10 registros individuales.	Centenas a miles de registros en conjuntos.

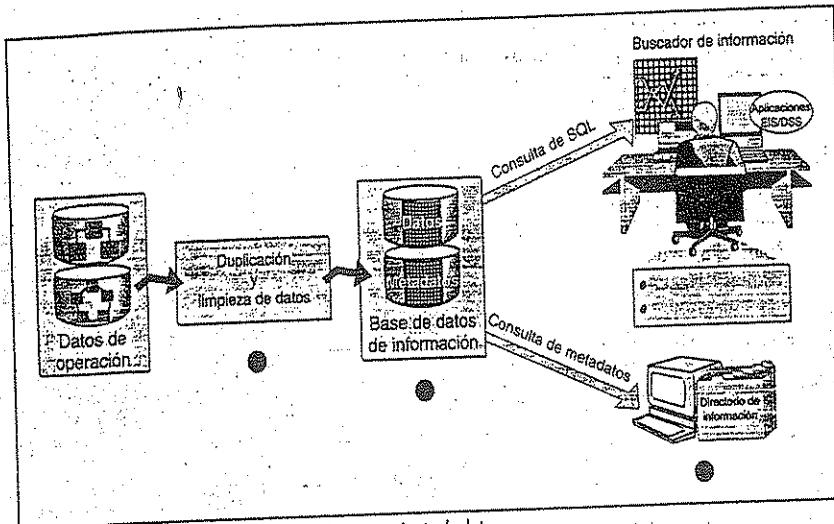


Figura 12-1. Elementos de un sistema de almacenamiento de datos.

tes relacionales o no relacionales. Cabe indicar que casi todos los datos externos son *transformados* y *limpiados* antes de ser introducidos a la bodega. En secciones posteriores nos ocuparemos en detalle de la duplicación de datos.

2. *La base de datos de información* es una base de datos relacional que organiza y almacena copias de datos de múltiples fuentes de datos en un formato que satisface las necesidades de los buscadores de información. Puede concebirse como el servidor de apoyo de decisiones que transforma, acrecienta y añade valor a los datos procedentes de diversas fuentes de producción. También almacena *metadatos* (o datos sobre datos), los cuales describen el contenido de la base de datos de información. Los *metadatos del nivel del sistema* describen las tablas, índices y extractos fuente a un administrador de base de datos (DBA); los *metadatos del nivel semántico* describen el contenido de los datos a un buscador de información. La base de datos de información puede ser una base de datos personal en una PC, una base de datos de tamaño mediano en un servidor local o una base de datos masivamente paralela en un servidor empresarial. La mayoría de los principales mecanismos de bases de datos de SQL pueden servir como bases de datos de información. Abundaremos en los mecanismos de bases de datos de información en los dos capítulos siguientes.
3. *El directorio de información* combina las funciones de un directorio técnico, un directorio de negocios y un navegador de información. Su propósito primordial es ayudar al buscador de información a encontrar qué datos están disponibles en las diferentes bases de datos, en qué formato se hallan y cómo acceder a ellos. También contribuye a que los DBA administren la bodega de datos. El directorio de información obtiene sus metadatos mediante el descubrimiento de cuáles bases de datos están en red y la posterior consulta de sus depósitos de metadatos. Intenta mantenerlo todo al día. De acuerdo con ciertas estimaciones, los buscadores de información de las grandes empresas dedican el 80% de su tiempo a reunir buscadores de información de las grandes empresas dedicarán el 80% de su tiempo a reunir

datos y el 20% restante a analizarlos. En ocasiones ignoran incluso el nombre de los objetos que buscan, dónde se localizan y cómo acceder a ellos. El directorio de información contribuye a remediar estos problemas al fungir como directorio de negocios.

Los DBA usan el directorio de información para acceder a los metadatos del nivel del sistema y llevar un registro de las fuentes de datos, objetivos de datos, reglas de depuración, reglas de transformación y detalles de reglas y reportes predefinidos.

Ejemplos de directorios de información/metadatos son el *Directory Manager* de Prism, *DataGuide* de IBM, *Information Warehouse Guide* de HP, *Infoharvester* de Minerva, *Extract and Metadata Exchange* de ETI, *Data Dictionary* de BrownStone, *Data Shopper* de Platinum/Reitech, *OpenBridge* de Informatica y *Passport Relational Dictionary* de Carleton.

4. *El soporte de herramientas de EIS/DSS* se proporciona vía SQL. La mayoría de los proveedores soportan ODBC y algún otro protocolo. Algunos de ellos —Red Brick, por ejemplo— ofrecen dialectos de SQL extendidos para acelerar consultas y uniones. Las herramientas se interesan más en el acceso secuencial de grandes cantidades de datos que en el acceso a un solo registro. Esto significa que se deben sintonizar los índices de tablas para consultas y lecturas secuenciales, en oposición a las actualizaciones. En el siguiente capítulo se le dará una idea de las herramientas de EIS/DSS disponibles y de lo que pueden hacer por usted. Tenga en cuenta que la mayoría de los proveedores de bodegas de datos han establecido alianzas con los principales proveedores de herramientas de EIS/DSS.

En síntesis, el almacenamiento de datos es el proceso de automatización de los sistemas de EIS/DSS con el uso de los cuatro elementos que acabamos de describir. Usted debe estar en condiciones de conjuntar datos de diferentes fuentes, duplicarlos, depurarlos, almacenarlos, catalogarlos y ponerlos después a la disposición de las herramientas de EIS/DSS. El truco es poder automatizar el proceso y hacerlo funcionar como un reloj.

¿Qué es lo que se automatiza?

Una de las principales diferencias entre un sistema de OLTP y una bodega de datos es la capacidad para describir minuciosamente el pasado. Los sistemas de OLTP son deficientes para la correcta representación de una empresa a un mes o un año de distancia. Un buen sistema de OLTP se halla en constante evolución.

Ralph Kimball, fundador de Red Brick
(Abril de 1996)

El almacenamiento de datos es una estructura para la automatización de todos los aspectos del proceso de apoyo de decisiones. En lugar de preguntarles a los administradores de bases de datos (DBA) qué información se halla disponible, ahora los buscadores de información pueden consultar directamente el directorio de información. Por supuesto que, como todos los buenos DBA, el directorio de información obtiene sus definiciones de datos de los metadatos almacenados en los diversos servidores.



do un caso de negocios en un asunto de bodega de datos empresarial, con la excelente justificación de que es necesario impedir que el personal de producción se vuelva loco.

Duplicación contra acceso directo

Las aplicaciones capaces de tolerar datos con antigüedad de unas cuantas horas a un día son mucho mejores candidatas a duplicación.

Glenn Froemming, DBMS Magazine
(Marzo de 1996)

Con la vasta difusión de la tecnología de cliente/servidor y las bases de datos en federación de acoplamiento holgado, es impráctico desde muchas perspectivas—desempeño, seguridad, disponibilidad, obligaciones históricas y control local—crear un solo depósito centralizado de datos. La administración de datos duplicados se usará cada vez más para eliminar los obstáculos de capacidad, desempeño y organización del acceso a datos centralizados.

La administración de copiado automatizado —o administración de datos duplicados— es ya una tecnología clave para compartir datos en un entorno de bases de datos en federación. Las aplicaciones de apoyo de decisiones que emplean bodegas de datos son candidatas perfectas a tecnología de datos duplicados. Estas aplicaciones suelen tolerar cierto grado de obsolescencia —aunque el término correcto sería *volatilidad*— en sus datos. La duplicación de datos para apoyo de decisiones reduce al mínimo la disruptión de sistemas de producción y permite modelar las bases de datos de información a la medida de las necesidades específicas. Por el contrario, el *acceso directo a datos* es requerido por aplicaciones —en su mayoría de OLTP de producción— incapaces de tolerar cualquier “volatilidad” en sus datos. Estas aplicaciones requieren de “datos vivos” que reflejen el estado de la empresa. Este tipo de datos vivos se obtienen en situaciones distribuidas mediante uno de cuatro métodos:

- *El uso de bases de datos en federación que soporten duplicación de datos síncrona (o continua):* las bases de datos destino deben sincronizarse dentro de la misma frontera de transacciones que la base de datos primaria (o fuente). Una base de datos destino que le permite a un usuario la actualización directa se llama *réplica*. Para mantener una política de actualización de sitio único, la réplica que se actualiza se convierte en la nueva base de datos fuente y debe propagar de inmediato sus actualizaciones. En general, la tecnología de duplicación síncrona es una propuesta riesgosa en entornos de bases de datos en federación. Requiere soporte de protocolos de grabación en dos fases entre bases de datos heterogéneas.
- *El uso de un servidor de base de datos centralizado:* todos los datos se conservan en un servidor con enorme facilidad de escalabilidad y tolerancia a fallas. En caso de adaptarse a sus necesidades organizacionales, esta solución es la que le dará muchos dolores de cabeza.
- *El uso de un producto de servidor múltiple de base de datos distribuida de proveedor único:* como puede advertirse, no hemos dicho que de proveedores múltiples, porque está tecnología padece aún muchas lagunas (véase el siguiente recuadro de información).
- *El uso de un monitor de TP para servidores frontales o de primer plano de bases de datos de proveedores múltiples:* los servidores de bases de datos deben soportar el protocolo



lo de XA de X/Open para poder ser administrados por un monitor de TP. Como se comprobará en la Quinta parte, esta tecnología puede resultar muy atractiva en muchas situaciones.

En suma, se necesitan ambos tipos de acceso a datos: duplicado y directo. Los aspectos relacionados con el acceso directo son ya muy conocidos por la industria; se dispone de abundantes soluciones comerciales. A su vez, la administración de datos duplicados en una estructura de bodega de datos está abriendo nuevas y muy interesantes oportunidades. A medida que una mayor cantidad de PC sean habilitadas para multimedia, comenzaremos a ver aparecer federaciones de bases de datos de información que incluyan al cliente en escritorio, donde se captura y visualiza la información local; al servidor local, que proporciona el almacenamiento de sobreflujo, y a servidores globales que contengan colectivamente una cantidad infinita de información y almacenamiento. La eficiente administración de duplicación y copiado se convierte en el pegamento que une a estas nuevas federaciones de bases de datos.



Información

El modelo de base de datos distribuida

La duplicación es más fácil de entender y mucho más fácil de implementar que los protocolos de grabación en dos fases.

Gartner Group
(Agosto de 1995)

Contra lo que los proveedores dicen y lo que la mayoría de los usuarios querrían pensar, los servidores de duplicación no cumplen rigurosamente con la integridad de los datos; tienden a hacerse de la vista gorda y arreglar los problemas después.

John Tibbetts y Barbara Bernstein
(Octubre de 1995)

Los paquetes comerciales estándar fuera de plataforma de base de datos distribuida ofrecerán, una vez que alcancen su pleno desarrollo, acceso transparente a los datos de una red. La base de datos distribuida registra la ubicación de los datos en la red, y enrutaría sus solicitudes a los nodos de base de datos correctos, con lo que volverá transparente su ubicación. Para ser plenamente distribuido, un servidor de base de datos debe soportar actualizaciones de sitios múltiples adoptando una disciplina de grabación en dos fases para transacciones (véase Figura 12-3). Esto debería permitirle unir datos de tablas que residen en varias máquinas. Asimismo, debería actualizar automáticamente datos de tablas que residen en varias máquinas.



En esta sección analizaremos la mecánica de la automatización total de este proceso de extracción (véase Figura 12-5). La mecánica del copiado tiene que ver con los siguientes asuntos: ¿cómo se especifica el extracto? ¿Cómo se mezclan datos de múltiples fuentes? ¿Los datos pueden transformarse como parte del copiado? ¿Quién orquesta el proceso de copiado? ¿Cómo se copian los datos en las bases de datos de información? ¿Cómo se regeneran las copias? ¿Qué tan estrechamente sincronizadas están las réplicas (o extractos) con la fuente? ¿En qué momento pueden actualizarse las réplicas? ¿Cuáles son las fronteras de transacciones de una copia?

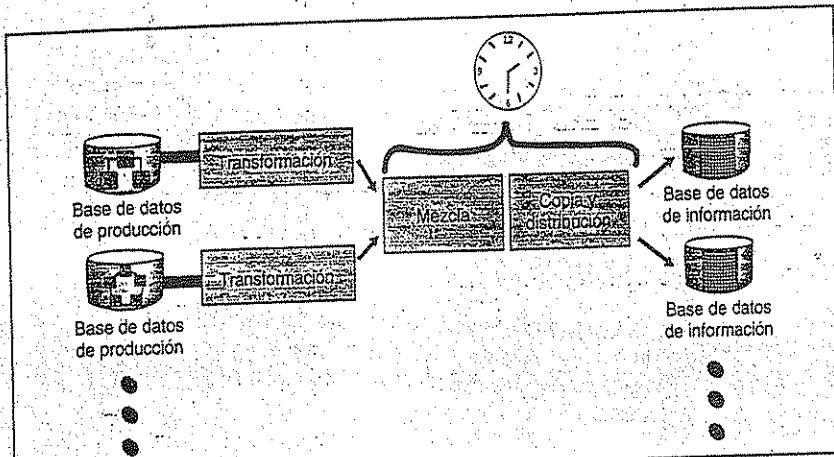


Figura 12-5. Proceso de duplicación y transformación de datos.

Regeneración y actualizaciones

Las bases de datos de información son ocupadas por datos originados en varias bases de datos de producción. Por lo general, los datos se copian o extraen por medio de una de dos técnicas:

- La *regeneración* consiste en el remplazo de la totalidad de los datos destino por datos de la fuente (véase Figura 12-6). Esto funciona adecuadamente cuando se manejan pequeñas cantidades de datos, de escasos requerimientos en cuanto a la frecuencia de actualización (lo que quiere decir que los datos tienen poca volatilidad). Este procedimiento también se emplea para realizar cargas pesadas iniciales en la base de datos destino.
 - La *actualización* se reduce al envío de los datos modificados al destino (véase Figura 12-7). Las actualizaciones pueden ser *síncronas*, lo que significa que la copia destino se actualiza en el mismo ámbito de grabación que la tabla fuente, o *asíncronas*, lo que significa que la tabla destino se actualiza en una transacción independiente de aquella de la fuente. Las actualizaciones síncronas son útiles en entornos de producción para la creación de bases de datos duplicadas que ofrecan alta disponibilidad. Las actualizaciones asíncronas son útiles en situaciones de almacenamiento de datos.

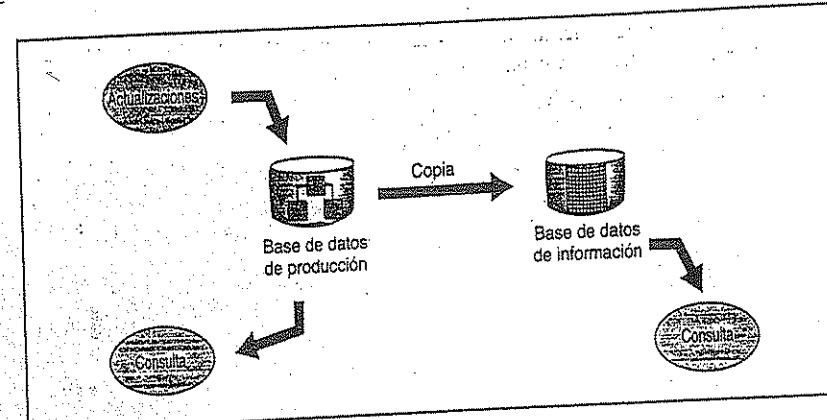


Figura 12-6. Duplicación por medio de regeneración

Usted especifica el nivel de sincronización que desea mantener entre la fuente y el destino por un lado y el intervalo de actualizaciones por el otro. Esto quiere decir que controla el nivel de obsolescencia de los datos que puede tolerar.

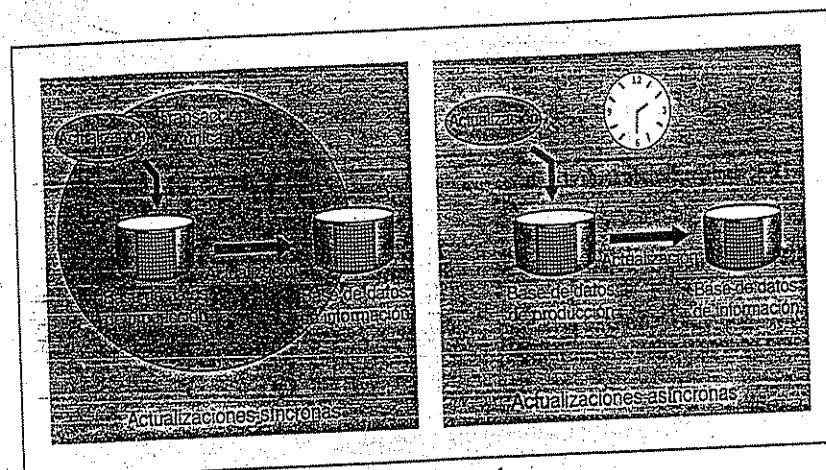


Figura 12-7. Duplicación a través de actualizaciones sincronas y asincronas.

Montaje de las actualizaciones

Algunos productos de bodega permiten controlar con toda precisión la frecuencia de las actualizaciones. Por ejemplo, se pueden especificar los intervalos para el envío de datos de *actualización asincrónica* de la fuente a uno o más destinos. Los cambios en las tablas fuente son retenidos en una o más *tablas de montaje* para la subsiguiente propagación a tablas destino (véase Figura



Además, distintas vistas de los mismos datos fuente pueden enviarse a diferentes destinos de copia. Las uniones de tablas múltiples también pueden emplearse para definir las transformaciones de copiado.

- Los *agregados* permiten transmitir únicamente agregaciones de datos como promedios, sumas, máximos, etc. También en este caso esto se especifica una vez con SQL, en tanto que la herramienta de copiado ejecutará el agregado cada vez que tenga lugar una transferencia.
- Las *funciones derivadas* permiten especificar datos inexistentes, pero que resultan de cierto cálculo (o función) a partir de los datos existentes. Por ejemplo, en la base de datos destino puede definirse una nueva columna de datos que sea la suma de dos columnas de la base de datos fuente. La nueva columna será creada y actualizada automáticamente como parte del proceso automatizado de copiado.

Además de limpiar y mezclar los datos, estas funciones pueden contribuir a la reducción del tráfico en la red entre fuentes y destinos, porque sólo se copian los datos que se necesitan.

Algunas de las herramientas de duplicación más sofisticadas permiten extraer y copiar datos de fuentes no relacionales, como Lotus Notes, Internet Web Servers, hojas de cálculo, abastecedores de noticias y bases de datos de CICS e IMS. La mayoría de las bodegas de datos son bases de datos de SQL, de manera que la mayor parte de las herramientas de extracción ofrecen transformaciones unidireccionales de las fuentes de datos extranjeras a SQL. Como cabe esperar, esto supone un alto grado de limpieza de datos. Casi todas las herramientas brindan retrollamadas que permiten aplicar a los extractos de datos funciones de limpieza personalizadas. Las transformaciones de datos suelen describirse en el directorio de información o en una herramienta de administración de bodega.

De acuerdo con Gartner Group, el mercado de herramientas para la extracción, limpieza y duplicación de datos crecerá de \$65 millones de dólares en 1994 a \$210 millones en 1999. Esto representa un aumento anual del 26 por ciento. Ejemplos de herramientas de extracción y duplicación de datos son *Passport* de Carleton, *Pipeline* de Platinum, *Data Mover* de Legent, *Replication Server* de Sybase, *Snapshot* de Oracle7, *OmniReplicator* de Praxis, *OpenIngres Replicator* de Ingres, *Entire Transaction Propagator* de Software AG, *Warehouse Manager* de Prism, *Extract* de ETI y las herramientas *DataJoiner*, *DataPropagator* y *DataRefresher* de IBM. Ejemplos de herramientas de transformación y limpieza son *InfoPump* de Trinziec, *Enterprise Integrator* (EI) de Apertus Technologies y *Copy Manager* de IBI.

Réplicas verdaderas

El tiempo es un concepto relativo en los sistemas duplicados, así que no recomendaría ningún tipo de esquema de detección de colisiones o resolución con base en él.

Glenn Froemming, DBMS Magazine
(Abril de 1996)

Las réplicas son *copias* de datos que usted puede actualizar (véase Figura 12-10). Cuando esto ocurre, la réplica actualizada debe encontrar la manera de resincronizar su estado con la base de datos primaria original. Normalmente, las actualizaciones sólo tienen lugar en una sola réplica designada y el servidor primario debe abstenerse de la realización de cambios generados fuera de la réplica. Así, el sitio de la actualización cambia de la base de datos primaria a la réplica.

Como resultado, la réplica comienza con una imagen completa de la base de datos primaria, a la cual le envía todas las actualizaciones subsecuentes, ya sea en forma continua o periódica. La base de datos primaria se encarga entonces de la propagación de los cambios que recibe a sus bases de datos destino, a través de los procesos normales. La restricción de actualización de sitio único puede relajarse con el empleo de versiones de *registro de salida* de datos duplicados, tecnología de amplio uso en las bases de datos de objetos.

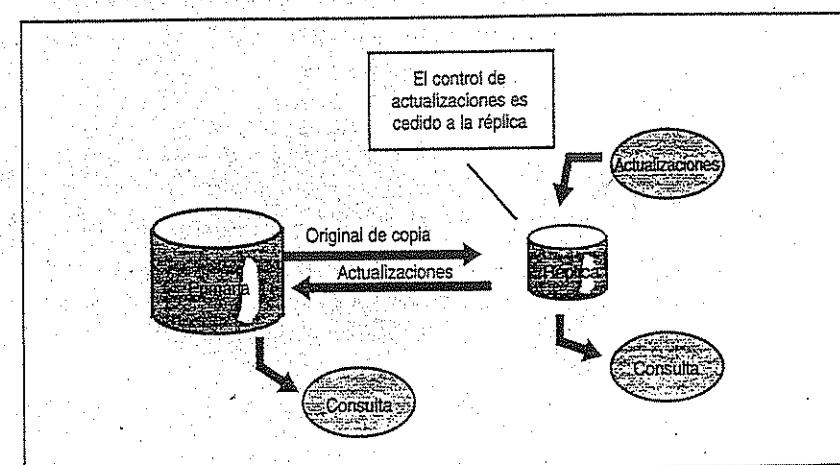


Figura 12-10. Réplicas, o conversión de las copias en datos maestros.

Ingres implementa una base de datos que permite que tanto los datos maestros como las réplicas sean actualizados por separado, y que emplea un *solucionador de conflictos* para sincronizar los datos. Cuando los cambios en la réplica se aplican a los datos maestros, un *detector de colisiones de actualizaciones* resuelve los conflictos mediante una de cuatro políticas especificadas por el usuario: prioridad de la actualización más antigua, prioridad de la actualización más reciente, aplicación de una acción especificada por el usuario o interrupción de todas las duplicaciones. Este último método puede ser el más adecuado, pero ¿es posible permitirle el paro total de la aplicación hasta que alguien concilie los datos? También se puede especificar el criterio prevaleciente por ubicación o intervalo de tiempo. Entre los participantes más recientes en la carrera de "datos en todas partes" están *Extract* de ETI, *OmniReplicator* de Praxis y el servicio *Symmetric Replication* de Oracle 7.1. Pero tenga cuidado: por auténticas que puedan ser, las réplicas bien podrían contaminar los datos.



crecimiento pronunciado están creando un mercado altamente competitivo. Los proveedores compiten por la oferta de productos con más funciones y mayor facilidad de uso. Las herramientas de EIS/DSS alcanzarán finalmente precios acordes con ventas de gran volumen. Cuando esto suceda, se convertirán sin duda en herramientas de todos los días, tal como ocurre ahora con las hojas de cálculo.

En nuestro ramo el crecimiento suele acompañarse de argumentos contrapuestos de proveedores, contraargumentos y una explosión de nuevas siglas. EIS/DSS no es la excepción. En la Figura 13-1 se muestra la evolución de las herramientas de EIS/DSS. Como puede verse, hemos introducido en ella una nueva y extraña terminología; esto es natural en los nuevos territorios. Esta figura le permitirá darse cuenta que las herramientas de EIS/DSS se vuelven cada vez más inteligentes (o al menos que no se han quedado quietas). El santo grial del sector de las herramientas de EIS/DSS es la consulta "sin intervención". Esto significa que en un momento no muy lejano podremos sentarnos cómodamente y enviar a nuestros agentes de información personal —armados con algoritmos de investigación— al cumplimiento de misiones de descubrimiento de datos. Claro que todavía falta mucho para que lleguemos a ello. Este capítulo es una descripción general del estado de las herramientas de EIS/DSS. Comenzaremos por herramientas de consulta y análisis simples. Pasaremos después a OLAP, término genérico para designar a las herramientas que permiten visualizar datos desde múltiples perspectivas (o dimensiones). El paso siguiente en esta escala evolutiva son las minas de datos, donde los propios datos nos dirán cosas de sí mismos, lo que nos descubrirá patrones ocultos. En última instancia, hay agentes de información personal que se parecen al asistente de datos de Danny De Vito.

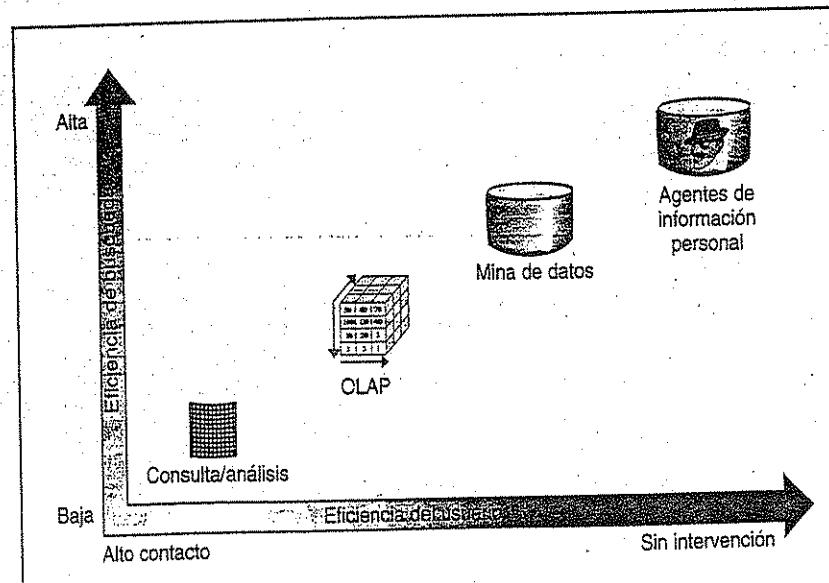


Figura 13-1. Evolución de las herramientas de EIS/DSS.



HERRAMIENTAS DE CONSULTA/REPORTE

La consulta de RDBMS usual le dirá qué, no por qué.

Christine Comaford, columnista, PC Week
(Febrero de 1996)

Las herramientas de consulta y análisis de datos permiten formular una consulta sin tener que generar un programa ni aprender SQL. Usted sencillamente apunta y hace clic para generar las instrucciones de selección (SELECT) y los criterios de búsqueda (la cláusula WHERE). La herramienta despliega entonces los resultados en forma comprensible, por lo general una tabla. En su mayoría, las herramientas de consulta ofrecen gráficas comunes de negocios —como gráficas de barras, gráficas de pastel, histogramas y gráficas lineales— que permiten proyectar los datos resultado de una consulta. Casi todas las herramientas permiten asimismo correr la consulta y llenar una modalidad de hoja de cálculo a partir de los resultados. Algunas permiten correr la consulta y explorar después geográficamente los resultados con el empleo de un paquete de mapas. La mayoría de las herramientas brindan facilidades de creación de scripts para la planificación automática de la ejecución de consultas y reportes.

Las mejores herramientas de consulta cuentan con dispositivos de seguridad para impedir la presentación de "consultas desbocadas". Estas consultas entregan conjuntos de resultados muy extensos. El dispositivo de seguridad más común sirve para que la herramienta proporcione un *gobernador de consultas* que anule una de éstas en caso de que la ejecución sea excesivamente prolongada o de que entregue un número excesivo de filas. Algunos gobernadores guardan grandes conjuntos de resultados en el servidor y se los entregan al cliente en incrementos fijos. Otros exhiben un cuadro diálogo en el que se indica el número de filas que se obtendrá de una consulta; si el número es exagerado, se puede hacer clic en el botón "Cancel" (cancelar).

Ejemplos de herramientas de consulta y análisis de datos son *Quest* de Gupta, *Forest & Trees* de Trinzie, *SAS System* del SAS Institute, *Visualizer* de IBM, *DataPrism* de Brio, *Q+E* de Intersolv, *Focus* de IBI, *Esperant* de Software AG y *Impromptu* de Cognos.



Herramientas para la consulta de BLOB

Información

La novedad más reciente en herramientas de consulta es la *consulta de BLOB*, por la cual es posible buscar patrones dentro de los campos de BLOB de una base de datos. Como se recordará, un BLOB es un tipo de datos de SQL-92 que sirve como contenedor de grandes documentos de texto o datos con multimedia. La buena noticia es que ahora disponemos de herramientas para la consulta del contenido de BLOB (al menos de aquellos que contienen imágenes). Estas herramientas permiten que usted le ordene al DBMS: "Busca todas las imágenes parecidas a ésta".

Una herramienta de consulta de BLOB permite buscar imágenes en una bodega de datos por color, textura, forma y posición. Se pueden crear consultas visuales manipulando



base de datos especializados para almacenar datos en matrices a lo largo de dimensiones relacionadas llamadas *hipercubos*. La mayoría de los MDBMS emplean esquemas de indexación intensiva para optimizar el acceso a estos cubos. Los proveedores de MDBMS suelen suministrar herramientas de OLAP del cliente que han sido optimizadas para sus mecanismos; no existen estándares para acceso a datos de MDBMS. Entre las herramientas de OLAP/MDBMS están *Essbase* de Arbor, *Accumate* de Kenan, *Lighthip Server* de Pilot, *TM/I* de Sinper, *Express* de Oracle/IRI, *CrossTarget/Diver* de Dimensional Insight, *OLAP++* de SAS, *DecisionSuite Server* de Information Advantage y *Holos* de Holistic System. *Commander OLAP* de Commshare es una herramienta de primer plano que usa el mecanismo de MDBMS de Essbase.



OLAP: ¿A RDBMS o a MDBMS?

Debate

El problema es que, ocultamente, la mayoría de los servidores de OLAP extraen datos de un DBMS bídimesional y pueblan un cubo de datos multidimensional.

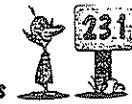
Christine Comaford, columnista

PC Week

(Febrero de 1996)

Todo mundo coincide en que OLAP es una maravilla. Pero la pregunta del millón es: ¿necesita usted un MDBMS para obtener todos los beneficios de OLAP? ¿O puede hacerse de ellos con su antiguo y confiable RDBMS? Los defensores de MDBMS sostienen que el cambio se impone. Exponen los siguientes argumentos:

- **MDBMS permite ahorrar dinero.** En teoría, los MDBMS requieren de menos espacio en disco, porque emplean matrices dispersas para almacenar datos en todo número de dimensiones, y a menudo relacionan los datos mediante un lenguaje de cálculo implantado. En consecuencia, hacen un mejor uso del hardware básico al ofrecer software plenamente optimizado para OLAP. En contraste con ello, los proveedores de RDBMS adoptan el método de "fuerza bruta", pues sencillamente pretenden resolver el problema con costoso hardware masivamente paralelo; por ejemplo, Himalaya de Tandem, SP2 de IBM y GIS 3600 de AT&T (es decir, Teradata).
- **Las consultas de MDBMS son un orden de magnitud más rápido.** Los proveedores de MDBMS le enseñarán resultados de evaluaciones comparativas que lo dejarán pasmado.
- **Oracle se rindió al comprar una compañía de MDBMS.** En junio de 1995, Oracle compró IRI, fabricante de Express MDB, por \$300 millones de dólares. Los promoto-



res de MDBMS consideran que esta compra fue el reconocimiento —nada menos que por el líder del mercado— de que los MDBMS son mejores para OLAP que los RDBMS.

Por supuesto que ninguno de estos argumentos le parece de suficiente peso al ramo de RDBMS, el cual se apresura a señalar que:

- **Los datos de MDBMS tienen que cargarse para poder realizar una consulta.** Las consultas de MDBMS se retardan considerablemente si antes debe cargarse un gigabyte de datos desde una fuente de RDBMS (la bodega).
- **Los MDBMS no han sido estandarizados.** En la actualidad se carece aún de estándares de MDBMS.
- **Los MDBMS carecen de arquitectura de servidor.** Casi todos ellos son sistemas de usuario único. No disponen de infraestructura de servidor multiusuarios ni del correspondiente middleware cliente/servidor.
- **Los MDBMS no se prestan a consultas específicas.** Sólo operan con hipercubos predefinidos.
- **Los RDBMS están introduciendo nuevas funciones de consulta.** Se les adecuó tradicionalmente a OLTP, en el que la transacción promedio se compone de unas cuantas llamadas a la base de datos. Sin embargo, a causa del extraordinario crecimiento del almacenamiento, los proveedores de DBMS se hallan en la carrera de introducir mayores capacidades de consulta. Ahora sus bases de datos deben ser capaces de manipular consultas que emitan cientos de llamadas a la base de datos, algunas de las cuales necesiten explorar quizás muchos gigabytes de datos.

Los proveedores de DBMS afirman estar ya en condiciones de vencer estos desafíos. Por ejemplo, IBM lanzó con DB2 V2 un nuevo optimizador de consultas fundamentado en costos y basado en su investigación "Starburst". Sybase complementó su Sistema 11 con *Sybase IQ*, que introduce una estructura de indexación de mapas de bits de carga baja que permite la total indexación de una base de datos íntegra. Por su parte, Oracle debió lanzar la indexación de mapas de bits y un nuevo optimizador de consultas paralelas con Oracle 7.3 en 1996. Si estos productos no le satisfacen, siempre hay Red Brick.

Entonces, ¿quién tiene la razón? No se trata obviamente de una cuestión de blanco y negro. Tanto los proveedores de MDBMS como de RDBMS adecuarán sus productos y harán todo lo que sea necesario por montarse a la ola de OLAP. No obstante, tal como lo descubrieron amargamente los proveedores de bases de datos de objetos (ODBMS), los RDBMS son hoy por hoy los "reyes de la colina" y no será fácil que una sola tecnología, por superior que sea, los desplace. Tal vez lo mejor será optar por la coexistencia. Por ejemplo, los MDBMS pueden convertirse en aditamentos de la estructura de RDBMS. Sin embargo, ya se ve cómo esto está terminando en la fusión IRI/Oracle; Oracle se halla en proceso de incluir el MDBMS. Si tiene éxito, éste pasará a ser sencillamente un nuevo tipo de datos *multidimensionales* dentro de RDBMS. □



Las herramientas de EIS/DSS, OLAP y las bases de datos multidimensionales son las nuevas y más atractivas tecnologías de embodegamiento de datos. Nuevas empresas entran y salen de este mercado, el cual está aún por establecerse. Lo más que pudimos hacer en este capítulo fue ofrecerle una estructura para el examen de estas nuevas tecnologías, buen primer paso hacia la comprensión de lo que pueden hacer por usted. Sería aún muy prematuro señalar ganadores; la función apenas empieza.

Capítulo 14

Bases de datos: Presentación de participantes



Hemos atestiguado enormes cambios en el mercado de DBMS, y suponemos que los seguirá habiendo hasta fines del siglo. Se espera que los grandes proveedores crecerán aún más, surgirán nuevos participantes en los nichos marginales, algunos de los mayores proveedores tendrán problemas y todavía aparecerán nuevos proveedores.

Gartner Group
(Febrero de 1995)

En los últimos capítulos hemos mencionado productos de bases de datos a diestra y siniestra sin presentarle formalmente a los participantes. Lo que ocurre es que deseábamos darle una idea de esta tecnología antes de introducirlo a los productos. En este capítulo ofreceremos una visión panorámica del mercado de base de datos y expondremos las tendencias actuales. Después nos ocuparemos formalmente de los principales participantes en este campo, con muchos de los cuales ya estarás familiarizado a estas alturas.

EL MERCADO DE BASES DE DATOS DE CLIENTE/SERVIDOR

De acuerdo con IDC, los ingresos del mercado de bases de datos de SQL fueron en 1995 de \$5 900 millones de dólares y seguirán creciendo a una tasa del 30% anual. Menos del 10%



Gartner Group prevé que en el año 2000 los "cinco grandes" proveedores de bases de datos serán IBM, Informix, Microsoft, Oracle y Sybase.² En segundo término estarán Computer Associates, a causa de Ingres; Progress Software, debido a su penetración vertical de mercado, y NonStop SQL de Tandem, por su facilidad de ampliación superior al 99% al pasar de 16 a 64 procesadores. Las previsiones de Gartner al año 2000 parecen indicar que el mercado de bases de datos se consolidará en torno a los triunfadores actuales. En las siguientes secciones expondremos lo que estos proveedores ofrecen hoy en día.

Oracle

Oracle volvió a estar en mente de todos en 1995 con el lanzamiento de su base de datos *Oracle7.1*, con capacidad para más de 90 plataformas (60 de las cuales son de Unix). Oracle brinda un entorno muy consistente entre todas ellas, desde herramientas e interfaces de administración hasta lenguaje de definición de datos (DDL: *data definition language*) y directorio de SQL. Se beneficia para ello de SMP sobre NetWare, OS/2, NT, AIX y SCO. En el extremo superior, soporta bases de datos masivamente paralelas con hardware especializado como SP2 de IBM (con acoplamiento holgado con RS/6000) y GIS 3600 de AT&T (también conocido como Teradata). En el extremo inferior, Oracle ha anunciado ya su *Personal Oracle7* y *Oracle7 Workgroup Server*.

Soporta actualmente un optimizador de consultas paralelas, duplicación bidireccional de datos y características de base de datos distribuida. Oracle7 soporta BLOB, procedimientos almacenados y desencadenantes. *Oracle7.3*, introducido a principios de 1996, mejora el desempeño de embodegamiento de datos y ofrece nuevos aditamentos para medios y texto; incluye también un servidor integrado de HTTP y un mejor mecanismo de autenticación. Entre las opciones de aditamentos están un servidor de correo electrónico, OLAP y un servidor de video. *Oracle8*, esperado para 1997, soportará extensiones de objetos basadas en SQL 3. Incluirá asimismo *Project Sedona*, un "servidor de componentes" que soporta objetos de CORBA, OLE, SQL 3 y Java. Se trata de una arquitectura de objetos de cliente/servidor en 3 planos. Los componentes del servidor de Sedona —ú objetos de negocios— podrán atender a clientes en correderos de objetos tanto CORBA como OLE; los objetos almacenarán su estado en la base de datos de Oracle8. Sedona incluirá un depósito de componentes y una herramienta de desarrollo visual.

Familia DB2 de IBM

IBM renovó en 1995 su familia de bases de datos relacionales con la introducción de *DB2 2.1*, también conocido como "Common Server". Se trata de la versión exportable de DB2, que corre en OS/2, NT, AIX, HP-UX, Solaris y otros Unix. El Common Server DB2 soporta extensiones de objetos avanzados, ODBC, CLI, desencadenantes, BLOB, SMP (en caso de que esté disponible) y un optimizador avanzado de consultas basado en costos. Las versiones para macrocomputadora y AS/400 de DB2 —que por lo pronto no forman parte de la base de código del DB2 Common Server— también fueron perfeccionadas para soportar más paralelismo, un optimizador de consultas más inteligente y procedimientos almacenados.



IBM también introdujo en 1995 *DB2 Parallel Edition for SP2*, un servidor de base de datos paralelo de nada compartido. En el nivel más bajo, IBM ofrece en la actualidad la base de datos de SQL Approach de Lotus, con características de creación de scripts, asistentes y un primer plano fácil de usar semejantes a Basic. Además, DB2 para OS/2 tiene ventas anuales por alrededor de 200 000 copias.

Informix

Informix es la compañía de bases de datos de crecimiento más rápido en el mercado de Unix, con ventas anuales de licencias de bases de datos a un crecimiento del 50%. Tras haberse iniciado como proveedor de servidores departamentales de revendedores de valor agregado (VAR: *value added reseller*), con un producto de extremo inferior llamado *Standard Edition*, en los dos últimos años esta compañía dirigió su atención al nivel más alto del mercado de bases de datos y rearquitecturó su base de datos nuclear para adaptarla al paralelismo. Generó 800 000 líneas de nuevo código. El resultado fue *Informix 7.1*, que soporta SMP en grupo. A principios de 1996 Informix introdujo un producto de bases de datos masivamente paralelas, *Informix 8.0 XPS*, que corre en SP2 de IBM, GIS 3600 de AT&T y Goldrush de ICL/Fujitsu.

Informix soporta procedimientos almacenados, desencadenantes, BLOB y la parafernalia usual de SQL. Su punto fuerte es OLTP. Sin embargo, con su recién descubierta facilidad de ampliación, se halla ahora en una buena posición para competir eficazmente en el lucrativo mercado de almacenamiento de datos.

Sybase

Sybase volvió al mercado a fines de 1995 con el lanzamiento de *System 11*. La versión anterior de este producto, *System 10*, fue un fracaso comercial. Su desempeño era deficiente y no podía ampliarse a más allá de cuatro máquinas. En consecuencia, Sybase se excluyó del lucrativo mercado de SMP. Peor aún, sus clientes votaron contra su sistema por el solo hecho de no adóptarlo. Pero todo ha cambiado con *System 11*, cuya capacidad para SMP ha obtenido excelentes comentarios. Este producto corre en la actualidad en DEC Unix, HP-UX, AIX, Solaris y Windows NT. *System 11* se amplía al nivel más alto con *Sybase MPP*, que sirve como mecanismo de base de datos de información de nivel más alto sobre hardware masivamente paralelo, incluidos GIS 3600 de AT&T, SP2 de IBM y servidores SPARC de Sun.

Sybase fue pionera en procedimientos almacenados y desencadenantes. Soporta BLOB y muchas características vanguardistas de SQL. Con su nuevo *System 11*, esta compañía ha incursionado en el almacenamiento de datos, sin abandonar el terreno de OLTP, en el que tradicionalmente se ha desempeñado tan bien. Participó recientemente en una aventura de adquisición. Compró tanto PowerSoft como el mecanismo de base de datos de Watcom, al que rebautizó como *SQL Anywhere*. Empleará esta esbelta pero poderosa base de datos de escritorio para establecer su presencia en el nivel más bajo del mercado. También se está expandiendo en el sector del middleware y añadiendo capacidades de MOM a sus bases de datos de SQL.

² Fuente: Gartner Group, "The Future of Databases" (9 de agosto de 1995).

■ MATISSE es una base de datos de objetos/relacional con auténtica facilidad de ampliación. El mecanismo de MATISSE ha sido excepcionalmente adaptado para soportar grandes BLOB. Puede manipular objetos de hasta 264 gigabytes de tamaño y circularlos casi a las velocidades del disco nativo. El mecanismo también es sensacional cuando se aplican evaluaciones comparativas de TPC-C de SQL ordinario. De este modo, puede disponerse de una base de datos de objetos capaz de ejecutar SQL y BLOB al mismo tiempo. Esta combinación única hace de MATISSE una de las mejores bases de datos implantadas para el Web.⁴

Además de los tres que hemos elegido, cientos de productos innovadores buscan sus nichos en el muy lucrativo mercado de las bases de datos en línea. La explosión de los nuevos tipos de medios —especialmente en el Web— ha dado lugar a nuevas y muy jugosas oportunidades. Claro que los "cinco grandes" no pueden renovar sus mecanismos de la noche a la mañana para soportar los nuevos tipos de datos; antes deben satisfacer las necesidades de su base instalada. Como resultado de ello, están adquiriendo a diestra y siniestra a quienes participan en los nichos, con el propósito de llenar los vacíos. En la carrera por el liderazgo en el ámbito de las bases de datos, todo puede suceder.

Capítulo 15

Bodegas de datos: Prism, IBM y Sybase



Las bodegas de datos se convertirán en un imperativo estratégico, no de lujo, para la mayoría de las organizaciones.

Gartner Group
(Agosto de 1995)

Con más de 15 "estructuras" de bodegas de datos y 100 productos comerciales entre los cuales elegir, ¿por cuáles optar en un capítulo dedicado a estos productos? Las tres estructuras por las que nos hemos decidido son: *Warehouse Manager* de Prism, *Information Warehouse* de IBM y *Warehouse Works* de Sybase.¹ Prism fue fundada por W.H. Inmon, "padre del almacenamiento de datos". Ofrece un producto de base de datos probado con características muy atractivas. IBM introdujo en 1991 la primera estructura de bodega de datos, y lanzó también, en 1993, la primera línea completa de productos de bodega de datos. A fines de 1995 introdujo dos innovadores productos de almacenamiento: *DataJoiner* y *Visual Warehouse*. Finalmente, Sybase representa una propuesta de un producto de almacenamiento de datos neto completamente ajustado.

⁴ Advertencia: Uno de los autores de este libro es alto ejecutivo de ADB MATISSE. Sin embargo, jura que todo esto es cierto.

¹ "Estructuras" (*frameworks*) es ya un término muy sobrecargado. En este caso significa una colección de productos y sus protocolos comunes.



INFORMATION WAREHOUSE DE IBM

Information Warehouse de IBM prácticamente ofrece una solución integrada de almacenamiento de datos de arriba abajo. La solución de IBM incluye la base de datos de información, las herramientas de administración de duplicaciones y copiado, un catálogo de información, un datamart, herramientas de EIS/DSS y una herramienta de administración de sistemas integrados. Prové también un mecanismo de flujo de trabajo que permite automatizar aún más el proceso de almacenamiento e integrarlo con el resto de la empresa. IBM publica las API para sus diversas herramientas de almacenamiento de datos, entre ellas de EIS/DSS, duplicación, directorio de información y administración del sistema. Como resultado de ello, contribuyó a la creación de un mercado de herramientas de terceros que opera dentro de la estructura de IBM.

En la Figura 15-2 se muestran los diversos productos de IBM que componen la solución de almacenamiento de datos de esta compañía. He aquí una breve descripción de lo que hacen estos productos:

- **DataPropagator Relational** brinda duplicación con agregación y limpieza de datos entre los miembros de la familia DB2, tales como DB2 sobre OS/2, MVS, AIX, VM, OS/400, HP-UX, Solaris y Windows NT. La bodega destino puede suscribir actualizaciones o reacciones totales y elegir también el momento de su aplicación. DataPropagator posee una excepcional arquitectura en dos partes compuesta de la siguiente manera: 1) un componente de *montaje*, cuya ejecución se realiza en el punto de los datos fuente, y 2) un componen-

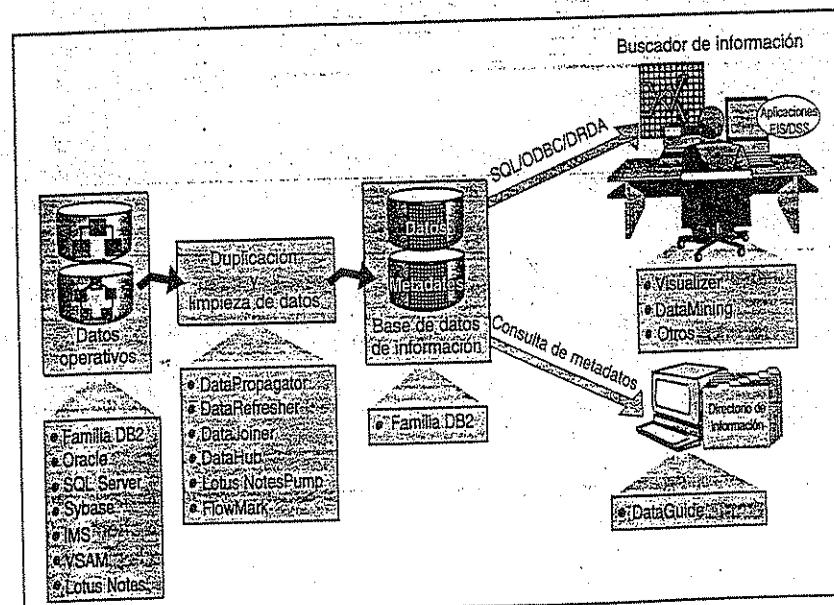


Figura 15-2. Estructura de Information Warehouse de IBM.



te de *aplicación*, con ejecuciones ya sea en el punto fuente o destino. El área de montaje es sencillamente un registro (o cola) de la base de datos relacional de cambios a los datos fuente; puede poblarla de diferentes fuentes de datos (véanse los puntos siguientes). El componente de aplicación traslada los cambios del área de montaje al destino en intervalos definidos por el usuario. La aplicación puede transformar los datos montados antes de copiarlos mediante agregaciones, uniones o cualquier operación de SQL.

- **DataPropagator NonRelational** extrae datos de IMS y los traslada al área de montaje de DataPropagator. Éste duplica entonces los datos del área de montaje para sus suscriptores, las bodegas destino.
- **DataRefresher** extrae datos de VSAM y archivos planos y los traslada al área de montaje de DataPropagator. Éste duplica entonces los datos del área de montaje para las bodegas destino suscriptoras.
- **DataJoiner** accede a y une transparentemente datos de una amplia variedad de fuentes relacionales y no relacionales, como Sybase, Oracle, SQL Server, productos de la familia DB2, IMS, VSAM y cualquier fuente de datos observante de ODBC. Los datos extraídos son conducidos al área de montaje de DataPropagator para su duplicación para bodegas destino suscriptoras.
- **Lotus NotesPump** extrae datos de bases de datos de Notes y los conduce al área de montaje de DataPropagator para su duplicación para bodegas destino suscriptoras.
- **DataHub** es un producto de OS/2 que ofrece un solo punto de administración para una bodega de datos. Además de permitir la administración de la duplicación de datos, brinda un entorno uniforme de "arrastrar y soltar" para la administración de bases de datos heterogéneas, en su mayoría de la plataforma múltiple de la familia DB2. Se pueden desplegar objetos de base de datos, administrar autorizaciones y correr utilerías de bases de datos. Esta herramienta proporciona un creador de itinerarios para automatizar la ejecución de tareas en momentos específicos. Con la aparición de la versión 2, un nodo de DataHub puede participar en la administración global generando alertas de SNMP en beneficio de los recursos que administra.
- **FlowMark** es un producto de administración de flujo de trabajo que le permite automatizar aún más la bodega al administrar la ejecución de procesos de almacenamiento de múltiples pasos. Por ejemplo, usted podría emplear FlowMark para realizar las siguientes tareas: 1) planear una captura de datos de fuentes múltiples, 2) copiar los resultados en una bodega, 3) ejecutar un conjunto de consultas de OLAP, 4) formatear un reporte y 5) enviar los resultados por correo electrónico a una lista de distribución.
- Los productos de la *familia DB2* aportan el mecanismo de SQL para la base de datos de información. DB2 corre en múltiples plataformas, como OS/2, NT, diversas modalidades de Unix, OS/400 y macrocomputadoras. DB2 en AIX y DB2 en MVS ya ofrecen soporte de consultas y uniones paralelas. Estas dos funciones deberían contribuir a acelerar las búsquedas en grandes entornos de almacenamiento de datos. Al traspasar una bodega ciertas dimensiones, se necesitará ya sea una base de datos de SQL masivamente paralela o un mecanismo de bases de datos multidimensional especializado para la ejecución de consultas de OLAP intensivo.

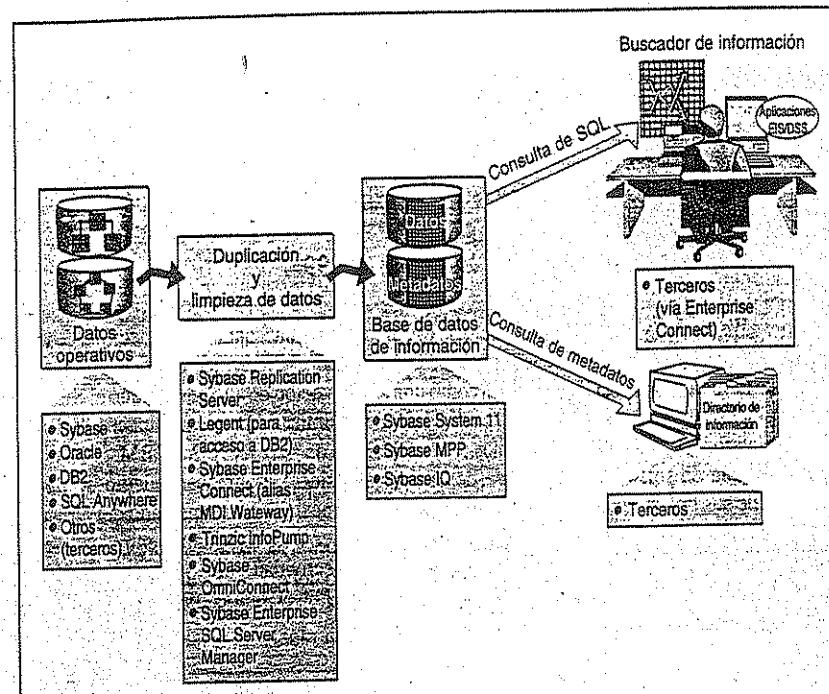


Figura 15-3. Estructura de Warehouse Works de Sybase.

- *Trinzie InfoPump* fue integrado recientemente al Sybase Replication Server para ofrecer transferencias de gran volumen y limpieza de datos. Aporta un lenguaje de creación de scripts que permite definir transformaciones de datos como parte del proceso de transferencia de gran volumen. Los scripts suelen ser invocados por desencadenantes cuando los datos se modifican.
- *Sybase Enterprise SQL Server Manager* es la herramienta de administración de sistemas distribuidos más reciente de Sybase; incluye un administrador del servidor duplicado. Una consola de GUI le permite a usted rastrear objetos de duplicación, vigilar el desempeño, probar conexiones en red y proceder tras las alertas. Sybase incorporó en su herramienta protocolos de administración basados en Tivoli, lo que la abre a terceros.
- *Sybase System.II* brinda el mecanismo de base de datos de información de nivel más bajo sobre uniprocesadores y máquinas de SMP. Corre actualmente en DEC Unix, HP-UX, AIX, Solaris y Windows NT.
- *Sybase MPP* sirve como mecanismo de base de datos de información de nivel más alto sobre hardware masivamente paralelo, como GIS 3600 de AT&T, SP2 de IBM y los servidores SPARC de Sun.



■ *Sybase IQ* aumenta al optimizador de consultas de la base de datos con capacidades de indexación avanzadas que han sido optimizadas para búsquedas específicas. Sybase sostiene que las consultas específicas con IQ son de un orden de magnitud más veloz. IQ emplea una estructura de indexación apta para bits de carga baja que permite indexar por completo una base de datos. La idea es nunca tener que hacer una exploración de tabla lineal una vez que todo ha sido completamente indexado. Asimismo, no se tiene que ocupar de la predefinición de vías de consultas para el mecanismo de búsqueda.



Todavía hay dinero por hacer en bases de datos

Advertencia

El almacenamiento de datos es evidentemente un mercado explosivo, y 1995 fue el primer año en mostrar ingresos sustanciales en ese mercado.

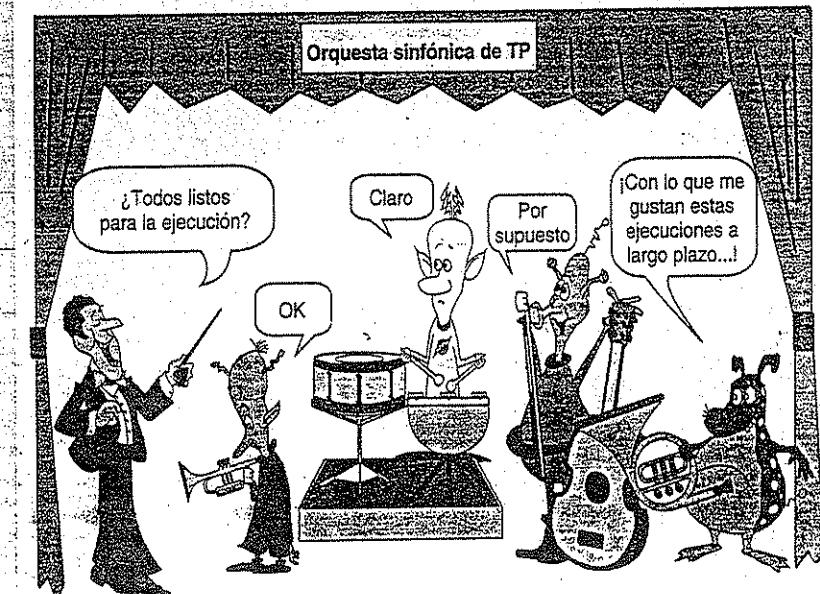
Dennis McEvoy, vicepresidente
de Products Group
Sybase
(Octubre de 1995)

Con esto concluimos nuestra introducción conceptual a los servidores de bases de datos de SQL. Fue un largo recorrido, como es de esperarse de una tecnología que representa a la mayoría de las aplicaciones cliente/servidor actualmente en producción. La tecnología de base de datos se encuentra aún en sus inicios. Los servidores de bases de datos de SQL se están convirtiendo en productos comerciales y aprendiendo a coexistir en acomodos de bases de datos en federación. Los sistemas de bases de datos decisivos para el cumplimiento de objetivos seguirán vendiéndose en paquetes de soberbia integración entre hardware escalable tolerante de fallas y software. Las nuevas áreas de crecimiento se encontrarán en los mercados masivos de productos orientados a bases de datos. ¿Qué significa esto para la instalación de una bodega de datos en cada escritorio y laptop móvil? ¿Quién mantendrá alimentadas estas bodegas con información continua en tiempo real? ¿Cómo se desempeñarán las bodegas de datos en el Web? ¿Qué herramientas nos ayudarán a clasificar toda esta información en tiempo real?

Sí, todavía están por hacerse fortunas en la tecnología de base de datos. Las compañías de bases de datos —tales como Sybase, Oracle, Gupta, Informix, Tandem, IBM San José e Ingres— están transformando a Silicon Valley (donde viven los autores de este libro) en "Database Valley". Y lo han logrado con apenas parte de la tecnología relacional y de SQL de las investigaciones de IBM. Prevemos (pues nos hallamos justamente en un recuadro de debate) que la fusión de "bodegas de datos" y "carreteras de la información" generará en las bases de datos oportunidades que harán parecer minúsculas a todas las que hemos presenciado hasta ahora. De modo que lo mejor está aún por llegar. Ciertamente creemos que nuestro valle se llamará Database Valley para fines de esta década. □

Parte 5

Procesamiento de transacciones de cliente/servidor



Capítulo 16

La magia de las transacciones



La idea de sistemas distribuidos sin administración de transacciones es como una sociedad sin una ley contractual. Las leyes pueden no gustarnos, pero necesitamos un medio para resolver las cosas cuando surgen diferencias. A nada se aplica mejor esto que a los mundos de PC y cliente/servidor.

Jim Gray (Mayo de 1993)¹

Las transacciones son más que simples eventos de negocios: se han convertido en toda una filosofía de diseño de aplicaciones que garantiza robustez en los sistemas distribuidos. Bajo el control de un monitor de TP, una transacción puede manejarse desde su punto de origen —habitualmente en el cliente— y a lo largo de uno o más servidores, para volver después al cliente originario. Al terminar una transacción, todas las partes involucradas están de acuerdo en cuanto a si fue un fracaso o un éxito. La transacción se convierte en el contrato que liga al cliente con uno o más servidores.

En este capítulo abordaremos primeramente las propiedades ACID que hacen de las transacciones bienes tan deseables en la computación de cliente/servidor. Después explicaremos las tra-

¹ Fuente: Jim Gray, "Where is Transaction Processing Headed?", en OTM Spectrum Reports.

MODELOS DE TRANSACCIONES

¿En qué momento debe comenzar una transacción? ¿Cuándo debe terminar y poner sus efectos a disposición del mundo exterior? ¿Cuáles son las unidades adecuadas de recuperación en caso de fallas? ¿Las transacciones de cómputo pueden ser reflejo de sus contrapartes en la realidad? Para contestar estas preguntas, analizaremos la *transacción simple*, describiémos sus limitaciones y veremos rápidamente las extensiones propuestas.

¿Qué es una transacción simple?

Las transacciones simples son los caballitos de batalla de la actual generación de sistemas para transacciones. Se les llama simples porque todo el trabajo que se lleva a cabo dentro de los límites de una transacción ocurre al mismo nivel (véase el área sombreada de la Figura 16-1).

La transacción comienza con *inicio_transacción* (*begin_transaction*) y termina ya sea con *grabación_transacción* (*commit_transaction*) o *aborts_transacción* (*abort_transaction*). Es una propuesta de todo o nada; es imposible grabar o abortar partes de una transacción simple. Todas las acciones son indivisibles, que es lo que queríamos. En la Tabla 16-1 se comparan los comandos usados en diferentes monitores de TP para delinear los límites de la transacción.

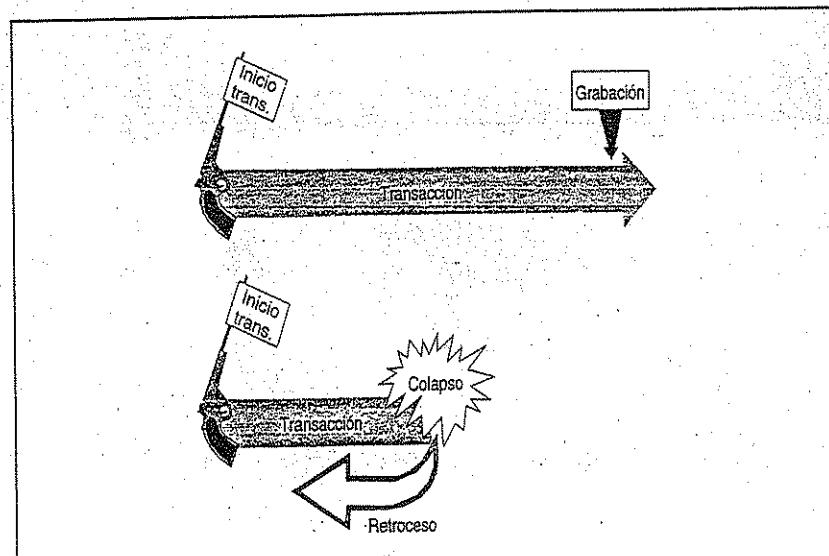


Figura 16-1. Transacción simple: Una propuesta de todo o nada.

Tabla 16-1. Comparación de delimitadores de transacciones simples de importantes monitores de TP
(Adaptada de OTM Spectrum Reports, febrero de 1993).

Sistema	Delimitador de transacción		
	Arranque	Grabación	Aborto
Tuxedo	TPBEGIN	TPCOMMIT	TPABORT
Top End	tx_begin	tx_commit	tx_rollback
RPC de Encina	transaction	onCommit	onAbort
X/Open	tx_begin	tx_commit	tx_rollback
TP de OSI	C-BEGIN	C-COMMIT	C-ROLLBACK
RSC de Tandem	Begin_Transaction	End_Transaction	Abort_Transaction
CICS	SYNCPOINT	SYNCPOINT	SYNCPOINT o ROLLBACK

Nos gustan nuestras transacciones simples

Debate

La mayor virtud de la transacción simple es su *simplicidad* y la facilidad con la que ofrece las características ACID. Miles de aplicaciones comerciales fueron creadas con base en el sencillo concepto de las transacciones simples. Históricamente, el desarrollo de la transacción simple se debió a su utilidad para aplicaciones bancarias; resulta excelente para modelar actividades breves.

Pero a medida que la disciplina de las transacciones ha ido permeando todas las facetas de la computación, hemos descubierto que el modelo de la transacción simple no es el más adecuado en todos los entornos. Se han escrito millones de líneas de código para compensar sus deficiencias. El modelo es particularmente frágil en lo que se refiere al manejo de transacciones de negocios que se extienden por prolongados períodos, de días o incluso meses. Es relativamente débil en labores por lotes. Y es una nulidad en situaciones que requieren de vueltas atrás parciales sin tener que eliminar el trabajo de una transacción completa; la rígida aplicación de "todo o nada" del principio ACID se interpone en el camino.

Por razones políticas, las transacciones simples en las que se hace uso de grabaciones en dos fases por lo general tienen vedado el acceso por fronteras interempresariales; quizás el MOM asíncrono sea un método preferible en estas circunstancias. Con MOM se pierde la protección ACID de extremo a extremo a cambio de la relajación de la sincronización estricta, impuesta por un protocolo global de grabación en dos fases. También, estamos experimentando dificultades con el modelo simple en entornos de cliente/servidor en los que el "tiempo para pensar" del cliente forma parte del circuito de transacción. Existen maneras de librarse cada uno de estos problemas, pero para ello se requiere generar un código personalizado. ¡No sería maravilloso que se pudiera extender el modelo de la transacción para que se hiciera cargo de todas estas situaciones automáticamente en lugar nuestro?



sita. El monitor de TP de cada ubicación debe administrar la parte que le corresponde de la transacción. Dentro de una ubicación, el monitor de TP coordina las transacciones con los subsistemas ACID y administradores de recursos locales, incluidos administradores de bases de datos, administradores de colas y transportes de mensajes. Por ejemplo, el monitor de TP garantizará que cuando una base de datos sea actualizada, se emita un mensaje y se haga un registro en una cola de flujo de trabajo. Todas estas acciones ocurrirán (exactamente una vez), o bien no ocurrirá ninguna. Además, uno de los monitores de TP debe coordinar las acciones de los demás monitores de TP. Todo esto se lleva a cabo con el empleo de un protocolo de *grabación en dos fases*, que coordina la grabación o el aborto de la transacción a través de sitios múltiples (véase el siguiente recuadro de detalles).

¿Qué es un protocolo de grabación en dos fases?

Detalles

El protocolo de *grabación en dos fases* sirve para sincronizar actualizaciones en diferentes máquinas a fin de que todas sean o bien un fracaso o bien un éxito. Esto se logra centralizando la decisión de grabar, pero concediendo a cada participante el derecho de voto. Es como un matrimonio católico: frente al altar se tiene la última oportunidad de repudiar la transacción. Si ninguna de las partes presenta objeciones, el matrimonio se consuma.

No debería sorprender entonces que cada implementación comercial introduzca su propia variante de protocolo de grabación en dos fases. Como de costumbre, no interoperan, y existen por supuesto organismos de estándares ocupados en que todo esto trabaje en conjunto. En diciembre de 1992 —después de un ciclo de desarrollo de cinco años—, ISO publicó su estándar *OSI TP*, que define muy rígidamente la implementación de una grabación en dos fases (véase Figura 16-4). Abordemos la mecánica de este protocolo:

1. *En la primera fase de una grabación*, el nodo administrador de grabación —también conocido como *nodo raíz* o *coordinador de transacción*— envía comandos de *preparar-grabación* (*prepare-to-commit*) a todos los nodos subordinados a los que se les solicitó directamente participar en la transacción. Los subordinados quizás tengan piezas adicionales de la transacción en otros nodos (o administradores de recursos), a los que deben propagar el comando de preparar-grabación. Esto se convierte entonces en un árbol de transacción, con el coordinador en la raíz.
2. *La primera fase de la grabación termina* cuando el nodo raíz recibe las señales de *listo-para-grabación* (*ready to commit*) de todos sus nodos subordinados directos participantes en la transacción. Esto significa que la transacción ha sido ejecutada exitosamente hasta este punto en todos los nodos y que éstos se hallan preparados para realizar una grabación final. El nodo raíz registra el hecho en un lugar seguro (esta información sirve para la recuperación en caso de falla en el nodo raíz).

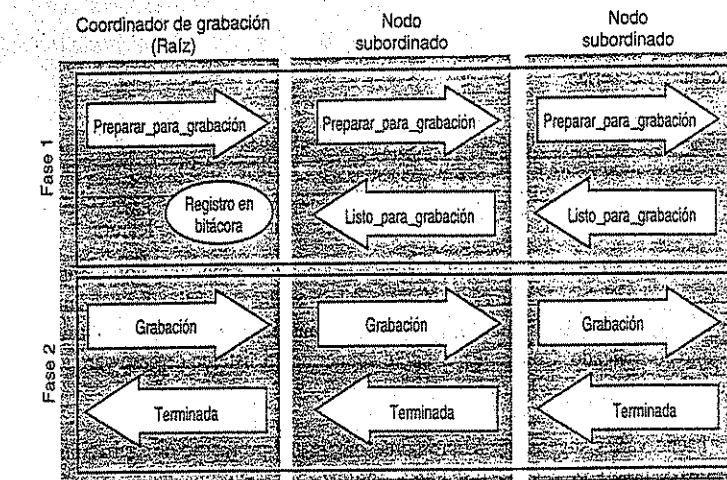


Figura 16-4. Mecánica de la grabación en dos fases de OSI TP.

3. *La segunda fase de la grabación comienza* en cuanto el nodo raíz toma la decisión de *grabar la transacción*, con base en el voto afirmativo unánime. Instruye entonces a sus subordinados para que graben. Éstos instruyen a su vez a sus subordinados para que hagan lo propio, y la orden se difunde por todo el árbol.
4. *La segunda fase de la grabación termina* cuando todos los nodos involucrados han grabado sin accidentes su parte de la transacción y la han vuelto durable. La raíz recibe todas las confirmaciones y puede decirle a su cliente que la transacción ha terminado. Descansará hasta la siguiente transacción.
5. *La grabación en dos fases aborta* si cualquiera de los participantes emite una indicación de *rechazo* (*refuse*), lo que significa que su parte de la transacción fracasó. En este caso, el nodo raíz notifica la realización de vuelta atrás a todos sus subordinados, quienes a su vez proceden de igual forma con sus propios subordinados.

La especificación XA de X/Open define un conjunto de API que opera con el protocolo base OSI TP. Para participar en una grabación en dos fases definida por XA, los monitores de TP y administradores de recursos (como bases de datos y colas de mensajes) deben trazar sus protocolos privados de grabación en dos fases con los comandos de XA. También deben acceder a que sea un tercero quien dirija la transacción, algo que no están acostumbrados a hacer. La especificación XA permite a los participantes retirarse de la transacción global durante la fase 1 si no tienen que actualizar recursos. En XA, un monitor de TP puede usar una grabación en una fase si trata con un solo administrador de recursos. Nos extenderemos en XA en el siguiente capítulo.

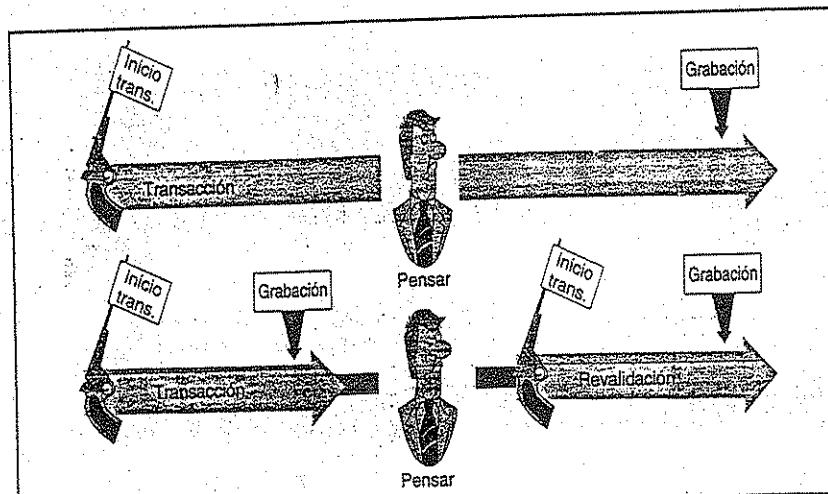


Figura 16-6. Excluya a la gente del circuito creando dos transacciones planas.

- **Transacciones de negocios que se extienden por largos períodos.** Éstas son las usuales transacciones de ingeniería de diseño asistido por computadora (CAD: computer-aided design) que pueden requerir de componentes administrados por CAD para trabajarse a lo largo de varios días y pasar de un ingeniero a otro (véase Figura 16-7). La transacción de CAD debe ser capaz de suspenderse y reanudarse luego de cada interrupción, preservar el trabajo en curso entre interrupciones y saber dónde se quedó y qué debe hacerse a continuación. En esencia, se convierte en un administrador de flujo de trabajo. Obviamente, las transacciones simples deben aumentarse por un programa de flujo de trabajo para manipular estas labores de larga vida. Ésta es un área en la que modelos alternativos de transacciones —como las transacciones de registro de entrada y salida de bases de datos de objetos, administración de réplicas, uso de versiones y flujo de trabajo— se muestran muy promisorios.

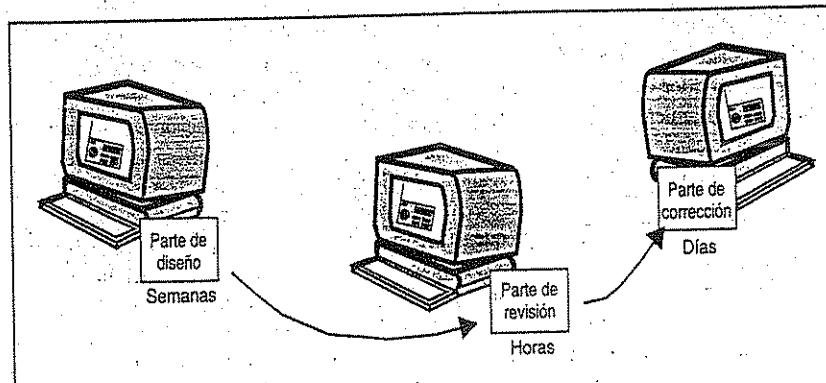


Figura 16-7. Transacciones de larga vida de días o semanas de duración.



- **Transacciones de negocios voluminosas.** En este caso el problema clásico es: ¿cómo manejar un millón de actualizaciones de registros bajo control de transacciones? (véase Figura 16-8). ¿Es forzoso que la transacción entera dé marcha atrás en caso de que ocurra una falla después de actualizado el registro 999,999? Sí; éste es el todo o nada si se emplea una sola transacción simple para el millón de actualizaciones. Por otra parte, si usted realiza cada actualización como una transacción independiente, el proceso resulta mucho más lento —tiene que hacer un millón de grabaciones distintas—, y ¿dónde reiniciaría en caso de falla? La solución propuesta en esta área son las transacciones encadenadas o syncpoints. Pero se trata de una solución no exenta de retrasos, porque introduce más grabaciones y quizás cierto código de reinicio. Nos parece preferible reiniciar una transacción simple ocasional, que adoptar las alternativas. Después de todo, ¿con qué frecuencia falla una transacción voluminosa?

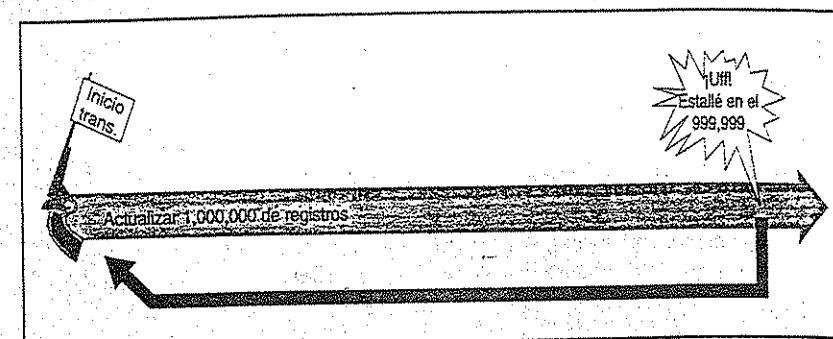


Figura 16-8. Transacción simple: Si falla, reiniciar la tarea de un millón de actualizaciones.

- **Transacciones de negocios entre compañías o en Internet.** El problema aquí es de carácter político. Muy pocas compañías permitirán que un monitor de TP (o base de datos) externo sincronice en tiempo real una transacción en sus sistemas, mediante una grabación en dos fases. La solución políticamente correcta puede ser realizar un intercambio entre compañías con el uso de colas de mensajes para transacciones de acoplamiento holgado. Una solución de MOM permite a las organizaciones dividir la unidad de trabajo en muchas transacciones que puedan ejecutarse asincrónicamente, procesarse en máquinas diferentes y ser coordinadas por monitores de TP independientes dentro de cada compañía (véase Figura 16-9). Se pierde consistencia instantánea, pero se mantiene el control entre compañías. Desde la perspectiva del software, se termina por descomponer una sola transacción simple de grabación en dos fases, en tres transacciones simples independientes ejecutadas en el monitor de TP de la compañía A, MOM y el monitor de TP de la compañía B. La transacción de MOM se cerciora de que la transacción de la computadora de la compañía A a la computadora de la compañía B ocurra con seguridad. Asumimos que MOM ofrece una cola duradera que proporciona la "D" de ACID en tiempo de grabación.

En resumen, la mayor parte de los problemas de las transacciones simples son producto de la rigidez (e interbloqueo) impuesta por la disciplina de todo o nada en situaciones en las que se requiere de mayor flexibilidad. Casi todos estos problemas pueden eludirse dividiendo las transacciones.

