UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ARQUITECTURA DE DATOS

CÁTEDRA: MICROPROCESADORES Y ARQUITECTURA DEL COMPUTADOR

CATEDRÁTICO: ING. AYLAS MIGUEL LUIS

ALUMNOS:

- DELGADILLO PEREZ MAIQUER
- HUIZA LOPEZ EDITH
- LUCAS CHANCHA JUDITH
- MARCELO CASTRO MELISSA

SEMESTRE: VI

HUANCAYO - PERÚ

2017

Este presente trabajo está dedicado a nuestros padres quienes nos han apoyado incondicionalmente. También está presente monografía está dedicada a todos que con su trabajo creador contribuyen a la grandeza y prestigio de nuestra querida Facultad.

ÍNDICE

Pág.

I. INTRODUCCIÓN

Organizar la información en base de una rápida recuperación es un factor determinante en muchos sectores de la actividad, sin olvidar la necesidad tanto en el ámbito de la decisión, como el consultar y analizar siempre más información, basta por ejemplo con pensar en el problema cotidiano de que administra una biblioteca o un almacén .De ahí la utilidad de organizar la información en estructuras fácilmente administrable y que aseguren una elevada velocidad de acceso

En la actualidad, las bases de datos tienen una importancia decisiva en la totalidad de las áreas de aplicación de la informática, como la ingeniería, la medicina, la educación, la biblioteconomía, los negocios, etc. Esto ha fomentado el desarrollo de una gran cantidad de conceptos y técnicas para la gestión eficiente de los datos.

Las bases de datos pueden tener cualquier tamaño y complejidad. Cuando la cantidad de información es grande y las relaciones entre los diferentes datos son muchas, es necesario organizar y controlar toda esta información almacenada, para que los usuarios puedan buscar, obtener y actualizar los datos cuando les sea necesario. Una base de datos puede ser creada y mantenida de forma manual (como el catálogo de fichas de una biblioteca), o bien estar informatizada .En este último caso, la creación y mantenimiento de la base de datos puede realizarse mediante un conjunto de programas de aplicación diseñados específicamente para dichas tareas, o bien mediante un sistema de gestión de bases de datos (SGBD).

En este informe trataremos de dar una visión completa de los conceptos ya mencionados. Así pues, el presente trabajo trata de proporcionar una visión completa de los aspectos implicados en el trabajo con bases de datos. Aunque no profundizaremos en algunos temas muy especializados, profundizaremos técnicamente en los temas más necesarios.

II. MARCO TEÓRICO

2.1 BASE DE DATOS

La arquitectura de base de datos comienza precisamente con la definición de la base de datos, el cual se entiende un conjunto de datos estructurados y permanentes agrupados por su homogeneidad y relacionados entre ellos, organizados con la mínima redundancia para ser usados en aplicaciones diversas, de modo controlado.

Una base de datos representa algunos aspectos del mundo real, aquellos que le interesan al usuario. Y que almacena datos con un propósito específico. Con la palabra "datos" se hace referencia a hechos conocidos que pueden registrarse, como ser números telefónicos, direcciones, nombres, etc.

Concepto de Datos:

Datos son los hechos que describen sucesos y entidades. Datos es una palabra en plural que se refiere a más de un hecho. A un hecho simple se le denomina "data-ítem" o elemento de dato.

Los datos son comunicados por varios tipos de símbolos tales como las letras del alfabeto, números, movimientos de labios, puntos y rayas, señales con la mano, dibujos, etc.

Lo importante es considerar que estos símbolos se pueden ordenar y reordenar de forma utilizable y se les denomina información.

Los datos son símbolos que describen condiciones, hechos, situaciones o valores. Los datos se caracterizan por no contener ninguna información. Un dato puede significar un número, una letra, un signo ortográfico o cualquier símbolo que represente una cantidad, una medida, una palabra o una descripción.

La importancia de los datos está en su capacidad de asociarse dentro de un contexto para convertirse en información. Por si mismos los datos no tienen capacidad de comunicar un significado y por tanto no pueden afectar el comportamiento de quien los recibe. Para ser útiles, los datos deben convertirse en información para ofrecer un significado, conocimiento, ideas o conclusiones.

Datos de una DB (base de datos) se refiere a archivos, bases de datos, documentos de texto, imágenes y, voz y video codificados en forma digital.

Concepto de Información:

La información no es un dato conjunto cualquiera de ellos. Es más bien una colección de hechos significativos y pertinentes, para el organismo u organización, que los percibe.

Es común que los términos datos e información se tomen incorrectamente como sinónimos. Recordemos que en una DB puede ser lo que para algunos usuarios es información sin embargo, esta deberá considerarse siempre como datos hasta que estos hayan salido del sistema a través de un SGBD.

2.1.1 CARACTERÍSTICAS

En la práctica una base de datos es un conjunto de archivos, convenientemente ordenados, que debe responder a las siguientes características:

- Estar completamente integrado
- Asegurar la velocidad de acceso a la información
- Permitir un acceso competente a la información
- Asegurar la privacidad a la información
- Asegurar la reconversión de los datos en caso de mal funcionamiento
- Representa algún aspecto del mundo real, llamado minimundo o universo de discurso (UdD) del cual provienen los datos. Los cambios en el minimundo se reflejan en la base de datos.
- Es un conjunto de datos lógicamente coherente, con significado implícito. Un montón de datos sin relación entre sí, agrupados de forma aleatoria, no se considera una base de datos.
- Toda base de datos se diseña, se crea y se carga con datos, con un objetivo determinado, y está dirigida a un grupo de usuarios, interesados en el contenido y en el uso de la base de datos.

2.1.2 APLICACIONES

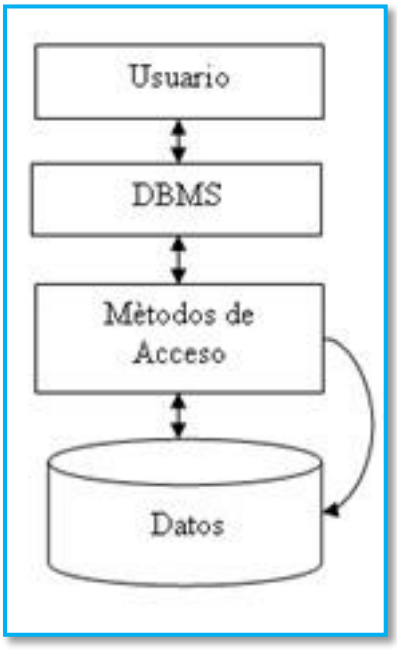
Las Bases de Datos y el sistema administrador resultan ser la columna vertebral de cualquier Empresa, siendo esta una unidad económico-social, integrada por elementos humanos, materiales y técnicos, que tiene por objeto obtener un resultado a través de su participación en la sociedad, con o sin afán de lucro., como pueden ser:

- ✓ Industrias manufactureras,
- ✓ Hospitales,
- ✓ Bancos,
- ✓ Escuelas,
- ✓ Instituciones Gubernamentales,
- ✓ Etc.

Donde para operar se deben tener una gran cantidad de datos como:

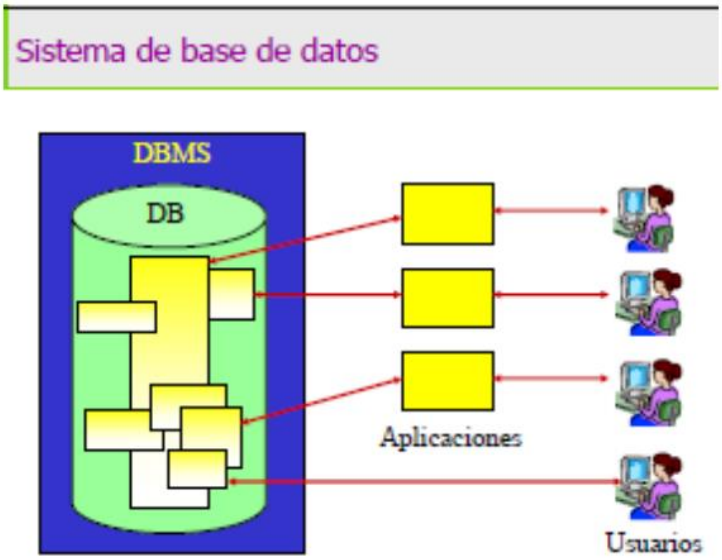
- o Datos de producción,
- o Información de pacientes ,
- o Cuentas contables,
- o Datos de alumnos y profesores,
- o Censos de población y de recursos,
- o Etc.

2.1.3 COMPONENTES DE UNA BASE DE DATOS



Existen programas denominados sistemas gestores de bases de datos, abreviado SGBD (del inglés database management system o DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas; También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.



En el diagrama anterior se percibe claramente como la DB y DBMS son entes separados aunque interrelacionados.

