TEMA 1. Introducción a las bases de datos.

1. Introducción. Evolución de la tecnología de bases de datos

La evolución a lo largo de varias décadas en el procesamiento de los datos y en la gestión de la información ha dado lugar a la sofisticación de la tecnología moderna de las bases de datos. En los años 50, los primeros sistemas de procesamiento de datos ejecutaron las tareas administrativas para reducir el papeleo. Más recientemente, los sistemas se han expandido hacia la producción y la gestión de la información. Actualmente, la función más importante de los sistemas de bases de datos consiste en proporcionar el fundamento a los sistemas de información para la gestión corporativa. Veamos un poco de su historia.

Inicialmente, los sistemas computacionales se utilizaron en los negocios para funciones de contabilidad y se intento imitar los procedimientos de papel de tal forma que los archivos del computador se correspondían con los archivos de papel y los registros en los archivos del computador contenían la información que podía almacenar una carpeta individual de un archivo manual. A estos sistemas se les llamaron *sistemas de procesamiento de datos*, debido a que ejecutaban las funciones habituales de tratamiento de registros.

En los años 60 el acceso era secuencial y, dado que el almacenamiento en disco era caro, se almacenaba en cinta magnética. Como los datos eran utilizados por diferentes aplicaciones, se debía reorganizar la información, reordenando los datos según un identificador y fusionando después la información. Normalmente se procesaban los datos en bloques, es decir, todos los registros de un archivo se procesaban al mismo tiempo, normalmente por la noche al cerrar el negocio.

Los sistemas de gestión de archivos puramente secuenciales eran eficaces cuando se trataba de producir informes una o dos veces al mes, pero para muchas tareas rutinarias no era suficiente, se necesitaba un acceso directo a los datos, es decir, la capacidad de acceder y procesar directamente un registro dado sin ordenar primero el archivo o leer los registros en secuencia.

Los problemas de redundancia en la introducción de datos, y por tanto de mayor probabilidad de error, así como la reorganización de la información dado que el acceso era secuencial, fueron resueltos parcialmente con la introducción de los *archivos de acceso directo* y, particularmente, de los archivos secuenciales indexados (ISAM - Indexed Sequential Access Method) que se utilizaron ampliamente en los años 70.

A diferencia de los de acceso secuencial, los archivos de acceso directo permiten la recuperación de los registros aleatoriamente, por lo que pueden recuperarse inmediatamente. Los archivos ISAM son los archivos más utilizados en procesos de tipo comercial. Estos archivos permiten que uno o más campos de datos - llamados conjuntamente **clave** - se utilicen precisamente para indicar qué registro se recuperará. Este potente y práctico método dotó de gran flexibilidad a las aplicaciones comerciales pero esto sólo fue una solución parcial. Para lograr una solución más completa a estos problemas, fue necesario introducir los sistemas de gestión de bases de datos.

A finales de los años 60 y principios de los 70 se dio la transición de que, los sistemas computacionales comerciales pasaron del procesamiento de los datos al procesamiento de la información. La información era mucho más que simples registros relacionados con el negocio. A finales de los años 60 esto condujo a una fuerte demanda de *sistemas de información para la gestión*. Estos sistemas utilizarán los datos ya existentes en el computador para dar respuesta a un amplio espectro de preguntas de gestión o administración.

En este contexto se hace una distinción entre datos e información. Los datos pueden considerarse como hechos aislados y la información corresponde a los datos procesados, es decir, organizados o resumidos.

Por tanto, podríamos decir que una **base de datos** es una colección de elementos de datos interrelacionados que pueden utilizarse (o ser procesados) por uno o más programas de aplicación, y que un **sistema de bases de datos** es aquel sistema que está formado por una base de datos, por un sistema computacional de propósito general llamado sistema de gestión de bases de datos (SGBD) que manipula la base de datos, así como por el hardware y el personal apropiados. Un sistema de bases de datos bien diseñado, cuenta con funciones que facilitan la manipulación de la información (inserción, borrado, modificación de registros….) de tal forma que transforma los datos puros en información.

2. Sistemas de Gestión de Archivos

Los sistemas de información tradicionales almacenan la información en ficheros que son tratados por los sistemas de gestión de archivos, es decir, por un programa o conjunto de programas que se encargan de gestionar un conjunto de archivos de datos.

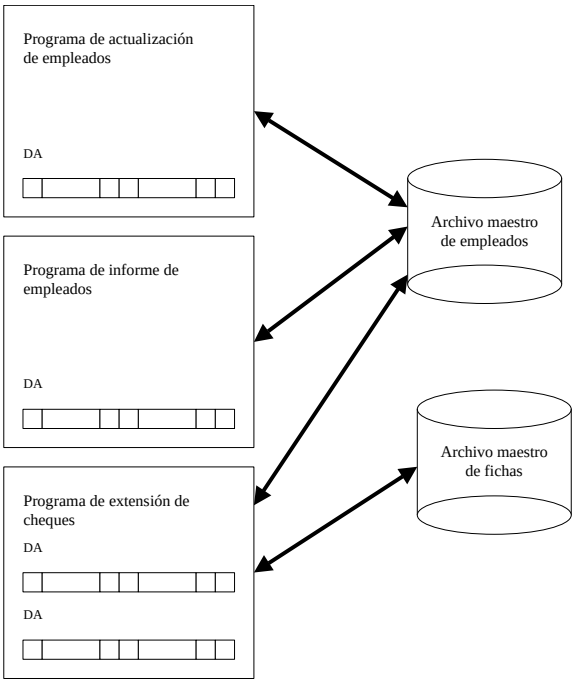
En una aplicación convencional con archivos, éstos se diseñan siguiendo las instrucciones de los correspondientes programas. Esto es, se decide si debe existir ó no archivos, cuántos deben ser, qué organización contendrá cada uno, qué programas actuarán sobre ellos y cómo lo harán. Esto tiene la ventaja, en principio, de que los programas son bastante eficientes, ya que la estructura de un archivo está pensada “para el programa” que lo va a usar. Sin embargo, esta forma de actuar está llena de graves inconvenientes. Por un lado, los programas que se realizan con posterioridad a la creación de un archivo pueden ser muy lentos, al tener que usar una organización pensada y creada “a la medida” de otro programa previo. Por otra parte, si se toma la decisión de crear nuevos archivos para los programas que se han de realizar, se puede entrar en un proceso de degeneración de la aplicación, ya que:

