¿Versión Access o versión SQL Server/Oracle?

Para definir qué sistema de base de datos requerirá, es conveniente tener en claro conceptos básicos acerca de las potencialidades de cada una de ellas, que facilite la búsqueda de datos y no requiera de inversiones extra. Aquí les ofrecemos una serie de conceptos que seguro serán de utilidad a la hora de decidirse por una u otra.

Lic. David Walfisch
Gerente General de Intelektron

Es usual que al ofrecer una solución a un Cliente o al tener que comprar para nuestro uso un sistema, nos encontremos con la necesidad de definir entre una versión Access o una versión SQL Server u Oracle. A priori, posiblemente exista una diferencia en el precio del producto y en los requerimientos (por ejemplo contar ya con un Servidor con SQL instalado y una cantidad de licencias de uso). Pero si no entendemos claramente cuáles son las diferencias técnicas reales, resulta difícil comparar el costo respecto del beneficio.

En esta nota quisiera explicarle al lector algunos conceptos que, si bien tienen una base técnica de Sistemas, espero sean fácilmente comprensibles y aplicables a la hora de decidir.

¿Cómo definir adecuadamente una "Base de Datos"?

Cada vez que pregunto a alguien si un archivo con extensión ".MDB" es una base de datos, invariablemente recibo una respuesta afirmativa. Y si bien la pregunta tiene una "trampita", resulta un buen disparador, ya que en realidad NO es una base de datos, sino sólo una parte: el archivo de base de datos y, en este caso, de Microsoft Access.

Una base de datos se compone de dos partes principales: el Archivo (uno o varios) donde se guardan los datos y el Motor de la base, que son las funciones y los procedimientos que permiten acceder a los datos para almacenarnos, leerlos, filtrarlos, modificarlos, etc.

Entonces, cuando hablamos de un archivo con extensión ".MDB" o ".DBF", se trata de una parte de una base de datos (Microsoft Access o dBASE respectivamente en estos ejemplos), que para ser utilizados requieren de un motor que la soporte.

Este motor podría ser la aplicación de Access que provee Microsoft en algunas versiones de su paquete de oficina (Microsoft Office), pero habitualmente, cuando adquirimos un software, no se requiere tener instalado el Access, sino que la misma aplicación contiene una serie de bibliotecas y recursos (archivos DLLs), que contienen al motor para acceder a los datos.

Estos "Drivers" hoy se denominan "Providers" y hay uno distinto para cada tipo de base de datos (o archivo de base de datos) a utilizar.

Un lenguaje habitual para realizar consultas (de lectura, modificación o eliminación de los datos o registros) es SQL, "Structured Query Language" o "Lenguaje de Consultas Estructurado". De allí deriva el nombre del producto de Microsoft, el Microsoft SQL Server.

Tanto este producto, como el ORACLE, el INFORMIX, y por ejemplo una versión de software libre usualmente utilizada como la MySQL, sí son "Bases de Datos" completas, ya que proveen tanto Motor como Archivo. Más adelante, explicaré sus principales ventajas. Ahora bien: cuando tengo un software que utiliza un archivo ".MDB", donde se ejecuta la aplicación, corre además el provider, que es quien aporta (junto con otros elementos) al motor que procesa y administra los datos. El archivo puede estar en la misma PC o alojado en algún disco compartido de la red por cuestiones de espacio, seguridad o tal vez porque múltiples aplicaciones necesiten acceder al mismo archivo.

Ahí es cuando surge habitualmente la confusión de pensar que porque el archivo está en un Servidor (de archivos en este caso) la aplicación es "Cliente-Servidor" (o Client-Server en inglés) y esto es un error.

Para poder explicar claramente la diferencia, vamos a tener que avanzar un poco más en el marco teórico y explicar el concepto de Multi-Capas.

¿Cómo es la arquitectura de Software Multi-Capas?

En la mayoría de las aplicaciones, y de manera muy evidente en las comerciales, podemos identificar al menos tres capas (o divisiones), que componen al sistema: la capa que el usuario utiliza para interactuar (donde puede recibir información o ingresar datos o ejecutar instrucciones a través de una interfase), la capa donde se almacenan y administran los datos y la capa que se encarga de los procesos y de ejecutar las acciones.

La capa de "Interfase" (o interfases en caso que se presente más de una), es la capa más evidente y percibida por los usuarios. Por ejemplo, en un programa de Windows, suelen ser las ventanas con sus botones, textos, imágenes, casillas para el ingreso de datos, combos desplegables, menús, etc. En una interfase tipo Web se presentan elementos similares pero contenidos en una página que se presenta desde un explorador, como el Internet Explorer, Chrome, FireFox o Safari.

La capa de "Lógica de Negocios" contiene los procesos y las acciones. Es un tanto más difícil de conceptualizar por los usuarios, porque lo que ellos ven es simplemente que tocan un botón y el programa ejecuta

Continúa en página 120

una acción. El botón, pertenece a la "Interfase" pero lo que hace, si el sistema está diseñado siguiendo la estructura de tres capas (o eventualmente más de tres), debería programarse por separado y que el botón únicamente dispare la ejecución de ese código contenido en una biblioteca.

Es posible que se esté preguntando cuál es la ventaja… Y en realidad, si bien el esfuerzo de diseño y programación es superior, presenta un aspecto positivo fundamental: se puede generar código reutilizable, de tal manera que hoy tenemos un sistema que nos brinda una funcionalidad y que lo accedemos desde Windows y que, si posteriormente es necesario, hacer una Interfase Web o Linux, es posible encarar los esfuerzos en crear una versión nueva orientada sólo a la forma de presentar las pantallas para la interacción con el usuario.

