Los Sistemas de Almacenamiento de la Información

Tabla de Contenidos

1.- Introducción

La mejor manera de entender la existencia del módulo de bases de datos en el currículo del ciclo formativo es reflexionando sobre dónde y de qué manera se almacenan y gestionan los datos que utilizamos diariamente. Muchas de nuestras acciones cotidianas, sean o no laborales, están relacionadas con las bases datos y los datos que ellas contienen. Piensa por ejemplo, en las siguientes situaciones:

- Utilizar la agenda del móvil para realizar una llamada.
- Pagar con tarjeta en un establecimiento.
- Solicitar una cita médica.
- Participar en la admisión para estudiar un ciclo formativo.
- Utilizar el GPS.

Casi todo lo que nos rodea está relacionado en alguna medida con los datos, su almacenamiento y gestión. Además, el gran volumen de datos que se maneja en la mayoría de los casos hace necesaria la existencia de técnicos perfectamente formados y capaces de trabajar con ellos.

Comenzaremos la unidad haciendo una pequeña aproximación histórica sobre cómo ha evolucionado el tratamiento de los datos.

En la década de los setenta los procesos básicos que las empresas realizaban para el tratamiento de la información se centraban en cuestiones relacionadas con contabilidad y facturación. Las necesidades de almacenamiento y gestión de información podían satisfacerse utilizando un número relativamente reducido de archivos en papel agrupados y ordenados, los típicos ficheros clásicos.

Este sistema podía ser útil cuando el volumen de datos manejado no era muy grande y se podía extraer la información que se necesitaba con cierta facilidad. Pero a medida que el archivo manual aumentaba y que la información que se necesitaba era más compleja, fue necesario sustituir este sistema por otro informatizado. En ese momento, la informática adaptó sus herramientas para que los elementos que el usuario manejaba en el ordenador se pareciesen a los que utilizaba manualmente. Así en informática se sigue hablado de ficheros, formularios, carpetas, directorios,…

La información debía ser trasladada desde el papel al formato digital y por lo general, era necesario almacenarla para su posterior recuperación, consulta y procesamiento. De este modo, para llevar a cabo un tratamiento eficiente de ésta era necesario establecer métodos adecuados para su almacenamiento. El elemento que permitió llevar a cabo el almacenamiento de datos de forma permanente en dispositivos de memoria masiva fue el **fichero** o **archivo**, del que hablaremos más detenidamente dentro de un momento. Así pues, por una parte se tenían los ficheros que contenían los datos con una estructura determinada y por otro los programas de aplicación que accedían a estos datos para producir información.

A partir de innovaciones tecnológicas de almacenamiento y comunicación de redes, los programas informáticos que manejaban la información tuvieron que:

- Implementar la posibilidad de realizar consultas y actualizaciones de los mismos datos simultáneamente por parte de diferentes usuarios.
- Interrelacionar los archivos.
- Eliminar la repetición innecesaria de datos ya que pone en riesgo la coherencia de los datos.

A raíz de estas necesidades surgen las bases de datos y será objeto de estudio de este módulo.

Con todo esto, se puede definir a la informática 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.

Hemos visto que 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 / Blu-ray)
- 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.- Ficheros

El elemento que permite almacenar datos de forma permanente en dispositivos de almacenamiento secundario es el fichero.

FICHERO DE CLIENTES

Campos:	DNI	NOMBRE	APELLIDOS	DIRECCION	TELEFONO
Registro 1:	73564765M	Javier	Barquín Arce	C/ Alta, 234	918342156
Registro 2:	56558765W	Luis	Gómez de Miguel	Avda. de Castilla, 2A	956235567
Registro 3:	13874521M	María Belén	Márquez Ruiz	C/ Floranes, 2	568732212
Registro 4:	75675317R	Carmen	Rodríguez Mata	Paseo Pereda, 123	942665544

Los ficheros están formados por registros lógicos que contienen datos relativos a un mismo elemento u objeto. A su vez, los registros están divididos en campos que contienen cada una de las informaciones elementales que forman un registro.