2.1.4 MODELOS DE BASES DE DATOS

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guardan los datos), así como de los métodos para almacenar y recuperar datos de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

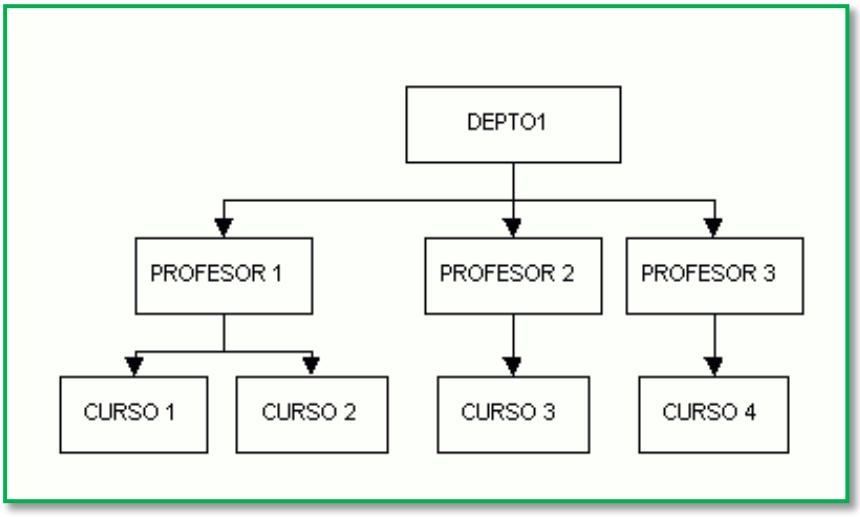
Algunos modelos con frecuencia utilizados en las bases de datos son:

2.1.4.1 Bases de datos jerárquicas

En este modelo los datos se organizan en forma de árbol invertido (algunos dicen raíz), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

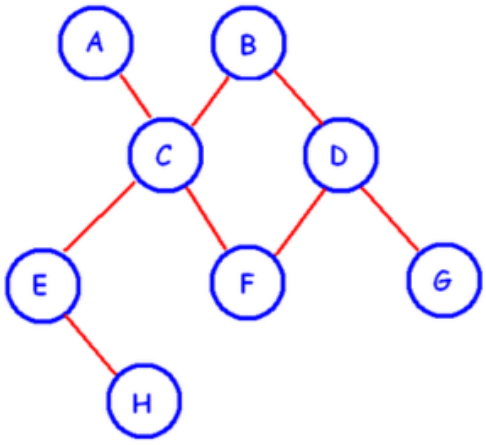
Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.



Ejemplo de un modelo de una base de datos jerárquica

2.1.4.2 Base de datos de red

Este es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).



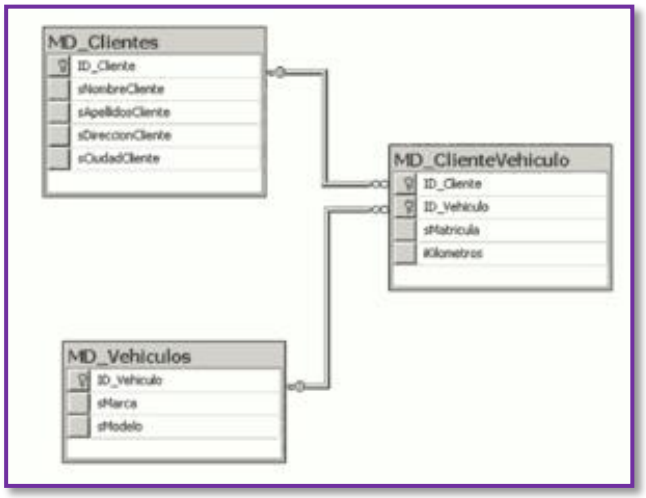
Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

2.1.4.3 Bases de datos relacionales

Este es el modelo utilizado en la actualidad para representar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que esta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

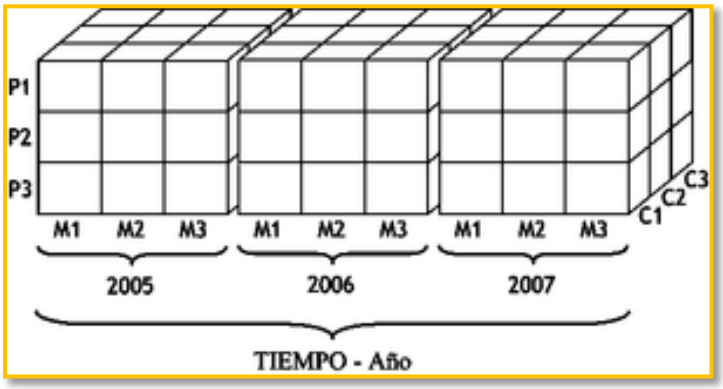
El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.
Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.



Ejemplo de tablas y relaciones.

2.1.4.4 Bases de datos multidimensionales

Son bases de datos ideadas para desarrollar aplicaciones muy concretas, como creación de Cubos OLAP. Básicamente no se diferencian demasiado de las bases de datos relacionales (una tabla en una base de datos relacional podría serlo también en una base de datos multidimensional), la diferencia está más bien a nivel conceptual; en las bases de datos multidimensionales los campos o atributos de una tabla pueden ser de dos tipos, o bien representan dimensiones de la tabla, o bien representan métricas que se desean aprender.



Cubos representando 4 dimensiones en base de datos multidimensional

2.1.4.5 Bases de datos orientadas a objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

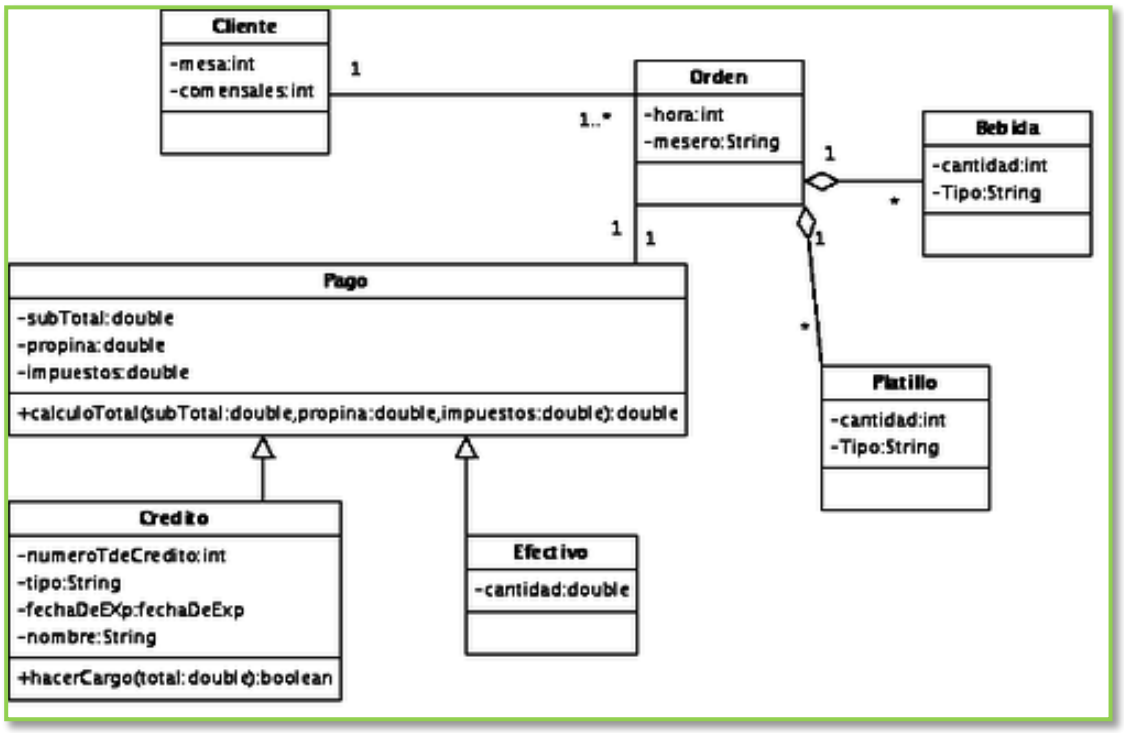
Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- **Encapsulación -** Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia -** Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- **Polimorfismo -** Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos.

Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

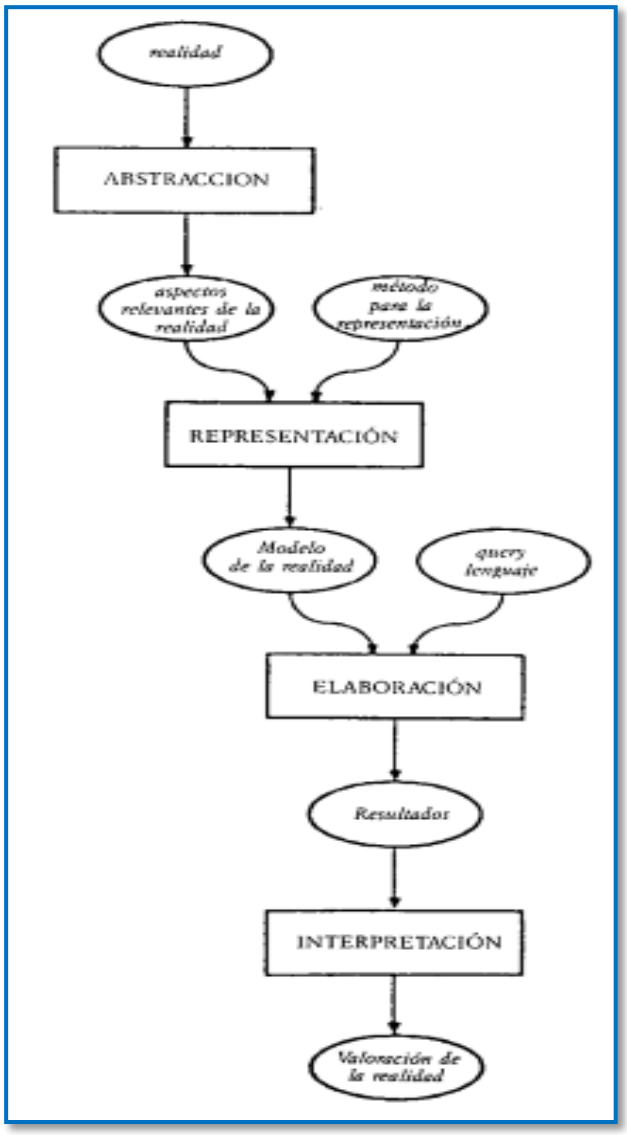
SQL:2003, es el estándar de SQL92 ampliado, soporta los conceptos orientados a objetos y mantiene la compatibilidad con SQL92.



Ejemplo de base de datos conteniendo objetos y herencias

2.1.5 IMPORTANCIA DE SABER MODELAR LA REALIDAD

A menudo cuando buscamos adquirir conocimientos más profundos de una realidad parcialmente conocida, se hace un modelo fijando la atención al aspecto que más interesa profundizar. Del estudio del modelo a menudo es posible obtener la información que nos permita tomar decisiones en función de una finalidad previamente fijada. Para comprender mejor este aspecto clave, escogeremos un ejemplo sencillo tomado de la figura siguiente:



Supongamos que se debe averiguar la aerodinamica del prototipo de un coche;en este caso la realidad es el coche en si mismo.Pero lo q interesa al terminar el estudio de la aerodinamica,es el perfil del coche,eventualmente el material con que se construira la carroceria,descuidando todos los aspectos q no esten relacionados al objetivo que nos proponemos como peso total,el tipo de motor,el interior ,etc.Determinamos los intereses particulares hemos hecho una ABSTRACCION,es decir hemos separado ciertos aspectos de la realidad cognoscitiva de los otros,que sin embargo conocemos.

