**SISTEMAS DE BASE DE DATOS**:

Un sistema de base de datos es un sistema computarizado cuyo propósito general es mantener información, además de permitir a los usuarios recuperarla y actualizarla cuando se lo necesite. Esta información es cualquier cosa necesaria para apoyar el proceso general de atender los asuntos de esa organización o individuo. Este sistema está formado por cuatro componentes principales: la información, el equipo, los programas y los usuarios.

INFORMACION: Estos pueden estar integrados y compartidos, en sistemas monousuario o multiusuario. Integrada significa que la base de datos puede considerarse como la unificación de varios archivos de datos, donde se elimina del todo o en parte cualquier redundancia entre ellos. Compartidos significa que los elementos individuales de información en la DB pueden compartirse entre varios usuarios distintos, en el sentido de que todos los usuarios puedan tener acceso al mismo elemento de información, por ejemplo los datos del archivo *Alumnos* para el sistema de biblioteca estarán compartidos por el Departamento de Alumnos.

EQUIPO: Son aquellos volúmenes de almacenamiento secundario, junto con dispositivos IO, además de la memoria, procesador, etc.

PROGRAMAS: Entre la base de datos física misma (es decir, los datos tal y como están almacenados en realidad), y los usuarios del sistema, existe una capa de software conocida como *Sistema de administración de base de datos*, DBMS. Este maneja todas las solicitudes de acceso a la base de datos formulada por los usuarios, además de distanciar a los usuarios de la base de datos al nivel del equipo (como los lenguajes de alto nivel evitan la necesidad de ocuparse de detalles de nivel de maquina).

USUARIOS: Existen tres clases amplias de usuarios:

*Programador de aplicaciones*: es quien se encarga de escribir los programas de aplicación que utilizaran la base de datos, estos programas operan sobre los datos (inserción de información nueva, modificación, recuperación y eliminación de datos existentes). Donde todas estas funciones se llevan a cabo dirigiendo las solicitudes apropiadas al DBMS.

*Usuario final*: es el que interactúa con el sistema, obteniendo acceso a la base de datos a través de las aplicaciones desarrolladas por los programadores de aplicación. Estas aplicaciones suelen ser amigables y fáciles de usar para aquellas personas sin conocimiento de base de datos,

*Administrador de base de datos*: es el técnico responsable de poner en práctica las decisiones del DA[[1]](#footnote-1), a diferencia de este -que es un gerente-, el DBA es un profesional en procesamiento de datos, su tarea es crear la base de datos en sí y poner en vigor los controles técnicos necesarios para apoyar las políticas dictadas por el DA, además de encargarse de garantizar el funcionamiento adecuado del sistema y establecer un enlace con los usuarios. El DBA no es solo una persona sino que un equipo de varias personas.

**BASE DE DATOS**:

Una base de datos es un conjunto de datos relacionados entre sí, persistentes en el tiempo que son utilizados por programas de aplicación de una organización. Este se diseña, construye y carga datos para un propósito específico, está dirigida a un determinado grupo de usuarios y la utilizan aplicaciones específicas. La gran ventaja de una base de datos a una empresa es que ofrece un control centralizado de la su información.

¿POR QUE UTILIZAR UNA BASE DE DATOS?

*Es compacto*: no hacen falta archivos de papeles que pudieran ocupar espacio.

*Es rápido*: las computadoras pueden obtener y modificar datos con mucha más velocidad que nosotros, además de poder hacer consultas sin necesidad de búsquedas visuales o manuales que requieren mucho más tiempo.

*Es menos laborioso*: se elimina gran parte del trabajo de mantener archivos a mano, las tareas realizadas por una computadora son mejores realizadas.

*Es actual*: se dispone en cualquier momento de información precisa y al día.

VENTAJAS DE UN SISTEMA DE BASE DE DATOS:

*Es posible disminuir redundancia*: en los sistemas sin base de datos cada aplicación tiene sus propios archivos privados. Esto puede provocar considerable redundancia en los datos almacenados, con el consecuente desperdicio de espacio de almacenamiento.

