Departamento Académico de Computación

DIVISIÓN ACADÉMICA DE INGENIERÍA

DEPARTAMENTO ACADÉMICO DE COMPUTACIÓN

Notas del curso de Desarrollo de Aplicaciones Informáticas

Autor: Dr. Felipe López G. Enero 2009

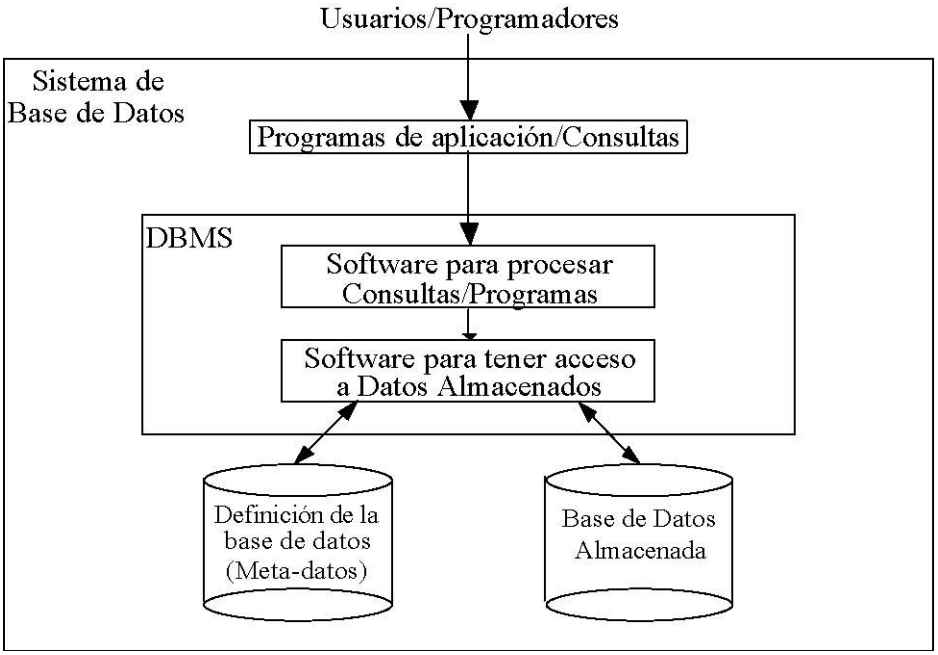
(Actualizado por la Dra. Alejandra Flores Mosri Enero 2017)

Departamento Académico de Computación

Introducción al procesamiento con bases de datos

Conceptos de bases de datos

Sistema de base de datos



Una **base de datos** es una **colección de datos relacionados** la cual tiene las siguientes propiedades implícitas:

- representa algún aspecto del mundo real, llamado el **minimundo** o el **Universo de Discurso (UD)**. Cambios en el minimundo se reflejan en la base de datos;
- su colección de datos es lógicamente coherente con un significado inherente; esto es, un conjunto de datos al azar normalmente no constituye una base de datos;
- se diseña, se construye y se llena con datos para un propósito específico;
- puede ser usada desde múltiples aplicaciones por diversos grupos de usuarios.

Un **DBMS (DataBase Management System)** (o SABD-Sistema de Administración de Bases de Datos) es una colección de programas que permite a usuarios crear y mantener bases de datos. Es un software de *propósito general* que facilita los procesos de:

- definición (especificación de estructuras, tipos de datos y restricciones),

- construcción (almacenamiento de los datos en algún medio) y

- manipulación (consultas y actualización) de bases de datos para diversas aplicaciones.

Un DBMS puede ser de *propósito especial* para manipular bases de datos que tienen un propósito específico.

