

C.F.G.S. DESARROLLO DE APLICACIONES WEB

**MÓDULO:
BASES DE DATOS**

Unidad 1

Sistemas de Almacenamiento de la Información

Exámen → 15 Octubre



INDICE

OBJETIVOS	3
1.- Sistemas Lógicos de Almacenamiento.	4
1.1.- Evolución de los sistemas de almacenamiento de la información.	4
2.- Ficheros.....	6
2.1.- Introducción: conceptos sobre ficheros.	6
2.2.- Tipos de Ficheros.	7
2.3.- Inconvenientes de los ficheros.	15
3.- Bases de datos.	17
3.1.- Introducción.	17
3.2.- Arquitectura de las bases de datos	19
3.3.- Modelos de bases de datos.....	21
4.- Sistemas gestores de bases de datos.	31
4.1. Componentes.	31
4.2.- Funciones.	32
4.3.- Arquitectura Cliente/Servidor.....	34



OBJETIVOS

En esta unidad el alumno aprenderá lo que es una base de datos y que funciones y componentes tiene, valorando su utilidad.

Con ella se pretende que el alumno conozca los sistemas de almacenamiento de la información que se han venido utilizando hasta ahora y sus inconvenientes, para que pueda comprender mejor la importancia de los sistemas de bases de datos actuales.

Los objetivos principalmente serán:

- Analizar los sistemas lógicos de almacenamiento y sus funciones
- Identificar los tipos de bases de datos según el modelo de datos utilizado.
- Identificar los tipos de bases de datos según la ubicación de la información
- Valorar la utilidad de un sistema gestor de base de datos
- Describir la función de cada elemento de un sistema gestor de bases de datos
- Clasificar los sistemas gestores de bases de datos.



1.- Sistemas Lógicos de Almacenamiento.

En cualquier actividad económica es necesario tomar decisiones.

Para tomar decisiones acertadas se requiere manejar una buena información que se obtendrá a partir de los **datos**. Entendemos los datos como hechos aislados. Cuando los datos se organizan y se tratan se obtiene **información**.

Para manejar los datos con eficacia utilizaremos una **base de datos**, que nos ayudará a almacenar y procesar esos datos, extraer la información necesaria y tomar decisiones.



Las bases de datos han evolucionado a partir de los **sistemas de archivos** que presentaban una serie de problemas y limitaciones que actualmente han sido superados.

Dentro de las bases de datos existen distintos modelos con sus ventajas e inconvenientes. Actualmente el modelo más extendido sigue siendo el **modelo relacional**.

Así pues, después de conocer el desarrollo de los distintos sistemas de almacenamiento, examinar algunos conceptos de bases de datos, entender los distintos modelos de base de datos existentes pondremos nuestro interés en el modelo de datos relacional.

1.1.- Evolución de los sistemas de almacenamiento de la información.

■ **Fichero manual:**

Tradicionalmente, las personas que se encargaban de guardar y hacer un seguimiento de la información, lo hacían mediante un sistema de ficheros manual, que se componía de un **conjunto de carpetas etiquetadas cuyo contenido estaba relacionado y se guardaban en un armario o archivo**.

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**.

Por ejemplo, en los colegios o centros de enseñanza, se guardaban así archivados los expedientes de cada uno de los alumnos con sus datos personales, notas de cada curso, etc.



En una consulta médica se guardaban en carpetas las historias clínicas de los pacientes.



■ Sistemas de ficheros:

Las primeras aplicaciones que manejaban los datos utilizando el ordenador se concentraban en tareas propias de oficina como gestión de entradas y salidas de pedidos, nóminas, facturación, etc.

Los datos necesarios se guardaban en ficheros en el ordenador y estas aplicaciones accedían a ellos para obtener los informes que se solicitaban de cara a la toma de decisiones en la empresa.

Teníamos por una parte los ficheros necesarios para contener los datos con una estructura determinada y por otro los programas de aplicación que accedían a estos datos para producir información.

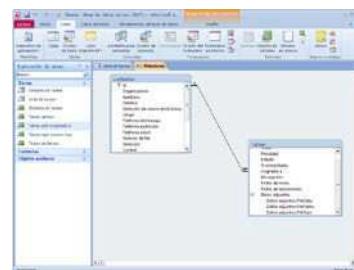


Hasta no hace mucho tiempo el lenguaje de programación COBOL era el más extendido para implementar aplicaciones de gestión tanto empresariales, como para los bancos, etc; por su fiabilidad para trabajar con distintos tipos de ficheros.

■ Sistemas de bases de datos

A finales de los años 60 surgen las bases de datos. En una base de datos se almacenan todos los datos que necesita la empresa y los programas que utilicen esos datos no se tienen que preocupar del almacenamiento físico de los mismos. Cualquier cambio en la estructura de los datos no afectará a los programas de aplicación que los utilicen.

Una base de datos es un conjunto de datos relacionados entre sí, organizados y estructurados, con información referente a algo.



Podremos utilizar una base de datos para cosas tan sencillas como mantener un registro de todos los vídeos de nuestras películas, nuestra biblioteca personal o la música que tenemos almacenada o tan complicadas como llevar toda la gestión de nuestra empresa.

Las bases de datos son tratadas usando **sistemas gestores de bases de datos (SGBD)** que proporcionan una serie de programas y utilidades que acceden y gestionan los datos.

Hoy en día los sistemas de archivos están superados, pero es importante conocer sus características, sus limitaciones y los problemas que presentaban para poder comprender mejor cómo funcionan las bases de datos.



2.- Ficheros.

2.1.- Introducción: conceptos sobre ficheros.

Para almacenar la información de modo permanente se utilizan **dispositivos de almacenamiento masivo** denominados **memoria secundaria**, ya que los datos **guardados en la memoria principal** desaparecen al desconectar el ordenador.

La información contenida en los dispositivos de almacenamiento se estructura en unidades denominadas **ficheros**.

Para entender mejor el funcionamiento del sistema de ficheros empezaremos por conocer la terminología básica aplicándolo a nuestro ejemplo:

Por ejemplo en un taller mecánico, supongamos que quieren almacenar los datos de los clientes que empiezan a ser habituales, en un fichero de CLIENTES. De ellos interesa mantener, por ejemplo, el nombre y los apellidos, el teléfono y la dirección.

- **Datos:** son los hechos, o aspectos que necesitamos almacenar para obtener, a partir de ellos, alguna información. Inicialmente los datos cuando no están organizados de forma lógica no tienen mucho significado.

Los datos en nuestro ejemplo serían los datos personales de los clientes recogidos anteriormente, sin tratar.

- **Campo:** Es un carácter o conjunto de caracteres que tiene significado específico. Se utiliza para definir y guardar datos. Es la mínima unidad de información creada con sentido en sí misma.

Definiríamos un campo para guardar el nombre, en otro campo para el teléfono, etc.

- **Registro:** Es un conjunto de campos lógicamente relacionados que describen una persona, lugar o cosa. Es también la unidad de tratamiento de los ficheros de datos.

Un registro lo formarían todos los datos relativos a un cliente: su nombre, sus apellidos, su teléfono y su dirección. (Por ejemplo los datos de Luis Gómez)

- **Fichero:** Es un conjunto de registros relacionados.

El fichero estaría formado por los datos de todos los clientes del taller.

Campos:	FICHERO DE CLIENTES					
	Campo	DNI	NOMBRE	APELLIDOS	DIRECCION	TELEFONO
Registro 1:	73564765M	Javier	Barquin 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	Rodriguez Mata	Paseo Pereda, 123	942665544	

Registro

Datos



Los **dispositivos de almacenamiento masivo** según la forma de acceder a la información se clasifican en:

- **Dispositivos secuenciales:** la información se guarda en posiciones consecutivas, de forma que para acceder a un dato hay que recorrer los datos anteriores. Por ejemplo la cinta magnética.
- **Dispositivos direccionables:** permiten el acceso directo a los datos. En estos dispositivos el espacio destinado a almacenamiento está dividido en segmentos direccionables de forma individual. Por ejemplo el disco duro.

El **acceso a un registro** es el procedimiento que se utiliza para seleccionarlo. Este acceso está condicionado por el tipo de soporte en el que se encuentre almacenada la información. Los tipos de acceso son:

- **Secuencial:** los registros se leen uno detrás de otro desde el principio del fichero hasta localizar el registro buscado o hasta el final del fichero. Puede utilizarse tanto con dispositivos secuenciales como direccionables.
- **Directo:** permite seleccionar un registro sin tener que leer los anteriores, accediendo directamente a él mediante su clave. Solo puede utilizarse en dispositivos direccionables.
- **Indexado:** para seleccionar un registro consultamos previamente de forma secuencial en una tabla que contiene la clave más alta y la dirección de comienzo de cada bloque de registros. Una vez localizado se utiliza el acceso directo a ese bloque de registros y, dentro del bloque, la lectura secuencial hasta localizarle. Necesita dispositivos de almacenamiento direccionables.
- **Dinámico:** permite el acceso directo o por índice a un registro y a partir de ese se accede a los demás de forma secuencial. Necesita también soportes direccionables.

2.2.- Tipos de Ficheros.

2.2.1.- Organización Secuencial.

En un fichero con organización secuencial los registros se escriben sobre el dispositivo de almacenamiento en posiciones físicamente contiguas, sin dejar huecos entre ellos, en el mismo orden en que hayan sido introducidos.

En este tipo de ficheros hay una correspondencia entre el **orden físico** (orden en el que están grabados los registros) y el **orden lógico** (orden en el que se han dado de alta y recuperado los registros). Puede utilizarse con soportes tanto secuenciales, como direccionables.

Con el fin de mejorar las prestaciones de la organización secuencial surgen una serie de organizaciones que son una variante de esta y que pueden ser utilizados con soportes direccionables, que trataremos a continuación

- ✓ Organización secuencial encadenada
- ✓ Organización secuencial indexada
- ✓ Organización secuencial indexada - encadenada





■ Las ventajas de los ficheros con organización secuencial son:

- ✓ Rapidez en el acceso a un bloque de registros contiguos
- ✓ No es necesario realizar operaciones de **compactación** del archivo
- ✓ No se desperdicia espacio en el dispositivo de almacenamiento porque no hay huecos
- ✓ Se pueden utilizar cualquier tipo de registros: de longitud fija, variable o indefinida.

■ Los inconvenientes de este tipo de organización son:

- ✓ Para acceder al registro n hay que recorrer los n-1 registros anteriores. El acceso es secuencial.
- ✓ Para realizar una **consulta** hay que crear un proceso en el que se compare el valor del campo que se pretende localizar con el valor del mismo campo correspondiente a cada registro leído del fichero.
- ✓ La adición de registros se realiza a continuación del último registro ya existente. No se pueden insertar nuevos registros. *Al final*
- ✓ No se pueden eliminar registros. Para eliminar un registro se marca de modo que no se muestre, pero el registro existe y ocupa espacio en el dispositivo del almacenamiento. (**borrado lógico**)
- ✓ Para mantener ordenado y compactado el fichero, hay que crear un fichero nuevo a partir del existente.