*Es posible evitar la inconsistencia*: una base de datos es inconsistente si hay dos entradas que representan lo mismo -es decir, hay redundancia- en donde estas no coinciden, en donde no se sabrá cuál es el correcto o el más actualizado. Por ende si no hay redundancia, no habrá inconsistencia.

*Es posible compartir los datos*: no solo las aplicaciones ya existentes pueden compartir la información de la base de datos, sino también que aquellas aplicaciones desarrolladas en un futuro puedan podrán trabajar con los mimos datos almacenados.

*Es posible hacer cumplir los estándares*: el DBA al tener un control centralizado de la base de datos, se puede garantizar la observancia de todas las normas aplicadas para la representación de datos. La normalización de formatos de los datos almacenaos es deseables sobre todo como apoyo para el intercambio de información, o migración de los datos entre sistema.

*Es posible aplicar restricciones de seguridad*: el DBA al tener jurisdicción completa sobre la base de datos, este puede asegurar que el acceso a la base de datos sea solo a través de los canales apropiados, además de poder definir las verificaciones de seguridad por realizar cuando se intente acceder a información delicada –estableciendo diferentes verificaciones para cada tipo de acceso (consulta, modificación, etc.)-.

*Es posible mantener la integridad*: el problema de la integridad radica en asegurar que la información de la base de datos sea correcta. Aunque no haya redundancia en la misma, es evidente que la base de datos puede contener información errónea –un control centralizado de la base de datos puede contribuir a evitar este tipo de problemas-

*Es posible equilibrar requerimientos opuestos*: al conocer los requerimientos generales de la empresa –en contraste con los requerimientos de cualquier usuario individual- el DBA puede estructurar el sistema con miras a proporcionar un servicio general.

COMPARACION CON UN SISTEMA DE ARCHIVOS:

*El sistema de archivos*:

No conoce la estructura interna de los registros.

No provee restricciones de seguridad e integridad.

No provee mecanismos de recuperación y concurrencia.

No existe el diccionario de datos.

No proporciona una correcta independencia de los datos.

No pueden compartirse e integrarse los datos de la misma manera que en una base de datos.

**INDEPENDENCIA DE LOS DATOS**:

La independencia de los datos es un objetivo primordial de los sistemas de bases de datos. Esta independencia puede definirse como la inmunidad de las aplicaciones ante los cambios en la estructura de almacenamiento y en la técnica de acceso, lo cual indica que las aplicaciones en cuestión no dependen de una estructura de almacenamiento o una técnica de acceso específicas. En un sistema de base de datos no es recomendable tener aplicaciones dependientes de datos por dos razones:

1. Cada aplicación requiere una vista diferente de los mismos datos, por Ej.: existan dos aplicaciones, cada una con su archivo privado *Saldo* en la cual uno lo almacena al valor en binario y otro en decimal. Aunque es posible integrar los dos archivos en uno solo y eliminar la redundancia, uno deberá hacer una conversión de los datos.
2. El DBA debe tener libertad para modificar la estructura de almacenamiento o la técnica de acceso para adaptarlas a cambios en los requerimientos, sin tener que modificar las aplicaciones ya existentes, por Ej.: podrían agregarse datos de otra clase a la base de datos, cambiar prioridades de las aplicaciones.

*Campo almacenado*: es la unidad más pequeña de información almacenada que recibe un nombre. La base de datos incluirá, en la mayor parte de los casos, muchas ocurrencias de cada uno de los diversos tipos de campo almacenado. Por Ej.: nombre, dni, etc.

*Registro almacenado*: es un conjunto de campos almacenados, relacionados entre sí, que cuenta con su propio nombre. Una ocurrencia de un registro almacenado está formada por un grupo de ocurrencias de campos almacenados relacionados entre sí. Por Ej.: persona, materia, etc.

*Archivo almacenado*: es el conjunto –con nombre- de todas las ocurrencias de un tipo de registro almacenado.

