# Bases de Datos

## Tema 1: Almacenamiento de la Información

# T.1.-Almacenamiento de la Información

# 0.- Concepto de información

**Organización lógica de los datos**

- Una estructura de datos es una colección de datos que se caracterizan por su **organización y por las operaciones** que sobre dicha estructura puedan definirse.
  - En función de esto, los datos pueden ser:

  –**Simples:** no se pueden dividir en elementos de tamaño menor.

  Ejemplo: 1999, como valor de 'Año'.

  –**Compuestos:** formados como agregación de datos simples.

  Ejemplo: 7/10/2015, como valor de 'Fecha', compuesto por el día (7), el mes (10) y el año (2015).

# 0.- Concepto de información

- Un **registro** es la estructura capaz de almacenar datos en forma de unidades homogéneas de información dentro de un almacén de información: un **archivo.**

**Está compuesto por** estructuras de menor tamaño denominadas **campos.**

- Se distinguen:
  - –Registros lógicos.
  - –Registros físicos.

# 0.- Concepto de información

- ✓ **Registro lógico: Unidad de información *homogénea*** compuesta por datos referentes a un determinado objeto o concepto, siendo el **campo la unidad** elemental de información dentro de un registro lógico.

- ✓ **Registro físico: Unidad de transmisión o de** almacenamiento de información. Conjunto de información que, en función de las características de la máquina, puede grabarse o leerse de una sola vez.

  Por lo general, un registro físico está formado por varios registros lógicos

# 0.- Concepto de información

# 0.- Concepto de información
## Almacenamiento de la Información

- **Almacenamiento Primario:** Medios sobre los que la CPU puede acceder directamente y por tanto más rápidamente. Es la memoria RAM.
- **Almacenamiento Secundario:** Dispositivos de almacenamiento secundario. En los discos duros cada pista se subdivide en bloques o sectores de tamaño fijo determinado por el S.O (512, 1024 ó 4096 bytes). La transferencia de información entre memoria y disco tiene lugar en unidades de bloque. Cuando se produce una orden de lectura se copian uno o varios bloques en el llamado buffer de memoria (área reservada de Memoria principal), si se realiza una escritura se copia el bloque correspondiente al disco.
- **Almacenamiento Intermedio:** Es parte de la memoria principal que se utiliza para agilizar la E/S de datos, es el Buffer.

# 1.- Ficheros

- Son estructuras de información que crean los Sistemas Operativos de los ordenadores para poder almacenar datos. Suelen tener un nombre y una extensión que determina el formato de la información que contiene.
- El formato y el tipo de fichero determinan la forma de interpretar la información que contiene, realmente el fichero sólo contiene un conjunto binario de 0 y 1s que debe ser interpretado.

# 1.- Ficheros

Un **archivo,** desde el punto de vista lógico, es un conjunto de registros *afines considerados, a efectos de* un proceso, como una colección de unidades simples de información, de las mismas características en cuanto a estructura, significado y tipo de tratamiento.

Físicamente, se puede definir como un conjunto de datos que tienen entre sí una relación lógica y están almacenados en un soporte de información adecuado para la comunicación con el ordenador.

- Un archivo almacena información referente a un mismo

tema de forma **estructurada,** con el fin de manipular

los elementos que lo componen de forma individual.

# 1.- Ficheros. Clasificación

Los ficheros pueden ser clasificados de muchas formas:

- **Según su Contenido**, pueden ser:
  - **Binarios**, tienen que ser interpretados.
  - De imagen: .jpg, .gif, .tiff, .bmp, .wmf, .png, .pcx
  - De video: .mpg, .mov, .avi, .qt
  - Comprimidos: .zip, .rar, .gz, .tar, .Z, .mbz
  - Ejecutables o Compilados: .exe, .a, .o, .cgi, .com
  - Procesadores de Texto: .doc, .docx, .odt
  - Hojas de cálculo: .xlsx, .ods

# 1.- Ficheros. Clasificación

Generalmente los ficheros que componen una BD son binarios. Por ejemplo el software de gestión de la BD Oracle guarda la información en múltiples ficheros (datafiles, tempfiles, logfiles, etc.), Access guarda la información de una BD con extensión .mdb o .accdb

- **De Texto**: También llamados Ficheros planos o ficheros ASCII (American Standard Code for Information Interchange) Tablas ascii/unicode

# 1.- Ficheros. Clasificación

- Los ficheros de texto, no necesitan ser interpretados, pero suelen tener extensiones que sirven para conocer el tipo de información que contienen. Por ejemplo:
  - Ficheros de configuración: .ini, .inf, .conf
  - Ficheros de código fuente: .java, .c, .sql
  - Ficheros de Páginas web: .html, .php, .css, .xml
  - Formatos enriquecidos: .rtf, .ps, .tex

# 1.- Ficheros. Clasificación

➢ **Según su Organización:**

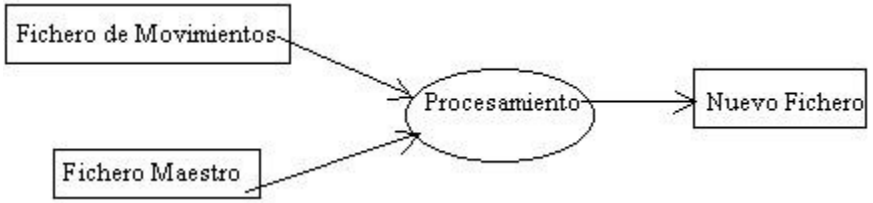
En función de la organización interna de los datos:

- **Secuencial:** Los registros se escriben sobre el dispositivo de almacenamiento en posiciones físicamente contiguas.
  - ✓ Ventajas: velocidad de acceso, fragmentación externa mínima.
  - ✓ Inconvenientes: consulta secuencial, dificultad en la inserción o eliminación de registros intermedios.
- Los archivos secuenciales son típicamente utilizados en aplicaciones de proceso por lotes y son óptimos para dichas aplicaciones si se procesan todos los registros. La organización secuencial de archivos es la única que es fácil de usar tanto en disco como en cinta. En el caso de registros de longitud fija, el acceso al registro i vendrá dado por: NºBloque=i/FB y el registro concreto estará en i mod FB.
- Para las aplicaciones interactivas que incluyen peticiones o actualizaciones de registros individuales, los archivos secuenciales ofrecen un rendimiento pobre.

# 1.- Ficheros. Clasificación

**Ventajas de los ficheros secuenciales:**

- Consulta muy rápida en procesamiento secuencial.
- Las modificaciones  obligan a hacer una copia del fichero.




- Proceso lento para consultas puntuales
- Aprovechan mucho el espacio de almacenamiento (sólo se precisa el justo para los datos)
- Posibilidad de usar cualquier tipo de soporte.
- Problema para procesar un fichero por más de una clave (campo de registro), ya que si un registro está ordenado en función de una clave no puede estarlo por otra. Las soluciones a este problema son: o bien se tienen dos ficheros iguales o más (tantos como clasificaciones diferentes haya) cada uno ordenado con respecto a una clave, o bien se clasifica el fichero cada vez que se quiera acceder (lo cual es muy lento).

# 1.- Ficheros. Clasificación

## Organización Secuencial

Variantes:

- **Secuencial encadenada.** Los registros se procesan en el orden lógico (uno detrás de otro), pero este no tiene porqué coincidir con el orden físico (los registros se enlazan por punteros). Es imprescindible un soporte de acceso directo.