Dirección de memoria	Marca de borrado	Nombre	Apellidos	Teléfono
1200		Alfredo	Bárcena	768334472
1300	*	Isabel	De los Ríos	987335612
1400		Carmen	Sierra	955347612
1500		Fernando	Ruiz	674992455
1600		Juan Carlos	Abad	573982277

En el momento de utilizar los ficheros con organización secuencial tenemos que tener en cuenta el soporte sobre el que están grabados, pues algunas operaciones que se pueden hacer en los soportes direccionables no se pueden hacer en los soportes secuenciales, como pueden ser las modificaciones y borrado lógico de registros.

La organización secuencial es aconsejable para ficheros con un índice de utilización muy elevado y que sean estables. Este tipo de ficheros son útiles cuando en cada operación de actualización o de consulta se van a procesar la mayoría de los registros. No son adecuados cuando se necesita procesar frecuentemente registros aislados, con un índice de utilización muy bajo.

2.2.2.- Organización secuencial encadenada.

Son ficheros de organización secuencial gestionados mediante punteros que nos permiten tener los registros ordenados según un orden lógico diferente del orden físico en el que están grabados.

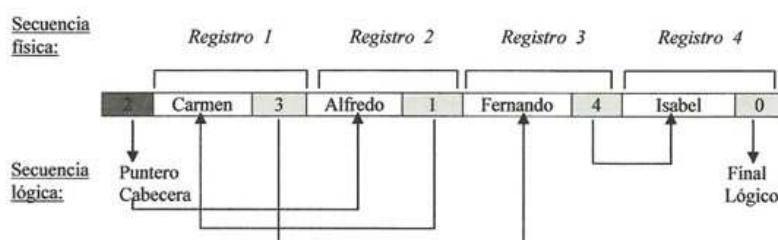


Los **punteros** son un campo adicional, por lo que aumentan el tamaño de los registros, y forman parte integrante de la estructura del registro, indicando cual es el siguiente o el anterior registro en secuencia lógica y no en secuencia física.

Las características de los punteros son:

- Estar en cada registro en una posición fija y definida para todos los registros
- Poseer una longitud constante

Registros encadenados con punteros



Para poder utilizar los datos del fichero el sistema operativo utiliza un indicador (puntero), que se coloca señalando al primer registro de datos del fichero, y se va desplazando, siempre en la misma dirección, cada vez que se lee o graba un registro. La estructura del fichero se completa con un **registro de cabecera** que contiene información acerca del fichero y un registro que sirve de **marca de final de fichero**, que el sistema utiliza para saber cuál es el último registro del fichero. El registro de final de fichero se graba, por primera vez en el momento de la creación del fichero y se va desplazando cuando se añaden nuevos registros al final del mismo.

En estos ficheros la secuencia física y la secuencia lógica no coinciden, pudiendo ocurrir que el último registro en secuencia física sea el primero en secuencia lógica y viceversa.

- La **consulta** es secuencial, cada vez que se consulta un registro en él se lee la posición del registro siguiente en secuencia lógica.
- La **adición** de registros se realiza al final, pues los registros se almacenan secuencialmente. Cuando se quiere insertar un registro en una dirección intermedia, el **registro físicamente se añade al final**, pero se modifican los **punteros** para mantener la secuencia lógica.
- La **eliminación** de registros se efectúa **modificando el puntero en el registro anterior**, para que apunte al registro siguiente al que queremos borrar, de modo que el fichero mantiene su tamaño, ya que el borrado que se produce es borrado lógico, pero no físico.
- Para **modificar** un registro se reescribe sobre la información anterior.

Estos ficheros necesitan soportes direccionables.





2.2.3.- Organización secuencial indexada.

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

Se les llama también **ficheros indexados** porque se basan en la utilización de índices.

En este tipo de ficheros se distinguen tres áreas:

Área primaria:	<p>En esta área se escriben los registros cuando se crea el fichero. Es la zona donde están contenidos los registros ordenados ascendente por el valor de su clave. Esta área del fichero está dividida en segmentos. Cada segmento almacena "n" registros consecutivos y almacenados en posiciones contiguas. Es un área de organización secuencial, donde el acceso a cada registro se realiza en una doble operación</p> <ul style="list-style-type: none">• Acceso directo al segmento donde se haya ubicado el registro buscado• Acceso secuencial posteriormente, a los registros del segmento, hasta localizar el registro buscado o alcanzar el final del segmento, en caso de que no se halle.
Área de índices	<p>Es creada por el sistema al mismo tiempo en que se almacenan los datos. En esta área los registros están formados por dos campos:</p> <ul style="list-style-type: none">• El valor del campo clave del último registro de un bloque o segmento. Los bloques están constituidos por un número fijo de registros consecutivos.• El segundo campo contiene la dirección de comienzo de cada uno de los segmentos en los que se halla dividida el área primaria
Área de overflow:	O área de excedentes. Es la zona destinada a contener los registros almacenados posteriormente a la creación del fichero, por lo que no han sido incluidos en el área primaria. Estos registros tendrán claves intermedias a las de los registros previamente almacenados en el área primaria.

Estos índices son similares a los de los libros. Si nos interesa leer un capítulo concreto podemos recurrir al índice que nos dice en qué página comienza, y abrimos el libro por esa página, sin tener que mirar en todas las páginas anteriores para localizarlo



- El **acceso a los registros** se realiza mediante una consulta secuencial al área de índices para determinar el segmento donde se encuentra el registro buscado.
- Con el valor del segmento seleccionado, se recorren secuencialmente los registros de ese segmento. Si el registro no está comprendido en el segmento, se continúa la búsqueda de forma secuencial en el área de overflow hasta la localización del registro o hasta terminar de leer los registros de esa área.
- **El borrado** La eliminación de los registros debe realizarse mediante marcas. Se generan huecos que realmente son posiciones de memoria ocupadas por registros marcados pero que no han sido eliminados físicamente del fichero. La única posibilidad de eliminar estos huecos, es en futuras operaciones en las cuales necesitemos reorganizar el fichero.
- **La inserción** de registros se hace en el área de overflow. No está permitida la instrucción de nuevo registro en el área primaria después de la creación del fichero.
- En esta organización cuando el numero de registros borrados es grande, o las cadenas de desbordamientos son largas su utilización deja de ser eficiente, siendo necesario **reorganizar el archivo**.

Esta organización es muy utilizada, tanto para procesos en los que intervienen pocos registros como para aquellos en los que se maneja el fichero completo ya que aprovecha las ventajas de las organizaciones secuencial y relativa

2.2.4.- Organización indexada-encadenada.

Este tipo de organización aprovecha lo mejor de la organización secuencial encadenada e indexada.

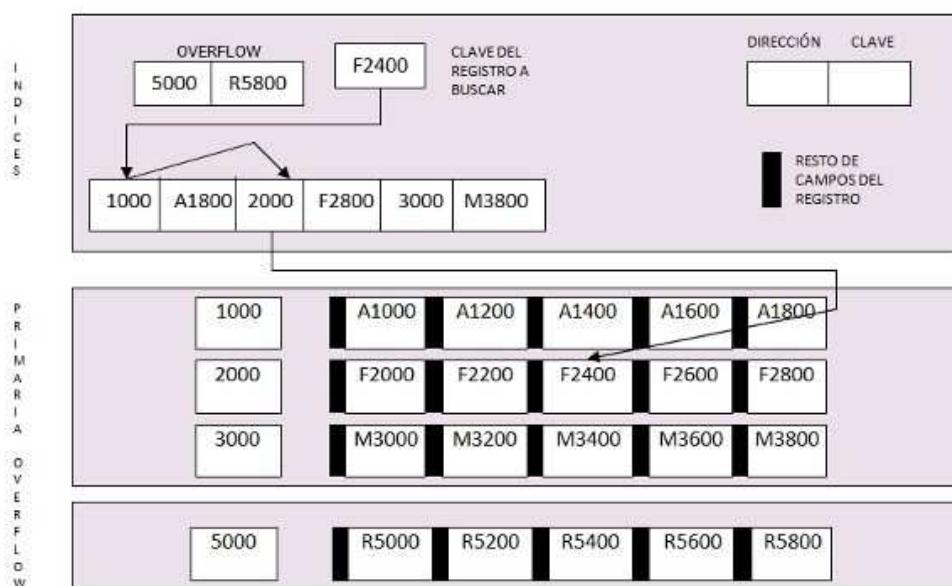
Se caracteriza por la utilización de **punteros e índices** de forma simultánea, lo que implica un aumento del espacio ocupado, pero proporciona una gran rapidez en la búsqueda de registros.

La estructura de esta organización es la misma que la de la organización indexada, a la que se han añadido punteros entre los registros de la zona primaria y la de overflow. De esta manera se consigue, respecto a la organización, indexada, mejorar los tiempos de búsqueda de registros en la zona de overflow y mantener la organización lógica de registros.

- Para **eliminar registros** se marcan, en lugar de ser borrados físicamente.
 - Las **adiciones** se realizan sobre la zona de overflow ya que no se pueden añadir registros en el área primaria una vez creado el fichero.
- Estos ficheros deben ser reorganizados con frecuencia ya que la no eliminación física de los registros marcados y las adiciones crean un overflow grande, y si no se reorganizan llegan a funcionar como ficheros secuenciales.



- Para **acceder a un registro** se busca en el área de índices la dirección de inicio del bloque de registros a la que pertenece el registro buscado. Si no se encuentra en el área de índices, el último registro del bloque apuntará a un bloque de área de overflow, que se lee de forma secuencial. Si no se encuentra en el área de overflow, se terminará la búsqueda al acabar el área de overflow.
- Si se desea **consultar** todo el fichero, el último registro de cada bloque del área de overflow tiene un puntero al primer registro del área siguiente y así se continuará la búsqueda hasta acabar de leer el fichero.
- Los nuevos registros se **insertan** y quedan enlazados entre sí mediante punteros conservando el orden lógico que marca la clave o índice principal.



Del tratamiento de los índices y punteros se encarga el sistema operativo por lo que no va a crear problemas al usuario cuando maneja este tipo de ficheros. El usuario sabe lo que sucede cuando solicita una consulta de un registro, pero no sabe cómo se realiza internamente esa consulta.

2.2.5.- Organización relativa directa.

La **organización relativa** está basada en la independencia entre el orden en el que se dan de alta los registros y la posición en que se graban en el soporte.

Son ficheros en los que el almacenamiento físico de los registros se realiza mediante el empleo de una clave que relaciona la posición del registro dentro del fichero y la posición de memoria donde está almacenado.

Hay dos tipos de organización relativa: **directa y aleatoria**.





En el caso de la **organización relativa directa** se emplean claves numéricas, por lo que los registros poseen direcciones numéricas enteras, de forma que **la secuencia lógica de almacenamiento de los registros en el fichero coincide con la secuencia física de almacenamiento de los registros** sobre el dispositivo, ya que las posiciones físicas de almacenamiento coinciden con el valor de la clave.

