Sistemas Gestores de Bases de Datos

Gestión y diseño de bases de datos

1. Tipos de Sistemas de Información

En la evolución de los sistemas de información ha habido dos puntos determinantes, que han formado los dos tipos fundamentales de sistemas de información.

1.1 Sistemas de Información orientados a procesos.

Antes de aparecer los SGBD (década de los setenta), la información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos que iban soportados sobre un sistema operativo. Éstos consistían en un conjunto de programas que definían y trabajaban sus propios datos. Los datos se almacenan en archivos y los programas manejan esos archivos para obtener la información. Si la estructura de los datos de los archivos cambia, todos los programas que los manejan se deben modificar; por ejemplo, un programa trabaja con un archivo de datos de alumnos, con una estructura o registro ya definido; si se incorporan elementos o campos a la estructura del archivo, los programas que utilizan ese archivo se tienen que modificar para tratar esos nuevos elementos. En estos sistemas de gestión de archivos, la definición de los datos se encuentra codificada dentro de los programas de aplicación en lugar de almacenarse de forma independiente, y además el control del acceso y la manipulación de los datos vienen impuesto por los programas de aplicación.
En estos sistemas de información se crean diversas aplicaciones (software) para gestionar diferentes aspectos del sistema. Cada aplicación realiza unas determinadas operaciones. Los datos de dichas aplicaciones se almacenan en archivos digitales dentro de las unidades de almacenamiento del ordenador (a veces en archivos binarios, o en hojas de cálculo, o incluso en archivos de texto).
Cada programa almacena y utiliza sus propios datos de forma un tanto caótica. La ventaja de este sistema (la única ventaja), es que los procesos son independientes por lo que la modificación de uno no afectaba al resto.
Esto supone un gran inconveniente a la hora de tratar grandes volúmenes de información.
Surge así la idea de separar los datos contenidos en los archivos de los programas que los manipulan, es decir, que se pueda modificar la estructura de los datos de los archivos sin que por ello se tengan que modificar los programas con los que trabajan. Se trata de estructurar y organizar los datos de forma que se pueda acceder a ellos con independencia de los programas que los gestionan.

Inconvenientes de un sistema de gestión de archivos:

- ✓ **Redundancia e inconsistencia de los datos:** se produce porque los archivos son creados por distintos programas y van cambiando a lo largo del tiempo, es decir, pueden tener distintos formatos y los datos pueden estar duplicados en varios sitios. Por ejemplo, el teléfono de un alumno puede aparecer en más de un archivo. La redundancia aumenta los costes de almacenamiento y acceso, y trae consigo la inconsistencia de los datos: las copias de los mismos datos no coinciden por aparecer en varios archivos.

- ✓ **Dependencia de los datos física-lógica:** o lo que es lo mismo, la estructura física de los datos (definición de archivos y registros) se encuentra codificada en los programas de aplicación.

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

- ✓ **Dificultad para tener acceso a los datos:** proliferación de programas, es decir, cada vez que se necesite una consulta que no fue prevista en el inicio implica la necesidad de codificar el programa de aplicación necesario. Lo que se trata de probar es que los entornos convencionales de procesamiento de archivos no permiten recuperar los datos necesarios de una forma conveniente y eficiente.

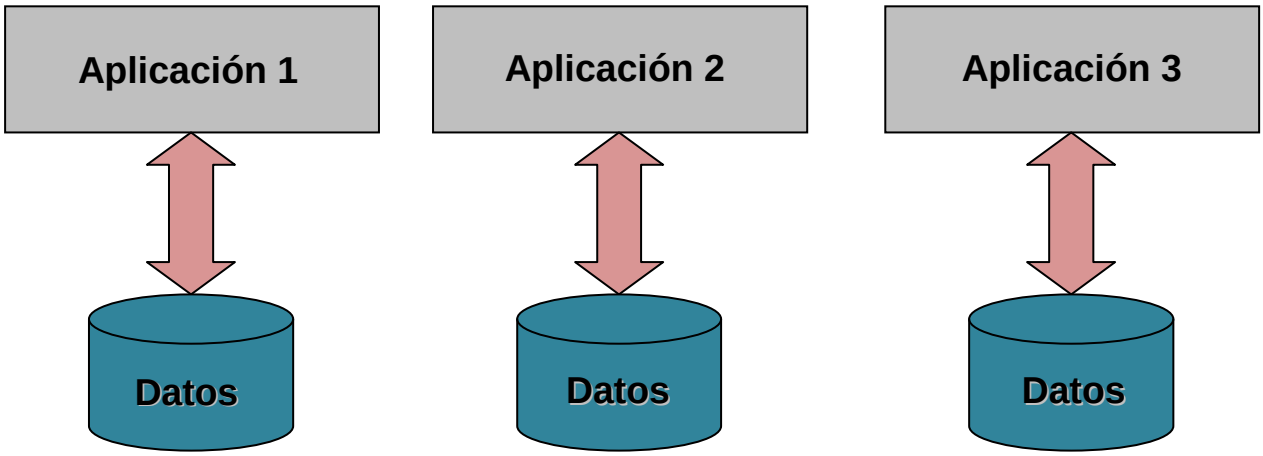
- ✓ **Separación y aislamiento de los datos:** es decir, al estar repartidos en varios archivos, y tener diferentes formatos, es difícil escribir nuevos programas que aseguren la manipulación de los datos correctos. Antes se deberían sincronizar todos los archivos para que los datos coincidiesen.

- ✓ **Dificultad para el acceso concurrente:** pues en un sistema de gestión de archivos es complicado que los usuarios actualicen los datos simultáneamente. Las actualizaciones concurrentes pueden dar por resultado datos inconsistentes, ya que se puede acceder a los datos por medio de diversos programas de aplicación.

- ✓ **Dependencia de la estructura del archivo con el lenguaje de programación:** pues la estructura se define dentro de los programas. Esto implica que los formatos de los archivos sean incompatibles. La incompatibilidad entre archivos generados por distintos lenguajes hace que los datos sean difíciles de procesar.
  Para poder saber cómo se almacenan los datos, es decir qué estructura se utiliza de los mismos, necesitamos ver el código de la aplicación; es decir el código y los datos no son independientes.

- ✓ **Problemas en la seguridad de los datos:** Resulta difícil implantar restricciones de seguridad pues las aplicaciones se van añadiendo al sistema según se van necesitando.
  Ya que cada aplicación se crea independientemente; es por tanto muy difícil establecer criterios de seguridad uniformes.

- ✓ **Problemas de integridad de datos (datos inconsistentes):** es decir, los valores almacenados en los archivos deben cumplir con restricciones de consistencia. Ya que un proceso cambia sus datos y no el resto. Por lo que el mismo dato puede tener valores distintos según qué aplicación acceda a él. Por ejemplo, no se puede insertar una nota de un alumno en una asignatura si previamente esa asignatura no está creada. Otro ejemplo, las unidades en almacén de un producto determinado no deben ser inferiores a una cantidad. Esto implica añadir gran número de líneas de código en los programas. El problema se complica cuando existen restricciones que implican varios datos en distintos archivos.