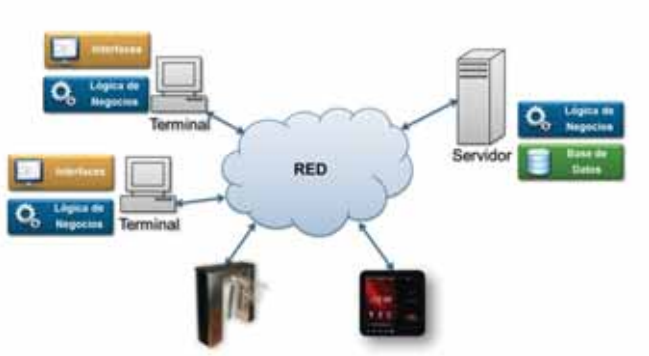
Por último, ni la capa de "Interfase" ni la de "Lógica de Negocios" debería encargarse de administrar datos (leer, agregar, eliminar o modificar) sino que debería solicitar a otro componente del sistema, la capa de "Base de Datos", que se encargue de esas tareas. Y al igual que con el caso de las múltiples interfases, gracias a tener esta estructura, será mucho más fácil y controlable realizar un nuevo desarrollo que permita ahora al software interactuar con un nuevo tipo de base de datos o vincularse a través de la integración con bases existentes.

¿Sistemas Stand-Alone o Sistemas Client-Server?

Una vez comprendidos los conceptos anteriores, sólo tenemos que aplicarlos a diferentes casos para interpretar sus características, algunas ventajas y cómo es habitual, también desventajas.

En un sistema Stand-Alone, las tres capas (salvo una pequeña excepción que analizaremos) se encuentran en una misma PC.

Es la clásica aplicación donde instalamos el software y ubicamos el archivo de base de datos en la PC o a lo sumo en un disco compartido en la Red.



Cuando una terminal solicita un informe con algún tipo de filtro, la petición que llega al motor de la base en el Servidor, que filtra los datos y devuelve solo la información solicitada.

Entonces tenemos la capa de "Interfase" presentando en pantalla, la capa de "Lógica de Negocios" procesando en la misma PC (para lo cual según la carga necesitaremos una computadora con determinadas características para que la performance sea adecuada) y de la capa de "Base de Datos", el motor se ejecuta en su totalidad en la misma computadora. Aquí viene una clave para comprender el porqué de este artículo: si el archivo de base de datos está en un disco propio, cuando pido, por ejemplo, un informe filtrado, este motor local tomará del archivo el total de los datos, los filtrará según los criterios aplicables y luego generará el resultado

Si, en cambio, el archivo está en la RED, deberán circular por la RED todos los datos hasta la PC, para recién allí ser filtrados por el motor, dando como resultado procesos mucho más lentos y consumos más altos de ancho de banda durante estos procesos.

Si además cuento con un sistema multiusuario (varias terminales accediendo a los datos e inclusive potencialmente de manera simultánea), en cada aplicación de las terminales se repite una situación similar y además cada una deberá controlar con ciertas reglas no estar trabajando con datos que otra esté accediendo para modificar, ya que esto podría generar inconsistencias.

En estos casos, al estar la "Lógica de Negocios" junto con el motor completamente en cada terminal, podemos hablar de una "aplicación distribuida". Ventajas: el proceso se reparte entre cada terminal, haciendo menos necesario contar con un Servidor fuerte. Desventajas: cada terminal debe por ende tener cierta potencia, y la actualización de las versiones es más demandante, ya que si contáramos con terminales con versiones distintas, probablemente terminaríamos teniendo alguna incompatibilidad o generaríamos resultados inconsistentes.

En cambio, en una aplicación Cliente-Servidor, la capa de "Interfase" está en su mayor parte en cada terminal, la "Lógica de Negocios" puede ubicarse sólo en un servidor, sólo en las terminales, o de una manera combinada (diferentes niveles de centralización-descentralización), pero lo que siempre queda del lado en un Servidor es la capa de "Base de Datos" y esto presenta múltiples ventajas. Tomando el ejemplo anterior, cuando una terminal solicita un informe con algún tipo de filtro, sólo circula por la red una petición que llega al motor de la base en el Servidor; éste se encarga de filtrar los datos y por último devuelve a la aplicación únicamente la información solicitada, generando un resultado con mayor velocidad y con una carga mucho menor a la RED.

Además, este tipo de Bases de Datos "completas" (motor+archivo, todo en una única aplicación servidor), suelen tener opciones de optimización, seguridad para el acceso a los datos, posibilidades de backup, controles mejorados para administrar usuarios simultáneos y herramientas de análisis muy superiores a otras opciones.

Como contrapartida, requieren no sólo disponer del producto de software, en general costoso y con diferentes tipos de licencia por usuarios, sino que además suele ser necesario contar con un Servidor (hardware) potente, un sistema operativo adecuado y, principalmente, con administradores con conocimientos que le permitan tanto la instalación como el mantenimiento del sistema, sin dudas, con un incremento en los costos.

Ahora, al menos, tendremos una mejor idea de por qué nos convendrá técnicamente optar por una u otra opción.

Más datos: **david@intelektron.com / www.intelektron.com / (54 11) 4305-5600**