Bases de datos NoSQL

Tabla de contenidos

01 **Introducción a las bases NoSQL**

Qué son, en qué se diferencian con las bases relacionales.

02 **Tipos de bases de bases NoSQL**

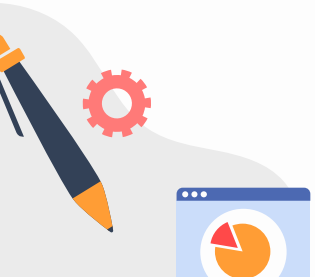
Documentales, clave-valor, grafos y basadas en columnas.

03 **SQL vs NoSQL**

Ventajas y desventajas. Usos.

04 **MongoDB**

Qué es, cómo se almacenan los datos.



01
Introducción a bases de datos NoSQL

Introducción a las bases NoSQL

Las bases NoSQL son un enfoque utilizado en el diseño de bases de datos que permite el almacenamiento y consulta de datos **fuera de las estructuras tradicionales** que se encuentran en las bases de datos relacionales.

Pueden almacenar la **misma información** que se almacena en las bases de datos relacionales, sólo que lo hacen de una **forma diferente**.

La decisión sobre qué enfoque utilizar depende del **contexto** y varía según el **caso de uso**.

Bases no relacionales: cómo se guarda la información

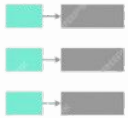
JSON (JavaScript Object Notation): formato liviano que se utiliza para almacenar e intercambiar información. Se utiliza principalmente para representar objetos y arreglos.

BSON (Binary JSON): formato de serialización con codificación binaria que es más compacto que JSON. Admite tipos de datos adicionales que están fuera de JSON estándar, como datos binarios y tipos de fecha.

Como el diseño no requiere un esquema, nos brinda **escalabilidad**

clientes.json

```
[
  {
    "nombre":"Martin Perez",
    "edad":17,
    "materias": [
      {
        "nombre":"Base de datos",
        "duracion": 4
      }
    ]
  },
  {
    ...
  }
]
```



