Introducción a las Bases de Datos

Autor: Alfonso Palomares
Corrección: Javier Rebagliatti

1. Las Bases de Datos

1.1. Introducción

Cada paso que damos a diario estamos topándonos con una base de datos. Sea a la hora de ir al supermercado y validar los precios de los productos con el nomenclador de precios, ya sea a la hora de realizar un viaje y definir el precio del pasaje.
En la mayoría de los casos, encontramos que los datos con los que tratamos, tienen ciertos agrupadores y que a partir de ellos definimos en consecuencia según la necesidad del caso. Todos estos datos, pueden permanecer almacenados en distintos medios, ya sea papel, nomencladores, o en sistemas digitales.
Algunos ejemplos de aplicación digital para el almacenamiento de datos pueden ser las transacciones digitales en un banco, como también liquidaciones digitales de sueldo.
Los anteriores ejemplos son aplicaciones digitales de problemas que anteriormente se resolvían de otra forma, pero que conceptualmente conllevan el mismo concepto y la misma solución, reemplazando el sistema informático al manual.
Existen además, otras aplicaciones no tan convencionales para la utilización de bases de datos, tal es el caso de las bases de datos multimediales, para almacenar fotos, videos, pdf's, etc, bases de datos con información geográfica (GIS), sistemas de procesamiento analítico (OLAP), entre otros.

Definiremos entonces, que una **base de datos** *es una colección de datos que cobran sentido a partir de relaciones entre ellos*. Estos datos serán parte de un sistema, el cual podrá representar una porción de datos de un universo de datos mayor.

1.2. Historia del Almacenamiento de Datos

Para comenzar este relato, comenzaremos señalando el origen de las bases de datos en los sistemas organizacionales de las antiguas bibliotecas, en las ciudades mesopotámicas, registros de censos, diversas actas notariadas o nomencladas, entre otros casos, que representen la definición de base de datos, es decir, que sean un grupo de datos u objetos que tengan sentido entre sí por estar relacionados.
Para el caso de las bibliotecas, los objetos eran los libros, agrupados por autor, tema, versión, tipo de formato en el que se encontraba, su estado, entre otros atributos de dichos objetos. Mientras que las relaciones que los involucraban estaban dadas por alguno de estos agrupadores ya mencionados. Esta organización de las publicaciones que se mantienen, se llevaba y se lleva a cabo mediante una base de datos para poder aprovechar estos datos.
Lo mismo ocurría con el caso de los registros censales. Estos registros, constan de atributos como cantidad de personas censadas, lugar de procedencia, entre otros datos. Y se pueden relacionar con otros censos, para poder formar alguna decisión u opinión al

respecto, por tanto, podemos también decir que tenemos datos relacionados que tienen un sentido aplicado.
Los comienzos de la informática están plenamente ligados al intento de gestionar más rápida y eficientemente las bases de datos que existían en otros medios, tales como archivos en papel.
Un claro ejemplo, es con los datos censales. Una de las primeras aproximaciones a una computadora, es la máquina Hollerith, inventada en 1884, para procesar censos en Estados Unidos, que ordenaba y contaba las tarjetas censales. Fue muy famosa en la Alemania Nazi, ya que se procesaron datos de los judíos germanos en 1939.

Los conceptos sobre los datos no se alteraron mucho, pero si la informática, los métodos de procesamiento, como la velocidad del cálculo.
Como acabamos de señalar, la informática comenzó almacenando sus datos en las tarjetas perforadas, como intento de automatización en el proceso de censos y notariales, luego llego a nosotros la posibilidad de almacenar información en forma de cinta, luego los discos sucedieron a la cinta, tanto ópticos como magnéticos. Con ésto tenemos, a grandes rasgos la evolución del soporte de la información, donde no entraremos en detalle porque no es materia del presente texto.
Con respecto a la organización de los datos, podemos tomar dos grandes grupos de organización de datos. Los de lectura secuencial y los de lectura directa o no secuencial, siendo los primeros los que necesitaban recorrer toda la información que se tuviera hasta reconocer aquella que se estaba buscando y los segundos logran acceder a esta información directamente. Empecemos a ver esta evolución hasta poder detenernos en el sistema que gestiona nuestras bases de datos.

