IES CHAN DO MONTE
C.S. de Desarrollo de Aplicaciones Multiplataforma
Módulo Base de datos

UNIDAD 1: Sistemas de almacenamiento de la información

Índice

1. Sistemas de información

En el entorno del mercado actual, la competitividad y la rapidez de maniobra de una empresa son imprescindibles para su éxito. Para conseguirlo **existe cada vez una mayor demanda de datos** y, por tanto, **más necesidad de gestionarlos**.

¿Dónde y de qué manera se almacenan y gestionan los datos que utilizamos diariamente?

Nos damos cuenta de que **cualquier *acción de nuestra vida cotidiana* o en la mayoría de los ámbitos de actividad**, está relacionado con los **datos**, **su almacenamiento** y **gestión**.

Esta demanda de datos estuvo siempre patente en empresas y sociedades, pero en estos años, la demanda se ten disparado mucho más debido al acceso multitudinario a Internet.

- ✓ Cuando seleccionamos nuestro canal favorito en la TDT.
- ✓ Al utilizar la agenda del móvil para realizar una llamada telefónica.
- ✓ Cuando operamos en el cajero automático.
- ✓ Al solicitar un certificado en un organismo público.
- ✓ Cuando acudimos a la consulta del médico.
- ✓ Al inscribirnos en un curso, plataforma OnLine.
- ✓ Al realizar un pedido en una empresa.
- ✓ Si utilizamos un GPS.
- ✓ Cuando reservamos unas localidades para un evento deportivo o espectáculo.
- ✓ Cuando jugamos en un videojuego.
- ✓ Cuando consultamos cualquier información en Internet. (Bibliotecas, enciclopedias, museos, etc.)
- ✓ Al registrarte en una página de juegos OnLine, redes sociales o foros, etc.

El propio nombre **Informática** hace referencia al hecho de ser una ciencia que trabaja con la **información**. Desde que se comenzaron a fabricar ordenadores, la información fue considerada como uno de los pilares de los ordenadores digitales. Por lo eres las bases de datos son una de las aplicaciones más antiguas de la informática.

En informática se conoce cómo **dato** cualquier *elemento informativo que tenga importancia para el sistema*. Desde el inicio de la informática, el dato se reconoció como el elemento fundamental de trabajo en un ordenador. Por lo tanto, se han realizado numerosos estudios y aplicaciones para mejorar la gestión que se realiza de los datos desde las computadoras.

Para poder almacenar datos y cada vez más datos, el ser humano ideó nuevas herramientas: archivos, cajones, carpetas y fichas en las que se almacenaban estos datos.

Antes de la aparición del ordenador, el tiempo requerido para manipular estos datos era enorme. Sin embargo el proceso de aprendizaje era relativamente sencillo ya que se usaban elementos que el usuario reconocía perfectamente.

Por esa razón, la informática adaptó sus herramientas para que los elementos que el usuario maneja en el ordenador se parezcan a los que utilizaba manualmente:

Así en informática se sigue hablando de ficheros, formularios, carpetas, directorios, ...

Un **sistema de información** es un conjunto de actividades que administran la información relevante en una entidad, generalmente, una empresa. El sistema de información es lo que se encarga de la distribución de los datos según unos determinados requisitos, de la correcta compartición de la información entre sus usuarios y de su almacenamiento en soportes adecuados.

Realmente un sistema de información sólo incluye la información que nos interesa de la empresa y los elementos necesarios para gestionar esa información. Un sistema de información genérico está formado por los siguientes elementos:

- **Recursos físicos**. Carpetas, documentos, equipamiento, discos, ...
- **Recursos humanos**. Personal que maneja la información.
- **Protocolos**. Normas que debe cumplir la información para que sea manejada (formato de la información, modelo para los documentos, ...)

Las empresas necesitan implantar estos sistemas de información debido a la competencia, que las deber a gestionar de la manera más eficiente sus datos para una mayor calidad en la organización de las actividades de los subsistemas empresariales.

Los grandes volúmenes de información manejados por un sistema se agrupan en conjuntos más pequeños para poder ser nombrados y representados en las aplicaciones que los emplean:

- La unidad más pequeña que permite representar información es el **bit**, que admite dos valores: **0 y 1**.
- Un grupo de **8 bits** forman un **byte**.

- Cada dato a lo que se pode hacer referencia en un sistema de información se denomina campo y está **formado por un conjunto de bytes**.
- Los datos que forman parte de una entidad común se agrupan en uno **registro**, que está formado por un **conjunto de campos**.
- **Todos los registros** del incluso tipo forman un **archivo**.
- El **conjunto de archivos** que representa la información de un sistema forma una **base de datos**.

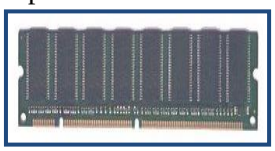
2. Estructura básica de almacenamiento: el archivo

2.1 Almacenamiento de la información

Todas las aplicaciones **informáticas trabajan con datos o información que deben ser almacenados en un medio físico,** como por ejemplo, discos duros, DVD, CD o Pendrive. Estos medios forman una jerarquía que distingue entre tes niveles de almacenamiento: primario, secundario e intermedio.

- **Almacenamiento primario**:

Se refiere a aquellos medios sobre los que la CPU del ordenador puede acceder directamente y por tanto, más rápidamente. Nos referimos a la memoria principal.



La **memoria** da al **procesador almacenamiento temporal para programas y datos**. Todos los programas y datos deben transferirse a la memoria desde un dispositivo de entrada o desde el almacenamiento secundario (disco duro, disquete), antes de que los programas puedan ejecutarse o procesar los datos.

Esta memoria permite almacenar los datos de entrada, instrucciones de los programas que se están ejecutando en ese momento, los datos intermedios resultados del procesamiento y los datos finales que se preparan para la salida.

A esta memoria se la conoce como **RAM y es una memoria volátil**.

- **Almacenamiento Secundario**:

La mayoría de la información guardada en la RAM se borra cuando se apaga la computadora. Por lo tanto, se necesitan **formas permanentes de almacenamiento para guardar y recuperar programas de software y archivos de datos que desee usar a diario**. Los dispositivos de secundario fueron desarrollados para satisfacer esta necesidad.

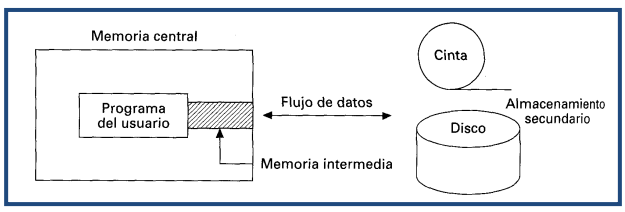
El **almacenamiento secundario** es un **medio de almacenamiento permanente** (no volátil). Son dispositivos más lentos, pero de mayor capacidad.

Los tipos más comunes de dispositivos de almacenamiento secundario son: Unidades de disco duro, unidades de CD, unidades de DVD, pendrive, memorias flash.

El disco duro es el sistema de almacenamiento más importante y en el se guardan los archivos de los programas - como los sistemas operativo (Windows 7, Windows 2008,…), las hojas de cálculo (Excel) los procesadores de texto (Word, OpenOffices), los juegos etc..

- **Almacenamiento intermedio:**

La transferencia de información entre las unidades de almacenamiento secundario y la memoria es mucho más lenta en comparación con el acceso a memoria RAM por parte de la CPU. Para acelerar las operaciones de entrada/salida entre las unidades de almacenamiento secundario y la memoria principal y la CPU tenga que esperar por los datos para procesarlos, se utilizan unas áreas de almacenamiento intermedio o Buffers.



Un **buffer (o búfer)** en informática es un espacio de memoria, en el que se almacenan datos para evitar que el programa o recurso que los requiere ya sea hardware o software, se quede sin datos durante una transferencia.

Leyendo o escribiendo la información del disco una sola vez y luego manteniéndola en la memoria hasta que no sea necesaria, se aceleran todas las lecturas o escrituras posteriores con respecto a la primera. La memoria utilizada para el almacenamiento intermedio es parte de la memoria RAM y se le conoce como *buffer caché*.

2.2 Sistemas de archivos de datos

Los **datos para su persistencia** deben ser **almacenados en componentes de almacenamiento permanente**. En estos medios, **los datos se estructuran en archivos** (también llamados ficheros).

Los **archivos de datos** son *un **conjunto de datos estructurados***, que se tratan como una unidad y que se encuentran almacenados sobre un dispositivo de almacenamento externo.

Como estructura de datos, los archivos permiten el almacenamiento permanente y la manipulación de gran número de datos. Los archivos **están formados por una colección de registros** y estos, constituyen una **agregación de campos** que se caracterizan por su **tamaño** o longitud y el **tipo de dato**.

- **Archivos**
  - Un **archivo** o fichero es una **colección de registros relacionados entre sí**, con aspectos en común y organizados para un propósito específico. Como por ejemplo: el fichero de nóminas de una empresa, el fichero de alumnos del instituto, etc.
- **Registros**

  Cuando los ficheros almacenan datos, se dice que constan de registros.
  - Un **registro** es una colección de información, normalmente, relativa a un mismo elemento u objeto. El registro es una **colección de campos lógicamente relacionados,** que pueden ser tratados como una unidad por algún programa.

    Un ejemplo de registro sería la información de un determinado alumno, que contiene los campos de nombre, apellidos, dirección, teléfono, fecha de nacimiento, etc. Si el archivo contiene datos de 1000 personas, constará de 1000 registros.
  - Los registros pueden ser de **longitud fija** o de **longitud variable**, y además, pueden estar o no constituidos por el mismo número de campos por registro.
- **Campos**
  - Un **campo** es cada una de las **unidades elementales en las que se divide el registro**, tales cómo nombre, apellidos, teléfono, etc.
  - El campo se caracteriza por **su tamaño** o longitud y por su **tipo de datos** (cadena de caracteres, entero, lógico, ...). Los campos pueden incluso variar su longitud, pero la mayor parte de los lenguajes de programación no manejan campos de longitud variable.

NOMBRE	DIRECCIÓN	FECHA DE NACIMIENTO	ESTUDIOS	SALARIO

- **Clave**
  - El **campo o grupo de campos que identifican unívocamente a cada registro** de un archivo se denomina *clave (key)*.

Operaciones

Las operaciones que se pueden realizar sobre un archivo son:

- **Creación** de un archivo: Es la operación mediante la que se **establece la estructura y posición** del archivo en el dispositivo de almacenamiento.
- **Consulta**: Permite al usuario **acceder al archivo** de datos para **conocer el contenido** de uno, varios o todos sus registros.
- **Actualización** (altas, bajas, modificación, consulta): Permite tenor **actualizado** el archivo mediante las siguientes operaciones:
  - **Inserción** de un registro nuevo en el archivo.
  - **Supresión** de un registro existente.
  - **Modificación** del contenido de uno registro.
- **Clasificación**: Es la **ordenación de los registros** del archivo teniendo en cuenta el valor de un campo específico. Esta ordenación puede ser ascendente o descendente.
- **Reorganización**: Las operaciones realizadas en los archivos, modifican la estructura inicial de este, lo que implica que las operaciones de acceso a los registros sean cada vez más lentas. La reorganización consiste en **realizar, a partir del archivo modificado, una copia en un nuevo archivo, con el fin de obtener una estructura lo más eficiente posible**.
- **Destrucción** (borrado): Cuando el archivo deja de tener validez, conviene **borrarlo del dispositivo** de almacenamiento para no desperdiciar espacio.
- **Fusión**: Permite obtener un **archivo a partir de otro**s.
- **Rotura o estalido**: Operación para **obtener varios archivos** a partir de un archivo inicial.

3. Tipos de ficheros y formatos

Un ordenador almacena muchos tipos de información, desde datos administrativos, contables y bancarios hasta música, películas, páginas webs, etc.

- El **formato y tipo de fichero determina la forma de interpretar la información que contiene**, ya que, en definitiva, el único que se almacena en un fichero es una ristra de bits (ceros y unos), y se hace necesario tener un modo de interpretar la información que contiene.

Por ejemplo, para almacenar una imagen en un ordenador, se puede usar un fichero binario bmp que almacena un vector de datos con los colores de los píxeles que forman la imagen. Además, la imagen posee una paleta de colores y unas dimensiones, información que también se incluye en el fichero. Todos estos datos se ordenan segundo un formato, y el sistema operativo, o la utilidad que trate los gráficos, debe conocer este formato para poder extraer los píxeles y mostrarlos por pantalla en la forma y dimensiones correctas. Si abrimos el gráfico con una utilidad como el bloc de notas, que sólo sabe interpretar texto, el resultado será ilegible e incomprensible.

- Los ficheros se clasifican **según su contenido**:
  - **Ficheros de texto**
  - **Ficheros binarios**.