- Gran parte de la información aparecerá duplicada en más de un archivo (redundancia) ocupando la aplicación más espacio del necesario.

- Al existir la misma información en varios archivos, los procesos de actualización se complican de forma innecesaria, dando lugar a una propagación de errores.

- Se corre el riesgo de tener datos incongruentes entre los distintos archivos. Por ejemplo, para el caso de una empresa cuyo sistema contiene información de los empleados, podría darse el caso de que hubiera dos domicilios diferentes del mismo individuo en dos archivos distintos (por estar uno actualizado y el otro no).

En estas aplicaciones convencionales con archivos, el conocimiento a cerca del contenido de un archivo (qué datos contiene y cómo están organizados) está incorporado a los programas de aplicación que utilizan el archivo. Como ejemplo de utilización de un sistema de gestión de archivos se puede ver, en la figura de la derecha, una aplicación de nóminas de una empresa donde cada uno de los programas que procesan el archivo maestro de empleados contienen una *descripción de archivo* (DA) que describe la composición de los datos del archivo. Si la estructura de los datos cambiaba, todos los programas que accedían al archivo tenían que ser modificados. Como el número de archivos y programas crecía con el tiempo, todo el esfuerzo de un departamento se perdía en mantener aplicaciones existentes en lugar de desarrollar otras nuevas.



Resumiendo, podemos decir que las deficiencias que sufren los sistemas de gestión de archivos son:

- Redundancia e inconsistencia de datos. Muchas aplicaciones utilizaban sus propios archivos, entonces, además de que podían existir campos repetidos lo cual obligaba a introducir varias veces los datos o a que no se actualizaran hasta tiempo después, estos podían tener distintas longitudes en los diferentes ficheros, lo cual hacía inconsistentes los datos en las diferentes versiones. Es decir, esta redundancia conduce a un almacenamiento y coste de acceso a los datos más alto.

- Pobre control de los datos. En los sistemas de archivos no había un control centralizado a nivel de los datos, dado que el mismo elemento podía tener varios nombres dependiendo del archivo en que estuviera contenido. Esto daba lugar a confusiones debido a los homónimos (un mismo término que tiene diferentes significados en diferentes contextos) y sinónimos (términos diferentes que significan lo mismo), cosa que se evitaba, como iremos viendo a lo largo de este tema, con el sistema de bases de datos.

- Capacidades inadecuadas de manipulación de los datos. Los archivos secuenciales indexados (ISAM) permitieron que las aplicaciones tuvieran acceso a un registro particular mediante una clave, pero esto fue suficiente mientras solamente se quiso un registro único. El problema vino cuando se quería obtener relaciones más fuertes entre los datos contenidos en diferentes archivos, ya que estos sistemas son incapaces de proporcionarlas. Para ello se desarrollaron los sistemas de bases de datos, para facilitar la interrelación entre los datos en archivos diferentes.

- Esfuerzo excesivo de programación. Un nuevo programa de aplicación requería con frecuencia un conjunto completamente nuevo de definiciones de los archivos (aunque alguno ya existiera, seguro que habían muchos más por definir para que hubiera una consistencia en los datos). Entonces el programador tenía que

recodificar todas las definiciones de los elementos de los datos necesarios ya existentes, así como codificar todos los elementos nuevos. Existía una interdependencia muy fuerte entre los programas y los datos que daba lugar a dos problemas; Dificultad de acceso a los datos (cada vez que se quería obtener un conjunto de datos del sistema con unas características determinadas había que programar) y un aislamiento de los datos (dado que los formatos de los datos podían ser diferentes en cada archivo, dando problemas de integridad y además, cada vez que se introducían nuevas restricciones a estos datos había que reprogramar). Como veremos en el apartado siguiente, las bases de datos brindan una separación entre el programa y los datos, de modo que los programas pueden ser, en cierta medida, independientes de los detalles de definición de los datos. Al garantizar un acceso a un fondo común de datos compartidos y al soportar lenguajes poderosos para la manipulación de los datos, los sistemas de bases de datos eliminan una gran cantidad de programación inicial y de mantenimiento.

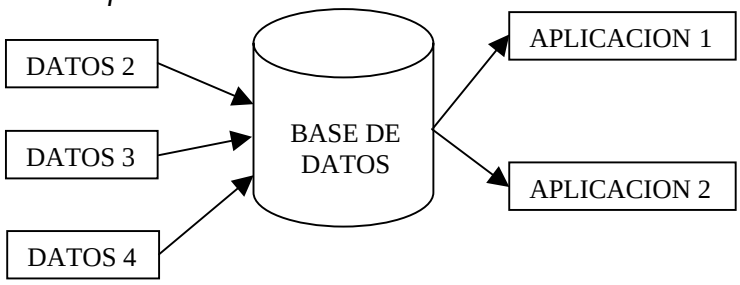
Por tanto, los problemas de mantener grandes sistemas basados en archivos condujeron, a finales de los sesenta, al desarrollo de los sistemas de bases de datos. La idea detrás de estos sistemas es sencilla: tomar la definición de los contenidos de un archivo y la estructura de los programas individuales, y almacenarla, junto con los datos, en una base de datos. Utilizando la información de la base de datos, el sistema gestor de la base de datos que la controla puede tomar un papel mucho más activo en la gestión de los datos y en los cambios a la estructura de la base de datos.