Por ejemplo, un registro `persona` contendría campos como `nombre, apellidos, dni, fecha de nacimiento, teléfono, edad,` etc.

Cada campo tiene tres propiedades fundamentales:

- **Nombre**

  Identifica al campo de forma única dentro de un mismo registro por tanto 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**

  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

Como los ficheros suelen ser muy voluminosos, sólo se pueden llevar a la memoria principal partes de ellos para poder procesarlos. La cantidad de información transferida entre el soporte en la que se almacena el fichero y la memoria principal del ordenador en una sola operación de lectura / escritura recibe el nombre de registro físico o bloque.

Normalmente en cada operación de lectura / escritura se transfieren varios registros del fichero, pues un bloque (registro físico) suele contener varios registros lógicos. El número de registros lógicos que entran en un bloque es el **factor de blocaje** y a esta operación de agrupar varios registros en un bloque se le llama **bloqueo** de registros.

4.- Tipos de ficheros

4.1.- Según su función

Ficheros permanentes

Contienen la información necesaria para el funcionamiento de una aplicación. Su vida es larga y normalmente no pueden generarse de forma inmediata a partir de otros ficheros. Dentro de ellos se distinguen:

- **Ficheros maestros**

  Contienen información que refleja el estado actual de los datos. Estos ficheros se actualizan periódicamente para adaptarlos a cada nueva situación pero su estructura no varía. Por ejemplo, el fichero con los datos de los usuarios de una plataforma educativa.

- **Ficheros constantes**

  Contienen datos fijos para la aplicación. Su información no sufre modificaciones frecuentes y suelen utilizarse como ficheros de consulta. Por ejemplo, el fichero con los códigos postales.

- **Ficheros históricos**

  Contienen datos que fueron considerados como actuales en un periodo o situación anterior. Se utilizan para la reconstrucción de situaciones. Por ejemplo, el fichero con los usuarios que han causado baja en la plataforma educativa.

Ficheros temporales

Contienen información necesaria para un proceso específico dentro de una aplicación. Se generan a partir de los datos de los ficheros permanentes. Tienen una vida corta y sólo se utilizan para obtener resultados o actualizar la información de los ficheros permanentes. Se clasifican en:

- **Ficheros intermedios**

  Almacenan resultados de una aplicación que serán utilizados por otra dentro de la misma tarea. Por ejemplo, para modificar el rol de un grupo de usuarios de la plataforma educativa, primero se crearía un fichero con los datos de los usuarios que cumplen la condición y luego se efectuaría la modificación.

- **Ficheros de maniobras**

  Se utilizan para no perder información generada por un proceso que por falta de espacio en memoria principal no se puede conservar.

- **Ficheros de resultados**

  Se generan a partir de los resultados finales de un proceso que van a ser transferidos a un dispositivo de salida, por ejemplo un fichero de impresión.

4.2.- Según la longitud de los registros

Los registros que componen un fichero pueden o no tener todos la misma longitud. Esto puede ser debido a la existencia de campos de longitud variable o por contener campos que se repiten un número variable de veces o por ambas causas.

4.2.1.- 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.

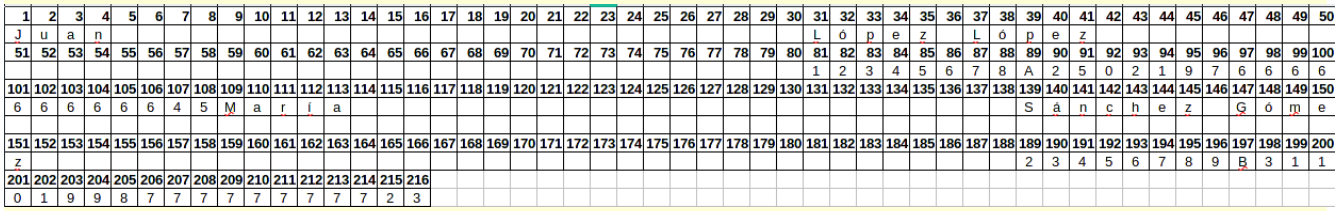
Su principal ventaja es que son fáciles de manipular puesto que para saber en qué posición del disco está un registro sólo hay que multiplicar el número del registro por el tamaño. Como desventaja principal tienen que son poco flexibles y pueden llegar a desperdiciar espacio.

