

Bases de datos – Introducción a los modelos

Índice del contenido

- ¿Qué es una base de datos?
- Características de las bases de datos
- Clasificaciones de las bases de datos
- Historia de las bases de datos
- Modelos de las bases de datos
- Modelo relacional
- Beneficios de las bases de datos relacionales:
- Modelo no relacional
- Tipos de bases de datos (modelos no relacionales)
- Beneficios de las bases de datos no relacionales
- Comparamos modelos relacional vs no relacional – ejemplo
- En resumen:
- Modelo relacional vs modelo no relacional
- Modelo relacional
- Modelo no relacional
- ¿Y cuando usar un modelo u otro?

Es momento de **aprender a crear, diseñar, administrar y trabajar con bases de datos en python**. Pero antes déjame decirte que si llegaste hasta aquí es casi seguro que has leído y practicado [programación estructurada](#) y [orientada a objetos](#), en python además de haber aprendido a utilizar un [software de control de versiones](#).

Antes de comenzar déjame dejarte en claro que la lectura de este post puede no ser muy sencilla de comprender, pero te aseguro que si lo lees detenidamente e investigas por tu cuenta cada concepto o palabra que te suene extraña saldrás con una muy buena **introducción a la comprensión de las bases de datos**. Este es el primer post he recopilado mucha información de diferentes fuentes y es parte de una serie dentro del 3er módulo de pythones "Desarrollo" perteneciente al apartado de bases de datos.

¿Qué es una base de datos?

La wikipedia la define como:

Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Ahora bien, esta es una definición muy genérica, me imagino que te darás cuenta!. Porque una biblioteca puede ser una base de datos, un jodido álbum de fotos de tus vacaciones es una base de datos. Hasta un museo, es una base de datos.. Claro, pero en informática una base de datos está definida (te coloco múltiples definiciones) algo así:

*1.-Una **base de datos** es una colección de información organizada de forma que un software pueda seleccionar rápidamente los fragmentos de **datos** que necesite.*

*2.-Una **base de datos** es un sistema de archivos electrónico. Las **bases de datos** tradicionales se organizan por campos, registros y archivos.*

3.-Una base de datos es un conjunto de datos almacenados y organizados con el fin de facilitar su acceso y recuperación mediante el uso de un ordenador.

Una base de datos es una serie de datos organizados y relacionados entre sí los cuales son recolectados y explotados por un software o sistema de información.

Bien ya que te ha quedado medianamente la definición es momento de preguntarnos **¿Y para qué sirven?.**

¿Y qué harías si no existieran y tuvieses que recopilar información en un formulario?. Claro que eso no es problema, si tiene 4 o 5 campos los guardamos en un archivo de texto y ya. ¿No?.. Claro, ahora imagínate un censo.. Sí, ese que se hace cada par de años.. ¿Cómo diablos organizamos y filtramos todos esos datos?. Pues para ello existen las bases de datos, para facilitar todo ese desmadre!

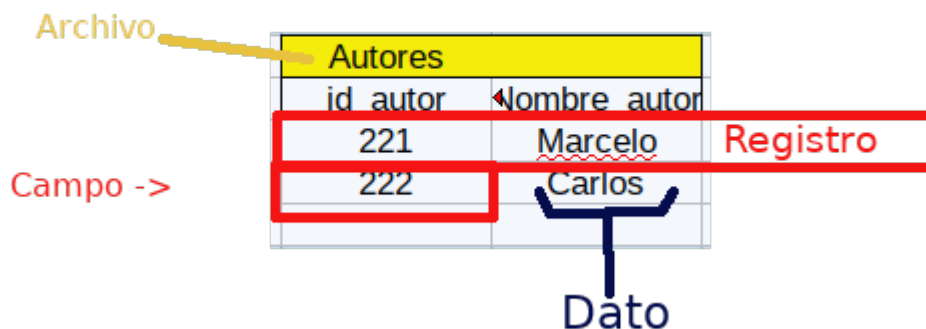
Veamos las características, beneficios y clasificación de las mismas!.

Así entonces decimos que **una base de datos se organiza en campos, registros y archivos:**

Un Campo: Es el área de almacenamiento que nos brinda la base de datos para almacenar datos de un tipo específico. Por ejemplo datos de tipo entero.

Un Registro: Es una colección de datos iguales o que están relacionados y pueden o no ser de tipos diferentes.

Un Archivo: Es un conjunto de registros (datos relacionados de diferentes tipos ubicados en campos.)