Archivos de texto

Desde cualquier editor de textos, o desde un lenguaje de programación, se puede almacenar muy fácilmente información en un archivo de texto, por tanto es un muy buen punto de partida para evaluar su uso a la hora de pensarlo como una base de datos.
Entonces, tratemos de pensar en los archivos de texto como medio de almacenamiento, para lo cual, rápidamente encontraremos que deberíamos usar un separador tanto para identificar el fin o el comienzo de cada parte de un dato, como el comienzo de un nuevo dato.

Un ejemplo claro sería tener grupos de bebidas, con el nombre, el color, si tiene burbujas o no, etc, delimitar con un pipe (|) cada atributo de una bebida, y para comenzar un nueva bebida podríamos utilizar el retorno de carro. Esto nos daría un archivo formado de la siguiente manera:

```
nombre    | color    | burbujas
coca cola | negro    | si
pepsi     | negro    | si
vino      | blanco   | no
```

Otro ejemplo, podría ser tener un conjunto de aplicaciones para nuestra computadora.

Donde almacenamos el nombre de la aplicación, la empresa que lo creo, la fecha de lanzamiento y en que sistema operativo puede funcionar. Podríamos ahora, utilizar las comas como separador de los atributos de cada aplicación, y el punto y coma para separar diferentes aplicaciones. Quedando de la siguiente manera:

```
office, microsoft, 2011, win7; winzip,
WinZip Computing, S.L, 2004, winXP;
photoshop 7, adobe, 2010, winXP;
```

Otra posibilidad, sería usar tabuladores, espacios, o cualquier otro caracter de corte para diferenciar los datos, teniendo en cuenta que no pueda estar contenido en el texto que forma parte del dato en sí, como un nuevo separador diferente para separar las tuplas[1]

Como se ve, para lograr encontrar algo dentro de un archivo de texto, se debería recorrerlo registro a registro, o tupla a tupla, hasta dar con lo que se esta buscando, teniendo la posibilidad de que tengamos que recorrerlo todo, ya que no se tiene en claro donde se encontrará lo buscado. Además, para asociar los datos de una tupla a una estructura dentro de nuestro lenguaje de programación, debemos desglosar la tupla para colocar dentro de la estructura los datos correspondientes, además de convertir los datos de tipo texto al formato de destino. Para el caso anterior, para completar una estructura de dos atributos tipo texto y uno numérico, debemos realizar cortes en la cadena de texto de entrada hasta detectar el separador de corte, para este caso el pipe, y mapear el dato con la estructura. Para el caso del documento, se debe hacer una conversión de datos.

Archivos Binarios

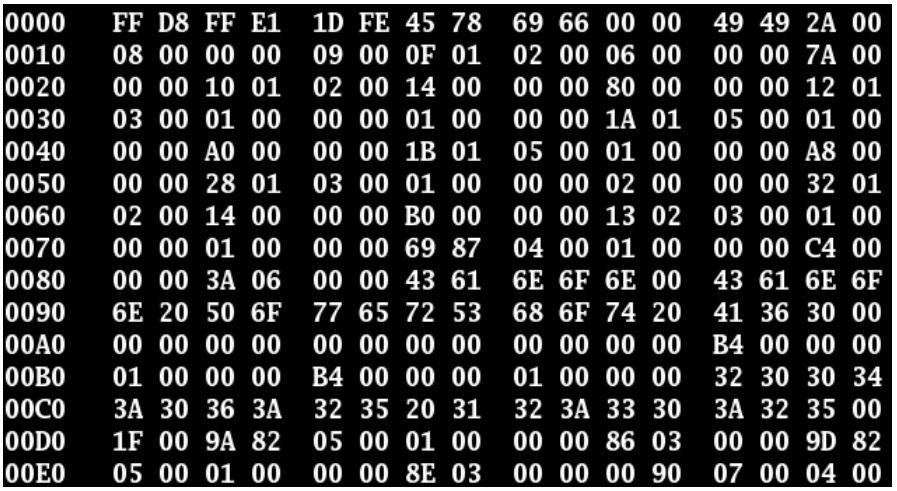
Como se desprende claramente de lo anterior, realizar el almacenamiento en uno o más archivos de texto no es la mejor solución. Tiene no solo aparejados los problemas en cuanto a la búsqueda de datos, sino también problemas en la seguridad de los mismos, dado que cualquiera, con los permisos sobre los archivos indicados, puede alterar un archivo de texto, u obtener el detalle de dicha información.
Por estos motivos, surgió la necesidad de generar archivos binarios para almacenar la información.
Al trabajar con archivos del tipo Binario, se gana a la hora de realizar el mapeo entre estructuras de datos y la información, ya que esta relación es directa, sin tener que realizar conversiones. Además, la forma de corte de los registros está ligada íntimamente al formato de la estructura, como también así el fin de cada registro. Esto ayudó a optimizar espacios, y a lograr mayor seguridad sobre la información guardada en dichos archivos, ya que no es simple interpretar un archivo binario sin tener el formato de las estructuras.
Para ejemplificar, tomaremos una estructura en lenguaje C. Para lo cual, intentaremos describir brevemente un vaso con dicha estructura. Luego, varias instancias de esta estructura se almacenarán en el archivo binario.

