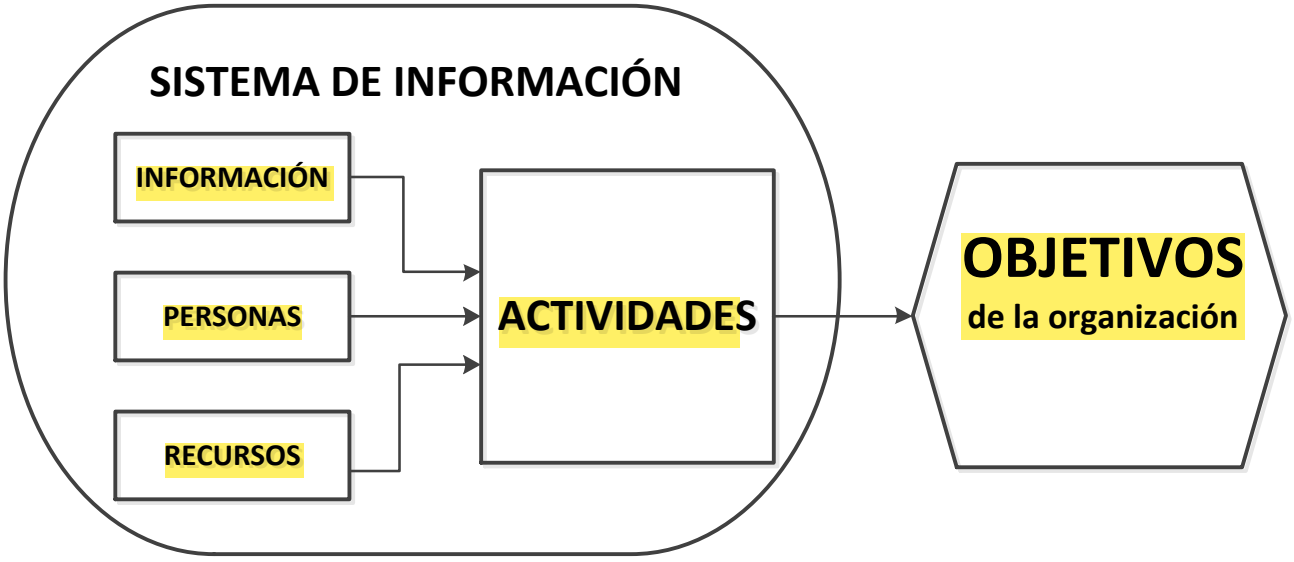
UD1. SISTEMAS DE ALMACENAMIENTO DE LA INFORMACIÓN

ÍNDICE

1. SISTEMAS DE INFORMACIÓN

1.1. DEFINICIÓN

Un **sistema de información (SI)** es un conjunto organizado de elementos, que pueden ser personas, información (datos), actividades (técnicas de trabajo) o recursos materiales. Estos elementos interactúan entre sí para tratar y administrar los datos, dando lugar a información más elaborada y distribuyéndola de la manera más adecuada posible en función de los objetivos de la organización.



1.2. SISTEMA DE INFORMACIÓN INFORMÁTICO

El término SI es usado **erróneamente** en muchas ocasiones como sinónimo de **sistema de información informático**. Este error se comete porque en la mayoría de los casos los recursos de un SI son casi en su totalidad sistemas informáticos, pero un SI no tiene por qué disponer de dichos recursos.

Podemos decir que los **sistemas de información informáticos** es un subconjunto de los sistemas de información en general.

Un **sistema de información informático** es el conjunto de herramientas informáticas que trata el desarrollo y la administración de la infraestructura tecnológica de una organización.

2. LOS FICHEROS DE INFORMACIÓN

2.1. INTRODUCCIÓN

En la década de los setenta, los procesos básicos que se llevaban a cabo en una empresa se centraban en cuestiones relacionadas con contabilidad y facturación. Las necesidades de almacenamiento y gestión de información podían satisfacerse utilizando un número relativamente reducido de archivos en papel agrupados y ordenados, los típicos **ficheros clásicos**.


Fichero clásico

DEL PAPEL AL ORDENADOR:

Al llevar a cabo una primera informatización, se pasó de tener los datos en formato papel a poder acceder a ellos de manera mucho más rápida a través del ordenador. En ese momento, 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 hablado de **ficheros**, **formularios**, **carpetas**,...

2.2. ¿QUÉ ES UN FICHERO INFORMÁTICO?

Todos los sistemas informáticos requieren de una memoria no volátil en la que se almacene la información. Esta memoria es la que se suele llamar **MEMORIA EXTERNA**. Así podremos disponer de la información (recuperarla) cuando deseemos.

Pero esta información debe organizarse de una manera coherente a su uso, es decir debemos darle una **ESTRUCTURA DE DATOS**.

FICHERO o ARCHIVO: Conjunto de información relacionada, tratada como un todo y organizada de forma estructurada.

Los ficheros están formados por **registros lógicos** que contienen datos relativos a un mismo elemento u objeto (por ejemplo, los datos de usuarios de un curso de formación).

