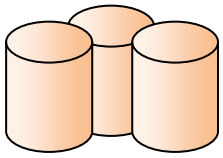
Sistemas de almacenamiento de la información

Índice



1. Introducción

Es evidente que estamos en la era de la información y que cualquier empresa, organización, por muy pequeña que sea, necesita para su funcionamiento tratar grandes cantidades de datos, información que le llega del exterior, que es generada en la propia empresa o información que suministra al exterior.
Toda esta información se tiene que poder almacenar y organizar de manera que pueda ser recuperada y utilizada de forma eficiente.
La forma en que esta información es almacenada a nivel físico en el disco (en código binario, en determinadas pistas del disco, etc.) es tarea del sistema operativo y no será objeto de nuestro estudio. Nosotros nos centraremos en la forma lógica en que está almacenada, es decir las diferentes estructuras que permiten organizar esa información de forma eficiente.
Estudiaremos las ventajas e inconvenientes de las diferentes estructuras y sus aplicaciones.

2. Evolución de los sistemas de almacenamiento

Inicialmente los soportes disponibles para almacenar los datos, limitaban muchísimo la forma en que se podía organizar la información. Cuando sólo se disponía de tarjetas perforadas *PUNCHED CARD* y posteriormente de cintas magnéticas *MAGNETIC TAPES*, estos soportes sólo permitían acceso secuencial a la información y la cantidad de información que podían almacenar era muy limitada.

Debida a la necesidad de organizar los datos de alguna forma que se pudiera almacenar y tratar de forma coherente, surgieron los ficheros (archivos) *FILES*. En cada fichero se almacenaban los datos referentes a un mismo tema, por ejemplo un archivo de clientes, un archivo de facturas, etc.

Luego con la aparición de los discos magnéticos *MAGNETIC DISKS*, se pudieron diseñar archivos directos e indexados que permitían un mejor acceso a la información.

Estos archivos presentaban una serie de inconvenientes que abrieron la puerta a las bases de datos *DATABASES* que también pasaron por varios formatos, empezando por las bases de datos jerárquicas *HIERARCHICAL DATABASES*, luego aparecieron las bases de datos en red *NAVIGATIONAL DATABASE – NETWORK MODEL* que fueron desbancadas por las bases de datos relacionales *RELATIONAL DATABASE* que se mantuvieron en exclusividad (exceptuando algún tipo de propósito específico) hasta el boom de Internet que propició la aparición de otro tipo de base de datos, las bases de datos NoSQL.

Hoy en día se siguen utilizando ficheros convencionales para propósitos muy concretos pero en la inmensa mayoría de aplicaciones han sido relegados por las bases de datos.

Para la inmensa mayoría de aplicaciones de gestión se utilizan bases de datos relacionales y para el mundo Internet aunque se siguen utilizando bases de datos relacionales, las bases de datos NoSQL se están abriendo camino.

Las bases de datos NoSQL por el momento resuelven problemas muy concretos con ciertos inconvenientes frente a las bases de datos relacionales por lo que se prevee que sigan conviviendo con las anteriores.

3. Ficheros

El fichero (también denominado archivo) *FILE* es la estructura básica de almacenamiento de la información *DATA STORAGE*. Un fichero puede albergar información de todo tipo (cartas, imágenes, películas, música, documentación escrita, etc) para reconocerlo tiene un nombre y una extensión que se indica detrás del nombre delante de un punto. Por ejemplo hoja.bmp es un fichero que se llama hoja y que contiene una imagen en formato binario (.bmp), documento.txt es un archivo de texto plano.

3.1. Tipos de ficheros y formatos

La información guardada en un fichero se almacena en binario *BINARY*, en forma de bits ("ceros" y "unos") y por tanto para dar sentido a lo almacenado es necesaria su interpretación por parte del sistema operativo o de los programas. El formato y tipo de fichero determina la forma de interpretar la información que contiene. Por ejemplo en un fichero .bmp la imagen se almacena en forma de píxeles cada uno de los cuales tiene asociado el código binario del color almacenado en ese píxel, si abrimos el archivo sin utilizar el programa adecuado no veremos una imagen.