REQUISITOS PARA LOGRAR LA INDEPENDENCIA DE LOS DATOS:

*Representación de datos numéricos*: un campo numérico puede almacenarse en forma aritmética interna o como cadena de caracteres. El DBA deberá escoger la base apropiada, la escala, el modo y la precisión.

*Representación de datos de caracteres*: puede almacenarse en cualquiera de los tipos de códigos de caracteres.

*Unidades para datos numéricos*: podrían cambiarse las unidades de un campo numérico -pulgadas a cm. por Ej.- durante un proceso de conversión al sistema métrico.

*Codificación de los datos*: en algunas situaciones podría ser conveniente representar datos almacenados mediante códigos –por Ej. Rojo=1, Verde=2-.

*Materialización de los datos*: Casi siempre el campo lógico percibido por una aplicación correspondiente a un campo almacenado único – aunque puede haber diferencias en el tipo de datos o unidades-, en casos así, puede considerarse directo –por Ej. Edad y fecha de nacimiento-. Pero a veces un campo lógico se materializa mediante un cálculo sobre un conjunto de ocurrencias –por Ej. Cantidad total -, siendo este un proceso de materialización indirecto.

*Estructura de los registros almacenados*: podrían combinarse dos registros almacenados existentes para formar uno solo, por Ej. Un registro – nombre, sexo- y otro –fechaNacimiento-. Esto se podría presentar cuando se integran a la base de datos aplicaciones anteriores a su instalación.

**ADMINISTRADOR DE BASE DE DATOS**:

El sistema de administración de la base de datos –DBMS- es el conjunto de programas que maneja todo acceso a la base de datos. Conceptualmente sucede lo siguiente:

1. Un usuario solicita acceso, empleando algún sublenguaje de datos determinado.
2. El DBMS interpreta esa solicitud y la analiza.
3. El DBMS inspecciona, en orden, el esquema externo de ese usuario la correspondencia externa/conceptual asociada, el esquema conceptual, la correspondencia conceptual/interna, y la definición de la estructura de almacenamiento.
4. El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenados.

FUNCIONES DEL DBMS:

*Definición de datos*: el DBMS debe ser capaz de aceptar definiciones de datos en versión fuente y convertirlas en la versión objeto apropiado. Es decir, el DBMS debe poder incluir componentes procesadores de lenguaje para cada uno de los diversos lenguajes de definición de datos –DLL-.

*Manipulación de datos*: el DBMS debe ser capaz de atender las solicitudes del usuario para extraer, y quizás poner al día, datos que ya existen en la base de datos. Es decir, debe incluir un componente procesador de lenguaje de manipulación de datos -DML-. Estas solicitudes DML pueden ser *planeadas* o *no planeadas*.

*Solicitud planeada*: es aquella cuya necesidad se previó mucho tiempo antes de que tuviera que ejecutarse por primera vez, donde el DBA habrá afinado el diseño físico de la base de datos para garantizar un buen desempeño de estas solicitudes.

*Solicitud no planeada*: es una consulta *ad hoc*, es decir una solicitud que no se previó, sino que surgió de improviso. El diseño de la base de datos puede o no ser ideal para solicitudes no planeadas.

*Seguridad e integridad de los datos*: el DBMS debe supervisar las solicitudes de los usuarios y rechazar los intentos de violar las medidas de seguridad e integridad definidas por el DBA.

*Recuperación y concurrencia de los datos*: el DBMS debe cuidar del cumplimiento de ciertos controles de recuperación y concurrencia.

*Diccionario de datos -metadatos-*: el DBMS debe incluir una función de diccionario de datos –siendo este una base de datos para el sistema, no para el usuario-, el contenido de este son datos acerca de datos, en donde se definen objetos del sistema, además de cuales programas utilizaran cuales partes de la base de datos, o que terminales están conectadas al sistema.

*Rendimiento*: el DBMS deberá ejecutar todas las funciones en la forma más eficiente posible.