En efecto ,distintas circunstancias,cada uno de nosotros,frente a una realidad ejercemos una eleccion distinguiendo las caracteristicas que nos interesan y por consiguiente la ignoramos.Una vez desterminado los aspectos significativos de la realidad que se quiere modelar,nace la necesidad de REPRESENTARLA.

Prosiguiendo con el ejemplo del coche,la representacion consistira en la realizacion de una maqueta en la que destacaremos unicamente las caracteristicas relativas al perfil y posiblemente el material utilizado para construir el chasis.

Respecto a la representacion en este momento ya podemos trabajar,osea podemos ejecutar las oportunas ELABORACIONES en la maqueta para tratar de obtener la informacion que necesitamos,el resultado de la elaboracion debe ser llevado al contexto real del cual se ha extraido el modelo;este proceso es en algun sentido la opracion inversa de la abstraccion que llamaremos INTERPRETACION. En efecto,la interpretacion de los resultados de la elaboracion da la pauta para descidir como actuar en la realidad y por consiguiente,en definitiva,dar sentido al mismo modelo.

2.1.6 PROYECCION CONCEPTUAL

En la fase de proyección conceptual de una base de datos, el objetivo principal concierne al análisis de una porción de la realidad y de los acontecimientos que se verifican a ella, que sean los intereses para el entorno y los acontecimientos que se verifican a ella, que sean de interés para el entorno de la aplicación en el cual la base de datos debe poder ser utilizada.

El análisis de la realidad puede ser visto en dos direcciones: la primera de forma horizontal, añade a la realidad considerada en su extensión, por consiguientes a todos los aspectos que lo componen. La segunda, que llamaremos vertical, concierne en cambio a nuestra relación con la realidad considerada: lo que definitivamente de ella nos interesa.

Procesos de análisis de datos:

- UNIFICACION
- CLASIFICACION
- GENERALIZACION

A. **UNIFICACION:** Consiste en reducir varios objetos en una sola unidad: una cabeza, dos brazos, dos piernas y un tronco forman la estructura del hombre.

B. **CLASIFICACION:** permite dividir los individuos en clases y por consiguiente dar una cierta estructura a la realidad.

LA CLASIFICACION SE PUEDE HACER POR :

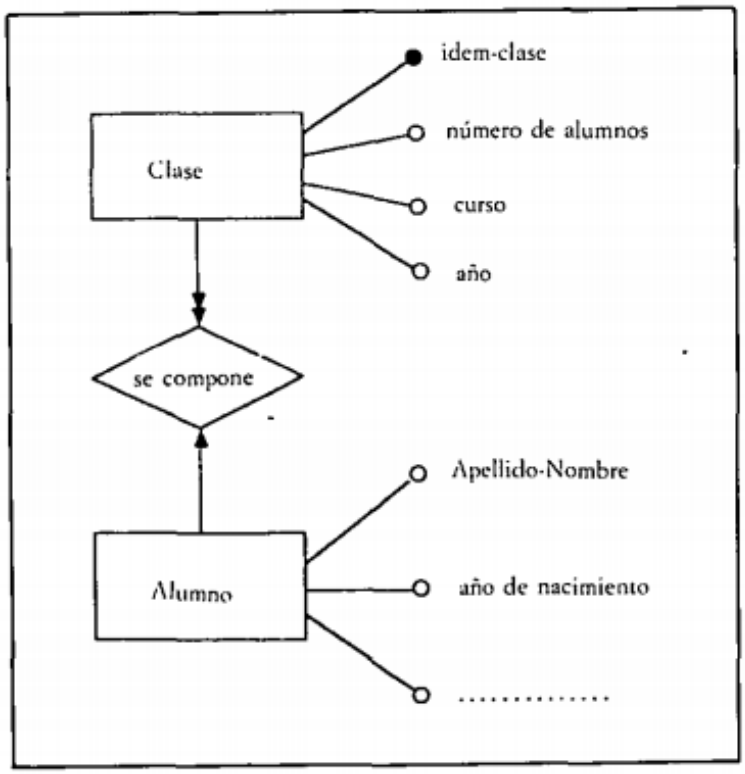
- ✓ Extensión .- se da explícitamente la lista de los individuos que pertenecen a la clase.

- ✓ Designación.- se determina la clase a través de la cual se especifica una propiedad que especifica a sus elementos

Se observa que el que en la base de datos la clase siempre se la extensión, en cuanto el archivo contiene a los mismos datos de varios individuos que componen la clase; al revés, la busque de la información está hecha habitualmente por designación, o sea especificando una propiedad que caracteriza a los individuos que buscamos.

C. **GENERALIZACION:** La propiedad más importante de la generación es la capacidad de transmisión de características a los individuos que buscamos.

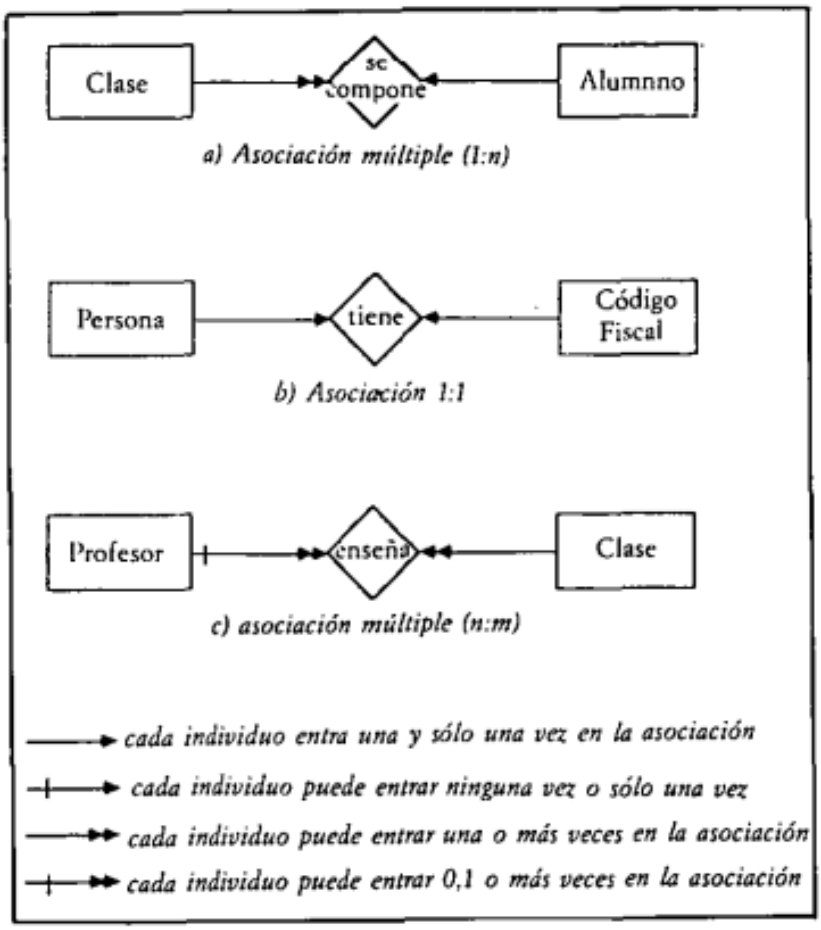
Por lo tanto en cada análisis, la realidad se presenta como una colección de entidades conectadas entre ellas por una red de asociación. En el esquema conceptual, ya sea por entidad como por asociación, vienen especificadas los atributos, es decir, la propiedad i las características que nos interesan para el análisis de la porción de la realidad a la cual se refiere.



LA ASOCIACION:

Es un aspecto relevante del esquema conceptual atañe a la tipología de asociación entre entidades. Las asociaciones son la red de conexionado de una base de datos y solo consideraremos, aquellas que pueden ayudar con una contribución efectiva en la instrumentación.

Tipos:



MODELO DE BASE DE DATOS

Es un tipo de modelo de datos que determina la estructura lógica de una base de datos y de manera fundamental determina el modo de almacenar, organizar y manipular los datos.

a) Modelo reticular:

El modelo de red expande la estructura jerárquica, permitiendo relaciones N:N en una estructura tipo árbol que permite múltiples padres. Antes de la llegada del modelo relacional, el modelo en red era el más popular para las bases de

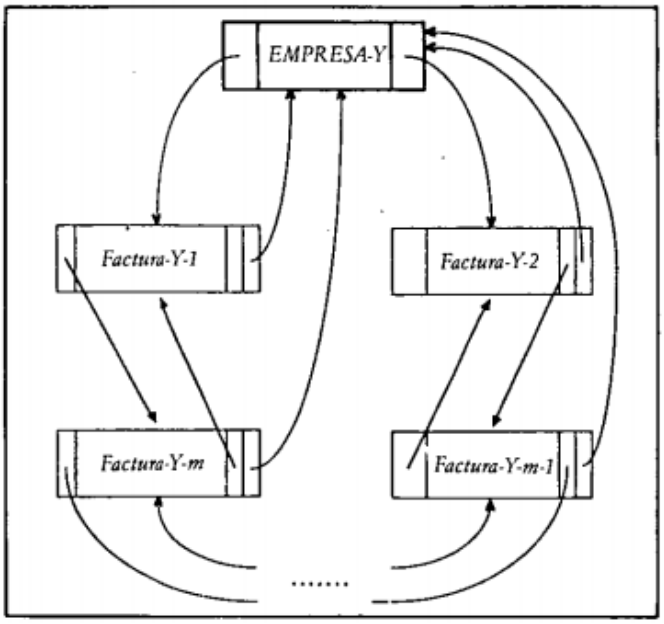
datos. Este modelo de red (definido por la especificación CODASYL) organiza datos que usan en dos construcciones básicas, registros y conjuntos. Los registros contienen campos que puede estar organizados jerárquicamente, como en el lenguaje COBOL. Los conjuntos definen relaciones N:N entre registros: varios propietarios, varios miembros. Un registro puede ser un propietario de varios conjuntos, y miembro en cualquier número de conjuntos.