Características de aplicaciones con bases de datos

Naturaleza auto-descriptiva de un sistema de base de datos

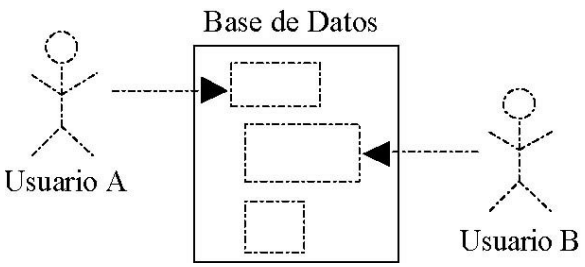
El sistema de base de datos contiene no sólo a la base de datos misma sino también una definición completa de su estructura. La definición se almacena en el **catálogo** del sistema. Esta información también se conoce como **meta-datos** o **diccionario de datos** o **directorio de datos.** El catálogo es usado por el DBMS para poder trabajar con *cualquier base de datos*. En un solo catálogo se mantienen las estructuras de las diversas bases de datos requeridas por los usuarios; éste se define una sola vez, salvo cambios posteriores, y después es usado por diversas aplicaciones. Normalmente la redundancia es poca y los cambios son más fáciles de hacer.

Independencia entre programas y datos

El DBMS proporciona a los usuarios una **representación lógica** de los datos sin incluir muchos de los detalles de cómo éstos son almacenados. El código interactúa con los datos sin influir en la estructura de éstos.

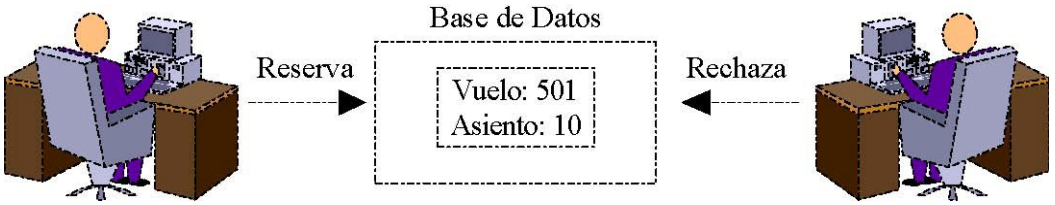
Soporte de múltiples vistas de los datos

Una base de datos normalmente tiene muchos usuarios, cada uno de los cuales puede requerir diferentes perspectivas o **vistas** de la base de datos. Una vista puede ser un subconjunto de la base de datos o puede contener datos **virtuales** que son derivados de otros datos de la base.



Procesamiento de transacciones multiusuario

El DBMS debe incluir software para **control de la concurrencia** a fin de asegurar que varios usuarios que tratan de actualizar los mismos datos, lo hagan en una forma controlada de tal manera que los resultados sean correctos. Ejemplo:



Usuario A reserva el asiento 10 en el vuelo 501

Usuario B reserva el asiento 10 en el vuelo 501

A este tipo de procesamiento se le conoce como de **transacciones multiusuario**.

Ventajas de usar un DBMS

- **Control de la redundancia** Cuando se hace el diseño de una base de datos se debe almacenar *una sola vez* cada dato lógico, tal como el nombre o fecha de nacimiento de una persona. Esto evita inconsistencias en la información y a la vez ahorra espacio de almacenamiento. En algunos casos es deseable, o necesario, tener una **redundancia controlada** para optimizar tiempos de respuesta en consultas o porque así lo exige el modelo de datos empleado. El DBMS debe proporcionar este recurso.

- **Restricciones de integridad** Esta característica se encuentra relacionada con la del punto anterior. Estas restricciones van desde permitir que a la base sólo ingresen **datos corre ctos** , por ejemplo valores en un rango, hasta forzar que un registro **esté relacionado** con otros registros de una cierta manera. Normalmente estas restricciones se derivan del significado o **semántica** de los datos del minimundo. La mayoría de las restricciones pueden ser controladas por el DBMS; otras deben controlarse por medio de módulos especializados programados.