A su vez, los registros están divididos en 1 o más **campos** que contienen cada una de las informaciones elementales que forman un registro (por ejemplo, el nombre del usuario, el primer apellido, email, teléfono…).

ESTRUCTURA LÓGICA DE UN FICHERO

REGISTRO 1	CAMPO1	CAMPO2	CAMPO3
REGISTRO 2	CAMPO1	CAMPO2	CAMPO3
	. . .	. . .	. . .
REGISTRO n	CAMPO1	CAMPO2	CAMPO3

EJEMPLO DEL CONTENIDO DE UN FICHERO DE DATOS DE USUARIOS

Usuario	Nombre	Ape1	Ape2	…	email
a23MariaGC	María	García	Pérez	…	a23MariaGC@iessanclemente.net
. . .		. . .	. . .	. . .	. . .
p20pablo	Pablo	Fiuza	Gil	…	pablo_profe@iessanclemente.net

Los ficheros suelen tener un gran volumen de datos por lo que se llevan a la memoria principal partes de ellos para poder procesarlos.

REGISTRO FÍSICO O BLOQUE: Cantidad de información que es transferida entre el soporte en el que se almacena el fichero, y la memoria principal del ordenador en una sola operación de lectura/grabación.

Normalmente en cada operación de lectura/grabación se transfieren varios registros del fichero, es decir **un bloque suele contener varios registros lógicos**.

FACTOR DE BLOQUEO: Número de registros lógicos de un bloque.

A la operación de agrupar varios registros en un bloque se le llama bloqueo de registros.

2.3. TIPOS

2.3.1. Ficheros según su función

2.3.1.1. Permanentes o maestros

Aquellos cuyo contenido refleja la situación actual de una entidad, empresa u organización. Tienen una vida equivalente a la de la aplicación que los usa.

2.3.1.2. De movimientos o transacciones

Sirven para actualizar los ficheros maestros. Existen 3 tipos de operaciones:

- **Altas:** añadimos nuevos registros al maestro
- **Bajas:** eliminamos registros del maestro
- **Modificaciones:** actualizamos registros del maestro

Normalmente estos ficheros se van llenando con actualizaciones necesarias, hasta que se alcanza un tamaño determinado, o el usuario o administrador decide actualizar el fichero maestro.

2.3.1.3. De maniobra o trabajo

Tienen información relativa a un proceso que todavía no se ha finalizado. Son creados por las aplicaciones y cuando el proceso termina se eliminan.

Su creación, uso y eliminación es **TRANSPARENTE** al usuario.

Ejemplo: Ficheros temporales como los *.tmp de Windows, que Windows mantiene mientras necesita y los borra cuando termina el proceso que los generó.

2.3.2. Ficheros según su organización

2.3.2.1. Organización secuencial

Para acceder al registro N, hemos de acceder a los N-1 anteriores. Podemos almacenarlo en soportes secuenciales o directos. No necesitamos ninguna clave de acceso. Cuando queremos localizar un dato debemos ir comprobando 1 a 1 si está.

Ejemplo: Un fichero secuencial puede ser un fichero con los nombres y puntuaciones ordenadas de los jugadores de un juego.

JUAN	3.500
CARMEN	2.000
SARA	1.500

2.3.2.2. Organización directa

Se puede acceder de forma aleatoria a cualquier registro.

Necesitamos soportes de acceso directo, pero también se pueden recorrer secuencialmente. Estos ficheros tienen una **CLAVE** que puede ser de 2 tipos:

- guarda la posición que ocupa el registro en el fichero => **DIRECCIONAMIENTO DIRECTO.**
- lo más habitual es que tengamos que aplicar una **función hash** a la clave para que nos dé su posición en el fichero => **DIRECCIONAMIENTO INDIRECTO**.

F(CLAVE) = Dirección/Posición

2.3.2.3. Ficheros indexados

Existen distintas formas de diseñar ficheros indexados (ISAM, VSAM...). Generalizando podemos decir que los ficheros indexados se componen de **3 partes o áreas**:

a) **Área primaria o de datos:** Contiene los datos en sí. Los registros se almacenan ascendentemente por su campo clave.
b) **Área de índices:** Fichero secuencial que divide el área primaria en distintos segmentos. De cada uno almacena la dirección de comienzo y la clave más alta.
c) **Área de excedentes o de overflow:** Se utiliza para añadir nuevos registros que no se introducen en el área primaria para evitar una reorganización del mismo.

Los **ficheros indexados**, requieren un campo clave por el que vamos a ordenar los datos, y que usaremos para realizar las búsquedas. Deberán estar almacenados en soportes que permitan el acceso directo.

2.4. INCONVENIENTES DE LOS SISTEMAS DE FICHEROS

2.4.1. Redundancia e inconsistencia de los datos

La concurrencia de distintos programadores durante un período largo de tiempo, puede ocasionar que el mismo dato esté en varios sitios.

Ejemplo: En el CPD (Centro de Proceso de Datos) de un Banco, la dirección y el número de teléfono del cliente pueden aparecer en 2 ficheros:

- en el fichero de cuentas de ahorros
- en el fichero de cheques

Esto incrementa el **coste de almacenamiento** y además puede provocar **inconsistencia** en los datos, si los datos varían de un fichero a otro.