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
  - **teléfono**: 666666666
  - **edad**: 45
- Persona 2
  - **nombre**: Maria
  - **apellidos**: Sánchez Gómez
  - **dni**: 23456789B
  - **fecha-de-nacimiento**: 31/10/1998
  - **teléfono**: 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	á	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 + 3 *108), etc.

4.2.2.- Ficheros de registros de longitud variable

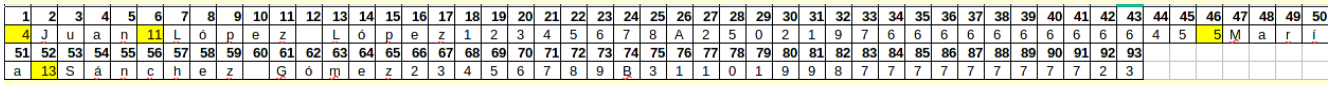
En este tipo de fichero los registros pueden ser de distintos tipos o bien ser todos del mismo tipo pero tener campos de longitud variable. Por lo tanto el espacio de almacenamiento que ocupa cada registro no es fijo sino que debe almacenarse como parte del registro.

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	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	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.

5.- Organización de 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 (por ejemplo, búsquedas, ordenación,…) sobre los ficheros se pueden realizar siempre utilizando estas operaciones básicas.

El acceso a un fichero está íntimamente ligado a su organización. La organización del fichero indica cómo están dispuestos los registros en el soporte de almacenamiento. Existen dos tipos de accesos:

- **Secuencial**

  Para acceder a un determinado registro N hay que recorrer sucesivamente los N-1 registros anteriores en la misma secuencia en que fueron escritos, hasta encontrar el registro adecuado.

- **Directo**

  Dada una llave (clave) se puede acceder directamente al registro sin necesidad de recorrer todos los anteriores.

Organización secuencial

Es el tipo más básico de organización. Los registros se almacenan secuencialmente uno a continuación del otro y los registros nuevos se añaden al final del fichero.

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 comparando el valor del campo que se pretende localizar con el valor del mismo campo correspondiente a cada registro leído del fichero, uno a uno desde el inicio del fichero hasta que se localice el registro buscado o se llegue al final del fichero, que estará indicado con una marca especial (EOF –End Of File). 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. En cualquier caso, el borrado es lógico, ya que consiste en marcar el registro de forma que al leerlo se identifique como no válido

Ejemplo:

Si se elimina el registro con clave 19:	[10][27][19*][21]
Si ahora se añade un nuevo registro:	[10][27][19*][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]

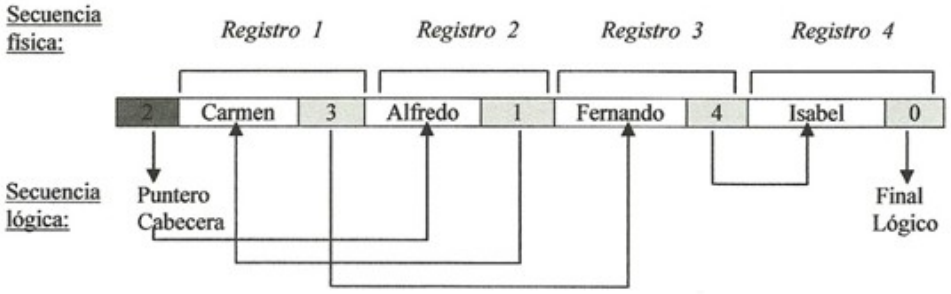
La organización secuencial es la única que puede gestionar un dispositivo de almacenamiento no direccionable, como por ejemplo una cinta magnética, lo que no impide que pueda ser utilizada por dispositivos de acceso directo.

Si los registros almacenados se identifican por medio de la información de uno de sus campos, a este campo se le denomina clave o llave. Si se guardan por este campo es más rápido realizar cualquier operación de lectura. La lectura en la organización secuencial siempre se realiza hacia delante.

