# SISTEMAS DE ALMACENAMIENTO DE INFORMACIÓN.

## Introducción a los sistemas de información

Se denomina sistema de información, al conjunto de personas, procesos, datos, aplicaciones, redes y tecnología que soporta y optimiza el día a día de una organización, permitiendo la resolución de problemas y la toma de decisiones de sus directivos.

Este conjunto de componentes: recoge, procesa, almacena y distribuye la información necesaria para el funcionamiento del negocio y también provee de mecanismos de análisis para conocer deficiencias o posibilidades de mejora.

Ejemplo: En una empresa de paquetería, como UPS o MRW, un cliente deja un paquete con destino a una dirección. Una vez pagado, un operario, lo refleja en un ordenador, utilizando una aplicación, que asignan un código al paquete, le emite una etiqueta y le indica que hacer para que comience su viaje a la dirección de destino, por ejemplo “Bilbao 18:50”.

En esta primera interacción, tenemos personas, que, con ayuda de la tecnología, introducen datos, usando una aplicación en un equipo informático, que imprime una etiqueta y que calcula en que camión debe viajar para comenzar su viaje al destino, que puede ser muy lejano.

Cuando este paquete se descarga de ese primer camión, un operario leerá el código del paquete y el terminal le indicará, “Barcelona 23:35”, y después cuando se descargue, “Terraza 06:10”, y así hasta que llegue a la oficina destino.

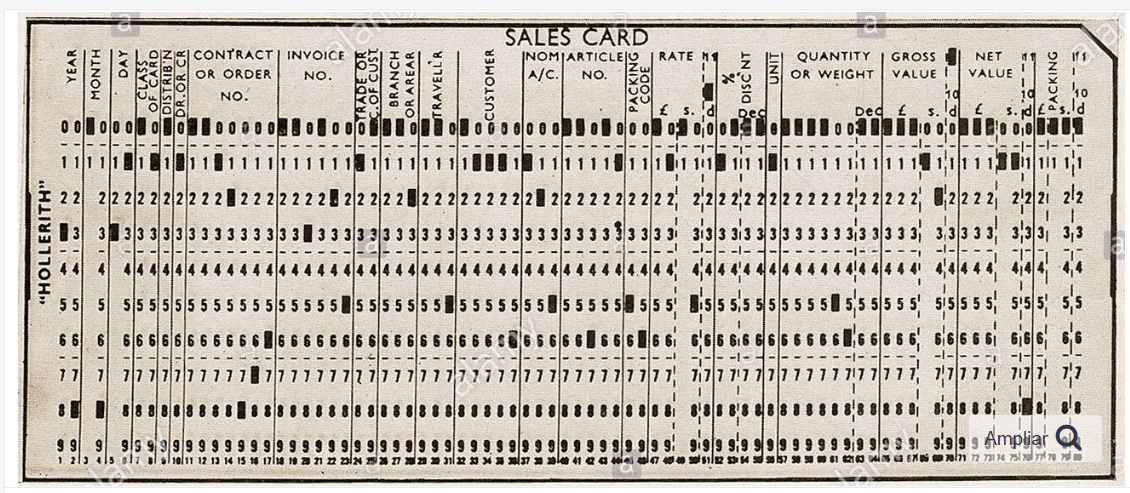
En todo sistema de información será habitual tener que almacenar datos y que estos estén accesibles desde distintos lugares geográficos para que el sistema de información permita desarrollar las actividades propias del modelo de negocio de forma óptima, solucionar los incidentes que se presenten y tomar decisiones para mejorar u optimizar el funcionamiento.

### Sistemas de información orientados al proceso: ficheros

Las bases de datos surgen en el momento en el que se constata que las necesidades de recogida, consulta y modificación de la información no pueden realizarse con la tecnología existente. Históricamente surgen con el fin de manipular la gran cantidad de datos necesarios para el proyecto de alunizaje del Apolo.

Sin embargo, ¿qué había antes y que queda de esa tecnología anterior a las bases de datos?

En un principio se utilizaron tarjetas perforadas para almacenar datos. La primera vez que se utilizaron fue para recopilar el censo estadounidense, que constitucionalmente tiene que realizarse cada 10 años. La población había crecido tanto que era imposible cumplir ese mandato constitucional. Con el uso de las tarjetas perforadas se completó el censo en un tiempo récord y el éxito hizo que comenzaran a utilizarse como medio de entrada de datos y almacenamiento para el análisis y procesos de datos.



De la tarjeta perforada se pasó a las cintas de papel perforado que no eran otra cosa que una sucesión de tarjetas perforadas. Estas evolucionaron a las cintas magnéticas, en las que las perforaciones se transformaron en señales magnéticas.



Normalmente la primera aplicación que se automatizaba en un negocio eran las nóminas debido a la complejidad de los cálculos. Se usaban un fichero maestro con los empleados, normalmente en cinta, donde se leía el primero y a continuación, se leían las tarjetas perforadas con sus horas trabajadas y circunstancias de productividad. Con eso el programa calculaba la nómina. Los resultados se grababan en una nueva cinta que sustituía a la maestra de empleados. Este proceso se hacía sin intervención humana y es el origen del procesamiento secuencial, por lotes, o “batch”.

