LOS SISTEMAS DE ALMACENAMIENTO DE LA INFORMACIÓN

Autor: Manuel Antonio Montoro Navarro

Tabla de Contenidos

1.- Introducción

La informática se define como "*La disciplina que se ocupa del proceso automático de la información*". Por lo tanto, la información es la razón de ser de la tecnología informática y del ordenador, que en el fondo es sólo una máquina que procesa información.

Con la información se realizan dos tareas principales:

- **Procesamiento** Combinar información para obtener información resumida o derivada.
- **Almacenamiento** Guardar la información para poder utilizarla posteriormente.

En este tema se va a tratar el almacenamiento de la información, especialmente a largo plazo.

2.- Almacenamiento de la información

La información se almacena en dispositivos denominados *memorias* que permiten recuperarla cuando se le solicita.

Existen muchas tecnologías de construcción de las memorias, con características distintas, ventajas e inconvenientes, lo que hace que se convierta en una labor fundamental de diseño de un equipo de proceso de datos el elegir la memoria o memorias adecuadas para cada parte del equipo.

Las características comunes de todos los dispositivos de almacenamiento son:

- **Unidad de almacenamiento**

  Es la cantidad de información más pequeña que se puede leer o escribir en el dispositivo. Puede oscilar desde 1 byte hasta miles de bytes, dependiendo del dispositivo y la tecnología utilizada. Si la memoria fuera un archivador con cajones donde se almacena la información, la unidad de almacenamiento sería la capacidad de un cajón.

- **Tipo de acceso**

  Indica la forma en que se puede acceder a una unidad de almacenamiento determinada dentro del dispositivo. Existen dos tipos de acceso:

  - **Aleatorio** Se puede acceder directamente si sabemos cual es su dirección.
  - **Secuencial** Sólo se pueden acceder de una en una (en secuencia).

Independientemente de la tecnología, la memoria de un equipo se divide en dos niveles de almacenamiento: primario y secundario.

2.1.- Almacenamiento primario

Dispositivos de memoria que son directamente accesibles al procesador del ordenador. Está formado por la memoria RAM, memorias ROM o FLASH y la memoria caché (que son memorias RAM más pequeñas y rápidas que la RAM común). **Toda la información que deba ser accedida por el procesador debe estar almacenada en el almacenamiento primario.** Por lo tanto, si un dato que debe acceder el procesador no está en el almacenamiento primario hay que transferirlo o copiarlo desde el almacenamiento en el que esté situado hacia el almacenamiento principal (usualmente a la RAM).

Debido a su relativamente pequeño tamaño, la unidad de almacenamiento en la memoria primaria es de una palabra de entre 8 bits (1 byte) y 64 bits (8 bytes), según la arquitectura del procesador.

Los dispositivos de almacenamiento primario siempre utilizan el tipo de acceso aleatorio.

2.2.- Almacenamiento secundario

Son el resto de dispositivos de almacenamiento que no son directamente accesibles por el procesador. Usualmente su capacidad de almacenamiento es varios órdenes de magnitud mayor que la del almacenamiento primario y su velocidad de acceso varios órdenes de magnitud menor que la de éste. Los dispositivos de almacenamiento secundario suelen ser:

- Discos magnéticos (discos duros)
- Discos ópticos (CD / DVD / Bluray)
- Cintas magnéticas
- Discos de estado sólido (SSD)

En el caso del almacenamiento secundario y debido a su gran capacidad, la unidad mínima de almacenamiento no es una palabra ya que se necesitarían direcciones muy grandes. En su lugar se utilizan una unidad llamada *bloque* o *página*. Un bloque es un conjunto de bytes que puede oscilar entre los 128 bytes y los 4Kbytes.

El tipo de acceso puede ser aleatorio (discos en general) o secuencial (cintas).

3.- Sistemas de ficheros

A continuación se describirá de forma somera los conceptos relacionados con los ficheros.

No hay que confundir los ficheros que se van a estudiar en este tema con los archivos desde el punto de vista del sistema operativo del ordenador. El nombre es similar pero el concepto es distinto.

3.1.- Registros

La información se almacena en forma de *registros*.

Un registro es una colección de valores o datos que están relacionados entre si. Cada uno de los valores o datos se llama *campo*. Por ejemplo, un registro persona contendría campos como `nombre, apellidos, dni, fecha de nacimiento, teléfono, edad,` etc.

Cada campo tiene tres propiedades fundamentales:

- **Nombre**

  Sirve para identificar de forma inequívoca al campo dentro de un mismo registro, por lo que no puede estar repetido dentro de un registro.
- **Tipo**

  Determina el conjunto de los posibles valores que puede tener el campo (número entero, número con decimales, texto, fecha,...).
- **Tamaño**

  Indica el espacio de almacenamiento que ocupa el valor o dato. Puede ser fijo (el campo siempre ocupa el mismo espacio sea cual sea el valor que contenga el campo) o variable (el campo puede ocupar más o menos espacio según el valor que tenga, aunque se suele poner un límite máximo a la longitud del contenido).

La lista con los campos que tiene un registro, indicando las propiedades de cada uno, determina el formato o la *estructura* del registro.

Ejemplo: En nuestro ejemplo, podríamos definir el registro como sigue:

- **Registro**: persona
  - **Nombre campo:** nombre
    - **Tipo**: Texto
    - **Tamaño**: Fijo 30 caracteres.
  - **Nombre campo:** apellidos
    - **Tipo**: Texto
    - **Tamaño**: Fijo 50 caracteres.
  - **Nombre campo:** dni
    - **Tipo**: Texto
    - **Tamaño**: Fijo 9 caracteres.
  - **Nombre campo:** fecha-de-nacimiento
    - **Tipo**: Fecha
    - **Tamaño**: Fijo 8 caracteres
  - **Nombre campo:** telefono
    - **Tipo**: Texto
    - **Tamaño**: Fijo 9 caracteres
  - **Nombre campo:** edad
    - **Tipo**: Entero
    - **Tamaño**: Fijo 2 cifras

3.2.- Ficheros

Un *fichero informático* es un conjunto de registros, almacenado en algún sistema de almacenamiento, usualmente secundario.

Cada registro se almacena en uno o más bloques del dispositivo de almacenamiento. De cómo se realice esta asignación dependerá el rendimiento del sistema, como se analizará más adelante.

Los ficheros son la unidad básica de información que utilizan los programas. Dicho de otra forma, todos los datos acaban siendo almacenados en ficheros.

Las operaciones básicas que se pueden realizar sobre un fichero son:

- **Consulta** Acceso a la información que contiene un registro.
- **Inserción** Añadir un nuevo registro al fichero.
- **Modificación** Modificar el valor de uno o más campos de un registro existente en el fichero.
- **Borrado** Eliminar un registro del fichero.

El resto de operaciones sobre los ficheros se pueden realizar siempre utilizando estas operaciones básicas.

Los ficheros se clasifican en ficheros de registros de longitud fija o variable.

3.2.1.- Ficheros de registros de longitud fija

En este tipo de ficheros todos los registros son del mismo tipo y todos los campos tienen longitud fija. Por lo tanto ocupan el mismo espacio de almacenamiento.

Su principal ventaja es que son más fáciles de manipular puesto que para saber en qué posición del disco está un fichero sólo hay que multiplicar el número del registro por el tamaño.