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 se realizan pocas modificaciones y borrados. Sin embargo, los datos en sistemas reales no siempre se adaptan a estas restricciones, por lo que para mejorar su rendimiento, han surgido variaciones de la organización secuencial: organización secuencial encadenada y organización secuencial indexada.

Organización secuencial encadenada

En este caso los registros almacenan, además de su propia información, un puntero (tipo de dato que almacena una dirección de memoria) con la dirección del registro siguiente, según el orden lógico del fichero. Desde el punto de vista lógico, el fichero está ordenado según el valor de alguna llave. Los registros se encuentran almacenados en direcciones físicas totalmente arbitrarias, pero los punteros permiten asegurar la secuencia lógica del fichero.



Para añadir un registro al final del fichero se localiza la posición del último registro del fichero. El puntero de este último registro contendrá una dirección nula. Una vez localizado el final del fichero, el nuevo registro se escribirá en la zona libre, colocando en su campo de puntero una dirección nula. Finalmente se modifica el último registro para actualizar el valor de su puntero, de forma que contenga la dirección del nuevo registro.

La operación de consulta es secuencial. Para leer un determinado registro se accede al primero en la lista y se comprueba si es el registro buscado. Si no es así, se lee la dirección del siguiente gracias al puntero y se accede a dicha dirección. El proceso continúa hasta que se localiza el registro deseado o se llega al final del fichero.

Para insertar es necesario localizar la posición en que se desea insertar, es decir, entre qué dos registros se quiere situar (registro anterior y posterior). Físicamente el registro se escribe en una dirección de memoria arbitraria que se encuentre disponible, colocando en su campo de puntero la dirección del registro al cual va a preceder. Luego se modifica el registro anterior para actualizar el valor de su puntero, de forma que contenga la dirección del nuevo registro.

En cuanto a la operación de modificación, si no implica un aumento en la longitud del registro ni una alteración del campo clave, se localizará el registro y simplemente habrá que sobreescribir en la misma posición. En caso contrario, primero se insertará un nuevo registro que incluya la modificación y posteriormente se borrará el registro con la información desactualizada.

Para eliminar un registro del fichero sólo hay que destruir su dirección del puntero del registro anterior, copiando en dicho puntero la dirección del registro siguiente al que se desea borrar, es decir, la dirección que contiene el puntero del registro a eliminar.

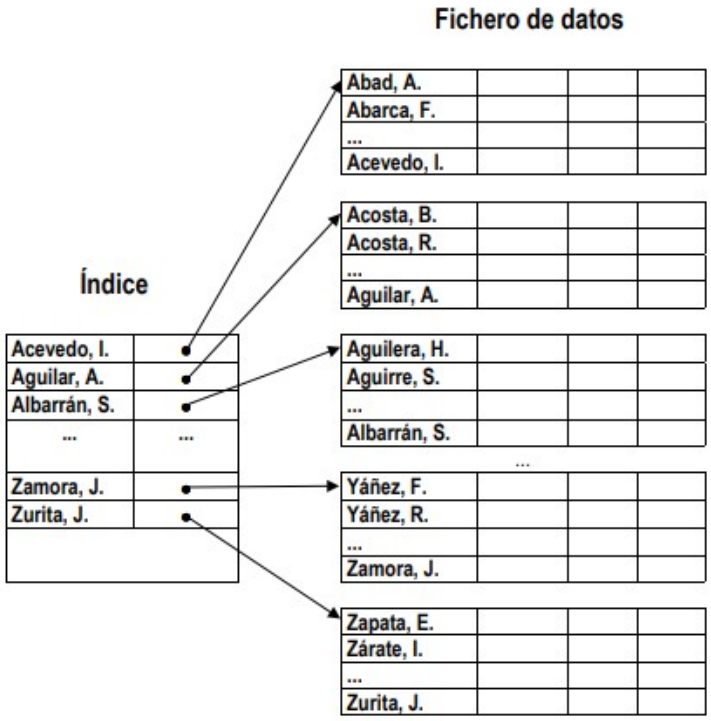
Organización secuencial indexada

En esta organización el fichero está formado por dos estructuras o zonas:

- **Zona de registros**, que contiene todos los registros del fichero, ordenados según el valor de una llave. Esta zona está dividida en tramos lógicos. Cada tramo está formado por una serie de registros consecutivos. Por cada tramo de la zona de registros hay un registro en la zona de índices.
- **Zona de índices**, que se procesa sólo de forma secuencial y cuya estructura es la de un fichero secuencial puro en que cada registro contiene dos campos: un campo llave (que contendrá algunos valores de la llave del fichero) y un campo dirección (que contendrá la dirección de un registro del fichero).

Este tipo de organización sólo se puede utilizar en soportes de almacenamiento direccionables.

Como se observa en la imagen, en la tabla de índices cada fila hace referencia a cada uno de los segmentos de la zona de registros. La clave corresponde al último registro del segmento y el índice apunta al registro inicial. Una vez que se accede al primer registro del segmento, dentro de él se localiza de forma secuencial el registro buscado.



Organización directa (aleatoria)

En este tipo de organización los registros se almacenan en una posición física que dependerá del espacio disponible en memoria masiva, de ahí que la distribución de los registros sea aleatoria dentro del soporte de almacenamiento.

La dirección se calcula para cada registro aplicando una fórmula o algoritmo matemático sobre un campo del registro. Cuando la dirección asignada a un registro está ya ocupada por otro se produce una colisión. Para resolver el problema se puede optar por dos alternativas.

- Buscar una nueva dirección libre en el mismo espacio de fichero hasta encontrar una posición libre donde almacenar la información.
- Reservar una zona de desbordamiento e ir almacenando los sinónimos (registros con diferentes llaves pero con una misma dirección física) en esta zona de manera consecutiva a media que van apareciendo.

Dependiendo de la función de transformación elegida se diferencian los siguientes métodos de direccionamiento

- **Directo**

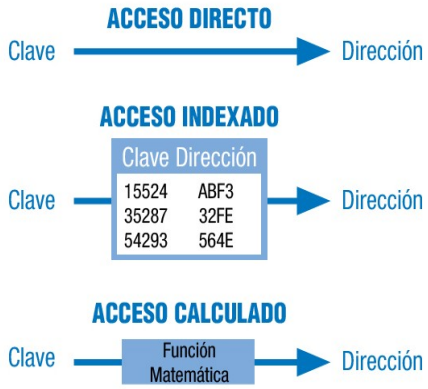
  Consiste en asignar a cada llave una dirección lógica, es decir, la transformación asigna como dirección el valor de la llave.

- **Indexado**

  Consiste en asociar a cada llave una dirección lógica por medio de una tabla. Este método no produce sinónimos pero necesita espacio adicional para la tabla.

- **Calculado (Hashing)**

  En este caso la dirección de cada registro se obtiene mediante una serie de cálculos matemáticos realizados con la llave. Si la clave es alfanumérica deberá previamente ser transformada en un número. Una buena función de hash será aquella que produzca el menor número de colisiones.



6.- Sistemas de bases de datos

Inicialmente, cuando las primeras empresas y organizaciones empezaron a usar sistemas informáticos trabajaban con sistemas de ficheros. Es decir, se trabajaba con programas que manejaban información almacenada en ficheros. Cada equipo trabajaba con sus propios datos y programas y se encargaba de su mantenimiento y gestión. Al principio el sistema funcionó pero con el tiempo y, sobre todo, con el incremento de la cantidad de información así como de los usuarios que la manejaban surgieron problemas (integridad y duplicidad de información, seguridad, etc., que llevaron finalmente a la organización de la información mediante un sistema más ordenado y manejable basado en la centralización de la gestión y la organización de los datos en forma de bases de datos.

Los principales problemas relacionados con el uso de ficheros como sistema de almacenamiento de información fueron los siguientes:

- **Separación y aislamiento de los datos**

  Cuando los datos están separados en distintos ficheros es más complicado acceder a ellos, y es responsabilidad del programador de la aplicación el coordinar el acceso a los distintos ficheros para asegurar que se extraen los datos correctos.