Ejemplo: Si se cambia la dirección sólo en el fichero de cheques y en el otro no, nos preguntaríamos cual es la dirección correcta.

2.4.2. Dificultad para tener acceso a los datos

Supongamos que uno de los gerentes del banco quiere saber los nombres de todos los clientes que viven en cierta parte de la ciudad. El gerente llama al CPD y pide la lista. Pero hay un **problema:**

- **no existe un programa que obtenga ese listado**, sólo existe el que genera la lista de todos los clientes.

Existen **2 SOLUCIONES POSIBLES:**

- generar la lista total y comprobar 1 a 1 los clientes que nos interesan
- crear el programa que genere la lista

Cualquiera de las dos soluciones es poco práctica.

2.4.3. Aislamiento de los datos

Los datos están repartidos en distintos ficheros que pueden tener distintos formatos. Esto implica una complicación adicional a la hora de desarrollar programas nuevos para obtener los datos apropiados.

2.4.4. Usuarios múltiples

Muchos sistemas permiten que varios usuarios actualicen información a la vez. Esto puede provocar que se genere información inconsistente.

Ejemplo: Si dos clientes titulares de una misma cuenta quitan dinero a la vez, el saldo resultante puede ser inconsistente.

Saldo Inicial 10.000 Euros

CLIENTE 1 ____Quita 5.000€____ Sin actualizar

CLIENTE 2 ____ Quita 7.000€____ ? ¿Cómo se actualiza el saldo?

2.4.5. Problemas de seguridad

No todos los usuarios deben tener acceso a la misma información.

Ejemplo: Una persona que prepara cheques de nóminas de los empleados debe poder ver los datos de los empleados, pero no los de los clientes.

Como los programas se generan en función de las necesidades que van surgiendo, es complicado establecer en un Sistema de Ficheros la seguridad.

2.4.6. Problemas de integridad

Los valores de los datos deben satisfacer las restricciones.

Ejemplo: El saldo de una cuenta no debe bajar nunca de un límite.

El sistema debe hacer que se cumplan esos limitantes. Esto provoca un PROBLEMA importante ya que cuando se añaden nuevas restricciones es complicado cambiar los programas para que verifiquen que éstas se cumplen.

2.5. CONCLUSIÓN

Para evitar estos problemas de los sistemas de ficheros, surgen las **bases de datos**.

3. INTRODUCCIÓN A LAS BASES DE DATOS (BBDD)

3.1. INTRODUCCIÓN

Actualmente, en las empresas y en los organismos oficiales se maneja una gran cantidad de datos, por lo que se hace necesario disponer de hardware (HW) y software (SW) que permita acceder de una manera **rápida, sencilla** y **fiable**.

Tradicionalmente la información se almacenaba en conjuntos de ficheros, lo que suponía la existencia de información redundante y a veces inconsistente. Existían problemas por ejemplo cuando se agregaban nuevos campos a un fichero, ya que:

AGREGAR CAMPO => CAMBIAR ESTRUCTURA DEL FICHERO =>
=> MODIFICAR PROGRAMAS QUE USAN EL FICHERO =>
= > PÉRDIDA DE TIEMPO Y DINERO

Resumiendo, los **sistemas gestores de ficheros generan problemas por la dependencia entre ficheros y aplicaciones**.

En los años 60 surgieron las bases de datos (BBDDD). En una BD se almacenan los datos. Los programas que manejan esos datos no se deben preocupar de cómo están almacenados físicamente, así ningún cambio en la estructura de los datos afectará a las aplicaciones.

❖ **EJEMPLO COMPARATIVO ENTRE LOS 2 ENFOQUES (Ficheros versus BBDD)**

- **Enfoque tradicional con FICHEROS:** Disponemos de un fichero maestro de alumnos que es usado/accedido por varios programas de aplicación. El formato de registro es:

Nombre alumno	Dirección	Población	Código Postal

Si queremos variar la estructura del registro añadiendo un campo más, por ejemplo, el campo relativo a la *fecha de nacimiento*, tendríamos que cambiar toda la estructura del fichero maestro y lo que es más costoso, modificar los programas que acceden a ese fichero.

- **Enfoque de BD:** Disponemos de una **TABLA** (estructura de datos para una BD relacional) con las siguientes columnas:

Nombre alumno	Dirección	Población	Código Postal

Si añadimos una nueva columna a la tabla, por ejemplo, la referente al teléfono, NO AFECTARÁ a los programas de aplicación que manejan esa tabla.

En las BBDD existe INDEPENDENCIA de los datos, con respecto a los procedimientos que los utilizan. Esto implica que ningún cambio en la estructura de datos afectará a los programas que los usen.

3.1.1. Definición de BD

Una BD es un conjunto de **datos** interrelacionados y almacenados sin redundancias perjudiciales o innecesarias, **los cuales se caracterizan por**:

- servir a 1 ó más aplicaciones de la mejor manera posible,
- existir independencia entre los datos y los programas que los manejan.