En este tipo de organización **no se puede almacenar un registro cuya clave esté por encima de los límites máximos del fichero**, ya que cada dirección sólo puede ser ocupada por un registro.

■ **Las ventajas** de este tipo de organización de ficheros son:

- **Acceso directo a los registros.** No se necesita un algoritmo de transformación
- **Permite realizar operaciones de escritura y lectura simultáneamente**, ya que primero se localiza el registro y luego se realiza la operación deseada: inserción, eliminación, consulta, modificación, etc.
- **El acceso a los datos** se realiza de **dos formas diferentes**:
 - ✓ **Directamente**, mediante la clave del registro
 - ✓ **Secuencialmente**, a partir del primer registro almacenado en el fichero, por lo que son muy rápidos en el tratamiento individual de registros.

Direcciones de memoria	Clave	Datos	Espacios o huecos libres
1			
2	2	Cliente A	
3			
4	4	Cliente B	
5	5	Cliente C	
6			
7			
8	8	Cliente D	
9			
10	9	Cliente E	
	10		

■ **Los inconvenientes** de este tipo de organización de ficheros son:

- Al realizar **un acceso secuencial**, en una consulta sobre todos los registros del fichero hay que recorrer todas las direcciones aunque estén vacías.
- Deja **gran cantidad de posiciones libres de memoria dentro del fichero (huecos)**, debido a que las claves de los registros pueden indicar posiciones de almacenamiento no contiguas, lo que implica una falta de aprovechamiento del soporte del almacenamiento respecto al número real de registros almacenados.
- **Se producen colisiones**, ya que puede existir más de un registro con la misma clave. Esto causa errores.



2.2.6.- Organización relativa aleatoria.

Son ficheros con organización relativa y clave alfanumérica o bien numérica pero que se debe transformar obteniéndose un valor numérico entero que facilite la correspondencia directa entre la clave y la dirección de memoria.

En este caso, las direcciones lógicas de almacenamiento, las claves, no coinciden con las direcciones físicas, que son las posiciones de cada registro.

El valor de la clave debe estar en relación con la capacidad máxima del soporte físico. No se pueden almacenar registros cuya dirección de almacenamiento sea mayor que los límites máximos del fichero.

La dirección del almacenamiento de un registro dentro del dispositivo se obtiene de su clave de la siguiente forma:

- Clave alfanumérica se aplican algoritmos de transformación para obtener valores enteros positivos.
- Si la clave es numérica se aplica un algoritmo que permita obtener un rango de valores comprendidos en el intervalo de valores de las direcciones de memoria disponibles, de modo que existe una relación directa entre la dirección lógica (clave) y la dirección física (memoria).

El algoritmo de transformación o hashing debe cumplir las siguientes condiciones:

- Que sea fácil de aplicar, estableciendo una relación directa entre dirección lógica y dirección física
- Que deje el mínimo número de huecos posible, maximizando el espacio disponible en el dispositivo de almacenamiento.
- Que las claves de registros diferentes nos den direcciones diferentes. Producir el menor número de registros que con distintas claves creen las mismas direcciones de almacenamiento. Cuando a partir de dos o más claves diferentes se obtiene la misma dirección se dice que se producen sinónimos y que esos registros producen colisiones. En este caso solo uno de ellos puede ser almacenado en esa dirección y habrá que prever algún procedimiento para calcular la posición en que se tiene que grabar el otro registro.

■ Las ventajas de este tipo de organización de ficheros son:

- Acceso inmediato a los registros mediante su clave
- No es necesario ordenar el fichero
- Se pueden realizar operaciones de escritura y lectura a la vez
- Son muy rápidos en el tratamiento individual de registros.
- Se pueden realizar accesos secuenciales.



Direcciones de memoria	Clave	Datos	
1			
2	AB85	Cliente C	
3			
4	DG49	Cliente A	
5	EH23	Cliente D	
6			
7			
8	BS12	Cliente F	
9	KL92	Cliente E	
10			
	FB43	Cliente B	

■ **Los inconvenientes** de este tipo de organización de ficheros son:

- Las consultas sobre todo el fichero son lentas
- El fichero contiene gran cantidad de huecos o espacios libres
- El algoritmo para la conversión de las claves y el algoritmo necesario para el almacenamiento de sinónimos han de ser creados de modo que dejen el menor número de huecos libres y se genere el menor número de sinónimos.

La organización relativa es la organización que tiene un menor tiempo de acceso a un registro en acceso directo. Se usan cuando el acceso a los datos de un registro se hace siempre empleando la misma clave y la velocidad de acceso a un registro es lo que más nos importa.

2.3.- Inconvenientes de los ficheros.

Aunque estos sistemas fueron utilizados durante mucho tiempo tienen inconvenientes importantes ya que se trata de sistemas orientados hacia los procesos, debido a que en ellos se da mayor importancia al tratamiento que reciben los datos, que se almacenan en ficheros diseñados para una determinada aplicación. Las aplicaciones son independientes unas de otras y los datos no se transfieren entre ellas, sino que se duplican cuando se necesitan.

Presentan dos tipos de problemas: respecto a los ficheros y respecto a los datos.



■ Problemas respecto a los ficheros:

Se deben a la necesidad de controlar la integridad semántica, el control de las autorizaciones, y la concurrencia de accesos de varios usuarios al mismo fichero simultáneamente.

• **Integridad semántica:**

Es un conjunto de restricciones, también llamadas Reglas de Validación, que permiten o no almacenar determinados valores de un objeto en la base de datos para evitar que se pierda la consistencia.

Ejemplo: Por ejemplo una restricción para que el saldo de una cuenta corriente no baje de un cifra determinada.

Cada fichero puede tener diferentes reglas de validación. Es difícil crear programas que tengan en cuenta a la vez todas estas reglas, con lo que se produce información inconsistente para el sistema.

• **Control de autorizaciones:**

Trata de evitar que se produzcan accesos indebidos a los datos, para lo que a cada usuario se le da un identificador y una clave. En este caso al estar los elementos del sistema distribuidos sin organizar a lo largo del sistema, no se puede controlar el acceso a cada elemento del sistema.

Ejemplo: No todo el personal del banco puede acceder a todos los datos. El departamento de nóminas sólo necesita acceder a los datos de los empleados y no a los de los clientes.

Es difícil prever y crear los controles de seguridad adecuados y puede que usuarios no autorizados accedan a datos de forma indebida.

• **Control de concurrencia:**

Es el control del acceso simultáneo de varios usuarios a los mismos datos. Ya que cuando varios usuarios acceden a la vez al mismo fichero para modificar información, si no hay un programa que controle el orden de acceso no habrá seguridad acerca de cuál de todas las modificaciones será guardada y en qué orden.

Ejemplo: cuando varios clientes acceden a una misma cuenta para retirar sus fondos. Puesto que se puede acceder a los datos con distintos programas será muy difícil de coordinar.

El control de concurrencia permite el acceso simultáneo de lectura de los datos y accesos sucesivos individualizados, en cola, de modo que hasta que no termine un usuario de realizar modificaciones, los demás no podrán realizar otras nuevas.

■ Problemas respecto a los datos:

Se deben a su estructura física, a su modo de estar almacenados en diferentes archivos.

• **Redundancia:**

Es la repetición innecesaria de información en varios ficheros.

Ejemplo: Nombres y números de teléfono de los clientes del banco. Podrían aparecer duplicados, tanto en el archivo de clientes de cuentas corrientes como, por ejemplo, en el de cuentas de cheques.





- **Inconsistencia:**

Es información redundante en la que las copias de los datos de los distintos ficheros no concuerdan entre sí.

Ejemplo: Cuando el teléfono de un cliente ha cambiado y no hemos reflejado el cambio en todos los ficheros que lo contienen.

- **Aislamiento:**

O fragmentación de la información. Se produce cuando los datos referentes a un objeto se almacenan en distintos ficheros, siendo difícil obtener a la vez toda la información relativa al mismo objeto.

Ejemplo: los datos personales de un cliente pueden estar almacenados en un fichero, mientras que se han creado otros ficheros recoger los datos relativos a sus cuentas bancarias, los préstamos e hipotecas, etc.

- **Dificultad de acceso a los datos:**

Es un problema organizativo en que, para eliminar el aislamiento, no existe una relación de todos los ficheros con los datos que contienen.

Ejemplo: cuando queremos localizar los datos de los clientes que viven en la ciudad correspondiente a un código postal determinado. Tenemos un fichero y una aplicación que extrae los datos de todos los clientes pero no teníamos previsto obtener ese tipo de información. Será necesario escribir un nuevo programa.

Las bases de datos surgen como un nuevo planteamiento de los sistemas orientados hacia los datos, para mejorar tanto las prestaciones como el rendimiento.



3.- Bases de datos.

3.1.- Introducción.

Los problemas inherentes a los sistemas de archivos pretenden eliminarse con la aparición de las bases de datos en los años 60.

Las **ventajas** que aportan las bases de datos sobre los sistemas de ficheros son:

- **Control sobre la redundancia de datos.** En los sistemas de bases de datos **todos** estos ficheros están integrados, por lo que **no se almacenan varias copias de los mismos datos**.
- **Consistencia de datos.** Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente.



- **Más información sobre la misma cantidad de datos.** Al estar todos los datos integrados, se puede extraer información adicional sobre los mismos.
- **Compartición de datos.** En los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados. Además, las nuevas aplicaciones que se vayan creando pueden utilizar los datos de la base de datos existente.
- **Se simplifica el esfuerzo de programación y mantenimiento de los programas.**
- **Mantenimiento de estándares.** Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales.

Como **conceptos** podemos decir que:

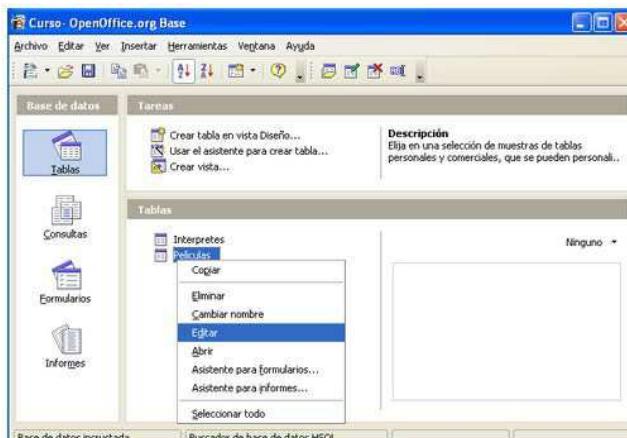
Una **base de datos** es un **conjunto de datos relacionados entre sí, organizados y estructurados, con información referente a algo.**

Podría definirse como un gran almacén de datos que se define una sola vez y a los que pueden acceder varios usuarios simultáneamente.

