

UNIDAD 1: Bases de Datos y Sistemas de almacenamiento de la Información



Obxectivos de la Unidad 1:

- Analizar los sistemas lógicos de almacenamiento y sus características.
- Identificar los tipos de bases de datos según el modelo de datos utilizado y de la localización de la información.
- Conocer las ventajas de utilizar un SGBD y las diferencias que existen con respecto a una BD.
- Conocer los componentes de una Base de Datos así como definir que es una Base de Datos.
- Clasificar los SGBD.
- Conocer qué es y como funciona la fragmentación de la información.

Sumario

1.- Sistemas de almacenamiento de la información.....	4
2.- Ficheros.....	7
3.- Características de las bases de datos.....	11
4.- Tipos de bases de datos.....	15
Fragmentación.....	23
5.- Diferentes visiones de los datos en las bases de datos.....	27
6.-Bases de Datos (BD) y Sistemas de Gestión de Bases de Datos (SGBD).....	30
6.1-Componentes de los SGBD.....	32
7.-SGBD Libres-Propietarios.....	39
8.-Otros sistemas de almacenamiento: XML, servicio de directorios,.....	40
8.1.-Sistemas de almacenamiento XML.....	40
8.2.-Servicio de directorio (LDAP).....	42
9.- Evolución de las Bases de Datos y los SGBD.....	43
10.-Bibliografía.....	44

1.- Sistemas de almacenamiento de la información.



En los años 60, los procesos empresariales básicos requerían almacenamiento y gestión de documentos en papel que se guardaban en archivos físicos, agrupados y ordenados de diferentes formas.

A medida que la tecnología informática fue creciendo y avanzando, se fue transfiriendo la información a dispositivos físicos para que fueran accedidos desde los ordenadores, permitiendo una mayor velocidad de acceso a la información y la posibilidad de guardar grandes cantidades de información.

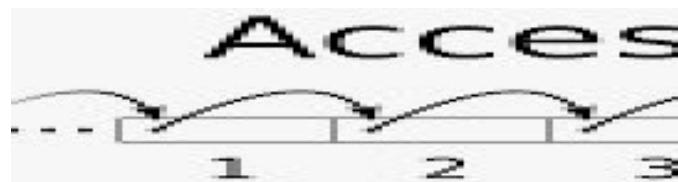
El término de *Bases de Datos* no apareció hasta mediados de los años sesenta, época en la cual la información era representada haciendo uso de un **conjunto de ficheros**, generalmente planos (es decir, el contenido no tenía formato, sólo era texto). Estos ficheros no estaban relacionados entre sí, y los datos almacenados representaban las relaciones existentes en la información que representaban mediante referencias simbólicas y/o físicas. La redundancia (información repetida) era grande y la integridad de la información representada (que la información que se guardaba era reflejo de la realidad) dejaba mucho que desear.



Aun así, muchos desarrolladores de software "bautizaban" a sus sistemas de ficheros como *Bases de Datos*, sin preocuparse de que cumplieran o no una serie de propiedades que deben acompañar al uso de este término, como se verá mas adelante.



Los procedimientos y estructuras para el almacenamiento y mantenimiento de la información correspondientes a un determinado dominio de un problema han evolucionado a medida que ha evolucionado la tecnología. Inicialmente los dispositivos de almacenamiento sólo permitían un acceso serial a la información, por lo que las estructuras de datos, mediante las cuales se podía representar la información, debían ser muy simples (archivos con organización secuencial) y los procedimientos de acceso a esta información requerían un alto tiempo de cómputo puesto que eran puramente seriales.



Además, los procedimientos encargados del mantenimiento de la información eran totalmente dependientes del hardware utilizado para ello, lo que suponía una continua modificación del software encargado de esta tarea cuando el hardware cambiaba.



- En un principio vemos que existían dos problemas que serán solucionados cuando aparezcan las verdaderas bases de datos: Acceso secuencial a la información => lentitud
- Los programas tenían una dependencia de los dispositivos físicos donde se guardaba la información



Dispositivo de acceso secuencial: Cintas Magnéticas

Con la aparición de los dispositivos de almacenamiento que permitían el acceso directo (generalmente denominado aleatorio) las estructuras mediante las cuales se podía estructurar la información se fueron haciendo más complejas. En esta época (histórica) se desarrollaron procedimientos de acceso directo a la información, si bien seguían siendo estos procedimientos los encargados de describir la estructura del almacenamiento de la información que trataban. Si la estructura cambia debido a cualquiera de las múltiples razones posibles (cambios en los requerimientos del cliente del sistema o cambios en el entorno del sistema, lo que puede suponer la modificación de la estructura de los registros que se almacenan en los archivos), los procedimientos debían ser modificados para reconocer la nueva estructura de la información. En esta época, aunque el almacenamiento de la información sí era independiente del dispositivo hardware utilizado, la estructura de la información no era independiente de los procedimientos que la manejaban.



Con los dispositivos de almacenamiento aleatorio resolvíamos parte del problema en la lentitud a la hora de recuperar información, pero seguíamos teniendo el problema de la dependencia del hardware (entre otros problemas que veremos después)

2.- Ficheros

Vamos a hacer un pequeño paréntesis y vamos a explicar de forma breve como se guardaba la información en los archivos y que tipos de acceso disponemos para guardar o recuperar la información.

Definiciones:

Un campo (Field) es el elemento de datos básico. Un campo individual contiene un valor único. Esta caracterizado por su longitud y por el tipo de datos. Dependiendo del diseño del archivo, los campos pueden ser de tamaño fijo o variable.

Registro (Record) es una colección de campos relacionados que pueden tratarse como una única unidad. Por ejemplo:, un registro de empleados va contener campos como nombre, numero de seguridad social, etc.

También dependiendo del diseño, los registros pueden ser de longitud fija o de longitud variable. Un registro va a tener una longitud variable si algunos de los campos son de tamaños variables o si el numero de campos es variable. Cada campo tiene un nombre de campo.

Archivo (File) es una colección de registros similares. El archivo es tratado como una entidad individual por los usuarios y las aplicaciones y puede ser referenciada por el nombre. Los archivos tienen nombres únicos y pueden crearse y borrarse.

Una vez que tenemos definidos los registros, podemos guardarlos en un archivo. Dependiendo de cómo podamos acceder al mismo, disponemos de:

1.- Ficheros de acceso secuencial: es aquel donde los registros están ubicados consecutivamente sobre un dispositivo de almacenamiento. De tal forma que para acceder a un registro determinado 'd' debemos pasar obligatoriamente por todos los registros que le preceden.

Suponiendo una cinta magnética, para leer registro 'd' debemos pasar antes por el 'a', 'b', 'c'.

----- Para leer el registro 'd', la cabeza
+- a b c d ... lectora, deberá pasar antes por los
| ----- que le preceden.
+> cabeza lectora.

2.- Fichero de acceso directo: es aquel que se encuentra almacenado en un dispositivo direccionable; y donde sus registros poseen un campo que denominaremos campo clave y que identifica inequívocamente a cada registro. En los ficheros de acceso directo el campo clave es el número del registro en el fichero. Así se establece una correspondencia directa entre los valores del campo clave y las direcciones lógicas en el soporte. Los registros se almacenan según el orden de entrada y no quedan ordenados.

De esta forma en un Fichero de Acceso Directo nos referimos a un registro por medio de su posición en este. Así, podremos obtener el reg número 4 sin pasar antes por los demás.

Reg	D A T O	
1	LUIS	Fíjese que los datos están almacenados en el orden en el
2	CARLOS	que han sido introducidos por el
3	TERESA	usuario.
4	JOAQUIN	
5	INMA	Accedemos a los datos por medio
6	JOSE	del valor de la posición del
		registro.

3.-Fichero de acceso indexado: es aquel que se encuentra almacenado en un dispositivo direccionable; y donde sus registros poseen un campo que denominaremos campo clave principal y que identifica inequívocamente a cada registro. La clave principal debe ser aquel campo del registro estructurado que tenga siempre un valor diferente a los ya introducidos, es decir, dentro del fichero Indexado no puede haber dos registros con los campos clave principal iguales.

Además del campo clave principal pueden existir otros campos claves secundarios que realizan la misma tarea que el campo clave, sin embargo, sus valores pueden repetirse. En los Ficheros de Acceso Indexado el campo clave puede ser cualquiera de los campos de un registro estructurado. Así se establece una correspondencia directa entre los valores del campo clave y el propio registro al que pertenece. Los registros se almacenan ordenados alfabéticamente por el campo clave, esto nos facilita la búsqueda y listados ordenados por los distintas claves.

Para cada campo clave, el fichero genera una tabla, donde dicha clave aparece ordenada alfabéticamente y se relaciona con la dirección de los datos.

De esta forma en un Fichero de Acceso Indexado nos referimos a un registro por medio de alguna de los campos que posea el fichero al cual se la ha asociado un índice. Es decir, leer el registro cuyo DNI sea: 46.399.554, en este caso leería el registro correspondiente a INMA. También podríamos haber dicho, leer los registro cuya Edad=16 y primero nos leería el registro correspondiente a los datos de Luis y en la siguiente petición de lectura los datos de Teresa. La diferencia entre índice principal y secundario, está en que el índice principal es único (relacionando así inequívocamente al registro al que pertenece) mientras que los índices secundarios pueden tener valores repetidos.

Índice Primario	Índice Secundario	Índice Secundario	Índice Secundario	
(Direc)	(D.N.I.)	(Nombre)	(Provincia)	(Edad)
1	55.366.546	LUIS	Las Palmas	16
2	42.386.225	CARLOS	Salamanca	17
3	32.387.203	TERESA	Oviedo	16
4	46.399.554	INMA	Palencia	20
5	60.643.434	JOAQUIN	Salamanca	17
6	22.543.986	JOSE	Las Palmas	23

Como podemos observar, esto sería un ejemplo de un fichero indexado.

Para cada campo clave, el fichero genera una tabla, donde dicha clave aparece ordenada alfabéticamente y se relaciona con la dirección de los datos. Así las tablas para la clave principal (DNI) y la clave secundaria (Nombre) serían:

(D.N.I.)	(Direc)	(Nombre)	(Direc)
22.543.986	6	CARLOS	2
32.387.203	3	INMA	4
42.386.225	2	JOAQUIN	5
46.399.554	4	JOSE	6
55.366.546	1	LUIS	1
60.643.434	6	TERESA	3

Tabla de Ac. Clave Princ.	Tabla de Ac. Clave Secund.
---------------------------	----------------------------

Obsérvese como ambas tablas aparecen ordenadas alfabéticamente (o de menor a mayor en el caso del DNI). Como ya dijimos, esto nos da grandes facilidades a la hora de realizar búsquedas y/o listados ordenados.

