Bases de datos

Libros:

-Database System Concepts. 7th Edition, Silbereeschatz,Korth, Sudarshan.

-Fundamentos de Bases de Datos, 5ta edición, Edition Silberschatz, Korth, Sudarchan.

-Fundamentals of Database Systems. Elmasri, Navathe. Addision Wesley,

-An introduction to Database Systems. Vol1 Vol2 C.J. Date, Addison Wesley.

Un Sistema de bases de datos o DBMS es una colección de datos interrelacionados y un conjunto de programas para acceder a ellos.

Según Navathe 97 “

Un sistema de software de propósito general que facilita el proceso de definir, construir y manipular base de datos para diversas aplicaciones”.

***Problemas de sistemas de archivo:***

Redundancia e inconsistencia de datos, cuando la misma información se repite en diferentes lugares, puede causar problemas de sobreutilización de espacio de almacenamiento, y además causar una potencial inconsistencia de datos.

Integridad, los datos almacenados tienen que cumplir ciertas condiciones de integridad de datos (una cuenta de banco no puede ser negativo), estas condiciones pueden cambiar en el transcurso del tiempo, cuando esto pasa, se deben actualizar todos los programas que manipulen la info modificada.

Problemas de Atomicidad, todas las operaciones de una transacción se deben ejecutar o ninguna.

Acceso concurrente a los datos, Para el óptimo uso de los recursos en necesario proveer accesos concurrentes a los datos y esto trae una potencial inconsistencia en los datos.

Seguridad en el acceso a los datos, no todos los usuarios en un sistema de BD deben tener acceso a todos los datos. Utilizando sistemas de archivos esto es muy difícil de garantizar, ya que la gestión de datos no está centralizada.

Aislamiento, ya que los datos están dispersos en varios archivos, y los archivos pueden estar en diferentes formatos, es difícil escribir nuevos programas de aplicación para recuperar los datos apropiados.

Dificultad en el acceso a los datos, cuando se necesita de ciertos datos diferentes de archivos diferentes, la obtención consulta y modificación de los datos no pueden hacerse directamente en forma práctica y eficiente. Tendrían que desarrollarse sistemas de recuperación de datos para realizar esa operación específica, o desarrollar un sistema de recuperación de datos para uso general y ajustarlo de acuerdo a las necesidades.

***Modelo Lógico Basados en Objetos:***

Modelo Entidad-Relación, es solo y exclusivamente un método del que disponemos para diseñar en un gestor de BD. Este modelo se representa a través de diagramas y está formado por varios elementos.

Este diagrama que ayuda a entender los datos y como se relacionan entre ellos, debe de ser completado con un pequeño resumen con la lista de los atributos y las relaciones de cada elemento.

Elementos:

**Entidad**, estas representan cosas u objetos (reales o abstractos) que se diferencian claramente entre si

Como por ejemplo se podría crear las siguientes entidades:

-Coche (objeto físico): contiene la información de cada taller.

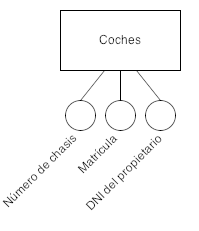
-Empleado (objeto físico): información de los trabajadores.

-Cargo del empleado (cosa abstracta); información de la función del empleado.



Estas entidades se representarían así:

Los atributos definen o identifican las características de entidad (**es el contenido de esta entidad**). Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Los atributos pueden ser de distintos tipos (**numéricos, texto, fecha, etc.**).



Por ejemplo, los atributos de nuestra entidad coche, nos data información sobre los coches de nuestro supuesto taller.

Otros posibles atributos serian, numero de chasis, la matricula, DNI del propietario, marca, modelo y muchos más que completen la información de cada coche.

Los atributos se representan como círculos que descienden de una entidad y no es necesario representarlos todos, sino los más significativos.

En un modelo relacional un ejemplo de tabla adentro de un base de datos podría ser: 

**Relación**, es un vínculo que nos permite definir una dependencia entre varias entidades, nos permite exigir que varias entidades compartan atribuidos de forma indispensable. Por ejemplo, los empleados del taller (**de la entidad ‘empleados’**) tienen un cargo (**según la entidad ‘cargo del empleado’**). Un atributo de la entidad ‘Empleados’ especifica que cargo tiene en el taller, y tiene que ser idéntico al que ya existe en la entidad ‘Cargo del empleado’.

Las relaciones se muestran en los diagramas como rombos, que se unen a las entidades mediante líneas.

Mediante una tabla quedaría representado de la siguiente forma.



**Relaciones de cardinalidad**, existen distintos tipos de relaciones según como participen en ellas las entidades, en el ejemplo cada empleado puede tener un cargo, pero un mismo cargo pueden compartir varios empleados. Esto complementa a las representaciones de las relaciones, mediante un intervalo en cada extremo de la relación que especifica *objetos* o cosas (de cada entidad) pueden intervenir en esa relación.

**UNO A UNO**, una entidad se relaciona únicamente con otra y viceversa. Por ejemplo, si tuviésemos una entidad con distintos chasis y otra con matrículas deberíamos de determinar que cada chasis solo puede tener una matrícula (y cada matricula un chasis).

**UNO A VARIOS O VARIOS A UNO,** determina que un registro de una entidad puede estar relacionado con varios de otra entidad, pero en esta entidad solo puede existir una vez.

