Bases de Datos

1º DAM/DAW/ASIR

Fundamentos de Bases de Datos

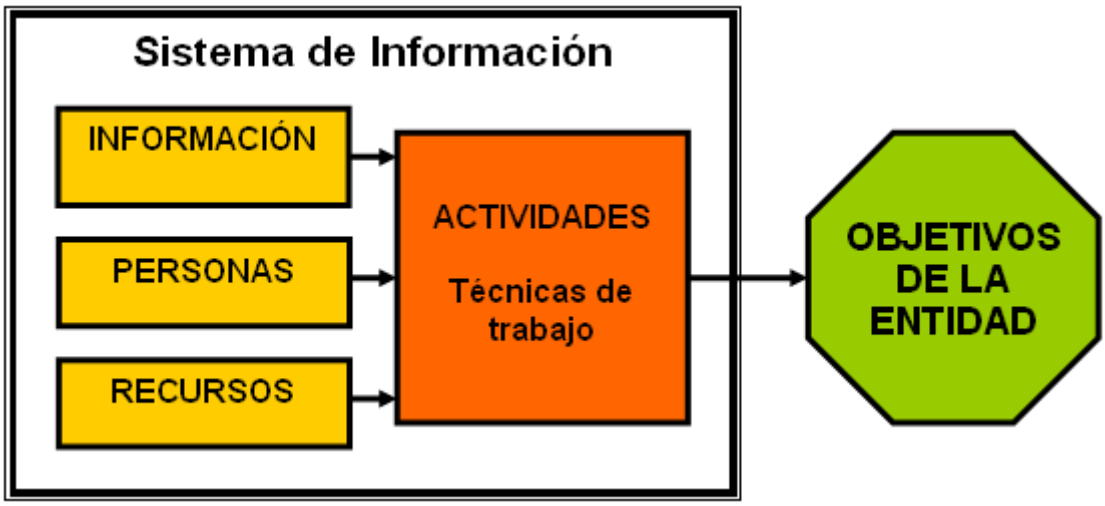
Sistema de Información

Un Sistema de Información es un conjunto de elementos destinados al tratamiento y administración de datos e información, organizados y listos para su posterior uso, generados para cubrir una necesidad (objetivo).

Esta definición de *Sistema de Información* abarca cualquier tipo de sistema, tengan o no componentes informáticos o se apoyen en ellos. En este curso sólo nos interesarán los Sistemas de Información de carácter informático, más concretamente los que conocemos como *Sistemas Gestores de Bases de Datos* (a partir de ahora *SGBD*), que definiremos más adelante.

Hay que tener en cuenta que, dentro del campo de la informática, existen muchos tipos de Sistemas de Información que no son *SGBD*. Por ejemplo, cualquier aplicación informática que permite el tratamiento de cantidades considerables de información para cualquier cometido puede ser considerado un Sistema de Información. En cualquier aso, es muy probable que cualquiera de estas aplicaciones (o *Sistemas de Información*) utilicen un *SGBD* como soporte lógico de almacenamiento de toda esa información.

Parte del presente curso consiste en el desarrollo de aplicaciones que conectan con Bases de Datos para gestionar datos, por lo que de cierta manera también vamos a crear nuestros propios Sistemas de Información, apoyándonos en herramientas de desarrollo para crearlos y en algún SGBD para dar soporte a nuestros datos.



¿Es necesario siempre usar un SGBD?

En nuestro campo, un fichero se podría considerar como la unidad mínima de almacenamiento de cualquier tipo de información. No es más que un conjunto de registros almacenados con una estructura homogénea, que permite ser accedido para su tratamiento desde alguna aplicación informática. Existen muchos tipos de ficheros atendiendo al *formato* en el que se escriben: texto plano, imagen, PDF, . . . y para trabajar con cada uno de esos formatos existen una o varias aplicaciones. Y por lo general, un tipo de aplicación sólo abrirá un tipo de fichero, el tipo de fichero para el que ha sido diseñada: El Bloc de Notas abre ficheros de texto plano, Adobe Reader abre ficheros PDF, Picasa abre ficheros de imagen, . . . Aunque también podemos encontrarnos con aplicaciones que son capaces de trabajar con diferentes formatos, similares entre sí (Microsoft Word, por ejemplo).

Una Base de Datos es un conjunto de ficheros interrelacionados entre sí, creados por un SGBD, que deben ser gestionados y accedidos a través de dicha herramienta. Por lo general se tratara de una gran cantidad de información repartida entre varios ficheros, que es accedida desde una herramienta específica de forma que el usuario se abstrae de cómo está organizada dicha información. No nos interesa conocer qué ficheros forman la Base de Datos, dónde están ni cuanto ocupan; la herramienta que utilizamos se ocupa de todo ese trabajo. Nosotros, como usuarios de una Base de Datos, tendremos a nuestra disposición todos

los datos y numerosas herramientas que nos permitan trabajar con ellos de una forma cómoda (utilidades para buscar, para ordenar, para eliminar, . . .)

Hay que tener en cuenta que, aunque las Bases de Datos sean la solución a los problemas de almacenamiento y consulta de grandes cantidades de información, no siempre serán la solución que necesitemos. Si imaginamos una pequeña aplicación que apenas necesita almacenar unas pocas filas de datos, no será muy recomendable depender de una segunda aplicación (normalmente costosa en tamaño y rendimiento y, probablemente, también en dinero) para gestionar esos pocos datos. Aunque existen soluciones pequeñas para este tipo de aplicaciones, normalmente añadirán una carga extra de trabajo mayor que el beneficio que produce usarlas cuando la ocasión lo requiere. Si bien es cierto que existen pequeños SGBD que pueden ser una opción muy interesante en este tipo de casos (*SQLite*, por ejemplo, tiene soporte en dispositivos móviles a través de *Android*).