[1]Un conjunto de elementos de un dato que forman un registro.

Estructura

```
struct Vaso {
    long nroSerie;
    int altura;
    bool tinePie;
    float diámetro;
} vaso;
```

Ejemplo de parte de la estructura vista anteriormente en formato binario, almacenando varias instancias de la estructura mencionada. El formato para representar el archivo binario es el Hexadecimal.



0000	FF	D8	FF	E1	1D	FE	45	78	69	66	00	00	49	49	2A	00
0010	08	00	00	00	09	00	0F	01	02	00	06	00	00	00	7A	00
0020	00	00	10	01	02	00	14	00	00	00	80	00	00	00	12	01
0030	03	00	01	00	00	00	01	00	00	00	1A	01	05	00	01	00
0040	00	00	A0	00	00	00	1B	01	05	00	01	00	00	00	A8	00
0050	00	00	28	01	03	00	01	00	00	00	02	00	00	00	32	01
0060	02	00	14	00	00	00	B0	00	00	00	13	02	03	00	01	00
0070	00	00	01	00	00	00	69	87	04	00	01	00	00	00	C4	00
0080	00	00	3A	06	00	00	43	61	6E	6F	6E	00	43	61	6E	6F
0090	6E	20	50	6F	77	65	72	53	68	6F	74	20	41	36	30	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	B4	00	00	00
00B0	01	00	00	00	B4	00	00	00	01	00	00	00	32	30	30	34
00C0	3A	30	36	3A	32	35	20	31	32	3A	33	30	3A	32	35	00
00D0	1F	00	9A	82	05	00	01	00	00	00	86	03	00	00	9D	82
00E0	05	00	01	00	00	00	8E	03	00	00	00	90	07	00	04	00

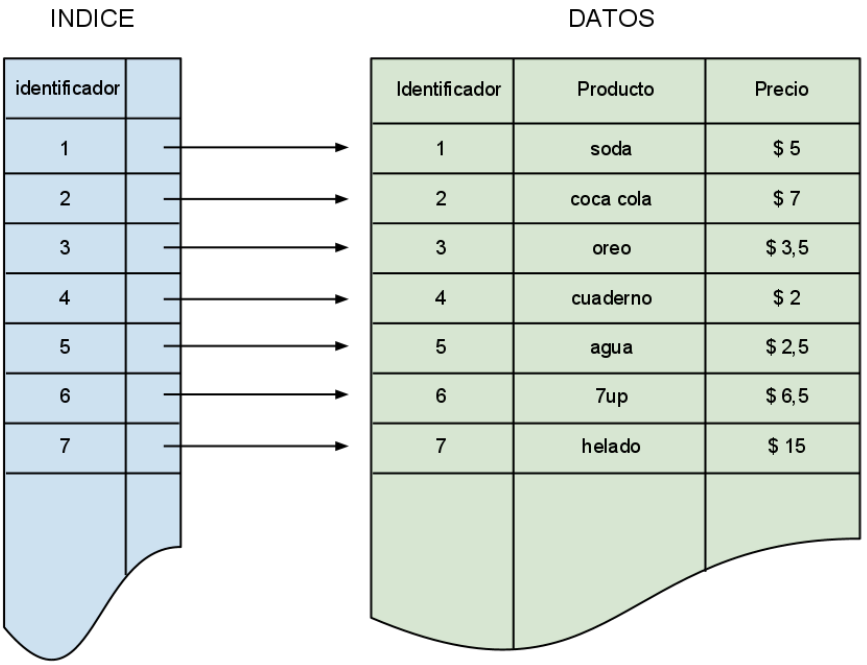
Estos archivos agregan otra problemática. Son dependientes no sólo del lenguaje de programación usado para generar dicho archivo, sino que además, necesitamos conocer el programa que genera dicho archivo para poder interpretar la estructura, como el orden y el tipo de dato de cada atributo dentro del registro.