El orden lógico sería: Manzana,Naranja, Pera, Limón

# 1.- Ficheros. Clasificación
## Organización Secuencial

En la Organización **Secuencial encadenada l**os registros contienen un campo adicional que indica qué registro es el siguiente en la secuencia lógica.

- ✓ Ventajas: velocidad de acceso, facilita la inserción de nuevos registros.
- ✓ Inconvenientes: aumento de tamaño para almacenar la posición del siguiente, la eliminación es lógica (no libera espacio físico).

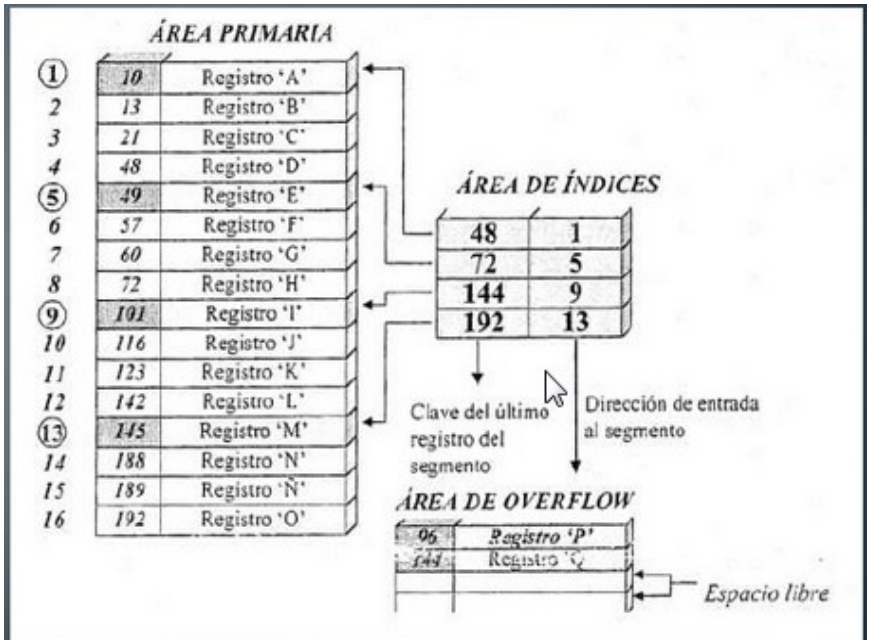
# 1.- Ficheros. Clasificación
## Organización Secuencial

- **Secuencial indexada:** En este modo de organización, al fichero le acompaña un fichero de índice que almacena las direcciones de los registros, a los que se accede secuencialmente, permitiendo el acceso casi directo a los registros del fichero de datos.
  - ✓ Ventajas: velocidad de acceso, facilita la inserción de nuevos registros.
  - ✓ Inconvenientes: aumento de tamaño para almacenar las posiciones, aumento de tiempo en las operaciones de actualización.

- Los registros se organizan en una secuencia basada en un campo clave presentando dos características, un **índice** del archivo para lograr el acceso más rápido a los datos y un **archivo de desbordamiento**. El índice proporciona una capacidad de búsqueda para llegar rápidamente al registro deseado y el archivo de desbordamiento es similar al archivo de registros usado en un archivo secuencial, pero está integrado de forma que los archivos de desbordamiento se ubiquen siguiendo un puntero desde su registro predecesor.
- Cada registro del archivo índice tiene dos campos, un **campo clave** igual al del archivo principal y un **puntero** al archivo principal. Para encontrar un campo específico se busca en el índice hasta encontrar el valor mayor de la clave que es igual o precede al valor deseado de la clave, la búsqueda continua en el archivo principal a partir de la posición que indique el puntero.

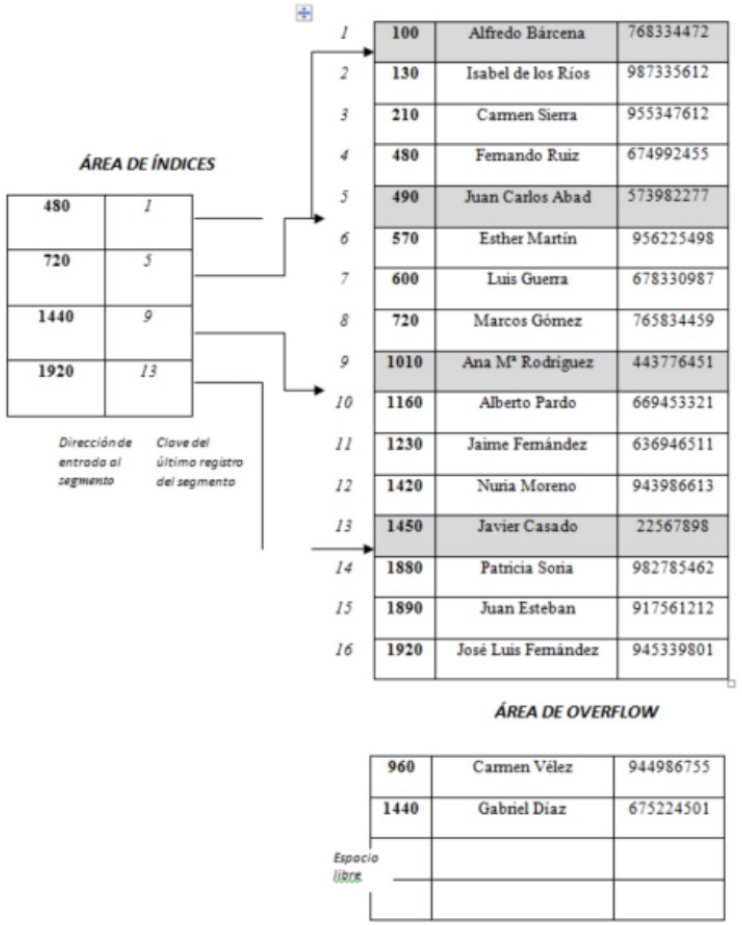
# 1.- Ficheros. Clasificación

## Organización Secuencial Indexada




En esta organización se dispone de una tabla de índices adicional. Un índice es una referencia que permite obtener de forma automática la ubicación de la zona del archivo físico donde se encuentra el registro buscado. Es decir, permite localizar un registro por medio de su clave sin recorrer previamente todos los que le preceden. Un ejemplo sería un diccionario.

# Ejemplo Fichero Secuencial Indexado



# 1.- Ficheros. Clasificación
# Organización Secuencial Indexada

Las características más relevantes de un archivo indexado, son las siguientes:

- Permiten utilizar el modo de acceso secuencial y el modo de acceso directo para leer la información guardada en sus registros.
- Solamente se puede grabar en un soporte direccionable.
- El diseño del registro tiene que tener 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.
- Implica un mantenimiento de la tabla de índices y una previsión inicial de la cantidad máxima de registros que va a contener.
- Consta de dos ficheros, el de índices y el de datos, el primero contiene la clave del último registro de cada bloque y en el segundo están los registros de datos, clasificados en orden ascendente por el campo clave.

# 1.- Ficheros. Clasificación
## Organización Directa

- **Directa:** Se utiliza una clave numérica que identifica los registros que componen un archivo. Normalmente se utilizan funciones de Hash para transformar la clave en una posición física.

✓ Ventajas: acceso inmediato a los registros a partir de su clave, permite realizar operaciones simultáneas de lectura/escritura, muy rápidos en el tratamiento individual de registros.

✓ Inconvenientes: consultas lentas para el fichero completo, el disco puede contener huecos libres. Sólo se puede utilizar con registros de longitud fija.