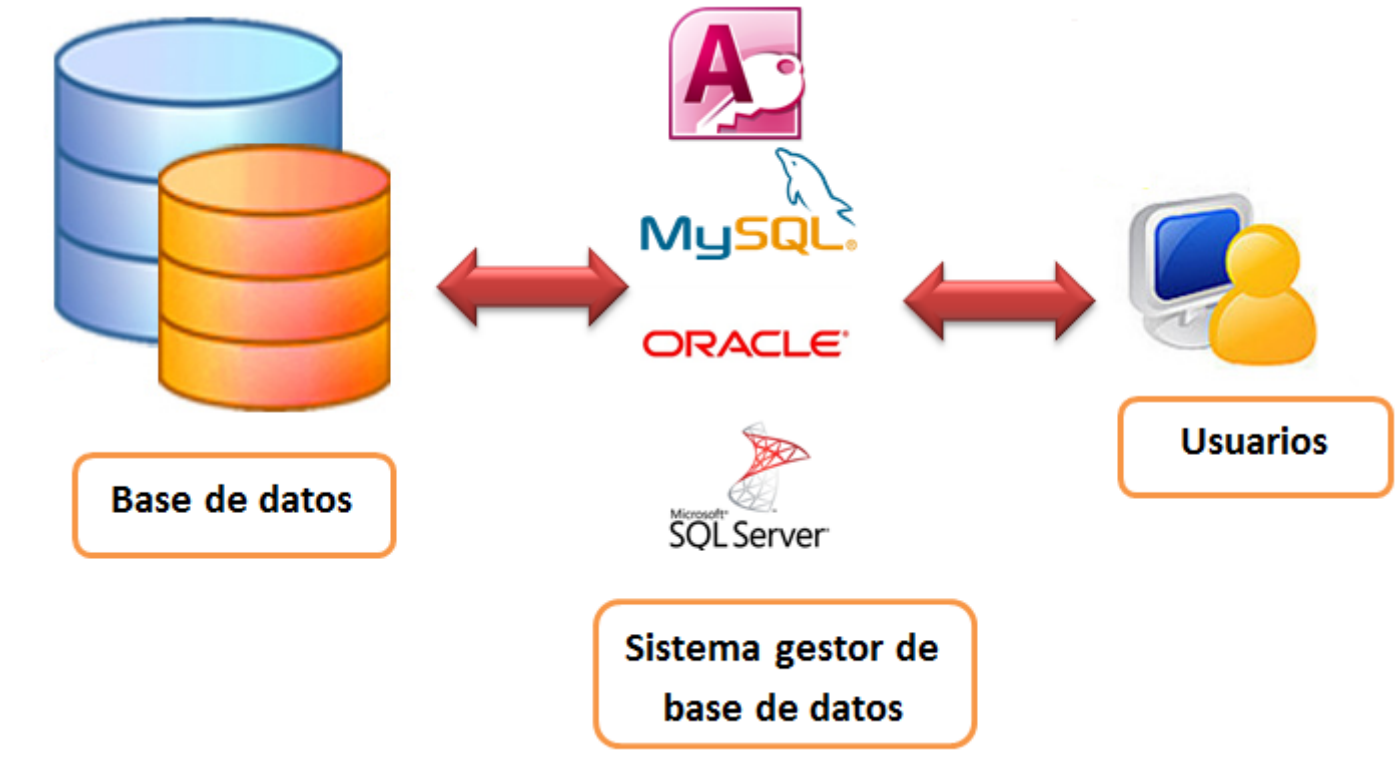
Otra ventaja que presenta el uso de Bases de Datos frente al empleo de ficheros es el ahorro que esto supone. Hay que pensar que en el caso de que no contemos con un SGBD para el desarrollo de una aplicación determinada, deberemos desarrollar a medida toda la parte que se refiera al almacenamiento y la gestión de todos los datos de dicha aplicación. En un SGBD toda esa parte ya está implementada, tiene un interfaz de acceso común y adaptable a, prácticamente, cualquier entorno. Además, existen lenguajes ampliamente conocidos para el acceso y gestión de los datos almacenados (*SQL* en el caso de los SGBD relacionales, por ejemplo).

Por otro lado hay que tener en cuenta, según veremos más adelante, que en la actualidad existen muchos tipos de SGBD, desde lo que se conoce como el clásico SGBD relacional hasta pequeños SGBD que apenas ocupan 1MB de espacio y no requieren instalación, otros almacenan la información como ficheros XML, . . . por lo que la distancia que separa el acceso directo a ficheros y los SGBD cada vez es menor y prácticamente todo puede considerarse una Base de Datos, que junto con su aplicación correspondiente forman numerosos y diferentes Sistemas de Información.

Sistema gestor de Bases de Datos

Un SGBD es una herramienta software, más o menos compleja, que permite la creación y gestión de una Base de Dato. En el punto anterior hemos visto que una Base de Datos estaba compuesta de varios ficheros relacionados entre sí. El SGBD es la herramienta que se encarga de organizar esos ficheros manteniendo la información siempre accesible para el usuario de la forma más eficiente posible, tanto en espacio como en velocidad de acceso.

Cuando gestionamos una Base de Datos a través de un SGBD, estamos añadiendo un nivel de abstracción puesto que "no nos enteramos" exactamente de lo que ocurre con los datos, cómo se almacenan, dónde se almacenan, cuánto ocupan, . . . puesto que es esta herramienta la que se encarga de que la Base de Datos sea consistente, esté siempre accesible, disponga de espacio en disco, y de otras muchas tareas que, dependiendo del SGBD concreto, pueden ser más o menos complejas.



Hay que tener en cuenta que actualmente existen diferentes tipos de SGBD en cuanto al nivel de conocimientos técnicos que se requieren para trabajar con ellos. Por citar un ejemplo, un SGBD como *MySQL* nunca podrá ser utilizado por usuarios con escasos conocimientos técnicos puesto que no está pensado como aplicación para el usuario final, sino como SGBD de apoyo para otras aplicaciones, aplicaciones web o sitios web que requieran manejar grandes cantidades de datos (Sistemas de Información). Por otro lado, *Microsoft Access* o *LibreOffice Base* son SGBD pensados para un usuario menos técnico. No se necesitan tantos conocimientos técnicos para su manejo, y por tanto se acercan más al usuario. De hecho se incluyen en las *suites ofimáticas* más conocidas. Por tanto, es lógico pensar que entonces no estarán pensados para servir de apoyo a otras aplicaciones, sino para funcionar de forma independiente como aplicación final de usuario.