Id_genero	Nombre	Id_pelicula	Título	Director	Año	Formato	Visionada
1	Ciencia-Ficción	1	Blade Runner	Ridley Scott	01/01/83	DVD	<input checked="" type="checkbox"/>
2	Aventuras	2	La Guerra de las Galaxias	George Lucas	01/01/77	VHS	<input checked="" type="checkbox"/>
3	Historico	3	Indiana Jones en Busca del Arca Perdida	Steven Spielberg	01/01/81	DVD	<input checked="" type="checkbox"/>
4	Comedia	4	Misión Dolor Baby	Clint Eastwood	01/01/04	DVD	<input type="checkbox"/>
5	Drama	5	La Comunidad del Anillo	Peter Jackson	01/01/01	DVD	<input checked="" type="checkbox"/>
6	Thriller	6	Señora Robada	Bernardo Bertolucci	01/01/96	VHS	<input checked="" type="checkbox"/>
7	Suspense	7	Gladiator	Ridley Scott	01/01/00	DVD	<input checked="" type="checkbox"/>
8	Terror	8	Rocky	John G. Avildsen	01/01/76	VHS	<input checked="" type="checkbox"/>
9	Fantasia	9	Los Lunes al Sol	Fernando León	01/01/02	DVD	<input type="checkbox"/>
		10	Mar Adentro	Alejandro Amenabar	01/01/04	DVD	<input type="checkbox"/>

Estos datos están interrelacionados y existe una mínima duplicidad.

El **Sistema gestor de la base de datos** (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener la Base de datos.





Para que este sistema sea efectivo debe cumplir:

- Los datos deben estar compartidos entre distintas personas, entre distintas localidades geográficas, etc.
- El uso de los datos debe estar controlado. El control lo facilita el SGBD y lo realizan los administradores.
- Los datos se integran de forma lógica, eliminando las redundancias y manteniendo la consistencia. La estructura lógica hace que se pueda mantener la consistencia entre muchos ficheros diferentes.

3.2.- Arquitectura de las bases de datos

En 1975, [el comité ANSI-SPARC](#) propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era separar los programas de aplicación de la base de datos física. En esta arquitectura el esquema de una base de datos se define con 3 niveles de abstracción:

- Físico (interno)
- Conceptual
- Externo

■ Nivel Físico o interno.

Describe como se almacenan físicamente las estructuras de datos en el ordenador, es decir la estructura física de la base de datos. Este esquema especifica: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, las rutas de acceso, etc.

La descripción del nivel físico se realiza mediante un **esquema interno**, que es un conjunto de definiciones y reglas que permite definir las tablas y cómo se relacionan entre sí.

A este nivel se describen los datos desde el punto de vista de la máquina que los soporta. Los usuarios que trabajan a este nivel son los **diseñadores de la base de datos o los Administradores**. Ningún usuario como tal tiene que ver con esta vista.



- Modelo entidad/relación
↓
Normalizar
↓
Modelo relacional => SQL
(pasar a tablas)

■ Nivel Conceptual →

O nivel lógico global. Describe la organización de los datos en la base de datos y las relaciones existentes entre ellos.

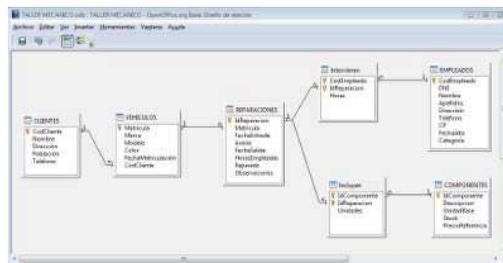
La descripción de este nivel se realiza mediante un **esquema conceptual**, que permite definir las entidades, los atributos y sus propiedades, las relaciones, operaciones de los usuarios y las restricciones y reglas de validación.





Esto implica hacer un análisis de las necesidades de información de los usuarios y definir las clases de datos para satisfacer esas necesidades.

Este nivel se ocupa de la estructura organizacional de los datos sin ocuparse de las estructuras físicas de almacenamiento. A este nivel los usuarios que intervienen son los **programadores**, encargados de crear las estructuras lógicas necesarias para guardar la información.



■ Nivel Externo.

O de usuario. Describe la base de datos como es percibida por los usuarios. Para los usuarios las tablas y sus registros existen físicamente. Cada usuario verá una base de datos (**esquema externo**) distinto según sea el nivel de acceso que se le haya concedido, ya que tendrá acceso a aquéllos datos que necesite, a las relaciones que emplee y las restricciones de uso que se le hayan definido.

Los objetos a los que puede acceder un usuario o grupo de usuarios forman su nivel externo: tablas, vistas, formularios, informes, etc. Es decir,

El nivel externo es la percepción de la base de datos por el usuario, de modo que hay tantos niveles externos distintos como grupos de usuarios haya en la base de datos.

A los usuarios cuyos conocimientos de informática no tienen por qué ser grandes hay que ocultarles la complejidad interna de las bases de datos, de modo que les sea transparente.

Al esquema externo también se le llama **vista**. A este nivel acceden dos tipos de **usuarios**: Los que acceden a su propio esquema externo y desde él realizan las consultas y acciones que deseen y los usuarios finales que acceden a la base de datos desde sistemas de menús y transacciones predefinidas. Este último grupo sólo realiza lecturas o adiciones de datos, poseyendo un esquema externo muy limitado.

Con la arquitectura a tres niveles se introduce el concepto de **independencia de datos**. Se definen dos tipos de independencia:

- **Independencia lógica**: se refiere a la posibilidad de modificar el esquema conceptual de la base de datos sin tener que modificar los esquemas externos, ni los programas. Por ejemplo: si se borra una entidad las vistas que no se refieran a ella no se alteran.
- **Independencia física**: se refiere a la posibilidad de modificar el esquema interno sin tener que modificar ni el esquema conceptual ni los esquemas externos. Por ejemplo: se pueden reorganizar los archivos físicos o añadir nuevos archivos de datos para mejorar el rendimiento.



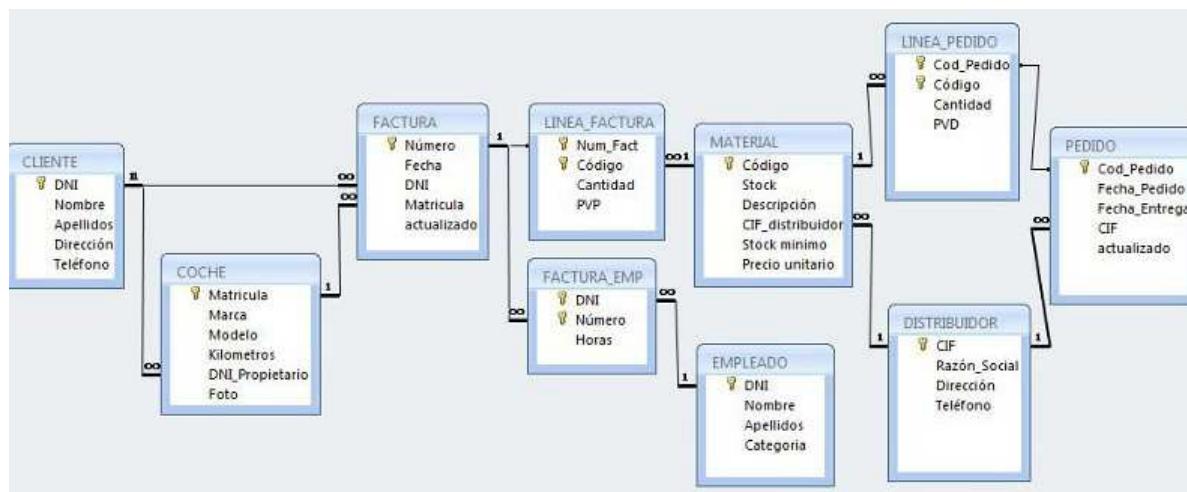
Para implementar esta arquitectura el SGBD utiliza una serie de procedimientos adicionales que implican un gasto de recursos durante la ejecución de una consulta o de un programa reduciendo su eficiencia. Por esta razón no todos los SGBD implementan la arquitectura completa.

3.3.- Modelos de bases de datos.

Llamamos **modelo** a un instrumento que se aplica a una parcela del mundo real para obtener una estructura de datos a la que denominamos **esquema**.

Modelar consiste en definir un mundo abstracto de forma que las conclusiones que se puedan sacar de él coincidan con las manifestaciones del mundo real.

Un modelo de datos es un conjunto de herramientas conceptuales que permiten describir los datos, sus relaciones y las reglas de integridad que deben cumplir.



Este esquema se especifica durante el diseño, y no es de esperar que se modifique a menudo. Sin embargo, los datos que se almacenan en la base de datos pueden cambiar con mucha frecuencia: se insertan datos, se actualizan, etc.

Es muy importante que el esquema sea correcto y se debe tener muchísimo cuidado al diseñarlo.

En su **evolución** las bases de datos se han basado en 3 modelos de datos

- **Jerárquico.**
- **En red.**
- **Relacional.**



Los primeros sistemas, introducidos en los años 60 se basaron en el **modelo jerárquico** que estructura todas las relaciones entre los datos como jerarquías. A finales de los 60 aparecieron los sistemas basados en el modelo en red. En estos dos modelos los datos se relacionaban con punteros físicos.

En 1970, Codd publicó un artículo que abrió una nueva perspectiva para los sistemas de gestión de la información argumentando que los datos deberían relacionarse mediante interrelaciones naturales y lógicas. Los datos se representan como tablas denominadas relaciones y se les aplica el cálculo y el álgebra relacional. Se denominó modelo relacional y es el estándar más empleado en la actualidad.

En estos momentos las bases de datos evolucionan en dos direcciones fundamentalmente: **el modelo de bases de datos orientadas a objetos** y la utilización de plataformas cliente-servidor para **bases de datos orientadas a internet**.

Los modelos de datos pueden clasificarse en:

- **Modelos de datos de alto nivel o conceptuales:** disponen de conceptos para describir la estructura de datos que necesitamos almacenar y sus relaciones que son muy cercanos a la forma de percibir la realidad por parte de los usuarios. Ejemplos de este son el modelo entidad/relación y el modelo orientado a objetos.
- **Modelos de datos de bajo nivel o físicos:** disponen de conceptos que describen detalles sobre el almacenamiento de los datos en el ordenador, el formato de los registros, la estructura de los ficheros y los métodos de acceso utilizados. Hay muy pocos modelos físicos en uso. Las estructuras más conocidas son las estructuras de hash y los árboles B+.
- **Modelos de datos lógicos:** se centran más en las operaciones que en la descripción de la realidad. Estos modelos pueden ser entendidos por los usuarios finales, pero no están muy lejanos de la forma en que los datos se organizan físicamente. Ejemplos de este son los modelos relacional, en red y jerárquico.

A la hora de conocer los distintos modelos de datos es necesario tener en cuenta los siguientes conceptos:

- **Independencia de los datos:** existe cuando un cambio en el tipo de datos se aplica automáticamente en cascada a través de la base de datos, por lo que no es necesario hacer cambios en los programas que se refieren a ese tipo de datos.
- **Independencia estructural:** existe cuando los cambios en la estructura de la base de datos no afectan a la capacidad de las aplicaciones de tener acceso a los datos.