Pero aún no hemos resuelto el problema del acceso a la información, ya que estos archivos se acceden también de forma secuencial.

Archivos Binarios indexados

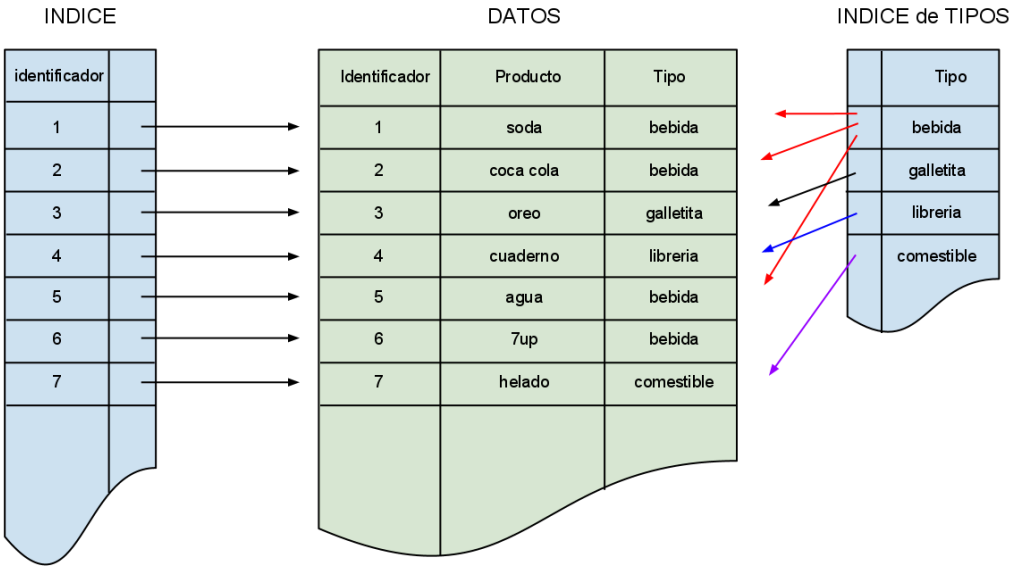
Primero, nos toca definir los índices. Un índice es un conjunto de señaladores o punteros que asocia un valor dado a uno o más registros. Por ejemplo, dentro de un libro, el índice nos muestra los títulos como punto de entrada y nos apunta el destino con el número de página donde encontramos la información detallada para ese título. Esto es rápidamente transportable a un archivo binario y su o sus índices, donde el libro sería el archivo binario y las entradas del índice del libro serían cada uno de los señaladores o punteros del archivo de índice al registro correspondiente.
Teniendo la información que identifica cada registro en el índice, nos ayuda a encontrar el

registro dentro del archivo binario muy fácilmente, ya que podemos recorrer secuencialmente el archivo de índice y con el señalador hacer un salto dentro del archivo binario de datos para encontrar el registro buscado. Esto acelera mucho la lectura de datos, ya que siempre la información dentro de los archivos de índices es menor que dentro del archivo de datos.
Además, en caso de querer tener los datos ordenados, solamente nos hace falta ordenar el índice de los mismos, ya que teniendo luego los punteros a los datos no nos importa como esté ordenado el archivo de datos, tampoco nos importa si tiene espacios entre registros. Lo único que se debe cuidar es que los señaladores del índice se mantenga asociados siempre al mismo registro con los que inicialmente fueron relacionados.
Para verlo gráficamente, podemos ilustrar este tipo de índices de la siguiente manera:



Debemos contemplar la posibilidad de que para un mismo archivo binario, tener varios índices, que determinen diferentes agrupaciones. Esto, también es visible en un libro, donde uno puede observar, un índice de capítulos, otro por temas y hasta algunos por conceptos.

Alguno de estos índices, tienen referenciados más de un registro, lo cual nos permite utilizarlo de agrupador. Por ejemplo, si almacenamos la información de las bebidas del gráfico anterior, pero agregando ahora el tipo de bebidas, quedando identificador, producto, y tipo en cada registro, podríamos tener un índice asociado a los identificadores, y también otro asociado al tipo de bebida, conteniendo este último sólo los tipos diferentes de todo el archivo binario y cada entrada de dicho índice muchos apuntadores a todos los registros que contengan dicho tipo de bebida. Claramente, cada entrada en el índice de tipos de bebida, nos permite alcanzar todos los registros con idéntico tipo.
Para poder tener un análisis completo de esta explicación, extenderemos el gráfico anterior, agregando un nuevo índice.