Características deseables

En general, de cualquier Sistema de Información, se esperan una serie de características, que se exponen a continuación:

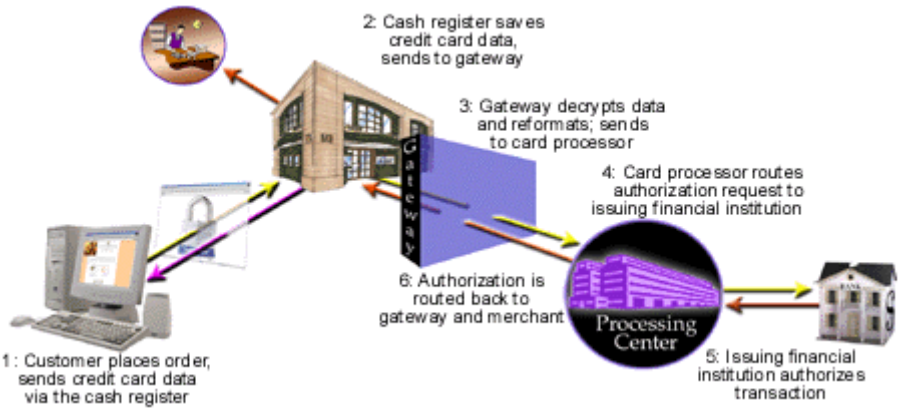
- **CRUD**: El acrónimo CRUD hace referencia a las 4 operaciones básicas que cualquier SGBD debería proporcionar: Create (crear), Update (modificar), Read (leer), Delete (eliminar). Además, hay que tenerlas muy en cuenta a la hora de diseñar correctamente nuestra Base de Datos, puesto que la aplicación que la gestione deberá facilitar estas operaciones de una forma correcta.
- **Recuperación de la información**: En cierta manera hace referencia a la *R* de CRUD pero también es importante la forma en que esta información se puede leer, de forma que se espera que cualquier fragmento de información pueda ser recuperado, y además de diferentes formas. Por ejemplo, deberíamos ser capaces de buscar a un cliente determinado en nuestras Base de Datos haciendo referencia a más de un criterio (DNI, apellidos, ciudad donde vive, . . .)
- **Consistencia**: De todo SGBD se espera que los resultados de una lectura o búsqueda sean consistentes de forma que si realizamos la misma búsqueda dos veces obtengamos el mismo resultado (siempre y cuando los datos no hayan sido modificados). Si otro usuario realiza la misma búsqueda deberá también recibir la misma información. Es decir, un mismo dato no puede tener más de un resultado posible.A este respecto hay que tener en cuenta que un mal diseño en una Base de Datos puede no devolver el mismo resultado para una misma búsqueda. Supongamos que la información de contacto de un cliente difiere de la información de contacto de ese cliente en alguno de los pedidos que ha realizado. ¿Qué información debo usar para contactar con él?
- **Validez**: El SGBD debe proporcionar mecanismos que permitan validar la información que se registra en la Base de Datos. Por ejemplo, si debemos indicar la fecha de un pedido en el campo de un formulario, es de esperar que el SGBD sea capaz de verificar si el formato del dato introducido es del tipo correcto.
- **Velocidad**: Quizás es una de las características más importantes. La información en un SGBD se tiene que poder crear, modificar, leer y eliminar rápidamente. En este aspecto influye en gran medida el diseño que se haya realizado de la Base de Datos. Un diseño pobre puede incidir muy negativamente en la velocidad de acceso a los datos por parte del SGBD.
- **Transacciones atómicas**: Una transacción atómica consiste en realizar una serie de acciones como si se tratara de una sola operación, y sin interferir de ninguna manera en la correcta ejecución de dichas acciones. Es muy importante tener en cuenta que dichas transacciones deben ejecutarse completamente, o bien no ejecutar ninguna de sus acciones, nunca deben ejecutarse a medias. Más adelante, en este tema, se detallarán como deben funcionar estas transacciones atómicas.
- **Persistencia y soporte para Backups**: Toda la información almacenada por el SGBD debe ser persistente. Es decir, no debe desaparecer. Una vez se haya realizado una operación, la información no puede cambiar. De otra manera no podríamos "confiar" en nuestra Base de Datos. Si bien en condiciones normales es relativamente fácil asegurar la persistencia de la información, hay que tener en cuenta que existen diferentes situaciones que pueden poner en peligro los datos. Supongamos que estamos dando de alta a una serie de clientes y en el mismo momento en que terminamos se va la luz. El SGBD tendrá que ser capaz de almacenar correctamente los datos de todas aquellas operaciones que han sido confirmadas (y no hacerlo en aquellas que no lo hayan sido). Por otro lado, ante situaciones imprevistas como borrados accidentales, fallos en los discos duros, . . . se necesitarán herramientas para la creación y restauración de *Backups*. En definitiva, mecanismos para recuperar la información en caso de fallos de aquellos elementos ajenos a nuestro SGBD pero de los que se depende (disco duro, acciones accidentales de los usuarios, . . .)
- **Capacidad de extender sus características**: Puede resultar muy interesante la capacidad de extender las características del SGBD de manera que a medida que pasa el tiempo podamos incorporar nuevas características a la Base de Datos que allí tengamos almacenada. En este caso también habrá que tener en cuenta el diseño de la Base de Datos ya que solamente un buen diseño permitirá extender sus capacidades sin comprometer el rendimiento.
- **Seguridad**: Todo Sistema de Información debe proporcionar ciertos mecanismos de seguridad que protejan la información que allí se almacena. Éstos mecanismos pueden variar y siempre serán mayores cuanto más sensible sea la información almacenada, pero al menos se deberá contar con una correcta gestión de cuentas de usuario y privilegios en cuanto a los accesos que los diferentes usuarios realizan al SGBD para trabajar con la Base de Datos. Por supuesto, el buen uso de esa gestión de privilegios será el que proporcione la seguridad deseada a nuestros datos.

Funciones

Por eso, podríamos decir que las funciones de todo SGBD son las siguientes:

- **Crear y organizar la Base de Datos** Esta herramienta se encarga de la creación y gestión de la Base de Datos, abstrayendo a sus usuarios de dónde y cómo se organiza internamente. El administrador de la Base de Datos dispone de las herramientas necesarias para crear y organizar las Base de Datos sin la necesidad de conocer la implementación de todo lo demás.
- **Control de acceso** No todos los SGBD lo incorporan, pero la mayoría dan soporte a un control en el acceso a la información mediante el empleo de un sistema de cuentas de usuario y privilegios por el que el administrador de la Base de Datos podrá decidir qué usuarios puede realizar determinadas tareas sobre objetos determinados de la Base de Datos. De esa manera se limita el acceso a la información más crítica y se limitan las acciones que determinados usuarios pueden hacer sobre determinados objetos.
- **Evitar la redundancia e inconsistencia de los datos** Quizás sea una de las funciones más importantes que debe llevar a cabo cualquier SGBD. La *redundancia de datos* se refiere a la necesidad de asegurar que un dato no se encuentra duplicado en algún otro lugar de la Base de Datos. Es un error muy común y a la vez muy peligroso puesto que puede producir lo que se conoce como *inconsistencia de datos* que tiene lugar cuando, habiendo redundancia, alguna de las copias duplicadas no se actualiza correctamente y el mismo dato toma más de un valor diferente.
- **Evitar anomalías en el acceso concurrente** Puesto que normalmente varios usuarios intentarán acceder simultáneamente a la misma Base de Datos, habrá que dar solución a los problemas que pueda ocasionar la modificación simultánea de un mismo dato por más de un usuario. Cualquier SGBD con soporte multiusuario deberá dar soporte a esta característica.
- **Garantizar la correcta ejecución de las transacciones**