3. Sistemas de Bases de Datos. SGBD.

Como ya hemos dicho, los SBD surgen como alternativa a los sistemas de archivos, intentando eliminar o al menos reducir sus inconvenientes. Desde el punto de vista lógico (programas y usuarios), los datos y la definición de sus relaciones se almacenan en un único lugar, que es común. Físicamente, los datos se almacenan en uno o varios ficheros. El acceso de los datos se realiza, a través del sistema de gestión de bases de datos (SGBD o DBMS, Data Base Management System en inglés), mediante sentencias específicas que pueden incluirse dentro de lenguajes de alto nivel. Con esto, podemos definir un sistema de base de datos de la siguiente forma:

Un sistema de base de datos es un sistema formado por un conjunto de datos y un paquete software para la gestión del mismo, de tal modo que se controla el almacenamiento de datos redundantes, los datos resultan independientes de los programas que los usan, se almacenan las relaciones entre los datos junto con éstos y se puede acceder a los datos de diversas formas.

El esquema de funcionalidad de un SBD, se podría representar de la siguiente forma:



Los requisitos que debe cumplir un buen sistema de base de datos son:

- *Acceso múltiple.* Diversos usuarios pueden acceder a la base de datos, sin que se produzcan conflictos ni visiones incoherentes.

- *Utilización múltiple.* Cada usuario podrá tener una imagen o visión particular de la estructura de la base de datos.

- *Flexibilidad.* Se podrán usar distintos métodos de acceso, con tiempos de respuesta razonablemente pequeños.

- *Confidencialidad y seguridad.* Se controlará el acceso a los datos (incluso a nivel de campo), impidiéndoselo a los usuarios no autorizados.

- *Protección contra fallos.* Deben existir mecanismos concretos de recuperación en caso de fallo de la computadora.

- *Independencia física.* Se puede cambiar el soporte físico de la base de datos sin que esto repercuta en la base de datos ni en los programas que la usan.

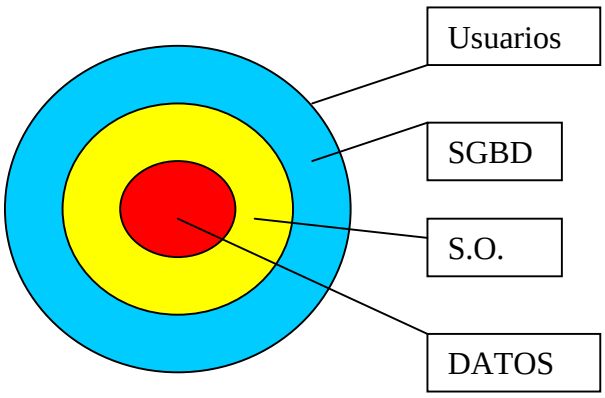
- *Independencia lógica.* Capacidad para que se puedan modificar los datos contenidos en la base, las relaciones existentes entre ellos o incluir nuevos datos, sin afectar a los programas que los usan.

- *Redundancia controlada.* Los datos se almacenan una sola vez.

- *Interfaz de alto nivel.* Existe una forma sencilla y cómoda de utilizar la base, al menos se cuenta con un lenguaje de programación de alto nivel, que facilita la tarea.

- *Interrogación directa ("query").* Existen facilidades para que se pueda tener acceso a los datos de forma conversacional.

Resumiendo, tenemos que: Las bases de datos permiten el almacenamiento, gestión y aprovechamiento de grandes volúmenes de información archivados durante periodos largos de tiempo en una computadora. Por ejemplo, en una empresa permiten crear, borrar, actualizar, recuperar relacionar y procesar nóminas pedidos, albaranes, facturas, etc. Y, el software que permite manejar la información almacenada en la base de datos es el Sistema de Gestión de Base de Datos (SGBD).



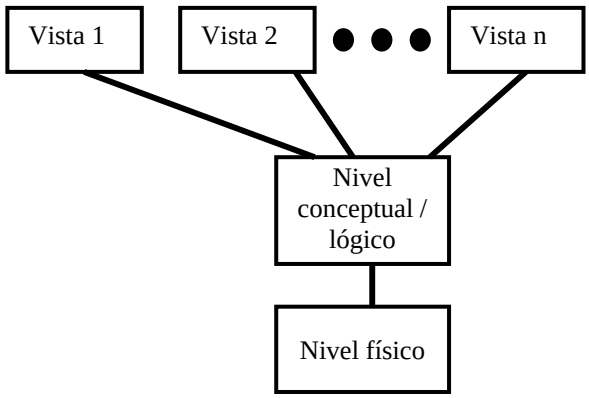
3.1. El SGBD

Ya hemos comentado que la base de datos es un almacenamiento persistente de información, es decir, la información supera en el tiempo a las operaciones de tratamiento y programas que se realizan sobre ella. Por ello, debe existir una independencia con los programas de datos que podemos definirla como la independencia de la representación de la información respecto a las aplicaciones que la

utilizan. De esta forma, se consigue una representación conveniente para todos los usos posibles de los datos y la estandarización de procedimientos El fundamento de esta independencia con los programas de datos radica, como ya veremos, en almacenar en la BD tanto los datos como las relaciones entre ellos y además, utilizar como unidad de almacenamiento el campo y el registro. Con esto, distintas aplicaciones podrán obtener las distintas 'vistas' de los datos que necesitan así como que será posible modificar la estructura de almacenamiento de la información sin afectar a las aplicaciones que los utilizan.