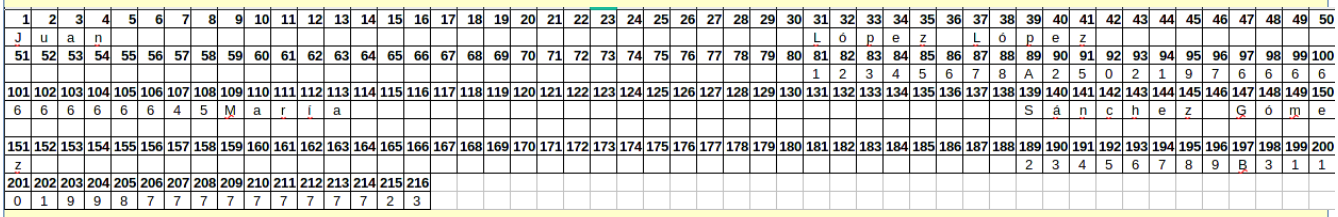
Como desventaja principal tienen que son poco flexibles y pueden llegar a desperdiciar espacio.

Ejemplo:

En apartados anteriores definimos el registro `persona` con longitud fija puesto que todos sus campos tienen longitud fija. Para representar los datos de las personas:

- Persona 1
  - **nombre**: Juan
  - **apellidos**: López López
  - **dni**: 12345678A
  - **fecha-de-nacimiento**:25/02/1976
  - **telefono**: 666666666
  - **edad**: 45
- Persona 2
  - **nombre**: Maria
  - **apellidos**: Sanchez Gómez
  - **dni**: 23456789B
  - **fecha-de-nacimiento**: 31/10/1998
  - **telefono**: 777777777
  - **edad**: 23

En un fichero de registros de longitud fija la información se almacenaría como muestra la imagen:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
J	u	a	n																											L	ó	p	e	z		L	ó	p	e	z									
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
																														1	2	3	4	5	6	7	8	A	2	5	0	2	1	9	7	6	6	6	6
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
6	6	6	6	6	6	4	5	M	a	r	í	a																										S	a	n	c	h	e	z		G	ó	m	e
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
z																																						2	3	4	5	6	7	8	9	B	3	1	1
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216																																		
0	1	9	9	8	7	7	7	7	7	7	7	7	7	2	3																																		

Como se puede ver, cada registro ocupa 30+50+9+8+9+2=108 bytes y por lo tanto 2 registros ocupan 2 * 108 = 216 bytes. Se puede comprobar la gran cantidad de espacio desperdiciado en los campos `nombre` y `apellidos`. El primer registro comienza en el byte número 1, el segundo en el 109 (1 + 108), el tercero en el 217 (1 + 2 *108), el cuarto en el 325 (1 + 2 *108), etc.

3.2.2.- Ficheros de registros de longitud variable

En este tipo de fichero los registros pueden ser de distintos tipos o bien ser todos del mismo tipo pero tener campos de longitud variable. Por lo tanto el espacio de almacenamiento que ocupa cada registro no es fijo sino que debe almacenarse como parte del registro.

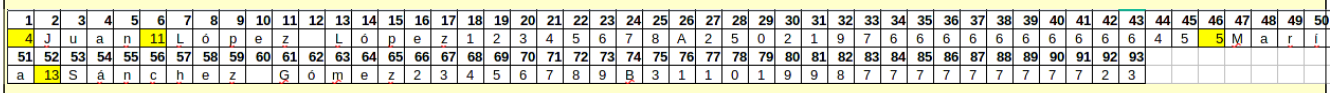
Su principal desventaja es que son más difíciles de manipular puesto que hay que realizar cálculos complejos o utilizar estructuras auxiliares (ver los índices más adelante) para poder manipularlos de forma eficiente.

Tienen a su favor que aprovechan el espacio de forma más eficiente.

Ejemplo:

En apartados anteriores definimos el registro `persona` de longitud fija. Supongamos ahora que los campos `nombre` y `apellidos` tienen longitud variable pero un tamaño máximo de 30 y 50, respectivamente. Para almacenar la longitud se colocará como primer valor de los campos `nombre` y `apellidos` la longitud que ocupan, a fin de saber donde terminan.

Ahora, la información se almacenaría de la siguiente manera:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
4	J	u	a	n	11	L	ó	p	e	z		L	ó	p	e	z	1	2	3	4	5	6	7	8	A	2	5	0	2	1	9	7	6	6	6	6	6	6	6	6	6	4	5	5	M	a	r	í
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93							
a	13	S	á	n	c	h	e	z		G	ó	m	e	z	2	3	4	5	6	7	8	9	B	3	1	1	0	1	9	9	8	7	7	7	7	7	7	7	7	2	3								

Los bytes con el fondo amarillo son los que almacenan las longitudes de los campos `nombre` y `apellidos`.

Como se puede ver, ahora los registros no ocupan siempre lo mismo. El primer registro ocupa 45 bytes mientras el segundo ocupa 48. Se ahorra mucho espacio (93 bytes frente a 216 que ocupaban en la versión con longitud fija, un ahorro del 57%) pero ahora la búsqueda de los registros es más difícil porque hay que ir leyendo los registros uno a uno y sumando las longitudes para poder determinar donde empieza o acaba cada uno. Se puede mejorar un poco el rendimiento si se incluye la longitud de cada registro completo al inicio pero esto añade algo más de espacio y de complejidad (si hay que cambiar el contenido de un campo de longitud variable lo más seguro es que haya que actualizar la longitud del registro). Además se complican las inserciones, modificaciones y borrados, como se verá más adelante.

3.2.3.- Organización del espacio del almacenamiento de los ficheros

Como se comentó anteriormente, una decisión importante que hay que tomar a la hora de trabajar con ficheros es decidir cómo se colocan los datos de cada registro en los bloques del dispositivo de almacenamiento durante su grabación.

Existen dos formas básicas de organización del espacio de los ficheros: secuencial y relativa.

Organización secuencial

En la organización secuencial se van almacenando los registros en bloques consecutivos en el mismo orden en que se van añadiendo al fichero. Un nuevo registro siempre se añade al último bloque del fichero si tiene espacio suficiente y si no, se escribe en el siguiente.

Ejemplo:

A un nuevo fichero secuencial se le añade el registro de clave 10.

El fichero quedaría: [10]

A continuación se añaden los registros con claves 27, 19 y 21.

El fichero quedaría ahora: [10][27][19][21]

La inserción de nuevos registros es muy eficiente puesto que únicamente hay que elegir el último bloque disponible.

La búsqueda es poco eficiente ya que si se desea localizar un registro determinado hay que ir consultando los registros uno a uno desde el inicio del fichero hasta que se localice el registro buscado. Como media habrá que recorrer la mitad de los registros para localizar el deseado y en el caso de que una búsqueda sea fallida (el registro que se busca no está en el fichero) hay que recorrer **todo** el fichero.

Ejemplo:

Si se desea localizar el registro con clave 19 habrá que ir al inicio del archivo e ir leyendo registros hasta que se localice el 19, lo que implicará leer (y descartar) los registros 10 y 27.

Si se busca el registro con clave 53 (que no se encuentra en el fichero) habrá que leer los cuatro registros antes de decidir que efectivamente, el registro no está almacenado.

El borrado de datos también es ineficiente puesto que al ir borrando registros van quedando "huecos" dentro del fichero. Este problema se puede paliar modificando el sistema de inserción para que "reutilice" huecos al insertar registros, siempre y cuando éstos ocupen una longitud menor o igual que la del hueco. Sin embargo este sistema complica el sistema de inserción, ralentizándolo. En algunos sistemas se realiza periódicamente una operación llamada "compactación" por lo que se crea un nuevo fichero a partir de uno antiguo, copiando los registros no borrados y reemplazando luego el fichero original. Esta operación toma tiempo y en muchos casos requiere una parada del sistema mientras se realiza, lo que la hace inapropiada para sistemas que deben funcionar de manera continua.