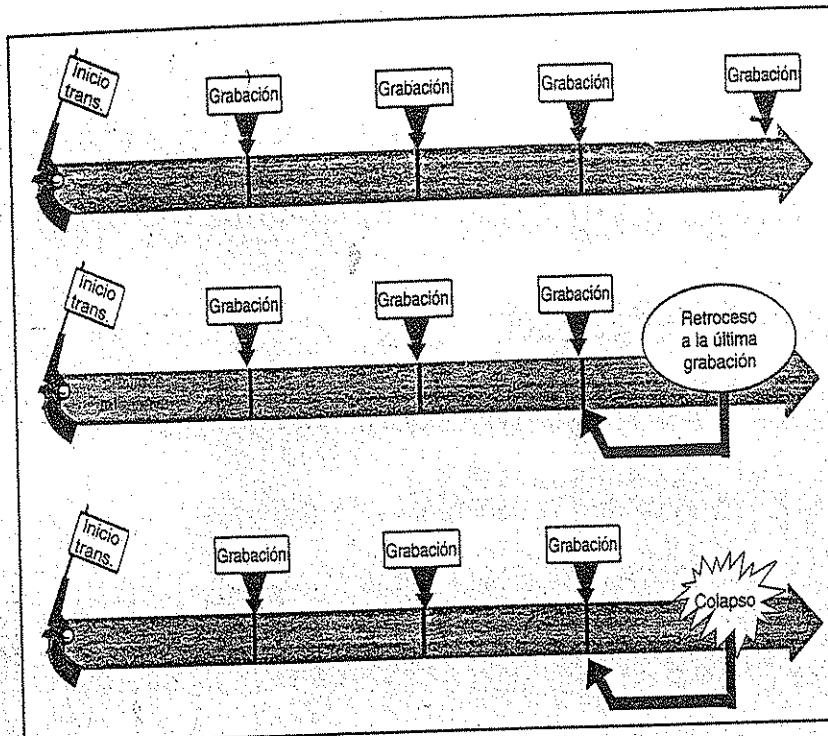


Figura 16-11. Transacciones encadenadas: Grabaciones durables.

programa. Esto permite tratar a la cadena en su totalidad como unidad atómica de trabajo. Aparte de disponer del pastel, puede comérselo.⁴

Transacciones anidadas

Las *transacciones anidadas* ofrecen la posibilidad de definir transacciones dentro de otras transacciones. Lo hacen dividiendo una transacción en jerarquías de "subtransacciones" (a la manera en que un programa se compone de procedimientos). La transacción principal pone en marcha las subtransacciones, las cuales se comportan como transacciones dependientes. Una subtransacción también puede poner en marcha sus propias subtransacciones, haciendo muy recursiva la estructura entera (véase Figura 16-13).

⁴ El término "saga" fue originalmente propuesto por Bruce Lindsay, de Almaden Research de IBM. El concepto fue desarrollado en su totalidad por Héctor García-Molina y K. Salem en 1987.

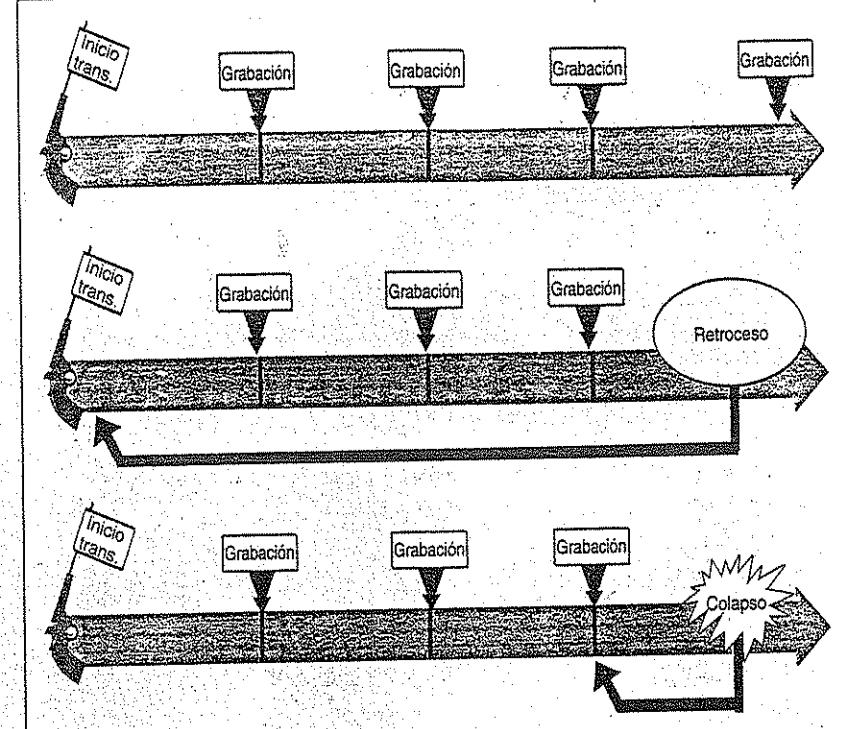


Figura 16-12. Sagas: Grabaciones durables pero con opción de retroceso.

Cada subtransacción puede emitir una grabación o retroceso para las piezas de trabajo asignadas. Cuando una subtransacción graba, sus resultados sólo son accesibles para el padre que la generó. La grabación de una subtransacción se vuelve permanente después de que ésta emite una grabación local y todos sus antecesores graban. Si una transacción madre realiza un retroceso, todas sus transacciones hijas retroceden, así hayan emitido grabaciones locales.

El mayor beneficio del anidamiento es que una falla en una subtransacción puede controlarse y ésta procesarse de nuevo con otro método, permitiendo que la transacción principal no se pierda. El anidamiento ayuda a que los programadores generen transacciones más granulares. La única implementación comercial de transacciones anidadas que conocemos es el monitor de TP de Encina. C (o C++) para transacciones de Encina permite declarar directamente en el código el anidamiento, en el punto en que aquél comienza a parecerse a invocaciones de procedimientos regulares. En algunos casos, el anidamiento puede ser desastroso; provoca más problemas que soluciones. Claro que con el monitor de TP de Encina en el mercado, la decisión es suya.

alcance de una transacción distribuida los recursos del escritorio, como la interfaz del usuario, las bodegas locales de datos, o los agentes personales.

En este capítulo explicaremos con cierto detalle qué son los monitores de TP y qué funciones realizan. Expondremos el modelo de X/Open para la interacción de los monitores de TP con otros administradores de recursos en un entorno abierto. Concluiremos con una lista de los beneficios que ofrecen los monitores de TP. Creímos necesario incluir esta lista porque en los ámbitos de LAN de PC y Unix aún no se han comprendido del todo bien los beneficios de los monitores de TP. Se les ha tratado con temor y abandonado en manos de los "sumos sacerdotes" de la ciencia de la computación, o sencillamente se les ha menospreciado en calidad de reliquias. Pero francamente no merecen una cosa ni la otra. Los monitores de TP son una delicia de programación, y de redes de cliente/servidor comunes y corrientes hacen surgir la magia de las transacciones. Pero basta de palabrería, que esto no es un recuadro de debate.

MONITORES DE TP

Los monitores de TP aparecieron originalmente en las macrocomputadoras para brindar entornos de tiempo de ejecución robustos capaces de soportar aplicaciones de OLTP de gran escala: reservaciones en aerolíneas y hoteles, operaciones bancarias, cajas bancarias automáticas, sistemas de autorización de crédito y sistemas de intermediación bursátil. Desde entonces, OLTP se ha difundido a prácticamente todo tipo de aplicaciones de negocios, incluidos hospitales, manufactura, sistemas comerciales de punto de venta, despachadoras de gasolina automatizadas y servicios de directorio telefónico. Los monitores de TP ofrecen todos los servicios que se requieran para que estas aplicaciones de OLTP sigan funcionando en el estilo al que están acostumbradas: con alta reactividad, disponibilidad y buena administración. El paso de OLTP a plataformas de cliente/servidor hizo surgir una nueva especie de monitores de TP, a fin de que este nuevo entorno fuera adecuado para aplicaciones decisivas en el cumplimiento de objetivos.

¿Qué es un monitor de TP?

No debería sorprendernos que la industria aún carezca de una definición de monitor de TP de aceptación generalizada. Nosotros nos atenemos a la definición de Jeri Edwards, para quien un monitor de TP es "un sistema operativo para el procesamiento de transacciones". Esta definición contiene la esencia misma de un monitor de TP. Pero, ¿qué hace en la realidad un sistema operativo para el procesamiento de transacciones? ¿Cómo se relaciona con el resto del mundo? ¿Qué servicios presta? Responderemos todas estas preguntas. Para decirlo brevemente, el monitor de TP es excelente para dos cosas:

- *La administración de procesos*, lo cual incluye poner en marcha los procesos del servidor, canalizar trabajo en dirección a ellos, vigilar su ejecución y equilibrar sus cargas de trabajo.
- *La administración de transacciones*, lo que significa que garantiza las propiedades ACID para todos los programas que operen bajo su protección.

Monitores de TP y OS: El prodigioso acto de canalización

La introducción original de los monitores de TP respondió a la necesidad de correr clases de aplicaciones capaces de atender a cientos y en ocasiones miles de clientes (piense nada más en la aplicación de reservaciones de líneas aéreas). Si a cada uno de estos miles de clientes se les hubieran dado todos los recursos necesarios sobre un servidor —normalmente una conexión de comunicaciones, medio MByte de memoria, uno o dos procesos y una docena de manipuladores de archivos abiertos—, incluso el más grande de los servidores en macrocomputadora se habría venido abajo (véase Figura 17-1). Por fortuna, no todos los clientes requieren de servicios al mismo tiempo. Sin embargo, cuando los requieren los desean *inmediatamente*. Se nos ha dicho que las personas que se encuentran en el otro extremo tienen una "tolerancia de espera" de dos segundos o menos. Los monitores de TP ofrecen un sistema operativo —encima de los OS existentes— que conecta en tiempo real a estas miles de personas impacientes con un fondo de procesos compartidos del servidor.

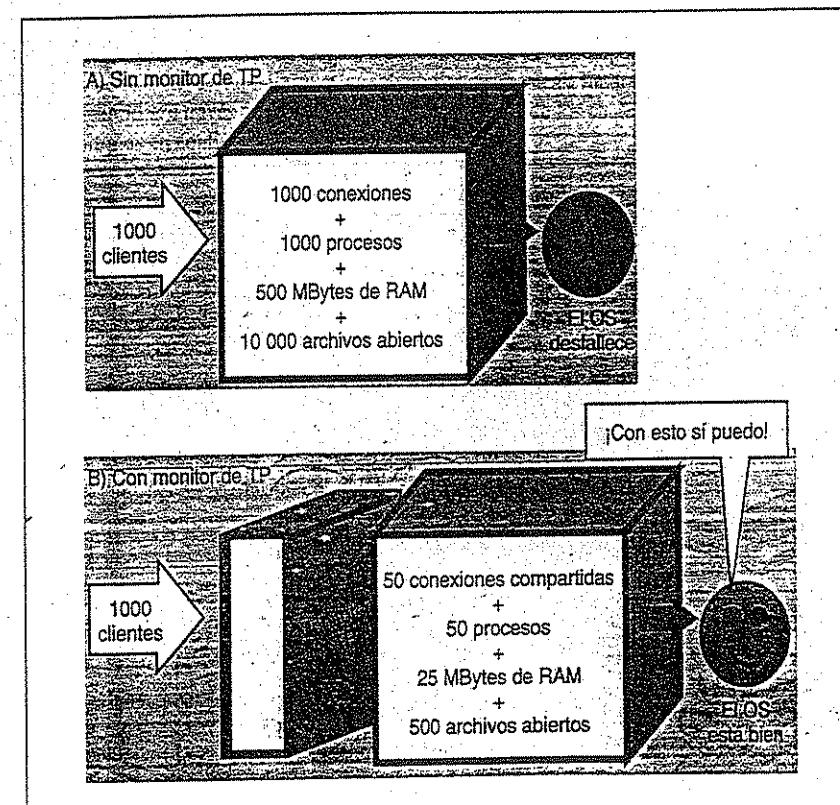


Figura 17-1. Por qué un sistema operativo servidor necesita un monitor de TP.



Con los monitores de TP, los programadores de aplicaciones no tienen que preocuparse por asuntos como la concurrencia, fallas, conexiones defectuosas, equilibrio de cargas y sincronización de recursos entre múltiples nodos. Todo esto se vuelve transparente para ellos, a la manera en que un sistema operativo vuelve transparente el hardware para programas ordinarios. Para decirlo llanamente, los monitores de TP ofrecen los mecanismos de tiempo de ejecución para la operación de transacciones, y lo hacen encima del hardware y los sistemas operativos ordinarios. Asimismo, aportan una estructura para el funcionamiento de las aplicaciones servidor.

Tipos de interacción de cliente/servidor con monitor de TP

Los sistemas operativos ordinarios deben conocer la naturaleza de las labores y recursos que administran. Esto también se aplica a los monitores de TP: deben brindar un entorno optimizado para la ejecución de las transacciones que operan bajo su control. Esto significa que deben cargar los programas del servidor, asignar dinámicamente las solicitudes de clientes recibidas a procesos servidores, procurar la recuperación necesaria tras la ocurrencia de fallas, entregar respuestas a los clientes y cerciorarse de que el tráfico de alta prioridad tenga preferencia.

¿De qué clase de suposiciones parte el monitor de TP para sus tipos de transacciones en interacciones de cliente/servidor? Por lo general responden a una de cuatro categorías: conversacional, de RPC, de colas y por lotes (véase Figura 17-3). Las transacciones por lotes suelen ejecutarse en el modo de baja prioridad. Las transacciones RPC y conversacionales implican normalmente a un usuario humano que necesita atención inmediata; se ejecutan en el modo de alta prioridad. Las transacciones en cola basadas en MOM pueden ser de ambos tipos.

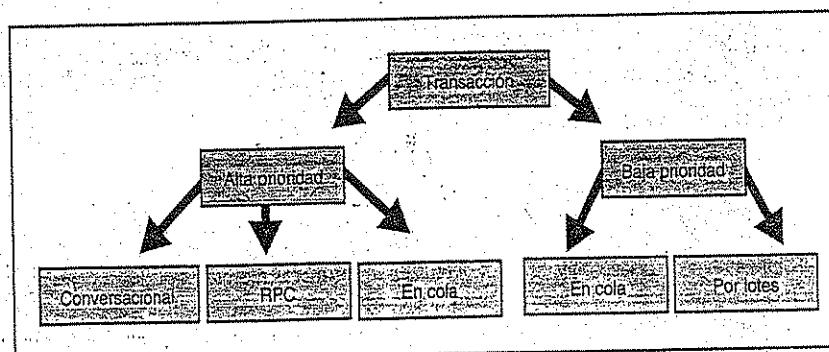


Figura 17-3. Perfiles de transacciones en cliente/servidor.

Además, los monitores de TP deben estar preparados para comunicarse con todos los administradores de recursos en los que se ejecuta la transacción, ya sea que se encuentren en la misma máquina o a lo largo de una red. Cuando los administradores de recursos se hallan en redes, el monitor de TP sincroniza la transacción con los monitores de TP remotos por medio de una grabación en dos fases.



RPC, colas y conversaciones para transacciones

A primera vista, los intercambios de transacciones en cliente/servidor parecen usar los modelos tradicionales de comunicación de NOS: colas RPC y comunicaciones en conversaciones de puerto a puerto. Pero esta impresión es falsa. En realidad echan mano de versiones muy aumentadas de estos mecanismos tradicionales de comunicación. Sin embargo, la mayoría de los elementos de valor agregado se vuelven transparentes para el programador; parecen intercambios ordinarios encerrados por llamadas de arranque y finalización de transacción. Las versiones para transacciones complementan los ya conocidos intercambios de NOS con las siguientes extensiones de valor agregado:

- Aportan *delimitadores de transacciones* que permiten a un cliente especificar los límites de inicio y fin de transacción. La mecánica de grabación actual es *delegada* por lo general a uno de los monitores de TP del servidor, porque se asume que el cliente no es confiable para ello.
- Introducen —secretamente— un intercambio en tres vías entre un cliente, el servidor y el monitor de TP (el administrador de transacción). Una nueva transacción le es asignada a un identificador (ID) específico por el monitor de TP coordinador. Todos los subsecuentes intercambios de mensajes entre los participantes son sellados con este ID de transacción. Los intercambios de mensajes permiten que el monitor de TP controle lo que Jim Gray llama la “red en dinámica expansión” de administradores de recursos que participan en una transacción distribuida. Los monitores de TP necesitan esa información para coordinar la grabación en dos fases con todos los participantes en una transacción.
- Implantan información de estado de transacción en cada uno de los mensajes intercambiados. Esta información contribuye a que el monitor de TP identifique el estado de la transacción distribuida y deduzca lo que debe hacer a continuación.
- Permiten que un monitor de TP imponga la semántica de *exactamente una vez*, lo que significa que el mensaje sólo se ejecuta una vez.
- Garantizan que un proceso servidor se halle en el extremo receptor del mensaje. Las RPC y MOM tradicionales no ponen la menor atención en asuntos de este tipo; dan por supuesto que un programa aparecerá “automáticamente” en el extremo receptor.
- Ofrecen enruteamiento al servidor con base en las clases de servidor, cargas de servidor, recuperaciones automáticas y otros factores.

Como puede ver, esto implica mucho más que un simple intercambio de RPC o MOM. La bibliografía especializada designa a estos servicios complementados como *RPC transaccional (TRPC: transactional RPC)*, *colas para transacciones* y *conversaciones para transacciones*. El factor distintivo es que todos los administradores de recursos y procesos invocados por estas llamadas pasan a formar parte de la transacción. El monitor de TP es informado de todas las llamadas de servicios; emplea esta información para orquestar las acciones de todos los parti-



Al destruir esta conexión directa, los monitores de TP controlan todo el tráfico que enlaza a cientos (o miles) de clientes con programas de aplicación y recursos posteriores. Los monitores de TP aseguran que las transacciones se lleven a cabo con exactitud, ejercen el equilibrio de cargas y elevan el desempeño general del sistema. Sobre todo, independizan los procesos respecto a todos los administradores de recursos. Nos permiten trabajar sin necesidad de recursos posteriores.

En pocas palabras, los monitores de TP tratan a los procesos como ciudadanos de primera clase. Les conceden existencia propia, ajena a la base de datos o GUI. Esto significa que usted puede distribuir procesos entre máquinas y redes de acuerdo con la ubicación más razonable. En contraste con ello, en el caso de cliente/servidor de base de datos, la aplicación se halla profundamente enterrada en la herramienta frontal (el cliente grande) o en una base de datos bajo la forma de procedimientos almacenados (el servidor amplio). El modelo de base de datos no trata a los procesos como ciudadanos independientes de primera clase. Así, aunque los procedimientos almacenados tengan una apariencia de 3 planos, su empaquetamiento es epítome de los 2 planos:

Como explicaremos más adelante, el método de 3 planos es el único sensato en un mundo intergaláctico de servidores múltiples. No pierda de vista sin embargo que los monitores de TP son sólo uno de los medios para la implementación de soluciones de cliente/servidor en 3 planos. A medida que avancemos, le mostraremos otros métodos. Como es natural, usted habrá de elegir el estilo que mejor responda a sus necesidades. □

ESTÁNDARES DE ADMINISTRACIÓN DE TRANSACCIONES: DTP DE X/OPEN Y OSI-TP

Los monitores de TP necesitan estándares porque son el software enlazador por excelencia. Las aplicaciones que coordinan podrían correr en diferentes plataformas con acceso a diferentes bases de datos y administradores de recursos. Con toda probabilidad, estas aplicaciones fueron desarrolladas con diferentes herramientas. Además, se desconocen absolutamente entre sí. La única manera de lograr que estas piezas dispares trabajen juntas es la propuesta de "estándares abiertos" que especifiquen las relaciones entre un monitor de TP con administradores de recursos, otros monitores de TP y sus clientes.

La mayor parte de la actividad de estándares en torno a los monitores de TP procede de dos fuentes: la Organización de Estándares Internacionales (ISO: *International Standard Organization*) —las especificaciones OSI-CCR y OSI-TP— y las especificaciones de *procesamiento de transacciones distribuidas* (DTP: *distributed transaction processing*) de X/Open. Los estándares ISO de ISO especifican los protocolos de mensajes (es decir, los FAP) que permiten que los monitores de TP interoperen. La especificación OSI-TP, de la que nos ocupamos en el capítulo anterior, es el más importante de estos estándares; define, entre otras cosas, el protocolo de grabación en dos fases. X/Open ha tomado la delantera en lo que se refiere a la definición de las API dentro de una estructura general de procesamiento de transacciones. Juntos, tanto el DTP de X/Open como OSI-TP constituyen el fundamento para la "administración de transacciones". Dedicaremos esta sección a DTP de X/Open.



El modelo de referencia DTP de X/Open, cosecha 1991

El modelo DTP de X/Open es una arquitectura de software que permite que múltiples programas de aplicación compartan recursos provistos por administradores de recursos múltiples, así como que su trabajo sea coordinado en transacciones globales.

X/Open, DTP Reference V2
(Diciembre de 1993)

En 1991, el grupo XTP de X/Open publicó el *Modelo de referencia para el procesamiento de transacciones* (*transaction processing reference model*), el cual ha merecido la entusiasta acogida de la industria. El principal propósito de este modelo es definir los componentes de un sistema basado en transacciones y localizar las interfaces entre ellos. En este modelo de 1991 se definieron tres componentes: programas de aplicación, administradores de transacciones y administradores de recursos (véase Figura 17-5). De acuerdo con la definición de X/Open:

- Un *administrador de recursos* es cualquier pieza de software que administre recursos compartidos —por ejemplo: un administrador de base de datos, una cola persistente o un sistema de archivos para transacciones— y permita que las actualizaciones de sus recursos sean coordinadas externamente vía un protocolo de grabación en dos fases.
- Un *administrador de transacciones* es el componente que coordina y controla a los administradores de recursos. El administrador de transacciones y el administrador de recursos se comunican vía la *interfaz XA* de X/Open, publicada en 1991. El administrador de transacciones se sirve de llamadas de API *xa_** para interactuar con los administradores de recursos; éstos emplean a su vez llamadas de API *ax_** para interactuar con el administrador de transacciones. Por ejemplo, éste emite un *xa_arranque* (*xa_start*) para indicar a un administrador de recursos que debe unirse a una nueva transacción. Emite *xa_preparar* (*xa_prepare*), *xa_grabar* (*xa_commit*) y *xa_retroceso* (*xa_rollback*) para indicarle que debe realizar una grabación en dos fases. Y emite *xa_fin* (*xa_end*) para indicarle al administrador de recursos que abandone la transacción. XA define llamadas adicionales para la recuperación de transacciones "dudosas". En sentido inverso, un administrador de recursos emite una llamada de *ax_reg* para registrar dinámicamente su presencia con el administrador de transacciones.²
- Un *programa de aplicación* emplea las API generales provistas por un administrador de recursos (SQL, por ejemplo), pero emite directamente las llamadas acotadas para transacciones al administrador de transacciones a través de la *interfaz TX* de X/Open, publicada en 1992. Una aplicación llama a *tx_inicio* (*tx_begin*) para arrancar una transacción, *tx_grabación* (*tx_commit*) para grabarla, a *tx_retroceso* (*tx_rollback*) para abortarla y a *tx_fijar_controles_transacción* (*tx_set_transaction_controls*) para determinar el modo de encadenamiento. Las transacciones pueden ser encadenadas o no encadenadas (modo por omisión).

² X/Open permite que los administradores de recursos se asocien a una transacción global sólo una vez que hayan sido directamente llamados por la aplicación. Usan la llamada de *ax_reg* para registrar dinámicamente su presencia.

El protocolo de capa superior que se persigue para cada una de estas API es el FAP de OSI-TP. Bajo el FAP de OSI-TP, los administradores de recursos de comunicación pueden soportar múltiples protocolos de transporte, como TCP/IP, OSI y APPC. Por supuesto que los protocolos propietarios pueden ser usados entre dominios homogéneos de administradores de transacciones. El uso de OSI-TP es obligatorio para comunicaciones entre dominios heterogéneos de administradores de transacciones. En teoría, deberíamos estar en condiciones de alcanzar cierto nivel de interoperabilidad entre proveedores múltiples (después de todo ésta es la idea que se halla en el fondo de todos estos estándares). No obstante, X/Open no aborda la relación entre las diferentes API de comunicación; por ejemplo, no dice nada sobre si una aplicación con XATMI puede intercambiar mensajes con una aplicación con CPI-C vía los administradores de recursos de comunicación especificados por X/Open.

En la Figura 17-7 se muestra en qué forma puede ser conducido un intercambio de transacción global. La aplicación en el nodo izquierdo interactúa con el recurso remoto a través de su administrador de recursos de comunicación. El administrador de transacciones del nodo en el que se origina la solicitud, funge como coordinador de la grabación usando los servicios del administrador de recursos de comunicación. El coordinador de grabación es el monitor de transacciones raíz, y el monitor remoto es un *subordinado*.

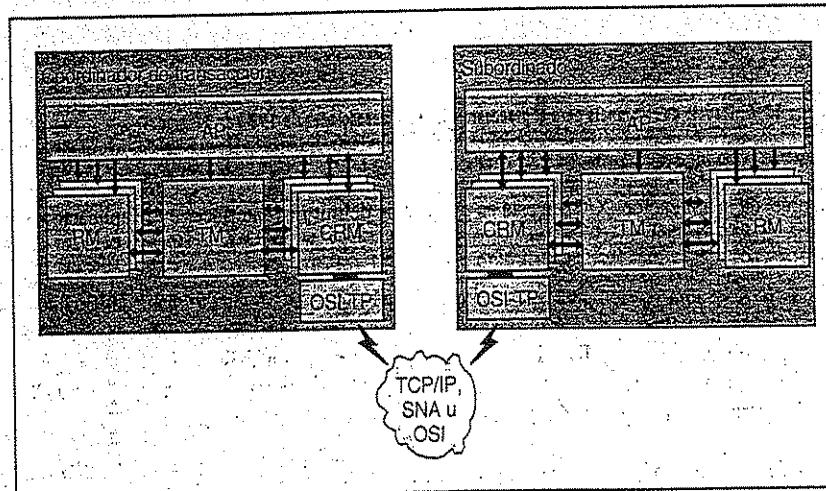


Figura 17-7. Una transacción distribuida.

Desde luego que en un intercambio pueden participar más de dos nodos. Las transacciones globales que operan entre administradores de transacciones distribuidas se administran mediante árboles de relaciones de administradores de transacciones (véase Figura 17-8). En este ejemplo se muestra que B es *superior* tanto a C como a D, pero *subordinado* de A, que funge como coordinador de la grabación. Durante la grabación en dos fases, el nodo superior se encarga de la coordinación de grabación de sus subordinados y reporta los resultados en sentido ascendente por la cadena.

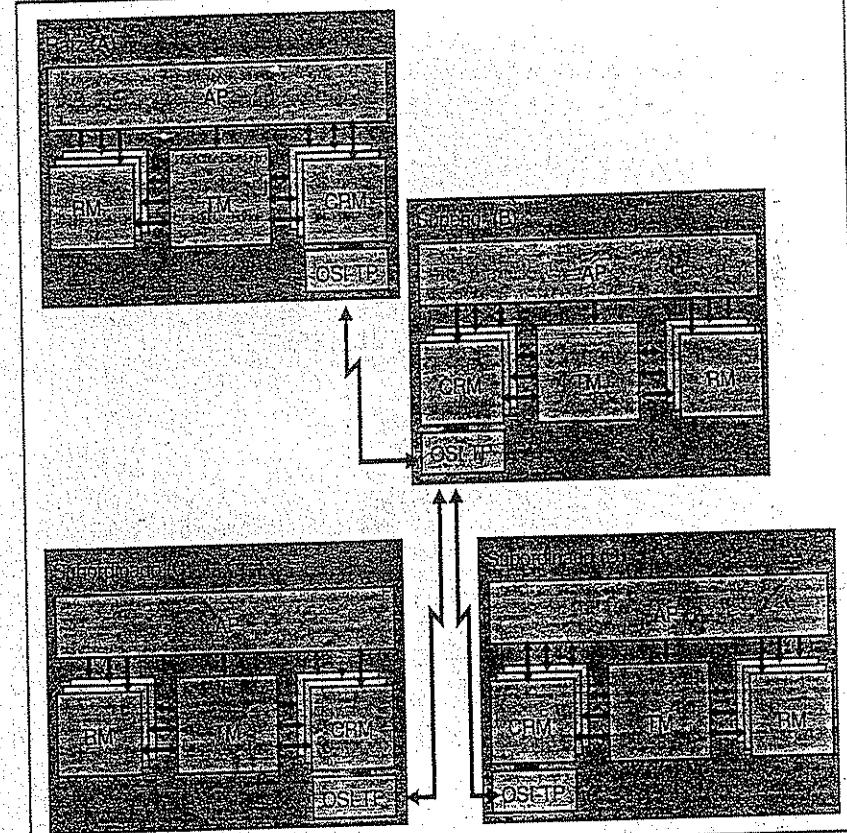


Figura 17-8. Estructura de árbol de una transacción global.



¿Por qué tres estándares de interfaces?

¿A qué se debe que necesitemos tres interfaces en el nivel de aplicación? ¿No tendría más sentido una sola API desde la perspectiva de la exportabilidad y la interoperabilidad? Eso es lo que pensamos nosotros, pero la línea "oficial" de X/Open es que cada una de ellas aporta un rico conjunto de paradigmas de comunicación entre aplicaciones. Una explicación más verosímil es que la comisión creó una especificación que dejará satisfechos a sus tres aguerridos contingentes: DCE, CICS y Tuxedo. ¡Así es como se crean los estándares!

¿NECESITA UN MONITOR DE TP?

Se impone una felicitación para la comunidad de OLTP. Ahora ya podemos unir casi todos nuestros sistemas.

Alfred Spector, director general, Transarc
(Septiembre de 1995)

Debido a que es probable que muchos de nuestros lectores desconozcan buena parte de lo referente a los monitores de TP, ofrecemos a continuación una lista de los beneficios que brindan los monitores de TP a las aplicaciones cliente/servidor. A pesar de que los monitores de TP surgieron para servir a aplicaciones "decisivas para misiones críticas" de gran tamaño, las nuevas versiones son aptas para el manejo de aplicaciones cliente/servidor que van desde unos cuantos hasta miles de nodos. Creemos que no pasará mucho tiempo antes de que en todas las estaciones de trabajo conectadas a redes, y no sólo en los servidores, resida un componente de monitor de TP.

He aquí, una lista de los beneficios que pueden obtenerse del uso de la actual generación de monitores de TP orientados a cliente/servidor:

- **Estructura de desarrollo de aplicaciones cliente/servidor.** Los proveedores de herramientas visuales soportan crecientemente en forma directa a las RPC y están volviendo a los monitores de TP transparentes para los desarrolladores. Las RPC definidas por IDL son más fáciles de integrar con herramientas de presentación final, que los procedimientos almacenados propietarios. En la parte del servidor, los monitores de TP ofrecen cápsulas de servidor de propósito general (clases de servidor) que corren en sus RPC. El monitor de TP introduce en los servidores un estilo de programación determinada por eventos al permitirle asociar RPC (manejadores de eventos) con eventos del servidor. Además, el entorno de tiempo de ejecución del monitor de TP impone la disciplina ACID sin necesidad de otro código especializado que no sea arrancar/terminar transacción. Un monitor de TP puede ser concebido como el ofrecimiento de una *estructura preconstruida* que le ayuda a usted a formar, operar y administrar una aplicación cliente/servidor (no se verá obligado a empezar de cero). Complementados con herramientas GUI de proveedor abierto, los monitores de TP constituyen una plataforma excelente para el veloz desarrollo de aplicaciones cliente/servidor robustas y de alto desempeño.
- **Muros de protección.** En un ámbito de cliente/servidor, es importante protegérse contra todo aquello que pueda entorpecer el entorno distribuido. Los monitores de TP implementan "muros de protección" entre aplicaciones y administradores de recursos, así como entre las mismas aplicaciones. Soportan muros de protección de acoplamiento ajustado como grabaciones en dos fases y muros de acoplamiento holgado como los provistos por las colas para transacciones. La unidad de protección es la transacción ACID.
- **Alta disponibilidad.** Los monitores de TP están diseñados para sortear todo tipo de fallas. La difusión de los principios ACID a todos los componentes permite crear sistemas autorremediables. Los monitores de TP siempre están al tanto del estado de los recursos de cliente/servidor bajo su control. Con ACID, usted puede detectar una falla en el momento mismo en que ocurre. En caso de presentarse una falla de hardware, el monitor de TP puede

reiniciar el proceso fallido o comutar a un proceso en otro nodo. De este modo es factible contar con arquitecturas sin un solo punto de falla.

- **Equilibrio de cargas.** Los monitores de TP se especializan en la administración de procesos, y soportan técnicas de equilibrio de cargas tanto estáticas como dinámicas. Soportan la prioridad de solicitudes y pueden duplicar *dinámicamente* procesos del servidor en el mismo nodo servidor o en nodos diferentes. En el caso estático, un fondo de clases de servidor puede destinarse al manejo de ciertas cargas pico (entre turnos de trabajo, por ejemplo), y desarmarse después para soportar otras combinaciones de labores durante el día. El software de equilibrio de cargas de los monitores de TP se asocia en forma excelente con la nueva generación de hardware del servidor de SMP (y MPP) de la actualidad.
- **Integración de MOM.** Los monitores de TP son el complemento ideal de los MOM. Juntos ofrecen soporte a transacciones de larga vida y aplicaciones del tipo flujo de trabajo. Los monitores de TP pueden actuar como coordinadores de transacciones en labores intercambiadas mediante colas para transacciones. Los eventos formados en cola pueden activar procesos del servidor administrados por el monitor de TP.
- **Facilidad de ampliación de funciones.** Los monitores de TP alientan la creación de procedimientos modulares reutilizables para el encapsulamiento de administradores de recursos. Con un monitor de TP, usted exporta la llamada a función y no los datos mismos. Esto significa que puede seguir añadiendo nuevas llamadas de funciones y permitir que el monitor de TP distribuya esas funciones entre múltiples servidores. Los monitores de TP le permiten crear aplicaciones de alto grado de complejidad con sólo agregar procedimientos. Garantizan que procedimientos que ignoran todo unos de otros, trabajarán al unísono con ACID. Además, el monitor de TP le permite mezclar administradores de recursos, para que siempre pueda empezar con el mismo administrador de recursos y pasar más tarde a otro sin perder sus inversiones en llamadas de función. Todas las funciones —incluso las de herencia— pueden formar parte del fondo de procedimientos reutilizables administrado por el monitor de TP. En otras palabras, los monitores de TP le permiten añadir en cualquier punto recursos de servidor heterogéneos sin alterar la arquitectura ya existente de la aplicación. El Standish Group llama a esto "escalabilidad matriz".
- **Costo reducido del sistema.** Con monitores de TP usted ahorra dinero. De acuerdo con el Standish Group, los monitores de TP pueden resultar en ahorros totales en costos del sistema de más del 30% —dependiendo de la escala del sistema— en comparación con un método más centrado en bases de datos. Además, las investigaciones de esta misma fuente indican la posibilidad de alcanzar significativos ahorros en "tiempo de desarrollo", de hasta el 40 o 50%. Junto con ello, el efecto de canalización de los monitores de TP puede resultar en grandes ahorros en la adquisición de administradores de recursos. Esto se debe a que los proveedores de bases de datos cobran por el número de usuarios activos; la canalización reduce ese número, lo que equivale a menores costos por concepto de licencias. Por ejemplo, el Standish Group estima que con un monitor de TP usted puede ahorrar el 62% en un sistema Oracle para 128 usuarios. Gracias a su equilibrio de cargas, los monitores de TP también ofrecen un mejor desempeño con el uso de los mismos recursos del sistema; esto quiere decir que usted puede correr su aplicación en hardware de menor costo. Finalmente, los monitores de TP no lo atan a una solución de base de datos de proveedor específico, lo que vuelve más competitivo el proceso de adquisición e incrementa los ahorros en cos-



En contraste con los administradores de bases de datos, los monitores de TP extienden la noción de transacciones a *todos* los recursos, no sólo a los centrados en bases de datos. Los monitores de TP controlan la ejecución de funciones en un servidor único o entre servidores en la red, método llamado *Tp pesado (TP-Heavy)*. Aquí nos ocuparemos del actual debate en la industria entre TP ligero y TP pesado. Tal como lo ha dicho Jim Gray, "sus problemas no se van a acabar por el solo hecho de adoptar el concepto de RPC o incluso de TP ligero".

Pero mientras estos dos bandos de TP se enfrentan, la mayor parte del ámbito de cliente/servidor centrado en PC (y Unix) es (*TP-Less*) ajeno a TP. En el campo de PC sigue privando un notorio desconocimiento de qué es la administración de transacciones y por qué se le necesita. Sin embargo, la administración de transacciones es la segunda naturaleza de la mayoría de las personas dedicadas a IS e interesadas en la "reducción de tamaño" desde entornos de macrocomputadoras. Esta gente sería incapaz de desplegar una aplicación de OLTP en LAN de PC sin algún tipo de monitor de TP. En consecuencia, está generando la demanda de una nueva generación de monitores de TP basados en LAN. El Standish Group calculó que los ingresos totales en monitores de TP fueron en 1995 de \$17,200 millones de dólares, cifra que se espera que crezca a \$20,000 millones en 1997. Estos números indican que el mercado de OLTP representa una gran oportunidad para los sistemas de cliente/servidor.

En este capítulo trataremos el debate entre TP ligero y TP pesado. Es importante saber qué falta en el procesamiento de transacciones centrado en bases de datos. Por supuesto que además habrá un recuadro de debate en el que revelaremos nuestra postura en esta pugna. Jim Gray está de nuevo en lo cierto: "TP está donde haya dinero: tanto literalmente (los bancos son en su mayoría sistemas de TP) como en modo figurado (CICS ha generado más ingresos que cualquier otra pieza de software)." Así que quizás valga la pena explorar qué tipo de sistema de TP —ligero o pesado— es el que mejor responde a las necesidades de cliente/servidor.

LOS ORÍGENES DE TP LIGERO

Mis esperanzas en cuanto a las transacciones giran alrededor del impacto de la distribución del procesamiento, cuando se descubra que los datos no lo son todo y que el proceso es igualmente importante.

Jim Gray

En los viejos tiempos de las macrocomputadoras, las divisiones eran muy claras: los servidores de base de datos se concentraban en la administración de datos, mientras que los monitores de TP lo hacían en la administración de procesos y aplicaciones. Cada parte se mantenía alejada del terreno ajeno, empeñada en hacer cada vez mejor lo que le correspondía. Ésta era una situación clásica de beneficio mutuo, pues todos prosperaban por igual. Esta dichosa coexistencia llegó a su fin en 1986, cuando Sybase se convirtió en el primer proveedor de bases de datos en integrar componentes del monitor de TP en el mecanismo de las bases de datos.



Sybase rompe la tregua

¿Qué hizo Sybase? Se recordará que en la Cuarta parte explicamos que Sybase canaliza todas las solicitudes de clientes hacia un servidor de proceso único multihilos. Se trata de una *canalización* de N a 1. Puede llamársele un caso de *exceso de canalización*, porque la base de datos y las aplicaciones del usuario comparten el mismo espacio de direccionamiento: una segura invitación al desastre.

Pero Sybase no se detuvo en la canalización; también pasó a ser el primer proveedor de bases de datos en introducir procedimientos almacenados y desencadenantes, dos funciones que definitivamente corresponden al lado de la casa dedicado a procedimientos. Con esta nueva arquitectura, Sybase se convirtió en el ganador indiscutible de las guerras de evaluaciones comparativas entre bases de datos. Claro que la mayoría de los proveedores de bases de datos no se quedaron atrás. Hoy, casi todos ellos ofrecen algún nivel de canalización y soporte de procedimientos almacenados en sus mecanismos de bases de datos. Los desarrolladores de aplicaciones y los proveedores de herramientas tampoco tardaron en explotar los beneficios de los procedimientos almacenados, con lo que hizo su aparición la arquitectura de *TP ligero* cliente/servidor.

Dada la popularidad de los servidores de bases de datos en las LAN de PC, ¿indica esto que los monitores de TP han muerto? ¿Son sencillamente un anacronismo heredado de la época de las macrocomputadoras? ¿El TP ligero integrado a bases de datos es la nueva plataforma por excelencia para los servidores de aplicaciones y OLTP? Es evidente que la respuesta a todas estas preguntas es no; de lo contrario, no les habríamos dedicado a los monitores de TP una parte entera de nuestro libro. Pero comencemos por revisar los hechos en forma fría y analítica. Cuando lleguemos al recuadro de debate externaremos nuestras opiniones sobre la dirección que está siguiendo todo esto.

¿Qué es TP ligero?

TP ligero es simplemente la integración de las funciones del monitor de TP a los mecanismos de bases de datos. Hasta ahora, sólo unas cuantas funciones del monitor de TP han sido objeto de esta integración, entre ellas la función de embarque, cierto nivel de canalización, administración de transacciones de función única y llamadas semejantes a RPC. Se ignora si los proveedores de bases de datos planean reinventar la rueda y desarrollar en TP ligero todas las funciones restantes del monitor de TP. La lista de funciones aún no implementadas es muy larga; la gente del monitor de TP lleva una ventaja de diez años.

¿Qué es TP pesado?

TP pesado son los monitores de TP tal como se les define en este capítulo. La nueva generación de productos de *TP pesado* para LAN de cliente/servidor incluye a CICS, Encina, Tuxedo, Pathway de Tandem, Top End y ACMS de Digital. Todos estos monitores de TP soportan la arquitectura cliente/servidor y permiten que las PC inicien desde el escritorio algunas transacciones de servidores múltiples muy complejas. Todos estos productos son soportados por herra-

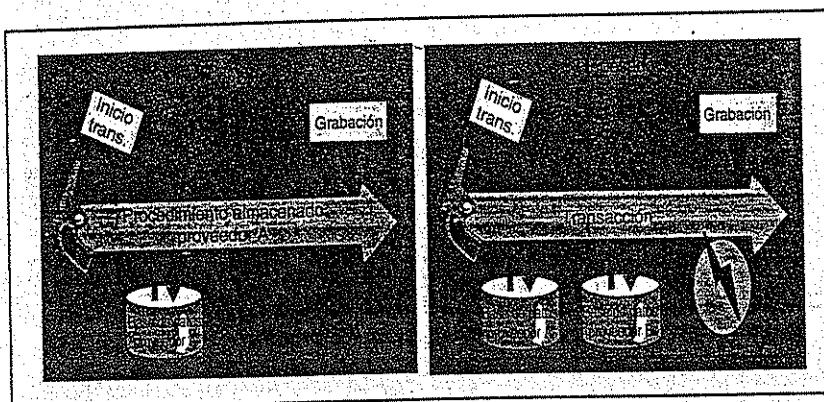
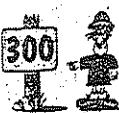


Figura 18-2. TP ligero contra TP pesado: Sincronización de administradores de recursos heterogéneos.

Sin embargo, el problema es que los gateways se construyen en el supuesto de que un procedimiento almacenado dentro de una base de datos única es toda la aplicación (así como el punto de origen de la transacción). Los gateways no permiten que aplicaciones múltiples (o procedimientos almacenados) participen en una transacción. También, lo atan a usted a un entorno de proveedor de base de datos de TP ligero propietario (o procedimiento almacenado). Por el contrario, TP pesado convierte sus aplicaciones en recursos neutrales.

TP ligero contra TP pesado: Administración de procesos

Un procedimiento almacenado de *TP ligero* es invocado, ejecutado bajo la protección de ACID (dentro de una grabación de fase única) y quizás retenido en la memoria caché para su futura reutilización. Esto es todo. En contraste, los procesos de *TP pesado* son preiniciados y administrados como clases de servidor (véase Figura 18-3). Si la carga de una clase de servidor resulta

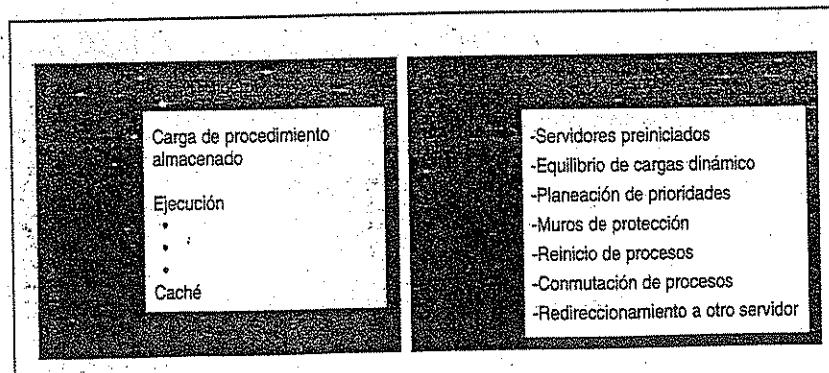


Figura 18-3. TP ligero contra TP pesado: Administración de procesos.



demasiado pesada, se inician automáticamente más procesos. Las clases de servidor soportan prioridades y otros atributos de clase de servicio. Los procesos del servidor cuentan con muros protectores en torno suyo para que los programas que corren en su interior no interfieran entre sí. Si un proceso de clase de servidor fracasa, se le reinicia o en su defecto la transacción puede ser reasignada a otro proceso servidor de la misma clase. El entorno en su conjunto opera bajo la permanente supervisión del monitor de TP. El concepto de clase de servidor contribuye a que el monitor de TP sepa qué clase de servicio necesita el usuario de un grupo particular de funciones. Se trata de un entorno administrado inteligentemente.

TP ligero contra TP pesado: Invocaciones cliente/servidor

La invocación de procedimiento almacenado de *TP ligero* es extremadamente ajena al estándar. Los proveedores ofrecen su propio mecanismo propietario de invocación de RPC. Las RPC no se definen por el uso IDL. Además, no están integradas a servicios de directorio global, seguridad y autenticación. Los vínculos de comunicaciones no se reinician automáticamente, ni están bajo la protección de transacciones. Adicionalmente, TP ligero no soporta intercambios de MOM o conversacionales.

Por el contrario, el entorno de *TP pesado* está más que abierto a diferentes estilos de comunicación (véase Figura 18-4). La RPC puede usar DCE como su base. Usted puede integrar fácilmente colas de MOM transaccional a la transacción global. La mayoría de los proveedores de monitores de TP también soportan APPC/CPC-I para comunicaciones de puerto a puerto.

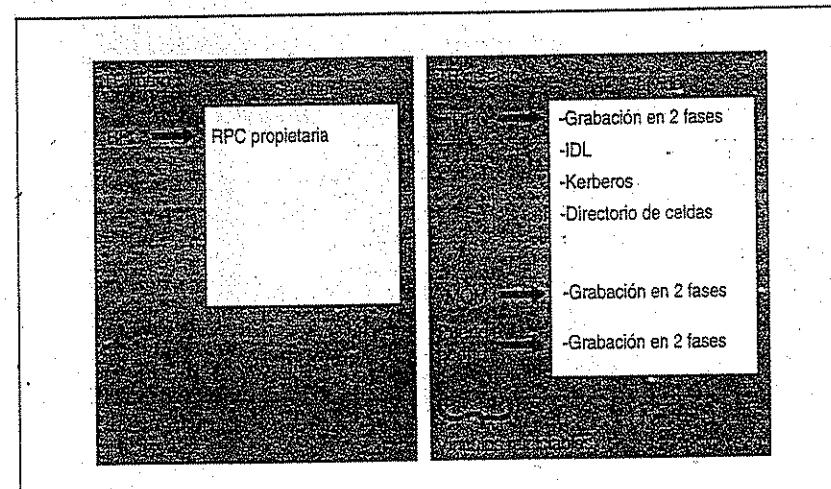


Figura 18-4. TP-Lite contra TP-Heavy: Invocación cliente/servidor.



años de incubación a un producto para desarrollar las facilidades adecuadas en áreas como distribución en línea de nuevos procesos, depuración remota, estadísticas integradas, herramientas de administración y conmutaciones automáticas durante fallas (y conciliaciones posteriores). Véase el recuadro de debate que aparece enseguida para opiniones más firmes al respecto.



Por fin: ¿TP ligero o TP pesado?

Debate

¿TP ligero o TP pesado? Lo más probable es que ninguno. El debate sobre la necesidad de un monitor de TP sólo es de interés a corto plazo, ya que las infraestructuras para transacciones actuales son inapropiadas para soportar la reingeniería de los procesos de negocios.

Gartner Group
(22 de noviembre de 1993)

Los atributos que fueron la causa del enorme predominio de TP ligero están a punto de convertirse en la causa de su relegamiento frente al renacer de TP pesado, disfrazado de sistemas de objetos.

Mohsen Al-Ghosein, arquitecto de TP
Microsoft
(Septiembre de 1995)

Roma no se construyó en una noche, como tampoco los monitores de TP. Y hasta donde nuestra vista alcanza, los monitores de TP aventajan considerablemente a TP ligero en el área de administración de procesos. TP ligero no se acerca siquiera a la administración de entornos en los que una transacción se extiende a lo largo de varias máquinas o administradores de recursos (las así llamadas transacciones de dominios múltiples). TP pesado ofrece administración global y permite que administradores de recursos de proveedores múltiples (incluidas bases de datos) se conecten al sistema, lo que nos permite elegir. Así pues, usted puede depender de TP pesado para crear el "mosaico" entero. Por el contrario, TP ligero brinda una solución de nivel básico, dominio único y base de datos única para el procesamiento de transacciones.

Por lo tanto, y como bicicleta, TP ligero es muy útil en situaciones en las que usted trata con una base de datos de proveedor único y un número de usuarios de reducido a intermedio. Además, así como las bicicletas nos permiten gozar de la experiencia de andar sobre ruedas, TP ligero les hará gozar a miles de programadores del procesamiento de transacciones. TP ligero es ideal en situaciones de nivel básico, porque es menos complejo; basta con un componente servidor: la base de datos. Los proveedores de TP ligero también saben hacer negocios en el ámbito de cliente/servidor, lo cual representa una ventaja muy importante.



Sin embargo, la tecnología de TP pesado es extremadamente importante para el futuro de la computación de cliente/servidor. Piense en lo que podría hacer con una Harley-Davidson en lugar de una bicicleta. Los monitores de TP nos permiten mezclar componentes en todo tipo de combinaciones, incluso en las más descabelladas; al mismo tiempo, garantizan que todo funcionará como relojito. En otras palabras, nos permiten proveer a la mezcla e igualación que distinguen al ámbito abierto de cliente/servidor. Desafortunadamente, los proveedores de monitores de TP enfrentan dificultades para transmitir su mensaje. Siguen usando terminología antigua, completamente ajena a la cultura de LAN de PC.

Los proveedores de TP pesado deberían centrar su atención en trasladar a cada escritorio y sistema operativo de 32 bits una versión "amigable" del monitor de TP Gartner (véase epígrafe anterior) quería que el procesamiento de transacciones pasara a la siguiente fase y comenzara a ocuparse del flujo de trabajo, tema de la Sexta parte. Estamos completamente de acuerdo. Mohsen Al-Ghosein, de Microsoft, piensa que el procesamiento de transacciones debería trasladarse a los objetos. También coincidimos plenamente con él (aunque sospechamos que se refiere a los objetos de OLE, mientras que nosotros nos inclinamos por los objetos de CORBA). Pero dada la mentalidad actual de los proveedores de TP pesado, nos daremos por satisfechos si vemos ocasionalmente un monitor de TP en un servidor de LAN de PC o Unix. No cuentan con la cantidad de emuladores o el empaquetamiento correcto de sus productos, y no pueden ir más allá de la "venta a MIS" de sus orígenes. No obstante, MIS y todos los demás nos beneficiaríamos enormemente de la ACIDación de todas las PC en redes. ¿No sería maravilloso que todas las PC pudieran participar en una transacción global? Creemos que la mayoría de los intercambios en el "futuro post-carestía" se realizarán en forma de transacciones globales.

Como se comprobará en el siguiente capítulo, los monitores de TP abiertos comienzan ya a mejorar radicalmente la imagen de OLTP. Consiguieron consolidarse en el mercado de cliente/servidor popularizando aplicaciones en 3 planos e introduciendo herramientas de GUI y middleware flexible como MOM y corredores eventuales. Sin embargo aún les falta mucho para atrapar los corazones y las mentes de la corriente principal de cliente/servidor. Su comercialización masiva sigue siendo insuficiente, y excepcional la "jerga CICS" en sus productos. La gente sigue creyendo que se necesita ser un experto en computación para tratar con monitores de TP (y transacciones en general).

Más adelante (después de que examinemos algunos conceptos más) le explicaremos por qué pensamos que la respuesta reside en los objetos distribuidos. En pocas palabras, los objetos pueden asumir las funciones provistas por monitores de TP, groupware y servidores de Internet. Además, cuentan con la forma indicada para colocar un administrador de transacciones en cada PC. Sí, el futuro es del embarque de funciones y cliente/servidor en 3 planos, pero esta visión se hará realidad por medio del middleware de "objetos de TP" y CORBA. No lo olvide: éste fue un recuadro de debate, y para la temprana etapa en la que aún nos encontramos usted ha recibido una dosis extrafuerte de opiniones. □



PANORAMA DEL MERCADO DE MONITORES DE TP

Hoy en día, el 90% del procesamiento de transacciones decisivo para misiones se realiza con uno de los más de 100,000 monitores de TP instalados.

**Jim Johnson, presidente
Standish Group
(Abril de 1995)**

De acuerdo con el Standish Group, los ingresos por concepto de software de monitor de TP correspondientes a 1995 fueron de \$1,186 millones de dólares. En el mismo año, las ventas totales de sistemas relacionados con monitores de TP —incluyendo hardware y herramientas— alcanzaron los \$17,000 millones. Hoy, el 90% de las aplicaciones de monitor de TP se hallan implementadas en sistemas de alto nivel como *CICS on MVS* de IBM, *IMS/TP* de IBM y *Pathway* de Tandem Computers. Sin embargo, una nueva generación de monitores de TP “abiertos” acaba de llegar al mercado, tales como *Encina*, *Top End*, *Tuxedo* y *CICS client/server*. Estos productos corren en múltiples sistemas operativos, y en su mayoría se basan en cliente/servidor (en oposición a los basados en terminales). Según Standish, las ventas de monitores de TP abiertos experimentaron un crecimiento del 382% de 1993 a 1994. Esta misma fuente supone que esos productos representarán más de la tercera parte de las ventas totales de monitores de TP en 1998 (véase Figura 19-1).

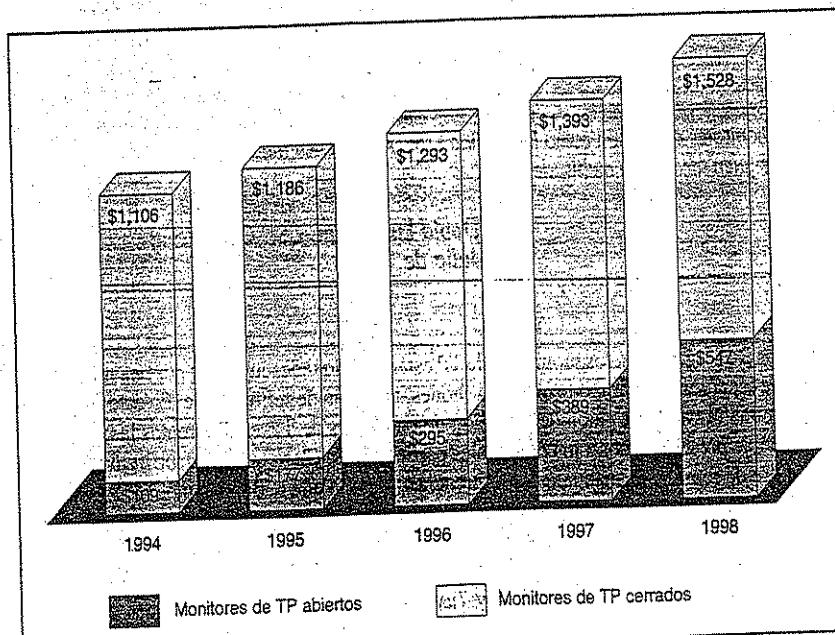
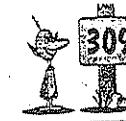


Figura 10-1 Ingresos anuales estimados (en millones de dólares) en monitores de TP. Fuente: Standish, 1995.



TENDENCIAS

Los monitores de TP compiten con los objetos por el puesto de plataforma de servidor de aplicaciones por excelencia para la computación de cliente/servidor en 3 planos, el segmento de más rápido crecimiento en el mercado de cliente/servidor. Además, persiguen nuevas fuentes de transacciones, como Internet, intranets y el comercio electrónico servidor a servidor. También en todas estas áreas enfrentan la sólida rivalidad de los objetos de CORBA. He aquí una breve descripción de los principales movimientos que advertimos en la industria de monitores de TP:

- *Los monitores de TP se convierten en entornos de servidor de aplicaciones exportables.* Los monitores de TP pueden correr ya entre casi todos los principales sistemas operativos para servidores. En consecuencia, constituyen entornos de aplicaciones servidor exportables. Usted genera una vez su aplicación en 3 planos y la exporta a los entornos de servidor de su gusto.
 - *Los monitores de TP se convierten en agentes de tráfico universales.* Además de soportar a sus clientes tradicionales, los monitores de TP ahora interceptan y enrutan llamadas de otros tipos de clientes, como Lotus Notes, Internet y MOM.
 - *Los monitores de TP se convierten en corredores de recursos.* Aparte de bases de datos de SQL, los monitores de TP ahora soportan todo tipo de administradores de recursos de segundo plano, como multiplicidad en los sistemas de archivos, bases de datos jerárquicas, colas persistentes, almacenes de imágenes, depósitos de HTML y bases de datos de documentos de Lotus Notes.
 - *Los monitores de TP descubren las herramientas de cliente/servidor.* Además de sus venerables mesas de trabajo de COBOL, los monitores de TP ya se asocian con conocidas herramientas de cliente/servidor, como Powersoft de Sybase, SuperNOVA 4GL de Four Season Software, Jam 74GL de JYACC, Delphi de Borland, Parts de Digitalk, Centura de Gupta y VisualAge de IBM. Las cosas están llegando al grado en que usted puede acceder a servicios de monitor de TP desde cualquier herramienta que soporte DLL, como Visual Basic, por ejemplo. También una nueva generación de herramientas de cliente/servidor en 3 planos soporta monitores de TP, incluidas Dynasty, Open Horizon, Forte, Magna, Unify Vision y VisualGen de IBM. Los proveedores de monitores de TP aprendieron la lección: sin herramientas equivalentes, los programadores no pasarán del desarrollo en 2 planos a 3.
 - *Los monitores de TP conocen los objetos.* En la actualidad, la mayoría de los monitores de TP ofrecen bibliotecas clase C++ para acceder a sus servicios. Algunos de ellos permiten que clientes de CORBA llamen a sus servicios mediante interfaces de CORBA definidas por IDL. Además, ciertos monitores de TP ya implementan interfaces con el servicio de transacciones de objetos definido por CORBA. Esto permitirá que aplicaciones administradas por monitores de TP participen en transacciones globales con objetos de CORBA. Explicaremos las implicaciones de esto en la Séptima parte.

La buena nueva en cuanto a los monitores de TP es que se hallan muy bien situados para conquistar el mercado de cliente/servidor en 3 planos, en rápido crecimiento. Se espera que el mercado de 3 planos crezca al 20% de las aplicaciones totales de cliente/servidor en 1997 (sobre el 5% actual). La mala noticia es que los monitores de TP están quedando en medio entre

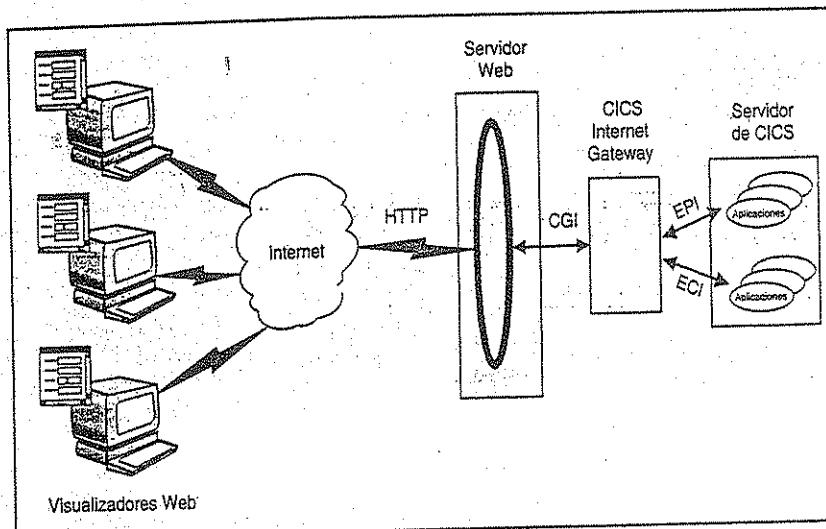


Figura 19-3. El monitor de TP y el Web: El CICS Internet Gateway.

usuarios registrados. El precio de nivel básico para 10 usuarios es de \$3,500 dólares, lo que hace a CICS más accesible para usuarios de nivel bajo.

Tuxedo de BEA/Novell

Tuxedo es una de las únicas tecnologías que permiten a la gente construir aplicaciones distribuidas que son agnósticas de OS.

Sheldon Laube, CTO, Novell
(Septiembre de 1995)

Tuxedo de BEA/Novell, originalmente de Unix Systems Lab (USL), ofrece un entorno de monitor de TP que corre en más de 36 plataformas de Unix, como HP-UX, AT&T GIS, AIX, Solaris y OSF/1. Cuenta con soporte de cliente en DOS, Unix, OS/2 y OS de Mac. Además, Tandem lanzó en 1995 una "personalidad" paralela a Tuxedo encima de su monitor de TP Pathway. Dependiendo de la fuente, Tuxedo posee actualmente entre el 25 y 32% del mercado de monitores de TP para Unix, menos que en 1992, cuando alcanzó el 50%.¹ Sin embargo, sus ingresos no dejaron de crecer en 1995 en un saludable 140%.

En 1995, Novell decidió finalmente sacar a Tuxedo de su encierro en Unix y colocarlo como una tecnología clave de middleware para el enlace de Windows NT, Unix, OS/2 y sistemas de

macrocomputadoras a redes con NetWare. A fines de ese mismo año, Novell lanzó *Tuxedo 6.1*, que corre en Unix, NetWare y NT. Sin embargo, con la desaparición del SuperNOS de Novell en el mismo periodo, algunos observadores de la industria previeron la posibilidad de que Tuxedo entrara al piso de remates. No se equivocaron. En febrero de 1996, Novell cedió el control del desarrollo de no NetWare y la responsabilidad de distribución de Tuxedo a BEA Systems, sólida y joven empresa de California.²

Tuxedo 6.1 incursiona en nuevos terrenos con las siguientes características:

- *Integración del servicio de directorio de NetWare* (NDS: *NetWare directory service*). Novell ofrece un producto de gateway integrado, llamado *NetWare TransactionLink*, que corre únicamente en servidores de NetWare. Permite que clientes de NetWare usen NDS para acceder a aplicaciones de Tuxedo aparte de hacerlo a sus ya existentes servicios de archivo de NLM, impresión y correo electrónico. Bastaría con hacer clic en objetos en un árbol de NDS para correr aplicaciones en servidores remotos de Tuxedo. Eventualmente Novell planea remplazar el espacio para nombres de Tuxedo por NDS para ofrecer un directorio común que permita a los clientes acceder inconsútilmente a aplicaciones de Tuxedo y recursos de NetWare. Cabe indicar que actualmente sólo se dispone de NDS sobre NetWare. Así, no se moleste en esperar esta gran unificación.
- *Administrador de aplicaciones gráficas*. Tuxedo 6.1 incluye una utilería basada en Motif, llamada *Application Manager*, que permite administrar y manejar servidores de Tuxedo remotos. Esta utilería puede vigilar aplicaciones y suministrar estadísticas de desempeño en una base de información de administración (MIB: *management information base*) de SNMP. Si se da un cuello de botella, el software puede reconfigurar parámetros "sobre la marcha" y recuperar el equilibrio de cargas y la planeación de prioridades.
- *Soporte de SNMP*. Tuxedo 6.1 proporciona un agente de SNMP y MIB. Esto significa que famosas estructuras de administración empresarial —como *OpenView* de HP y *SystemView* de IBM— pueden acceder remotamente a la información reunida por el *Application Manager* de Tuxedo.
- *Correaje de eventos de publicación y suscripción*. Tuxedo cuenta ahora con un *Event Broker* que media entre publicadores de eventos y suscriptores. El corredor notifica automáticamente a los suscriptores cuando un evento que los afecta ocurre; por ejemplo, cuando el precio de ciertas acciones excede un límite. Con Tuxedo usted puede registrar las acciones que éste debería emprender cuando ocurre un evento. Puede activar una serie de pasos de procesamiento entre aplicaciones múltiples en respuesta a eventos especificados

¹ BEA no es un proyecto ordinario. Su meta es convertirse en la "compañía de las transacciones". Su propósito empresarial se centra, exclusivamente, en la construcción e integración de monitores de TP. Además de Tuxedo, adquirió el mecanismo de CICS sobre Unix de VISystems y a los dos proveedores principales de servicios de Tuxedo a nivel empresarial, Information Management Company (IMC) e Independence Technologies, Inc. (ITI).



embargo, no olvide que la mayoría de las soluciones de Top End se ofrecen en paquete; AT&T no abandonará su camino para vender directamente a MIS monitores de TP.

Component Coordinator de Microsoft

Después de analizar los sistemas de procesamiento de transacciones más complejos del mundo nos preguntamos: ¿qué se necesita para que las PC o redes de PC sean aptas para estas tareas?

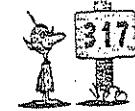
Bill Gates, presidente
Microsoft
(Enero de 1995)

Microsoft competirá con los proveedores de TP establecidos por el control de nuevas formas emergentes de trabajo de producción distribuida.

Gartner Group
(Enero de 1995)

Aunque Microsoft no participa por ahora en el mercado de monitores de TP, ha contratado a algunos de los mejores talentos de OLTP. En 1995 emitió una especificación preliminar para *OLE/Transactions*, la cual es rival directa del servicio de transacciones de objetos (OTS: object transaction service) de CORBA. No obstante, no cabe duda de que Microsoft piensa que el futuro del software está en las arquitecturas en 3 planos y los objetos basados en OLE. Las transacciones son el vínculo que permitirá que todo esto trabaje al unísono.

Microsoft planea convertirse en uno de los principales participantes en el mercado de transacciones adoptando un método de la base a la cima para el procesamiento de transacciones. Para empezar, planea incorporar sin costo administradores y registros cronológicos de transacciones basados en OLE en su software principal, incluidos sistemas de archivo, SQL Server, Excel, Access, Exchange y el Internet Information Server (IIS). Después ofrecerá un Component Coordinator que combine la función de un ORB empresarial y un monitor de TP. Este elemento coordinará objetos de OLE y otros recursos a través de una empresa usando principios de ACID. En su versión definitiva, prevista para 1997, este coordinador soportará interacciones de MOM, así como invocaciones de objetos semejantes a RPC. Era de esperarse una versión reducida del Component Coordinator en SQL Server 6.5 (previsto para fines de 1996). Se trata de un coordinador de transacciones distribuidas que soporta grabación en dos fases y el estándar de XA. Aunque esta versión no soporta OLE Automation, incluye una API de OLE/TP que da acceso a programadores de C++ a funciones de programación de transacciones.



Pathway de Tandem

Pathway de Tandem es un monitor de TP de interfaz final para sistemas de MPP de acoplamiento holgado. Posee alrededor del 10% del mercado de monitores de TP, pero en los últimos años fue objeto de transformaciones radicales. En 1993 añadió soporte para una arquitectura cliente/servidor con una RPC para transacciones propietaria llamada RSC. Ésta ofrece una RPC delegada que corre en todas las plataformas de cliente, tales como DOS, OS/2, Windows, NT, Mac y casi todas las modalidades de Unix que se conocen. En 1995, más del 70% de las nuevas aplicaciones de Pathway ya se basaban en el modelo de cliente/servidor. Pathway se jacta de poseer algunas de las mayores aplicaciones cliente/servidor (las cuales soportan en algunos casos hasta más de 17,000 clientes).

Pero la historia de Pathway no termina ahí. En 1994, Tandem introdujo una arquitectura de "personalidad múltiple" por encima de su mecanismo central de transacciones. Esto significa que Pathway puede ejecutar API de monitores de TP de terceros además de las propias. En 1995, Tandem lanzó una personalidad paralela a Tuxedo encima de Pathway. En 1996 esperaba lanzar una personalidad paralela a CICS, una personalidad de servidor de comercio en Internet y una personalidad CORBA basada en SOM de IBM. Una aplicación de monitor de TP que corre en cualquiera de estas personalidades puede invocar servicios de aplicaciones de cualquier otra personalidad; el mecanismo del monitor de TP funge como un enorme conmutador. Además, el monitor de TP puede equilibrar dinámicamente la carga de los procesos para que corran en miles de procesadores holgadamente acoplados. Si alguno de éstos falla, el monitor de TP proporciona intercambio automático.³

CONCLUSIÓN

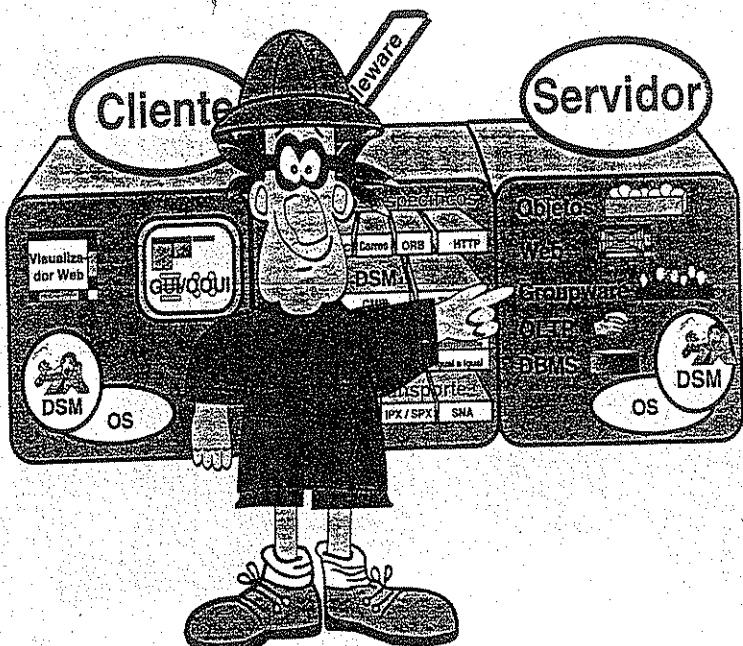
Como puede verse, los monitores de TP han recorrido ya un largo camino. Sus proveedores han tenido éxito en la creación de la necesidad de sus productos en el ámbito de las aplicaciones cliente/servidor en 3 planos. Además, los monitores de TP no dejan de reinventarse y hallar nuevos usos a sus servicios. Los proveedores de bases de datos seguirán combatiendo a los monitores de TP y las aplicaciones de 3 planos en general.

El mayor reto para los monitores de TP provendrá de los proveedores de ORB de CORBA. Los ORB de CORBA ofrecen el middleware ideal para el funcionamiento de componentes distribuidos capaces de participar por igual en los tres planos. Adicionalmente, los monitores de TP ya comienzan a descubrir los objetos, aunque siguen situados en la periferia de CORBA. En otras palabras, ven a CORBA sólo como un cliente más, y a los objetos como un medio para empaquetar bibliotecas de clase.

Como lo comprobará usted más adelante, CORBA es mucho más que sólo otro medio para emitir una RPC. Brinda la mejor plataforma de middleware abierto sobre la que los monitores

³ Advertencia: Si esto le parece demasiado deslumbrante, es porque uno de los autores de este libro dirigió el desarrollo de la línea de productos Pathway en los últimos cuatro años. Sin embargo, ella asegura que en este caso la publicidad no miente.

Introducción a la Sexta parte



Introducción a la Sexta parte

Sí es de quienes piensan que en la variedad está el gusto, groupware es para usted.

Joe Paone, Internetwork
(Octubre de 1995)

Si las transacciones ACID y los monitores de TP les parecieron divertidos, amigos marcianos, espérense a conocer el groupware. Y, a propósito, tenemos para ustedes un juego terrícola: se llamará "Los ciegos y el elefante". Les vendaremos los ojos a ver si adivinan qué es groupware. El que gane pasará una noche en la ciudad. ¡Listos?

Vamos a ver entonces. ¿Qué es groupware? La respuesta del marciano número uno es: "Correo electrónico." El marciano número dos dice que es un "almacén de documentos" para multimedia. Y el marciano número tres afirma que los otros dos están equivocados: "Pero si es flujo de trabajo!" ¿Tenemos otros participantes? ¿Alguien cree que tiene que ver con conferencias electrónicas? ¿Con calendarización y planificación grupales? Como se deduce de la caricatura, el elefante de groupware es todo lo mencionado. ¡Está bien, está bien! Todos pasarán una noche en la ciudad. ¿Quieren un guía de fiestas? Tenemos un voluntario.

La Sexta parte trata de la categoría amorfia de cliente/servidor conocida como groupware; amorfia porque es tan reciente que aun no conocemos del todo sus potencialidades. Los promotores de groupware afirman que su tecnología nos permite crear nuevas clases de aplicaciones cliente/servidor, diferentes a todas las que hemos visto en macro y minicomputadoras. Esto es posible gracias a los elementos de persona a persona en las comunicaciones de cliente/servidor. La revolución de las PC fue construida alrededor de la computación *personal*; groupware puede provocar una revolución de software análoga en torno a la computación *interpersonal*. Claro que esto incluye a los marcianos. ¿Es lucrativo el groupware? Al menos así debe creerlo IBM: pagó \$3,500 millones de dólares por adquirir Lotus Notes, el producto de groupware por excelencia. Si alguien está dispuesto a girar un cheque de \$3,500 millones de dólares por una tecnología, es que debe tratarse de algo grande. ¿Qué dónde está Groupware Valley? Los marcianos siempre han de llevarnos la delantera!

Empezaremos por definir el groupware. (No se rían: *podemos* hacerlo.) Despues trataremos de lo que diferencia al groupware de las bases de datos de SQL y los monitores de TP. Luego examinaremos las tecnologías constitutivas del groupware, entre ellas el procesamiento de documentos con multimedia, el flujo de trabajo, el correo electrónico, conferencias y calendarización y planificación grupales. Claro que, en lo que se refiere al groupware, el todo es mayor a la suma de sus partes, de manera que tendremos que analizar de dónde surge esa sinergia. Terminaremos con *Lotus Notes* y sus competidores más cercanos, entre ellos *Collabra Share* de Netscape, *Exchange* de Microsoft y *GroupWise XTD* de Novell.

que Lotus Notes está muy cerca de conseguirlo. Después de hacernos de una definición práctica, analizaremos las tecnologías de base y cómo groupware combina las piezas dentro del marco de cliente/servidor.

¿POR QUÉ ES IMPORTANTE EL GROUPWARE?

La gente ya no habla de los sistemas operativos; sencillamente parte del supuesto de que existen. En definitiva, el groupware evolucionará de la misma manera.

Esther Dyson, editora, versión 1.0

La empresa de investigación de mercado Workgroup Technologies calcula que el mercado de groupware crecerá a casi \$6,500 millones de dólares en 1998, desde un nivel de \$2,300 millones en 1995. Los segmentos de más rápido crecimiento serán el correo electrónico, el flujo de trabajo y los servicios de implantación de groupware. Lotus Notes — producto amorfó de groupware que escapa a toda definición— ha vendido más de 5 millones de licencias a 7,000 compañías. Lotus prevé que Notes correrá en 20 millones de sedes para fines de 1997. De acuerdo con Dataquest, los ingresos por concepto de venta de Notes se dispararán a \$1,200 millones de dólares en 1998, sobre una base de \$261 millones en 1994. Esto se traduce en una tasa compuesta de crecimiento anual de 52%.¹ Por su parte, el Input Research Institute prevé que los ingresos por productos y servicios de Notes ascenderán a \$4,000 millones de dólares en 1999. Lotus calcula que por cada dólar en ventas de Notes, los distribuidores obtienen otros 7 u 8 en diseño de aplicaciones, administración de sistemas y capacitación.

¿Cuál es la causa de este súbito interés por el groupware? De acuerdo con David Coleman, director del boletín "GroupTalk", este rápido crecimiento se debe a que el groupware puede transformar a una compañía al modificar el estilo de comunicación entre las personas y, en consecuencia, el cambio en los procesos de negocios. Por ejemplo, se puede usar groupware para automatizar el servicio a clientes y volver más sensible a una compañía. Groupware posee también el potencial para aplanar a las organizaciones y eliminar capas de burocracia (véase el siguiente recuadro de debate).

Groupware permite que participantes directos —dondequiera que se encuentren— colaboren en una labor por medio de redes de cliente/servidor. Prevemos el crecimiento de "empresas virtuales" formadas por grupos no afiliados de personas que colaborarán en proyectos específicos. Groupware ayuda a administrar (y rastrear) el producto a lo largo de sus diversas fases; asimismo, permite que los participantes intercambien ideas y sincronicen su trabajo. Sirve de registro de la "memoria colectiva" del grupo.

El groupware hace posible en muchos casos que los departamentos desarrollen y desplieguen sus propias aplicaciones. Cualquier persona capaz de crear una simple hoja de cálculo puede aprender a crear una aplicación de Lotus Notes, para la que se requieren escasas habilidades de programación.

¹ Estas cifras, reunidas antes de que IBM adquiriera a Lotus, no incluyen la considerable minindustria que se ha desarrollado alrededor de Notes.

La habilidad de los departamentos para desarrollar y crear sus propias aplicaciones de groupware de cliente/servidor está dando como resultado rendimientos de inversión fenomenales. Por ejemplo, en un estudio sobre 65 usuarios de Notes realizado por IDC en 1994 se constató que el retorno de inversión iba del 16 al 1,666% en una inversión mediana de \$100,000 dólares. En más de la mitad de los casos se presentaron rendimientos superiores al 100%, y una cuarta parte obtuvo rendimientos mayores al 200%. En vista de estas extraordinarias cifras, IDC repitió el estudio con una nueva serie de usuarios y obtuvo resultados similares. El fenómeno groupware —como las hojas de cálculo y las hypercards de Macintosh— se nutre solo. La diferencia es que groupware es una aplicación cliente/servidor autorregeneradora; es cuestión de redes y relaciones interpersonales. Casi todos los productos de groupware soportan también API abiertas, que permiten que terceros (y tiendas de IS) añadan nuevas funciones por encima de la fundación.



Groupware y reingeniería

Debate

¿Estamos invirtiendo en groupware para infundir energía de colaboración en las organizaciones? ¿O lo estamos haciendo sobre todo para detener la hemorragia?

Michael Schrage, académico
MIT Sloan School

La burocracia en la mayoría de las organizaciones es muy resistente; hará falta mucho más que una buena dosis de groupware para eliminarla. De hecho, el groupware puede ser mal utilizado y dar lugar a la automatización de la burocracia, con lo que su permanencia quedaría garantizada. En el actual movimiento de "reingeniería" se piensa que se ha descubierto el problema: se han estado aplicando a las operaciones empresariales los procesos de línea de ensamblaje de Ford. Es preciso replantear por lo tanto nuestra manera de trabajar; esto es, proceder a la reingeniería del proceso. Les deseamos suerte. Ojalá consigan deshacerse de algunos puestos con la reingeniería.