**VARIOS A VARIOS**, determina que una entidad puede relacionarse con otra con ninguno o varios registros y viceversa. Por ejemplo, en el taller un coche puede ser reparado por varios mecánicos distintos y esos mecánicos pueden reparar varios coches distintos.

Los indicadores numéricos indican el primero es número mínimo de registros en una relación y posteriormente el máximo (si no hay límite se presenta con una ‘n’).

**Claves**, es el atributo de una entidad, al que le aplicamos una restricción que lo distingue de los demás registros (**no permitiendo que el atributo especifico se repita en la entidad**) o le aplica un vínculo. Existen distintos tipos,

**Superclave:** aplica una clave o restricción a varios atributos de la primera entidad, para así asegurarse que en su conjunto no se repitan varias veces y así no poder entrar en dudas al querer identificar un registro.

**Clave Primaria:** identifica inequívocamente un solo atributo no permitiendo que se repita en la misma entidad. Como sería la matricula o el número de chasis de un coche (no puede existir 2 veces el mismo).

**Clave externa** o **clave foránea:** tiene que estar estrictamente relacionado con la clave primaria de otra entidad, para así exigir que exista previamente esa clave. Por ejemplo, un empleado del taller indispensablemente tiene que tener un cargo (**que lo representamos numéricamente**), por lo cual si intentásemos darle un cargo inexistente el gestor de bases de datos nos devolvería error.

Modelo de Bases de Batos Orientados a Objetos, este agrupa la información en paquetes relacionados entre sí, los datos de cada registro se combinan en un solo objeto, con todos sus atributos. De esta manera toda la información está disponible en el objeto, ya que sus datos quedan agrupados en lugar de distribuidos en diferentes tablas. En los objetos no solo pueden guardarse los atributos, sino que los métodos, lo que refleja la afinidad de estas bases de datos con los **lenguajes de programación orientados a objetos**, al igual que en estos, cada objeto presenta un conjunto de acciones que pueden llevarse a cabo.

El modelo de BD con el que decidamos trabajar dependerá del uso que queramos darle. Las bases de datos de objetos son especialmente adecuadas si ya estamos trabajando con **lenguajes de programación orientados a objetos**, como java, porque los objetos del código fuente se pueden integrar fácilmente en la base de datos. Si recurrimos a una base de datos relacional, lo que suele ser más frecuente, nos costara incorporar objetos complejos a la estructura tabular.

Uno de los inconvenientes de este modelo es que su uso está **poco extendido**. Aunque se conoce desde la década de 1980, hasta ahora solo se han desarrollado unos pocos SGBD para bases de datos orientadas a objetos.

Lo que supone una ventaja en ciertas situaciones puede convertirse en un inconveniente en otras, la complejidad de los objetos garantiza que hasta las consultas y anotaciones más complejas puedan llevarse a cabo mucho más rápido que en los modelos relacionales. Sin embargo, si los procesos son sencillos en comparación, no se puede prescindir de la estructura compleja, lo que puede conllevar a problemas de **ralentización.**

**Modelo jerárquico**: la información se representa con colecciones de registros y las relaciones con punteros. Los registros se organizan como árboles.

**Modelo de red**: ídem al anterior pero los registros se pueden organizar como grafos.

**Modelo Relacional**: Los datos se representan como tablas y las relaciones entre los datos también.

***Motores de bases de datos SQL***

Garantizan propiedades ACID, que utilizan SQL como lenguaje, pueden ser:

-Cliente-Servidor (ambiente multiusiarios):

-**MySQL(InnoDB, etc.).**

**-Oracle.**

**-Postgres.**

-Embebidas (Incluido en las aplicaciones, son útiles para aplicaciones monousuario):

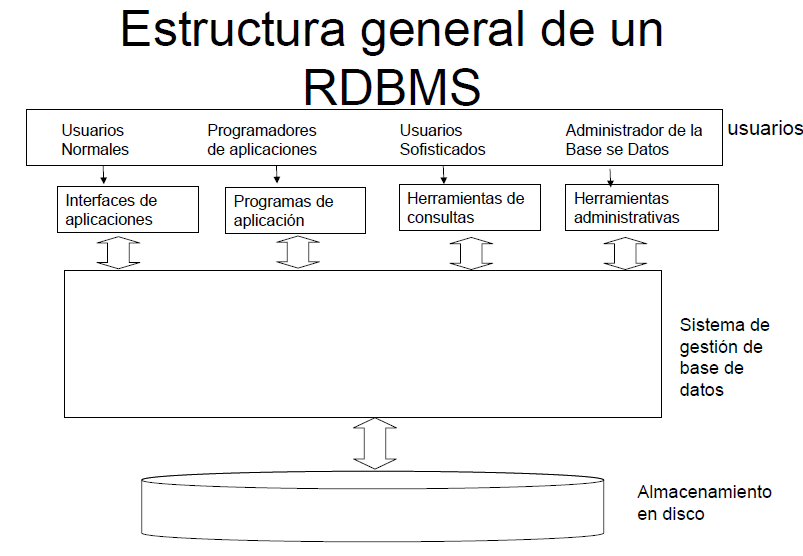
**-Sqlite(No posee chequeo de tipos).**

**-MySQL(librería MySQLd).**

**Bases de datos NoSQL**, estas bases de datos NO tienen esquema, No garantizan las propiedades ACID, No perminen JOINs, escalan horizontalmente, entre otras caractiristicas.

Existen orientadas a:

-**Documentos (mongoDB formatos BSON).**

** -Columnas(Cassandra).**

**-Etc.**