Debido a que una misma BD se puede usar por distintas aplicaciones y distintos usuarios, **en la BD se guarda información de 2 TIPOS**:

- **Datos del usuario:** Datos de las aplicaciones de usuario, datos guardados en *tablas de usuario*.

  Ejemplo: Datos de alumnos: apellidos, nombre, dirección, población.
- **Datos del sistema:** Es aquella información que la BD necesita para gestionarse a sí misma.

  Ejemplo: Datos de usuarios: nombre, contraseña, privilegios...

  Estos datos son imprescindibles porque la BD debe conocer:
  - **quienes acceden**
  - cómo acceden, y,
  - a qué acceden.

3.2. OBJETIVOS DE LA ORGANIZACIÓN DE UNA BD

Independientemente de cómo esté organizada, una BD debe cumplir los siguientes objetivos para que sea considerada como tal:

3.2.1. Versatilidad para representar la información

Los mismos datos deben poder ser usados de distintas maneras, según para qué se utilicen.

Ejemplo: Un **programador** usará un conjunto de datos de distinto modo a un **usuario** que sólo quiere hacer una consulta.

3.2.2. Desempeño

Las BBDD han de atender las peticiones con la velocidad adecuada al uso que se vaya a hacer de los datos.

Ejemplo: La reserva de plazas en aerolíneas debe ser casi inmediata.

3.2.3. Redundancia mínima

En los sistemas gestores de ficheros existía un alto nivel de redundancia, lo que hacía que los mismos datos apareciesen repetidos en varios archivos, incrementando el coste de almacenamiento, y ocasionando inconsistencia de los datos repetidos, ya que en un momento dado podían tener valores diferentes.

Uno de los objetivos de las BBDD es eliminar la redundancia, siempre y cuando esto no afecte al rendimiento de la misma ni la haga muy compleja (hay casos en los que se permite cierta redundancia para evitar un problema mayor).

3.2.4. Capacidad de acceso rápido

Los usuarios de BD continuamente están pidiendo información acerca de los datos. Esto obliga a que un objetivo a conseguir por las BBDD sea lograr un acceso rápido a los mismos.

3.2.5. Integridad

Debido a que los datos son usados por muchos usuarios y de distintas maneras, el sistema de BBDD ha de asegurar que **los datos que se almacenan son los que realmente se tienen que almacenar**.

3.2.6. Seguridad y privacidad

- **Seguridad:** Protección frente a fallos del sistema o frente a usos de personas no autorizadas.
- **Privacidad:** Los datos serán accesibles (públicos) para unos usuarios y no accesibles (privados) para otros.

3.2.7. Afinación

Ajuste de la organización física de los datos para acceder a ellos más rápidamente. Con el tiempo el volumen de datos que necesita la entidad/empresa/organismo va aumentando, por lo que la organización física de los datos debe ser muy buena para que el acceso a los mismos no se vaya haciendo más lento a medida que aumenta la cantidad de datos.

3.2.8. Interfaz con el pasado y el futuro

Este objetivo perseguido por las BBDD hace referencia a que los **cambios que se realicen**, en los programas o en los datos, **no impidan trabajar con lo existente hasta el momento**.

Debe existir:

- **Independencia FÍSICA de los datos:** Cuando se modifica la organización física de los datos (**por ejemplo**, cambiar el soporte físico donde se almacenan) no debe afectar a los programas que haya en uso.
- **Independencia LÓGICA de los datos:** Cuando se cambia la estructura lógica de los datos (**por ejemplo,** añadir nuevos campos a una tabla), éstos no deben afectar a los programas que hasta ahora usaban esos datos.

3.3. ARQUITECTURA DE UNA BD

Los usuarios de una BD deben tener **una visión lo más abstracta posible** de los datos almacenados en ella; es decir, no tienen necesidad de saber cómo están organizados y almacenados. La visión de los datos por parte del usuario debe ser una **VISIÓN TRANSPARENTE**.

Por esto, la arquitectura (el diseño) de las BBDD está constituida por **3 niveles**, según la visión que se tenga de los datos:

3.3.1. Nivel interno o físico

En este nivel se describe cómo se almacenan realmente (físicamente) los datos. Este nivel se representa mediante **un único esquema interno** para la BD.

Se describen en el esquema interno de cada BD:

- los ficheros que guardan la información
- la organización de los ficheros
- ubicación de los ficheros
- registros y tipos de registros
- información de los campos
- los índices...

3.3.2. Nivel conceptual

En este nivel se describen cuáles son los datos reales que están almacenados en el BD y qué relaciones existen entre ellos.

Aunque una estructura sencilla del nivel conceptual, tenga que representarse de manera compleja en el nivel físico, el usuario del nivel conceptual no tiene porqué darse cuenta de la complejidad de la organización de la BD a nivel físico.

El nivel conceptual se representa con un único esquema conceptual.

Este nivel de abstracción lo emplean los programadores de la BD, que son quienes deciden qué información se guarda en la BD.

IMPORTANTE: Los administradores también emplean este nivel para determinadas tareas, porque para otras necesitan analizar la BD desde el nivel físico.

En este nivel no se tienen en cuenta ni la organización física ni los métodos de acceso a los ficheros.