Veremos posteriormente que los archivos no pueden ser considerados bases de datos al no cumplir algunas de las características necesarias.

Como los sistemas son dinámicos, los requerimientos cambian con el tiempo, la información a ser tratada en cada problema también cambia y, por tanto, es necesario, de alguna manera, independizar la estructura de la información (los archivos encargados de almacenaría) de los procedimientos encargados de su tratamiento, si no se estaría siempre abocado a la dedicación de una gran cantidad de esfuerzo a la modificación de todos aquellos procedimientos encargados del mantenimiento de la información.

Cuando se reconoce que los sistemas evolucionan y que, por tanto, la información y la estructura de la misma no son estáticas sino que va cambiando con el tiempo, es cuando aparece el concepto de las *Bases de Datos*. Si se desea que cualquier modificación en la cantidad, contenido y estructura de la información que se desea mantener acerca de un determinado problema no afecte a los procedimientos desarrollados previamente para el mantenimiento de la misma, es necesario tener en cuenta que **existe una independencia de los datos con respecto a los procedimientos.**

Por ejemplo, si utilizamos un registro (estructura de datos compuesta por campos) para guardar información de un alumno, y desarrollo un procedimiento para leer dicha información desde un archivo secuencial (en el que los registros están guardados uno detrás de otro), dicho procedimiento va a depender de la estructura física que compone el registro, de tal forma que si cambio el formato de dicho registro (por un cambio en la visión conceptual del problema) será necesario modificar el procedimiento para adaptarlo al nuevo formato.



Esta independencia de la información con respecto a los procedimientos que maneja debe satisfacerse a dos niveles de abstracción para que sea efectiva; por tanto, se habla de:

Independencia lógica de los datos, por la que la modificación de la representación lógica general del dominio del problema no afecta a los programas de aplicación que los manipulan, siempre que esta modificación no elimine ninguno de los ítems de datos que estos programas requieran.

Independencia física de los datos, por la que la distribución en las unidades de almacenamiento y la estructura física de la información almacenada es independiente de los cambios de la estructura lógica general de la información y, por tanto, de los procedimientos que manejan la misma.

Por ejemplo, si tengo un gestor de base de datos relacional (utiliza tablas para guardar la información), puedo modificar la estructura de las tablas, añadiendo nuevos campos, sin que tenga que modificar nada de los procedimientos que hacen uso de esa tabla. Si modifico físicamente los registros para que tengan un mayor rendimiento a la hora de mostrarlos, no va a afectar en nada al diseño lógico (la tabla) que estoy utilizando.

3.- Características de las bases de datos

La información que forma parte de una base de datos puede organizarse de múltiples formas pero con independencia de la arquitectura de la base de datos, ésta debe cumplir una serie de características para ser considerada como tal, algunas de las cuales se describirán a continuación:

Versatilidad para la representación de la información: Si bien la información que forma parte del dominio de un problema es única y caracteriza a ese problema o sistema, pueden existir diferentes visiones de esa información. Visiones parciales en las que sólo se tiene en cuenta parte del dominio del problema y/o visiones globales que observan el problema desde diferentes puntos de vista.

Ejercicio: Pensar en un negocio y ver cual serían las diferentes visiones que tendrían de la información.

Por ejemplo, una empresa dedicada a la venta de coches:

- Comercial: le interesaría saber que coches lleva vendidos, cuales, en qué tiempo, con qué coches tiene más beneficio.
- Taller: interesaría saber que coches fueron vendidos (tipo) y en qué tiempo, para determinar cuando van a venir a revisión y las piezas que tiene que pedir.
- Jefe: ganancias que lleva por la venta de los coches, en períodos de tiempo, por vendedor (para despedir claro ☺)

Un procedimiento, un programa de aplicación, que maneja la información correspondiente a un problema puede, por tanto, tener en cuenta sólo parte del conjunto de información, mientras que otro procedimiento puede considerar a otro conjunto diferente, o no, de información del mismo problema.

Si se considera que un procedimiento "ve" la información que maneja como un registro, la organización de la información en la base de datos debe permitir que diferentes procedimientos puedan construir diferentes registros a partir de la información existente en la base de datos. Estos registros (lógicos) estarán formados por ítems de datos que forman parte del dominio del problema y que son derivados del conjunto de los ítems de datos existentes en ese problema y, además, cada uno de estos registros lógicos construidos por los procedimientos **deben** ser independientes de los registros físicos existentes en la base de datos para almacenar la información.

Por ejemplo, si se quisiera implantar una BD que gestionara una empresa de teléfonos móviles, esta tendría que almacenar todo la información necesaria para su gestión, pero el uso que se haga de esa información variará según el empleado que la maneje. Así, un vendedor le querrá tener información acerca de los productos que vende su empresa, tipos de móviles, precios y características, pero un directivo querrá saber la evolución económica de la empresa, ventas, número de clientes....Como vemos, son diferentes visiones parciales de la misma BD.

Desempeño: Las bases de datos deben asegurar un tiempo de respuesta adecuado en la comunicación hombre-máquina, permitiendo el acceso simultáneo al mismo o distinto conjunto de ítems de datos por el mismo o distinto procedimiento.

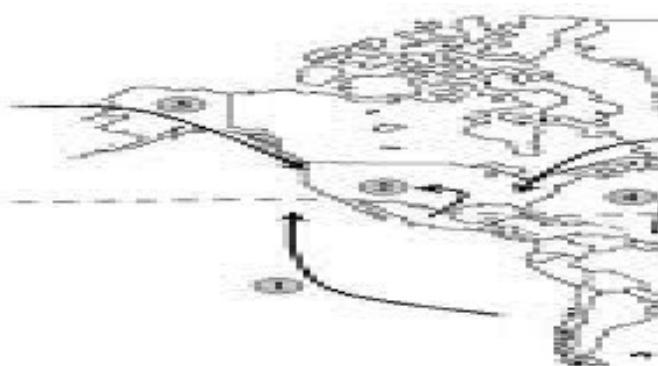
Mínima redundancia: Una de las principales razones por las que surgió la tecnología de las bases de datos fue el evitar la alta redundancia que se presentaba en los sistemas de procesamiento de la información debido al uso de archivos con estructuras planas. Redundancia quiere decir repetición de la información, como por ejemplo almacenar en dos sitios distintos la dirección de los clientes.

Otro ejemplo de redundancia

CodLibro	Titulo	Autor	Editorial
1001	Variable compleja	Murray Spiegel	McGraw Hill
1004	Visual Basic 5	E. Petroustos	Anaya
1005	Estadística	Murray Spiegel	McGraw Hill
1006	Oracle University	Nancy Greenberg	Oracle Corp.
1006	Oracle University	Priya Nathan	Oracle Corp.
1007	Clipper 5.01	Ramalho	McGraw Hill

La existencia de redundancia es nefasta debido a la posibilidad de inconsistencia (que una información no se ajuste a la realidad) en la información almacenada en la base de datos. La redundancia implica la existencia de varias copias de un mismo ítem de datos, las cuales pueden, en un momento dado, tener distintos valores. Además, hay que tener en cuenta que esta duplicación implica unas necesidades de almacenamiento innecesarias.

Un objetivo principal de las bases de datos es eliminar la redundancia siempre que ello no implique una complejidad de la misma y/o una disminución en el desempeño. Si existen diversas copias de un mismo ítem de datos, es necesario establecer procedimientos que garanticen la consistencia de la información cuando estas copias sean utilizadas por diferentes procedimientos al mismo tiempo. Por otro lado, si sólo existe una copia de un ítem de datos es necesario establecer procedimientos que permitan el acceso a esta copia por varios procedimientos, para garantizar un desempeño aceptable.



Por ejemplo, si tengo una base de datos distribuida (en diferentes lugares) y tengo la lista de clientes con sus datos en todas ellas, será necesario que si se produce un cambio se ‘propague’ por todas las demás.

Capacidad de acceso: Los usuarios de la base de datos reclaman a ésta continuamente información sobre los datos almacenados. Estas interrogaciones a la base de datos, que pueden ser conocidas o no, cuando se diseñó la misma, solicitan información correspondiente a distintos ítems de datos, así como sus relaciones, representadas en la base de datos y, por añadidura, agrupados, formateados, etc., de múltiples formas.

Una base de datos debe ser capaz de responder, en un tiempo aceptable, a cualquier consulta sobre la información que mantiene, sin restricciones graves en cuanto a los ítems, relaciones, formato, etc., solicitados en la misma, y respondiendo al usuario rápidamente.



Nota: Veremos más adelante que podemos dotar a las bases de datos de mecanismos para aumentar la ‘velocidad’ de acceso (como los índices en las bases de datos relaciones). Pero para ello debemos de conocer de antemano el tipo de consultas que se van a realizar, por lo que puede suceder que si disponemos de miles de registros, alguna consulta no planificada tarde un tiempo de respuesta muy alto....

Simplicidad: Las bases de datos deben estar basadas en representaciones lógicas simples que permitan la verificación en la representación del problema que representan y, más aún, la modificación de los requerimientos en el mismo, de tal forma que la inclusión de nuevos ítems de datos y relaciones no ocasione una complejidad excesiva.

Integridad: La integridad de una base de datos hace referencia a que los datos almacenados en la misma sean correctos y se correspondan a los requerimientos del dominio que del problema que estamos modelizando. Así los procedimientos que manejan la información deben garantizar la integridad de estos datos debido a dos posibles problemas: problemas durante el procesamiento de la información (fallos hardware, software, del sistema...) y problemas que puedan surgir debido al acceso concurrente de muchos usuarios al mismo tiempo. Por otro lado, en una base de datos deben establecerse procedimientos que verifiquen que los valores de los datos se ajustan a los requerimientos y restricciones extraídos del análisis del problema.

Ejercicio: indicar algún ejemplo de restricción o requerimiento del mundo real.

Seguridad y Privacidad: La seguridad de una base de datos hace referencia a la capacidad de ésta para proteger los datos contra su pérdida total o parcial por fallos del sistema o por accesos accidentales o intencionados a los mismos. Mientras que la privacidad de una base de datos hace referencia a la reserva de la información de la misma a personas no autorizadas.