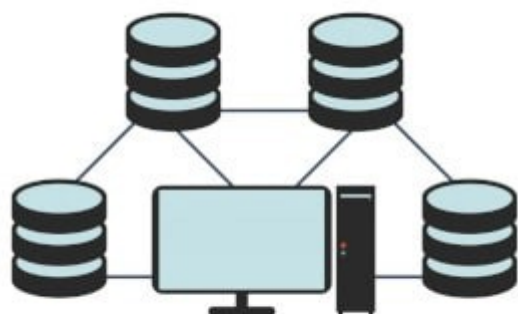


Así un conjunto de archivos que a su vez cada uno es un conjunto de registros, que a su vez son un conjunto de campos relacionados y ordenados de diferentes tipos de datos **conforman una base de datos.**

Para hacértelo más fácil si no tienes ni la más remota idea de bases de datos imagina que estás trabajando con un conjunto de archivos en excel dentro de una carpeta que almacenan estos datos como en la imagen y los cuales están relacionados, por ejemplo dentro de una empresa. E imagina también que existe un programa que hace "consultas" a estos ficheros buscando darte los datos que tu le pidas!.

Características de las bases de datos:

- **Nos permiten acceder y consultar los datos rápida y fácilmente.**
- **Nos permiten almacenar enormes volúmenes de datos.**
- **Permiten ser accedidas fácilmente por cualquier software o sistema.**



- **Control de concurrencia:** Permite que varios usuarios visualicen y modifiquen la información al mismo tiempo.
- **Proporcionan mecanismos de seguridad mediante usuarios, roles y contraseñas.**
- **Las bases de datos cuentan con control de redundancia (evitar la duplicación de determinados datos, por ejemplo tener un producto dos veces con diferentes precios. Vaya lío cuando alguien venga a comprar!)**

Clasificaciones de las bases de datos:

Así como son de útiles las bases de datos existen diversos tipos de bases de datos para diferentes propósitos en el manejo de datos. Y aquí es donde se pone interesante, una base de datos puede clasificarse “según”:

- Si son **estáticas o dinámicas**: Puede que existan **bases de datos estáticas**, es decir aquellas en las que los datos no pueden ser modificados, son de solo lectura, para consultas únicamente. Y existen las **bases de datos dinámicas** que son aquellas que se utilizan para lectura y escritura a la vez, de las cuales podremos consultar, modificar y añadir datos.
- Si son **centralizadas o distribuidas**: Podemos tener una base de datos en un único servidor, o una base de datos formada por varias partes (particiones) alojadas en diferentes servidores o nodos en diferentes puntos geográficos. La organización de bases de datos distribuidas es la más actualizada a día de hoy!.
- **Clasificación por el teorema de CAP**: Esta es una de las clasificaciones más importantes para aquellos que nos dedicamos al desarrollo de software, pues esta actividad comprende también el diseño y elección del modelo de gestión y bases de datos. El teorema de CAP está enfocado a **bases de datos distribuidas (múltiples bases de datos NO alojadas en un solo nodo u ordenador / servidor.)**. El teorema de **CAP** las clasifica según su:

1.- Consistencia: Los datos deben “Coincidir” en todos los nodos o servidores. Si tenemos múltiples bases de datos en diferentes servidores, en ellos los datos deben ser iguales, no puede haber diferentes valores para el precio de un mismo producto por ejemplo (recurriendo al ejemplo anterior).

2.- Disponibilidad: Se debe poder seguir consultado o almacenando datos aunque alguno de los nodos no esté disponible. Debemos poder por ejemplo consultar el precio de un producto almacenado en una base de datos por más que por ejemplo el administrador de un servidor que funcionaba como nodo haya limpiado el mismo con agua y jabón.

3.-Tolerancia a particiones: Permite que la base de datos esté particionadas o divididas en diferentes nodos / servidores. Y funcionen correctamente a pesar de ello.

Pero, **este teorema también indica que en sistemas de bases de datos distribuidos no se puedan cumplir las tres características.** Y por ello existen diversos modelos de almacenamiento y gestión de bases de datos que permiten cumplir alguna de las características con éxito pero flaquean en otras. Ahora, en adelante veremos estos **modelos (relacional y no relacional) aunque existen más** y podrás dilucidar en la diferencia de uno con otro en cual de estas tres características fallan. Pero te iré introduciendo a ello poco a poco a medida que te cuento la historia y el nacimiento del **modelo relacional.**

Historia de las bases de datos.