Parecería que hemos resuelto todos los problemas que nos hemos propuesto resolver. Tenemos archivos binarios que son difíciles de descifrar, tenemos archivos de índices que nos permite que la información sea accedida más rápido, tenemos estructuras fácilmente mapeables a nuestro lenguaje de programación. Pero todavía hay algo que nos falta.
Todavía estos conjuntos de datos son dependientes del lenguje y del programa para el cual fueron creados. Para resolver esto, debemos generar una capa entre nuestros programas y la gestión de los datos. Por lo cual, damos la bienvenida al sistema gestor de bases de datos.

1.3. Sistema Gestor de Bases de Datos (SGBD)

Fueron varios los factores que llevaron a crear un sistema que gestione nuestros datos. Inicialmente, con el deseo de poder utilizar los datos de una aplicación desde diferentes enfoques, luego, existiendo la necesidad de resguardar nuestros datos, teniendo en claro con quien y como compartíamos cada porción de nuestra información.
Además, siempre en determinado momento, surge la necesidad de matizar datos desde diferentes orígenes, lo cual hace que tengamos que cruzar diferentes bases de datos.
Esto, es muy complicado hacerlo desde cualquiera de nuestros sistemas que programamos en un lenguaje de alto nivel, ya sea con archivos de texto o binarios, con o sin índices.

Por lo cual, se detectó la necesidad de generar un sistema independiente de nuestros programas, donde almacenar los conjuntos de datos, delegando en este sistema la administración de los archivos, las formas de accesos, la forma de almacenamiento y, sobretodo, el lenguaje para la interacción de estos datos.
Es decir, que al utilizar un SGBD, deberemos definir nuestros datos conforme a las estructuras que el SGBD nos proponga, accediendo a esos datos según los permisos que dicho sistema tenga formulados para nuestros sistemas y la operación de los datos de dicho esquema será realizada, no ya con el lenguaje de programación de nuestro programa, sino con el lenguaje común para todos los esquemas definido en el SGBD.

Trabajando de esta manera, el sistema gestor puede trabajar con más de una base de datos a la vez, actuando de forma transparente para cada una de ellas. Además, más de un programa puede interactuar con dichos datos, sin importar con que lenguaje este realizado cada programa, necesitando simplemente los accesos en el SGBD a los datos utilizados.

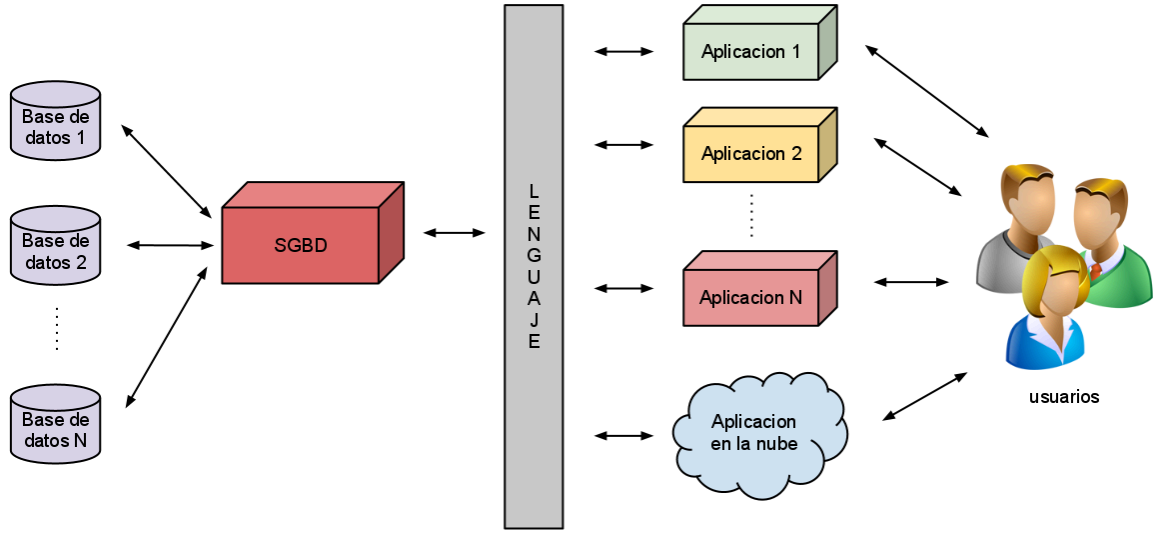
Las ventajas de usar un SGBD no solamente residen en lo expuesto anteriormente, sino que además, teniendo en cuenta que son sistemas específicos, están altamente optimizados, permitiendo la aplicación de teorías sobre la normalización de los datos para su mejor utilización, logrando la consistencia de los datos, es decir que nos garantiza que todas las partes de los datos se logran guardar de forma atómica. Podemos señalar también, que la utilización de un gestor nos permite tener accesos muy veloces a la información.