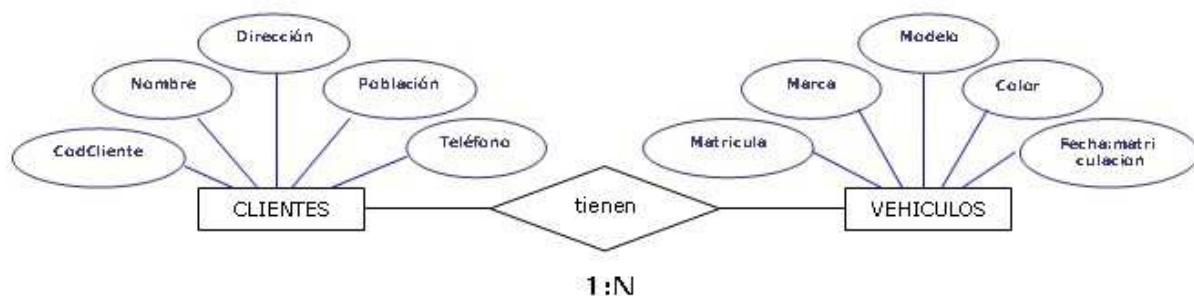
Cada SGBD soporta un modelo lógico, por tanto los modelos de datos sirven para clasificar los distintos tipos de bases de datos.



3.3.1. Modelo entidad-relación.

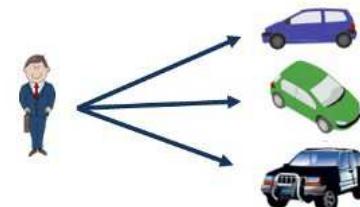
Este modelo fue propuesto por Peter Chen en 1976 para la representación conceptual de los problemas del mundo real. Es un modelo muy extendido y potente para la representación de los datos. Se simboliza haciendo uso de grafos y de tablas.

Se basa en una percepción del mundo real que consiste en una colección de objetos llamados entidades y las relaciones entre esos objetos. Cada entidad representa un objeto que se distingue de otros por tener un conjunto de atributos propio.



En el ejemplo anterior tendríamos las entidades **CLIENTES** y **VEHICULOS** y la relación establecida entre esas dos entidades: tienen.

- Como **atributos** de la **entidad** CLIENTES tenemos: CodCliente, Nombre, Dirección, Población y Teléfono.
- Los **atributos** de la **entidad** VEHICULOS son: Matricula, Marca, Modelo, Color y Fecha matriculación.
- La **relación** tienen describe la asociación entre los datos de ambas entidades. Cuando la relación es 1:N (uno a muchos) significa que un cliente podrá tener muchos vehículos (más de uno); mientras que cada vehículo es de un solo cliente.



El modelado entidad-relación es una técnica para el modelado de datos utilizando diagramas **entidad-relación**. No es la única técnica pero sí la más utilizada. Mediante una serie de procedimientos se puede pasar del modelo entidad-relación (E-R) a otros, como por ejemplo el modelo relacional que veremos posteriormente.

3.3.2. Modelo jerárquico.

Se le llama también modelo en árbol, ya que utiliza para su representación una estructura de tipo árbol invertido.



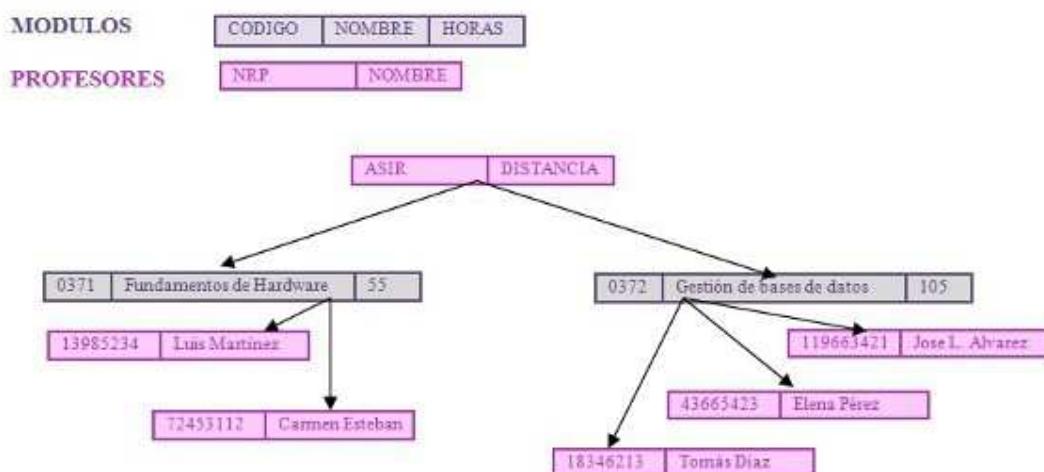


Una base de datos jerárquica es un conjunto de registros lógicamente organizados con una estructura de árbol invertido. Dentro de la jerarquía el nivel superior se percibe como el "padre" de los registros situados debajo de él, de forma que:

- Cada padre puede tener muchos hijos
- Cada hijo sólo tiene un padre

■ Las **características** principales son:

- Una colección de árboles forma una base de datos.
- A los registros se les denomina segmentos o nodos, que contienen atributos o campos
- Los nodos están organizados en niveles. Cada nodo contiene los campos comunes a los nodos hijos, vinculados a él. Al nodo más alto en la jerarquía o estructura de árbol se le denomina raíz
- Padre es un nodo vinculado a otros de nivel inferior. Un nodo no puede ser padre de sí mismo
- Hijos son los nodos vinculados con otros de nivel superior. Todos los hijos de un padre están al mismo nivel. Todo nodo no raíz tiene un padre y todo nodo padre puede tener varios hijos.
- Las relaciones entre registros se representan mediante arcos o lazos.
- No es posible representar relaciones N:M entre registros, ni **relaciones reflexivas** entre ellos.



■ **Ventajas:**

- Conceptualmente es simple y eso facilita su diseño.
- Seguridad: se ejecuta por el sistema, no depende de los programadores.
- Independencia de los datos: un cambio en el tipo de dato se aplica en cascada por el sistema
- Integridad: ya que cada registro hijo está siempre relacionado con su padre
- Eficiencia: es muy eficiente siempre que se tengan muchas transacciones que impliquen relaciones 1:N que sean permanentes



■ Inconvenientes:

- La ejecución es compleja, no en cuanto a las dependencias de los datos, pero sí obliga a tener conocimientos en cuanto a las características del almacenamiento físico.
- Difícil de administrar: si se hace algún cambio en la ubicación de los registros habrá que modificar todas las aplicaciones que accedan a la base de datos
- No tiene independencia estructural: puesto que la navegación por los registros se hace siguiendo una ruta de acceso (padre-hijo, de izquierda a derecha, etc) si se hacen cambios en la estructura los programas no funcionarán.
- Complejidad a la hora de programar aplicaciones: es necesario que tanto los programadores como los usuarios sepan cómo están distribuidos los datos para acceder a ellos.
- Solo representa relaciones 1:N. Muchas relaciones en el mundo real no se ajustan a este tipo.

Llegó a ser el sistema de bases de datos para "[mainframes](#)" líder en los años 70 y 80 desarrollado por IBM. Actualmente no es importante entre los estándares de bases de datos pero sirvió de base para los desarrollos posteriores, de hecho muchas de sus características y ventajas se replicaron en las bases de datos actuales.

3.3.3. Modelo en red.

Es un modelo de datos propio de los sistemas comerciales de los años 70, que aún está vigente si no se piden demasiadas modificaciones al sistema.

Fue creado para representar relaciones más complejas y eficientes que las del modelo jerárquico y así imponer un estándar de bases de datos que ayudara a los diseñadores y a los programadores.

■ Características:

- Agregado de datos es un conjunto de elementos a los que se les asigna un nombre
- Registro es un conjunto de campos
- Cada campo contiene elementos.
- Se define elemento como la unidad más pequeña de información que es independiente y significativa en sí misma
- A una relación se le denomina conjunto
- Cada conjunto se compone de al menos dos tipos de registro: propietario (registro padre) y miembro (registro hijo)
- Un conjunto es una relación 1:N entre propietario y miembro



■ **Ventajas:**

- Simplicidad conceptual: es comprensible a la vista y eso facilita el diseño
- Puede manejar relaciones M:N y relaciones reflexivas
- El acceso a los datos es más flexible porque no requiere una ruta preordenada
- Se cumple la integridad de la base de datos porque un miembro no puede existir sin propietario
- Ofrece una independencia suficiente de los datos para aislar los datos del almacenamiento físico, por tanto si se hacen cambios en las características de los datos no hay que cambiar los programas de aplicación.
- Cumple los estándares, que incluyen un lenguaje de manipulación y de definición de datos, por tanto la administración y portabilidad es más fácil.

■ **Inconvenientes:**

- Complejidad del sistema: el acceso a los datos se hace leyendo un registro cada vez, por tanto los programadores y los usuarios finales deben conocer las estructuras internas, por tanto no es fácil de utilizar.
- Falta de independencia estructural: si se hacen cambios en la estructura de la base de datos, es necesario cambiar las aplicaciones. Aunque logra la independencia de los datos, no produce independencia estructural

En muchos casos se parece al modelo jerárquico con la diferencia de que el modelo en red permite que un registro tenga más de un parente, es decir que un registro puede aparecer como miembro en más de un conjunto; pero debido a las desventajas de este modelo, fue sustituido por el modelo relacional.

3.3.4. Modelo relacional. 

Fue desarrollado por Codd para IBM en los años 70, pero inicialmente los ordenadores carecían de prestaciones para poder ejecutarlo. Actualmente es el modelo más utilizado para modelar problemas reales y administrar datos dinámicamente.

Su principal ventaja es que permite que el usuario y el diseñador operar en un entorno que se percibe como un conjunto de tablas y los detalles físicos complejos los maneja el sistema.

■ **Características:**

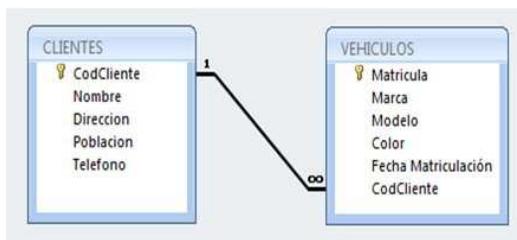
- Este modelo representa los datos y las relaciones entre ellos como una colección de tablas,
- De manera simple, una relación representa una tabla que no es más que un conjunto de filas, cada fila es un conjunto de campos y cada campo representa un valor que describe el mundo real.
- Las tablas son independientes pero se relacionan mediante un vínculo común.
- Proporciona una redundancia y una inconsistencia mínima.
- La independencia de datos de las aplicaciones y del dispositivo de almacenamiento



Ejemplo: Los datos de los CLIENTES se guardan en una tabla y los de los VEHICULOS en otra, de forma independiente. El vínculo entre las tablas CLIENTES y VEHICULOS es el CodCliente (código de cliente), que permite establecer que el cliente Raquel Marcos es propietario de dos vehículos: un Ford Focus y un Suzuki Vitara. En este ejemplo la relación es 1:N, un cliente puede tener muchos vehículos.