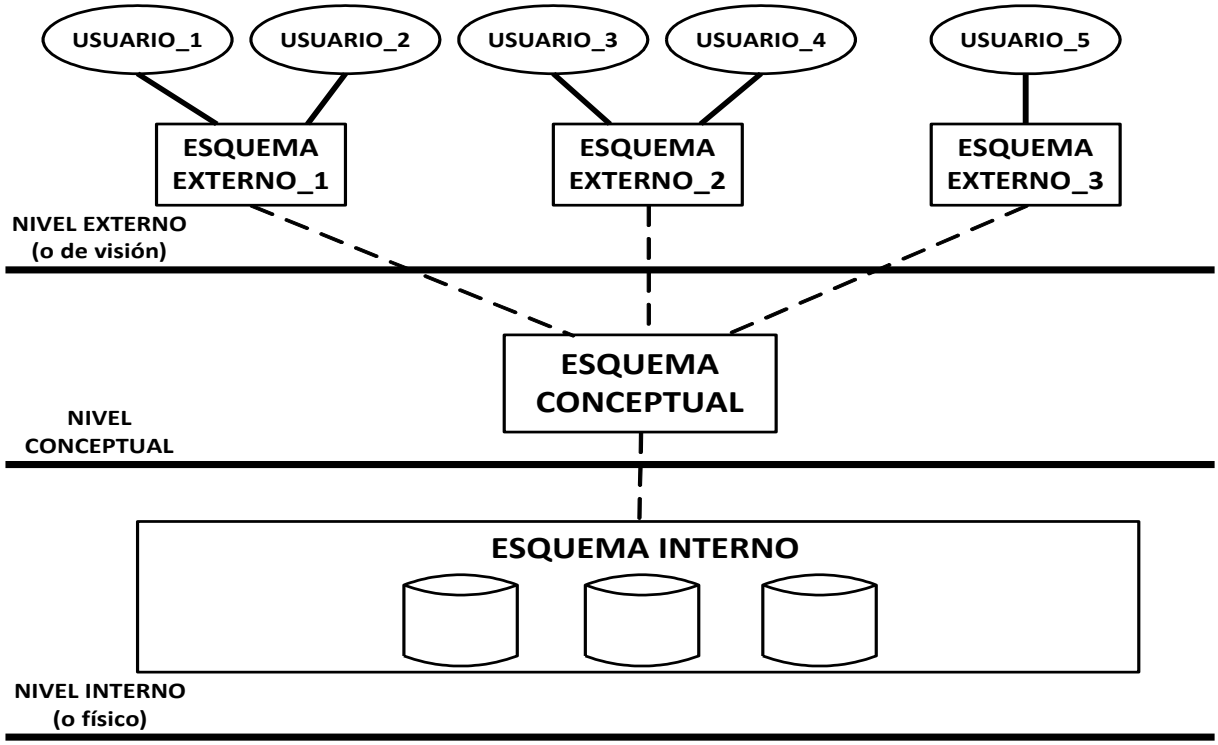
3.3.3. Nivel externo o de visión

Es el nivel más cercano a los usuarios. Representa la visión individual de un usuario o de un grupo de usuarios de la BD.

Estas **visiones particulares de los usuarios** nos las proporcionan los programas de aplicación, que sólo manejan una parte de la información de la BD, la que necesitan según el fin para el que han sido desarrollados.

Pueden existir, y es lo habitual, **más de un esquema externo** (más de una vista).

RESUMIENDO GRÁFICAMENTE:



3.4. SISTEMAS GESTORES DE BD (SGBD)

Un SGBD es un conjunto de programas que permiten administrar y gestionar la información de la BD.

Un SGBD tiene por **COMPONENTES principales**:

3.4.1. Gestor de la BD

Conjunto de programas transparentes al usuario que se encargan de garantizar la privacidad, seguridad, integridad de los datos y el acceso concurrente a ellos.

El gestor proporciona una **interfaz** entre:

- los datos almacenados
- los programas que manejan esos datos, y,
- los usuarios.

Cualquier operación que queramos realizar *contra* la BD la ha de realizar el gestor de la misma.

El gestor almacena en el **Diccionario de Datos** la descripción de la BD, los usuarios permitidos y las autorizaciones pertinentes.

Existe un usuario **administrador** encargado de centralizar las tareas.

3.4.2. Diccionario de Datos (DD)

Es una BD donde se almacena toda la descripción de la BD. Debe contener todo lo que cualquier usuario pueda querer saber sobre la BD:

- las descripciones externa, conceptual e interna de la BD
- las transformaciones entre los 3 niveles

- las restricciones sobre los datos
- el acceso a los datos
- las descripciones de las cuentas de usuario
- las autorizaciones de cada usuario
- los esquemas externos de cada programa...

3.4.3. Administrador de la BD (ABD ó DBA)

Persona (o grupo) responsable de la seguridad y del control de los datos. Tiene el control centralizado de los datos y de los programas que pueden acceder a ellos.

El administrador se sirve del gestor para sus tareas.