La información de la base de datos es de vital importancia y valor para la organización/empresa responsable de la misma. A medida que esta información es más importante, tanto más valioso se hace para la empresa el mantener su seguridad y privacidad. Para conseguir estas características, una base de datos debe satisfacer, al menos, los siguientes requisitos:

- Seguridad contra la destrucción de los datos causada por el entorno: fuego, robo, inundaciones y cualquier otra forma.
- Seguridad contra la destrucción de los datos causada por fallos del sistema (hardware y software), de forma que los datos puedan reconstruirse.
- Seguridad contra accesos no autorizados a la base de datos.
- Seguridad contra accesos indebidos a los datos.

Por lo que deben existir en la base de datos tanto procedimientos de recuperación de la información perdida total o parcialmente por cualquier causa, como procedimientos que supervisen el acceso a los datos por los usuarios de la base de datos.

Afinación: La afinación hace referencia a la organización física de la información de la base de datos, la cual determina directamente el tiempo de respuesta de los procedimientos que operan sobre la misma.

Si una de las características que debe tener una base de datos es un buen desempeño, la organización física de los datos debe ser tal que ésta pueda ser alcanzada. Pero la base de datos evoluciona con el tiempo, el volumen de información va haciéndose cada vez más importante y, por añadidura, tanto los ítems de datos como las relaciones entre ellos pueden ampliarse y/o modificarse. Esto implica que una buena organización física de los datos en un momento dado, puede no ser tan buena en otro.

Por ello, la base de datos debe ser flexible a la modificación de esta organización física, lo que puede suponer además una migración de los datos según evolucione la base de datos, sin que por ello se vean afectados los procedimientos u otras representaciones de los datos pero, sin embargo, se consiga un desempeño más alto.

Interfaz con el pasado y futuro: Una base de datos debe estar abierta a reconocer información organizada físicamente por otro software (de base de datos o no) de distinta forma a la que utiliza la base de datos. Puede existir información antes de la implantación, por la empresa, de la base de datos (actual) que es valiosa para la organización.

4.- Tipos de bases de datos

Según la variabilidad de los datos almacenados

Bases de datos estáticas: Son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

Bases de datos dinámicas: Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de un supermercado, una farmacia, un videoclub o una empresa.

Según su modelo de datos (Un modelo de datos es básicamente una "descripción" de algo conocido como *contenedor de datos* (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores):

Bases de datos jerárquicas: El sistema jerárquico más comúnmente conocido es el sistema IMS de IBM. Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se los conoce como *hojas*.

Esta base de datos tiene como objetivo establecer una jerarquía de fichas, de manera que cada ficha puede contener a su vez listas de otras fichas, y así sucesivamente. Por ejemplo, una ficha de clientes puede contener una lista de fichas de facturas, cada una de las cuales puede contener a su vez una lista de fichas de líneas de detalle que describen los servicios facturados.

En la figura siguiente tenemos una ocurrencia del tipo de registro Curso, de manera que como cabeza principal tenemos una instancia del segmento curso, de la cual dependen una o varias instancias de los segmentos Requisito y Oferta; a su vez, de Oferta dependen otros que son Profesor y Estudiante.

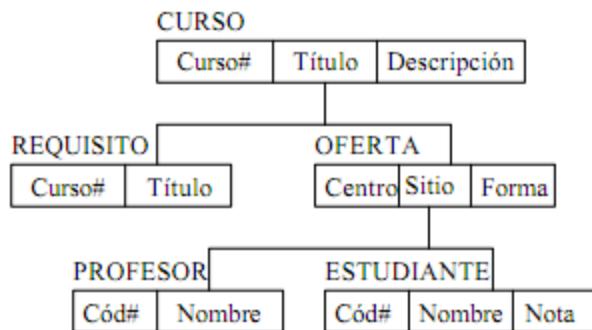


Figura 1. Ejemplo de tipo de registro.
Los tipos de segmento son CURSO, REQUISITO, OFERTA, PROFESOR, y ESTUDIANTE. CURSO es el tipo de segmento raíz.

Para movernos por un registro de estructura jerárquica lo que se hace es posicionarse inicialmente en la raíz de una instancia, e ir navegando por sus hijos según nos convenga consultando o modificando los datos pertinentes.

Una base de datos de este tipo, no permite el acceso directo a las instancias de un segmento hijo, si no es seleccionando previamente las instancias de los padres de los que depende.

Por ejemplo, no se puede seleccionar un estudiante si no es previa selección de una oferta y de un curso.

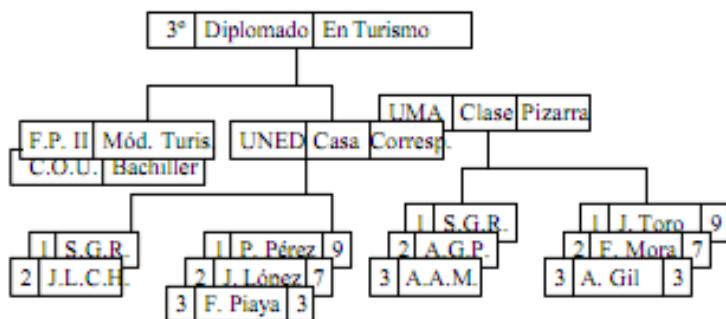


Figura 2. Instancia de un registro.

Problemas de este modelo:

- La jerarquía existente entre los tipos de objetos que se manipulan (Cursos, Estudiantes, Profesores, etc.), y las dependencias existentes, hacen que sea imposible el acceso directo a instancias de cada uno de ellos, con lo que se pierde en independencia y facilidad de uso.
- Si un mismo segmento debe participar en varios tipos de registro, deben incluirse mecanismos que eviten la repetición de datos. Es más, en el ejemplo anterior se ve que una instancia del segmento Profesor: 1 S.G.R. aparece dependiendo de la oferta de la

UNED, y de la UMA. Está claro que los datos no se deben repetir, ya que ello puede provocar que posteriormente se modifique una de las instancias pero no la otra, con la consiguiente inconsistencia entre ambas copias de los mismos datos.

Base de datos de red: Éste es un modelo ligeramente distinto del jerárquico.; su diferencia fundamental es la modificación del concepto de *nodo*: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

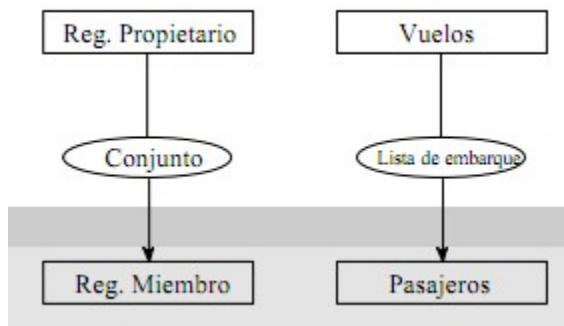
Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

Podemos considerar al modelo de bases de datos en red como de una potencia intermedia entre el jerárquico y el relacional que estudiaremos más adelante. Su estructura es parecida a la jerárquica aunque bastante más compleja, con lo que se consiguen evitar, al menos en parte, los problemas de aquél.

Los conceptos fundamentales que debe conocer el administrador para definir el esquema de una base de datos jerárquica, son los siguientes:

- Registro: Viene a ser como cada una de las fichas almacenadas en un fichero convencional.
- Campos o elementos de datos. Son cada uno de los apartados de que se compone una ficha.
- Conjunto: Es el concepto que permite relacionar entre sí tipos de registro distintos.

Para ilustrar el concepto de conjunto, supongamos que tenemos un tipo de registro de clientes, y un tipo de registro de vuelos de avión, y supongamos que queremos asociar ambas informaciones, de manera que para cada vuelo queremos saber cuáles son los pasajeros que viajan en él. La forma de hacerlo es a través de un conjunto. Un conjunto relaciona dos tipos de registro. Uno de ellos es el registro propietario del conjunto, y el otro es el miembro.



Cada tipo de conjunto, posee, a su vez, una serie de ocurrencias de conjunto, donde cada ocurrencia está formada por una instancia del tipo propietario, y una, varias o ninguna instancia del tipo miembro. Por ejemplo. Una ocurrencia de conjunto puede ser:

IB-763 Málaga Helsinki 27/8/97 17:00

33387698-K Juan Linares

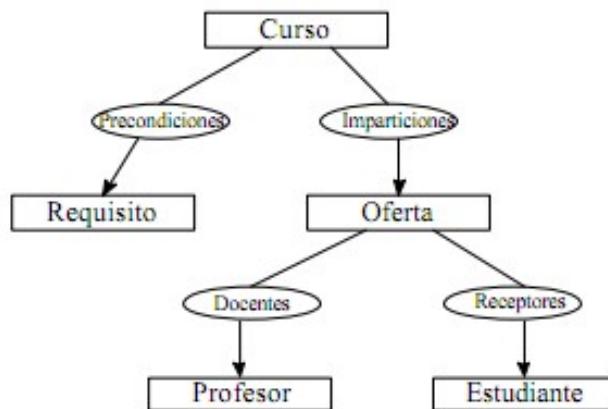
83698637-H Pedro Hernández

24885764-G Luis Caro

64653627-J Pablo Mármol

Una restricción bastante importante de este modelo, es que una ocurrencia de registro miembro puede pertenecer como máximo a una sola instancia de un determinado conjunto, aunque puede participar en varios tipos de conjuntos distintos.

Este modelo en red es más potente que el modelo jerárquico, ya que aquél puede simularse, aplicando una jerarquía de conjuntos en varios niveles. Por ejemplo, el ejemplo jerárquico del punto anterior quedaría ahora como:



Problema: una misma instancia de registro miembro no pueda aparecer en más de una instancia de conjunto, hace que sea difícil de expresar algunas situaciones.

Por ejemplo, en el caso de las lista de embarque, está claro que no sólo cada vuelo lo componen varios pasajeros, sino que, además, un mismo pasajero ha podido embarcar en varios vuelos a lo largo de su vida.