Bien para comprender mejor las cosas es necesario (a menos que tengas más de 50 años) contarte un poco de historia y darle su debido agradecimiento a una persona, que aunque tu creas que todo es difícil, vas a querer darle un abrazo a este tipo. De quien hablo es de [Edgar Frank Codd](#). Y entenderás porque en un momento.. Primero déjame contarte una historia que seguro si les preguntas a tus padres o abuelos podrán confirmar..

Antes de que apareciera este tipo en los maravillosos 70' la informática se encontraba en su despertar con la aparición de los ordenadores de uso doméstico. Unas porquerías enormes que apenas tenían unos jodidos megas de RAM y eran más víctimas de golpes e insultos que de utilidad.. Seguro recordarás en la casa de la abuela o habrás visto una foto de estos enormes truños..



Pues por aquellas épocas estaban naciendo los

lenguajes de programación y el almacenamiento era muy escaso, si bien sabrás un disco duro no era lo que es hoy en día que tienen un reducido tamaño físico y pueden alojar una gran cantidad de información. Más bien se trabajaba con cintas, y comenzaban a aparecer los discos floppy y toda esa baratija que pronto serán una reliquia y se pagará fortuna por uno de ellos. Pues sucedía que los programadores de aquella épocas tenían (por llamarlo así) "el libre albedrío" de programar y diseñar la gestión y almacenamiento de los datos como se les diera la real gana. **Y los programas repartían la información en el disco duro en campos y registros dentro de ficheros.**

Como si tuvieras 200 archivos de texto repartidos por todo el pc, donde para relacionar un dato con otro se utilizaban punteros (direcciones físicas dentro de ficheros). Porque en estas épocas se recurre a los **modelos jerárquicos** (estructuras con relación padre-hijo que generaban una amplia redundancia) y al posterior **modelo de red** (estructura de relación de todo con todo lo cual solucionaba la redundancia pero era extremadamente complejo.)



Así que imagínate, cuando había que agregar nuevos datos había que hacer sitio, re-calcular punteros y toda esa frustración en un lenguaje como **Cobol**. Los programas una vez creados se hacían más y más complejos, lentos, extremadamente difíciles de mantener por lo que las empresas debían gastar fortunas en programadores e ibuprofeno para sus dolores de cabeza. Todo esto era una mezcla de fideos y spaghetti revolcados en salsa de tomate donde además para representar la relación entre ficheros, por ejemplo (alumnos, profesores, materias) se debía crear otro fichero con sus punteros y otro programa que le recorra los caminos marcados por esos punteros para llegar de un registro a otro. **Era ENFERMIZO!**

En resumen:

- La información se repartía en el disco duro en campos y registros dentro de ficheros que necesitaban de otros ficheros para establecer una relación.
- Para representar la relación entre uno y otro se utilizaban punteros.
- Así teníamos muchísimos ficheros por todo el disco donde la información estaba distribuida por todas partes, mal organizada, insegura, atomizada y creaba redundancia (mismos datos almacenados en varios lugares).

- Se gastaba muchísimo dinero y costaba mucho mantener programas y administrar grandes cantidad de información.
- Básicamente, en esas épocas no quieras imaginarte el dolor de cabeza de los pobres programadores en cobol..

Modelos de las bases de datos

Modelo relacional

Pero llegó el señor, **Edgar Codd** creador del modelo relacional en 1970 y dijo, ya basta de punteros, ficheros y todas esas mierdas vamos a hacerlo a mi modo!. **Llegó, se paró ante la mirada atenta de todos y dijo, a partir de ahora:**

- Utilizarán algo llamado “base de datos”.
- Una [base de datos](#) se compone de varias tablas, denominadas relaciones.
- No pueden existir dos tablas con el mismo nombre ni registro.
- Cada tabla es a su vez un conjunto de [campos](#) (columnas) y [registros](#) (filas).
- La relación entre una tabla padre y un hijo se lleva a cabo por medio de las llaves primarias y llaves foráneas (o ajenas).
- Las llaves primarias son la clave principal de un registro dentro de una tabla y estas deben cumplir con la integridad de los datos.
- Las llaves ajenas se colocan en la tabla hija, contienen el mismo valor que la llave primaria del registro padre; por medio de estas se hacen las formas relacionales.
- Y hoy se van a casa temprano!..
¿Qué es lo que lo hace tan “grandioso” a este tipo?
Pues este tipo es quién **establece las bases del modelo relacional** que utilizamos en la actualidad, donde:

- Se reemplaza el uso de los benditos punteros por “valores” que sirven como **claves**.
- Ofrece **representar la información como un conjunto de tablas**.
- El usuario ve y trabaja solo con **tablas**.
- Se cuenta con un **gestor del sistema de almacenamiento** de datos que recibe órdenes a través de un **lenguaje comprensible por el humano y no procedimental (SQL)**. Al que le dices lo que quieres básicamente y este lo encuentra y te lo da sin necesidad de tu escarbar manualmente en las tablas.
- Separa la representación física de la lógica, es más oculta la representación física de los datos.
- Todas estas tablas, datos e información se sirven desde un lugar centralizado y no distribuido como antes.