A finales de los años 1950 aparece el disco magnético, que permite acceder a los registros sin leer los anteriores y actualizarlos. Las aplicaciones se volvieron más complejas, principalmente por el uso de las redes de comunicaciones y las grandes aplicaciones transaccionales, por lo que se necesitaban métodos de acceso y almacenamiento más sofisticados.

El uso de ficheros continua en nuestros días. Si las necesidades de acceso y almacenamiento de una aplicación no son sofisticadas, se utilizan los ficheros, que no son otra cosa que una sucesión de registros, como los que se encontraban en las antiguas cintas perforadas y magnéticas, a los que pueden accederse de forma directa a cada registro, sin tener que procesar todos los que tiene delante.

Cuando una empresa o departamento, utiliza ficheros en lugar de bases de datos, la información que almacena está **descentralizada**, el departamento almacena y gestiona sus propios datos mediante una serie de programas escritos especialmente para él y no comparte información ni aplicaciones con otros departamentos. Los datos comunes están duplicados en cada departamento.

### 2.1 Diseño lógico vs diseño físico

Una de los inconvenientes de esta forma de trabajar es que, como la estructura de los datos se encuentra especificada en los programas de aplicación, cualquier cambio en la estructura de los datos almacenados en el fichero afecta a las aplicaciones.

El programador debe identificar todas las partes de cada programa afectado por el cambio, modificarlos y volverlos a probar, lo que lleva mucho tiempo y está sujeto a que se produzcan errores. A este problema, se le denomina *falta de independencia de datos lógica-física.*

Ejemplo, incluimos un nuevo dato a los registros de un fichero:

José Martínez; C/Albéniz 21, 3º-4ª; 33024; Gijón;

Todos los programas que abren este fichero, deben tener en cuenta las características del nuevo registro.

Cuando se trabaja con una base de datos, se pueden modificar las estructuras donde se almacenan los datos (que ahora en lugar de ser ficheros se denomina tablas), para añadir, por ejemplo, un nuevo dato a cada registro, sin tener que modificar los programas de aplicación que no lo utilizan.

También se puede modificar el lugar físico en el que se almacenan los datos o la tabla en la que se almacena, sin tener que modificar las aplicaciones. Por ejemplo, decidir almacenar de forma separada la información de contacto de un empleado y sus salarios, o mover los datos a un nuevo directorio o unidad, sin tener que alterar los programas de aplicación.

### 2.2 Gestión de ficheros e interacción con ficheros.

Esta forma de trabajar, con **datos redundantes y duplicados** entre departamentos, que **desaprovecha el espacio de almacenamiento**, también origina a menudo **inconsistencias en los datos**, es decir, como la información está repetida en varios departamentos, puede haber información que no coincida. Por ejemplo: el domicilio de un mismo cliente no coincide, porque lo tienen en facturación y en ventas, pero sólo uno de ellos ha sido informado del cambio de domicilio y, o no le llegan las facturas o no le llegan los pedidos.

Además, la estructura física de los ficheros de datos (cómo se llaman o dónde están) y de sus registros (qué campos contiene cada registro), están definidas dentro de los programas que los utilizan, lo que hace que sea **difícil acceder a ellos cuando se usa más de uno**, pues debe sincronizarse su lectura y actualización para evitar que un usuario lea datos incorrectos mientras otro los actualiza, aumentando el tiempo de proceso.

Debido a este **aislamiento de la información**, cuando hay necesidad de transferir la información a un nuevo programa hay que desarrollar un software denominado de migración específico.

En resumen, hay una gran **falta de flexibilidad** originada por la dependencia total de la estructura física de los datos, es decir, donde están los datos, que ficheros hay y que registros tiene cada uno.

### 2.3 Tipos de ficheros (planos, indexados, acceso directo...)

Cuando se trabaja con ficheros nos encontramos con los siguientes tipos básicos:

**Texto plano:** Almacenan secuencias de caracteres con una codificación determinada (Por ejemplo, ASCII). Son legibles con un software de edición de textos como el *block de notas*. Ejemplos de ficheros de texto plano: .HTML, .XML, .txt o .csv.

**Binarios**: Contienen información codificada en binario por una aplicación, que es el la que entiende el formato de la información y puede modificarla. No son legibles con editores de texto. Ejemplos de ficheros binarios: aplicaciones (.exe), documentos (.pdf, .doc, docx), ficheros de imagen y audio (avi, mp3, mp4, jpg).

Los ficheros de texto plano también se pueden clasificar de acuerdo a su organización interna en:

**Secuenciales:** La información se almacena de forma contigua. Para acceder a un dato hay que recorrer todos los anteriores.

11342122#Alfonso Martínez Gonzalez#687732112$10876653#Margarita Sánchez Alvarez#654432212$11356678#Jose Martínez Alvarez#656774378!

**Aleatorios o de acceso directo:** Cada línea se organiza con unos tamaños fijos. Se puede acceder directamente al principio de cada línea. Al final se incluyen caracteres especiales de salto de línea. Los datos ocupan un numero de caracteres fijo, aunque no lo ocupen totalmente.

Como todos ocupan el mismo espacio, se puede acceder a cualquier registro, ya que se conoce el número de caracteres de cada registro.