  ✓ **Texto**: los **bits** almacenados pueden **ser traducidos**, polo sistema operativo, **a caracteres alfabéticos y números** que entiende el ser humano,

  ✓ **Datos binarios**, tratados cómo componentes de **estructuras de datos más complejas**, como ficheros que almacenan sonido, video, imágenes, etc.

- **Ficheros de texto**, también llamados ficheros planos o ficheros ascii.

  Nota; El vocablo **ascii** es un acrónimo de American Standard Code for Information Interchange -Código Estándar Estadounidense para el Intercambio de Información-. Es un **estándar que asigna un valor numérico a cada carácter**, con el que se pueden representar los documentos llamados de texto plano.

Los ficheros de texto, aun que **no necesitan tener un formato para ser interpretado**, suelen tener extensiones para conocer qué tipo de texto tiene el fichero, como por ejemplo:

- **Ficheros de configuración**: Su contenido es texto sobre configuraciones del sistema operativo o de alguna aplicación. Pueden tener extensión .ini, .inf, .conf
- **Ficheros de código fuente**: Su contenido y texto con programas informáticos. Como por ejemplo: .c, .java, .sql
- **Ficheros de páginas web**: Las páginas webs son ficheros de texto con hipertexto que interpreta un navegador: .html, .php, .xml, .css
- **Formatos enriquecidos**: Son texto que incluyen códigos de control para ofrecer una visualización del texto más elegante.

- **Ficheros binarios**, son todos los que no son de texto y requieren un **formato para ser interpretado**. Como por ejemplo:
  - **De imagen**: .jpg, .gif, .mp, entre otros
  - **De vídeo**: .mpg, .avi, .mov
  - **Comprimidos o empaquetados**: .zip, .gz,. Z, .tar, .lhz
  - **Executables o compilados**: .exe, .com, .cgi
  - **Procesadores de textos:** .odt, .doc
  - **Ficheros de datos**:.dat

Nota: Los **ficheros que componen una base de datos son de tipo binario**, ya que la información que almacena debe tener una estructura lógica y organizada para que las aplicaciones puedan acceder a ella de manera universal, esto es, siguiendo un estándar. Esta estructura lógica y organizada, generalmente es muy difícil de expresar mediante ficheros de texto, por lo tanto, la información de una base de datos se guardaría en uno o varios ficheros binarios.

4. Organizaciones físicas de datos

4.1 Estructura y métodos de acceso

➢ Denominamos **acceso,** al procedimiento necesario para **situarse en uno registro determinado**, con el fin de realizar una **operación de lectura o escritura** en él.

Según las características del soporte empleado y el **modo en que estén organizados los registros**, existen básicamente tres modos de acceso:

- **Acceso secuencial**, que implica el acceso a un archivo segundo a orden de almacenamiento de sus registros.
- **Acceso directo**, en el que los registros pueden **leerse y escribirse directamente en la posición física** que ocupan en el archivo.
- **Acceso por índice,** consiste en crear un **índice ordenado con las claves del archivo**. Para acceder a un registro se **busca secuencialmente la clave en el índice**, que lleva asociada la dirección real del registro en el archivo, en el cual se lee y escribe directamente.

4.2 Organización

➢ La **organización** de un archivo define **el modo en el que los registros se disponen sobre el soporte de almacenamiento**.

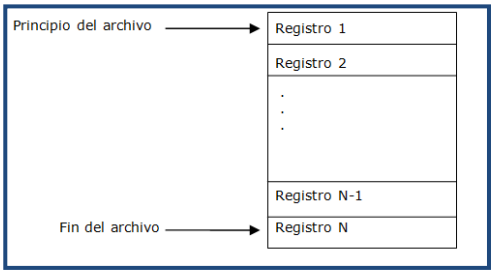
Es el modo como se estructuran los datos en un archivo. En general, se consideran los siguientes tipos de ficheros teniendo en cuenta su organización:

4.2.1 Ficheros secuenciales

✓ Un archivo con organización secuencial es una **sucesión de registros almacenados consecutivamente** sobre el soporte externo, de tal manera que para acceder al registro n, es obligatorio pasar por todos los n-1 registros que le preceden.

✓ El **orden físico** en la que fueron grabados los registros es el **orden de lectura** de estos.

✓ Los ficheros organizados secuencialmente contienen un registro particular (el último) que almacena una **marca de fin de archivo** (detectable mediante la función **EOF o FF**)



Ventajas:

- **Buen aprovechamiento del soporte** de almacenamiento ya que no hay espacios entre los registros.
- Es la **más adecuada cuando la tasa de actividad es alta**, y decir, suponen una rapidez de acceso cuando hay un gran porcentaje de registros consultados (cuando se deben tratar la mayor parte de los registros)

Inconvenientes:

- **Lenta localización de un único registro**, ya que este tipo de organización exige comenzar a leer desde el principio del archivo.
- Es necesario **un archivo auxiliar** si se quieren **realizar actualizacións** (bajas, altas o modificaciones)

4.2.2 Ficheros con organización relativa, directa o aleatoria

✓ Los datos **se sitúan** en el archivo y **se accede la ellos directamente mediante su posición**, y decir, al lugar relativo que ocupan.

✓ La **posición de cada registro** en el soporte **depende sólo del valor de su clave**. Por lo tanto, **cada registro de un fichero con organización directa debe tener una clave** que lo identifique de forma única. Los registros se van a almacenar segundo el valor de la clave.

- ✓ El **espacio físico** del que se disponen para lo fichero **se divide en un conjunto de posiciones**, todas de igual tamaño y capacidad, enumeradas de 1 a N, siendo 1 la primera y N la última posición del fichero.
- ✓ La **clave va a servir para indicar en qué posición se encuentra el registro** con **respecto a primera posición del fichero** (**posición relativa**). La correspondencia entre la clave y su posición debe ser programada y la determinación entre lo registro y su posición física se obtiene mediante una fórmula.

Nota: En la práctica el programador no gestiona directamente direcciones absolutas, sino direcciones relativas respeto al principio del archivo. Esto permite diseñar el programa con independencia de la posición absoluta del archivo en el soporte.

La **relación entre la posición relativa y la clave** pode obtenerse de las siguientes formas:

- **Direccionamento directo**
  Consiste en tratar el **valor clave como la propia dirección relativa**. Esto pode hacerse cuando las claves son numéricas y consecutivas y su rango varía de 1 a P, siendo P un valor cercan al número de posiciones del fichero.

  Posición relativa= clave

- **Direccionamento aleatorio (indirecto o calculado)**
  Cuando las claves no son numéricas consecutivas (como por ejemplo DNI) o son claves alfabéticas no se pode emplear el direccionamiento directo.
  En estos casos se empleará un **algoritmo de direccionamiento**: se aplicará **una función de conversión de clave o función hash**.

Suponiendo que N es el número de posiciones disponibles para el archivo, el algoritmo de direccionamiento convierte cada valor de la clave en una dirección relativa d, comprendida entre 1 y N.
El algoritmo de transformación de clave y el tratamiento de sinónimos corre a cargo del programador

Problemas que presentan los algoritmos de transformación de claves :

- Al aplicarlos sobre **claves de registros diferentes** de un incluso fichero, **devuelvan la misma dirección** relativa, con el cual se produce una **colisión**.
  Como los dos registros no pueden ser almacenados en la misma dirección deben preverse que esto suceda y tratar las posibles colisiones que se produzcan.
- Otro problema que puede ocurrir es que **no se generen todas las direcciones comprendidas entre 1 y N** (se generen **huecos**)

MANEJO DE COLISIONES

El tratamiento de una colisión pode hacerse usando almacenamiento externo (en un área separada llamada de desbordamiento u overflow) o almacenamiento interno (en el espacio reservado para lo fichero).

- **Tratamiento de colisiones externo**: En el **área de overflow o desbordamiento**. Consiste en tener una zona aparte del área de datos para almacenar los registros desbordados.
- **Tratamiento de colisiones interno**: Cuando se produce un sinónimo, lo registro que colisiona se almacena en la **celda más próxima vacía** (usando el acceso secuencial cíclico – desde la última posición se pasa a la primera).

Ejemplos de algoritmos de conversión de claves:

Direccionamiento por el **centro del cuadrado** (para claves numéricas)

- Se **multiplica** la **clave al cuadrado**
- Del resultado, se **eliminan las cifras de los extremos**
- Hacer que el número quede entre **0 y 1**
- **Multiplicar** el **número obtenido por el número máximo de direcciones** disponibles del fichero

Ejemplo: Clave = 3240 Posiciones = 1200

Cuadrado de la clave: 10497600

Dígitos centrales válidos: 4976

4976/10000= 0,4976

0,4976*1200=512,12 → Posición relativa: 512

Direccionamiento por **división** (para claves numéricas)

- Obtener **el resto de dividir la clave por el número de direcciones** asignadas al archivo. Se obtiene un número entre 0 y N-1
- **Sumarle 1 al número obtenido**

Ejemplo: Clave = 97234 Posiciones = 1200

97234 MOD 1200 = 31

Posición relativa: 32

Direccionamiento por **truncamiento** (para claves numéricas)

- **Eliminar los primeros k o bien los últimos m dígitos** de un número de n dígitos.
- Hacer que el número quede entre **0 y 1**
- **Multiplicar el número obtenido polo número máximo de direcciones** disponibles del fichero

Ejemplo: Clave = 2321345 Posiciones = 1200

Eliminamos los 3 primeros dígitos: 1345

1345/10000= 0,1345

0,1345*1200=161,4 → Posición relativa: 161

Para claves numéricas

- **Convertir la clave en numérica**. Como por ejemplo: A=1, B=2, ... Así ABC5 -> 1235
- Aplicar **cualquier método** de los empleados con la clave numérica obtenida.

Ventajas

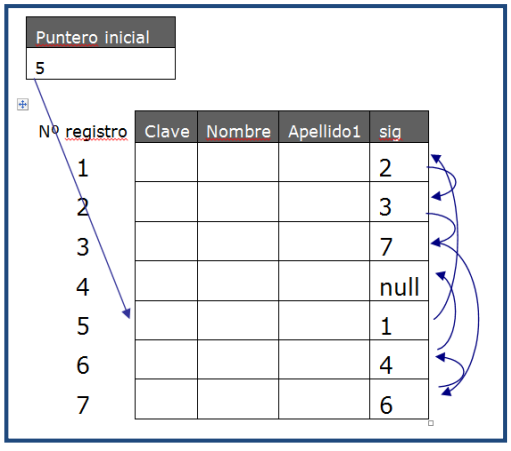
- El acceso a un registro concreto es directo. **Mayor rapidez de acceso**.
- Las **actualizaciones** de cualquiera registro también son **inmediatas**, sin que se deban emplear registros auxiliar de copia.

Inconvenientes:

- **Mal aprovechamiento del soporte** (por existir huecos).
- Se deben tener **programadas soluciones para cuando se producen colisiones**.

4.2.3 Ficheros secuenciales encadenados

- Son **ficheros secuenciales** gestionados mediante **punteros**.
- **Cada registro posee un puntero** que indica la **dirección del siguiente registro** y que se pode modificar en cualquier momento.
- El puntero permite **recorrer los datos en un orden concreto**.
- Cuando aparece un nuevo registro, se añade al final del archivo, pero los punteros se reordenan para que se mantenga el orden.



Ventajas:

- El fichero mantiene un orden según se añadieron los registros y otro orden en base a una clave.
- Para **ordenar** el fichero sólo se deben **modificar los punteros**.
- Tiene las **mismas ventajas** que la **organización secuencial**

Inconvenientes:

- No se borran registros, sino que se **marcan para ser ignorados**, por lo que se **derrocha el espacio**.
- **Añadir registros o modificar las claves** son operaciones que requieren **recalcular los punteros**.

4.2.4 Ficheros con organización secuencial-indexada

- Este tipo de organización trata de **subsanar los inconvenientes de las organizaciones secuencial y relativa**, refundir en una sola organización las ventajas de las anteriores:
  - **Secuencial**: **ahorro de espacio** en el soporte.