Beneficios de las bases de datos relacionales:

- Se utilizan para tener acceso más rápido a los datos.
- A través de una base de datos unificada, la aplicación navegará para responder a la solicitud del usuario.
- Evita la duplicidad en los registros.
- Garantiza la integridad referencial, así, al eliminar un registro elimina todos los registros relacionados dependientes.
- Son más amigables en su utilización, favoreciendo la realización de informes y la optimización de los tiempos y procesos.

Así organiza los datos en tablas que poseen campos organizados en columnas y filas (como un excel) centralizadas en una base de datos.. Algo así:

+ Opciones

  		id_venta	id_cliente	id_usuario	fecha_emision	1	igv	subtotal	total
<input type="checkbox"/>  Editar  Copiar  Borrar		1	14	2	2016-06-28		27.90	155.00	182.90
<input type="checkbox"/>  Editar  Copiar  Borrar		2	5	2	2016-06-28		40.32	224.00	264.32
<input type="checkbox"/>  Editar  Copiar  Borrar		3	5	2	2016-06-28		459.36	2552.00	301.36
<input type="checkbox"/>  Editar  Copiar  Borrar		4	1	2	2016-06-29		23.94	133.00	156.94
<input type="checkbox"/>  Editar  Copiar  Borrar		5	3	2	2016-07-01		14.04	78.00	92.04
<input type="checkbox"/>  Editar  Copiar  Borrar		8	2	1	2016-07-03		27.72	154.00	181.72
<input type="checkbox"/>  Editar  Copiar  Borrar		12	3	2	2016-07-20		645.00	354.00	453.00
<input type="checkbox"/>  Editar  Copiar  Borrar		9	8	1	2016-08-17		12.00	14.00	185.00
<input type="checkbox"/>  Editar  Copiar  Borrar		14	7	2	2016-08-23		68.00	68.00	566.00
<input type="checkbox"/>  Editar  Copiar  Borrar		13	4	2	2016-09-21		45.00	85.00	855.00
<input type="checkbox"/>  Editar  Copiar  Borrar		10	5	3	2016-09-22		14.00	545.00	344.00
<input type="checkbox"/>  Editar  Copiar  Borrar		11	4	3	2016-11-17		45.00	45.00	455.00



☐ Seleccionar todo

Para los elementos que están marcados:



Editar



Copiar



Borrar



Exportar

Esto es una tabla de una base de datos en Mysql vista desde PhpMyAdmin.

El nombre de modelo relacional proviene de la relación (matemática) entre los datos de estos conjuntos (tablas).

Bien puede parecer difícil de comprender al principio pero verás que a medida que sigas leyendo estas entradas se te pondrá cada vez más fácil. Veremos este modelo más adelante en otra entrada, es mi objetivo como introducción explicar ambos modelos, compararlos y lograr discernir para qué casos nos va a servir cada uno de ellos.

Luego ya profundizaremos y aprenderemos ambos como unos campeones...

Modelo no relacional

Las bases de datos no relacionales, también conocidas hoy en día como **bases de datos NoSQL** (porque no utilizan SQL para consultas) utilizan variados modelos de datos para acceder y administrarlos.