3.4.4. Los lenguajes

El SGBD ha de proporcionar lenguajes para definir y manipular los datos de la BD. Estos lenguajes los usan los administradores y los usuarios.

Los lenguajes PRINCIPALES son los siguientes:

- **Lenguaje de Definición de Datos (LDD ó DDL):** Para definir:
  - el esquema conceptual
  - los distintos esquemas externos

  Con este lenguaje **creamos los objetos** de la BD.
- **Lenguaje de Manipulación de Datos (LMD Ó DML):** Con este lenguaje podemos insertar datos, modificarlos, eliminarlos y recuperarlos.

  Con este lenguaje **actualizamos y consultamos** los datos de la BD.
- **Lenguaje de Control de Datos (LCD ó DCL):** Se usa para controlar el acceso a la información de la BD, definiendo privilegios y tipos de acceso. De esta tarea se encarga el DBA.

Los lenguajes de las BBDD tienen una gramática sencilla fácil de entender por usuarios no expertos.

No todos los lenguajes son iguales para todas las BBDD, pues dependen del **modelo de datos** que tenga el SGBD utilizado.

Los SGBD pueden procesar peticiones en LDD, LMD ó LCD formuladas en programas escritos en diversos lenguajes de programación como C, PHP, Java, Visual Basic .Net, Python...

3.5. MODELOS DE BASES DE DATOS

Un modelo de BD consiste en un **GRUPO DE HERRAMIENTAS CONCEPTUALES** (gráficos, normas...) para describir los datos, sus relaciones, su semántica y sus limitantes (restricciones).

Una vez que se ha realizado el **diseño conceptual** de la BD según el modelo conceptual elegido, el esquema resultante ha de traducirse a un modelo lógico de datos.

Los **3 modelos de datos** más difundidos son:

- el modelo de datos **jerárquico**
- el modelo de datos **en red**
- el modelo de datos **RELACIONAL** (el que nosotros estudiaremos con detenimiento)

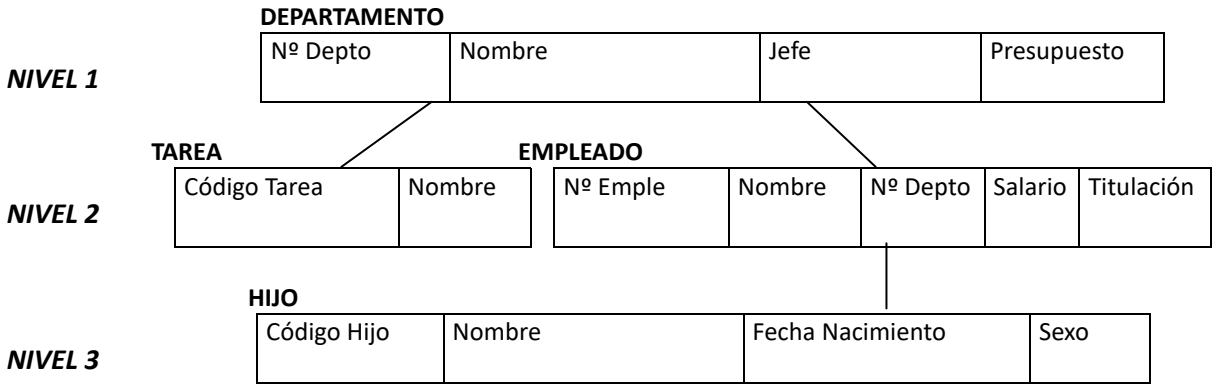
3.5.1. Modelo jerárquico

Se sirve de árboles para la representación de los datos. Un árbol está compuesto por una jerarquía de elementos llamados **nodos**.

El nivel más alto de la jerarquía tiene un único nodo denominado **raíz**.

Cada nodo representa un tipo de registro llamado **segmento** con sus correspondientes campos.

Ejemplo: Tenemos un **esquema jerárquico** que almacena la información referente a los DEPARTAMENTOS de una empresa:



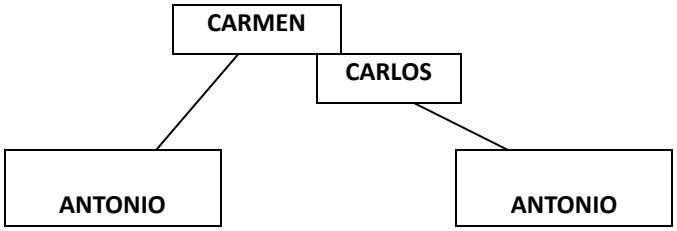
- En el primer nivel tenemos el **nodo raíz**, el s**egmento DEPARTAMENTO**, con la información de los departamentos
- En el segundo nivel, que depende del primero tenemos 2 segmentos:
  - El **segmento TAREA**, con información de las tareas que se hacen en los departamentos.
  - El **segmento EMPLEADO**, con información de los empleados que trabajan en cada departamento.

PROBLEMAS DEL MODELO JERÁRQUICO:

Como se puede observar en el ejemplo, el modelo jerárquico sólo permite establecer relaciones 1:N, es decir 1 a varios. Para el ejemplo 1 departamento puede tener asignadas varias tareas, pero una tarea concreta sólo puede ser realizada por un departamento.