# 1.- Ficheros. Clasificación

➢ **En función del uso** que se hace de ellos:

- **Permanentes**: se almacenan en memoria auxiliar, para permitir su uso posterior aunque se apague el equipo. A su vez, se clasifican en:

  - Maestros.

  - Constantes.

  - Históricos

- **Temporales:** su tiempo de vida es de corta

Duración, pueden ser intermedios, de maniobras o de resultados.

# 1.- Ficheros
## Métodos de acceso

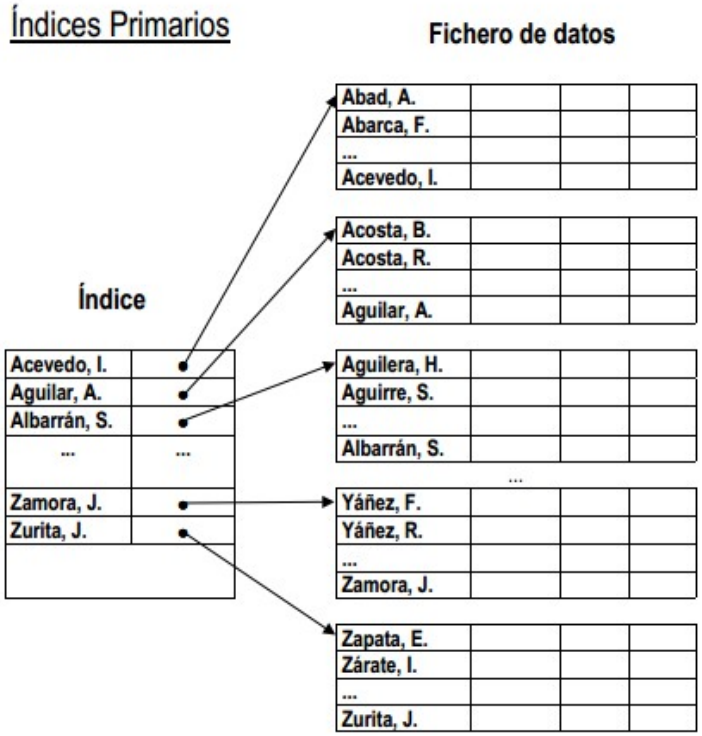
Para poder utilizar la información almacenada en un archivo, las aplicaciones deben acceder a la misma y almacenarla en memoria. Hay distintas formas de acceder a un archivo. Para una aplicación, elegir adecuadamente la forma de acceder a un archivo suele ser un aspecto importante de diseño, ya que, en muchos casos, el método de acceso tiene un impacto significativo en el rendimiento de la misma.

Dependiendo de que se pueda ir de una posición a otra de un archivo, pasando por todas las posiciones anteriores o no, se distinguen dos métodos de acceso principales: ***acceso secuencial*** y ***acceso directo***.

# 1.- Ficheros
# Tipos de Índices

- Otra clasificación de Tipos de índices es:
  - **Índices primarios**: el índice es el campo clave de ordenación del fichero (su valor no puede repetirse en otro registro del fichero).
  - **Índices de agrupamiento**: el campo de ordenación no es el campo clave, es decir, pueden haber valores repetidos .
  - **Índice secundario**: cuando el campo de indización no es el de ordenación.

# 1.- Ficheros
# Índices Primarios




**Entradas**: registros de longitud fija.

**Campo de indexación**: campo clave de ordenación del fichero de datos.

**Índice no denso**.

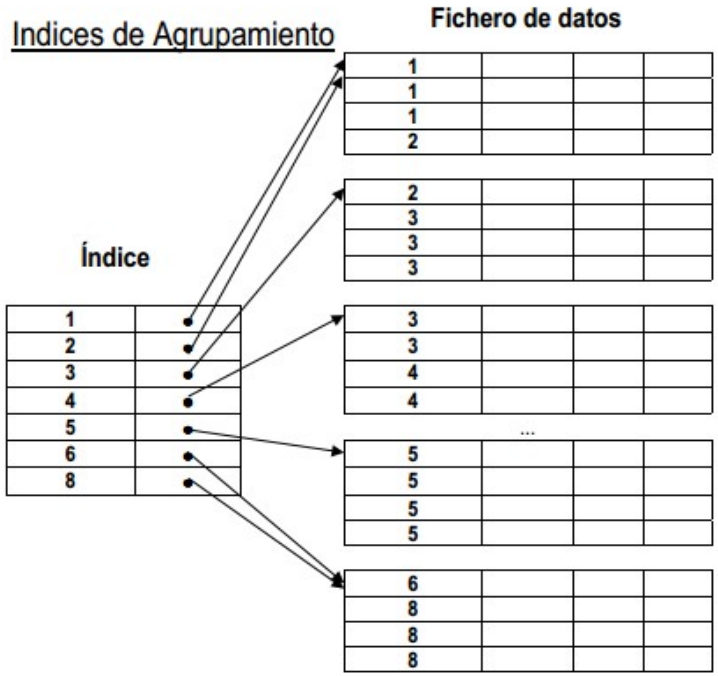
**Entrada**: valor de la clave del primer/último registro del bloque y puntero a dicho bloque.

**Búsqueda binaria** sobre el índice: visita menos bloques de disco.

**Problema**: son ficheros ordenados (opciones: fichero de desbordamiento desordenado; lista enlazada de registros de desbordamiento).

**Importante**: sobre un fichero ordenado por clave sólo puede definirse un índice primario.

# 1.- Ficheros
## Índices de Agrupamiento




**Entradas**: registros de longitud fija.

**Campo de indexación**: campo no clave de ordenación del fichero de datos (campo de agrupamiento).

**Índice no denso**

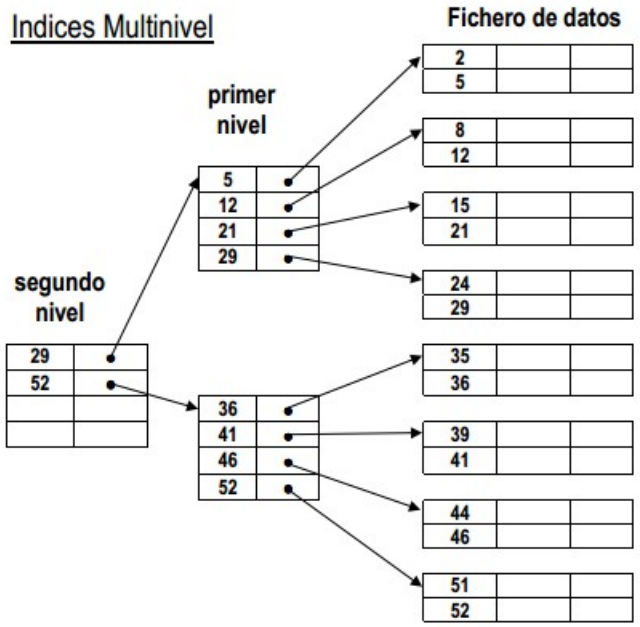
**Entradas**: una por cada valor distinto del campo de agrupamiento. El puntero apunta al primer bloque que contiene un registro con dicho valor.

**Búsqueda binaria** sobre el índice: visita menos bloques de disco.

**Problema**: son ficheros ordenados. (opción: reservar un bloque entero para cada valor distinto del campo de agrupamiento).