El modelo en red es una generalización del modelo jerárquico, en tanto está construido sobre el concepto de múltiples ramas (estructuras de nivel inferior) emanando de uno o varios nodos (estructuras de nivel alto), mientras el modelo se diferencia del modelo jerárquico en que las ramas pueden estar unidas a múltiples nodos. El modelo de red es capaz de representar la redundancia en datos de una manera más eficiente que en el modelo jerárquico.

Las operaciones del modelo de red se realizan por de navegación: un programa mantiene la posición actual, y navega entre registros siguiendo las relaciones entre ellos. Los registros también pueden ser localizados por valores claves.

Aunque no es una característica esencial del modelo, las bases de datos en red implementan sus relaciones mediante punteros directos al disco. Esto da una velocidad de recuperación excelente, pero penaliza las operaciones de carga y reorganización.

Entre los SGBS más populares que tienen arquitectura en red se encuentran Total e IDMS. IDMS logró una importante base de usuarios; en 1980 adoptó el modelo relacional y SQL, manteniendo además sus herramientas y lenguajes originales.



La mayoría de base de datos orientadas (introducidas en 1990) usan el concepto de navegación para proporcionar acceso rápido entre objetos en una red.

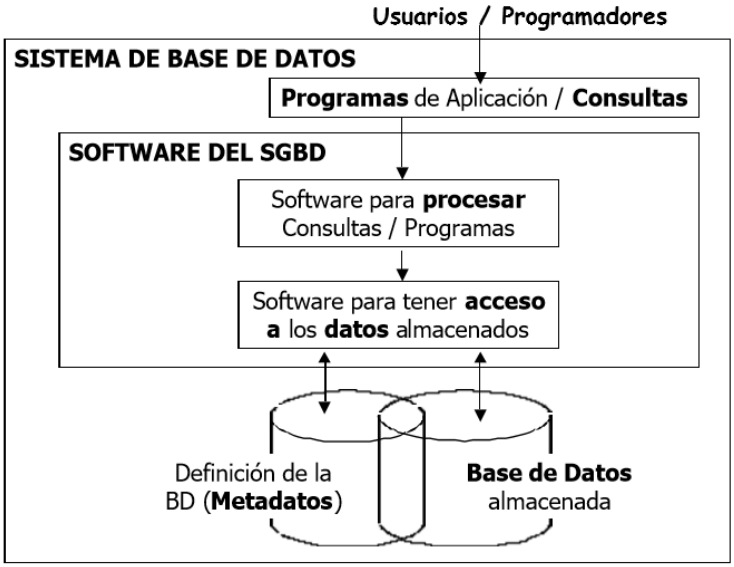
2.2 SISTEMA DE GESTION DE BASE DE DATOS

Un sistema de gestión de bases de datos (SGBD, o en inglés database management system DBMS) es un conjunto de programas que permite a los usuarios crear y mantener una base de datos. Es un sistema software de propósito general, que facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones.

Definir una base de datos consiste en especificar los tipos de los datos, las estructuras de los datos y las restricciones de los datos.

Construir una BD es el proceso de almacenar los datos en algún medio de almacenamiento controlado por el SGBD.

Manipular la BD es:
a) consultar los datos para obtener cierta información,
b) actualizar la base de datos (modificar o eliminar datos, o introducir nuevos) para reflejar los cambios ocurridos en el minimundo, o
c) generar informes a partir de los datos almacenados.



El objetivo principal de un SGBD es proporcionar un entorno a la vez práctico y eficiente a la hora de almacenar y recuperar la información de la base de datos. No es obligatorio utilizar software de SGBD de propósito general para implementar una base de datos informatizada. Sería posible construir un conjunto de programas propio para crear y

mantener la base de datos, es decir, crear software de SGBD de propósito específico. Al conjunto formado por la base de datos y el software (tanto del SGBD como el de los programas de aplicación) lo llamaremos sistema de bases de datos (SBD).

Características del enfoque de bases de datos:

Existen diferentes enfoques en cuanto a los SGBD, Y antes de la llegada de su llegada, las empresas almacenaban su información empleando el enfoque clásico de procesamiento de ficheros, en el cual la definición e implementación de los ficheros necesarios para una aplicación específica se realiza como parte de la programación de la aplicación. Las características que distinguen el enfoque de bases de datos del clásico de procesamiento de ficheros son varias:

- **Naturaleza autodescriptiva de los sistemas de bases de datos**

Además de la base de datos en sí misma, el sistema contiene una descripción completa de la base de datos. Esta descripción se almacena en el catálogo del sistema y consiste en información sobre la estructura de cada fichero, el tipo y formato de almacenamiento de cada elemento y las restricciones que se aplican a los datos. La información contenida en el catálogo se llama metainformación1. El catálogo es usado por el software del SGBD y a veces, por usuarios que necesitan información sobre la estructura de la BD. El catálogo es necesario porque el SGBD no está escrito para una determinada aplicación, sino para cualquier aplicación de bases de datos, de forma que el SGBD tiene que consultar el catálogo para conocer la estructura de los ficheros de cada BD (la de una universidad, o la de un banco). En cambio, en el procesamiento de ficheros clásico, la definición de los datos es parte del código de los programas de aplicación, así que un programa "sólo puede trabajar con una base de datos" específica, cuya estructura se describe en el propio código (un ejemplo es un programa escrito en lenguaje Pascal que contenga declaraciones de estructuras de datos).

- **Separación entre los programas y los datos**

En el procesamiento de ficheros tradicional, como ya hemos indicado, la estructura de los ficheros de datos está integrada en los programas, así que un cambio en la estructura de un fichero puede implicar la modificación de todos los programas que acceden al mismo. En cambio, los programas de acceso del SGBD se escriben para que sean independientes de cómo y dónde estén almacenados los datos. La estructura de los ficheros se guarda en el catálogo del SGBD, separada de los programas de acceso. Esta propiedad es la independencia entre programas y datos.

Por ejemplo, es posible escribir un programa que sólo pueda tener acceso a registros de ACTOR de 62 caracteres de longitud. Si añadimos otro dato (campo) a cada

registro de ACTOR, por ejemplo el lugar de nacimiento, ese programa no podrá seguir funcionando: habrá que modificarlo. Sin embargo, en un entorno de SGBD, basta modificar la descripción en el catálogo de los registros de ACTOR, y no se cambia ningún programa. La próxima vez que el programa del SGBD consulte el catálogo, tendrá acceso a la nueva estructura de los registros de ACTOR y la utilizará de forma adecuada.

La flexibilidad que proporciona esta independencia entre programas y datos es crucial para conseguir, sin esfuerzo excesivo, la adaptación continua del sistema de información a la evolución de las organizaciones. La característica que hace posible la independencia entre programas y datos es la abstracción de los datos proporcionada por el SGBD.

- **Datos compartidos y procesamiento de transacciones multiusuario**

Un SGBD multiusuario, como su propio nombre indica, debe permitir el acceso simultáneo a la base de datos por parte de varios usuarios. Esto es imprescindible si los datos de diversas aplicaciones se deben integrar y mantener en una sola base de datos. El SGBD debe incluir software de control de concurrencia para asegurar que, cuando varios usuarios intenten actualizar los mismos datos, lo hagan de manera controlada, de forma que el resultado final sea correcto.

Un ejemplo sería el caso de varios encargados de realizar reservas de asientos numerados en una sala de cine: el SGBD debe asegurar que sólo un empleado tenga acceso a un asiento específico en un momento dado, para asignarlo a un cliente, y que en cuanto un empleado reserve un asiento, los demás lo vean inmediatamente. Cada operación de reserva sería una transacción.

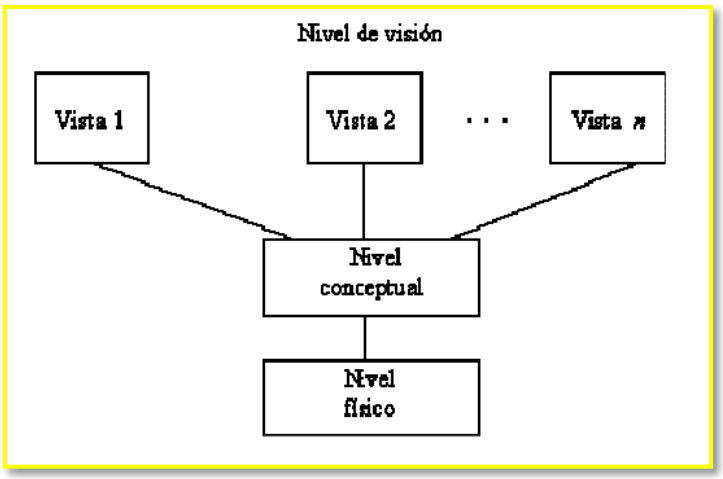
Una función fundamental del SGBD es asegurar que las transacciones concurrentes se realizan de manera correcta, sin interferencias entre ellas.

- **Soporte de múltiples vistas de los datos**

Como ya hemos indicado, un sistema de BD suele tener muchos usuarios. Algunos de ellos no deberían poder acceder a todos los datos (por cuestiones de seguridad), o simplemente no necesitan acceder más que a una parte de ellos.

Por ejemplo, en un sistema de gestión de una productora de películas de cine, el personal de nóminas necesita ver sólo la parte de la base de datos que contiene información acerca de los empleados de la productora, y no necesita saber nada acerca de la recaudación de las películas proyectadas en diferentes salas de cine. Por tanto, cada usuario (o grupo de usuarios) puede necesitar una vista o perspectiva diferente de la BD.