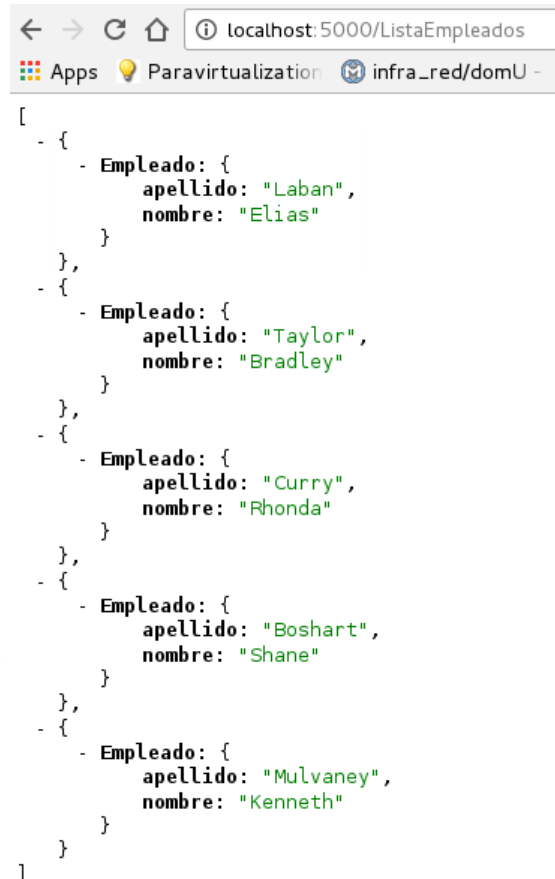
Normalmente este tipo de modelo está adaptado y optimizado para aquellas bases de datos donde hay que administrar un inmenso volumen de datos y donde a su vez se requiere un rápido acceso a ellos, además de un modelo flexible.

Utilizadas en numerosas aplicaciones como aplicaciones móviles o video-juegos las **bases de datos no relacionales logran esta optimización flexibilizando algunas de las restricciones de coherencia de datos en otras bases de datos.**

En la imagen a la derecha puedes contemplar el diseño de una base de datos no relacional en JSON, lo que en realidad es un documento donde se almacena la información. Pero es considerado un almacenamiento de datos no relacional. Veamos algunos beneficios de este modelo y dejame explicarte algo respecto de él.

Y tú te preguntarás, si he comprendido el modelo relacional y es el más utilizado actualmente, para que diablos voy a aprender sobre el **modelo no relacional**. ¿Acaso romper esas relaciones de las que hablabas no sería complicado como antes con toda esa mierda de punteros y ficheros regados por mi pc?.

Pues actualmente el modelo no relacional está ganando terreno a pasos agigantados en el diseño de software por su velocidad y capacidad de almacenamiento. Y saber escoger, utilizar, diseñar tú gestión de datos para tú software con ambos modelos te va a convertir en la jodida navaja suiza de los desarrolladores. Nada de "ficheritos" eso quedó en el pasado, no me malinterpretes.. Que el modelo no sea relacional no implica que siempre se trabaje con ficheros, existen diferentes formas de representación de una bases de datos no relacional.



The screenshot shows a web browser window with the address bar displaying 'localhost:5000/ListaEmpleados'. The browser tabs include 'Apps', 'Paravirtualization', and 'infra_red/domU'. The main content area displays a JSON array of employee records. The JSON is formatted with syntax highlighting, showing an array of objects, each representing an employee with 'apellido' and 'nombre' fields.

```
[
  - {
    - Empleado: {
      apellido: "Laban",
      nombre: "Elias"
    }
  },
  - {
    - Empleado: {
      apellido: "Taylor",
      nombre: "Bradley"
    }
  },
  - {
    - Empleado: {
      apellido: "Curry",
      nombre: "Rhonda"
    }
  },
  - {
    - Empleado: {
      apellido: "Boshart",
      nombre: "Shane"
    }
  },
  - {
    - Empleado: {
      apellido: "Mulvaney",
      nombre: "Kenneth"
    }
  }
]
```

Tipos de bases de datos (modelos no relacionales).

- **Clave-valor:** Esta es un modelo de base de datos no relacional muy similar a los diccionarios.¿Recuerdas?. Donde utiliza este método simple de clave-valor para almacenar datos como un conjunto de pares donde la clave representa un valor único. Son actualmente usadas por SnapChat, Netflix, Nike, etc.Puedes leer más sobre ellas aquí: [Bases de datos clave-valor amazon](#).
- **Documentos:** Como te mostré anteriormente el caso de JSON (JavaScript Object Notation), también dentro de este tipo entra XML o MongoDB entre otras. Y consisten en documentos como ficheros donde podemos almacenar de forma eficiente e intuitiva facilitándonos el almacenamiento y consulta de datos. La naturaleza flexible, semi estructurada y jerárquica de estos documentos y las bases de datos permite que evolucionen según las necesidades de las aplicaciones.
- **Gráficos:** Bueno aquí ya nos vamos un poquito al diablo, pero sí. También son bases de datos que permiten expresar las relaciones y diferencias entre diferentes datos y tipos de datos. Algo así como las estadísticas!
- **Entre otras:** Existen algunas más como el almacenamiento en memorias del tipo caché, entre otras que no viene al caso explicar tanto solo para causarte pesadillas..