En el movimiento de la reingeniería se afirma que no sirve de nada lanzar tecnología a un proceso con desempeño deficiente. Estamos de acuerdo; nadie pretende oponerse a la evidencia de que no tiene sentido automatizar un proceso que, para comenzar, ni siquiera debería existir. Sin embargo, el groupware es una tecnología heterodoxa; puede automatizar todo tipo de procesos, incluyendo los malos. Puede servir para automatizar procesos ineficientes y volverlos "eficientemente ineficientes". O puede automatizar las estructuras producto de la reingeniería y agilizarlas para alcanzar mejoras de la magnitud que los gurús prometen. Pero es probable que veamos muchos casos de mal uso de la tecnología hasta que la gente aprenda esta lección. Replantear nuestra manera de trabajar requiere mucho mayor esfuerzo que echarle un paquete de groupware comercial encima. De todas maneras los proveedores de groupware amasarán fortunas, pero el valor de sus productos para las compañías que los adquieran será radicalmente diferente. □



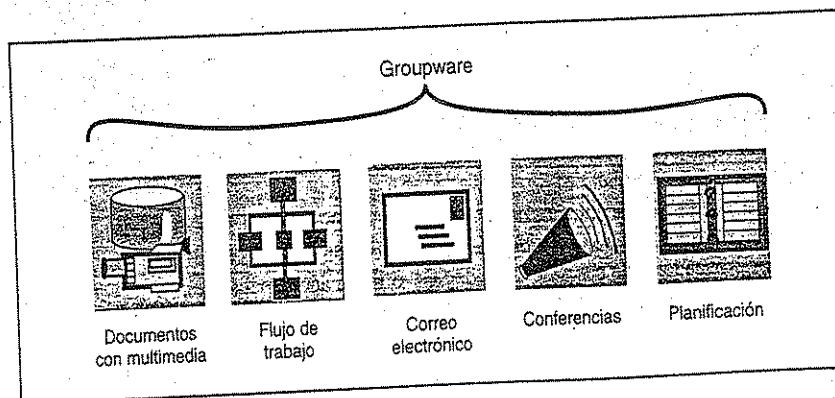
¿En qué se diferencia el groupware de los monitores de TP?

Los monitores de TP fueron tratados en la Quinta parte, sobre la administración de procesos de transacciones en redes de cliente/servidor. ¿Qué resulta de compararlos con el groupware? En lo que atañe a almacenes de documentos, los monitores de TP pueden complementar adecuadamente al software de groupware. El monitor de TP trata al almacén de documentos como a cualquier otro administrador de recursos. Si soporta grabación en dos fases, el monitor de TP coordinará gustosamente una transacción distribuida que incluya al almacén de documentos. Sin embargo, los monitores de TP y el groupware compiten en el área del flujo de trabajo: pensamos que el flujo de trabajo del groupware es una tecnología mucho más desarrollada que la transacción de larga vida del monitor de TP (aunque menos protegida).

El actual modelo de flujo de trabajo, y de groupware en general, no está orientado a las transacciones en el sentido de ACID. El groupware es apto para reflejar los cambiantes estados de la información a través del tiempo, pero no se conduce del todo bien en lo que respecta a reflejar el estado de los datos en tiempo real en un momento dado. Por ejemplo, el groupware no utiliza grabaciones en dos fases para sincronizar cambios distribuidos entre administradores de recursos. Sería magnífico que monitores de TP y groupware combinaran esfuerzos para dotar al flujo de trabajo de propiedades ACID (si esto fuera posible). Abordaremos el flujo de trabajo más adelante.

LOS COMPONENTES DEL GROUPWARE

Como ya se dijo, el groupware se apoya en cinco tecnologías básicas: administración de documentos con multimedia, flujo de trabajo, correo electrónico, conferencias y planificación (véase Figura 20-1). Su magia es producto de la combinación de estas tecnologías y la creación de una nueva sinergia. La tecnología de administración de documentos con multimedia y flujo de trabajo procede de los sistemas de proceso de imágenes de documentos electrónicos; el correo electrónico y la planificación provienen de la automatización de oficinas, y las conferencias son cualidad propia.



Pero antes de abordar directamente el groupware, veamos rápidamente qué ofrecen estas tecnologías componentes.

Del proceso electrónico de imágenes a la administración de documentos con multimedia

La tecnología de administración de documentos del groupware echa raíces en el proceso electrónico de imágenes. Si quisieramos ser puristas, dirímos que el proceso electrónico de imágenes es sólo otra modalidad de groupware de propósito especial. Claro que la gente del proceso electrónico de imágenes —creadora de una gran industria multimillonaria— podría reponer que el groupware es apenas una variante del proceso de imágenes. En todo caso, tenemos que ocuparnos del proceso electrónico de imágenes, porque se trata de una importante industria de cliente/servidor precursora del groupware.

El proceso electrónico de imágenes comenzó en pequeño. En la década de los sesenta, muchas empresas remplazaron enormes bodegas de documentos informativos por microfilmes y sistemas de recuperación asistidos por computadora. A mediados de los ochenta, la aparición de PC, LAN, exploradoras, tableros de compresión y estuches de disco óptico permitió la automatización del almacenamiento de imágenes, así como los sistemas de rastreo centrados en datos para la localización de esas imágenes. Estas nuevas tecnologías volvieron económicos el almacenamiento y despliegue de imágenes en línea. En algunas aplicaciones, los ahorros en costos asociados con la reducción de personal y un más veloz acceso en línea a documentos (en segundos y ya no en días) justificaron el incremento en los costos de los nuevos sistemas de cliente/servidor. Llenar un archivero de cuatro cajones cuesta \$25,000 dólares, y mantenerlo \$2,160 al año. Pero, sobre todo, el 3 por ciento del papel se pierde; el costo promedio de recuperación de un documento es de \$120,00 dólares. Se calcula que sólo en las empresas estadounidenses se acumulan 3,000 millones de documentos en papel; cifras de este tipo fueron la causa de que surgiera la industria del proceso electrónico de imágenes.

Arquitectura cliente/servidor de proceso electrónico de imágenes

Los sistemas de proceso electrónico de imágenes son por definición aplicaciones cliente/servidor orientadas a bases de datos (véase Figura 20-2). Las PC clientes toman y manipulan las imágenes; funguen como elementos de primer plano para los datos almacenados en servidores de imágenes. La PC cliente hace por lo general estas cosas:

- La exploradora integrada a la PC del cliente digitaliza la imagen por medio de un proceso similar al de las máquinas de fax. (No es casual que en ocasiones las máquinas de fax sirvan como exploradoras remotas.)
- Después de ser digitalizada, la imagen se exhibe y se comprueba su calidad; se le vuelve a explorar en caso necesario.
- Mientras la imagen se despliega, un operador extrae información de la imagen, e introduce los datos en los campos de un formato de GUI. Como mínimo, se le asigna al documento

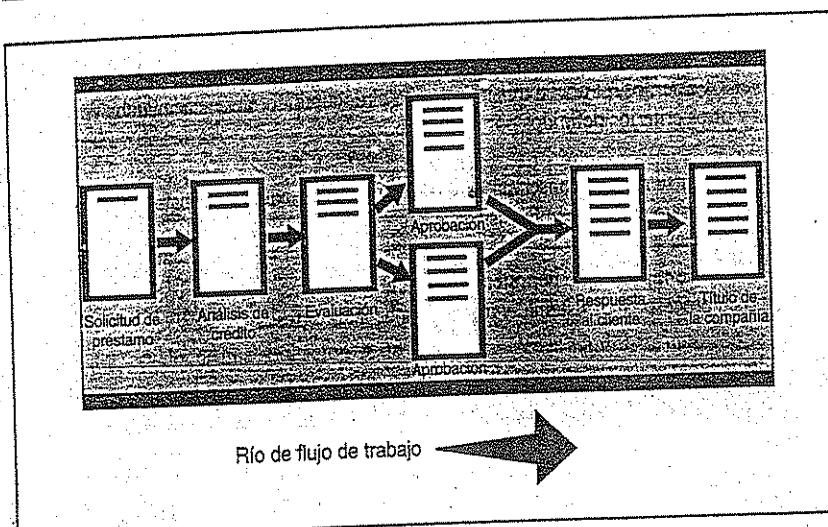


Figura 20-3. El río del flujo de trabajo.

entorno sumamente dinámico. Algunos flujos de trabajo pueden ser difusos y no del todo comprensibles; otros son deterministas y altamente repetitivos. En todos los casos, estos flujos de trabajo nos ayudan a colaborar en bien del cumplimiento de las labores.

Para apreciar el verdadero significado del flujo de trabajo, es necesario conocer sus orígenes. La tecnología del flujo de trabajo tiene raíces históricas aferradas firmemente al ámbito de la administración de imágenes y de la tecnología de manufactura integrada por computadora. FileNet Corporation —proveedor de proceso de imágenes— fue uno de los precursores de esta tecnología en 1984. Ésa y otras compañías de proceso electrónico de imágenes —como ViewStar, Sigma, Wang e ImagePlus de IBM— descubrieron que el flujo de trabajo podía ser útil para automatizar procesos de gran volumen anteriormente basados en papel (véase Figura 20-4).

El flujo de trabajo es especialmente aplicable a "fábricas de papel", es decir, a grandes oficinas que procesan rutinariamente documentos que representan transacciones de negocios (préstamos, procesamiento de reclamaciones y declaraciones fiscales, por ejemplo). El papel es a estas fábricas lo que la materia prima a la manufactura: "la molienda del molino" que produce los bienes de las organizaciones. Cuando el papel se convirtió en imagen electrónica, el flujo de trabajo automatizó el desplazamiento de los documentos de una operación de procesamiento de imágenes a otra. Tanto el controlador del flujo de trabajo como el trabajo mismo son reproducciones electrónicas de formaciones industriales reales.

Los sistemas de flujo de trabajo de proceso de imágenes son costosos, rígidos y centralizados, y por lo general implican la intervención de un profesional de IS altamente calificado para la realización del diseño y la integración. Tienden a ser propietarios y no cuentan con interfaces adecuadas con otras aplicaciones.

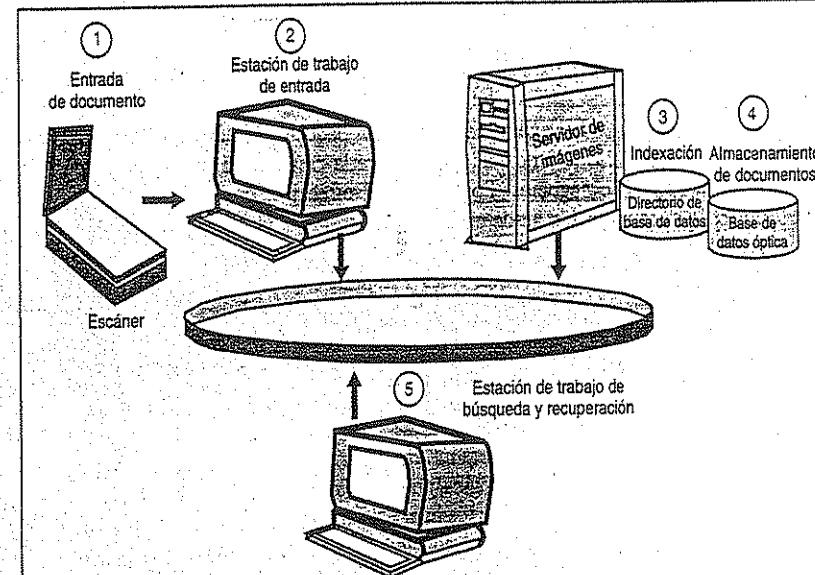
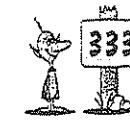


Figura 20-4. Flujo de trabajo en sistemas de proceso electrónico de imágenes.

En cuanto a sus rasgos positivos, pueden manejar enormes cargas de trabajo y poseen excelentes controles integrados de seguridad y versión. Son muy aptos para la planificación de tareas relacionadas con documentos, que rastrean hasta su conclusión. Su costo de inicio es de \$3,000 dólares por sede.

Los nuevos sistemas de flujo de trabajo

El groupware introduce una nueva especie de software de flujo de trabajo de cliente/servidor para las masas. Los nuevos paquetes de flujo de trabajo llegan más lejos que sus contrapartes de proceso de imágenes en las siguientes áreas:

- **Soporte de necesidades específicas del usuario.** Los nuevos paquetes de flujo de trabajo resuelven necesidades de automatización de procesos tanto estructurados como no estructurados. Pueden automatizar procesos perfectamente entendibles, así como procesos más nebulosos.
- **Bajo costo.** Los nuevos paquetes de flujo de trabajo se venden a precios de software para PC; se paga de \$100 a \$500 dólares por sede.
- **Integración con otras aplicaciones.** Los nuevos paquetes de flujo de trabajo pueden integrarse con aplicaciones existentes ya sea generando un proceso o enviándoles algún tipo de

- El *flujo de trabajo ad hoc* se ocupa de procesos de trabajo de vida corta y no estructurados. Éstos pueden involucrar a grupos de personas a cargo de un problema común. Piénsese en un proyecto de corta duración con un plazo límite. El flujo de trabajo sirve para asignar roles, rastrear y enrutar el trabajo en proceso, vigilar los plazos y controlar quién hizo qué y cuándo. Es una herramienta excelente para el seguimiento de trabajo entre personas físicamente dispersas. Este tipo de flujo de trabajo es como conducir un automóvil. La navegación se centra en el conductor, pero se necesitan señalizaciones y un mapa para deducir el destino. El conductor también debe conocer la serie de opciones disponibles en cada curva. El flujo de trabajo ad hoc sirve para la automatización creciente, aunque todo aquello que el sistema no puede manejar se deja en manos humanas. Aprovecha en su totalidad la potencia del escritorio para ayudar a la gente a circular por los caminos de la región.

Rutas del flujo de trabajo

Los paquetes de flujo de trabajo moderno soportan el mismo tipo de topologías comunes en las comunicaciones humanas (véase Figura 20-5). Por lo general permiten especificar una ruta que define las operaciones establecidas que recorre una unidad de trabajo. Asimismo, le permiten definir reglas para la especificación de condiciones de aceptación en el desplazamiento de una operación a la siguiente. Se pueden crear rutas secuenciales, rutas paralelas (vías alternas), rutas con lazos de retroalimentación (de repetición de trabajo, por ejemplo), rutas circulares, rutas radiales (de rayos de rueda) y rutas totalmente interconectadas. Las primeras cuatro rutas se usan en los flujos de trabajo orientados a procesos, mientras que las dos últimas se usan en flujo de trabajo ad hoc.

Divisiones y uniones de flujo de trabajo

Los objetos de flujo de trabajo pueden tomar diferentes rutas y volverse a unir después en una sola en un punto de "reunión". Además, un objeto de flujo de trabajo puede dividirse en múltiples partes y unirse en una sola en su recorrido por el río del flujo de trabajo (véase Figura 20-6). Esto se efectúa por medio de divisiones y uniones, tal como se explica en los siguientes ejemplos:

- Las *divisiones y (and-splits)*, sirven para hacer explotar un objeto en muchas partes. Por ejemplo, un conjunto de chips en una placa es rastreado como grupo hasta que los chips se dividen; después cada uno sigue su propio camino.
- Las *divisiones o (or-splits)* sirven para desprender unas cuantas partes de un grupo. Por ejemplo, unos cuantos chips pueden dividirse de la placa para su prueba al azar; volverán al grupo más tarde mediante una unión o.
- Las *uniones o (or-joins)* permiten que ciertos miembros se reintegren al grupo. Por ejemplo, en una línea de manufactura, una parte defectuosa puede ser destinada a una operación de repetición de trabajo; una vez reparada, puede reintegrársele al grupo por medio de una unión o.

- Las *uniones y (and-joins)* son puntos de reunión que permiten la reagrupación de objetos a fin de que sigan una ruta en calidad de grupo. Por ejemplo, usted puede usar una unión y, para empaquetar muchas unidades en un contenedor que pueda embarcarse como unidad.

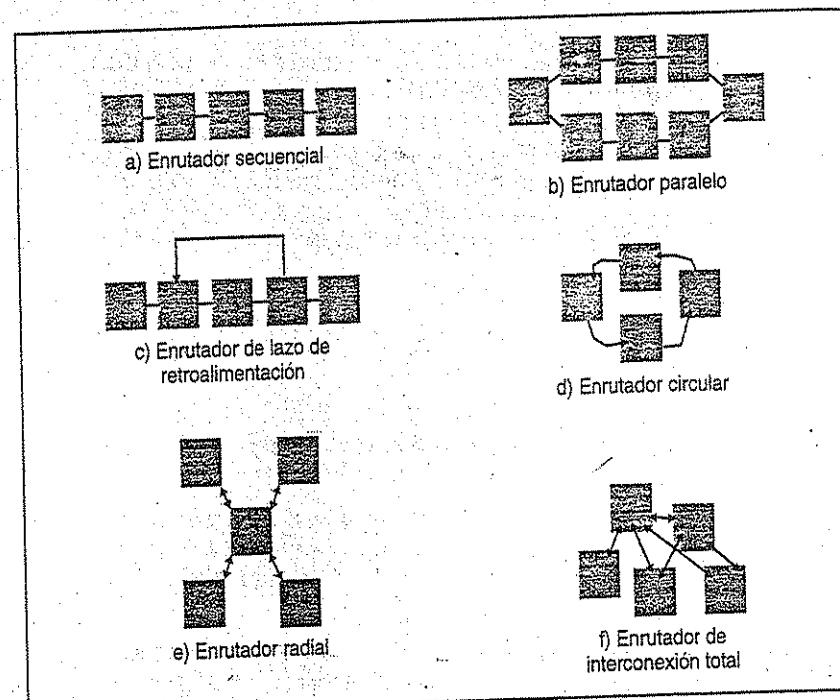


Figura 20-5. Los flujos de trabajo pueden adoptar toda clase de patrones.

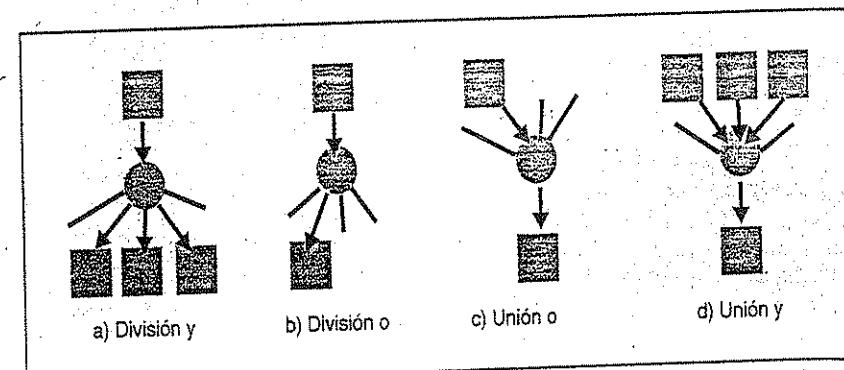


Figura 20-6. Flujo de trabajo: labores de división y unión.



Flujo de trabajo: Conozca a los participantes

Hemos construido literalmente un nuevo género de actividad en torno a la confusión e interés generados por el flujo de trabajo. Muchos usuarios no tienen idea de lo que es la tecnología.

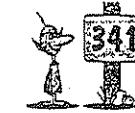
**Tom Koulopoulos, presidente
Delphi Consulting
(Septiembre de 1995)**

El flujo de trabajo es una tecnología en boga. Delphi Consulting estima que el gasto en flujo de trabajo alcanzó la marca de los \$1,000 millones de dólares en 1995, por encima de los \$722 millones en 1994. BIS Strategic Decisions calcula a su vez que el número total de sedes con flujo de trabajo podría alcanzar los 5.8 millones en 1998 —lo que representa un mercado de \$3,000 millones de dólares— sobre 514,000 en 1994.

Más de 100 proveedores compiten en la actualidad en el mercado de flujo de trabajo y son frecuentes objetivos de adquisición. Por ejemplo, BancTec adquirió recientemente a Recognition International, Wang adquirió a Sigma y FileNet compró Watermark. El mercado de flujo de trabajo todavía está en movimiento; no hay un líder evidente. De acuerdo con Delphi, en 1994 más del 40% del mercado de flujo de trabajo correspondió a estos cinco productos: *Workflow* de FileNet (17%), *ViewStar* de ViewStar (7%), *FlowMark* de IBM (6%), *Floware* de Recognition International (6%) y *Open Workflow* de Wang (5%). Nótese que la mayoría de los "cinco líderes" son proveedores de proceso de imágenes.

Algunos de los paquetes de flujo de trabajo más novedosos provienen de pequeños proveedores, cada uno de los cuales posee una participación de mercado inferior al 5%. Muchos de ellos ofrecen productos de flujo de trabajo en torno a Lotus Notes, como el propio Lotus, *At Work* de Quality Decision Management, *ActionWorkflow Builder for Notes* de Action Technologies y *WorkMAN* de Reach Software. Entre los productos más prometedores están *InConcert* de XSoft, *TeamRoute* de Digital y *Formflow* de Delrina. Finalmente, Microsoft concibe el flujo de trabajo como extensión del sistema de correo electrónico; Action Technologies se asoció con Microsoft para dotar de características de flujo de trabajo a *Windows 95* y a una versión futura de *Exchange*.

Los proveedores de software ofrecen actualmente flujo de trabajo como parte de sus aplicaciones empaquetadas. Por ejemplo, SAP construyó desde cero un mecanismo de flujo de trabajo para su *R/3 Release 3.0*, mientras que Oracle desarrolló su propio software de flujo de trabajo para su nueva serie de software de aplicación *SmartClient 10*. En contraste, PeopleSoft ha obtenido de terceros —como Action Technologies, Delrina e IBM/Lotus— las licencias para su mecanismo de flujo de trabajo. También los proveedores de sistemas habilitan ya el flujo de trabajo en sus aplicaciones mediante licencias de terceros. Por ejemplo, el mecanismo de flujo de trabajo de Action está siendo implantado en productos de Sybase, Saros, Verity, Platinum y LaserData. Finalmente, Novell posee la licencia del mecanismo de flujo de trabajo *Ensamble* de FileNet para incluirlo en su sistema de manejo de mensajes *Groupwise*.



La Workflow Coalition

La *Workflow Management Coalition* (WfMC) fue fundada en agosto de 1993 en calidad de organismo internacional no lucrativo para el desarrollo y promoción de estándares de flujo de trabajo. Su membresía está abierta a todas las partes interesadas o implicadas en la creación, análisis o desarrollo de sistemas de administración de flujo de trabajo. En 1996, WfMC tenía más de 150 miembros, entre ellos la totalidad de los principales proveedores de flujo de trabajo y los mayores integradores de sistemas.

En enero de 1995, WfMC publicó un glosario de términos relativos al flujo de trabajo, primer paso hacia la creación de un vocabulario común para la industria. La meta final es crear un lenguaje para la especificación de *procesos* de flujo de trabajo que pueda ser interpretado y concretado por una amplia variedad de *mecanismos* de flujo de trabajo. Según WfMC, un proceso de flujo de trabajo consiste en un conjunto de *actividades*. Una actividad es un paso lógico que contribuye a la realización de un proceso de flujo de trabajo; la ejecuta una *herramienta*, que es una aplicación fuera del sistema de flujo de trabajo. En la Figura 20-9 se muestra la reunión de las piezas del rompecabezas del flujo de trabajo en el marco conceptual de WfMC.

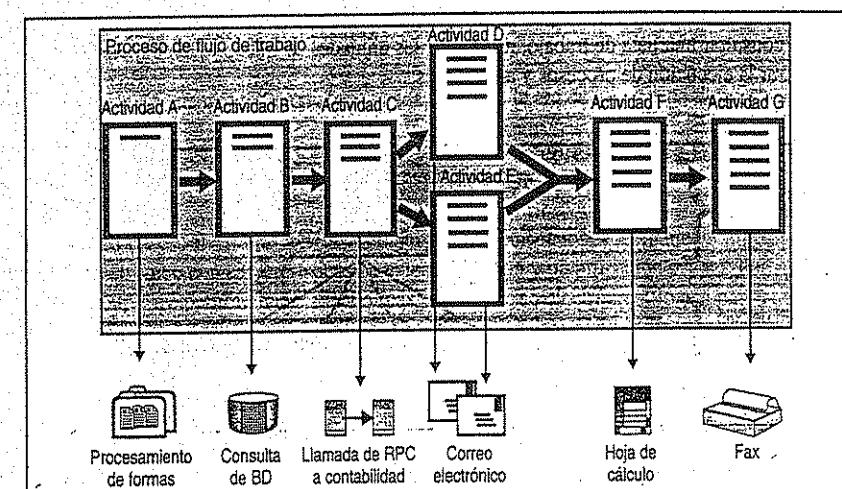


Figura 20-9. El proceso del flujo de trabajo como una serie de actividades desempeñadas por herramientas.

El objetivo a largo plazo de WfMC es definir un conjunto de API que pueda ser usado por aplicaciones y herramientas del cliente para invocar y definir funciones de flujo de trabajo y controlar un mecanismo de flujo de trabajo. WfMC trabaja también en un protocolo de interoperabilidad que permite la comunicación entre sí de mecanismos de flujo de trabajo de proveedores múltiples. En diciembre de 1995, WfMC dio a conocer su primer conjunto de API, llamado *API del cliente de flujo de trabajo* (*Workflow Client API*). Esta API es utilizada por aplicaciones cliente de flujo de trabajo para invocar los servicios de mecanismos de flujo de trabajo de proveedores múltiples.

Infraestructura del correo electrónico

La infraestructura requerida para crear redes principales de correo ubicuas ya se encuentra disponible. Existe una distinción decisiva entre la aplicación de correo —el primer piano— y la infraestructura del correo —el segundo plano. En términos ideales, el primer y el segundo planos deberían comunicarse por medio de líneas de cliente/servidor, que es lo que Lotus Notes 4.0 hace actualmente (véase Figura 20-11). Los antiguos paquetes de correo electrónico basados en LAN unen a los planos primero y segundo en el mismo proceso y emplean un servidor de archivos de la LAN para el almacenamiento de correo. Éste es el método de Lotus cc:Mail, el producto de correo electrónico más popular de la industria. Casi todos los productos de correo electrónico para LAN de PC siguen el modelo de servidor de archivos de cc:Mail, lo que sin

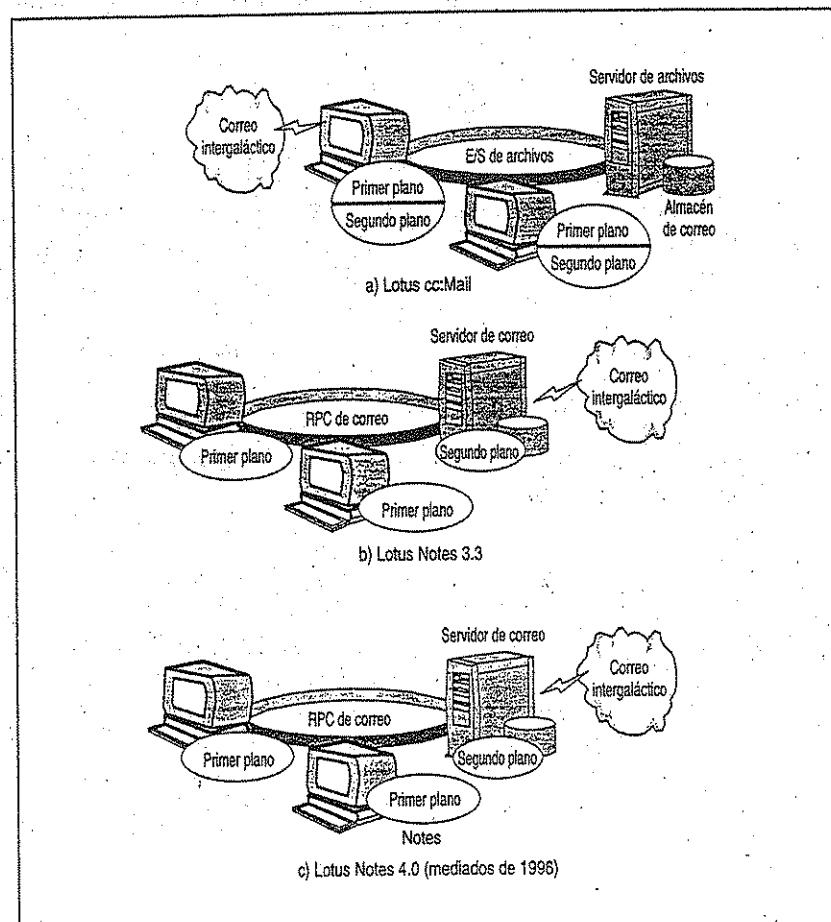


Figura 20-11. Correo electrónico de cliente/servidor: Lotus Notes contra cc:Mail.

embargo ya comienza a cambiar. Por ejemplo, Lotus rediseñó cc:Mail en líneas de cliente/servidor. Éste y Notes comparten hoy el mismo servidor posterior de correo. Microsoft y Novell ya desplazan también sus conocidos productos de correo electrónico —Microsoft Mail y Novell Groupwise— al modelo de cliente/servidor.

Sin embargo, hace falta mucho más que cc:Mail o Notes para crear una infraestructura de correo intergaláctico. ¿Cómo conectar cc:Mail o Notes a otras redes de correo? Enlazándose a una *red principal de correo* de servidor a servidor. Hay dos formas de hacerlo:

- **Gateways:** se necesita un gateway para cada sistema diferente de correo electrónico al que se desee acceder. Pero esto podría convertirse fácilmente en una pesadilla administrativa. Los gateways también limitan algunas funciones, como la capacidad de búsqueda de una dirección. Asimismo, son inefficientes en cuanto a la provisión de servicios sincronizados de administración de directorio, el eficaz enrutamiento de mensajes, la administración del sistema global, etcétera.
- **Red principal de correo:** se necesita un gateway para la red principal, y punto. Pero la pregunta es: ¿cuál red? Los contendientes son el estándar internacional X.400, el servicio de manejo de mensajes (MHS: Message Handling Service) de Novell y el servicio de correo del protocolo simple de transporte de mensajes (SMTP: simple message transport protocol) de Internet. El péndulo parece inclinarse en favor de X.400, ahora mucho más simple y barato que antes (véase el siguiente recuadro de detalles). El SMTP de Internet también es muy popular. Desde luego que es muy probable que las redes principales de MHS, SMTP y X.400 se interconecten pronto a través de gateways. Así que todas ellas podrían salir ganando.

La separación de funciones de correo en líneas de cliente/servidor facilitará la creación de clientes frontales totalmente independientes de los mecanismos de correo posteriores. Ocupémonos de los estándares de la API de correo que lo harán posible.



Redes principales de correo X.400

Detalles

El protocolo de correo X.400 ha conseguido por fin convertirse en la red principal de correspondencia de la industria. Ha sido adoptado ya por la totalidad de los principales proveedores de servicios públicos del mundo. La mayoría de los proveedores de correo electrónico —como Lotus, Microsoft, HP, IBM y SoftSwitch— ofrecen ya productos acordes con X.400. Varios de los más grandes —como IBM, Tandem y Digital— se sirven de X.400 para enlazar sus sistemas de manejo de mensajes. Para cuando usted lea este libro, deberá disponerse ya de versiones X.400 de Lotus Notes y cc:Mail, lo que le dará a X.400 una apariencia amigable en todas las plataformas. X.400 posee las siguientes características:

- **Soporte de intercambios de BLOB.** X.400 define un medio para el intercambio de imágenes, fax y otros anexos binarios de mensajes.



de distribución. Las libretas de direcciones pueden ser personales (alojadas en la memoria caché del sistema local) o formar parte de un directorio global. La API debería permitirle leer y escribir información de directorios, lo mismo que navegar a través de jerarquías de libretas de direcciones. Usted debería estar en condiciones de añadir o eliminar grupos o miembros de libretas de direcciones, y de recorrerlas.

- **Manipulación de objetos de correo.** Las API deben darle acceso a los subcomponentes de un objeto de correo. Por ejemplo, un mensaje puede consistir en campos de encabezamiento y varias unidades de datos.
- **Servicios de autenticación y seguridad.** Esto incluye API que permiten el acceso de una aplicación al sistema de correo y la autenticación de sus usuarios.

- **Interfaz del proveedor de servicios.** Esta interfaz permite que proveedores de servicios presten sus propios servicios posteriores a las API de correo frontales. Por ejemplo, un servidor de correo de Lotus Notes es accedido a través de la RPC de Lotus propietaria. En la sección de ODBC de la Cuarta parte abordamos ya el concepto de la interfaz del proveedor de servicios. Ésta crea un entorno abierto para los proveedores de servicios de correo; no les ofrece gran cosa a los desarrolladores de aplicaciones con integración de correo.

La separación entre el primer plano y el segundo plano de correo permite que una sola API trabaje con múltiples segundos planos de correo. Además, diferentes proveedores pueden ofrecer sus propios servicios especializados de conexión (como almacenes de mensajes, por ejemplo). Pero, ¿cuál es el conjunto de API común que nos da acceso a toda esta potencia de proveedores de correo?

¡Sorpresa! La industria de correo electrónico cuenta con más de un conjunto común de API abierta. Recuerde que *siempre* se debe elegir entre más de un estándar. En el caso del correo electrónico, al principio hubo cinco estándares, que en la actualidad se han reducido a tres: VIM, MAPI y CMC. Además, tanto Novell como Lotus soportan ya MAPI y CMC, además de VIM. Es posible, entonces, que al final sólo haya un estándar común: MAPI.

- La *mensajería independiente de proveedor* (VIM: *vendor independent messaging*) es una interfaz respaldada conjuntamente por Lotus, Apple, IBM, Borland, MCI, Oracle, WordPerfect y Novell. Fue diseñada desde un principio como interfaz entre plataformas. Se compone de 55 llamadas de API —10 de ellas opcionales—, que soportan correo simple, almacenamiento de mensajes y servicios de libreta de direcciones. Ofrece asimismo una *interfaz de correo simple* (SMI: *simple mail interface*), compuesta por dos llamadas: SMISendMail y SMISendDocuments. Su mayor cualidad es su soporte entre plataformas, y los proveedores tras él. Su mayor defecto es que no brinda una interfaz del proveedor de servicios (SPI: *service provider interface*).

- La *API para manejo de mensajes* (MAPI: *messaging API*) es la arquitectura de sistemas abiertos de Windows (WOSA: *Windows open systems architecture*) de Microsoft habilitada para correo electrónico. Fue en sus inicios la primera API cliente de Windows, DLL de Windows que opera fundamentalmente en segundo plano de Microsoft Mail. Ahora la soportan prácticamente todos los proveedores. Las API frontales de MAPI fueron generadas para el *Mail Spooler* de Windows. Los proveedores de servidores de correo pueden



redireccionar las llamadas al Mail Spooler a sus servicios posteriores con la SPI de MAPI. La MAPI simple se compone de 12 llamadas de API, las cuales prestan servicios de correo simple, almacen de mensajes y libreta de direcciones. La *MAPI extendida*, tecnología parcialmente implantada en Windows 95, soporta 100 llamadas de API, que permiten que las aplicaciones manejen grandes cantidades de mensajes, realicen filtración de correo, administren almacenes de mensajes y tengan acceso a información de direccionamiento compleja. La MAPI extendida expone la SPI a la aplicación. Existen tres tipos de SPI: de transporte, libreta de direcciones y almacen de mensajes. La cualidad de MAPI es su virtual universalidad.

- Las *llamadas comunes de correo* (CMC: *common mail call*) son la interfaz de la X.400 API Association (XAPIA), la cual fue publicada en junio de 1993 como parte de una tregua negociada en las guerras de la API de correo; los campamentos tanto de VIM como de MAPI la apoyaron. Microsoft y Lotus ofrecen una biblioteca DLL gratuita para CMC. Ésta se compone de 10 llamadas de API, subconjunto de las llamadas de VIM y MAPI (véase Figura 20-13). No incluye funcionalidad de VIM extendida ni MAPI extendida. Sólo realiza correo simple. Es el “pariente pobre” de las API de correo electrónico.

Las guerras de API de correo electrónico han llegado a su fin: todo parece indicar que la ganadora fue MAPI. Microsoft dirige ahora su atención a la fusión de correo electrónico y flujo de trabajo. En octubre de 1995 anunció la *estructura de flujo de trabajo* de MAPI, especificación preliminar para permitir la interoperación de mecanismos de flujo de trabajo y sistemas de manejo de mensajes. Esta estructura hará posible que los usuarios de correo electrónico echen a andar un proceso de flujo de trabajo, conozcan su estado y rastreen las tareas de flujo de trabajo en un recuadro. Microsoft ha sometido ya esta estructura a la consideración de la Workflow Management Coalition (se trata de un trabajo en proceso).

La buena noticia es entonces que ya disponemos de una API de correo universal que bien podría ser habilitada para flujo de trabajo. Ello facilitará la mezcla y empate de clientes de correo de

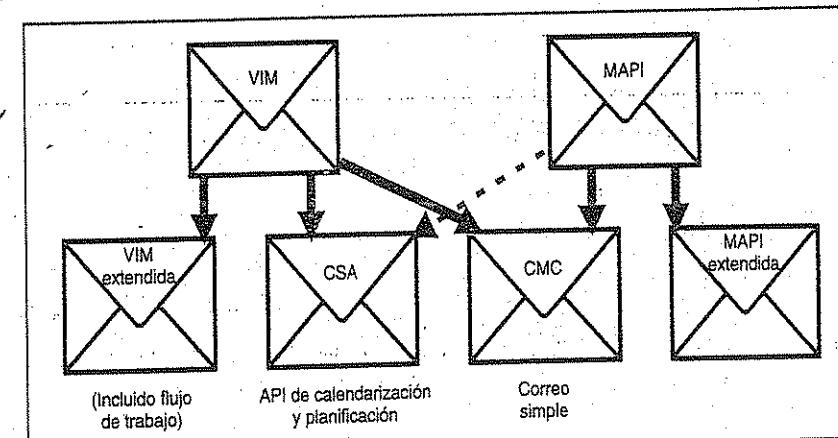


Figura 20-13. CMC: la gran unificación de MAPI simple y VIM simple.



Groupware: ¿sólo sensacionalismo publicitario?

Debate

En el futuro, las aplicaciones sin integración de groupware correrán el riesgo de ser una isla en medio del océano de las comunicaciones globales. El groupware unirá a individuos, agentes, procesos, empresas y corporaciones.

Karl Wong, Dataquest
(Diciembre de 1995)

Como toda nueva tecnología de cliente/servidor, el groupware está recibiendo su dosis de estremecedora escándalo comercial. Para colmo, cualquier pieza de software de usuarios múltiples puede ser llamada "groupware". Todo el software cliente/servidor tiene que ver en una forma u otra con comunicaciones grupales. Aunque tuvimos el cuidado de definir las tecnologías constitutivas del groupware, el término sigue siendo muy confuso. Dos personas —incluso de Lotus— le darán definiciones distintas de Lotus Notes, a pesar de lo cual éste se vende como pan caliente. Es inaudible que satisface necesidades en algún lado.

El groupware es un blanco móvil. Las aplicaciones de groupware no cesan de extenderse a nuevos territorios como resultado de cambios en la tecnología. Por ejemplo, el groupware avanza en la actualidad sobre dos nuevas áreas: Internet y telefonía. Hablaremos de Internet en la Octava parte. Las API de telefonía —como TAPI de Microsoft y la TSAPI rival de Novell/AT&T— permiten habilitar telefónicamente aplicaciones de groupware. Esto significa que sus aplicaciones de groupware podrán contestar, ubicar, mostrar en pantalla y enrutar llamadas telefónicas. Las API de telefonía también permiten enrutar mensajes, manejar correo de voz e integrar voz, fax y correo electrónico en un sistema coherente. En cierto sentido, ahora una aplicación de groupware puede añadir a su repertorio de funciones casi todo lo que en la actualidad puede hacer un sistema moderno de intercambio de ramificación privado (PBX: *private branch exchange*). De acuerdo con Dataquest, los sistemas telefónicos basados en PC experimentarán un crecimiento del 2,000%, desde los 154 millones de dólares en 1994 a los 2,800 millones en 1998. De manera que bienvenido a otra área de gran crecimiento.

Cualquier tecnología nueva se inicia entre confusiones. La clave está en distinguir la verdad en el sensacionalismo comercial, y comprender después qué podemos hacer con la tecnología. En el caso del groupware, la oportunidad está en la creación de aplicaciones de cliente/servidor —diferentes a las que conocemos— con tecnología de bases de datos de documentos con multimedia, correo electrónico, flujo de trabajo, conferencias, calendarización y planificación. La industria de groupware está creando las interfaces comunes entre estas piezas dispares. Así, nos basta aprender a usarlas, y quizás hasta integrarlas con bodegas de base de datos, Internet y tecnología de objetos distribuidos. □

Capítulo 21

Groupware: Presentación de participantes



Casi todas las grandes organizaciones harán sus evaluaciones este año y presentarán un sistema el próximo. Esta guerra va a ser sumamente rápida.

Tom Austin, Gartner Group
(Marzo de 1996)

El groupware es una industria en rápido crecimiento y en estado de flujo constante. El groupware fue por mucho tiempo sinónimo de Lotus Notes. A principios de 1996 aún no había un producto de groupware tan completo como Notes. Sin embargo, es de suponer que Notes enfrentará ferrea competencia de tres nuevos productos: *GroupWise XTD* de Novell, *Collabra Share* de Netscape y *Exchange* de Microsoft. Todos ellos están respaldados por compañías con una mentalidad y una presencia de mercado perfectamente asentadas. Además, seguramente los tres intentarán igualar a Notes como completos productos de groupware "todo en uno".

Este capítulo comienza con una breve descripción del mercado de *groupware* y las principales tendencias dentro de la industria. Después se analizará Notes 4.0 con cierto detalle, pues se trata del producto que da la pauta a todos los proveedores de *groupware*, y por lo tanto al que desean combatir todos los competidores. Hablaremos luego rápidamente de los competidores



En realidad, Notes es un producto multifacético de groupware cliente/servidor. Y tal como se comprobó en el capítulo anterior, es difícil definir al groupware en menos de 25 palabras. El secreto de un buen paquete de groupware es que sea capaz de crear un todo que es mucho más que la suma de sus partes. Notes lo hace muy bien.

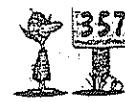
¿Qué es Lotus Notes?

La versión 1 de Notes fue de conferencias, la versión 2 de información compartida, la versión 3 de flujo de trabajo y la versión 4 de aplicaciones interempresariales.

David Marshak, vicepresidente
Seybold
(Mayo de 1995)

Lotus Notes permite que grupos de usuarios interactúen y comparten información que bien puede ser de naturaleza altamente desestructurada. Su desarrollo de aplicaciones cliente/servidor y entorno en tiempo de ejecución ofrecen las siguientes funciones (véase Figura 21-1):

- Un *servidor de base de datos de documentos* almacena y administra el acceso de clientes multiusuarios a datos semiestructurados, incluidos texto, imágenes, audio y video. La versión 4 soporta hasta 1,000 clientes de Notes concurrentes sobre plataformas de servidor de SMP de 32 bits (orden de magnitud mayor a la versión 3).
- Un *servidor de correo electrónico* administra acceso de clientes multiusuarios a correo. Notes incluye una infraestructura de red principal de correo; se dispone de X.400 como componente opcional. La versión 4 integra plenamente opciones de red de X.400 y SMTP/MIME de Internet.
- Una *infraestructura de red principal de servidor/servidor* soporta tanto enrutamiento de correo como duplicación de base de datos. El mecanismo de duplicación sincroniza copias de la misma base de datos, la cual puede residir en máquinas servidores múltiples (o clientes). La versión 4 soporta ya una nueva característica de *paso de servidor*, que permite conectarse con un servidor de Notes y tener acceso desde ahí a cualquier otro servidor para el cual se tenga autorización. Esta característica también permite tener acceso a múltiples bases de datos en servidores múltiples al mismo tiempo.
- Un *entorno cliente de GUI* presenta vistas de las bases de datos de documentos y ofrece un primer plano para correo electrónico. Los usuarios pueden navegar por las bases de datos y el contenido de sus documentos. Las vistas son consultas almacenadas que actúan como filtros de información en las bases de datos. El primer plano de correo electrónico es sencillamente una vista especializada de una base de datos de correo. Notes puede anexar formas de GUI (privadas o públicas) a las diversas bases de datos usadas para entrada de datos. La versión 4 introdujo el nuevo cliente *Notes Mail*; éste soporta una flexible interfaz de usuario trilateral que integra el cc:Mail, la visualización del Web y al cliente tradicional de Notes.



nal de Notes. Ofrece asimismo un buzón de entrada (*in-box*) universal capaz de aceptar todo tipo de correo, faxes y formatos.

■ Los *servicios distribuidos* incluyen firmas electrónicas, seguridad y listas de control de acceso, servicios de administración de base de datos, administración de sistemas y un espacio global para nombres basado en X.500.

■ Las *herramientas de desarrollo de aplicaciones* incluyen: 1) un generador de formas de GUI; 2) herramientas y plantillas para la creación de bases de datos; 3) un lenguaje primitivo de creación de scripts consistente en fórmulas, y 4) un conjunto de API abierta, que contiene la API de Notes, VIM, MAPI y ODBC. La versión 4 soporta *agentes inteligentes*. Éstos son aplicaciones de Notes con scripts útiles para la automatización de tareas repetitivas, como duplicación de bases de datos, manipulación de datos, las acciones de buzón de entrada y los servicios de manejo de mensajes. La versión 4 introduce igualmente *LotusScript 3.0*, lenguaje de programación orientado a objetos semejante a BASIC y capaz de ser aplicado entre plataformas. Además, usted puede generar aplicaciones de Notes mediante el uso de herramientas muy conocidas de cliente/servidor de terceros, como *Delphi*, *SQLWindows*, *PowerBuilder*, *New Era*, *VisualAge*, *Notes ViP* y *Visual Basic* (vía los VBX *HiTest* proporcionados por Lotus).

Todas las comunicaciones de Notes —cliente/servidor y servidor/servidor— se realizan con una RPC propietaria. Notes soporta controladores de cliente/servidor para pilas NetBEUI, TCP/IP, IPX/SPX y AppleTalk. Para las comunicaciones servidor a servidor se dispone de controladores de APPC y X.25 opcionales.

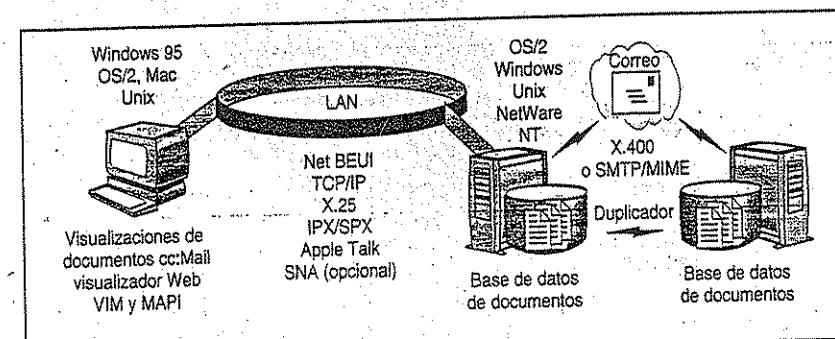


Figura 21-1. Componentes de Lotus Notes.

La base de datos de documentos con multimedia

Ray Ozzie —fundador de Iris Associates, compañía que desarrolló el Notes original bajo contrato con Lotus— describe los cimientos de la arquitectura de Notes como “un mecanismo de base de datos para información semiestructurada y no estructurada”. Este modelo se asemeja

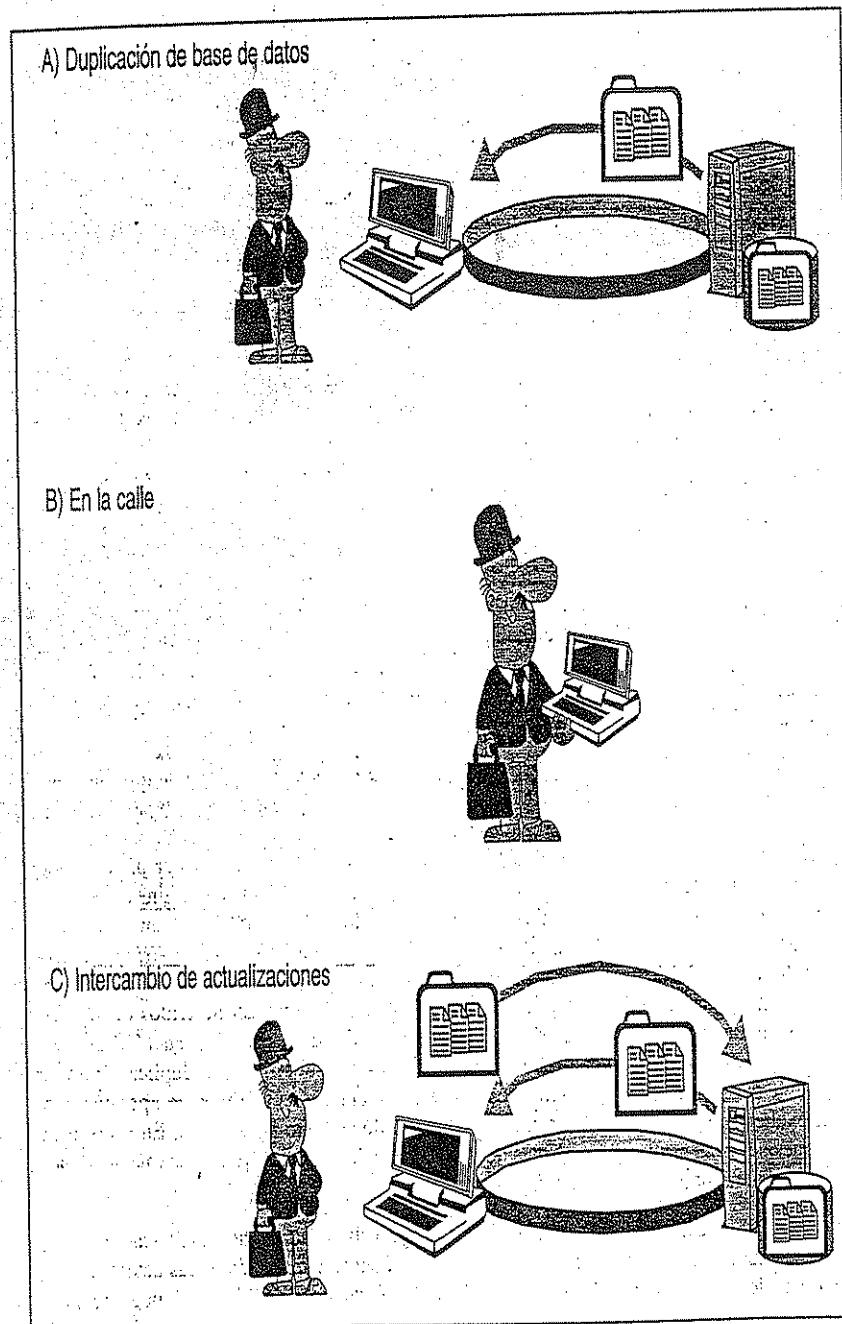


Figura 21-3. Bases de datos duplicadas pueden utilizarse en la calle.

primer guardado, mientras que en el caso del segundo se notificaba una sobrescritura en cambios ajenos. La decisión de sobreescibir o no datos se dejaba al usuario (lo que por supuesto no podía satisfacer al primer usuario). Pero en la versión 3 (y posteriores) de Notes se introdujo una capacidad de uso de versiones que permite que un documento editado se convierta en respuesta del documento original. O bien, la versión más recientemente actualizada puede volverse el documento principal, en tanto que todas las versiones anteriores siguen siendo respuestas.

El uso de versiones no garantiza que la versión más reciente sea la más exacta, y es incapaz de unir cambios en una sola copia de los datos. Así, incluso la versión 4 de Notes no es una tecnología conveniente para aplicaciones de bases de datos de OLTP o aplicaciones que requieren controles de alta concurrencia que supongan actualizaciones inmediatas. Sin embargo, el uso de versiones elimina la pérdida de datos en actualizaciones concurrentes o duplicación. Resulta muy útil por ello para su uso específico: aplicaciones de groupware centradas en documentos, área que OLTP no se atreve siquiera a tocar.

Cómo crear una aplicación de Notes

Por lo general se crea una nueva *base de datos* de Notes con una de las plantillas provistas por Lotus, la cual es sometida a personalización. Una base de datos es simplemente un nuevo archivo; puede asignársele un ícono identificador, título, páneles de ayuda y un *documento de políticas* en el que se explique de qué trata el asunto. Se usan *formatos* para introducir o visualizar información en una base de datos. Para crear un formato, se puede partir de uno de los formatos preexistentes para modificarlo con el editor de GUI. Los formatos ofrecen campos de entrada de datos, campos de texto y áreas gráficas para la inserción (o incorporación) de imágenes u otras fuentes de datos multimedia. Lotus proporciona un lenguaje macro —similar a las fórmulas de hoja de cálculo de 1-2-3— para asociar comandos con eventos y acciones específicos. Además, se puede usar *LotusScript* para generar manipuladores de eventos. En el futuro usted podrá generar sus manipuladores de eventos de Notes con el lenguaje Java. Los formatos que crea se asocian con la base de datos. Puede designarlos como *públicos*, lo que significa que están a disposición de todas las aplicaciones cliente con acceso a la base de datos, o como *privados*, lo que significa que sólo su creador podrá usarlos.

Las *vistas* son consultas almacenadas que muestran el contenido de una base de datos o de un documento en particular. Sirven para la navegación y filtración de información; como ejemplo puede citarse el caso de muestra de documentos de hasta un mes “por región” o “por vendedor”. La vista exhibirá la lista de documentos en formato tabular o resaltado. Todas las bases de datos cuentan con una o más vistas creadas por el diseñador para facilitar el acceso a la información. Los usuarios también pueden crear vistas *privadas* para disponer de listados o criterios de acceso no previstos por el diseñador de la base de datos.

En la parte del cliente, el usuario cuenta con un *espacio de trabajo*, una visualización estilo libreta que organiza las bases de datos por temas (véase Figura 21-4). Una libreta consta de seis páginas de trabajo con código de color, cada una de las cuales tiene una cejilla de carpetas en donde se identifica la categoría en la que usted desea organizar sus bases de datos. Cada página de trabajo puede contener de cero a cientos de bases de datos. Cada base de datos se repre-

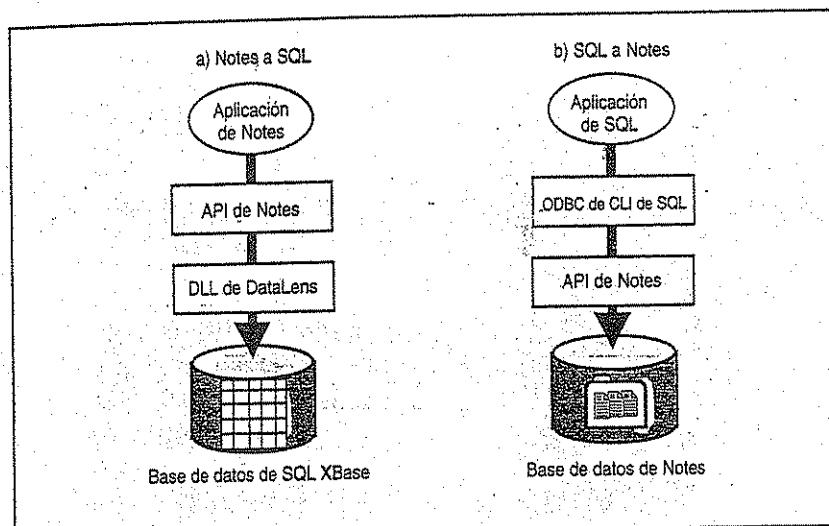


Figura 21-5. Intercambios bidireccionales de datos entre bases de datos de Notes y SQL.

Correo electrónico de Notes

Deseamos hacer de Notes el sitio ideal para vivir en la red.

Jim Manzi, exdirector general
Lotus

Desde la perspectiva de la estación de trabajo cliente, el correo electrónico es simplemente otra base de datos de Notes, la cual contiene un grupo de documentos de correo. Los procedimientos para la lectura del correo recibido, su clasificación o la creación de un documento de correo son los mismos que se emplean para crear y leer documentos de cualquier otra base de datos de Notes. Sencillamente se usan formatos y vistas a la medida de los documentos de correo propios. Por supuesto que una de las diferencias es que los documentos de correo que usted crea serán enviados al buzón de alguien. Notes ofrece indicadores visuales para hacerle saber que ha recibido correo. Se abre la base de datos y lo lee. El diseño de Ray Ozzie parece ser muy consistente.

El servidor de correo electrónico de Notes está abierto. Si el primer plano de correo de Notes no es de su gusto, puede usar las API de VIM o MAPI para crear el suyo. O simplemente puede utilizar VIM o MAPI para la integración de correo en sus aplicaciones, con el servidor de Notes como segundo plano. Con la versión 4, los clientes de cc:Mail pueden usar el servidor de correo de Notes. ¿Qué tipo de servicios presta un servidor de correo de Notes? Ejerce funciones de red principal de correo, con las siguientes características:

- **Optimización de enrutamiento.** Las técnicas incluyen prioridad en mensajes no acoplados y selección de ruta de adaptación dinámica basados en costos de vinculación.
- **Hilos enrutadores independientes.** Todas las comunicaciones de servidor a servidor son manejadas por hilos de transferencia independientes. Los hilos permiten la ocurrencia de múltiples transferencias simultáneas en diferentes rutas de la red. Además, impiden que mensajes grandes de correo retrasen otras tareas del servidor.
- **Notificación de falla de entrega.** Los remitentes pueden ser notificados cuando no es posible realizar una entrega (y los motivos de ello).
- **Soporte de espacio para nombres de X.500.** Notes soporta en su totalidad la asignación de nombres jerárquicos compatibles con X.500 como medio propio de identificación de usuarios dentro del sistema. Esto facilita a los directorios de Notes interoperar en el nivel de asignación de nombres con otros sistemas compatibles con X.500. Evita dolores de cabeza por conflictos de asignación de nombres.
- **Gateways de correo y servicios de directorio.** Notes proporciona gateways de correo electrónico a la redes de correo electrónico más populares, como X.400, SMTP, cc:Mail, MHS, PROFS, Exchange, VinesMail, Fax, VAXmail y SoftSwitch (véase Figura 21-6).
- **Firmas electrónicas.** Notes usa la criptografía de llave pública de RSA para todo lo relacionado con la seguridad de Notes, incluida la codificación. (En la Tercera parte explicaremos que RSA es apta para firmas electrónicas, pero muy lenta en lo que se refiere a codificación general.) Si usted firma un mensaje, a Notes le lleva unos segundos más enviarlo, porque debe generar una firma electrónica de RSA. Si la sola codificación de un nombre tarda un par de segundos, ¿cuánto tardará la codificación de un documento de tamaño mediano?

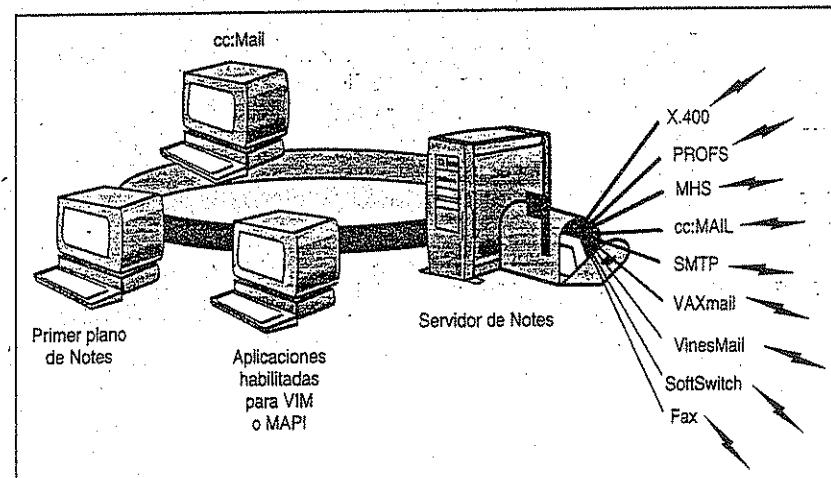
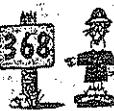


Figura 21-6. Alcance del correo electrónico de Notes a través de gateways.



¿Qué tan revolucionario es Notes?

Debate

Notes es un producto muy interesante que invita a ser manejado y a crear aplicaciones cliente/servidor sólo por el gusto de hacerlo. Sus bases de datos tienden a multiplicarse como conejos. Una vez en ellas, resultan contagiosas. A partir de la versión 4, Notes comenzó a ocuparse de las necesidades de cliente/servidor de usuarios móviles y empresas intergalácticas. Usted puede trasladar sus bases de datos de Notes en una laptop o hacerlas duplicar con destino a los cuatro puntos cardinales. De igual forma, Lotus se empeña en abrir aún más el entorno de Notes al nivel de API, para que ISV y programadores de IS puedan incorporarse a él y aportar adiciones. En más de un sentido, entonces, Lotus Notes puede ser una "aplicación arrasadora" que haga por cliente/servidor lo que Lotus 1-2-3 hizo por las PC y DOS.

Notes nos gusta, y lo recomendamos ampliamente para ciertas clases de aplicaciones. Pero es importante entender lo que puede y no puede hacer. Es ideal para aplicaciones que recogen información con multimedia, que realizan muy pocas actualizaciones y deben integrarse con correo electrónico. Pero no es tan eficaz en el manejo de aplicaciones relacionadas con datos estructurados, que son intensivas en consultas y requieren actualizaciones de usuarios múltiples con altos niveles de integridad. No manipula transacciones adecuadamente: no sabe siquiera cómo se escribe ACID. Para estos tipos de aplicaciones son preferibles los monitores de TP, bodegas de datos, MOM para transacciones y bases de datos de SQL y objetos. □

de correo y directorios (MADMAN: *mail and directories MANagement*). Daremos más información sobre SNMP y MIB en la Novena parte.

GROUPWISE XTD DE NOVELL

Novell tiene muchas piezas, algunas de ellas muy bien integradas.

**David Marshak, vicepresidente
Seybold**
(Mayo de 1995)

Alguien dijo una vez que *GroupWise* de Novell ha sufrido más cambios de nombre que un exconvicto. Es cierto. El producto, originalmente llamado *WordPerfect Office*, fue rebautizado en mayo de 1994 como *WordPerfect Symmetry*, para evitar confusión con Microsoft



Office. Pero en julio de ese mismo año Novell adquirió a *WordPerfect*, y puso por nombre *GroupWise* al producto. En septiembre de 1994 se lanzó *GroupWise 4.1* para plataformas múltiples, y en 1996 la versión del producto para cliente/servidor fue rebautizada como *GroupWise XTD*.

En enero de 1996 Novell vendió su división *WordPerfect* a Corel, pero conservó el exitoso producto *GroupWise*. A fines de 1995, este producto había vendido más de 5 millones de copias, convirtiéndose así en el paquete de correo electrónico para LAN más popular después de *Lotus cc:Mail* y *Microsoft Mail*. *GroupWise* creció 128% en 1995 y es en este momento el producto de Novell de más rápido crecimiento. En la más reciente estrategia de esta empresa, llamada "Red inteligente global", se le consideró pieza clave de infraestructura. El propósito de Novell es apoyar a la enorme base instalada de redes de NetWare y ofrecer una plataforma común para la administración de aplicaciones tanto de red como de groupware.

Entonces, ¿qué es *GroupWise*? Novell lo llama "una aplicación de manejo de mensajes". En su encarnación actual, es un producto de correo electrónico con funciones adicionales de groupware. Es un paquete integrado que incluye correo electrónico, calendarización, planificación, administración de tareas, mensajes de Internet y faxes. Soporta lo que Novell denomina "mensajería basada en reglas"; le permite asociar acciones para la entrada y salida de mensajes de correo electrónico. Todas las funciones de *GroupWise* operan con una interfaz de usuario común.

El software cliente de *GroupWise* se ejecuta en Windows, DOS, Macintosh, OS/2, HP-UX, AIX, SCO, DG-UX, Solaris, SunOS y Unix SVR4. El software del mecanismo de servidor groupware corre en numerosas variantes de Unix, NetWare y OS/2. Además, el cliente de Windows soporta MAPI simple, mientras que el cliente de Mac soporta AOCE y AppleScript. En la parte del servidor, Novell ofrece gateways a casi todos los sistemas de redes de mensajes de correo electrónico.

En diciembre de 1994, Novell firmó un acuerdo con Collabra para integrar un producto software de conversación colaborativa en *GroupWise*. En 1995 firmó un contrato similar con FileNet para brindar soporte de flujo de trabajo para *GroupWise*. El siguiente paso será *GroupWise XTD*, que ya deberá estar en circulación para cuando usted lea este libro. *GroupWise XTD* cuenta con duplicación de servidor a servidor y de servidor a cliente, flujo de trabajo y algunas capacidades de administración de formatos y documentos electrónicos. Realiza duplicación de mensajes en lugar de duplicación de base de datos.

GroupWise XTD incluye un *buzón de entrada universal*, que le permite el acceso a todo tipo de mensajes recibidos en una ubicación, incluidos correo electrónico, solicitudes de planificación, tareas delegadas, correo de voz, faxes y formatos electrónicos. Perfil y administra información de la misma manera, independientemente del tipo de datos de que se trate. En teoría debería ser posible clasificar, enrutar, copiar y delegar un mensaje de voz del mismo modo en que se haría con un mensaje de correo electrónico. Esta característica libera al usuario de la necesidad de reunir información de distintas aplicaciones o ubicaciones. Además, XTD brindará una capacidad de *carpeta duplicada* que permitirá que equipos compartan aplicaciones, información y unidades de buzón de entrada. Podrá compartir información relacionada con un tema o proyecto en particular. En enero de 1996 Novell dio a conocer *WebAccess*, un producto adicional que permite el acceso al buzón de entrada de *GroupWise* con cualquier visualizador Web.

El largamente pospuesto producto Exchange de Microsoft —lanzado finalmente a mediados de 1996— es un sistema de mensajes de cliente/servidor. Exchange ofrece a los clientes de Microsoft Mail correo electrónico basado en servidor, fax y capacidades de integración de correo de voz. El código del servidor sólo corre en Windows NT. Exchange también integrará planificación grupal, formatos electrónicos y aplicaciones de productividad empresarial bajo una interfaz de usuario unificada. Es parte integral de BackOffice de Microsoft. Por lo tanto, se apoya en la infraestructura común de NT para efectos de seguridad, autorización de usuarios y administración de redes. Por ejemplo, le permite modificar directamente el perfil de un usuario desde el interior de la *utilería de administrador del usuario* de NT.

Originalmente catalogado como “verdugo de Notes”, Exchange en su encarnación más reciente está más cerca de Collabra en cuanto a funciones que de Lotus Notes (véase Figura 21-9). Como Collabra, Exchange ofrece carpetas públicas para grupos de discusión encima de un sistema centrado en correo. En contraste, Notes brinda un entorno de desarrollo de aplicaciones de groupware sobre una base de datos de documentos compartida.

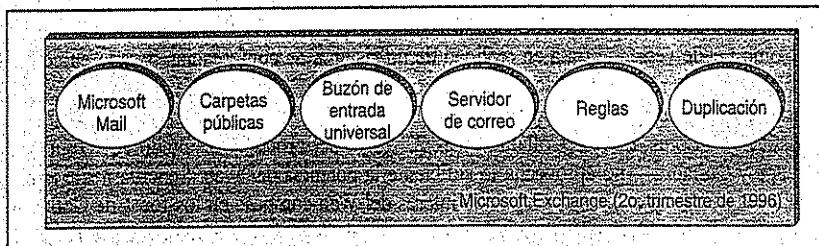


Figura 21-9. Exchange: la próxima generación de Mail Server de Microsoft.

Si usted es uno de los 7 millones de usuarios de Microsoft Mail, en Exchange se sentirá como en casa. Exchange puede concebirse como una versión para cliente/servidor de Microsoft Mail. Sustituye las carpetas compartidas de éste por una nueva facilidad de *carpetas públicas* que corre en servidores. Los clientes con acceso a ellas pueden destinarnos mensajes directamente, en lugar de tener que enviarlos a un usuario o grupo de usuarios en particular. Esto significa que usted puede usar las carpetas públicas para crear foros públicos en los que grupos de usuarios puedan crear y responder a envíos. Exchange lleva un registro de los hilos de mensajes en las carpetas públicas, que despliega en configuración de árbol en la ventana de un foro. Esto le permite ver cómo se relacionan los mensajes de una carpeta pública a otra.

Además, Exchange cuenta con un servicio de “publicar y suscribir” sobre sus carpetas públicas. Por ejemplo, le permite suscribirse a una de ellas y recibir notificaciones automáticamente cada vez que alguien destina un mensaje a esa carpeta. Un diseñador de formatos integrado le permite crear formatos personalizados para la emisión de mensajes. También puede generar formularios con Visual Basic.

Como GroupWise y cc:Mail, Exchange aporta un “buzón de entrada universal” que permite reunir correo electrónico, faxes y mensajes de las carpetas públicas. Podrá filtrar mensajes, clasificarlos y crear vistas múltiples de carpetas de mensajes. Por ejemplo, podrá clasificar los mensajes por tema, fecha, emisor o contenido especial de un campo personalizado. También deberá definir reglas (o acciones) a las que se invocará cuando se reciba un mensaje. Estas reglas —llamadas *AutoAssistants*— se ejecutan en el servidor.

De igual manera que Collabra y GroupWise, Exchange soporta duplicación de servidor a servidor por medio de correo electrónico. Por el contrario, Notes emplea una RPC especializada para acelerar el proceso de duplicación y mejorar su fuerza. A diferencia de Notes, Exchange no permitirá la duplicación de bases de datos con destino a clientes remotos, lo que puede representar un problema para usuarios móviles. En la actualidad Exchange es considerado en esencia como un medio para sincronizar comunicaciones de correo electrónico. No se le tiene por sustituto de Notes. Constituye más bien el sistema de correo electrónico de próxima generación de Microsoft.



¿Groupware sigue siendo sinónimo de Notes?

Debate

Las próximas víctimas del Web serán las aplicaciones para grupos de trabajo, como Lotus Notes y Microsoft Exchange. IBM y Microsoft se encuentran, por primera vez, cometiendo el mismo error en la misma posición.

Charles Ferguson, presidente, Vermeer (Noviembre de 1995)

En vez de eliminar la necesidad de aplicaciones como Notes, el Web no hará en muchos casos sino incrementarla, motivo de fondo para la adquisición de Collabra por Netscape.

Jamie Lewis, presidente, Burton Group (Octubre de 1995)

Tras la abierta alianza de Lotus con el Web, Internet ha dejado de representar una amenaza para Notes. Por el contrario, Internet e intranets pueden convertirse en importantes fuentes de nuevos ingresos para las proveedoras de groupware. Lotus Notes constituye la tecnología ideal para el soporte de bases de datos de documentos, misma materia de la que está hecha Internet. Los servidores de Notes crean un paradigma de base de datos sobre archivos ordinarios, de un orden de magnitud superior al tratamiento de miles de millones de archivos en bruto.

En 1996 no había aún un producto de groupware comparable a Notes en sus grandes cualidades de interplataformas cruzadas y facilidad de ampliación y escalabilidad. Sin

Introducción a la Séptima parte

CORBA y OLE serán quizá el terreno de juego del futuro próximo.

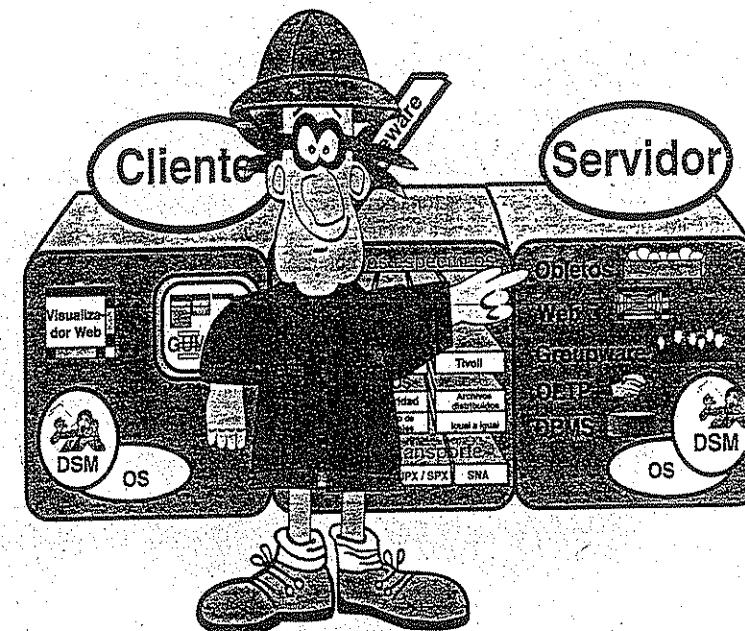
**Mark Betz, director de Objetos
Block Financial
(Marzo de 1996)**

¿Disfrutaron de la noche en la ciudad? ¿Quieren jugar otra vez a "Los ciegos y el elefante"? Esos sí que no se ya a poder; ustedes los marcianos son muy traviesos. Pero les tenemos preparada una fantástica aventura, que promete ser también muy divertida. Exploraremos el territorio desconocido de los objetos distribuidos. Y no nos referimos precisamente a los ovnis, esa marcianada de los "*objetos voladores no identificados*". ¡Oh, perdón! Por supuesto que ustedes, apreciables marcianos, no son ni objetos ni no identificados. No, nos referimos a objetos de cómputo. ¿Vuelan? Claro; algunos lo hacen en redes inalámbricas. Sabemos además de la existencia de objetos nómadas, que viven en las redes, y de "corredores", que son quienes organizan comunidades de objetos. Si, es otra nueva frontera, y debe estar repleta de oro. ¿Ya empacaron y están listos para otra aventura?

La Séptima parte trata de objetos distribuidos. El término *distribuidos* es importante, porque quiere decir que nos las veremos con objetos que participan en relaciones de cliente/servidor con otros objetos. ¿Más instalaciones? Sí, pero se supone que ésta es la "madre de las instalaciones". Además de hacer todo lo que hemos descrito hasta ahora en este libro, se dice que los objetos *lo hacen mejor*. He aquí por qué:

- Los *objetos* son por sí mismos una impresionante combinación de datos y funciones, con propiedades mágicas como polimorfismo, herencia y encapsulado. Esta magia obra maravillas en entornos distribuidos.
 - Los *corredores de objetos* son el sistema distribuido por autonomía. Hacen posible que los objetos se descubran dinámicamente entre sí e interactúen entre máquinas y sistemas operativos.
 - Los *servicios de objetos* nos permiten crear, administrar, nombrar, mover, copiar, almacenar y restaurar objetos.
 - Los *monitores de TP de objetos* pueden revelarse como los más potentes y flexibles administradores de transacciones. Objetos y transacciones son una combinación explosiva.
 - El *groupware de objetos* puede modificar por completo nuestro estilo de interacción. El nuevo groupware se hará para objetos nómadas, administradores de eventos inteligentes y servicios de duplicación de objetos.
 - Las *bases de datos de objetos* son lo último en sistemas de administración de BLOB, documentos y casi cualquier tipo de información, especialmente los nuevos tipos de información.

Introducción a la Séptima parte



- La tecnología de *Web de objetos* es nada menos que el fundamento para la próxima generación de Internet e intranets, como se explica en la Octava parte.
 - La tecnología de *vinculación de objetos* nos permite crear redes sumamente flexibles entre programas que no saben nada unos de otros. Las redes surgen de documentos de escritorio aparentemente ordinarios.
 - Las *estructuras de objetos* revolucionarán probablemente nuestra manera de componer sistemas distribuidos. Nos dotan de subsistemas de software prefabricado, flexible y personalizable.

Empaquetados como componentes, los objetos representan quizá la infraestructura por excelencia para la formación de sistemas de cliente/servidor. Decimos "quizá" porque su éxito dependerá de algo más que únicamente una fantástica tecnología. ¿Qué productos se ofrecen? ¿Los principales participantes apoyan esta tecnología? ¿Se cuenta ya con estándares básicos?

Estamos seguros de que a ustedes, amigos marcianos, les va a fascinar esto de los objetos distribuidos. Tiene todos los elementos de una nueva fiebre del oro. Pero antes tenemos que entender claramente cómo es que esta tecnología que tiene ya 20 años de existencia está prepa-



objetos en las máquinas y límite de redes para crear soluciones de cliente/servidor. No repetiremos aquí los portentos de la programación orientada a objetos (*object-oriented programming*), Smalltalk y C++, porque estamos seguros de que ya los conoce. Daremos por lo tanto un paso más: objetos distribuidos. Estamos profundamente convencidos de que ésta es el área en la que los objetos desarrollarán todo su potencial, y de que en ese proceso terminarán por convertirse en el nuevo "modelo básico de computación". También creemos que Internet e intranets necesitan de los objetos distribuidos para cumplir su promesa intergaláctica, de lo que hablaremos en la Octava parte. Finalmente, pensamos que sin fundamentos sólidos de objetos distribuidos, la administración de sistemas de cliente/servidor es una causa perdida, de lo que hablaremos en la Novena parte. En este capítulo abordaremos los objetos distribuidos y los componentes. Es la información básica que necesitará para comprender los capítulos sobre CORBA, OLE y OpenDoc.

QUÉ PROMETEN LOS OBJETOS DISTRIBUIDOS

Así como las bases de datos fueron el centro para el diseño de aplicaciones en los años setenta y ochenta, los componentes son el centro del diseño de aplicaciones para los noventa y el próximo siglo.