Haz la prueba, busca una imagen en tu ordenador y ábrela con el Bloc de notas (botón derecho del ratón, elige la opción *Abrir con*, elegir *programa predeterminado*, desmarca la casilla *Usar siempre el archivo seleccionado...* (¡ojo!), *otros programas*, busca *Bloc de notas*) verás una ristra de caracteres ininteligibles porque cada byte almacenado en el archivo es interpretado como un caracter en vez del nº de color del píxel.

Tradicionalmente, los ficheros se han clasificado de muchas formas, según su organización (secuencial, directa, indexada), según su uso (maestros, históricos, movimientos) o según su contenido (binario, texto), pero desde la aparición de las bases de datos ya no se utilizan las primeras clasificaciones, no obstante las comentaremos aquí ya que los términos se siguen utilizando.

- La organización de un fichero dicta la forma en que están almacenados los datos en él.

En la **organización secuencial** *SEQUENTIAL ORGANIZATION* los datos se organizan en registros *RECORDS* y campos *FIELDS* y cada registro se almacena detrás del último en el orden en que son grabados.

Con lo cual no hay huecos entre registros pero para poder leer la información almacenada en un registro determinado tenemos que leer todos los registros anteriores lo que puede ser muy lento si el archivo tiene muchos registros.

En un archivo con **organización directa** *DIRECT ORGANIZATION* (*DIRECT/RELATIVE FILE*) se reserva un lugar para cada registro y cada registro se coloca en el lugar que le corresponde de acuerdo a su nº de registro. El registro nº 1 es el primero, el nº 2 es el segundo y así sucesivamente. En este caso si primero se graba el nº 10 y después el nº 1, el nº 1 seguirá siendo el primero mientras que en un fichero con organización secuencial el primero sería el registro nº 10.

La organización directa tiene la ventaja que se puede acceder directamente a la información de un registro conociendo su nº de registro mientras que como desventaja el archivo puede estar lleno de huecos (imagina que primero se graba el registro nº 10000 y después registros del orden de 1 a 100, tendremos 9900 registros vacíos ocupando espacio, otra desventaja es que para identificar un registro sólo podemos utilizar su nº de registro, no podemos utilizar campos alfanuméricos (por ejemplo un código que contenga letras).

En un fichero con **organización indexada** *INDEXED ORGANIZATION* los registros se almacenan conforme se van grabando en posiciones contiguas pero luego podemos recuperar los registros directamente mediante un tabla de índice. Además de los datos del archivo se almacena una tabla que nos dice dónde se encuentra cada registro como el índice de un libro que nos dice en qué página se encuentra cada apartado (dicho de una forma muy simplificada, ya volveremos sobre los índices en un tema posterior) y de esta forma eliminamos los huecos pero podemos recuperar un registro determinado sin tener que pasar por todos los que le preceden y el índice puede estar basado en un campo o varios campos alfanuméricos. Como desventaja las tablas de índice *INDEXES* ocupan espacio, pero sobre todo los índices se van degradando con el tiempo (algo parecido a cuando tenemos un disco duro sin desfragmentar) y la lectura de los registros se puede ralentizar considerablemente. A pesar de estos inconvenientes, sus ventajas hacen que este tipo de organización se sigue utilizando como veremos en las bases de datos.

- Según su uso tenemos los archivos maestros, históricos y de movimientos.

Los **archivos maestros** *MASTER FILE* son los que contienen la mayoría de los datos de un sistema de información, por ejemplo los datos de nuestros clientes, de nuestros empleados, etc...

Un **archivo de movimientos** *TRANSACTION FILE* almacena las modificaciones que se tienen que realizar sobre un archivo maestro. Se utilizaban muchísimo cuando todavía las aplicaciones no eran interactivas, se guarda en él todas las modificaciones a realizar sobre un archivo maestro y luego se ejecuta el programa de mantenimiento del maestro que lee cada registro del fichero de movimientos y actualiza el maestro (por ejemplo inserta nuevos clientes, borra algunos clientes, modifica la dirección de un cliente, etc.).