Una vista puede ser un subconjunto de la base de datos, y puede contener datos virtuales (no almacenados, sino que se derivan o calculan a partir de otros datos). Los usuarios normalmente no necesitan saber (de hecho, no lo saben) si ven y utilizan todos o sólo parte de los datos, y tampoco si son datos derivados o no.



2.2.1 CARACTERÍSTICAS FUNDAMENTALES DE UN SISTEMA DE GESTIÓN DE BASE DE DATOS

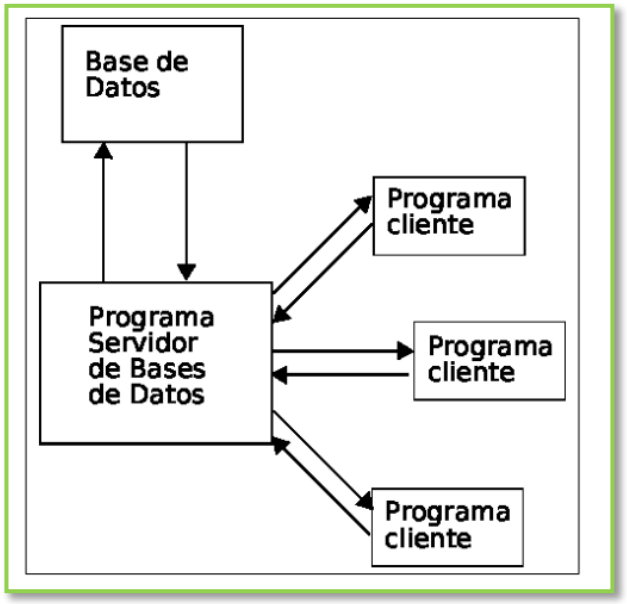
Un SGBD permite el almacenamiento, manipulación y consulta de datos pertenecientes a una base de datos organizada en uno o varios ficheros. En el modelo más extendido (base de datos relacional) la base de datos consiste, de cara al usuario, en un conjunto de tablas entre las que se establecen relaciones. A pesar de sus semejanzas (ambos manejan conjuntos de tablas) existen una serie de diferencias fundamentales entre un SGBD y un programa de hoja de cálculo, la principal es que un SGBD permite:

- ✓ El método de almacenamiento y el programa que gestiona los datos (servidor) son independientes del programa desde el que se lanzan las consultas (cliente).

- ✓ En lugar de primarse la visualización de toda la información, el ojetivo fundamental es permitir consultas complejas, cuya resolución está optimizada, expresadas mediante un lenguaje formal.

- ✓ El almacenamiento de los datos se hace de forma eficiente aunque oculta para el usuario y normalmente tiene, al contrario de lo que ocurre con las hojas de cálculo, poco que ver con la estructura con la que los datos se presentan al usuario.

- ✓ El acceso concurrente de múltiples usuarios autorizados a los datos, realizando operaciones de actualización y consulta de los mismos garantizando la ausencia de problemas de seguridad (debidos a accesos no autorizados) o integridad (pérdida de datos por el intento de varios usuarios de acceder al mismo fichero al mismo tiempo.



Esquema cliente-servidor en una base de datos

2.2.2 BENEFICIOS DE LAS SGBD Y UNA BD

El beneficio de la arquitectura de tres niveles es por la explicación de independencia de datos, definida como la capacidad de modificación del esquema a un nivel de sistema sin tener que modificar el esquema superior de forma inmediata. Lo que más tenemos que destacar en un SGBD lo suficientemente funcional, es lo siguiente:

- Versatilidad en la representación de los datos, ofreciendo las visiones de la información almacenada de todas las formas necesarias.
- Corto tiempo de respuesta y acceso simultaneo a datos.
- Mínima redundancia.
- Simplicidad y Privacidad.
- Seguridad, teniendo la capacidad de proteger los datos ante pérdidas totales o parciales (incendios, accesos no autorizados, uso incorrecto de los datos...).
- Afinación de datos, organizándolos de la forma más óptima para obtener unos rápidos tiempos de respuesta.
- Integridad, otorgando a los datos fiabilidad, frente a fallos de Hardware y Software.

2.2.3 INCONVENIENTES DE LOS SGBD

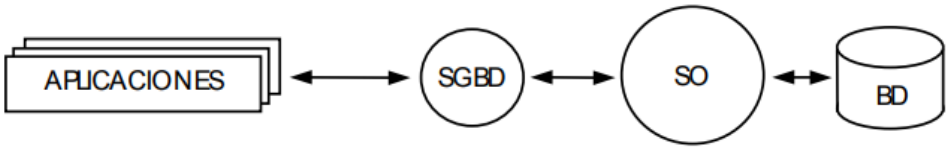
Típicamente, es necesario disponer de una o más personas que administren de la base de datos, en la misma forma en que suele ser necesario en instalaciones de cierto porte disponer de una o más personas que administren los sistemas operativos. Esto puede llegar a incrementar los costos de operación en una empresa. Sin embargo hay que balancear este aspecto con la calidad y confiabilidad del sistema que se obtiene. Si se tienen muy pocos datos que son usados por un único usuario por vez y no hay que realizar consultas complejas sobre los datos, entonces es posible que sea mejor usar una planilla de cálculo.

- **Complejidad:** los software son muy complejos y las personas que vayan a usarlo deben tener conocimiento de las funcionalidades del mismo para poder aprovecharlo al máximo.
- **Tamaño:** la complejidad y la gran cantidad de funciones que tienen hacen que sea un software de gran tamaño, que requiere de gran cantidad de memoria para poder correr.
- **Coste del hardware adicional:** los requisitos de hardware para correr un SGBD por lo general son relativamente altos, por lo que estos equipos pueden llegar a costar gran cantidad de dinero.

2.2.4 ARQUITECTURA DE UN SISTEMA DE GESTIÓN DE BASES DE DATOS

Un SGBD no es, ni más ni menos, que un programa de ordenador que maneja Bases de Datos. Como tal, tiene unas características comunes que se encuentran, implementadas de una forma o de otra, en todos los productos comerciales disponibles en el mercado. Nuestra pretensión es dar una visión general de esas características sin centrarnos en producto alguno. El hecho fundamental que justifica la utilización de un SGBD es que, si antes los programas llamaban directamente al sistema operativo para manejar sus ficheros, ahora es el SGDB es el encargado de facilitar los servicios de acceso de las aplicaciones a los datos.

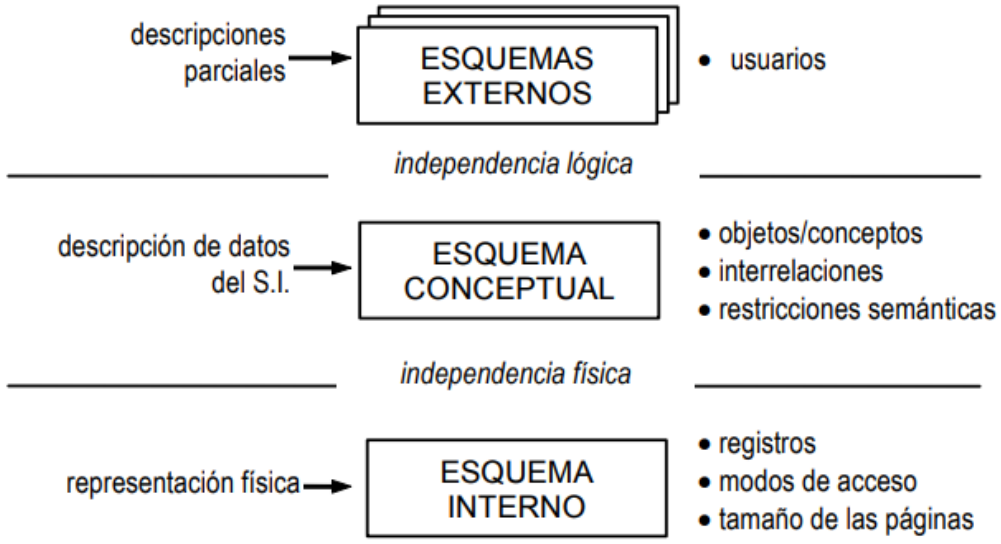
Los programas piden al SGBD ciertos datos de la BD y éste, con la información y herramientas de que dispone, las convierte en operaciones de acceso a los distintos ficheros de la BD, operaciones que son ejecutadas por los métodos de acceso; entendemos por tales el conjunto de rutinas del Sistema Operativo que gestionan al nivel más próximo al hardware el acceso a los ficheros.



Como se observa, el SGBD introduce un nuevo nivel de independencia entre los usuarios y el hardware que no existe en un sistema clásico en el que los programas se comunican directamente con los métodos de acceso. Uno de los objetivos de las BD es la independencia de datos. Para alcanzar dicho objetivo se han hecho distintas propuestas para definir la estructura ideal de un SGBD. El grupo ANSI/SPARC34 (1977) propuso una arquitectura a tres niveles, donde se distinguen tres esquemas de BD:

- Esquema Conceptual (EC)
- Esquema Interno (EI)
- Esquemas Externos (EE)

El EC es la descripción del sistema de información (objetos e interrelaciones) con independencia del SGBD que se vaya a utilizar; el EI es la descripción del EC en términos de representación física (la forma de almacenarlo en el ordenador), y los



EE son las vistas parciales (subconjuntos del conjunto global de información) que tienen los distintos usuarios.

Sin embargo, se encontraron con que no disponían de un Modelo Conceptual general y accesible desde cualquier SGBD que nos permita definir el esquema conceptual. Generalmente, los modelos de datos soportados por los SGBD

comerciales suelen ser demasiado pobres semánticamente, por lo que se prefiere realizar la descripción del sistema de información en algún modelo más expresivo para luego "traducirlo" a un SGBD concreto. La intención es disponer de una primera descripción lo más completa posible e independiente de las herramientas que se vayan a utilizar para gestionarlo, incluso con la posibilidad de que no se vaya a mecanizar mediante un ordenador.