A estos sistemas se les llama sistemas de gestión de ficheros. Se consideran también así a los sistemas que utilizan programas ofimáticos (como **Word** o **Excel** por ejemplo) para gestionar sus datos (muchas pequeñas empresas utilizan esta forma de administrar sus datos). De hecho estos sistemas producen los mismos (si no más) problemas.

1.2 Sistemas de Información orientados a los datos. Bases de Datos.

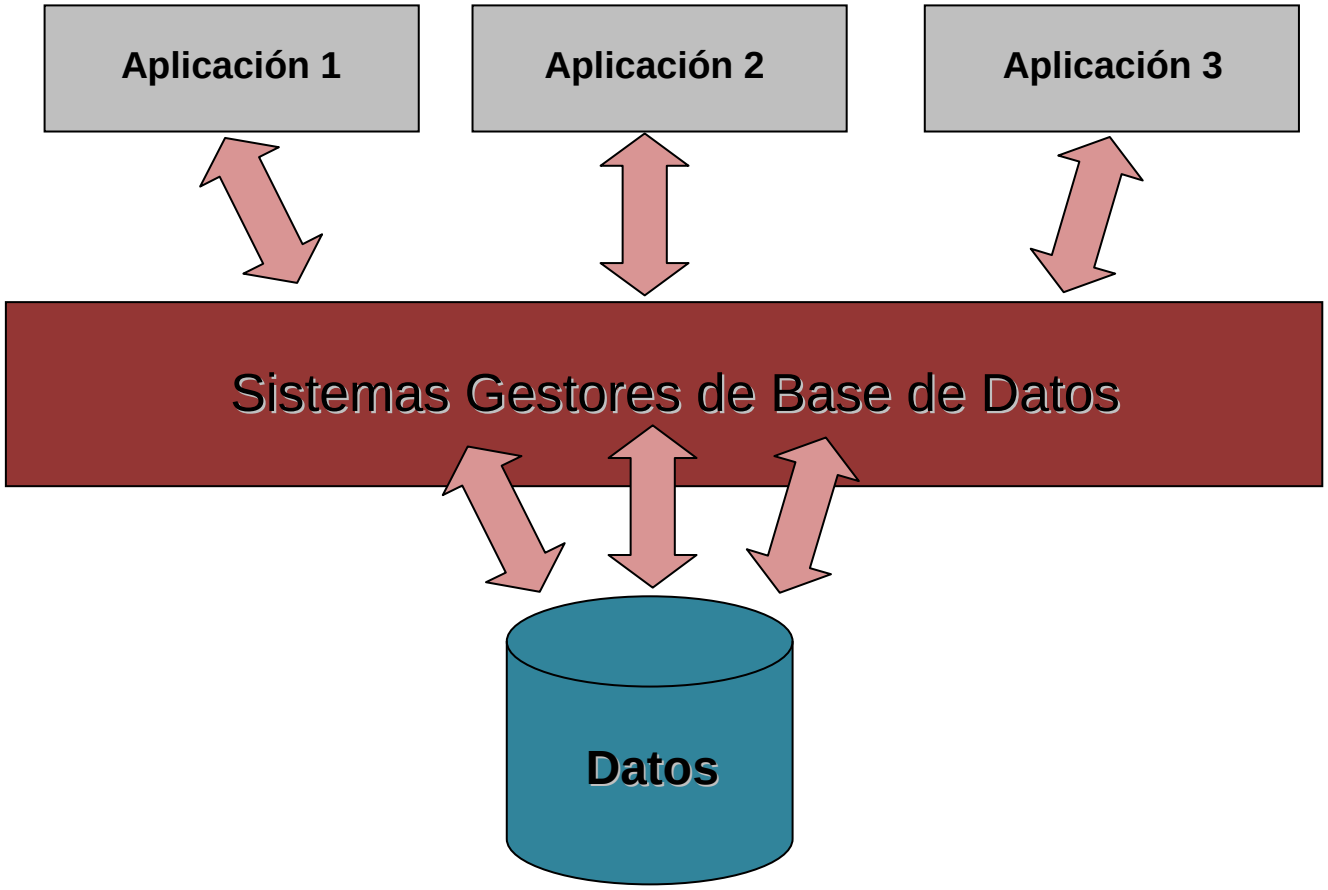
Todos estos inconvenientes hacen posible el fomento y desarrollo de SGBD. El objetivo primordial de un gestor es proporcionar eficiencia y seguridad a la hora de extraer o almacenar información en las BD. Los sistemas gestores de BBDD están diseñados para gestionar grandes bloques de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información.
En este tipo de sistemas los datos se centralizan en una **base de datos** común a todas las aplicaciones. En esos sistemas los datos se almacenan en una única estructura lógica que es utilizable por las aplicaciones. A través de esa estructura se accede a los datos que son comunes a todas las aplicaciones. Cuando una aplicación modifica un dato, dicho dato la modificación será visible para el resto de aplicaciones.
Una BD es un gran almacén de datos que se define una sola vez; los datos pueden ser accedidos de forma simultánea por varios usuarios; están relacionados y existe un número mínimo de duplicidad; además en las BBDD se almacenarán las descripciones de esos datos, lo que se llama **metadatos** en el diccionario de datos.

Ventajas

- ✓ **Independencia de los datos y los programas y procesos.** Esto permite modificar los datos sin modificar el código de las aplicaciones.
- ✓ **Menor redundancia**. No hace falta tanta repetición de datos. Sólo se indica la forma en la que se relacionan los datos**.**
- ✓ **Integridad de los datos**. Mayor dificultad de perder los datos o de realizar incoherencias con ellos.
- ✓ **Mayor seguridad en los datos.** Al permitir limitar el acceso a los usuarios. Cada tipo de usuario podrá acceder a unas cosas.
- ✓ **Datos más documentados.** Gracias a los **metadatos** que permiten describir la información de la base de datos.
- ✓ **Acceso a los datos más eficiente.** La organización de los datos produce un resultado más óptimo en rendimiento.
- ✓ **Menor espacio de almacenamiento.** Gracias a una mejor estructuración de los datos.
- ✓ **Acceso simultáneo a los datos.** Es más fácil controlar el acceso de usuarios de forma concurrente.

Desventajas

- ✓ **Instalación costosa.** El control y administración de bases de datos requiere de un software y hardware poderoso.
- ✓ **Requiere personal cualificado.** Debido a la dificultad de manejo de este tipo de sistemas.
- ✓ **Implantación larga y difícil.** Debido a los puntos anteriores. La adaptación del personal es mucho más complicada y lleva bastante tiempo.
- ✓ **Ausencia de estándares reales.** Lo cual significa una excesiva dependencia hacia los sistemas comerciales del mercado. Aunque, hoy en día, una buena parte de esta tecnología está aceptada como estándar de hecho.



2. Arquitectura de los Sistemas Gestores de Bases de Datos

Un sistema gestor de bases de datos o **SGBD** (aunque se suele utilizar más a menudo las siglas **DBMS** procedentes del inglés, ***Data Base Management System***) es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos.
En estos sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos, garantizando además la seguridad de los mismos.
El éxito del SGBD reside en mantener la seguridad e integridad de los datos. Lógicamente tiene que proporcionar herramientas a los distintos usuarios.
Entre las herramientas que proporciona están:

- ✓ **Herramientas para la creación y especificación de los datos**: así como la estructura de la base de datos. Especificación de la estructura, el tipo de los datos, las restricciones y relaciones entre ellos mediante lenguajes de definición de datos. Toda esta información se almacena en el diccionario de datos, el SGBD proporcionará mecanismos
  para la gestión del diccionario de datos.