David Vaskevitch, director de Enterprise Microsoft (1995)¹

La tecnología de los objetos modifica radicalmente el estilo de desarrollo de sistemas de software. La propuesta es precisa: podremos conjuntar complejos sistemas de información de cliente/servidor simplemente reuniendo y extendiendo componentes de software reutilizables. Cualquier uno de los objetos puede ser modificado o reemplazado sin afectar al resto de los componentes en el sistema ni su modo de interactuar. Los componentes pueden lanzarse como grupos de bibliotecas de clase prearmados en *estructuras*, de cuyas piezas se sabe que, en su totalidad, pueden trabajar juntas para realizar una tarea específica. Los componentes revolucionarán nuestra manera de crear sistemas de cliente/servidor. Prometen ofrecernos lo último en capacidades de mezclar e igualar.

Beneficios de los objetos distribuidos

La tecnología de objetos distribuidos es extremadamente apta para la creación de sistemas flexibles de cliente/servidor porque los datos y lógica de negocios se encapsulan dentro de los objetos, permitiéndoles así ubicarse en cualquier punto dentro de un sistema distribuido. La granularidad de la distribución está muy mejorada. Los objetos distribuidos poseen el inherente potencial para permitir que componentes granulares de software sean conectados y usados (*plug-and-play*), interoperen entre redes, corran en diferentes plataformas, coexistan con aplicaciones de herencia a través de envolturas de objetos, merodeen en redes y se administren.

¹ Fuente: David Vaskevitch, Client/Server Strategies, 2a. edición (IDG, 1995).



solo, lo mismo que a los recursos que controlan. Los objetos son por naturaleza entidades autogestionarias. Deben permitirnos administrar sistemas muy complejos mediante la difusión de instrucciones y alarmas. Cada objeto receptor reaccionará de distinta manera al mensaje, con base en su tipo de objeto.

¿Cuál es la causa de este súbito interés por los objetos distribuidos?

La tecnología de objetos cuenta con la capacidad para revolucionar la computación de cliente/servidor porque facilita a programadores, usuarios y administradores de sistemas el desarrollo, uso y administración de software.

Ronald Weissman, director
NeXT

Todavía es necesario responder a esta pregunta: ¿a qué se debe el renovado interés en la tecnología de objetos, con 20 años de existencia? Es ya una tecnología madura. Con ninguna otra tecnología podríamos componer el tipo de aplicaciones que necesitamos. De igual manera, la industria ya ha puesto los cimientos para la infraestructura de los objetos distribuidos, integrada por un bus de software de objetos y por la tecnología para que los componentes puedan ser conectados y usados en este bus. Las aplicaciones monolíticas lo son porque se les forma como un todo. El bus de objetos y la infraestructura de componentes vuelven innecesario levantar sistemas de información desde cero. Nos permiten crear con partes aplicaciones completas.

Sin embargo, los objetos distribuidos (y los componentes) no bastan por sí solos para el cumplimiento de ese objetivo. Se les debe empaquetar en *series* de componentes capaces de operar juntos. En estas series se combina lo mejor de cliente/servidor y de la tecnología de objetos distribuidos. Nos permiten "poner en orden" sistemas de información enteros mediante la reunión de componentes de objetos comerciales. Podremos así armar —en tiempo récord— aplicaciones cliente/servidor sumamente flexibles, a la medida de las necesidades de un cliente. Los componentes podrán presentarse en series prearmadas, cuyas piezas colaboren en la ejecución de una tarea específica. Prevemos que los componentes y las series de cliente/servidor que los integran ofrecerán grandes y nuevas oportunidades a ISV, integradores de sistemas y desarrolladores de IS dentro de las empresas.

Para que los objetos tengan éxito, deben residir en entornos de cliente/servidor abiertos y aprender cómo "conectarlos y usarlos" entre redes y sistemas operativos. En teoría, la tecnología de objetos es ideal para la creación de sistemas de cliente/servidor, porque datos y lógica de negocios se encapsulan dentro de los objetos, lo que les permite ubicarse en cualquier punto dentro de un sistema distribuido. La *granularidad* de la distribución mejora enormemente. Los objetos pueden ocultar fácilmente los elementos específicos de plataformas y dar la apariencia de que las piezas interoperan sin costura.

Las aplicaciones cliente/servidor orientadas a objetos pueden permitirse ser mucho más flexibles que las aplicaciones verticales tradicionales; las estructuras dan lugar a que los usuarios finales mezclen e igualen componentes sin que por ello la aplicación distribuida sea *menos* robusta. Los objetos distribuidos permiten que componentes granulares de software se



La tecnología de componentes —en todas sus formas— promete transformar radicalmente el modo en el que los sistemas de software se desarrollan. Por ejemplo, los objetos distribuidos nos permiten agrupar complejos sistemas de información de cliente/servidor con sólo ensamblar y extender componentes. La meta de los componentes de objetos es brindar a usuarios y desarrolladores de software los mismos niveles de interoperabilidad de aplicaciones de conectar y usar que ya están a disposición de consumidores y fabricantes de partes electrónicas o circuitos integrados personalizados.

El impulso detrás de los componentes

La revolución de los componentes está siendo conducida desde el escritorio, donde los proveedores están cayendo en la cuenta de que, para ser rentables, deben reformular la arquitectura de sus aplicaciones y series existentes para adecuarlas a la realidad de los componentes. Las actuales aplicaciones para computadoras de escritorio son monolíticas y bofás. Contienen todas las características imaginables, independientemente de que se desee o no contar con ellas. Casi todos empleamos en realidad menos del 10% de las características de una aplicación; el resto no nos sirve para otra cosa que para abultar la complejidad y el volumen. Pero nos vemos obligados a esperar —aparentemente hasta la eternidad— para disponer de las nuevas características que realmente necesitamos, en forma de versiones mejoradas o remplazos. Esto se debe a que los proveedores tienen que presentar en cada nueva versión los clientes de características del producto, lo que desemboca en largos y costosos ciclos de desarrollo. Adoptan el método de “largo alcance” porque ignoran qué características necesitarán los usuarios; prueban hacerlo todo para todos.

Las mejoras y el mantenimiento de estas aplicaciones hinchadas, sobrecargadas de características y monolíticas, son muy costosos para los proveedores. Cada cambio pone en peligro la frágil integridad del monolito y requiere de largos ciclos de pruebas de regresión (y recursos). Mantener estas aplicaciones tampoco es precisamente un picnic. Los pequeños *proveedores independientes de software (ISV: independent software vendor)* enfrentan los mismos problemas, sólo que más agudamente. Cuentan con recursos mucho más limitados para resolverlos.

Para darle una idea de la magnitud de este problema, piense que a WordPerfect le llevó sólo 14 “años” de desarrollo hacerle mejoras a su producto de la versión 3 a la 4. Pero necesitó de 250 años para hacer pasar al mismo producto de la versión 5 a la 6. De seguir así las cosas, lograr la versión 8 podría costarle hasta 4 464 años de desarrollo. Microsoft experimenta el mismo fenómeno. Por ejemplo, las dimensiones de Excel pasaron de los 4 MBytes en 1990 a más de 16 MBytes en la actualidad. Cuenta con que esta situación cambiará con los componentes de OLE. La historia de Lotus es similar. Lanzó 1-2-3 en 1982 con un disco flexible; hoy se requiere de 11MBytes de espacio en disco para instalarlo. En su caso tiene la mirada puesta en OLE y OpenDoc para resolver el problema.

Los componentes al rescate

Los objetos se han librado de las garras de un solo lenguaje o programa. Los programadores se han librado de los límites de un compilador o familia de bibliotecas de clase. Los objetos pueden estar en cualquier parte, trabajar juntos y abrir todo un nuevo mundo de oportunidades a la próxima generación de arquitecturas de sistemas.

Martín Anderson, presidente
Integrated Objects
(Junio de 1995)

Desde hace tiempo se sabe que la programación orientada a objetos es una solución a los problemas anteriormente descritos. Sin embargo, los objetos no ofrecen por sí solos una infraestructura para que software creado por diferentes proveedores pueda interactuar en el mismo espacio de direccionamiento, y mucho menos entre espacios de direccionamiento, redes y sistemas operativos. La solución es complementar los objetos clásicos con una infraestructura de componentes estándar.

OpenDoc y OLE son actualmente los estándares de componentes líderes para el escritorio; CORBA representa un estándar de componentes para la empresa. CORBA y OpenDoc se complementan; OpenDoc usa a CORBA como su bus de objetos. Estos nuevos estándares de componentes cambiarán la economía del desarrollo de software. Las aplicaciones monolíticas, tanto en el escritorio como en la empresa, serán remplazadas por series de componentes. El impacto de esta nueva tecnología en la situación de usted será el siguiente:

- Los *usuarios de potencia* encontrarán en ella su segunda naturaleza para ensamblar sus aplicaciones personalizadas con componentes comerciales. Harán uso de scripts para unir las partes y personalizar su comportamiento.
- Pequeños desarrolladores e ISV descubrirán que los componentes reducen costos y derriban las barreras de acceso al mercado de software. Podrán crear componentes individuales con la certeza de que se integrarán armónicamente con el software existente creado por grandes compañías de desarrollo; no tendrán que reinventar todas las funciones. Lograrán una integración de grano fino, en lugar de la actual integración de “remedio casero”. Además, llegarán más rápidamente a la etapa de comercialización, gracias a que dispondrán de antemano del grueso de una aplicación.
- Los *grandes desarrolladores, fabricantes de IS e integradores de sistemas* usarán series de componentes para crear (o ensamblar) en tiempo récord aplicaciones cliente/servidor de alcance empresarial. Lo común será que alrededor del 80% de las funciones que necesitan estén a su disposición bajo la forma de componentes comerciales. El 20% restante será el valor agregado que aporten. La prueba de los sistemas de cliente/servidor que resultarán de

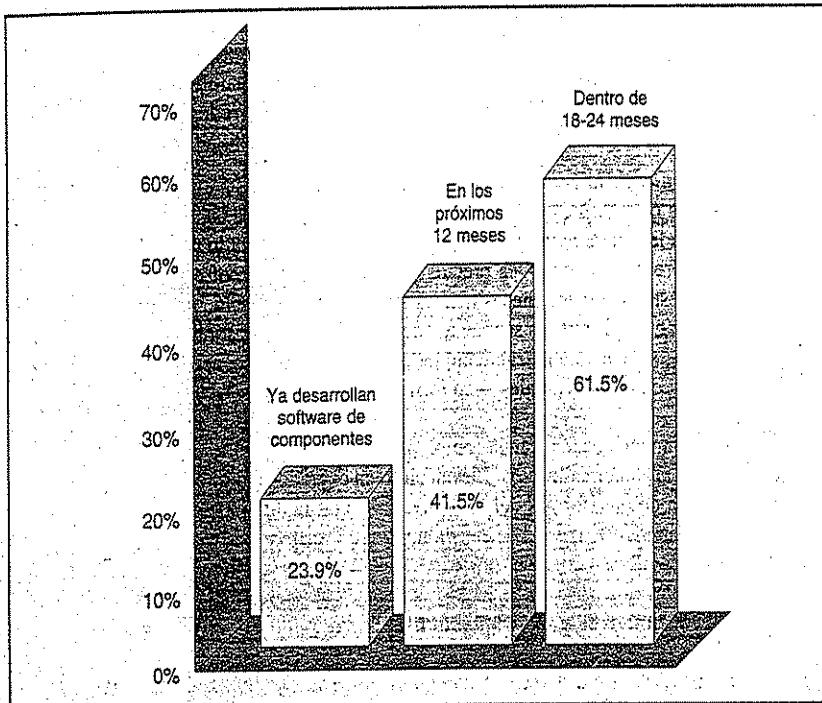


Figura 22-2. Periodos previstos por desarrolladores para iniciarse con componentes (Fuente: Strategic Focus, enero de 1995).

conectaremos. El bus de objetos constituye la base. A las familias de IC de software que operan en común se les llama *series*. En teoría, usted podrá comprar sus CI de software —o componentes— por medio de catálogos de partes estándar. De acuerdo con Gartner Group, los componentes harán surgir tres nuevos mercados: 1) el mercado de componentes como tal; 2) un mercado de herramientas para el ensamblaje de componentes y 3) un mercado de aplicaciones personalizadas desarrolladas con componentes.³

¿Por qué esto no sucedió antes?

Entonces, la pregunta del millón es: ¿por qué no ocurrió más rápido? ¿Por qué tuvimos que esperar 23 años para seguir los pasos de los ingenieros de hardware? Es cierto que durante casi 23 años la industria del software no dejó de hablar de reutilización, objetos y metodologías que nos habrían librado de las crisis del momento. Esta vez, la diferencia es que tenemos dos estándares entre los cuales elegir: OpenDoc/CORBA y OLE/DCOM. Sin estándares no habría componentes. Por lo tanto, si esto no ocurrió antes fue sencillamente porque la industria carecía de la adecuada infraestructura, o estándares, de componentes. Pero ahora tenemos ambos.

³ Fuente: Gartner Group, "Object Orientation for the Rest of Us" (marzo de 1995).

Entonces, ¿qué es un componente?

Un componente es una pieza de software suficientemente pequeña para crear y mantener, suficientemente grande para desplegar y soportar, y con interfaces estándar para interoperabilidad.

Jed Harris, expresidente
CI Labs
(Enero de 1995)

Los componentes interoperan mediante el uso de modelos de interacción de cliente/servidor neutrales en lo que se refiere a mensajes. A diferencia de los objetos tradicionales, pueden interoperar entre lenguajes, herramientas, sistemas operativos y redes. Sin embargo, también son semejantes a los objetos en el sentido de que soportan herencia, polimorfismo y encapsulado. Cabe indicar que algunos componentes —Ivar Jacobson los llama componentes de *caja negra*— no pueden extenderse por herencia. Los componentes de OLE corresponden a la categoría de caja negra. En cambio, los componentes tanto de CORBA como de OpenDoc soportan herencia. Como resultado de ello, permiten la integración de componentes ya sea de caja blanca o caja negra. Un componente de *caja blanca* es un componente que se comporta como objeto clásico.

Dado que los componentes significan cosas diferentes para personas distintas, definiremos las funciones que un componente *mínimo* debe proporcionar. En la siguiente sección ampliaremos nuestra definición para incluir en ella las características que deben ofrecer los supercomponentes. Nuestra definición de componente es un compuesto de lo que brindan CORBA, OpenDoc, Java y OLE. En su mayoría, las primeras definiciones de componentes eran en realidad listas de intenciones. Pero ahora que ya se dispone de estándares, podemos derivar de ellos una definición. Así pues, un componente mínimo posee las siguientes propiedades:

- **Es una entidad comercializable.** Un componente es una pieza binaria de software autónoma en paquete comercial que puede adquirirse en el mercado abierto.
- **No es una aplicación completa.** Un componente puede combinarse con otros componentes para formar una aplicación completa. Está diseñado para desempeñar un conjunto limitado de tareas dentro del dominio de una aplicación. Los componentes pueden ser objetos de grano fino, como los objetos de tamaño C++; objetos de grano mediano, como los controles de GUI, u objetos de grano grueso, como un applet de Java.
- **Se le puede emplear en combinaciones impredecibles.** Como los objetos, un componente puede usarse en formas absolutamente imprevistas por el desarrollador original. Usualmente, los componentes pueden combinarse con otros componentes de la misma familia llamada *serie*— mediante conectar y usar (*plug-and-play*).
- **Tiene una interfaz claramente definida.** Como los objetos clásicos, un componente sólo puede ser manipulado a través de su interfaz. Éste es el medio por el cual el componente expone su función al mundo exterior. Un componente de CORBA/OpenDoc también offre-

- **Creación de scripts:** un componente debe permitir que su interfaz sea controlada por medio de lenguajes de creación de scripts. Esto significa que la interfaz debe ser autodescriptiva y soportar acoplamiento posterior.
- **Metadatos e introspección:** un componente debe ofrecer, a solicitud expresa, información sobre sí mismo. Esto incluye una descripción de sus interfaces, atributos y las series que soporta.
- **Control de transacciones y candados:** un componente debe proteger sus recursos en transacciones y cooperar con otros componentes en el ofrecimiento de integridad total o nula. Adicionalmente, debe ofrecer candados para serializar el acceso a recursos compartidos.
- **Persistencia:** un componente debe ser capaz de guardar su estado en almacenamiento persistente y de restaurarlo más tarde.
- **Relaciones:** un componente debe estar en condiciones de asociarse dinámicamente o permanentemente con otros componentes. Por ejemplo, un componente puede contener otros componentes.
- **Facilidad de uso:** un componente debe ofrecer un número limitado de operaciones para alentar su uso y reutilización. En otras palabras, el nivel de abstracción debe ser lo más elevado posible para que el componente invite a su uso.
- **Autocomprobación:** un componente debe probarse por sí mismo. Usted debería estar en condiciones de correr diagnósticos provistos por el componente para la determinación de problemas.
- **Manejo de mensajes semánticos:** un componente debe ser capaz de comprender el vocabulario de series particulares y de las extensiones específicas de dominio que soporta.
- **Autoinstalación:** un componente debe estar en condiciones de instalarse solo y de registrar automáticamente su entrada en funciones en el registro del sistema operativo o de componentes. También debe eliminarse del disco cuando se le solicite.

Esta lista debería darle una clara idea del nivel de calidad y funcionalidad que esperamos de nuestros componentes. Lo interesante del asunto es que tanto OpenDoc/CORBA como OLE/DCOM ya ofrecen algunas de estas funciones. Las implementaciones típicas de OpenDoc/CORBA permiten la incorporación de este comportamiento en componentes ordinarios vía *mixins* en tiempo de instalación. Algunas implementaciones de CORBA —SOM, por ejemplo— permiten incluso la inserción de este comportamiento de sistema en componentes binarios en tiempo de ejecución. OLE permite la adición del comportamiento en tiempo de instalación mediante la combinación de componentes integrados por múltiples interfaces; un componente externo puede llamar entonces a las interfaces adecuadas a través de la técnica de reutilización conocida como *agregación*.

Objetos de negocios: Los componentes por excelencia

Los componentes contribuirán a que los usuarios se concentren en tareas, no en herramientas, así como una cocina bien surtida permite fijar la atención en la preparación y disfrute de succulentos platillos, y no en la marca de los ingredientes.

Dave LeFeuvre, Novell
(Enero de 1995)

Los objetos distribuidos son componentes por definición. La infraestructura de los objetos distribuidos es en realidad una infraestructura de componentes. Los programadores pueden lograr fácilmente una actitud de colaboración generando código para las dos partes involucradas. Sin embargo, lo difícil es conseguir que componentes sin conocimiento previo entre sí hagan lo mismo. Para llegar a este punto, usted necesita estándares que fijen las reglas de participación para diferentes fronteras de interacción entre componentes. Juntas, estas diferentes fronteras de interacción definen la *infraestructura* de un componente distribuido.

En el nivel básico, una infraestructura de componentes aporta un bus de objetos —el *corredor de solicitudes de objetos* (ORB: *object request broker*)—, el cual permite que los objetos interoperen entre espacios de direccionamiento, lenguajes, sistemas operativos y redes. El bus también ofrece mecanismos para que los componentes intercambien metadatos y se descubran entre sí. En el siguiente nivel, la infraestructura complementa el bus con *servicios al nivel del sistema* adicionales, los cuales hacen posible la creación de componentes superinteligentes. Son ejemplos de estos servicios las licencias, seguridad, control de versiones, persistencia, negociación de series, manejo de mensajes semánticos, creación de scripts y transacciones.

La última meta es permitirle a usted la creación de componentes que se comporten como *objetos de negocios*. Estos son componentes que siguen el modelo de sus contrapartes reales en algún dominio al nivel aplicación. Por lo general desempeñan funciones de negocios específicas, un cliente, automóvil u hotel, por ejemplo. Usted puede agrupar estos objetos de negocios en series visuales asentadas en un escritorio pero con redes subyacentes de cliente/servidor.

En consecuencia, el nirvana total en la actividad de componentes de cliente/servidor son componentes de objetos de negocios superinteligentes que hacen mucho más que simplemente interoperar: colaboran en el nivel semántico para lograr la realización de labores. Por ejemplo, los agentes nómadas de una red global deben estar en condiciones de colaborar en el establecimiento de negociaciones con otros agentes. Los agentes son un ejemplo de objetos de negocios. La infraestructura brinda estándares de colaboración al nivel aplicación en forma de *estructuras de aplicaciones*. Estas estructuras vigilan el cumplimiento de las reglas de participación entre componentes independientes y les permiten colaborar en series.

En la Figura 22-3 se muestra la evolución de los componentes de la interoperabilidad a la colaboración. Esta evolución corresponde a las fronteras de servicios de la infraestructura de componentes. El bus de componentes ofrece interoperabilidad simple; los servicios del sistema aportan componentes superinteligentes, y las estructuras de aplicaciones contribuyen con la semántica del nivel de aplicación para que los componentes colaboren en series.

CONCLUSIÓN

Los objetos son de interés para todos: gerentes, niños de 3 años y superprogramadores.
La tecnología orientada a objetos atrae a todos estos grupos diferentes.

Jim Gray
(Febrero de 1995)

Los componentes distribuidos —modelados como objetos de negocios— se ajustan a la perfección a arquitecturas cliente/servidor en 3 planos. Ofrecen soluciones ampliables y flexibles para entornos de cliente/servidor intergalácticos, así como para Internet e intranets. Los objetos de negocios pueden descomponerse y dividirse naturalmente entre múltiples planos para satisfacer las necesidades de una aplicación. Son glóbulos inteligentes, autodescriptivos y autogestionarios que pueden ser desplazados y ejecutados donde mejor convenga. Pero, sobre todo, los objetos de negocios son evolutivos: no le obligan a deshacerse de las aplicaciones servidor con las que ya cuenta y a empezar desde cero. Puede encapsular lo que ya tiene y añadir crecientemente nueva inteligencia, a razón de un componente por vez. En los siguientes capítulos describiremos las infraestructuras de componentes distribuidos de CORBA/OpenDoc y OLE/DCOM que hacen posible en gran medida esta magia.

Capítulo 23

CORBA: De ORB a objetos de negocios



Orbe (orb): Esfera enjoyerada coronada por una cruz que forma parte de las insignias de un soberano y que simboliza poder y justicia monárquicos.

American Heritage Dictionary

ORB: Tendido de pavimento en el camino de terracería llamado computación distribuida.

Chris Stone, presidente de OMG

La arquitectura común de corredores de solicitudes de objetos (CORBA: *common object request broker architecture*) es el proyecto de middleware más importante (y ambicioso) que haya sido emprendido jamás en la industria. Es resultado de un consorcio —llamado Object Management Group (OMG)— integrado por más de 650 compañías en representación de la totalidad del espectro de la industria de la computación. La excepción notable es Microsoft, con su propio bus de objetos, el *modelo de objetos de componentes distribuidos (DCOM: distributed component object model)*. Para el resto de la industria, la próxima generación de middleware es CORBA. El bus de objetos de CORBA define la forma de los componentes que lo habitan y cómo interoperan. En consecuencia, al haber optado por un bus abierto, la industria optó también por la creación de un campo de juego abierto para los componentes.

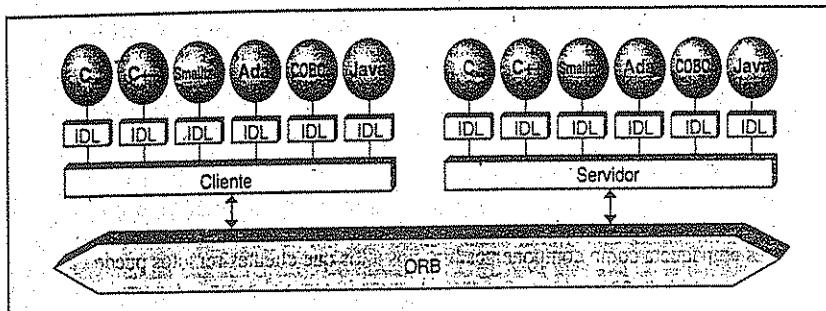


Figura 23-1. Las ligaduras de lenguaje de IDL de CORBA brindan interoperabilidad de cliente/servidor.

datos. La gramática de IDL es un subconjunto de C++ con palabras clave adicionales para soportar conceptos distribuidos; también soporta plenamente características y pragmas (enunciados de comentarios) de preprocesamiento de C++ estándar.

La ambiciosa meta de CORBA es "IDLizar" todo el middleware cliente/servidor y todos los componentes residentes en un ORB. OMG espera alcanzar esta meta siguiendo dos pasos: 1) convertirlo todo en clavos y 2) darles a todos un martillo.

- El "clavo" es el IDL de CORBA. Éste permite a los proveedores de componentes especificar en un lenguaje de definición estándar la interfaz y estructura de los objetos que ofrecen. Un contrato definido en IDL obliga a los proveedores de servicios de objetos distribuidos con sus clientes. Para que un objeto pueda solicitar algo de otro objeto, debe conocer la interfaz del objeto destino. El *depósito de interfaces* de CORBA contiene las definiciones de todas estas interfaces. Contiene los metadatos, gracias a los cuales los componentes pueden descubrirse dinámicamente entre sí en tiempo de ejecución. Esto hace de CORBA un sistema autodescriptivo.
- El "martillo" incluye el conjunto de servicios distribuidos que brindarán los proveedores de OMG. Estos servicios determinarán qué objetos habrán de encontrarse en la red, cuáles métodos proporcionarán y qué adaptadores de interfaces de objetos soportarán. La ubicación del objeto deberá ser transparente para el cliente y la implementación del objeto. No tendrá por qué importar que el objeto se halle en el mismo proceso o al otro lado del mundo.

¿Le suena conocido todo esto? Debería. Estamos describiendo la "ola de los objetos" de la computación de cliente/servidor, aunque en este caso entre objetos cooperadores, en oposición a procesos cooperadores. El objetivo de esta nueva ola es crear con objetos "legoware" de lenguajes, multiOS y proveedores múltiples. Proveedores como Sun, HP, IBM, Digital, Tandem y NCR ya están haciendo uso de CORBA y su interfaz estándar definida en IDL para incorporarse a la carretera de los objetos. El IDL es el contrato en el que se reúnen todos los elementos.



Componentes de CORBA: De objetos del sistema a objetos de negocios

Los objetos pueden variar enormemente en tamaño y número. Pueden representarlo todo, desde el hardware hasta todo lo que implica llegar a aplicaciones de diseño completo. ¿Cómo decidir qué debe ser un objeto?

Erich Gamma y cols. autores
Design Patterns
(Addison-Wesley, 1994)

Note que hemos estado usando indistintamente los términos "componentes" y "objetos distribuidos". Por definición, los objetos distribuidos de CORBA son componentes, debido a la forma en que se les empaqueta. En los sistemas de objetos distribuidos, la unidad de trabajo y distribución es un componente. La infraestructura de objetos distribuidos de CORBA les facilita a los componentes ser más autónomos, autoadministrables y colaborativos. Este esfuerzo es mucho más ambicioso que todos los intentos realizados hasta ahora por modalidades competitivas de middleware. La tecnología de objetos distribuidos de CORBA nos permite conjuntar complejos sistemas de información de cliente/servidor con sólo ensamblar y extender compo-



neutral, lo que hace posible llamar a objetos entre fronteras de lenguajes y sistemas operativos. Por el contrario, otros tipos de middleware proporcionan por lo general bibliotecas de API específicas de lenguajes de bajo nivel. Además, no distinguen entre implementación y especificación; la API está estrechamente acoplada con la implementación, lo que la vuelve sumamente sensible a cambios.

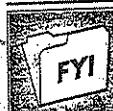
Sistema autodescriptivo. CORBA suministra metadatos en tiempo de ejecución para la descripción de todas las interfaces del servidor conocidas por el sistema. Todo ORB de CORBA debe soportar un *depósito de interfaces* que contenga información en tiempo real sobre las funciones ofrecidas por un servidor y sus parámetros. Los clientes recurren a los metadatos para saber cómo invocar servicios en tiempo de ejecución, mientras que las herramientas les son útiles para generar código "sobre la marcha". Los metadatos son generados automáticamente ya sea por un precompilador de lenguaje de IDL o por compiladores aptos para generar IDL directamente de un lenguaje OO. Por ejemplo, el compilador MetaWare C++ genera IDL directamente de definiciones de clase de C++ (y también escribe directamente esa información en el depósito de interfaces). Para mayor conocimiento, ninguna otra modalidad de middleware cliente/servidor brinda este tipo de metadatos en tiempo de ejecución, ni definiciones independientes de lenguaje de todos sus servicios. Como se comprobará más adelante, los objetos y componentes de negocios requieren toda la flexibilidad de acoplamiento posterior que se les pueda otorgar.

Transparencia local/remota. Un ORB puede correr en modo autónomo en una laptop, o interconectarse con todos los demás ORB del universo (al usar servicios interORB de CORBA 2.0). Puede servir de intermediario para llamadas entre objetos dentro de un solo proceso, en procesos múltiples en ejecución dentro de la misma máquina o en procesos múltiples en ejecución entre redes y sistemas operativos. Todo esto sucede de manera transparente para los objetos propios. Advierta que el ORB puede intermediar entre objetos de grano fino —como los de clases C++— así como entre objetos de grano grueso. En general, un programador de CORBA en cliente/servidor no tiene que ocuparse de transportes, ubicaciones del servidor, activación de objetos, pedidos de bytes entre plataformas disímiles, ni sistemas operativos destino; CORBA vuelve transparente todo esto.

Seguridad y transacciones integradas. El ORB incluye en sus mensajes información de contexto para el manejo de seguridad y transacciones entre fronteras de máquinas y ORB.

Manejo de mensajes polimórficos. En contraste con otras modalidades de middleware, un ORB no se limita a invocar una función remota; invoca una función de un objeto destino. Esto significa que la misma llamada de función tendrá diferentes efectos, dependiendo del objeto que la reciba. Por ejemplo, una invocación de método *configurarse_a_si_mismo* (*configure_yourself*) se comporta de manera diferente si se le aplica a un objeto de base de datos que si se aplica a un objeto de impresora (véase el siguiente recuadro de información).

Por supuesto que CORBA tiene también su dotación de deficiencias. Las examinaremos en un recuadro de debate al final de este capítulo, titulado "*¿Estamos en El Año del ORB?*".



información

ORB contra RPC

Todos los corredores —lo mismo de acciones que de objetos— cobran por sus servicios.

Bill Andreas, arquitecto en jefe
HyperDesk

¿En qué se diferencian las invocaciones de métodos de ORB de las RPC? Los mecanismos son muy parecidos, pero aun así hay algunas diferencias importantes. En el caso de una RPC, usted llama a una función específica (los datos son aparte). En cambio, en el caso de un ORB llama a un método dentro de un objeto específico. Diferentes clases de objetos pueden responder a la misma invocación de método de diferente manera, gracias a la magia del polimorfismo. Dado que cada objeto administra sus propios datos de instancia privada, el método se implementa en esos datos de instancia específicos (véase Figura 23-3).

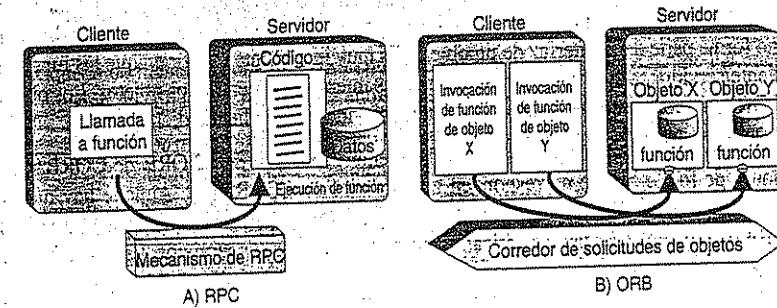


Figura 23-3. ORB contra RPC.

Las invocaciones de métodos de ORB poseen una precisión digna de un bisturí. La llamada se dirige a un objeto específico que controla datos específicos, y que después implementa la función a la manera específica de su clase. Por el contrario, las llamadas de RPC carecen de especificidad; todas las funciones del mismo nombre se implementan de la misma manera. En su caso no existen servicios diferenciados.

Desde luego que algunos ORB se erigen sobre un servicio de RPC, de manera que se termina pagando una sanción de desempeño por este "refinado" nivel de servicio. El desembolso vale la pena si se obtiene provecho de los nuevos niveles de granularidad distribuida provistos por los objetos. De no ser así, usted no habrá hecho otra cosa que



Identificadores de depósitos globales de CORBA 2.0

Información

Los ORB de CORBA 2.0 cuentan con identificadores globales —llamados *identificadores de depósitos*— para la identificación exclusiva y global de un componente y su interfaz a lo largo de ORB y depósitos de proveedores múltiples. Los identificadores de depósitos son cadenas únicas generadas por sistemas que sirven para guardar consistencia en las convenciones de nombramiento de múltiples depósitos, a fin de impedir colisiones de nombres. Se les genera a través de *pragmas* en IDL. El pragma especifica si generarlos por medio de *identificadores universales únicos* (UUID: *universal unique identifier*) de DCE o de prefijos únicos aportados por el usuario y agregados a los nombres del alcance de IDL. El identificador de depósitos mismo es una cadena compuesta por una jerarquía de nombres de tres niveles.

tarse. Las invocaciones dinámicas ofrecen máxima flexibilidad, pero su programación es difícil; sin embargo, son muy útiles para herramientas de descubrimiento de servicios en tiempo de ejecución.

La parte del servidor es incapaz de distinguir la diferencia entre una invocación estática o dinámica; las dos poseen la misma semántica de mensajes. En ambos casos, el ORB localiza un adaptador de objetos del servidor, transmite los parámetros y transfiere el control a la implementación de los objetos por medio del talón (o esqueleto) de IDL del servidor. He aquí lo que hacen los elementos de CORBA en la parte del servidor de la Figura 23-4:

- Los *talones de IDL del servidor* (OMG les llama *esqueletos*) brindan interfaces estáticas para cada uno de los servicios exportados por el servidor. Lo mismo que los del cliente, estos talones se crean usando un compilador de IDL.
- La *interfaz de esqueleto dinámico* (DSI: *dynamic skeleton interface*) —introducida en CORBA 2.0— ofrece un mecanismo de vinculación en tiempo de ejecución para servidores que deben manejar la recepción de llamadas de métodos destinadas a componentes que carecen de esqueletos (o talones) compilados con base en IDL. El esqueleto dinámico busca valores de parámetros en un mensaje recibido para deducir a quién va dirigido; es decir, para identificar al objeto y método de destino. En contraste, los esqueletos compilados normales se definen para una clase de objetos en particular, y de ellos se espera una implementación de método para cada método definido en IDL. Los esqueletos dinámicos son muy útiles para la implementación de puentes genéricos entre ORB. También pueden ser usados por intérpretes y lenguajes de creación de scripts para la generación dinámica de implementaciones de objetos. La DSI es el equivalente en el servidor de una DLL. Puede recibir invocaciones del cliente estáticas o dinámicas.
- El *adaptador de objetos* se sitúa encima de los servicios de comunicación del núcleo del ORB y acepta solicitudes de servicios en nombre de los objetos del servidor. Aporta el

entorno de tiempo de ejecución para la representación de objetos del servidor, la transmisión de solicitudes dirigidas a ellos y la asignación a estos mismos de identificadores de objetos, llamados en CORBA *referencias de objetos*. El adaptador de objetos también registra las clases que soporta y sus instancias en tiempo de ejecución (es decir, objetos) con el *Depósito de Implementación*. CORBA especifica que cada ORB debe soportar un adaptador estándar, llamado *adaptador básico de objetos* (BOA: *basic object adapter*). Los servidores pueden soportar más de un adaptador de objetos.

- El *depósito de implementaciones* constituye un depósito de información en tiempo de ejecución sobre las clases que soporta un servidor, los objetos representados y sus identificadores. Funge asimismo como punto común para el almacenamiento de información adicional asociada con la implementación de ORB. Como ejemplos pueden citarse información de seguimiento, rastros para auditoría, seguridad y otros datos administrativos.
- La *interfaz de ORB* consiste en unas cuantas API para servicios locales que son idénticos a los provistos en la parte del cliente.

Concluye así nuestra visión panorámica de los componentes del ORB y sus interfaces.

CORBA 2.0: EL ORB INTERGALÁCTICO

CORBA 1.1 se limitó a la creación de aplicaciones de objetos exportables; la implementación del núcleo de ORB quedó como “ejercicio para los proveedores”. El resultado fue cierto nivel de exportabilidad de componentes, pero no interoperabilidad. CORBA 2.0 aportó la interoperabilidad al especificar un *protocolo entre ORB en Internet* (IIOP: *Internet inter-ORB protocol*) de cumplimiento obligatorio. El IIOP es básicamente TCP/IP con algunos intercambios de mensajes definidos por CORBA, los cuales hacen las veces de protocolo de red principal común. Todo ORB que se declare observante de CORBA debe implementar nativamente el IIOP, o brindar un “medio puente” para él. Nota: se le llama medio puente porque IIOP es la red principal “estándar” de CORBA. Así, todo ORB propietario puede conectarse con el universo de ORB traduciendo solicitudes a y desde la columna vertebral IIOP.

Además de IIOP, CORBA soporta *protocolos entre ORB específicos de entornos* (ESIOP: *environment-specific inter-ORB protocols*) para la interoperación “fuera de cuadro” en redes específicas. CORBA 2.0 especifica a DCE como el primero de muchos ESIOP opcionales. El ESIOP de DCE brinda un entorno robusto para ORB de misión crítica (véase el siguiente cuadro de debate). Lo soportan ya *ORB Plus* de HP y *ObjectBroker* de Digital; eventualmente lo soportará SOM de IBM.

Los puentes entre ORB y los IIOP pueden servir para crear topologías muy flexibles vía federaciones de ORB. En la Figura 23-5 se observa una red principal (backbone) de IIOP con varios ORB propietarios conectados a ella vía medios puentes. Advierta la presencia del ESIOP de DCE. Los ORB pueden segmentarse en dominios con base en necesidades administrativas, implementaciones de ORB de proveedores, protocolos de redes, cargas de tráfico, tipos de servicios y asuntos de seguridad. Las políticas de cada lado de la cerca pueden entrar en conflicto, de manera que se pueden crear muros de protección (firewalls) alrededor de la red principal del

lo. Desde luego que Microsoft está, cuando menos, dos años atrás de CORBA en su búsqueda de objetos distribuidos. De manera que a los puristas de DCE no les queda sino aguardar a que Microsoft dé en el blanco, o aprender a coexistir con IIOP.

DCE necesita a CORBA y CORBA necesita a DCE. DCE favorece a CORBA con una infraestructura de RPC de misión crítica. CORBA favorece a DCE con una plataforma abierta para objetos. Nada impide que se tiendan puentes a IIOP, en tanto ambas orillas usen el mismo modelo de objetos de CORBA 2.0. Así, a pesar de nuestro profundo aprecio por IIOP, creemos que también DCE/ESIOP es crucial para el éxito de CORBA. CORBA necesita de los dos. La grandeza de OMG fue haber sintetizado tecnologías rivales. Por ejemplo, fundió dos propuestas en pugna para ofrecernos invocaciones tanto estáticas como dinámicas. La fusión de DCE e IIOP en CORBA 2.0 es de igual categoría: nos da dos tecnologías de ORB complementarias. Entonces, saquemos provecho de esta situación y ocupémonos de cosas más importantes, como los objetos de negocios. □

SERVICIOS DE OBJETOS DE CORBA

Los servicios de objetos de CORBA son conjuntos de servicios del nivel del sistema, empaquetados con interfaces especificadas de IDL. Se les puede concebir como adiciones y complementos a la funcionalidad del ORB. Sirven para crear un componente, nombrarlo e introducirlo en el entorno. A mediados de 1996, OMG ya había definido estándares para trece servicios de objetos:

- El *servicio de ciclo de vida* define las operaciones para la creación, copia, desplazamiento y eliminación de componentes en el bus.
- El *servicio de persistencia* ofrece una interfaz única para el almacenamiento persistente de componentes en una amplia variedad de servidores de almacenamiento, como bases de datos de objetos (ODBMS), bases de datos relacionales (RDBMS) y archivos simples.
- El *servicio de nombramiento* permite que componentes en el bus localicen a otros componentes por su nombre; asimismo, soporta contextos de nombramiento en federación. Este servicio hace posible ligar a objetos con directorios de redes o contextos de nombramiento ya existentes, como X.500 de ISO, DCE de OSF y NIS+ de Sun.
- El *servicio de eventos* permite a componentes en el bus registrar o des registrar dinámicamente su interés en eventos específicos. Este servicio define al bien conocido objeto llamado *canal de eventos*, que recoge y distribuye eventos entre componentes que lo ignoran todo unos de otros.
- El *servicio de control de concurrencia* aporta un administrador de candados capaz de obtenerlos en beneficio ya sea de transacciones o hilos.
- El *servicio de transacciones* provee coordinación de grabación en dos fases entre componentes recobrables por medio de transacciones simples o anidadas.

- El *servicio de relaciones* ofrece un medio para la creación de asociaciones (o vinculaciones) dinámicas entre componentes que se desconocen entre sí. También brinda mecanismos para el recorrido de los vínculos que agrupan a estos componentes. Este servicio puede ser útil para reforzar la contención de integridad referencial, el seguimiento de relaciones y para cualquier otro tipo de vínculos entre componentes.
- El *servicio de externalización* contribuye con un medio estándar para la introducción y obtención de datos de un componente con una variante del mecanismo de flujo.
- El *servicio de consulta* proporciona operaciones de consulta para objetos. Se trata de un superconjunto de SQL basado en la inminente especificación SQL3 y en el *lenguaje de consulta de objetos (OQL: object query language)* del Object Database Management Group (ODMG).
- El *servicio de licencias* cuenta con operaciones para la medición del uso de componentes a fin de asegurar compensaciones justas por concepto de utilización. Soporta cualquier modelo de control de uso en cualquier punto del ciclo de vida de un componente. Soporta asimismo cobro por sesión, por nodo, por creación de instancias y por ubicación.
- El *servicio de propiedades* consta de operaciones para asociar valores nombrados (o propiedades) con cualquier componente. Con este servicio es posible asociar propiedades dinámicamente con el estado de un componente, por ejemplo, un título o fecha.
- El *servicio de tiempo* ofrece interfaces para el tiempo de sincronización en un entorno de objetos distribuidos. También ofrece operaciones para la definición y administración de eventos activados por tiempo.
- El *servicio de seguridad* brinda una estructura completa para la seguridad de objetos distribuidos. Soporta autenticación, listas de control de acceso, confidencialidad y no repudio. Se encarga asimismo de la delegación de credenciales entre objetos (véase el siguiente cuadro de información).

A fines de 1996 debieron estar listos otros tres servicios: negociador, recolecciones e iniciación. El servicio *negociador* equivale a la "sección amarilla" para objetos; permite que éstos publiciten sus servicios y compitan por la adjudicación de labores. El servicio de *recolecciones* ofrece interfaces de CORBA para la creación y manipulación genéricas de las recolecciones más comunes. Finalmente, el servicio de *iniciación* permitirá el arranque automático de solicitudes al momento de invocación de un ORB. Todos estos servicios enriquecen el comportamiento de un componente y tienden el robusto entorno en el que éste pueda residir y operar en condiciones de protección.

En la Figura 23-6 se muestran los calendarios de horarios de *solicitud de propuestas (RFP: request for proposal)* que usa OMG para el desarrollo de especificaciones de servicios de objetos. Las RFP de OMG son solicitudes de una tecnología. Resultan en respuestas de los miembros sobre cómo implementar un estándar en particular. Los miembros deben basar sus propuestas en productos existentes o en desarrollo (se necesita alguna prueba del concepto). Por lo general una RFP se cumple mediante la fusión de las propuestas presentadas por varias organizaciones. A partir de la emisión de una RFP por parte de OMG, transcurren alrededor de 12 a 16 meses hasta la obtención de un estándar práctico. Como puede verse, OMG ya casi ha

El ORB puede manejar la seguridad de una amplia variedad de sistemas, desde dominios autorizados (con una sola máquina o proceso) hasta situaciones entre ORB intergalácticas. Los componentes que no son responsables de instrumentar su propia seguridad son más fáciles de desarrollar, administrar y exportar entre entornos. Además, insertar la seguridad en el ORB puede reducir al mínimo la carga de desempeño. El servicio de seguridad de CORBA —adoptado por OMG en marzo de 1996— se ocupa de *todos* estos requerimientos; se trata quizás del estándar de seguridad de cliente/servidor más completo que haya habido hasta ahora. □

Servicios de objetos: Middleware sobre pedido

Es importante advertir que los servicios de objetos de CORBA constituyen un método excepcional para la creación de middleware *sobre pedido*. No hay nada parecido en los actuales sistemas clásicos de cliente/servidor. Con CORBA, los proveedores de componentes pueden desarrollar sus objetos sin preocuparse en absoluto por los servicios del sistema. Entonces, dependiendo de las necesidades del cliente, el desarrollador (o integrador de sistemas) puede mezclar el componente original con cualquier combinación de servicios de CORBA para crear la función que se necesita. Esto es posible mediante la subclasificación de la clase original, y su posterior mezcla con las clases de servicios de objetos requeridas a través de la herencia múltiple. Todo se realiza además por intermedio del IDL; no se precisa de código fuente. Por ejemplo, usted puede desarrollar un componente denominado "automóvil" y crear una versión concurrente, persistente y apta para transacciones del automóvil por herencia múltiple de los servicios correspondientes.

Además, algunos proveedores de ORB se beneficiarán de sus extensiones de CORBA metaclases, con lo que a usted le será posible crear sus *mixins* en tiempo de creación de objetos. Una *metaclase* es una clase y al mismo tiempo un objeto en tiempo de ejecución. Por ejemplo, SOM de IBM extiende CORBA mediante el tratamiento de clases como objetos de primera clase. Esto significa que se pueden crear y personalizar nuevas clases en tiempo de ejecución. Las fábricas de objetos pueden hacer uso de estas facilidades de metaclases para componer una clase en tiempo de ejecución con base en la solicitud de un cliente. Usted puede crear clases sobre pedido por herencia múltiple de los servicios de objetos existentes. Por ejemplo, la fábrica puede tomar un componente ordinario, como un "auto", y volverlo apto para transacciones, bloqueable y seguro por herencia múltiple de las clases de servicios de objetos existentes. Este método es la modalidad más avanzada de middleware sobre pedido. Lo maravilloso del asunto es que bien podría ocurrir que el proveedor del componente original no sepa nada de transacciones, seguridad ni candados. Estos servicios se añaden dinámicamente al componente en tiempo de creación en fábrica con base en los requerimientos del cliente.

Si la herencia múltiple no es de su gusto, algunas implementaciones de ORB le permiten agregar métodos "sobre la marcha" a clases existentes. En particular, puede añadir llamadas de verificación *anteriores* y *posteriores*, para que sean activadas antes y después de ejecutado cualquier método ordinario. Estas llamadas anteriores y posteriores pueden servirle para demandar cualquiera de los servicios de CORBA existentes o, para el caso, cualquier cosa exis-

tente en un ORB. Incluso puede anexar scripts a desencadenantes anteriores/posteriores. Por ejemplo, puede usar un desencadenante *anterior* para obtener un candado del servicio de concurrencia, mientras que con el desencadenante *posterior* se soltaría el candado.

Por medio de la combinación de la tecnología de metaclasas con los servicios de CORBA, usted podrá crear entornos de middleware "personalizados a última hora" para la ejecución de componentes particulares. Ésta es una demostración de la máxima flexibilidad de los objetos. Es probable que la mayoría de los desarrolladores de componentes adopten una metodología más conservadora y creen sus mixins en tiempo de compilación o a través de una herramienta en tiempo de formación. En ambos casos, esto seguirá siendo mucho más flexible que cualquier otra cosa que pueda hacerse con el actual middleware cliente/servidor.

FACILIDADES COMUNES DE CORBA

Las *facilidades comunes* son recopilaciones de componentes definidos en IDL que prestan servicios de uso directo a objetos de aplicaciones. Se les puede concebir como un paso más en la jerarquía semántica. Las dos categorías de facilidades comunes —horizontal y vertical— definen reglas de participación en las que es indispensable la colaboración efectiva de los componentes de negocios. Para darle una idea del estado actual de las cosas, en octubre de 1994 OMG emitió la *solicitud de propuestas 1 (RFPI)* de facilidades comunes para recibir sugerencias de tecnología para documentos compuestos. En marzo de 1996 adoptó OpenDoc como tecnología de documentos compuestos, a la que ha llamado *facilidad de componentes de documentos distribuidos (DDCF: distributed document component facility)*. DDCF especifica los servicios de presentación para componentes y un estándar de intercambio de documentos basado en Bento de OpenDoc.

Las facilidades comunes, actualmente en proceso de elaboración, incluyen agentes móviles, intercambio de datos, estructuras de objetos de negocios e internacionalización. Al igual que el sistema de carreteras, las facilidades comunes son un proyecto interminable. Las labores proseguirán hasta que CORBA defina interfaces en IDL para todos los servicios distribuidos de que tenemos noticia en la actualidad, lo mismo que para aquellos aún por inventar. Cuando esto ocurra, CORBA ofrecerá interfaces en IDL para prácticamente todos los servicios distribuidos conocidos hasta la fecha (muchas de las cuales serán versiones "IDLizadas" de middleware existente).

OBJETOS DE NEGOCIOS DE CORBA

Un objeto de negocios es una representación de un ente activo en el dominio empresarial, incluidos al menos su nombre y definición de negocios, atributos, comportamiento, relaciones y restricciones. Un objeto de negocios puede representar, por ejemplo, una persona, lugar o concepto.

OMG, Business Object Task Force

Los objetos de negocios ofrecen un medio natural para la descripción de conceptos independientes de aplicaciones tales como cliente, pedido, competidor, dinero, pago, automóvil y pa-

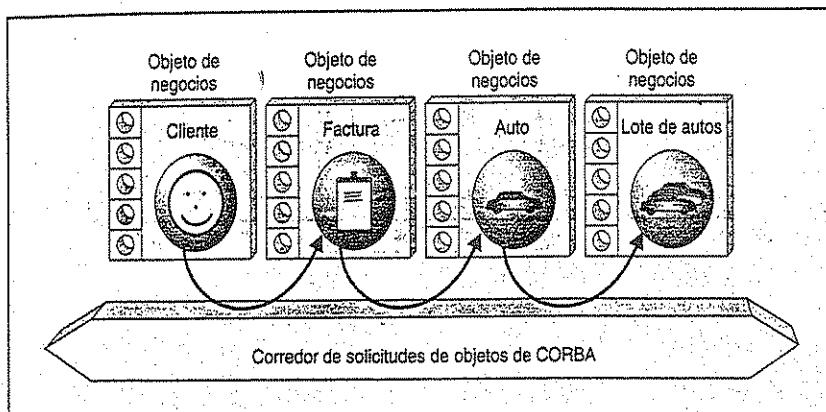


Figura 23-7. Sistema de reserva de autos con objetos de negocios cooperadores.

¿En qué se diferencia esto de una aplicación tradicional? Con apenas un mínimo esfuerzo, usted puede reutilizar algunos de estos objetos de negocios en otro contexto de aplicación. Por ejemplo, en un programa de venta de automóviles podrían reutilizarse casi todos estos objetos, especialmente si fueron diseñados para operar con más de una serie semántica. Los objetos auto, cliente y factura, por ejemplo, podrían soportar múltiples vistas para el manejo de diferentes situaciones de negocios. En un caso extremo, los objetos de negocios podrían especializarse mediante herencia para tomar en cuenta las particularidades de la actividad comercial de venta de automóviles. Como se verá en la siguiente sección, un objeto de negocios no es una entidad monolítica. Dentro se descompone en un conjunto de objetos cooperadores capaces de reaccionar a situaciones de negocios diferentes. Los objetos de negocios son sumamente flexibles.

Anatomía de un objeto de negocios de CORBA

Los objetos de negocios representan directamente el modelo a escala de la empresa, el cual se vuelve parte de un sistema de información. Toda persona, lugar, cosa, evento, transacción o proceso en la empresa, puede ser representado por un objeto activo en el sistema de información.

Cory Casanave, director de OMG
(Junio de 1995)

Un objeto de negocios de OMG es una variante del paradigma *modelo/vista/controlador (MVC)*. MVC es un patrón de diseño de objetos, usado para la creación de interfaces en Smalltalk y en casi todas las bibliotecas de clase de GUI. Se compone de tres tipos de objetos. El *modelo* representa al objeto de aplicación y sus datos encapsulados. La *vista* representa visualmente al objeto en la pantalla. El *controlador* define el modo en que reacciona la interfaz del usuario a entradas del usuario y eventos de GUI.



En el modelo de OMG, un objeto de negocios también se compone de tres tipos de objetos (véase Figura 23-8):

- Los *objetos de negocios* encapsulan el almacenamiento, metadatos, concurrencia y reglas de negocios asociados con una entidad de negocios activa. Definen asimismo la manera en que reacciona el objeto a cambios en las vistas o el modelo.
- Los *objetos de procesos de negocios* encapsulan la lógica de negocios al nivel empresarial. En los sistemas tradicionales de *modelo-vista-controlador*, el controlador está a cargo del proceso. En el modelo de OMG, las funciones de procesos de corta vida son manejadas por el objeto de negocios. Los procesos de larga vida que involucran a otros objetos de negocios son manejados por el objeto de procesos de negocios, especialización del objeto de negocios que se ocupa de los procesos de larga vida y del entorno en su conjunto. Por ejemplo, sabe cómo manipular un flujo de trabajo o una transacción de larga vida. El objeto de procesos suele fungir como el lazo que une a los demás objetos. Define, por ejemplo, la manera en la que reacciona el objeto a un cambio en el entorno. Este tipo de cambio puede ser provocado por la ejecución de una transacción de negocios o por la recepción de un mensaje procedente de otro objeto de negocios. Cabe señalar que algunos objetos de negocios pueden estar completamente orientados a procesos y no asociados con datos ni presentaciones específicos.
- Los *objetos de presentación* representan visualmente al objeto para el usuario. Cada objeto de negocios puede poseer múltiples presentaciones para múltiples propósitos. Las presentaciones se comunican directamente con el objeto de negocios para la exhibición de datos en la pantalla. Y en ocasiones también se comunican directamente con el objeto de procesos. OMG reconoce asimismo la existencia de interfaces no visuales para objetos de negocios.

Un componente de objeto de negocios ordinario consiste en un objeto de negocios, uno o más objetos de presentaciones y un objeto de procesos. Estas entidades actúan como un solo cuerpo.

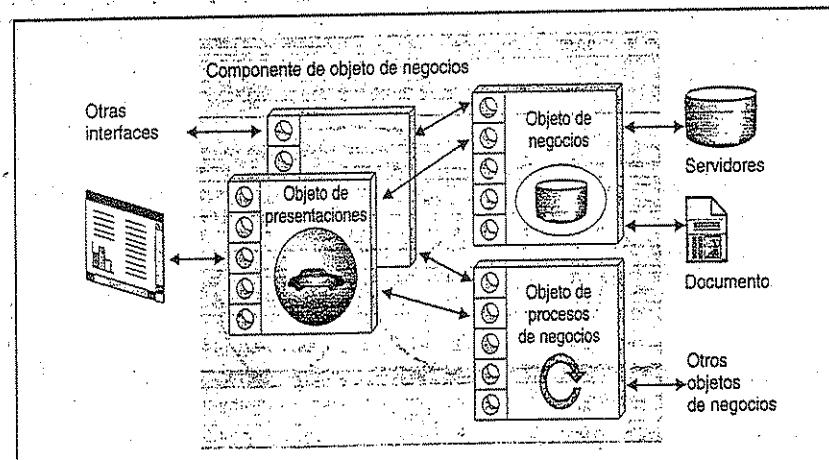


Figura 23-8. Anatomía de un objeto de negocios.



Suponemos que para principios de ese año los ORB se habrán vuelto obvios. Y que se dejará de hablar de ellos. Si quiere saber por qué, siga leyendo.

El argumento más sólido en favor de esta predicción se reduce primero a analizar qué le falta a CORBA, cuándo podemos esperar las piezas faltantes y cuáles, si algunos, son los obstáculos de esta tecnología. Pero, además, tenemos un as bajo la manga, con el que demostraremos sin lugar a dudas que 1997 es el año del ORB. Lo interesante para el frente de CORBA es que en esta ocasión nuestra lista de defectos es mucho más corta que la que presentamos en la edición anterior de este libro. Pero eso no quiere decir que no tengamos malas noticias sobre el frente de CORBA:

- **Los ORB comerciales son lentos e inefficientes.** La primera generación de ORB de CORBA es totalmente inconveniente para entornos de cliente/servidor para misión crítica. Casi ninguno de los ORB actuales se aviene a recoger la basura, ni realiza equilibrio de cargas ni control de concurrencia. Ninguno de los servidores de CORBA en el mercado es capaz de manejar millones de objetos de grano fino; sencillamente no son escalables. Además, no existen servidores tolerantes de fallas para ORB. Lo inquietante es que los objetos son inherentemente ampliables, y en alto grado, así como fácilmente duplicables para brindar tolerancia de fallas. Pero el problema es que, en la parte del servidor, los ORB comerciales en existencia no están a la par de los monitores de TP y los ODBMS, y ni siquiera de los RDBMS; no son productos de servidor maduros.

- **Dónde está MOM?** MOM significa *middleware orientado a mensajes (message-oriented middleware)*. Ofrece colas de mensajes asíncronos en las partes tanto del cliente como del servidor. Permite que clientes y servidores funcionen de acuerdo con sus propios tiempos y velocidades asignados, sin necesidad de hallarse activos simultáneamente. Los ORB deben proporcionar servicios de MOM para soportar usuarios

móviles y facilitar las comunicaciones en entornos heterogéneos. El grupo de trabajo de facilidades comunes de OMG se ocupa actualmente de una RFP de MOM para introducir en CORBA clases de manejo de mensajes y formación de colas.

- **El código del servidor no es muy exportable.** La especificación de CORBA no define suficientemente todas las interfaces requeridas para la generación de código del servidor exportable. En contraste con ello, la especificación para la generación de clientes plenamente exportables es muy completa. OMG está consciente de esta deficiencia y ha iniciado ya la búsqueda de una solución.
- **CORBA estándar no soporta metaclasses.** La habilidad de tratar a las clases como objetos de primera categoría es muy importante para objetos de negocios y middleware personalizable. En la sección sobre los servicios de objetos nos referimos a algunos de los milagros que las metaclasses son capaces de hacer. La flexibilidad en tiempo de ejecución provista por ellas es absolutamente necesaria para los objetos de negocios. Sería magnífico que OMG estandarizara las metaclasses de SOM (o algo equivalente) para crear un campo de juego nivelado para todos los proveedores. Hoy, los proveedores de CORBA basado en SOM llevan ventaja sobre sus competidores, a causa del soporte de metaclasses de SOM. Lo malo es que estas implementaciones de metaclasses de SOM no son exportables entre ORB de CORBA, lo que en realidad resulta vergonzoso.
- **IDL debe soportar extensiones de nivel semántico.** El IDL de OMG debe brindar extensiones para estructuras de mensajes semánticas. La idea es que los componentes puedan interactuar entre sí a nivel semántico. El IDL de CORBA debería aportar "pegamento" para resolver los desajustes entre lo que espera un componente del cliente y lo que ofrece un componente del servidor. En otras palabras, hace falta una modalidad de manejo de mensajes flexible y autodescriptiva. La Business Object Task Force comienza ya a resolver estos asuntos.

¿Alguno de estos factores representa un obstáculo? El único gran obstáculo es el primer elemento de nuestra lista: la carencia de robustas implementaciones comerciales. Sin que las demás cuestiones dejen de ser importantes y demandar soluciones, lo cierto es que en ausencia de implementaciones robustas del servidor, los ORB nunca desplazarán a las demás modalidades de middleware o entornos de aplicaciones cliente/servidor. En nuestra opinión, OMG ha creado un notable conjunto de estándares (con las deficiencias ya anotadas). La pelota está ahora del lado de los proveedores. Éstos deben ofrecer implementaciones comerciales suficientemente vigorosas para satisfacer la corriente básica de cliente/servidor. Quizá sean los proveedores de monitores de TP los mejor capacitados para ofrecer los primeros ORB realmente robustos. Creemos que la próxima generación de monitores de TP se llamará ORB. Debe tomarse en cuenta que los proveedores de monitores de TP intervinieron en la definición del servicio de transacciones de objetos de CORBA. Están perfectamente al tanto de la sinergia entre ORB y entornos clásicos de procesamiento de transacciones.

Así pues, ¿están listos los ORB para su temporada mayor? Están cerca. La actual generación de ORB es útil en entornos homogéneos para desarrollar una mejor comprensión de



sus clientes para crear una fachada para sus servicios. Es como el envío de Mosaic en esteroides. Este capítulo trata de los fundamentos de los documentos compuestos. En los dos siguientes capítulos nos referiremos a productos reales, como OpenDoc y OLE.

DOCUMENTOS COMPUSTOS: ¿POR QUÉ TANTO ALBOROTO?

Un *documento compuesto* no es más que una alegoría para la organización de conjuntos de componentes, tanto visualmente como a través de relaciones de contención. Es una *estructura* de integración de componentes visuales. El documento es en esencia un punto de reunión de componentes y datos que pueden proceder de las fuentes más variadas. Dado que los documentos son tan familiares, crean un paradigma muy natural para la introducción a gran escala de objetos "para las masas". Para la mayoría de los usuarios, su primera interacción con objetos ocurrirá por medio de los documentos compuestos.

El escritorio ilimitado

Los documentos compuestos son "lugares" de residencia de componentes. El propio escritorio se halla en proceso de convertirse en un gigantesco documento compuesto que integre en forma "ilimitada" servicios de aplicaciones y sistemas operativos. ¿Exageramos? Tal vez, pero basta pensar en que OLE ya es parte integral de Windows 95. Y OpenDoc será incluido en cada copia de OS/Warp y Macintosh. Tanto OpenDoc como OLE son estructuras para la creación de documentos compuestos y la administración de los componentes residentes en esos documentos. Eventualmente, todos los objetos que aparezcan en el escritorio —en Windows, OS/2 o Mac— serán componentes. La mayoría de estos componentes podrán contener otros componentes.

Así, el escritorio moderno se transformará sencillamente en un gigantesco contenedor de componentes, algunos de los cuales serán a su vez contenedores de otros componentes, y así sucesivamente. Los componentes pueden ser desplazados vía arrastrar y soltar desde el escritorio a cualquier contenedor visual (y viceversa) para adecuarse a las necesidades de "espacio" de trabajo del usuario. Pero, ¿no es esto lo mismo que introducir archivos en carpetas del escritorio? Sí, excepto que los componentes visuales que serán desplazados de esta manera son más inteligentes que cualquier cosa que resida actualmente en los escritorios. Adicionalmente, saben cómo colaborar con otros componentes para compartir bienes visuales y almacenamiento. En lugar de ver recuadros dentro de recuadros, lo que usted verá se asemeja más bien a un tapiz visual. Se trata en realidad de un tapiz autorganizado creado por componentes inteligentes.

Documentos de todas las formas

Un documento compuesto es principalmente un contenedor visual de componentes. Con las más recientes tecnologías de documentos compuestos —como OpenDoc—, estos componentes pueden ser de formas irregulares. Con formas irregulares podemos crear combinaciones de contenedor/componente cuyo único límite es la imaginación. Por ejemplo, es fácil imaginar que habrá contenedores en forma de aviones, estadios, jardines, ciudades, centros comerciales,



etcétera. Usted podrá crear aplicaciones arrastrando componentes desde paletas y soltándolos en contenedores. Es como un juego de *SimCity*, salvo que cada uno de estos componentes vive y puede interactuar directamente con un usuario (véase Figura 24-1). Lo maravilloso del asunto es que estos componentes pueden ser suministrados por diferentes proveedores sin conocerse entre sí. Los propios contenedores son también componentes; también pueden incrustarse dentro de otros componentes. Todo es sumamente recursivo.

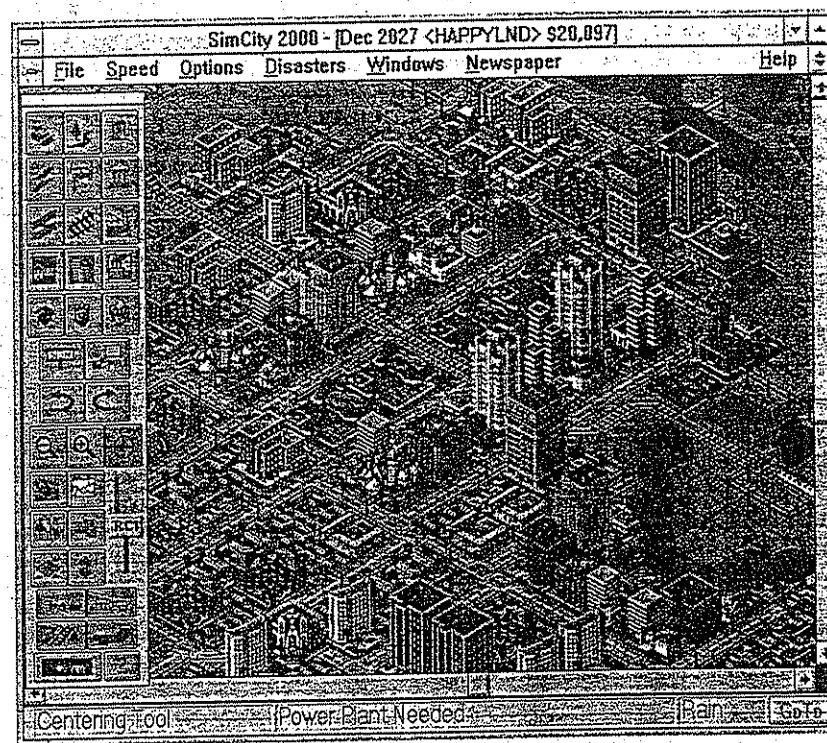


Figura 24-1. Si SimCity fuera un documento compuesto.

Una sede para todo tipo de datos

Los datos de los documentos compuestos son mucho más que texto en hojas de cálculo y documentos en papel. Pueden ser cualquier cosa: películas, sonidos, animación, controles, calendarios en red o carpetas virtuales. Cada nuevo tipo de medio que se desarrolla —video, sonido, animación, simulación, etc.— se puede representar por un componente en un documento. Los componentes de acceso a bases de datos pueden aportar información visual a usuarios y a otros componentes. Por ejemplo, por medio de scripts usted puede transmitir datos a un



centrales ya existentes. Consta de cuatro servicios de valor agregado para sus componentes: *diagramación de documentos*, *almacenamiento estructurado*, *creación de scripts/automatización* y *transferencia uniforme de datos*. En esta sección se ofrece una breve introducción a estas tecnologías. Nos extenderemos más sobre ellas en los capítulos dedicados a productos.

Diagramación de documentos

El servicio de *diagramación de documentos* define las reglas de participación que permiten que componentes independientemente desarrollados compartan la ventana del contenedor. Sirviéndose del contenedor como mediador, los componentes deben cooperar para producir un documento aparentemente incosútil para el usuario final. Recuérdese que el término "documento" se usa en el sentido de SimCity. Los contenedores activan los componentes incrustados en un documento y asignan a cada uno de ellos una pieza de los bienes visuales. Los componentes despliegan sus datos en el área que se les ha concedido e interactúan con el usuario.

Los contenedores distribuyen eventos a sus componentes y les notifican cuando ocurre algo interesante en sus inmediaciones. El contenedor también decide el uso de recursos compartidos; por ejemplo, una barra de menús compartida. Un contenedor debe ser capaz de aceptar un componente soltado dentro de sus límites y deducir qué hacer con él. Los componentes incrustados negocian con sus contenedores en caso de necesitar bienes adicionales en pantalla.

Almacenamiento estructurado

Un documento tradicional es un bloque monolítico de datos —en el interior de un archivo— controlado por una sola aplicación. Por el contrario, un documento compuesto consiste en muchos pequeños bloques de datos de contenido; cada bloque es controlado por su propio componente de software. El documento compuesto ofrece los protocolos que impiden que los componentes contaminen el documento común en el que residen.

Los documentos compuestos deben ser capaces de dividir un solo archivo en compartimientos de almacén que puedan ser asignados a componentes individuales. Cada componente se encarga de almacenar los datos que deseé en su almacén. Se les llama documentos compuestos porque pueden alojar diferentes tipos de datos. Cuando un documento compuesto es abierto por primera vez, sabe cómo encontrar y activar los componentes que manipulan los datos. En este caso, un componente es la información más el código que los manipula. El contenedor nunca toca ni manipula datos pertenecientes a sus componentes incrustados.

El *almacenamiento estructurado* es la tecnología que crea "un sistema de archivos dentro de un archivo" al proporcionar una capa de indirección sobre sistemas de archivo existentes. Cada componente recibe una estructura semejante a un directorio para organizar y describir el contenido de su almacenamiento; los datos como tales se almacenan en flujos. Estos flujos alojan lo mismo diminutos registros que gigantescos BLOB de datos; por ejemplo, una película completa, como *Lo que el viento se llevó*.



Los datos en un documento compuesto pueden provenir de una amplia variedad de fuentes, incluidos componentes foráneos y bases de datos de SQL. El contenedor aporta los enlaces adecuados para activar los componentes foráneos asociados con distintos elementos de datos. Lo hace incrustando directamente los datos externos en un documento o manteniendo apuntares —o vínculos— en el documento con las fuentes de datos externas. En ambos casos, el documento sigue siendo editable por varias aplicaciones; cada una de ellas percibe sus datos en formato nativa.

Creación de scripts y automatización

La interconexión y especialización de componentes del nivel de aplicación se conocen ahora como creación de scripts... Por supuesto que generar scripts es programar, pero no quisieramos decírles a los usuarios finales que lo que en realidad hacen es programar, ¿verdad?

Dave Thomas, presidente
Object Technology International
(Marzo de 1995)

La creación de scripts —también llamada *automatización*— es una de las grandes características del modelo de componentes de documentos compuestos. Los scripts permiten a los usuarios personalizar sus aplicaciones. En situaciones de documentos compuestos, hacen posible que usuarios de potencia y generadores de scripts creen relaciones personalizadas entre componentes en el documento por medio de facilidades estándar de edición de documentos. Los scripts permiten a los programadores e integradores de sistemas crear relaciones de cliente/servidor en las que se emplee la metáfora del documento para primeros planos del cliente. También les permiten crear documentos inteligentes.

Scripts adjuntos pueden permitirle a un documento realizar actividades de rastreo cada vez que se le lee o escribe. El documento puede notificarles a sus dueños, vía correo electrónico, cada vez que se lee su documento. El script también puede controlar lo que se exhibe en un documento basado en la autoridad de un usuario. Podría solicitar una contraseña, validar firmas digitales y consultar con un servidor de autenticación y una base de datos de capacidades antes de permitirle a usted ver partes del documento. Un documento inteligente puede acomodarse por si solo a sus preferencias. Los scripts también pueden introducir contenido dinámicamente; por ejemplo, usted podría recurrir a un script para consultar una bodega de datos e introducir los datos.

Scripts y documentos compuestos se complementan magníficamente entre sí. El documento aloja varios scripts y los invoca cuando se disparan ciertos eventos; por ejemplo, cuando usted abre o cierra un documento. A cambio de ello, los scripts protegen el documento y contribuyen con la inteligencia que lo vuelve autosuficiente. El resultado de la unión de estas dos tecnologías son documentos inteligentes autogestionarios. Usted también puede hacer uso de esta tecnología para crear componentes móviles que deambulen a lo largo de redes y realicen toda clase de trabajos útiles. Por primera vez disponemos de tecnología comercial que nos permite

CONCLUSIÓN

Los documentos compuestos ofrecen una estructura y metáfora visual para la organización de componentes. Brindan protocolos de negociación por medio de los cuales componentes incrustados puedan fundir sus elementos de interfaz del usuario en el espacio de ventana del contenedor y compartir un archivo de documento. Los scripts insertados añaden inteligencia al documento y lo protegen de amenazas. Dado que los propios sistemas operativos de escritorio se están convirtiendo en estructuras de documentos compuestos, ofrecerán un mercado masivo de componentes capaces de operar con las nuevas reglas de participación. Recuerde que estas reglas de participación son fundamentalmente las provistas por el bus y servicios de objetos subyacentes. Los documentos compuestos introducen un alto nivel de colaboración.

Los documentos compuestos son particularmente atractivos porque brindan una potente metáfora para la integración de componentes. Las estructuras de documentos compuestos le permiten visualizar los componentes, almacenarlos en un archivo común, intercambiarlos por medio de transferencias de datos y extenderlos con el uso de scripts y automatización. Además, todo se construye sobre un bus de objetos, el cual permite que estos componentes operen en redes intergalácticas y sean empaquetados en bibliotecas dinámicas de lenguaje independiente.

Por supuesto que el precio de vivir dentro de una estructura es la aceptación de las restricciones (y reglas de participación) que impone a fin de acrecentar los beneficios. Se trata de una propuesta de todo o nada. Las restricciones impuestas por una estructura de documentos compuestos incluyen sus protocolos para compartir un archivo común, para la negociación de propiedades visuales y para el intercambio de información con el mundo exterior. Los componentes ya no serán independientes. Residirán dentro del documento. También, recurrirán al contenedor como intermediario para recibir eventos, compartir recursos y comunicarse entre sí. En respuesta, alcanzan altos niveles de colaboración visual y pueden ser distribuidos a través de canales comerciales masivos. Se trata de una relación de intercambio clásica.

Los documentos compuestos también son una tecnología clave para sistemas de cliente/servidor empresariales, así como para Internet e intranets. Los servidores pueden enviar planos frontales flexibles (una fachada) para sus servicios mediante el uso de documentos compuestos. Adicionalmente, los documentos compuestos pueden servir para empaquetar y embarcar en redes todo tipo de componentes autónomos, como agentes nómadas, componentes móviles y flujo de trabajo. Llamamos a esta tecnología *ubicaciones embarcables*. Así que bienvenido a este desafiante nuevo mundo. En los dos capítulos siguientes expondremos lo referente a *OpenDoc* y *OLE*, los dos estándares *de facto* de los documentos compuestos. Hablaremos de las ubicaciones embarcables en la Octava parte.

Capítulo 25

El modelo de componentes de OpenDoc



OpenDoc representa una mayor amenaza para Microsoft que sus amigos del Departamento de Justicia. Es un furioso juez Stanley Sporkin generado en C++.

Editorial
MacWEEK
(Marzo de 1995)

En marzo de 1996, OMG adoptó a OpenDoc como base para su tecnología de documentos compuestos. Esto significa que CORBA cuenta ahora con una arquitectura consistente tanto para el cliente como para el servidor. Además, OpenDoc hace posible que clientes y servidores de CORBA intercambien componentes a través de contenedores de documentos móviles. OpenDoc permite a CORBA acceder por la puerta grande al sector de objetos móviles. Le permite asimismo estar presente en las partes tanto del cliente como del servidor de Internet e intranets. De este modo, la novedad es que CORBA ya no es sólo tecnología de middleware y del servidor; ahora también está activa en los terrenos del cliente. En este capítulo se le ofrece una visión "panorámica" de OpenDoc y su ubicación en el ámbito CORBA. Se explica también el modelo de documentos compuestos de OpenDoc.

máquina o a través de redes. Dado que se le incluye en todo tiempo de ejecución de OpenDoc, un desarrollador de partes puede tener acceso a cualquier servicio o a cualquier ORB compatible con CORBA. Esto abre literalmente un universo de posibilidades.

SOM es también una tecnología de empaquetamiento de componentes. Permite que desarrolladores de OpenDoc empaquen sus partes en formato binario y las emitan después como DLL. El soporte de herencia múltiple de SOM —tanto en implementación como en interfaz— hace posible que los desarrolladores deriven de las ya existentes nuevas partes de OpenDoc. Esto se logra subclasiificando una parte existente de OpenDoc por medio del IDL de CORBA, para reutilizar o anular después las implementaciones de métodos suministradas en los binarios de DLL. Además, SOM les permite a los desarrolladores añadir métodos a partes existentes sin impacto alguno en los programas que las usan.

Bento

Bento —debe su nombre a los platos japoneses con divisiones para diferentes alimentos— define un formato contenedor para usarse en archivos, flujos de redes, tableros de recortes, etcétera. Un contenedor de archivos de Bento permite a las aplicaciones almacenar y recuperar grupos de objetos en un solo archivo estructurado, junto con sus referencias —o vínculos— de otros objetos.

El formato de contenedor de Bento es neutral en cuanto a plataformas; puede almacenar todo tipo de datos. En un documento de Bento, cada objeto posee un identificador persistente que se desplaza junto con él de un sistema a otro. Bento también soporta referencias entre objetos de diferentes documentos. En caso de existir varias versiones de un documento, Bento almacena únicamente los cambios incrementales. Esto facilita el mantenimiento de diferentes versiones del mismo documento.

El diseño de Bento está optimizado para intercambios de documentos. Un contenedor de Bento es un portador excelente para el intercambio de documentos compuestos entre aplicaciones que corren en diferentes plataformas. Bento también puede servir para desplazar grupos de objetos con sus scripts anexos, lo que lo convierte en una muy adecuada tecnología básica para componentes móviles.

Transferencia uniforme de datos

Las API de almacenamiento de OpenDoc proporcionan *transferencia uniforme de datos* entre y dentro de aplicaciones. Las mismas invocaciones de métodos utilizadas para el almacenamiento copiar y pegar y enlaces. Los datos pueden ser representados en una amplia variedad de formatos. OpenDoc permite mover partes enteras y las partes contenidas en ellas en una sola operación. Además de transferencia de datos, los enlaces aportan llamadas de notificación para la regeneración dinámica de la información transferida.

Administración de documentos compuestos

La administración de documentos compuestos define los protocolos que permiten a las partes compartir un espacio visual y coordinar el uso de recursos compartidos como entradas de teclado, menús y el enfoque de selección. Las *partes* son los elementos fundamentales de OpenDoc. Todo documento cuenta con una parte de nivel máximo (o raíz) en la que se incrustan todas las demás partes. Una parte puede contener a otras. Los *marcos* son áreas de la pantalla que representan a una parte. Representan también la parte en las negociaciones de espacio durante el despliegue de un documento.

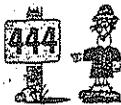
Las partes están asociadas con *editores de partes*. Éstos son elementos activos que manejan los datos de la parte e interactúan con el usuario. Un componente de OpenDoc es la combinación del editor de partes y sus datos. Usted puede seleccionar un editor en tiempo de ejecución para que trabaje con un tipo de parte en especial. OpenDoc alienta a los proveedores a ofrecer visualizadores de partes que puedan distribuirse libremente con un documento. El visualizador permite exhibir la parte sin alterar su contenido.