11342122 Alfonso Martínez González 687732112  
10876653 Margarita Sánchez Álvarez 654432212  
11356678 José Martínez Álvarez 656774378

**Indexados:**

Se usan ficheros auxiliares en los que se almacena información sobre donde está cada registro del fichero aleatorio usando uno de los campos del fichero y almacenándolo de forma ordenada.

10876653 2  
11342122 1  
11356678 3

11342122 Alfonso Martínez González 687732112  
10876653 Margarita Sánchez Álvarez 654432212  
11356678 José Martínez Álvarez 656774378

## Sistemas de información orientados a los datos: bases de datos.

Las necesidades de gestionar cantidades relativamente grandes de información, sin duplicidades, sin inconsistencias, su uso en proyectos con necesidades interrelacionadas, pero distintas, que evolucionaban y cambian, dejan patente que hay que desarrollar un sistema que gestione los datos, de forma estructurada e independiente de las aplicaciones que van a utilizarlos.

Una base de datos es **un conjunto estructurado de datos que representan entidades y sus interrelaciones.** La representación es única e integrada, a pesar de que debe permitir utilizaciones varias y simultáneas.

### 3.1 Arquitectura de una base de datos: nivel interno, conceptual y externo.

Una base de datos trata de representar la realidad, mejor dicho, la pequeña parte de la realidad que le interesa a una empresa (su mini-mundo). Las empresas funcionan dentro de pequeños mundos (para una universidad, su mundo son los estudiantes, la asignatura, las aulas, etc. mientras que para un banco su mini-mundo es el dinero, los clientes, sus cuentas…).

Las empresas funcionan en ese “mundo” y toman decisiones observando ese mundo, por ello las BD deben ser capaces de representan esos “mini mundos”.

Las bases de datos se pueden ver con tres niveles de abstracción. El objetivo de usar esos tres niveles es separar lo que ve el usuario del cómo se ha implementado físicamente la base de datos.

Las ventajas de tener tres niveles es que:

* Permiten que los usuarios cambien con el tiempo cómo quieren ver los datos (por ejemplo, mostrar la fecha de nacimiento y después querer ver la edad).
* Los usuarios no tienen por qué tener que lidiar con aspectos como dónde se almacenan los datos.
* Los usuarios no tienen por qué conocer con tipos de datos o estructuras de almacenamiento de datos.
* Los cambios en los distintos modelos no tienen por qué interferir entre sí.

#### Nivel conceptual o lógico

Uno de los niveles de diseño de una base de datos es ese mini-mundo, también llamado **modelo conceptual**, o **segundo nivel** de definición de la base de datos.

Se definen entidades, que identifican por ejemplo personas, lugares u objetos de los que se recopilan datos (clientes, cuentas bancarias, asignaturas). En estas entidades se almacenará el conjunto de todos los clientes, cuentas, estudiantes o asignaturas con las que cuenta la organización o empresa.

Cada entidad tiene ciertos atributos, que son características para describir esas entidades y que la organización considera importantes. Por ejemplo, para un cliente, su nombre, DNI, dirección o para una cuenta bancaria su número de cuenta y su saldo.

Algunas entidades se relacionarán con otras, por ejemplo, la cuenta bancaria tendrá un propietario, que será un cliente. Un estudiante se relacionará con varias clases.

La base de datos se diseña para ser un modelo útil para la organización y para sus operaciones en el mini-mundo de su interés.

Los cambios en el mundo real se hacen en esas entidades, en sus atributos o en sus relaciones. Por ejemplo, un cambio de dirección de un cliente se hace en la entidad cliente, en su atributo dirección.

Para seguir la pista de los hechos en el mundo real (mini-mundo real) no es suficiente con entidades básicas que representan el mini-mundo sino también entidades que registren los cambios, por ejemplo, no es sólo necesaria un saldo de una cuenta bancaria, sino que es necesario también

#### Nivel externo

Como no todos los usuarios de una organización tienen las mismas necesidades, ese mini-mundo se debe de poder adaptar a las necesidades de los distintos usuarios o departamentos de la organización. El nivel conceptual tiene un nivel superior que es el nivel externo.

Este nivel es la forma con la que los usuarios conocen los datos y por ello puede haber tantos niveles externos como usuarios. El mini-mundo se representa de la forma que es adecuada para cada usuario.

El usuario interactúa sólo con ciertos aspectos del mini-mundo, está sólo interesado en ciertas entidades, quizás en sólo algunos de sus atributos o relaciones.

Si se crean nuevas entidades o se añaden nuevos atributos a entidades ya existentes o nuevas relaciones, el usuario no estará al tanto de ellas a no ser que sea él el que lo ha pedido.

A estas formas de ver el mini-mundo se le denominan vistas y suelen incluir un subconjunto de entidades, atributos (o campos calculados con ellos) y relaciones.

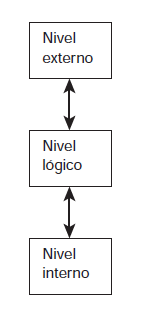
Además de ver la base de datos modelada para el usuario, es una barrera o frontera por debajo de la cual no se le deja ver.

#### Nivel interno

El nivel interno define la forma en la que se van a almacenar los datos. Cubre la implementación física de la base de datos. Y por debajo hay otro nivel físico gestionado por el sistema operativo bajo la dirección del gestor de la base de datos, que se encargará de colocar los datos en los dispositivos de almacenamiento. También es el encargado del desempeño, como la creación de índices.