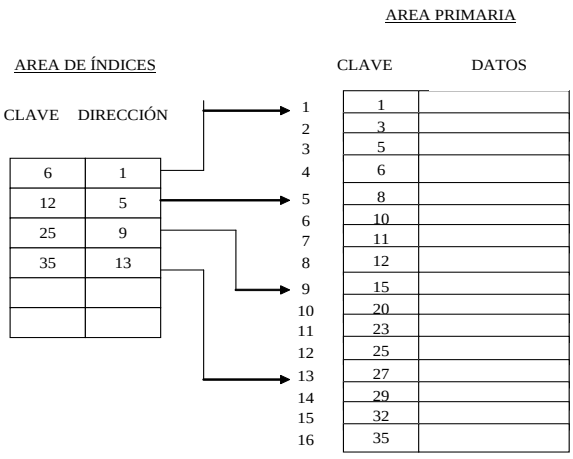
    Los registros se **graban secuencialmente ordenados por el campo clave**.
  - **Relativa**: **mínimo tiempo de acceso a un registro,**

    Se van a **crear unos índices o tablas** que, empleándolas juntamente con el archivo, van a permitir acceder directamente a un grupo de registros
- En esta organización se distinguen **tres clases de áreas** dentro del fichero:
  - **ÁREA PRIMARIA**
    - ✓ Es donde se almacenan **los registros durante la creación del archivo**.
    - ✓ Se trata de un fichero **organizado secuencialmente ordenado polo campo clave**.
  - **ÁREA DE ÍNDICES**
    - ✓ Se pode considerar como un **fichero auxiliar**, **organizado secuencialmente**, que sirve para **acceder a los registros del área primaria a través del valor de su clave**.
    - ✓ Sus registros están formados por dos campos:
      - **Valor clave**: Contiene el **valor de la clave más alta** de los registros que forman **un grupo**.
      - **Dirección:** Contiene la **dirección de comienzo de cada grupo**.
  - **ÁREA DE DESBORDAMENTO**
    - ✓ Cuando se **crea el fichero** esta área **está vacía**.
    - ✓ Posteriormente, se almacenarán en esta área **los registros procedentes del área primaria** cuando en esta se añadan nuevos registros.
    - ✓ Los registros se irán **colocando por el orden de llegada**, con el cual se **rompe la secuencia lógica del fichero** (los registros ocupan la primera posición libre, independientemente de la posición lógica que tuvieran que ocupar).

Como por ejemplo: Supongamos que tenemos las siguientes claves numéricas ordenadas de menor a mayor y que cada grupo cuenten 4 registros:

1,3,5,6,8,10,11,12,15,20,23,25,27,29,32,35

En la imagen podemos ver cómo se organizarían en el área primaria los anteriores registros y cómo sería la estructura del área de índices cuando creamos el archivo con organización secuencial indexada.



4.2.5 Ficheros indexado-encadenados

- Utiliza **punteros e índices**, es una variante encadenada del caso anterior.
- Hay un **fichero de índices** equivalente al comentado en el caso anterior y otro fichero de tipo encadenado con punteros a los siguientes registros.
- Cuando se **añaden registros se utiliza un área de desbordamiento u overflow**. En ese archivo los datos se almacenan secuencialmente, se accede la ellos se busca un dato y no se encuentra en la tabla de índices.

Ventajas:

- Posee las **mismas ventajas que los archivos secuenciales indexados**, además de una mayor rapidez al reorganizar el fichero (sólo se modifican los punteros)

Inconvenientes:

- Requieren compactar los datos a menudo para **reorganizar índices** y quitar el fichero de desbordamiento.

5. Operaciones relacionadas con el uso de fichero en la base de datos

Borrado y recuperación de registros

- ✓ Algunos de los tipos de fichero vistos anteriormente **no admiten el borrado real de datos**, sino que permiten **añadir un dato que indica se lo registro está borrado o no**.
- ✓ Esto es interesante ya que **permite anular una operación de borrado**. Por ello, esta técnica de marcar registros se utiliza casi siempre en todos los tipos de archivos.

Fragmentación y compactación de datos

La fragmentación en un archivo hace referencia a la posibilidad de que **ésta tenga huecos interiores debido al borrado de datos o a otras causas.**

Problemas:

- **Mayor espacio de almacenamiento**.
- **Lentitud en las operaciones de lectura y escritura** del fichero

Por lo eres es **importante compactar los datos**. Esta técnica permite eliminar los huecos interiores a un archivo. Las formas de realizarla son:

- ***Reescribir el archivo para eliminar los huecos***. Es la mejor opción, pero lógicamente es la más lenta al requerir releer y reorganizar todo el contenido del fichero.
- ***Aprovechar huecos***. De manera que los nuevos registros se inserten en esos huecos. Esta técnica suele requerir un paso previo para reorganizar esos huecos.

Compresión de datos

En muchos casos, para **ahorrar espacio de almacenamiento**, se emplean técnicas de **compresión de datos.**

- ✓ **Ventaja**: los **datos ocupan menos espacio**.
- ✓ **Desventaja**: al manipular los datos se tienen que descomprimir, lo que hace que la **manipulación de los datos sea lenta**.

Cifrado de datos

Otra de las opciones habituales sobre fichero de datos es **utilizar técnicas de cifrado para proteger** los ficheros en caso de que alguien **no autorizado** consiga el fichero.

Para leer un fichero de datos, haría falla descifrar el fichero. Para descifrarlo, necesitaremos una clave o bien aplicar métodos de descifrado; lógicamente cuanto mejor sea la técnica de cifrado, más difícil será descifrar los datos mediante la fuerza bruta.

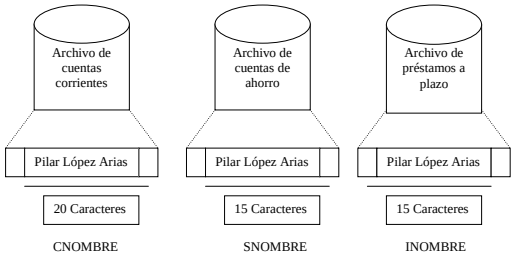
6. Inconvenientes de un sistema de gestión de archivos

Los principales **problemas en la gestión de datos con un sistema de fichero** son:

Redundancia e inconsistencia de los datos

Se produce porque **cada aplicación emplea sus propios archivos de datos**, aunque muchos de estos datos sean comunes a varios programas.

Como por ejemplo: en un banco, los datos personales de un cliente (nombre, apellidos, dirección, teléfono) aparecerían en el archivo de cuentas corrientes, en uno de cuentas de ahorro y en un archivo préstamos. En los distintos archivos, por ejemplo, el campo dirección pode tener distintos formatos y además están duplicados en distintos sitios.



- La **redundancia:**
  - ✘ **Aumenta los costos de almacenamiento y acceso**
  - ✘ **Trae consigo: la inconsistencia de los datos** (cada vez que se produjera un cambio de dirección deberíamos cambiarlo en todos los archivos en los que este dato figura, ya que de otro modo no podremos saber cuál es la dirección correcta).

Los **sistemas de bases de datos** pueden **eliminar la redundancia de los datos**, ya que todas las aplicaciones **comparten un almacén común de datos**. La información se almacena una única vez en la base de datos.

- **Dependencia de los datos física-lógica**
  - La **estructura física de los datos** (definición de archivos y registros)
    - **Está codificada nos programas de aplicación**:

      Cualquier cambio en esa estructura implica al programador i**dentificar, modificar y probar todos los programas que manipulan esos archivos**.

- **Dificultad para tener acceso a los datos**
  - **Cada vez que se necesite una consulta que no fue prevista en el inicio:**

    Es preciso codificar el programa de aplicación necesario.

    Los entornos convencionales de procesamiento de archivos no permiten recuperar los datos de manera eficiente.

- **Separación y aislamiento de los datos**
  - **Al estar repartidos en varios archivos los datos, y tener diferentes formatos:**

    Es difícil escribir nuevos programas que relacionen los datos de distintos archivos. Antes deberían sincronizarse todos los archivos para que los datos coincidieran.

  Como por ejemplo, imaginemos que tenemos un archivo de CLIENTES y otro de PRODUCTOS y queremos relacionar los clientes con los productos que les interesa comprar. Con el procesamiento de archivos, esto es difícil porque no podemos establecer conexiones fuertes entre los datos contenidos en archivos diferentes. Los analistas de sistemas y programadores deben determinar que partes de cada archivo son necesarias, deben decidir cómo se relacionan los archivos entre sí y coordinar el procesamiento de los archivos de suerte que se obtengan los datos correctos. Se coordinar dos archivos es difícil, imagine la tarea de coordinar diez o más.

- **Dificultad para el acceso concurrente**
  - **En un sistema de gestión de archivos es complicado que los usuarios actualicen los datos simultáneamente.**

    Las actualizaciones concurrentes pueden dar como resultado datos inconsistentes, ya que se pode acceder a los datos desde distintos programas de aplicación, por ejemplo, un usuario puede estar consultando un dato mientras otro lo modifica,

- **Dependencia de la estructura del archivo con el lenguaje de programación**
  - **Los formatos físicos de los archivos y registros son parte del código de la aplicación (Los programas de aplicación dependen de los formatos de los archivos).**

    Problema → Cualquier cambio en los formatos de los archivos también hay que modificar los programas que los emplean.

- **Problemas en la seguridad de los datos**
  - **Resulta difícil implantar restricciones de seguridad pues las aplicaciones se van añadiendo al sistema según se van necesitando.**

- **Problemas de integridad de datos**
  - **Los valores almacenados en los archivos deben cumplir ciertas restricciones de consistencia.**

    Esto implica añadir gran número de líneas de código nos programas. El problema se complica cuando existen restricciones que implican varios datos en distintos archivos.

  Ejemplo de restricciones: no se puede insertar una nota de un alumno se previamente esa materia no está creada. Otro ejemplo, las unidades en almacén de un producto determinado no deben ser inferiores la una cantidad.

7. Sistemas de bases de datos

Los sistemas de base de datos superan las limitaciones de los sistemas orientados a los archivos.

- **Al tolerar una estructura de datos centralizada** (base de datos), integrada:

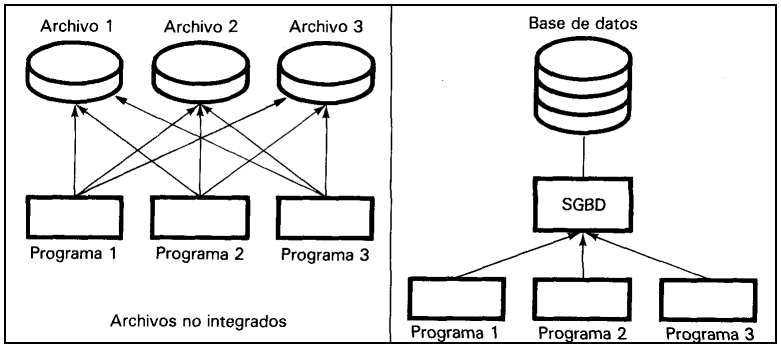
  Los sistemas de base de datos **eliminan los problemas de redundancia y control de los datos**.

Si una **base de datos está centralizada** está **disponible para toda la empresa**.
Por ejemplo, si se cambió el nombre de un cliente, dicho cambio está disponible para toda la empresa.

Definición de base de datos:

Una base de datos es un **conjunto de datos organizados y relacionados lógicamente** y que han sido diseñadas para satisfacer los requerimientos de información de una empresa o organización.

- Al **proporcionar un software para almacenamiento, acceso y modificación de los datos** (Sistema Gestor Base Datos o SGDB): _
  - ✓ El sistema **esconde ciertos detalles de cómo se almacenan y mantienen los datos**.
  - ✓ **Simplifican el esfuerzo de programación y mantenimiento de los programas**.



Si comparamos el sistema de procesamiento de archivos con el sistema de base de datos_

- ✓ Los **programas de procesamiento de archivos** tienen que **acceder directamente a los datos contenidos en los archivos**, en contraste los **programas de aplicación de base de datos acuden al SGBD (Sistema Gestor de Base de Datos) para acceder a los datos almacenados.**
- ✓ Esta diferencia es significativa: **hace más fácil la tarea de programar la aplicación**.

  Los **programadores no deben preocuparse de la forma en que se almacenan los datos**, más bien quedan libres para concentrarse en cuestiones importantes para el usuario, y no en aspecto del sistema de computación.

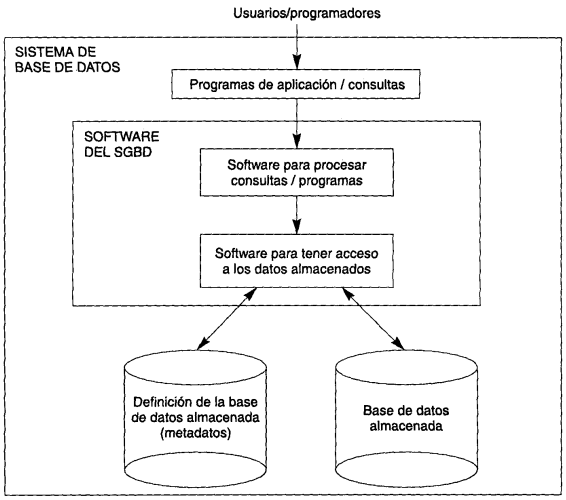
7.1 Sistema Gestor de base de datos

Definimos ***Sistema Gestor de Bases de Datos*** o **SGBD** (DataBase Management System – SGBD), es el **software** que permite a los usuarios **procesar, describir, administrar y recuperar los datos almacenados en una Base de Datos**.