Arquitectura de creación de scripts abierta

La *arquitectura de creación de scripts abierta* (OSA: open scripting architecture) es una extensión de Apple Events de Macintosh. OSA permite que las partes expongan su contenido por medio de eventos semánticos que pueden ser invocados por cualquier lenguaje de generación de scripts observante de OSA. Los comandos emitidos a través de estos eventos semánticos operan en un *especificador de objetos*, el cual identifica en forma natural los objetos que el usuario ve en la pantalla (o dentro de algún otro contexto). Y puesto que se trata de objetos, los comandos son polimórficos. Por ejemplo, "siguiente" ("next") puede significar la siguiente celda o la siguiente palabra, dependiendo del tipo de parte que reciba la orden. Un especificador de objetos describe aquello que el usuario ve. Esta descripción es traducida por OSA en la referencia de objetos real.

Una parte susceptible de recibir un script debe ser preparada para mostrar en tiempo de ejecución la lista de objetos que contiene y las operaciones que soporta. OpenDoc puede suministrar mensajes de eventos del sistema de creación de scripts a las partes. La creación de scripts permite coordinar la interacción entre partes. OpenDoc también hace posible adjuntar scripts a eventos semánticos. Esto vuelve "maleable" una parte de OpenDoc, lo que significa que un script generado por el usuario puede ser activado al desencadenarse un evento semántico. Al interceptar el evento, el script puede modificar el comportamiento de la parte.

Además, las partes de OpenDoc pueden diseñarse para ser *grabables*. En este caso, el editor de partes intercepta toda recepción de acción, la convierte en un evento semántico y después se remite a sí mismo el evento, a lo cual se le llama un *cuello de botella*. Usted puede emplear cuellos de botella para revisar scripts adjuntos y registrar todos los eventos. Los eventos así registrados pueden ser convertidos al lenguaje de generación de scripts de su preferencia y reproducidos tiempo después.



sólo ofrece ComponentGlue en Windows y Mac. Sin embargo, esta tecnología forma parte de la arquitectura de OpenDoc y puede adaptarse a otras plataformas. En teoría, ComponentGlue podría permitir que usuarios visualizaran (aunque no editarán) objetos de OLE incrustados en un documento de OpenDoc en plataformas sin soporte nativo de OLE de Microsoft, como OS/2 y Unix. En la práctica, está por verse.

QUÉ HACE OPENDOC POR SISTEMAS DE CLIENTE/SERVIDOR

La tecnología de componentes de OpenDoc brinda nuevas oportunidades para la creación de sistemas de cliente/servidor. OpenDoc lleva el bus de CORBA al escritorio, con lo que hace de éste el centro de conexión de todas las comunicaciones entre programas. Se puede usar el mismo modelo de objetos para conectar tanto objetos empresariales como objetos visuales de grano mediano que residen en un escritorio común. Todo es sumamente consistente. Ahora es posible rempaquetar aplicaciones monolíticas de escritorio en partes capaces de conectar y usar (*plug-and-play*) mutuas en el mismo escritorio o a través de una red.

Cliente/servidor, estilo OpenDoc

Desde el punto de vista de cliente/servidor, un documento de OpenDoc actúa como un punto central de integración de múltiples fuentes de datos que residen en diferentes servidores (véase Figura 25-2). Las partes pueden vincularse con bases de datos empresariales, administradores de flujo de trabajo, depósitos de imágenes, correo electrónico o la hoja de cálculo local. El documento es la aplicación cliente/servidor. Funge como depósito de relación o "vínculos" de cliente/servidor con fuentes de datos externas y funciones remotas. Los usuarios finales pueden

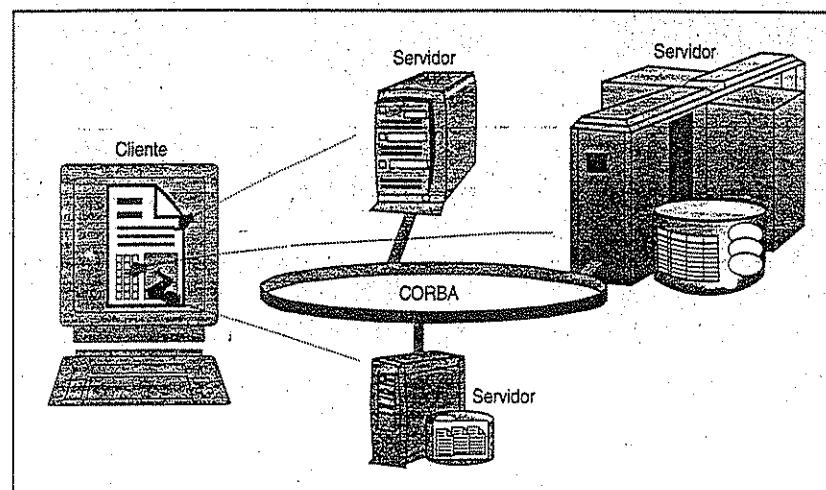
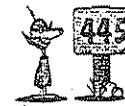


Figura 25-2. Un documento de OpenDoc como punto central de integración para vínculos de cliente/servidor.



ensamblar estas aplicaciones abriendo sencillamente un documento contenedor de OpenDoc y arrastrando partes hacia él. Pueden disponer de partes vivas en un documento visual tal como se hace ahora con programas de esquematización de páginas. La creación de una aplicación se reduce a labores de pegado y esquematización; no se requiere programación. Los usuarios de potencia, las compañías de IS y los integradores de sistemas pueden crear sistemas de cliente/servidor más sofisticados generando scripts de OpenDoc que orquesten complejas colaboraciones entre partes.

Es fácil imaginar otras alegorías para contenedores visuales en los que las partes puedan ser arrastradas, soltadas, reorganizadas y manipuladas en respuesta a las necesidades del usuario. Como ejemplo de contenedores de OpenDoc pueden citarse formatos de negocios, aviones, bases de datos en primer plano, planos, escritorios, jardines y cualquier otra representación visual que pueda ser útil como contenedor de partes. Anticipamos que los distribuidores de software ofrecerán "contenedores del diseñador" para complementar el ramo de partes; partes y contenedores trabajarán juntos e inseparablemente. Por ejemplo, el contenedor de una línea aérea podría estar ocupado por partes en representación de asientos, pasajeros, tripulación, equipaje, etcétera.

Los documentos vivos de OpenDoc pueden guardarse, embarcarse en redes de diferentes plataformas y volverse a abrir más tarde con los mismos vínculos de cliente/servidor. Recuerde que hablamos de documentos con multimedia vivientes, dotados de inteligencia de aplicación y vínculos con fuentes de datos externas. Usted puede mover estos documentos compuestos nombradas —a través de un administrador de flujo de trabajo— de un lugar a otro, en reflejo de un proceso de negocios dentro de y entre empresas.

En resumen, OpenDoc alentará nuevos niveles de manipulación directa en el escritorio. Los usuarios finales podrán crear aplicaciones personalizadas eligiendo un contenedor y ocupándolo con partes activas residentes en el escritorio o en servidores en cualquier punto de la red. Imagíñese la posibilidad de acceder a múltiples fuentes de datos y objetos de negocios a través de múltiples conexiones de cliente/servidor desde un solo contenedor o documento visual. Kurt Piersol, el arquitecto de OpenDoc de Apple Computers, llama a esto "todo un sistema de información en un documento".

Cómo mejora OpenDoc a CORBA

OpenDoc introduce mejoras de componentes en todos los aspectos actuales de CORBA, como:

- *Cuentas de referencias*. Los objetos de OpenDoc llevan un conteo de su uso. Esto contribuye a la liberación de memoria del sistema cuando los componentes dejan de usarse.
- *Series nombradas de extensiones*. OpenDoc permite que los componentes creen series nombradas de servicios llamadas *extensiones*. Un cliente puede pedirle a un componente que soporte una serie nombrada de funciones en particular y obtener una referencia de ella; las extensiones también son objetos. Estas extensiones hacen posible que grupos de partes



Preguntas por hacer a sus proveedores de OpenDoc

Debate

Para que usted tenga éxito con OpenDoc, los proveedores de sistemas y partes deben poner a su disposición una infraestructura de partes y las herramientas de aplicación adecuadas. He aquí algunas preguntas que convendría hacer al proveedor de OpenDoc de su elección:

- ¿Cómo manejará OpenDoc las partes entre plataformas? ¿Cómo se logrará la conversión común entre Windows, OS/2, Mac y variantes de Unix? ¿Qué lenguaje de creación de scripts debo usar entre estas plataformas?
- ¿De qué herramientas entre plataformas se dispone para la creación y ensamblado de partes? ¿Esas herramientas soportarán partes de OpenDoc en sus paletas? ¿Soportarán OCX de OLE? ¿Soportarán partición de componentes entre fronteras de cliente/servidor? ¿Soportarán componentes del servidor en ORB de CORBA?
- ¿Cómo se distribuirán, certificarán y mantendrán las partes? ¿Quién certificará series de partes, incluidas las de cliente/servidor? ¿A quién puedo recurrir en caso de que una parte se descomponga? ¿A quién debo recurrir cuando una serie de partes de proveedores múltiples no funcione de acuerdo con lo publicitado?
- A quién debo recurrir cuando un componente de OLE con envoltura de OpenDoc no funcione: a Microsoft o a CI Labs? □

Capítulo 26

OLE/DCOM: El otro bus de componentes



Dentro de muchos años, un Charles Darwin de la computación volverá la vista atrás y se preguntará cómo fue posible que las API de Windows de Microsoft hayan evolucionado hasta convertirse en un sistema operativo orientado a objetos.

Kraig Brockschmidt, autor de
Inside OLE 2, segunda edición
(Microsoft Press, 1995)

Si CORBA es el estándar de componentes distribuidos líder de la industria, entonces la *vinculación e incrustación de objetos* (*OLE: object linking and embedding*) de Microsoft es el "otro estándar" de facto. ¿Por qué es tan importante OLE? Por Microsoft. Todo lo que hace y hará se basa en OLE. Windows mismo está metamorfoseándose en OLE. O, si se prefiere, OLE es el fundamento de Windows orientado a objetos. Para cuando se lance Cairo, todo Windows será OLE. Pero, ¿y esto qué tiene que ver con los objetos distribuidos? Quizá resulte sorpresivo, pero lo cierto es que OLE se apoya en un ORB llamado *modelo de objetos de componentes distribuidos* (*DCOM: distributed component object model*). Aunque por ahora DCOM es un ORB de máquina única, Microsoft planea trasladarlo —junto con la mayor parte del OLE actual— a NT 4.0 y Cairo. OLE y DCOM sentarán las bases para el futuro entorno de computación distribuida de Microsoft. Windows 95 participará como cliente.

que DCOM constituirá el bus y servicios de objetos subyacentes. Todos los servicios del sistema—including el nuevo sistema de archivos—se generarán como objetos de DCOM. El modelo de documentos compuestos será el escritorio de Windows. Contendrá objetos que el usuario verá y manipulará. Dado que todo se generará sobre DCOM, con cualquier cosa podrá extenderse el sistema operativo base o la interfaz del usuario de documentos compuestos. Por ejemplo, los OCX pueden extender la GUI de Windows, en tanto que componentes de DCOM de bajo nivel pueden extender el almacenamiento persistente de Windows.

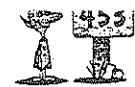
OLE: Interfaces en todas partes

OLE se compone de varias “interfaces”, así como de API estilo Win32. Las interfaces de OLE son similares en concepto a las de CORBA; definen un conjunto de funciones conexas. Una *interfaz* define un contrato entre componentes. Un componente de OLE puede soportar una o más interfaces. DCOM proporciona un protocolo de negociación para interfaces, que permite que los clientes adquieran apuntadores —en tiempo de ejecución— para las interfaces soportadas por un componente. Los programadores pueden generar sus propias interfaces por medio del *lenguaje de definición de interfaces (IDL: interface definition language)* de DCOM. Éste también ofrece un *lenguaje de descripción de objetos (ODL: object description language)* para la descripción de estas interfaces en una *biblioteca de tipos*. Éste es el equivalente en DCOM del depósito de interfaces de CORBA. Los clientes pueden descubrir dinámicamente qué interfaces soporta un objeto y qué parámetros requieren aquéllas. Como CORBA, DCOM cuenta con invocaciones de métodos tanto estáticas como dinámicas.

Como se indicará después, OLE consiste en alrededor de 100 interfaces, cada una de las cuales soporta en promedio unas seis funciones de miembros. Además, la biblioteca de OLE/DCOM proporciona alrededor de 120 API estilo Win32. Esto quiere decir que OLE introduce aproximadamente 720 llamadas de funciones nuevas (por encima de Win32). Algunas de estas interfaces son simplemente clases abstractas sin implementación. Sólo definen el contrato de interfaz. El implementador de componentes debe aportar el código que implementa la interfaz. Esto puede suponer un poquito de trabajo. Microsoft brinda herramientas para aligerar algo la carga. Además de las interfaces definidas por Microsoft, usted puede crear sus propias interfaces, llamadas en OLE *interfaces personalizadas*. También puede sustituir los componentes incluidos por Microsoft, siempre y cuando empate sus contratos de interfaz.

Entonces, ¿qué es un componente de OLE?

Un componente de OLE es definido por una *clase* que implementa una o más interfaces y una *fábrica de clases*, la interfaz que sabe cómo producir una instancia de componente de esa clase. A diferencia de las partes de OpenDoc, un componente de OLE no es una unidad autónoma predefinida. Todas las partes de OpenDoc poseen las mismas interfaces básicas, aunque también soportan extensiones. Por el contrario, un componente de OLE es un grupo de interfaces. Lograr que un componente de OLE alcance el mismo nivel de sofisticación de una parte de OpenDoc supone el concurso de muchas interfaces.



A fines de 1994, Microsoft introdujo *controles personalizados* (u OCX), los cuales poseen un conjunto predefinido de interfaces (a la manera de las partes de OpenDoc). Un OCX es un componente de grano mediano claramente definido, empaquetado como una parte de OpenDoc. Sin embargo, carece de algunas de las funciones de ésta. Por ejemplo, no permite la incrustación de otras partes, para lo que, en cambio, si están facultados los contenedores de OLE. De este modo, un OCX es un conjunto de contratos entre un componente visual de OLE y su contenedor. Por supuesto que éstos contratos tienen que ser implementados por el proveedor de OCX.

Como CORBA, OLE soporta componentes de todos los tamaños: de grano fino, mediano y grueso. Un ActiveX es un componente minimalista; un OCX es un componente de grano mediano. Los contenedores de OLE son componentes de grano grueso del tamaño de una aplicación.

TECNOLOGÍAS CONSTITUTIVAS DE OLE

Al igual que OpenDoc, OLE emplea la conocida alegoría de documentos para la organización visual de los componentes en el escritorio. También como OpenDoc, define las reglas de participación para que OCX y otros componentes visuales compartan la propiedad de la pantalla dentro de una sola ventana y almacenen sus datos en un solo archivo contenedor. Como OpenDoc, OLE suministra interfaces para que los componentes intercambien información con otros componentes vía vinculaciones, tablero de recortes y arrastrar y soltar. Y como OpenDoc, da soporte de automatización para que los componentes coordinen sus acciones por medio de scripts de automatización.

En la Figura 26-1 aparecen las tecnologías constitutivas de OLE y su relación entre sí. Adviértese que esta figura es muy semejante a la que se vio en el capítulo dedicado a OpenDoc, aunque los nombres de las tecnologías son distintos. El bus de objetos es DCOM en vez de SOM/CORBA. El almacenamiento estructurado es provisto por archivos compuestos en lugar de Bento. OLE posee su propio modelo para la automatización y creación de scripts. Incluso los modelos de transferencia uniforme de datos son diferentes. Y, como era de esperarse, OLE y OpenDoc tienen puntos muy distintos de diseño de documentos compuestos; OLE no soporta por ahora partes de forma irregular, y su edición *in situ* es limitada.

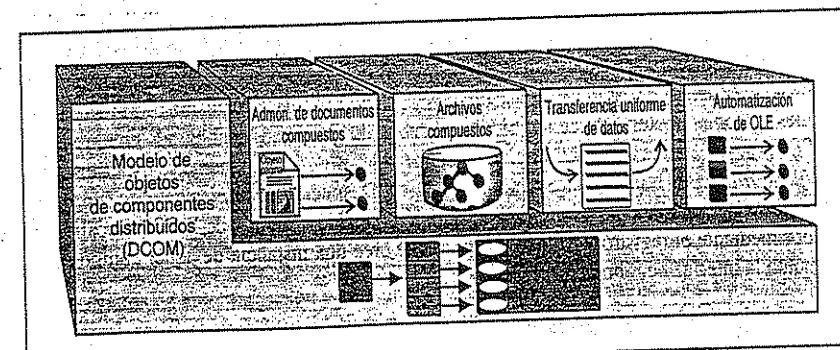


Figura 26-1. Tecnologías constitutivas de OLE.

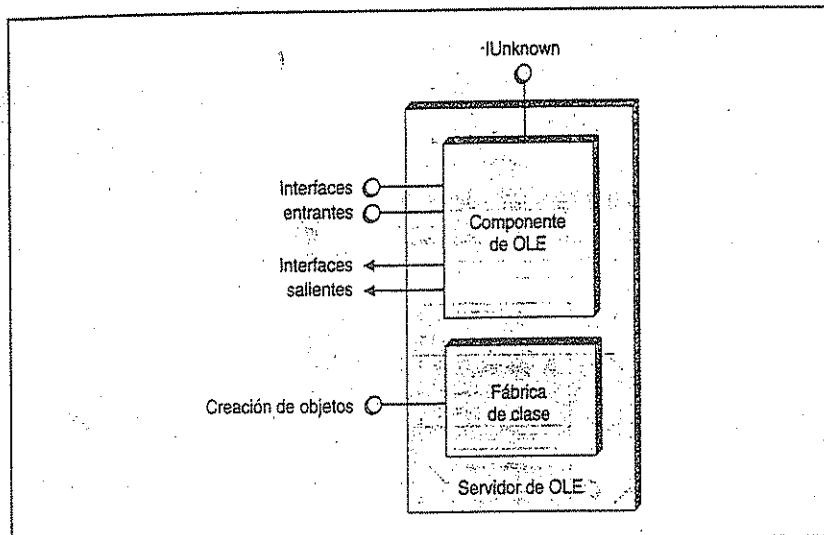


Figura 26-3. Anatomía de un componente de DCOM.

La automatización se basa en las facilidades de invocación dinámica de OLE, llamadas *interfaces despachables*. Al igual que las invocaciones de métodos dinámicas de CORBA, el despacho dinámico de OLE permite a un cliente invocar un método o manipular una propiedad por medio de un mecanismo de vinculación posterior. Un identificador de despacho le es transmitido a un método de *invocación*, el cual resuelve a qué método llamar en tiempo de ejecución.

Desde luego que los clientes de automatización deben descubrir en tiempo de ejecución qué interfaces proporciona un servidor de automatización. Esto incluye los métodos soportados por una interfaz y los tipos de parámetros requeridos por cada método. Incluye también las propiedades que un componente muestra a sus clientes. Éstos pueden obtener toda esta información en tiempo de ejecución en *bibliotecas de tipos*, el equivalente en OLE del depósito de interfaces de CORBA. ¿Cómo se introduce inicialmente esta información en bibliotecas de tipos? Se le crea con un archivo de lenguaje de definición de objetos (ODL) que describe las interfaces.

Transferencia uniforme de datos de OLE

Al igual que OpenDoc, OLE dispone de un mecanismo generalizado de transferencia de datos entre componentes que puede servir en las situaciones más diversas y entre una amplia variedad de medios. Los datos pueden intercambiarse mediante protocolos como el portapapeles, arrastrar y soltar, enlaces y documentos compuestos. Estos datos intercambiados pueden arrastrarse y posteriormente pegarse o soltarse en el mismo documento, un documento o una aplicación diferentes. La transferencia de datos real puede tener lugar sobre memoria compartida o con archivos de almacenamiento. Notificaciones asíncronas pueden enviarse a un cliente enlazado al modificarse los datos fuente. La transferencia uniforme de datos de OLE ofrece una interfaz única para transferencia de datos, la cual opera con múltiples protocolos.

Almacenamiento estructurado y servicios de persistencia de OLE

El sistema de almacenamiento estructurado de OLE consta de un sistema de archivos dentro de un archivo. A la implementación actual de esta arquitectura se le conoce como *archivos compuestos* (anteriormente *DocFiles*). Los archivos compuestos introducen una capa de indirección sobre sistemas de archivos existentes. Con Cairo se convertirán en el sistema de archivos como tal. Los archivos compuestos dividen un archivo en un grupo de *almacénamientos* (o directorios) y *flujos* (o valores de datos en bruto). Este sistema de directorio interno puede servir para organizar el contenido de un documento. OLE permite a los componentes controlar su propio almacenamiento en el documento compuesto. Los directorios describen los flujos; la estructura jerárquica facilita la navegación en el documento de objetos de OLE. Los archivos compuestos ofrecen facilidades de transacciones rudimentarias para la grabación de información en almacenes o la restauración de sus estados previos.

OLE también brinda un conjunto de interfaces para que un cliente se comunique con un objeto *persistent*. Estas interfaces definen las capacidades de un objeto persistente. En un extremo, un objeto persistente puede ignorarlo todo sobre el almacenamiento estructurado; sólo sabe almacenar y manipular su estado en un archivo regular. En el otro, un objeto sabe navegar por los almacenes de un archivo compuesto. En medio se encontrarían objetos capaces de manipular un solo flujo. En OLE, el cliente crea un objeto para manejarlo después como almacén persistente de información de estado. El objeto inicializa después su estado leyendo el objeto de almacenamiento. El cliente también puede pedirle al objeto que registre su estado de almacenamiento.

OLE incluye un servicio de nombramiento persistente, llamado *monikers*. Un *moniker* es un nombre inteligente que puede asociarse a datos persistentes. Todos los monikers tienen la misma interfaz, pero pueden poseer diferentes algoritmos de vinculación para el hallazgo de los datos asociados con un nombre. El algoritmo de vinculación es definido por el identificador de clase que implementa el moniker. Por ejemplo, los monikers pueden servir como alias (o sobrenombres) de un archivo remoto, un elemento dentro de un archivo o una consulta de SQL para la extracción de datos de una base de datos relacional.

Servicio de documentos compuestos de OLE

Cuando se piensa en OLE, quizás la primera idea que le viene a la mente son los documentos compuestos.

Chris Weiland y cols.,
Windows Tech Journal

El servicio de documentos compuestos de OLE define las interfaces entre una aplicación de *contenedores* y los componentes del *servidor* a los que controla. En este caso, un servidor es un componente visual que “atiende” a un contenedor. Las interfaces contenedor/servidor definen protocolos para la activación de servidores y la edición de su contenido “en el mismo lugar” dentro de la ventana del contenedor. Las aplicaciones de contenedores administran el almacenamiento y la ventana para la exhibición de los datos de un documento compuesto. Cada com-

Introducción a la Octava parte

El Web podría desatar la última fiebre del oro del milenio.

Tom Halfhill, director
BYTE Magazine
(Marzo de 1996)

En lo que se refiere específicamente a Internet, sabemos muy bien que está ocurriendo algo muy importante, pero creo que nadie esperó que fuera tan popular tan rápidamente.

Bill Gates, presidente
Microsoft
(Abril de 1996)

Todavía tienen fiebre del oro, marcianos? Esta parte es sobre cliente/servidor con el World Wide Web, que promete ser "la última fiebre del oro del milenio". Aunque no sea por otro motivo, será la última fiebre del oro de la que trataremos en este libro. Si es que aún no se han dado cuenta, el Web es en la actualidad el tema más candente en el planeta. No hay sitio que escape a él. Está en todas partes: en el cine, en las carteleras, en los periódicos, en los programas nocturnos de la televisión. Es la más reciente gran frontera tecnológica de nuestro planeta y nuestra mayor esperanza para el futuro. Es un mundo virtual, libre de hambre y enfermedades. Es la manera más rápida de hacer miles de millones de dólares.

Pero, ¿qué tiene que ver todo esto con cliente/servidor? Todo. Para los principiantes, el Web es la aplicación cliente/servidor más grande del mundo. Es la primera aplicación que lleva cliente/servidor hasta las masas. Y es la primera aplicación cliente/servidor de proporciones intergalácticas. Si, podrán interactuar con el avatar de Michael Jackson. ¿Qué es un avatar? Es la personalidad electrónica de cada quien en el Web. El Web está creando *habitats* habitados por avatares. Bienvenidos a los desafiantes nuevos *mundos virtuales*.

Pero el Web es más que diversión, juegos y avatares; también es un gran negocio. Los patrocinadores del Web—es decir, quienes pagan las golosinas gratis—son el mercado accionario y las mayores empresas de Estados Unidos. Estos patrocinadores ven intranets y comercio electrónico en el Web. Las *intranets* les permiten reconstruir sus redes corporativas privadas usando la tecnología de cliente/servidor del Web. El *comercio electrónico* está a punto de convertir al Web en el centro comercial más grande del mundo; es una nueva frontera para el comercio de toda clase. El mercado de valores ve en el Web una cosa: dinero. Así que esto mantiene las acciones al alza. Es una locura autogeneradora.

¿Cuándo estallará la burbuja? Jamás, si es que el Web cumple sus promesas. Por ahora, lo más que obtenemos son páginas Web "moderadas" y gran cantidad de alteraciones y ruido. Para convertirse realmente en el mayor centro comercial del mundo, el Web necesita apoyarse en un muy sólido fundamento de cliente/servidor. Ya explicaremos que la infraestructura del Web debe combinarse con objetos distribuidos para cumplir su destino intergaláctico. Los objetos nos permiten crear mundos virtuales más realistas, llamados *ubicaciones* *embarcables*. Las ubicaciones son contenedores de componentes que pueden enviarse del servidor a los clientes para crear una nueva generación de aplicaciones cliente/servidor móviles basadas en el Web.

Introducción a la Octava parte

La Octava parte trata del Web desde la perspectiva de cliente/servidor. Comenzaremos por el hipervinculado Web actual. Analizaremos la tecnología de cliente/servidor tras las páginas Web y los vínculos de hipertexto. Después nos referiremos a las extensiones en uso para volver al Web más interactivo; esto es, para hacerlo comportarse más a la manera típica de un sistema de cliente/servidor. Estas extensiones incluyen formatos y tablas de HTML en la parte del cliente y el protocolo CGI en la parte del servidor. CGI está usándose para conectar al Web todo tipo de servidores, pero en realidad no pasa de ser un remedio casero; no representa una solución a largo plazo. Abordaremos también las extensiones de seguridad para el Web, como muros de protección, certificados y protocolos protegidos como SSL y S-HTTP. Estas siglas le serán perfectamente comprensibles cuando termine de leer esta parte.

El corazón de la Octava parte es el Web de objetos (*Object Web*). Creemos que el Web está acercándose a los objetos en forma excelente. Java es sólo el primer paso en esta dirección. Los capítulos sobre Object Web abarcarán el tema de Java a profundidad. Luego veremos en qué forma los ORB—como CORBA y Network OLE—extienden a Java en lo que se refiere a comunicaciones entre objetos. Nos ocuparemos también del uso que se les dará a los documentos compuestos como contenedores de componentes del Web que puedan ser directamente manipulados "en el mismo lugar". Estos contenedores sirven asimismo para embarcar ubicaciones (*o mundos virtuales*) de los servidores a los clientes. El Object Web es una arquitectura cliente/servidor en 3 planos que combina lo mejor de ambos mundos: ORB y el Web. El Object Web también será una gran incubadora de componentes formados con Java, OLE y OpenDoc, lo más parecido al paraíso de los componentes que pueda imaginarse.

Concluiremos la Octava parte con una panorámica de mercado y productos. Es la guía al oro. Pero antes de iniciar este viaje, es importante entender que la vida en Internet se mide en "años animales", no en años humanos. Esto significa que la tecnología de Internet avanza muy rápidamente. Parecería que Internet se transforma cada dos años. En la actualidad iniciamos la fase de Java de Internet, en dirección al Object Web. Así que abróchense bien los cinturones de seguridad, porque éste será un trayecto muy acelerado con muchos retumbos. Esperamos que se diviertan.



A fines de 1993, el visualizador Web gráfico Mosaic introdujo el primer entorno de aplicación cliente/servidor verdadero por encima de Internet. Así fue el nacimiento de cliente/servidor estilo Web. Este nuevo modelo de cliente/servidor consiste en clientes ligeros, portátiles y "universales" en comunicación con servidores superamplios o superobesos. Cliente/servidor en el Web hace algunas cosas a la perfección; por ejemplo, publicación electrónica, tableros personales, compartimiento de documentos y grupos de conversación. En 1995, la tecnología del Web se amplió para soportar aplicaciones cliente/servidor ligeramente más interactivas, mediante el uso de formatos HTML y del protocolo del servidor CGI. Además, Netscape y otros introdujeron nuevos protocolos para hacer del Web un ámbito más seguro. Ejemplos de estos nuevos protocolos son la capa de sockets de seguridad (SSL: *secure sockets layer*), *HTTP seguro* (*S-HTTP: secure HTTP*) y todo tipo de muros de protección.

El Web que conocemos ahora no es todavía la idílica carretera de la información, "la madre de todos los sistemas de cliente/servidor". Para alcanzar ese preciado objetivo, es necesario el abaratamiento de un ancho de banda abundante y el sacroso enlace entre el Web y la tecnología de objetos distribuidos. Java es el primer paso en esta dirección. En la Figura 27-1 aparece la progresión de tecnologías del Web. En 1994, el Web era sobre todo un colosal —y muy de moda— medio unidireccional para la publicación y difusión de documentos electrónicos. A fines de 1995, se había convertido ya en una plataforma de aplicaciones cliente/servidor más interactiva; la *interfaz de gateway común* (*CGI: common gateway interface*) se usa ahora para acceder a todos los entornos de servidor conocidos.

En 1996, el Web descubrió finalmente los objetos. Java es el primer paso hacia la creación de un *Web de objetos* de cliente/servidor.

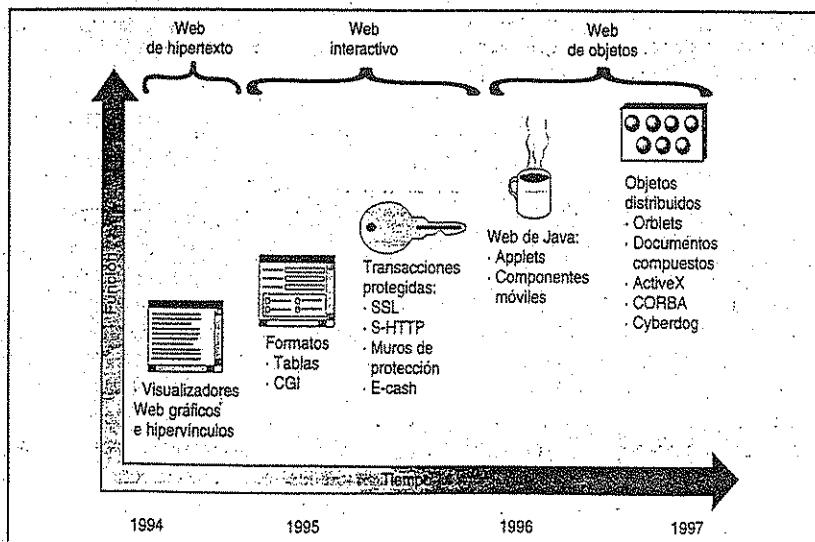


Figura 27-1. Evolución de las tecnologías del Web.



¿Podrá haber algo después de Java? Quizá le sorprenda saberlo, pero la respuesta es sí. Java es un paso necesario pero aún insuficiente hacia la creación de un Web de objetos. Para el siguiente paso se requiere más cafeína de objetos. Nos referimos a la combinación de Java con la infraestructura de objetos distribuidos; su taza de Java incluirá muy pronto ORBlets de CORBA, COMlets de OLE y estructuras de documentos compuestos como OLE y OpenDoc. Confiamos en que le guste el café cargado.

¿Hay algo que aún no se haya dicho sobre Internet? ¿Es posible que a la prensa, periódicos, diarios locales y revistas nacionales de computación se les haya escapado algo? Creemos que sí, pues de otro modo no nos lanzaríamos a un recorrido de cinco capítulos sobre el Web desde la perspectiva de cliente/servidor.

Este capítulo es el inicio del recorrido en la era de hipertexto del Web; en él presentaremos las tecnologías de cliente/servidor básicas en las que se apoya el Web, como HTML, HTTP y los visualizadores Web. En el capítulo 28, "Cliente/servidor en el Web: La era interactiva", introduciremos las tecnologías interactivas del Web, como formatos, CGI y protocolos de seguridad necesarios para el comercio electrónico basado en el Web. El capítulo 29, "Cliente/servidor en el Web: La era de objetos de Java", tratará del Web de Java. El capítulo 30, "Cliente/servidor en el Web: La era de objetos distribuidos", del Web de objetos más allá de Java; es decir, de la unión de Java con tecnologías de componentes distribuidos como CORBA, OLE y OpenDoc. Finalmente, en el capítulo 31, "Cliente/servidor en el Web: Presentación de participantes", concluirá nuestro paseo, con una visión panorámica del mercado de cliente/servidor del Web.

CLIENTE/SERVIDOR, ESTILO WEB

Internet es el mayor experimento mundial en anarquía.

Eric Schmidt, director técnico, Sun
(Octubre de 1995)

El Web construye el orden de cliente/servidor sobre lo que Eric Schmidt, de Sun, llama "el mayor experimento mundial en anarquía". En esta primera encarnación, sumamente popular, el Web es simplemente un sistema de hipertexto global. *Hipertexto* es un mecanismo de software que enlaza documentos con otros documentos afines en la misma máquina o a lo largo de redes. El documento enlazado puede contener en sí mismo enlaces con otros documentos, lo que puede extenderse hasta la eternidad. Un enlace puede apuntar también a otros recursos externos, como archivos de imágenes, sonido o programas ejecutables. A la larga, el Web podría vincular entre sí a todos los documentos producidos en este planeta.

Lo fantástico del Web es su simplicidad. El modelo de cliente/servidor del Web alcanza su escala intergaláctica mediante el uso de protocolos sumamente exportables sobre TCP/IP. La exportabilidad, independencia respecto de plataformas e independencia respecto de contenido se reafirman en cada nivel; son los mantras de los arquitectos del Web. Pero, ¿qué hace que cientos de miles de servidores distribuidos se comporten como una sola aplicación? Esta magia es resultado de la introducción de cuatro nuevas tecnologías sobre la infraestructura existente



pales protocolos y piezas de cliente/servidor. Con eso le bastará para volverse extraordinariamente peligroso.

Su primera interacción de cliente/servidor en el Web

El modelo que vemos emerge es un cliente universal, capaz de navegar en redes locales o en Internet y de recurrir a cualquier aplicación en cualquier momento.

Marc Andreessen, vicepresidente
Netscape
(Septiembre de 1995)

En la Figura 27-2 se muestra la operación conjunta de las piezas de cliente y servidor en el Web:

1. *Usted selecciona un URL destino.* Una interacción de cliente/servidor en el Web comienza cuando usted especifica un URL destino desde su visualizador Web. Procede a ello ya sea haciendo clic en un vínculo de hipertexto, seleccionando un URL en una lista o tecleando explícitamente en el URL (medida habitualmente último recurso).
2. *El visualizador envía una solicitud de HTTP al servidor.* El visualizador toma el URL especificado por usted, lo inserta en una solicitud de HTTP y la envía al servidor destino.
3. *El servidor entra en acción y procesa la solicitud.* En la parte receptora, el servidor HTTP gira en un lazo, a la espera del arribo de solicitudes a su puerto conocido (en el caso de HTTP, el puerto predeterminado es 80). La recepción de la solicitud provoca el establecimiento de una conexión de sócket entre el cliente y el servidor. El servidor recibe el mensaje del cliente, localiza el archivo HTML solicitado, se lo hace llegar al cliente junto con cierta información de estado y finalmente cancela la conexión.
4. *El visualizador interpreta los comandos de HTML y exhibe el contenido de la página.* El visualizador exhibe un indicador de estado mientras aguarda la recepción del URL solicitado. Cuando finalmente recibe el URL, identifica su tipo. Si se trata de un archivo HTML, interpreta las etiquetas y exhibe el contenido en su ventana. De no ser así, invoca una aplicación ayudante asociada con un tipo particular de recurso y le cede el archivo de retorno. El ayudante presenta el contenido en su propia ventana o en un área convenida dentro de la ventana del visualizador. Por ejemplo, la mayoría de los visualizadores no saben cómo proceder con un archivo de video, de manera que lo ceden a una reproductora de video —el ayudante en este caso— para que se encargue de la proyección de la película en una ventana distinta.

Esta interacción primitiva de cliente/servidor representa más del 90% de las transacciones actuales en el Web. En el siguiente capítulo abordaremos los protocolos de CGI y los formatos de los visualizadores y explicaremos de qué modo extienden el modelo básico. Pero antes debemos exponer los detalles de los protocolos de URL, HTTP y HTML. Es preciso conocer el funcionamiento de estos protocolos para comprender el CGI, los formatos, la seguridad de Internet y los objetos del Web.

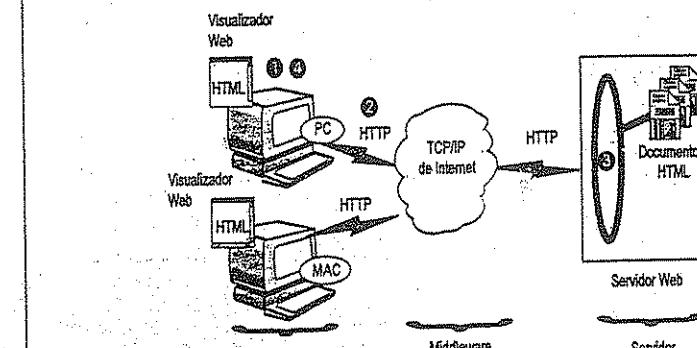


Figura 27-2. Interacción de cliente/servidor en el Web.

ENTONCES, ¿QUÉ ES UN URL EXACTAMENTE?

A la información le gusta ser libre: libre de plataformas, destinos de salida, instrucciones de formateo, formatos de archivos propietarios y ubicaciones físicas.

Art Fuller, Data Based Advisor
(Junio de 1995)

Un URL ofrece un esquema de nombramiento de propósito general para la especificación de recursos de Internet con el uso de una cadena de caracteres ASCII imprimibles. Estos caracteres imprimibles permiten enviar URL en mensajes de correo, imprimirlas en tarjetas de presentación o exhibirlas en carteles. Un URL suele componerse de cuatro partes (véase Figura 27-3):

- El *esquema de protocolo* le indica al visualizador Web qué protocolo de Internet emplear al accesar a un recurso en un servidor. Además de HTTP, URL soporta todos los protocolos principales de Internet, como Gopher, FTP, News, Mailto y WAIS. HTTP es el protocolo nativo del Web; está dirigido a páginas y programas servidores Web. Gopher es un precursor del Web; exhibe información sobre servidores en forma de jerarquía de menús. FTP es el protocolo más antiguo de Internet para la transferencia de archivos. News es un protocolo de grupos de interés que permite especificar un grupo de información o artículo. Mailto le permite enviar correspondencia a una dirección designada de correo electrónico. WAIS es el medio para especificar el nombre de dominio de una base de datos destino por buscar, así como una lista de criterios de búsqueda. Por supuesto que su elección específica de un esquema de protocolo influirá directamente en la interpretación de la información sobre la ruta o vía de acceso en el URL.

Estructura general de un documento HTML

Lo único que le interesa a la gente por ahora es conectarse fácilmente. Cuando lo logre, se dará cuenta de que lo que importa es el contenido.

Christine Comaford, columnista
PC Week
(Marzo de 1996)

En la figura 27-5 aparece la estructura de un documento HTML. La razón básica de esta estructura es permitirle al visualizador la comprensión de la organización de un documento. Un documento HTML correctamente estructurado empieza con <HTML> y termina con </HTML>. Además, todo documento HTML debe contar con una sección de encabezado en la parte superior, encerrada por las etiquetas <HEAD> y </HEAD>, y con una sección de cuerpo de texto, encerrada por las etiquetas <BODY> y </BODY>. El encabezado contiene información en la que se describe el contenido del cuerpo, su título, URL y si el documento es localizable. Un visualizador Web exhibe este elemento de texto marcado por las etiquetas <TITLE> y </TITLE> en la barra de título de su ventana. Este texto es también el que describe al documento en una lista de consulta del usuario (o lista de *marcadores*).

Las etiquetas <BODY> y </BODY> contienen la porción del documento que usted ve exhibida dentro del área del cliente de la ventana de un visualizador Web. Ahí es donde usted visualiza el contenido de un documento HTML. El HTML le permite estructurar aún más el cuerpo de un documento usando una jerarquía de subencabezados que pueden anidarse hasta en seis niveles. Usted especifica los subencabezados en orden descendente, por medio de las etiquetas <H1> a <H6>. Cuando un visualizador Web se topa con una etiqueta de subencabezado, termina el párrafo en curso y exhibe el texto del subencabezado alineado a la izquierda y con una fuente tipográfica distinta. En la Figura 27-6 se muestra una presentación en pantalla de una versión particular del visualizador Web del HTML de la Figura 27-5.

```
<HTML>
<HEAD>
<TITLE>Mi documento</TITLE>
</HEAD>
<BODY>
<H1>Este es un subencabezado H1</H1>
<H2>Este es un subencabezado H2</H2>
<H3>Este es un subencabezado H3</H3>
<H4>Este es un subencabezado H4</H4>
<H5>Este es un subencabezado H5</H5>
<H6>Este es un subencabezado H6</H6>
</BODY>
```

Figura 27-5. HTML para "Mi documento".

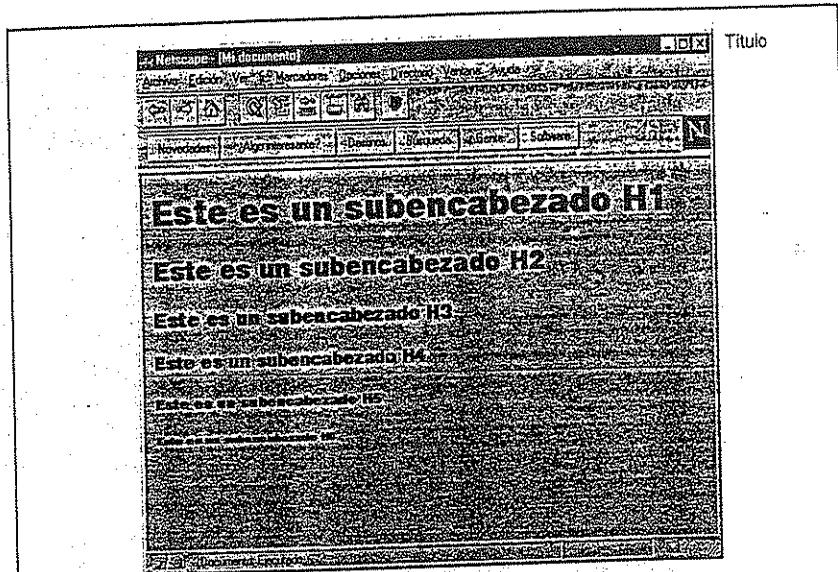


Figura 27-6. Presentación de "Mi documento" en HTML por un visualizador Web.

Cómo estructurar el flujo de texto en un documento HTML

Por lo común, el texto de un documento se divide en párrafos bajo diversos subencabezados. Para indicar un nuevo párrafo se usa la etiqueta <P>; para trazar una línea horizontal, la etiqueta <HR>, y para introducir un bloque de texto en estilo de formato previo —como el listado de un programa o una tabla—, la etiqueta <PRE>. Cabe señalar que <P> y <HR> son etiquetas únicas; en su caso no se requiere de los pares </P> y </HR>.

La Figura 27-7 es un ejemplo de documento HTML con las etiquetas <P>, <HR> y <PRE>. En la Figura 27-8 se muestra la presentación de ese documento por un visualizador Web usual. Nótese que todo el texto entre etiquetas <P> es reorganizado por el visualizador Web para adecuarlo al tamaño de la ventana, las fuentes tipográficas y la forma general del documento. La mayoría de los visualizadores Web aplican un espacio en blanco equivalente a una línea para separar entre si los párrafos. La línea horizontal se traza dejando cierto espacio en blanco arriba y abajo de ella. Es conveniente usar líneas horizontales tanto como sea posible. Contribuyen a la creación de una apariencia uniforme y su transferencia por la red es más eficiente que un mapa de bits o una línea de caracteres subrayados.

Finalmente, adviértase que el texto marcado por las etiquetas <PRE> y </PRE> no es muy atractivo, pero en él se respeta la diagramación que especificamos en el HTML. Básicamente, lo que hicimos fue indicarle al visualizador que el texto <PRE> está fuera de sus límites. En consecuencia, el visualizador no puede reorganizarlo, ajustarlo ni emplear una atractiva fuente tipográfica proporcionada para exhibirlo. En cambio, presenta el texto "tal como está", usando una fuente no proporcionada para mantener el espacio en blanco, los tabuladores, los inicios de

alineación predeterminada es BOTTOM, lo que significa que la imagen será alineada de acuerdo con el extremo inferior del texto. En la Figura 27-10 se observa la versión "normal" de la página de Objetos distribuidos. Todo lo que hicimos fue insertar después de la etiqueta <P>. De este modo le indicamos al visualizador Web que colocara el texto junto a la imagen de Zog, a la izquierda.



Figura 27-10. Presentación de una imagen de Zog por un visualizador Web.

Hipervínculos

En nuestra opinión, lo que realmente hizo que el Web se difundiera por todas partes fueron los hipervínculos. Estos ofrecen los ganchos que permiten navegar transparentemente de un servidor a otro con sólo hacer clic con el ratón. Un documento se hipervincula con otros documentos u otras ubicaciones en el mismo documento por medio de un par de etiquetas de *ancla*, de esta manera: . Éste es un vínculo importante .

Entre las etiquetas <A> y se inserta el texto sobre el que el usuario podrá hacer clic para pasar a la página vinculada. Casi todos los visualizadores resaltan y subrayan el texto para destacarlo. Además de texto, también se puede colocar una etiqueta para vincular una imagen. Los vínculos visuales "normales" combinan texto con gráficos.

El HREF —o atributo de referencia— especifica el documento de destino. El URL puede especificarse en una de tres maneras: 1) *absoluto*, lo que significa que contiene el nombre de anfitrión y el nombre de archivo del documento destino; 2) *relativo*, lo que significa que el nombre de anfitrión de destino y el directorio de inicio para la ruta son los mismos que los del documento que contiene la etiqueta de ancla, y 3) *local*, lo que quiere decir que el archivo reside en la máquina cliente, no en el servidor Web.



Los vínculos de HTML también permiten señalar a otras anclas en un archivo HTML. Esto significa que se pueden crear vínculos no sólo con un archivo, sino también con un punto específico de un archivo. Para hacerlo, se debe añadir el nombre del ancla después del nombre de archivo, separándolo con el signo de número "#". Por ejemplo, la etiqueta lo trasladará a "mi marcador". Por supuesto que antes debe haber creado "mi marcador" en el archivo de destino usando una etiqueta de ancla con un atributo NAME. Por ejemplo, Texto. En la Figura 27-11 aparece un vínculo de este tipo.

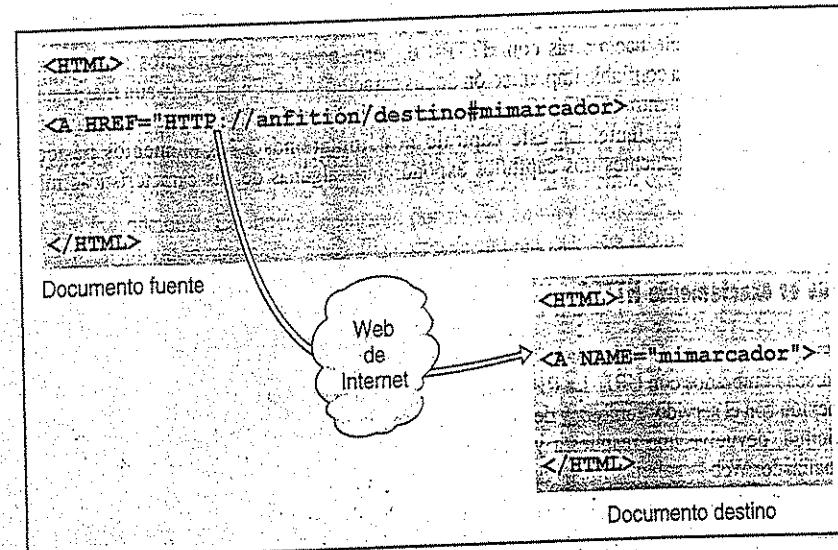


Figura 27-11. Hipervinculación con un punto ancla dentro de un documento destino.

Versiones de HTML

Existen varias versiones diferentes de HTML, y es necesario que sepa un poco acerca de ellas. HTML ha sido definido por un grupo de trabajo de la *Internet Engineering Task Force (IETF)*, así como por un consorcio de la industria, llamado W3C.¹ Existen varias versiones de HTML. La primera versión —conocida como HTML 1.0— ha circulado en el mercado desde hace un par de años. El estándar actual es HTML 2.0, que ofrece mejoras menores sobre HTML 1.0. HTML 3.0 (originalmente conocido como HTML+) ofrece mejoras sustanciales sobre HTML 2.0. Sin embargo, de acuerdo con W3C, pasará todavía algo de tiempo antes de que dispongamos de un nuevo estándar HTML 3.0, o incluso de HTML 2.1. Mientras tanto, W3C emitirá estándares fragmentarios para diferentes y nuevas características, como tablas, applets, marcos y objetos activos.

¹ El World Wide Web Consortium (W3C) está encabezado por Tim Berners-Lee. Lo integran más de 100 compañías proveedoras de Web. Su propósito es acelerar la producción de estándares del Web.

- La *línea de solicitud* se compone de tres campos de texto, separados por espacios en blanco. El primer campo especifica el método —o comando— por aplicar al recurso de un servidor. El método más común es GET, por medio del cual se le pide al servidor enviar una copia del recurso al cliente. En el siguiente recuadro de detalles se explican los demás métodos de HTTP. El segundo campo especifica el nombre del recurso destino; es el URL sin el protocolo ni el nombre de dominio del servidor. El tercer campo identifica la versión del protocolo usada por el cliente; por ejemplo, HTTP/1.0.
- Los *campos de encabezamiento de solicitud* ofrecen información adicional sobre la solicitud, y sobre el cliente mismo, al servidor. Actúan como parámetros de RPC. Cada campo de encabezado consiste en un nombre, seguido por dos puntos (:) y el valor del campo. El orden en que se transmiten los campos de encabezado no es significativo.
- El *cuerpo de entidad* es empleado en ocasiones por los clientes para transmitirle al servidor información de masa.

En la sección inferior de la Figura 27-12 se incluye un ejemplo de una solicitud GET de HTTP arquetípica; el cliente solicita un archivo llamado "archivo.html" del servidor. El primer campo de encabezado *accept* le indica al servidor que el cliente sabe cómo manejar archivos de texto de HTML. El segundo campo de aceptación le indica al servidor que el cliente también puede manejar todos los formatos de audio. Finalmente, el campo *user-agent* le hace saber al servidor que el cliente es un visualizador MacWeb. Así, las diversas "aceptaciones" le señalan al servidor qué tipos de datos puede manipular el cliente; el campo de *user-agent* indica el nombre de implantación del cliente. Listaremos todos los campos de encabezado de HTTP en el siguiente recuadro de detalles.

HTTP en un vistazo

Detalles:

Basta analizar detenidamente la especificación de HTTP para descubrir la gran cantidad de cosas que no se han implantado aún. El actual surtido de servidores en realidad no le hace justicia.

John Labovitz, O'Reilly

Métodos de HTTP

Como se advierte en la Tabla 27-1, HTTP aún no está completo. Está evolucionando para convertirse en un protocolo de cliente/servidor más maduro, al menos en lo que se refiere a los métodos que soporta. HTTP/1.1 soporta 13 métodos, contra 3 de HTTP/1.0. Los métodos más ampliamente implantados hasta ahora siguen siendo GET y POST. Ya nos hemos detenido en GET en este capítulo. Trataremos de POST en el capítulo siguiente.

Tabla 27-1. Un vistazo a los métodos de HTTP.

Método	HTTP/1.0	HTTP/1.1	Descripción del método
GET	S	S	Recuperar el URL especificado.
HEAD	S	S	Idéntico a GET, salvo que el servidor no envía el documento en respuesta; sólo envía los encabezados. Los clientes lo usan para obtener metadatos de recursos o para probar la validez de vínculos de hipertexto.
POST	S	S	Enviar estos datos al URL especificado.
PUT	N	S	Almacenar estos datos en el URL especificado, en remplazo del contenido anterior.
PATCH	N	S	Similar a PUT, salvo que contiene una lista de diferencias entre la versión original del URL y el contenido deseado tras la aplicación del método.
COPY	N	S	Copiar el recurso identificado por el URL en la(s) ubicación(es) especificada(s).
MOVE	N	S	Trasladar el recurso indicado por el URL a la(s) ubicación(es) especificada(s). Este método es equivalente a COPY/DELETE (copiar/borrar).
DELETE	N	S	Borrar el recurso identificado por el URL.
LINK	N	S	Establecer una o más relaciones de vinculación entre el recurso identificado por el URL y otros recursos.
UNLINK	N	S	Eliminar una o más relaciones de vinculación en el URL especificado.
TRACE	N	S	Notificar todo lo que se reciba del cliente en el cuerpo de entidad de la respuesta.
OPTIONS	N	S	Solicita información sobre las opciones de comunicación disponibles en la cadena de solicitud/respuesta para el URL especificado. Permite a los clientes determinar las capacidades de un servidor sin recuperar un recurso.
WRAPPED	N	S	Permite que solicitudes se envuelvan en conjunto y quizás también se codifiquen para reforzar la seguridad y/o privacidad de la solicitud. El servidor de destino debe desenvolver el mensaje y cederlo al manipulador apropiado.

Encabezados de entidad de HTTP

Tabla 27-5. Encabezados de entidad de HTTP.

Encabezado	HTTP/1.0	HTTP/1.1	Descripción
Allow	S	S	Lista los métodos soportados por el recurso de URL.
Content-Encoding	S	S	Especifica codificación de respuesta, como compresión y compresión con zip.
Content-Language	N	S	Especifica lenguaje natural de respuesta (por ejemplo, francés).
Content-Length	S	S	Extensión en bytes del cuerpo de entidad.
Content-Type	S	S	Tipo de contenido MIME de respuesta.
Content-Version	N	S	Contiene un número de versión del recurso.
Derived-From	N	S	Identifica la versión anterior del recurso.
Expires	S	S	Contiene fecha/hora después de la cual el documento caduca.
Last-Modified	S	S	Contiene fecha/hora de la modificación más reciente del recurso.
Link	N	S	Contiene información de vinculación del documento.
Title	N	S	Contiene el título del documento.
Transfer-Encoding	N	S	Identifica una transformación aplicada al documento (o cuerpo del mensaje).
URL-Header	N	S	Contiene la parte del nombre del recurso del URL.

¿Cómo es una respuesta de HTTP?

En la Figura 27-13 se muestra la sintaxis de una respuesta de HTTP. Ésta consiste en una *línea de encabezado de respuesta*, uno o más *campos de encabezado de respuesta* opcionales y un *cuerpo de entidad* opcional. Las líneas se separan por medio de un carro de retorno/avance de línea (crlf). El cuerpo de entidad debe ir precedido por un espacio en blanco. He aquí los detalles:

- La *línea de encabezado de respuesta* envía la versión de HTTP, el estado de la respuesta y una explicación del estado de devolución.
- Los *campos de encabezado de respuesta* envían información en la que se describen los atributos del servidor y el documento HTML enviado al cliente. Cada campo de encabezado

se compone de un nombre, seguido por dos puntos (:) y el valor del campo. El orden en que el servidor remite los campos de encabezado no es importante.

- El *cuerpo de entidad* contiene, por lo general, un documento HTML que un cliente haya solicitado.

En la sección inferior de la Figura 27-13 aparece una respuesta usual del servidor a una solicitud GET. El código de resultado 200 indica que la solicitud fue exitosa. El campo de encabezado *server* (servidor) identifica al servidor como un NCSA/1.3. El campo *MIME-version* (versión de MIME) indica que el servidor soporta MIME 1.0 (véase el siguiente recuadro de detalles). El campo *content-type* (tipo de contenido) describe el objeto remitido como un documento de texto de HTML. El campo *content-length* (extensión de contenido) indica la extensión del contenido. A esto le sigue el propio documento HTML, objeto de la solicitud. El servidor envía los datos solicitados y después cancela la conexión TCP/IP. Esto es todo. ¡Felicidades! Ya es usted todo un experto en HTTP.

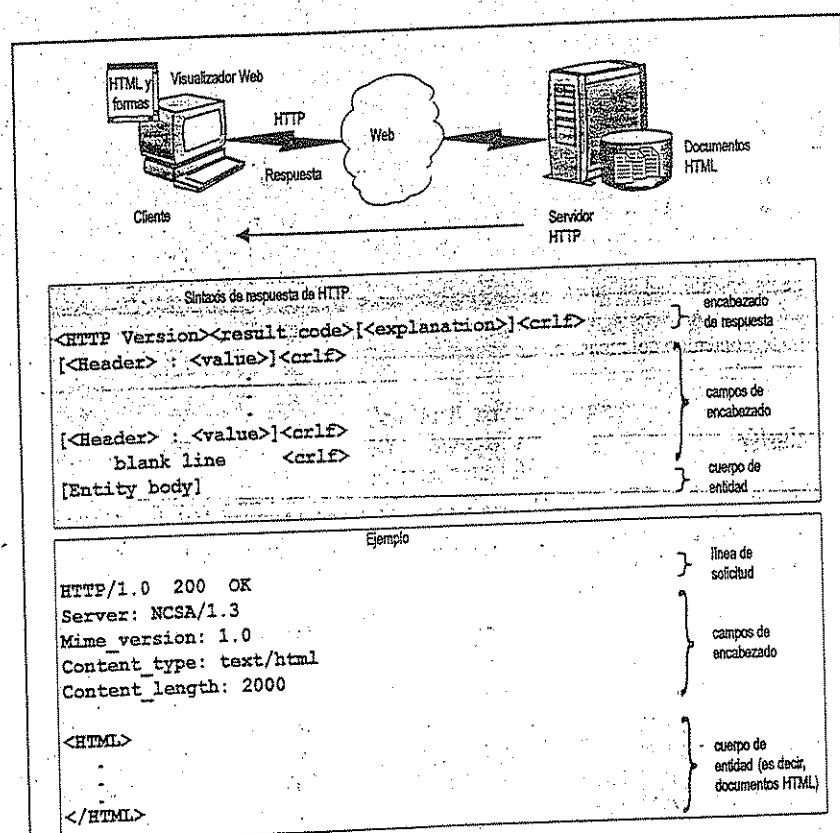


Figura 27-13. Formato de respuesta de HTTP.

CLIENTE/SERVIDOR EN 3 PLANOS, ESTILO WEB

Navigator es más una terminal 3270 que otra cosa, con la pequeña diferencia de que no sólo sirve para acceder a una macrocomputadora.

Marc Andreessen, vicepresidente
Netscape
(Septiembre de 1995)

Los visualizadores Web son las versiones modernas de las terminales 3270 del pasado. Sí, estamos regresando al futuro. Pero, ¿dónde están los "formatos de llenado" de 3270? Se trata de los equivalentes en computación de los formatos en papel que nos hemos pasado llenando toda la vida. Pues resulta que el Web es abundante en formatos. Están en todas partes. En caso de que aún no los haya visto, un *formato de Web* es una página HTML con uno o más campos de introducción de datos y un botón "Submit" ("poner a consideración") obligatorio. Se hace clic en este botón para enviar el contenido de datos del formato a un servidor Web. Esto da lugar a que el visualizador reúna todas las entradas del formato, las introduzca en un mensaje de HTTP e invoque después ya sea un método GET o POST de HTTP en la parte del servidor.

En el extremo receptor, el servidor Web usual no sabe qué hacer con un formato; no es un documento HTML ordinario. Así, sencillamente le da la vuelta e invoca al programa o recurso nombrado en el URL para decirle que se haga cargo de la solicitud. El servidor transmite la solicitud de método y sus parámetros al programa posterior por medio de un protocolo llamado *interfaz de gateway común (CGI: common gateway interface)*.

El programa en segundo plano ejecuta la solicitud y remite los resultados en formato HTML al servidor Web, haciendo uso del protocolo de CGI. El servidor Web trata a los resultados como un documento normal, que remite al cliente. Así, en cierto sentido el servidor Web funge como conductor entre el cliente Web y un programa posterior, que es el que efectivamente lleva a cabo el trabajo. En la Figura 28-1 se presentan los elementos de esta nueva arquitectura cliente/servidor en 3 planos, estilo Web. El primer plano corresponde a un visualizador Web, que soporta formatos interactivos; el segundo, a un servidor HTTP cualquiera complementado con programas CGI, mientras que el tercero se compone de los tradicionales servidores de segundo plano.

Además de formas, Netscape fue pionero en el concepto de *tablas* de HTML. Éstas son nuevas etiquetas que permiten a un visualizador exhibir información de filas múltiples —resultados de consulta recibidos de un servidor, por ejemplo— dentro de un accesorio de tablas gráficas. Muchos de los visualizadores más conocidos soportan ya las etiquetas de tablas de Netscape. Tal como veremos, W3C y IEFT publicaron un documento preliminar en el que se especifican las extensiones de tablas de HTML 2.0. Afortunadamente, este documento es compatible hacia atrás con las extensiones de Netscape.

La tecnología de CGI hace posible que los clientes de Internet actualicen bases de datos en servidores de segundo plano. De hecho, actualizaciones e inserciones ocupan el centro mismo del comercio electrónico en línea. Sin embargo, la mayoría de los proveedores de servicios en línea no permiten la actualización de sus bases de datos sin alguna modalidad draconiana de

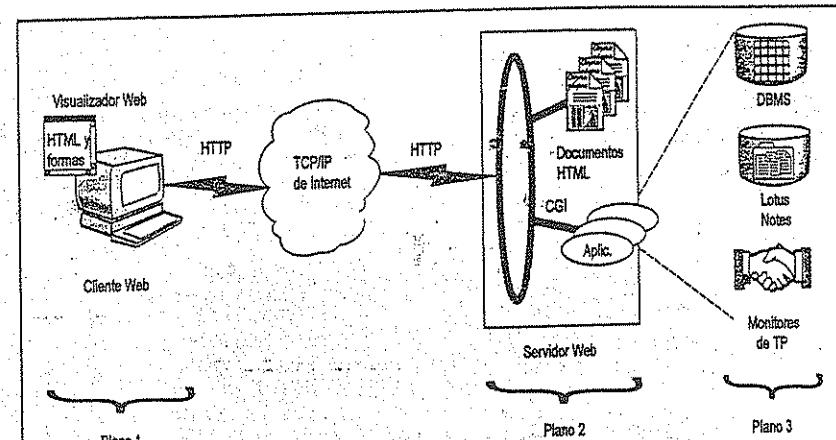


Figura 28-1. Cliente/servidor en 3 planos, estilo Web.

seguridad de cliente/servidor. Aquí es donde entran en juego protocolos como la *capa de sockets de seguridad (SSL: secure sockets layer)*, *HTTP protegido (S-HTTP: secure HTTP)* y los muros de protección (*firewall*) de Internet. Éstos brindan una más de las tecnologías necesarias para hacer de Internet el mayor centro comercial del mundo.

Los proveedores de software están empleando la tecnología que acabamos de describir para conectar visualizadores Web a prácticamente toda clase de sistemas de cliente/servidor, incluidas bases de datos de SQL, monitores de TP, servidores de groupware, colas de MOM, redes principales de correo electrónico, corredores de objetos distribuidos, etc. Todos estos sistemas cuentan en la actualidad con gateways que aceptan solicitudes de CGI. Ofrecen además *transformadores* para hacer corresponder dinámicamente sus datos con HTML a fin de que puedan exhibirse en visualizadores Web. En otras palabras, los servidores crean una página Web "sobre la marcha" para presentar sus resultados.

Como dijeron en la película *Field of Dreams*, "Tú hazlo; ya vendrán". En el caso del Web, hicieron al cliente universal, y detrás de él llegaron todos los servidores. Lamentablemente, y tal como habrá de descubrirlo usted inminentemente, estos servidores viven del otro lado de un cuello de botella llamado CGI.

FORMATOS DE HTML 2.0 BASADOS EN EL WEB

HTML se está convirtiendo poco a poco en una interfaz de programación de aplicaciones.

Bill Macchione, vicepresidente de tecnología
Ziff-Davis
(Noviembre de 1995)

En septiembre de 1995, W3C publicó la especificación de HTML 2.0, que incluye *formatos* para la incrustación en documentos de campos de introducción de texto, botones de opción,



Usted puede anular estas omisiones mientras interactúa con el formato. Expongamos los detalles:

- **Texto** (text) es el campo de entrada predeterminado; permite introducir una sola línea de datos de texto. Usted puede especificar el número máximo de caracteres en el campo por medio del atributo adicional MAXLENGTH.
 - **Contraseña** (password) es un campo de entrada de texto en el que aparecen asteriscos (*) cuando usted teclea un valor en él.
 - **Texto oculto** (hidden) es un campo que no aparece en la forma; puede contener un valor predeterminado que usted envía a la aplicación servidor.
 - **La casilla de verificación** (checkbox) es un campo de dos estados que corresponde a activado o desactivado. Usted puede disponer de un conjunto de casillas desactivadas con el mismo nombre. El visualizador Web ignora las casillas desactivadas; envía las unidades activadas al servidor como pares nombre/valor.
 - **Botón de opción** exhibe un grupo de botones de radio. Todos los botones de opción tienen el mismo nombre, y sólo uno puede activarse. Esto significa que un grupo de botones de opción sólo remite un único valor al servidor.
 - **Botón de reiniciación** (reset) exhibe un botón oprimible del tipo "reset". Usted hace clic en el botón de reiniciación para eliminar el contenido de un formato y restaurar sus valores predeterminados. Dentro de un botón de reiniciación se puede exhibir cualquier palabra usando la propiedad VALOR. Si no se especifica una propiedad NAME, el valor no le será remitido a la aplicación servidor.
 - **Botón de envío para consideración** (submit) exhibe un botón oprimible del tipo "submit". Cuando usted hace clic en el botón de envío, el visualizador reúne los datos de todos los campos de una forma, aparea cada unidad de datos con un nombre y expide después los pares nombre/valor a la aplicación servidor. Cada forma debe tener exactamente un campo del tipo "envío para consideración". HTML no soporta múltiples botones de envío; es decir, uno para cada tarea. Si usted inserta múltiples botones de envío en una forma, todos ellos le remitirán lo mismo. La etiqueta FORM sólo puede enviar el contenido del formato a una única aplicación servidor. Se pueden exhibir las palabras que se deseé dentro del botón de envío usando la propiedad VALUE. Si no se especifica la propiedad NAME, el VALOR no le será remitido al servidor.
 - **Imagen** (image) es un tipo especial de presentación; exhibe una imagen en vez de un botón. El formato se presenta finalmente para su consideración haciendo clic en la imagen.

Ahora que ya sabe para qué sirven estos campos de entrada, repase el HTML de la Figura 28-4. Esta vez tendrá más sentido para usted.

El campo de selección

Una etiqueta SELECT le permite crear un cuadro de lista descendente en el cual un usuario elige uno o varios elementos (u opciones). Los elementos seleccionados se convierten en los

<FORM METHOD="POST" ACTION="HTTP://WWW.MIHOST.ORG/COT-BIN/EJEMPLOFORMA">

Campo de texto <INPUT NAME="nombre" VALUE="Juan Carlos" TYPE="TEXT" SIZE="40">

Área de texto <TEXTAREA NAME="areaTexto" ROWS="5" COLS="20"></TEXTAREA>

Datos <INPUT TYPE="TEXT" NAME="dato1" VALUE="dato1" />

Contraseña <INPUT NAME="contrasena" VALUE="12345" TYPE="password" SIZE="10" />

Botones de opción <INPUT TYPE="RADIO" NAME="t1" VALUE="M" checked="" />

<INPUT TYPE="RADIO" NAME="t1" VALUE="F" />

Castillas de verificación <INPUT TYPE="checkbox" NAME="c1" VALUE="checked" checked="" />

<INPUT TYPE="checkbox" NAME="c1" VALUE="checked" />

<INPUT TYPE="checkbox" NAME="c2" VALUE="checked" checked="" />

<INPUT TYPE="checkbox" NAME="c2" VALUE="checked" />

Seleción <SELECT NAME="selección" />

<OPTION value="1" selected="">Opción 1</OPTION>

<OPTION value="2">Opción 2</OPTION>

<OPTION value="3">Opción 3</OPTION>

<OPTION value="4">Opción 4</OPTION>

<OPTION value="5">Opción 5</OPTION>

<OPTION value="6">Opción 6</OPTION>

<OPTION value="7">Opción 7</OPTION>

<OPTION value="8">Opción 8</OPTION>

<OPTION value="9">Opción 9</OPTION>

<OPTION value="10">Opción 10</OPTION>

<INPUT TYPE="SUBMIT" NAME="enviar" VALUE="ENVIAR" />

Figura 28-4. HTML de la forma muestra.

Campos de entrada

Los campos de entrada (**INPUT**) sirven para introducir y capturar datos. Se pueden especificar ocho tipos de entradas con la etiqueta HTML:

```
<INPUT TYPE="field-type" NAME="Nombre de campo" VALUE="valor predeterminado">
```

La propiedad TYPE le permite especificar un tipo de entrada, el cual puede ser ya sea texto, contraseña, texto oculto, botón de opción, botón de reiniciación, botón de casilla de verificación, de envío o imagen. Las propiedades NAME y VALUE crean los pares nombre/valor que usted pone a consideración de la aplicación servidor. NAME especifica el nombre simbólico de la variable; no es el nombre exhibido. VALUE contiene los datos reales. En nuestro ejemplo, la mayoría de los campos tiene valores predeterminados.



Mes	Maui	Kawai	Big Island	Waikiki
Enero	9,400	2,100	3,200	30,900
Febrero	8,210	1,450	2,640	24,550
Marzo	11,234	2,410	3,560	31,100

Hawaii Hotel Reservaciones para el T.I.

1997-1998

Figura 28-6. Presentación del ejemplo de tabla 1 por un visualizador Web común.

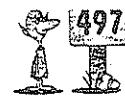
Figura 28-7. Ejemplo de tabla 2 de HTML.

Hawaii Hotel Reservaciones - 1997				
	Maui	Kauai	Bil. Islands	Hawaii
Enero	5400	2100	3200	30.500
Febrero	6210	1450	2.640	24.550
Marzo	1234	2410	3.560	31.100

Figura 28-8. Presentación del ejemplo de tabla 2 por un visualizador Web común.

CGI: LA PARTE DEL SERVIDOR WEB

¿Cómo se transmiten los datos de un formato a un programa situado en un servidor HTTP? Se les transmite por medio de un protocolo de cliente/servidor de extremo a extremo que incluye tanto a HTTP como a CGI. El mejor modo de explicar la dinámica de este protocolo consiste en recorrer una invocación de método POST. Como explicaremos más adelante, debe evitar a toda costa usar GET, aunque sea el método por omisión para la consideración de formatos.



Un escenario de CGI

En la Figura 28-9 se muestra la operación conjunta de los programas del cliente y el servidor para el procesamiento de una solicitud de formato. He aquí la explicación paso a paso de esta interacción:

1. *El usuario hace clic en el botón de envío (submit) del formato.* Esto da lugar a que el visualizador Web reúna los datos del formato y los junte en una larga cadena de pares valor/nombre, separados entre sí por el símbolo inglés conocido como ampersand (&). Convierte espacios entre los datos en símbolos de más (+). El resultado no es precisamente bonito.

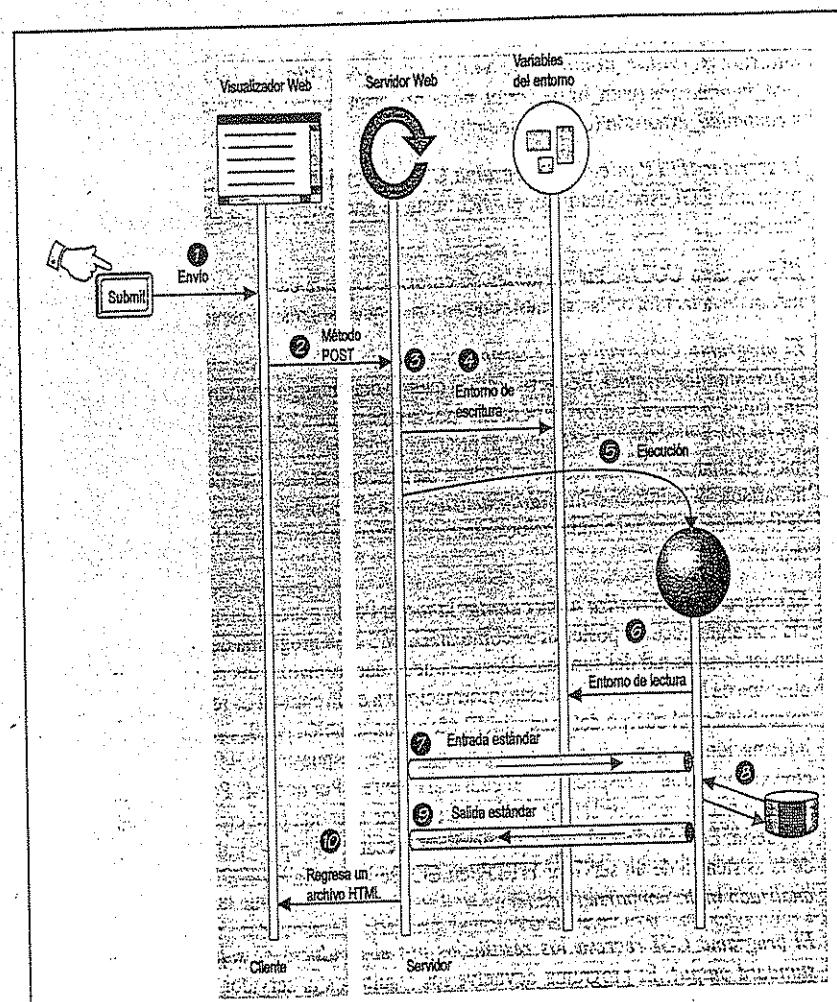


Figura 28-9. POST de HTTP: Un escenario de cliente/servidor de extremo a extremo.

volver a contar con ella en formatos subsecuentes sin que el usuario tenga que reintroducirla, ni estar consciente siquiera de que está siendo transmitida. En otras palabras, los campos ocultos actúan como variables para el mantenimiento de estado entre presentaciones de formatos. Pero, ¿cómo pasa esta información de un formato a otro? A través del programa CGI.

En la Figura 28-10 aparece una transacción electrónica que requiere sumisión de formatos múltiples, los cuales desembocan en un pago electrónico. El primer conjunto de formatos permite elegir la mercancía por colocar en el carrito de compras electrónico. El siguiente formato solicita una dirección y fecha de entrega. En el último usted recibe la factura y la solicitud de alguna forma de pago.

¿Cómo se mantiene el estado de esta transacción entre invocaciones de formatos? El procedimiento es el siguiente: el programa CGI procesa un formato y se lo presenta con el formato siguiente; esto continúa hasta que usted hace el pago final o aborta la transacción. En este caso, el truco es que el programa CGI emplea campos invisibles para almacenar información de los formatos anteriores en el formato siguiente. Por ejemplo, registra en los campos invisibles los bienes que usted seleccionó en los formatos previos. Usted nunca ve el contenido de estos campos ocultos. No sabe siquiera que existen, a menos que visualice el HTML del documento. Sin embargo, cuando usted presenta a consideración el formato, todos estos campos ocultos le son transmitidos al programa CGI junto con cualesquiera datos nuevos.

Así, en esencia, el programa CGI almacena el estado de la transacción en los formatos que remite al cliente, en lugar de almacenarlos en su propia memoria. ¿Qué le parece esta treta? Le advertimos que sería una verdadera obra maestra.

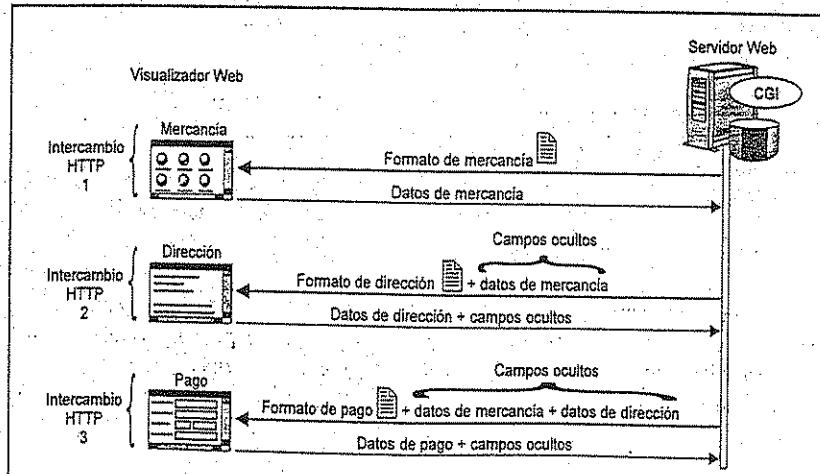


Figura 28-10. Mantenimiento de estado por campos ocultos entre invocaciones del Web.



Advertencia

Aléjese de GET

Aunque GET es el método predeterminado de HTTP para la presentación del contenido de un formato, le recomendamos no usarlo por ningún motivo, y recurrir *siempre* a POST en su lugar. Para comprender esta recomendación, lo mejor es analizar una interacción GET de extremo a extremo. En la parte del cliente, GET provoca que el conjunto del contenido "nombre=valor" del formato se añada al URL tras un signo de interrogación (?). Por el contrario, POST añade el contenido al cuerpo del mensaje de HTML.