- Generalizando para todos los casos, podemos decir que este modelo **NO PERMITE que 1 hijo tenga más de un padre**.

  Volviendo al ejemplo anterior, si en la misma empresa trabaja un matrimonio (Carlos y Carmen) y tienen 1 hijo en común (Antonio), deberíamos repetir la información del hijo en el modelo:



- Para acceder a cualquier segmento hay que partir del segmento raíz => *sistema lento*
- No se permite más de una relación entre segmentos.

Ejemplo: Si tenemos una BD jerárquica que almacena información de los MUNICIPIOS, el segmento con información de **personas** y el segmento con información de **viviendas**, deberían poder relacionarse de 2 maneras distintas:

- una indicando la relación de propiedad de la vivienda (quién posee la vivienda), y,
- otra indicando la relación de habitabilidad de la persona (quién habita la vivienda).

UTILIZANDO EL MODELO JERÁRQUICO NO SE PODRÍA ESTABLECER ESTA RELACIÓN

Dos de los **gestores de BD** más conocidos **que emplean el modelo de datos jerárquico** son: *IMS de IBM* y *System 2000 de Intel.*

3.5.2. Modelo en red

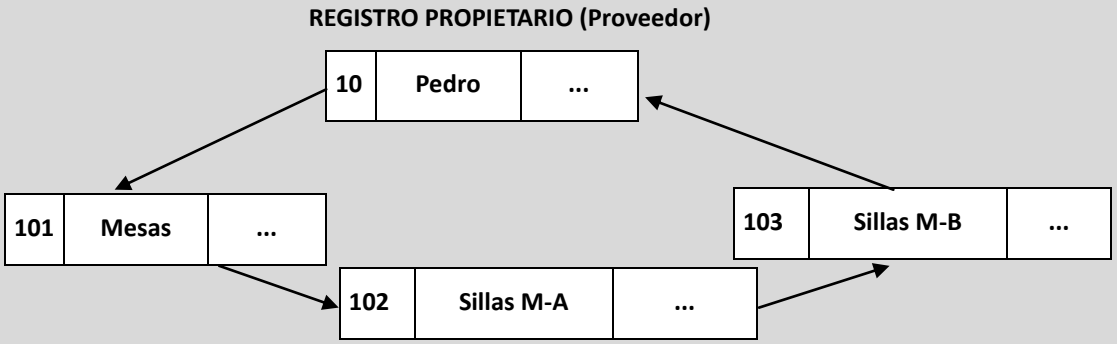
Las **entidades** del mundo real se representan como NODOS, y las **relaciones** entre ellas como LÍNEAS que unen los nodos.

En estos modelos cualquier componente puede vincularse con cualquier otro.

Se puede describir como el modelo jerárquico, con la misma terminologías de *padres e hijos*, pero ahora **un hijo puede tener varios padres**.

El modelo de red más conocido es el **CODASYL** (**C**onference on **Da**ta **Sy**stem **L**anguages).

Ejemplo: Supongamos una BD en red en la que se almacene información de los PROVEEDORES y de los ARTÍCULOS servidos por los mismos. Veamos un ejemplo de un caso particular:



El registro propietario apunta al primer registro miembro (el 101), éste al 2º(el 102), y así sucesivamente hasta llegar al último registro miembro que apunta al propietario.

La **representación interna** se realiza mediante *listas circulares doblemente enlazadas.*

Gestores de BD que utilizan el modelo de datos en red son:
DMS 1100 de UNIVAC, EDMS de Xerox y *PHOLAS de Philips*.

IMPORTANTE: Las representaciones lógicas basadas en árboles o estructuras en red, a menudo limitan el crecimiento de la BD. Este es un inconveniente importante de este modelo.

3.5.3. Modelo Relacional

Un **SGBD relacional** usa *tablas bidimensionales* para representar los datos y las relaciones entre ellos. Fue **Codd** quien desarrolló en IBM el modelo de datos relacional.

ALGUNAS VENTAJAS DE ESTE MODELO:

- ofrece una visión conceptual sencilla de los datos que existen en la BD
- puede entenderlo y usarlo cualquier usuario sin ser un experto informático
- los usuarios no necesitan saber dónde se encuentran físicamente los datos
- se puede ampliar el esquema conceptual sin modificar los programas de aplicación.

Podemos decir que **el elemento principal** de este modelo es la **RELACIÓN**, de ahí su nombre de modelo relacional. Cada relación se representa mediante una tabla.

Es el modelo de datos más empleado en los gestores de BD actuales.

Ejemplos de gestores de BD relacionales son:

- System R y QBE de IBM

- Oracle
- Informix
- DBase
- MySQL
- MariaDB
- Microsoft Access
- Microsoft SQL Server
- Postgres
- SQLlite...

3.6. ARQUITECTURA CLIENTE/SERVIDOR (C/S)

Las **distintas configuraciones** de la arquitectura Cliente/Servidor son:

3.6.1. Basada en anfitrión

La **máquina cliente y la servidora son la misma**. Los usuarios se conectan directamente a la máquina dónde se encuentra la BD.