CLIENTES				
CodCliente	Nombre	Dirección	Población	Teléfono
1	Francisco Álvarez	C/ La Mata 9.	Alcántara	925767788
2	Raquel Marcos	C/ La Amapola 7.	Toledo	925998811
3	Carlos Revuelta	C/ La Arboleda 12.	Madrid	919090771
4	José María Cabello	C/ La plaza 72	Madrid	914409071
5	Jorge Peña	C/ Fresnedo, 23	Guadalajara	949788896
6	Dolores Manzano	C/ Autonomía, 8	Madrid	916767560

VEHICULOS					
Matricula	Marca	Modelo	Color	Fecha Matriculación	CodCliente
4534 FNG	Ford	Focus	Negro	14/04/2007	2
1203 CLL	Citroën	C4	Magenta	23/08/2005	5
3367 GHB	Suzuki	Vitara	Cobalto	16/05/2009	2
1004 JLG	Kia	Rio	Rojo	02/07/2009	6
6709 BFG	Peugeot	206	Gris plata	12/10/2006	3



■ Ventajas:

- **Independencia estructural:** los programadores, usuarios, diseñadores no necesitan conocer la ruta de acceso a los datos. Los cambios en la estructura de la base de datos no afectan a la capacidad de acceso a los datos.
- **Simplicidad conceptual:** debido a que el sistema se encarga del almacenamiento físico de los datos, los diseñadores se centran en la representación lógica de la base de datos.
- **Facilidad para diseñar y administrar y utilizar a base de datos,** debido a la independencia de los datos y estructural.
- **Capacidad para hacer consultas de forma rápida y sencilla** (mediante el lenguaje SQL)
- **Un SGBD relacional incluye elementos de software que realizan más tareas y más complejas para los usuarios y los diseñadores.**

■ Inconvenientes:

- Requiere una elevada inversión en hardware y software para evitar que sea lento, aunque esto está cambiando gracias a la evolución de la capacidad del hardware y a las mejoras de los sistemas operativos.
- El diseño deficiente es bastante común debido a la facilidad de uso de esta herramienta para personas inexpertas. **A medida que la base de datos crece, si el diseño es inapropiado, el sistema es más lento y se producen anomalías.**
- Debido a la facilidad de uso, los usuarios finales a menudo crean subconjuntos de bases de datos que pueden producir datos **inconsistencia**.



Como las desventajas son mínimas comparadas con las ventajas, se ha convertido en el modelo predominante en la actualidad, no obstante debido a que la complejidad de la realidad cada vez es mayor, se buscan alternativas de modelado con un mayor componente visual. Este modelo será estudiado ampliamente en la próxima unidad.

3.3.5. Modelo orientado a objetos.

Los modelos de bases de datos intentan representar cada vez con más fidelidad los problemas del mundo real que cada vez son más complejos. Uno de los modelos que se han desarrollado recientemente (años 90) es el modelo de bases de datos orientada a objetos.

Se denomina así porque su estructura básica es un **objeto**, que recoge tanto datos como sus relaciones. Supone una forma diferente de definir y utilizar las entidades.

Un objeto se describe como un conjunto de hechos, pero incluye también información sobre la relación que tienen los hechos dentro del objeto y con otros objetos; así como todas las operaciones que puedan ser realizadas en él.

Ejemplo: Consideremos un objeto que representan las deudas pendientes de nuestros clientes. Este objeto contiene dos atributos: el NIF del cliente y el saldo adeudado. Contiene además un método ya que si el saldo adeudado es menor de 300 Euros y no han transcurrido más de 3 meses no se le cobran intereses, pero si una de las dos condiciones no se cumple se le recargará un 5% de interés. Supongamos que queremos modificar este recargo elevándolo a un 6%. En este modelo de datos no implicaría modificar los programas de aplicación, bastaría con modificar el método recargo.

El modelo de datos orientado a objetos está basado en los siguientes componentes:

- Los objetos del modelo: equivale a una entidad individual del modelo E-R.
- Los atributos que describen las propiedades de ese objeto.
- Los objetos que comparten características similares se agrupan en clases.
- Una clase es un conjunto de objetos similares con estructura (atributos) y comportamiento (métodos) compartidos. Se podría comparar a una entidad del modelo E-R pero se diferencian en que una clase contiene una serie de procedimientos llamados métodos.
- Un método representa una acción del mundo real. Por ejemplo: localizar el nombre de un cliente, cambiar el teléfono de un cliente o imprimir su dirección. Son equivalentes a los procedimientos en un lenguaje de programación. Definen el comportamiento de un objeto.
- Las clases se organizan en una jerarquía de clase que se parece a un árbol invertido donde cada clase tiene solo un parente. Por ejemplo la clase cliente y la clase proveedor comparten una clase: persona



- La herencia es la capacidad de un objeto de heredar los atributos y los métodos de los objetos que están sobre él en una jerarquía de clase. Por ejemplo las clases cliente y proveedor, como subclases de la clase persona heredarán los atributos de la clase persona.

■ Ventajas:

- Agrega contenido semántico. En el ejemplo anterior dentro del objeto factura se incluyen las relaciones entre el cliente y la factura y entre la factura y los artículos.
- La representación visual facilita la comprensión de relaciones complejas dentro de los objetos y entre ellos.
- Mantiene la integridad de la base de datos al implementar la herencia entre objetos.
- Independencia estructural de los datos ya que son objetos autónomos.

■ Inconvenientes:

- No existen estándares de modelo de datos orientados a objetos. Sobre todo método de acceso a datos estándar.
- El método de acceso se parece al jerárquico y en red.
- El modelado y ejecución es difícil debido a que tienen mucho contenido semántico y no hay estándares.
- La complejidad y elevados requerimientos del sistema hace que las transacciones sean lentas.

Son una buena elección para aquellos sistemas que necesitan un buen rendimiento en la manipulación de tipos de datos complejos ya que proporcionan los costes de desarrollo y mantenimiento más bajos.

3.3.6. Modelos de bases de datos e internet.

La evolución de los sistemas de bases de datos ha ido siempre marcada por la búsqueda de nuevas herramientas para reflejar el mundo real lo mejor posible.

Cada modelo ha ido sustituyendo al anterior intentando eliminar sus defectos: el modelo de red sustituyó al modelo jerárquico porque podía representar relaciones M:N (muchos a muchos).

El modelo relacional reemplazó al modelo de red porque permite representar la realidad de una forma más simple, ofrece mayor independencia de los datos y admite consultas utilizando un lenguaje relativamente fácil.

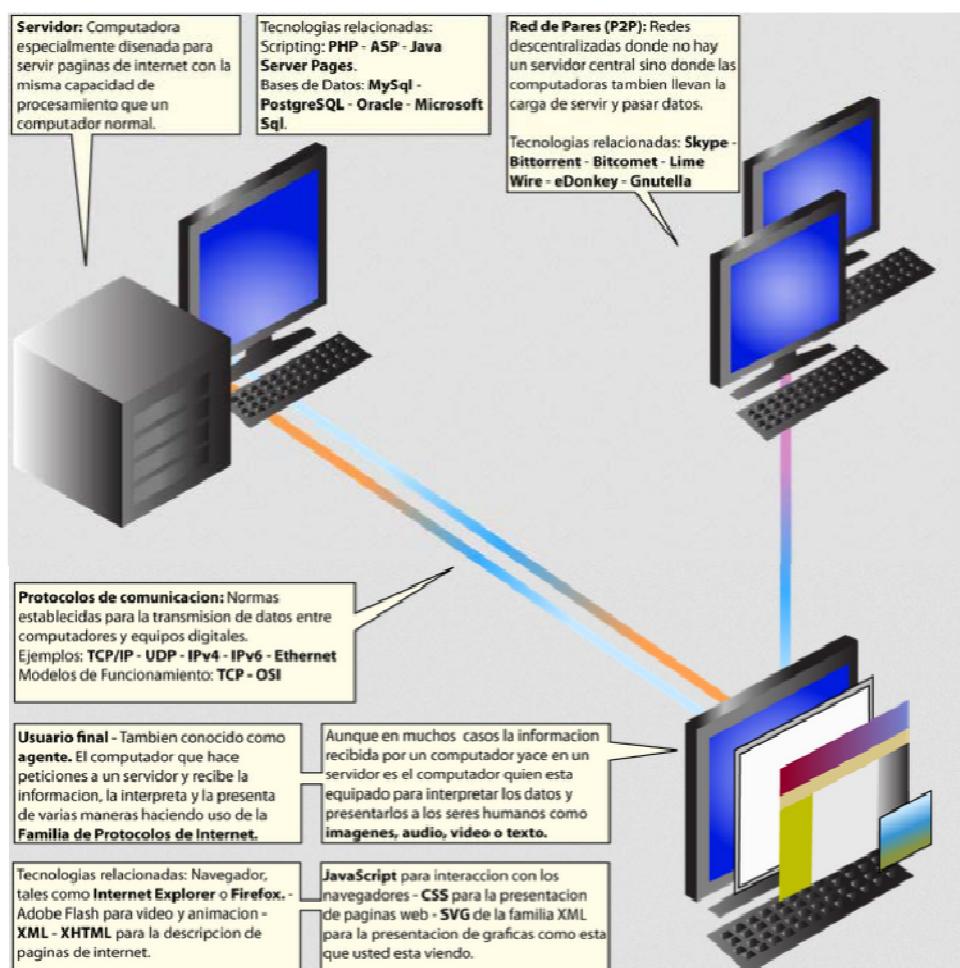
A medida que las aplicaciones se van volviendo más complejas aparece el modelo de datos orientado a objetos y el modelo relacional ampliado que incluye muchas características del modelo orientado a objetos. Ambos intentan recoger la mayor información semántica posible, pero desde orientaciones distintas: el modelo orientado a objetos se enfoca más a aplicaciones de ingeniería y científicas muy especializadas y el modelo relacional ampliado se dirige más a aplicaciones de negocios.



Sin embargo el mercado de las bases de datos ha cambiado sustancialmente con el creciente desarrollo del uso de internet en las transacciones comerciales. Ahora los esfuerzos se dirigen a la creación y desarrollo de bases de datos que se comuniquen fácilmente por internet.

Se busca:

- Acceso flexible a internet.
- Que se conecten fácilmente con distintas estructuras de datos
- Que el diseño del modelo conceptual sea sencillo.
- Que disponga de herramientas de diseño, consulta, desarrollo de aplicaciones e interfaz gráfica potentes y que faciliten el trabajo.



A pesar de todo hay que tener en cuenta que las transacciones comerciales que se realizan por internet al final se comunican con una base de datos que deberá estar bien diseñada. De no ser así ninguna interfaz de internet podrá resolver los problemas.



4.- Sistemas gestores de bases de datos.

4.1. Componentes.

Los SGBD son paquetes de software complejos que deben proporcionar una serie de servicios que permiten almacenar y explotar los datos de forma eficiente.