Ejemplo:

Si se elimina el registro con clave 19:	[10][27][ ][21]
Si ahora se añade un nuevo registro:	[10][27][ ][21][33]
Y tras la compactación:	[10][27][21][33]

La modificación puede ser simple, si los registros son de longitud fija, o más compleja, si los registros son de longitud variable. En este último caso, si el tamaño del registro modificado es mayor que el que tenía originalmente, no cabrá en el hueco en el que estaba almacenado. La solución es eliminar el registro antiguo y almacenar el registro con los cambios al final del fichero, como si se realizara una inserción.

Ejemplo:

Si los registros son de longitud fija y se modifica el registro de clave 21, el nuevo fichero quedaría: [10][27][21][33]

Si los registros son de longitud variable y el nuevo contenido del registro 21 tuviera un tamaño superior al del hueco que lo alberga, habría que realizar la modificación mediante borrado e inserción: [10][27][33][21]

Como se puede ver, la organización secuencial es simple de implementar pero no es muy eficiente. Aún así puede ser la adecuada para ficheros que se suelen procesar siempre de forma completa y en los que la principal operación de modificación es la inserción y sufre pocas modificaciones y borrados. En este caso puede ser una organización suficientemente eficiente. Por desgracia, los datos en sistemas reales no siempre se adaptan a estas restricciones, por lo que el sistema secuencial no será el más adecuado para ellas. No obstante, y a fin de mejorar su rendimiento, han surgido variaciones de la organización secuencial que palían los inconvenientes arriba comentados. Las más empleadas son:

- **Organización secuencial indexada**

  En esta organización los registros con los datos se graban en un fichero secuencialmente, pero se pueden recuperar con acceso directo gracias a la utilización de un fichero adicional, llamado índice, que contiene información de la posición que ocupa cada registro en el fichero de datos, solucionando así el principal problema de la organización secuencial pura.

  Los sistemas más potentes permiten tener más de un índice por fichero de forma que se pueden organizar búsquedas por distintos campos.

- **Organización secuencial encadenada**

  Esta organización permite tener los registros ordenados según un orden lógico diferente al orden físico en el que están grabados gracias a la utilización de unos campos adicionales llamados punteros, que indican la posición del siguiente registro.

  De esta forma, aunque los registros se vayan insertando siempre al final del fichero, los campos de posición permiten establecer otro orden a la hora de recorrer el fichero, lo cual puede ser importante si el acceso debe hacerse de forma secuencial pero en un orden distinto al de la inserción.

Organización ordenada o relativa

En este tipo de organización los registros se almacenan en orden según el valor de uno de sus campos, llamado *campo de ordenación*. Usualmente este campo también tiene un valor único para cada registro. En este caso se denomina *campo clave*.

Ejemplo:

A un nuevo fichero relativo se le añade el registro de clave 10.
El fichero quedaría: [10]

A continuación se añaden los registros con claves 27, 19 y 21.

El fichero quedaría ahora: [10][19][21][27]

La inserción de registros es poco eficiente porque hay que mantener la ordenación. En muchos casos esto exige, o bien mover registros hacia el final del archivo para hacer hueco, o bien utilizar ficheros de maniobra.

Las búsquedas, sin embargo, son muy eficientes si se realizan por el campo de ordenación ya que se puede realizar búsqueda binaria cuyo tiempo de búsqueda es logarítmico respecto al número total de registros. Si la búsqueda se hace por otros campos tiene la misma eficiencia que la organización secuencial simple.

Respecto a los borrados, su eficiencia es similar a la de los ficheros secuenciales.

La modificación no presenta problemas salvo que:

- Los registros sean de longitud variable y la longitud del nuevo contenido sea mayor que el del contenido actual. Esto exigirá mover el resto de registros hasta el final del fichero para hacer espacio, lo que puede ser muy costoso.
- Se modifique el contenido del campo de ordenación. Esto puede ocasionar que el registro deba ser movido, lo cual también es costoso.

3.2.4.- Métodos de acceso

Se refiere al procedimiento seguido para acceder a uno o más registros determinados de un fichero.

Una de las operaciones más costosas y frecuentes es la búsqueda de información. Para mejorar su eficiencia se utilizan *índices*, que son estructuras de datos que relacionan valores de un campo (normalmente el campo clave) de un registro con su dirección de memoria.

Son similares a los índices analíticos que vienen al final de los libros de texto o científicos en los que aparece una lista de la palabras o términos que aparecen en el libro junto con la primera página en la que aparece dicho término o con una lista de todas las páginas en las que aparece dicho término, según el tipo de índice que se desee.

Hay varios tipos de índices, que se discuten a continuación:

Índices Primarios

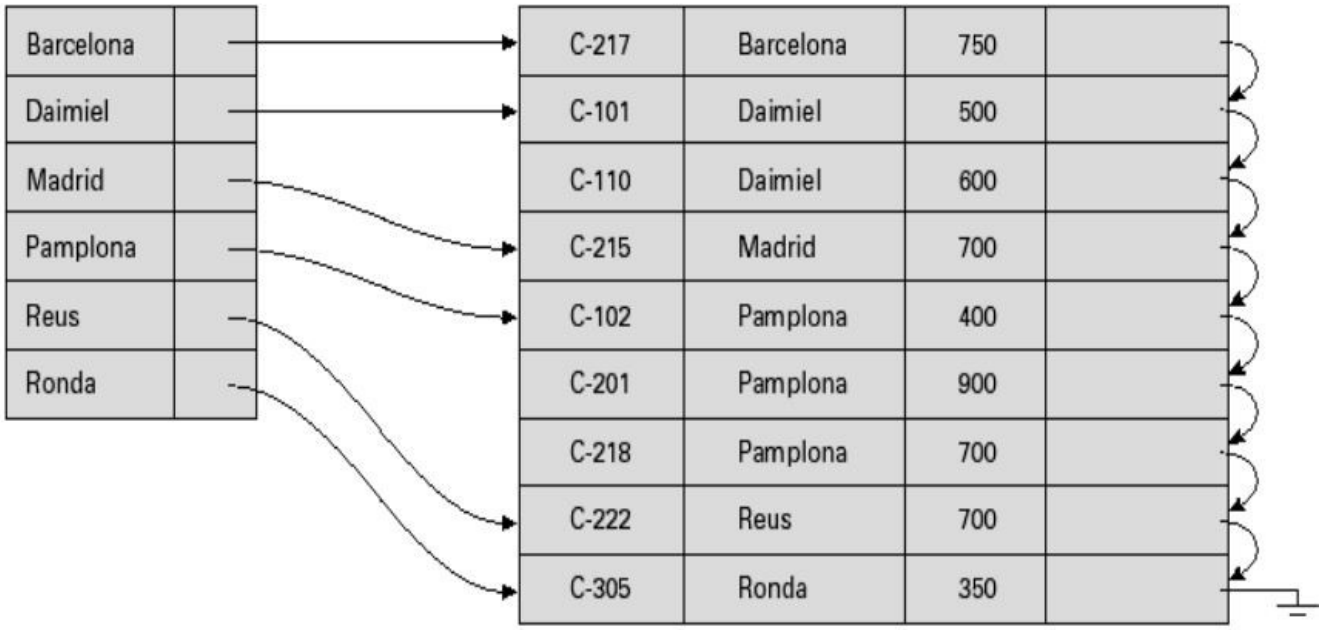
Un índice primario es un fichero ordenado que contiene registros de longitud fija con dos campos: uno es el campo clave del fichero de datos y el otro es un apuntador al bloque del fichero de datos donde se encuentra el registro correspondiente.

Son mucho más pequeños que los ficheros de datos.