- **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 estructura física de los datos está codificada en los programas de aplicación, cualquier cambio en dicha estructura es difícil de realizar. El programador debe identificar todos los programas afectados por el cambio, modificarlos y volverlos a probar, lo que implica mucho tiempo y está sujeto a que se produzcan errores.

- **Formatos de ficheros incompatibles**

  Dado que la estructura de los ficheros se define en los programas de aplicación, es completamente dependiente del lenguaje de programación. La incompatibilidad entre ficheros generados por distintos lenguajes hace que los ficheros sean difíciles de procesar de modo conjunto.

- **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 no existe un esquema general de muestre la organización de la información y cada departamento tiene sus datos 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.

No debemos olvidar que en última instancia todo se almacena en ficheros. La diferencia es que cuando usamos bases de datos no trabajamos directamente con ficheros, sino con estructuras de datos más fáciles de manejar.

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.

Los datos están organizados en estructuras que se definen una única vez y que se utilizan simultáneamente por varios equipos y usuarios.

Los datos en lugar de almacenarse en ficheros desconectados entre si y con información redundante, están centralizados y organizados, de forma que se minimice la redundancia y se facilite su gestión.

La base de datos no pertenece a un equipo, se comparte por toda la organización.

La base de datos no sólo contiene los datos de la organización 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 que, en muchos casos, se organiza en otra base de datos.

6.1.- Arquitectura

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.

6.2.- 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 muchas 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.

> *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.

6.3.- 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

6.3.1.- 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` que informa sobre cual es la oficina en la que trabaja cada empleado, o dicho de otra forma, qué empleados trabajan en una oficina determinada. Más adelante estudiaremos este modelo con más detalle.

6.3.2.- 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 la información se estructura a través de un esquema de árbol. Las únicas relaciones entre entidades que se permiten son de padres a hijos. Una instancia de una entidad (padre) puede estar relacionada con una o más instancias de otra (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. XML usa este tipo de modelo.

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. El esquema que se obtiene en cuanto a las relaciones de sus nodos es un grafo. Es difícil de administrar.

Modelo relacional

Este modelo es posterior a los dos anteriores. Los datos y las relaciones entre estos se representan mediante una colección de tablas bidimensionales. Dedicaremos a este apartado el siguiente tema.

Modelo orientado a objetos

El objetivo de este modelo es cubrir las limitaciones del modelo relacional utilizando el potencial de los lenguajes de programación orientados a objetos. En este modelo se almacena el comportamiento de las entidades además de su información.

Los conceptos más importantes que incorpora del paradigma de la programación orienta a objetos son la encapsulación, la herencia y el polimorfismo.

Existen más modelos lógicos, como por ejemplo el modelo objeto-relacional , el modelo declarativo o el modelo NoSQL.

6.3.3.- Modelos físicos

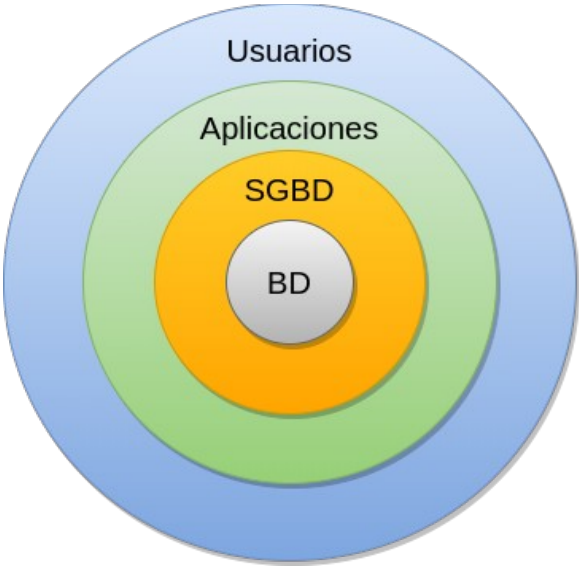
Describen las estructuras de almacenamiento concretas en el sistema de gestión de bases de datos, indicando qué se almacena, donde y cómo. El modelo físico depende fuertemente del sistema de gestión de bases de datos empleado, por lo que no se puede dar una descripción general del mismo, debiendo referirse a la documentación del sistema de gestión de bases de datos concreto elegido.