Por otra parte, los usuarios que normalmente utilizan la BD no necesitan conocer todo este nivel de complejidad de los datos para trabajar con ellos. Por ejemplo, para el caso de un usuario inexperto, el saber cómo han sido almacenados en memoria los datos es algo que no le interesa pero sí le interesa recuperar aquella información de su interés. Entonces, para poder representar el nivel de complejidad en el que estamos trabajando, vamos a definirnos tres niveles de abstracción de los datos de forma que se simplifique la interacción con el usuario. Estos tres niveles son:

- Nivel físico: se describe cómo se almacenan los datos en la memoria, es decir, las estructuras de datos complejas de bajo nivel.
- Nivel lógico y conceptual: describe qué datos son almacenados en la BD y las relaciones que existen entre ellos. Es decir, el pequeño número de estructuras simples que describen la BD. Usado por los administradores.
- Nivel de vistas: da una visión lógica de los datos relacionados con una aplicación dado que no todas las aplicaciones estarán interesadas en los mismos datos almacenados en la BD.



Anteriormente se ha comentado cuales eran las características que debía cumplir un buen SBD y, por otra parte, se ha dicho que el SGBD era el software que permitía manejar la información almacenada en la BD. Por lo tanto, las tareas que debe desempeñar el SGBD son:

- Ocultar al usuario los detalles del almacenamiento de la información, mostrando una visión 'abstracta' de esta.
- Permitir la integración de distintos tipos de información y permitir compartirlos entre distintas aplicaciones y usuarios.
- Encargarse de garantizar la seguridad de la información, controlando el acceso a la misma. El SGBD controla la integridad de la información comprobando la consistencia de la misma cuando se realizan operaciones de inserción, modificación o borrado.
- Organizar el acceso concurrente a la información por parte de distintas aplicaciones y usuarios, eliminando la posibilidad de interferencias o conflictos entre diferentes acciones.

3.2. El Administrador de la BD y el Administrador de los Datos

El Administrador de la Base de Datos es la persona encargada de la operación del sistema, y es el responsable de decidir:

- Los datos que se deben almacenar en la base de datos.
- La política de mantenimiento, tratamiento de los datos y seguridad de la información.

El administrador de la BD suele ser un informático y es el que decide la mejor forma de desarrollar las directivas del administrador de datos, organizando la administración del sistema y la operación de los usuarios.

El Administrador de los Datos es una persona relacionada con las actividades de gestión y dirección en la empresa que conoce a fondo los flujos de información dentro de la empresa y las necesidades de utilización de la misma por cada departamento.

3.3. Lenguajes de la Base de Datos

La interacción con la BD se realiza a través de lo que se llaman lenguajes. En las BD tendremos dos tipos de lenguajes; un lenguaje para especificar el esquema de la BD, que es lo que se conoce como **Lenguaje de Definición de Datos** (**DDL**), y otro para poder expresar consultas y actualizaciones de la BD, conocido como **Lenguaje de Manipulación de Datos** (**DML**). Estos lenguajes nos permiten realizar operaciones interactivas o diferidas sobre la base de datos.

El DDL es el lenguaje que contiene el conjunto de definiciones o instrucciones que permiten definir el diseño o esquema completo de la base de datos.

El DML es el lenguaje que permite a los usuarios acceder o manipular los datos, donde manipular significa tanto recuperar información, como insertar nueva información, borrarla o modificarla. Existen básicamente dos tipos de lenguajes de manipulación; los procedimentales y los no procedimentales.

Los **lenguajes de manipulación de datos procedimentales** son los que especifican qué datos queremos y cómo obtenerlos.

Los **lenguajes de manipulación de datos no procedimentales** son aquellos en que el usuario especifica qué datos desea obtener pero no cómo obtenerlos. Son los más fáciles de aprender pero pueden ser menos eficientes.

Hemos dicho que una de las funciones de los DML es recuperar información y esto se hace a través de una consulta. Podemos definir *consulta* como una instrucción de solicitud para recuperar información y por tanto, esto puede expresarse en un lenguaje que llamaremos *Lenguaje de Consulta*. Debemos anotar que, en muchas ocasiones, se suele usar la expresión Lenguaje de Consulta como sinónimo de Lenguaje de Manipulación de Datos pese a que esto sea incorrecto.

Como avance de lo que veremos posteriormente podemos decir que, el SQL (Structured Query Language) es un lenguaje combinado de manipulación y definición de datos, y es

el estándar más utilizado en las bases de datos relacionales. Por ejemplo, algunas sentencias en SQL son:

```
SELECT * FROM MI_TABLA;
INSERT INTO MI_TABLA VALUES ('CLAVE1',12124);
DELETE FROM MI_TABLA;
UPDATE MI_TABLA SET MI_CAMPO='CLAVE2' WHERE MI_CAMPO='CLAVE1';
```

Tanto al SGBD como a la base de datos se le pueden introducir las instrucciones interactivamente por el operador o incorporarse a programas de aplicación escritos en cualquier lenguaje de propósito general (C, Pascal, Basic, …). En SQL, este modo de programación se denomina SQL embebido (embed SQL).

4. Modelos de Datos

En la actualidad estamos inmersos en varias décadas de largo esfuerzo por desarrollar sistemas de gestión de bases de datos cada vez más poderosos. Este proceso ha sido testigo del desarrollo evolutivo de los sistemas basados en los **modelos de datos**, y que no son más que métodos conceptuales para estructurar los datos, es decir, que nos permiten describir los datos y las relaciones entre ellos. En este apartado veremos qué son los modelos de datos y revisaremos algunos de los más significativos, dado que dieron lugar a determinadas Bases de Datos que estudiaremos más tarde.

