



Conozca en Profundidad

¿Porque Servidores de Bases de Datos?

Sistema de Capacitación y material instructivo del Software Esbiza

TITULO DEL DOCUMENTO: ¿PORQUÉ SERVIDORES DE BASES DE DATOS?
 AUTOR: CLAUDIO A. NIPOTTI
 FECHA ORIGINAL: 17-OCT-2006
 NÚMERO Y FECHA DE ESTA REVISIÓN: 3, 18-NOV-2007

Como se dice siempre, "Base de datos es cualquier conjunto organizado de datos" y el ejemplo clásico que se brinda es el fichero de tarjetas de cartulina.

Este documento en realidad se enfoca en los "**Servidores de bases de datos**", una clase de programas que realmente mueve al mundo, al ser el centro de todos los grandes sistemas empresariales y de gobierno, por red o por internet. Eso lo vamos a tratar a lo largo de su contenido, que está orientado principalmente a servir como introducción a personas sin conocimientos sobre el tema.

¿Beneficios?

Las siguientes son algunas características del uso de servidores de bases de datos:

a) **Globalización de la información:** La información está por igual al alcance de todos los miembros de la organización que tengan acceso a la base de datos.

b) **Protección de datos sensibles:** Se les llama datos sensibles a aquella información que tiene un alto grado de importancia personal o financiera, y que por lo tanto requiere privacidad. En todo el mundo hay leyes que regulan la tenencia y uso de este tipo de datos. En el comercio electrónico abundan, desde el número de tarjeta de crédito de una persona hasta el historial de sus compras o los domicilios a donde ordenó el envío.

Es importante protegerlos no solamente de su robo o uso ilegítimo sino también de la pérdida por fallas de hardware o software, pérdida que puede ser irreparable y causar trastornos inimaginables en la actividad de empresas y particulares. Un servidor de bases de datos requiere un nombre de usuario y contraseña al conectarse a él.

c) **Eliminación de redundancia:** Las técnicas de normalización aplicadas eliminan toda redundancia.

d) **Eliminación de inconsistencias:** Por ejemplo, que cada registro de historia médica pertenezca a un paciente existente y no a uno que se borró o que nunca existió. Los datos ingresados pueden ser controlados por una aplicación, pero normalmente la misma base de datos puede contener y exigir que se cumplan reglas preprogramadas que aseguren la consistencia de los datos, aún cuando se use otra aplicación que no lo controle (ver punto e).

e) **Independencia entre datos y tratamiento:** Todavía hoy, muchos programadores o estudiantes de programación solamente conciben los datos como patrimonio de un determinado programa, que es el único que los accede. Un servidor de bases de datos es un repositorio podríamos decir universal: cualquier aplicación que se conecte al servidor (y para eso están los drivers, como ODBC, JDBC, ADO, etc.) puede acceder a los datos. Inclusive hojas de cálculo, por citar una aplicación algo alejada del programa tradicional de gestión.

f) **Multiusuario:** Deberíamos hacer un documento muy largo para ilustrar los distintos conflictos que pueden ocurrir cuando muchos usuarios acceden al mismo tiempo, y las técnicas que los servidores de bases de datos usan para manejarlos. Imagínese solamente un usuario tratando de borrar un registro que otro está viendo como parte de un informe. Aquí no hay soluciones mágicas, el registro se va a borrar finalmente y el informe ya no va a ser tan actual, pero se trata de garantizar que al menos ese informe se termine de elaborar con los datos que había al iniciarse y evitar que se produzca un error de software o de consistencia.

g) **Abstracción:** Un motor de bases de datos usa (internamente) recursos de manejo de archivos y técnicas de programación complejísimo para cumplir sus objetivos. Por no mencionar que los datos de una misma persona (por poner un caso) pueden estar físicamente en distintos archivos o discos o servidores.

La abstracción significa que el operador, usuario o aplicación que solicita los datos los recibe sin tener que conocer todos los mecanismos que se pusieron en marcha para traerle la información, ni siquiera conoce su ubicación física, que puede cambiar.

¿Cómo eran los sistemas antes de los Servidores de Bases de Datos?