- **Restricción a accesos no autorizados (Seguridad)** Normalmente el DBMS debe proporcionar un subsistema de **seguridad y autorización** que el DBA utiliza para asignar cuentas y privilegios (restricciones) a los usuarios de una base de datos. Con este subsistema se puede definir el tipo de operaciones que un usuario puede efectuar y a qué partes de la base puede tener acceso. El uso de vistas es otra forma de controlar el acceso.

- **Múltiples interfaces de usuario** Debido a los diversos tipos de usuarios que usan una base de datos, el DBMS debe proporcionar una variedad de interfaces para los mismos. Éstas pueden incluir: lenguajes de consulta; lenguajes de programación, con capacidades gráficas inclusive; interfaces basadas en menús e interfaces de lenguaje natural.

- **Representación de vínculos complejos entre datos** El DBMS debe tener la capacidad de representar una gran variedad de vínculos complejos entre los datos, así como poder recuperarlos y actualizarlos fácil y eficientemente.

- **Respaldo (Backup) y Recuperación (Recovery)** El DBMS debe tener los recursos para recuperarse de fallas de hardware y de software. El subsistema de **respaldo y recuperación** es el responsable por esta tarea. Por ejemplo, si el sistema falla a mitad de una transacción, el DBMS debe poder restaurar la base de datos hasta el estado previo que tenía antes de la falla, y así asegurar que su contenido es válido.

Implicaciones adicionales del enfoque de base de datos

La utilización de bases de datos en las organizaciones presenta varias implicaciones las cuales pueden beneficiar a las funciones computacionales de las mismas.

- **Potencial para establecer estándares** En una organización grande el enfoque de bases de datos permite al DBA establecer estándares entre los usuarios de las mismas. Esto facilita la comunicación y la cooperación entre los departamentos, proyectos y usuarios dentro de la organización. Los estándares pueden ser definidos para los nombres y formatos de los datos, formatos de despliegue, estructuras de reportes, terminología, etc.

- **Disponibilidad de información "al día"** Cuando una actualización a la base de datos es hecha por un usuario, todos los demás usuarios pueden ver inmediatamente dicha actualización. Esta disponibilidad de información "al día" es esencial para muchas transacciones, tales como sistemas de reservaciones y bases de datos de bancos.

Usuarios de bases de datos

Para bases de datos **personales**, normalmente la misma persona define, construye y manipula la base de datos. Para bases de datos **multiusuario grandes**, con varias decenas o cientos de usuarios, diversos tipos de personal están involucrados:

- **Administrador de la base de datos (DBA)** Es el responsable por la administración de los recursos tanto del DBMS como de las bases de datos que tenga la organización. El DBA se encarga de autorizar el acceso a las bases de datos, coordinar y supervisar su uso, adecuar el sistema para un uso eficiente, etc. El DBA puede ser una persona o un equipo de personas de la organización.

- **Diseñadores de la base de datos** Son los responsables de identificar los datos a ser almacenados en una base de datos y de escoger las estructuras apropiadas para representar y almacenar estos datos. Normalmente definen las vistas de los diversos grupos de usuarios para al final integrar una base de datos con todas ellas.

- **Programadores de aplicaciones (ingenieros de software)** Son los encargados de determinar los requerimientos de los usuarios, de desarrollar las especificaciones para los programas y de implementarlos. También se encargan de hacer las pruebas, depuraciones, documentación y mantenimiento de las aplicaciones.

- **Usuarios finales** Son las personas que tienen acceso a la base de datos para hacer consultas y actualizaciones y generar reportes. El rango es amplio y puede ir desde usuarios simples (capturistas) hasta usuarios experimentados (ingenieros, científicos, analistas de negocios, etc.).