Índices de agrupamiento

Cuando los registros de un fichero están ordenados por un campo no clave (que no tiene un valor único para cada registro), este campo se denomina campo de agrupamiento ya que el fichero contiene grupos de registros con el mismo valor del campo.

Los índices de agrupamiento son índices que se crean sobre estos campos. En este caso cada entrada del índice contiene un valor del campo de agrupamiento y el número de registro *del primer registro del grupo*. De esta forma se puede acceder rápidamente al primer registro del grupo y a partir de ahí recorrer el fichero de forma secuencial.



Índices secundarios

Los índices secundarios son iguales a los primarios o de agrupamiento pero cuando el campo sobre el que se indexa no es el campo por el que está ordenado el fichero.

4.- Sistemas de bases de datos

Antes de la aparición de las bases de datos en los años 60, la gestión de los datos se hacía a través de los **sistemas de ficheros**. Un sistema de ficheros es un conjunto de programas que presta servicio a los usuarios finales. Cada programa define y maneja sus propios datos. Los sistemas de ficheros surgieron al tratar de informatizar el manejo de los archivadores manuales con objeto de proporcionar un acceso más eficiente a los datos.

Los sistemas de ficheros presentan una serie de inconvenientes, primero respecto a los archivos, y segundo respecto a los datos.

Los problemas respecto a los ficheros se deben a la necesidad de controlar la integridad semántica, el control de autorizaciones y la concurrencia de accesos.

Los problemas respecto a los datos se deben a su estructura física, a su modo de estar almacenados en diferentes archivos. Destacan los siguientes problemas: redundancia de información, inconsistencia, fragmentación de la información y dificultad de acceso a los datos.

Vamos a ver algunos de estos problemas:

- **Separación y aislamiento de los datos**

  Cuando los datos están separados en distintos ficheros, es responsabilidad de la aplicación el coordinar el acceso a los distintos ficheros.

- **Duplicación de datos**

  Pueden existir ficheros distintos con copias de los mismos datos, lo que provoca problemas de consistencia de datos (si las copias no son idénticas) y de desperdicio de espacio (ya que hay varias copias de los mismos datos).

  Por ejemplo, puede ocurrir que el departamento de Contabilidad de una empresa tenga un fichero de empleados con unos datos y el departamento de Personal otro fichero de empleados con algunos datos iguales y otros distintos.

- **Dependencia de los datos de las aplicaciones**

  Ya que la definición y manipulación de los ficheros se realiza completamente por las aplicaciones, cualquier cambio en cualquiera de las dos exige la modificación de la aplicación.

- **Incompatibilidad de ficheros**

  Dado que la estructura de los ficheros está definida en la aplicación que los maneja, otra aplicación no puede acceder a los ficheros.

- **Consultas fijas**

Dado que la aplicación es la que conoce la estructura de los ficheros, cualquier consulta que se quiera realizar sobre ellos deberá estar codificada en la aplicación. Dicho de otra manera, no se puede realizar ninguna consulta que no esté codificada en la aplicación.

- **Control de concurrencia**

  El acceso de varias aplicaciones al mismo fichero a la vez es complicado ya que se pueden provocar inconsistencias al modificar una aplicación un fichero mientras otra lo lee.

- **Catálogo**

  Conforme el número de aplicaciones y ficheros crece en una organización se vuelve complicado el saber donde está almacenado un dato. La información está ahí pero no se sabe exactamente en qué fichero, ya que cada departamento tiene los suyos y los gestiona de forma separada a los demás departamentos.

Debido a estos problemas surgieron las bases de datos como un modelo de organización de los datos.

> *Una base de datos es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización.*

Una base de datos es un conjunto de datos *organizados* en estructuras que se definen una única vez y que se utilizan simultáneamente por varios equipos y usuarios.

En lugar de almacenarse en ficheros desconectados entre si y con información redundante, los datos de una base de datos están centralizados. La base de datos no pertenece a un equipo sino a *toda la organización*.

Además, la base de datos no sólo contiene la información sino que también almacena una descripción de dicha información. A esta descripción se la conoce como *metadatos* y se almacena en una estructura denominada *diccionario de datos* o *catálogo*.

Para poder utilizar estos datos, tenemos que hacer uso de los **Sistemas Gestores de Bases de Datos**, que son un software que sirve de interfaz entre la base de datos, los usuarios y las aplicaciones.

4.1.- Arquitectura de un sistema de bases de datos

En 1975, el comité ANSI (American National Standard Institute) propuso un estándar para la creación de sistemas de bases de datos basado en una arquitectura con tres capas o niveles.

El objetivo de esta arquitectura es el de separar los distintos niveles de abstracción desde los que se puede ver el esquema de una base de datos, abarcando desde la vista más concreta hasta la de más alto nivel. Los tres niveles o capas son:

- **Nivel interno**

Describe la estructura física de la base de datos mediante un esquema interno. Este esquema describe todos los detalles para el almacenamiento de la base de datos así como los métodos de acceso. Se habla de ficheros, discos, directorios, índices, etc.

- **Nivel medio**

  Describe la estructura de la base de datos completa para toda la organización mediante un esquema conceptual. Se ocultan los detalles de cómo se almacenan los datos y se centra en describir estos datos. En este nivel se habla de entidades, atributos, relaciones y restricciones.

- **Nivel externo**

  Se describen las vistas de usuario a través de esquemas externos. Estos utilizan un modelo conceptual para describir la parte de la base de datos que atañe a un usuario o grupo de usuarios, ocultando el resto. El usuario final percibe este esquema a través del uso de aplicaciones.

Hay que recordar que los tres niveles contienen descripciones *de los mismos datos,* pero en distintos niveles de abstracción. Los únicos datos que existen están en el nivel físico almacenados en un dispositivo de almacenamiento.

Esta arquitectura intenta obtener la *independencia de los datos*, que es la capacidad de un sistema de permitir cambios en un nivel sin que éste afecte a los niveles superiores. Se pueden definir dos tipos de independencia de datos:

- **Independencia lógica**

  Es la capacidad de modificar el esquema conceptual (nivel medio) sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema para ampliar la base de datos o para reducirla. Si se reduce, las aplicaciones que no se refieran a los datos eliminados no deberían verse afectadas.

- **Independencia física**

  Es la capacidad de modificar el esquema interno sin tener que alterar el resto de esquemas. Por ejemplo, puede ser necesario reorganizar los ficheros para mejorar el rendimiento o añadir/quitar índices. Es más fácil de conseguir que la independencia lógica.

4.2.- Modelos de datos

Una base de datos contendrá información (datos) correspondiente a "cosas" que existen en el mundo real (personas, objetos, situaciones, relaciones, etc.) pero, obviamente, no contendrá **toda** la información posible sobre estas "cosas". Por ejemplo, si en nuestra base de datos guardamos información sobre cada cliente de la empresa, sólo se guardará la información sobre los mismos que sea pertinente o útil para nuestra organización, descartándose el resto de información por irrelevante

(nombre y apellidos, dirección, DNI podrían ser datos necesarios pero color de pelo o de ojos, número de hijos o estado civil parece que no lo sería).

La descripción simplificada de una parte del mundo real se denomina *modelo* y es un concepto ampliamente utilizado en muchos campos del conocimiento. Un modelo consiste en una descripción simplificada de una parte de la realidad a fin de poder ser manipulado más fácilmente. Por ejemplo, si se quiere saber cómo se comportará un diseño de avión ante las corrientes de aire no se construye el avión completo (con motores, asientos, baños, etc.) y se prueba en vuelo sino que se crea una maqueta a escala con la forma externa más parecida posible al avión real pero sin sus "tripas" y se prueba con ventiladores en lugar de lanzarlo al aire. Esta maqueta es un modelo del avión real en el que se han descartado un montón de cosas (motores, depósitos de combustible, sistemas de navegación, etc.) y se ha centrado en la forma externa. Obviamente no es un avión real pero es perfectamente válido para estudiar su comportamiento aerodinámico.