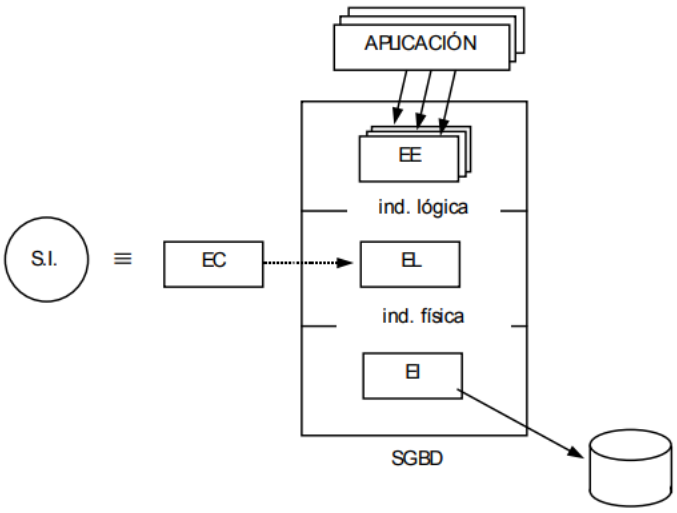
Por esta razón muchos autores prefieren distinguir dos esquemas en lugar del EC propuesto por ANSI/SPARC, lo que la convierte en una arquitectura a cuatro niveles:

- **Esquema Conceptual:** visión desde un punto de vista organizativo, independiente del SGBD que se utilice, e incluso de la utilización o no de sistemas de bases de datos. En este esquema se describe la información de la organización (objetos y relaciones) desde un punto de vista no informático.

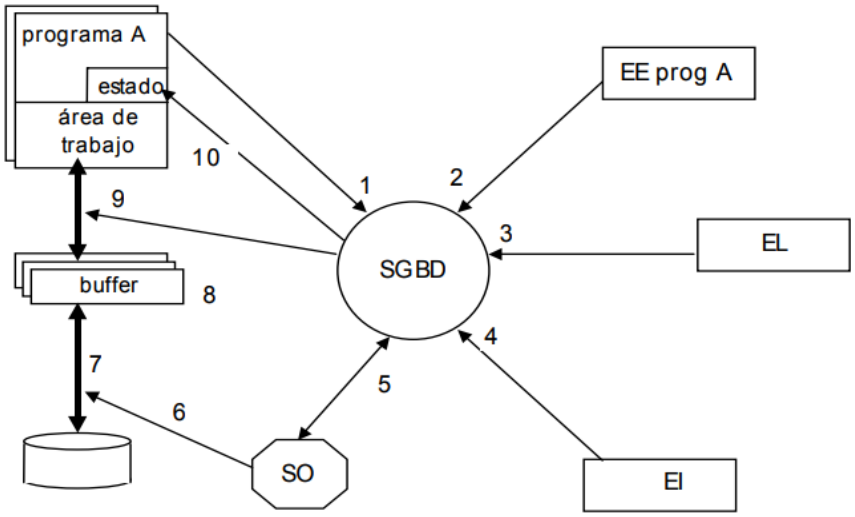
- **Esquema Lógico:** visión expresada en términos de un SGBD concreto, o mejor dicho, de un modelo de datos soportado por un SGBD. En este esquema lógico se representan las entidades y relaciones de acuerdo a las características de dicho modelo, sin entrar todavía en detalles de representación física.

- **Esquema Interno (o Esquema Físico):** descripción de la representación en la memoria externa del ordenador de los datos del esquema lógico, sus interrelaciones y los instrumentos para acceder a ellos.

- **Esquemas Externos:** cada uno de ellos describe los datos y relaciones entre ellos de interés para una aplicación dada. Estos esquemas pueden verse como subconjuntos del Modelo Lógico de la BD.



El SGBD dispone de la definición de cada uno de estos esquemas para satisfacer las peticiones de los programas. Se muestra en este momento una posible interacción entre el SGBD, el Sistema Operativo y los programas de aplicación a la hora de atender una petición de acceso a los datos.



Además de estas operaciones, el SGBD se ocupa de los controles de privacidad e integridad que hayan sido previstos tanto en el esquema lógico como en el esquema externo. Asimismo recoge información para la posible reconstrucción de la base de datos en caso de fallo.

Conviene hacer una aclaración en este punto. La mayoría de los SGBD comerciales trabajan como si únicamente tuvieran un esquema: el lógico. La percepción del usuario es que sólo tiene que definir la estructura de la base de datos en función del modelo de datos que soporta el programa, siendo totalmente transparente para él la organización que utiliza a nivel físico de los datos. Únicamente en sistemas muy grandes, el administrador de la base de datos ha de tomar decisiones sobre cómo se deben guardar los datos en los sistemas de almacenamiento de la máquina, generalmente para aumentar el rendimiento del sistema.

2.2.5 ESTRUCTURA GENERAL DEL SISTEMA DE BASES DE DATOS

Los SGBD son sistemas software muy complejo, compuesto de varios módulos software que se encargan de cada una de las responsabilidades del sistema completo. Algunas de estas funciones las puede proporcionar el sistema operativo (SO), pero en general los sistemas operativos sólo proporcionan los servicios más básicos y los SGBD deben construirse sobre esta base (por esto, en el diseño de un SGBD se debe considerar la interfaz entre el SGBD y el SO).

2.2.5.1 Módulos componentes de un SGBD

Los siguientes son los componentes de procesamiento de consultas:

- El **compilador de consultas** trata cada consulta de alto nivel (escrita en LMD) que se introduce de forma interactiva, analiza su sintaxis, intenta optimizarla (transformarla en otra equivalente pero más eficiente) y genera una llamada al *procesador de consultas* para que la ejecute5.

- El **precompilador de LMD embebido** extrae las sentencias en LMD de un programa escrito en un lenguaje host y las envía al **compilador de LMD**, el cual intenta optimizarlas y las convierte en código objeto (instrucciones de bajo nivel que entiende el *procesador de consultas*) para el acceso a la BD. El resto del programa se envía al compilador del lenguaje host. El código objeto de las sentencias LMD se enlaza (*link*) con el código objeto del resto del programa, formando una *transacción programada* cuyo código ejecutable incluye llamadas al *procesador de consultas* de la base de datos.
- El **compilador (o intérprete) de LDD** procesa las definiciones de esquema escritas en LDD, y almacena las descripciones de los esquemas (metadatos) en el catálogo del SGBD. Otros módulos del SGBD necesitan conocer la información contenida en el catálogo.

- El **procesador de consultas** en tiempo de ejecución se encarga de recibir solicitudes de recuperación o actualización, y las ejecuta sobre la base de datos. El acceso a los datos (a disco) se realiza mediante el *gestor de datos almacenados*.

Los siguientes son los COMPONENTES DE GESTIÓN DE ALMACENAMIENTO, que proporcionan la interfaz entre los datos almacenados y los programas de aplicación y envío de consultas al sistema.

- **Subsistema de control de concurrencia y recuperación** (o **gestor de transacciones**), que...
  - Asegura la consistencia y coherencia de los datos cuando varios usuarios actualizan a la vez la misma información en la BD.
  - Detecta fallos o caídas del sistema, en cuyo caso debe llevar a cabo la restauración de la base de datos a un estado consistente (correcto).

- **Subsistema de integridad y seguridad**, encargado de...
  - Determinar si las actualizaciones de los datos son correctas o por el contrario violan alguna restricción de integridad, en cuyo caso realiza la acción adecuada.
  - Asegurar que se cumplen las restricciones de seguridad en el acceso a la base de datos o a determinados datos.

- **Gestor de datos almacenados y de la memoria intermedia**, que controla el acceso a la información del SGBD almacenada en disco (datos o metadatos). Se encarga de la reserva de espacio de almacenamiento en disco y las estructuras de datos usadas para representar la información en disco.

Este componente puede emplear los servicios básicos del SO para transferir datos de bajo nivel entre el disco y la memoria principal del ordenador. Es el responsable de otros aspectos de transferencia de datos, como por ejemplo el manejo de las áreas de almacenamiento intermedio (*buffers*) en la memoria principal, donde se llevan los datos desde el disco para que después otros módulos del SGBD puedan procesarlos. También se encarga de decidir qué datos tratar en la memoria caché.

Además de los componentes anteriores, se necesitan varias **ESTRUCTURAS DE DATOS** como parte de la implementación física del sistema:

- **Ficheros de datos** en disco, que almacenan la base de datos en sí.
- El **catálogo** del SGBD, que almacena metadatos acerca de la estructura de la base de datos.
- **Estructuras de acceso** (ficheros de índices, por ejemplo), que permiten acceso rápido a elementos de datos que tienen valores particulares.
- **Datos estadísticos** sobre los datos en la base de datos. Suele considerarse contenida en el catálogo.

Esta información es necesaria para seleccionar las formas eficientes de ejecutar una consulta (optimización).

2.2.5.2 Utilidades del sistema de bases de datos

El SGBD es el componente software más importante en un sistema de bases de datos, pero no el único. Existen otras aplicaciones o utilidades que ayudan al ABD a manejar el sistema. Estas utilidades realizan las siguientes funciones:

- ***Utilidades para Carga.*** Se utilizan para cargar ficheros de datos ya existentes (de texto, por ejemplo) en la base de datos. Normalmente se indica el formato de los datos del fichero fuente y el que deben tener en la base de datos destino, y la utilidad de carga (herramienta de conversión) convierte los datos de un formato a otro para almacenarlos en la BD. Esto permite el intercambio de información entre diferentes SGBD comerciales (por ejemplo de *Oracle* a *Access*).

- ***Utilidades para Respaldo.*** Crean una copia de seguridad (*backup*) del contenido de la base de datos.
  Esta copia puede utilizarse para restaurar la BD después de un fallo general del sistema.

- ***Utilidades para Reorganización de ficheros.*** Permiten modificar la organización de los ficheros de la BD, para mejorar el rendimiento. Pueden incluir utilidades para ordenar y comprimir ficheros.