Ejemplos de bases de datos

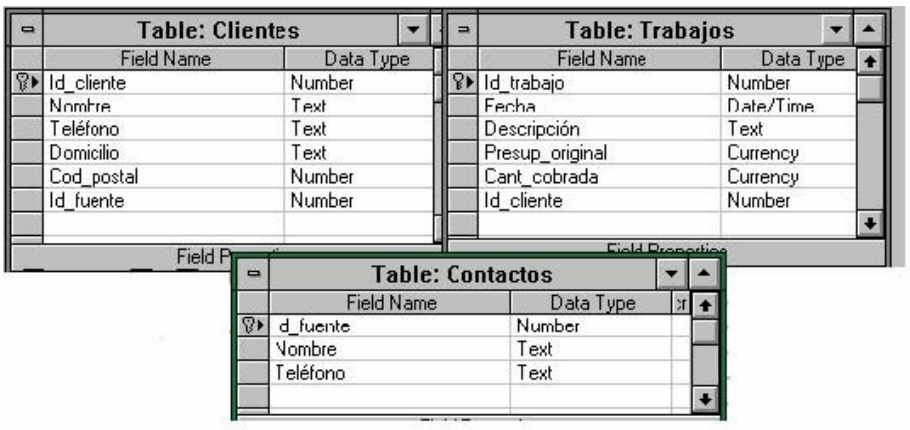
En esta sección se describen varios ejemplos de bases de datos que muestran el espectro de sistemas en las que éstas se pueden usar, así como las características importantes de cada tipo de base de datos.

Micro-empresa para pintar casas

Es una micro-empresa compuesta por dos personas de tiempo completo y algunos trabajadores eventuales, que se dedica a pintar casas. Es una micro-empresa que ya tiene varios años en el mercado, con una buena reputación, que consigue la mayor parte de sus trabajos por recomendaciones de clientes y, ocasionalmente, por contratistas.

Cuentan con una cartera de varias docenas de clientes y emplean una pequeña base de datos personal para guardar información importante acerca de trabajos realizados anteriormente. Esto con el fin de poder obtener datos relevantes cuando se va a efectuar un nuevo trabajo, ya sea para un cliente nuevo (recomendado o no) o para un cliente anterior.

Enseguida se muestra un ejemplo de esta base de datos: Las bases de datos personales, en general, no son grandes, contienen pocas tablas y almacenan poca información.



Los DBMS típicos para estas bases de datos son: Microsoft Access, Microsoft FoxPro, MySQL y las versiones para PC de los manejadores grandes (normalmente gratuitas).

Agencia de venta de automóviles

Las bases de datos pueden ser más complicadas que la del ejemplo anterior. Considérese ahora una agencia de venta de automóviles en la cual laboran dos supervisores, cuatro agentes de ventas y un administrador.

La base de datos contiene información sobre modelos en venta, clientes, ventas realizadas, etc. Esta base de datos es compartida por el personal de la agencia y está localizada en una red de área local formada por:

-un servidor de base de datos,
-una PC para el administrador,
-dos PC's para los supervisores, y
-cuatro PC's para los agentes de ventas.

La base de datos requerida para gestionar la información de esta aplicación es más complicada que la de la micro-empresa que pinta casas. Un ejemplo de las tablas que podría contener es el siguiente (la notación es del modelo relacional, que se verá más adelante):

Modelos(IdModelo, Descripción, PrecioActual, CantUnidades)
Unidades(NoSerie, FechaAdquisición, IdModelo)
Agentes(IdAgente, Nombre, Teléfono, FechaIngreso)
Clientes(IdCliente, Nombre, Domicilio, Teléfono, Notas)
VentasHechas(IdVenta, Fecha, IdAgente, IdCliente)
UnidadesVendidas(IdVenta, NoSerie, PrecioVenta)
Equipos(IdEquipo, Nombre, Descripción, Costo)
EquipoExtra(IdVenta, NoSerie, IdEquipo)

Obviamente, la información almacenada en esta base de datos es mucho mayor que en el primer caso. Además, esta base de datos es multiusuario.

Los DBMS típicos para estas bases de datos son: SQL Server, MySQL (versión avanzada) y las versiones menores o intermedias de los manejadores grandes como Oracle, Sybase y otros.