Un modelo de datos funciona de forma similar. Es una representación de la información de una parte de la realidad que contiene sólo la información que se considera relevante para el propósito para el que se construye el modelo. La información irrelevante no aparece en el mismo.

Un modelo de datos nos permite manipular la información de forma que se compruebe si es correcta, completa, consistente, etc. sin necesidad de complicaciones innecesarias por información que en realidad no nos importa.

Una definición más formal sería:

> *Un modelo de datos es una colección de herramientas conceptuales (mentales) que describen los datos y las relaciones que existen entre los mismos, así como sus restricciones.*

Las herramientas conceptuales permiten analizar o representar una realidad compleja por composición.

Las relaciones que existen entre los datos es una información importante. Por ejemplo, un nombre o DNI sueltos dan poca información pero juntos nos indican que una persona con el nombre dado tiene un DNI con el número que se proporciona.

Las restricciones sobre los datos son reglas que se aplican a los mismos y limitan o definen los mismos. Por ejemplo, un DNI debe constar de ocho números y una letra. Cualquier otra combinación de letras y números no es un DNI válido. Además, mediante un algoritmo ya definido, la letra está relacionada con los números de forma que para una combinación dada de 8 números sólo le corresponde una letra determinada y ninguna otra.

4.3.- Tipos de modelos

Dado que la arquitectura de la base de datos es de tres niveles (externo, medio e interno), hay que modelar los datos en cada nivel. Por lo tanto, existen tres tipos de modelos:

- Modelos conceptuales
- Modelos lógicos
- Modelos físicos

4.3.1.- Modelos conceptuales

Se utilizan para describir los datos de una forma parecida a como los manipulamos las personas, aunque formalizando la descripción a fin de que sea precisa y sin ambigüedades.

Son modelos muy flexibles y permiten expresar datos, relaciones y restricciones explícitamente.

Existen muchos modelos pero el más extendido y utilizado por su sencillez y eficiencia es el *Modelo Entidad-Relación* o MER.

El MER representa la realidad a través de *entidades*, que son "cosas" que existen de forma independiente y que se distinguen de "cosas" similares por sus características. Por ejemplo, si estamos creando un modelo de la información que se gestiona en el departamento de nóminas de una empresa tendríamos, con casi total seguridad, una entidad denominada **`EMPLEADO`** que contendría las características relevantes para nosotros que tiene cada una de las personas que trabaja para la empresa tales como `DNI, nombre y apellidos, dirección, salario, número de hijos`,...

A estas características de las entidades se les denomina *atributos*. Un atributo de una entidad es un dato que tienen todas las ocurrencias de dicha entidad.

A cada una de las distintas ocurrencias de una entidad se le denomina *instancia* de esa entidad. Por ejemplo, Juan Antonio Pérez, con DNI 88888888A, que vive en la Avenida de los Tejos, 16, cobra 1500 euros al mes y no tiene hijos sería una instancia de la entidad `EMPLEADO` y Manuel Sánchez Carvajal, con DNI 99999999Z, dirección en Calle Ajoporro, 23, sueldo 1456 euros mensuales y con un hijo sería otra. Son dos empleados distintos pero tienen características comunes, aunque el valor concreto de las mismas sean distintos. Hay que hacer hincapié aquí en que lo importante es que los dos empleados tienen las mismas características (por ejemplo, dirección) aunque el valor concreto de cada una será posiblemente distinto para cada uno (aunque podría ser igual, por ejemplo, dos empleados podrían ser padre e hijo y viven en el mismo domicilio).

Asimismo el modelo también describe las relaciones entre las entidades. Estas relaciones describen vínculos entre instancias de dos o más entidades. Por ejemplo, la entidad **`EMPLEADO`** y la entidad **`SUCURSAL`** podrían estar vinculadas con la relación **`Trabaja en`** que informa de en cual oficina trabaja cada empleado, o dicho de otra forma, qué empleados trabajan en una oficina determinada.

Más adelante estudiaremos este modelo con más detalle.

4.3.2.- Modelos lógicos

Los modelos lógicos utilizan una descripción más formal para describir los datos de forma que se puedan realizar operaciones sobre los mismos de forma no ambigua o mecánica. Dicho de otra

forma, son modelos más cercanos a la forma de trabajar de la máquina que los modelos conceptuales, que están más cercanos a la forma de trabajar de las personas. Los modelos lógicos han sufrido una evolución más patente a lo largo del tiempo conforme la tecnología evolucionaba y las bases de datos se hacían más grandes y complejas.

Modelo Jerárquico

En el modelo jerárquico las únicas relaciones entre entidades que se permiten son de padres a hijos. Una instancia de una entidad (llamada entidad padre) puede estar relacionada con una o más instancias de otra (entidad hija). Es fácil de programar pero tiene el inconveniente de que es muy rígida y hay muchas relaciones que no cumplen esta restricción. Estas relaciones deben representarse utilizando "trucos" tales como duplicar la información, ocasionando problemas de consistencia.

Modelo en red

Más avanzado que el jerárquico permite que una entidad tenga más de un padre, eliminando la restricción que tenía el jerárquico y mejorando la consistencia. Aún así es difícil de administrar.

Modelo relacional

En este modelo los datos y las relaciones se representan utilizando el concepto de tabla, lo cual hace que la manipulación sea uniforme.

Modelo orientado a objetos

En estos modelos se intenta almacenar el comportamiento de las entidades además de su información. Además permiten establecer relaciones entre los datos que no se pueden hacer fácilmente con los otros modelos, tales como herencia. A pesar de ser más avanzadas que las anteriores, su uso no se ha extendido mucho por la falta de un estándar, por lo que el mercado está muy fragmentado y cada sistema es distinto completamente a los demás.

Modelo declarativo

En estos modelos los datos se almacenan como datos y reglas. Permiten consultas muy complejas. Son utilizados principalmente en campos como los sistemas expertos e Inteligencia Artificial.

4.3.3.- Modelos físicos

Los modelos físicos de datos describen las estructuras de almacenamiento concretas en el sistema de gestión de bases de datos, indicando qué se almacena, donde y cómo.

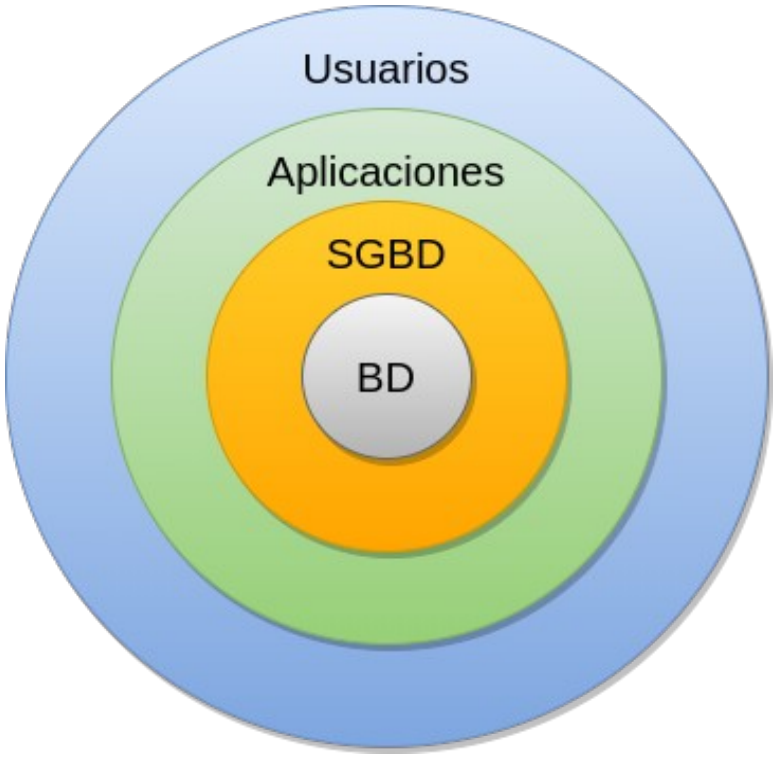
El modelo físico depende fuertemente del sistema de gestión de bases de datos empleado, por lo que no se puede dar una descripción general del mismo, debiendo referirse a la documentación del sistema de gestión de bases de datos concreto elegido.