- ✓ **Herramientas para administrar y crear la estructura física:** requerida en las unidades de almacenamiento.

- ✓ **Herramientas para la manipulación de los datos:** de las bases de datos, para añadir, modificar, suprimir o consultar datos.

- ✓ **Herramientas de recuperación:** en caso de desastre.

- ✓ **Herramientas para la creación de copias de seguridad:** para restablecer la información en caso de fallos en el sistema.

- ✓ **Herramientas para la gestión de la comunicación:** de la base de datos.

- ✓ **Herramientas para la creación de aplicaciones:** que utilicen esquemas externos de los datos.

- ✓ **Herramientas de instalación:** de la base de datos.

- ✓ **Herramientas para la exportación e importación:** de datos.

2.1 Niveles de abstracción de una base de datos.

Introducción

En cualquier sistema de información se considera que se pueden observar los datos desde dos puntos de vista:

- ✓ **Nivel externo**: Esta es la visión de los datos que poseen los usuarios del Sistema de Información.

- ✓ **Nivel físico**: Esta es la forma en la que realmente están almacenados los datos.

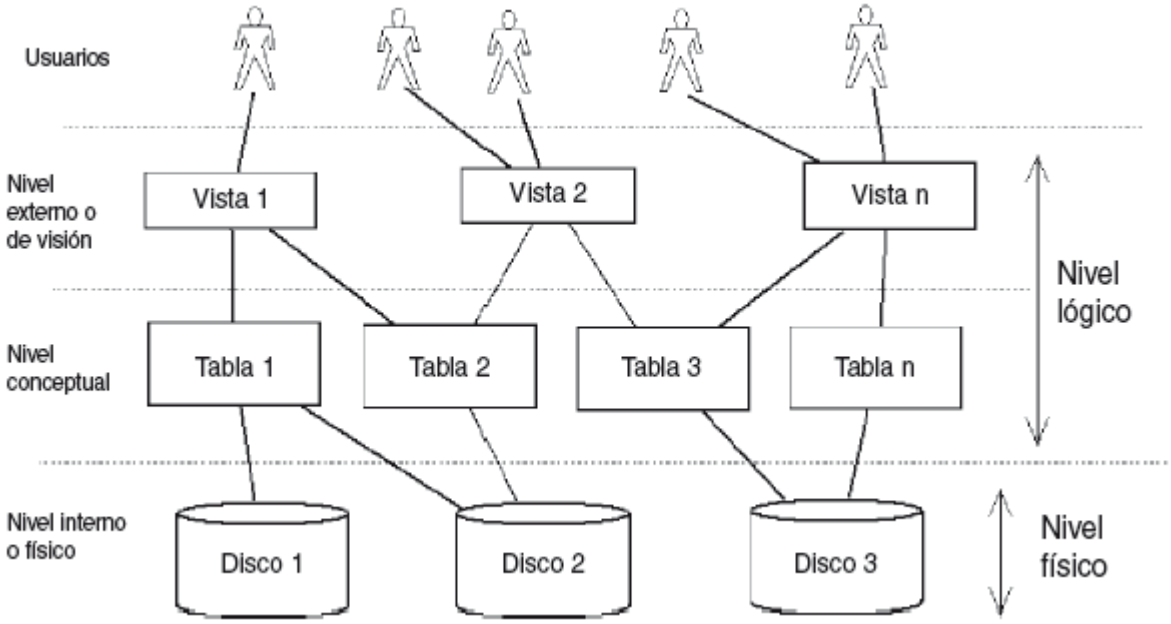
Realmente la base de datos es la misma, pero se la puede observar desde estos dos puntos de vista. Al igual que una casa se la pueda observar pensando en los materiales concretos con los que se construye o bien pensando en ella con el plano en papel.
En todo sistema de información digital, los usuarios ven los datos desde las aplicaciones creadas por los programadores. A ese nivel se manejan formularios, informes en pantalla o en papel,...
Pero la realidad física de esos datos, tal cual se almacenan en los discos queda oculta a los usuarios. Esa forma de ver la base de datos está reservada a los administradores. Es el nivel físico el que permite ver la base de datos en función de cómo realmente se están almacenando en el ordenador, en qué carpeta, qué archivos se usan…
En el caso de los Sistemas de Base de datos, se añade un tercer nivel, un tercer punto de vista, es el **nivel conceptual**. Ese nivel se sitúa entre el físico y el externo.
En cada nivel se manejan esquemas de la base de datos, al igual que al construir una casa, los distintos profesionales manejan distintos tipos de planos (eléctricos, de albañilería, de tuberías de agua,…). Con lo cual una base de datos requiere diseñar al menos tres esquemas (en realidad son más).
En 1975, el comité ANSI-SPARC (*American National Standard Institute - Standards Planning and Requirements Committee*) propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era el de separar los programas de aplicación de la BD física. En esta arquitectura el esquema de una BD se define en tres niveles de abstracción distintos:

Nivel interno o físico: el más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. Describe la estructura física de la BD mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etcétera.
Esta visión sólo la requiere el **administrador/a**. El administrador la necesita para poder gestionar más eficientemente la base de datos.

Nivel externo o de visión: es el más cercano a los usuarios, es decir, es donde se describen varios esquemas externos o vistas de usuarios. Cada esquema describe la parte de la BD que interesa a un grupo de usuarios en este nivel se representa la visión individual de un usuario o de un grupo de usuarios.
En realidad son varios. Se trata de la visión de los datos que poseen los **usuarios y usuarias finales.** Esa visión es la que obtienen a través de las aplicaciones. Las aplicaciones creadas por los desarrolladores abstraen la realidad conceptual de modo que el usuario no conoce las relaciones entre los datos, como tampoco conoce dónde realmente se están almacenando los datos.
Los esquemas externos los realizan las **programadoras/es** según las indicaciones formales de los y las **analistas**.
Realmente cada aplicación produce un esquema externo diferente (aunque algunos pueden coincidir) o **vista de usuario**. El conjunto de todas las vistas de usuario es lo que se denomina **esquema externo global**.

Nivel conceptual: describe la estructura de toda la BD para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos, relaciones, operaciones de los usuarios y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento. Representa la información contenida en la BD.
Se trata de un esquema teórico de los datos en el que figuran organizados en estructuras reconocibles del mundo real y en el que también aparece la forma de relacionarse los datos. Este esquema es el paso que permite modelar un problema real a su forma correspondiente en el ordenador.

Este esquema es la base de datos de todos los demás. Como se verá más adelante, es el primer paso a realizar al crear una base de datos. En definitiva es el plano o modelo general de la base de datos.
El esquema conceptual lo realizan **diseñadores/as** o **analistas**.



Niveles de abstracción de la arquitectura ANSI.

3. Componentes de los Sistemas Gestores de Bases de Datos

Los SGBD son paquetes de software muy complejos que deben proporcionar una serie de servicios que van a permitir almacenar y explotar los datos de forma eficiente. Los componentes principales son los siguientes.