Sistema de transacciones bancarias (cuentas de cheques)

Este es un ejemplo de un sistema de base de datos de mucho mayor complejidad que los dos anteriores. El sistema debe poder:

-almacenar los datos generales de los clientes y de las cuentas que poseen,
-registrar las transacciones que se realizan y el lugar en que se efectúan,
-generar estados de cuenta,
-llevar a cabo consolidaciones, etc.

Además, el sistema debe permitir acceso multiusuario ya que las diversas transacciones se pueden efectuar:

- en ventanilla (en varias decenas de sucursales), }
- a través de cajero automático (varias decenas),
- por Internet,
- por teléfono, etc.

Esta base de datos atiende simultáneamente a cientos de usuarios. Así, la base de datos es grande y compleja, pudiendo constar de varias decenas de tablas, conteniendo cientos de miles de datos.

Agencia de viajes turísticos

Este ejemplo es el de un sistema de base de datos asociado a un sitio de Internet que se utiliza para: promover lugares turísticos y las características de los mismos, obtener datos y direcciones electrónicas de los clientes que visitan el sitio, y almacenar solicitudes de reservación para estos lugares, incluyendo hoteles y servicios.

La información que se maneja para la base de datos incluye datos multimedia: fotos, videos, sonido, etc. Además, la aplicación permite a los usuarios navegar por la misma, mostrando ventanas que contienen más detalles sobre determinadas características de los lugares.

Las características principales que distinguen a este ejemplo de los tres anteriores son las siguientes:

-gran parte de la información manejada es estructurada (como pueden ser los datos de los clientes), pero otra parte es no estructurada (la información multimedia)

-el acceso a la información por parte de los clientes se hace a través de un navegador estándar, por lo que no se necesita software adicional en su PC se usa la tecnología estándar orientada a Internet para transferir los datos entre el navegador del cliente, la aplicación del sitio y la base de datos.

Conceptos y arquitectura de un sistema de base de datos

Esquemas e instancias

En cualquier modelo de datos es importante distinguir entre la *descripción* de la base de datos y la *base de datos en sí misma*.

La descripción de la base de datos se conoce como **esquema de la base de datos (o metadatos)**. Este esquema se especifica durante el diseño de la base de datos y se espera que

no cambie con frecuencia. Un esquema dibujado se conoce como **diagrama del esquema**. Cada unidad en el esquema es un **elemento del esquema**.

Ejemplo:

PROFESOR

Id_prof	Nom_prof	Categoría

CURSO

Clave	Nom_cu	Creds	Id_prof

ALUMNO

Matri	Nom_al	Carr	Prom

CURSA

Matri	Clave	Calif

Un diagrama de esquema muestra sólo *algunos aspectos* del esquema, tal como el nombre de los elementos y de los campos de datos; pero no muestra otros, como el tipo de los datos, los vínculos entre ellos o restricciones sobre los mismos.

Los datos actuales en una base de datos pueden cambiar frecuentemente. Al conjunto de datos que está en la base de datos en un momento particular del tiempo se le conoce como **estado de la base de datos** (o conjunto de **ocurrencias** o **instancias**). En un estado dado, cada elemento del esquema tiene su propio *conjunto actual* de instancias.

Ejemplo:

CURSO

Clave	Nom_cu	Creds	Id_prof
300	Computacion I	9	1
310	Algoritmica	9	1
605	Economia V	8	5

Cada que se actualiza la base de datos (inserción, modificación o eliminación de datos), ésta cambia de un estado a otro.

Cuando se **define** una nueva base de datos, sólo se especifica su esquema al DBMS. Su estado inicial es el "estado vacío". Cuando se **cargan** los primeros datos, éstos definen el "estado inicial" de la base de datos. Después, cada actualización hace que la base de datos pase de un estado a otro.