Si hablamos de un SGBD con soporte para transacciones (SGBD transaccional), éste tendrá que proporcionar soporte para una correcta ejecución de las mismas. A ese respecto existen un conjunto de normas conocidas como **ACID**, que marcan aquellas características que todo SGBD debe soportar para un correcto funcionamiento de las transacciones.



- Atomicidad (**A**tomicity): Se debe asegurar si la operación se ha llevado a cabo o no. No puede existir un término medio
- Consistencia (**C**onsistency): Todas las operaciones que empiezan a ejecutarse deben asegurarse de que pueden terminar sin romper ninguna de las reglas de integridad de la Base de Datos
- Aislamiento (**I**solation): Si se ejecutan dos transacciones sobre los mismos datos al mismo tiempo (operaciones concurrentes), deben ser independientes y no deben generar ningún error
- Durabilidad (**D**urability): Una vez se realice la operación, ésta debe persistir en el tiempo. Incluso ante un fallo del sistema, si la operación se realizó correctamente no se podrá deshacer

Como ejemplo para entender el concepto de transacción hay que imaginar el movimiento de dinero entre dos cuentas bancarias. Se trata de una sola operación que conlleva la ejecución de más de una tarea (retirar el dinero de la cuenta origen e incrementarlo en la cuenta destino) de forma que nunca debería ejecutarse a medias (no podemos sólo retirar el dinero de la cuenta origen o sólo incrementar el saldo en la de destino). Se trata de una transacción, y por tanto deberá cumplir las reglas ACID para funcionar correctamente.

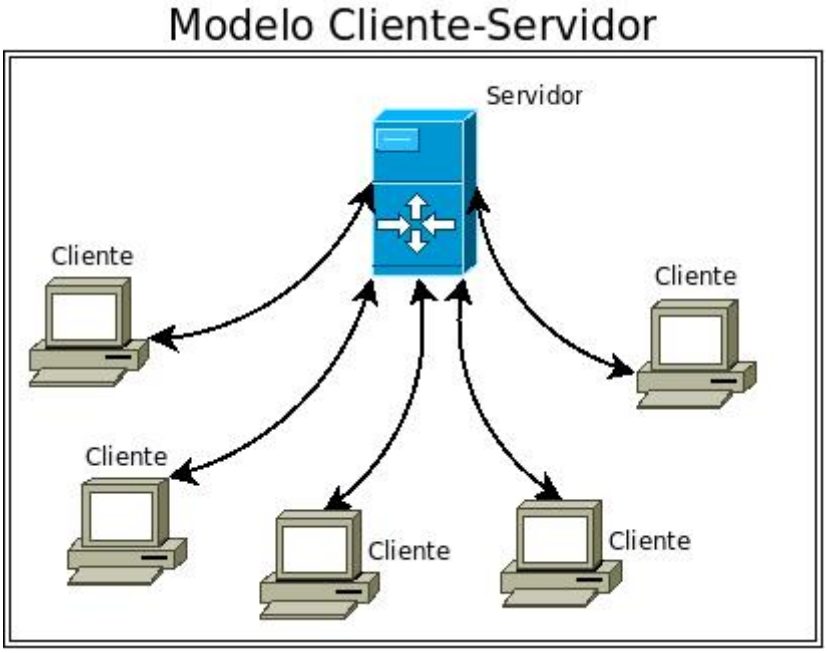
- **Respaldo y recuperación** La gran mayoría de SGBD proporcionan herramientas adicionales para el respaldo y la recuperación. Debe ser posible realizar una copia de respaldo de toda la Base de Datos en cualquier momento para su posterior recuperación en caso de fallo.

No todos los SGBD implementan todas las funcionalidades que se han descrito anteriormente puesto que en determinadas ocasiones no nos serán necesarias. Por ejemplo, Microsoft Access no da soporte a acceso concurrente ni tampoco da solución a ciertos problemas de seguridad y control de acceso, pero no es uno de sus objetivos. Se trata de una Base de Datos incluida en un

paquete ofimática y no debería usarse en entornos más complejos. Por otro lado, sistemas como Oracle, SQL Server o MySQL implementan muy buenos soportes para acceso concurrente, seguridad, control de acceso y otras muchas funcionalidades, puesto que son soluciones destinadas a entornos más complejos. Otros SGBD apenas implementan algunas de las características deseables de estas herramientas, puesto que están destinados para entornos con poca capacidad de cálculo, como puede ser cualquier dispositivo móvil. Por citar un ejemplo, el sistema operativo Android utiliza un SGBD llamada SQLite por sus reducidas dimensiones y necesidades en cuanto a la capacidad de cálculo