Las razones para hacer esta separación

* Diferentes usuarios necesitas distintas vistas de los mismos datos.
* La forma en que un usuario ve los datos puede cambiar con el tiempo.
* Los usuarios no tienen por qué lidiar con las complejidades de las estructuras de almacenamiento.
* Se deben poder efectuar cambios en el modelo conceptual sin afectar a todos los usuarios.
* Se deben poder cambiar estructuras de datos sin afectar a las vistas de usuario.
* Se deben poder cambiar la distribución de los archivos o de los dispositivos de almacenamiento sin afectar al modelo conceptual ni a las vistas de usuario.







## Sistemas gestores de bases de datos

Además de la necesidad vista en el programa Apolo, de gestionar grandes cantidades de datos con relaciones complejas, la introducción de las líneas de comunicaciones, de terminales y discos duros, hace que surja la necesidad de que los programas permitan a varios usuarios consultar los mismos ficheros on-line y de forma simultánea. Mas adelante se unió la necesidad de tener que actualizarlos también. Las aplicaciones comienzan a integrarse (facturas con pedidos, pedidos con productos, etc..), los datos también y tuvo que eliminarse la redundancia, es decir que los datos que aparecían en ficheros de más de una aplicación deberían estar en un solo lugar.

Al principio con la integración, con hardware con un acceso rápido y con ficheros indexados fue suficiente. Hasta principios de los años 70 se les llamó Data Banks y luego Data Bases.

Pero el software de gestión de ficheros era demasiado elemental, por ejemplo, no permite que varios usuarios actualicen simultáneamente datos si no es delegando en la programación. La utilización de estos conjuntos de ficheros era excesivamente compleja, así que fue saliendo al mercado un software más sofisticado, lo que se denominaron sistemas de gestión de bases de datos (SGBD o en inglés Data Base Management Sistemas).

¿Que aportaba?

* El programador podía olvidarse de trabajar con ficheros, en el SGBD podía incluir en distintas entidades los datos que necesitase.
* El sistema tenía previstas las interrelaciones entre entidades, que no era posible trabajando con ficheros.
* Con un SGBD no era necesario crear ficheros a medida y controlar la redundancia en ellos, con un SGBD todas las aplicaciones trabajarían con él y la integración evitaría las redundancias.
* En un SGBD se podrían conectar múltiples operarios y cada uno podría tener su propia visión de los datos.

En los años 80, aparecen los ordenadores *minis* y después *micro*, que extienden la informática a todas las empresas e instituciones.

El desarrollo de aplicaciones necesitaba ser rápido y sencillo, no era realista que los SGBD fuesen tan complejos y necesiten personal tan cualificado.

Aparecen los SGBD relacionales (RDBMS), que suponen un gran avance al facilitar la programación de aplicaciones que usan BD a la vez que permiten que los programas sean independientes del hardware.

La estandarización del lenguaje SQL supone una explosión de uso de los SGBD relacionales.

### 4.2 Funciones

Los SGBD pretenden satisfacer las necesidades que han dejado patentes las aplicaciones que de forma on-line trabajaban directamente con sistemas de ficheros:

Realizar **consultas complejas** a los datos sin tener que escribir, compilar y ejecutar un programa específico.

Los usuarios podrán realizar consultas de cualquier tipo y complejidad directamente al SGBD y este tendrá que responder inmediatamente sin saber a priori que el usuario va a hacer esa consulta.

El usuario debería poder formular la consulta con un lenguaje sencillo, que el sistema debe interpretar directamente.

La solución fue el lenguaje SQL. Permite realizar consultas no predefinidas y complejas.

Ejemplo:

* Conocer el número de alumnos de menos de 21 años con una nota media superior a 7 que están matriculados en Bases de Datos de DAM.
* Nombres de los profesores y de los alumnos que no han superado dos o menos asignaturas.

El SGBD debe permitir **hacer cambios** (ser **flexible**) a medida que el sistema de información crezca o se modifique y los procesos no deben verse afectados por esos cambios tecnológicos o del sistema de información.

Para conseguirlo, los SGBD han implementado una separación por niveles en el que ni los usuarios que trabajan con el SGBD, ni los que diseñan el SI, necesitan saber nada sobre el sistema físico que soporta la BD. A esto se le denomina *independencia física* *de los datos.*

También que los usuarios no tengan que hacer cambios si se modifica el esquema o definición conceptual. Estos cambios no afectarán a la visión que los usuarios tienen de la BD. A esto se le denomina *independencia lógica de los datos.*

Deberá facilitar la **eliminación de redundancia**.

Con ello se evitará que el mismo dato deje de ser coherente al estar duplicado en varios sitios a la vez por olvidar modificar alguno de los lugares donde está repetido.

Si el SGBD permite datos redundantes, el mismo deberá actualizar los datos en los lugares donde estén repetidos. Sino el sistema acabará siendo incoherente.

Un tipo de redundancia peligrosa son los campos calculados. Estos campos nos permiten responder rápidamente a consultas globales, ya que ahorran procesar datos, pero hay que rehacerlos si se modifican los datos elementales. Conviene que el SGBD lo haga automáticamente.