IB-763 Málaga Helsinki 27/8/97 17:00
 33387698-K Juan Linares
 83698637-H Pedro Hernández
 24885764-G Luis Caro
64653627-J Pablo Mármol

IB-722 Málaga Zurich 21/9/97 7:00
 63553572-K Alfredo Sánchez
 24746928-G Antonio Fernández
64653627-J Pablo Mármol

Bases de datos relacionales: Éste es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar F.Codd, de los laboratorios IBM en San José (California). Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL (*Structured Query Language* o *Lenguaje Estructurado de Consultas*), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Marca	Modelo	Color	Matrícula	Situación
Lamborgh.	Diablo 630	Amarillo	MA-2663-BC	En renta
Ferrari	F-40	Rojo	MA-8870-BC	Disponible
Shärrö R.	Decade	Blanco	VD-870-GTH	Disponible
De Tomaso	Pantera	Blanco	ML-7890-B	En renta
Pontiac	Trans-Am	Negro	KNIGHT	En taller
Austin M.	S3'40	Marrón	CA-5647-AB	Disponible
Jaguar	Destruktor	Verde	AD-768-TTY	En renta

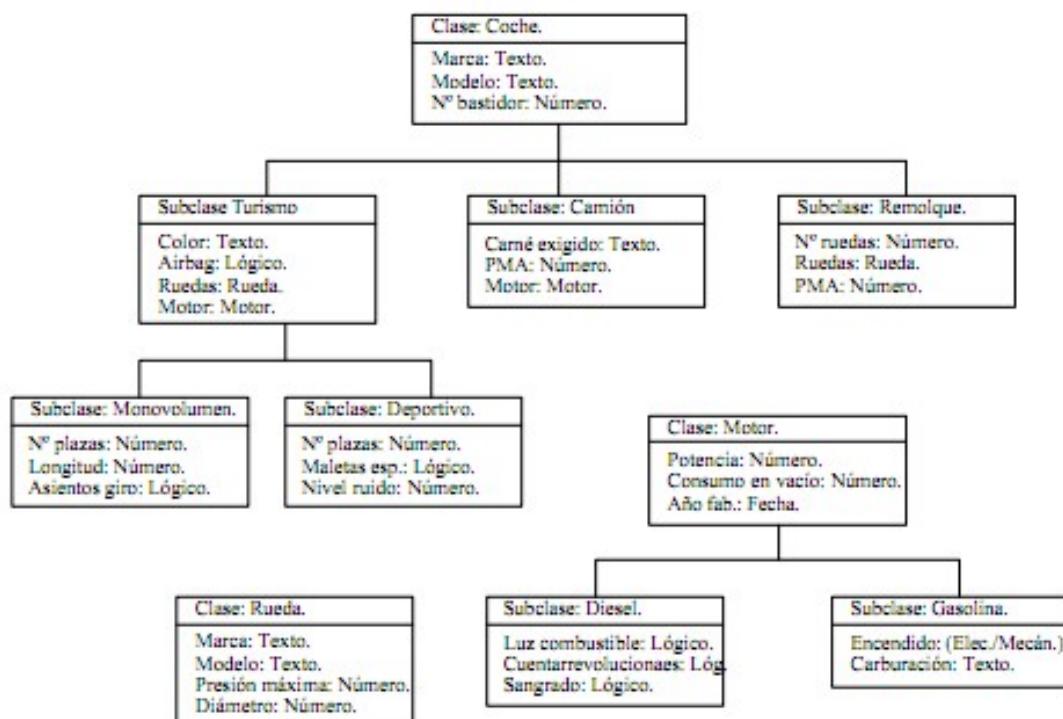
Figura 7. Ejemplo de tabla relacional.

Bases de datos orientadas a objetos: Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los *objetos* completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

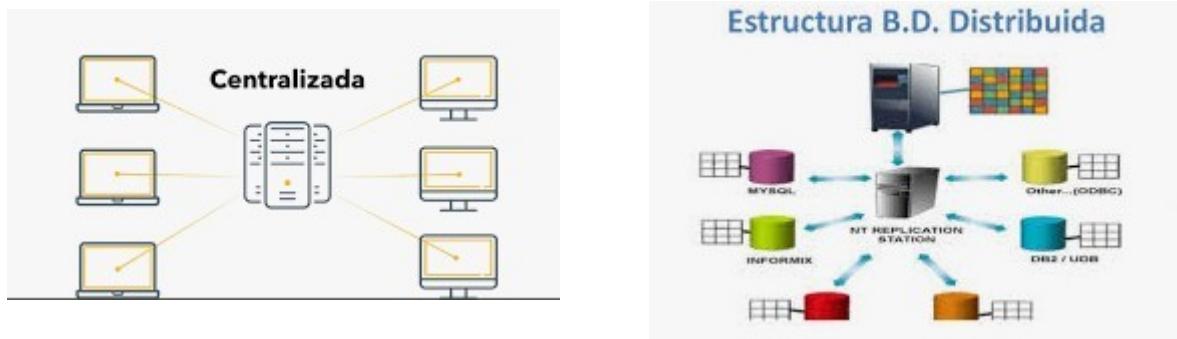
- Encapsulación: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

SQL 2003, es el estándar de SQL92 ampliado, soporta los conceptos orientados a objetos y mantiene la compatibilidad con SQL92.



Según el lugar de almacenamiento de los datos:

Bases de datos Distribuidas / Centralizadas:



En un sistema de base de datos distribuida, los datos se almacenan en varios computadores. Los computadores de un sistema distribuido se comunican entre sí a través de diversos medios de comunicación, tales como cables de alta velocidad o líneas telefónicas. No comparten la memoria principal ni el procesador.

Los procesadores de un sistema distribuido pueden variar en cuanto su tamaño y función. Pueden incluir microcomputadores pequeños, estaciones de trabajo y sistemas de computadores grandes de aplicación general. Estos procesadores reciben diferentes nombres, tales como localidades, nodos o computadores.

Un sistema distribuido de bases de datos consiste en un conjunto de localidades, cada uno de las cuales puede participar en la ejecución de transacciones (operaciones que afectan a varias tablas, por ejemplo) que accedan a datos de una o varias localidades. La diferencia principal entre los sistemas de base de datos centralizados y distribuidos es que, en los primeros, los datos residen en una sola localidad, mientras que, en los últimos, se encuentran en varias localidades.

Características:

Un sistema distribuido de base de datos consiste en un conjunto de localidades, cada una de las cuales mantiene un sistema de base de datos local. Cada localidad puede procesar transacciones locales, o bien transacciones globales entre varias localidades, requiriendo para ello comunicación entre ellas.

Las localidades pueden conectarse físicamente de diversas formas, las principales son:

- Red totalmente conectada
- Red prácticamente conectada
- Red con estructura de árbol
- Red de estrella
- Red de anillo

Las diferencias principales entre estas configuraciones son:

- Coste de instalación: El coste de conectar físicamente las localidades del sistema
- Coste de comunicación: El coste en tiempo y dinero que implica enviar un mensaje desde la localidad A a la B
- Fiabilidad: La frecuencia con que falla una línea de comunicación o una localidad.
- Disponibilidad: La posibilidad de acceder a información a pesar de fallos en algunas localidades o líneas de comunicación.

Las localidades pueden estar dispersas, ya sea por un área geográfica extensa (a lo largo de un país), llamadas redes de larga distancia; o en un área reducida (en un mismo edificio), llamadas redes de área local. Para las primeras se utilizan en la comunicación líneas telefónicas, conexiones de microondas y canales de satélites; mientras que para las segundas se utiliza cables coaxiales de banda base o banda ancha y fibra óptica.

Consideraciones al distribuir la base de datos:

Existen varias razones para construir sistemas distribuidos de bases de datos que incluyen compartir la información, fiabilidad y disponibilidad y agilizar el procesamiento de las consultas. Pero también tiene sus desventajas, como desarrollos de software más costosos, mayor posibilidad de errores y costos extras de procesamiento.

VENTAJAS Y DESVENTAJAS:

Ventajas de la distribución de datos:

La principal ventaja de los sistemas distribuidos es la capacidad de compartir y acceder a la información de una forma fiable y eficaz.

Utilización compartida de los datos y distribución del control:

La ventaja principal de compartir los datos por medio de la distribución es que cada localidad pueda controlar hasta cierto punto los datos almacenados localmente. En un sistema centralizado, el administrador de base de datos de la localidad central controla la base de datos. En un sistema distribuido existe un administrador global de la base de datos que se encarga de todo el sistema. Parte de esta responsabilidad se delega al administrador de base de datos de cada localidad. Dependiendo del diseño del sistema distribuido, cada administrador local podrá tener un grado de autonomía diferente, que se conoce como autonomía local. La posibilidad de contar con autonomía local es en muchos casos una ventaja importante de las bases de datos distribuidas.

Fiabilidad y disponibilidad:

Si se produce un fallo en una localidad de un sistema distribuido, es posible que las demás localidades puedan seguir trabajando. En particular, si los datos se repiten en varias localidades, una transacción que requiere un dato específico puede encontrarlo en más de una localidad. Así, el fallo de una localidad no implica necesariamente la desactivación del sistema.

El sistema debe detectar cuando falla una localidad y tomar las medidas necesarias para recuperarse del fallo. El sistema no debe seguir utilizando la localidad que falló. Por último, cuando se recupere o repare esta localidad, debe contarse con mecanismos para reintegrarla al sistema con el mínimo de complicaciones.

La disponibilidad es fundamental para los sistemas de bases de datos que se utilizan en aplicaciones de tiempo real. Por ejemplo, si una línea aérea no puede tener acceso a la información, es posible que pierda clientes a favor de la competencia.

Agilización del procesamiento de consultas:

Si una consulta comprende datos de varias localidades, puede ser posible dividir la consulta en varias sub-consultas que se ejecuten en paralelo en distintas localidades. En los casos en que hay repetición de los datos, el sistema puede pasar la consulta a las localidades más ligeras de carga.

Desventajas de la distribución de los datos:

La desventaja principal de los sistemas distribuidos es la mayor complejidad que se requiere para garantizar una coordinación adecuada entre localidades.

El aumento de la complejidad se refleja en:

- Coste del desarrollo de software: es más difícil estructurar un sistema de bases de datos distribuidos y por tanto su coste es menor.
- Mayor tiempo extra de procesamiento: El intercambio de mensajes y los cálculos adicionales son una forma de tiempo extra que no existe en los sistemas centralizados.

Ejercicio: pensar en algún ejemplo de sistema distribuido.

Fragmentación

Fragmentación es la descomposición o partición de una tabla en pedazos llamados *fragmentos*.

La fragmentación básicamente se puede hacer de dos formas:

- **Fragmentación Horizontal.** selecciona registros completos de una relación
- **Fragmentación Vertical.** selecciona columnas completas de una relación

Reglas a Cumplir por Fragmentación

- **Condición de Compleción.**
Todos los datos de la relación global deberán ser mapeados a algún fragmento.
- **Condición de Reconstrucción.**
Deberá ser siempre posible reconstruir la relación global a partir de sus fragmentos.
- **Condición de Conjuntos Disjuntos.**
Es conveniente que los fragmentos sean disjuntos.

Tipos de fragmentación:**Fragmentación Horizontal**

Definir *fragmentos horizontales* se hace a través del *operador de selección del álgebra relacional* operando sobre una relación global.