Estas ventajas devienen de la maduración en que se encuentran los SGBD, con varias décadas ya en el mercado y la cantidad de teoría en el manejo de información aplicada a estas soluciones.

Con respecto a la forma de almacenar los datos, cada SGBD tiene sus propias características, pero formalmente, todos toman los registros de cada grupo de datos como una colección de registros, permitiendo interactuar entre diferentes registros y colecciones según la necesidad del resultado esperado. La gran mayoría genera archivos donde internamente maneja sus datos conforme a un propio sistema de archivos, tal como hace un sistema operativo.
Esto le permite utilizar su propia administración de los archivos internos, aplicar sus propias reglas de permisos y realizar optimizaciones al uso y ubicación de los datos.

Es quien coordina, además, el acceso concurrente a los datos a la hora de la lectura y quien garantiza que los datos guardados conserven el orden en que fueron realizados dichos movimientos de datos.

Podemos definir, entonces a un **SGBD** *como un sistema que administra los datos de forma eficiente y compacta, que nos ofrece un lenguaje común y determinado para el manejo de esos datos y que se ocupa de administrar los permisos sobre esos datos*.



Con respecto al lenguaje común propuesto en el sistema, volveremos más adelante, lo propio ocurrirá con la forma en que trabaja los datos dentro de su propio sistema de archivos y demás temas. Ayudaremos a guiarlos para realizar un modelo correcto de su base de datos para lograr explotar al máximo los beneficios presentados del SGBD.

1.4. Cuando no usar un SGBD

Parecería difícil determinar cuándo se puede dar el caso donde no necesitemos un sistema gestor de datos. Claramente, cada vez que necesitemos trabajar con una base de datos, acudiremos a un SGBD para que nos administre nuestra base de datos.
Pero entonces debemos preguntarnos, ¿cuando no necesitamos una base de datos?.

La respuesta la encontramos en la misma definición. Si tenemos un conjunto de datos, que no está relacionado o no necesita estar relacionado para tener sentido, no necesitamos una base de datos.
Por ejemplo, un simple listado donde cada uno de las filas encuentre toda su definición dentro de la misma fila, si necesidad de complementar dicha información de la fila con otra fila o con al fila de otro listado, podemos decir que es un problema que no necesita de una base de datos para resolverse. Como este, podemos encontrar muchas otros casos.

1.5. Ejemplos de SGBD

Existen en el mercado, muchos gestores de bases de datos, siendo unos destacados en un punto y otros en otro.
Para este caso, los agruparemos por si son libres o no, omitiendo por el momento nuestra preferencia o nuestro análisis subjetivo sobre cada uno de los gestores aquí propuestos.
En el caso de omisión de algún gestor, es simplemente por desconocimiento del mismo, sin tener animosidad alguna.

SGBD libres

- PostgreSQL
- MySQL
- SQLite
- Firebird
- DB2 Express-C
- Apache Derby
- MariaDB

SGBD pagos

- dBase
- Fox Pro
- IBM DB2
- IBM Informix
- Microsoft SQL Server
- Oracle
- PervasiveSQL
- Sybase

1.6. Repaso.

Volvamos rápidamente sobre los conceptos aquí vistos.

1. Una base de datos es un conjunto de datos que tienen relación entre sí.
2. Un registro o tupla es un dato en particular, divisible en cada uno de los componentes del mismo.
3. La evolución para almacenar datos, fue desde los archivos de texto a las bases de datos, pasando por los archivos binarios y el descubrimiento de los índices.
4. Un SGBD se ocupa de administrar los datos, de forma eficiente y compacta. Se ocupa de administrar los usuarios que acceden a la misma, como también que información puede observar cada usuario. Provee además un lenguaje estándar para trabajar con los datos que contiene.