Deberá garantizar la **integridad** de los datos.

El SGBD debe asegurar la calidad de los datos, si hay un error en un programa, si un operario incluye un dato erróneo, si hay una avería de un disco, si una transacción se queda a medias, si hay un corte de corriente, etc.

Si el SGBD detecta que un programa quiere hacer una operación que va en contra de las reglas definidas no lo deberá permitir. Por ello para diseñar las BD deberá permitir definir reglas de integridad de usuario además de las inherentes al modelo de datos.

Además, deberá proporcionar herramientas para casos de desastre, que permitan recuperar o restaurar datos estropeados. Los SGBD disponen de herramientas de copia de seguridad y de restauración, así como del mantenimiento de un diario donde anota lo que se va escribiendo en la BD.

Permitir **usuarios concurrentes.**

Tanto las empresas de líneas aéreas como los servidores web tienen simultáneamente miles de sesiones de usuario a la vez.

Cuando son todos de lectura el problema es únicamente de rendimiento (memoria, velocidad del disco, cachés, etc.).

Pero cuando uno actualiza datos, puede interferir en los demás de forma que se obtengan datos erróneos. Para facilitar los accesos concurrentes, la BD utiliza el concepto de transacción.

Una transacción es un conjunto de operaciones que se ejecutan como una unidad. O se ejecutan todas o no se ejecuta ninguna.

Ejemplo: Imaginemos un programa pensado para llevar a cabo la operación de transferencia de dinero de una cuenta X a otra Y. Supongamos que la transferencia efectúa dos operaciones: en primer lugar, el cargo a X y después, el abono a Y. Este programa se debe ejecutar de forma que se hagan las dos operaciones o ninguna, ya que si por cualquier razón (por ejemplo, por interrupción del flujo eléctrico) el programa ejecutase sólo el cargo de dinero a X sin abonarlos a Y, la BD quedaría en un estado incorrecto. Queremos que la ejecución de este programa sea tratada por el SGBD como una transacción de BD.

Para lograrlo, los SGBD utilizan la técnica denominada de bloqueo (lock). Los bloqueos ponen limitaciones a los accesos a esos datos.

**Seguridad**.

Los SGBD deben aportar confidencialidad a los datos, permitiendo autorizar a aquellas personas que tienen derechos de acceso a los datos y denegando su acceso a los que no los tienen.

Aunque esto sólo puede parecer importante a nivel militar o de inteligencia, en España hay leyes que exigen la protección de los datos personales.

Para lograrlo, los usuarios deben identificarse y autenticarse contra el SGBD, de forma que cada usuario tenga acceso sólo a la parte del modelo conceptual al que pueda tener acceso.

Por otra parte, los SGBD permiten encriptar tanto los datos físicos como algunos de los campos de la BD (por ejemplo: las contraseñas).

### 4.2 Componentes

Una vez se ha elegido e instalado un SGBD, el fabricante del software nos ofrecerá un conjunto de herramientas para almacenar y acceder de forma eficiente a los datos:

* Software que nos permitirá almacenar datos de forma eficiente, con todas las funciones enumerados en el punto anterior y no tener que utilizar directamente el sistema de ficheros del sistema operativo.
* Un lenguaje de programación que nos permitirá almacenar y acceder a los datos (crear, leer y modificar), como crear, consultar y modificar el diseño de la base de datos (metadatos), como mínimo por línea de comandos.
* Herramientas para gestionar las copias de seguridad, el mantenimiento de usuarios y sus permisos, programar tareas, etc.
* Un entorno gráfico para simplificar la gestión del SGBD como alternativa a la línea de comandos.

### 4.3 Usuarios y permisos.

Los SGBD permiten crear usuarios y darles permisos para leer sus vistas, que simplifican la visión que tiene del modelo conceptual de la BD. También para leer y escribir e incluso para poder consultar, actualizar o modificar cualquiera de las entidades de la BD.

Esos permisos se hacen de forma granular, es decir, indicando explícitamente que usuario puede hacer qué cosa.

Además, hay un tipo de usuario, muy importante, especial, que realiza tareas de administración y control de la BD. Una empresa o institución que tenga SI construidos en torno a BD necesita que alguien lleve a cabo una serie de funciones centralizadas de gestión y administración, para asegurar que la explotación de la BD es la correcta. Este conjunto de funciones se conoce con el nombre de administración de BD (ABD), y los usuarios que hacen este tipo especial de trabajo se denominan *administradores de BD*.

Entre sus atribuciones:

* Cambios en los esquemas.
* Asegurar la disponibilidad de los datos (con copias de seguridad o reconstruyendo la BD).
* Resolución de emergencias.
* Diseño físico (reestructurar el almacenamiento, fijar los caminos de acceso).
* Control del rendimiento (modificaciones en los esquemas para mejorarlo, en los parámetros del SGBD o en el SO).
* Control de la seguridad (autorizaciones / restricciones).

### 4.4 Lenguajes

Para comunicarse con el SGBD, el usuario, ya sea un programa de aplicación o un usuario directo, se vale de un lenguaje. Hay muchos lenguajes diferentes, según el tipo de usuarios para los que están pensados y el tipo de cosas que los usuarios deben poder expresar con ellos.