El objetivo principal

- ✓ **Proporcionar una manera eficiente y segura de almacenamiento o extracción de la información** en las Bases de Datos.
- ✓ Están diseñados para **gestionar grandes bloques de información**, que implica tanto la definición de estructuras para lo almacenamiento como de mecanismos para la gestión de la información.

- Una **base de datos** es un **gran almacén de datos que se define una sola vez.**
- Los **datos** pueden **ser accedidos de manera simultánea por varios usuarios**; están relacionados y existe un mínimo de duplicidad.
- Además, en los sistemas de bases de datos **se almacenan las descripciones de los datos**, lo que se conoce como **metadatos**, en el **diccionario de datos**.



Los SGBD:

- Realizan **las transformaciones necesarias para almacenar y recuperar la información almacenada**.

El SGBD un sistema de **software de propósito general** que facilita el proceso de **definir, construir y manipular bases de datos** para diversas aplicaciones.

- ✓ Para **definir** una base de datos hay que **especificar los tipos de datos, las estructuras y las restricciones** de los datos que se almacenarán en ella.
- ✓ **Construir** una base de datos es el proceso de **guardar los datos** en algún medio de almacenamiento controlado por el SGBD.
- ✓ En la **manipulación** de una base de datos intervienen funciones como consultar la base de datos para obtener datos específicos, actualizar la base de datos para reflejar cambios en el minimundo y generar informes a partir de los datos

- Todas las **interacciones entre la base de datos y los usuarios corren a cargo del SGBD** (comprobaciones de seguridad incluidas).

7.2 Funciones

Los SGBD deben prestar las **siguientes funciones:**

Creación y definición de la Base de Datos ✓ Se especifica **estructura, el tipo de datos, las restricciones y las relaciones** entre estos mediante **lenguaje de definición de datos (LDD)**. ✓ Toda esta información **se almacena en el diccionario de datos**. ✓ El SGBD proporciona **mecanismos para la gestión del diccionario de datos.**	**Manipulación de los datos** Realizando **consultas, inserciones y actualizaciones de los mismos empleando lenguajes de manipulación de datos (LMD).**
Acceso controlado a los datos de la base de datos Empleando **mecanismos de seguridad de acceso para los usuarios**.	**Mantener la integridad y consistencia de los datos** Empleando mecanismos para **evitar que los datos se degraden por cambios no autorizados**.
Acceso compartido a la base de datos **Controlando la interacción entre usuarios concurrentes**.	**Herramientas para la creación de copias de seguridad**
Mecanismos de respaldo y recuperación Para **restablecer la información en caso de fallos en el sistema**.	

7.3 Ventajas de los SGBD

Independencia datos/programas:

- ✓ La **gestión de los datos se hace a alto nivel**, es decir, por comandos o menús fácilmente inteligibles por el usuario, **quedando ocultos todos los detalles físicos**.
- ✓ Las **aplicaciones** de bases de datos **no necesitan conocer el formato de los registros y archivos** que procesa. Sólo saber la longitud y el tipo de dato que necesitan procesar de la base de datos.
- ✓ **El SGBD es el que localiza los datos en los registros**.

Flexibilidad.

- ✓ En ocasiones es necesario modificar la estructura de una base de datos cuando cambian los requerimientos.
- ✓ Los **SGBS permiten efectuar cambios en la estructura de la base de datos sin afectar a los datos almacenados y los programas de aplicación existente**.

Evitan la redundancia.

- ✓ **Todos los datos** de una aplicación **se almacenan** en un medio muy sencillo llamado **base de datos, por lo que evitan que estén duplicados en varios sitios.**

Evitan la inconsistencia.

- ✓ Cuando **existen varias copias** de un dato, **puede ocurrir que haya inconsistencia**, es decir, que las copias no concuerden (al modificar un dato en un sitio y no en las copias).
- ✓ **Las bases de datos evitan este problema**.

Evita programación para las tareas más comunes.

- ✓ El software de bases de datos **incorpora las funciones básicas sobre datos** que, de otra manera, deberían codificarse mediante un lenguaje de programación.

Menor tiempo de creación de aplicaciones.

- ✓ Una de las características es que la creación de una aplicación nueva - como la obtención de cierta información de la base de datos para imprimir en un informe - requiere poco tiempo una vez que la base de datos está creada y funcionando.
- ✓ Se **estima que el tiempo de creación de una aplicación con un SGBD es de sexta o cuarta parte de lo requerido en un sistema de procesamiento de archivos tradicional**.

Es posible aplicar restricciones de seguridad para el acceso a los datos.

- ✓ El SGBD contiene un **subsistema de seguridad y autorización que permita imponer controles para acceder o modificar las bases de datos**. Cuando muchos usuarios comparten una misma base de datos, es probable que no todos tengan la autorización para tener acceso a toda la información que contiene.

8. Arquitectura de los sistemas de bases de datos

Como ya se comentó, **la independencia de los datos y los programas** es una de las grandes ventajas de los SGBD.

Definición:

La **independencia de los datos/programas** es la propiedad que asegura que **los programas de aplicación sean independientes de los cambios que se realizan en la estructura de los datos y de los detalles de su representación física.**

Objetivo:

- ✓ Que **las aplicaciones no dependan de cómo se almacenen los datos físicamente o de su organización lógica.**
- ✓ **Puede modificarse la estructura de los datos sin que afecte a los programas**. Ej.: Modificar nombres y formato de los ficheros, añadir o borrar campos, añadir o borrar índice (mecanismos para mejorar los accesos a los datos), dividir la base de datos en distintos discos, etc.
- ✓ Esto implica que las aplicaciones (o los usuarios) tienen una **Visión Abstracta de los datos**:
- ✓ El SGBD **esconde detalles sobre cómo se almacenan y mantienen los datos**, de forma que **facilita el acceso a estos datos**, ya que sólo es necesario pedir al SGBD qué datos se requieren, sin especificar cómo conseguirlos.

Este es el **principal objetivo de la Arquitectura ANSI/SPARC**(American National Standard Institute - Standards Planning and Requirements Committee) d

- La idea básica es poner un **nivel intermedio entre las aplicaciones (o los usuarios) y el sistema de almacenamiento físico**.

El principal objetivo de esta arquitectura era lo de **separar los programas de aplicación de la base de datos física.**

En 1975, el comité **ANSI/SPARC** (American National Standard Institute / Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los sistemas gestores de bases de datos.

En esta arquitectura, cada **sistema de base de datos se organiza de acuerdo la tres niveles** de abstracción distintos:

NIVELES DE LA ARQUITECTURA ANSI/SPARC

El nivel interno o físico

- ✓ Es el **más cercano al almacenamiento físico**, es decir, a **cómo están almacenada la base de datos** en el ordenador.
- ✓ **Describe los detalles de cómo se almacenan físicamente los datos**, así como los caminos de acceso para la base de datos.

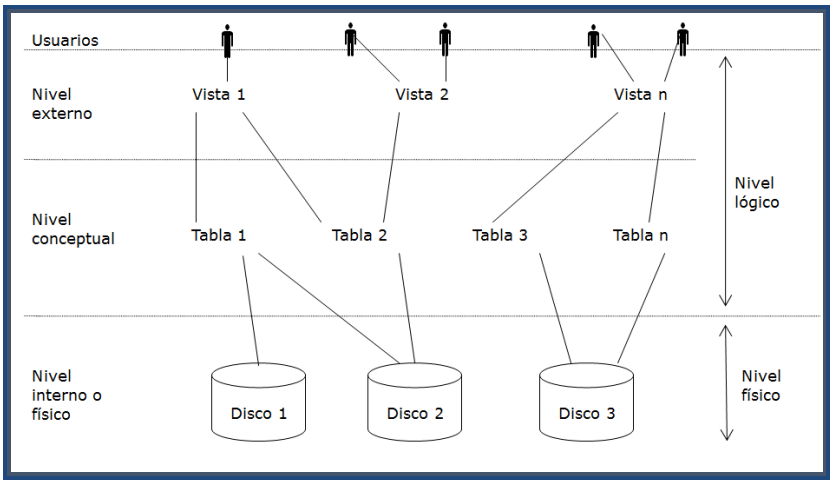
El nivel conceptual

- ✓ Describe la **estructura de toda la base de datos** para una comunidad de usuarios.
- ✓ El esquema conceptual oculta los detalles de las estructuras físicas de almacenamiento y se **concentra en describir entidades, tipos de datos, relaciones, operaciones de los usuarios y restricciones**.
- ✓ Describe **qué datos se almacenan en la base de datos y qué relaciones existen entre los datos**.

El nivel externo

- ✓ El **más cercano a los usuarios**, es decir, es donde se describen varios esquemas externos o vistas de usuarios.
- ✓ Cada esquema externo describe **la parte de la base de datos que interesa el un grupo de usuarios determinado, y oculta a ese grupo el resto de la base de datos**.

Esta arquitectura describe los datos a tres niveles de abstracción. Los **únicos datos que existen realmente están en el nivel físico**, almacenados en discos u otros dispositivos.



Los SGBD basados en esta arquitectura permiten que cada grupo de usuarios haga referencia a su propio esquema externo. El SGBD debe transformar cualquier petición de usuario (esquema externo) a una petición expresada en términos del esquema conceptual, para finalmente ser una petición expresada en el esquema interno que se procesará sobre la base de datos almacenada.

Al proceso y transformar solicitudes y resultados de un nivel la otro se le denomina correspondencia o transformación (mapping o mapeo).

Ejemplo de la arquitectura ANSI/SPARC en una BD relacional

➢ **Nivel externo**: **Visión parcial de las tablas de la BD según el usuario**. Por ejemplo, la vista que se muestra en la siguiente imagen, obtiene el listado de notas de alumnos con los siguientes datos: Curso, Nombre, Nombre de asignatura y Nota.

Curso	Nombre	Nombre de asignatura	Nota
1	Ana	Programación en lenguajes estructurados	6
1	Ana	Sistemas informáticos multiusuario y en red	8
2	Rosa	Desa. de aplic. en entornos de 4.ª Generación y H. Case	5
2	Juan	Desa. de aplic. en entornos de 4.ª Generación y H. Case	7
1	Alicia	Programación en lenguajes estructurados	5
1	Alicia	Sistemas informáticos multiusuario y en red	4

➢ **Nivel conceptual: Definición de todas las tablas, columnas, restricciones, claves y relaciones.** En este ejemplo, disponemos de tres tablas que están relacionadas:

- Tabla **ALUMNOS**. Columnas: NMatrícula, Nombre, Curso, Dirección, Población. Clave: NMatrícula. Además tiene una relación con NOTAS, pues un alumno puede tener notas en varias asignaturas.
- Tabla **ASIGNATURAS**. Columnas: Código, Nombre de asignatura. Clave: Código. Está relacionada con NOTAS, pues para una asignatura hay varias notas, tantas como alumnos la cursen.0
- Tabla **NOTAS**. Columnas: NMatrícula, Código, Nota. Está relacionada con ALUMNOS y ASIGNATURAS, pues un alumno tiene notas en varias asignaturas, y de una asignatura existen varias notas, tantas como alumnos.

➢ **Nivel interno**: En una BD, las tablas se almacenan en **archivos de datos de la BD**. **Si hay claves, se crean índices para acceder a los datos**, todo esto contenido en el disco duro, en una pista y en un sector, que sólo el SGBD conoce. Ante una petición, sabe a qué pista, a qué sector, a qué archivo de datos y a qué **índices acceder**.

➢ Con la **arquitectura a tres niveles** se introduce el **concepto de independencia de datos**. Hay dos tipos de independencia con respeto a los datos:

La independencia lógica con respecto a los datos

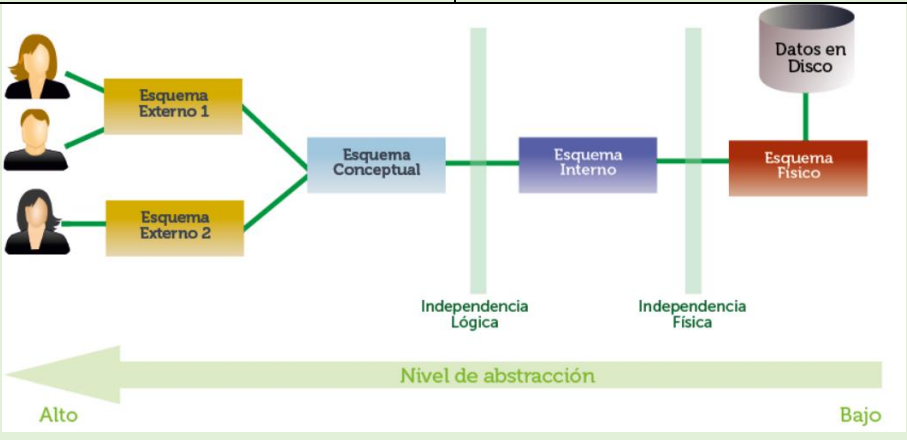
Capacidad de **modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación**.