- ***Utilidades para Monitorización*** o vigilancia del rendimiento. Permiten supervisar el uso de la BD y proporcionan datos estadísticos al ABD, que los utiliza para tomar decisiones con el objetivo de mejorar el rendimiento o funcionamiento del sistema de bases de datos.

- ***Utilidades para Control de accesos de los Usuarios*** de la base de datos.

- ***Utilidades para acceso al Diccionario de Datos***

2.3 ARQUITECTURA CLIENTE-SERVIDOR

Según TIC (Tecnología de Información y Comunicación), (2016) La estructura cliente - servidor es una arquitectura de computación en la que se consigue un procesamiento cooperativo de la información por medio de un conjunto de procesadores, de tal forma que uno o varios clientes, distribuidos geográficamente o no, solicitan servicios de computación a uno o más servidores.

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Atendiendo a esta visión descentralizada, la arquitectura cliente - servidor consiste en una arquitectura distribuida de computación, en la que las tareas de cómputo se reparten entre

distintos procesadores, obteniendo los usuarios finales el resultado final de forma transparente, con independencia del número de equipos (servidores) que han intervenido en el tratamiento. Se puede decir por tanto que la arquitectura cliente - servidor es un tipo de arquitectura distribuida, posiblemente la más extendida.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

En la arquitectura C/S el remitente de una solicitud es conocido como **cliente**.

Sus características son:

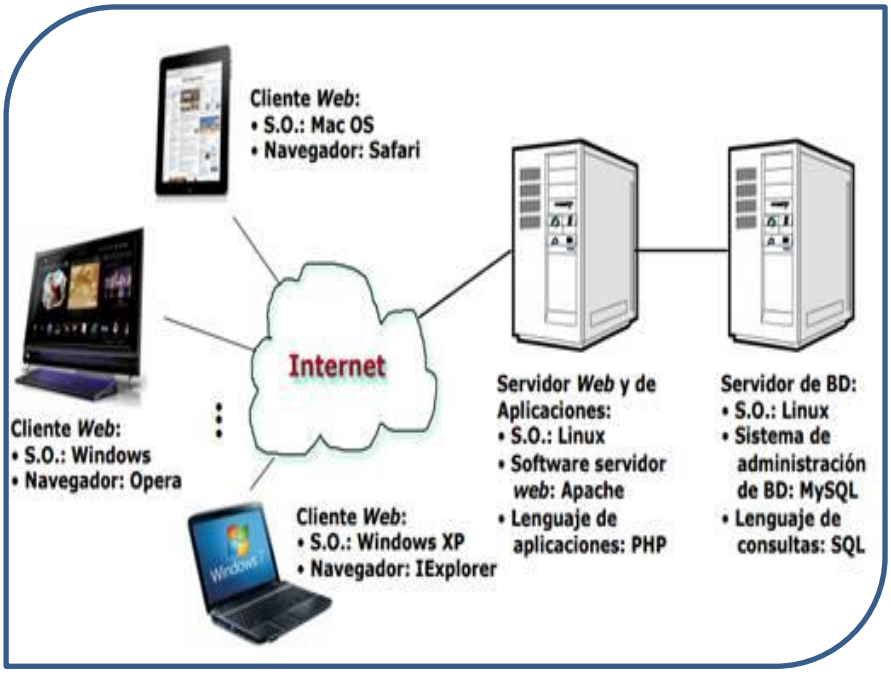
- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

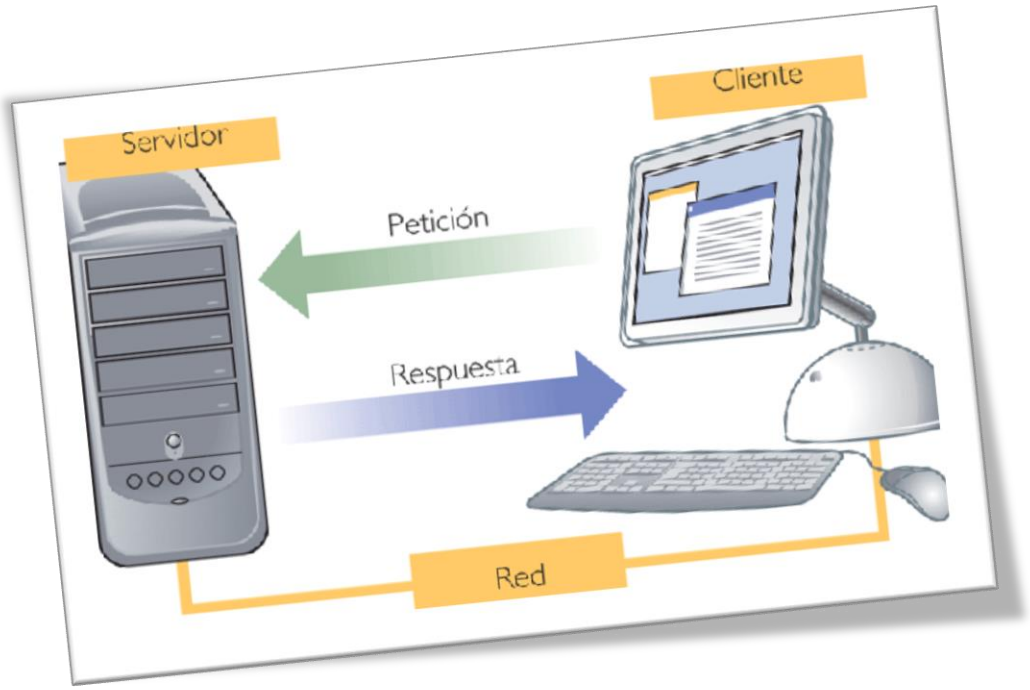
Al receptor de la solicitud enviada por cliente se conoce como **servidor**.

Sus características son:

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales.

Ejemplo de Arquitectura Cliente Servidor:



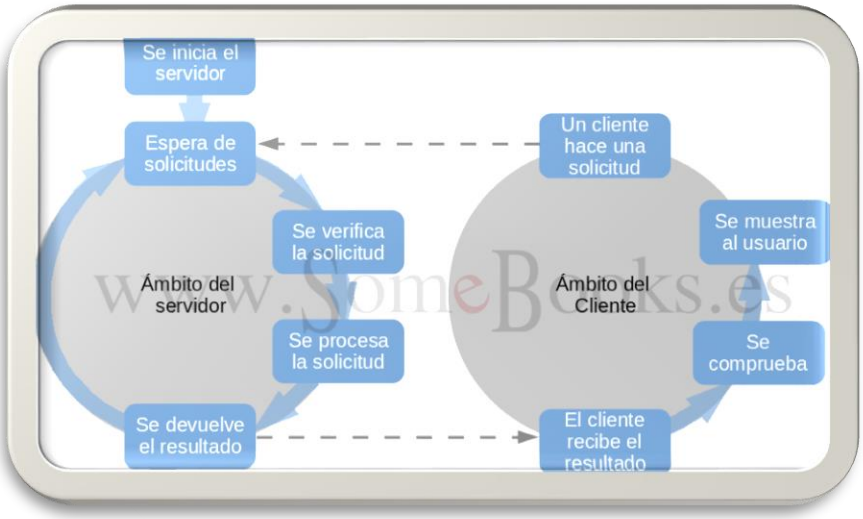


2.3.1 ESQUEMA DE FUNCIONAMIENTO DE UN SISTEMA SEGÚN LA ARQUITECTURA CLIENTE – SERVIDOR

☺ El Esquema de funcionamiento de un Sistema Cliente/Servidor sería:

☺ El cliente solicita una información al servidor.

- ☺ El servidor recibe la petición del cliente.
- ☺ El servidor procesa dicha solicitud.
- ☺ El servidor envía el resultado obtenido al cliente.
- ☺ El cliente recibe el resultado y lo procesa



2.3.2 ELEMENTOS QUE FORMAN PARTE DE UNA ARQUITECTURA CLIENTE – SERVIDOR

Un sistema Cliente/Servidor es un Sistema de Información distribuido basado en las siguientes características:

- Servicio: unidad básica de diseño. El servidor los proporciona y el cliente los utiliza.
- Recursos compartidos: Muchos clientes utilizan los mismos servidores y, a través de ellos, comparten tanto recursos lógicos como físicos.
- Protocolos asimétricos: Los clientes inician "conversaciones". Los servidores esperan su establecimiento pasivamente.
- Transparencia de localización física de los servidores y clientes: El cliente no tiene por qué saber dónde se encuentra situado el recurso que desea utilizar.
- Independencia de la plataforma HW y SW que se emplee.
- Sistemas débilmente acoplados. Interacción basada en envío de mensajes.
- Encapsulamiento de servicios. Los detalles de la implementación de un servicio son transparentes al cliente.

- Escalabilidad horizontal (añadir clientes) y vertical (ampliar potencia de los servidores).
- Integridad: Datos y programas centralizados en servidores facilitan su integridad y mantenimiento.

Elementos principales:

CLIENTE

Igual que antes, al hablar de forma genérica sobre un cliente, nos referimos a un ordenador, normalmente con prestaciones ajustadas, que requiere los servicios de un equipo servidor.

Sin embargo, bajo el punto de vista de la arquitectura cliente/servidor, un cliente es un proceso que solicita los servicios de otro, normalmente a petición de un usuario.

En entornos cliente/servidor, suele utilizarse el término front-end para referirse a un proceso cliente.

Normalmente, un proceso cliente se encarga de interactuar con el usuario, por lo que estará construido con alguna herramienta que permita implementar interfaces gráficas (GUI). Además, se encargará de formular las solicitudes al servidor y recibir su respuesta, por lo que deberá encargarse de una parte de la lógica de la aplicación y de realizar algunas validaciones de forma local.

SERVIDOR

Cuando hablamos de una forma genérica, si mencionamos a un servidor, nos referimos a un ordenador, normalmente con prestaciones elevadas, que ejecuta servicios para atender las demandas de diferentes clientes.

Sin embargo, bajo el punto de vista de la arquitectura cliente/servidor, un servidor es un proceso que ofrece el recurso (o recursos) que administra a los clientes que lo solicitan.