El lenguaje especializado en la escritura de los esquemas, es decir, en la descripción de la BD, se conoce genéricamente como DDL (*data definition language*).

El lenguaje especializado en la utilización de la BD (consultas y mantenimiento) se conoce como DML (*data management language*).

Lo más frecuente es que el mismo lenguaje disponga de construcciones para las dos funciones, DDL y DML.

Por ejemplo, en el lenguaje SQL, el más utilizado en la BD relacionales, se dispone de sentencias de ambos tipos:

1. Sentencias de tipo DML, como por ejemplo SELECT para hacer consultas o INSERT, UPDATE y DELETE para el mantenimiento de los datos.
2. Sentencias de tipo DDL, como por ejemplo CREATE TABLE, para definir las tablas, columnas y restricciones.

Los lenguajes DML pueden ser de dos tipos:

1. Declarativos: Se indica que se quiere hacer y no como.
2. Procedimentales: Se detalla paso a paso como se deben realizar las operaciones en la BD.

Los lenguajes pre-relacionales (Jerárquicos y en Red) eran procedimentales. SQL es declarativo.

Los lenguajes DDL son siempre declarativos.

Cuando se construye una aplicación, el lenguaje no suele tener instrucciones para el acceso a la BD. Se usas habitualmente llamadas a funciones (por ejemplo, ODBC o JDBC) que son las que envían la sentencia a ejecutar al SGBD.

### 4.5 Flujo de datos y control

Para entender el funcionamiento de un SGBD, vamos a enumerar los principales pasos para la ejecución de una consulta enviada a un SGBD por una aplicación:

1. La aplicación ejecuta una llamada al SGBD y le envía la consulta. El SGBD verifica la sintaxis de la consulta y que el usuario de la aplicación esté autorizado a hacerla. Utiliza el esquema externo y conceptual.
2. Si la consulta es válida, el SGBD determina usando el esquema interno la forma de responderla. La aplicación no indica nada sobre cómo obtener los datos, es el SGBD el que lo debe determinar. El resultado implicará recuperar datos de las páginas donde se encuentren los datos buscados.
3. Una vez que sabe las páginas, el SGBD comprueba si, por suerte, está en alguno de sus buffers y si no lo está, con la ayuda del S.O. la buscará en disco y la cargará en los buffers. Si ya está se ahorra el acceso a disco.
4. Una vez en memoria, extrae los registros buscados de entre los que están en la página, interpretando la codificación según indique el esquema interno.
5. El SGBD construye la respuesta, según el esquema externo sobre cómo deben mostrarse los datos al usuario.
6. Devuelve los datos al programa y da por terminada la ejecución de la consulta.

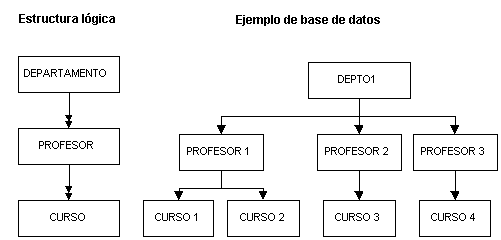
### 4.6 Tipos de SGBD

#### 4.6.1 Según el modelo lógico de datos: modelo jerárquico, de red, relacional, orientado a objetos.

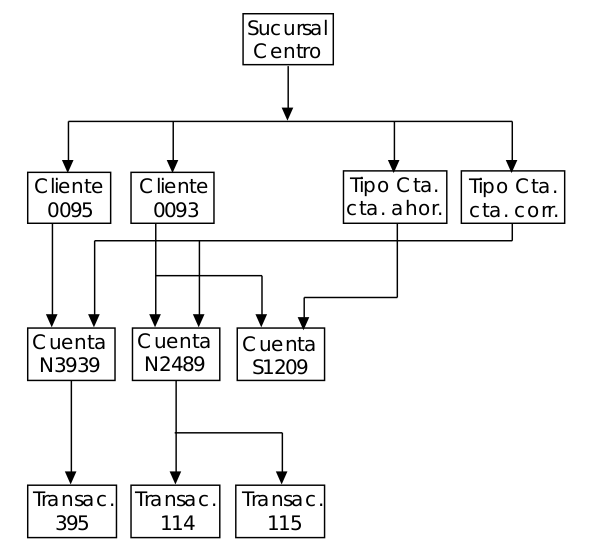
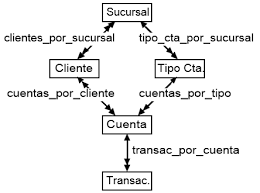
El *modelo de base de datos* es la arquitectura mediante la que se almacena e interrelaciona la información que se va a gestionar.

Los modelos más utilizados son:

**Jerárquico:** Refina la idea del fichero indexado, creando una estructura de jerarquía entre datos de varios ficheros en forma de árbol. Se usa en grandes máquinas (mainframes) y su implantación comercial más conocida es IMS/DL1 de IBM, que sigue en funcionamiento en sistemas de información muy antiguos muy grandes.