**Importante**: sobre un fichero ordenado por un campo no clave sólo puede definirse un índice de agrupamiento.

# 1.- Ficheros
## Índices Multinivel




**Objetivo**: reducir más que con la búsqueda binaria el trozo de índice en donde seguir buscando.

**Primer nivel**: fichero ordenado con entradas de tamaño fijo y un valor distinto del campo de indexación en cada una.

**Siguientes niveles**: índices primarios sobre el nivel anterior.

Número de registros por bloque: **r**

- primer nivel $i_1$ entradas
- segundo nivel $i_2 = \lceil i_1/r \rceil$ entradas
- tercer nivel $i_3 = \lceil i_2/r \rceil$ entradas ...

Se necesita un nivel más si el anterior ocupa más de un bloque.

Un índice multinivel con $i_1$ **entradas** en el primer nivel tiene $\lceil \log_r i_1 \rceil$ **niveles**.

Reducen el número de accesos a bloque al hacer búsquedas, **pero son ficheros ordenados**.

# 1.- Ficheros

**Inconvenientes de los sistemas de ficheros:**

- ✓ Separación y aislamiento de los datos.
- ✓ Duplicación o Redundancia de datos que origina fallos en la Integridad de la información.
- ✓ Dependencia de la codificación de los datos en los programas.
- ✓ Formatos de ficheros incompatibles.

- Estos inconvenientes de los sistemas de ficheros se pueden atribuir a dos factores:

- ➢ La definición de los datos se encuentra codificada dentro de los programas de aplicación, en lugar de estar almacenada por separado de forma independiente.
- ➢ No hay control sobre el acceso y la manipulación de los datos, fuera de lo impuesto por los programas de aplicación.

# 2.- Evolución de los sistemas de ficheros a las bases de datos

En los sistemas de información clásicos soportados sobre ficheros los datos se recogen varias veces y se encuentran repetidos en distintos archivos. Esta redundancia, además de malgastar recursos, puede producir divergencias en los resultados. Estos sistemas, ponen el énfasis en el tratamiento que reciben los datos, que se organizan en ficheros que se diseñan para una determinada aplicación.

- Las aplicaciones se diseñan e implantan con total independencia unas de otras y los datos no se suelen transferir entre ellas, sino que se duplican siempre que los distintos trabajos los necesiten.

# 2.- Evolución de los sistemas de ficheros a las bases de datos

El sistema *clásico de gestión de ficheros genera además una* **ocupación excesiva de memoria secundaria**, un **aumento de los tiempos de proceso**, al repetirse los mismos controles y operaciones en los distintos ficheros y, sobre todo, **inconsistencias** debidas a que la actualización de los mismos datos que, cuando se encuentran en más de un fichero, no puede realizarse de forma simultánea en todos ellos.

- La solución es utilizar un "**sistema orientado a los datos**" en el que los datos sean recogidos y almacenados una sola vez, con independencia de los tratamientos que sobre los datos se realicen. En una **base de datos**, los datos se mantienen y organizan en un conjunto estructurado que no está diseñado para una aplicación concreta, sino que, por el contrario, tiende a satisfacer las necesidades de información de toda la organización.

# 2.- Evolución S.Ficheros => BD

## SISTEMAS DE BASES DE DATOS

**Definición**

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

**Problemas sistemas de ficheros**

- Separación/aislamiento de los datos
- Duplicación de datos
- Dependencia
- Formato
- Concurrencia
- Autorizaciones
- Catálogo

# 2.- Evolución S.Ficheros => BD

# 2.- Evolución S.Ficheros => BD

**Niveles de abstracción en una base de datos:**

- **Conceptual :** vista del usuario**.** Es el que percibe el usuario a través de las aplicaciones.
- **Lógico :** representación global de los datos, independiente tanto del *hardware* como de cada usuario en particular. Debe incluir:
  - La descripción de **todos los datos.**
  - La descripción de las **interrelaciones entre los datos.**
  - Las **restricciones de integridad y de confidencialidad**
- **Físico:** datos almacenados en forma de bits. Debe especificar, al menos:
  - Estrategia de almacenamiento.
  - Métodos y rutas de acceso.
  - Otros: compresión, encriptado, ajustes, etc.

# 3.- Concepto de Bases de Datos

Es una colección de información perteneciente a un mismo contexto, que está almacenada de forma organizada en ficheros en soporte secundario (no volátil) y con redundancia controlada. Los datos deben ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se debe apoyar en un modelo de datos, que permite captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación, comunes y bien determinados, facilitarán la seguridad del conjunto de los datos.

# 3.1- Concepto de Bases de Datos

En una base de datos debe existir **independencia** entre los programas y los datos que los manejan.

- En una base de datos, los mismos datos pueden ser utilizados por distintas aplicaciones y usuarios.
- La base de datos debe permitir **métodos para consultar, insertar, modificar** y **eliminar** datos.
- Una base de datos almacena información de dos tipos:

  1. **Datos del usuario,** el contenido de la base de datos.

  2. **Datos del sistema,** tanto la estructura como los datos de control necesarios para su manejo.
- El Sistema de Gestión de Bases de Datos es el conjunto de programas que permiten la implantación, acceso y mantenimiento de una base de datos. El SGBD y la base de datos constituyen el Sistema de Base de Datos.

# 3.2.- Componentes de una Base de Datos (1)

- **Dato**: Porción de información concreta sobre un concepto o suceso.
- **Tipo de dato**: indica la naturaleza del campo (numérico, alfanumérico , compuesto)
- **Campo o Columna**: es un identificador para un dato. Cada campo pertenece a un tipo de dato.
- **Registro, Tupla o Fila**: Conjunto de datos referente a un mismo concepto o suceso.
- **Campo Clave**: Campo especial que identifica de forma única a cada registro.
- **Tabla**: Conjunto de registros bajo un mismo nombre que representa al conjunto de todos ellos.
- **Consulta**: Instrucción para hacer peticiones a una BD. Puede ser una consulta simple de un registro concreto o una solicitud para seleccionar todos los registros que satisfagan un conjunto de criterios. Así como una petición de inserción, eliminación o modificación/actualización de registros.

# 3.2- Componentes de una Base de Datos (2)

- **Índice**: Estructura que almacena los campos clave de una tabla, organizándolos para hacer más fácil encontrar y ordenar los registros de la tabla. Para buscar un elemento que esté indexado, sólo hay que buscar en el índice de dicho elemento para, una vez encontrado, devolver el registro que se encuentre en la posición marcada por el índice.
- **Vista**: Transformación que se hace a una o más tablas para obtener una nueva tabla virtual, es decir, que no está almacenada en memoria secundaria, aunque si se almacena su definición.
- I**nforme**: Listado ordenado de los campos y registros seleccionados.
- **Guiones o Scripts**: Conjunto de instrucciones, que ejecutadas de forma ordenada, realizan operaciones de mantenimiento de datos en la BD.
- **Procedimientos**: Tipo especial de Scripts que está almacenado en la BD y forma parte de su esquema.

# 3.3.- Estructura de una BD

Los datos se almacenan a través de un **Esquema**, que es la definición de la estructura donde se almacenan los datos, contiene todo lo necesario para organizar la información mediante tablas, registros y campos. También contiene otros objetos necesarios para el tratamiento de los datos (procedimientos, vistas, índices, etc.). También se le llama **Metainformación**, e.d, información sobre la información o metadatos.

Los gestores suelen almacenar el esquema en tablas.