En el **archivo histórico** *HISTORY FILE* se suelen almacenar datos que queremos guardar pero que ya no se utilizan más que para sacar estádisticas o simplemente para tener constancia de lo que había en los archivos, por ejemplo un histórico de facturas en el que guardamos las facturas de años anteriores.

- Según su contenido.

El contenido de un archivo puede ser tratado como texto o como datos binarios, es decir, los bits almacenados pueden ser traducidos por el sistema operativo a caracteres alfabéticos y números (caracteres alfanuméricos) que entiende el ser humano, o pueden ser tratados como componentes de estructuras de datos más complejas que sólo entienden (saben interpretar) los programas diseñados a tal efecto.

Esta distinción nos abre una clasificación entre ficheros de texto y fichero binarios.

3.2. Formatos

- Ficheros de texto *TEXT FILES*

Los ficheros de texto suelen llamarse también ficheros planos o ficheros ASCII. ASCII es un estándar que asigna un valor numérico a cada carácter, y ese valor numérico convertido a binario es lo que se almacena. Esto permite que los caracteres así almacenados sean fácilmente legibles por el usuario ya que el sistema sólo tiene que realizar una conversión utilizando la tabla de códigos *CODES*.

Por eso se dice que estos ficheros son directamente legibles por el ser humano.

Hoy en día se utilizan también tablas de códigos más amplias para poder representar una gama más amplia de caracteres como por ejemplo la codificación UNICODE.

Los ficheros de texto, aunque no necesitan un formato para ser interpretados, suelen tener extensiones para indicar qué tipo de información se halla dentro del archivo, por ejemplo:

- Ficheros de configuración *CONFIGURATION FILES*: Son ficheros cuyo contenido es texto sobre configuraciones del sistema iperativo o de alguna aplicación. Pueden tener las extensiones .ini, -inf, .conf

- Ficheros de código fuente *SOURCE CODE*, contienen el texto de programas informáticos desarrollado por el programador. Ejemplos: .sql, .c, .vb, .java.

- Ficheros de páginas web *WEB PAGE FILE*, las páginas web son ficheros de texto que incluyen hipervínculos LINKS (enlaces a otros archivos u otras partes del texto) y son interpretados por los navegadores web *WEB BROWSER*. Ejemplo: .htm, .html, .php

- Ficheros de texto con o sin formato enriquecido.
  El tipo texto sin formato enriquecido es el .txt
  Con formato enriquecido, son textos que contienen códigos de control para ofrecer una visión de texto más elegante: .rtf, .ps, .tex (para fórmulas matemáticas)

- Ficheros binarios *BINARY FILES*

Los ficheros binarios requieren de un programa que interprete el código bunario almacenado y lo traduzca a algo inteligible por el ser humano.

Aquí tienes algunos tipos de archivos binarios:

- De imagen *IMAGE*: .bmp, .jpg, .jpeg, .gif, .tiff, .png, .psd (Adobe Photoshop), .cdr (CorelDraw), .ai (Adobe Illustrator)

- De vídeo *VIDEO*: .mpg, .mov, .avi, .qt, .mp4

- Ficheros comprimidos *COMPRESSED FILE*: .zip, .rar, .tar, .gz

- Ejecutables *EXECUTABLE FILES*: .exe, .com, .cgi

- Procesadores de texto *WORD PROCESSOR*: .doc, .docx, .odt

- De aplicaciones varias *APPLICATIONS*: .xls (Excel), .ppt, .pps (Powerpoint), odt (OpenOffice Writer), odp (OpenOffice Presentación), odx (OpenOffice Calc).

Referencia más amplia en http://www.hispazone.com/Articulo/91/Los-archivos:-tipos--extensiones-y-programas-para-su-uso.html

4. Las bases de datos

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Una base de datos normalmente lleva asociado un sistema gestor de base de datos *DATABASE MANAGEMENT SYSTEM* que consiste en un conjunto de programas que permiten gestionar la información almacenada en ella.

Por ejemplo un usuario puede utilizar el sistema gestor muy conocido Microsoft Access para guardar información sobre las películas que tiene en casa, o para llevar un control de los gastos de casa. Podría, utilizando dicho gestor crear una base de datos para las películas y otra para sus gastos.