Podemos modificar el esquema conceptual para ampliar la base de datos (añadiendo un nuevo tipo de registro o elemento de información), o para reducirla (como por ejemplo, eliminando una entidad, los esquemas externos que no se refieran la ella no se verán afectados).

La independencia física con respecto a los datos

Capacidad de **modificar el esquema interno sin tener que alterar el esquema conceptual, ni los externos.**

Como por ejemplo, se pueden reorganizar los archivos físicos con el fin de mejorar el rendimiento de las operaciones de consulta o actualización.



En los sistemas gestores de bases de datos basados en arquitecturas de varios niveles se hace necesario ampliar el diccionario de datos para que incluya información sobre cómo establecer las correspondencias entre las solicitudes de los usuarios y los datos, entre los distintos niveles.

Por tanto, la arquitectura de tres niveles puede facilitar el logro de la verdadera independencia con respecto a los datos, tanto física como lógica

9. Componentes de los sgbd

Los sistemas gestores de bases de datos están divididos en módulos que proporcionan una serie de servicios que permiten almacenar y explotar los datos de manera eficiente. Los **componentes principales** son: **el núcleo, los lenguajes, las utilidades y el diccionario de datos.**

9.1 Núcleo

Es el **conjunto de programas que coordinan y controlan el funcionamiento del SGBD**. Son programas transparentes al usuario que garantizan la seguridad e integridad de los datos:

- **Controlan la integridad y seguridad**, protegiendo los datos contra accesos no autorizados.
- Los SGBD ofrecen mecanismos para **implantar restricciones de integridad en la BD**. Los datos que se almacenan deben satisfacer ciertos tipos de restricciones de consistencia y reglas de integridad.
- **Gestionan el diccionario de datos** o catálogo para que pueda estar accesible al usuario los metadatos que describen los datos de la base de datos.
- Implementan funciones de comunicación entre niveles.
- Proporciona herramientas y mecanismos para la **planificación y realización de copias de seguridad y restauración**.
- Debe **asegurar el acceso concurrente** y ofrecer mecanismos para mantener la consistencia de los datos en caso de que varios usuarios actualicen la base de datos al mismo tiempo.
- La mayoría tienen un **optimizador de consultas**: para determinar la estrategia óptima para la ejecución de las consultas
- También suelen incorporar un **planificador(scheduler) para programar y automatizar la realización de ciertas operaciones y procesos.**
- Permiten la **importación y exportación de datos** (migraciones)

9.2 Funciones y lenguajes de los SGBD

Todos los SGBD ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: administradores, diseñadores, programadores y usuarios finales. Estas responden a los tres tipos de funciones que tienen que desempeñar:

- **Función de descrición**
  - ✓ Permite al diseñador de la base de datos **crear las estructuras apropiadas para integrar adecuadamente los datos**. Se dice que esta función es la que permite definir las tres estructuras de la base de datos (relacionadas con los tres niveles de abstracción).
    - Estructura interna
    - Estructura conceptual
    - Estructura externa
  - ✓ Esta función trabaja con los **metadatos**.

➢**Los metadatos son la información de la base de datos que sirven para describir los datos.**

Laura, Alicia y Manuel son datos; pero nombre es un metadato. También son metadatos decir que la base de datos contiene Empleados o que el DNI está formado por 9 caracteres de los cuales los ocho primeros son números y el noveno una letra en mayúsculas.

- ✓ Sirve para **crear, eliminar o modificar metadatos**, y para esto se usa **un lenguaje de descripción de datos o DDL**. Este lenguaje permite:
  - **Definir las estructuras de datos**
  - **Definir las relaciones entre los datos**
  - **Definir las reglas que tienen que cumplir los datos.**

- **Función de manipulación**
  - ✓ Permite **modificar y emplear los datos de la base de datos**, lo que se realiza mediante un **lenguaje de modificación de datos o DML**. Este lenguaje permite:
    - **Añadir datos**
    - **Eliminar datos**
    - **Modificar datos**
    - **Buscar datos**
  - ✓ Actualmente se suele distinguir aparte la función de buscar datos en la base de datos (**función de consulta**), para lo cual se proporciona un **lenguaje de consulta de datos o DQL**.
- **Función de control**
  - ✓ Mediante esta función los administradores **poseen mecanismos para proteger los datos**; de modo que se permite a cada usuario ver ciertos datos y otros no; o bien usar ciertos recursos concretos de la base de datos y prohibir otros. Es decir, simplemente permite controlar la seguridad de la base de datos.
  - ✓ El lenguaje que implementa esta función es el **lenguaje de control de datos o DCL**.

9.3 Utilidades

Son aplicaciones que facilitan el trabajo a los usuarios y programadores. Tienen la característica común de tener un interfaz donado de entender. Se basan en menús que guían al usuario para conseguir el objetivo final.

Entre estos se encuentran los **asistentes, los generadores de menús, los generadores de informes y los generadores de formularios**.

9.4 Diccionario de datos

El diccionario de datos, también conocido cómo **catálogo del sistema**, es el lugar donde se almacena información acerca de todos los datos que forman la base de datos. Describe la base de datos y los objetos que la forman.

En una **base de datos relacional**, el diccionario de datos proporciona información acerca de:

- La **estructura física y lógica** de la base de datos
- La **definición de todos los objetos de la base de datos**: tablas, vistas, índices, disparadores, procedimientos, ...
- El **espacio asignado y empleado por estos objetos**.
- Información acerca de las **restricciones de integridad**.
- Los **privilegios y papeles otorgados a los usuarios**.
- **Auditoría** de información, como los accesos a los objetos.

10. Modelos de datos

Una característica del enfoque de base de datos es que proporciona **cierto nivel de abstracción de los datos al ocultar detalles del almacenamiento**, es decir, el usuario va a emplear esos datos pero no sabe cómo están almacenados físicamente.

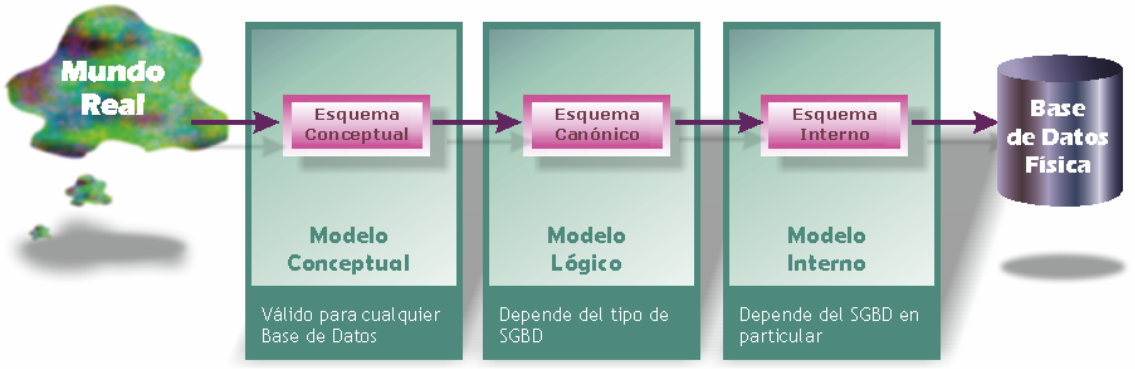
Los modelos de datos son el principal instrumento para ofrecer dicha abstracción.

Un modelo de datos es un **conjunto de conceptos** que pueden **servir para describir**, a **diferentes niveles de abstracción**, la **estructura de una base de datos**.

Con el concepto de **estructura de datos** nos referimos a los **tipos de datos, las relaciones y las restricciones** que deben cumplirse para esos datos.

Categorías de los modelos de datos

Se han propuesto muchos modelos de datos, y podemos clasificarlos **dependiendo de los tipos de conceptos** que ofrecen para describir la estructura de la base de datos.



En la práctica, al definir la base de datos desde el mundo real hasta llegar a los datos físicos se pasa por los siguientes esquemas:

- **Los modelos de datos conceptuales o de alto nivel :**
  - ✓ Disponen de **conceptos muy cercanos al modo como los usuarios perciben los datos**.
  - ✓ Son unas **herramientas de análisis y son independientes del software de SGBD** que se use para manipular la base de datos.

  En realidad, el diseño conceptual debe hacerse aun cuando la implementación final no use un SGBD, sino archivos convencionales y lenguajes de programación.

  Los modelos de datos conceptuales más empleados son: el modelo de **entidad-relación (ER) y el UML**

- **Modelos de datos de representación o lógicos (o de implementación).**
  - ✓ Describen **la estructura de la base de datos que puede procesar el software del SGBD**.
  - ✓ **Depende del tipo de SGBD y no de un SGBD específico**.

▪ **Relacionales**	▪ **Jerárquicos**	▪ **Red**
▪ **Objeto-relacionales**	▪ **Orientados a objetos**	▪ **Base de datos XML**

- **Los modelos de datos de bajo nivel o físicos:**
  - ✓ Proporcionan conceptos que **describen los detalles de cómo se almacenan los datos** en el computador, es decir, la implantación de la base de datos en un dispositivo de almacenamiento secundario.
  - ✓ **Describe las estructuras de almacenamiento y los métodos** usados para tener un acceso efectivo de los datos.

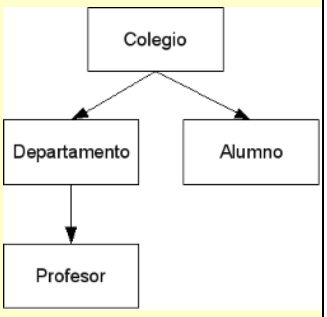
  Estos modelos no están normalizados ni existen en realidad como modelos, sino que son propios de cada producto comercial.

Modelos de datos de representación o lógicos (o de implementación).

A continuación vamos a ver los **modelos lógicos más importantes**:

Modelo jerárquico

- ✓ También conocido como **modelo en árbol** debido la que usa una estructura en árbol para organizar los datos.
- ✓ La información **se organiza con una jerarquía en la que la relación entre las entidades de este modelo es siempre del tipo padre/hijo**. De este modo hay una serie de nodos que contendrán atributos y que se relacionarán con nodos hijos de manera que puede haber más de un hijo para lo mismo padre
- ✓ Este modelo **está en desuso ya que no es válido para modelar la mayoría de los problemas de bases de datos**.



Modelo en red (codasyl)

- ✓ Es un **modelo que tuvo una grande aceptación aunque apenas se utiliza actualmente**.
- ✓ En este modelo, la información **se organiza en registros (también llamados nodos) y enlaces**. En los registros se almacenan los datos, mientras que los enlaces permiten relacionar estos datos (**Grafo**)
- ✓ Las bases de datos en red son parecidas las jerárquicas, sólo que en ellas pode **haber más de un padre**.
- ✓ En este modelo **se pode representar perfectamente cualquier tipo de relación entre los datos**, pero hace muy complicado su manejo.

Un ejemplo de base de datos en red

González	19283746	Arenal	La Granja
Gómez	19283746	Carretas	Cerceda
López	67789901	Mayor	Peguerinos
Abril	96396396	Preciados	Valsaín
Santos	32112312	Mayor	Peguerinos
Rupérez	24466880	Ramblas	León

C-101	100.000
C-215	140.000
C-102	80.000
C-305	70.000
C-201	180.000
C-217	150.000
C-222	140.000

Modelo relacional

- ✓ En este modelo **los datos se organizan en tablas que se relacionan entre sí**.
- ✓ Es el **modelo más popular y el que ocupa todavía el primer lugar en el mercado**, ya que son las más indicas para las tradicionales aplicaciones de gestión, como por ejemplo, la gestión de una biblioteca.
- ✓ **Este modelo es el que vamos a ver en este curso**.