Los componentes principales son:

- **Lenguajes de los SGBD:** Todos los SGBD ofrecen lenguajes apropiados a cada tipo de usuarios: administradores, diseñadores, programadores de aplicaciones y usuarios finales. Los lenguajes que intervienen en un SGBD se clasifican en:

- **LDD** (Lenguaje de definición de datos): se utiliza para definir el esquema conceptual y el esquema interno de la base de datos: los objetos de la base de datos, las estructuras de almacenamiento y las vistas de los distintos usuarios. Lo emplean los diseñadores de la base de datos y los administradores.
- **LMD** (Lenguaje de manipulación de datos): se utiliza para consultar y actualizar los datos de la base de datos. Lo emplean los usuarios para consultar, insertar, modificar o borrar datos en una base de datos. A menudo estas sentencias están embebidas en un lenguaje de alto nivel llamado [lenguaje anfitrión](#)

La mayoría de los SGBD incorporan lenguajes de cuarta generación que permiten al usuario desarrollar aplicaciones de forma fácil y rápida. Se denomina también herramientas de desarrollo.

- **El diccionario de datos:** Es una guía donde se describe la base de datos con todos los objetos que la forman. Se dice también que contiene "[metadatos](#)" porque es información sobre los datos. Contiene las características lógicas como: nombre, descripción, alias, contenido y organización, donde se almacenan los datos del sistema. Identifica los procesos donde se emplean los datos y los sitios donde se necesita acceder a la información

El diccionario proporciona información acerca de:

- La estructura lógica y física de la base de datos.
 - La definición de cada uno de los objetos: tablas, vistas, índices, funciones, procedimientos, etc.
 - Los valores que toman las columnas de las tablas por defecto
 - Información que permite garantizar [la integridad](#) de los datos almacenados
 - Los privilegios y control de acceso de los usuarios
 - Normas que garanticen la seguridad de los datos.
 - Estadísticas y auditorías de los accesos a los objetos, etc.
- **Seguridad e integridad de los datos:** son una serie de mecanismos que proporciona el SGBD para garantizar un acceso correcto, seguro y eficiente a los datos. Se hace mediante un componente software que se encarga de:





- Garantizar que el acceso a los datos se permita solo a los usuarios autorizados.
- Disponer de herramientas para planificar y realizar copias de seguridad y restauración.
- Realizar los procedimientos necesarios para recuperar los datos tras un fallo o pérdida temporal.
- Ofrecer mecanismos para implantar restricciones de integridad que los datos deberán cumplir.
- Controlar el acceso concurrente de varios usuarios a los datos sin que se pierda la **consistencia**.

■ **Los usuarios del SGBD:** existen distintos tipos de usuarios que acceden al sistema. Podemos considerar:

- Programadores: Son los responsables de la creación y ajuste de las aplicaciones que ataquen a los datos. Emplean DDL, DML y cualquier lenguaje anfitrión.
- Usuarios expertos: Emplean las utilidades de la base de datos y el DML para acceder a los datos y realizar sus propios procesos sobre los objetos para los que se les ha concedido permiso.
- Usuarios ocasionales: que utilizan programas de aplicación para acceder a las bases de datos, pero que solo pueden utilizar aquellos objetos para los que se les ha dado permiso de acceso.
- Diseñadores-Administradores: Los diseñadores planifican y desarrollan las bases de datos. Definen el esquema lógico y físico de la base de datos, optimizando el almacenamiento y generando la documentación de análisis necesaria para los programadores. Cuando las bases de datos están creadas los diseñadores tienen la función de administradores. Los administradores de la base de datos gestionan la seguridad (usuarios y permisos), y la integridad de los datos asegurando que las transacciones sean correctas y no se pierdan datos. También se ocupan de crear las copias de seguridad. Tienen el máximo nivel de acceso. Utilizan fundamentalmente DDL

■ **Herramientas de la base de datos:** todos los SGBD incluyen una serie de herramientas de administración que permiten a los administradores la gestión de la base de datos

- Definir el esquema lógico y físico de la base de datos: crear, modificar y manipular
- Controlar la privacidad de los datos: gestión de usuarios y permisos

Estas herramientas cada vez incluyen mayores prestaciones

4.2.- Funciones.

Un SGBD realiza funciones que garantizan la integridad y la consistencia de los datos en una base de datos. La mayoría de estas funciones son transparentes para los usuarios finales y que las realiza el propio SGBD.



1.- Administración del diccionario de datos:

Las definiciones de los datos y sus relaciones guardan en el [diccionario de datos](#). El SGBD utiliza este diccionario para buscar las estructuras y las relaciones de los datos que cada programa solicita. Cualquier cambio que se realice en la estructura de una base de datos queda automáticamente registrado aquí, sin que el usuario tenga que modificar los programas que accedan a la estructura modificada.

2.- Administración del almacenamiento de datos:

Esta función permite al SGBD crear las estructuras necesarias para el almacenamiento de datos, liberando al usuario de tener que definir y programar las características físicas de los datos. Permite almacenar no sólo los datos, sino formularios de entrada, definiciones de filtros relacionados, de informes, reglas de validación, procedimientos, estructuras para datos con formatos de vídeo, imagen, hoja de cálculo, gráficos, etc.

3.- Transformación y presentación de datos:

El SGBD transforma los datos que se introducen en las estructuras necesarias para guardarlos. Transforma las solicitudes lógicas en comandos que localizan y recuperan físicamente los datos, liberando así al usuario de esa tarea.

4.- Administración de la seguridad:

Crea un sistema de seguridad que establece unas reglas que determinan que usuarios pueden acceder a la base de datos, a qué datos pueden tener acceso y qué operaciones pueden realizar.

5.- Control de acceso de usuarios múltiples:

Es el control de concurrencia. Permite acceder a la base de datos a múltiples usuarios creando unos algoritmos complejos para no comprometer la integridad de la base de datos.

6.- Administración de tareas de respaldo y recuperación:

Proporciona utilidades que permiten realizar procedimientos de respaldo y recuperación rutinarios y especiales, como un fallo en el suministro eléctrico, un sector defectuoso en el disco, etc.

7.- Administración de la integridad de los datos:

Controla que se cumplan las reglas de integridad de los datos y de las relaciones, con lo que se reduce al mínimo la **redundancia** y se aumenta la **consistencia**.

8.- Lenguajes de acceso a la base de datos e interfaces de programación de aplicaciones:

Permite acceder a los datos utilizando un lenguaje de consulta.

Lenguaje de consulta: Se trata de un lenguaje que no tiene procedimientos. Debemos sustituir la palabra término, por el término en cuestión y las siglas de la UT. Si recordáis, en el glosario no añadimos los términos sin más, sino que les concatenábamos la coletilla (SIGXX, es decir que el usuario especifica al sistema qué debe hacer y no cómo). Tiene dos componentes:





- Lenguaje de definición de datos (DDL): contiene instrucciones para definir las estructuras donde se alojan los datos.
- Lenguaje de manipulación de datos (LMD): contiene instrucciones que permiten a los usuarios extraer datos de la base de datos.

También permite a los programadores acceder a los datos mediante lenguajes de procedimientos y proporciona utilidades administrativas para crear, ejecutar y mantener la base de datos.

9.- Interfaces de comunicación de bases de datos:

Permiten que la base de datos acepte solicitudes de usuarios conectados en una red de ordenadores y a través de internet. Esta comunicación con el SGBD puede establecerse de varias formas:

- Realizar peticiones mediante formularios desde el explorador de internet.
- Publicar informes en internet que pueda explorar cualquier usuario.
- Conectarse a otros sistemas mediante aplicaciones como correo electrónico.

PARA SABER MÁS

Para tener más información sobre el diccionario de datos:

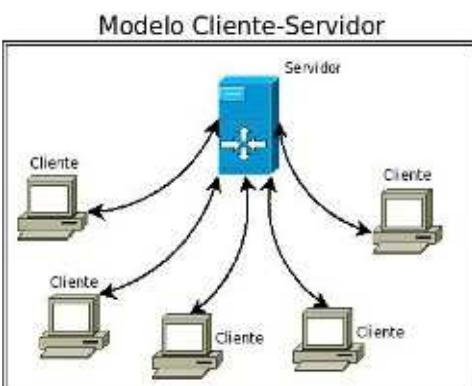
http://es.wikipedia.org/wiki/Diccionario_de_datos

4.3.- Arquitectura Cliente/Servidor.

El objetivo de un sistema de bases de datos es facilitar el desarrollo y ejecución de aplicaciones. Por tanto, desde un punto de vista amplio, un sistema de bases de datos posee una estructura compuesta de dos partes: un servidor y un conjunto de clientes.

El **servidor** permite llevar a cabo las funciones propias del SGBD, se puede decir que el servidor es en sí, el SGBD.

Un **cliente** de una base de datos es cada consumidor de recursos de la base de datos: las aplicaciones del servidor, las aplicaciones de usuario y cualquier otro elemento de aplicación que acceda al servidor.



Los dos elementos de la base de datos, clientes y servidor, se pueden ejecutar en la misma máquina o en máquinas distintas, interconectadas a través de algún sistema de comunicación. Lo habitual es que sean máquinas distintas tal como se ve en el esquema.



Según el número de servidores y la forma de acceder un cliente a los mismos se tienen los siguientes tipos de estructuras de bases de datos:

- Basada en anfitrión
- Cliente/servidor
- Cliente/multiservidor
- Distribuidas

Los sistemas de bases de datos pueden clasificarse también según la ubicación de la base de datos. Por ejemplo si la base de datos está localizada en un solo sitio se denomina sistema **centralizado** y el que soporta una base de datos distribuida en varios sitios se llama sistema **distribuido**.

- En los **SGBD centralizados** los datos se almacenan en un solo ordenador. Los SGBD centralizados pueden atender a varios usuarios, pero el SGBD y la base de datos en sí residen por completo en una sola máquina.
- En los **SGBD distribuidos** la base de datos real y el propio software del SGBD pueden estar distribuidos en varios sitios conectados por una red. El proceso distribuido exige la presencia de algún software que se encargue de gestionar las comunicaciones entre las distintas máquinas que participan en el proceso. Muchos SGBD distribuidos emplean una arquitectura cliente-servidor.

El **software** de este tipo de arquitecturas posee varios componentes que se pueden asociar al cliente o al servidor:

- **Software de gestión de datos:** normalmente reside en el servidor y lleva a cabo la gestión de los datos que requieren las aplicaciones.
- **Software de interacción con el usuario y presentación:** suele residir en el cliente e implementa las funciones de una interfaz gráfica de usuario.
- **Software de desarrollo:** suele residir en el cliente y se utiliza para desarrollar aplicaciones.
- **Otros elementos de software que facilitan la conexión cliente-servidor:** tanto en el cliente como en el servidor se instala software de sistemas operativos en red, de aplicaciones específicas de base de datos, de comunicaciones, etc.

4.3.1.- Distintas configuraciones de la arquitectura cliente/servidor.