# 3.4.- Evolución y Tipos BD

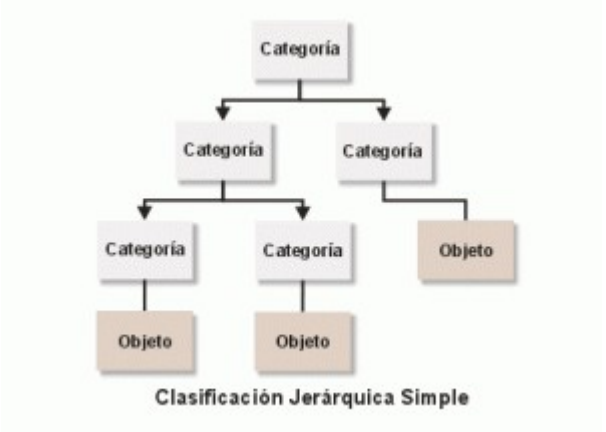
Según ha ido avanzando la tecnología, las BD han ido cambiando la forma de representar y extraer la información.

Década de 1950 unida a la utilización de cintas magnéticas => aplicaciones basadas en **Sistemas de Ficheros**.

Década de 1960 se generaliza la utilización de los discos magnéticos => BD **Jerárquicas y en Red**.

El **modelo jerárquico** se parece a un árbol invertido:

Representa bien las relaciones de 1 a N, pero no puede representar relaciones N a M. Además resulta complicado añadir nuevas relaciones y el Desarrollo de aplicaciones resulta complejo ya que El programador necesita conocer la estructura de datos utilizada.

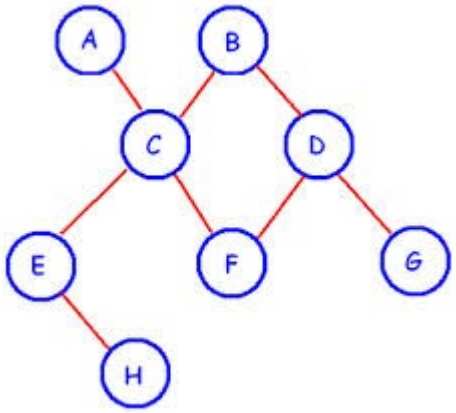


Clasificación Jerárquica Simple

# 3.4.- Evolución y Tipos BD

- El **modelo en Red** se desarrolló a partir del modelo jerárquico para resolver su falta de flexibilidad, por ejemplo, permite que un nodo de nivel inferior (hijo o miembro) tenga dos nodos de nivel superior(padre o propietario), permitiendo modelar relaciones de N a M.

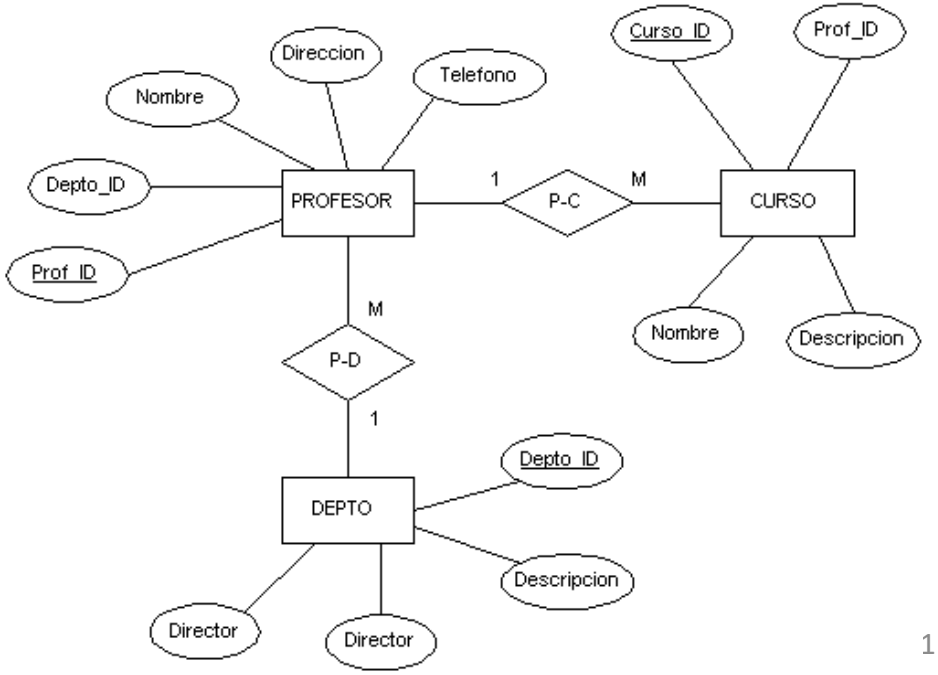
  Este modelo resulta mucho más difícil de implementar y mantener que el jerárquico y sigue presentando problemas con las relaciones, además el programador sigue teniendo que conocer bien las estructuras de datos para lograr que el modelo sea eficiente.



# 3.4.- Evolución y Tipos BD

- Década de 1970 Edgar Frank Codd publica el Modelo Relacional de Datos, basado en la lógica de predicados y teoría de conjuntos.
- Este Modelo supuso un gran avance, en vez de basarse en una relación de superior a inferior, permite que cualquier fichero se relacione con otro a través de un campo común.

La complejidad se redujo mucho, ya que los cambios se podían realizar sobre el esquema de la BD sin que ello afectara a la capacidad del sistema para acceder a los datos. Pero este modelo resultó inaplicable en 1970, ya que la facilidad de uso presentaba un coste importante de rendimiento y el hardware en aquellos años no podía implementarlo.



# 3.4.- Evolución y Tipos BD

- Década de 1980 IBM lanza su motor de BD **DB2** para la plataforma MVS. Y después IBM crea el **SQL.**
- Década de 1990 IBM lanza su nueva versión de DB2 para BD paralelas.
- Tras la aparición de Internet y el crecimiento de la información, se crean las BD **Distribuidas**, se multiplica el número de ordenadores que controlan una BD, surgiendo las BD **Multidimensionales**, que forman los denominados 'cubos de información'.

# 3.4.- Evolución y Tipos BD

- Las aplicaciones web generan información de forma masiva: Millones de usuarios accediendo simultáneamente y generando consultas a BD
- Las BD relacionales presentan problemas de escalabilidad y rendimiento
- La estructura E-R no se ajusta a la estructura de la información generada en la web: Objetos, XML, Grafos …
- Se requieren nuevos sistemas de almacenamiento de la información para solucionar las nuevas necesidades de la web
  - **NoSQL** al rescate – se conoce como “Big Data”

# 3.4.- Evolución y Tipos BD

**NoSQL Diferencias Relacional:**

- No se usa SQL como lenguaje de consultas
  - Cassandra utiliza el lenguaje CQL, MongoDB utiliza JSON o BigTable hace uso de GQL
- No se utiliza la estructura de tabla para el almacenamiento
  - Se basan en objetos, grafos, pares clave-valor …
- No se permiten operaciones JOIN
  - Para evitar la sobrecarga que generaría un join con los volúmenes de datos generados en la web
- Arquitectura distribuida
  - La información se distribuye entre varias máquinas para mejorar el rendimiento

# 3.4.- Evolución y Tipos BD

- **NoSQL Ventajas:**
- Máquinas con pocos recursos
  - ✓ Requieren menos computación que un SGBD Relacional por lo que se requieren menos recursos HW
- Escalabilidad horizontal
  - ✓ Al ser un sistema distribuido el rendimiento se puede mejorar añadiendo más nodos
- Información escalable
  - ✓ Al ser distribuidos son capaces de manejar grandes volúmenes de datos
- Se evita el principal cuello de botella de un SGBD basado en SQL
  - ✓ Para ejecutar cada sentencia hay que transcribirla, si se producen muchas peticiones se ralentiza el sistema