**ARQUITECTURA DE UN SISTEMA DE BASE DE DATOS**:

Brinda un marco para definir los conceptos generales. Es genérico, lo cual no quiere decir que un sistema de base de datos en particular lo cumpla, además, está basado en la arquitectura ANSI/SPARC. Esta arquitectura se divide en tres niveles denominados nivel interno, conceptual y externo.

*Nivel interno*: la vista interna es una representación de bajo nivel de toda la base de datos, se compone de varias ocurrencias de varios tipos de registro interno –registro almacenado-, índices, etc. Esta representación todavía aún está a un paso del nivel físico, ya que no maneja registros físicos –páginas o bloques-. La vista interna se define mediante el esquema interno, el cual, define los diversos tipos de registros almacenados, además de especificar cuáles índices hay, como se representan los campos almacenados, en que secuencia física se encuentran los registros almacenados, etc. El esquema interno se escribe con otro lenguaje más de definición de datos, el DLL interno.

*Nivel conceptual*: es una representación de toda la información contenida en la base de datos, pueden ser muy diferente de la forma como percibe los datos cualquier usuario individual. La vista conceptual se compone de varias ocurrencias de varios tipos de registro conceptual. La vista conceptual se define mediante un esquema conceptual, el cual incluye definiciones de cada uno de los tipos de registro conceptual. El esquema conceptual se escribe utilizando otro lenguaje de definición de datos, el DLL conceptual. Este esquema permite además agregar algunas verificaciones sencillas de integridad y seguridad, además de agregar reglas de negocios en general.

*Nivel externo*: es el nivel del usuario individual. Se usa un lenguaje primario y un sublenguaje de datos -DSL-, es decir un subconjunto del lenguaje total que se ocupe de manera específica de los objetos y operaciones de la base de datos. El DSL esta embebido dentro del lenguaje primario correspondiente –generalmente SQL-. Cualquier DSL es en realidad una combinación de por lo menos dos lenguajes subordinados: lenguaje de definición de datos -DLL-, con el cual es posible definir o declarar los objetos de la base de datos y un lenguaje de manipulación de datos -DML- con el que es posible manipular o procesar dichos objetos. La vista externa es la vista parcial que tiene un usuario de la base de datos. Es la concurrencia de todos los registros externos.

**MODELO DE DATOS**:

Un modelo de datos es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos nos permiten modelar la estructura de los datos, los operadores nos permiten modelar su comportamiento. El termino modelo de datos puede tener dos significados diferentes. El primer sentido lo podemos ver como un lenguaje de programación, para resolver una amplia variedad de problemas. El segundo es como un programa específico escrito en ese lenguaje. Y resuelve un problema determinado. Existen diferentes tipos de modelos de datos:

1. Modelo Relacional.
2. Modelo Orientado a Objetos.
3. Modelo Jerárquico.
4. Modelo de Red.

IMPLEMENTACION:

La implementación de determinado modelo de datos es una realización física, en una máquina real, de los componentes de la máquina abstracta que en conjunto constituyen ese modelo.

**MODELO RELACIONAL**:

Este modelo surgió como una solución a la dependencia de datos que tenían el modelo jerárquico y el modelo en red, siendo un modelo de datos basado en el concepto matemático de relación. Este modelo consta de tres partes fundamentales: la *estructura de datos* –tiene que ver con las relaciones como tales-, *integridad de los datos* –tiene que ver (entre otra cosas) con las claves primarias y externa- y la *manipulación de datos* –tiene que ver con los operadores (restringir, proyectar, juntar, etc.)- .El modelo relacional consta de cinco componentes:

1. Un conjunto abierto de *tipos escalares* (incluyendo en particular el tipo lógico o valor verdadero).

2. Un *generador de tipos de relación* y una interpretación propuesta de dichos tipos.

3. Herramientas para definir *variables de relación* de dichos de relación generados.

4. Una operación *asignación relacional* para asignar valores de relación a las variables de relación mencionadas.

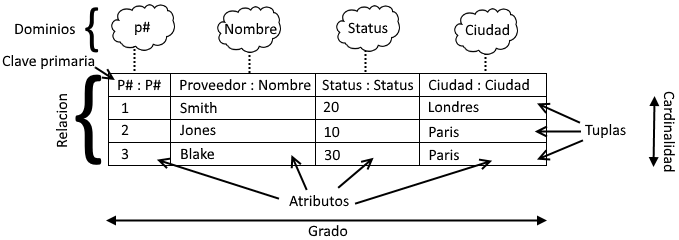
5. Un conjunto abierto de *operadores relacionales* genéricos para derivar valores de relación de otros valores de relación.

VENTAJAS DEL MODELO RELACIONAL:

*Sencillez y uniformidad de criterio*: los usuarios ven a la base de datos como un conjunto de relaciones.

*Sólida fundamentación teórica*: el modelo puede ser definido con rigor.

*Independencia de la interfaz*: los lenguajes relacionales al manejar conjunto de tupas poseen gran independencia de cómo están almacenados.



*Atributo*: en la representación tabular de la relación es una columna de la tabla.

*Tupla*: en la representación tabular de la relación es una fila de la tabla.

*Cardinalidad*: es el número de tuplas de una relación.

*Grado*: es el número de atributos de una relación.

*Dominio*: es un tipo de datos definido por el sistema o un tipo definido por el usuario.

**RELACION**:

Dado un conjunto de n tipos o dominios Ti(i = 1, 2,…, n), que no son necesariamente todos distintos, r es una relación sobre esos tipos si consta de dos partes: un *encabezado* y un *cuerpo*; donde:

El *encabezado* es un conjunto de n atributos de la forma Ai:Ti, donde los Ai –que deben ser todos distintos- son los nombres de atributo de r y los Ti son los nombres de tipo correspondientes (i = 1, 2,…, n);

El *cuerpo* es un conjunto de m tuplas t, en donde y es a su vez un conjunto de componentes de la forma Ai:vi en la cual vi es un valor de tipo Ti; es decir, el valor de atributo para el atributo Ai de la tupla t (i = 1, 2,…, n).

A los valores m y n se los denomina Cardinalidad y grado, respectivamente, de la relación r.

PROPIEDADES DE LAS RELACIONES:

*No existen tuplas duplicadas*: esta primera propiedad sirve para ilustrar el punto de que –en general- una relación y una tabla no son lo mismo, ya que –en general- una tabla puede contener filas duplicadas, mientras que una relación no puede contener una tupla duplicada –ya que si una “relación” contiene tuplas duplicadas, entonces no es una relación por definición matemática-.

*Las tuplas están en desorden, de arriba hacia abajo*: esta propiedad surge del hecho de que el cuerpo de la relación es un conjunto matemático; en matemáticas, los conjuntos no están ordenados. Por lo tanto no hay algo que se llame *la quinta tupla* o *la siguiente tupla*, en otras palabras no existe el concepto de direccionamiento posicional.

*Los atributos están en desorden de izquierda a derecha*: Esta propiedad surge del hecho de que el encabezado de una relación también es un conjunto –de atributos-. Por lo tanto no existe algo como *el primer atributo* o *el siguiente atributo*, en otras palabras, siempre se hace referencia a los atributos por nombre, nunca por posición.

*Cada tupla deben tener definida una clave primaria*.

VARIABLE RELACIONAL – RELACION:

Nos referiremos a *variable relacional*, cuando hacemos mención a la estructura de la misma. O sea al encabezado de una relación, sin importar un valor en particular de su cuerpo.

Nos referiremos a *relación* cuando hacemos mención a un valor específico en un momento dado para una variable relacional. Este valor es un conjunto de N tuplas pertenecientes al cuerpo.

1. Administrador de datos: este conoce la información y las necesidades de la empresa. Su tarea principal es decidir qué datos serán almacenados, además de establecer políticas para mantener y manejar los datos almacenados. [↑](#footnote-ref-1)