Ejemplo: s		
SNUM	NOMBRE	CD
S1	X	L
S2	Y	L
S3	Z	P
S4	A	P
S5	B	L

Una posible fragmentación Horizontal de esta tabla sería la sig:

FH1= $s \text{ WHERE } CD='L'$
FH2= $s \text{ WHERE } CD='P'$

FH1		
SNUM	NOMBRE	CD
S1	X	L
S2	Y	L
S5	B	L

FH2		
SNUM	NOMBRE	CD
S3	Z	P
S4	A	P

En general una fragmentación horizontal es correcta, si cumple que:

- El conjunto de calificaciones mapea todo el dominio del atributo(s) bajo el cual se hace la calificación.
- Si siempre es posible reconstruir la tabla global por medio del operador **UNION** del álgebra relacional:

$$R = F1 \text{ UNION } F2 \text{ UNION } \dots \text{ UNION } Fn$$
- Si todas las calificaciones de los fragmentos son mutuamente exclusivas, es decir, si al aplicar las calificaciones se producen fragmentos que al *intersectarlos* generan un conjunto vacío.

$$F = F1 \text{ INTERSECT } F2 \text{ INTERSECT } \dots \text{ INTERSECT } Fn$$

Fragmentación Vertical

Definir *fragmentos verticales* se hace a través del *operador de proyección del álgebra relacional* operando sobre una relación global.

Ejemplo:

E					
EMP#	NOMBRE	SALARIO	IMPTO.	#JEFE	DEPT#
e1	X	1000	100	J1	D1
e2	Y	1500	300	J1	D1
e3	Z	500	20	J2	D2
e4	A	4000	1000	J3	D3
e5	B	2000	350	J2	D2

EMP#	NOMBRE	#JEFE	DEPT#
e1	X	J1	D1
e2	Y	J1	D1
e3	Z	J2	D2
e4	A	J3	D3
e5	B	J2	D2

EMP#	SALARIO	IMPTO
e1	1000	100
e2	1500	300
e3	500	20
e4	4000	1000
e5	2000	350

$FV1 = E[EMP\#, NOMBRE, \#, JEFE, DEPT\#]$ $FV2 = E[EMP\#, SALARIO, IMPTO]$

Una característica importante de la fragmentación vertical, es que *todos los fragmentos deben incluir la clave primaria* de la relación global.

La razón es que si no incluimos la clave primaria no es posible reconstruir la relación original.

Para reconstruir la relación original debemos realizar un **JOIN** de todos los fragmentos.

$R = F1 \text{ JOIN } F2 \text{ JOIN } \dots \text{ JOIN } F3$

En fragmentación vertical no se cumple que los fragmentos sean disjuntos (la llave está repetida en todos los fragmentos).

Fragmentación Híbrida

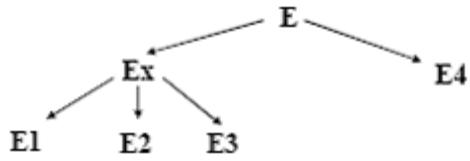
Consiste en *aplicar las operaciones de fragmentación vistas anteriormente de manera recursiva*, satisfaciendo las condiciones de *compleción* cada vez que se realiza la fragmentación.

La reconstrucción puede ser obtenida aplicando las reglas de reconstrucción en orden inverso.

De esta forma podemos fragmentar nuestra tabla global en los pedazos que queramos y como queramos.

Ejemplo: Considere la relación de empleado (E).

Una posible fragmentación híbrida sería:



E1 = Ex[Emp#, Nombre, #Jefe,Dept#] Where Dept# <10

**E2 = Ex[Emp#, Nombre, #Jefe,Dept#] Where Dept# >=10 And
Dept# <=20**

E3 = Ex[Emp#, Nombre, #Jefe,Dept#] Where Dept# >20

E4 = E[Emp#, Salario, Impto]

Ejercicio: crea una tabla de empleados con datos y indica como quedarían al realizar la fragmentación anterior.

5.- Diferentes visiones de los datos en las bases de datos

Para que una base de datos pueda satisfacer las características antes señaladas, y otras más, es necesario que los usuarios de la misma tengan una visión abstracta de los datos almacenados en la misma. Es decir, el usuario, a diferencia de lo que ocurría con el uso de las organizaciones clásicas de almacenamiento de la información, no tiene necesidad de conocer cómo se organizan los datos físicamente en la base de datos.

El usuario experto conoce las características del problema que representa la base de datos. El usuario conoce la organización todo o parte del sistema, la información que se maneja, las relaciones existentes entre esta información, cómo debe ser tratada esta información y en qué tiempo. En definitiva, el usuario conoce el dominio del problema que va a ser tratado con una base de datos. Si es ésta la visión de los datos que la base de datos presenta al usuario, el usuario podrá utilizar el sistema e integrarse en él adecuadamente.

Si por el contrario, la base de datos presenta al usuario la visión de cómo está organizada la información físicamente, tanto los datos como las múltiples relaciones que pueden existir entre ellos, éste no reconocería el problema de la organización que está tratando si no tiene una alta formación para reconocer en ocasiones estructuras físicas muy complejas.

Por ejemplo, a un usuario no se le puede decir que para obtener la información que necesita, lea los datos de una estructura, después vaya a otra estructura distinta y las relacione con un valor a través de un puntero,....

Dependiendo de quién acceda o use la base de datos, ésta debe presentarle una visión de los datos que sea capaz de reconocer, interpretar y manejar. Además, si una base de datos debe satisfacer las características antes expuestas, la organización física de los datos debe ser lo más independiente posible de los procedimientos que manejan la información y de los posibles cambios que surjan en el dominio del problema.

Se puede entonces hablar de que existen tres visiones de los datos en una base de datos:

Visión externa: Es la visión de los datos que tienen los usuarios finales de una base de datos. Un usuario (un operador de terminal) trata sólo una visión parcial de la información, sólo aquella que interviene en el dominio de actividad (el subsistema de la organización en el que interviene). Este usuario debe "ver" la información que maneja como un registro, una ficha de datos con independencia de a qué entidad pertenecen los ítems de datos, correspondientes a ese registro, en el dominio del problema (sistema) y en qué relaciones se ven implicados esos datos.

Estas "*visiones particulares*" de los usuarios son proporcionadas por los procedimientos o programas de aplicación que sólo manejan parte de la información de la base de datos.

Visión conceptual: Es la visión o representación del problema tal y como éste se presenta en el mundo real. Una base de datos representa la información que es observada en el mundo real con respecto a un determinado problema. En esta observación, en el análisis del problema, se determinan los objetos o entidades que intervienen en el mismo, las propiedades o características de estas entidades y las relaciones o dependencias que existen entre ellos.

La visión conceptual de una base de datos es una representación abstracta del problema e independiente, en principio, de cómo va a ser tratada esta información, de qué visiones externas pueda tener y de cómo esta información pueda ser almacenada físicamente. Así, la visión conceptual de una base de datos no cambia a no ser que cambie la naturaleza del problema.

Visión Física: La visión física de una base de datos es la representación de cómo la información es almacenada en los dispositivos de almacenamiento. Esta visión describe las estructuras u organizaciones físicas, dispositivos, volúmenes, ficheros, tipos de datos, punteros, etc., estructuras de mayor o menor complejidad que representan el dominio del problema de una forma entendible por el sistema informática.

Si los usuarios de las bases de datos pueden percibir tres visiones diferentes de los datos, es debido a que en una base de datos existen, al menos, tres formas diferentes de descripción de la información que almacena. Pero, como el dominio del problema que representa una base de datos es el mismo, entonces estas tres formas de descripción son, en realidad, tres niveles de abstracción diferentes que describen un mismo problema (Ver Figura 1.1)

En cada nivel se describen aquellos objetos de interés que pueden ser entendidos por los usuarios, de ese nivel, de la base de datos. Así, el usuario final sólo entiende de registros (visión o nivel de abstracción externo) mientras que el diseñador o analista de sistemas sólo entiende de tipos de entidades o clases de objetos que intervienen en el dominio del problema que la organización desea que se trate mediante una base de datos, de las relaciones existentes entre ellas y de los procedimientos que son llevados a cabo en la organización para la solución del problema que se está tratando (nivel de abstracción conceptual) y, por otro lado, es el administrador de la base de datos el encargado de describir el nivel físico para determinar aquella organización física que pueda garantizar el desempeño óptimo del sistema (nivel de abstracción físico o interno)

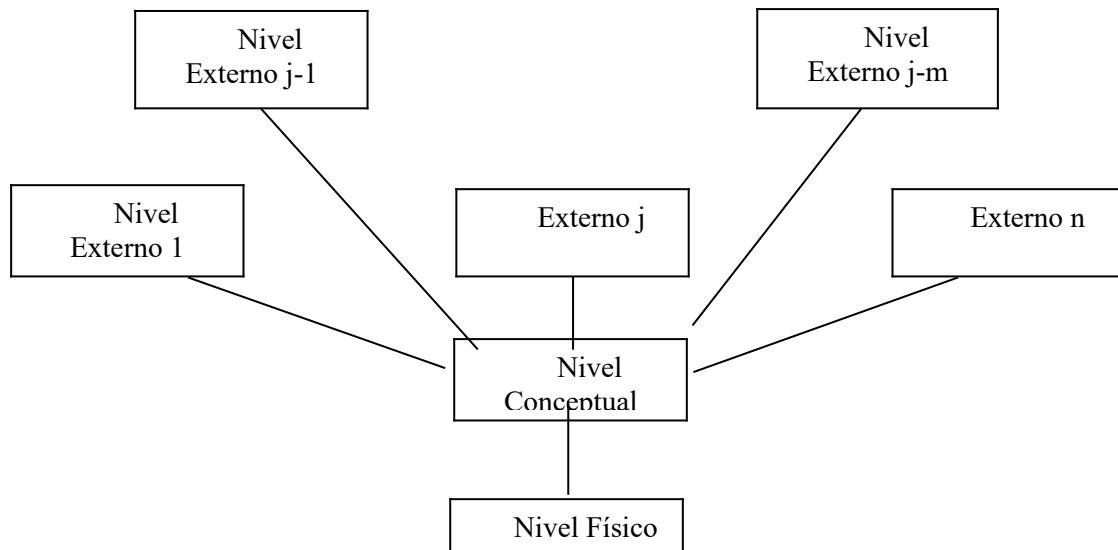


Figura 1.1: Tres visiones diferentes en las bases de datos

La descripción de los datos a estos tres niveles de abstracción diferentes garantiza la independencia de los datos, uno de los objetivos principales de las bases de datos.