En cualquier momento, un estado de la base de datos debe ser **válido**, esto es, el estado debe satisfacer tanto la estructura como las restricciones especificadas para la base de datos.

Lenguaje SQL

El lenguaje **SQL** (Structured Query Language) es el **estándar** para manejadores relacionales. Contiene elementos para:

– definir el esquema lógico de la base de datos. Esta definición se compila y se guarda en el catálogo del DBMS,

– especificar el esquema interno de la base de datos. No es común encontrar esta parte en los DBMS, salvo, posiblemente, en algunos de gran complejidad,

– especificar vistas de usuarios,

– manipular los datos de la base, lo cual incluye recuperar, insertar, eliminar y modificar

– administrar los recursos del DBMS y de las bases de datos.

En ocasiones, los DBMS, sobre todo los de mayor complejidad, brindan un lenguaje adicional **no estándar** que permite efectuar ciertas labores de programación con las bases de datos. Este lenguaje puede tener algunas pocas instrucciones de programación, o puede ser de cierta complejidad. Normalmente este lenguaje se usa para programar rutinas, llamadas **procedimientos almacenados**, con el objetivo de tener accesos muy eficientes a la base de datos (por el hecho de que siempre son almacenados junto con ella).

Por otro lado, el DBMS puede permitir trabajar con SQL en forma interactiva, a través de terminal o de PC comunicada con la base de datos, o puede ejecutar instrucciones de SQL embebidas en un lenguaje de programación de propósito general. En este último caso, el lenguaje de programación se conoce como **lenguaje anfitrión (host).**

Interfaces

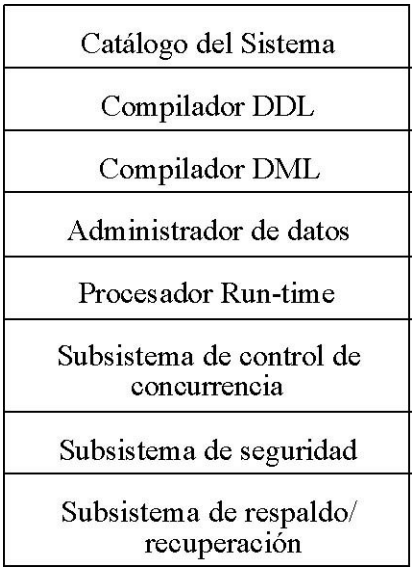
Son las diferentes formas de acceder a un ambiente de bases de datos. Algunas pueden ser proporcionadas por el DBMS; otras, programadas por medio de un lenguaje de programación.

- **Gráficas**, típicamente despliegan al usuario un diagrama el cual éste va llenando con información. Normalmente están combinadas con menúes basados en ventanas. La mayoría se implementan con aplicaciones web o móviles.
- **De lenguaje natural**, las cuales aceptan solicitudes escritas en lenguaje natural (normalmente, inglés). No son comunes y se puede decir que aún están en desarrollo.
- **Para usuarios paramétricos**, que se emplean para presentar al usuario un conjunto reducido de operaciones, muchas de ellas asignadas a teclas del teclado. Están destinadas a usuarios que efectúan operaciones repetitivas, como los cajeros de bancos.

- **Para el DBA**, que normalmente contienen instrucciones privilegiadas para ser usadas exclusivamente por el DBA.

Componentes de un DBMS

Un DBMS es un sistema de software grande y complejo. A continuación se describen brevemente las partes importantes de un DBMS.