Muchas veces se confunden base de datos y sistema gestor, uno contiene los datos y el otro es el programa que permite gestionar estos datos y a veces contiene la propia base de datos como en el caso de Microsoft Access.

Todas las bases de datos no tienen las mismas características por lo que se suelen clasificar en distintas categorías.

Podemos realizar dicha clasificación:

. Según su ubicación

. Según la organización lógica de la información.

. Según su propósito

4.1. Tipos de bases de datos según su ubicación

❖ **Bases de datos locales**

En modo local tenemos la base de datos y el usuario ubicados en el mismo ordenador. Un ejemplo de base de datos que funciona en modo local *LOCAL MODE* es Microsoft Access, MS Access es una base de datos fácil de manejar por usuarios poco expertos que funciona bien en modo local y mientras no tenga que albergar grandes cantidades de información.

Ventajas:

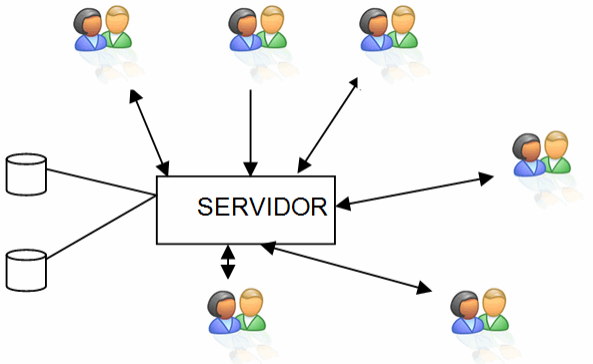
- Economía *ECONOMY* – Es la más barata.
- Simplicidad *SIMPLICITY* – No se necesita llevar controles de accesos concurrentes *CONCURRENCY*, de transmisión de datos *DATA TRANSMISSION*, etc.

Desventajas:

- Monousuario *SINGLE USER* – En un instante determinado sólo la puede utilizar una persona.
- Capacidad *STORAGE CAPACITY* – Suele tener una capacidad de almacenamiento limitado.

❖ **Bases de datos centralizadas**

En los sistemas centralizados *CENTRALIZED SYSTEMS* tenemos la base de datos completa en un mismo servidor *SERVER*, y todos los usuarios acceden a ese servidor. Que la base de datos esté en un mismo servidor no implica que esté en un solo archivo o en el mismo disco, puede estar repartida.



En modo Cliente/Servidor *CLIENT-SERVER*, la base de datos se encuentra en un ordenador *COMPUTER* (el Servidor) y los usuarios acceden simultáneamente a esa base de datos a través de la red *NETWORK* (sea una red local o Internet) desde sus ordenadores a través de un programa Cliente *CLIENT PROGRAM*.

A nivel de empresas es el sistema que más se utiliza en la actualidad.

Ventajas:

- Multiusuario *MULTI-USER* – Permite que varios usuarios accedan a la vez a la misma información.

- No redundancia – Al estar todos los datos en el mismo servidor, la información no se duplica y es más facil evitar fallos debidos a redundancias.

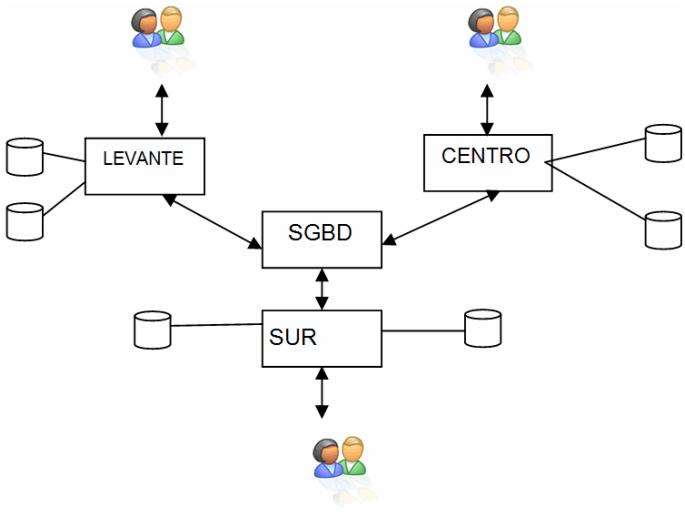
Desventajas:

- Complejidad – Tiene que incluir y gestionar un sistema de usuarios y subesquemas *SUBSCHEMAS*.
- Seguridad – Se tienen que realizar controles para garantizar la seguridad de los datos, tanto a nivel interno como a nivel de comunicaciones.