En el lado receptor, GET provoca que el servidor HTTP analice el URL e inserte la cadena entera de pares nombre/valor posterior al signo de interrogación (?) en la variable de entorno *cadena de consulta* (*query-string*). El servidor incurre entonces en una gran confusión, porque cree haber recibido una consulta. El programa CGI lee la variable de entorno *cadena de consulta* para obtener el contenido del formato.

¿Qué problema hay con esta imagen? Es letal. He aquí por qué: la mayoría de los sistemas operativos limita sus variables de entorno a entre 256 y 1,024 caracteres en *total*. Esto significa que los datos que excedan de esta extensión máxima serán recortados, o provocarán que el sistema operativo "estalle". Como ya se explicó, es imposible limitar la cantidad de datos de un campo de entrada de líneas múltiples. Por esta razón, use siempre POST cuando se trate de formas; no se le ocurra siquiera utilizar GET. □

SEGURIDAD DEL WEB

Internet ha dado una larga batalla por mantener la seguridad de los recursos ante el acceso ilimitado del mundo exterior.

**Michael Goulde, analista de Seybold
(Enero de 1996)**

La seguridad es un factor decisivo en el establecimiento y aceptación de aplicaciones comerciales del Web. Por ejemplo, si usted hace uso de un servicio bancario en su hogar, querrá estar seguro de que sus interacciones de cliente/servidor sean confidenciales y de ninguna manera se vean expuestas a manipulaciones indebidas. Además, tanto usted como el banco deben estar en condiciones de comprobar su respectiva identidad y de producir registros auditables de sus transacciones.

La seguridad en el Web es una cuestión bilateral que involucra tanto al cliente (el visualizador) como al servidor. Ambos tienen un papel que desempeñar aquí. En la actualidad la mayor parte

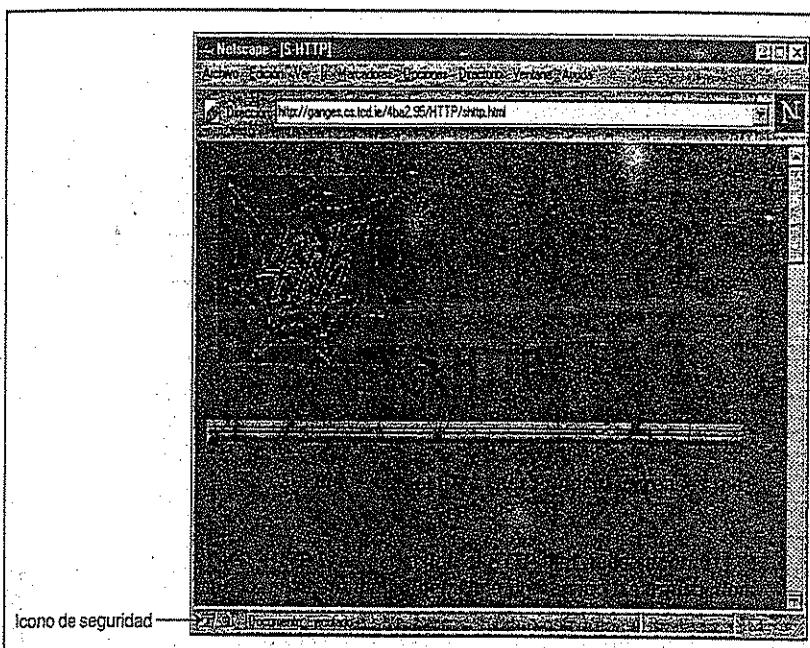


Figura 28-11. Claves de seguridad del Navigator de Netscape.

FYI

La cassetta di sicurezza

Información

¿Cómo pueden saber los clientes que la llave pública de un servidor es válida? ¿Cómo impedir que un servidor impostor le envíe su llave pública? La respuesta es la autoridad de certificación que firma la llave pública. Para el uso de sus características de seguridad, Commerce Server de Netscape exige un certificado firmado digitalmente. Sin un certificado, el servidor operará en modo no protegido. Netscape sólo autentificará a un servidor con llave firmada ya sea por 1) Netscape, a lo cual el público no puede tener acceso, o 2) VeriSign, compañía derivada de RSA. VeriSign firma llaves y extiende certificados digitales a cambio de una cuota, siempre y cuando se cumplan ciertas condiciones.

Para obtener una llave de VeriSign, antes se debe presentar una *solicitud de certificado* junto con \$290 dólares por el primer servidor y \$95 por cada servidor adicional. La solicitud debe incluir información sobre usted y su compañía. Si VeriSign se muestra satisfe-

cha con toda la información ofrecida, le enviará por correo electrónico un certificado firmado, válido por un año para habilitar la seguridad; la renovación anual de la licencia tiene un costo adicional de \$75 dólares por servidor. Usted puede instalar en su servidor el certificado válido firmado para la habilitación de seguridad. Por supuesto que deberá tomar las precauciones usuales para mantener la integridad del certificado firmado y de su llave privada.

Los certificados están protegidos por pares de llaves pública y privada vinculados por un algoritmo criptográfico. Estas llaves poseen la capacidad de codificar y decodificar información. Ninguna llave ajena podrá descifrar mensajes destinados a usted codificados con su llave pública. De igual manera, ninguna llave ajena puede hacerse pasar por usted y enviar mensajes codificados con su clave privada.

De acuerdo con Netscape, este más bien tortuoso proceso de aprobación fue instaurado para "protegerlo a usted, a su organización y a la autoridad de certificación". Además de representar ganancias para RSA/VeriSign, los certificados de seguridad expedidos por terceros autorizados contribuyen a resolver el siguiente problema: ¿cómo emitir para nuestros servidores el equivalente de los números de identificación nacional o licencias de conducir que nos identifican exclusivamente a nosotros? La solución fue en este caso introducir una autoridad —VeriSign— capaz de garantizar el par de llaves pública/privada del servidor.

El uso de toda carretera segura implica un costo. Bienvenido a la cassetta de cobro de seguridad. Lo interesante del asunto es que, en Estados Unidos, VeriSign enfrentará muy pronto la competencia del Servicio Postal. El negocio de la certificación creará fortunas. La gran oportunidad en puerta es la emisión de certificados digitales para millones de máquinas cliente. Esto permitirá que un cliente y un servidor autentifiquen mutuamente su identidad antes de iniciar una transacción electrónica. □

S-HTTP

S-HTTP es una variante de seguridad reforzada de HTTP desarrollada por EIT. En el otoño de 1994, EIT dio a conocer a los miembros de CommerceNet una implantación completa de referencia funcional de un cliente y servidor con S-HTTP. Se dispone ya de una implantación comercial de S-HTTP de Terisa Systems, empresa fundada por EIT y RSA Data Security en 1994. Terisa elabora un producto de software de caja de herramientas de seguridad que permite a desarrolladores de software integrar S-HTTP en sus clientes y servidores Web.

S-HTTP aporta codificación y seguridad al nivel de aplicación sobre comunicaciones ordinarias basadas en sockets. Cliente y servidor se comunican por medio de una sesión ordinaria de HTTP y negocian después sus requerimientos de seguridad; emplean un protocolo semejante a MIME para codificar el contenido de sus mensajes.

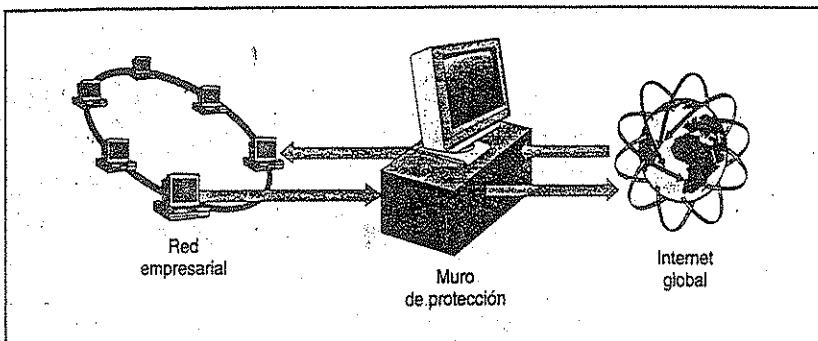


Figura 28-13. Muros de protección (firewalls): La patrulla fronteriza de las redes.

Los filtros de paquetes no mantienen contexto ni conocen a la aplicación con la que tratan. Toman sus decisiones con base únicamente en el análisis del encabezado de IP del paquete en cuestión, e interpretan a continuación las reglas que fueron programados a seguir. Los piratas informáticos pueden explotar esta ausencia de información general para conseguir que paquetes IP falsos obtengan autorización de paso del enrutador; por ejemplo, pueden imitar direcciones IP o máquinas autorizadas. Las redes que dependen únicamente de tecnología de filtración de paquetes son de antemano menos seguras que las resguardadas por muros de protección basados en representantes. Y dado que el mantenimiento de soluciones de filtración de paquetes es más complejo, también resultan mucho más susceptibles a infracciones de seguridad. Es difícil determinar qué huecos han quedado abiertos.

Muros de protección basados en proxy

Los muros de protección con proxy —también llamados muros de *aplicaciones*— son la modalidad de muros de protección más segura. Ejecutan un reducido número de programas —llamados *proxies*—, los cuales pueden ser protegidos y autorizados. Todo el tráfico procedente de Internet se canaliza al gateway proxy correspondiente a correo, HTTP, FTP, Gopher, etc. Los proxies transfieren entonces la información recibida a la red interna, con base en derechos de acceso de usuarios individuales. Dado que el proxy es una aplicación, toma decisiones con base en reglas de contexto, autorización y autenticación, no en direcciones IP. Esto quiere decir que el muro de protección opera al nivel más alto de la pila de protocolos. Esto permite implantar políticas de seguridad basadas en un amplio conjunto de medidas defensivas.

Los proxies son relays entre Internet y la red privada. La dirección de muro de protección del proxy es lo único visible para el mundo exterior. Por lo tanto, las direcciones IP de una red interna son absolutamente invisibles para el exterior. Las personas ajenas deben autenticarse con la aplicación proxy indicada. El proxy no permitirá el paso de paquetes que contengan una dirección de destino final de la red interna. Muchos proxies imponen la adecuación de los programas de aplicación de usuarios terminales para dirigirlos al proxy. Por el contrario,

los *proxies transparentes* son absolutamente transparentes para los usuarios finales, quienes ni siquiera se dan cuenta de estar haciendo uso de sus servicios.

En algunos muros de protección se combinan las técnicas de enrutador y proxy para ofrecer mayor seguridad. En la Figura 28-14 se muestra una conocida combinación de muros de protección que corre proxies en un “anfitrión bastión” y se sirve de un enrutador para bloquear todo el tráfico a y desde Internet, salvo el del anfitrión bastión. El enrutador está configurado para sólo permitir tráfico procedente del IP del anfitrión bastión hacia el interior. Esto significa que computadoras a ambos lados del muro sólo pueden comunicarse por medio de los proxies de la máquina bastión; todo el demás tráfico será bloqueado. La configuración de un enrutador de este tipo es muy sencilla. Así, ésta es una forma para dormir mejor durante la noche.

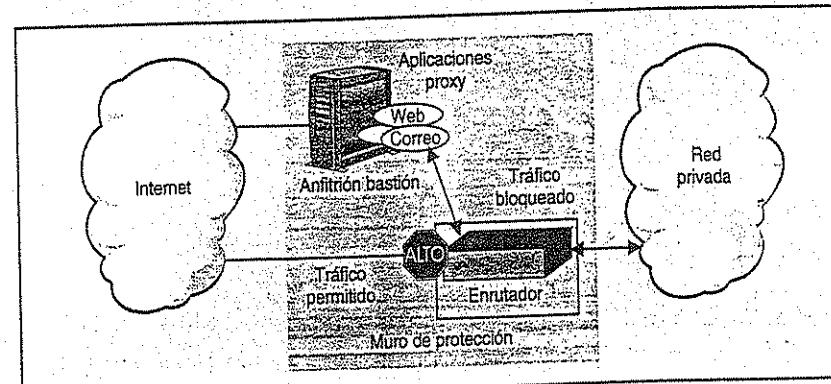


Figura 28-14. Muro de protección Combo: Proxies y enrutadores.

Pagos electrónicos

Bancos y comercios se han lanzado en estampida sobre Internet no porque ésta sea muy divertida, sino por razones estrictamente económicas. El costo de una transacción en cajas es de alrededor \$2 dólares; en cambio, el costo de una transacción electrónica es de menos de 15 centavos. El primer banco del mundo en incorporarse a Internet —Security First— abrió sus puertas digitales en octubre de 1995 (véase Figura 28-15). Ahora muchos bancos permiten a sus clientes el uso del Web para el pago de cuentas por medios electrónicos, consultas de saldo de sus cuentas de cheques y presentación de solicitudes de préstamos.

La mayoría de los bancos están creando su infraestructura de pagos electrónicos sobre SSL y S-HTTP (véase el siguiente recuadro de debate). Sin embargo, no hay motivo para que todas las transacciones de efectivo deban ocurrir en el contexto de diálogos de S-HTTP o SSL.

Por ejemplo, CyberCash permite el uso de la tarjeta de crédito personal, pero conduce la transacción a su propio canal, protegido con el banco emisor de la tarjeta. No recurre a SSL ni a S-HTTP, pero cuenta con su propia codificación; puede hacerlo porque proporciona el software que corre en ambos extremos del canal protegido. En cierto modo, lo que hace es simplemente evitar Internet

Debate

¿Visa o Mastercard?

Tim Vernel, comerciante de artículos deportivos en Internet

Este es ridículo.

Antes de enviar pagos o su tarjeta de crédito por el Web, seguramente deseará protección en caso de fraude. Es aquí donde Visa y Mastercard entran en escena. Hasta hace poco, cada una de ellas soportaba un estándar diferente para pagos electrónicos. El estándar de Mastercard —llamado *protocolo de pago electrónico seguro (SEPP: secure electronic payment protocol)*— es respaldado también por Netscape, RSA, IBM y CyberCash (Netscape lo llama *valija segura*). El estándar de Visa —llamado *tecnología de transacciones protegidas (STT: secure transaction technology)*— es respaldado a su vez por Microsoft, Spyglass, RSA e Internet Shopping. La buena noticia es que Visa y Mastercard han convenido trabajar en una solución común, denominada *transacción electrónica protegida (SET: secure electronic transaction)*. Es probable además que terminen apoyando la *iniciativa de pagos electrónicos conjuntos (JEPI: joint electronic payments initiative)* anunciada por W3C y CommerceNet en abril de 1996. JEPI busca compatibilidad entre los diferentes esquemas de pago electrónico, incluidos *e-cash*, *e-checks* y tarjetas de crédito electrónicas. Si esto se logra, los comerciantes en Internet se verán libres de la molestia de manejar múltiples sistemas de pago. Se trata de una guerra de la que no podemos prescindir.

INTERNET E INTRANETS

Las aplicaciones empresariales internas son la mayor oportunidad y el área de crecimiento más rápido.

**Mike Homer, vicepresidente de comercialización
Netscape**
(Octubre de 1995)

La tecnología de cliente/servidor en el Web ya se usa también en redes privadas llamadas *intranets*. Se trata de redes empresariales internas desarrolladas con el uso de la tecnología de cliente/servidor en el Web. Pueden ser redes empresariales independientes, o bien situarse del otro lado de un muro de protección. Una intranet simplemente influye en la tecnología pública en la que se basa el Web. Pueden convertirse en las nuevas redes principales empresariales. Netscape ha informado que las intranets representan más del 70% de sus ventas de servidores.

Intranet o no intranet

En 1996, los instrumentadores de aplicaciones cliente/servidor que deseen crear aplicaciones internas decisivas para misiones tendrán poca necesidad de tecnología derivada de Internet más allá de propósitos de soporte.

Gartner Group
(28 de diciembre de 1995)

Es lógico que las empresas ya se hayan estandarizado en TCP/IP para contar con soporte de HTTP en sus redes principales de cliente/servidor. HTTP permite el uso de software de cliente y servidor Web de muy bajo costo. Los visualizadores Web resultan muy sencillos para los usuarios, corren en plataformas de cliente múltiples y son prácticamente gratuitos. También los servidores Web son relativamente baratos. Brindan gateways a todas las plataformas de servidor conocidas, lo que los convierte en punto estratégico para la recopilación y distribución de información.

Asimismo, las herramientas de cliente/servidor en el Web están convirtiendo a éste en una plataforma de desarrollo de aplicaciones cliente/servidor para las masas. Estas masas incluyen a los desarrolladores empresariales, quienes ahora cuentan con la posibilidad de emplear tecnología de Internet en sus aplicaciones cliente/servidor de desarrollo interno. Sin embargo, no deje de leer las advertencias al respecto en el siguiente capítulo antes de subirse al tren de “Internet es el sistema”. En el mejor de los casos, el Web actual es apenas una infraestructura en construcción. El protocolo HTTP/CGI, carente de estado, puede ser excelente para consultas simples e información de revisión. Pero padece muchas deficiencias como para permitir la creación de aplicaciones cliente/servidor ricas, decisivas para objetivos escalables y de alcance empresarial.

Intranets contra LAN empresariales

Hace apenas unos años, parecía que las LAN se extenderían hasta convertirse en redes principales intergalácticas. Pero ahora da más bien la impresión de que Internet es la red principal intergaláctica; las LAN han terminado por ser simples extensiones de Internet. En algunos círculos se considera a las LAN empresariales como extensiones privadas de Internet muy veloces; en otras palabras, como intranets. Usted puede interconectar intranets geográficamente distribuidas a través de la red principal de Internet (usando muros de protección).

Construidas sobre LAN y WAN empresariales veloces, las intranets pueden dar lugar a la proliferación de “terminales Web” de bajo costo en las empresas. Las intranets con la velocidad de LAN ofrecen el gran ancho de banda que requieren esas terminales Web. Por supuesto que las PC no van a desaparecer, como han sostenido ciertos especialistas.

En la Tabla 28-1 se hace una comparación entre el estado actual del Web y los entornos de cliente/servidor tradicionales. Obviamente, el Web hace algunas cosas muy bien. Pero en su situación actual, dista mucho de ser el redentor que librará a los IS empresariales de sus aflicciones de cliente/servidor. Como explicaremos en el siguiente capítulo, el Web tiene que com-

JAVA Y HOTJAVA

Java ha hecho volar los cimientos de Bill Gates y destruido su modelo de programas de software comercial que sólo corren en su plataforma.

Scott McNealy, director general, Sun
(Diciembre de 1995)

A menos que haya pasado el último año oculto en una cueva, lo más probable es que ya haya oido hablar de Java, el nuevo lenguaje de programación orientado a objetos de Sun. Java es más que un lenguaje derivado de C++; es también un entorno de sistema operativo exportable. Además, permite generar componentes exportables que pueden distribuirse en el Web. Una de las partes más importantes del entorno de Java es *HotJava*, visualizador Web especial capaz de interpretar código generado con Java. HotJava fue desarrollado por Sun para mostrar las capacidades del lenguaje de programación Java. Tanto Netscape como Spyglass han adoptado ya características de HotJava. Cuando usted lea este libro, todos los principales visualizadores en el mercado ofrecerán sin duda capacidades semejantes a las de HotJava.

¿Hay algo que Java no pueda hacer?

Cada quien tiene su propia idea de Java; la realidad se impondrá en unos cuantos meses.

Christine Comaford, columnista, PC Week
(Abril de 1996)

¿Hay algo que Java no pueda hacer? Resulta que sí, mucho. En primer lugar, Java en sí mismo no es un ORB; carece de bus de objetos distribuidos. Precisa de facilidades semejantes a las de CORBA para comunicarse con las aplicaciones existentes a través de una red, así como para que se comuniquen entre sí los componentes de Java. Sin embargo, Sun y otros proveedores de CORBA se hallan ya en proceso de creación de ORB de Java compatibles con CORBA. En segundo lugar, el Java actual no cuenta con una estructura de documentos compuestos a la manera de OLE u OpenDoc. También en este caso se realizan ya acciones para soportar OLE y OpenDoc en Java. En tercero, Java no ofrece servicios de componentes distribuidos semejantes a los de CORBA. Como era de suponer, Sun y socios se empeñan ya en lograr que los servicios de CORBA estén disponibles en el entorno de Java.

Además, Java necesita mejores herramientas de cliente/servidor, compiladores y todo lo que se puede esperar de un entorno de desarrollo maduro. Lo interesante es que el mercado de herramientas de Java se encuentra en plena ebullición. Cuando usted lea esto, podrá disponer de excelentes herramientas de cliente/servidor con Java de Borland, Symantec, Sybase/PowerBuilder, JavaSoft, IBM, Microsoft y muchos otros (véase Figura 29-1). Dedicaremos el resto de esta sección a explicar el lenguaje Java, el exportable entorno de sistema operativo de Java y las facilidades de componentes.

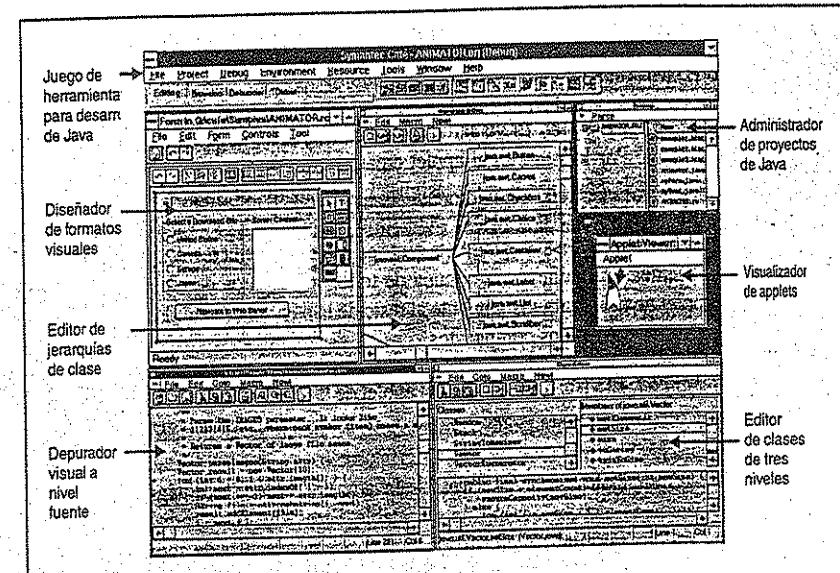


Figura 29-1. Cafe de Symantec: El primer armador visual de Java.

Cliente/servidor en el Web, estilo Java

Tras formar en Java la parte del cliente de una aplicación, el lanzamiento de una aplicación cliente se reduce a comutar a una página. La instalación es trivial; basta hacerlo con un servidor Web. Y no hay puertos, sólo una versión de la aplicación.

James Gosling, creador de Java
(Septiembre de 1995)

Java introduce un modelo completamente nuevo de interacción de cliente/servidor en el Web. Nos permite generar pequeños programas como componentes llamados *applets*, que pueden descargarse en un visualizador compatible con Java. Los *applets* nos permiten distribuir contenido ejecutable a lo largo del Web junto con datos. En la Figura 29-2 se muestra un escenario de interacción de cliente/servidor en el Web que incluye un applet de Java. He aquí los pasos a seguir:

1. **Solicitud del applet.** Un visualizador Web solicita un *applet* de Java cuando encuentra la nueva etiqueta HTML <APPLET>. Los atributos de la etiqueta incluyen el nombre del programa: el nombre de archivo de la clase. El programa suele residir en el mismo servidor en el que se origina la página HTML.

La magia de los códigos de bytes

Java viene a colocarse justo en el corazón del territorio de Microsoft y rompe así el cerco del escritorio.

**Dr. Jeff Sutherland, Homepage.Journal
(Marzo de 1996)**

Al igual que todos los sistemas de código móvil, Java ofrece tanto exportabilidad como seguridad. Un *applet* de Java es una unidad exportable de código móvil. Java logra la exportabilidad compilando *applets* en la *máquina virtual de Java*, la cual sigue el modelo de instrucciones de un procesador RISC virtual. Estas primitivas instrucciones se llaman *códigos de bytes (bytecodes)*. Los códigos de bytes trasladan las instrucciones compiladas al nivel más bajo posible sin volverlas dependientes de la máquina. El lenguaje Java se afirma especificando el tamaño de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos. Los programas son los mismos en cualquier plataforma; no existen incompatibilidades de tipos de datos entre arquitecturas de *hardware* y *software*.

Los códigos de bytes hacen de Java un lenguaje parcialmente compilado. La creación de códigos de bytes representa alrededor del 80% de la labor de compilación; el otro 20% es realizado por el tiempo de ejecución de Java. Así, puede decirse que Java es un 80% de compilación y un 20% de interpretación. Esta combinación 80/20 parece ofrecer una excelente exportabilidad de código; la abstracción de códigos de bytes fue diseñada para el eficiente transporte de código entre múltiples plataformas de *hardware* y *software*.

Por supuesto que ésto implica un costo. En este caso, se gana en exportabilidad lo que se pierde en desempeño; el código interpretado de Java es unas quince veces más lento que el código compilado nativo. Java también soporta compiladores regulares, así como compiladores *justo a tiempo*, capaces de generar código que corra a velocidades nativas de C++.

El verificador de Java

En la Figura 29-3 aparecen los pasos a seguir para crear y ejecutar un *applet* de Java. En primer término, usted corre su *applet* a través de un compilador de máquina virtual de Java para crear los códigos de bytes y almacenarlos en un servidor. Sus códigos de bytes serán copiados en una máquina cliente destino cuando un visualizador solicite su *applet*. Tan pronto como los códigos de bytes arriben a la máquina destino, correrán a través de un *verificador de Java*, sumamente suspicaz en cuanto al código que recibe.

El verificador corre los códigos de bytes por un guante de pruebas. Analiza la presencia de cosas como punteros falsificados, violaciones de acceso, desajustes de tipos de parámetros y sobreflujos de pilas. En cierto sentido, el verificador actúa como "portero", asegura que el

código que recibe de fuentes tanto locales como remotas sea seguro. No se permite la ejecución de ningún código que no apruebe los análisis del verificador. Si éste se muestra satisfecho, cede los códigos de bytes al cargador de clases.

El *cargador de clases* suele transmitir los códigos de bytes a un *intérprete*. Éste es el elemento de tiempo de ejecución que ejecuta las instrucciones de Java en la máquina destino. El intérprete puede proceder a la ejecución sin revisar nada, porque sabe que el código recibido ha sido purificado. Puede correr a toda velocidad sin demérito de la confiabilidad.

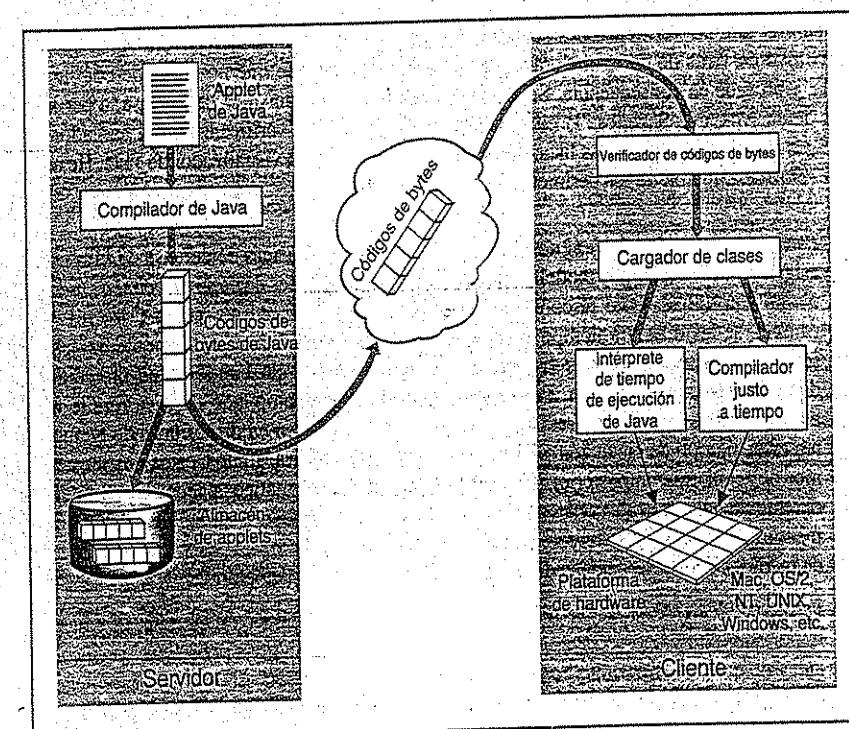


Figura 29-3. El ciclo de códigos de bytes: De la producción a la ejecución.

Sin embargo, incluso "a toda velocidad" un intérprete es de un orden de magnitud más lento que el código compilado nativo. Así, los diseñadores de Java intentan ahora arreglar la situación con técnicas de compilador *justo a tiempo*, lo que significa traducir un *applet* completo a código nativo antes de ejecutarlo. Esto puede incrementar un tanto la carga general durante el proceso de traducción, pero permite que el código resultante corra a casi las velocidades nativas.

En suma, Java rodea a sus applets de varios niveles de protección. Controla todo acceso a recursos del cliente y procesos de servidores remotos. En teoría, Java permite crear aplicaciones que no puedan ser invadidas desde el exterior y protegidas contra la intrusión de código no autorizado que pretenda entrometerse y generar virus o invadir sistemas de archivos. Claro que aún es muy pronto para cantar victoria.

El lenguaje Java

Java es C más-más-menos-menos.

*Bill Joy, cofundador
Sun
(Diciembre de 1995)*

Sun describe el lenguaje Java como "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, neutral en cuanto a arquitecturas, exportable, multihilos y dinámico". ¿Qué más podríamos añadir? Que aún no nos hemos ocupado de los aspectos de hilos múltiples y orientación a objetos del lenguaje.

A diferencia de C++, Java sólo soporta un solo modelo de herencia para implementaciones de clases de objetos. Sin embargo, soporta herencia múltiple al nivel de interfaces. Una *interfaz* de Java es muy semejante al IDL de CORBA; especifica los contratos de interfaces de una clase sin inmiscuirse en la implementación. Una clase de Java puede implementar una o más interfaces, así como añadir sus propias funciones. Sin embargo, puesto que Java carece de herencia de implementación múltiple, se debe reimplementar la funcionalidad de la interfaz en cada clase que implemente esta interfaz.

Java aborda el problema de colisiones de nombramiento con C++ introduciendo un concepto de espacio para nombres llamado *paquetes*. Un paquete, como los módulos del IDL de CORBA, reúne clases afines en paquetes nombrados.

A diferencia de C++, Java ofrece recolección de basura automática. Usted no tiene que ocuparse de la destrucción de objetos por medio de destructores. Esto quiere decir que se pueden crear instancias de nuevos objetos sin temor a fugas de memoria. Las características multihilos de Java también son muy útiles; facilitan enormemente la provisión de código exportable asegurado con hilos. Java integra asimismo manejo de excepciones; la documentación de un método debe indicar las excepciones que tolera. En general, Java es una delicia para el programador de C++. Conserva las bondades de C++ y elimina todo lo desagradable. Lo único que lamentamos es la pérdida de herencia de implementaciones múltiples; ocurre que nos gustan los mixins.



APPLETS: COMPONENTES, ESTILO JAVA

En el mundo anterior a Java, una página Web era esencialmente una hoja de papel. En el mundo de Java, un visualizador se convierte en una estructura.

*James Gosling, vicepresidente
JavaSoft
(Septiembre de 1995)*

Usted puede correr aplicaciones Java como programas independientes o como applets invocados por un visualizador. Técnicamente, un applet es una pieza de código que hereda su comportamiento de la clase Applet de Java, que después extiende con la nueva función. Un *applet* no es una aplicación completa. En realidad es un componente que corre en el entorno de un visualizador. En este caso, el visualizador funge como estructura para la ejecución de los componentes de Java, o *applets*.

¿Qué tipo de servicios de estructura de componentes prestan los visualizadores? Aunque el visualizador Java está lejos de ser un contenedor OLE u OpenDoc, ofrece a sus applets tres servicios útiles. Primero, el visualizador controla por completo el ciclo de vida de los applets. Segundo, provee a los applets de información de atributos de la etiqueta APPLET. Tercero, sirve como programa o proceso principal dentro del cual se ejecuta el applet; aporta la función básica. Detengámonos brevemente en los dos primeros servicios.

Administración del ciclo de vida de los applets

Un visualizador Web administra todas las fases del ciclo de vida de un *applet*. Recuérdese que el visualizador descarga primero el applet y después lo corre en su entorno. Tiene además la gentileza de informarle al applet acerca de los eventos más importantes ocurridos en su ciclo de vida. La interfaz *Runnable* de Java define los métodos que un visualizador invoca en un *applet* durante su ciclo de vida (véase Figura 29-5). La mayoría de los applets de Java implementan código que reacciona a estas invocaciones de métodos.

El visualizador invoca *Init* (Iniciación) después de cargar el applet en la memoria. Invoca *Start* (arranque) para indicarle al applet que haga todo lo necesario antes de correr. *Paint* (pintura) instruye al applet que exhiba su contenido. *Stop* (alto) y *Destroy* (destrucción) le indican anular todos sus hilos, detener la ejecución y liberar todos sus recursos. Esto ocurre cuando el visualizador remplaza la página Web que contiene al applet por otra página. En síntesis, el visualizador es el dueño del applet; le ofrece un entorno de ejecución que incluye administración de su ciclo de vida.

```
<INSERT
CLASSID="Java:NervousText.class"
CODE="http://java.acme.com/applets/NervousText.class"
WIDTH=300 HEIGHT=200 ALIGN=Left>
<PARAM NAME=text VALUE="Este es el visualizador Java">
</INSERT>
```

El siguiente ejemplo muestra cómo se usa INSERT para especificar un control de OLE (o ActiveX) para un reloj:

```
<INSERT
ID=clock1
CLASSID="uuid:{663C8FEF-1EF9-11CF-A3DB-080036F12502}"
TYPE="application/x-oleobject"
CODE="http://www.foo.bar/test.stm"
DATA="http://www.acme.com/ole/clock.stm"
WIDTH=300 HEIGHT=200 ALIGN=Left>
</INSERT>
```

El atributo "ID" (identificación) permite que otros controles en la misma página ubiquen el reloj. El atributo "DATA" (datos) se refiere al flujo persistente que usted puede emplear para inicializar el estado del objeto. El atributo "CODE" alude al archivo que contiene el código de este objeto. En caso de que esté presente un directorio global de OLE, quizás lo único que se necesite para ubicar el código sea "CLASSID" (identificación de clase). En el caso de OLE, CLASSID equivale a DCE UUID. En el de objetos de CORBA, es un identificador global. El prefijo sirve para identificar el sistema de objetos. Una vez más, le advertimos que la sintaxis de la etiqueta INSERT aún está "en construcción", aspecto en el que insistiremos más adelante. □

¿JAVA ES EN REALIDAD UN SISTEMA OPERATIVO DISFRASADO?

Dispondremos de un OS simple, aplicaciones pequeñas basadas en componentes y contenido vivo.

Eric Schmidt, director técnico, Sun
(Febrero de 1996)

Sí, Java es un nuevo sistema operativo, al menos para máquinas cliente de Internet. En la Figura 29-6 aparece la visión de Sun acerca de la máquina cliente de Internet de Java ideal. En la base se encuentran los chips RISC, que ejecutan nativamente los códigos de bytes de Java. La familia de procesadores de Java de Sun constará de tres líneas: *PicoJAVA*, *MicroJAVA* y *UltraJAVA*. Los procesadores variarán en precio/desempeño y capacidad de aplicación.

Sun preveía ofrecer muestras de chips PicoJAVA a principios de 1997; tendrían un precio de unos 25 dólares. Eric Schmidt, director técnico de Sun, calcula que los chips de Java podrían correr 50 veces más rápido que las implementaciones actuales. De lograrse este desempeño, Java se convertirá en una de las máquinas más populares de Internet. La especificación de

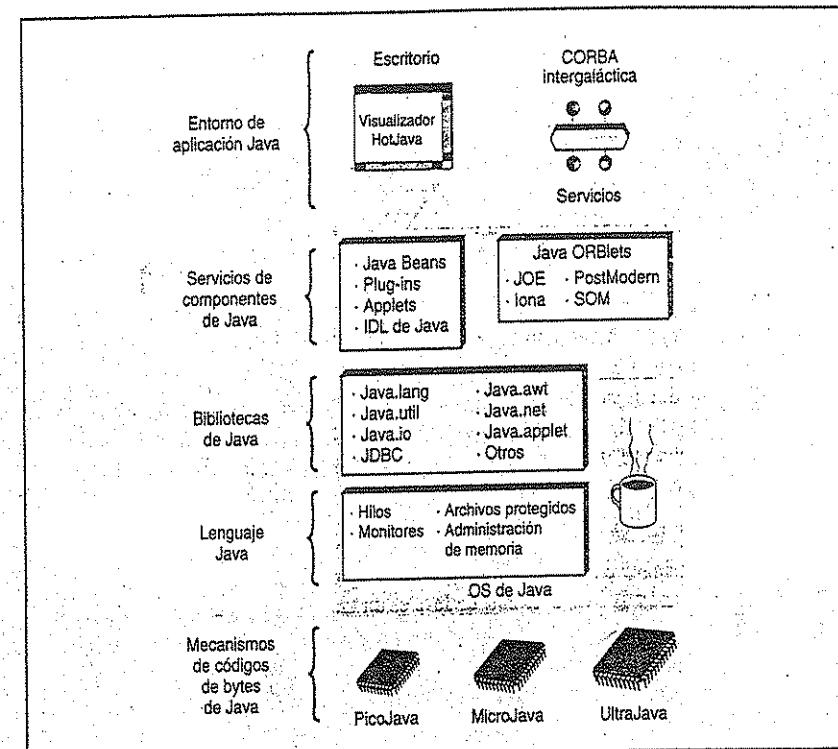


Figura 29-6. El cliente Java por excelencia.

máquina virtual de Java se halla disponible en la página de Java. En consecuencia, es de esperar que otros proveedores lancen al mercado sus propios chips RISC para Java.

En dirección ascendente, la capa siguiente es la del "sistema operativo" de Java. Éste consiste en el lenguaje Java, las bibliotecas de Java, JDBC y ORB. Como ya se explicó, el lenguaje Java propiamente dicho presta potentes servicios en tiempo de ejecución, como hilos, monitores, administración de memoria, seguridad, controles de acceso, etc. En febrero de 1996, Sun anunció Kona, un sistema operativo básico basado en Java. Kona brindará multihilos y enlace en red integrado, pero carecerá de sistema de archivos y no soportará memoria virtual.

Bibliotecas de Java

Las *bibliotecas de Java* extienden el lenguaje; aportan un entorno exportable para la generación de aplicaciones de Java aseguradas con hilos. Se componen de seis paquetes de Java en los que se implementan cientos de clases. He aquí una breve descripción de lo que ofrecen estos paquetes:



Suponemos que la especificación de JDBC estará terminada para cuando usted lea este libro. Sun integrará un administrador de manejadores de JDBC en futuras versiones de Java; también ofrecerá un puente JDBC a ODBC. Creemos que JDBC podría convertirse a la larga en un estándar de API para bases de datos aún más importante que ODBC u OLE/DB.

Cabe señalar que, con la asistencia de SunSoft, ODMG trabaja también en acoplamientos de Java para ODBMS. Estos vínculos serán la vía de acceso más directa para el almacenamiento de objetos de Java; se podrá prescindir por completo de SQL.

Java y CORBA: JOE, PostModern y Iona

Además de HotJava, Sun está desarrollando el *entorno de objetos de Java* (*JOE: Java object environment*), un ORB de CORBA exportable generado enteramente en Java. A causa de su reducida huella, JOE podrá descargarse a solicitud expresa o integrarse al entorno de tiempo de ejecución de Java. Permitirá que cualquier *applet* de Java acceda inconsultilmente a cualquier servicio de CORBA en un ORB intergaláctico. Sun no es el único proveedor en trabajar en un ORB de Java. Cuando usted lea esto, ya estarán disponibles también los ORB para Java de PostModern y Iona.

BlackWidow de PostModern implementa ya las partes de cliente y servidor de un ORB de CORBA 2.0 en menos de 100 KBytes de código de bytes de Java. Este reducido tamaño lo convierte en un ORBlet ideal; es decir, un ORB descargable a solicitud expresa, como cualquier otro applet de Java. Aún tendremos mucho que informar sobre Java y ORB en el capítulo siguiente.

CONCLUSIÓN

Creo que la gente apenas empieza a entender lo que significan realmente las redes.

James Gosling, creador de Java
(Marzo de 1996)

La cuestión básica es que Java ofrece un medio más simple y novedoso para el desarrollo, administración y desarrollo de aplicaciones cliente/servidor. Usted puede tener acceso a la versión más reciente de una aplicación haciendo sencillamente un clic con el ratón. Puede distribuir una aplicación entre millones de clientes colocándola en un servidor Web. La distribución será inmediata. Además, no tiene que preocuparse por la instalación y las actualizaciones.

Java también es magnífico para los servidores. Permite generar código móvil del servidor útil en disposiciones sumamente flexibles. Por ejemplo, los servidores pueden transmitir applets a otros servidores para buscar información. *Software ejecutable* puede trasladarse a donde más se le necesite. Java proporciona un entorno de ejecución flexible y justo a tiempo tanto en servidores como en clientes. ¿Significa esto que finalmente hemos llegado al nirvana de cliente/servidor? No, aún no. En el siguiente capítulo se explicará por qué.

Capítulo 30

Cliente/servidor en el Web: La era de objetos distribuidos



La tecnología del Web y la tecnología de objetos distribuidos se complementan naturalmente. Nos interesa que OMG y W3C trabajen juntos para definir un futuro común.

Tim Berners-Lee, director
W3C
(Febrero de 1996)

Java es el primer paso hacia la creación de un Web de objetos, pero no es suficiente. Tiene que complementarse con una infraestructura de objetos distribuidos, que es donde entra en juego CORBA de OMG. Como se indicará más adelante, PostModern, Sun y Iona ya han anunciado ORB de CORBA basados en Java.

El Web de objetos también necesita una infraestructura de documentos compuestos como OLE y OpenDoc. Los documentos compuestos con Java nos permitirán crear *lugares* *embarcables*. Un lugar es un conglomerado de componentes visuales; es un contenedor de OLE u OpenDoc embarcable repleto de componentes. Lo habitual es almacenar esos lugares en servidores. Usted podrá bajar a su máquina sus lugares preferidos con comandos de HTTP ordinarios. Para interactuar después con un lugar bastará con hacer clic en los componentes visuales que contiene. El contenido de un lugar es dinámico; todo se puede editar en él.



limitado conjunto de operaciones. Las aplicaciones para servidores de BlackWidow son objetos regulares de CORBA. En consecuencia, se puede disponer de ellas en forma permanente. No hay necesidad de incurrir en la sobrecarga de generar un script de CGI para cada invocación. Todos los clientes compatibles con CORBA 2.0 pueden realizar invocaciones en un servidor de BlackWidow.

Para crear clientes y servidores Web compatibles con CORBA, primero se deben definir las funciones que expondrá un objeto de servidor haciendo uso de IDL. Después se corre IDL en el precompilador de BlackWidow. El precompilador genera un esqueleto de código para los objetos del servidor en C++ o Java; también genera talones de Java para la parte del cliente. Los esquemas deben llenarse por supuesto con la lógica de la aplicación. El ORB de BlackWidow fue implantado en menos de 100 KBytes de códigos de bytes de Java. Por lo tanto, se le puede descargar fácilmente en la máquina cliente si aún no está en ella. □

Además de Sun, otros miembros de OMG comenzaron a darse cuenta del potencial de la unión de CORBA con el Web para Java. A fines de 1995, OMG inició un estudio conjunto con el consorcio W3C para la integración del IIOP de CORBA con HTTP y Java. OMG está aplicando asimismo un procedimiento de vía rápida para el desarrollo de ligaduras de lenguaje Java con CORBA. Además, algunos miembros de OMG —como el consorcio europeo ANSA— están ya muy adelantados en sus investigaciones sobre la operación conjunta de CORBA e Internet (véase el siguiente recuadro de debate). Como resultado de todo este trabajo, ya comienza a emerger un nuevo modelo de *Web de objetos* sobre los lineamientos que se muestran en la Figura 30-1.

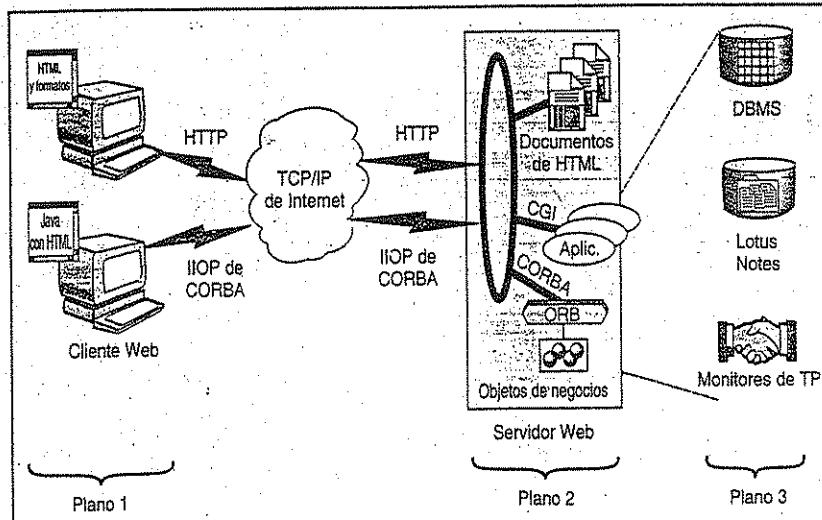


Figura 30-1. Modelo de Web de objetos basado en clientes de Java y ORB de CORBA.



En la figura se muestra un modelo de aplicaciones cliente/servidor en 3 planos que consiste en: 1) clientes de Java en el primer plano, 2) objetos de negocios de CORBA en el plano intermedio y 3) servidores tradicionales en el tercer plano. Los objetos de negocios de CORBA ofrecen la lógica de la aplicación y encapsulan bases de datos, monitores de TP y servidores de groupware existentes. De este modo, los objetos de negocios de CORBA remplazan a las aplicaciones CGI en el plano intermedio, lo cual es realmente magnífico.

Además, el cliente de Java puede comunicarse directamente con un objeto de CORBA por intermedio del ORB de Java. Esto significa que CORBA sustituye a HTTP/CGI como capa de middleware para las comunicaciones objeto a objeto, lo cual es también maravilloso. Al igual que HTTP, el IIOP de CORBA se sirve de Internet como red principal. Esto quiere decir que tanto IIOP como HTTP pueden correr en las mismas redes. HTTP sirve para descargar páginas Web, applets e imágenes; CORBA sirve para las comunicaciones de cliente a servidor de Java. Así, contamos con una solución evolutiva que no prescinde de las actuales aplicaciones para el Web. También debe señalarse que hoy los clientes de Java no pueden comunicarse entre procesos. CORBA resuelve asimismo este problema.



CORBA se encuentra con Internet

Debate

Sabemos de tres metodologías para la fusión de IIOP de CORBA con HTTP/CGI. La primera de ellas, adoptada por *Web Broker* de Digital, consiste en la generación de un gateway de CGI a CORBA. Este método es sumamente evolutivo, pero no resuelve el cuello de botella de CGI ni extiende a Java con una infraestructura de objetos distribuidos. La segunda metodología consiste en proveer gateways bidireccionales HTTP a IIOP y generar un servidor HTTP de CORBA que atienda solicitudes de HTTP. Éste es el método adoptado por ANSA. Si se le aplica adecuadamente, permitiría que CORBA remplazara por completo a HTTP en entornos antiguos y extendiera directamente a Java en nuevos entornos. La tercera metodología es el entorno de "vivir y dejar vivir" que describimos en la sección anterior (véase Figura 30-1).

En nuestra opinión, el tercer método es el más pragmático. Es también el más fácil de implantar y desplegar. En la parte del servidor, simplemente se añade un servidor IIOP de CORBA al servidor HTTP existente. En la parte del cliente, se debe integrar un ORB de CORBA. Esto puede hacerse ya sea descargando un ORBlet de Java del servidor o incorporando un ORB de Java al visualizador Web. El tercer método nos gusta porque permite que CORBA/IIOP y HTTP coexistan pacíficamente en la misma red principal. Sólo se usa al CORBA/IIOP para soportar las nuevas interacciones de cliente/servidor de Java; HTTP sigue atendiendo documentos HTML y las aplicaciones de herencia de CGI. □



negocios, los WebObjects ejecutan la lógica de la aplicación y extraen sus datos de almacenes de bases de datos y servidores de herencia.

Los desarrolladores suelen almacenar el estado persistente del WebObject en bases de datos relacionales como las de Oracle y Sybase. A diferencia de las aplicaciones CGI, los WebObjects ofrecen amplias facilidades para la administración de estado para permitirles a los desarrolladores manejar variables entre invocaciones de clientes. Por ejemplo, usted puede declarar variables que serán válidas para un cliente en particular por medio de invocaciones de HTTP. Como otras aplicaciones CGI, WebObjects genera dinámicamente páginas Web en respuesta a la solicitud de un cliente. Una nueva etiqueta WEBOBJECT es necesaria para especificar las páginas dinámicas. Esta etiqueta HTML no estándar también sirve para especificar las variables y acciones que se pasan a un WebObject.

 **¿Los WebObjects son la respuesta?**

Debate

Lo más importante en este momento es permitir que el Web acumule usuarios y establezca su ubicuidad hasta que se haya afianzado a tal punto que ni siquiera Microsoft pueda apoderarse de ella, después se le podrán añadir todos los accesorios que se deseé. Me temo que el microcósmico afán de perfección le dará a Microsoft el tiempo suficiente para apropiarse del Web.

**Steve Jobs, director general, NeXT
(Enero de 1996)**

¿Cuál es el problema de WebObjects? Ninguno, si se está dispuesto a optar por una buena solución evolutiva. Se cuenta con objetos del servidor sobre la infraestructura de HTTP existente. Pero el problema real es que no se libera la potencia de los objetos de Java sobre el cliente. El bus del servidor de WebObject no se extiende al cliente. Se sigue haciendo uso de HTTP para las comunicaciones cliente/servidor. Además, WebObjects requiere una nueva etiqueta HTTP para expresar la semántica de la interacción cliente/servidor. En este negocio nunca hay felicidad completa.

Pero hagamos de este un verdadero recuadro de debate. Siempre hemos sostenido que el mundo es capaz de tolerar cuando mucho dos buses de objetos distribuidos: CORBA y DCOM/OLE. No creemos que haya lugar para un tercer bus de objetos; en este caso, Objective C. Por supuesto que, como todos los que nos movemos en el ámbito de los objetos, no dejamos de agradarnos las excelentes características de Objective C y de OpenStep. Pero siempre hemos dicho que NeXT debería adoptar a CORBA como bus de objetos distribuidos. Así pues, ¿son los WebObjects la respuesta? Sí, si lo que usted necesita es una solución rápida que no requiera Java en el cliente. Pero ésta no es la arquitectura que elegiríamos para la creación de un Web de objetos a largo plazo. Para ser justos, Steve Jobs acierta cuando dice que en este negocio del Web no podemos darnos el lujo de un "microcósmico afán de perfección". Pero no está de más intentarlo. □



DOCUMENTOS COMPUUESTOS Y WEB DE OBJETOS

Necesitamos un entorno dinámico basado en componentes que sea seguro, independiente de plataformas y extensible, además de fácil de mantener y que nos ofrezca el grado de reutilización que anhelamos. No lo teníamos... hasta Java.

**Christine Comaford, columnista
PC Week
(Enero de 1996)**

Java y CORBA no son suficientes. El Web de objetos también debe complementarse con documentos compuestos. ¿Por qué? Porque los documentos compuestos pueden ofrecer literalmente magia en el Web, especialmente si se les combina con Java. Las estructuras de documentos compuestos —como OLE y OpenDoc— aportan dos importantes tecnologías: 1) un fundamento de componentes visuales para la creación de visualizadores Web abiertos y 2) la tecnología de contenedores para la distribución, alojamiento en memoria caché y almacenamiento de grupos de componentes afines y sus datos; es decir, de *lugares embarcables*. En esta sección explicaremos primero qué hacen por el Web estas tecnologías de documentos compuestos. Asimismo, ofreceremos dos escenarios arquitectónicos para la operación de OpenDoc y OLE con el Web!

Documentos compuestos como visualizadores Web abiertos

Imagínese un visualizador Web formado completamente por componentes. Imagínese además que el propio visualizador Web fuera un contenedor visual de componentes. Esto significaría que sería un componente en el que podrían insertarse otros componentes. Por ejemplo, el visualizador Web podría formarse como contenedor de OpenDoc u OLE. ¿Y qué podría hacer usted con esto?

A quienes se inician en estos misterios, este nuevo tipo de visualizador les ofrece una experiencia visual integrada diferente a todo lo que hayan visto hasta la fecha. Los componentes y los applets de Java podrán compartir inconsútilmente la propiedad visual de una ventana de un visualizador. Usted podrá editar el contenido de cualquier componente *in situ* sin importar qué tan profundamente se halle incrustado dentro de otros componentes. Podrá también arrastrar y soltar componentes con el visualizador, así como entre el visualizador y el escritorio circundante. Esto quiere decir que podrá incrustar componentes dentro de otros componentes y después desplazarlos a voluntad entre documentos, escritorios y redes.

¹ El desarrollo de una infraestructura de documentos compuestos como OpenDoc y OLE tiene un costo superior a los \$100 millones de dólares. Es sumamente improbable que los proveedores de visualizadores puedan recrear toda esta tecnología desde cero. Así, es de esperar que se alien a una de las dos infraestructuras de componentes en pugna.



da componentes comunes que pueden usarse "tal cual" o extenderse recurriendo a proveedores de partes. El pegamento común incluye a *Notebook* de Cyberdog, que permite almacenar las ubicaciones preferidas de Internet en un cuaderno jerárquico. El *Log* le recordará por dónde ha pasado, y le ofrecerá una visualización de bloc de notas de los sitios que visitó. Puede usar las facilidades de arrastrar y soltar de OpenDoc para arrastrar un URL del *Notebook* (o *Log*) a una ventana del visualizador y viceversa.

La módula de Cyberdog es la parte *PathFinder*, que contiene botones para el lanzamiento de otras partes de OpenDoc para Internet (véase Figura 30-2). Usted puede acceder de manera inconsútil a cualquier servicio de Internet haciendo simplemente clic en su botón. Todas las partes, con "inteligencia canina", comparten el *Notebook* y *Log* comunes y actúan como una sola aplicación. Al commutar entre partes no se experimentan cambios de contexto. Además, múltiples partes pueden activarse al mismo tiempo.

Cyberdog también ofrece una parte *Mail* acorde con MIME. Se puede crear un mensaje de correo electrónico aplicando estilos en cierto texto, en el que se puede arrastrar y soltar cualquier tipo de contenido visual, entre ellos URL que funjan como vínculos activos, no como direcciones de texto. Luego, el mensaje de correo —con los URL incrustados— se envía a otros usuarios de Cyberdog (véase Figura 30-3). Los agentes de correo de Cyberdog emplean los encabezados de mensajes de MIME para filtrar la entrada de correspondencia —con base en los diversos criterios especificados por el usuario— antes de hacerla llegar a las charolas de recepción adecuadas.

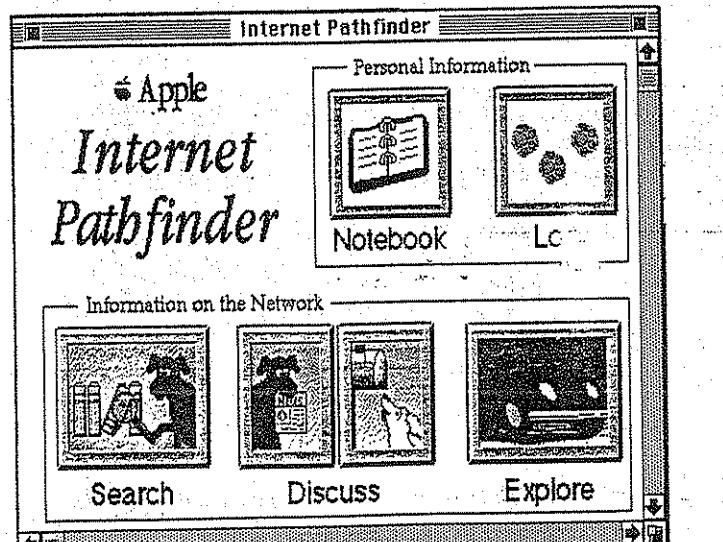


Figura 30-2. Parte contenedora del Pathfinder para Internet, de Cyberdog.

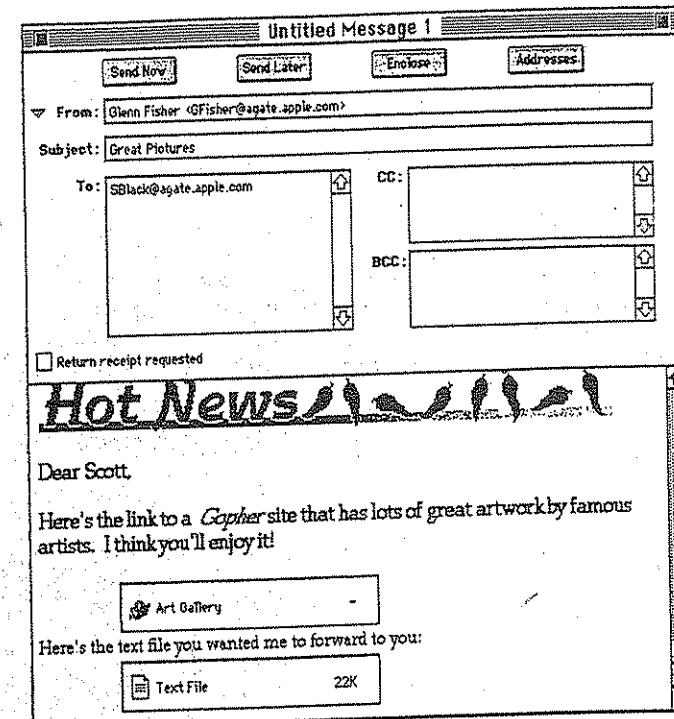


Figura 30-3. La parte *Mail* de Cyberdog.

Las partes de Cyberdog operan con cualquier aplicación de contenedores de OpenDoc. Esto quiere decir que se pueden arrastrar partes y soltarlas en el escritorio, y viceversa. Se pueden enviar partes en contenedores de correo electrónico. Finalmente, se pueden incrustar partes de Cyberdog en cualquier aplicación de OpenDoc para habilitarlas con Internet. Las partes de Cyberdog funcionan de la misma manera sin importar el contenedor que se use.

Con Cyberdog, un URL es una parte de primera clase. Se pueden arrastrar y soltar URL entre contenedores, visualizadores Web y el escritorio. Se les puede soltar incluso en un *Cyberbutton* para crear vínculos con otros recursos del Web. Por ejemplo, usted puede arrastrar la papelería de Cyberbutton desde la paleta de OpenDoc, soltar sobre ella un ícono de mapa de bits para personalizar la apariencia visual del botón y después activar éste soltando un URL sobre el ícono. Puede crear todos los botones personalizados que desee, colocarlos dentro de un mensaje de correo o página Web y distribuirlos por medio de correo electrónico al resto del mundo.

La parte del visualizador de Cyberdog soporta todas las extensiones de HTML 2.0 y casi todas las de HTML 3.0. Los ingenieros de Cyberdog planean proporcionar envolturas para applets de Java y conexiones de Netscape a fin de que se les pueda tratar como a

página entera, con lo que se destruye el contexto del usuario. Además, los servidores no disponen de un medio sencillo para hacerles saber a los clientes de los cambios sucedidos. Las ubicaciones resuelven estos problemas; ofrecen un ancla en el cliente a la que los servidores pueden llamar.

El cliente Web del futuro

La cápsula del sistema operativo y el visualizador de Internet serán una y la misma cosa.

Bill Gates, presidente
Microsoft
(Abril de 1996)

En la Figura 30-4 se observan los tres modelos de cliente Web. En el primer modelo, el visualizador es el escritorio; supone que la gente vive en sus visualizadores. Éste es el modelo actual de Netscape. En el segundo, todo lo que se halla en el escritorio ha sido habilitado para el Web; la idea es estar en condiciones de acceder al Web desde cualquier aplicación o componente sin necesidad de iniciar el visualizador. Éste es el modelo de Cyberdog; Microsoft lo soportará también en Windows 97. El tercer modelo es el de los lugares embarcables; permite acceder al Web desde las ubicaciones del usuario. Una ubicación puede sostener sesiones concurrentes múltiples con servidores de objetos del Web. Además, múltiples ubicaciones pueden estar concurrentemente activas en el mismo escritorio. El Web de objetos quizás termine por soportar estos tres modelos.

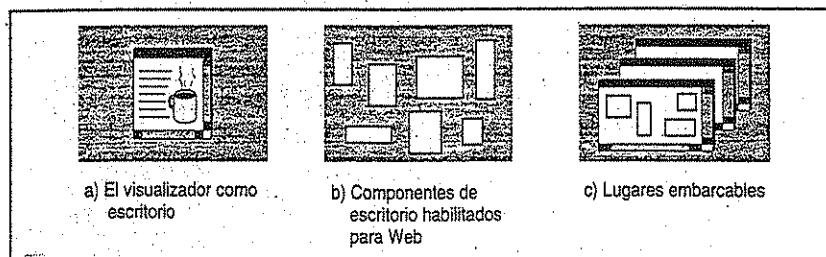


Figura 30-4. Evolución del modelo del cliente Web.

En síntesis, las estructuras de documentos compuestos —como OpenDoc y OLE— fueron creadas para transportar y exhibir componentes. Ahora nos permitirán desplegar páginas Web formadas por componentes. Y “componentizarán” al propio visualizador Web. Los documentos compuestos nos permitirán crear, ensamblar y distribuir una variedad infinita de contenido Web dinámico. Tanto OpenDoc como OLE soportarán componentes distribuidos encima de sus respectivos ORB, CORBA y DCOM. En la sección siguiente especularímos sobre la forma en que las piezas pueden operar juntas en un Web de objetos con CORBA u OLE (véase también el siguiente recuadro de debate).



Debate

¿Cuál Web de objetos?

En lugar de tener contenedores de OpenDoc o contenedores de OLE, o como se les vaya a llamar, usted tendrá documentos HTML que contengan todos estos diferentes tipos de cosas: objetos de Java, archivos de Macromedia Director, diferentes tipos de formatos de audio, formatos de video, formatos de documentos exportables, prácticamente todo.

Marc Andreessen, vicepresidente,
Netscape
(Diciembre de 1995)

Microsoft ha construido una infraestructura para la computación de cliente/servidor en la intranet empresarial que a Java le tomará años crear.

Dr. Jeff Sutherland, vicepresidente de
Objects
VMARK Software (Marzo de 1996)

La cuestión de fondo es que Java es incapaz de ofrecer por sí solo un Web de objetos para cliente/servidor en 3 planos. Necesita complementarse con un ORB y alguna modalidad de estructura de documentos compuestos. La creación de un Web de objetos implica a estos tres elementos. No coincidimos con Marc Andreessen en que HTML podrá evolucionar de la noche a la mañana para convertirse en una arquitectura de documentos compuestos; le llevaría dos años más y más de 100 millones de dólares recrear las características que OLE y OpenDoc ya poseen. Y Andreessen no puede darse el lujo de esperar dos años; Microsoft no se lo permitiría. Era de esperar que Microsoft contara ya con las tres piezas a fines de 1996, con el lanzamiento de Network OLE. También dispone de magníficas herramientas y de seguidores leales entre los desarrolladores de Windows/OLE (fuerza integrada por más de un millón de individuos).

El bando de Java contaría ya con ORBlets de CORBA antes del lanzamiento de Network OLE. Sin embargo, también debe ofrecer una arquitectura integrada de documentos compuestos para competir eficazmente con Microsoft. Y aquí es donde OpenDoc y Cyberdog entran en escena. OpenDoc es multiplataforma y neutral en cuanto a proveedores; sus especificaciones son controladas por CI Labs y OMG. No es inconcebible que CI Labs y OMG pongan las versiones para Internet de CORBA y OpenDoc bajo la custodia de W3C, el consorcio de múltiples proveedores que controla los estándares de Internet. No es de suponer en cambio, por ningún motivo, que Microsoft ponga bajo la custodia de W3C la especificación de OLE.

De manera que todo indica que se acerca otra gran guerra. Lo que estará en juego esta vez será el control del Web de objetos. Microsoft desplegará DCOM, OLE DocObjects y VB Script. Si logran unirse, los bandos de Internet desplegarán Java, CORBA y OpenDoc, y ofrecerán también puentes a OLE ActiveX. ¿Puede evitarse esta guerra? Sí, si ocurre una de dos cosas: 1) Microsoft cede a W3C la custodia de las especificaciones de OLE, DCOM y Windows o 2) la comunidad de Internet es incapaz de ofrecer una solución de Web de objetos unificada basada en estándares abiertos. ¿Cuál cree usted que sea más probable? □

HTML ordinarios, IIS fungirá como depósito y memoria caché de títulos de documentos de OLE. IIS rastrea un documento (o título) asignándole una identificación única global (GUID: *globally unique ID*); establece correspondencias entre GUID y URL mediante *monikers* de OLE, nombre de lujo para un servicio de nombramiento rudimentario (véase siguiente recuadro de detalles).



¿Qué es un moniker asíncrono?

Detalles

En Sweeper se introducirá un nuevo tipo de moniker, llamado *moniker asíncrono*. Los visualizadores lo usarán para bajar archivos por medio de distintos hilos de segundo plano. El *moniker* notifica al visualizador, vía una llamada de verificación, cuando ha terminado de bajar. A esto se llama "carga perezosa". La idea es permitirle al visualizador interactuar con el usuario incluso mientras se bajan grandes objetos de la red. El *moniker de URL* es una implementación del moniker asíncrono; sirve para encapsular URL de Internet. Cabe señalar que los monikers de URL de OLE adoptaron muchas de las características de una entidad similar de Cyberdog. □

IIS ofrecerá también una estructura de aplicación para la ejecución de objetos de negocios basados en OLE. Esta estructura permitirá que componentes del cliente —como ActiveX, applets de Java y applets de VB— invoquen objetos de negocios del servidor a través del ORB de Network OLE (también llamado DCOM). Asimismo, los objetos de negocios interactuarán entre sí por medio del ORB de OLE. Harán lo propio con aplicaciones de herencia mediante API para CGI, ODBC e ISAPI provistas por IIS.

El tercer plano lo ocupa BackOffice de Microsoft, y cualquier aplicación servidor a la que se pueda acceder por ISAPI u ODBC. Microsoft pretende ofrecer más adelante un monitor de TP basado en OLE —llamado *coordinador de componentes*— para orquestar transacciones entre diferentes administradores de recursos. Cuando esto suceda, los objetos de negocios serán aptos para transacciones. Por supuesto que un monitor de TP también brinda equilibrio de cargas y facilidad de ampliación. Adicionalmente, Cairo de Microsoft proporcionará un ODBMS para la administración y acceso a BLOB y otras modalidades ricas en datos. En la cima de todo esto, Microsoft proporcionará herramientas para el ensamblaje de aplicaciones en el Web de objetos de cliente/servidor en 3 planos. Estas herramientas permitirán la construcción (o ensamblaje) de las partes tanto de cliente como de servidor de la ecuación del Web de objetos.

En resumen, Microsoft ha reunido una impresionante versión del Web de objetos. ¿Alguna advertencia por hacer? Sin que se pretenda convertir esta sección en un recuadro de debate, parece haber tres problemas en esta versión: 1) IIS se centra en NT, 2) Network OLE es un ORB propietario aún no probado y 3) la arquitectura de documentos compuestos de OLE es propietaria. Si a usted no le importan ni el carácter propietario ni la dependencia de NT, entonces esta versión le resultará excelente. Sin embargo, nos cuesta trabajo imaginar la posible conciliación entre los estándares cerrados de Microsoft y la cultura abierta del Web. Aun así, hay que decirlo: el reto de Microsoft obligará a W3C a acelerar su calendario para una arquitectura Web de objetos abierta, como lo explicaremos en la siguiente sección.



¿Qué son DocObjects y ActiveX?

información

DocObjects son documentos de OLE capaces de actuar como sus propios contenedores. Microsoft los desarrolló originalmente para Office 95, donde se les usa para acumular documentos, hojas de cálculo y diapositivas de PowerPoint en una sola carpeta con tres argollas. La carpeta entera puede guardarse, visualizarse, imprimirse, copiarse o desplazarse como una sola entidad. DocObjects ahora se reposicionan como contenedores de componentes de OLE móviles para Internet.

DocObjects ofrecen un medio estándar para la conexión de aplicaciones tradicionales a la ventana del visualizador Web de Microsoft. Además, se les prepara también para convertirlos en contenedores de componentes de propósito general. Una página de DocObject puede contener OCX —o ActiveX— ligeros, así como otros tipos de componentes. En contraste con un contenedor simple de documentos compuestos, la carpeta de DocObject sabe cómo manejar paginación, encabezados, secciones al pie, páginas de títulos, visualizaciones y otras formaciones visuales.

El DocObject es un contenedor de documentos compuestos, de modo que ofrece su propio almacenamiento, lo que lo convierte en una unidad de administración de almacenamiento. Para nuestros lectores aficionados a la tecnología de OLE, hemos de apuntar que DocObjects introduce tres nuevas interfaces de OLE: IOleDocument y IOleDocumentView en la parte de componentes (u OLE Server) y IOleDocumentSite en la parte del contenedor.

Microsoft quería ver poblados sus títulos de DocObjects por sus más recientes controles OCX 96, cuya nueva denominación comercial es *componentes de ActiveX*. Estos nuevos controles son OCX de OLE austeros, con menos interfaces de OLE obligatorias por soportar. En términos técnicos, el control debe ser sencillamente un objeto de DCOM autorregistrable, debe implantar las interfaces IUnknown e IClassFactory de OLE y todas las demás que necesite para cumplir sus funciones. Así, se le libera de todo el equipaje extra de OLE. En teoría, el tamaño de estos controles ligeros podría llegar a ser de hasta 1/3 de los OCX actuales, lo que significa que su descarga de Internet podrá ser más veloz.

Microsoft desarrolla también un conjunto estándar de *categorías de componentes* y sus descripciones. Al no requerir de interfaces particulares, los ActiveX son libres de hacer lo que les plazca; Microsoft ha relajado, en efecto, los requerimientos de control de OLE, hasta reducirlos a cero. Esto supone una carga extra para los contenedores de OLE, de los que ahora se espera que descubran dinámicamente las interfaces que soporta un ActiveX. Las nuevas categorías de componentes de OLE brindan un medio para que los controles informen a sus contenedores de lo que hacen. Como era de esperarse, Microsoft define una categoría de *InternetAware*, controles que cargan sus datos vía monikers asíncronos. Cada componente de ActiveX estará representado en el registro de OLE por un *identificador de categoría*.



El efecto lemming

Debate

Microsoft persigue un propósito específico en Internet. Muy específico.

**Bill Gates, presidente
Microsoft
(Diciembre de 1995)**

La estrategia principal de Microsoft con Internet es "extender" a HTML en un estilo semejante a Java para lanzar la infraestructura de Internet basada en OLE propia de la compañía.

**Gartner Group
(Enero de 1996)**

Microsoft se ha mantenido firme en su visión de los componentes. Ha dedicado los últimos años a reformar sus herramientas en torno al ORB de DCOM y la tecnología de componentes de OLE. Ahora aplica toda su tecnología en la creación de un Web de objetos. Todo lo que genere en el Web se basará en su infraestructura de componentes distribuidos.

Por el contrario, parecería que cada uno de los competidores de Microsoft poseyera sólo una pieza del rompecabezas de los objetos, de manera que ninguno las tiene todas. De este modo, hará falta una seria colaboración entre proveedores para que tejan juntos un conjunto coherente de productos y herramientas para el Web de objetos abierto. Como mínimo, deben ser tan congruentes como las propuestas de Microsoft.

Lamentablemente, algunos de los competidores de Microsoft creen llevar la ventaja y que podrán imponerse por si solos. Llamamos a esto el "efecto lemming". Con ello queremos decir que sin un sólido fundamento de objetos, cada competidor caerá al barranco en un punto diferente. Por ejemplo, sin una infraestructura sólida de documentos compuestos, Netscape no podrá contra Microsoft. Lo mismo podemos decir de Java de Sun, Lotus Notes y Oracle. Ninguna de estas tecnologías puede sostenerse en pie por sí sola contra el Web de objetos de Microsoft. Si estas compañías no colaboran en la creación rápida de un Web de objetos abierto, Microsoft está destinada a convertirse en la próxima Microsoft. □

Capítulo 31

Cliente/servidor en el Web: Presentación de participantes



Hace dos años, Internet era una ola emergente. Ahora es un maremoto, y los proveedores están sufriendo para adaptarse a él.

**Jamie Lewis, presidente
Burton Group
(Diciembre de 1995)**

Una vez más, la industria de la computación vuelve a crecerse. Internet es un maremoto gigantesco que no dejará sin tocar nada a su paso. Está redefiniendo la cobertura de cliente/servidor y volviendo a barajar las cartas. Como es de imaginar, Internet generará más que su acción justa de nuevos y fugaces ganadores y perdedores en la industria de la computación. Además, nadie sabe con certeza qué rumbo seguirá el mercado de Internet.

Sería una aberración que pretendiéramos predecir el panorama de cliente/servidor en Internet de los próximos dos años. Por ejemplo, las compañías proveedoras de servidores —como Sun, IBM y Oracle— pueden servirse de Java para reconquistar el escritorio de manos de Microsoft. O bien, Microsoft podría servirse del Web de objetos de OLE para convertirse en la nueva compañía de servidores empresariales, con lo que entonces podría borrar a las ahora existentes compañías de servidores de la faz de la Tierra. En medio de estas dos posibilidades se hallan



■ **Consultoría de Internet.** Incluye a integradores de sistemas tradicionales, de los que se echará mano para la creación de las intranets empresariales. Incluye también a una nueva especie de consultores en Internet; por ejemplo, diseñadores gráficos y agencias de publicidad en línea y creación de imagen en Internet.

Estas seis categorías corresponden casi exactamente a las establecidas por *Hambrecht and Quist Research* en su estudio de clasificación del mercado *The Internet — Webbing The Information Economy* (22 de septiembre de 1995). Así, podemos basarnos en sus proyecciones para describir la dimensión del mercado total de Internet y de cada uno de sus segmentos.

De acuerdo con Hambrecht and Quist, el mercado de Internet representó \$1,180 millones de dólares en ventas en 1995; en la Figura 31-2 se muestra la división del pastel entre cada una de las seis categorías del mercado. Todo indica que quienes están haciendo más dinero por ahora son los proveedores de hardware y los ISP. A pesar de tantas turbulencias, el software cliente/servidor en el Web es apenas una lucecita de \$260 millones de dólares en la pantalla del radar de Internet. Cabe indicar que en el estudio de referencia no se tomaron en cuenta los \$1,000 millones de dólares gastados por las empresas en el desarrollo de sus páginas Web en 1995; parece que la creación de páginas Web corresponde a la categoría "por amor a la camiseta".

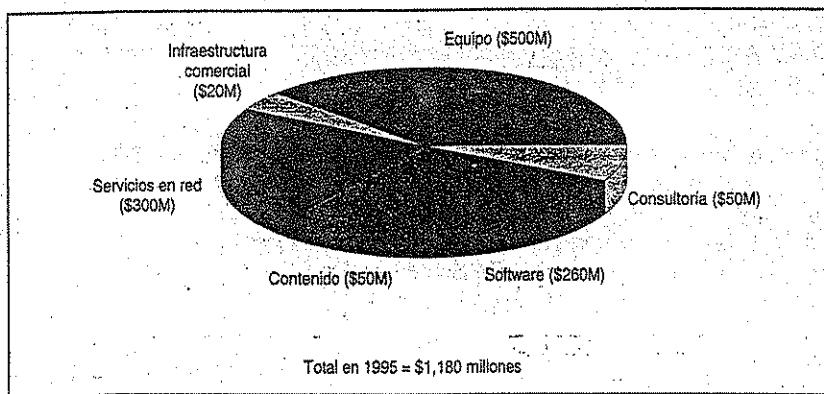


Figura 31-2. El mercado de Internet en 1995. (Fuente: Hambrecht and Quist)

Hambrecht and Quist pronostica que el mercado de Internet representará en el año 2000 los \$23,200 millones de dólares (véase Figura 31-3). El mayor crecimiento corresponderá al contenido de Internet; es de esperar que aumente de los actuales \$100 millones de dólares a más de \$10,000 millones para fines de la década. La segunda área de crecimiento será la de servicios en red; se espera que se eleve a \$5,000 millones de dólares en el año 2000, lo que significaría una tasa compuesta de crecimiento anual de 76%. Finalmente, también al mercado de software cliente/servidor en Internet le espera buen tiempo; se prevé que aumente a \$5,000 millones de dólares, a una tasa compuesta de crecimiento anual de 73%.

Las cifras de los pronósticos de Hambrecht and Quist explican el motivo de que la comunidad inversionista esté cerrando tan estrecha fila detrás de Internet; sus miembros suponen que se tratará de un negocio por \$23,200 millones de dólares en el año 2000. Las tasas de crecimiento

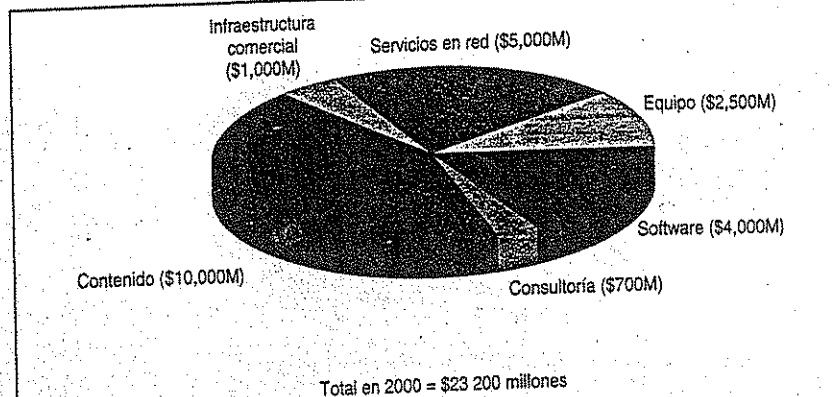


Figura 31-3. Proyección del mercado de Internet para el año 2000. (Fuente: Hambrecht and Quist)

proyectadas son astronómicas. Sin embargo, las cifras parecen coincidir con el extraordinario valor que el mercado accionario atribuye a todo lo que tiene que ver con Internet.