Ejemplos: Oracle, MySQL, SQL Server, Maria DB, SQLite, PostgreSQL

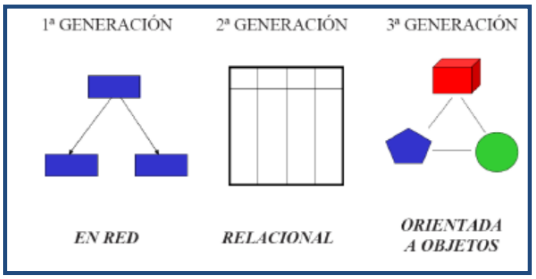
Un ejemplo de base de datos relacional

nombre-cliente	*dni*	*calle-cliente*	*ciudad-cliente*	*número-cuenta*
González	19283746	Arenal	La Granja	C-101
Gómez	19283746	Carretas	Cerceda	C-215
López	67789901	Mayor	Peguerinos	C-102
Abril	96396396	Preciados	Valsaín	C-305
González	19283746	Arenal	La Granja	C-201
Santos	32112312	Mayor	Peguerinos	C-217
Rupérez	24466880	Ramblas	León	C-222
Gómez	19283746	Carretas	Cerceda	C-201

número-cuenta	*saldo*
C-101	100.000
C-215	140.000
C-102	80.000
C-305	70.000
C-201	180.000
C-217	150.000
C-222	140.000

➤ Debido en parte a las mejoras que ha sufrido el hardware, **han surgido aplicaciones cada vez más sofisticadas, que las BD relacionales no son siempre las más apropiadas para el almacenamiento**-

Ejemplo: aplicaciones para el diseño y fabricación de ingeniería (CAD/CAM (Computer_Aided Design/computer/ Computer_Aided Manufacturing), CIM (Computer Integrated Manufacturing)), sistemas de información geográfica (GIS, Geographic Information System), sistemas multimedia, CASE (Computer_Aided Software Enginneering),



➤ Los requerimientos y las características de estas nuevas aplicaciones difieren en gran medida de las típicas aplicaciones de gestión:

- ✘ La **estructura de los objetos es más compleja** así como sus interrelaciones,
- ✘ Se necesitan **nuevos tipos de datos para almacenar imágenes y textos**, y hace falta definir operaciones no estándar, específicas para cada aplicación.
- ✘ Para hacer frente a estos requerimientos, surge una nueva generación de base de datos, las **bases de datos orientadas a objetos** puras (BDO) y las **base de datos objeto-relacionales** (BDOR).

Modelo de bases de datos orientadas a objetos

- ✓ Desde la **aparición de la programación orientada a objetos**, se intentó **adaptar las bases de datos a estos lenguajes**.
- ✓ La programación orientada a objetos permite **cohesionar datos y procedimientos**, haciendo que se diseñen **estructuras que posean datos (atributos) y en las que se definen los procedimientos/operaciones (métodos)** que se pueden realizar con los datos. Las bases de datos orientadas a objetos se basan en la misma idea.
- ✓ **La representación de los datos es en forma de objetos** (atributos y métodos).
- ✓ Estas bases de datos mezclan las potencialidades de BDOO con diferentes lenguajes de programación. Estas se coordinan muy bien con Delphi, Phyton, Ruby, JavaScript, Java, Visual BASIC, NET, C++, entre otros.
- ✓ **Su modelo conceptual es el UML**.

Algunos sistemas disponibles en el mercado están: DB4O, ZODB, GEMSTONE/OPAL de ServicLogic, ONTOS de Ontologic, Objectivity de Objectivity Inc., Versant de Versant Technologies, ObjecStore de Object Design y O2 de O2 Technology

Modelo de bases de datos objeto-relacionales

- ✓ Tratan de ser un **híbrido entre el modelo relacional y el orientado a objetos**.

  El problema de las bases de datos orientadas a objetos es que requieren reinvertir capital y esfuerzos de nuevo para convertir las bases de datos relacionales en bases de datos orientadas a objetos.
- ✓ En las bases de datos **objeto-relacionales se intentan conseguir una compatibilidad relacional** dando la posibilidad de **integrar avances de la orientación a objetos.**
- ✓ Estas bases de datos están basadas en el **estándar SQL 99**.

Las últimas versiones de la mayoría de las clásicas grandes bases de datos relacionales (Oracle, SQL Server, Informix, PostGreSQL...) son objetos relacionales.

Base de datos XML

- ✓ XML se ha convertido en el lenguaje más ampliamente utilizado para la representación de datos semi-estructurado, siendo actualmente el lenguaje estándar para **el intercambio de información**, integración de datos y de aplicaciones.

  Es decir, es especialmente útil **como lenguaje de formato de datos cuando las aplicaciones necesitan comunicarse**, **facilitando la integración de información** procedente de varias aplicaciones.
- ✓ Cada vez disponemos de más **documentos XML que es necesario almacenar y gestionar de un modo operativo**.
- ✓ En la actualidad, existen diferentes soluciones para almacenamiento y gestión de los documentos XML, que se pueden clasificar en dos grupos:
  - Las **BD XML nativas** como Tamino, eXcelon, eXist,
  - **Extensiones XML en SGBD convencionales**, como en Oracle, Microsoft SQL Server (a partir de la versión 2000), IBM DB2 XML Extender, POSTGRESQL

11. SGBD según su arquitectura

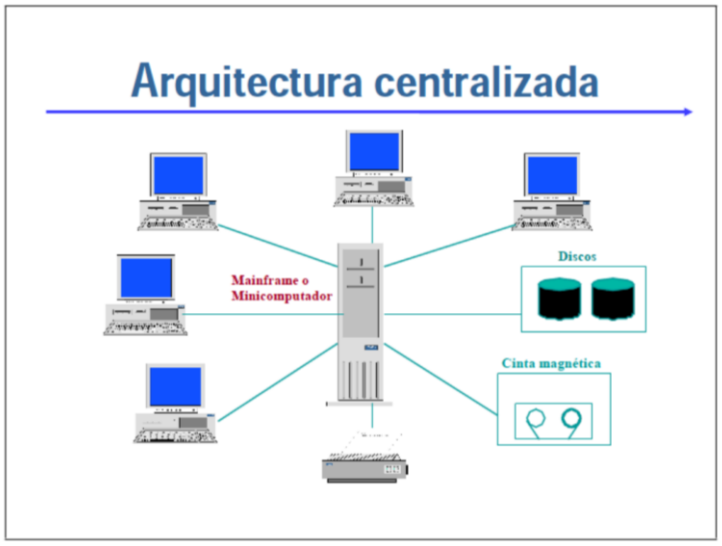
A continuación se presentan diferentes arquitecturas de base de datos:

- **Centralizadas**
- **Cliente-Servidor**
- **Distribuidas**

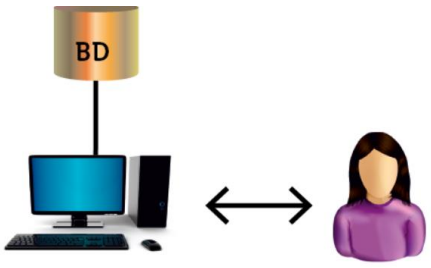
La arquitectura de un sistema de base de datos está influenciada en gran medida por el sistema informático subyacente en el que se ejecuta el sistema de base de datos. En la arquitectura de un sistema de base de datos se reflejan aspectos como la conexión en red, el paralelismo y la distribución:

11.1 Arquitecturas centralizadas

- ✓ El **SGBD está implantado en una sola plataforma u ordenador** desde **donde se gestiona directamente, de modo centralizado, la totalidad de los recursos**.
- ✓ Es la arquitectura de los centros de proceso de datos tradicionales.
- ✓ Las primeras arquitecturas eran llamadas **Arquitecturas centralizadas** y contaban:
  - **Con un macrocomputador o mainframe que proporcionaban el procesamiento principal** a todas las funciones del sistema:
    programas de aplicación, interfaces de usuario, el software SGBD, base de datos, los dispositivos de almacenamiento secundario como discos y cintas (estas para las copias de seguridad).
  - **Los usuarios accedían al sistema mediante terminales, que solamente mostraban en pantalla información**. Se llamaban terminales tontas, dado que su única función era solo la de visualización.
  - **Todo el procesamiento se hacía de manera remota en el mainframe**, que cuando concluía de procesar y se proponía desplegarle algo al usuario, debía comunicarse con ese terminal enviándole la información y los controles de pantalla. Dicha comunicación entre el computador central y los terminales "tontos" se hacía mediante algún tipo de red de computadores



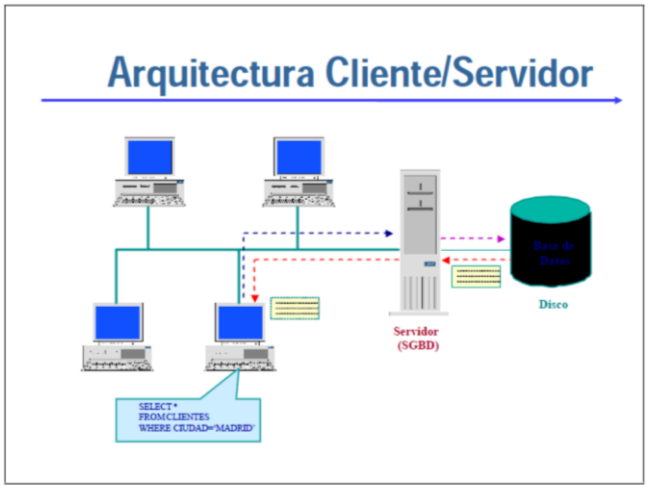
- ✓ Tales **sistemas comprenden el rango desde los SBD monousuario** ejecutándose en ordenadores personales hasta los **sistemas de base de datos de alto rendimiento** ejecutándose en grandes sistemas.



Los precios de Hardware desciendo notoriamente, surgen las PC's y la mayoría de los usuarios cambia los terminales que tenían por los nuevos PC's dado que eran relativamente baratos, y generalmente se rompían poco. En un primer momento, la arquitectura de los sistemas no varió, se seguía usando un SGBD centralizado con PC's remotas que se conectaban a una determinada máquina que era la encargada de realizar todas las funciones del SGBD.

Poco a poco los SGBD fueron evolucionando y la arquitectura centralizada comenzaba a ser una limitante en cuanto al procesamiento que el usuario exigía, lo cual llevó a las arquitecturas SGBD Cliente-Servidor.

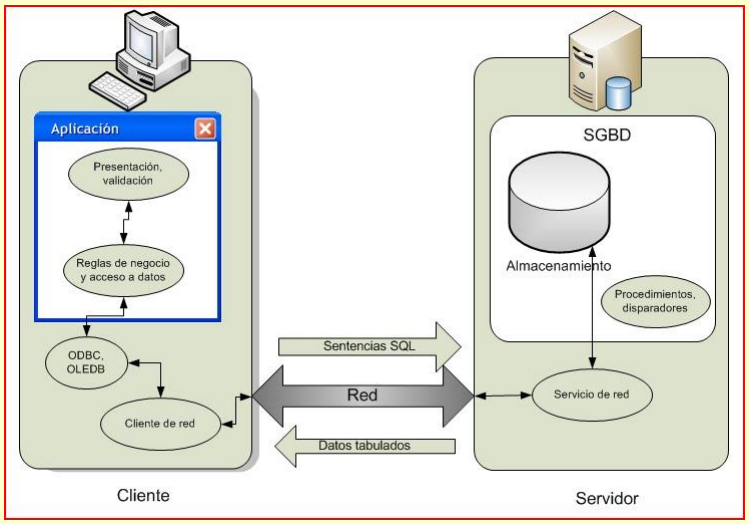
11.2 Arquitectura cliente-servidor



El **bajo costo del hardware**, los **nuevos PC's, y sobre todo el desarrollo importante que tuvieron las redes de computadoras** permiten que algunas tareas se ejecuten en un sistema servidor y que otras se ejecuten en los sistemas clientes. Esta división de trabajo ha conducido al desarrollo de sistemas de **bases de datos cliente-servidor**.

La idea principal de una arquitectura cliente-servidor consiste básicamente en que un programa, el **servidor gestiona un recurso compartido** y hace determinadas funciones cuando se las pide otro programa, el cliente, que es quien interactúa con el usuario.

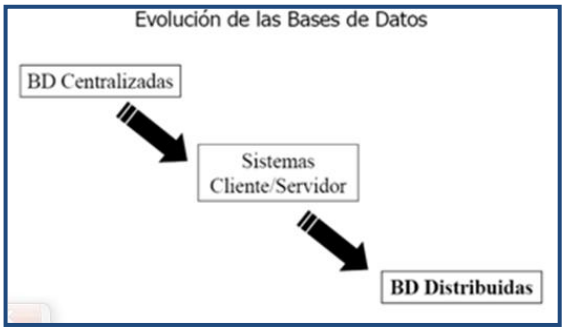
- ✓ La **arquitectura cliente-servidor tiene una estructura con varios ordenadores**, todos **interconectados** entre sí mediante algún tipo de red de computadoras.
- ✓ En el servidor **se almacena el sistema gestor de base de datos y la base de datos** y los programas de los clientes puedan emitir instrucciones para el acceso a la base de datos.
- ✓ El **cliente, provee capacidades de interfaz de usuario y procesamiento local**. Si alguno de los clientes solicita cierta funcionalidad adicional, entonces la máquina cliente se conecta con un servidor que proporcione dicha funcionalidad.
- ✓ **El cliente no tiene la responsabilidad de acceso a datos** y maneja meramente la gestionados por el servidor.
- ✓ El **servidor ejecuta y maneja las funciones** relativas **al acceso** compartido concurrente, **acepta sentencias originadas por aplicaciones del cliente, las procesa, y devuelve los resultados al cliente**.