❖ **Bases de datos distribuidas**

Tenemos la información repartida en distintas localizaciones unidas todas ellas mediante red y un sistema gestor de bases de datos distribuidas *DISTRIBUTED DATABASES*.

Las distintas localizaciones suelen ser distintas geográficamente.



Ventajas:

- Rendimiento *SYSTEM PERFORMANCE* - Una clara ventaja es que es posible ubicar los datos en lugares donde se necesitan con más frecuencia, aunque también se permita a usuarios no locales acceder a los datos según sus necesidades. Esto hace que la información se recupere de forma más rápida y ágil en las ubicaciones locales. Además los sistemas trabajan en paralelo, lo cual permite balancear la carga *TO BALANCE THE LOAD/LOAD BALANCING* en los servidores.
- Disponibilidad *AVAILABILITY* - En caso de que falle la base de datos de alguna localidad, el sistema no se colapsa, puede seguir funcionando excluyendo los datos de la localidad que haya fallado.
- Autonomía local - Un departamento puede controlar los datos que le pertenecen.
- Economía en la implantación *IMPLEMENTATION*- Es más barato crear una red de muchas máquinas pequeñas, que tener una sola máquina muy poderosa.
- Modularidad *MODULARITY* - Se pueden modificar, agregar o quitar sistemas de la base de datos distribuida sin afectar a los demás sistemas (módulos).

Desventajas

- Complejidad en el diseño de datos - Además de las dificultades que generalmente se encuentran al diseñar una base de datos, el diseño de una base de datos distribuida debe considerar la fragmentación *FRAGMENTATION/CHUNKING*, replicación *REPLICATION* y ubicación de los fragmentos *FRAGMENTS/CHUNKS* en sitios específicos, se tiene que trabajar tomando en cuenta su naturaleza distribuida, por lo cual no podemos pensar en hacer composiciones *JOINS* que afecten a tablas de varios sistemas, etc.
- Complejidad técnica - Se debe asegurar que la base de datos sea transparente, se debe lidiar con varios sistemas diferentes que pueden presentar dificultades únicas.

- Economía en el mantenimiento - La complejidad y la infraestructura necesaria implica que se necesitará mayor mano de obra.
- Seguridad - se debe trabajar en la seguridad de la infraestructura así como cada uno de los sistemas.
- Integridad - Se vuelve difícil mantener la integridad, aplicar las reglas de integridad a través de la red puede ser muy caro en términos de transmisión de datos.
- Falta de experiencia - las bases de datos distribuidas son un campo relativamente nuevo y poco común por lo cual no existe mucho personal con experiencia o conocimientos adecuados.
- Carencia de estándares - aún no existen herramientas o metodologías que ayuden a los usuarios a convertir un DBMS centralizado en un DBMS distribuido.

Más información en : http://es.wikipedia.org/wiki/Bases_de_datos_distribuidas

4.2. Tipos de bases de datos según la organización lógica de los datos

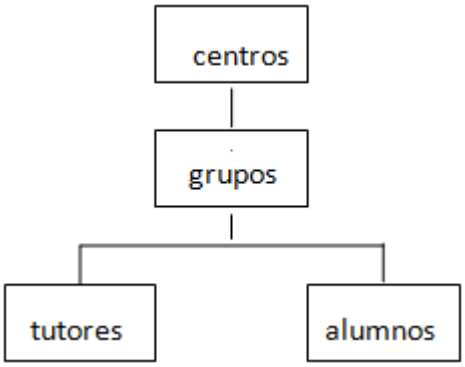
❖ **Las bases de datos jerárquicas**

En una base de datos jerárquica se organizan los datos utilizando estructuras arborescentes (en árbol).