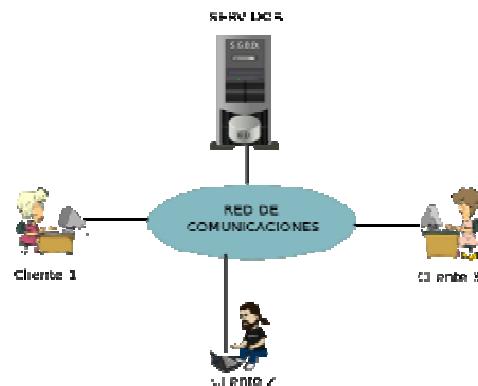
- **Basada en anfitrión:** cuando la máquina cliente y la máquina servidor es la misma. Los usuarios se conectan directamente a la máquina donde se encuentra la base de datos.
- **Cliente-servidor:** la base de datos reside en la máquina servidor y los usuarios acceden a la base de datos desde la máquina cliente a través de una red.
- **Proceso distribuido:** El empleo de arquitecturas cliente-servidor da origen al procesamiento distribuido, que consiste en repartir el proceso de los datos en varias máquinas interconectadas mediante algún tipo de red.



- **Basada en servidores de aplicaciones:** esta configuración permite el uso de aplicaciones en redes WAN e internet. Permite que las aplicaciones se ejecuten en máquinas clientes que no requieren ninguna administración. Cualquier PC que ejecute un navegador puede acceder a las aplicaciones.

■ Cliente-servidor.

Los sistemas cliente-servidor poseen **arquitectura en dos niveles** ya que se distinguen dos funcionalidades básicas: cliente y servidor. Estas funcionalidades se refieren a los ordenadores que ejecutan los procesos, de forma que un equipo informático puede actuar como servidor de bases de datos en determinadas aplicaciones y como cliente para otras.



Para su correcto funcionamiento, la base de datos, necesita estar instalada en un determinado equipo, con un Sistema Operativo y el software de red que permita la comunicación del servidor con los clientes.

- ✓ **La aplicación cliente:** es la responsable de verificar y aceptar las entradas de los usuarios. Si se acepta la petición del usuario envía una consulta al servidor de bases de datos. Esta petición es procesada por el servidor, que envía de vuelta a la aplicación cliente los resultados de la consulta. La aplicación cliente formatea los datos de acuerdo con la petición y los muestra al usuario. Normalmente el cliente posee un interfaz de programación de aplicaciones (API), que es el encargado de enviar las consultas al servidor. Los programadores pueden crear aplicaciones o Software de desarrollo.
- ✓ **Red de comunicaciones.** El software es independiente del tipo de redes utilizado para comunicar las aplicaciones cliente con el servidor, con lo que hay cierta independencia del software de la base de datos respecto del software de red.
- ✓ **El servidor de la base de datos:** acepta las consultas de los clientes, los procesa y devuelve los resultados. El lenguaje de consulta habitualmente empleado en los SGBD cliente/servidor es SQL, que implementa instrucciones tanto del LDD (lenguaje de definición de datos) como del LMD (lenguaje de manipulación de datos).

■ Ventajas de los sistemas de bases de datos cliente/servidor

- ✓ Distribuyen los procesos entre el cliente (que ejecuta el interfaz de usuario junto con las aplicaciones) y el servidor (que gestiona el motor de datos y el software de acceso centralizado a los datos). Esta división de tareas evita la disminución de velocidad de ejecución que ocurre en los procesos de datos centralizados, aunque se genera un cuello de botella por la velocidad del tráfico de red, ya que se emplean redes tanto para transmitir información como en las aplicaciones de bases de datos.



- ✓ Empleo de ordenadores personales estándar para el servidor y los clientes, en lugar de grandes equipos.
- ✓ Son sistemas escalables y modulares, es decir, se puede aumentar la cantidad de servicios prestados por los ordenadores y reemplazarlos por equipos nuevos sin que sea afectado por el cambio de SGBD. Aunque el empleo de diferentes equipos aumenta los problemas de mantenimiento físico y lógico.
- ✓ Disponibilidad de herramientas de desarrollo de calidad: lenguajes de 4^a generación orientados a objetos y a procedimientos y entornos gráficos de desarrollo, herramientas de modelado de datos, etc.

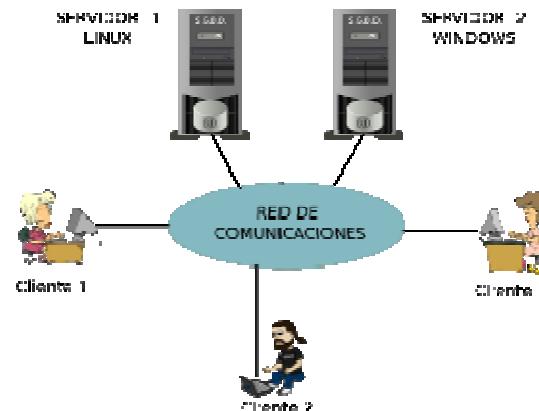
■ Proceso Distribuido.

Una base de datos **cliente-multiservidor** es aquella en la que el cliente se puede conectar a más de un servidor simultáneamente o bien sólo puede conectarse a un servidor en cada sesión cliente.

Cuando una aplicación cliente accede a datos de distintos servidores, se denomina **Sistema de Bases de Datos Distribuidas**.

La base de datos está distribuida en más de una máquina servidora. Los usuarios no tienen por qué conocer la ubicación física de los datos con los que trabajan y han de acceder simultáneamente a varios servidores.

Para el usuario (cliente) es indistinto si los orígenes de datos son únicos o múltiples, pero para el administrador es un trabajo extra considerable, ya que es necesario crear un sistema centralizado de administración de servidores y clientes o administrar de forma individualizada cada servidor para poder mantener en sus sitios usuarios, datos y aplicaciones, con controles de seguridad, acceso y concurrencia.



En la arquitectura de una Base de datos distribuida se dan los siguientes elementos:

- ✓ Varias bases de datos se pueden almacenar en SGBD de diferentes fabricantes.
- ✓ Los ordenadores donde se ejecutan estos sistemas pueden ser distintos.
- ✓ Que ejecuten distintos Sistemas operativos.
- ✓ Las redes que comunican los elementos de las bases de datos pueden tener distintos protocolos y arquitecturas.

Sea cual sea la implementación real de una base de datos distribuida, son dificultades a nivel de hardware, de software y de protocolos y sobre todo de administración, pero no son ningún problema para el usuario.



■ Servidores de aplicaciones.

Esta configuración permite el uso de aplicaciones en redes de área amplia (WAN) e Internet. Permite que las aplicaciones se ejecuten en máquinas clientes que no requieren ninguna administración. Cualquier ordenador que ejecute simplemente un navegador puede acceder a las aplicaciones.

La arquitectura de las Bases de datos basadas en servidor de aplicaciones se denomina arquitectura a tres niveles:

- Por un lado la parte servidor se ejecuta en el **servidor de bases de datos**, mientras que la parte cliente se ejecuta en dos niveles:
- Una máquina **servidor de aplicaciones**.
- La **máquina cliente**.
- La carga de trabajo se realiza de forma centralizada en el servidor de aplicaciones mientras que las máquinas cliente ejecutan la parte de la aplicación que actúa de interfaz de usuario.

La arquitectura cliente-servidor frente a la arquitectura basada en servidores de aplicaciones tiene las siguientes **diferencias**:

- ✓ La arquitectura cliente-servidor requiere que las aplicaciones se instalen en cada puesto de trabajo, aumentando los costes de administración, además impone grandes exigencias a la red, lo que imposibilita el uso de redes de área extensa (WAN) e internet.
- ✓ En la arquitectura basada en servidores de aplicaciones las aplicaciones se instalan en puestos de trabajo que pueden ser cualquier Pc con un navegador, sin necesidad de ninguna administración.



GLOSARIO

Borrado lógico

El registro no se borra físicamente, sino que se hace mediante la grabación de una marca en el mismo.

Clave

Un campo, o combinación de campos, que permita identificar cada registro de forma única, es decir, que no pueda haber dos registros que tengan la misma información en él.

Clave alfanumérica

Clave compuesta de letras, números y otros caracteres especiales.

Colisiones

Situación que se produce cuando de registros con claves distintas se obtiene la misma dirección de almacenamiento después de aplicar un algoritmo de transformación.

Comité ANSI-SPARC

Es un grupo de normalización creado en 1969 para estudiar el impacto de los S.G.B.D. en los sistemas de información y cuyos resultados, publicados en 1975 dieron lugar a la arquitectura a 3 niveles: interno, externo y conceptual.

Compactación

Son técnicas para disminuir el espacio ocupado en disco por los datos, eliminando los problemas derivados de eliminaciones y actualizaciones que dejan huecos que retardan las búsquedas.

Consistencia

Que los datos cumplan las reglas de validación y las restricciones establecidas

Diccionario de datos

Contiene información sobre los elementos que constituyen la base de datos: tablas, registros, campos, relaciones, descripciones, etc.

Esquema

Representación abstracta del mundo real. A la descripción de una base de datos mediante un modelo de datos se le denomina esquema de la base de datos.

Hashing

Técnicas para la resolución de colisiones entre sinónimos. Hash es un algoritmo para generar claves únicas.



Huecos

Direcciones de almacenamiento que no son ocupadas por ningún registro después de aplicar un algoritmo de transformación hash

Inconsistencia

Cuando la repetición de datos (redundancia) provoca que los datos duplicados no coincidan.

Integridad

Se refiere a que los datos de una base de datos sean correctos y completos. Cuando los contenidos se modifican pueden añadirse datos no válidos como por ejemplo un pedido que especifica un producto que no existe.

Lenguaje anfitrión

Lenguaje de alto nivel que permite incrustar instrucciones en otro lenguaje o lenguaje huésped.

Metadatos

Se denomina así al contenido del diccionario de datos. Almacena las descripciones de los datos que se guardan en la base de datos. Guarda “datos” sobre los “datos”.

Mainframes

Macrocomputador. En la actualidad se utiliza esta palabra para referirse a los grandes ordenadores. Con el aumento de potencia de los llamados miniordenadores, la frontera entre éstos y los mainframes está cada vez menos clara.

Objeto

Es un elemento que integra datos y los procedimientos para manipularlos.

Orden lógico

El orden en que son dados de alta y recuperados los registros

Orden físico

El orden en que están grabados los registros en el soporte

Punteros

Un puntero es un campo adicional que se añade a un registro para indicar cuál es el registro anterior o siguiente en orden lógico.

Redes WAN

Redes de área extensa capaces de cubrir distancia de 100 a 1000 km



Relaciones reflexivas

Vínculo de una entidad consigo misma

Redundancia

Datos que se repiten de forma innecesaria en una base de datos

Segmentos

Conjunto de registros almacenados en posiciones consecutivas en un dispositivo de almacenamiento y que se direccionan como un bloque. El espacio total de almacenamientos se divide en segmentos de una longitud determinada a los cuales se accede por su dirección.

Sinónimos

Direcciones de memoria que resultan de la transformación de las claves y que ya están ocupadas por otros registros