Un servidor es todo proceso que proporciona un servicio a otros. Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se lo conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Las principales funciones que lleva a cabo el proceso servidor se enumeran a continuación:

- Aceptar los requerimientos de bases de datos que hacen los clientes.

- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

MIDDLEWARE

Es la parte del software del sistema que se encarga del transporte de los mensajes entre el cliente y el servidor, por lo que se ejecuta en ambos lados de la estructura.

El middleware permite independizar a los clientes y a los servidores, sobre todo, gracias a los sistemas abiertos, que eliminan la necesidad de supeditarse a tecnologías propietarias.

Por lo tanto, el middleware facilita el desarrollo de aplicaciones, porque resuelve la parte del transporte de mensajes y facilita la interconexión de sistemas heterogéneos sin utilizar tecnologías propietarias.

Además, ofrece más control sobre el negocio, debido a que permite obtener información desde diferentes orígenes (uniendo tecnologías y arquitecturas distintas) y ofrecerla de manera conjunta.

Podemos estructurar el middleware en tres niveles:

- El protocolo de transporte, que será común para otras aplicaciones del sistema.
- El sistema operativo de red
- El protocolo del servicio, que será específico del tipo de sistema cliente/servidor que estemos considerando.

2.3.3 ARQUITECTURA MULTICAPAS

La arquitectura cliente/servidor genérica tiene dos tipos de nodos en la red: clientes y servidores. Consecuentemente, estas arquitecturas genéricas se refieren a veces como arquitecturas de dos niveles o dos capas.

Algunas redes disponen de tres tipos de nodos:

- Clientes que interactúan con los usuarios finales.
- Servidores de aplicación que procesan los datos para los clientes.
- Servidores de la base de datos que almacenan los datos para los servidores de aplicación.
- Esta configuración se llama una arquitectura de tres-capas.

Ventajas de las arquitecturas n-capas:

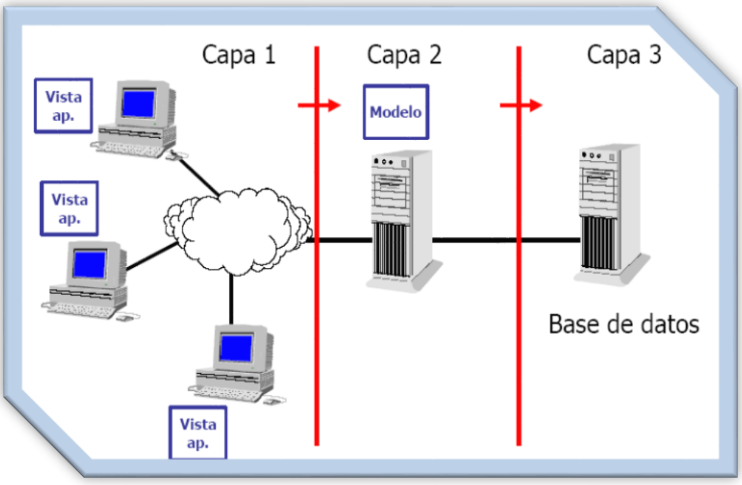
- La ventaja fundamental de una arquitectura n-capas comparado con una arquitectura de dos niveles (o una tres-capas con una de dos niveles) es que separa hacia fuera el proceso, eso ocurre para mejorar el balance la carga en los diversos servidores; es más escalable.

Desventajas de las arquitecturas de la n-capas:

- Pone más carga en la red, debido a una mayor cantidad de tráfico de la red.
- Es mucho más difícil programar y probar el software que en arquitectura de dos niveles porque tienen que comunicarse más dispositivos para terminar la transacción de un usuario.

MODELO CLIENTE/SERVIDOR 3 CAPAS

Esta estructura se caracteriza por elaborar la aplicación en base a dos capas principales de software, más la capa correspondiente al servidor de base de datos. Al igual que en la arquitectura dos capas, y según las decisiones de diseño que se



tomen, se puede balancear la carga de trabajo entre el proceso cliente y el nuevo proceso correspondiente al servidor de aplicación.

Ventajas:

- Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto a la estructura en dos planos).
- Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones. Provee escalabilidad horizontal y vertical.

- Se mantiene la independencia entre el código de la aplicación (reglas y conocimiento del negocio) y los datos, mejorando la portabilidad de las aplicaciones.

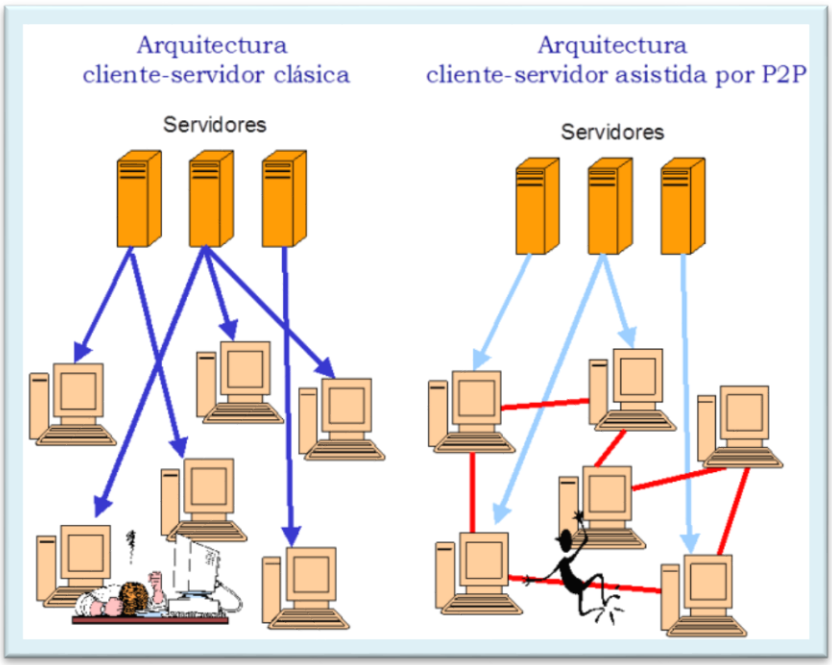
Inconvenientes:

- Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con Cliente/Servidor dos planos.
- Existen pocos proveedores de herramientas integradas de desarrollo con relación al modelo Cliente/Servidor dos planos, y normalmente son de alto costo.

2.3.4 MEJORAS DE LA ARQUITECTURA CLIENTE – SERVIDOR

Como ya habrás deducido, el modelo cliente/servidor está especialmente indicado en redes medias o grandes que necesiten un alto nivel de fiabilidad.

Centralización del control: Los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no



autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P- Red entre pares).

Escalabilidad: Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

Fácil mantenimiento: Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.

Inconvenientes de la Arquitectura Cliente – Servidor

La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultáneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes P2P como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.

El paradigma de C/S clásico no tiene la robustez de una red P2P. Cuando un servidor está caído, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.

El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el coste. El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es una Web, no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

Presenta dependencia del servidor ya que toda la red está construida al rededor del servidor y si éste deja de funcionar o lo hace con un rendimiento inadecuado, afectará a toda la infraestructura.

Afortunadamente, este último inconveniente está superado, al menos en parte, gracias a sistemas como los servidores redundantes, la tolerancia a fallos y los sistemas de almacenamiento en modo RAID.

III. CONCLUSIONES

- Últimamente, gracias a la evolución de la tecnología y la consiguiente reducción de los costos, hemos asistido a la grande y creciente difusión de los elaboradores y de su uso para facilitar y potenciar la posibilidad de tratamiento de la información. Paralelamente las nuevas tecnologías informáticas proponen nuevos modelos de sistema para la gestión de datos.
- Conocer el proceso interno, la estructura he implementación de base de datos nos muestra la importancia que realizan en el mundo laboral, y como cada ente que las utiliza es dependiente de ellas.
- Las bases de datos forman el núcleo de las principales aplicaciones, sitio web y servicios corporativos.
- El propósito de una base de datos es responder a consultas y ejecutar transacciones de datos.
- El objetivo principal de un SGBD es proporcionar un entorno a la vez práctico y eficiente a la hora.
- de almacenar y recuperar la información de la base de datos.
- La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.
- Esta arquitectura se basa en la existencia de dos tipos de aplicaciones ejecutándose de forma independiente.
- Una de las aplicaciones actúa como servidora la otra como cliente.
- El cliente pide datos, se envían en forma de consulta al servidor el servidor procesa la consulta y devuelve los datos al cliente y solo viajan los datos pedidos.

IV. BIBLIOGRAFÍA

- [EN 2002] Elmasri, R.; Navathe, S.B. **Fundamentos de Sistemas de Bases de Datos. *3ª Edición***. Madrid [etc.]: Addison-Wesley, Pearson Educación, 2002. (Capítulos 1 y 2)

- [EN 1997] Elmasri, R.; Navathe, S.B.: **Sistemas de bases de datos. Conceptos fundamentales**. ***2ª Edición***. Wilmington, Delaware, USA: Addison-Wesley Iberoamericana, 1997. (Capítulos 1 y 2)

- [MPM 1999] De Miguel, A.; Piattini, M.; Marcos, E. **Diseño de bases de datos relacionales**. Madrid: Ra-Ma, 1999. (Capítulos 1 y 2)

- [MP 1993] De Miguel, A.; Piattini, M.: **Concepción y diseño de bases de datos: del Modelo E/R al Modelo Relacional**. Madrid: Ra-Ma, 1993.

- [SKS 1998] Korth, H; Silberschatz, A., Sudarshan, S.:**Fundamentos de bases de datos. *3ª Edición***. Madrid: McGraw-Hill, 1998. (Capítulo 1)

V. REFERENCIAS

- http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/
- https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/
- https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos
- https://es.wikipedia.org/wiki/Base_de_datos
- http://www.monografias.com/docs114/telecomunicaciones-arquitectura-cliente-servidor/telecomunicaciones-arquitectura-cliente-servidor.shtml
- https://tecnologia-facil.com/que-es/que-es-p2p/
- https://sites.google.com/site/aimbsor/introduccion-a-los-sor/1-1-arquitectura-cliente-servidor