Un **árbol** *TREE* es una estructura jerárquica en la que los elementos se suelen denominar **nodos** *NODE* y existen dependencias entre los nodos.

La **dependencia es de 1:M** del tipo **padre/hijo**. Un hijo *CHILD* no puede tener más de un padre *PARENT*, pero un padre varios hijos.

Ejemplo:



Un ejemplo de base de datos jerárquica es el sistema IMS.

Habían caído en desuso pero los conceptos asociados a este tipo de base de datos están retomando fuerza con la aparición de nuevos sistemas gestores como por ejemplo las bases de datos nativas XML que también organizan la información de modo jerárquico.

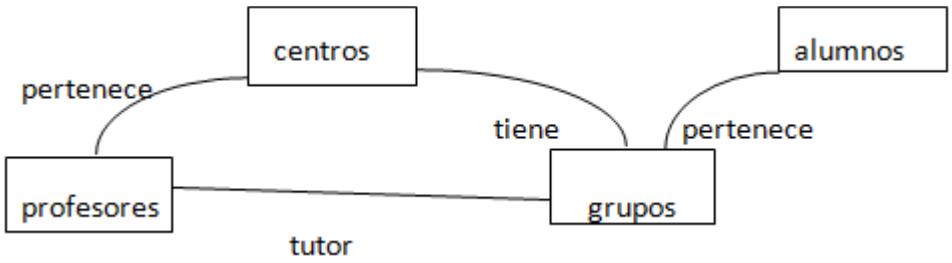
Para ampliar: http://en.wikipedia.org/wiki/Hierarchical_database_model

❖ **Las bases de datos en red**

En una base de datos en red se utiliza la estructura de grafo/red *NETWORK MODEL*, como en el caso anterior los distintos objetos están relacionados entre sí mediante relaciones del tipo 1:M pero en este caso un objeto puede estar relacionado como hijo con varios elementos que serán sus padres. Un este caso las relaciones que se crean se denominan SET y el equivalente al padre se denomina **propietario** *OWNER* y el equivalente al hijo se denomina **miembro** *MEMBER*.

Un ejemplo de sistema en red es el CODASYL. También existen modelos para realizar el diseño de datos orientado a bases de datos en red.

En red podríamos representar lo mismo que la estructura anterior y además lo siguiente:



Los sistemas jerárquico y en red constituyen la primera generación de los SGBD. Pero estos sistemas presentan algunos inconvenientes:

- Es necesario escribir complejos programas de aplicación para responder a cualquier tipo de consulta de datos, por simple que ésta sea.
- La independencia de datos es mínima.
- No incluyen controles de integridad.

Por lo que pronto fueron sustituidos por los sistemas relacionales, aunque últimamente el modelo red se vuelve a retomar con mejoras en bases de datos específicas.
Para ampliar: http://en.wikipedia.org/wiki/Navigational_database

❖ **Las bases de datos relacionales**

Esta es la estructura que se ha impuesto para aplicaciones de gestión *MANAGEMENT APPLICATIONS*, consiste en organizar los datos en forma de tablas *TABLES*, las relaciones *RELATIONSHIP* entre los objetos se consiguen incluyendo en la tabla del hijo, la clave del objeto padre. Como son las que utilizaremos durante todo el módulo hemos reservado un tema especial para ellas.

Están basadas en los principios de:

- Independencia de los datos: Los programas no dependen de los datos y los datos se puedne cambiar sin tener que cambiar los programas.
- Redundancia mínima: No hay que duplicar la información.
- Seguridad e integridad: No pueden haber errores en los datos y se trabaja con datos actualizados.
- Facilidad de recuperación de la información.

Las veremos con más detalles en el próximo tema.

❖ **Las bases de datos objeto-relacionales**

Debido al desarrollo de la programación orientada a objetos *OBJECT-ORIENTED PROGRAMING* las anteriores están dando paso a las bases de datos objeto-relacionales. Son bases de datos relacionales en las que los datos están organizados en tablas, pero esas tablas pueden contener objetos e incorporan conceptos orientados a objetos como por ejemplo la herencia.
Tenemos un tema dedicado a ellas.