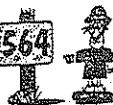
¿Estallará la burbuja? Lo ignoramos. Algunos expertos han predicho la "muerte de Internet" casi cada año desde mediados de la década de los setenta. Y ciertamente tal cosa no ha ocurrido. Coincidimos plenamente con los supuestos en los que se basan los pronósticos de Hambrecht and Quist. En todo caso, quizás sean un tanto conservadores. Creemos también que muchas de las grandes estrellas actuales de Internet se habrán eclipsado para el año 2000. Las revoluciones tienden a engendrar sus propias criaturas; Internet no será la excepción.

TENDENCIAS

Cliente/servidor es un sistema total. Se puede ajustar el centro de gravedad y decir: "Me olvidaré de los clientes y ganaré con los servidores", o "Apostaré a los clientes y renunciaré a los servidores".

Jim Clark, presidente
Netscape
(Octubre de 1995)

En el estudio de Hambrecht y Quist se identifica al correo electrónico como la aplicación predominante en la Internet de hoy (véase Figura 31-4). Se pronostica también que el Web tendrá 200 millones de usuarios en el año 2000, cifra verdaderamente fenomenal. Si tuviéramos que apostar, diríamos que el correo electrónico será absorbido por el Web mucho antes del año 2000; será sencillamente otra aplicación del Web. Así pues, lo que se impone preguntar es: ¿cómo ocurrirá el crecimiento del Web hasta abarcar a 200 millones de usuarios? ¿Cómo será un Web con 200 millones de usuarios? ¿Quién nos trasladará allá? ¿Cómo?



El Web liberará sin duda un torrente de actividad empresarial en torno a los componentes para cliente/servidor. Este nuevo modelo de distribución de software acelerará el viraje hacia los componentes y los applets; atraerá a millones de pequeños desarrolladores. En poco tiempo, estos desarrolladores ofrecerán los "applets arrasadores" que llevarán el Web a las masas. Veremos applets arrasadores en compras domésticas, banca electrónica, recreación, educación y muchas actividades aún por inventar.

Recuérdese que la cifra mágica es 200 millones de usuarios para el año 2000. El Web cuenta ya con 20 millones de usuarios. Muy pronto se les unirán 100 millones de personas con PC donde entran en juego las PC de \$500 dólares para el Web; éstas serán las computadoras volkswagen de bajo precio que abrirán el Web para las masas. Si lo conseguimos, éstas llegarán gus- tosas.

Pero, ¿qué obstáculos se perciben en este paisaje color de rosa? Tres vienen a la mente de inmediato: ancho de banda, más ancho de banda y un millón de applets. Es obvio que el Web no podrá convertirse en una utilería de información para las masas sin un abundante ancho de banda de bajo costo. El destino final del Web de cliente/servidor puede depender de la tasa de adopción de tecnologías como ISDN, ATM y módems para cable. Además del ancho de banda, el Web debe volver a crear toda la computación de cliente/servidor con componentes móviles. Ésta será evidentemente una labor colosal; se necesitará de millones de años Java de programadores para cumplirla. En suma, el Web precisa de abundancia de ancho de banda y applets para mantener su actual ritmo de desarrollo.

CONOZCA A LOS PARTICIPANTES

Es una guerra, una guerra mundial.

Mary McCaffrey, Internet World
(Marzo de 1996)

Miles de pequeñas empresas iniciales harán fortunas ofreciendo nuevo contenido, software de propósito especial, infobases e innovadores servicios en el Web. Además, algunos millones de personas se ganarán decorosamente la vida en el Web vendiendo accesorios, componentes, applets y servicios; será un gigantesco tianguis en línea. Así, hay muchas fortunas por hacer en el nivel de aplicaciones. Sin embargo, preveremos que la parte del león de los ingresos en middleware cliente/servidor para el Web irá a dar a los bolsillos de quienes ya se imagina: Netscape, Sun, Microsoft, Lotus/IBM, Apple y Oracle.

Netscape y Apple tienen ideas. Los demás también, pero junto con bolsillos llenos además de una sólida base tecnológica de cliente/servidor sobre la cual apoyarse. Los bolsillos llenos son muy importantes, porque buena parte de la infraestructura de cliente/servidor en el Web ya no



sirve. Aunque muchas de las compañías que se arremolinan en torno al Web esperan que todo les salga gratis, sabemos que lo "gratis" no es eterno. Alguien tendrá que pagar en algún momento todas las chucherías. En el caso del Web, el dinero procede de dos fuentes: 1) el mercado accionario y 2) las grandes empresas que adquieran tecnología de intranets e Internet. El mercado accionario es veleidoso; seguirá invirtiendo en Internet siempre y cuando vea ganancias al final del túnel. Así, sólo quedan las intranets empresariales. A corto plazo, serán las grandes empresas las que paguen la infraestructura del Web. A largo plazo, el Web se pagará solo al transformarse en el mayor centro comercial del mundo.

Ya no basta lo "normal"

Lo interesante del asunto es que se espera que las grandes empresas gasten más de \$5,000 millones de dólares en intranets en 1997. Por supuesto que, en su mayoría, no parten de cero en cuanto a la tecnología de cliente/servidor, lo que significa que esperarán que el Web se adapte a sus actuales infraestructuras. Esto quiere decir a su vez que la tecnología del Web debe ser capaz de lanzar interfaces inconsútiles con bases de datos, monitores de TP, NOS, groupware y plataformas de administración de sistemas ya en uso. Las empresas también esperan que sus sistemas sean robustos y decisivos para el cumplimiento de objetivos. Esto quiere decir que los participantes en la esfera del Web con las tecnologías de cliente/servidor más maduras serán quienes tengan mayores posibilidades de ganar. Ya no basta lo "normal".

Alguien dijo una vez que recorrer las propuestas de productos de cliente/servidor para Internet era como tratar de describir a un tren bala en marcha. Este mercado es tan intensamente competitivo que las estrategias y alianzas de proveedores cambian casi todos los días. Para obtener la información más reciente, sintonice el noticiero matutino o lea los periódicos. En las siguientes secciones haremos una breve descripción de lo que ofrecen hoy en día los principales proveedores de cliente/servidor para el Web.

Netscape

Quizá 1995 sea considerado en retrospectiva como el fin de la era de Windows y el principio de la era de las redes, ejemplificada por Netscape... Netscape pretende hacer de Navigator tanto el sistema operativo de facto como el "juego de oficina" para el ciberspacio.

Peter Lewis, New York Times
(Noviembre de 1995)

Netscape es una de las estrellas brillantes de Internet. En menos de un año, pasó de la nada a una compañía valuada en \$2,000 millones de dólares. Domina actualmente el mercado de visualizadores; su *Navigator 2.0* fue el primero en soportar applets de Java, seguridad de SSL, marcos y conexiones. Navigator 2.0 es más que un visualizador, es toda una serie para el cliente en Internet. Maneja correo electrónico, grupos de discusión hilados, FTP, intercambios,

de servidores muy fuerte, que incluye objetos del servidor multihilos, un ORB de CORBA ampliable, herramientas de desarrollo de objetos de negocios y la plataforma de administración de sistemas Solstice. Hace uso de las bibliotecas de clase de Persistence para brindar interfaces de objetos con los principales DBMS y ODBMS.

La posición de Sun como proveedor de hardware supone tanto ventajas como inconvenientes. En el aspecto positivo, Sun está en condiciones de proporcionar una línea completa de combinaciones de hardware/software para Internet integradas, como máquinas con Java de bajo costo, estaciones de trabajo de gráficos tridimensionales, enruteadores, muros de protección (*firewalls*) y superservidores.

En el aspecto negativo, a Sun se le percibe como proveedor de hardware principalmente Solaris, lo cual no proyecta la imagen de proveedor de soluciones de software entre plataformas. La creación de JavaSoft como subsidiaria de software independiente puede contribuir a modificar esta percepción, así como a generar para Sun altos ingresos en software por ventas de componentes de Java de plataformas múltiples.

No obstante, otra de las debilidades de Sun es la carencia de una arquitectura de documentos compuestos y ubicaciones embarcables. Esto desmerece considerablemente su posición como proveedor de componentes independiente. Frente a la embestida comercial de OLE, los documentos compuestos se han vuelto parte importante del sustrato del middleware; sencillamente ya no se les puede ignorar. Sun podría adquirir esta tecnología —sobre una plataforma de CORBA— sumándose al campamento de OpenDoc.

Microsoft

Microsoft tiene algo de lo que todos los demás carecen: puede regalarlo todo en los próximos 20 años.

Eric Schmidt, director técnico
Sun
(Febrero de 1996)

Hasta diciembre de 1995, parecía que Microsoft sufría de una inusual falta de claridad en su estrategia hacia Internet. Quizá esperaba que MSN sustituyera a Internet de golpe. Por supuesto que esto no sucedió; MSN fue un fracaso comercial para los estándares de Microsoft. Como resultado de ello, esta compañía decidió que no podía seguir ignorando a Internet. Su nueva estrategia de caballo de Troya parece consistir en aceptar a Internet y sus estándares para adecuarlos después a OLE. Puede lograrlo aportando cientos de extensiones propietarias.

En marzo de 1996, Microsoft anunció ActiveX y su nueva arremetida de Internet de OLE. Esta vez, cientos de ISV se montaron en el tren de Internet de Microsoft. Como explicamos en el capítulo anterior, Microsoft dispone de una muy consistente versión del Web de objetos. Si el otro multitudinario bando en Internet no consigue pronto el funcionamiento conjunto de su Web de objetos, el Web sencillamente se convertirá a OLE.



¿Exageramos? No. Tal como lo señala Eric Schmidt, de Sun, Microsoft es dueño de los bolsillos más prósperos de la industria. Puede darse el lujo de integrar lo que se le antoje ya sea a Windows 95 en el cliente o a NT en el servidor. La integración de IIS con NT lo convirtió de inmediato en relevante participante en el mercado de servidores Web. Su *Internet Explorer 3.X* se integrará totalmente a Windows 97. Además, fue capaz de obligar tanto a CompuServe como a America Online a integrar Explorer en sus paquetes. Toda esta integración gratuita puede recompensar las inversiones de Netscape en la parte del cliente.

Microsoft planea integrar IIS con su serie *BackOffice*, que incluye a Exchange, SQL Server y su ya próximo monitor de TP basado en OLE. A fines de 1996 tenía proyectado introducir un servidor proxy (o muro de protección), un servidor comercial y un servidor de medios en BackOffice. MSN está siendo repositionado como servicio Web "de pago". Además, las aplicaciones *Office* de Microsoft —que incluyen a Excel, Word y PowerPoint— serán habilitadas para Internet con adiciones que les permitirán interactuar directamente con IIS. Microsoft también está integrando el *Internet Assistant* —herramienta de publicación de HTML de nivel básico— con Word. Finalmente, soporta asimismo a Internet a través de una impresionante familia de herramientas, entre cuyos miembros están *Internet Studio*, *Visual Basic*, *Visual C++*, *Jakarta* y la recientemente adquirida *Vermeer*; estas herramientas fueron descritas en el capítulo anterior.

La cuestión básica es que Microsoft es un formidable participante en Internet, especialmente para sectores de sólo Windows. Aporta a la ecuación una versión completa del Web de objetos. Su mayor defecto es que esta versión no opera adecuadamente en plataformas que no sean de Windows. El Web es demasiado diverso como para atenerse exclusivamente a Windows. Así, éste es el talón de Aquiles de Microsoft. Sin embargo, para las multitudes para las cuales Windows "es todo", esto quizás ni siquiera parezca una desventaja. Para ellas, Internet será sencillamente otra extensión de Windows con estándares establecidos por Microsoft.

Apple

Nuestro propósito es volver a Internet tan amable como Mac lo es ya:

Larry Tessler, director científico
Apple
(Mayo de 1996)

Los usuarios de Mac tienen una presencia desproporcionadamente fuerte en el Web; quizás representen casi el 20% de la población cliente del Web. Aunque la Mac actual no es la mejor plataforma de servidor Web del mundo, es sumamente amigable y cortés; por ejemplo, el *Workgroup Server* de Apple es un buen servidor Web para pequeñas empresas que no esperan miles de movimientos al día. Apple puede convertirse en gran participante en servidores Web cuando, en 1997, lance su OS de Mac basado en micronúcleo, cuyo nombre de código es *Copland*. Mientras tanto, es el software cliente el que le da brillo a la Mac en el Web.



Parte 8. Cliente/servidor e Internet

El método que sigue Oracle en relación con Internet se centra casi exclusivamente en bases de datos. WebServer ofrece seguridad de extremo a extremo integrando SSL y S-HTTP a la seguridad de la base de datos. En vez de soportar CGI, WebServer hace uso de una API propietaria llamada *Oracle Web Request Broker*. Esta API enlaza directamente con los servicios de base de datos de Oracle; mantiene información de estado para que los clientes no tengan que conectarse al DBMS cada vez que emiten una solicitud de SQL. WebServer brinda asimismo un entorno de tiempo de ejecución de Java integrado, con extensiones para Oracle7; la idea es ofrecer la posibilidad de crear applets de Java para la parte del servidor. WebServer correrá en un tiempo más en casi todas las plataformas de servidor. Sus primeras versiones corren en NT y Solaris; su costo es de \$2,945 dólares.

Con ventas por \$3,500 millones de dólares, Oracle es la segunda compañía de software más importante del mundo. Podría convertirse en un importante proveedor de software servidor entre plataformas para intranets, cosa que sabe hacer bien. En el lado negativo, carece de una estrategia clara en cuanto al Web de objetos. Cuenta con una excelente herramienta para componentes, *PowerObjects*, pero no con una infraestructura de objetos distribuidos o documentos compuestos. Con una aceptable versión del Web de objetos, Oracle podría ser un extraordinario proveedor de software para intranets en el Web.

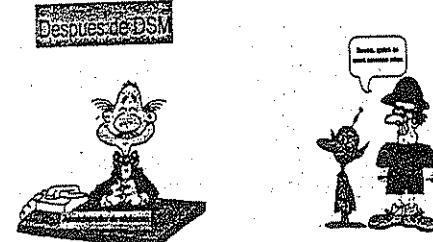
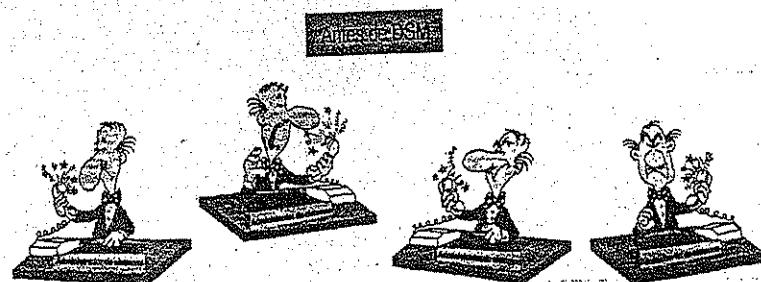
CONCLUSIÓN

Es mayor cada día el número de personas en la red que el de quienes ven CNN. La cantidad de hogares con acceso a la Red es semejante al número de televisores en blanco y negro de 1950, de modo que esperamos la misma explosión de crecimiento.

Dennis Tsu, director de Internet
SunSoft
(Marzo de 1996)

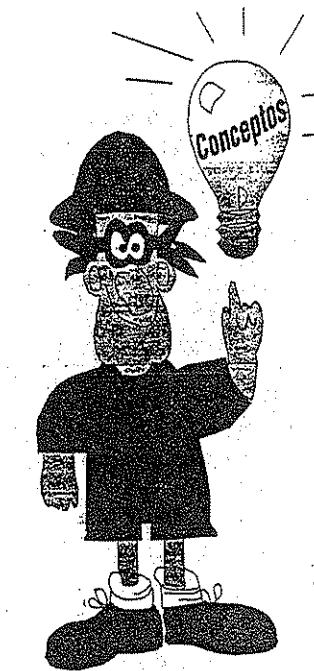
Para seguir con la analogía de Dennis Tsu, si el Web con HTTP actual es televisión en blanco y negro, entonces el Web de objetos es televisión a color. Hemos comenzado ya a ver parte de esos colores en Java, pero lo mejor está aún por venir. Será responsabilidad del Web de objetos crear el tipo de contenido queatraiga a las masas a Internet; también hará maravillas para los innumerables usuarios de intranets. ¿Esto terminará alguna vez? No. Le dijimos que la vida en el Web se mide en "años animales". ¿Ya entendió por qué?

Parte 9 Administración de sistemas distribuidos



Capítulo 32

Administración de sistemas distribuidos de cliente/servidor



Ahora que mis aplicaciones han abandonado la macrocomputadora, ¿cómo voy a administrar este desorden?

Gerente de Sistemas anónimo

La sola existencia de un problema no garantiza que haya solución.

Proverbio yiddish

Cliente/servidor ha sido la guillotina de los sistemas centralizados, a los cuales ha dividido en pequeñas piezas y esparcido a todo lo largo de las redes. Tras ello hicieron su aparición los "sistemas abiertos", que cercenaron aún más el software, de manera que cada pieza provenga de un proveedor diferente. ¿Cómo administrar estas piezas? Los temas de la administración y soporte son el talón de Aquiles de la computación de cliente/servidor. Aunque los beneficios de los sistemas de cliente/servidor son indiscutibles —reducción de costos de hardware y software, mayor flexibilidad en los sistemas, mayor facilidad de uso de planos frontales—, lo cierto es que ahora enfrentamos el hecho de que la administración y el soporte son mucho más costosos que en el caso de los sistemas centralizados. La computación de cliente/servidor dispersa aplicaciones entre múltiples sistemas en una red, y genera, por lo tanto, problemas temibles

variedad de proveedores, redes, paquetes de software y configuraciones de sistemas explica que cada sistema de cliente/servidor sea diferente al inmediatamente anterior. Además, apenas en fecha muy reciente, los proveedores procedieron al ajuste de sus herramientas de administración para volverlas acordes con su propio surtido de productos y adecuarlas en términos generales al nivel de grupos de trabajo o departamental.

Como resultado de ello, los centros de control de cliente/servidor de las grandes organizaciones efectivamente semejan el tablero de mando de la nave espacial *Enterprise*. ¿Quién se encarga de la correlación entre las diferentes plataformas de administración? Los operadores, desde luego. Éstos deben dominar diferentes grupos de comandos para cada interfaz del usuario y producto. Y a cada momento deben correlacionar información —sobre fallas, configuración y desempeño, por ejemplo— de las distintas consolas de administración. Dado que cada sistema cuenta con su propia base de datos de administración, no es raro que la misma información sea tecleada y vuelta a teclear innumerables veces. Un mismo error puede provocar la aparición de cientos de mensajes distintos en diferentes consolas al mismo tiempo. En muchos casos se corre el riesgo de superposición de funciones, lo que produce aún mayor confusión. Incluso en configuraciones limitadas y con la intervención de unos cuantos proveedores, la administración de la primera generación de sistemas de cliente/servidor resultó siempre sumamente compleja.

VIVIR EN EL CAOS Y APRENDER A QUERERLO

A veces la respuesta de las preguntas más difíciles es la más simple: las aplicaciones distribuidas necesitan una administración distribuida.

**James Herman, vicepresidente
Northeast Consulting Resources, Inc.**

A principios de la década de los noventa, la industria empezó a percatarse que la multiplicación de herramientas incompletas por parte de proveedores múltiples no hacía sino generar un desorden inmanejable que amenazaba con descarrilar todo el movimiento de cliente/servidor. Había que hacer algo, y pronto; la industria enfrentó el reto e introdujo velozmente muchas innovaciones. En los últimos años han surgido, en rápida sucesión, tres generaciones de arquitecturas administrativas: las plataformas de *administrador de administradores*, *administración de sistemas distribuidos (DSM)* y *DSM abierta* (véase Figura 32-1). Nos ocuparemos de estas arquitecturas en el resto de esta sección.

Administrador de administradores

NetView de IBM fue uno de los primeros intentos por resolver los problemas de la administración de sistemas distribuidos a nivel empresarial. Introdujo el concepto de *administrador de administradores*, que hizo posible una “macrocomputadora en las alturas” para la supervisión de una empresa en su totalidad. La idea fue crear una imagen de sistema único mediante una jerarquía de tres niveles de elementos administrativos: *puntos de entrada* de nivel inferior para

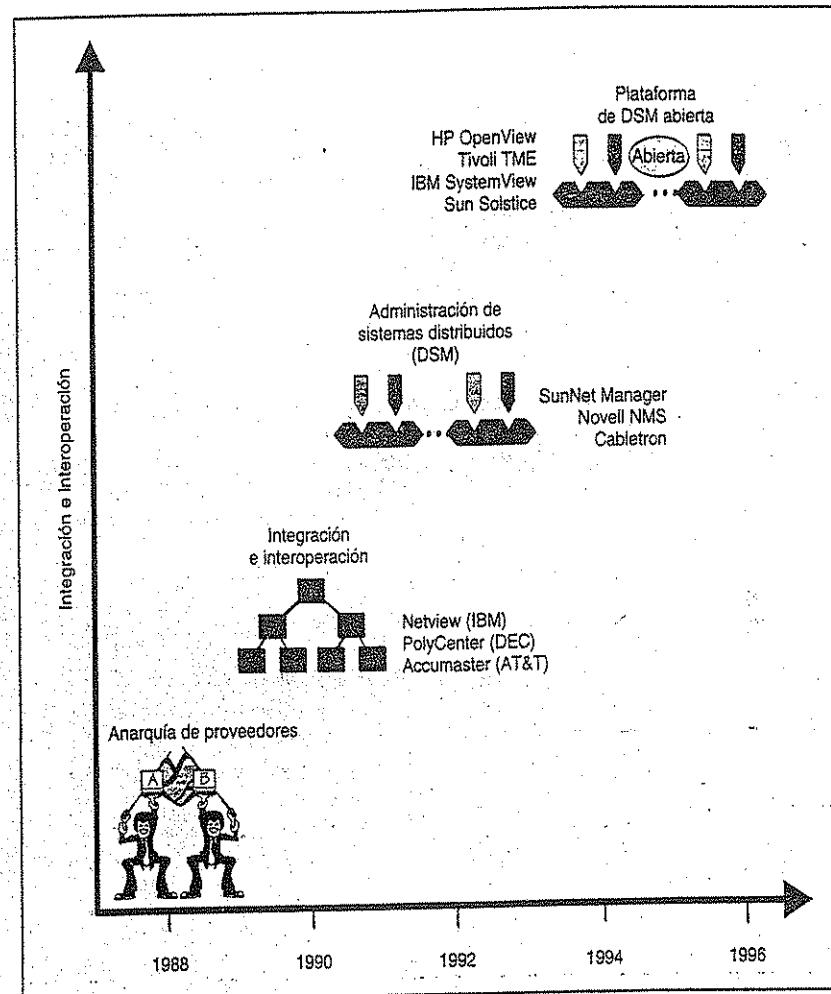


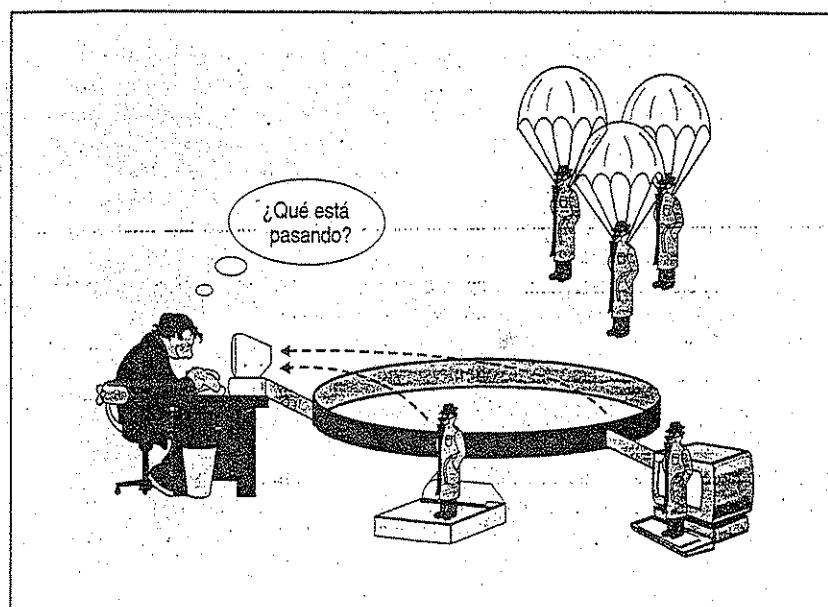
Figura 32-1. Tres generaciones de arquitecturas de administración de sistemas distribuidos.

la recolección de información administrativa y su envío en dirección ascendente; *puntos de servicio* de mandos medios para proceder sobre una parte de la información y enviar el resto en dirección ascendente, y *puntos focales* en la cima para el mantenimiento de una base de datos central de información administrativa y la presentación de una visualización uniforme de todos los recursos distribuidos. NetView/PC —el primer producto de *punto de servicio* lanzado por IBM— también hacía las veces de gateway para la conversión de información administrativa procedente de dispositivos no SNA a un formato comprensible para los puntos focales de la macrocomputadora por medio del *transporte de vectores de administración de redes (NMVT: network management vector transport)* de SNA.

administradas de la red e informan sobre su estado. La estación de administración puede hacer que sus agentes de software "desciendan en paracaídas" en cualquier punto de la red para velar por sus intereses y recabar los datos que necesita. Los agentes también son capaces de ejecutar comandos en su nombre.

Los agentes contribuyen a la creación de un "mundo reflejado" del universo de cliente/servidor por administrar. Cada agente vigila una porción del universo. Miles de agentes podrían ser lanzados en paracaídas a todo lo largo de la red. Todos ellos operan en forma simultánea, nunca descansan, están siempre alerta para detectar todo aquello que pueda marchar mal y no cesan de recabar datos que puedan ser de utilidad para un administrador. El software administrador clasifica estas enormes cantidades de información en tiempo real en busca de material sólido, las tendencias y patrones que puedan surgir. Se encarga de que el operador no se vea abrumado de datos. Para lograrlo, debe crear un modelo de datos de la "carga de realidad" que administra. Las "refinerías de datos" deben convertir los datos en información útil. Una estación de trabajo de administración debe incluir una base de datos relacional o de objetos para administrar y organizar la información recopilada y recordar sucesos del pasado.

Pero un mundo reflejado no se reduce a un servicio de información; es una *ubicación*. Se puede deambular por él. Para permitirlo, la estación de administración se sirve de imágenes icónicas para crear representaciones computacionales visuales de "que está pasando allá afuera". Múltiples vistas permiten amplificar, obtener visiones panorámicas y vagar por la red. La pantalla puede ser leída como un tablero de instrumentos cuando se necesita consultar el estado de un sistema de un solo golpe de vista, o bien es posible recorrer grandes cantidades de información organizada en muchas vistas si es necesario una labor detectivesca más detenida. En cualquier



nivel, la pantalla está activa; cambia para reflejar las cambiantes condiciones del sistema en todo momento. La forma en la que las cosas se presentan es un aspecto muy importante de la administración de sistemas. En términos ideales, usted debería ver y controlar todos los detalles de un sistema administrado desde una estación de trabajo de administración única. La apariencia visual e impresión general de la administración de sistemas son semejantes a las de los videojuegos, pero en segundo plano trata con agentes reales que controlan colectivamente cada aspecto de un sistema distribuido.

Una plataforma de administración brinda también el middleware necesario para la comunicación con todo tipo de agentes, desde agentes simples encargados de dispositivos de hardware (como enruteadores), hasta agentes complejos que persiguen objetos que vagan por la red. Los agentes simples se pueden localizar por medio de protocolos simples, como SNMP; los agentes complejos pueden ser invocados a través de un ORB. En el punto intermedio disponemos de agentes tradicionales para aplicaciones cliente/servidor, los cuales se comunican por medio de RPC y MOM. En suma, la administración de sistemas distribuidos emplea tecnología de cliente/servidor para administrar sistemas de cliente/servidor. Todo es absolutamente recurrente. Los mundos reflejados son ejemplo del uso que se le dará a la tecnología de cliente/servidor en los próximos años. Daremos un paso enorme si entendemos el funcionamiento del mundo reflejado de la administración de sistemas.

COMPONENTES DE UNA PLATAFORMA DE DSM ABIERTA

En los últimos años, las organizaciones de IT han implementado soluciones fragmentarias de DSM.

Meta Group
(Septiembre de 1995)

En la Figura 32-2 se muestran los principales componentes de una plataforma de administración abierta. Se trata de una combinación de los modelos conceptuales de administración distribuida de OSI, OSF-DME, UI-Atlas DM y SystemView de IBM. Pero, sobre todo, esta figura es también un modelo combinado representativo de plataformas comerciales de administración abierta como OpenView de HP, SystemView de IBM y Tivoli. Antes de entrar en detalle, presentaremos brevemente los componentes administrativos de una plataforma de administración abierta. Comenzaremos por la parte superior de la Figura 32-2 y seguiremos una dirección descendente:

- La *interfaz del usuario OOU* ofrece representaciones visuales de los objetos administrados. La estación de trabajo de administración debe descubrir automáticamente la topología de agentes de una red y exhibirlos después en una vista general. Se puede hacer uso de mapas de segundo plano para indicar la ubicación geográfica de los agentes. Al hacer clic sobre un ícono que representa un objeto administrado, aparece una vista de su estado del momento y de opciones para la observación y control de éste. En teoría debería ser posible definir visualmente combinaciones de evento/acción. Los cuadros de diálogo de consulta

APLICACIONES DE ADMINISTRACIÓN: EL MANEJO DE LA DIVERSIDAD Y LA COMPLEJIDAD

Con API estandarizadas y la previsible convergencia en un reducido número de plataformas de administración, surgirá un mercado completamente nuevo. Se abre así una gran oportunidad para el desarrollo de aplicaciones de administración de excelencia.

Gartner Group

Las aplicaciones de administración llevan a cabo las operaciones reales de la administración de sistemas distribuidos. Ellas nos permiten responder las siguientes preguntas: ¿Qué tan alto es el grado de desempeño de mi sistema de cliente/servidor? ¿Qué ocurre más allá, y dónde exactamente? ¿Quién está haciendo qué para quién? ¿Cómo instalar nuevo software? ¿Qué falló? ¿Cómo repararlo? ¿Todo está siendo objeto de respaldo? ¿Resistirá mi sistema un sismo de 8.0 grados? En esta sección pasaremos breve revista a las categorías de software de administración de sistemas que contribuyen a la resolución de estas preguntas.

¿Qué tan alto es el grado de desempeño de mi sistema de cliente/servidor?

Reunir información de tiempo real de varios componentes del sistema en un entorno de cliente/servidor e identificar las causas de problemas de desempeño puede representar verdaderos dolores de cabeza para administradores de sistemas y bases de datos. El software de administración debe ser capaz de recolectar abundante información, en tiempo real, de diferentes fuentes en un entorno de cliente/servidor. Luego debe proceder sobre esos datos, o presentarlos en forma gráfica o numérica para su análisis. Lo ideal sería que usted pudiera correlacionar los datos de desempeño con información obtenida de las herramientas de administración de configuración para identificar proactivamente cuellos de botella del sistema.

Las *herramientas de monitoreo de desempeño* reúnen datos estadísticos sobre niveles de utilización de recursos de componentes clave de un sistema de cliente/servidor, para más tarde generar alarmas en caso de que no se cumplan los criterios definidos por el administrador. Algunas de las herramientas pueden brindar incluso facilidades de creación de scripts para la generación de acciones de reparación o la realización de mantenimiento preventivo. Estas herramientas vigilan los niveles de utilización de recursos usuales, como redes, CPU, discos, memoria, espacio de archivos, procesos, transacciones del servidor, correo electrónico, módems y enruteadores. Los mejores sistemas recurren a histogramas y desviaciones estándar para monitorear el tiempo de respuesta de los servidores. Las herramientas de monitoreo de desempeño mantienen una base de datos sobre el desempeño histórico del sistema a fin de permitir la detección de tendencias. Se pueden obtener representaciones gráficas de datos de uso de cualquier recurso, trátese de una computadora, red o base de datos. La herramienta también debería recopilar automáticamente datos relevantes de cierto periodo, a fin de que usted pueda analizar la utilización de recursos durante horas pico antes de tomar decisiones de inversión de capital en hardware.

En 1996 comenzaron a aparecer *herramientas de monitoreo de desempeño de aplicaciones*. Las primeras de ellas se han concentrado en la vigilancia y ajuste preciso de DBMS. La DMI y



la MIB de aplicaciones estandarizan los parámetros que una aplicación necesita para comunicarse con una herramienta o agente de administración. Estos formatos estándar son justo lo que se necesitaba para el despegue de esta área de administración de sistemas. Lamentablemente, para la automatización de acciones correctivas en el área de ajustes de desempeño pueden usarse apenas unas cuantas herramientas genéricas. De lo que se precisa en este caso es de herramientas que automaten el equilibrio de cargas en respuesta a variaciones en las cargas. Parte de esta labor la efectúan componentes especializados del sistema, como los monitores de TP y los enruteadores.

¿Qué ocurre más allá, y dónde exactamente?

Las *herramientas de administración de inventarios*, también llamadas "administradores de activos", llevan un registro de los programas que corren en cada máquina, los niveles de software que operan, etc. Registran asimismo el inventario de hardware y mantienen una base de datos de información de inventarios. En su mayoría, estas herramientas cuentan con la capacidad de notificar automáticamente la detección de cambios en el hardware y software de la LAN. Algunas de las herramientas más "autoritarias" hacen más que únicamente reportar cambios; pueden revocar y restaurar archivos automáticamente de acuerdo con los parámetros fijados por un administrador, como en el caso, por ejemplo, de archivos de acceso.

En teoría, un administrador de inventarios automatizado debería estar en condiciones de determinar automáticamente la totalidad de software, estaciones de trabajo, periféricos, enruteadores y servidores directamente adscritos a la LAN. Desafortunadamente, en la mayoría de los casos es necesario que usted introduzca ciertos datos; no hay manera de reunir automáticamente información como el número de serie de los dispositivos. Además, resulta difícil llevar un seguimiento de todo el software empleado en la LAN, ya que no se cuenta con una sola lista íntegra de software. Los proveedores suelen depender en este caso de la lista de la Software Publisher Association (SPA). Esta lista se renueva dos veces al año y sólo contiene el software de proveedores que pagan por la inclusión de sus aplicaciones en ella. Algunas herramientas pueden pasar por alto software desconocido, pues aunque verifiquen el tamaño del archivo y número de versión, se limitarán a hacer suposiciones basadas en el tipo de aplicación de que se trata.

Las *herramientas de administración de configuración* pueden fijar parámetros de software y realizar ajustes precisos de sistemas complejos, como bases de datos relacionales o sistemas operativos. Por fortuna, cada vez se dispone de mayor número de herramientas genéricas de administración de software. Aun así, en términos de control de configuración la gente sigue pensando sobre todo en herramientas de configuración de hardware en existencia desde hace un buen tiempo. Estas herramientas pueden determinar umbrales y parámetros de ajuste de dispositivos. Asimismo, pueden reunir información de configuración de cualquier sistema administrado a través de su MIB. Usted puede rastrear cambios en la configuración del entorno de cliente/servidor en el curso del tiempo y emplear esta información para la realización de actividades cruciales de mantenimiento y ajuste, como la identificación y equilibrio de cargas en torno a cuellos de botella. Por ejemplo, una herramienta de administración de base de datos debe vigilar la utilización de recursos de la base de datos, la fragmentación del espacio de la base de datos, los bloqueos de aplicaciones, el rendimiento de aplicaciones, los recursos de la computadora, los procesos muertos y los puntos de consumo excesivo de recursos.

Casi todas las herramientas desempeñan una labor aceptable de reunión de datos en tiempo real sobre el estado de dispositivos y conexiones, pero dejan mucho que desechar en tareas como análisis de tendencias, registro de fallas y reporte sobre elementos de la red. El software de administración debería mantener una vigilancia permanente sobre el sistema para la detección de problemas potenciales, y emprender automáticamente acciones preventivas o correctivas para resolver problemas antes de que ocurran. En lugar de depender de la *autopsia* de problemas y de acciones reparadoras posteriores a los hechos, lo deseable sería evitar problemas potenciales sin necesidad de interacción humana. Esto haría menos tediosa la administración de sistemas de cliente/servidor, lo mismo que menos susceptible a errores humanos.

¿Resistiría mi sistema un sismo de 8.0 grados?

Ya no se trata sólo de la recuperación de desastres. El respaldo de redes está pasando a formar parte de una estrategia de administración de almacenamiento en la que se prescinde del eslabón más débil: usted.

Michael Peterson, PC Magazine

Como los datos en una red crecen continuamente, el administrador tiene otro fuerte dolor de cabeza: ¿cómo administrar este crecimiento dentro de los límites del presupuesto asignado al tiempo que se protege a los datos contra desastres? La recuperación de desastres lo incluye todo, desde la planeación de respaldos diarios hasta el mantenimiento de instalaciones de reserva. Las *herramientas de recuperación de desastres; respaldo y archivo* pueden poner en marcha el respaldo o restaurar la acción cuando la fuente y destino de la operación pueden localizarse en cualquier punto de la red. Estas herramientas ofrecen una interfaz del usuario para la planeación de operaciones en modo imperceptible. El respaldo puede efectuarse sobre cualquier tipo de medio. Las herramientas más avanzadas cuentan con *administración de almacenamiento jerárquico* para múltiples niveles de almacenamiento, como memoria caché, servidores de archivos y medios de archivamiento como máquina de discos ópticos o unidades de cinta. La herramienta debe permitir el monitoreo de los puntos donde residen todos los datos y la fijación de políticas como plazos de respaldo. Las mejores herramientas poseen la capacidad de discriminar patrones de acceso; tienen cierto entendimiento de cómo se relaciona la información.

Capítulo 33

Estándares de administración de sistemas distribuidos



A medida que nos acercamos a esta nueva relación de cliente/servidor distribuido, la pregunta es: ¿tendremos que disponer de un administrador en cada ubicación? A unos \$100,000 dólares al año por persona, sería muy costoso.

Gary Falksen, DBA with XES Inc.

En este capítulo presentaremos una docena de estándares de administración distribuida y un puñado de siglas nuevas. Abordaremos los estándares tradicionales de administrador/agente, como SNMP, SNMPv2, RMON, RMON-2, XMP y XOM. Nos ocuparemos también de un estándar para pequeños agentes de escritorio, el DMI de DMTF. Concluiremos, en fin, con los estándares de administración de sistemas basados en CORBA, como Tivoli y DME. Incluirímos por supuesto el recuadro de debate de rigor para expresar *nuestra* opinión sobre la administración de sistemas en los entornos de cliente/servidor. ¿Qué sería de nosotros sin recuadros de debate?

Hasta ahora se han definido dos protocolos de administración de redes "estándar": SNMP de Internet y CMIP de OSI. Ambos son protocolos administrador/agente. Ambos son capaces también de describir información de administración. Tienen mucho en común (incluida una similitud confusa en terminología), pero difieren en aspectos importantes. El método de SNMP es simple y directo, mientras que el de CMIP es tanto más potente cuanto más complejo. El

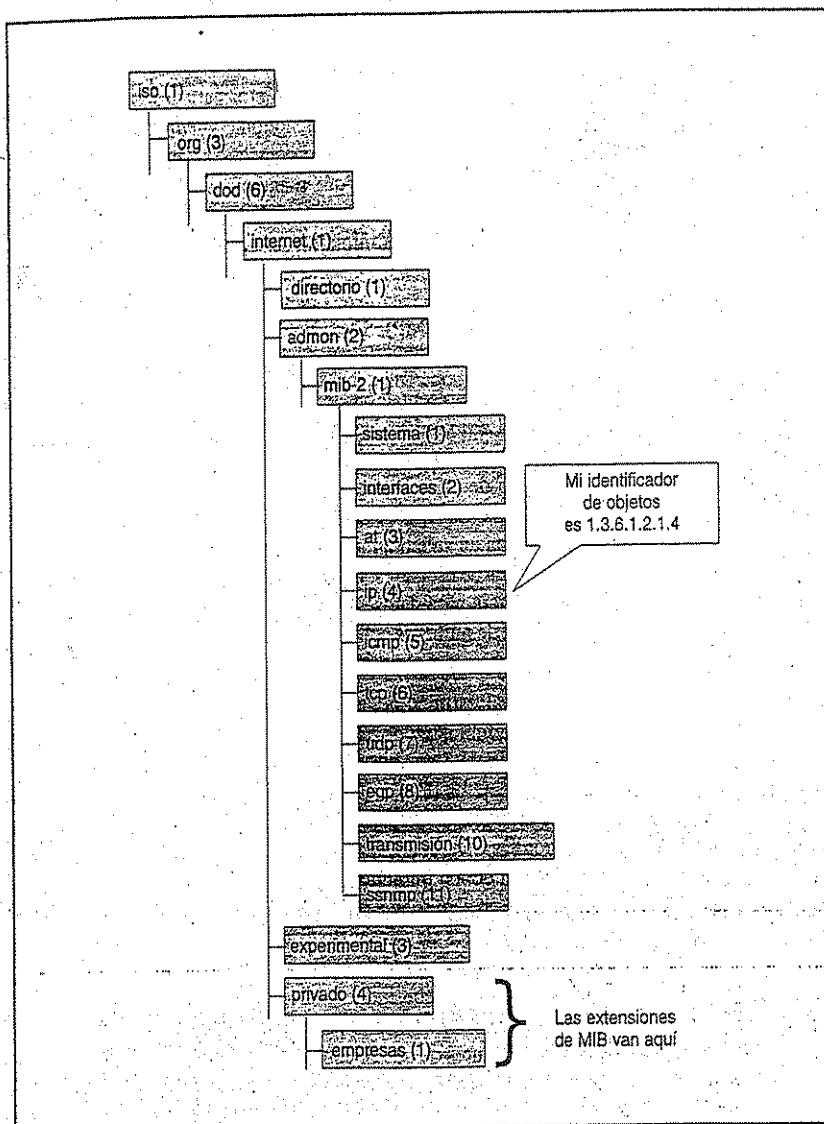


Figura 33-1. Grupos de objetos de MIB-II.

sirve para asignar *identificadores de objetos* empresariales a cada proveedor que registre uno. Los proveedores deben describir sus extensiones de MIB por medio de descripciones formales—definidas en RFC 1155 o RFC 1212—en un archivo de texto. Debe proporcionarse la información suficiente para permitir que una estación de administración cargue y compile la definición de MIB específica del proveedor y la incorpore a la biblioteca de descripciones de objetos



administrados. La estación de administración sólo puede acceder a información a la que sabe cómo llamar. Las extensiones de MIB privadas son el centro mismo del sistema de administración de SNMP y OSI, pero representan al mismo tiempo las áreas de mayor confusión. Los usuarios deben cerciorarse que las plataformas de administración que eligen puedan manejar las extensiones de MIB privadas para sus sistemas. Se sabe que diferentes sistemas de administración producen diferentes resultados al administrar los mismos datos de MIB, así que más vale tomar precauciones.

Herramientas de MIB

La mayoría de las plataformas de administración ofrecen herramientas para las buenas relaciones con las MIB. En esencia, una MIB es una modalidad de base de datos jerárquica distribuida a lo largo de estaciones administradas. Como cualquier base de datos, requiere herramientas que la vuelvan accesible. He aquí las herramientas más usuales para la simplificación de operaciones con MIB:

- Un *compilador de MIB* toma un archivo en formato RFC 1155 y lo convierte en un formato que pueda ser utilizado por la estación de administración. Asimismo, actualiza una MIB existente y añade nuevas definiciones específicas de proveedores.
- Un *visualizador de MIB* presenta el árbol de MIB en forma gráfica; permite buscar objetos por grupos o atributos. Se pueden regenerar datos de instancias desde cualquier nodo administrado. El visualizador permite la creación de alias para objetos de MIB (es decir, la asignación a éstos de nombres significativos para el usuario). En realidad no es otra cosa que una herramienta de consulta especializada para bases de datos MIB. Permite “recorrer la MIB”. Algunos proveedores han hecho posible la superposición de la información de MIB en mapas o imágenes de objetos de interés para el usuario.
- Un *generador de reportes de MIB* permite la creación gráfica de reportes de los datos administrados. Estos reportes pueden complementarse con gráficas de negocios, mapas y otras formas de presentación visual.

Las herramientas de MIB suelen integrarse con otras herramientas visuales de administración. Por ejemplo, una herramienta de detección de agentes puede exhibir la ubicación de los agentes en la red, además de lo cual usted puede emplear la herramienta de consulta de MIB para conocer los datos que contienen. Sin embargo, las MIB no son del todo amables con las personas. Fueron diseñadas para usarse por aplicaciones, no por individuos. Por lo tanto, lo mejor es mantener ocultos sus detalles tras las herramientas de presentación gráfica.

El SNMP de Internet

El *protocolo simple de administración de redes* (SNMP: *simple network management protocol*) es hoy en día el protocolo de administración de redes más extensamente implementado; lo soporta un número de dispositivos de redes en constante crecimiento. Tal como se le define en RFC 1157, el SNMP está diseñado para hacer exactamente lo que su nombre indica: efectuar

SNMP hasta el límite: Las extensiones RMON de MIB-II

El estándar de *monitoreo remoto de redes* (RMON: *remote network-monitoring*) definido en RFC 1757 es una extensión muy significativa de MIB-II que lleva a SNMP hasta sus límites.¹ El motivo que explica a RMON se halla en los proveedores de monitores de redes; necesitaban extensiones de SNMP que permitieran la participación de su equipo en la administración de redes. Un *monitor de redes* —Sniffer de Network General, por ejemplo— es un dispositivo “promiscuo” que, tras asentarse en la red, es capaz de interceptar y visualizar cualquier paquete, sin importar quién lo haya enviado a quién. Evidentemente, estos dispositivos de registro recopilan gran cantidad de información de enorme utilidad para una estación de administración de SNMP. Los monitores son “agentes secretos” por excelencia. Se introducen en la red y pueden ver todo lo que pasa por ella. De este modo, la pregunta es: ¿cómo obtiene una estación de administración esta información voluminosa por medio de SNMP? La respuesta: a través de las extensiones RMON de MIB-II.

Lo interesante de RMON es que se trata de la entidad más inteligente definida hasta ahora por Internet; rompe el molde de los dispositivos administrados de simplificación extrema y facultades nulas. Un monitor debe poseer la inteligencia suficiente para filtrar la información que recolecta y proceder en consecuencia sin involucrar directamente en cada acción a la estación de administración, ni atascar la red con enormes cantidades de transferencias de datos en masa. Mediante el *monitoreo preferente*, el equipo de registro (o *sonda*) emite diagnósticos permanentes sobre el tráfico en la red, notifica a la estación de administración la detección de fallas y proporciona valiosa información sobre el evento. Este activismo está muy lejos de la filosofía habitual de SNMP, para la cual cada nodo administrado es un conjunto de variables remotas definidas por MIB.

Con base en las extensiones de MIB-II sobre el SNMP estándar, RMON define las convenciones para indicarle a un monitor remoto qué datos recopilar. Recuérdese que SNMP no soporta comandos imperativos ni puede crear nuevas instancias de objetos. Así, RMON lo hace todo a través de convenciones que permiten que una variable de MIB represente un comando, y que otras variables representen los parámetros del comando; todo el proceso se distingue por su torpeza; pero es una muestra de lo que se puede hacer en casos desesperados. RMON define varios nuevos objetos de MIB-II que representan comandos. El monitor ejecuta el comando cuando la estación de administración se remite a esos objetos por medio del comando SET de SNMP. La especificación de RMON define la adición, eliminación o modificación de filas en las condiciones de MIB. En su mayor parte, RMON define nueve extensiones de grupos de objetos de MIB-II útiles para el almacenamiento de datos y estadísticas reunidos por un monitor (véase Figura 33-3).

Cada uno de los nueve grupos de RMON ofrece un conjunto de variables de control por medio de las cuales una estación de trabajo de administración puede controlar remotamente la operación de un agente de monitoreo. Estas variables pueden concebirse como tablas de estado que

¹ RFC 1757, introducida en febrero de 1995, sustituyó a RFC 1271; extiende los objetos con raíz en Ethernet de RFC 1271 para incluir a Token-Ring y FDDI.

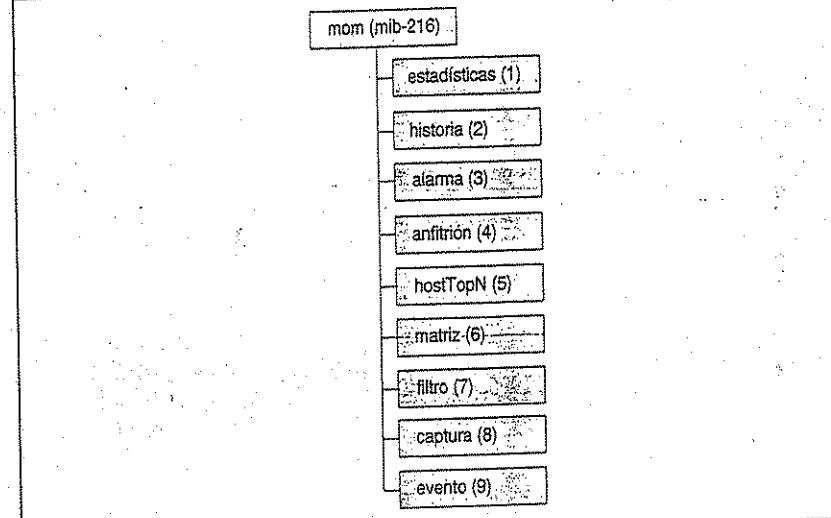


Figura 33-3. Extensiones de grupos de MIB-II de RMON.

indican al monitor qué recolectar y cómo manipular los eventos que genera. En un marco más moderno, todo esto habría podido hacerse mediante RPC, MOM o invocaciones de ORB. RMON es un extraño protocolo centrado en datos, que se sirve de variables de MIB para transmitir instrucciones, parámetros e información de control entre una estación de administración y un agente. Es un “pariente pobre”.

Y ahora con ustedes: RMON-2

En agosto de 1995 fue presentada a IETF la versión preliminar de MIB de RMON-2; es de esperar que cuando usted lea este libro ya se haya convertido en estándar. RMON-2 añade grupos a RMON. Recaba información sobre el funcionamiento de la capa de red (o, capa tres del modelo OSI). En contraste, el RMON original sólo se ocupaba de las capas uno y dos de OSI. RMON-2 permite que herramientas de administración compongan visualizaciones de extremo a extremo de la red en su totalidad. A diferencia de ello, RMON se ocupaba únicamente de segmentos de la red.

Con RMON-2 usted podrá obtener una descripción completa del funcionamiento de una red íntegra, no sólo de segmentos individuales de una LAN. Podrá administrar proactivamente sus redes y detectar responsables de retardos o estallidos de segmentos de redes. Algunos proveedores han ampliado incluso RMON-2 para reunir información de la totalidad de las siete capas del modelo OSI. Sin embargo, toda implementación que vaya más allá de la capa tres es absolutamente ajena al estándar.

- GET-NEXT también es idéntico a SNMP, excepto que la restricción atómica se relaja.
- GET-BULK es un nuevo comando emitido por una estación de administración. Es similar a GET-NEXT. Sin embargo, en lugar de ofrecer únicamente la siguiente variable, el agente puede remitir tantas variables sucesivas del árbol de MIB como quepan en un mensaje.
- SET es idéntico a SNMP. Es una operación en dos fases. En la primera fase, se verifica que todas las variables en la lista puedan ser actualizadas; en la segunda se procede a la actualización. Como en SNMP, es una propuesta de todo o nada.
- TRAP realiza una función similar a la que ejerce en SNMP, salvo que emplea un formato de paquetes diferente (el encabezado de trap de SNMP sólo reconoce el tipo de dirección de TCP/IP; SNMPv2 es más general). Al igual que en el caso de SNMP, la trap no está reconocida.
- INFORM es un nuevo comando, enviado por un administrador de SNMPv2 a otro administrador. Sirve para el intercambio de información de administración. Los mensajes pueden enviarse a todos los nodos administradores especificados en la MIB de M2M, o a un administrador en particular. La MIB de M2M permite que un nodo administrador superior defina los eventos subordinados de su interés.

Como SNMP, SNMPv2 carece de conexiones; usa un servicio de datagramas. La especificación incluye correspondencias con UDP, IPX, AppleTalk y el servicio sin conexión de OSI. Incluye asimismo un nuevo tipo de datos para redes de alta velocidad y mejores valores de remisión de errores.

La terrible parálisis de SNMPv2

Debate

Es Bosnia. Nos hemos convertido en serbios y croatas.

Marshall Rose, coautor de SNMPv2
(Marzo de 1996)

Cuando quedó demostrado que los derechos de acceso y la seguridad de SNMPv2 eran demasiado difíciles y complicados de implementar, el grupo de trabajo original de SNMPv2 volvió a la carga, pero esta vez con dos propuestas de seguridad rivales:

- El modelo de seguridad basado en el usuario (USEC: user-based security model) es defendido por Marshall Rose y Keith McCloghrie. USEC propone un modelo simple de seguridad minimalista. Se trata de un diseño centrado en agentes que requiere mínima intervención de la estación de administración. En consecuencia, también la administración de seguridad es mínima.

- SNMPv2* (o "V2 star") es defendido por Jeffrey Case y Steven Waldbusser (autor también de RMON-2). SNMPv2* introduce una estructura de seguridad sobre USEC para volverlo "más completo". Tal como Case lo ha expresado, "USEC posee tan pocas características que cabe preguntarse si las que tiene son suficientes". Seguramente piensa que no, y de ahí que haya presentado SNMPv2*, que ofrece múltiples servicios de autenticación y privacidad. Por el contrario, Rose, de USEC, piensa que SNMPv2* puede convertirse en "el CMIP de mediados de los noventa: muy capaz, pero extremadamente complejo, grande, costoso y desorganizado".

Como resultado de esta parálisis, IETF desintegró el grupo de trabajo de SNMPv2 en diciembre de 1995 para "permitir que los ánimos se enfrien". Se preveía convocarlo de nuevo en 1996, si los autores llegaban a un acuerdo. Mientras tanto, ambos bandos demandaban apoyo para establecer los estándares industriales *de facto* antes de la nueva convocatoria de IETF. Sin embargo, los proveedores en su mayoría aguardan en los márgenes el surgimiento de un claro ganador para implementar sus productos.

En nuestra opinión —ya que, después de todo, estamos en un recuadro de debate—, quizá ya es momento de que la industria prescinda de todas las variantes de SNMPv2 y opte por la administración de sistemas basada en objetos. SNMPv2 ya dejó pasar su oportunidad (como se insistirá en un recuadro de debate posterior). ¿Quién dijo que la administración de sistemas no era emocionante? □

LA ESTRUCTURA DE ADMINISTRACIÓN DE OSI

Muchas de las deficiencias de SNMP son resueltas por la administración de redes de OSI. Pero no falta quienes piensen que el remedio fue peor que la enfermedad, dada la complejidad y dimensiones de la administración de redes de OSI.

William Stallings, autor de
SNMP, SNMPv2 y CMIP
(Addison-Wesley, 1993)

La administración de sistemas distribuidos de OSI está definida por más de 30 estándares. ¿Está listo para una nueva dosis de siglas? De acuerdo con la visión de OSI, la administración de redes se divide en cinco componentes de nivel de aplicación llamados *áreas funcionales de administración de sistemas* (SMFA: *system management functional area*). Éstas son administración de fallas, administración de contabilidad, administración de configuración, administración de desempeño y administración de seguridad (véase Figura 33-5). Las SMFA se apoyan en los servicios de 13 *funciones de administración de sistemas* (SMF: *system management function*) definidas por OSI, las cuales pueden ser usadas por una o más SMFA. Las SMF descansan a su vez en el *elemento de servicios de información de administración común* (CMISE: *common management information services element*). El CMISE es una combinación de los protocolos definidos por los *servicios de información de administración común* (CMIS: *common management information service*) y el *protocolo de información de administración común* (CMIP: *common management information protocol*).

Protocolos de administración de OSI: CMIP, CMOT y CMOL

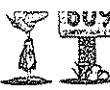
El verdadero significado de las siglas CMIP es "protocolo de información de administración (sumamente) complejo" (CMIP: complex management information protocol).

Marshall T. Rose, autor de
The Simple Book
(Prentice Hall, 1991)

CMIP es el protocolo de OSI para las comunicaciones administrador a agente y administrador a administrador. En aberto contraste con SNMP, CMIP es un protocolo orientado a conexiones, que opera encima de una pila completa de siete capas de OSI. *CMIP sobre TCP/IP* (o *CMOT*) ofrece una versión más ligera de CMIP para redes con TCP/IP. *CMIP sobre LLC* (o *CMOL*) es un CMIP más escueto todavía; fue diseñado por IBM y 3Com para operar directamente encima de la capa de vinculación lógica de IEEE 802.2. CMIP es mucho más rico en funcionalidad que su contraparte SNMP. Su protocolo ofrece los siguientes servicios:

- **Obtención (get)** solicita datos de la base de información de administración del agente. La solicitud puede ir dirigida a un solo objeto administrado o a un grupo de objetos administrados. Por cada valor del objeto administrado se pueden solicitar uno o más de sus atributos.
- **Reporte de eventos (event-report)** es una notificación enviada por un agente a un sistema de administración para indicar la ocurrencia de un evento. El servicio puede solicitar opcionalmente una confirmación. Junto con la notificación del evento se transmiten cinco parámetros para especificar la clase de objeto y la instancia en la que se originó el evento, el tipo de evento, la hora en que fue generado y toda información de usuario relativa al evento.
- **Acción (action)** es una solicitud por la cual un objeto administrado recibe la instrucción de llevar a cabo alguna acción en particular. La acción es implementada por un procedimiento, el cual es especificado como parte del objeto administrado. En caso de hallarse presente, el parámetro *acción-información* puede emplearse para transmitir parámetros de entrada e información adicional.
- **Creación (create)** es una solicitud hecha por un sistema de administración para crear una nueva instancia de una clase de objetos administrados.
- **Eliminación (delete)** es una solicitud presentada por un sistema de administración para borrar una instancia de una clase de objetos administrados.

Todos los servicios de CMIP pueden ejecutarse opcionalmente con confirmación. Para especificar el contexto de los objetos de administración de interés, CMIP emplea dos modalidades: *sondeo* y *filtración*. El sondeo marca un nodo en el árbol de información donde comienza el árbol de búsqueda; la filtración es una expresión de búsqueda booleana aplicada a los atributos de los objetos sondeados.



AGENTES DIMINUTOS: LA INTERFAZ DE ADMINISTRACIÓN DE ESCRITORIOS (DMI)

Ningún adaptador de proveedor único debería ser vendido sin un MIF... para tenerlo todo bajo control. Esta previsión significará muchas menos molestias posteriores para el usuario.

Jamie Lewis
PC Week

En el extremo contrario del espectro, el consorcio industrial denominado *Desktop Management Task Force (DMTF)* se encarga de definir estándares para la administración de todos los componentes de una PC, Mac o estación de trabajo, incluidos hardware, OS, aplicaciones, almacenamiento y periféricos. El consorcio DMTF se compone de 122 miembros formales; integran la dirección Compaq, Dell, Digital, HP, IBM, Intel, Microsoft, NEC, Novell, SCO, Symantec y SunSoft. En octubre de 1993, DMTF dio a conocer la primera versión de su *interfaz de administración de escritorios (DMI: desktop management interface)*. IBM, Microsoft y Apple anunciaron que la integrarían en futuras versiones de sus OS. Sin embargo, DMI es independiente de protocolos, plataformas y sistemas operativos.

DMI resuelve un problema real: facilita la administración de los alrededor de 10,000 componentes de PC en el mercado. Una de las razones de que la administración no se haya extendido hasta los componentes de PC a través de SNMP es el costo. Para cumplir con SNMP, los proveedores de aditamentos habrían tenido que crear una MIB privada, generar interfaces para los agentes de SNMP y negociar con los proveedores de plataformas de administración la interpretación de sus MIB. Además, la mayoría de las PC que operan con DOS y Windows carecen de la RAM suficiente para soportar múltiples agentes de SNMP o sus pilas de protocolos. En compensación, en ciertos casos se integraron agentes propietarios a tarjetas adaptadoras; por ejemplo, las tarjetas para Ethernet de 3Com o Cabletron. Con DMI, a los proveedores de componentes les basta con "adaptar" sus dispositivos; DMI hace el resto.

El soporte de DMI se difundió en 1996. Se le incorpora ya en los productos de proveedores de adaptadores (Intel, Madge y 3Com), fabricantes de PC (AST, Dell y Apple) y proveedores de administración de sistemas (Novell, IBM/Tivoli y Sun). Además, la soportan ya casi todos los programas de inventarios de LAN en el mercado, tales como *LAN Directory* de Frye, *LAN Inventory* de McAfee y *Administrator for Networks* de Norton-Lambert. Finalmente, las redes de Apple, IBM, Sun y Ki han anunciado una *tecnología de agentes comunes* que permite que componentes observantes de DMI sean administrados por las ya existentes consolas de administración basadas en SNMP. Los agentes comunes establecen correspondencias entre los MIF de DMI y las MIB de SNMP. IBM proporcionará agentes comunes para OS/2, Windows 95 y NT; Apple ofrecerá un agente común para Mac, y Sun y Ki ofrecerán agentes comunes para Solaris, HP-UX y otras variantes de Unix. Microsoft brinda ya una implantación parcial del estándar de DMI en la característica de conexión y manejo de Windows 95. En el futuro, pondrá a disposición de agentes de DMI, vía una interfaz de OLE, la información del registro de Windows y NT.

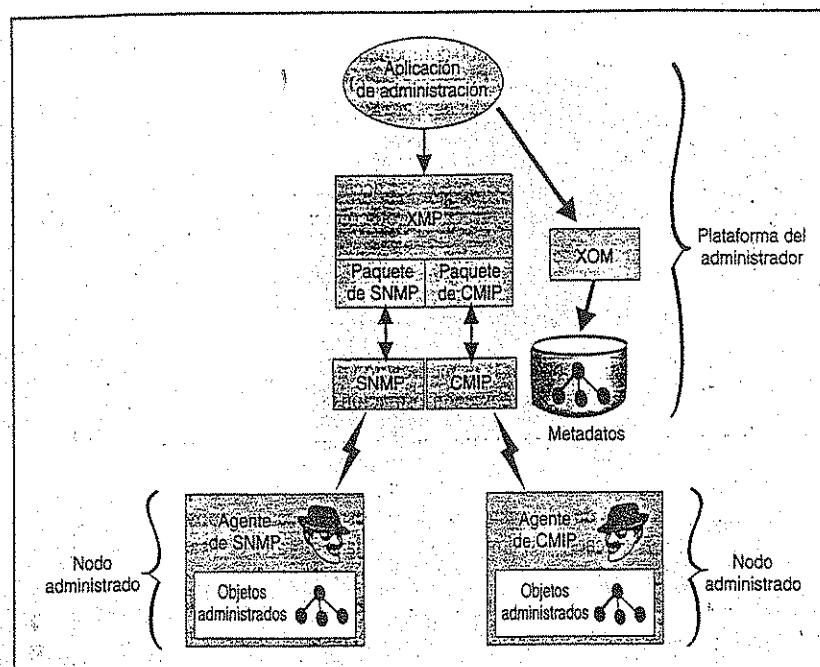


Figura 33-7. XOM, XMP y base de datos de metadatos de X/Open.

La API XMP de X/Open

La API de administración de X/Open (XMP: X/Open management API) se deriva de una interfaz anterior, llamada API de administración consolidada (CM-API: consolidated management API), de Bull y HP. La API XMP se usa para comunicaciones proceso a proceso basadas en estándares entre un sistema de administración y un sistema administrado. XMP define un conjunto de llamadas de API C que permite el acceso tanto a SNMP como a CMIP. La semántica de la interfaz se asemeja más a CMIP que a SNMP. A causa de las diferencias en la representación de los datos administrados, XMP no vuelve del todo transparente la aplicación a SNMP o CMIP. Pero como solución intermedia, ofrece diferentes "paquetes" basados en XOM para su uso con los diferentes protocolos (explicaremos XOM en la sección siguiente). Un paquete está definido para operaciones de SNMP estándar, mientras que el otro está definido para operaciones de CMIP estándar.

La API XOM de X/Open

XMP depende de otra API definida por X/Open, llamada *administrador de objetos de X/Open* (XOM: X/Open object manager). La API XOM sirve para manipular las estructuras de datos

asociadas con los objetos administrados. Las estructuras de datos de XMP se preparan con llamadas de API XOM. La actual versión de XOM ofrece un medio para el tratamiento de tipos complejos en C definidos por ASN.1. Tanto SNMP como CMIP usan ASN.1. XOM es suficientemente general para ser usado por protocolos no relacionados siquiera con la administración de sistemas. Por ejemplo, DCE emplea a XOM en sus servicios de directorio global.

Para soportar las jerarquías más complejas de objetos de CMIP, X/Open incluye un *kit de desarrollo de paquetes* (PDK: package development kit). El componente primordial del PDK es un *compilador de metadatos*, capaz de leer definiciones de GDMO y producir un paquete (en realidad un conjunto de tipos de datos de C). Se pueden usar entonces estructuras de C con XMP y XOM. El compilador de metadatos ocupa una base de datos local con información derivada de definiciones de GDMO. La información sobre clases de objetos administrados de CMIP incluye sus clases de objetos superior y subordinada, los atributos de la clase y si una clase de objetos dada soporta la creación y eliminación de solicitudes. Se puede acceder a la base de metadatos usando llamadas de XMP normales. Este proceso es similar en naturaleza al compilador de IDL y el depósito de interfaces de CORBA, aunque por lo general la arquitectura de CORBA es mucho más consistente y ofrece funciones más avanzadas.

EL ESTÁNDAR DME DE OSF

En julio de 1990, OSF emitió una solicitud de tecnología (RFT: request for technology) para un entorno de administración distribuida (DME: distributed management environment), el cual ofrecería una solución completa a la administración de sistemas y redes en entornos heterogéneos de múltiples proveedores. Cualquier institución, fuese miembro de OSF o no, podía responder. Veinticinco organizaciones presentaron tecnologías. En septiembre de 1991, OSF dio a conocer a las ganadoras, entre las que estaban OpenView de HP, WizDOM de Tivoli, Data Engine de IBM y los controladores de CMIP y SNMP de Groupe Bull. En mayo de 1992 dio a conocer una arquitectura muy completa, que combinaba una estructura de administración de redes tradicional con una estructura de objetos posmoderna basada en CORBA.

A fines de 1994, OSF redujo las dimensiones de su propuesta de DME. En vez de emplear el ORB de Tivoli, ahora se especifican interfaces administrativas con cualquier ORB compatible con CORBA. Todo indica que OSF concentra sus mayores esfuerzos en la administración de DCE. Como resultado, los proveedores de plataformas abiertas adquieren partes de la tecnología de DME directamente con sus creadores y las incorporan en sus productos. Por ejemplo, IBM obtuvo licencia para partes de OpenView de HP, que fueron incorporadas en NetView/6000 y LAN NetView. Despues, Digital obtuvo licencias para NetView de IBM; éste es el fundamento de Polycenter NetView. Finalmente, IBM adquirió a Tivoli a principios de 1996. De este modo, Tivoli es ahora el fundamento de la plataforma de administración de sistemas basada en CORBA de IBM, llamada *TME 10*.

A pesar de su reducción, la arquitectura de administración de sistemas de DME sigue siendo de gran interés. DME brinda una estructura completa para la comprensión de la administración de sistemas distribuidos en sus enfoques clásico y posmoderno. La arquitectura de DME concilia en forma original estos dos enfoques empleando para ello envolturas de objetos y un IDL y

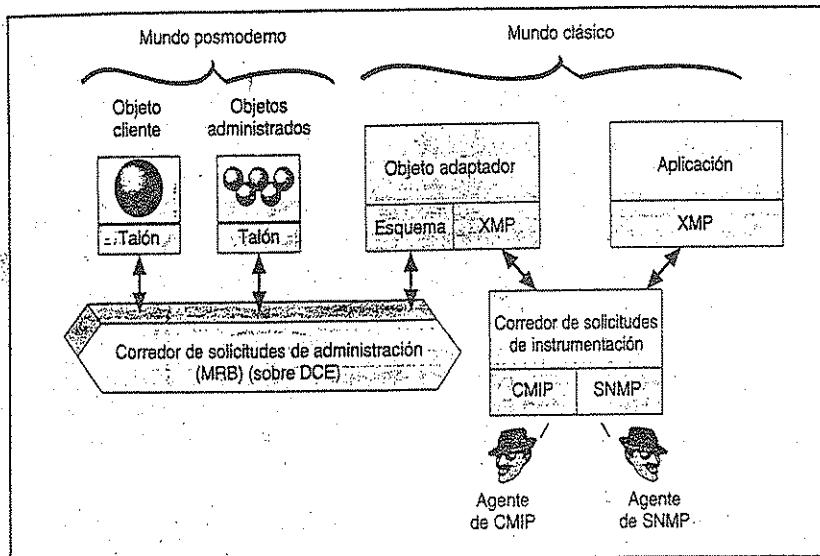


Figura 33-9. La estructura de administración de objetos de DME.

sentan a la interfaz del usuario y aplicaciones de administración. DME se ocupa en particular de la comunicación entre objetos de administración. Los adaptadores de objetos se usan para encapsular protocolos de administración estándar, como SNMP, CMIP y DCE. Con el apoyo de los servicios de directorio de DCE, el corredor de solicitudes de administración de DME ofrece un espacio de nombramiento uniforme para todos sus objetos (de herencia o posmodernos). Sabe cómo localizarlos e invocar sus servicios. Así que bienvenido al mundo posmoderno de la administración de sistemas con objetos distribuidos. La encarnación comercial de esta nueva arquitectura es el *Tivoli Management Environment (TME)*, del que hablaremos más adelante.

FYI

La nueva OSF

Información

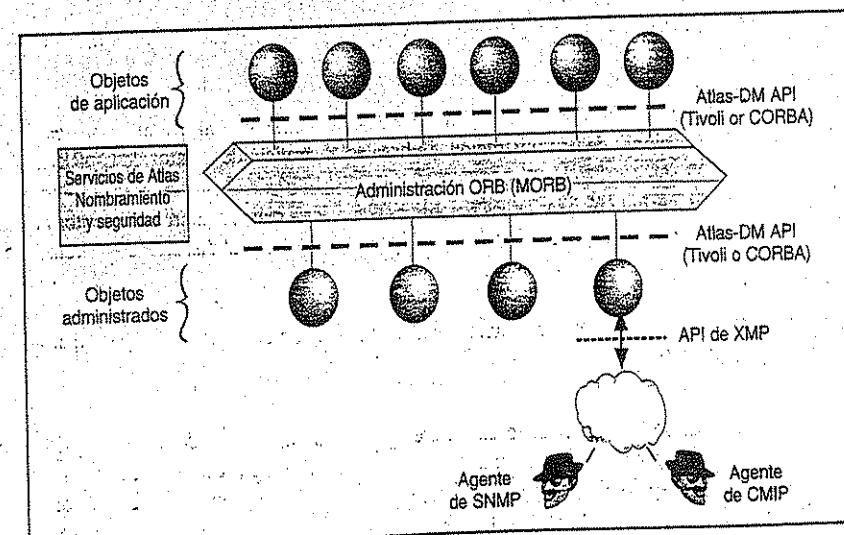
La nueva misión de OSF es concentrarse en la arquitectura, subcontratación, y distribución de software de infraestructura basado en estándares. Ha dejado de ser por tanto una organización de desarrollo completo de software. La nueva OSF ha adoptado un método secular para el ORB de administración; cualquier ORB compatible con CORBA podrá fungir como tal. Cuando los miembros de OSF optaron por un ORB de administración compatible con CORBA, decidieron que OSF cediera a OMG el ramo relacionado con ORB. Cualquier ORB compatible con CORBA puede ser un MRB. □



Estructura de administración distribuida de UI-Atlas

Unix International (UI) fue el otro consorcio Unix de la industria; sus 270 miembros definieron los requerimientos para la evolución de software distribuido para Unix en el marco de una estructura arquitectónica llamada *Atlas*. En julio de 1991, UI publicó sus requerimientos de Administración distribuida *Atlas* (*Atlas-DM: Atlas-distributed management*), cuyo concepto unificador básico son los objetos distribuidos. USL optó por implantar la primera versión de *Atlas* con base en la tecnología de Tivoli existente. La segunda versión, dada a conocer en 1994, se basaría en CORBA. En la época anterior a la adquisición de Novell, los requerimientos de UI fueron implantados por USL (o cualquier otra entidad contratada por ésta para el efecto). Hablamos en pasado porque UI se disolvió el 31 de diciembre de 1993. Aun así, expondremos *Atlas-DM* en esta sección debido a que se trata de una importante arquitectura que bien podría ser implantada en algunas variantes de Unix.

Arquitectónicamente, UI incluso llegó más lejos que DME de OSF en su soporte de objetos; en su primera versión, *Atlas* ignoró a SNMP y CMIP. Al igual que el DME posmoderno, *Atlas-DM* define a las aplicaciones de administración como conjuntos de objetos que interactúan entre sí y con objetos que representan a los recursos administrados. El *ORB de administración (MORB: management ORB)* de UI ofrece acceso transparente a objetos administrados a lo largo de la red (véase Figura 33-10). Los objetos representan recursos genéricos en el entorno distribuido, incluidos anfitriones, usuarios, LAN, DBMS, aplicaciones, discos, archivos y OS. Una llamada de API dinámica única —la *objcall* de Tivoli— se usa para invocar métodos en objetos (todos los parámetros se transmiten en cadenas de caracteres de ASCII). Las referencias de objetos pueden transmitirse en tiempo de ejecución. Además, Tivoli permite el descubrimiento dinámico.

Figura 33-10. La estructura de administración de objetos de *Atlas-DM*.

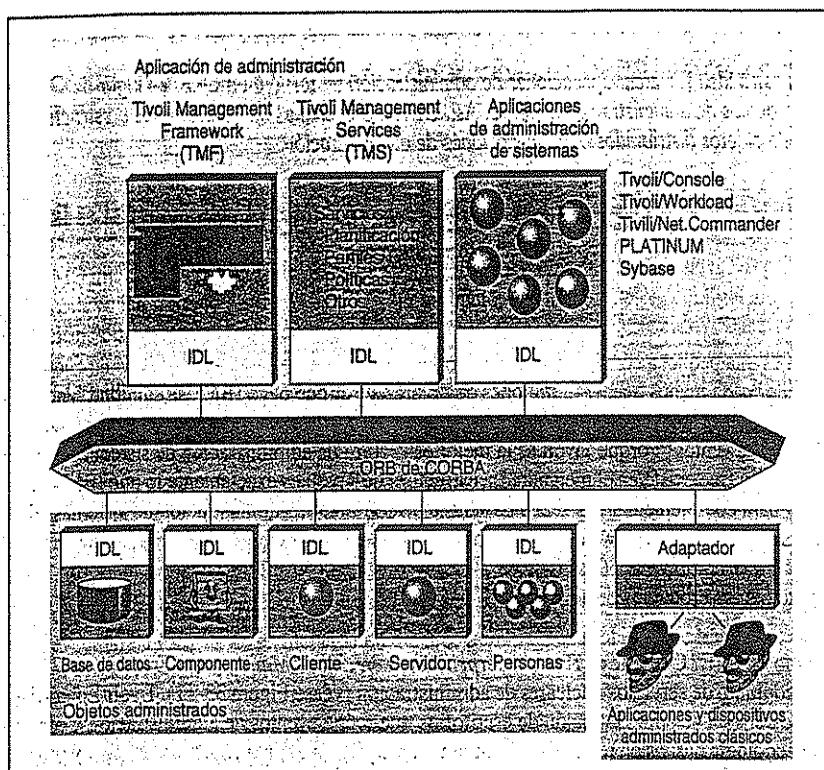


Figura 33-11. El Tivoli Management Environment (TME).

Administración de sistemas: Por qué los objetos son la respuesta

Debate

Un sistema de administración de redes se ve limitado por las capacidades del protocolo de administración de redes y por los objetos usados para representar el entorno por administrar.

William Stallings, autor de SNMP, SNMPv2 and CMIP (Addison-Wesley, 1993)

Creemos que CORBA ofrece un protocolo moderno y natural para la representación de entidades administradas, la definición de sus servicios, la especificación de datos de ins-

tancias y la invocación de métodos a través de un ORB. Pueden usarse los depósitos de interfaces e implantaciones de CORBA para descubrir e invocar dinámicamente métodos en esos objetos administrados en tiempo de ejecución. Los objetos administrados pueden llamar directamente a la estación de administración cuando tienen algo significativo que reportar (a diferencia de lo que ocurre en SNMP, que depende principalmente de sondeo). El servicio de eventos de CORBA es ideal para la distribución de eventos asíncronos de administración de sistemas. Por su parte, los servicios de objetos de CORBA ofrecen funciones útiles y abundantes que tendrían que ser reinventadas por SNMP o CMIP de OSI, como ciclo de vida, nombramiento, transacciones y persistencia. Con el uso de CORBA, la administración de sistemas distribuidos se convierte simplemente en otro servicio del ORB. Los objetos pueden administrarse a sí mismos. Las aplicaciones de administración ofrecen visualizaciones de grupos de objetos autoadministrados.