Lenguajes de los SGBD

Todos los SGBD ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: administradores, diseñadores, programadores de aplicaciones y usuarios finales.
Los lenguajes van a permitir al administrador de la BD especificar los datos que componen la BD, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios.

Los lenguajes del SGBD se clasifican en:

Lenguaje de definición de datos (LDD o DDL): se utiliza para especificar el esquema de la BD, las vistas de los usuarios y las estructuras de almacenamiento.
Es el que define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la BD.

Permite al diseñador de la base de datos crear las estructuras apropiadas para integrar adecuadamente los datos. Se dice que esta función es la que permite definir las tres estructuras de la base de datos (relacionadas con los tres niveles de abstracción).

- **Estructura interna**
- **Estructura conceptual**
- **Estructura externa**

Realmente esta función trabajo con **metadatos**. Los metadatos es la información de la base de datos que realmente sirve para describir a los datos. Es decir, ***Sánchez Rodríguez*** y ***Crespo*** son datos; pero ***Primer Apellido*** es un metadato. También son datos decir que la base de datos contiene ***Alumnos*** o que el ***dni*** lo forman 9 caracteres de los cuales los 8 primeros son números y el noveno un carácter en mayúsculas.
La función de definición sirve pues para **crear, eliminar o modificar metadatos**. Para ello permite usar un **lenguaje de descripción de datos** o **DDL**. Mediante ese lenguaje:

- Se definen las estructuras de datos
- Se definen las relaciones entre los datos
- Se definen las reglas que han de cumplir los datos

Lenguaje de manipulación de datos (LMD o DML): se utilizan para leer y actualizar los datos de la BD. Es el utilizado por los usuarios para realizar consultas, inserciones, eliminaciones y modificaciones. Los hay *procedurales*, en los que el usuario será normalmente un programador y especifica las operaciones de acceso a los datos llamando a los procedimientos necesarios. Estos lenguajes acceden a un registro y lo procesan.
Las sentencias de un DML procedural están embebidas en un lenguaje de alto nivel llamado *anfitrión*. Las BD jerárquicas y en red utilizan estos DML procedurales.

Mediante ese lenguaje se puede:

- Añadir datos
- Eliminar datos
- Modificar datos
- Buscar datos

Actualmente se suele distinguir aparte la función de buscar datos en la base de datos (**función de consulta**). Para lo cual se proporciona un **lenguaje de consulta de datos** o **DQL.**

Lenguaje de control de datos (LCD o DCL): Mediante esta función los administradores poseen mecanismos para proteger los datos; de modo que se permite a cada usuario ver ciertos datos y otros no; o bien usar ciertos recursos concretos de la base de datos y prohibir otros.
Es decir simplemente permite controlar la seguridad de la base de datos. El lenguaje que implementa esta función es el **lenguaje de control de datos** o **DCL.**

No procedurales son los lenguajes declarativos. En muchos SGBD se pueden introducir interactivamente instrucciones del LMD desde un terminal, también pueden ir embebidas en un lenguaje de programación de alto nivel. Estos lenguajes permiten especificar los datos a obtener en una consulta, o los datos a modificar, mediante sentencias sencillas. Las BD relacionales utilizan lenguajes no procedurales como SQL (*Structured Query Language*) o QBE (*Query By Example*).

La mayoría de los SGBD comerciales incluyen **lenguajes de cuarta generación (4GL)** que permiten al usuario desarrollar aplicaciones de forma fácil y rápida, también se les llama *herramientas de desarrollo*.
Ejemplos de esto son las herramientas del SGBD ORACLE: SQL Forms para la generación de formularios de pantalla y para interactuar con los datos; SQL Reports para generar informes de los datos contenidos en la BD; PL/SQL lenguaje para crear procedimientos que interractuen con los datos de la BD.

3.1 Recursos humanos de las bases de datos.

Intervienen (como ya se ha comentado) muchas personas en el desarrollo y manipulación de una base de datos. Habíamos seleccionado cuatro tipos de usuarios (administradores/as, desarrolladores, diseñadores/as y usuarios/as). Ahora vamos a desglosar aún más esta clasificación.

Informáticos

Lógicamente son los profesionales que definen y preparan la base de datos. Pueden ser:

- ✓ **Directivos/as.** Organizadores y coordinadores del proyecto a desarrollar y máximos responsables del mismo. Esto significa que son los encargados de decidir los recursos que se pueden utilizar, planificar el tiempo y las tareas, la atención al usuario y de dirigir las entrevistas y reuniones pertinentes.
- ✓ **Analistas**. Son los encargados de controlar el desarrollo de la base de datos aprobada por la dirección. Normalmente son además los **diseñadores de la base de datos** (especialmente de los esquemas interno y conceptual) y los directores de la programación de la misma.
- ✓ **Administradores/as de las bases de datos.** Encargados de crear el esquema interno de la base de datos, que incluye la planificación de copia de seguridad, gestión de usuarios y permisos y creación de los objetos de la base de datos.
- ✓ **Desarrolladores/as o programadores/as.** Encargados de la realización de las aplicaciones de usuario de la base de datos.
- ✓ **Equipo de mantenimiento.** Encargados de dar soporte a los usuarios en el trabajo diario (suelen incorporar además tareas administrativas como la creación de copias de seguridad por ejemplo o el arreglo de problemas de red por ejemplo).

Usuarios

- ✓ **Expertos/as.** Utilizan el lenguaje de manipulación de datos (**DML**) para acceder a la base de datos. Son usuarios que utilizan la base de datos para gestión avanzada de decisiones.
- ✓ **Habituales.** Utilizan las aplicaciones creadas por los desarrolladores para consultar y actualizar los datos. Son los que trabajan en la empresa a diario con estas herramientas y el objetivo fundamental de todo el desarrollo de la base de datos.
- ✓ **Ocasionales**. Son usuarios que utilizan un acceso mínimo a la base de datos a través de una aplicación que permite consultar ciertos datos. Serían por ejemplo los usuarios que consultan el horario de trenes a través de Internet.

El Administrador de la Base de Datos DBA

El DBA tiene una gran responsabilidad ya que posee el máximo nivel de privilegios. Será el encargado de crear los usuarios que se conectarán a la BD. En la administración de una BD siempre hay que procurar que haya el menor número de administradores, a ser posible una sola persona.
El objetivo principal de un DBA es garantizar que la BD cumple los fines previstos por la organización, lo que incluye una serie de tareas como:

- ✓ Instalar SGBD en el sistema informático.
- ✓ Crear las BBDD que se vayan a gestionar.
- ✓ Crear y mantener el esquema de la BD.
- ✓ Crear y mantener las cuentas de usuario de la BD.
- ✓ Arrancar y parar SGBD, y cargar las BBDD con las que se ha de trabajar.
- ✓ Colaborar con el administrador del S.O. en las tareas de ubicación, dimensionado y control de los archivos y espacios de disco ocupados por el SGBD.
- ✓ Colaborar en las tareas de formación de usuarios.
- ✓ Establecer estándares de uso, políticas de acceso y protocolos de trabajo diario para los usuarios de la BD.
- ✓ Suministrar la información necesaria sobre la BD a los equipos de análisis y programación de aplicaciones.