- ✓ La **arquitectura cliente/servidor reduce el tráfico de red y divide la carga de trabajo de la base de datos**. Las funciones íntimamente relacionadas con el usuario, tales como el manejo de entradas y la visualización de datos, se concentran en los clientes. Las funciones de intenso procesamiento de datos, tales como la entrada/salida de archivos y el procesamiento de consultas, se concentran en el servidor de la base de datos

11.3 Bases de datos distribuidas

En un sistema de base de datos centralizada todos los componentes (software, datos y soporte físico) residen en un único lugar. Existe otra arquitectura, cada vez más extendida, en la que se opta por un esquema distribuido en que los componentes se distribuyen en distintas computadoras comunicados a través de una red.



Una **base de datos distribuida** es una **colección de datos que pertenece lógicamente al mismo sistema pero que está dispersa físicamente entre los sitios de una red de computadoras**.

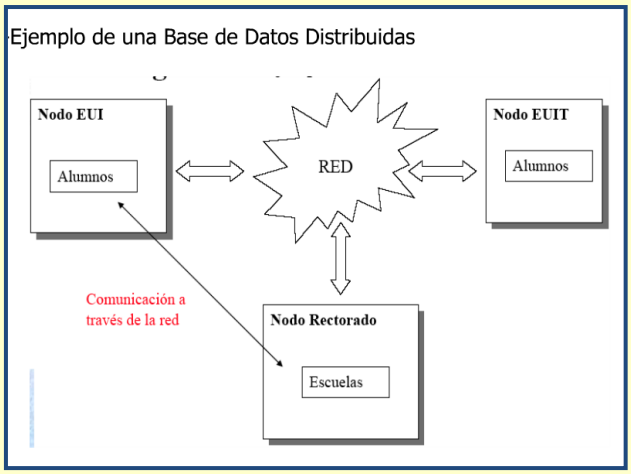
12. Base de datos distribuidas

- ✓ Un SGBDD se **compone de una única base de datos lógica,** que **físicamente está dividida en fragmentos ubicados en nodos distintos** e interconectados mediante una red de comunicaciones
- ✓ **La base de datos se almacena en varias computadoras situadas en lugares geográficos diferentes y se administran de forma separada.**
- ✓ Varios medios de comunicación, como las líneas telefónicas o las redes de alta velocidad son las que ponen en contacto las diferentes computadoras de un sistema distribuido.
- ✓ Las computadoras de un sistema distribuido pueden variar en tamaño y función, pudiendo abarcar desde las estaciones de trabajo a grandes computadoras, etc. A las computadoras que forman parte de un sistema distribuido suelen llamarse **nodos o emplazamientos**.

Las bases de datos distribuidas, a pesar de estar separadas físicamente, y correr en distintos sistemas operativos, la impresión para el usuario es la de estar trabajando en una base de datos "simple", es decir, que desde el punto de vista lógico, la base de datos se utiliza como si estuviese físicamente en el mismo sitio.

✓ Tipos de transacciones: Las **locales** y las **globales**.

- Una **transacción local** **es aquella que accede a los datos de un único emplazamiento en el cual se inició la transacción**.
- Por otra parte, una **transacción global** es **aquella que o bien accede a los datos situados en un emplazamiento diferente de aquel en el que se inició la transacción o bien accede a datos de varios emplazamientos distintos**.



Por ejemplo, una universidad puede tener varias escuelas en distintas ciudades, o un banco puede tener múltiples sucursales. Es natural que las bases de datos empleadas en las aplicaciones de estas organizaciones estén distribuidas en esos lugares. Muchos usuarios locales tienen acceso exclusivamente a los datos que están en el lugar, pero otros usuarios globales, cómo la oficina central de la compañía o el rectorado pueden requerir acceso ocasional a los datos almacenados en varios de estos lugares. Cabe señalar que, por lo regular, los datos en cada sitio local describen un "minimundo" en ese sitio. Las fuentes de los datos y la mayoría de los usuarios y aplicaciones de la base de datos local residen físicamente en ese lugar.

- **Nodos de las Escuelas:**

DNI	Escuela	Nombre	Nota ingreso	Beca

- **Nodo del Rectorado:**

Escuela	Situación	Número alumnos

- **Nuevo alumno en la secretaría del centro: transacción local.**
- **Nuevo alumno en el rectorado: transacción global**

Ventajas

Mejora en el rendimiento

✓ Cuando grandes bases de datos se distribuyen en varios sitios, **las consultas y transacciones que afectan a un solo sitio son más rápidas al ser más pequeña la base de datos local**.

✓ Los sitios **no se ven congestionados por muchas transacciones**, ya que **está dispersas entre varios** sistemas.

✓ De esto modo, cuando una transacción requiera acceso a más de un sitio, estos pueden hacerse en paralelo, de **forma que se reduce el tiempo de respuesta**.

Fiabilidad

Definida como la probabilidad de que un sistema esté activo en un tiempo dado.

✓ Al distribuirse los datos es **más fácil asegurar la fiabilidad del sistema, ya que si falla algún nodo es muy probable que no se vean afectadas la mayoría de las transacciones.**

Disponibilidad

Es la probabilidad de que un sistema esté activo de manera continuada durante un tiempo.

✓ **Aunque falle un nodo, el resto no tienen por qué verse afectados.**

Tipo de aplicaciones

Muchas aplicaciones tienen un marcado carácter distribuido al estar sus componentes dispersos en distintos sitios, por ejemplo: supermercados, bancos, tiendas etc.

Desventaja:

✓ Por el contrario, distribuir implica **una mayor complejidad en el diseño, la implementación y la gestión de los datos**, lo cual debe de incorporar, además de los componentes habituales en un SGBD:

Un SGBD-D es el software que gestiona BDD y suministra mecanismos de acceso que hace la distribución transparente al usuario.

La gestión de Base de Datos Distribuidas es el procesamiento de base de datos donde la ejecución de transacciones y la recuperación de datos y la actualización de los datos acontece a través de dos o más computadoras independientes, por lo general separadas geográficamente.

La figura siguiente muestra un sistema de base de datos distribuida que involucra a **los siguientes componentes**:

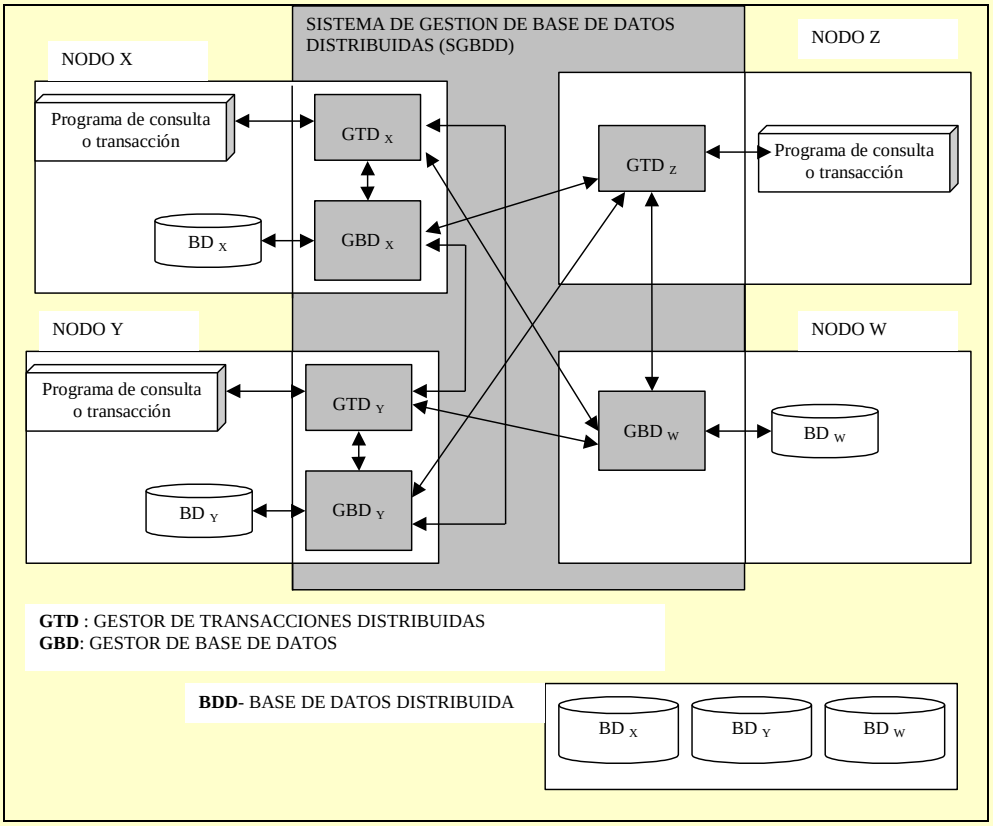
- El **Gestor de Transacciones Distribuidas (GTD)** es un programa que **recibe solicitudes de procesamiento de las transacciones o de los programas de consulta y a su vez las traduce en acciones para los sistemas de gestión de base de datos**. Una función importante del GTD es controlar y coordinar dichas acciones.

  El GTD se puede proporcionar como parte del SGBDD o puede desarrollarse por la propia organización para poner en práctica el sistema distribuido.

- **El Gestor de la Base de datos (GBD)** es un programa que **procesa cierta porción de la base de datos distribuida** de acuerdo con comandos de acción recibidos por los GTD.

- El **Un nodo** es una **computadora que ejecuta un GTD, un GBD o inclusive ambos**. Un nodo de transacción procesa un GTD y un nodo de base de datos procesa un GBD y su base de datos. En la figura siguiente, el nodo W es un nodo de base de datos ejecutando GBDW y almacenando BDW. El nodo Z es un nodo de transacción. Los nodos Y y Z son a la vez nodos de base de datos y Nodos de transacción...

- En el **diccionario o catalogo global de la BDD** se guardará información **sobre la ubicación de los datos**, sobre los **fragmentos de cada relación y sobre la duplicación de los datos**



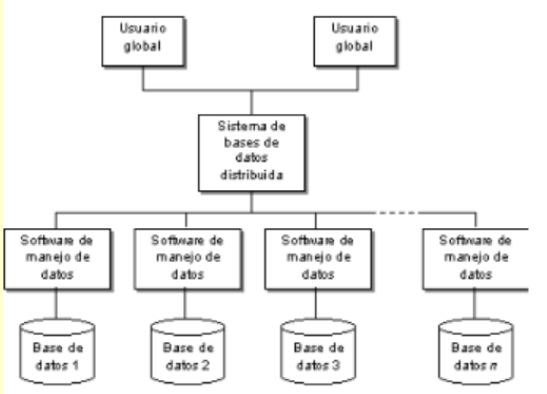
Varios SGBBD han sido implementados o están en desarrollo. Ej: System for Distributed database (SDD) comercializado por Computer Corporation America. R* de IBM. Distributed INGRESS de Relational Technology. El popular DB2 de IBM incluye soporte de distribución, lo mismo que SBBD ORACLE a partir de su versión 2 (actualmente se comercializa Oracle 11g) y sql server a partir 2000 (ultima SQL Server 2008 R2).

12.1 Tipos de base de datos distribuidas

Las BDD pueden ser:

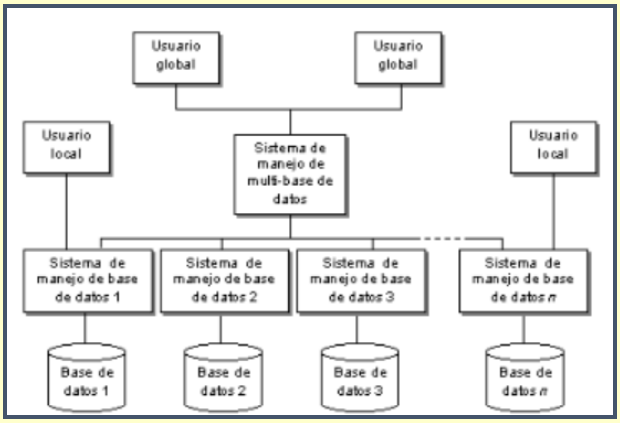
Homogéneas:

- ✓ **Todos los sitios tienen el mismo SGBD**, son conscientes de la existencia de los demás sitios y cooperan en el procesamiento de las solicitudes. Los nodos locales mantienen un mismo esquema y SGBD.



Heterogéneas:

- ✓ **Cada sitio puede tener un SGBD distinto** así como esquemas diferentes. Puede que algunos sitios no conozcan a otros.