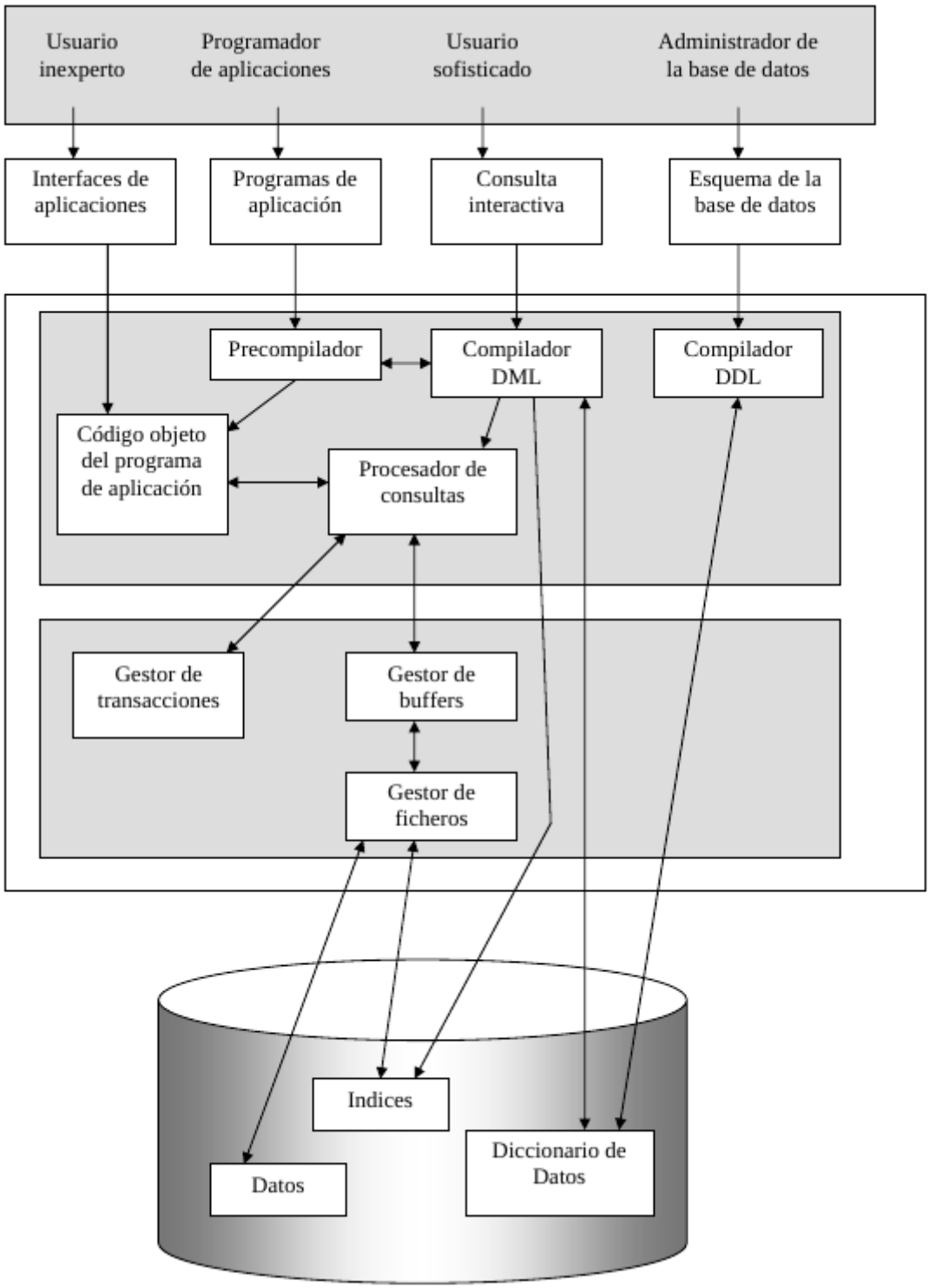
Componentes

Normalmente existe una separación clara entre los componentes principales de la gran mayoría de SGBD. Lo más común es la separación entre lado Cliente y lado Servidor (*Arquitectura Cliente-Servidor*), aunque no todos los SGBD siguen esa arquitectura. La gran parte de la funcionalidad la encontraremos en el lado del servidor, puesto que es donde se almacenarán siempre los datos. El lado cliente es una herramienta bastante simple que sirve de comunicación entre usuario y lo que se conoce como el Motor (Engine) del SGBD, que forma la parte servidor.



Si miramos dentro del motor de un SGBD, podemos encontrarnos los siguiente componentes:

- **Administrador de Almacenamiento**: Controla el acceso a la información de la Base de Datos en el disco. Se encarga tanto de los *buffers* como de los archivos donde se almacenan todos los datos.
- **Procesador de consultas**: Se encarga de recibir las peticiones de consultas y debe encontrar la manera más óptima de realizarlas a través de lo que se conoce como *Plan de Ejecución*. Emite órdenes al Administrador de Almacenamiento puesto que es quién accede finalmente a los datos para proporcionarlos.
- **Gestor de transacciones**: Asegura que se mantenga la integridad de la Base de Datos tras la ejecución de transacciones, sean o no fallidas.



Tipos

Atendiendo al modelo de datos utilizado, destacan los siguientes tipos, siempre teniendo en cuenta que algunas Bases de Datos pueden estar en varias de estas categorías. Por citar un ejemplo, *db4o* es un conocido SGBD orientado a objetos y a la vez es *NoSQL* puesto que no utiliza el lenguaje SQL para acceder a los datos que almacena.

Relacional

Se ajustan al modelo relacional, que es actualmente el modelo de Base de Datos más extendido. Es el modelo de datos más utilizado actualmente, basado en el concepto de tablas y las relaciones entre ellas como forma de relacionar la información entre sí. Destacan Ms Access [http://products.office.com/es-ES/access], Oracle [https://www.oracle.com/database/index.html], Microsoft SQL Server [https://www.microsoft.com/en-us/cloud-platform/sql-server] y MySQL [http://www.mysql.com] como los SGBD relacionales más extendidos actualmente.

id-cliente	*nombre-cliente*	*calle-cliente*	*ciudad-cliente*
19.283.746	González	Arenal	La Granja
01.928.374	Gómez	Carretas	Cerceda
67.789.901	López	Mayor	Peguerinos
18.273.609	Abril	Preciados	Valsaín
32.112.312	Santos	Mayor	Peguerinos
33.666.999	Rupérez	Ramblas	León
01.928.374	Gómez	Carretas	Cerceda

Tabla *cliente*

número-cuenta	*saldo*
C-101	500
C-215	700
C-102	400
C-305	350
C-201	900
C-217	750
C-222	700

Tabla *cuenta*

id-cliente	*número-cuenta*
19.283.746	C-101
19.283.746	C-201
01.928.374	C-215
67.789.901	C-102
18.273.609	C-305
32.112.312	C-217
33.666.999	C-222
01.928.374	C-201

Tabla de relación *cliente-cuenta*

Los SGBD relacionales proporcionan una serie de funcionalidades, entre las que destacan las siguientes:

- **Tipos de datos**: Cada columna tiene un tipo de dato definido de manera que el SGBD no permite almacenar valores de otro tipo en dicha columna
- **Restricciones**: Es posible definir restricciones que obliguen a cumplir una serie de requisitos a los valores que se almacen en una columna determinada
- **Integridad referencial**: En el momento de registrar algún nuevo dato que deba estar relacionado con otro, el SGBD comprobará que el segundo existe antes de permitir el registro. En caso contrario no lo permitirá
- **Consultas complejas**: Es posible realizar consultas muy complejas, incluso aquellas que inicialmente no habían sido tenidas en cuenta durante el diseño de la Base de Datos

Las Bases de Datos relacionales son apropiadas si:

- Es necesario realizar consultas muy complejas entre diferentes tablas
- Es necesario validar la información entre tablas
- Se permite que un dato pueda tener cualquier número de valores
- Es necesario construir nuevas consultas que no habían sido planificadas cuando se diseñó la Base de Datos

Por otro lado, no son apropiadas si:

- Es necesario definir una topología determinada para trabajar con la aplicación correspondiente, tal y como podría ser una jerarquía o red. Para estos casos existen tipos de SGBD específicos que proporcionan mayor rendimiento
- En muy contadas situaciones donde las necesidades de rendimiento son muy elevadas y las reglas a las que se someten las Bases de Datos relacionales ralentizan el funcionamiento global

XML

XML es un lenguaje para almacenar información de forma jerárquica. Es un lenguaje que no proporciona ninguna forma de crear, buscar, modificar o validar información. Sin embargo, se usa ampliamente para almacenar, transferir y recuperar datos jerárquicos y hay muchas herramientas para trabajar con estos ficheros de una manera sencilla.

Además, es un lenguaje muy utilizado para el intercambio de información entre aplicaciones remotas a través de Internet.

En definitiva, el lenguaje XML sólo especifica la forma de escribir el fichero y organizar la información mediante las etiquetas que da sentido a cada uno de los valores almacenados y la jerarquía entre estas etiquetas marca la dependencia de las mismas. Así, en función del uso que queramos darle, deberemos hacernos con la herramienta determinada que nos permita gestionar el fichero XML de la forma que más nos convenga. Para el caso que nos ocupa, existen herramientas como *xbird* que permiten acceder a ficheros XML para trabajar como si de una Base de Datos se tratara, proporcionando lo que se conoce como

procesador *XQuery* para dar soporte a un lenguaje de consultas y proporcionando un rendimiento óptimo en el acceso a los datos, pudiéndose trabajar con ficheros de hasta varios GBs.

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

```
- <Employees TotalRows="9">
  - <Employee id="1">
      <SalesPerson>Nancy Davolio</SalesPerson>
      <Title>Sales%20Representative</Title>
      <BirthDate>December 8, 1968</BirthDate>
      <HireDate>May 1, 1992</HireDate>
      <Extension>5467</Extension>
    </Employee>
  - <Employee id="2">
      <SalesPerson>Andrew Fuller</SalesPerson>
      <Title>Vice%20President%2C%20Sales</Title>
      <BirthDate>February 19, 1952</BirthDate>
      <HireDate>August 14, 1992</HireDate>
      <Extension>3457</Extension>
    </Employee>
  + <Employee id="3"></Employee>
  + <Employee id="4"></Employee>
  + <Employee id="5"></Employee>
  + <Employee id="6"></Employee>
  + <Employee id="7"></Employee>
  + <Employee id="8"></Employee>
  + <Employee id="9"></Employee>
</Employees>
```

Las Bases de datos XML son apropiadas si:

- Los datos son jerárquicos
- Las herramientas XML proporcionan las funcionalidades que necesitas
- Vas a utilizar aplicaciones software que entienden el formato XML

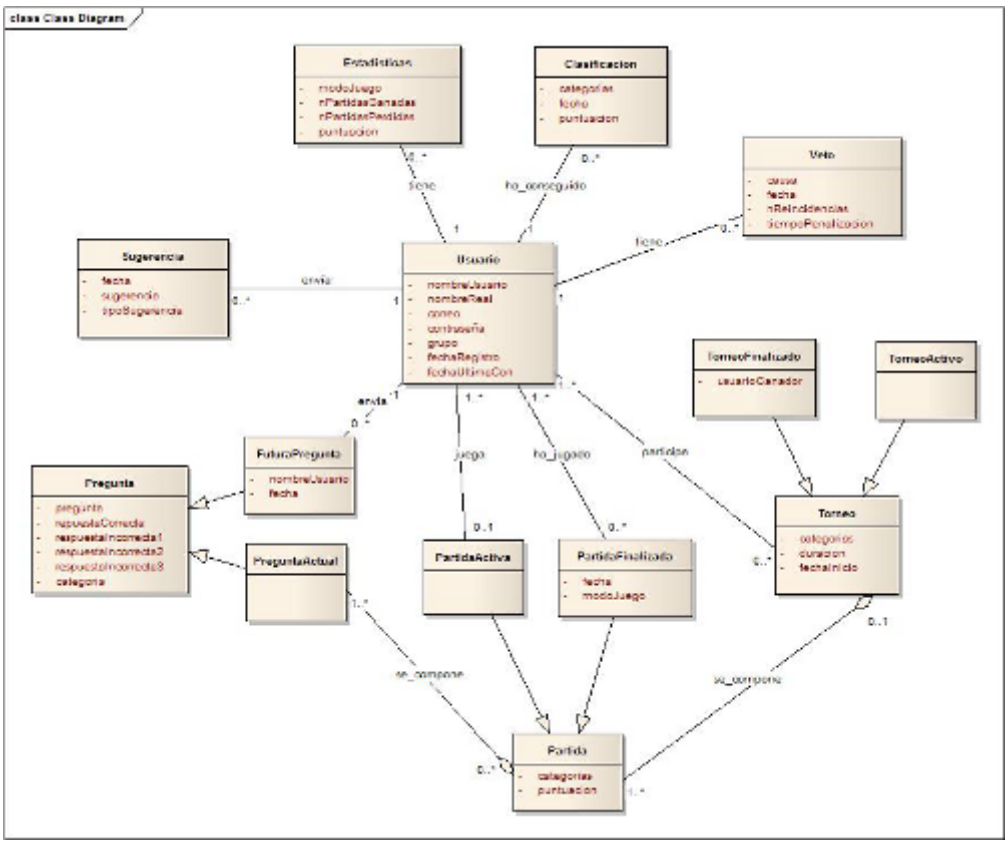
Por otro lado, no convienen si:

- Los datos no son jerárquicos
- Es necesario realizar validaciones complicadas
- Es necesario relacionar de alguna manera la información
- La Base de Datos es muy grande como para reescribir el fichero completo cada vez que se realiza algún cambio

Orientado a objetos

En este modelo todos los elementos de la Base de Datos se representan como objetos y siguen el paradigma de la *Programación Orientada a Objetos*. El objetivo de este tipo de Bases de Datos es el trabajo en unión con algún lenguaje de programación del mismo paradigma. De esa manera los objetos de la aplicación tienen una correspondencia directa con los objetos en la Base de Datos, puesto que pueden usar el mismo modelo de datos.