Se puede definir el *modelo de datos* como una colección de herramientas conceptuales para describir los datos, las relaciones de datos, la semántica de los datos y las ligaduras de consistencia entre los datos.

En base a esto, podemos clasificar los diferentes modelos en tres grupos; modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos.

4.1. El modelo físico de datos

El modelo físico se usa para describir los datos en un nivel bajo, es decir, en el nivel físico. Existen pocos modelos de datos físicos que estén en uso, aunque los más conocidos son el modelo de unificación y el modelo de memoria por marcos. En este curso no entraremos a detallar el modelo de datos físico.

4.2. Los modelos lógicos basados en objetos

Los modelos lógicos basados en objetos se usan para describir datos en los niveles lógico y de vistas. Se caracterizan porque proporcionan estructuras muy flexibles y permiten que las ligaduras de los datos se especifiquen de forma explícita. Dentro de este tipo de modelos se encuentran, por ejemplo; el *modelo entidad-relación*, el *modelo orientado a objetos*, el *modelo de datos semántico*, el *modelo de datos funcional* y otros. A continuación describiremos solamente el modelo entidad-relación.

Modelo Entidad-Relación

El modelo de datos entidad-relación (E-R) se basa en percibir el mundo real como una colección de objetos básicos, llamados *entidades*, y las *relaciones* entre estos objetos.

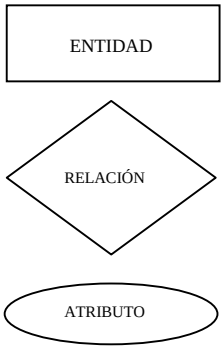
Por ejemplo, en una base de datos se almacena información de una serie de objetos o elementos. Estos objetos reciben el nombre de *entidades*. Entidad es cualquier objeto u ente sobre el que se almacena información. Así, en una base de datos académicos podrá haber información de las entidades alumno, profesor, asignatura, centro, plan de estudios, curso, etc. En una base de datos comerciales de una empresa aparecerán las entidades cliente, producto, vendedor, etc. De cada entidad se almacenan una serie de datos que se denominan *atributos o propiedades* de la entidad. Puede ser atributo de una entidad cualquier característica o propiedad de está que se considere relevante para la aplicación. Así pues, para una aplicación administrativa de un centro, el DNI, apellido y nombre, sexo, fecha de nacimiento, etc. son atributos de la entidad alumno.

En una base de datos se almacena, además de las entidades, las *relaciones* existentes entre ellas. Así por ejemplo, en la base de datos académicos antes citada hay relaciones entre las entidades curso y alumnos, alumnos y profesores, profesores y asignaturas, etc. Es decir, las relaciones asocian distintas entidades entre sí. Una entidad puede relacionarse con otras entidades y consigo misma.

Se llama **conjunto de entidades** al conjunto de todas las entidades del mismo tipo y **conjunto de relaciones**, al conjunto de relaciones del mismo tipo.

Además de entidades y relaciones, el modelo E-R representa ciertas ligaduras que los contenidos de las bases de datos deben cumplir. Una ligadura importante es la *correspondencia de cardinalidades*, que expresa el número de entidades con las que otra entidad se puede asociar a través de un conjunto de relaciones.

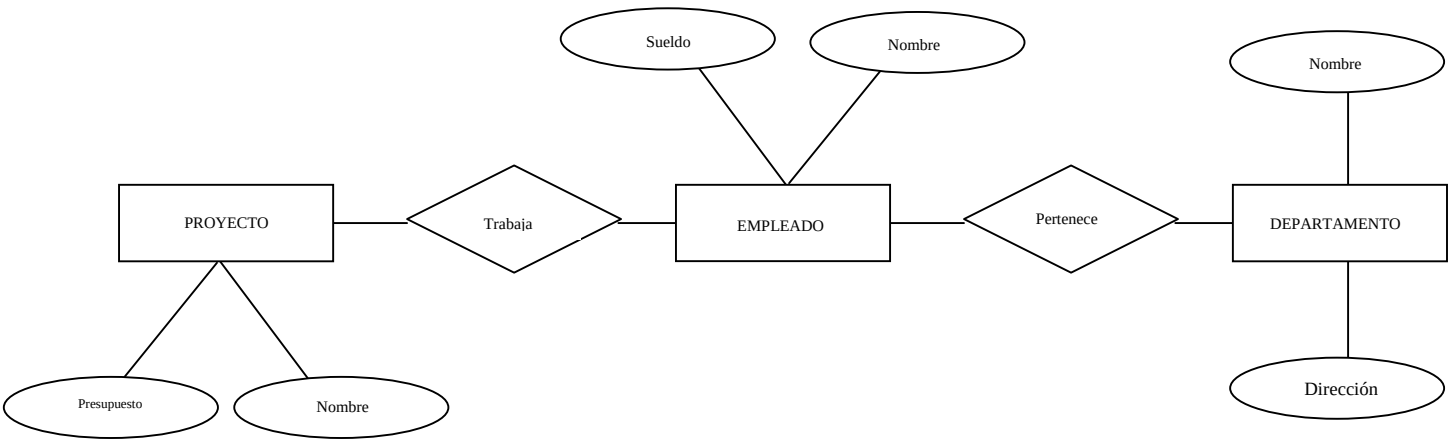
La totalidad de las estructuras lógicas de una BD se pueden expresa gráficamente mediante un **diagrama E-R** que consta de los siguientes componentes:



- Los rectángulos representan el conjunto de entidades.
- Los rombos representan las relaciones entre conjuntos de entidades.
- Las elipses representan los atributos.
- Las líneas unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

Además, cada componente se etiqueta con el nombre de la entidad o relación que representa.