5.- Sistemas Gestores de Bases de Datos

Un Sistema de Gestión de Bases de Datos (SGBD) es una aplicación o conjunto de aplicaciones que permite a los usuarios definir, crear y mantener bases de datos así como proporcionar acceso a las mismas. Es un sistema intermediario entre el usuario y las bases de datos.

El funcionamiento de un sistema de información con SGBDs sería el siguiente:

- En el nivel más externo están los usuarios, que son los productores y consumidores finales de la información del sistema. Hay que tener en cuenta, sin embargo, que los usuarios no tienen porque ser personas. Pueden ser otros sistemas externos que interactúan con el nuestro consumiendo y produciendo información (por ejemplo una cámara que reconoce matrículas en la entrada o salida de un parking).
- Los usuarios interactúan con aplicaciones que ofrecen una vista coherente de los datos y facilitan el manejo de los mismos, proporcionando un interfaz más amigable (formularios, ventanas de edición, informes, etc.)



- Las aplicaciones se comunican con el SGBD para solicitar datos o realizar altas o modificaciones sobre los mismos.
- Los SGBDs son los encargados de mantener las bases de datos físicas en las que se almacena la información y desde donde se recupera cuando sea necesario.

5.1.- Objetivos de un SGBD

Los objetivos principales de un SGBD son los siguientes:

- **Abstracción de la Información:** el sistema ahorra al usuario la necesidad de conocer los detalles de cómo se almacenan los datos, así como de su mantenimiento. Para ocultar estos detalles, se definen varios niveles de abstracción.
- **Independencia:** capacidad para modificar un esquema de definición sin afectar a los programas de aplicación. Existen dos niveles:
  - Independencia física: cuando es posible modificar el esquema físico sin que afecte a las aplicaciones. Se realizan para mejorar el rendimiento.
  - Independencia lógica: cuando es posible modificar el esquema conceptual sin obligar a escribir de nuevo las aplicaciones. Se realizan cuando cambia la estructura lógica de la BD.
- **Redundancia mínima:** evitar la repetición o redundancia de los datos en múltiples ficheros.
- **Consistencia:** si existen datos duplicados en ficheros y se realiza una actualización de ese dato, será necesario que el SGBD garantice la adecuada actualización del dato en todos los ficheros para evitar la inconsistencia del dato.
- **Seguridad:** protección de los datos frente al acceso accidental o intencionado por parte de individuos no autorizados.
- **Integridad:** medidas necesarias para conservar la corrección de los datos en la BD. Existen circunstancias que pueden hacer que los datos se estropeen:
  - Fallos del Hardware.
  - Actualizaciones incompletas.
  - Inserción de datos no válidos.
- **Respaldo y recuperación:** el SGBD debe proporcionar mecanismos eficientes para conservar copias de seguridad de cada fichero en prevención de posibles fallos. Las copias de seguridad deben realizarse regularmente y guardarse en un lugar seguro. Con respecto a las recuperaciones, para poder hacerlas con la máxima rapidez y eficacia se recurre a un fichero especial llamado Bitácora o Diario (Archivo Log), donde se registran todos los datos que se vayan modificando debido a las operaciones de la BD. En él se guarda tanto el valor que tenía el dato antes de ser modificado como el valor después de la modificación.
- **Control de la concurrencia:** dado que lo más habitual es que una BD trabaje en un entorno multiprogramación y multiusuario, es necesario controlar el que transacciones distintas accedan al mismo dato de forma simultánea para evitar la inconsistencia de datos.
- **Tiempo de respuesta:** se debe asegurar un tiempo de respuesta idóneo a las peticiones de usuario.

5.2.- Funciones de un SGBD

Para poder cumplir los objetivos anteriormente reseñados, un SGBD debe realizar las siguientes funciones:

- **Gestionar el diccionario de datos**

  Un diccionario de datos es un conjunto de archivos que mantiene el SGBD automáticamente y que contiene información sobre los datos que se almacenan en la misma. A esta información también se la conoce como *metadatos*. Aunque la información exacta almacenada en el diccionario depende del SGBD, de forma general, éste contiene lo siguiente:

  - Nombre, tipo y tamaño de cada dato.
  - Relaciones entre los datos.
  - Reglas de integridad sobre los datos.
  - Usuarios con autorización para realizar operaciones sobre los datos, indicando qué usuario puede realizar qué operación sobre qué dato.
  - Estadísticas varias de uso.

- **Garantizar la integridad transaccional**

  Se debe garantizar que todas las actualizaciones correspondientes a una transacción se llevan a cabo o bien que ninguna lo hace. Existen varias técnicas para resolver esto, siendo la más empleada el uso de *diarios* (logs).

- **Gestionar el acceso concurrente a los datos**

  Cuando se permite el acceso simultáneo de varios usuarios a los mismos datos se pueden producir problemas de consistencia de los datos si un usuario está escribiendo unos datos que otro usuario está leyendo en ese mismo instante. El sistema debe garantizar que los datos que accede cada usuario son consistentes. Las técnicas más empleadas para asegurar esto es el uso de *bloqueos* (locks).

- **Recuperar datos**

  Cuando se produce un fallo (humano o de los sistemas) es importante que la base de datos se recupere a un estado *consistente* lo más cerca posible del momento del fallo. Es inevitable que ocurra cierta pérdida de datos pero hay que minimizarla todo lo posible.

- **Proporcionar interfaces de uso**

  El SGBD debe ser accesible desde el exterior (si no, ¿para que sirve?). Para ello debe proporcionar canales o accesos por los que conectar con él (red, acceso local, memoria compartida, etc.)

- **Gestionar las restricciones sobre los datos**

  Las restricciones sobre los datos son reglas que estos deben cumplir en su estado consistente. El sistema debe garantizar que esto es así, evitando, por ejemplo, la modificación de un dato si dicha modificación viola una o más reglas de integridad.

- **Proporcionar herramientas de administración**

  El SGBD debería proporcionar herramientas para facilitar la administración y uso del mismo. A veces estas herramientas no se proporcionan directamente por el fabricante sino por terceras personas.

5.3.- Componentes de un SGBD

Aunque cada sistema es distinto, de forma general todos los SGBDs incluyen los siguientes componentes:

- **Lenguajes de datos**

  Se utilizan para dar instrucciones al SGBD.

  Normalmente se distinguen tres tipos de lenguajes según la funcionalidad que ofrecen:

  - **Lenguaje de Definición de Datos (Data Definition Language, DDL)**

    Este lenguaje se utiliza para manipular las definiciones de los objetos de la base de datos así como de su estructura, relaciones y restricciones.

  - **Lenguaje de Control de Datos (Data Control Language, DCL)**

    Este lenguaje se utiliza para manipular la seguridad de los datos (usuarios, grupos, privilegios, etc.)