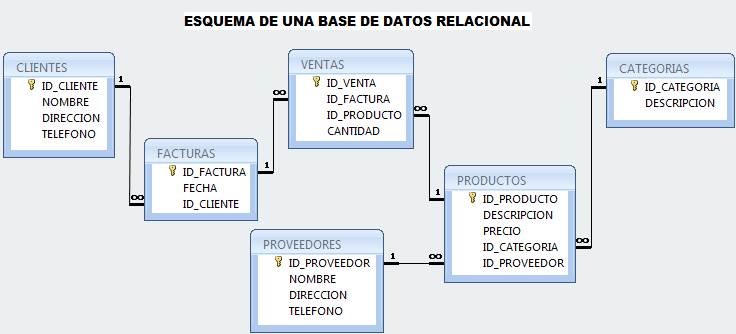


**Red:** Implementa mejoras al jerárquico como la flexibilidad a costa de aumentar la complejidad. Como en el jerárquico hay registros e interrelaciones, pero un registro no está limitado a ser “hijo” de un solo “padre”.



Ambos modelos hacían de la programación de aplicaciones fuese muy compleja.

**Relacional:** La información se representa con entidades y relaciones entre ellas, sin rutas para obtener los datos o jerarquías. Cada entidad e interrelaciones se representa en forma de tablas bidimensional (con filas y columnas). Es el modelo más extendido y utilizado hoy en día. Las más importantes son Oracle, IBM o Microsoft, que han evolucionado al tipo objeto-relacional, aunque aún se pueden encontrar relacionales puras como MySQL.



Surgen con la aparición del ordenador personal (primero mini y luego micro). Como todas las empresas se informatizan, exigen que el desarrollo de aplicaciones sea más sencillo. Ya que las BD en red y jerárquicas sólo las podía utilizar personal muy cualificado.

**Orientadas a objetos:** Los tipos de datos que se pueden definir en los sistemas relacionales son muy limitados. La necesidad de incorporar tecnologías multimedia (imagen y sonido) hace necesario que las bases de datos tengan que aceptar atributos de este tipo.

Otras aplicaciones no tienen suficiente con incorporar tipos especializados en multimedia. Necesitan tipos más complejos que el desarrollador pueda definir a medida, los objetos.

En las bases de datos orientados a objetos los datos se almacenan y representan en forma de objetos. Los objetos que se crean en un programa pueden almacenarse como se han creado y posteriormente leerlos de la base de datos.

Se usan en aplicaciones que requieren un alto rendimiento con muchos cálculos y que necesitan resultados rápidos, como sistemas en tiempo real, arquitectura y modelado 3D, telecomunicaciones o astronomía.

Algunas BD ODBMS pueden usarse en múltiples lenguajes. Por ejemplo, Gestione, permite enlazar con C++ y Java.

Las ODBMS no tienen un lenguaje de consultas estándar como el SQL de las bases de datos relacionales.

Otras populares con Cache y DB4o.

**Relacional con objetos:**

El rápido desarrollo de la web ha hecho que las BD puramente relacionales (RDBMS), se adapten también para ser usadas por servicios web, en los que se utilizarán lenguajes OO como C++ y Java con datos multimedia. Estas bases de datos ORDBMS (objeto-relacionales o relacionales con objetos) son bases de datos relacionales que soportan las características del modelo orientado a objetos como almacenamiento de objetos.

Ejemplos son PostgreSQL, Oracle, Sql Server, o DB2 de IBM.

#### 4.6.2 Según el número de sitios: centralizados, distribuidos.

Si utilizamos como criterio la ubicación física de la información existen bases de datos:

**Centralizadas**: La base de datos está en una sola máquina, el servidor de bases de datos

**Bases de datos distribuidas**: La información se reparte por distintos servidores, normalmente alejados físicamente.

### 4.7 Arquitectura cliente/servidor.

La tecnología que se utiliza habitualmente para distribuir datos es la que se conoce como entorno (o arquitectura) cliente/servidor (C/S). Todos los SGBD relacionales del mercado han sido adaptados a este entorno.

La idea del C/S es sencilla. Dos procesos diferentes, que se ejecutan en un mismo sistema o en sistemas separados, actúan de forma que uno tiene el papel de cliente o peticionario de un servicio, y el otro el de servidor o proveedor del servicio.

Por ejemplo, un programa de aplicación que un usuario ejecuta en su PC (que está conectado a una red) pide ciertos datos de una BD que reside en un equipo UNIX donde, a su vez, se ejecuta el SGBD relacional que la gestiona. El programa de aplicación es el cliente y el SGBD es el servidor.

Un proceso cliente puede pedir servicios a varios servidores y el cliente y el servidor pueden residir en un mismo sistema.