El protocolo simple de administración de redes (SNMP: *simple network management protocol*), el protocolo de administración predominante hoy, es demasiado limitado para los requerimientos de la administración de sistemas total. Debe remplazársele. Esto significa que en los próximos años experimentaremos una gran transición a software de administración más sofisticado. Los tres contendientes en el remplazo de SNMP son SNMPv2, CMIP de OSI y CORBA. Todos ellos requieren de más memoria y procesadores más inteligentes que SNMP. Así, la pregunta es por cuál de ellos optar. Nosotros nos inclinamos por CORBA. SNMPv2 y CMIP son ya antiquíllas, además su programación es increíblemente desordenada. Las MIB representan un anacronismo en la era de IDL y almacenes persistentes de objetos. Requieren que aplicaciones de administración interactúen con gran cantidad de atributos de bajo nivel, a los que manipulan por medio de *get* y *set*. SNMPv2 no permite el registro de operaciones de objetos administrados; CMIP sí, pero en forma desmanada. Creemos que el SNMP “simple” fue un excelente protocolo básico para la resolución de muchos problemas reales en la época de la carestía y de la administración simple de redes. Pero a medida que nos acercamos a la administración de sistemas total, entre más pronto se abandone SNMP, SNMPv2 y CMIP y se les reemplace por CORBA, la administración de sistemas será mejor.

En la Tabla 33-1 se hace una comparación entre las características de SNMP, SNMPv2, CMIP y CORBA. SNMP y SNMPv2 ceden una mínima cantidad de lógica de emisión de eventos a los agentes, de modo que los nodos administrados son muy simples. Las facultades residen en la estación de administración. Como resultado de ello, traps (o eventos) se usan con poca frecuencia, y la estación de administración debe sondear a los agentes para determinar qué ocurre. En contraste, CMIP y CORBA están sujetos a eventos, lo que significa que los agentes son más inteligentes; el sondeo de la estación de administración es limitado. En consecuencia, una estación de administración con CORBA puede manejar un número mucho mayor de objetos administrados.

En general, CMIP o CORBA resultan mejores que SNMP para la administración de grandes y complejas redes de proveedores múltiples. Claro que la simplicidad de SNMP permite que se le despliegue en más dispositivos, lo que a su vez facilita la administración de grandes redes. Los diseñadores de SNMP entendieron a la perfección la disyuntiva. Optaron por el mínimo común denominador y aceptaron las consecuencias. En SNMPv2 se



con *LAN NetView* y *OS/2 SystemView*. Estos proveedores tienen larga historia en la administración de PC, servidores de NOS y LAN departamentales. La información que reúnen nutre a la cadena alimenticia de la administración. Históricamente, estos sistemas no han sido capaces de ofrecer soluciones administrativas de alcance empresarial.

¿Quién será el ganador? Sería prematuro determinarlo. Habrá que estar atentos a adquisiciones y fusiones. La administración de cliente/servidor es demasiado compleja como para una respuesta monolítica de "un solo tamaño para todas las necesidades". Tampoco son la respuesta series preintegradas de productos monolíticos; son demasiado inflexibles. Los proveedores mejor posicionados son aquellos que ofrecen estructuras de administración abiertas sobre un bus de objetos distribuidos (como CORBA).

Las estructuras abiertas pueden complementarse con cientos de módulos de conexión de terceros. Reúnen excelentes productos de administración en una solución integrada por el usuario. Juntas, estas conexiones (y sus estructuras) deberían ser capaces de administrar PC, LAN, WAN, aplicaciones, servidores y el Web. Futuras conexiones se formarán con el uso de tecnología de componentes.

Los líderes en estructuras de administración de sistemas distribuidos orientadas a objetos son *TME* de Tivoli, *Solstice Enterprise Manager* de SunSoft, *Spectrum* de Cabletron, *Unicenter con estructuras de CORBA* de CA/Legent y *SystemView/Karat* de IBM. Es probable que para cuando usted lea este libro Symantec y HP también ofrezcan ya estructuras de objetos distribuidos.

CONCLUSIÓN

Los sistemas de cliente/servidor intergalácticos no funcionarán si no es posible administrarlos. La difusión de Internet e intranets exacerbará este problema. Los sistemas de administración distribuida están en pañales. Sus agentes parecerían de la Edad de Piedra. Por ejemplo, los agentes de SNMP son absolutamente pasivos; sólo hablan cuando se les dirige la palabra. Necesitamos una nueva generación de agentes inteligentes capaces de actuar independientemente de los sistemas de administración. Necesitamos sistemas de administración basados en componentes sobre un fundamento de objetos distribuidos. Necesitamos mejores estándares de objetos para aplicaciones de administración. Y necesitamos todo esto ya.

Desafortunadamente, los organismos de estándares avanzan muy lentamente. IETF gastó tres preciosos años en sus guerras de SNMPv2. El DME de OSF y Atlas de UI fueron esfuerzos abortados. CMIP de OSI es un fracaso comercial. Y OMG avanza con extrema lentitud en el área de administración de sistemas. Las dos estrellas en éste panorama, de otro modo desolador, son DMTF y Tivoli. La DMI de DMTF está obteniendo una gran aceptación comercial y ya se ocupa de los aspectos relacionados con la administración de aplicaciones. Tivoli creó el estándar de objetos *de facto* para la administración de sistemas con base en CORBA y los MIF de DMI. Se ocupa ahora de la cuestión de la administración de sistemas en el Web, con la colaboración de JavaSoft. Así pues, todavía tenemos esperanzas.

Parte 10 Resumen





HERRAMIENTAS DE DESARROLLO DE APLICACIONES CLIENTE/SERVIDOR

Las herramientas de desarrollo son la pieza clave en cliente/servidor. Encapsulan la tecnología de cliente/servidor descrita en este libro y facilitan a los usuarios la generación de aplicaciones. Las mejores herramientas son sumamente visuales. Casi todas las herramientas actuales de cliente/servidor sirven para crear sistemas departamentales de apoyo de decisiones. Sin embargo, ya han comenzado a surgir herramientas de Internet. Es probable que este hecho saque al mercado de herramientas de sus búnkeres departamentales. ¿Cómo clasificar estas herramientas de cliente/servidor? ¿Cuál es la herramienta más indicada para una labor precisa? Como en todo lo referente a cliente/servidor, parecería que todo mundo tuviera una opinión que emitir sobre herramientas. Por lo tanto, en esta sección presentaremos el esquema de clasificación que nos ha funcionado a nosotros.

El mejor y más reciente modelo de herramientas de cliente/servidor

La evaluación de herramientas de cliente/servidor puede ser, en efecto, sumamente entretenida. Es probable que se le bombardee con toneladas de atractivos anuncios de herramientas en los que le ofrezcan soluciones de cliente/servidor instantáneas y sencillas. Quizá en alguna feria comercial ya haya presenciado una de esas habilidosas demostraciones en las que se da la impresión de que para crear aplicaciones enteras de cliente/servidor basta con unos cuantos clics con el ratón. Pero con más de 300 herramientas en el mercado, ¿cuál de ellas elegir para efectos de evaluación? Es de suponer que, en estas épocas, no disponga usted ni de presupuesto ni de tiempo infinitos para la evaluación de todas y cada una de estas 300 herramientas. Así pues, podría empezar seleccionando herramientas por plataforma, con el inconveniente, sin embargo, de que casi todas ellas corren ya en todas las plataformas más conocidas. O podría seleccionarlas de acuerdo con su precio, aunque correría el riesgo de obtener tan poco como lo que pagó. O podría seleccionarlas por su marca, con lo que se expondría a dejar de lado herramientas de vanguardia con las más recientes y mejores características.

Para facilitar su evaluación, le proponemos un modelo simple, consiste en la división del mercado de herramientas en torno a cuatro ejes (véase Figura 34-1):

- **Raíces.** ¿Cuáles son los antepasados de la herramienta? En uno de los extremos de este eje se encuentran las herramientas que se originaron en macrocomputadoras; tienden a centrarse en CASE. En el extremo contrario se hallan las herramientas que se originaron en LAN de PC; tienden a centrarse en GUI estar sujetas a eventos. En la parte intermedia estarían las herramientas supermini 4GL. El método (o paradigma) seguido por una herramienta para el desarrollo de aplicaciones está íntimamente ligado a sus raíces.
- **Tecnología distribuida.** ¿Cuál es la tecnología de cliente/servidor? ¿Qué aplicaciones crea la herramienta? El eje va de clientes grandes con servidores pequeños que se especializan en el apoyo de decisiones a clientes pequeños con servidores amplios que realizan OLTP. En medio estarían Groupware y objetos distribuidos, que dividen la lógica más equilibradamente.

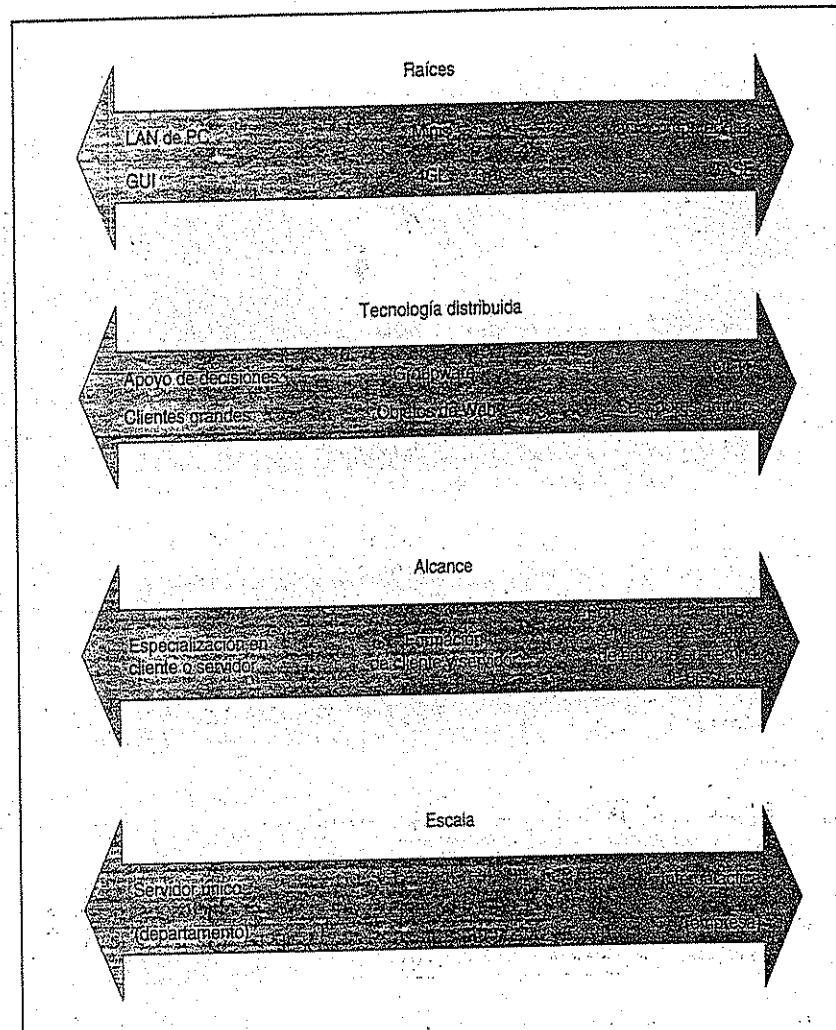


Figura 34-1. Los cuatro ejes de las herramientas de cliente/servidor.

- **Alcance.** ¿Cuántas funciones de cliente/servidor ofrece la herramienta? El eje va de herramientas que se especializan en el cliente, el servidor o el middleware a herramientas que ofrecen los tres (véase Figura 34-2). Algunas herramientas pueden llegar incluso a integrar un tiempo de ejecución que incluya administración de sistemas, administradores de recursos y un solo punto de instalación. Parecerá herejía, pero lo cierto es que entre mayor sea el contenido de la herramienta, menor integración tendrá que aportar usted. La integración es uno de los mayores dolores de cabeza en los sistemas de cliente/servidor.



Cliente/servidor consiste fundamentalmente en una relación entre programas que corren en distintas máquinas. Requiere por lo tanto de una infraestructura para hacer cosas en las que las PC independientes ni siquiera se ocupaban. Por ejemplo, en el diseño deben incluirse robustas comunicaciones entre procesos a lo largo de LAN o WAN. Interfaces gráficas que hagan uso de GUI y OOUI deben ser explotadas a fin de que las aplicaciones luzcan y puedan manejarse más como objetos reales que como procesos de programación. Las interfaces del usuario son entornos cada vez más complejos, sensibles y específicos, con el acento puesto en las tareas humanas. Los clientes con OOUI insertan a la gente en el circuito distribuido, lo que inevitablemente incrementa las complicaciones. Las personas cometen muchos errores, hacen cosas imprevistas y requieren habitualmente grandes cantidades de información de diversas fuentes. Las OOUI más avanzadas —como los documentos compuestos y las ubicaciones embarcables— convierten a las estaciones de trabajo del cliente en mundos virtuales en los que se sostienen numerosos diálogos paralelos con una amplia variedad de servidores.

Las OOUI también le conceden al usuario mucha mayor libertad que las GUI o los sistemas basados en terminales. Los usuarios son libres de organizar sus objetos visuales (y escritorio) a su gusto. Ya no se ven restringidos a la rígida lógica de las aplicaciones orientadas a tareas. Las OOUI carecen de paneles principales y pantallas de navegación. Dificultan la determinación de dónde comienza una aplicación y termina otra (o de qué es un objeto de aplicación y un objeto del sistema). Todo se reduce a objetos visuales en todas partes. Esto plantea dos importantes preguntas: ¿requiere cliente/servidor de un nuevo enfoque de desarrollo de sistemas? En aquellos casos en los que el diseño solía comenzar predominantemente en la base de datos, ¿qué es primero ahora: el cliente o el servidor?

¿Qué hace diferente a cliente/servidor?

El diseño de sistemas tradicionales (basados en terminales) empezaba con los datos. Las pantallas se desarrollaban primordialmente para dirigir el proceso de llenado de la base de datos, de manera que se les diseñaba después de que se definieran las transacciones y tablas. Por el contrario, las aplicaciones cliente/servidor requieren un método de diseño mucho más complejo:

- La interfaz es más flexible que las terminales, y al usuario se le permite mayor libertad.
- Los diseños de primer plano basados en objetos asignan mucho más inteligencia a la parte del cliente de la aplicación.
- Los mensajes entre el cliente y el servidor son acordes con el usuario y específicos de aplicaciones.
- El diseño debe optimizarse para aprovechar el paralelismo inherente a la aplicación distribuida.

Esto da como resultado un método de diseño exclusivo de las aplicaciones cliente/servidor: el diseño debe empezar tanto en el cliente como en el servidor. Así, en una aplicación cliente/



servidor se cuenta con dos puntos de partida: la GUI/OOUI y los datos (a menos que los datos de un proceso de negocios ya estén en su sitio). Los diseños de GUI/OOUI y datos se unen en los objetos de aplicación del plano intermedio. Los objetos de aplicación establecen correspondencias entre los objetos visuales y la base de datos, y viceversa. Encapsulan los datos compartidos y las reglas de negocios. ¿Le parece complejo? No se preocupe: no es tan difícil como parece.

La creación rápida de prototipos es esencial

Uno de los medios para evitar una situación del “huevo” y la “gallina” en el desarrollo —como la que se muestra en la Figura 34-3— es emplear una metodología de rápida creación de prototipos. La rápida creación de prototipos permite el desarrollo de sistemas en forma incremental. Se empieza por el cliente, y después se procede de la misma manera con el servidor. Se debe avanzar siempre con pequeños pasos, e ir afinando constantemente el diseño a medida que se progresá. Cerciórese de involucrar al usuario final durante todas las etapas de diseño de interfaces. En esta forma de desarrollo en delta, el sistema se afina paulatinamente hasta que el desarrollo de un prototipo funcional se halla suficientemente maduro para insertarlo en la producción. Es probable que este método represente una carga mayor para el programador que los métodos tradicionales, que descansaban en análisis y diseños de la parte posterior a la frontal. Con la creación rápida de prototipos se corre el riesgo de que, además por efecto de una decisión administrativa de reducción de costos, un prototipo “funcional” no optimizado sea prematuramente insertado en la producción. También el extremo contrario puede implicar un riesgo: el síndrome de creación perpetua de prototipos.

Estos riesgos son fácilmente rebasados por los beneficios que ofrece la creación rápida de prototipos en el desarrollo de un sistema de cliente/servidor con OOUI. En entornos de cliente/

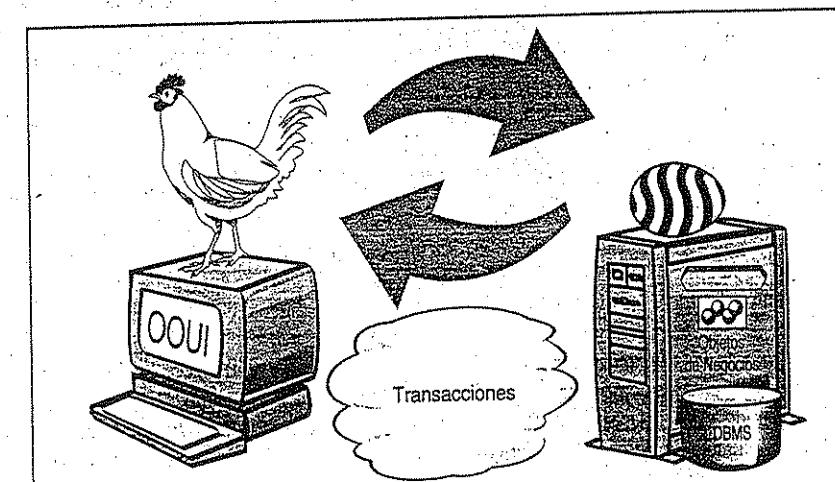


Figura 34-3. El huevo y la gallina de cliente/servidor.

ción a objetos/acciones del diseño del cliente puede contribuir a la transformación de los objetos visuales en un modelo de relaciones de entidades para los objetos de la base de datos. Las acciones se traducen en invocaciones de métodos en los objetos del servidor del plano intermedio que encapsulan a los datos persistentes. Los datos pueden almacenarse en una amplia variedad de almacenes de datos, como archivos de HTML, DBMS, ODBMS, Lotus Notes, macrocomputadoras y archivos simples. El objeto del servidor del plano intermedio es el punto de integración de los datos. Los clientes sólo pueden manipular datos invocando a estos objetos del servidor. Los objetos del servidor pueden optar por guardar los datos extraídos en memoria caché en un ODBMS o conservarlos en la memoria. Todo esto es transparente para el cliente.

9. *Publicación del IDL para los objetos del servidor.* La publicación del IDL anuncia al mundo lo que hace un objeto del servidor. Es un contrato de cumplimiento obligatorio entre el objeto del servidor y sus clientes. El IDL define las funciones exportadas por el objeto del servidor. Ofrece un nivel de abstracción más alto que los mensajes. Podría usarse, por ejemplo el IDL de RPC de DCE o, mejor aún, CORBA, cuyo depósito de interfaces ofrece un medio para el descubrimiento dinámico de servicios. Para cuando usted lea este libro, también Network OLE será una opción.
10. *Desarrollo del código de los objetos de negocios uno por uno.* Esto significa el desarrollo de código en los tres planos para cada objeto. Valide con el usuario tanto la interfaz del usuario como el desempeño del sistema.
11. *Paso del prototipo al sistema funcional.* Un sistema funcional es la suma de todos los objetos de negocios que contiene. Es la implantación de una ubicación. Usted desarrolla su sistema en forma paulatina, de modo que dispondrá de un sistema plenamente funcional una vez que codifique su último objeto de negocios.

Pero esto no es todo. Después de que haya desarrollado un prototipo aceptable, puede empezar a experimentar con cuestiones de distribución. ¿Dónde se almacenarán las ubicaciones? ¿Cómo son enviadas del servidor a los clientes? ¿Cómo se protegerán estas ubicaciones? Expusimos estos temas en la Octava parte. ¿Es todo? No todavía. También debe emplear el prototipo para saber cómo equilibrar las cargas entre servidores. Lo maravilloso de los objetos en 3 planos es que pueden ampliarse tanto descendente como ascendente. En un extremo, todos los objetos corren en una sola máquina. En el otro extremo, a cada objeto se le asigna su propia máquina. Así, emplee el prototipo para jugar con las cargas. Se trata de un acto dinámico de equilibrio de cargas entre los clientes y el servidor, y entre servidores.

CONCLUSIÓN

La tecnología de cliente/servidor les ofrece a los desarrolladores el potencial para crear aplicaciones visuales nuevas y revolucionarias. La creación de este tipo de aplicaciones requiere de la integración inconsútil de tecnología de OOUI, sistemas operativos, tecnología de ORB y DBMS. Para tener éxito, necesitaremos de nuevos métodos de desarrollos de sistemas que enfaticen la creación rápida de prototipos y la participación del usuario final. Esto nos permitirá explotar la sinergia entre objetos del cliente y el servidor. Estaremos también en mejor posición para com-



prender, desde las primeras etapas de un proyecto, las oportunidades (el paralelismo, por ejemplo) y los riesgos (desempeño y recuperación de errores, por ejemplo) introducidos por la distribución de una aplicación en una red.

El método de diseño basado en prototipos elimina la necesidad de especificaciones extensas. Pero, sobre todo, este método permite que el usuario participe en la especificación del producto y su gradual afinación. Se presta asimismo al diseño de aplicaciones distribuidas porque es posible pulir y ajustar la distribución de funciones a medida que se sabe más sobre el comportamiento del sistema en la realidad.

Así pues, el exitoso diseño de cliente/servidor en 3 planos se inicia en paralelo tanto en el cliente como en el servidor. Los dos puntos de partida son los objetos de OOUI y los objetos de datos. El "pegamento" que los une son los objetos de negocios del plano intermedio. Empiece por el cliente y derive después en el servidor, o viceversa. En cualquier caso, dé pasos cortos y repita. El prototipo visual le da vida anticipada al diseño. Por lo tanto, ¿quién es primero: el cliente o el servidor? ¡Los dos!



Además, los alumnos que optaron por el Web dedicaron a sus proyectos una cantidad de tiempo extraordinaria; hicieron mucho más de lo que les habíamos pedido. Algunos sencillamente acamparon en nuestro laboratorio.

¿Cuál es la moraleja de esta historia? Indica que la programación de cliente/servidor con el Web es muy popular entre nuestros alumnos de posgrado, muchos de los cuales son también programadores profesionales. Aparte de popular, la programación para el Web también puede ser adictiva y divertida. Pero, tal como ya lo sabe usted por este libro, para la verdadera operación en cliente/servidor nos hace falta todavía el Web de objetos. Afortunadamente, la programación con el Web de objetos será igualmente entretenida, si no es que más. Así pues, gracias al Web el desarrollo de cliente/servidor se está convirtiendo en una actividad divertida. Asimismo, el Web pone a disposición masiva al cliente/servidor a lo grande.

¿QUÉ MODALIDAD DE CLIENTE/SERVIDOR?

En el Capítulo 2, página 20, expusimos la idea de que cliente/servidor se halla en transición de la *era de Ethernet* a la *era intergaláctica*. Dedicamos después la mayor parte de este libro a explicar las tecnologías de cliente/servidor que pueden producir este cambio. ¿Cuál es entonces el punto básico?

Proponemos la Figura 35-1 como respuesta a la pregunta ¿qué modalidad de cliente/servidor? La era de Ethernet de cliente/servidor presenció una ola de aplicaciones centradas en archivos (la ola de NetWare), seguida por una ola centrada en bases de datos (la ola de Oracle); los monitores de TP y el groupware fueron olas menores. La gran ola siguiente es el Web de objetos. Creemos que, combinados con Internet, los objetos distribuidos son esenciales para hacer realidad la visión de cliente/servidor intergaláctico.

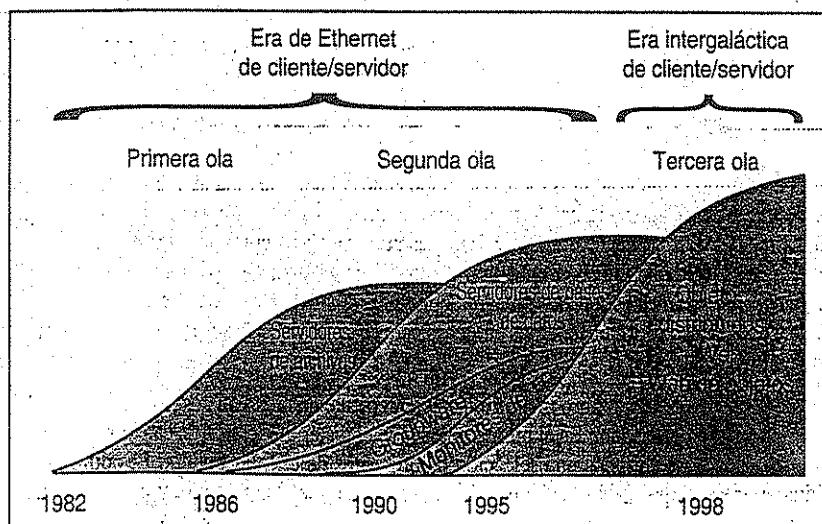


Figura 35-1. Olas de cliente/servidor.



Una vez que haya despegado, la tecnología del Web de objetos absorberá a todas las demás modalidades de computación de cliente/servidor, incluidos monitores de TP, bases de datos y groupware. Los objetos distribuidos pueden hacer todo esto, y mejor. Nos permitirán dividir grandes aplicaciones monolíticas en componentes de proveedores múltiples más manejables residentes y coexistentes en el bus intergaláctico. Representan también nuestra única esperanza para la administración y distribución de los millones de entidades de software que existirán en las redes intergalácticas. El Web es la gran aplicación que llevará los objetos hasta las masas.

¿En qué ola debo navegar?

No hace falta un recuadro de debate para afirmar lo obvio: el Web de objetos es el futuro de la tecnología de cliente/servidor. Los objetos y el Web comprenden todos los aspectos de la computación distribuida, entre ellos OOUI, documentos compuestos, transacciones, groupware, bases de datos y administración de sistemas. De este modo, la pregunta por responder es: ¿conviene que invierta usted en tecnologías intermedias o debe sumarse de una vez por todas a la ola del Web de objetos? Se trata de una disyuntiva entre el uso de tecnologías probadas y de aquellas otras "a punto de salir". En verdad nos encontramos en una dolorosa coyuntura tecnológica. La combinación de paradigmas suele desembocar en la obtención de lo peor de ambos mundos. Como nuestros alumnos, nos urge navegar de inmediato por la ola del Web de objetos y ver a dónde nos lleva. Pero al mismo tiempo oímos la voz de la prudencia, que nos dice que el Web de objetos aún no está listo para su momento estelar intergaláctico. El Web y objetos terminarán por fundirse, pero ¿queremos ser los primeros en navegar esta ola?

La cuestión de la facilidad de ampliación de cliente/servidor

Aparte de las olas tecnológicas, en la elección de una plataforma de cliente/servidor debemos tomar en cuenta la cuestión de la facilidad de ampliación. Es mucho más fácil hacer uso de cliente/servidor en pequeñas empresas y departamentos. Más del 80% de las instalaciones existentes de cliente/servidor son de servidor único y tienen menos de 50 clientes. Las instalaciones pequeñas son habitualmente más fáciles de usar y administrar. Casi todas las herramientas existentes de cliente/servidor atienden este mercado. Son útiles para la creación de aplicaciones cliente/servidor para el apoyo de decisiones, correo electrónico y groupware. Ofrecen excelentes facilidades para la formación de primeros planos con GUI y se prestan a operar con paquetes comerciales de middleware y servidor (principalmente DBMS de SQL y Lotus Notes). Sin embargo, e incluso en el nivel básico, carecemos de herramientas adecuadas para la creación de soluciones decisivas para el cumplimiento de objetos. Entendemos por esto aplicaciones de las que se puede depender en la realización de las operaciones diarias de negocios. Especialmente en el nivel básico, estos tipos de aplicaciones deben soportar transacciones, administración de sistemas integrada y alta disponibilidad.

Cliente/servidor intergaláctico es mucho más complejo. Requiere del sofisticado middleware descrito en este libro, como MOM, transacciones distribuidas y ORB. Requiere de herramientas probadas que puedan obtener provecho de este middleware. Cliente/servidor intergaláctico es, por definición, de servidores múltiples. Así, las herramientas deben ser capaces de utilizar



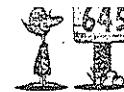
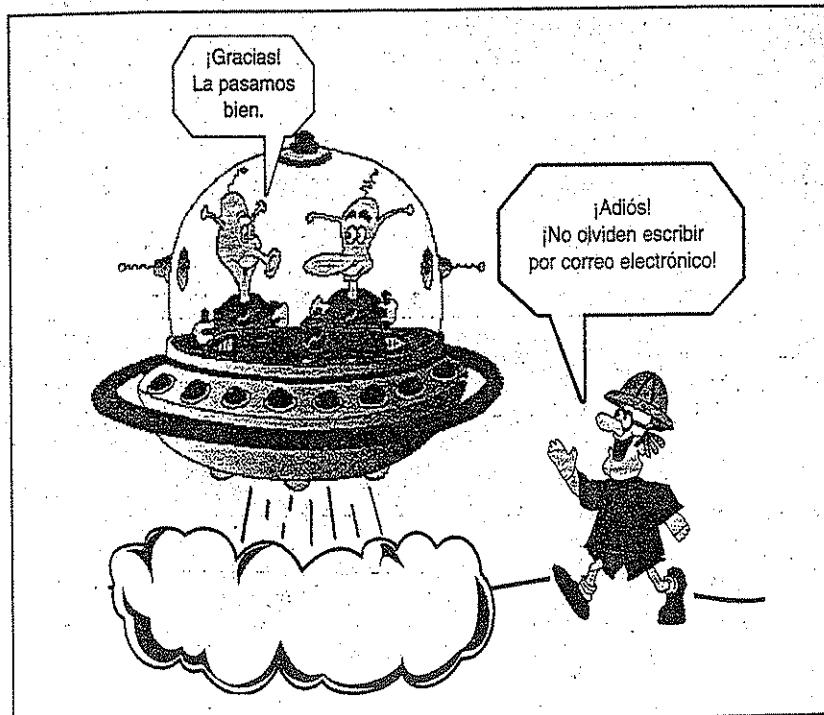
ES MOMENTO DE DECIR ADIÓS

La única manera de predecir acertadamente el futuro es inventarlo.

Alan Kay, gurú de PC

Sí, fue un viaje largo y accidentado. Esperamos que haya disfrutado de esta visita guiada tanto como nosotros gozamos haberle servido de guías. A nuestros amigos de Marte les deseamos un dichoso regreso a casa; fue un placer conocerlos. Esperamos que encuentren en alguna parte los tesoros de cliente/servidor. No olviden contarles a sus demás amigos marcianos de nuestro viaje guiado. Tenemos también un libro complementario para los interesados en objetos. Se llama *The Essential Distributed Objects Survival Guide*. (Wiley, 1996). Este otro libro, de 600 páginas de extensión, nos permitió darnos el lujo de explorar CORBA, OLE y OpenDoc a profundidad.

Por más que buscamos, no encontramos las sabias palabras con las que nos gustaría concluir este largo recorrido. Pero en realidad no habría mucho que añadir a todo lo ya dicho. Sólo quisieramos decir que éste fue nuestro intento por hallarle sentido al traumático cambio por el que atraviesa la industria de la computación. Se trata de un cambio difícil para muchos de



nosotros; para otros es el comienzo de un nuevo día en la computación, con el cielo como único límite. Bueno, pero ya basta de sensiblerías. Nos despedimos de ustedes con un recuadro de debate de partida acerca de la dirección que está siguiendo todo esto.



¿Sobreviviremos a la revolución de cliente/servidor?

Debate

Sí, pero para hacerlo necesitamos 100,000 nuevos applets. ¿Cómo alcanzaremos ese número? La tecnología de cliente/servidor hace posible redesplegar la mayoría de nuestras aplicaciones de cómputo en hardware de pequeño tamaño, donde los márgenes de ganancia son del grosor de una navaja. Pero si recomputarizamos la misma base de aplicaciones, usando LAN de PC e intranets en lugar de macrocomputadoras, la mayoría de nosotros nos quedaremos sin trabajo. Esto se debe a que ahora perseguimos un pastel más chico, en el que las utilidades son considerablemente menores. Si nuestras ganancias se reducen, eliminaremos la investigación que nos permita crear esas nuevas tecnologías. En consecuencia, nuestros clientes no dispondrán de nuevas aplicaciones ni nueva tecnología. Todos saldremos perdiendo. A esto se le llama el "efecto caníbal".

En cambio, necesitamos aprovechar la tecnología de cliente/servidor para extender las fronteras de la computación. En otras palabras, debemos movernos hacia nuevas fronteras, como la carretera de la información. Pero, ¿podremos crear rápidamente miles de nuevas aplicaciones cliente/servidor para poblar esas nuevas fronteras? Para conseguirlo, requerimos una base tecnológica, estándares y herramientas. Esta guía demuestra que la base tecnológica, los estándares y algunos sólidos productos ya existen. Lo que falta son las herramientas adecuadas para la explotación de la nueva frontera. No es suficiente con crear la aplicación; también se le debe empaquetar, utilizar y administrar eficazmente.

¿Quién va a empaquetar, utilizar y administrar estas aplicaciones? Sería absurdo esperar que todas las personas que han leído íntegro este libro lo han hecho para volverse expertas en cliente/servidor (no nos malinterprete; nos encantan las ventas). En cambio, debemos simplificar el empaquetamiento y distribución de nuestros productos de cliente/servidor. De todas las tecnologías expuestas en este libro, el Web de objetos es el que ofrece las mejores oportunidades para la creación —en tiempo récord— de nuevas aplicaciones cliente/servidor capaces de llegar más lejos que cualquier otra aplicación. A esto se debe que seamos optimistas en cuanto a las perspectivas de largo plazo. Pero a corto plazo enfrentaremos dificultades. Así, tendremos que detenernos hasta que descubramos cómo desencadenar el verdadero poder de esta tecnología. Es una lástima, pero así es. Lo bueno es que una vez superado ese periodo crítico, quienes permanezcan gozarán de una nueva fiebre del oro. Como dice el proverbio chino, "estamos condenados a vivir tiempos interesantes". □



administrador de base de datos, 589
 bases de datos (DBA), 205, 210-211
 control de servicios (SCM), 454
 flujo de trabajo, 268, 270
 objetos de X/Open (XOM), 612
 definición, 611-612
 paquetes, 612-613
 recursos de comunicación, definición, 286
 Administrator for Networks, 609
 Adobe, 438
 ADSL. Ver Línea de suscripción digital asimétrica
 ADT. Ver Tipos de datos abstractos
 AFS. Ver Sistema de archivos Andrew
 agente, 21-22, 226, 233
 definición, 584-585
 del sistema de directorio (DSP), 106
 del usuario del directorio (DUA), 106
 información, 233
 nómadas, 22, 393
 SQL, 153
 agregación, 392, 454
 aislamiento, definición, 258
 AIX, 92, 143, 169, 193, 238, 247-248, 250, 291, 314-315, 570
 alerta, 169
 Alexander, Christopher, 434, 639
 Al-Ghosein, Mohsen, 176, 304-305
 almacenamiento estructurado, 432
 Almaden Research, 272
 Alpha, 90
 alta velocidad, Ethernet de, 40

America Online (AOL), 506, 558
 ancho de banda, 33-54, 564
 Anderson, Martin, 385
 Andreas, Bill, 405
 Andreessen, Marc, 366, 468, 488, 526, 549, 567
 anexos, definición de, 331, 359
 ANSA, 536-537
 ANSI, 44, 152
 AnyNet/2, 38, 197
 AOCE, 369
 Apache, 566
 Apertus Technologies, 222
 API
 de administración consolidada (CM-API), 612
 de administración de X/Open (XMP), 612
 de administración de X/Open (XOM), 106
 de servicios de seguridad genéricos (GSSAPI), 143
 de telefonía, 352
 del cliente, 342
 del servicio de directorio de X/Open (XDS), 106
 para manejo de mensajes (MAPI), 348-350, 363-364, 370
 definición, 348
 extendida, 349
 aplicación, 542
 ayudante
 capa de, 115
 desarrollo de, 627, 631-634, 641-643
 servidores de, 14, 86-87
 aplicaciones
 de administración, 588
 estructuras de, 393
 habilitadas para correo, 343
 definición, 346

aplicaciones/transacciones, interfaz de administración de (ATMI), 282
 definición, 287
 Apollo, RPC de, 136
 APPC, 116, 282, 357
 Apple, computadoras, 86, 348, 438, 441, 534, 543-546, 567, 569-570, 609
 Apple Events, 441
 AppleScript, 369
 applet, 517, 519, 551, 564
 AppleTalk, 34, 84, 604
 Application Systems (AS), 248
 APPM, 34
 Approach de Lotus, 239
 arabica, 439, 571
 Arbor, 230, 249
 archivos estructurados, 546, 555, 563
 Ardis, 53
 arquitectura común de corredores de solicitudes de objetos (CORBA), 92, 93, 135, 294, 305, 309-311, 314, 316, 374, 385, 388-390, 397-426, 549, 554, 562, 567-568, 587, 594, 607, 613-615, 619, 630, 635-636, 643
 acoplamiento de lenguaje, 404
 adaptador básico de objetos (BOA), 409
 adaptador de objeto, 409
 administración de sistemas, 618-621
 beneficios, 403
 bus de objetos, 397-426
 cliente/servidor, 400, 406
 código del servidor, 425
 columnas vertebrales, 411

arquitectura común de corredores de solicitudes de objetos (CORBA) (continuación)
 contra DCE, 410-411
 contra DCOM, 410, 449-458
 contra OLE, 410, 449-458
 DCE/ESIOP, 411
 deficiencia, 425
 depósito de implementaciones, 409
 depósito de interfaces, 400, 404, 407, 456
 e Internet, 411
 e intérpretes, 408
 ECI gateway, 311
 esqueletos, 408
 estructuras, 417
 facilidades comunes, 402-403, 417
 federación de ORBS, 410
 firmas de métodos, 407
 guía de arquitectura de administración de objetos (OMA Guide), 403
 HTTP, gateways, 537
 identificadores de depósito, 408
 IDL, 399, 419, 524
 implementaciones comerciales, 424
 inter-ORB, 404, 409
 interfaz de esqueleto dinámico (DSI), 408
 interfaz de invocaciones dinámicas (DII), 407
 interfaz de ORB, 407, 409
 interfaz estática, 408
 invocación de método dinámica, 403, 407, 456
 invocación de métodos estáticos, 403, 407
 medio puente, 409
 mensaje semántico, 425
 mercados verticales, 417
 metadatos, 400-404
 middleware, 416
 objetos, 399
 objetos de negocios, 421
 ORB, 568
 ORBlet, 465, 532, 535, 549
 RFP, 414
 protocolo entre ORB en Internet (IIOP), 409, 411
 puentes, 409
 puentes genéricos, 408
 seguridad, 414-416
 servicios, 398-399
 servicio de ciclo de vida, 412
 control de concurrencia, 412
 eventos, 412
 externalización, 413
 licencias, 413
 nombramiento, 412
 objetos, 397-426
 y middleware, 416
 persistencia, 412
 propiedades, 413
 seguridad, 404-413
 transacciones, 412
 transacciones de objetos (OTS), 309, 314, 316, 419
 servicios de consulta, 413
 servicios de relaciones, 413
 talones, 408
 transacciones, 404
 transparencia local/remota, 404
 versión 1.1, 403
 versión 2.0, 399, 404
 y cliente/servidor, 403-404, 408
 y componentes, 402
 y el Web, 533-556
 y Java, 532, 538-539
 y metaclasses, 425
 y MOM, 425
 y OpenDoc, 438, 445, 447
 y transacciones, 317
 arquitectura de bases de datos relacionales distribuida (DRDA), 17, 179, 195-197, 198, 248
 cliente, 196
 definición, 196
 servidor, 196
 arquitectura de creación de scripts abierta (OSA), 439
 arquitectura de redes de sistemas (SNA), 193
 LU 6.2, 121
 asignar nombres, 103, 126
 asimétrico, multiprocesamiento, 64-65
 asíncrona, modo de transferencia (ATM), 43, 44-48, 54
 estándares, 46
 Forum, 46
 ASN.1. Ver Notación de sintaxis abstracta uno
 AST, 609
 At Work, 340
 AT&T, 53, 208, 244, 582
 GIS, 238, 307, 310-312, 315-316
 GIS 3600, 239, 249-250
 Paradyne, 51
 Atlas, 617-618
 administración distribuida (Atlas-DM), 594, 617-618
 definición, 617-618
 ATM, 564
 Ver Asíncrona, modo de transferencia
 atomicidad, definición de, 258



cliente/servidor
 (continuación)
 servidores de groupware,
 14
 soluciones para los
 negocios, 1, 55
 recursos compartidos, 11
 transparencia de ubicación,
 11
 y CORBA, 400, 403-404,
 406
 y objetos, 398
 y OpenDoc, 444-445

CLSID. Ver *Identificador de clase*

Club Med, 21

CM-API. Ver *API de administración consolidada*

CMC. Ver *Llamadas comunes de correo*

CMIP
 sobre LLC (CMOL), 608
 sobre TCP/IP (CMOT),
 608

Ver *Protocolo de información de administración común*

CMIS. Ver *Servicios de información de administración común*

CMOL
 definición, 608
 Ver *CMIP sobre LLC*

CMOT
 definición, 608
 Ver *CMIP sobre TCP/IP*

COBOL, 244
 codificación, 110, 141, 365,
 502, 511-512

con llave pública, 112,
 503-506, 512
 de llave secreta, 141
 de RC4, 503

Coffee, Peter, 447

Cognos, 227, 229

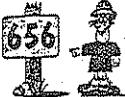
colas de mensajes, 113,
 126-128, 130
 colecciones, 155
 Coleman, David, 324, 326
 Colusa, 519
 Collabra, 370-373
 agente de duplicación, 371
 asistente de búsqueda, 371
 Forums, 371
 Collabra Share, 320,
 353-354, 370-371
 Comaford, Christine, 227,
 230, 472, 516, 541
 comercio electrónico,
 21, 309, 506, 509, 559
 COMlets, 465
 Commander OLAP, 230, 233
 CommerceNet, 505, 512
 COMMIT, 153
 Commshare, 230
 Compaq Computer Corp.,
 609
 compilador de metadatos,
 definición, 613
 Component Integration
 Laboratories (CI
 Labs), 389, 426, 438-
 439,
 447-448, 549, 554
 componentes, 238, 381, 398,
 402, 527-528, 564
 administración de propie-
 dad, 391
 autocomprobación, 392
 beneficios, 381, 387
 caja blanca, 389
 caja negra, 389
 ciclo de vida, 391
 conducir la revolución de,
 384
 configuración, 391
 control de transacciones,
 392
 herramientas, 448
 infraestructura, 385, 393
 introspección, 392, 398
 licencias, 391
 mensajes semánticos, 392
 metadatos, 392
 minimalista, 389
 negocios, 448
 nivel del sistema, 418
 notificaciones de eventos,
 391
 oportunidad de negocios,
 387
 paleta abierta, 391
 paquetes, 389
 persistencia, 392
 relaciones, 392
 scripts, 392
 seguridad, 391
 series, 381, 388-389, 393
 servicios de sistema, 393
 superinteligentes, 393
 versiones, 391
 y cliente/servidor, 435
 y CORBA, 402
 y documentos compuestos,
 428-429
 y groupware, 374
 y objetos de negocios, 393,
 418
 ComponentGlue, 443-444
 CompuServe, 20, 350, 506,
 558
 Computer Associates (CA),
 237, 623
 Computron, 343
 Comshare, 233
 comunicación entre procesos
 (IPC), 60
 conectividad de base de datos
 abierta (ODBC), 17,
 175, 179, 183, 184-
 187, 189, 192, 238,
 247-248, 363, 450,
 530-531, 552
 controladores, 187
 nivel 1, 185
 nivel 2, 185
 núcleo, 185

conectividad de base de datos
 abierta (ODBC)
 (continuación)
 SDK, 186
 versión 1.0, 185
 versión 2.0, 185
 versión 3.0, 185
 y Java, 532
 conexión y manejo, 84, 609
 conexiones (plug-ins), 542,
 554
 conferencias, 328, 352
 definición, 350
 comutación de paquetes,
 44-47
 comutación de Token Ring,
 42
 comutadores de LAN, 40
 consistencia, definición, 258
 consultas
 desbocadas, 227
 recurrentes, 155
 contenedor, 429, 543, 554,
 563
 contraseña secreta compar-
 tida, 506
 control
 de acceso a medios
 (MAC), 115
 de vinculación lógica
 (LLC), 15
 controlador, 420
 base de datos, 178, 187
 SQL, 173, 176
 controladores de bases de
 datos, definición, 187
 conversaciones, 115
 coordinador de transacciones
 distribuidas (DTC), 240
 Coordinated Computing, 406
 Copland, 569
 Copy Manager, 222
 CORBA. Ver *Arquitectura*
común de corredores
de solicitudes de
objetos
 corredor de solicitudes de
 administración
 (MRB), 614-615
 corredor de solicitudes de
 objetos (ORB), 15, 17,
 135, 162, 311, 393,
 403, 424, 515-532,
 533-556, 601, 641
 Atlas, 618
 contra RPC, 405-406
 definición, 380
 DME, 614-615
 mecánica, 406
 posicionamiento, 380
 red principal, 409
 y Java, 515-532, 533-556
 y el Web, 533-556
 y monitores de TP, 425
 corredores de objetos, 63
 correo electrónico, 343, 348,
 349, 352, 369, 486
 almacenamiento de
 mensajes, 347, 349
 API, 346, 349
 definición, 343
 gateways, 365
 libretas de direcciones, 348
 mercados, 343
 red principal, 345
 seguridad, 348
 servidor, 356
 correspondencia de priva-
 cidad garantizada, 511
 Covia, 17, 126
 Cox, Brad, 387
 CPI-C. Ver *Interfaz común*
de programación para
comunicaciones
 creación rápida de prototipos,
 633-634
 criptografía, 365
 Croft, Dan, 53
 CrossTarget/Diver, 230
 CSA. Ver *Planificación y*
calendarización API
 CSE Systems, 343

CT Object Framework, 623
 Cubbage, Paul, 86
 cursores persistentes, 156
 CyberCash, 510
 Cyberdog, 85-86, 534,
 543-546, 548-549,
 552, 554, 570
 CYCLADE Consultants, 418
 Chappell, David, 410

D

Danny De Vito, herramientas,
 205
 DAP. Ver *Protocolo de acceso*
al directorio
 Data Based Advisor, 469, 486
 Data Dictionary, 211
 Data Interpretation Systems
 (DIS), 248
 Data Mover, 222
 Data Shopper, 211
 DataAnalysér, 233
 Database Programming and
 Design, 197, 217
 Database Valley, 251
 DataBlades, 241
 DataDirect, ODBC/ClassLib,
 186
 DataDiscovery, 233
 datagramas, 118, 120, 598,
 604
 DataGuide, 211, 248
 DataHub, 180, 198, 247
 DataJoiner, 222, 243, 247
 DataLens, 363
 Datamart, 212, 248
 Datamation, 228, 454
 DataPivot, 229
 DataPrism, 227
 DataPropagator, 222
 NonRelational, 247
 Relational, 246
 Dataquest, 86, 323-324, 352
 DataRefresher, 222, 247



estándares inalámbricos, 53
estructura
de administración de OSI, 605, 607-608
de flujo de trabajo de MAPI, 349
de información de administración (SMI), 594, 595, 602
estructuras, 315, 388, 427, 435-436
Ethernet, 39-40, 51
ETI, 211, 222-223, 249
evaluaciones comparativas, 302
evento, 169, 618
administración, 592
administradores de eventos, 314
eventos semánticos, 441, 446
Excel, 185, 248, 316, 384, 569
Exchange, 106, 320, 340, 353-354, 372-374
carpetas públicas, 372
duplicación, 373
Expersoft, 15, 423
Explorer, 534, 543, 550, 569
Express, 230
Express MDB, 230
Express Technologies, 249
extensiones del correo de Internet de propósitos múltiples (MIME) 354, 478, 486, 498, 505
Extract, 222-223
Extract and Metadata Exchange, 211

F
fabricantes de IS, 385-387
Falksen, Gary, 593
FAP. Ver *Formato y protocolos*

FastTrack, 566
Fax, 365
FDDI. Ver *Interfaz, de datos distribuida por fibra*
federación
base de datos en, 177, 180, 214-215, 217
monitores de TP en, 291
nombramiento en, 103
Federal Express, 499
Ferguson, Charles, 373, 463
fibra, 49
Fielding, Roy, 478
FileNet, 332, 340, 369
filtros de paquetes, 508-509
Finkelstein, Richard, 89-90, 149, 160, 189-190, 299
firewall. Ver *Muro de protección*
firma
digital, 112, 506, 551
electrónica, 502
First Virtual, 511
Forrester, 351
Flores, Fernando, 338
Flow Mark, 247, 340
Floware, 340
flujo de trabajo, 212, 247, 328-329, 334, 352, 421, 445, 631
actividades, 341
ad hoc, 336
coalición, 341
definición, 332-333
e imágenes, 332-333
e imágenes electrónicas, 331
funciones, 334
herramientas, 341
mecanismos, 341
modelos, 335, 338
negociación, 338
nuevos paradigmas, 333
órigenes, 331-333
procesos, 341-343
reglas, 335
repetición de trabajos, 336
roles, 335
rutas, 335-336
servidor, 334
y groupware, 333
y MOM, 334
fluxos, 457
Focus, 227
Fore Systems, 46
Forest & Trees, 227, 233
Forester Research, 294-295, 515
formato
de imagen gráfica (GIF), 475, 476
y protocolos (FAP), 173, 177, 179-180, 196, 284
definición, 173
formatos, campos ocultos, 500
Formflow, 340
foros, 370
Forrester, 351
Forte, 309
Four Season Software, 309, 315
FoxPro, 236
Frame Relay. Ver *Relé de trama*
freeware, 512
Froemming, Glenn, 214, 222
Frye, 609
FT Systems, 302
FTP. Ver *Protocolo de transferencia de archivos*
Fuller, Art, 469, 486
funciones, 155
de servidor de base de datos de SQL, 157

G
Gamma, Erich, 401
García-Molina, Héctor, 272

Gartner Group, 215, 222, 235, 238, 243, 304, 316, 353, 388, 411, 450, 512-513, 538, 556, 570, 579, 582-583, 588, 623
Gates, Bill, 21, 316, 460, 548, 556, 558
gateways, 34, 38, 174-175, 299, 537
correo electrónico, 365
Gupta, 174
Informix, 174
Ingres, 158, 168, 190
MDI, 175
Oracle, 174
proveedor, 175
servicios de datos abiertos (ODS), 175
servidor de datos abiertos (ODS), 175
Sybase, 174-175
XDB, 174
GDMO, 613
definición, 607
GDS. Ver *Servicios de directorio global*
Gelernter, David, 583-585
General Magic, 519
gente, lugares y cosas, 547
Gibraltar, 552
GIF. Ver *Formato de imagen gráfica*
Gilder, George, 54
GlobeSpan, 51
gobernador de consultas, 227
Goldrush, 239
Gopher, 466, 470
Gosling, James, 517-518, 522, 525, 532
Gould, Michael, 501, 534, 554
grabación en dos fases, 128, 166, 214, 261, 269, 280, 284, 328, 361
definición, 264
limitaciones, 266
mecanismos, 264
y MOM, 269
grabación
administrador, 264
coordinador, 288
grabaciones delegadas, 266, 281, 290
Graham, Ian, 418
GRANT, 153
Gray, Jim, 183, 257-258, 262, 266, 279, 281, 296, 298, 396
Group Bull, 613
GroupTalk, 324
groupware, 14, 319, 351, 361, 631, 640
base de datos de documentos, 326
conferencias, 350
definición, 326
e imagen electrónica, 330
historia, 324
Internet, 354, 370, 373
memoria colectiva, 324
mercado, 324, 357
monitores de TP, 355
Web, 355
y componentes, 355
y reingeniería, 325
GroupWise, 340, 345, 353-354
de consulta, 225-234
Hewlett Packard (HP), 15, 136, 143, 208, 211, 311, 313, 345, 349, 407, 411, 423, 567, 609, 613, 624
híbridos, clientes, 77
Hickman, Angela, 52
hilos, 60, 142
hipertexto, 65, 487
hipervínculos, 463, 476-477, 487
HiTest, 357
Hoffman, Mark, 249
Holistic, 249
Holistic System, 230

GSSAPI. Ver *API de servicios de seguridad genericos*
GUI. Ver *Interfaz gráfica del usuario*
guía de la arquitectura de administración de objetos (OMA.Guide), 402
Gupta, 149, 152, 158, 167, 174, 195, 227, 251

H

Hackathorn, Richard, 209
Haderle, Don, 171, 225
Hades, 140
Hahn, Eric, 370
Halfhill, Tom, 460
Hambrecht and Quist, 560-561
Harris, Jed, 389, 439
Heinen, Roger, 550
herencia, 454
múltiple, 416
Herman, James, 580
herramientas, 448, 627-628, 630-634, 642-643, 645
de consulta, 225-234
Hewlett Packard (HP), 15, 136, 143, 208, 211, 311, 313, 345, 349, 407, 411, 423, 567, 609, 613, 624
híbridos, clientes, 77
Hickman, Angela, 52
hilos, 60, 142
hipertexto, 65, 487
hipervínculos, 463, 476-477, 487
HiTest, 357
Hoffman, Mark, 249
Holistic, 249
Holistic System, 230



Java (*continuación*)
compilador, 520
componente, 525, 527
conexión de base de datos
de Java (JDBC),
530-532
contenedor, 525
entorno de objetos Java
(JOE), 532
herramientas, 516
hilos, 524
interfaz, 524
interfaz Runnable, 525
Java.applet, 530
Java.awt, 530
Jkava.io, 530
Java.lang, 530
Java.net, 530
Java.util, 530
Kona, 529
lenguaje, 524
listas de control de acceso,
523
máquina virtual, 520, 528
MicroJAVA, 528
modo anfitrión, 523
muro de protección
(firewall), 523
nombrar, 524
ORB, 532
ORBlets, 532
paquetes, 524
PC, 528
PicoJAVA, 528
recolección de basura, 524
seguridad, 522-524
sistema operativo, 528-
530, 532
UltraJAVA, 528
verificador, 520-522
y C++, 524
y CGI, 534
y CORBA, 532, 538-539
y documentos compuestos,
541, 549
y ODBC, 532

y ODMG, 532
y OLE, 516, 543
y OpenDoc, 439, 516, 543
y ORB, 515-532, 533-556

Manager/X, 120
NetView, 580, 581, 613,
624

Server, 16, 118

LANcity, 51

LANDesk Management
Suite, 624

LaserData, 340

Laube, Sheldon, 312

LeFevre, Dave, 393

Legent, 222, 623-624

lenguaje de
aplicación SQLWindows
(SAL), 167

consulta de objetos (OQL),
413

definición de datos (DDL),
153

definición de interfaz de la
red (NIDL), 122

descripción de objetos
(ODL), 452

manipulación de datos
(DML), 153

lenguaje de consulta estructu-
rada (SQL), 150, 457

acoplamiento 156

agente, 153

alargadores, 555

BLOB, 154, 327

catálogo del sistema, 151

catálogos, 154

CLI, 179, 181-183, 190

conexiones, 153

controlador, 173, 176

cursor, 153

definición, 151

desencadenantes, 167, 169

esquema, 154

estándares, 151

FAP, 177, 179-180

gateways, 175

GET DIAGNOSTICS, 154

grupo, 154

herramientas, 174

herramientas frontales, 174

K

Kador, John, 403

Karat, 624

Kay, Alan, 644

Kezan, 230

Kerberos, 108, 110, 140-142,
506

versión 5, 143

Ki, redes, 609

Kimball, Ralph, 211, 224

Kingsley, Idehen, 184

Kona, 81

Koulopoulos, Tom, 340

L

Labovitz, John, 480

LAN

Directory, 609

Inventory, 609

Manager, 119, 134

lenguaje de consulta estruc-
turado (SQL)
(*continuación*)

integridad de referencias,
151, 169

nivel de aislamiento, 153

origenes, 150

plan, 157

procedimientos almacenados, 163, 165, 167

protección, 153

reglas, 167, 169

restricciones de dominios,
154

restricciones de verifica-
ción, 154

SELECT, 227

SQL dinámico, 154, 164

SQL estático, 164

SQL incrustado, 152, 154,
178, 182

SQLSTATE, 154

tablas temporales, 154

transacción, 153

unión, 154

lenguaje de definición de
interfaces (IDL), 136,
311, 390, 399, 404,
438, 630, 635

Ada, 399

C, 399, 407

C++, 399, 407

COBOL, 399

identificadores de depósi-
tos, 408

Objective C, 399

Smalltalk, 407

lenguaje de marcación de
hipertexto (HTML),
63, 309, 465, 467,
471-477, 491-495,
527, 543, 554

anclas, 476-477

conexiones, 542

documento, 472

estándar, 477

tablas, 488, 495

versión 1.0, 477

versión 2.0, 477, 489-490

versión 3.0, 477

etiquetas

A, 476-477

APPLET, 517, 525-527

B, 471

BODY, 472

DIR, 475

DL, 475

DYNsrc, 527

EMBED, 527

FORM, 490-494

H1, 472

H2, 472

H3, 472

H4, 472

H5, 472

H6, 472

HEAD, 472

HR, 473-475, 490

Href, 476-477

I, 471

IMG, 475-477, 527

INPUT, 492-493

INSERT, 527-528

LI, 475

MENU, 475

OL, 475

P, 473-474

PRE, 473-474

SELECT, 494

STRIKE, 471

TABLE, 495

TEXTAREA, 494

TITLE, 472

TT, 471

U, 471

UL, 475

WEBOBJECT, 540

formatos, 488, 490

(HTML+), 477

marcos, 542

puntos importantes,
477

tablas, 488, 495

versión 1.0, 477

versión 2.0, 477, 489-490

versión 3.0, 477

Lewis, Jamie, 42, 373, 557,
609

Lewis, Peter, 565

ley de telecomunicaciones de
1996, 51, 54

LFS. Ver *Sistema de archivos,*
local

licencias, 391, 413

LightShip, 233

Server, 230

Lindsay, Bruce, 272

línea de suscripción digital
asimétrica (ADSL), 49, 51

de cargo de bits (HDSL),
49, 51

Linthicum, David, 307, 427,
443

listas de control de acceso
(ACL), 109, 141, 314

LiveWire, 566

Pró, 566

locales/remotos, transparen-
cia de procesos, 60

localizador uniforme de
recursos (URL),
467-470, 476-477,
503, 527-528,
530-531, 542

Lockert, Dick, 205

Lorin, Hal, 96, 122

Lotus, 193, 340, 345, 348,
364, 438, 543, 556,
570-571

cc:Mail, 345

Lotus 1-2-3, 384

Lotus Notes, 17, 38, 222,
309, 324-325, 331,
340, 352, 359, 373-
374, 556, 570-571

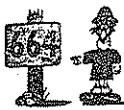
4.0, 350, 355-357, 366

acceso a base de datos de
SQL, 363

almacenamiento de mensa-
jes, 347

anexos, 358

API, 363



monitores de TP
(continuación)
desarrollo de herramientas,
309
disponibilidad, 292
equilibrio de cargas, 279,
293
estructura, 292
estándares, 291
evaluaciones comparativas,
302-303
fallas, 280
federación, 291
fluxos de trabajo, 328
herramientas, 304
groupware, 355
mercado, 307-318
procesos, 298
productos, 291, 307-318
protección, 292
RPC, 280
SMP, 293
tendencias, 309
transacciones, 279
vendedores, 307-318
y flujo de trabajo, 328
y MOM, 293
y objetos, 305, 309, 316
y ORB, 425
y tres planos, 284
Moore, Larry, 343
Mosaic, 464
Motorola, 51, 53
Mowbray, Thomas, 399, 454
MPP, 317
Ver Proceso paralelo masivo
MPTN. *Ver Red de transporte de protocolo múltiple*
MQI, X/Open, 290
MQSeries, 130, 282, 311,
374, 570-571
Three Tier (MQ3T), 130
MRB. *Ver Corredor de solicitudes de administración*

MSN. *Ver Microsoft Network*
MSS. *Ver Sistema de apoyo de decisiones*
multicanónico, 124
multidimensional, acceso,
228-229
multimedia, 77, 215, 327, 330
documentos, 327
multitarea preferente, 59
MUMPS, 182
mundo virtual, 75
muro de protección, 41, 63,
466, 502, 506-509,
523, 562
anfitrión bastión, 509
MVC. *Ver Modelo/vista/ controlador*
MVS, 88, 247, 311

N

Named Pipes, 16, 116, 120,
173, 175, 193
Navigator, 566
Gold, 566
NCR, 315
NCSA (Centro Nacional para Aplicaciones de Super-cómputo), 566
NDIS, 36-37
inalámbrica, 53
NDR. *Ver Representación de datos de red*
NEC, 609
Negroponte, Nicholas, 45
Nelson, Ted, 80
NEO, 15, 423, 535, 567-568
NEOnet, 535
Nested NetWare, 81
NetBEUI, 83-84, 120, 193,
357
NetBIOS, 34, 116, 118-120
bloque de control de red
(NCB), 120
comandos, 119-120

Netcom, 558
NetCommerce Server, 570
NetManage, 554
NetMap, 233
Netscape, 320, 353-354, 366,
370-371, 464, 467-
468, 488, 502-503,
506, 512, 516, 526,
534, 549, 556, 566-
567

Commerce Server,
503-504
extensiones, 488, 503
marcos, 542
Navigator, 488, 503
seguridad, 503-504
y CORBA, 568

NetView, 613, 623
definición, 581
puntos de entrada, 580
puntos de servicio, 581
puntos focales, 581

NetView/6000, 613
NetView/PC, 581
NetWare, 16, 87, 119,
133-135, 143, 236,
313, 619
3.X, 134
4.1, 89
servicio de directorio de
(NDS), 106, 135,
313
SMP, 238

NetWare Manager System
(NMS), 582-583

NetWare TransactionalLink,
313

Network General, 600
Network OLE 15, 84, 90,
143, 378, 450, 539,
550, 552, 636

New Era, 357
New York Times, 565
News, 466, 469
NeXT, 466, 534, 539-540

NFS, 102, 135

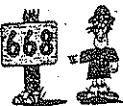
NIDL. *Ver Lenguaje de definición de interfaz de la red*
Nielsen, Henrik Frystyk, 478
NIS, 412
NLM. *Ver Módulos cargables de NetWare*
NMS. *Ver NetWare Manager System*
NMVT. *Ver Transporte de vectores de administración de redes*
nómadas, agentes, 22, 393
nombramiento en forma de árbol o jerárquico, 103
Norton Administrator, 624
Norton-Lambert, 609
NOS. *Ver Sistema operativo para redes*
notación de sintaxis abstracta uno (ASN.1), 195,
595, 606, 613
Notes
cc:Mail, 356
Mail, 356
VIP, 357
NotesView, 367
notificaciones, 618
Novell, 81, 195, 310-314,
320, 340, 345, 348,
353-354, 369-370,
393, 438, 609, 623
NSAPI, 534
número de puerto, 470
Nynex, 50

O

O'Connell, Brian, 571
O'Reilly and Associates, 480
Object Database Management Group (ODMG), 413
acoplamientos de JAVA, 532
Object Management Group (OMG), 17, 397-398,

418, 420, 438, 535-
536, 549, 554, 624
Business Object Task Force, 417
Object Technology International, 387, 433
ObjectBroker, 15, 409, 423
Objective C, 399, 540
objeto
administración de sistema,
381
ciclo de vida, 413
conurrencia, 412
estructuras, 380, 624
eventos, 412
externalización, 413
licencias, 413
nombramiento, 412
persistencia, 412
persistente, 457
propiedades, 413
relaciones, 413
seguridad, 413-416
servicio de consulta, 413
transacciones, 412
y cliente/servidor, 398
objetos, 15, 63, 105, 375, 573
y el Web, 515-532
y groupware, 374
y monitores de TP, 309
objetos administrados,
definición, 595
objetos binarios grandes
(BLOB), 62, 154, 156,
227, 236, 238, 327,
330-331, 345-346,
359, 563
consulta de, 227
herramientas de GUI, 327
objetos de negocios, 229
Ver Vinculación e Incrustación de objetos
OLE Team Development, 450
OLE Transactions, 450
OLE/DB, 185, 187
OLE/Transactions, 316

cooperadores (CBO),
418
definición, 418
facilidad de, 422
objetos distribuidos, 375,
573, 640
definición, 382
temas de seguridad, 414
y componentes, 382
objetos de presentación, 421
objetos de procesos de
negocios, 421
OC12, 44
OC48, 44
OCI. *Ver Interfaz del nivel de llamada de Oracle*
OCX 96, 553-554
ODBC, 188
Ver Conectividad de base de datos abierta
ODBMS. *Ver Sistema de bases de datos de objetos*
ODL. *Ver Lenguaje de descripción de objetos*
ODMG. *Ver Object Database Management Group*
ODS. *Ver Servidor de datos abiertos; Servicios de datos abiertos*
Office, 569
Oficina de Seguridad Nacional (NSA), 111
Ogden, Carol, 351
OLAP. *Ver Procesamiento analítico en línea*
OLE, 74
DB, 450
para las empresas, 450
TP de, 240, 451
Ver Vinculación e Incrustación de objetos
OLE Team Development, 450
OLE Transactions, 450
OLE/DB, 185, 187
OLE/Transactions, 316



principales o unidades autorizadas, 140

Prins, Rob, 418

prioridad, 59

Prism, 208, 211, 222, 244-245, 249

directorio de información, 244-245

Directory Manager, 245

Warehouse Manager, 243-251

procedimientos de almacenamiento, 62, 150, 156, 162-163, 165, 167, 176, 189, 191, 238, 284, 297-298, 300-302, 403

beneficios, 165

grabación, 298

inconvenientes, 165

y monitores de TP, 301

procesamiento analítico en línea (OLAP), 228-231

procesamiento de transacciones, 253, 319, 460

procesamiento de transacciones en línea (OLTP), 13, 163-164, 200-201, 205, 214, 262, 276, 358, 361

base de datos, necesidades, 206

datos, 201-202, 204

definición, 200

ligero, 200

pesado, 200

sistemas de producción, 202

proceso paralelo masivo (MPP), 317

Prodigy, 350, 558

producción, base de datos, 206

PROFS, 365

programación orientada a objetos (OOP), 380

programas cliente anatomía, 66

GUI simple, 68

OUI, 69

requerimientos de un OS, 76 sin GUI, 67

Progress, 196, 238

Project Sedona, 238

protocolo de

acceso a directorio ligero (LDAP), 566

acceso al directorio (DAP), 106

administración de X/Open (XMP), 594, 618 definición, 611-612

control de transmisión/protocolo Internet (TCP/IP), 502

pago electrónico seguro (SEPP), 512

reservación de recursos (RSVP), 48

tiempo real (RTP), 39

transferencia de archivos (FTP), 466, 470

transparencia de noticias en red (NNTP), 466

sistema de directorio (DSP), 106

protocolo de información de administración común (CMIP), 17, 587, 594-

595, 597, 612, 621, 624

acción, 608

creación, 608

definición, 605, 607

eliminación, 608

eventos, 607

filtración, 608

herencia, 607

objetos administrados, 607

obtención, 608

operaciones, 607

plantillas, 607

reporte de eventos, 608 sondeo, 607

protocolo de transferencia de hipertexto (HTMP)

63, 466, 467-468, 470, 478-480, 482, 484,

486, 496-499, 513, 536-537, 539-540

campos de encabezado, 480, 482

comando, 480

cuadro de entidad, 480, 484

definición, 478 e IIOP, 537

encabezados generales, 482

encabezamiento de solicitud, 480, 482, 484

encabezado de respuesta, 484

estado, 478-479, 499-500, 539

estándares, 478

gateways a CORBA, 537

HTTP 1.X, 478

HTTP/1.0, 478, 480, 482

HTTP/1.1, 478, 480, 482

HTTP-NG, 478

métodos, 480, 501

COPY, 481

DELETE, 481

GET, 481, 485, 488, 491, 501

HEAD, 481

LINK, 481

MOVE, 481

OPTIONS, 481

PATCH, 481

POST, 480, 488, 491, 498, 501

PUT, 480

TRACE, 480

UNLINK, 480

WRAPPED, 480

protocolo de transferencia de hipertexto (HTMP) (continuación)

puerto, 468

RPC, 478

servidor, 467, 498

solicitud, 480, 482, 484

protocolo entre ORB en Internet (IIOP), 409, 535-537, 555

y HTTP, 537

protocolo Internet (IP), 508-509

protocolo simple de administración de redes (SNMP), 17, 313, 315, 367, 594-595, 597-598, 600, 609, 612, 621

definición, 594, 597-598

GET, 597-598, 603

GET-NEXT, 598, 604

limitaciones, 598, 600, 602-603

operaciones, 597

SET, 598, 604

sondeo, 599

TRAP, 598, 604

protocolo simple de administración de redes versión 2 (SNMPv2), 587, 594-595, 604-

605, 621, 624

definición, 602-603

GET, 603

GET-BULK, 604

GET-NEXT, 604

INFORM, 604

operaciones, 603

SET, 604

TRAP, 604

versión 2* (SNMPv2*), 605

protocolo simple de transporte de mensajes (SMTP), 345, 356, 365, 466

protocolos

básados en sesiones, 117

de administración de Internet, 594-595, 597-598, 602-603

de descubrimiento, 120

entre ORB específicos de entornos (ESIOP), 409

proveedor de tiempo externo, 139

proveedores de servicios de Internet (ISP), 48, 558

independientes de software (ISV), 384-385

proxy, 407

basado en, 508-509

PSI, 558

PSM. Ver Módulos de almacenamiento persistente

publicación y suscripción, 313

publicar y suscribir, 372

puentes, 34, 120

punto de presencia (POP), 42, 51

puntos importantes, 477

Pyramid Technology, 88, 208

Q

Q+E Software, 185, 188, 227

QMF. Ver Query Management Facility

Quality Decision Management, 340

Query Dialog, 228

Query Management Facility (QMF), 248

Quest, 227

R

R/3, 340

radio en paquetes, 53

RAM Mobile Data, 53

rastreo de eventos, diagramas de, 418

rastros de auditoría, 109

RDA, 179, 186, 194

RDBMS. Ver Sistema de administración de bases de datos relacionales

Reach Software, 340

realización de formación, 407

RealTime Notes, 350

Recognition International, 340

recuperación de desastres, 592

recursos, administrador de, 265, 281

definición, 285

Red Brick, 208, 211, 213, 229, 231, 244

Brick Warehouse, 229, 244

Warehouse for Workgroups 213

red

celular, 52

de comunicaciones personales (PCN), 54

de transporte de protocolo múltiple (MPTN), 197

digital de servicios integrados (ISDN), 48, 50

BRI, 49

PRI, 49

digital de servicios integrados de banda ancha (B-ISDN), 47

inteligente global, 369

óptica síncrona (Sonet), 43, 47-48

redes de área amplia (WAN), 20, 33-54

de área local (LAN), 33-54

de valor agregado (VAN), 511



Solaris, 88, 90, 250, 314-315
 solicitantes, 101
 solicitud
 de propuesta (RFP), 414
 distribuida, 197
 Solstice, 535
 Enterprise Manager, 578, 624
 SOM. Ver *Modelo de objeto de sistema*
 SOM 3.0, 15, 423
 Sonet. Ver *Red óptica síncrona*
 SP2, 230, 238-239, 249-250
 SPARCserver, 239, 250
 Spec 1170, 93
 Spector, Alfred, 292, 298
 Spectrum, 582, 624
 SPI. Ver *Interfaz de proveedor de servicios*
 Spotlight, 233
 Sprint, 44
 SPX, 119
 Ver *Intercambio secuencial de paquetes*
 Spyglass, 467, 516
 SQL
 dinámico, 154, 162
 estático, 162, 198
 incrustado (ESQL), 154, 173, 178, 181-182, 189
 definición, 182
 Ver *Lenguaje de consulta estructurado*
 SQL, base de datos, 235-242, 640
 Web, 237
 y groupware, 326
 SQL 3, 238
 SQL Access Group (SAG), 155, 179, 183, 185
 CLI, 17, 179, 183
 definición, 183
 origenes, 183
 SQL para multimedia, 156

SQL Server, 159, 166, 168, 185-186, 237, 240-241, 244, 247, 316
 SUN RPC, 135
 SunNet Manager, 582-583
 SunSoft, Inc., 572, 624
 supercarretera digital, 64, 114, 553
 SuperNOS, 313
 SuperNOVA 4GL, 309, 315
 superservidores, 64
 Sutherland, Jeff, 520, 549
 SVR4. Ver *Unix SVR4*
 sweeper, 543, 550
 Switched Ethernet, 40
 Sybase, 65, 117, 143, 149, 151, 158-159, 163, 166, 168, 174-175, 186, 190-191, 195, 208, 222, 231, 238-240, 244, 247-248, 251, 296-297, 302, 309, 314-315, 340, 516, 532, 540, 566, 619, 631
 DB-Library, 175
 Enterprise Connect, 250
 Enterprise SQL Server Manager, 250
 IQ, 231, 251
 Stodder, David, 197, 217
 Stone, Chris, 397
 Strategic Focus, 387
 Stratus, 88
 Striker, 543
 STT. Ver *Tecnología de transacciones protegidas*
 subtransacciones, 272
 SuiteSpot, 566
 sumas de verificación criptográficas, 110, 141
 Summit Strategies, 86, 283
 Sun, 15, 93, 133, 135, 142, 239, 250, 407, 412, 423, 465, 516, 523-524, 528, 532-533, 535, 556, 567-568, 578, 609, 619

Symantec, 516, 609, 624, 631
 Symmetric Replication, 223
 syncpoint, 282
 definición, 271
 SysMan, 618
 System Management Server (SMS), 623

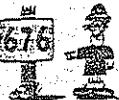
System R, 150
 System Strategies, 17, 126
 SystemView, 313, 585, 624
 Sytek, 119
 Syware, 186

T

T1, 44, 48, 51
 T3, 44
 tablas de montaje de datos, 220
 talones, 407-408, 455
 Tandem Computers, 88, 158, 185, 230, 238, 241, 244, 251, 275, 282, 291, 297, 307-308, 310-312, 317
 CORBA, 317
 Himalaya, 230
 NonStop Kernel, 555
 NonStop SQL, 238, 241, 244
 Parallel CICS, 311
 Pathway, 291, 297, 307-308, 310-312, 317
 RSC, 261, 282, 317
 servidor de comercio, 317
 SOM, 317
 Tuxedo, 312
 Tanenbaum, Andrew, 99
 TAPI, 352
 TCI, 51
 TCP/IP, 34, 37, 84, 106, 120, 194, 357, 466, 595, 598
 TeamRoute 340
 tecnología
 de agentes comunes, 609
 de comunicaciones privadas (PCT), 506
 de transacciones protegidas (STT), 512
 telefonía, 355

telescript, 519
 Telnet, 466
 Tempo, 315
 Teradata, 175, 208-209, 238, 244, 249, 315
 tercero autorizado, 141
 Terisa Systems, 505-506
 Tessler, Larry, 569
 The, Lee, 228
 Thomas, Dave, 387, 433
 Tibbetts, John, 215
 tiempo para pensar, 261
 tipos de datos abstractos (ADT), 155, 167
 Tivoli, 578, 585, 609, 613, 618-619, 624
 Management Environment (TME), 618-619, 624
 Management Framework (TMF), 619
 TLI. Ver *Interfaz de capa de transporte*
 TM/1, 230
 TME. Ver *Tivoli Management Environment*
 TME 10, 613
 TMF. Ver *Tivoli Management Framework*
 Token Ring, 39
 Top End, 260, 298, 307-308, 310-311, 315-316
 colas para transacciones recuperables (RTQ), 315
 TP ligero, 14, 162, 166, 304
 definición, 297-298
 origenes, 296
 TP pesado, 14, 166, 304
 definición, 297-298
 TPC-C, 302
 TRAN, 291
 transacción, 21, 128, 153, 275, 282, 412
 anidadas, 21, 258, 262, 270, 272, 314
 definición, 272

ConTracts, 270
 de carrito de compras, 262, 270
 de larga vida, 262, 270, 328
 de niveles múltiples, 262
 definición, 257
 delimitadores, 281
 encadenadas, 258, 270-271
 definición, 271
 global, 286, 288
 migrantes, 270
 monitores de TP, 279
 revalidación, 262
 voluminosas, 269
 sagas, 262
 Savepoint, 271
 simple, 257-258, 260, 262
 deficiencias, 261
 limitaciones, 266-270
 origenes, 261
 syncpoint, 271
 y componentes, 392
 transacciones, 21
 de períodos largos, 268, 270, 421



- World Wide Web (Web)
(continuación)
terminales, 514
visualizadores, 467, 534,
543, 547-548
y base de datos de SQL,
489
y groupware, 489
y MOM, 489
y monitores de TP, 489
y objetos, 515-532
World Wide Web Consortium
(W3C), 447, 489-490,
512, 527, 533, 534-
536, 549, 552, 554
WOSA, 348
- X
- X.25, 45, 357
X.400, 345-346, 356, 365
definición, 345-346
XAPIA, 349
X.400 API Association
(XAPIA), 349
X.435, 346
X.500, 89, 135, 346, 357
de ISO, 412
servicio de directorio, 106
X.509, 346
- X/Open, 17, 93, 106, 150,
155, 187, 189, 198,
215, 261, 594
administración de API,
611, 613
CLI, 179, 183-184, 186,
189, 198, 248
CPI-C, 287
DTP, 284-285
modelo de referencia de
transacción, 285
modelo de transacción,
286
SysMan, 618
TX, interfaz, 286
TxRPC, 287
XA+, 286
XATMI, 287
XMP, 611, 613
XOM, 611, 613
XTP, grupo de, 284
XA, 265
definición, 285-286
interfaz, 285
XA de X/Open, 156
XA+, definición, 286
XAPIA, 350
Ver X.400 API Association
XATMI, 17
definición, 287
XBase, 236
- XDB, 152, 158, 174, 195
XDR. Ver *Representación de
datos externos*
XDS. Ver *API del servicio de
directorio de X/Open*
XES Inc., 593
XMP. Ver *API de administra-
ción de X/Open*
XNS, 119
XOM. Ver *Administrador de
objetos de X/Open*
XPE, 623
XSoft, 340
- Y
- Yahoo!, 499, 559
Yankee Group, 41
- Z
- Zenith, 51
Ziff-Davis, 489
Zimmermann, Phil, 512
ZipLock, 510
Zisman, Mike, 543