  - **Lenguaje de Manipulación de Datos (Data Manipulation Language, DML)**

    Este lenguaje se utiliza para manipular el *contenido* de la base de datos. Permite consultar los datos así como crear, modificar y eliminar datos.

- **Diccionario de datos**

  Usualmente se implementa como otra base de datos cualquiera del sistema aunque se manipule de forma distinta.

- **Herramientas**

  Ofrecen un medio de administrar y gestionar el SGBD.

- **Optimizador de consultas**

  Traduce el DML a las operaciones básicas a realizar sobre el modelo físico (búsquedas, consulta de índices, etc.). Además procura realizar la traducción de forma que la consulta sea lo más eficiente posible sin que el usuario deba preocuparse por ello.

- **Gestor de transacciones**

  Se ocupa de gestionar las transacciones y el acceso concurrente para asegurar la consistencia de la base de datos.
- **Planificador**

  En algunos SGBDs se ocupa de lanzar tareas que se deben iniciar de forma automática, ya sea en respuesta a determinadas condiciones de funcionamiento o a una planificación temporal.
- **Gestión de replicación**

  Algunos SGBDs proporcionan mecanismos para realizar copias offline de los datos, ya sea a otros SGBDs de reserva o a sistemas de copias de seguridad (backups).

5.4.- Usuarios de los SGBDs

Generalmente se distinguen cuatro grupos de usuarios que utilizan los SGBDs. Estos son:

- **Administradores**

  Se ocupan del mantenimiento del sistema completo (instalación, puesta en marcha, gestión de copias, gestión de almacenamiento, etc.). Son los usuarios con más "poder" del sistema.
- **Diseñadores**

  Realizan el diseño de la base de datos y se encargan de crear los modelos correspondientes y a implementarlos en el SGBD, a veces en colaboración con los administradores.
- **Programadores**

  Realizan programas que interactúan con los datos almacenados en el SGBD a través de éste. Suelen tener acceso completo a los datos aunque no se les suele permitir el modificar la estructura de los mismos.
- **Usuarios finales**

  Son clientes de las bases de datos que las usan sin necesitar tener conocimiento alguno sobre su funcionamiento, ubicación, etc.

6.- SGBDs propietarios y libres

Aunque es un movimiento que existe desde antes, con la popularización de Internet ha proliferado el software libre como alternativa al software comercial cerrado que existía antes.

El mundo de los SGBDs no ha sido refractario a estos cambios que han ocasionado que actualmente se pueda disponer de una amplia paleta de SGBDs tanto propietarios como libres. Este es un factor como otro cualquiera que hay que tomar en cuenta a la hora de seleccionar el SGBD a utilizar para una organización o departamento.

Entre los factores a tener en cuenta están:

- **Precio**

  Usualmente el software libre es gratuito mientras el propietario no lo es.
- **Funcionalidad**

  El software propietario suele ofrecer mejores funcionalidades que el gratuito.
- **Facilidad de uso**

  El software propietario suele ofrecer mejores o más sencillas herramientas de administración, uso, etc.
- **Soporte**

  El software comercial suele ofrecer mejor soporte aunque bastantes de software libre también ofrecen servicio de soporte (normalmente no gratuito).
- **Comunidad**

  Los proyectos de software libre suelen ofrecer una buena comunidad de usuarios dispuestos a ayudar ante los problemas.
- **Control**

  Con el software libre el control sobre el SGBD es total ya que nos permite seguir adelante aunque el fabricante del SGBD cese el desarrollo y fabricación del mismo o el soporte.

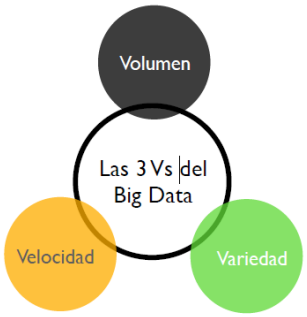
7.- Big Data

En esta nueva era tecnológica en la que nos hayamos inmersos, a diario se generan enormes cantidades de datos, del orden de petabytes. Hoy en día, cualquier dispositivo como puede ser un reloj, un coche, un *smartphone*, etc., está conectado a Internet generando, enviando y recibiendo una gran cantidad de datos. Tanto es así, que se estima que el 90% de los datos disponibles en el mundo ha sido generado en los últimos años.

Esta realidad descrita anteriormente demanda la capacidad de enviar y recibir datos e información a gran velocidad, así como la capacidad de almacenar tal cantidad de datos y procesarlos en tiempo real. Así pues, la gran cantidad de datos disponibles junto con las herramientas, tanto hardware como software, que existen a disposición para analizarlos se conoce como **big data**. No existe una definición precisa del término big data, ni tampoco un término en castellano que permita denominar este concepto. A veces se usan en castellano los términos datos masivos o grandes volúmenes de datos para hacer referencia al big data. Por este motivo, a menudo el concepto de big data es definido en función de las características que poseen los datos y los procesos que forman parte de este nuevo paradigma de computación. Esto es lo que se conoce como las Vs del big data.

Algunos autores coinciden en que big data son datos cuyo **volumen** es demasiado grande como para procesarlos con las tecnologías y técnicas tradicionales, requiriendo nuevas arquitecturas

hardware, modelos de programación y algoritmos para su procesamiento. Además, se trata de datos que se presentan en una gran **variedad** de estructuras y formatos: datos sintéticos, provenientes de sensores, numéricos, textuales, imágenes, audio, vídeo... Finalmente, se trata de datos que requieren ser procesados a gran **velocidad** para poder extraer valor y conocimiento de ellos. Esta concepción se conoce como las **tres Vs del big data**.



Otros autores amplían las características que han de tener los datos que forman parte del big data, incluyendo “otras Vs” como lo son: **volatilidad**, referida al tiempo durante el cual los datos recogidos son válidos y a durante cuánto tiempo deberán ser almacenados. **Valor**, referido a la utilidad de los datos obtenidos para extraer conocimiento y tomar decisiones a partir de ellos. **Validez**, referida a lo precisos que son los datos para el uso que se pretende darles. El uso de datos validados permitirá ahorrar tiempo en etapas como la limpieza y el preprocesamiento de los datos. **Veracidad**, relacionada con la confiabilidad del origen del cual provienen los datos con los que se trabajará así como la incertidumbre o el ruido que pudiera existir en ellos. **Variabilidad**, frente a la variedad de estructuras y formatos, hace referencia a la complejidad del conjunto de datos, es decir, al número de variables que contiene. Estas características, unidas a las tres Vs descritas anteriormente, se conocen como las **ocho Vs del big data**.

Dado que no existe una definición uniforme para el término big data, muchos autores definen el término en función de aquellas características que consideran más relevantes, por lo que es común encontrar “las cinco Vs del big data”, “las siete Vs del big data” o “las diez Vs del big data” según cada autor, apareciendo distintos términos para describir el big data, como también pueden ser visualización o vulnerabilidad, entre otros.

Son muchas las **soluciones a nivel hardware y software**, que se han propuesto a los problemas derivados del almacenamiento y el procesamiento de big data. A continuación, se describen los fundamentos de tres de ellas

7.1.- Almacenes de datos

Las bases de datos tradicionales, siguiendo la definición anterior, están basadas generalmente en **sistemas relacionales u objeto-relacionales**. Para el acceso, procesamiento y recuperación de los datos, se sigue el modelo ***Online Transaction Processing* (OLTP)**. El modelo OLTP, traducido al castellano como **procesamiento de transacciones en línea**, permite gestionar los cambios de la base de datos mediante la inserción, actualización y eliminación de información de la misma a través de transacciones básicas que son procesadas en tiempos muy pequeños. Con respecto a la recuperación de información de la base de datos, se utilizan operadores clásicos (concatenación, proyección, selección, agrupamiento...) para realizar consultas básicas y sencillas (realizadas, mayoritariamente, en lenguaje SQL y extensiones del mismo). Finalmente, las opciones de visualización de los datos recuperados son limitadas, mostrándose fundamentalmente los resultados de forma tabular y requiriendo un procesamiento adicional y más complejo en caso de querer presentar datos complejos.