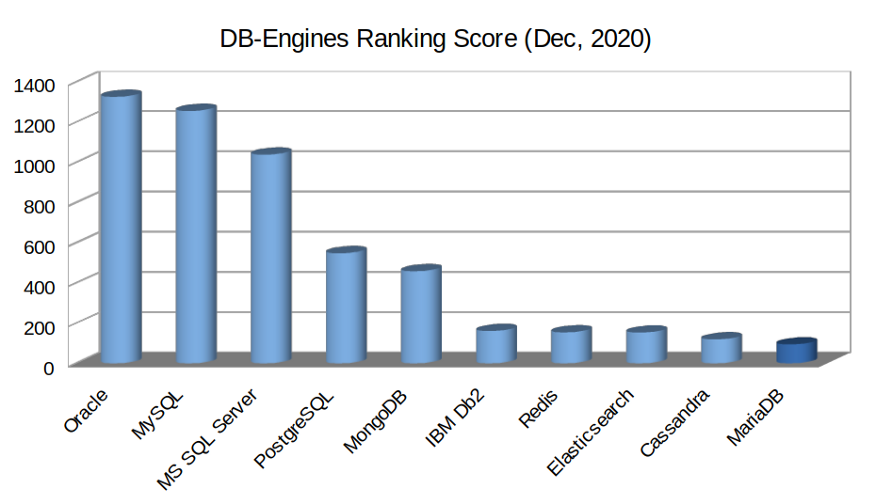


La facilidad para disponer de distribución de datos no es la única razón, ni siquiera la básica, del gran éxito de los entornos C/S en los años noventa. Tal vez el motivo fundamental ha sido la flexibilidad para construir y hacer crecer la configuración informática global de la empresa, así como de hacer modificaciones en ella, mediante *hardware* y *software* muy estándar y barato.

### 4.8 Sistemas gestores de bases de datos propietarios y libres.

Hace 20 años elegir qué base de datos utilizar era mucho más fácil que actualmente. Se elegía un RDBMS. Pero actualmente, elegir la base de datos es una tarea muy delicada, ya que a las bases de datos tradicionales SQL se han unido las NoSQL, las NoSQL y las bases de datos en la nube.

Sin embargo, el ranking de BD en 2021 sigue estando encabezado por BD relacionales. Entre ellas hay versiones open-zurce, gratuitas, y comerciales.



Entre los RDBMS comerciales (relacionales), Oracle destaca como líder seguido por MS SQL Server y por IBM DB2, usada en grandes organizaciones.

Entre las gratuitas, MySQL y PostgreSQL son líderes, mientras que MariaDB, compatible con MySQL, que se está comenzando a hacer popular.

Entre las NoSQL, MongoDB es la dominante.

**Oracle**: Es la BD número uno de las bases de datos con soporte comercial. Dispone de versiones para Windows y para la variante de Linux.

**Microsoft SQL Server**: Es una de las principales BD comerciales con soporte de Microsoft. De uso habitual si se desarrolla con productos Microsoft.

**IBM Db2**: Una de las más utilizadas en empresas de tamaño medio y grande. Dispone de versiones para Windows y Unix/Linux.

**MySQL**: Es una de las más populares y usadas RDBMS, especialmente en aplicaciones WEB (Facebook y Uber la usan). Dispone de licencias gratuitas y comerciales. Fue adquirida por Oracle.

**PostgreSQL**: Su origen está en la Universidad de Berkeley y en su BD Ingres. Es una de las más usadas y avanzadas base de datos de código abierto.

**MongoDB**: Es la DBMS NoSQL líder. No es relacional.

**Maread**: Es compatible con MySQL. Un servidor MySQL puede remplazarse por uno con MariaDB.

### 4.9 Bases de datos centralizadas y bases de datos distribuidas:

### Arquitectura. Ventajas y desventajas.

A finales de los 80, los SGBD relacionales se utilizan prácticamente en todas las empresas, excepto cuando se necesita un rendimiento elevado, en cuyo caso se opta por SGBD pre-relacionales (jerárquicos o en red)

Las empresas se encuentran con departamentos que han ido comprando ordenadores departamentales y personales y han ido haciendo aplicaciones con BD. El resultado múltiples BD de distintos tipos y proveedores. A esto hay que unir las fusiones de empresas.

La necesidad de tener una visión global de la empresa y de interrelacionar

diferentes aplicaciones que utilizan BD diferentes, junto con la facilidad que dan las redes para intercomunicar ordenadores, ha conducido a los SGBD actuales, a permitir que un programa pueda trabajar con diferentes BD como si se tratase de una sola. Es lo que se conoce como base de datos distribuida



Esta distribución ideal se consigue cuando las diferentes BD son soportados por una misma marca de SGBD. No es tan sencillo si los SGBD son heterogéneos. En la actualidad, gracias principalmente a la estandarización del lenguaje SQL, los SGBD de marcas diferentes pueden darse servicio unos a otros y colaborar para dar servicio a un programa de aplicación. No obstante, en general, en los casos de heterogeneidad no se llega a poder dar en el programa que los utiliza la apariencia de que se trata de una único BD.

Además de esta distribución “impuesta”, de querer acceder de forma integrada distintas BD preexistentes, también se puede hacer una distribución “deseada”, diseñando una BD distribuido físicamente, y con ciertas partes replicadas en diferentes sistemas. Las razones:

1) Disponibilidad. La disponibilidad de un sistema con una BD distribuida es más alta, porque si queda fuera de servicio uno de los sistemas, los demás seguirán funcionando y sus datos continuarán estando disponibles.

2) Coste. Una BD distribuida puede reducir costes. En el caso de un sistema centralizado, todos los equipos usuarios, que pueden estar distribuidos por distintas y lejanas áreas geográficas, están conectados al sistema central por medio de líneas de comunicación. El coste total de las comunicaciones se puede reducir haciendo que un usuario tenga más cerca los datos que utiliza con mayor frecuencia; por ejemplo, en un ordenador de su propia oficina o, incluso, en su ordenador personal.