12.2 Distribución de los datos: fragmentación y replicación

Una de las decisiones más importantes que el diseñador de bases de datos distribuidas debe tomar es el posicionamiento de los datos en el sistema y el esquema bajo el cual lo desea hacer. Para esto existen varias alternativas principales: fragmentación y la replicación

12.2.1 Fragmentación

El objetivo de la **fragmentación** consiste en **dividir la relación (tabla) en un conjunto de relaciones** más pequeñas tal que algunas de las aplicaciones de usuario sólo hagan uso de un fragmento.

Fragmentos: Cada relación global puede ser dividida en porciones llamados fragmentos. El mapa resultante se denomina esquema de fragmentación. Una relación global puede dividirse en n fragmentos y un fragmento sólo puede pertenecer a una relación global.

Fragmentación horizontal.

Se realiza sobre las filas de las tablas, es decir que cada fragmento será un subconjunto de las filas de cada tabla. Una tabla T se divide en subconjuntos, T1, T2, ...Tn. Los fragmentos se definen mediante una operación de selección. Cada fragmento se almacena en un nodo. Su reconstrucción se realizará mediante la unión de los fragmentos componentes.

Fragmentación vertical

La fragmentación vertical de una tabla T produce una serie de fragmentos T1, T2, ..., Tr cada uno de los cuales contiene un **subconjunto de los atributos** de T así como la clave primaria de T.

Fragmentación mixta

Como su nombre indica es una **combinación de la fragmentación vertical y la fragmentación vertical**.

Ejemplo (frag. Horizontal)

TABLA EMPLEADO

EMP_ID	NOMBRE	DEPT	SALARIO
E1	SANDRA	D1	5000000
E2	ALFREDO	D1	4000000
E3	GUILLERMO	D2	8000000
E4	JUAN	D3	9000000

FRAGMENTO 1 DE LA TABLA EMPLEADO

EMP_ID	NOMBRE	DEPT	SALARIO
E1	SANDRA	D1	5000000
E2	ALFREDO	D1	4000000

FRAGMENTO 2 DE LA TABLA EMPLEADO

EMP_ID	NOMBRE	DEPT	SALARIO
E3	GUILLERMO	D2	8000000

FRAGMENTO 3 DE LA TABLA EMPLEADO

EMP_ID	NOMBRE	DEPT	SALARIO
E4	JUAN	D3	9000000

Ejemplo (frag. vertical)

TABLA EMPLEADO

MP_ID	NOMBRE	DEPT	SALARIO	EXP
E1	SANDRA	D1	5000000	PROGRAMADOR
E2	ALFREDO	D1	4000000	ANALISTA
E3	GUILLERMO	D2	8000000	DISEÑADOR
E4	JUAN	D3	9000000	PEON

FRAGMENTO 1 DE LA TABLA EMPLEADO

EMP_ID	NOMBRE	DEPT	EXP
E1	SANDRA	D1	PROGRAMADOR
E2	ALFREDO	D1	ANALISTA
E3	GUILLERMO	D2	DISEÑADOR
E4	JUAN	D3	PEON

FRAGMENTO 2 DE LA TABLA EMPLEADO

EMP_ID	EXP
E1	PROGRAMADOR
E2	ANALISTA
E3	DISEÑADOR
E4	PEON

FRAG_1 = Π EMP_ID, DEPT, NOMBI

Ventajas:	**Inconvenientes:**
✓ **Utilización.** Generalmente las aplicaciones trabajan con vistas en vez de con relaciones completas. ✓ **Eficiencia**. Los datos se almacenan dónde más se utilizan. ✓ **Paralelismo**. Las transacciones pueden dividirse en subconsultas que operan con fragmentos mejorando la disponibilidad de los datos y el acceso concurrente. ✓ **Seguridad.** Los datos no necesarios localmente no se almacenan y se evita su uso por los usuarios no autorizados ✓ **Disminuye el tráfico de red**.	✓ **Consultas más lentas** al tener que buscar datos de diferentes fragmentos en distintas sedes. ✓ **Aumenta la complejidad** para garantizar la integridad, consistencia y recuperabilidad.

12.2.2 Replicación

- La Replicación de Datos es un servicio esencial para muchas aplicaciones distribuidas y **consiste en que cada nodo debe tener su copia completa de la base de datos**.
- En general, existe una **copia principal y varias copias secundarias, a las cuales se propagan las modificaciones de forma asíncrona**.
- Se proveen productos llamados replicadores, cuya función principal es mantener la consistencia entre las copias. Funcionan de forma transparente a las aplicaciones que se ejecutan en el servidor SGBD.
- Existen dos **tipos de propagación de modificaciones**:

Incremental:	**Total**:
La información que se envía desde la copia principal a las secundarias **son las variaciones en los datos**.	Se envía **toda la copia principal completa**.
Principal Utilidad: Hacer que **el sistema sea menos sensible a los fallos**, ya que si la copia principal no estuviera disponible se puede seguir usando alguna de las copias secundarias.	

Es fácil ver que este esquema tiene un alto costo en el almacenamiento de la información. Debido a que la actualización de los datos debe ser realizada en todas las copias, también tiene un alto costo de escritura, pero todo esto vale la pena si tenemos un sistema en el que se va a escribir pocas veces y leer muchas, y dónde la disponibilidad y fiabilidad de los datos sea de máxima importancia.

13. SGBD propietarios y libres

La **elección de un gestor de bases de datos en una empresa** no es algo ni mucho menos trivial. De partida, puede llegar a ser una inversión tanto en hardware como en software muy cuantiosa.

A la hora de elegir siempre se buscan productos con **reconocido prestigio, fiabilidad, velocidad, rendimiento, facilidad de administración y conexión con otros productos, bien documentados**, con una buena evolución y soporte. Productos de los que sea fácil obtener información, con buenas herramientas, y para los que incluso podamos recibir cursos de formación si fuese necesario. Productos que estén siendo utilizados en muchos entornos productivos y que nos den la suficiente confianza.

Con el advenimiento de Internet, el **software libre se ha consolidado como alternativa, técnicamente viable y económicamente sostenible al software propietario**, contrariamente a lo que a menudo se piensa, convirtiéndose el **software libre como otra alternativa para ofrecer los mismos servicios a un costo significativamente reducido**, encontrando estas alternativas tanto para sistemas operativos, herramientas de ofimática, software especializado, sistemas de gestión de bases de datos. Son productos cada vez más evolucionados, con más funcionalidades y las empresas que los desarrollan tienen también cada vez más volumen de negocio (cuestión importante para continuar la evolución de los productos).

No obstante, cada vez más los fabricantes de software propietario y comerciales ofrecen versiones gratuitas (no libres), aunque con limitaciones en su funcionalidad, de sus productos para probarlos. Estas versiones se suelen denominar *express.*

Con independencia de la versión del producto disponemos, tanto en los productos de software libre como comerciales, de cada vez **más documentación en línea que facilita** enormemente su aprendizaje aunque en este sentido los productos de software libre destacan ya que los usuarios y comunidades generan una gran cantidad de documentación que está permanentemente actualizada.

SGBD libres más conocidos son: **MySQL, PostgreSQL, FireBird, MariaDB, SQLite, Ingres, EntrepriseDB, MonetDB, LuciDB**.	**SGBD propietarios** más potentes y utilizados son: **Oracle, DB2 de IBM y SQL Server de Microsoft-**

Dentro de las **funcionalidades comunes** que ofrecen los anteriores SGBD tanto libres como comerciales tenemos:

- **Incorporan un lenguaje que es compatible con ANSI-SQL** para Definición de Datos (LDD), Manipulación de Datos (LMD), Control de usuarios y administración de la seguridad (LCD).
- **Mecanismos para definir la integridad referencial**: conjunto de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.
- **Procedimientos almacenados** son un **conjunto de instrucciones SQL,** a las que se les asigna un nombre, son compiladas y almacenadas en un servidor de base de datos. Después de que se define un procedimiento almacenado en la base de datos, éste puede ser llamado desde un programa de aplicación, mediante su nombre.
- **Disparadores o trigger: clase de procedimiento almacenado que se realiza de acuerdo con un evento**, cuando éste ocurra dentro de la base de datos.
- **Cursores:** soportan el desarrollo de programas de base de datos, permitiendo al usuario **moverse libremente a través de un conjunto de resultados de consultas**
- **Vistas**: Una vista de base de datos **es un resultado de una consulta SQL de una o varias tablas; también se le puede considerar una tabla virtual**.
- **índices:** Un índice es un mecanismo de la BD utilizado para agilizar el acceso a una fila de una tabla.
- **Seguridad:** Un SGBD cuenta con un **subsistema de seguridad y autorización que se encarga de garantizar la seguridad de porciones de la BD contra el acceso no autorizado**. Los SGBD tienen opciones que permiten manejar la seguridad, para proteger los datos contra acceso, alteración o destrucción no autorizados, conceder privilegios de acceso a los usuarios, protección frente a caídas o fallos en el software o en el equipo, mecanismos de copia de seguridad y restauración. También suelen tienen varios archivos de auditoria en donde se registran las operaciones que realizan los usuarios., la actividad del sgbd, etc..
- **Integridad de las transacciones y control de concurrencia**: proporciona los mecanismos que garanticen la **integridad de los datos cuando varios usuarios accedan a ellos concurrentemente.**

Sistemas Gestores de Bases de Datos Libres.

SGBD	Descripción	URL
MySQL	Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Distribuido bajo dos tipos de licencias, comercial y libre. Multiplataforma, posee varios motores de almacenamiento, accesible a través de múltiples lenguajes de programación y muy ligado a aplicaciones web.	http://www.mysql.com/
PostgreSQL	Sistema Relacional Orientado a Objetos. Considerado como la base de datos de código abierto más avanzada del mundo. Desarrollado por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales. Es multiplataforma y accesible desde múltiples lenguajes de programación.	http://www.postgresql.org/
Firebird	Sistema Gestor de Base de Datos relacional, multiplataforma, con bajo consumo de recursos, excelente gestión de la concurrencia, alto rendimiento y potente soporte para diferentes lenguajes.	http://www.firebirdsql.org/
Apache Derby	Sistema Gestor escrito en Java, de reducido tamaño, con soporte multilenguaje, multiplataforma, altamente portable, puede funcionar embebido o en modo cliente/servidor.	http://db.apache.org/derby/
SQLite	Sistema relacional, basado en una biblioteca escrita en C que interactua directamente con los programas, reduce los tiempos de acceso siendo más rápido que MySQL o PostGreSQL, es multiplataforma y con soporte para varios lenguajes de programación.	http://www.sqlite.org/

Sistemas Gestores de Bases de Datos Comerciales.

SGBD	Descripción	URL
ORACLE	Reconocido como uno de los mejores a nivel mundial. Es multiplataforma, confiable y seguro. Es Cliente/Servidor. Basado en el modelo de datos Relacional. De gran potencia, aunque con un precio elevado hace que sólo se vea en empresas muy grandes y multinacionales. Ofrece una versión gratuita **Oracle Database 10*g* Express Edition.**	http://www.oracle.com/us/products/database/product-editions-066501.html?ssSourceSiteId=ocomes
MYSQL	Sistema muy extendido que se ofrece bajo dos tipos de licencia, comercial o libre. Para aquellas empresas que deseen incorporarlo en productos privativos, deben comprar una licencia específica. Es Relacional, Multihilo, Multiusuario y Multiplataforma. Su gran velocidad lo hace ideal para consulta de bases de datos y plataformas web.	http://www.mysql.com/
DB2	Multiplataforma, el motor de base de datos relacional integra XML de manera nativa, lo que IBM ha llamado pureXML, que permite almacenar documentos completos para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.	http://www.ibm.com/developerworks/ssa/downloads/im/udbexp/
INFORMIX	Otra opción de IBM para el mundo empresarial que necesita un DBMS sencillo y confiable. Es un gestor de base de datos relacional basado en SQL. Multiplataforma. Consume menos recursos que Oracle, con utilidades muy avanzadas respecto a conectividad y funciones relacionadas con tecnologías de Internet/Intranet, XML, etc.	http://www-01.ibm.com/software/es/data/informix/discover-informix/index.html
Microsoft SQL SERVER	Sistema Gestor de Base de Datos producido por Microsoft. Es relacional, sólo funciona bajo Microsoft Windows, utiliza arquitectura Cliente/Servidor. Constituye la alternativa a	http://www.microsoft.com/spain/sql/2008/overview.aspx
	otros potentes SGBD como son Oracle, PostgreSQL o MySQL.	
SYBASE	Un DBMS con bastantes años en el mercado, tiene 3 versiones para ajustarse a las necesidades reales de cada empresa. Es un sistema relacional, altamente escalable, de alto rendimiento, con soporte a grandes volúmenes de datos, transacciones y usuarios, y de bajo costo.	http://www.sybase.es/products/databasemanagement/adaptiveserverenterprise