Beneficios de las bases de datos no relacionales

Bueno, los beneficios son variados pero en el mejor de los casos y en el que más me gusta hacer hincapié es en la velocidad de consulta y la facilidad de añadir nuevos datos para el programador. Por otro lado podemos ver aplicaciones que combinan ambos modelos para diferentes objetivos.

- Las bases de datos NoSQL están diseñadas para varios patrones de acceso a datos. Las bases de datos de búsqueda NoSQL están diseñadas para hacer análisis sobre datos semiestructurados.
- Proporcionan una variedad de modelos de datos para el desarrollador, a diferencia del modelo relacional que es casi único.
- Permite un escalado horizontal y admite un amplio volumen de datos. Podemos almacenar más datos, más rápido lo que nos beneficia en una mejora de latencia y un amplio rendimiento.

Comparamos modelos relacional vs no relacional.

Ejemplo:

Más en "criollo" vamos a suponer que estamos creando un programa que va a administrar una biblioteca de imágenes de las cuales tienen diferentes autores.

En una base de datos de modelo relacional estos datos serán almacenados en tablas separadas. Una de ellas contendrá las **imágenes** y la otra los **autores o diseñadores de las mismas** y la relación entre ellas se establecerá mediante claves primarias y externas (en otra tabla "idimgautores").

Imágenes			
id_imagen	Imagen	Fecha	Peso
1	Lemonelise.png	12/05/04	1,5
2	Platillapapel.jpg	12/05/04	0,75

Autores		idimgautores	
id_autor	Nombre_autor	id_imagen	id_autor
221	Marcelo	1	221
222	Carlos	2	221

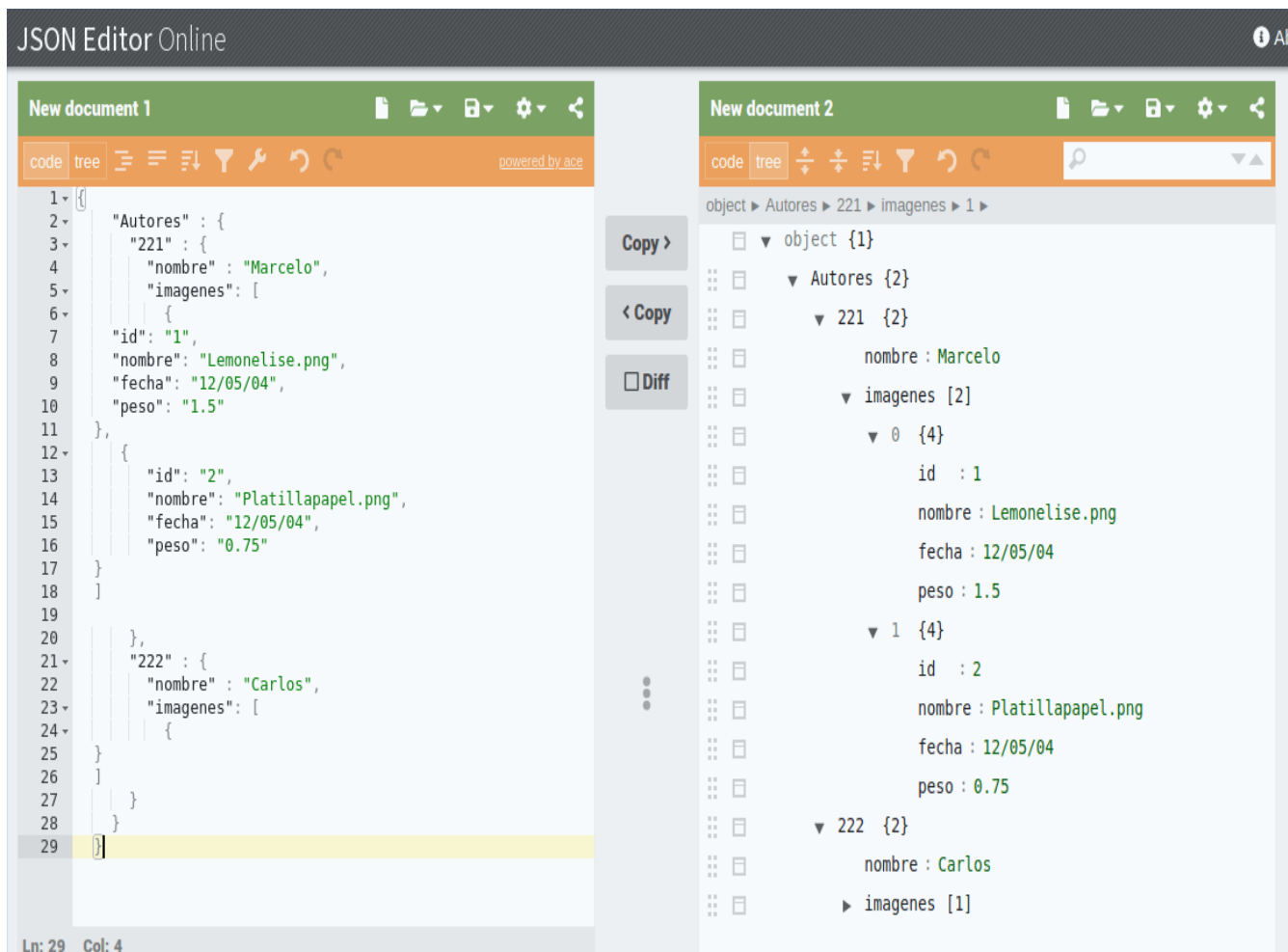
Por ejemplo la tabla "**imágenes**" contendrá las columnas "id_imagen", "imagen" en sí, la "fecha de creación" y el "peso".