Es decir:

- Que pueda ser modificada la organización física (sin que por ello haya cambiado o deba cambiar la descripción conceptual) de los datos, y sin que por ello tengan que ser modificados los programas de aplicación (nivel de abstracción o nivel externo) que manipula esta información.
- Que pueda ser modificada la representación conceptual del problema que está siendo representado en la base de datos (debido a la consideración de nuevas entidades, o relaciones, o cambios en las características de las mismas) y sin que por ello tenga que ser modificada la estructura física de la información (nivel interno).
- Evidentemente, las visiones externas pueden cambiar conforme nuevos requerimientos o necesidades funcionales o de operación son incorporados al dominio del problema por la organización y/o su entorno, y sin que por ello deba ser modificada ninguna de las descripciones de los datos a ninguno de los restantes niveles de abstracción.

6.-Bases de Datos (BD) y Sistemas de Gestión de Bases de Datos (SGBD)

Existen muchas definiciones de lo que es una base de datos.

Una **Base de Datos** es una colección de archivos relacionados que almacenan tanto una representación abstracta del dominio de un problema del mundo real cuyo manejo resulta de interés para una organización, como los datos correspondientes a la información acerca del mismo. Tanto la representación como los datos están sujetos a una serie de restricciones, las cuales forman parte del dominio del problema y cuya descripción está también almacenada en esos ficheros.

De la que podemos extraer las siguientes consideraciones:

Se trata de una *colección de archivos relacionados* es decir, a diferencia de los sistemas clásicos que manejan organizaciones clásicas de almacenamiento, en una base de datos los archivos no son independientes entre sí, la base de datos puede ser vista como un único depósito en el cual se almacena toda la información correspondiente al dominio de un problema.

En estos archivos se encuentra almacenada tanto la *representación abstracta* del dominio del problema: es decir, la visión física, lógica y cada una de las visiones externas de la información, como *los datos* conocidos acerca del mismo en un momento dado.

El que tanto la *representación como los datos* están sujetos a una serie de restricciones implica:

- Que las restricciones innatas al problema están representadas. Restricciones acerca de las propiedades de las entidades, datos y relaciones existentes en el dominio del problema. Por ejemplo, en caso de representar la gestión de un colegio de la E.S.O., existe la restricción de que la edad de los niños no puede superar los 18 años. Esta es una restricción propia del dominio.
- Que el acceso a la información almacenada está sujeto a una serie de restricciones que garantizan la integridad de la misma. Estas restricciones impiden que algún procedimiento viole las reglas que vinculan los datos entre los diferentes niveles de representación. Por ejemplo, yo no puedo asignar un curso a un alumno sin que este esté dado de alta en el centro.

Es importante conocer la diferencia entre lo que es una base de datos y lo que es un *Sistema de Gestión de Bases de Datos*, términos que se confunden muy a menudo cuando se está trabajando con la información haciendo uso de esta tecnología.

Cuando se habla de bases de datos se habla de información que está almacenada cumpliendo toda una serie de características y restricciones como las que se han expuesto anteriormente.

Pero para que la información pueda ser almacenada como se ha descrito y el acceso a la misma satisfaga las características exigidas a una base de datos para ser denominada como tal, es necesario que exista una serie de procedimientos (un sistema software) que sea capaz de llevar a cabo tal labor. A este sistema software es a lo que se le denomina *Sistema de Gestión de Bases de Datos*.

Así, un SGBD es una colección de programas de aplicación que proporcionan al usuario de la base de datos los medios necesarios para realizar las siguientes tareas:

- Definición de los datos a los distintos niveles de abstracción (físico, lógico y externo).
- Manipulación de los datos en la base de datos. Es decir, la inserción, modificación, borrado y acceso o consulta de los mismos.
- Mantenimiento de la integridad de la base de datos. Integridad en cuanto a los datos en sí, sus valores y las relaciones entre ellos.
- Control de la privacidad y seguridad de los datos en la base de datos.

Y, en definitiva, los medios necesarios para el establecimiento de toda aquellas características exigibles a una base de datos.

6.1-Componentes de los SGBD

Cuando se utiliza el término de "componentes" de un *SGBD* se está realizando una, y quizás peligrosa, generalización, puesto que estos componentes son muy variados. Así, un *SGBD* cuenta tanto con herramientas software como con personal humano especializado en la realización de las tareas y acciones necesarias para la gestión adecuada de la información.

6.1.1 El Lenguaje de definición de datos

Si para garantizar la independencia de los datos es necesaria la definición de éstos a diferentes niveles de abstracción es, por tanto, necesario que el *SGBD* cuente con un componente que permita la realización de esta tarea. El lenguaje de definición de los datos -*Data Definition Language (DDL)*- es un lenguaje artificial basado en un determinado modelo de datos que permite la representación lógica de los datos.

La representación de los datos obtenida en este proceso de compilación es almacenada en otro componente del *SGBD* denominado *Diccionario de Datos*.

6.1.2 El Lenguaje de definición del almacenamiento de los datos

En la mayoría de los *SGBD* el mismo lenguaje DDL permite la definición de los datos en el nivel de representación físico, si bien en otros es un subcomponente de éste denominado lenguaje de definición del almacenamiento de los datos -*Data Storage Definition Language (DSDL)*-.

En cualquier caso, haciendo uso del *DDL* o un subcomponente de éste, el *DSDL*, se definen los datos correspondientes al dominio de un problema a los dos niveles de abstracción (conceptual y físico), y a esta definición de los datos se le denomina *Esquema de la Base de Datos*.

El esquema de la base de datos es una representación de los datos correspondientes al dominio de un problema mediante un lenguaje de definición de datos, el cual está basado en un modelo de datos.

Pongamos por ejemplo lo siguiente:

Queremos modelizar el que un profesor imparte en una serie de cursos. Este sería el problema del mundo real que queremos modelizar para ser gestionado por una base de datos. A partir de aquí

- crearíamos un modelo de datos:
Profesor-----imparte-----cursos
- crearíamos los correspondientes esquemas a cada uno de los niveles

Esquema global (Tipo de objeto)	Esquema Interno	Esquema externo
Curso	Curso	En Oracle Forms
Código Carácter(5)	Cod_curso Byte(3)	
Nombre Carácter(50)	Nombre Byte(50)	Nombre Varchar2(50)
Num_hours Numérico(3)	Num_hours Byte(2)	Horas Number(3,0)
Descripción Carácter variable(200)	Descripción Byte(200)	Descripción Varchar2(200)
Profesor	Profesor	

El DDL nos permite definir el esquema conceptual y el DSDL el esquema físico.

En el esquema estarán definidas:

- Las características del problema a un nivel de descripción lógico. Esta definición no variará a no ser que cambie el problema.
- Cada una de las clases de objetos, y sus propiedades, que formen parte del dominio del problema o sistema que se desea tratar con el SGBD.
- Cada una de las relaciones, y sus propiedades, existentes entre estas clases de objetos.
- Todas aquellas restricciones concernientes tanto a las clases de objetos y sus propiedades como con las relaciones entre ellos.
- Las características del problema desde un punto de vista físico u operacional. Esta descripción puede variar con el tiempo en base a las necesidades o requerimientos operacionales, y contemplará, por ejemplo:
 - Las unidades físicas en las cuales los datos van a ser almacenados.
 - Los volúmenes y archivos utilizados.
 - Las características físicas y lógicas de los medios de almacenamiento y métodos de acceso a la información: clusters, bloques, índices, tablas hash, etc.

Además del DSDL, el DDL cuenta con un sublenguaje encargado del control y seguridad de los datos. Este sublenguaje se denomina lenguaje de control de datos -*Data Control Language (DCL)*- y permite el control del acceso a la información almacenada en el diccionario de datos (definición de privilegios y tipos de acceso), así como el control de la seguridad de los datos.

6.1.3 El Lenguaje de manipulación de datos

Otro componente esencial de los *SGBD* es el lenguaje de manipulación de los datos -*Data Manipulation Language (DML)*-. *El DML* es un lenguaje artificial mediante el cual se realizan dos funciones bien diferentes en la gestión de los datos:

1. La definición del nivel externo o de usuario de los datos.
2. La manipulación de los datos; es decir, la inserción, borrado, modificación y recuperación de los datos almacenados en la base de datos.

Al igual que el *DDL*, *el DML* está basado en un modelo de datos y, por tanto, los *SGBD* basados en distintos modelos de datos tienen diferente *DML*. Se trata también de un lenguaje basado en una gramática completa, sencilla y, generalmente, fácil de entender por usuarios no expertos.

Dependiendo del modelo de datos en el cual se soportan y, por supuesto, del *SGBD*, existen dos tipos de *DML*:

Procedimentales: los cuales requieren que en las sentencias del lenguaje se especifique qué datos se van a manipular y qué acciones/operaciones deben realizarse para ello.

No Procedimentales: los cuales sólo requieren que en las sentencias del lenguaje se especifique qué datos se van a manipular, siendo el propio *DML* el encargado de determinar los procedimientos más efectivos para ello.

Naturalmente, estos últimos son mucho más fáciles de entender y manejar por usuarios no expertos, si bien cuando los requerimientos en cuanto al tiempo de respuesta del *SGBD* en los procesos de manipulación de los datos son importantes, será necesaria la modificación del código que generan estos lenguajes para asegurar un desempeño adecuado del sistema.

Como se ha comentado anteriormente, el *DML* tiene también la función de describir la visión externa de los datos. Efectivamente, mediante el *DML* se definen *las "vistas"* o visiones parciales que los usuarios tienen del esquema de la base de datos definido mediante el *DDL*. Estas vistas de los datos son denominadas Subesquemas y pueden realizarse de varias formas:

- Haciendo uso única y exclusivamente del *DML*. Así, con sentencias propias de este lenguaje se definen distintas visiones parciales -orientadas al usuario final- del esquema de la base de datos. Estas visiones parciales (vistas) son, generalmente, almacenadas en el diccionario de datos.
- Haciendo uso de un lenguaje huésped (host) como *C*, *COBOL*, *FORTRAN*, etc., mediante el cual se realizan los programas de aplicación que permiten al usuario manipular los datos de la base de datos. En el código fuente de estos programas están presentes sentencias del *DML*, que son las encargadas de estos procesos, mientras que las sentencias realizadas en el lenguaje huésped tienen como objetivo el control del flujo de la información y la interfaz de usuario.

En este caso, los programas fuentes deben de ser precompilados, antes de ser compilados, con el compilador correspondiente, para generar el código máquina, y convertir las sentencias *DML* inmersas en el código fuente a un código entendible por el compilador del lenguaje huésped.