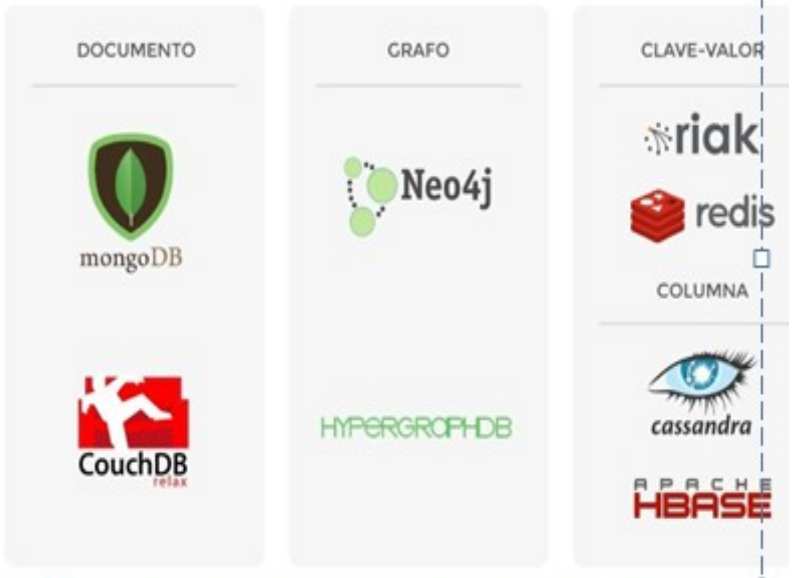
# 3.4.- Evolución y Tipos BD

- **NoSQL Inconvenientes:**
- Dependiendo de los requisitos del sitio pueden ser más eficientes pero para muchos sitios un sistemas de bases de datos relacional sigue siendo lo óptimo.
- Sólo se deben utilizar cuando son necesarias y van a mejorar realmente el rendimiento.
  - El modelo de datos cambia rápidamente y se necesita un alto rendimiento y escalabilidad, es probable que lo que mejor sean soluciones NoSQL.
  - Si por el contrario, la estructura de los datos no suele cambiar y el crecimiento de los datos es “moderado”, las tecnologías SQL serán la mejor opción

# 3.4.- Evolución y Tipos BD

## NoSQL – Tipos de BD

- Clave – Valor
- Columnas
- Documentales
- Grafo
- Orientadas a Objetos



# 3.4.1.- Bases de datos relacionales

En el modelo relacional, la información se representa en forma de tablas, que almacenan información lógicamente relacionada. Estas tablas se relacionan formando vínculos o relaciones entre ellas que se establecen *implícitamente por la presencia* de campos de conexión comunes.

Cada tabla se compone de filas y columnas, cada fila almacena un registro con tantos campos como columnas tenga la tabla.

# 3.4.2.- Bases de Datos orientadas a objetos

- La información se almacena como objetos igual que en un lenguaje de programación orientado a objetos como Java.
- Técnicamente podrían considerarse bases de datos NoSQL aunque no encajan en el concepto NoSQL de no relacionales, distribuidas y escalables.
- Ejemplos:
  - Db4o: adquirida en 2008 por Versant, actualmente tienen su propio producto y desde 2011 ya no trabajan en ella. Se puede descargar la versión libre.
  - Zope: es un framwork con servidor de aplicaciones para el desarrollo de sitios web OO con python

# 3.4.2.- Bases de Datos orientadas a objetos

Las bases de datos orientadas a objetos surgen para cubrir nuevos tipos de aplicaciones:

- Diseño y fabricación asistido por ordenador (CASE,CAD/CAM).
- Bases de datos gráficas y de imágenes.
- Bases de datos científicas.
- Sistemas de información geográfica.
- Bases de datos multimedia.
- Acceso uniforme a sistemas de múltiples bases de datos.

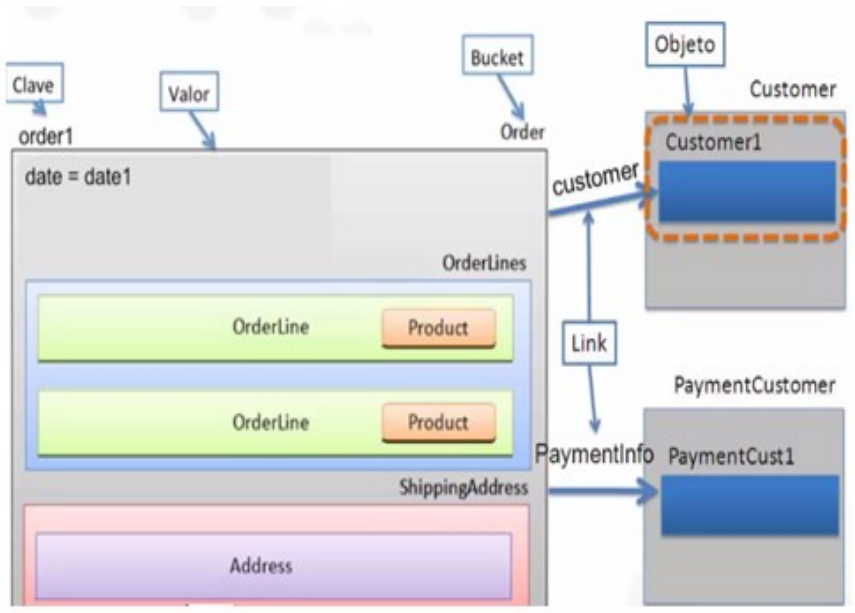
# 3.4.2.- Bases de Datos orientadas a objetos

- Gestionan objetos en los cuales están encapsulados los **datos** y las **operaciones** que actúan sobre ellos.
- Desarrollo bajo orientación a objetos un único modelo subyacente implementado en el sistema de datos, al que pueden acceder directamente las aplicaciones.
- Intentan satisfacer necesidades de aplicaciones más complejas que las tradicionales que utilizan nuevos tipos de datos para almacenar imágenes o grandes bloques de texto.
- Permiten al diseñador de la base de datos especificar la estructura de objetos complejos así como las operaciones que se pueden aplicar a estos objetos

# 3.4.3.- BD Clave-Valor

## NoSQL – Tipos de BD

- **Clave – Valor**
  - Son las más sencillas de entender. Guardan tuplas que contienen una clave y su valor. Cuándo se quiere recuperar un dato, se busca por su clave y se recupera el valor. El valor se suele almacenar como un binario (BLOB), lo que hace muy eficientes las lecturas y escrituras.



# 3.4.3.- BD Clave-Valor

- Clave – Valor
  - Ejemplos:
    - DynamoDB: desarrollada por Amazon, es una opción de almacenaje que podemos usar desde los Amazon Web Services. La utilizan el Washington Post y Scopely.
    - Redis: desarrollada en C y de código abierto, es utilizada por Craiglist y Stack Overflow ( a modo de caché).