Probablemente el lector esté familiarizado con alguno de los métodos de almacenamiento que se detallan a continuación. Están ordenados desde el más primitivo hasta el más avanzado. No obstante, ninguno otorga los beneficios de un Servidor de Bases de Datos.

a) **Archivos Planos:** Los datos se grababan en archivos de texto que podrían leerse inclusive con el bloc de notas de Windows, sin ninguna seguridad ni posibilidades avanzadas de búsqueda, como no fuera leerlos uno a uno (secuencialmente)

b) **Archivos Binarios:** Por ejemplo, el caso de Turbo Pascal. Los archivos eran un poco más avanzados en su estructura, pero las funciones eran muy básicas y aunque siempre hubo bibliotecas de funciones de terceros, normalmente todo lo que fuera búsqueda y ordenación se tenía que programar "a mano".

c) **Archivos DBF:** A partir del producto **dBASE (I, II, III y IV)**, aparecieron otros como **Fox Base**, **Clipper** y **Fox Pro**. Eran lenguajes muy especializados en el manejo de datos, podían manejar un par de millones de registros, aunque llegados a este punto la lentitud era notoria, pero sus archivos, comparados con los motores de bases de datos, carecían de seguridad tanto frente al acceso indebido como a la corrupción. Era muy frecuente que se dañaran. **Visual Fox Pro** es un caso aparte, sigue vigente y puede utilizar motores de bases de datos, prescindiendo de las DBF.

Hay un lenguaje de consulta que se verá luego que hasta las tablas DBF no existía, inclusive en estas tablas, que tenían muchas ventajas sobre las técnicas anteriores, las búsquedas eran realmente artesanales. En Visual Fox Pro apareció por primera vez un dialecto bueno de este lenguaje, pero menos potente que en un motor de bases de datos.

¿Cómo son los sistemas que utilizan Servidores de Bases de Datos?

Hay 3 cualidades que permiten explicar porqué un Servidor de Bases de Datos supera a otros sistemas de almacenamiento:

a) **Propiedades ACID:** Las propiedades ACID, que se explican luego, son el punto fuerte de toda base de datos transaccional. Las propiedades ACID resuelven muchos conflictos, especialmente debidos a

dos factores: la concurrencia, es decir, dos o mas usuarios tratando de modificar la misma información, y la interrupción de procesos de grabación debido a errores o fallas varias.

b) **Lenguaje SQL**: Un lenguaje poderoso como SQL, que permite obtener los resultados deseados emitiendo ordenes que son bastante simples una vez que se aprende parte del lenguaje.

c) **Tolerancia a fallas**: Una parte muy grande de toda la ingeniería de software aplicada al desarrollo del servidor se orienta a hacerlo resistente a fallas. Ya mencionamos las propiedades ACID, además tenemos replicación y agrupación (clustering), o sea, la copia continua en distintos servidores (todos tienen el mismo contenido, pueden repartirse la carga de trabajo) y el funcionamiento de varios servidores como uno solo respectivamente (donde también se administra mejor la carga de trabajo, o sea la cantidad de solicitudes recibidas desde los programas cliente).

Propiedades ACID

Vemos que ACID hace referencia todo el tiempo a transacciones. Una transacción no es, como podría suponerse, una transacción comercial o un intercambio. En terminología de bases de datos significa "un conjunto de operaciones que deben realizarse o cancelarse todas juntas". Por ejemplo, si registramos que Pedro compró un televisor en cuotas, también tenemos que registrar las cuotas que va a tener que pagar Pedro con sus fechas e importes. Si por una falla de hardware o software se grabara una u otra cosa, no tendría sentido: comprar sin un plan de pago o un plan de pago sin nada comprado no sirven para nada. Por eso, al estar dentro de una transacción, se garantiza que se grabarán ambos o el error que ocurra dejará todo sin grabar.

ACID significa:

Atomicidad: las operaciones se tratan en forma atómica agrupadas en transacciones.

Consistencia: mediante reglas de integridad que mantienen coherencia entre datos.

Aislamiento (Isolation): se manejan posibles conflictos entre varias transacciones simultáneas impidiendo que unas interfieran con otras y todas se ejecuten y finalicen en el orden en que se iniciaron.

Durabilidad: las transacciones confirmadas son permanentes.

Tolerancia a fallas

Para comparar, se ha revisado un programa en Clipper que todavía se usa en muchas oficinas. Clipper era tecnología de punta hace 15 años... hoy no debería pasar esto.