6.1.4 El diccionario de datos

El diccionario de datos es uno o un conjunto de archivos que contienen información acerca de los datos que pueden ser almacenados en la base de datos. Se trata de una "metabase de datos"; es decir, una base de datos que contiene información sobre otra base de datos.

En el diccionario de datos se almacenan todas las definiciones realizadas por *el DDL* sobre el problema que va a ser tratado por el *SGBD* y, algunas (las que se deseen) de las realizadas por el *DML*. Así, en el diccionario de datos se encuentra almacenado:

- El esquema lógico de la base de datos.
- El esquema físico de la base de datos.
- Los subesquemas de la base de datos.

Es decir, la representación de los datos a los tres niveles de abstracción. Pero además, en el diccionario de datos se encuentra mucha más información almacenada; información correspondiente con:

- Las restricciones de privacidad y acceso a los datos almacenados en la base de datos. Estas restricciones han sido definidas haciendo uso del *DDL* y/o su sublenguaje, el *DCL*.
- Las reglas, normas o restricciones referentes a la seguridad de los datos.
- Otra serie de información que permite garantizar la integridad de los datos almacenados en la base de datos.

6.1.5 El gestor de la base de datos

El gestor de la base de datos, a veces denominado monitor, es un componente software encargado de garantizar el correcto, seguro, íntegro y eficiente acceso y almacenamiento de los datos. Este componente es el encargado de proporcionar una interfaz entre los datos almacenados y los programas de aplicación que los manejan.

El que este componente realice sus funciones asignadas correctamente dependerá de muchos factores, entre los que se pueden citar: el volumen de la base de datos, las estructuras físicas definidas para el almacenamiento de los mismos, los procedimientos desarrollados para la manipulación de los datos, las características del hardware y la calidad del propio gestor.

Puede verse al gestor de la base de datos como un intérprete entre el usuario (de cualquier tipo) y los datos. Toda operación que se quiera realizar "contra la base de datos debe ser previamente permitida por el gestor de la misma, el cual, una vez interpretada y validada, o bien realiza la operación devolviendo el resultado de la misma al programa/procedimiento que la solicitó, o bien la rechaza. Así, el gestor de la base de datos es el responsable de:

- Garantizar la privacidad de los datos, permitiendo sólo el acceso a los mismos a los usuarios autorizados.
- Garantizar la seguridad de los datos, realizando los procedimientos necesarios para que los datos puedan ser recuperados tras un fallo que ocasione una pérdida o deterioro temporal de los mismos.
- Garantizar la integridad de los datos, gestionando que los datos que se almacenan en la base de datos satisfagan las restricciones definidas en el esquema de la misma.

- Garantizar el acceso concurrente a la base de datos de forma que varios usuarios puedan acceder al mismo o distinto dato sin que ello ocasione un pérdida de la integridad de la base de datos.
- Interaccionar con el sistema operativo y, en particular, con el gestor de archivos del mismo, de forma que los procedimientos DML puedan ser entendidos por el sistema operativo para el correcto almacenamiento y recuperación de la información. Para ello, el gestor de la base de datos cuenta con un subcomponente denominado *procesador de consultas*.

6.1.6 El administrador de la base de datos

Otro de los componentes de los *SGBD* es el administrador de la base de datos -*Data Base Administrator (DBA)*-. Se trata de un componente humano de suma importancia en el resultado que el uso de las bases de datos va a tener en la resolución de un determinado problema. El *DBA* tiene una serie de responsabilidades en cuanto a la definición, administración, seguridad, privacidad e integridad de la información que es tratada, así como en el desempeño del *SGBD* en el procesamiento de la misma.

Entre las tareas asignadas al *DBA* se encuentran:

- La definición del esquema lógico de la base de datos. Es decir, la codificación mediante sentencias del *DDL* del conjunto de definiciones que representan las características del problema que va a ser tratado haciendo uso del *SGBD*. En esta definición se incluyen aquellas especificaciones necesarias para que el *SGBD* pueda mantener la integridad de los datos almacenados en la base de datos.
- La definición del esquema físico de la base de datos. Es decir, la especificación de las estructuras de almacenamiento y los métodos de acceso a la información almacenada en los dispositivos físicos de almacenamiento. Esta definición se realiza haciendo uso del *DSDL* (o el propio *DDL*) mediante un conjunto de sentencias que son compiladas y traducidas a una especificación entendible por la máquina que es almacenada en el diccionario de datos junto con el esquema canónico.
- La definición de los subesquemas o visiones externas o de usuario de la base de datos. Aquellas vistas parciales de la base de datos que son almacenadas en el diccionario de datos son definidas por el *DBA*, el cual es el único que tiene acceso y, por tanto, privilegios suficientes para la gestión de este componente.
- El control de la privacidad de los datos, mediante la concesión de privilegios a usuarios o grupos de éstos para el acceso a la información almacenada en la base de datos. Esta tarea se realiza en base al esquema de la base de datos y en base a las operaciones básicas que pueden realizarse con los datos (consulta, inserción, modificación y borrado), concediéndose privilegios para una o varias de estas acciones a grupos de datos definidos en el esquema de la base de datos.
- Mantenimiento de los esquemas. Así, el *DBA* es el responsable de:
 - Introducir las modificaciones necesarias en el esquema lógico; modificaciones producidas por un cambio en el problema tratado por el *SGBD* o una ampliación del mismo.
 - Introducir las modificaciones necesarias en la representación física de los datos, de forma que esta representación evolucione paralelamente a la extensión de la base de datos y a la introducción de nuevos requerimientos funcionales y/o de desempeño.
 - Introducir las modificaciones y nuevas definiciones de los subesquemas o visiones externas o de usuario, aportando una explotación efectiva de la base de datos.
- La especificación de los procedimientos necesarios para el mantenimiento de la seguridad de los datos almacenados en la base de datos. Es decir, cuando, cómo y de qué forma se deben realizar y definir los procesos que garanticen que los datos puedan ser recuperados aun después de un fallo que dé lugar a una pérdida temporal de los mismos.

6.1.7 Los usuarios de la base de datos

Tradicionalmente se ha considerado a los usuarios de las bases de datos como un componente más de los *SGBD*, posiblemente debido a que estos sistemas fueron de los primeros en considerar a éstos como una parte importante del correcto, adecuado y necesario funcionamiento de los mismos.

Naturalmente, en un sistema de esta complejidad existen muchos tipos de usuarios que acceden e interactúan con el mismo, así se pueden considerar:

Usuarios terminales: aquellos usuarios que, a través de programas de aplicación, interactúan con la base de datos. Son usuarios no especializados que tienen la visión del problema que les proporcionan las visiones externas o subesquemas que utilizan los programas de aplicación a los cuales tienen privilegios de ejecución.

Usuarios técnicos: aquellos que desarrollan los programas de aplicación que van a ser utilizados por los usuarios terminales de la base de datos. Son profesionales informáticos que, haciendo uso de lenguajes de tercera o cuarta generación, preparan procedimientos que son invocados desde una interfaz orientada al usuario para gestionar las operaciones necesarias en la gestión del problema.

Usuarios especializados: aquellos que utilizan el *SGBD* como una herramienta en el desarrollo de otros sistemas más o menos complejos. Estos usuarios necesitan una buena gestión de la información que es procesada por otro sistema comercial o desarrollado por ellos y, por tanto, utilizan al *SGBD* como un submódulo de sus sistemas particulares, interactuando con él en la medida que le es necesario. Por ejemplo, en el desarrollo de sistemas expertos, el *SGBD* puede ser el encargado de la gestión de la base y metabase de conocimiento del mismo.

Usuarios críticos: cuya denominación, aunque algo "ruda", consideramos la más acertada. Estos usuarios pueden tener desde mucho, hasta ningún conocimiento técnico de la tecnología de base de datos, y/o del *SGBD* en el cual se soporta la base de datos con la cual interactúan pero, independientemente de ello, requieren de la base de datos información en un formato y detalle y bajo unos requerimientos que generalmente no están previstos de antemano (en el proceso de análisis del problema y diseño del esquema) y en un tiempo mínimo.

Se trata de aquellos usuarios gerencias o pertenecientes al staff de las empresas en las cuales se ha instalado la base de datos, los cuales, en base a expectativas de gestión, administración, mercado, marketing o simplemente por interés personal, realizan consultas no previstas sobre la información almacenada en la base de datos.

Como hoy en día la mayoría de los *SGBD* cuentan con un lenguaje de cuarta generación integrado en su producto, las interacciones de estos usuarios contra la base de datos pueden realizarse sin que los usuarios técnicos tengan que estar preparándoles continuamente procedimientos nuevos, y de un corto tiempo de vida, para la realización de las mismas. Sin embargo, al tratarse de consultas complejas, y debido a que el lenguaje de cuarta generación no genera un código objeto efectivo para interactuar con la base de datos, y al no estar previstas las estructuras físicas para este tipo de interrogantes, el desempeño de estos procesos es bajo, ocasionando la "crítica" generalizada de estos usuarios, los cuales son, generalmente, los que tienen poder de decisión sobre la inversión y contratación en la empresa (de ahí el nombre que los autores han dado a los mismos).

7.-SGBD Libres-Propietarios

La elección de una base de datos ha sido por mucho tiempo un punto de discusión necesaria dentro de los departamentos de sistemas de las empresas, debido a que dicha decisión acarrea muchas importantes consecuencias para la organización, a veces de manera permanente. Para aplicaciones de misión crítica esta discusión es aún mucho más importante, y los factores que inciden dentro de la decisión de adoptar una u otra plataforma son muy variados y complejos a la vez.

La característica que más se nota en un sistema de base de datos es la velocidad de procesamiento, pero en la medida en que aumenta la complejidad de un proyecto informático otras características se hacen necesarias.

Características de las bases de datos libres

Pueden ser usadas sin restricciones de ningún tipo.

Pueden ser estudiadas (debe permitir el acceso a su código fuente).

Pueden ser redistribuidas (la copia no constituye delito).

Pueden ser modificadas y es permitido distribuirlas con las modificaciones.

Ejemplos de gestores de base de datos libres

Firebird
BDB
MySQL
PostgreSQL
Sqlite

Ejemplos de gestores de base de datos propietarios

dBase
FileMaker
Fox Pro
IBM DB2 Universal Database (DB2 UDB)
IBM Informix
MAGIC
Microsoft SQL Server
Open Access
Oracle
Paradox
PervasiveSQL
Progress (DBMS)
Sybase ASE
Sybase ASA
Sybase IQ
WindowBase

8.-Otros sistemas de almacenamiento: XML, servicio de directorios,... (Información obtenida del profesor Juan Marcos Filgueira Gomis)

Además de almacenar datos en archivos y bases de datos, existen otras formas de archivar y administrar información. Dos de los más notables son el formato XML y los servicios de directorio LDAP, que también almacenan información de forma estructurada.