La revolución en la generación, almacenamiento y procesamiento de los datos, así como la irrupción del big data, han puesto a prueba el modelo de funcionamiento, rendimiento y escalabilidad de las bases de datos relacionales tradicionales. En la actualidad, se requiere de soluciones integradas que aúnen datos y tecnología para almacenar y procesar grandes cantidades de datos con diferentes estructuras y formatos con el objetivo de facilitar la consulta, el análisis y la toma de decisiones sobre los mismos. En este sentido, la **inteligencia de negocio**, más conocida por el término inglés ***business intelligence***, investiga en el diseño y desarrollo de este tipo de soluciones.

Estas nuevas soluciones requerirán un modelo de procesamiento diferente a OLTP. Esto es así, ya que el objetivo perseguido por la inteligencia de negocio está menos orientado al ámbito transaccional y más enfocado al ámbito analítico. Las nuevas soluciones utilizan el modelo ***Online analytical processing*** (OLAP). La principal diferencia entre OLTP y OLAP estriba en que mientras que el primero es un sistema de procesamiento de transacciones en línea, el segundo es un sistema de **recuperación y análisis** de datos en línea. Por tanto, OLAP complementa a SQL aportando la capacidad de analizar datos desde distintas variables y dimensiones, mejorando el proceso de toma de decisiones.

Un **almacén de datos**, más conocido por el término ***data warehouse,*** es una solución de *business intelligence* que combina tecnologías y componentes con el objetivo de ayudar al uso estratégico de los datos por parte de una organización. Esta solución debe proveer a la empresa, de forma integrada, de capacidad de almacenamiento de una gran cantidad de datos así como de herramientas de análisis de los mismos que, frente al procesamiento de transacciones, permita transformar los datos en información para ponerla a disposición de la organización y optimizar el proceso de toma de decisiones.

7.2.- Bases de datos documentales u orientadas a objetos

La gran variedad y heterogeneidad en los tipos de datos almacenados y procesados en los últimos años ha puesto sobre la mesa la cuestión de si las bases de datos relacionales son el modelo más óptimo para trabajar con según qué tipos de datos. Como alternativa a ellas, en los últimos años han proliferado las bases de datos NoSQL.

NoSQL es el término utilizado para referirse a un tipo de bases de datos que permiten almacenar y gestionar tipos de datos que tradicionalmente han sido difíciles de gestionar por parte de las bases de datos relacionales. Así pues, NoSQL hace referencia a bases de datos documentales, bases de datos orientadas a grafos, buscadores, etc. En contraposición a las bases de datos relacionales, las NoSQL se caracterizan principalmente por:

- Independencia del esquema.
- No relacionales.
- Distribuidas.

Las **bases de datos documentales** trabajan con documentos, entendidos como una estructura jerárquica de datos que, a su vez, puede contener subestructuras. En una primera aproximación, es fácil pensar en que estos documentos pueden ser documentos de Microsoft Word, documentos pdf, páginas web... Las bases de datos documentales pueden, efectivamente, trabajar con estos tipos de documentos. Sin embargo, el término documento en este contexto posee un mayor nivel de abstracción.

Los **documentos** pueden consistir en datos binarios o texto plano. Es posible que se traten de datos semiestructurados, cuando aparecen en formatos como **JavaScript Object Notation (JSON)** o **Extensible Markup Language (XML)**. Por último, también pueden ser datos estructurados conforme a un modelo de datos particular como, por ejemplo, **XML Schema Definition (XSD)**.

Actualmente, **XML** y **JSON** son los formatos de intercambio de datos más utilizados en el desarrollo de aplicaciones web.

7.3.- Bases de datos sobre grafos

En los últimos años han aparecido nuevos tipos de datos como aquellos que vienen dados en forma de grafo. Algunos de estos datos son los provenientes de interacciones en redes sociales, datos geográficos expresados en forma de mapas, etc.

Una **base de datos orientada a grafos** es, por tanto, un sistema de bases de datos que implementa métodos de creación, lectura, actualización y eliminación de datos en un modelo expresado en forma de grafo. Existen dos aspectos fundamentales en este tipo de sistemas: almacenamiento y procesamiento de los datos.

Almacenamiento de los datos

En una base de datos orientada a grafos, los datos pueden almacenarse siguiendo el modelo relacional, lo que implica mapear la estructura del grafo a una estructura relacional, o bien, almacenarse de forma **nativa** utilizando modelos de datos propios para almacenar estructuras de tipo grafo. La ventaja de mapear los grafos a una estructura relacional radica en que la gestión y consulta de los datos se realizará de forma tradicional a través de un *backend* conocido como, por ejemplo, MySQL. Por su parte, la ventaja del almacenamiento nativo de grafos radica es que existen modelos de datos e implementaciones que aseguran y garantizan el buen rendimiento y la escalabilidad del sistema.

Procesamiento de los datos

En este sentido, también es posible distinguir el procesamiento no nativo de los grafos, lo cual implica el tratamiento de estos datos siguiendo las técnicas tradicionales utilizadas en los lenguajes de modificación de datos de las bases de datos relacionales, o el procesamiento nativo de los datos de grafos, el cual es beneficioso porque optimiza los recorridos del grafo cuando se realizan consultas aunque, en ocasiones, invierta demasiado tiempo y memoria en consultas que no requieren de recorridos complejos.

La motivación para requerir de sistemas de bases de datos específicos, orientados a trabajar con este tipo de datos, radica en tres aspectos principales:

- **Rendimiento**: Los sistemas de bases de datos orientados a grafos optimizan el rendimiento de las consultas sobre este tipo de datos con respecto a las bases de datos relacionales.
- **Flexibilidad**: Los sistemas de bases de datos orientados a grafos son más flexibles que los sistemas de bases de datos relacionales.
- **Agilidad**.

Existen distintos lenguajes de marcado de grafos, a partir de los cuales es posible generar archivos con formato de grafo para ser tratados por una base de datos de este tipo. Algunos de los lenguajes más utilizados son ***GraphML***, ***eXtensible Graph Markup and Modeling Language*** (**XGMML**), ***Graph Exchange Language*** (**GXL**) o ***Graph Modelling Language*** (**GML**), entre otros. La mayoría de ellos son variantes o extensiones del lenguaje **XML** para el modelado de grafos.

Los sistemas de bases de datos orientados a grafos han proliferado mucho en los últimos años, existiendo numerosas tecnologías que dan soporte a ellos, como pueden ser ***Microsoft Infinite Graph***, ***Titan***, ***OrientDB***, ***FlockDB***, ***AllegroGraph*** o ***Neo4j***

8.- Resumen

En este tema se han descrito los problemas del almacenamiento de datos de forma permanente y a gran escala. Desde los sistemas de almacenamiento hasta los sistemas de ficheros, enumerando las ventajas e inconvenientes de cada implementación.

A continuación se han descrito las bases de datos, como solución a la disparidad de sistemas de almacenamiento y a los problemas de inconsistencias y redundancias en la información perteneciente a una organización.

Por último se han descrito los SGBD como el software que hace posible el manejo de estas bases de datos y que proporciona servicios adicionales como transacciones o acceso concurrente a los datos.