7.- Sistemas Gestores de Bases de Datos

Un Sistema de Gestión de Bases de Datos (SGBD), también llamado DBMS (DataBase Management System), 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 SGBD 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 porqué 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.



7.1.- Funciones

Las funciones principales que destacar de un SGBD son:

- **Gestionar el diccionario de datos (catálogo)**

  Un diccionario de datos es una base de datos que mantiene el SGBD automáticamente y que contiene información sobre los datos que se almacenan en la misma, por tanto es una base de datos sobre los datos. 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, por ejemplo, frecuencia de las transacciones, número de accesos realizados a los objetos de la base de datos,...

- **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. Una transacción es un conjunto de acciones que cambian el contenido de la base de datos. 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).

- **Recuperación de 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. 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.

7.2.- Componentes

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.

- **Objetos**

  Objetos que se mantienen en la base de datos. El número y tipo depende del SGBD, por ejemplo

  - Tablas base y vistas (tablas derivadas).
  - Consultas.
  - Dominios y tipos definidos de datos.
  - Restricciones de tabla y dominio y aserciones.
  - Funciones y procedimientos almacenados.
  - Disparadores o triggers.

- **Herramientas para**

  Administrar y gestionar el SGBD (seguridad, integridad, concurrencia, migraciones,...).
- **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).

7.3.- Usuarios

Generalmente se distinguen cuatro grupos de usuarios que utilizan las bases de datos. Estos son:

- **Administradores**

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

  Realizan el diseño lógicode la base de datos, debiendo identificar los datos, las relaciones entre ellos, sus restricciones, etc. Para ello han de conocer a fondo los datos y procesos a representar en la base de datos. Si estamos hablando de una empresa, será necesario que conozcan las reglas de negocio en la que esta se mueve. Para obtener un buen resultado, el diseñador de la base de datos debe implicar en el proceso a todos los usuarios de la base de datos, tan pronto como sea posible.
- **Programadores de aplicaciones**

  Implementan los programas de aplicación que servirán a los usuarios finales. Estos programas ofrecerán la posibilidad de realizar consultas de datos, inserción, actualización o eliminación de los mismos.
- **Usuarios finales**

  Trabajan en el nivel externo mediante vistas. Son clientes de las bases de datos que las usan sin necesitar tener conocimiento alguno sobre su funcionamiento, ubicación, etc.

7.4.- Tipos de SGBD

Existen numerosos SGBD en el mercado que se pueden clasificar según los siguientes criterios:

- **Número de usuarios**
  - Monousuario
  - Multiusuario
- **Modelo lógico en el que se basan**
  - Jerárquico
  - En red
  - Relacional
  - Objeto-relacional
  - Orientado a objetos
- **Número de sitios**
  - Centralizados: en un solo servidor o equipo.
  - Distribuidos: en varios equipos que pueden ser homogéneos y heterogéneos.
- **Ámbito de aplicación**
  - Propósito General: orientados a toda clase de aplicaciones.
  - Propósito Específico: centradas en un tipo específico de aplicaciones.
- **Tipos de datos**
  - Sistemas relacionales estándar: manejan tipos básicos (int, char, etc.).
  - XML: para el caso de bases de datos que trabajan con documentos xml.
  - Objeto-relacionales: para bases relacionales que incorporan tipos complejos de datos.
  - De objetos: para bases de datos que soportan tipos de objeto con datos y métodos asociados.
- **Lenguajes soportados**
  - SQL estándar.
  - NoSQL o nuevo lenguaje de consulta: menos estructurado y orientado a bases documentales o de tipo clave-valor. Es muy útil para manejar consultas de grandes cantidades de datos distribuidos en clusters de servidores.