# 3.4.4.- BD Columnas

- Columnas
  - Pensadas para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros. Todos los casos de un solo elemento de datos (por ejemplo, Nombre de Persona) se almacenan de modo que se puede acceder como una unidad

    De gran utilidad en consultas analíticas (Data mining o Bussiness Inteligence)

| Id | Apellido | Nombre | Salario |
|---|---|---|---|
| 1 | Gómez | José | 40000 |
| 2 | Jiménez | Mary | 50000 |
| 3 | Salabarrieta | Cathy | 44000 |

El sistema por filas lo serializa:

```
1, Gómez, Jose, 40000;
2,Jimenez,Mary,50000;
3,Gutierrez,Cathy,44000;
```

Pero el sistema por COLUMNAS lo serializa así:

```
1,2,3;
Gómez,Jiménez, Gutiérrez;
José, Mary, Cathy;
40000,50000,44000;
```

# 3.4.4.- BD Columnas

- Ejemplos:
  - Cassandra: incluida en esta sección, aunque en realidad sigue un modelo híbrido entre orientada a columnas y clave-valor. Es utilizada por Facebook y Twitter (aunque dejaron de usarla para almacenar tweets).
  - HBase. Escrita en Java y mantenida por el Projecto Hadoop de Apache, se utiliza para procesar grandes cantidades de datos. La utilizan Facebook, Twitter o Yahoo.

# 3.4.5.- BD Documentales

- **Documentales**
  - Son aquellas que gestionan datos semi estructurados. Es decir documentos. Estos datos son almacenados en algún formato estándar como puede ser XML ó JSON. Permiten consultas sobre el contenido de los documentos. Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales

```
{
  Name: "Genbeta Dev",
  Tipo: "Blogging",
  Categorias:
  [
    {
      Título: "Desarrollo",
      Artículos: 89
    },
    {
      Título: "Formación",
      Artículos: 45
    }
  ]
}
```

# 3.4.5.- Bases de datos documentales

- Una base de datos se crea y mantiene de forma continuada con el objetivo de resolver necesidades de información concretas de un colectivo, una empresa o el conjunto de la sociedad. Las bases de datos documentales pretenden cubrir las nuevas necesidades de *información*.
- En una base de datos documental, cada registro se corresponde con un **documento,** de tipo:
  - Audiovisual, gráfico o sonoro.
  - Publicación impresa.
  - Electrónico.
  - De archivo.
  - Etc.

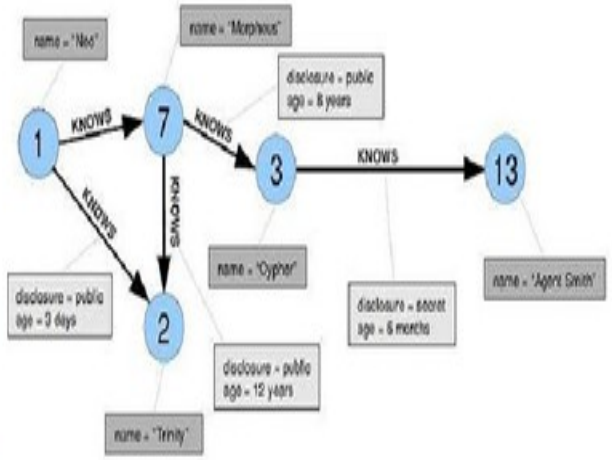
  Estos recursos electrónicos pueden consultarse directamente en formato electrónico o utilizarse para elaborar productos impresos, como bibliografías, informes, etc.

# 3.4.5.- Bases de datos documentales

- Documentales
  - Ejemplos:
  - MongoDB: probablemente la base de datos NoSQL más famosa del momento. En octubre del 2013, MongoDB conseguía 150 millones de dólares en financiación, convirtiéndose en una da las startups más prometedoras. Algunas compañías que actualmente utilizan MongoDB son Foursquare o eBay.
  - CouchDB: es la base de datos orientada a documentos de Apache. Este sistema es utilizado por compañías como Credit Suisse y la BBC

# 3.4.6.- BD Grafos

- **Grafo**
  - Basadas en la teoría de grafos utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes sociales. Para que sea eficiente se debe normalizar de forma que cada nodo tenga una sola columna y cada relación dos.



# 3.4.6.- BD Grafos

- Ejemplos:
  - Infinite Graph: escrita en Java y C++ por la compañía Objectivity. Tiene dos modelos de licenciamiento: uno gratuito y otro de pago.
  - Neo4j: base de datos de código abierto, escrita en Java por la compañía Neo Technology. Utilizada por compañías como HP, Infojobs o Cisco.

# 3.5.- Usos de las BD

- Los usos más frecuentes de las BD son:
  - ✓BD Administrativas
  - ✓BD Contables
  - ✓BD para Motores de Búsqueda
  - ✓Científicas
  - ✓Configuraciones
  - ✓Bibliotecas
  - ✓Censos
  - ✓Virus
  - ✓Militares
  - ✓VideoJuegos
  - ✓Deportes

# 4.- **Criterios de uso de ficheros o bases de datos**

- El uso de un sistema de ficheros:

– Facilita la **localización de la información en el** dispositivo de almacenamiento secundario.

– Reduce el tiempo de **desarrollo.**

– Asegura un **control exhaustivo sobre el** contenido de los archivos.

– Reduce el tiempo de **respuesta con archivos** de gran tamaño.

– Permite adaptar el formato de modo que el usuario pueda **consultar su contenido**

# 4.- **Criterios de uso de ficheros o bases de datos**

- El uso de una base de datos aporta:
- Mayor eficiencia en la recogida, validación e introducción de los datos en el sistema
- Reducción del espacio de almacenamiento.
- Independencia de los datos respecto de los programas que los utilizan y viceversa
- Asegura la coherencia de los resultados.
- Mejor disponibilidad de los datos para el conjunto de los usuarios.
- Mayor calidad y normalización en la documentación de la información, que se integra con los datos.

# 5.- Sistemas Gestores de Bases de Datos (SGBD)

- Un SGBD es el conjunto de herramientas que facilitan la consulta, uso y actualización de una BD. Un ejemplo es Oracle 11g, que incorpora un conjunto de herramientas software que son capaces de estructurar en múltiples discos duros los ficheros de una BD, permitiendo el acceso a sus datos tanto a partir de herramientas gráficas como a partir de potentes lenguajes de programación (PL-SQL, php, c++ …)

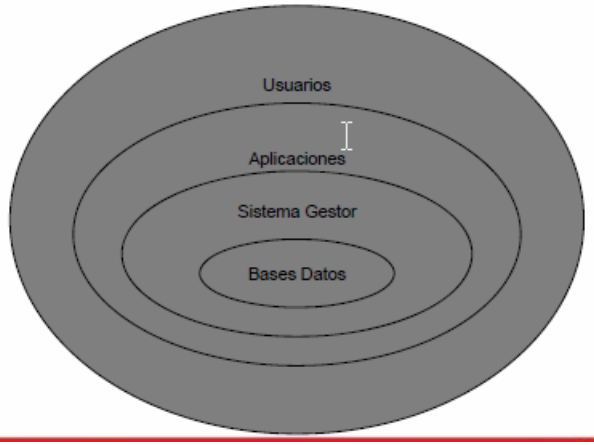
# 5.- Sistemas Gestores de Bases de Datos (SGBD)

- **Ejemplos de gestores de base de datos libres:**
  - ✓ Firebird
  - ✓ BDB
  - ✓ MySQL
  - ✓ PostgreSQL
  - ✓ Sqlite
- **Ejemplos de gestores de base de datos propietarios:**

- ✓ dBase
- ✓ FileMaker
- ✓ Fox Pro
- ✓ IBM DB2 Universal Database (DB2 UDB)
- ✓ IBM Informix
- ✓ MAGIC
- ✓ Microsoft SQL Server
- ✓ Open Access
- ✓ Oracle

# 5.1.- Concepto de SGBD

*Un sistema de gestión de la base de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma. Es una herramienta que sirve de interfaz entre el usuario y las bases de datos.*



# 5.2.- Funciones de un SGBD (1)

1.- Permiten a los usuarios almacenar datos, acceder a ellos y actualizarlos de forma sencilla

2.- Garantizan la integridad de los datos, no permitiendo operaciones que dejen un conjunto de datos incompleto o incorrecto.