Puesto que el paradigma de orientación a objetos es relativamente nuevo, y aunque en la programación está bastante extendido, en las Bases de Datos no está muy aceptado. Existen ya algunos SGBD comerciales pero su uso es minoritario. Entre ellos destaca *db4o* e |InterSystems Caché [http://www.intersystems.com/es/nuestros-productos/cache/vision-general/]



Las Bases de Datos orientadas a objetos son apropiadas si:

- La aplicación se está programando con algún lenguaje de programación orientado a objetos
- No es necesario realizar consultas muy complejas

Por otro lado, no convienen si:

- Es necesario comunicarse con herramientas externas para almacenar cierta información en otros modelos más habituales como el relacional
- Es necesario realizar consultas complejas
- La aplicación no se va a programar con un lenguaje orientado a objetos
- Es necesario realizar ciertas validaciones que este tipo de bases de datos no proporciona

Objeto-Relacional

Aparecen como una extensión al modelo relacional, por lo que los datos se almacenan en forma de tablas. Y además permiten la utilización de ciertas características propias de la Programación Orientada a Objetos (POO) como pueden ser la Herencia o el uso de tipos más complejos como son las colecciones o los campos estructurados. Cabe destacar PostgreSQL [http://www.postgresql.org] como SGBD objeto-relacional, que actualmente está bastante extendido.

Las Bases de Datos objeto-relacionales son apropiadas si:

- La arquitectura de la aplicación está orientada a objetos
- Es necesario realizar consultas complejas de datos relacionados entre sí

- Es necesario realizar validaciones de datos
- En el equipo de trabajo se mantiene una separación entre programadores de la aplicación y la base de datos

Por otro lado, no convienen si:

- Para el desarrollo de la aplicación no se va a utilizar un lenguaje orientado a objetos

NoSQL

Aparecen como un cambio de tendencia frente a los clásicos SGBD relacionales. Su principal característica es que no emplean el lenguaje SQL como principal lenguaje de consultas (de ahí su nombre, No only SQL). Tampoco soportan operaciones *JOIN* como en los sistemas relacionales y no suelen garantizar *ACID*.

Su principal característica es la escalabilidad horizontal. Es decir, el rendimiento de la Base de Datos se mantiene a medida que se van añadiendo nodos (equipos) al sistema. Esto comenzó a ser una necesidad cuando las principales compañías de Internet (Google, Amazon, Twitter, . . .) comenzaron a necesitar grandes sistemas que dieran respuesta en tiempo real. Los clásicos SGBD relacionales no daban respuesta a sus necesidades, puesto que emplean gran parte del tiempo en realizar las validaciones y verificar la coherencia entre los datos. Los sistemas NoSQL restan importancia a este aspecto por lo que permiten ofrecer un mayor rendimiento al conjunto para determinados tipos y estructuras de datos.

En general, las Bases de Datos NoSQL son apropiadas si:

- Es necesario manejar grandes cantidades de datos
- Es necesario escalar (añadiendo nodos al sistema)

Por otro lado, no son apropiadas para cualquier tipo de proyecto puesto que no siempre el rendimiento será lo más importante, o al menos no a costa de perder reglas de validación o coherencia o no garantizar *ACID*.

Algunos ejemplos conocidos de SGBD NoSQL son MongoDB [https://www.mongodb.com] y Cassandra [https://cassandra.apache.org], aunque también podemos considerar como SGBD NoSQL a *db4o*, que al tratarse de un SGBD orientado a objetos no emplea SQL como lenguaje principal de consulta, aunque difiere bastante en funcionamiento de los dos anteriores y de algunas de las características principales de los SGBD NoSQL.

Clasificación según accesos simultáneos soportados

Monousuario

Son aquellos SGBD que no permiten que más de un usuario pueda estar conectado con la Base de Datos al mismo tiempo. Normalmente no soportan ningún tipo de control de usuario por lo que la Base de Datos se reducirá a un simple fichero al que se podrá acceder desde la aplicación como si de cualquier otro tipo de documento se tratara. Así, no soportan el control de concurrencia ni cualquier otra característica propia de los sistemas multiusuarios. Algunos SGBD monousuario son *Ms Access* y *LibreOffice Base*.

Multiusuario

Permiten conexiones simultáneas con una misma Base de Datos. Tiene soporte para control de cuentas de usuario, y por tanto incluyen todo el soporte para el control de concurrencia, propio de cualquier sistema multiusuario. Algunos SGBD multiusuario son *MySQL*, *PostgreSQL*, *Microsoft SQL Server*, *Oracle*, *MongoDB* y *Cassandra*.

Clasificación según tamaño

SGBD ligero

Se trata de pequeños SGBD (incluso < 1MByte de tamaño) que dan soporte a algunas de las principales funciones de estas herramientas, ideales para entornos donde no es necesario un gran rendimiento o bien no se dispone de gran potencia de cálculo (dispositivos móviles). El más conocido en la actualidad es *SQLite*

SGBD Ofimática

Algo más potentes que los SGBD considerados como ligeros, con algunas funciones más, muy utilizados en Ofimática para pequeñas Bases de Datos (normalmente algunos MBytes). *Ms Access* y *LibreOffice Base* son dos ejemplos de esta categoría

SGBD de alto rendimiento

Ideales para entornos dedicados a la gestión de datos o bien para aplicaciones que manejan ya una gran cantidad de datos y necesitan algunas funciones que otros SGBD no proporcionan, como control de usuarios, concurrencia, soporte para transacciones u otras operaciones. *MySQL*, *Microsoft SQL Server*, *Oracle* y *PostgreSQL* son algunos ejemplos Otros SGBD de alto rendimiento, aunque son algunas características diferentes a los anteriores, son *MongoDB* y *Cassandra*.