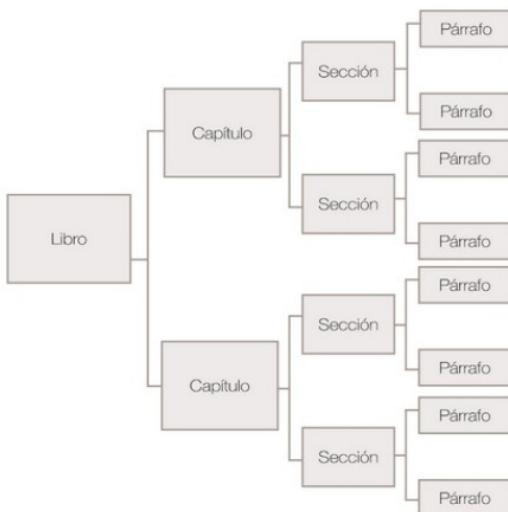
8.1.-Sistemas de almacenamiento XML

El estándar XML se utiliza para crear documentos que contienen datos estructurados y para almacenar información de archivos de comunicación, comunicar datos entre aplicaciones web e información utilizada por los propios programas.

```
<?xml version="1.0" encoding="UTF-8"?>
- <biblioteca>
  - <libro>
    <título>La vida está en otra parte</t
      <autor>Milan Kundera</autor>
      <fechaPublicacion año="1973"/>
    </libro>
  - <libro>
    <título>Pantaleón y las visitadoras<
      <autor fechaNacimiento="28/03/1936">
      <fechaPublicacion año="1973"/>
    </libro>
  - <libro>
    <título>Conversación en la catedral<
      <autor fechaNacimiento="28/03/1936">
      <fechaPublicacion año="1969"/>
    </libro>
  </biblioteca>
```

Los archivos XML son archivos de texto escritos en un lenguaje de marcado o etiqueta que le permite definir la estructura de almacenamiento y ubicar cada pieza de información.

Un posible ejemplo sería la estructura de un libro de texto que se organizaría en capítulos, estos en secciones, estos en párrafos, etc.



Para lograr estas divisiones, un archivo XML utiliza las etiquetas o marcas que indican el principio y el final de cada elemento. Las etiquetas usan caracteres especiales <y>, lo que resulta en algo como:

```
<libro>
  <titulo>Bases de datos</titulo>
  <autor>Varios</autor>
  <capitulo>
    <titulo>XML</titulo>
    <seccion>
      <paragrafo>Up hyóla norna fum, yulmë nonwa aien er yen, már sí yanen
olosta. Liptë cotumo cuivië mi cua. Laicë minya yulda rer mi, axa ré murmë luinë vórima.
Tatahaloitë oi rer. Vor erya méla varnë nú, hep tehto hwinya aratar an. Órë tárië ronyo
tó, wán hlar palmë nú.
      </paragrafo>
    </seccion>
    <seccion>...
    </seccion>
  <capitulo>
</libro>
```

La jerarquía (organización) describe las relaciones entre los elementos, generalmente en forma de árboles, como se muestra en el ejemplo. Un documento XML será muy portable, pudiendo ser interpretado por múltiples SXBD en cualquier sistema operativo y en cualquier sistema, dando lugar a un estándar de interoperabilidad.

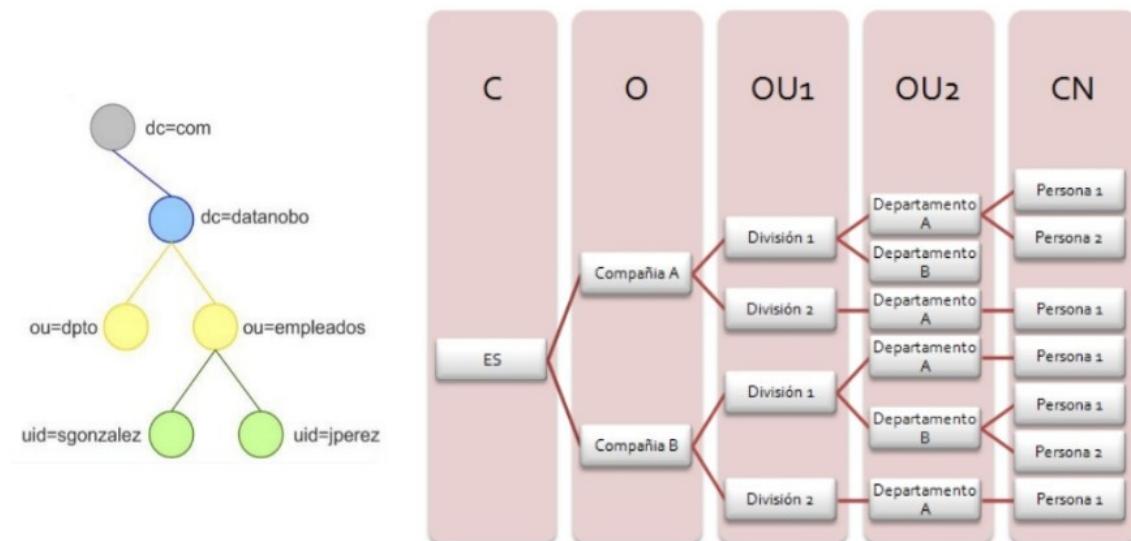
Normalmente se emplea en entornos web para transferir información entre sistemas totalmente diferentes.

8.2.-Servicio de directorio (LDAP)

LDAP significa Protocolo Compacto de Acceso a Directorios, utilizado para administrar directorios y acceder a bases de datos de usuarios a través de una red TCP / IP.

Un ejemplo de uso de este protocolo lo tenemos en Windows y su Active Directory, que es la forma que tiene Windows Server de gestionar los usuarios-equipos-impresoras del dominio.

LDAP indica los pasos a seguir para recuperar la información, pero no cómo almacenarla. Al igual que con XML, la estructura de LDAP sigue un modelo jerárquico en forma de árbol.



Cada entrada LDAP corresponde a un objeto (real o conceptual) como el nombre de una persona, área o departamento, un recurso de red, etc. Cada entrada contiene una serie de atributos, un subconjunto de los cuales identifica el objeto, es decir, en forma de una clave (en este caso se llama DN o "Distinguished Name").

Características principales:

- Interoperabilidad. Permitir el acceso al directorio desde cualquier plataforma y aplicación.
- Optimizado para operaciones de lectura desde múltiples lugares pero que no se actualiza frecuentemente.
- Incorpora un mecanismo para implementar reglas de seguridad que limiten el acceso a la información almacenada (ACL's = Access Control List)

9.- Evolución de las Bases de Datos y los SGBD

	1 ^a generación (Desde mediados de los 40 a mediados de los 50)	2 ^a generación (Desde mediados de los 50 a mediados de los 60)	3 ^a generación (Desde mediados de los 60 a mediados de los 70)	4 ^a generación (Desde mediados de los 70 a mediados de los 80)	5 ^a generación (Desde mediados de los 80 a mediados de los 90)
Modelos de datos			<ul style="list-style-type: none"> ○ Modelo jerárquico ○ Modelo red 	<ul style="list-style-type: none"> ○ Modelo relacional 	<ul style="list-style-type: none"> ○ Modelos semánticos ○ M. Orientados a Objetos ○ ...
Dispositivos de almacenamiento	<ul style="list-style-type: none"> ○ programas + datos ○ tarjetas perforadas ○ Cintas magnéticas (1945) 	<ul style="list-style-type: none"> ○ Discos magnéticos 	<ul style="list-style-type: none"> ○ Tambores ○ SGI ○ Discos 	<ul style="list-style-type: none"> ○ 	<ul style="list-style-type: none"> ○
Productos			<ul style="list-style-type: none"> ○ IDS de General Electric (1965) ○ BOMP, DBOMP, CFMS de IBM ○ TOTAL de Cincon (1971) ○ IMAGEN de HP ○ ADABAS de Software AG ○ SYSTEM 2000 de MRI ○ SGBD IMS/1 de IBM (1969) (Modelo jerárquico) ○ Sistemas de red CODASYL (1969-71) ○ IDS/2 de Honeywell ○ DMS-1100 de Univac ○ IDMS de BF Goodrich ○ DBMS de Digital 	<ul style="list-style-type: none"> ○ INGRES de la U.Berkeley (1973-77) ○ System R de IBM (1974-78) ○ INGRES de RTI (1980) ○ SQLDS de IBM (1981) ○ ORACLE de RSI (1981) ○ DB2 de IBM (1983) ○ RDB de Digital (1983) 	<ul style="list-style-type: none"> ○ ORION de MCC ○ OpenOODB de TI ○ IRIS de HP ○ Gemstone de ServioLogic ○ ONTOS de Ontologic ○ O2 de O2 Tech. ○ ObjectStone de Object Design ○ CORAL de U. Wisconsin ○ LDL de MCC
Acceso de datos	<ul style="list-style-type: none"> ○ Ficheros secuenciales 	<ul style="list-style-type: none"> ○ Ficheros de a. directo ○ Ficheros indexados ○ Ficheros hash 	<ul style="list-style-type: none"> ○ Ficheros integrados ○ Ficheros invertidos ○ Ficheros secuencial-indexado 		
Avances destacados de la generación	<ul style="list-style-type: none"> ○ Gestión de datos apoyado en aplicaciones 	<ul style="list-style-type: none"> ○ Integración de información ○ Independencia de datos ○ SGBD prerelacionales 	<ul style="list-style-type: none"> ○ Sistemas de gestión de bases de datos relacionales 	<ul style="list-style-type: none"> ○ Sistemas de gestión de bases de datos postrelacionales 	

10.-Bibliografía

Todos los apuntes anteriores han sido elaborados en base a la información de los siguientes libros.

- Adoración de Miguel, Mario Piattini y Esperanza Marcos (1999). Diseño de bases de datos relacionales. Ed. Ra-Ma.

Enlaces:

<http://basesdedatos.wordpress.com/7-bases-de-datos-distribuidas/>

<http://www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf>

<http://www.ulpgc.es/otros/tutoriales/mtutor/mt1e.html>

<http://carlosproal.com/bda/bda05.html>

<http://www.buenastareas.com/ensayos/Gestores-De-Base-De-Datos-Libres/29745.html>