Efectuar tareas de explotación como:

- ➢ Vigilar el trabajo diario colaborando en la información y resolución de las dudas de los usuarios de la BD.
- ➢ Controlar en tiempo real los accesos, tasas de uso, cargas en los servidores, anomalías, etcétera.
- ➢ Llegado el caso, reorganizar la BD.
- ➢ Efectuar las copias de seguridad periódicas de la BD.
- ➢ Restaurar la BD después de un incidente material a partir de las copias de seguridad.
- ➢ Estudiar las auditorías del sistema para detectar anomalías, intentos de violación de la seguridad, etcétera.
- ➢ Ajustar y optimizar la BD mediante el ajuste de sus parámetros, y con ayuda de las herramientas de monitorización y de las estadísticas del sistema.
- ➢ En su gestión diaria, el DBA suele utilizar una serie de herramientas de administración de la BD.
- ➢ Con el paso del tiempo, estas herramientas han adquirido sofisticadas prestaciones y facilitan en gran medida la realización de trabajos que, hasta no hace demasiado, requerían de arduos esfuerzos por parte de los administradores.

3.2 Estructura multicapa.

El proceso que realiza un SGBD está en realidad formado por varias capas que actúan como interfaces entre el usuario y los datos. Fue el propio organismo ANSI (en su modelo X3/SPARC que luego se comenta) la que introdujo una mejora de su modelo de bases de datos en 1988 a través de un grupo de trabajo llamado **UFTG** (***User Facilities Task Group,*** grupo de trabajo para las facilidades de usuario). Este modelo toma como objeto principal al usuario habitual de la base de datos y modela el funcionamiento de la base de datos en una sucesión de capas cuya finalidad es ocultar y proteger la parte interna de las bases de datos.
Desde esta óptica para llegar a los datos hay que pasar una serie de capas que desde la parte más externa poco a poco van entrando más en la realidad física de la base de datos.

Modelo de referencia de las facilidades de usuario

Facilidades de usuario

Son las herramientas que proporciona el SGBD a los usuarios para permitir un acceso más sencillo a los datos. Actúan de interfaz entre el usuario y la base de datos, y son el único elemento que maneja el usuario. Son, en definitiva, las páginas web y las aplicaciones con las que los usuarios manejan la base de datos. Permite abstraer la realidad de la base de datos a las usuarias y usuarios, mostrando la información de una forma más humana.

Capa de acceso a datos

La capa de acceso a datos es la que permite comunicar a las aplicaciones de usuario con el diccionario de datos. Es un software (un driver o controlador en realidad) que se encarga traducir las peticiones del usuario para que lleguen de forma correcta a la base de datos y ésta pueda responder de forma adecuada.

Diccionario de datos

Se trata del elemento que posee todos los metadatos. Gracias a esta capa las solicitudes de los clientes (que son conceptuales antes de llegar aquí) se traducen en instrucciones que hacen referencia al esquema interno de la base de datos.
El **diccionario de datos** es el lugar donde se deposita información acerca de todos los datos que forman la BD. Es una guía en la que se describe la BD y los objetos que la forman.
El diccionario contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información.
En una BD relacional, el diccionario de datos proporciona información acerca de:

- ✓ La estructura lógica y física de la BD.
- ✓ Las definiciones de todos los objetos de la BD: tablas, vistas, índices, disparadores, procedimientos, funciones, etcétera.
- ✓ El espacio asignado y utilizado por los objetos.
- ✓ Los valores por defecto de las columnas de las tablas.
- ✓ Información acerca de las restricciones de integridad.
- ✓ Los privilegios y roles otorgados a los usuarios.
- ✓ Auditoría de información, como los accesos a los objetos.

Un diccionario de datos debe cumplir las siguientes características:

- ➢ Debe soportar las descripciones de los modelos conceptual, lógico, interno y externo de la BD.
- ➢ Debe estar integrado dentro del SGBD.
- ➢ Debe apoyar la transferencia eficiente de información al SGDB. La conexión entre los modelos interno y externo debe ser realizada en tiempo de ejecución.
- ➢ Debe comenzar con la reorganización de versiones de producción de la BD. Además debe reflejar los cambios en la descripción de la BD. Cualquier cambio a la descripción de programas ha de ser reflejado automáticamente en la librería de descripción de programas con la ayuda del diccionario de datos.
- ➢ Debe estar almacenado en un medio de almacenamiento con acceso directo para la fácil recuperación de información.

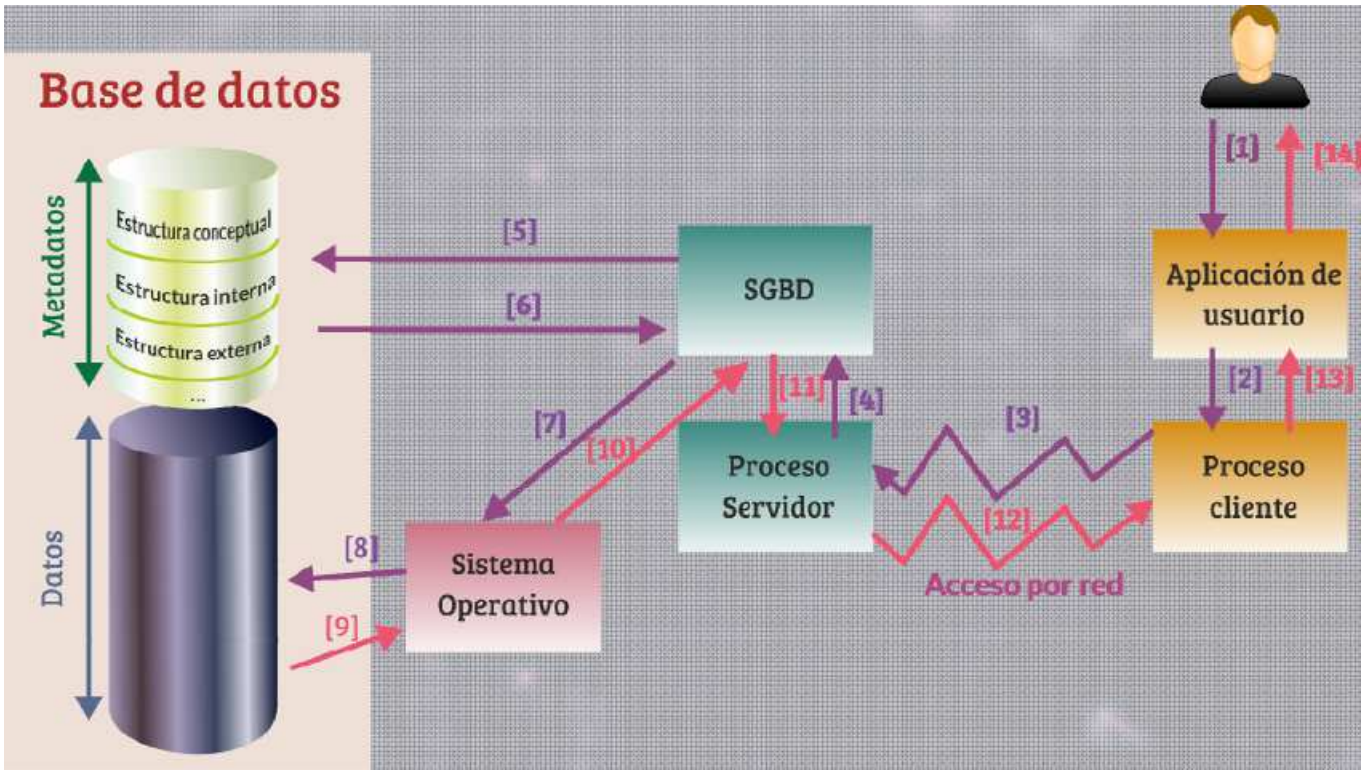
Núcleo

El núcleo de la base de datos es la encargada de traducir todas las instrucciones requeridas y prepararlas para su correcta interpretación por parte del sistema. Realiza la traducción física de las peticiones.