Tipos de Bases de Datos

Bases de Datos centralizadas

Son el tipo de Bases de Datos más común. Normalmente, en grandes corporaciones, se dedica un único equipo para el almacenamiento y gestión de los datos utilizando algún tipo de SGBD. Toda la Base de Datos se encuentran ubicada lógica y físicamente en ese equipo. Es la solución más habitual puesto que normalmente es suficiente y además es mucho más sencilla de implementar.

Ventajas

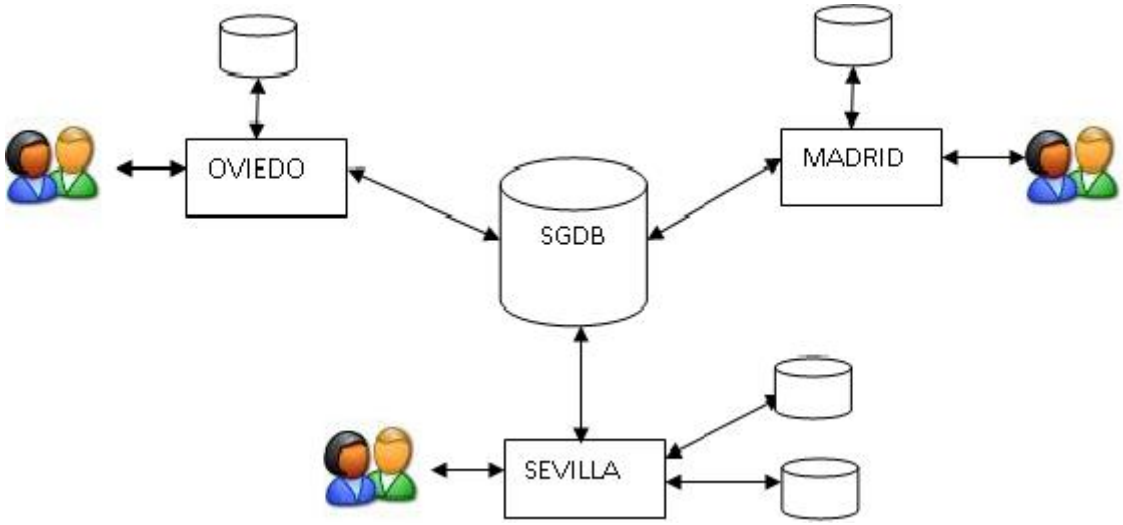
- Fácil implementación. Al tratarse de una solución ampliamente adoptada es más sencillo de implementar y más fácil actuar ante posibles fallos
- Diseño sencillo. Al tratarse de una solución conocida, el diseño lógico es también más sencillo de realizar

Desventajas

- En caso de fallo, al existir una sola ubicación física de la Base de Datos, éste afectará a todos los datos almacenados en ella

Bases de Datos distribuidas

Una Base de Datos distribuida es un conjunto de bases de datos relacionadas lógicamente entre sí que se encuentran localizadas en diferentes ubicaciones físicas. Puede tratarse de dos máquinas conectadas entre sí mediante una *LAN* o bien a través de Internet. Como consecuencia, dichas Bases de Datos pueden realizar tareas de forma autónoma o bien de forma distribuida. En el caso de las operaciones realizadas de forma distribuida, el usuario de cualquiera de las dos (o más) máquinas tendrá acceso a toda la información como si éstos estuvieran siendo accedidos de forma local.



Ventajas

- Es más barato crear una red de máquinas de bajo rendimiento que disponer de una sola máquina muy potente
- En cuanto a la disponibilidad, un fallo en cualquiera de los nodos no afecta al resto por lo que el resto de la Base de Datos seguirá accesible

- Puesto que los datos se encuentran distribuidos, éstos pueden ubicarse donde más interesa. Por ejemplo en el departamento con el que tienen relación. Además, este departamento dispondrá de la autonomía necesaria para controlar dicha información

Desventajas

- Un sistema de Bases de Datos distribuida es complejo de poner en marcha y mantener
- En cuanto a la seguridad, al disponer de varios nodos, se dispone de varios sistemas diferentes que habrá que proteger
- Es un sistema bastante nuevo, por lo que existe poca experiencia ante fallos poco comunes
- Al disponer de varios nodos es posible que se necesite más mano de obra especializada en cada uno de ellos

Arquitectura de una aplicación

Una vez que hemos visto qué es un SGBD, una Base de Datos y los diferentes tipos con los que nos podemos encontrar justo con sus ventajas e inconvenientes, conviene echar un vistazo de una manera más global para ver cuál es la posición de un SGBD y la Base de Datos en el conjunto de una aplicación ya desarrollada.

Como ya se ha visto más arriba, muchos de los SGBD trabajan siguiendo lo que se conoce como arquitectura cliente-servidor. Siguiendo esta misma filosofía, existe la *programación por capas*, que consiste en una arquitectura en la que el principal objetivo es la separación de las diferentes capas lógicas de una aplicación (presentación, negocio y datos). Por ello, la arquitectura más conocida es la arquitectura de 3 capas, en la que la estructura de la aplicación se separa en tres niveles. Aunque no siempre las 3 capas están separadas físicamente, es conveniente que lo estén lógicamente. La separación física vendrá dada por la propia naturaleza de la aplicación (aplicación web) o bien por las propias necesidades en cuanto a rendimiento de la misma (uno o más equipos para dar soporte a la capa de negocio o datos).

- **Capa de presentación** Es la capa que ve el usuario. Presenta la aplicación al usuario, le permite interactuar con el mismo y le muestra la información que éste solicita. En definitiva es lo que se conoce como interfaz de usuario.

- **Capa de negocio** En esta capa se encuentra toda la lógica de la aplicación y será donde se realicen todos los procesos de tratamiento de la información obtenida en la capa de datos y cuyos resultados se muestran al usuario mediante la capa de presentación.

- **Capa de datos** En esta capa residen los datos. En ella se encuentran los SGBD y reciben peticiones para acceder o escribir datos desde la capa de negocio.

En aplicaciones de tamaño medio es bastante habitual encontrar un modelo lógico basado en 3 capas pero físicamente distribuido solamente en 2. En estos casos lo habitual será que el usuario cargue la capa de presentación en su propio equipo (ya sea ejecutando la aplicación o cargando la web en su navegador) y que las capas de negocio y datos se encuentren en otro equipo remoto (el servidor). A medida que la aplicación se haga más exigente las capas de negocio y datos pueden separarse en diferentes equipos (normalmente físicamente más cerca) y según se exige un mayor rendimiento se pueden añadir equipos a cualquiera de estas dos capas.