❖ **Las bases de datos NoSQL**

Fuente: http://sg.com.mx/revista/42/nosql-la-evolucion-las-bases-datos#.VBZwqfl_uSp

Los RDBMS (Relational DataBase Management Systems) actuales tienen ciertos inconvenientes surgidos a raíz de la aparición de nuevas aplicaciones con nuevos requerimientos.

- Leer datos es costoso cuando manejamos cantidades de datos desorbitadas.

- Las transacciones no son siempre necesarias y tienen ciertos inconvenientes.
- Es complicado escalar.
- No todos los sistemas de información se representan bien con el modelo relacional.

Como respuesta a estos problemas surgió el paradigma NoSQL. NoSQL no es un sustituto a las bases de datos relacionales, es solo un movimiento que busca otras opciones para escenarios específicos, "No uses sólo SQL".

NoSQL no es una solución única, su fortaleza está en su diversidad. El desarrollador cuenta con un abanico de soluciones y puede elegir la mejor para su problema específico.

Por ejemplo tenemos:

- **Almacenes Key-Value** *KEY-VALUE SYSTEMS*

Estas son las bases de datos más simples en cuanto su uso (la implementación puede ser muy complicada), ya que simplemente almacena valores identificados por una clave. ideales para entornos altamente distribuidos y aplicaciones que necesitan escalar horizontalmente. Su limitante está en que no permiten realmente un modelo de datos, todo lo que guardan es un valor binario; aunque algunas implementaciones como Redis modifican este comportamiento permitiendo otro tipo de valores como Listas. Este tipo de base de datos son muy útiles para almacenar sesiones de usuario (usando su username como clave). Son extremadamente rápidos pero no permiten consultas complejas más allá de buscar por su clave. Las más usadas son VMWare Redis, Amazon SimpleDB, Oracle BerkeleyDB y Tokyo Cabinet.

- **Bases de datos columnares** *COLUMN-ORIENTED DATABASES*

Como su nombre lo indica, guardan los datos en columnas en lugar de filas. Esto supone mayor velocidad en lecturas si queremos consultar un número reducido de columnas, son ideales para calcular resúmenes de columnas, pero no son muy eficientes a la hora de realizar escrituras. Por ello este tipo de soluciones es usado en aplicaciones con un índice bajo de escrituras pero muchas lecturas.

- **Bases de datos orientadas a documentos** *DOCUMENT-ORIENTED DATABASES*

Los datos se organizan en documentos y se pueden realizar búsquedas complejas en esos documentos. Ejemplos: MongoDB, Apache CouchDB y Apache Cassandra
Para ampliar: http://en.wikipedia.org/wiki/Document-oriented_database
Englobadas dentro de este tipo tenemos también las **Bases de datos XML nativas** *NATIVE XML DATABASES* en las de los documentos son documentos XML. Por ejemplo: eXist
Se verán el curso que viene en Acceso a Datos.

- **Bases de datos orientadas a grafos** *GRAPH DATABASE*

Como su nombre lo indica, estas bases de datos almacenan los datos en forma de grafo. Esto permite darle importancia no solo a los datos, sino a las relaciones entre ellos. Es mucho más eficiente navegar entre relaciones que en un modelo relacional. Este tipo de bases de datos sólo son aprovechables si la información se puede representar fácilmente como una red. Algo que ocurre mucho en redes sociales o sistemas de recomendación de productos, donde además se tiene la ventaja de poder aplicar algoritmos estadísticos para determinar recomendaciones que se basan en recorrer grafos. Por ejemplo: Neo4J, Hyperbase- DB e InfoGrid.