Mientras la tabla "**autores**" contendrá el "id_autor", "nombre del autor".

Y finalmente una tercera tabla llamada "**idimgautores**" contendrá las columnas "id_imagen", "id_autor". **Donde se almacenará la id de la imagen y la id de autor creando la relación.** Por lo que efectivamente en el ejemplo podemos afirmar que ambas imágenes son del autor "Marcelo" con id "221", que las subió el mismo día, etc.

Bueno es un ejemplo burdo pero práctico para que entiendas mejor, por supuesto es una ganga en excel con un dibujito cutre..

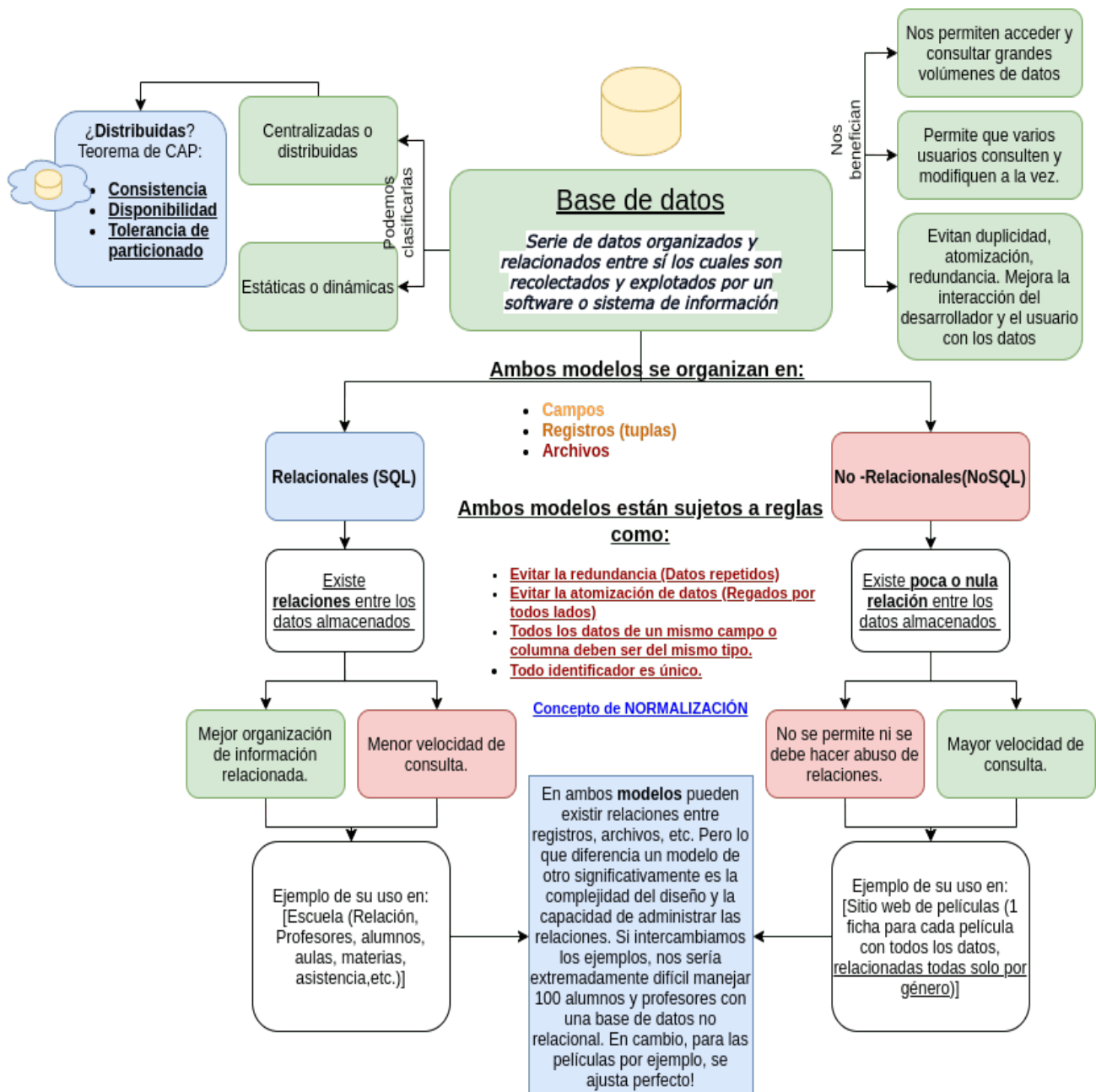
*Pero, **en el caso de una base de datos no relacional** esto mismo se haría de otra manera (algo así) **eliminando de algún modo las relaciones entre los datos y almacenando todo "en un solo documento"**, que puede ser un documento JSON por ejemplo:*