- **Catálogo del sistema,** es una mini-base de datos almacenada en disco que puede estar separada físicamente o no de las bases de datos controladas por el DBMS. Contiene información como nombres de archivos, campos de datos, tipos de datos, detalles de almacenamiento, información de transformación entre esquemas y restricciones.
- **Compilador DDL (Data Definition Language),** procesa las definiciones de los esquemas y almacena sus descripciones (meta-datos) en el catálogo.
- **Compilador DML (Data Manipulation Language)**, analiza y traduce las instrucciones de SQL que actualizan (insertan, borran y cambian) y consultan a la base de datos. Normalmente genera llamados al procesador run-time para que éste las ejecute.
- **Administrador de datos,** controla el acceso a la información almacenada en disco, ya sea que forme parte de una base de datos o del catálogo. Usa servicios del sistema operativo para intercambiar datos entre disco y memoria central.
- **Procesador run-time,** recibe operaciones de recuperación y de actualización y las ejecuta sobre la base de datos; para esto se apoya en el administrador de datos.
- **Subsistema de control de concurrencia,** controla el acceso simultáneo a una base de datos realizado por varios usuarios.
- **Subsistema de seguridad**, restringe el acceso a la base de datos, tanto a usuarios no autorizados como a partes de ésta que sólo determinado grupo de usuarios puede usar.

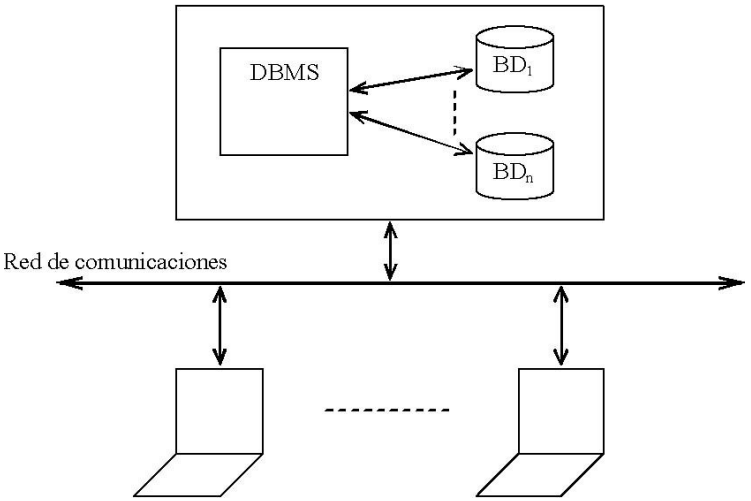
- **Subsistema de respaldo/recuperación,** permite crear respaldos de las bases de datos en otro dispositivo, ya sea disco o cinta magnética, para poder recuperar la información, posteriormente, en caso de una falla catastrófica.

Arquitecturas entre bases de datos y aplicaciones

Existen básicamente tres tipos de arquitectura para el procesamiento de bases de datos desde aplicaciones, relacionadas todas ellas con el sitio en el cual se encuentra cada parte del sistema. La más simple es la **arquitectura local** en la cual todos los componentes del sistema (base de datos, DBMS y aplicaciones) se encuentran en la misma computadora. Esta arquitectura normalmente es la que siguen los sistemas personales. Los otros dos tipos se describen a continuación.

Arquitectura cliente-servidor

Es una arquitectura en la cual el sistema de base de datos se divide en dos partes: el servidor y el cliente.