Sistema operativo

Es una capa externa al software SGBD pero es la única capa que realmente accede a los datos en sí. En realidad los SGBD no acceden directamente al disco, sino que piden al Sistema Operativo que lo haga.

3.3 Funcionamiento del SGBD.



En esta arquitectura describe los datos a tres niveles de abstracción. En realidad los únicos datos que existen están a nivel físico almacenados en discos u otros dispositivos. Los SGBD basados en esta arquitectura permiten que cada grupo de usuarios haga referencia a su propio esquema externo.
El SGBD debe de transformar cualquier petición de usuario (esquema externo) a una petición expresada en términos de esquema conceptual, para finalmente ser una petición expresada en el esquema interno que se procesará sobre la BD almacenada. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación, el SGBD es capaz de interpretar una solicitud de datos y realiza los siguientes pasos:

1) El usuario solicita unos datos y crea una consulta.

2) La Aplicación del usuario convierte esta consulta en un proceso realizado por el cliente del SGBD.

3) La consulta viaja a través de un medio (red).

4) El SGBD verifica y acepta el esquema externo para ese usuario. Se convierte el proceso del usuario en un "Proceso de Servidor" interno.

5) Transforma la solicitud al esquema conceptual.

6) Verifica y acepta el esquema conceptual.

7) El proceso lanzado por el usuario llama al SGBD indicando la porción de la base de datos que se desea tratar.

8) Transforma la solicitud al esquema físico o interno. El SGBD obtiene el esquema físico.

9) El SGBD traduce la llamada a los métodos de acceso del Sistema Operativo que permiten acceder realmente a los datos requeridos.

10) Selecciona la o las tablas implicadas en la consulta y ejecuta la consulta. El Sistema Operativo accede a los datos tras traducir las órdenes dadas por el SGBD.

11) Transforma del esquema interno al conceptual, y del conceptual al externo. Los datos pasan del disco a una memoria intermedia o ***buffer***. En ese buffer se almacenarán los datos según se vayan recibiendo

12) Los datos pasan del buffer al área de trabajo del usuario (**ATU**) del proceso del usuario. Los pasos se repiten hasta que se envíe toda la información al proceso de usuario.

13) En el caso de que haya errores en cualquier momento del proceso, el SGBD devuelve indicadores en los que manifiesta si ha habido errores o advertencias a tener en cuenta. Esto se indica al área de comunicaciones del proceso de usuario. Si las indicaciones son satisfactorias, los datos de la ATU serán utilizables por el proceso de usuario.

14) Finalmente, el usuario ve los datos solicitados.

Para una BD específica sólo hay un esquema interno y uno conceptual, pero puede haber varios esquemas externos definidos para uno o para varios usuarios.

3.4 Formas de ejecución de un SGBD.

Actualmente casi todos los Sistemas Gestores de Bases de Datos funcionan de manera semejante, en realidad hay tres posibilidades de funcionamiento:

SGBDs monocapa: Es la más sencilla, pero la que tiene menos escalabilidad (posibilidad de crecer). El Sistema Gestor se instala en una máquina y los usuarios acceden directamente a esa máquina y ese Sistema Gestor. En estos sistemas no se accede de forma remota a la base de datos.

SGBDs bicapa: Estructura clásica, la base de datos y su SGBD están en un servidor al cual acceden los clientes. El cliente posee software que permite al usuario enviar instrucciones al SGBD en el servidor y recibir los resultados de estas instrucciones. Para ello el software cliente y el servidor deben utilizar software de comunicaciones en red. Hay dos posibilidades:

- **Estructura Cliente-Servidor.** La base de datos está en un solo servidor al que acceden los clientes (incluso simultáneamente).

- **Cliente Multi-servidor.** En este caso los clientes acceden a un conjunto de servidores que distribuyen la base de datos. El cliente no sabe si los datos están en uno o más servidores, ya que el resultado es el mismo independientemente de dónde se almacenan los datos. Se usa cuando el número de clientes ha crecido mucho y un solo servidor no podría atender sus peticiones.

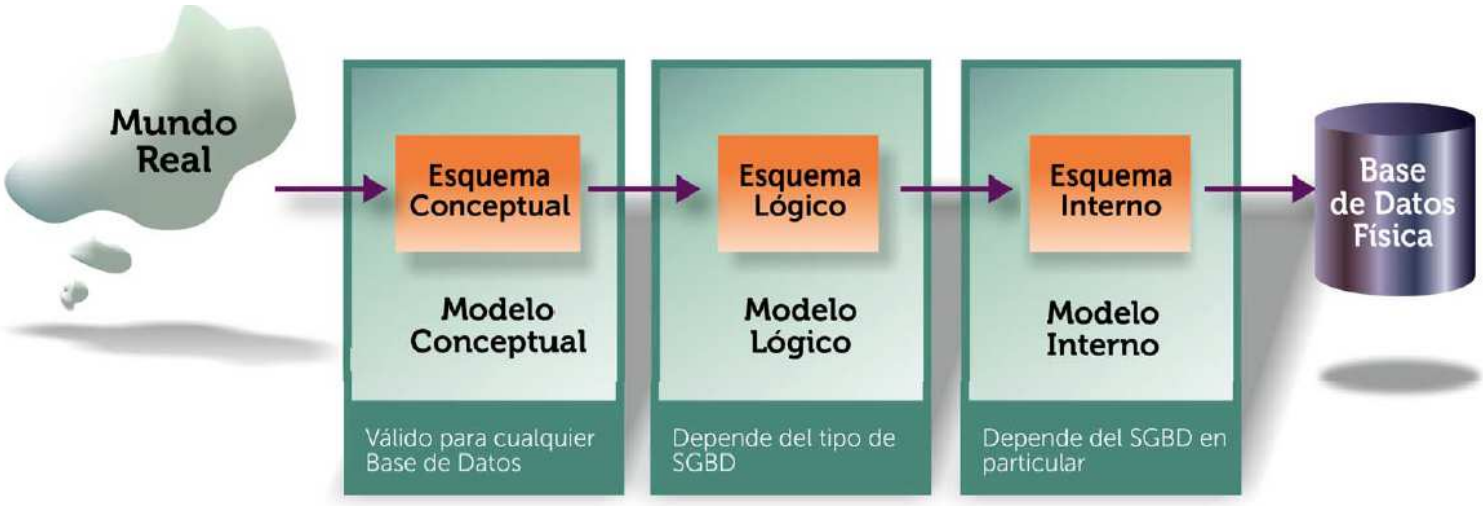
SGBD de tres o más capas: Es una estructura de tipo cliente/servidor, pero en la que hay al menos una capa intermedia entre las dos. Esa capa se suele encargar de procesar las peticiones y enviarlas al SGBD con el que se comunica. Un ejemplo habitual es que la tercer capa sea un **servidor web** que

evita que el cliente se conecte directamente al SGBD. Ese servidor web se encarga de traducir lo que el cliente realiza a una forma entendible por la base de datos.
Esta forma de trabajar permite además que para usar una base de datos, baste un simple navegador al cliente.
Puede haber más capas con otros fines. Estas capas siempre están entre el cliente y el servidor.