Veamos un ejemplo:



4.3. Los modelos lógicos basados en registros

Los modelos lógicos basados en registros también se usan para describir datos en los niveles lógico y de vistas pero, a diferencia de los modelos basados en objetos, se usan tanto para especificar la estructura lógica completa de la base de datos como para proporcionar una descripción de alto nivel de la implementación. Los modelos basados en registros se llaman así debido a que la BD se estructura en registros de formato fijo de diferentes tipos. En cada tipo de registro se define un número fijo de campos o atributos, y cada campo tiene normalmente una longitud fija. Los tres modelos basados en registros más ampliamente aceptados son; el jerárquico, en red y el relacional. Veamos en que consisten estos tres modelos que posteriormente dieron lugar a que las bases de datos se clasificaran tradicionalmente en tres grupos: *jerárquicas, de red* y *relacionales*.

Modelo Jerárquico

En el modelo jerárquico, los datos se representan mediante colecciones de registros y las relaciones entre los datos se representan mediante enlaces, los cuales pueden verse como punteros. Los registros en la base de datos se organizan como colecciones de árboles.

Modelo de red

El modelo de red es parecido al jerárquico ya que, los datos y las relaciones entre los datos también se representan mediante registros y enlaces respectivamente. La diferencia es que aquí los registros se organizan como colecciones de grafos dirigidos.

Modelo Relacional

En el modelo relacional se usa una colección de tablas para representar tanto los datos como las relaciones entre esos datos. Cada tabla tiene varias columnas y cada columna tiene un nombre único. A diferencia de los anteriores, no usa punteros sino que relaciona los registros a través de los valores que contiene.

5. Clasificación de las bases de datos

Entidades y atributos son conceptos abstractos. En una base de datos, aunque la tecnología evoluciona constantemente, la información de cada entidad se almacena en registros, y cada atributo en los campos de dicho registro, de forma análoga a cómo se almacenaban en los archivos, aunque con la salvedad de que en las bases de datos cada entidad necesita registros con una estructura específica. El hecho de ir implementando nuevas bases de datos de acuerdo con la evolución sufrida por los modelos de datos, hizo que tradicionalmente las bases de datos se clasificaran en tres grupos: *jerárquicas, de red* y *relacionales*. Las dos primeras se diferencian en los tipos de relaciones que permiten, pudiendo decirse que la estructura jerárquica es un caso particular de la estructura de red. Por otra parte, las bases de datos relacionales son conceptualmente distintas, pues en ellas las relaciones se almacenan y manipulan de forma completamente distinta. A lo largo de esta sección veremos con más detalle estos tres grupos de bases de datos.

Si tenemos en cuenta que en la actualidad han ido apareciendo nuevos y más potentes modos de manipular datos así como nuevas tecnologías, podríamos ampliar la anterior clasificación considerando que, los sistemas de bases de datos se pueden clasificar de forma conveniente atendiendo a las estructuras de datos que manejan y a los operadores presentados al usuario y que le permiten acceder a la información almacenada en ella.

Desde esta perspectiva, los sistemas más antiguos se han denominado *pre-relacionales*, y se clasifican en tres categorías. Luego aparecen los sistemas *relacionales*, que marcan la frontera y definen el "antes y el después". Y posteriormente, los sistemas *post-relacionales*, que están todavía en evolución y marcan la pauta de las nuevas tendencias y tecnologías.

Pre-relacionales:
- De lista invertida (CA-DATACOM/DB, etc.)
- Jerárquicos (IMS de IBM, etc.)
- De red (CA-IDMS/DB, etc.)

Relacionales:
- Relacionales (ORACLE, DB2, SQL/DS, Rdb/VMS, INGRES, INFORMIX, SQLSERVER, etc.)

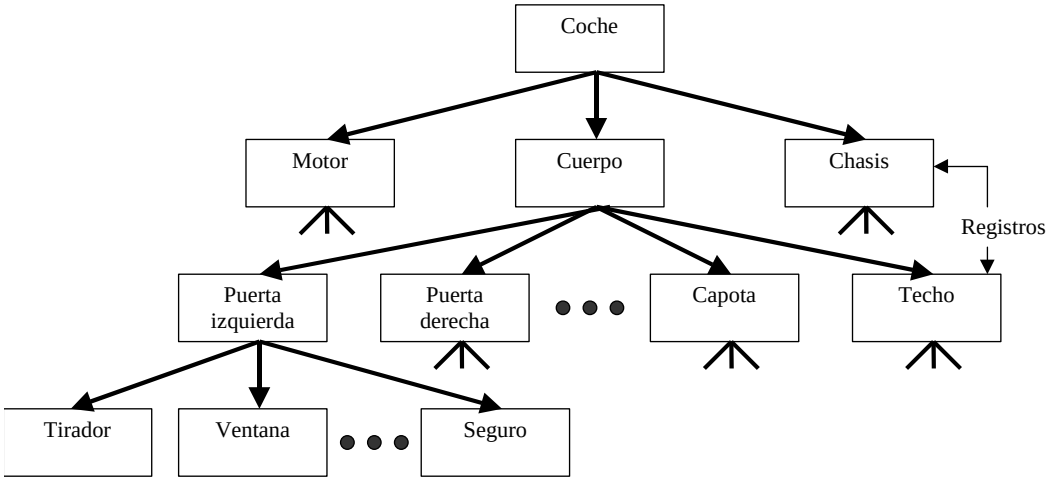
Post-relacionales:
- Sistemas deductivos de administración de bases de datos
- Sistemas semánticos de administración de bases de datos
- SGBD de relación universal
- SGBD orientados a objetos
- Sistemas extensibles de administración de bases de datos
- Sistemas expertos de administración de bases de datos

Analicemos un poco las características de los tres modelos básicos.