El modelo más extendido es el relacional. Esto se debe principalmente a su flexibilidad y sencillez de manejo. El lenguaje SQL se ha convertido en un estándard para el manejo de datos de un SGBD relacional. Sin embargo, hay una creciente popularidad de otros lenguajes como NoSQL (Not Only SQL) que se han desarrollado para adaptarse a datos masivos y dispersos en muchos servidores.

7.4.1.- SGBD comerciales

Actualmente existen multitud de SGBD comerciales. La elección de un SGBD es una decisión muy importante a la hora de desarrollar proyectos. A veces, el sistema más avanzado, "el mejor" según los entendidos, puede no serlo para el tipo de proyecto que estemos desarrollando. Hemos de tener en cuenta qué volumen de carga debe soportar la base de datos, qué sistema operativo utilizaremos como soporte, cuál es nuestro presupuesto, plazos de entrega, etc.

A través de la siguiente tabla se exponen los SGBD comerciales más utilizados y sus características más relevantes:

SGBD	Descripción
ORACLE	Reconocido como uno de los mejores a nivel mundial. Es multiplataforma, confiable y seguro. Es cliente / servidor. Basado en el modelo de datos relacional. De gran potencia, aunque con un precio elevado hace que sólo se vea en empresas muy grandes y multinacionales. Ofrece una versión gratuita Oracle Database Express Edition 11*g* Release 2.
MYSQL	Sistema muy extendido que se ofrece bajo dos tipos de licencia, comercial o libre. Para aquellas empresas que deseen incorporarlo en productos privativos, deben comprar una licencia específica. Es relacional, multihilo, multiusuario y multiplataforma. Su gran velocidad lo hace ideal para consulta de bases de datos y plataformas web.
DB2	Multiplataforma, el motor de base de datos relacional integra XML de manera nativa, que permite almacenar documentos completos para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.
Microsoft SQL SERVER	Producido por Microsoft. Es relacional, sólo funciona bajo Microsoft Windows, utiliza arquitectura cliente / servidor. Constituye la alternativa a otros potentes SGBD como son Oracle, PostgreSQL o MySQL.
SYBASE	Un SGBD con bastantes años en el mercado. Tiene tres versiones para ajustarse a las necesidades reales de cada empresa. Es un sistema relacional, altamente escalable, de alto rendimiento, con soporte a grandes volúmenes de datos, transacciones y usuarios, y de bajo costo.

Otros SGBD comerciales importantes son: DBase, Access, Interbase y Foxpro.

7.4.2.- SGBD libres

Son la alternativa a los SGBD comerciales. Son sistemas distribuidos y desarrollados libremente. En la siguiente tabla se relacionan los más utilizados actualmente, así como sus principales características.

SGBD	Descripción
MySQL	Es relacional, multihilo y multiusuario con más de seis millones de instalaciones. Distribuido bajo dos tipos de licencias, comercial y libre. Multiplataforma, posee varios motores de almacenamiento, accesible a través de múltiples lenguajes de programación y muy ligado a aplicaciones web.
PostgreSQL	Sistema relacional orientado a objetos. Considerado como la base de datos de código abierto más avanzada del mundo. Desarrollado por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y / o apoyados por organizaciones comerciales. Es multiplataforma y accesible desde múltiples lenguajes de programación.
Firebird	Relacional, multiplataforma, con bajo consumo de recursos, excelente gestión de la concurrencia, alto rendimiento y potente soporte para diferentes lenguajes.
Apache Derby	Sistema gestor escrito en Java, de reducido tamaño, con soporte multilenguaje, multiplataforma, altamente portable, puede funcionar embebido o en modo cliente / servidor.
SQLite	Sistema relacional, basado en una biblioteca escrita en C que interactúa directamente con los programas, reduce los tiempos de acceso siendo más rápido que MySQL o PostgreSQL, es multiplataforma y con soporte para varios lenguajes de programación.
MariaDB	Este SGBD fue en origen un fork de MySQL. Incorpora numerosas mejoras sobre éste y está siendo muy utilizada en aplicaciones web en los últimos años.