4. Tipos de Sistemas Gestores de Bases de Datos

Introducción

Como se ha visto en los apartados anteriores, resulta que cada SGBD puede utilizar un modelo diferente para los datos. Por lo que hay modelos conceptuales diferentes según que SGBD utilicemos.



Modelos de datos utilizados en el desarrollo de una BD

No obstante existen modelos lógicos comunes, ya que hay SGBD de diferentes tipos. En la realidad el modelo ANSI se modifica para que existan dos modelos internos: el modelo lógico (referido a cualquier SGBD de ese tipo) y el modelo propiamente interno (aplicable sólo a un SGBD en particular). De hecho en la práctica al definir las bases de datos desde el mundo real hasta llegar a los datos físicos se pasa por los esquemas señalados en la Ilustración anterior.
Por lo tanto la diferencia entre los distintos SGBD está en que proporcionan diferentes modelos lógicos.

Diferencias entre el modelo lógico y el conceptual

El modelo conceptual es independiente del DBMS que se vaya a utilizar. El lógico depende de un **tipo** de SGBD en particular.

El modelo lógico está más cerca del modelo físico, el que utiliza internamente el ordenador.

El modelo conceptual es el más cercano al usuario, el lógico es el encargado de establecer el paso entre el modelo conceptual y el modelo físico del sistema.

Algunos ejemplos de modelos conceptuales son:

- **Modelo Entidad Relación**
- **Modelo RM/T**
- **Modelo UML**

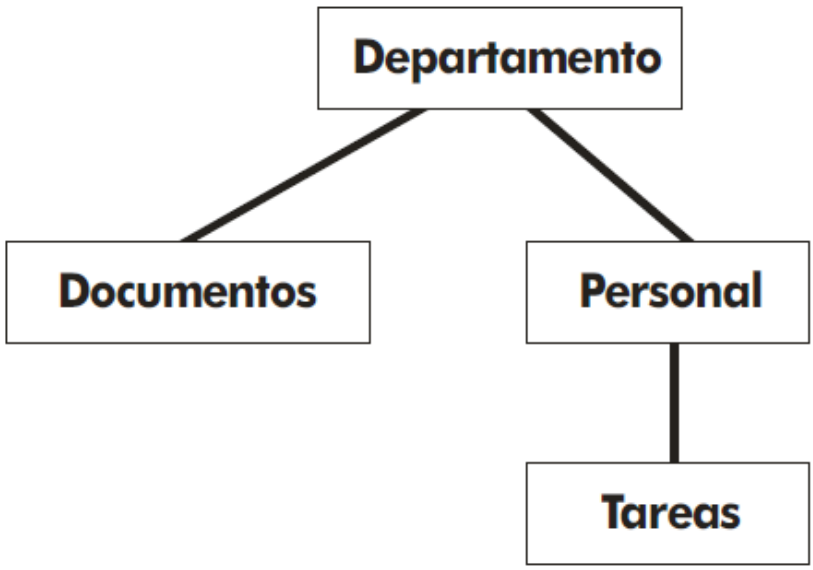
Ejemplos de modelos lógicos son:

- **Modelo relacional**
- **Modelo Codasyl**
- **Modelo Jerárquico**

A continuación se comentarán los modelos lógicos más importantes.

4.1 Modelo jerárquico.

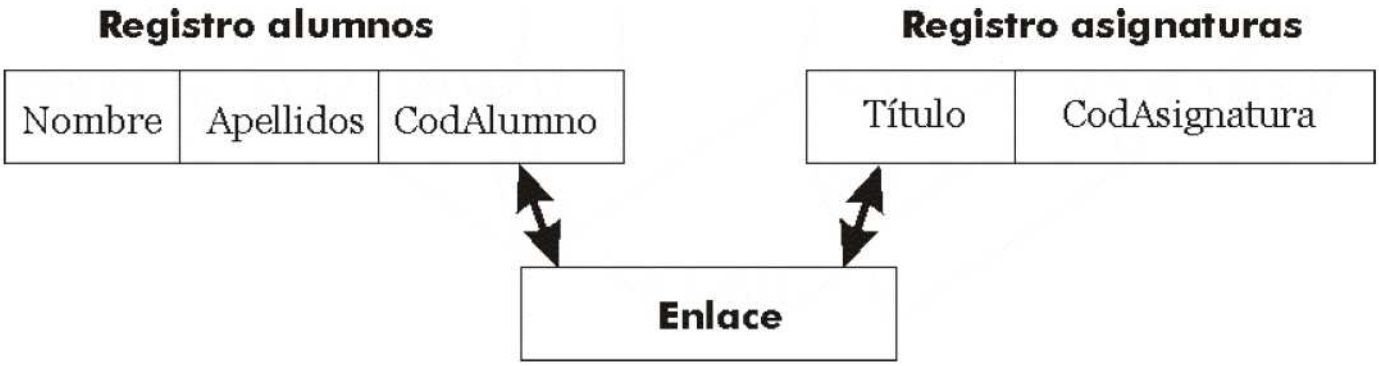
Era utilizado por los primeros SGBD, desde que IBM lo definió para su IMS (***Information Management System,*** Sistema Administrador de Información) en 1970. Se le llama también modelo en árbol debido a que utiliza una estructura en árbol para organizar los datos.
La información se organiza con un jerarquía en la que la relación entre las entidades de este modelo siempre es del tipo **padre / hijo**. De esta forma hay una serie de nodos que contendrán atributos y que se relacionarán con nodos hijos de forma que puede haber más de un hijo para el mismo padre (pero un hijo sólo tiene un padre).
Los datos de este modelo se almacenan en estructuras lógicas llamadas **segmentos**. Los segmentos se relacionan entre sí utilizando **arcos.**
La forma visual de este modelo es de árbol invertido, en la parte superior están los padres y en la inferior los hijos.



Este esquema está en absoluto desuso ya que no es válido para modelar la mayoría de problemas de bases de datos.

4.2 Modelo en red (Codasyl).

Es un modelo que ha tenido una gran aceptación (aunque apenas se utiliza actualmente). En especial se hizo popular la forma definida por Codasyl a principios de los 70 que se ha convertido en el modelo en red más utilizado.
El modelo en red organiza la información en **registros** (también llamados **nodos**) y **enlaces**. En los registros se almacenan los datos, mientras que los enlaces permiten relacionar estos datos. Las bases de datos en red son parecidas a las jerárquicas sólo que en ellas puede haber más de un padre.
En este modelo se pueden representar perfectamente cualquier tipo de relación entre los datos (aunque el Codasyl restringía un poco las relaciones posibles), pero hace muy complicado su manejo.