5.1 Base de datos jerárquica

Los primeros SBD introducidos a mediados de los 60, estaban basados en el modelo jerárquico que resume que: *todas las interrelaciones entre los datos pueden estructurarse como jerarquías*. Con esto, los datos se representan mediante una estructura en árbol. En un sistema jerárquico de BD los archivos se conectan entre sí mediante punteros físicos o campos de datos añadidos a los registros individuales. Un **puntero** (apuntador) es una dirección física que identifica dónde puede encontrarse un registro sobre el disco. En una jerarquía, un **hijo** (un registro "subordinado" en una interrelación jerárquica) puede solamente tener un **padre** (un registro "propietario" en una interrelación jerárquica), pero un padre puede tener varios hijos. A este tipo de relación se le llama *relación 1-a-Muchos*.

Veamos esto con una de las aplicaciones más importantes de los sistemas de gestión de base de datos primitivos como puede ser el planeamiento de la producción de empresas de facturación. Si un fabricante de automóviles decidía producir 10.000 unidades de un modelo de coche y 5.000 unidades de otro modelo, necesitaba saber cuántas piezas pedir a sus proveedores. Para responder a la cuestión, el producto (un coche) tenía que descomponerse en ensamblajes (motor, chasis, etc.), que a su vez se descomponían en subensamblajes (válvulas, cilindros, bujías, etc.) y luego en sub-subensamblajes, etc. El manejo de esta lista de piezas, conocido como una "cuenta de materiales", era un trabajo a medida para los ordenadores.



Ejemplo de base de datos jerárquica.

La cuenta de materiales para un producto tenía una estructura jerárquica natural. Para almacenar estos datos, se desarrollo el modelo de datos *jerárquico* (ver la figura). En este modelo, cada *registro* de la base de datos representa una pieza específica. Los registros tenían relaciones *padre/hijo*, que ligaban cada pieza a su subpieza, y así sucesivamente.

Para acceder a los datos en la base de datos, un programa podría:

- Hallar una pieza particular mediante su número (como por ejemplo la puerta izquierda).
- Descender al primer hijo (el tirador de la puerta).
- Ascender hasta su padre (el cuerpo).
- Moverse de lado hasta el siguiente hijo (la puerta derecha).

La recuperación de los datos en una base de datos jerárquica requería, por tanto, navegar a través de los registros moviéndose hacia arriba, hacia abajo y hacia los lados, un registro cada vez.

Uno de los sistemas de gestión de base de datos jerárquica más populares fue el Information Management System (IMS) de IBM, introducido en 1968. Las ventajas del IMS y su modelo jerárquico son las siguientes:

- Estructura simple. La organización de una base de datos IMS era fácil de entender. La jerarquía de la base de datos se asemejaba al diagrama de organización de una empresa o un árbol familiar.

- Organización padre/hijo. Una base de datos IMS era excelente para representar relaciones padre/hijo, tales como “A es pieza de B” o “A es propiedad de B”.

- Rendimiento. IMS almacenaba las relaciones padre/hijo como punteros físicos de un registro de datos a otro, de modo que el movimiento a través de la base de datos era rápido. Y dado que la estructura era sencilla, IMS podía colocar los registros padre e hijo cercanos unos a otros en el disco, minimizando la entrada/salida de disco.

IMS sigue siendo el SGBD más ampliamente instalado en los maxicomputadores IBM.

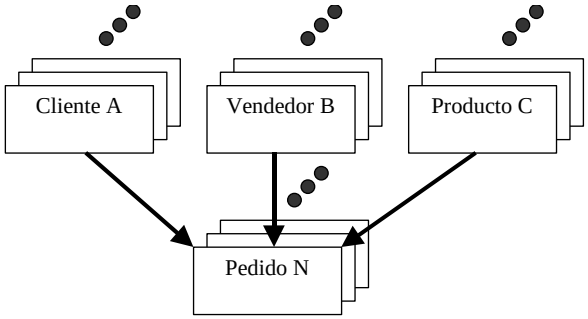
Los problemas que presentan las BD jerárquicas son:

- Únicamente pueden representar relaciones de 1-a-Muchos.
- Las relaciones Muchos-a-Muchos requieren la redundancia de información dado que, si un registro tipo aparece como "hijo" en más de dos relaciones, se debe de replicar. Como consecuencia, esto puede producir problemas de integridad y consistencia de los datos.
- Los lenguajes de manipulación asociados son fuertemente navegacionales.

5.2 Bases de datos de red

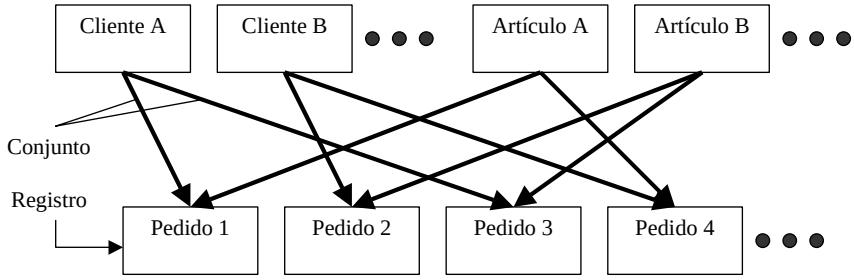
Rápidamente se comprobó que el modelo jerárquico tenía algunas limitaciones importantes, ya que no todas las interrelaciones podían expresarse fácilmente en una estructura jerárquica, por lo que surgieron las redes. Estas redes se denominan diagramas. Una **red** es una interrelación de datos en la cual un registro puede estar subordinado a registros de más de un archivo. A causa de la necesidad obvia de manipular tales interrelaciones, a finales de los años 60 se desarrollaron los sistemas *de red*. Al igual que los sistemas de bases de datos jerárquicos, los sistemas de bases de datos de red emplearon punteros físicos para enlazar entre sí los registros de diferentes archivos.