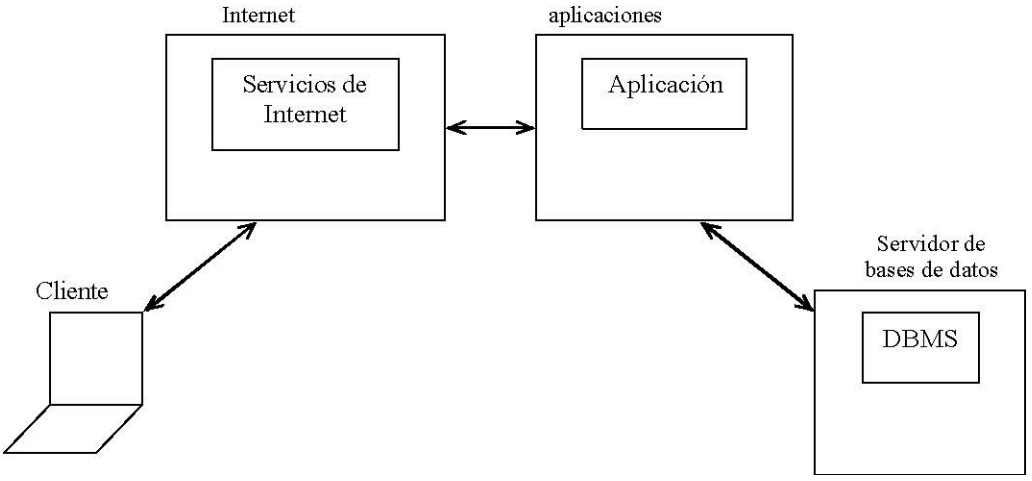


El servidor está formado precisamente por el DBMS llevando a cabo la administración y la manipulación de las bases de datos que controla. Los clientes son las diversas aplicaciones que trabajan con la información que está en las bases de datos, tanto aplicaciones escritas por usuarios como aplicaciones integradas (proporcionadas por el fabricante del DBMS o por terceros). Los clientes hacen peticiones al servidor (el DBMS), éste las recibe y las procesa, y envía las respuestas de vuelta a los clientes.

Es común usar computadoras personales o estaciones de trabajo sencillas, del lado de los clientes, y estaciones de trabajo poderosas o **mainframes**, del lado del servidor. Obviamente debe existir una red y software de comunicación para que clientes y servidor puedan intercambiar información y así poder implementar este tipo de arquitectura. Este tipo es el que se podría emplear para el segundo y tercer ejemplos, sobre todo el segundo, mencionados en la sección 1.2.

Arquitectura multicapa

También es conocida como arquitectura de **procesamiento distribuido**. En este caso el sistema se descompone en varias **capas**, cada una llevando a cabo un tipo de procesamiento específico.



Por ejemplo, en la capa más cercana al usuario se podría tener un programa con interfaces gráficas poderosas para facilitar la actualización/consulta de la información de la base de datos a través de ventanas. En la siguiente capa se podría tener un **servidor de Internet** que llevara el control de todas las **páginas** que se mostrarían al usuario como interfaces de la aplicación. La tercera capa podría ser un **servidor de aplicaciones** el cual contendría las aplicaciones que implementan la lógica (reglas) del negocio. Y, finalmente, la última capa contendría al **servidor de bases de datos**.

Servicios en la nube

Para este tipo de arquitectura se utilizan granjas de servidores virtuales que implementan arquitecturas multicapa.

La mayor ventaja de utilizar una nube es que la administración de los recursos recae en el administrador de la nube, minimizando de esta forma costos. También permite a la capacidad de los servicios crecer de acuerdo a la demanda de estos.

Existen varios tipos de servicios en la nube:

IaaS – Infrastructure as a Service. Unicamente se rentan los servidores (capacidad de procesamiento, almacenamiento y memoria), el equipo de TI es responsable de instalar y mantener tanto los sistemas operativos, el DBMS, las bibliotecas de programación (PHP, .NET, etc.), los servicios hacia los clientes (Apache, IIS, Tomcat, etc.) y las aplicaciones que decidan programarse/instalarse.

PaaS – Platform as a Service. Se rentan los servidores con sistemas operativos y cierto software como los DBMS, las bibliotecas de programación y los servicios a clientes. El proveedor de la nube es responsable de mantener los sistemas operativos y el software al día

con parches operativos y de seguridad. El cliente es responsable de instalar/programar las aplicaciones.

SaaS – Software as a Service. Se rentan las aplicaciones. El proveedor de la nube es responsable de toda la arquitectura incluyendo las aplicaciones y debe prestar el servicio a sus clientes siguiendo acuerdos de servicio específicos.

MaaS – Malware as a Service. Se rentan servicios de malware, pueden ser servicios para distribuir malware o bien para recopilar información de clientes afectados con malware.