3.6.2. Cliente/Servidor típica

La **BD está en el servidor** y los usuarios acceden a la BD desde su máquina cliente a través de la red.

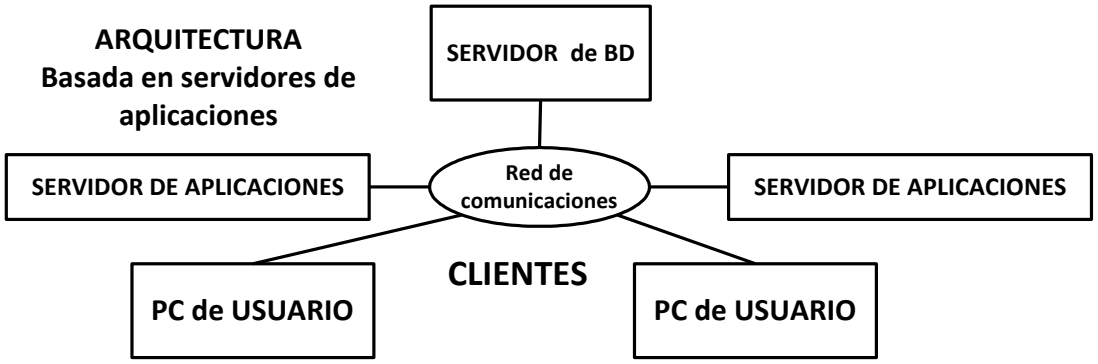
3.6.3. Procesamiento distribuido

La **BD está repartida en MÁS DE UNA máquina servidora**. Los usuarios no tienen porqué conocer la ubicación física de los datos y acceden a varios servidores a la vez.

3.6.4. Basada en servidores de aplicaciones

Esta configuración permite el uso de aplicaciones en redes de área extensa (WAN) e Internet. Hace posible que las aplicaciones se ejecuten en máquinas cliente que no requieren ninguna administración.

Cualquier PC que ejecute un navegador de Internet puede acceder a las aplicaciones.



DIFERENCIA ENTRE C/S Y ARQUITECTURA BASADA EN SERVIDORES DE APLICACIONES

- La arquitectura C/S requiere que las aplicaciones se instalen en cada cliente.
- En la arquitectura basada en servidores de aplicaciones, cualquier PC puede acceder a las aplicaciones simplemente con tener instalado un navegador web.

3.7. USUARIOS DE LA BD

Según el modo en que interactúan con el sistema se distinguen 3 tipos de usuarios:

- **Programadores de aplicaciones:** son **profesionales de la programación** que desarrollan las aplicaciones que actúan sobre los datos de la BD.
- **Usuarios casuales:** estos usuarios no escriben programas pero **sí realizan consultas a la BD** en el lenguaje de consultas apropiado para la BD.
- **Usuarios ingenuos:** son usuarios que no tiene porqué tener conocimientos de informática y que interactúan con el sistema llamando a alguno de los programas de aplicación escritos previamente.

4. OTROS SISTEMAS DE ALMACENAMIENTO

4.1. XML (eXtensible Markup Language = *lenguaje de marcas extensible*)

Los sistemas gestores de bases de datos son el software esencial en el que se apoyan la mayoría de las aplicaciones para el almacenamiento y acceso a sus datos.

XML es un modelo de organización de datos muy extendido para usos muy variados entre los que destacan la comunicación entre aplicaciones o la transformación y representación de los datos.

XML almacena los datos en una estructura de árbol con espacios de nombres para diferentes categorías de datos.

Ejemplo: documento XML

En el siguiente ejemplo, se muestran los nombres de tres usuarios en XML.

```
<users>
        <user>
                <firstName>John</firstName> <lastName>Doe</lastName>
        </user>
        <user>
                <firstName>María</firstName> <lastName>García</lastName>
        </user>
        <user>
                <firstName>Nikki</firstName> <lastName>Wolf</lastName>
        </user>
</users>
```

4.2. JSON (JavaScript Object Notation)

Es un formato ligero (poco pesado) de intercambio de datos. Su sintaxis es más compacta y fácil de escribir y leer que la del XML.

Ejemplo: documento JSON

En el siguiente ejemplo, se muestran los nombres de tres usuarios en XML.

```
{"users":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"María", "lastName":"García" },
    { "firstName":"Nikki", "lastName":"Wolf" }
]}
```

4.3. SERVICIO DE DIRECTORIO

Como ejemplos de directorios tenemos desde una guía de teléfonos hasta cualquier revista que contenga la programación televisiva.

No hay que confundir servicio de directorio con el **repositorio de directorio**, que es la base de datos que contiene la información sobre los objetos gestionados por el servicio de directorio.

Un **servicio de directorio** es una o varias aplicaciones que almacenan y organizan la información de los usuarios de una red, sobre recursos de red, y permite a los administradores gestionar el acceso de usuarios a los recursos sobre dicha red. Además, los servicios de directorio actúan como una capa de abstracción entre los usuarios y los recursos compartidos.

Ejemplos: Open LDAP, Microsoft Active Directory, Apple Open Directory…