- **Bases de datos orientadas a objetos** *OBJECT-ORIENTED DATABASES*

Trata de almacenar en la base de datos los objetos completos (estado *STATE* y comportamiento *BEHAVIOR*). La información que contienen se organiza en colecciones *COLLECTIONS* de objetos con sus **atributos/propiedades** *ATTRIBUTES/PROPERTIES* y el comportamiento en **operaciones/métodos** *OPERATIONS/METHODS*. Por ejemplo: db4o, Versant, Neodatis y Objectivity/DB.
Se verán el curso que viene en Acceso a Datos.
Para ampliar: http://en.wikipedia.org/wiki/Object_database

Para ampliar sobre bases de datos NoSQL:

http://www.dosideas.com/base-de-datos/657-nosql-el-movimiento-en-contra-de-las-bases-de-datos.html

http://rafinguer.blogspot.com.es/2009/11/movimiento-nosql-la-alternativa-las.html

http://google.dirson.com/post/1827-bigtable-sistema-almacenamiento-google/

http://blog.classora.com/2013/07/30/bases-de-datos-nosql-cassandra-vs-bigtable/

4.3. Tipos de bases de datos según su propósito

- **Bases de datos geográficas**

En ellas se almacenan mapas *MAPS* y símbolos *SYMBOLS* que representan superficies geográficas. Por ejemplo la empleada por Google Earth.

Para ampliar: http://en.wikipedia.org/wiki/Geographic_information_system

- **Bases de datos deductivas (lógicas)**

Es un sistema que almacena hechos *FACTS* y reglas *RULES* que permiten a través de procedimientos de inferencia *INFERENCE,* extraer nuevos hechos. Se basan en la lógica matemática y por eso también son denominadas bases de datos lógicas *LOGICAL*.

Para ampliar: http://html.rincondelvago.com/bases-de-datos-deductivas.html
http://en.wikipedia.org/wiki/Deductive_database

- **Las Big Data**

Big data es un término aplicado a conjuntos de datos que superan la capacidad del software habitual para ser capturados, gestionados y procesados en un tiempo razonable.

Para ampliar:

http://en.wikipedia.org/wiki/Big_data

http://mike2.openmethodology.org/wiki/Big_Data_Definition

5. LEY DE PROTECCION DE DATOS

Ya que las bases de datos tienen como propósito almacenar datos, para cerrar este tema no podemos olvidar citar la LOPD Ley Orgánica de Protección de Datos.

Su objetivo principal es regular el tratamiento de los datos y ficheros, de carácter personal, independientemente del soporte en el cual sean tratados, los derechos de los ciudadanos sobre ellos y las obligaciones de aquellos que los crean o tratan.

El órgano de control del cumplimiento de la normativa de protección de datos dentro del territorio español, con carácter general es la Agencia Española de Protección de Datos (AEPD).

Los datos personales se clasifican en función de su mayor o menor grado de sensibilidad, siendo los requisitos legales y de medidas de seguridad informáticas más estrictos en función del mayor grado de sensibilidad de los mismos.

Es obligatorio, en todo caso la declaración de los ficheros de protección de datos a la "Agencia Española de Protección de Datos".

Los interesados a los que se soliciten datos personales deberán ser previamente informados de modo expreso, preciso e inequívoco:

- De la existencia de un fichero o tratamiento de datos de carácter personal, de la finalidad de la recogida de éstos y de los destinatarios de la información.
- Del carácter obligatorio o facultativo de su respuesta a las preguntas que les sean planteadas.
- De las consecuencias de la obtención de los datos o de la negativa a suministrarlos.
- De la posibilidad de ejercitar los derechos de acceso, rectificación, cancelación y oposición.
- De la identidad y dirección del responsable del tratamiento o, en su caso, de su representante.

Se permite sin embargo, el tratamiento de datos de carácter personal sin haber sido recabados directamente del afectado o interesado, aunque no se exime de la obligación de informar de forma expresa, precisa e inequívoca, por parte del responsable del fichero o su representante, dentro de los **tres meses** siguientes al inicio del tratamiento de los datos.

Excepción: No será necesaria la comunicación en tres meses de dicha información si los datos han sido recogidos de "**fuentes accesibles al público**", y se destinan a la actividad de **publicidad** o prospección **comercial**, en este caso "en cada comunicación que se dirija al interesado se le informará del origen de los datos y de la identidad del responsable del tratamiento así como de los derechos que le asisten".

Las sanciones tienen una elevada cuantía, siendo España el país de la Unión Europea que tiene las sanciones más altas en materia de protección de datos. Dichas sanciones dependen de la infracción cometida, las sanciones se clasifican en leves, graves y muy graves.