3.- Proporcionan, junto al S.O, un sistema de seguridad que garantiza el acceso a la información exclusivamente a los usuarios autorizados.

4.- Proporcionan un diccionario de metadatos, que contiene el esquema de la BD y que es fácilmente accesible.

5.- Permite el uso de transacciones, garantizando que todas las operaciones de la transacción se realicen correctamente, y en el caso de que haya alguna incidencia se deshagan los cambios.

# 5.2.- Funciones de un SGBD (2)

6.- Ofrecen estadísticas sobre el uso del gestor, posibilitando la monitorización del uso de la BD.

7.- Permiten la concurrencia.

8.- Independizan los datos de la aplicación o usuario que esté utilizándolos, haciendo más fácil su migración a otras plataformas.

9.- Ofrecen conectividad con el exterior. El protocolo ODBC (Open DataBase Connectivity) se utiliza frecuentemente como forma de comunicación entre BD y aplicaciones externas.

10.- Incorporan herramientas para realizar copias de seguridad y restauración de la información.

# 5.3.- Lenguaje SQL(Structure Query Language)

Es el interfaz de comunicación con el usuario. Está estandarizado por la ISO (International Organization for Standarization). Se compone de cuatro lenguajes:

- **DDL**: Lenguaje de Definición de Datos. Se ocupa de la creación/eliminación de la estructura de la BD (CREATE, DROP)
- **DML**: Lenguaje de Manipulación de Datos. Permite seleccionar datos (SELECT), insertarlos (INSERT), eliminarlos (DELETE) y actualizarlos (UPDATE).
- **DCL**: Lenguaje de Control de Datos. Incluye GRANT y REVOKE, que permiten al administrador gestionar el acceso a los datos contenidos en la BD.
- **TCL**: Lenguaje de Control de Transacciones. Permite ejecutar varios comandos de forma simultánea como si fuera un comando atómico o indivisible (COMMIT) y si ocurriera algún incidente podría deshacer los pasos dados (ROLLBACK).

# 5.3.- Lenguaje SQL(Structured Query Language)

- El Lenguaje de definición de datos (**DDL**, *Data Definition Language)* Permite especificar los datos, su estructura y las relaciones que existen entre ellos. Debe proporcionar los datos para los niveles interno, conceptual y externo, especificando las características de los mismos en cada nivel:

  − A nivel interno, se debe indicar el espacio físico para los datos, tamaño y tipo de datos, así como los métodos de acceso.

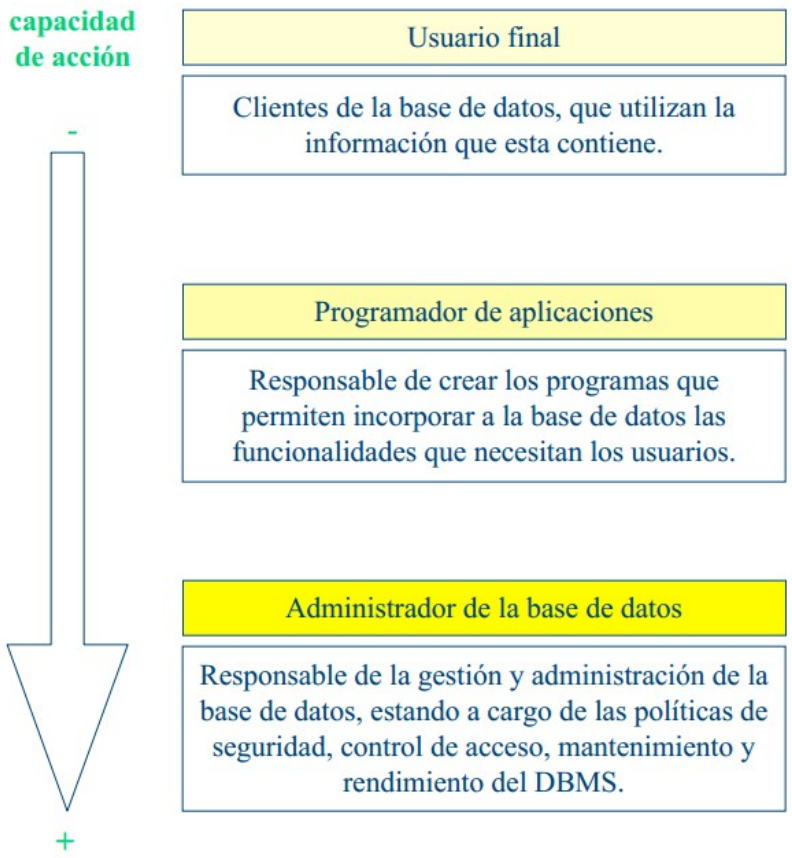
  − A nivel conceptual y externo, se deben definir entidades, sus atributos, interrelaciones entre ellos, permisos de acceso, etc.
- El lenguaje de manipulación de datos (**DML**, *Data Manipulation Language)* Permite recuperar, añadir, suprimir o modificar los datos respetando siempre las normas de seguridad y especificaciones del administrador. Proporciona un conjunto de procedimientos que facilitan las tareas del administrador.
- El Lenguaje de Control de Datos (**LCD** o DCL, *Data Control Language).* Permite:
  - Gestionar la capacidad de los ficheros.
  - Obtener estadísticas de uso.
  - Gestionar los procedimientos de copia de seguridad.
  - Rearranque en caso de caída del sistema.
  - Protección frente a accesos no autorizados.

# 5.4.- Tipos

- Según modelo lógico:
  - Jerárquico
  - En red
  - Relacional
  - Objeto-relacional
  - Orientado a objetos
- Según número de sitios:
  - Centralizado
  - Distribuido
- Según tipo de datos:
  - Relacionales
  - XML
  - Objeto-relacionales
  - Orientados a objetos
- Según número de usuarios:
  - Monousuario
  - Multiusuario
- Según Ámbito de aplicación
  - Propósito general
  - Propósito específico
- Según lenguajes soportados:
  - SQL
  - NoSQL

Según su capacidad y potencia se pueden distinguir Los GBD **Ofimáticas**, como el Microsoft Access con VBA (*Visual Basic for Applications)* como lenguaje y los GBD **Corporativos** como Oracle, DB2 y MySQL.

# 6.- Usuarios de la base de datos



# BIBLIOGRAFÍA


http://admonberty.wordpress.com/2013/06/15/unidad-iii-archivo-secuencial-indexado/

- http://kochshito26.wordpress.com/2013/06/12/administracion-de-archivos-%E2%86%94-unidad-iii-archivo-secuencial-indexado/
- http://robotica.uv.es/pub/Libro/PDFs/CAPI6.pdf
- http://alcazaba.unex.es/asg/103173/Curso%202000-01/Tema6-2-2.doc
- http://es.slideshare.net/Fportavella/ficheros-con-organizacin-secuencial-indexada
- http://www3.uji.es/~mmarques/f47/teoria/te

# BIBLIOGRAFÍA


http://www3.uji.es/~aliaga/e44/Tema_03.pdf

http://html.rincondelvago.com/algoritmos-de-busqueda.html

https://www.youtube.com/watch?v=yoeV4Ex8C8U&list=PLs1sXiNvW4OyJCZs5WR3OjPZTIlqNcvQi&index=2&t=0s

https://www.youtube.com/watch?v=h6aGdYj7xAw