En resumen:

Modelo relacional vs modelo no relacional

Aquí como es tradición te he hecho un diagrama para que logres comprender todo el artículo de una manera visualmente atractiva.



Modelo relacional:

Este tipo de modelo consiste en la organización de la información en trozos pequeños que se “relacionan” entre ellos mediante la relación de sus identificadores.

Modelo no relacional:

Este modelo es lo totalmente opuesto al anterior y se basa en organizar la información sin un identificador que sirva para representar la relación entre un conjunto de datos y otro. Se organiza normalmente en documentos y en algunos casos es muy útil, sobre todo si no tenemos un esquema de lo que se va a almacenar.

¿Cuándo usar un modelo u otro?.

Existen muchos modelos como los jerárquicos, de red, pero la mayoría hoy están obsoletos y sólo salen a flote el **relacional y el no relacional**. Se libra actualmente una batalla entre **SQL vs noSQL**.

Decidir cual usar no es una tarea fácil en el caso de desarrollar una aplicación puesto que ambas tienen sus pros y sus contras, pero básicamente se tiene que tener muy en cuenta los requerimientos de nuestra aplicación y otros factores como el presupuesto dedicado al almacenamiento, entre otros.

Si hablamos de una aplicación que trabajará con muchos datos, y en estos no importa demasiado la relación entre ellos. Pero si nos interesa la disponibilidad y velocidad, no lo sé pongámosle una "aplicación de películas online", donde probablemente debamos almacenar unas 10000 películas con sus trailers, resúmenes, imágenes de portada, entre otras. Es muy probable que nos decidamos por diseñar una base de datos no relacional.

En el caso de trabajar con una cantidad considerable de datos de usuarios pero no un inmenso volumen podemos recurrir a bases de datos relacionales, que nos permitirán trabajar con más comodidad respecto a los usuarios, nombres, emails, rangos, etc. Y podremos relacionarlos y filtrar la información mediante el uso de SQL como lenguaje de consulta.

En cada caso es cuestión de realizar un análisis para evaluar los pros y los contras de elegir un modelo u otro. Luego procedemos al diseño de la estructura de los datos (una no menos importante actividad) y luego procederemos a conformar dicha estructura y comenzaremos a cargar datos.

Si quieres leer más sobre SQL vs noSQL, puedes hacerlo en [link](#). Este tema vamos a tratarlo en profundidad más adelante.

Esto ha sido todo en esta primera entrada de introducción a bases de datos. Se que puede leerse algo complicado para esos lectores que son novatillos y que recién comienzan a rondar el mundo de la informática. Pero te aseguro que en esta entrada encontrarás toda la información filtrada, ordenada, condensada. Y que si la lees con detenimiento tendrás una muy buena base de **introducción sobre bases de datos** para comenzar a desarrollarte en este área.