El sistema en Clipper revisado confiesa en su documentación ser absolutamente vulnerable a:

- Interrupciones anormales del sistema (errores que terminan su ejecución)
- Apagar el ordenador con el sistema en uso
- Sufrir un corte de energía con el sistema en uso
- Variaciones de tensión en suministro eléctrico
- Incorrecta puesta a tierra o aislación de la red
- Encendido de motores conectados a la misma red eléctrica
- El ordenador donde se usa el sistema se desconecta de la red (!!!)

Como vemos, la mayoría de los problemas se deben a que los datos no son independientes de la aplicación, y, para peor, son directamente accesibles por el programa. Esto se entenderá mejor cuando expliquemos el funcionamiento de un esquema Cliente / Servidor.

El sistema tomado como ejemplo parece haber sido diseñado por expertos pues es capaz de informar las causas de las fallas... y culpando siempre al usuario. El tiempo invertido en informarlas podría haberse aprovechado mejor aprendiendo a utilizar la tecnología que corresponde.

El único factor al que no escapa ninguna información digital es un mal funcionamiento del disco.

En la segunda pantalla vemos las complicaciones que acarrear los efectos de eventos que para un servidor de bases de datos son prácticamente inocuos. Incluye realizar un backup en *diskettes* y enviarlo al proveedor.

Lo notable es que lo que se acaba de describir no sucede únicamente con sistemas antiguos: de hecho, muchos sistemas recién liberados utilizan internamente esta forma tan vulnerable de manejar los datos. Desde luego, no es el caso del **Software Esbiza**.

El modelo Cliente / Servidor

Se ha dado una idea aproximada de algunas funciones de un Servidor de Bases de Datos (seguridad, controlar la concurrencia de operaciones sobre un mismo dato, localizar los datos solicitados sin importar en qué disco o ubicación se encuentran) pero no de su funcionamiento.

En este modelo, los datos están almacenados en uno o más discos, y solamente son accesibles por el servidor.

Los puestos de trabajo ejecutan distintos tipos de aplicaciones que reciben el nombre genérico de *clientes* en contraste con el *servidor*.

Generalmente la aplicación cliente es un sistema especialmente diseñado para el usuario, o al menos especialmente diseñado para cumplir una función que coincide con la necesidad del usuario. Por ejemplo los programas para gestionar distintas actividades: administrativa, laboratorios, plantas de fabricación, etc.

Pero cliente es también cualquier aplicación –como SQLyog, MySQLfront, el cliente de modo texto **mysql**, etc.- capaz de enviar sentencias SQL al servidor y recibir resultados. Estos últimos también se utilizan para crear y modificar tablas.

El servidor recibe e interpreta estas sentencias, y es el *único* que accede a los datos propiamente dichos para ejecutar estas sentencias, y en caso de que haya que devolver resultados, los devuelve a la aplicación cliente, que recibe la información pedida pero nunca tuvo acceso a los archivos verdaderos. Por eso no importa que los ordenadores que ejecutan las aplicaciones clientes se congelen, colapsen, se apaguen, o pierdan la conexión de red. Las que sufrirán consecuencias serán las sentencias que estaban enviando, por ejemplo no llegarán a destino, pero esto simplemente impedirá la comunicación con el servidor, sin causar que éste acceda descontroladamente a los datos y los arruine. Es obvio que la maravillosa arquitectura de un software servidor determina que las solicitudes irreconocibles no se ejecuten.

Conclusión

Definitivamente, debe elegirse Cliente / Servidor. Mientras los desarrolladores de sistemas de gestión en lenguajes antiguos siguen tratando de descubrir cómo evitar que un usuario borre un registro que otro está modificando, los desarrolladores de sistemas cliente / servidor simplemente disfrutan el uso de un Servidor de Bases de Datos, que siempre sorprende con una nueva forma de solucionar problemas.

Nota Legal: Todas las marcas mencionadas son propiedad de sus respectivos dueños. Turbo Pascal y dBase IV son propiedad de Borland International Inc; Fox Base, Fox Pro, Visual Fox Pro y SQL Server son propiedad de Microsoft Corporation; SQLyog es propiedad de Web Yog, Oracle es propiedad de Oracle Corporation, MySQL es propiedad de MySQL AB; IBM/DB2 es propiedad de International Business Machines INC.; Clipper es propiedad de Computer Associates.

Recuerde que como suscriptor de Esbiza, puede solicitar mayor información a esbiza@gmail.com o por teléfono. Para ver más artículos como este visite www.esbiza.com.ar