4.3 Modelo relacional.

En este modelo los datos se organizan en tablas cuyos datos se relacionan. Es el modelo más popular y se describe con más detalle en los temas siguientes que veremos.

4.4 Modelo de bases de datos orientadas a objetos.

Desde la aparición de la programación orientada a objetos (**POO** u **OOP**) se empezó a pensar en bases de datos adaptadas a estos lenguajes. La programación orientada a objetos permite cohesionar datos y procedimientos, haciendo que se diseñen estructuras que poseen datos (**atributos**) en las que se definen los procedimientos (**operaciones**) que pueden realizar con los datos. En las bases orientadas a objetos se utiliza esta misma idea.
A través de este concepto se intenta que estas bases de datos consigan arreglar las limitaciones de las relacionales. Por ejemplo el problema de la herencia (el hecho de que no se puedan realizar relaciones de herencia entre las tablas), tipos definidos por el usuario, disparadores (triggers) almacenables en la base de datos, soporte multimedia...
Se supone que son las bases de datos de tercera generación (la primera fue las bases de datos en red y la segunda las relacionales), lo que significa que el futuro parece estar a favor de estas bases de datos. Pero siguen sin reemplazar a las relacionales, aunque son el tipo de base de datos que más está creciendo en los últimos años.

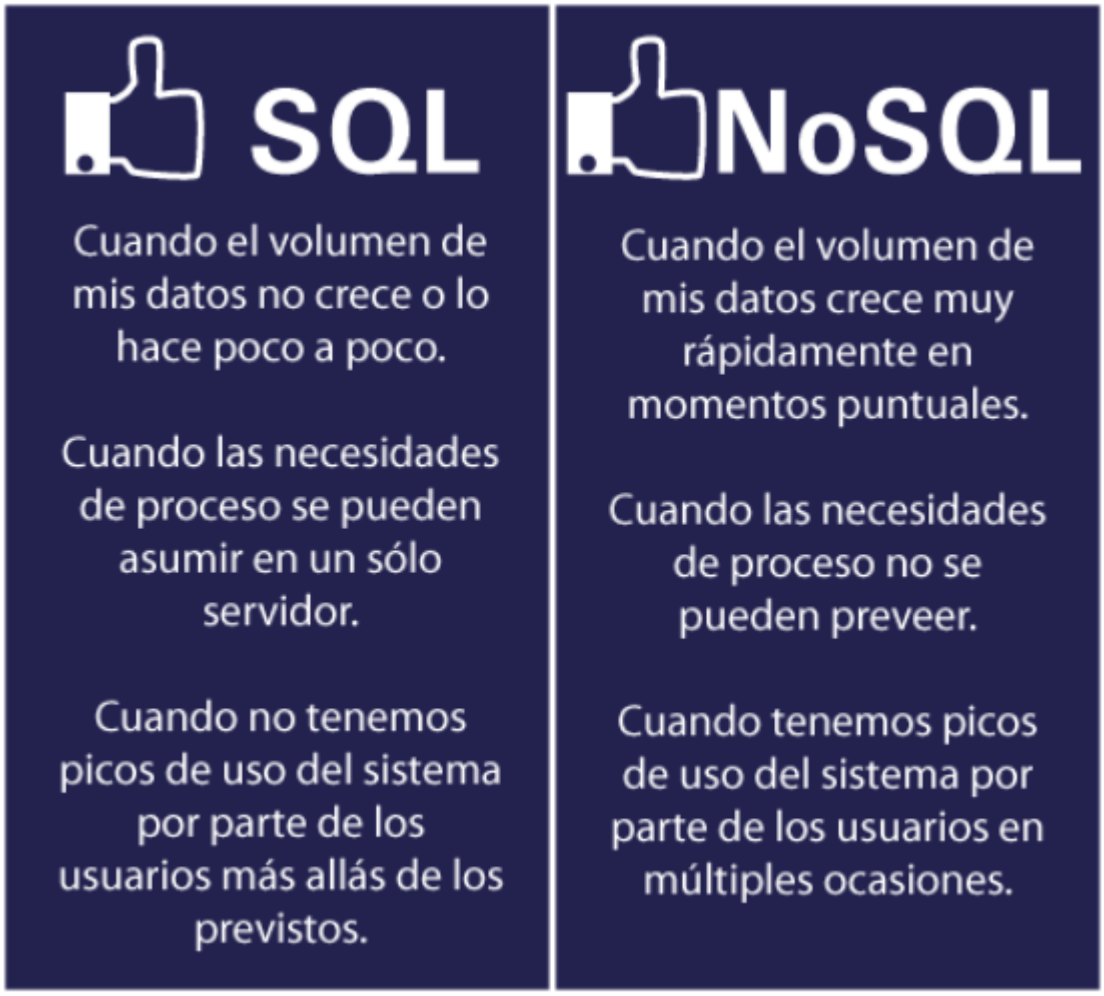
Su modelo conceptual se suele diseñar en **UML** y el lógico actualmente en **ODMG** (***Object Data Management Group,*** grupo de administración de objetos de datos, organismo que intenta crear estándares para este modelo).

4.5 Bases de datos objeto-relacionales.

Tratan de ser un híbrido entre el modelo relacional y el orientado a objetos. El problema de las bases de datos orientadas a objetos es que requieren reinvertir capital y esfuerzos de nuevo para convertir las bases de datos relacionales en bases de datos orientadas a objetos. En las bases de datos objeto relacionales se intenta conseguir una compatibilidad relacional dando la posibilidad de integrar mejoras de la orientación a objetos.
Estas bases de datos se basan en el estándar **SQL 99**. En ese estándar se añade a las bases relacionales la posibilidad de almacenar procedimientos de usuario, triggers, tipos definidos por el usuario, consultas recursivas, bases de datos OLAP, tipos LOB,...
Las últimas versiones de la mayoría de las clásicas grandes bases de datos relacionales (**Oracle**, **SQL Server**, **Informix**, ...) son objeto relacionales.

4.6 Bases de datos NoSQL.

Bajo este nombre se agrupan las bases de datos (con arquitecturas muy diversas) pensadas para grabar los datos de manera veloz para así poder atender a miles y miles de peticiones. Es decir, es el modelo de las bases de datos que se utilizan en los grandes servicios de Internet (como **twitter**, **Facebook**, **Amazon**,...).
La idea es que los datos apenas necesitan validarse y relacionarse y lo importante es la disponibilidad de la propia base de datos. El nombre NoSQL, hace referencia a que este modelo de bases de datos rompe con el lenguaje SQL (el lenguaje de las bases de datos relacionales, las bases de datos dominantes hasta la actualidad) para poder manipular los datos con lenguajes de otro tipo.


SQL
Cuando el volumen de mis datos no crece o lo hace poco a poco.
Cuando las necesidades de proceso se pueden asumir en un sólo servidor.
Cuando no tenemos picos de uso del sistema por parte de los usuarios más allás de los previstos.
NoSQL
Cuando el volumen de mis datos crece muy rápidamente en momentos puntuales.
Cuando las necesidades de proceso no se pueden preveer.
Cuando tenemos picos de uso del sistema por parte de los usuarios en múltiples ocasiones.