Resumiendo, la estructura sencilla de una base de datos jerárquica se convertía en una desventaja cuando los datos tenían estructuras más complejas. Por ejemplo, en una base de datos de procesamiento de pedidos, un simple pedido podría participar en tres relaciones padre/hijo diferentes, ligando el pedido al cliente que lo remitió, al vendedor que lo aceptó y al producto ordenado, tal como se muestra en la figura. La estructura de datos simplemente no se ajustaría a la jerarquía estricta de IMS.



Ejemplo de múltiples relaciones hijo/padre

Para manejar aplicaciones tales como el procesamiento de pedidos, se desarrolló un nuevo modelo de datos de red. El modelo de datos de red extendía el modelo jerárquico permitiendo que un registro participara en múltiples relaciones padre/hijo, reduciendo o eliminando de este modo las redundancias. Estas relaciones eran conocidas como *conjuntos* en el modelo de red. Entre mediados de los años 60 y principios de los 70 se desarrollaron y se comercializaron exitosamente varios SGBD en redes por lo que, en 1971, este modelo de datos se normalizó, es decir, se publicó un estándar oficial para bases de datos de red que se conoció como el modelo CODASYL.



Ejemplo de base de datos en red (CODASYL).

Hay que comentar que IBM nunca desarrolló un SGBD de red por sí mismo, sino que extendió el IMS a lo largo de los años. Sin embargo, durante los años setenta, otras compañías de software crearon productos que implementaban el modelo de red, tales como el IDMS de Cullinet, el Total de Cincom y el SGBD Adabas.

Para un programador, acceder a una base de datos de red era muy similar a acceder a una base de datos jerárquicos. Un programa de aplicación podía:

- Hallar un registro padre específico mediante una clave (como por ejemplo un número de cliente).
- Descender al primer hijo en un conjunto particular (el primer pedido remitido por ese cliente).
- Moverse lateralmente de un hijo al siguiente dentro del conjunto (la orden siguiente remitida por el mismo cliente).
- Ascender desde un hijo a su padre en otro conjunto (el vendedor que aceptó el pedido).

Una vez más el programador tenía que recorrer la base de datos registro a registro, especificando esta vez qué relación recorrer además de indicar la dirección.

Las bases de datos en red tenían varias ventajas:

- Flexibilidad. Las múltiples relaciones padre/hijo permitían a una base de datos de red representar datos que no tuvieran una estructura jerárquica sencilla.
- Normalización. El estándar CODASYL reforzó la popularidad del modelo de red y los vendedores de minicomputadoras, como Digital Equipment Corporation y Data General, implementaron bases de datos de red.
- Rendimiento. A pesar de su superior complejidad, las bases de datos de red reforzaron el rendimiento aproximándolo al de las bases de datos jerárquicas. Los conjuntos se representaron mediante punteros a registros de datos físicos, y en algunos sistemas, el administrador de la base de datos podía especificar la agrupación de datos basada en una relación de conjunto.

Por otra parte, las bases de datos de red tenían también sus desventajas. Al igual que las bases de datos jerárquicas, resultaban muy rígidas; Las relaciones de conjunto y la estructura de los registros tenían que ser especificadas de antemano; Modificar la estructura de la base de datos requería típicamente la reconstrucción de la base de datos completa.

Tanto las bases de datos jerárquicas como de red eran herramientas para programadores. Por ejemplo, para responder a una pregunta como “Cuál es el producto más popular ordenado por el cliente A”, un programador tenía que escribir un programa que recorriera su camino a través de la base de datos. Con frecuencia la anotación de las peticiones para informes a medida duraba semanas o meses y para el momento en que el programa estaba escrito, la información que se entregaba con frecuencia ya no merecía la pena.

5.3 Bases de datos relacionales

El uso de punteros físicos en las BD jerárquicas y de red era a la vez su punto fuerte y débil. Fuerte porque permitieron la recuperación rápida de los datos que tuvieran interrelaciones predeterminadas. Débil porque estas interrelaciones tenían que definirse antes de que el sistema fuera puesto en explotación. Era difícil, si no imposible, recuperar datos basados en otras interrelaciones, cosa que con el tiempo fue inaceptable.

En 1970, E. F. Codd publicó un artículo revolucionario que desafió fuertemente el juicio convencional de la "condición" de las bases de datos. Codd argumentó que los datos deberían relacionarse mediante interrelaciones naturales, lógicas, inherentes a los datos, más que mediante punteros físicos. Es decir, si la información lógica necesaria para hacer la combinación estaba presente en los datos, las personas deberían ser capaces de combinar los datos de fuentes diferentes. Esto abrió una nueva perspectiva para los sistemas de gestión de información ya que las interrogaciones a las bases de datos no necesitarían, en adelante, limitarse a las interrelaciones indicadas por los punteros físicos.

Los sistemas de bases de datos que soportan la recuperación de los datos, tomando en consideración las interrelaciones lógicas, podrán resolver fácilmente los problemas anteriormente planteados.

Dada la relevancia de estas bases de datos, y que son el objetivo de este curso, las trataremos más ampliamente en el tema 2.