Key-Value

Wide-Column

Graph

Document

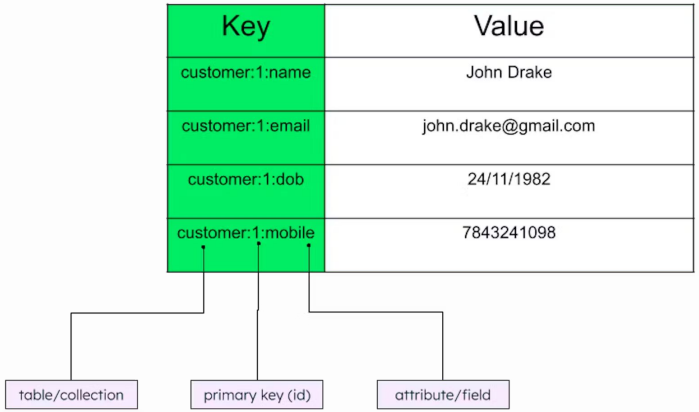
02

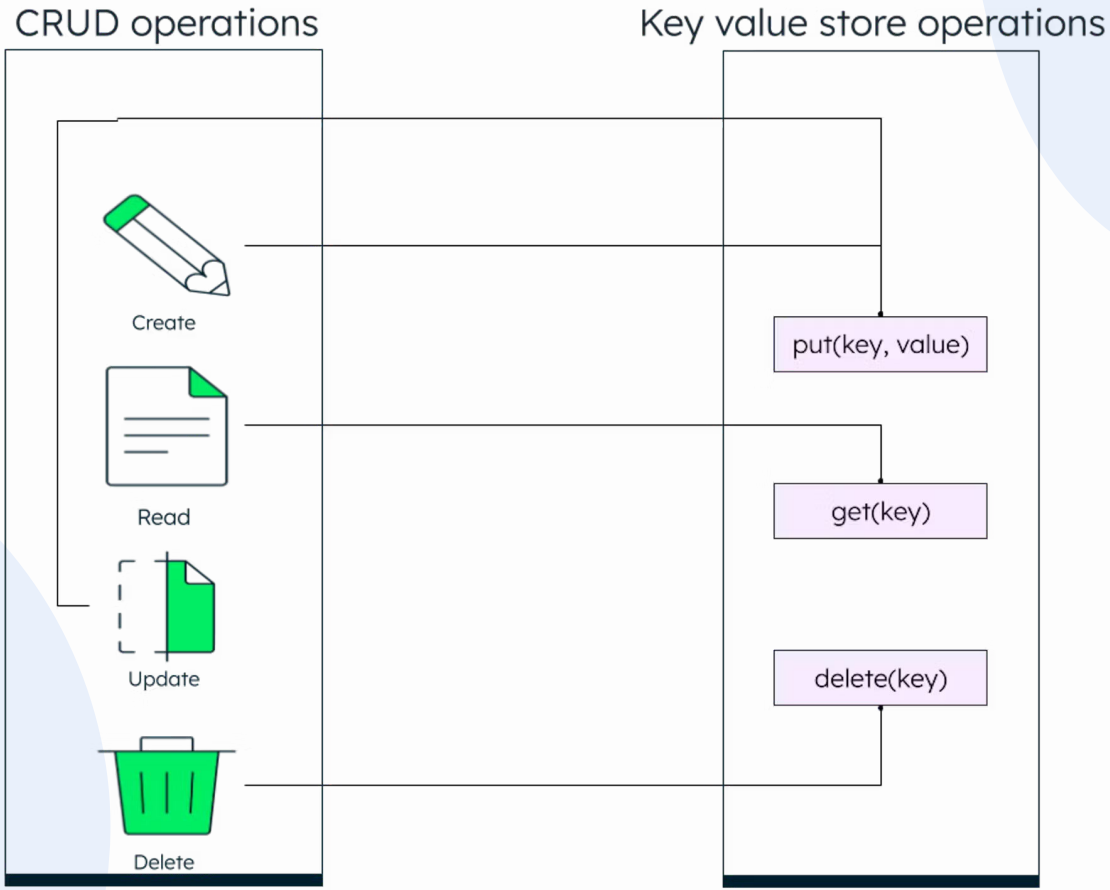
Tipos de bases NoSQL

Tipos: clave-valor

Son las más simples, los elementos se almacenan con un nombre de atributo (clave) junto a su valor. La clave puede ser similar a lo que utilizaríamos en una base SQL, como un ID. El valor puede ser un tipo más complejo como un objeto o un arreglo.

Se utiliza generalmente para almacenar en caché la información de sesión del usuario.




CRUD operations
Key value store operations
Create
Read
Update
Delete
put(key, value)
get(key)
delete(key)

Relational database storage

customer_id	name	email	dob	mobile
1	John Drake	john.drake@gmail.com	24/11/1982	7843241098
2	Mary Chile	mary.chile@outlook.com	05/06/1981	8903424531
3	Mac Adams	mac_1979@gmail.com	23/04/1979	0920421454
4	Jill Smith	jellyjill@gmail.com	14/02/1987	8795092014

Key value database storage

Key	Value

customer_1	{
	"name": "John Drake",
	"email": "john.drake@gmail.com",
	"dob": "24/11/1982",
	"mobile": 7843241098
	}

customer_2	{
	"name": "Mary Chile",
	"email": "mary.chile@outlook.com",
	"dob": "05/06/1981",
	"mobile": 8903424531
	}

customer_3	{
	"name": "Mac Adams",
	"email": "mac_1979@gmail.com",
	"dob": "23/04/1979",
	"mobile": 0920421454
	}

customer_4	{
	"name": "Jill Smith",
	"email": "jellyjill@gmail.com",
	"dob": "14/02/1987",
	"mobile": 8795092014
	}

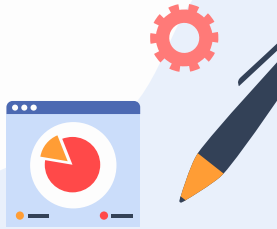
Cuándo utilizar una base clave-valor

- La aplicación está diseñada en consultas simples basadas en claves.
- Acceso aleatorio a datos en tiempo real, por ejemplo, atributos de sesión de usuario en una aplicación en línea, como juegos o finanzas.
- Perfiles de usuario, carrito de compra, sesión de navegación.

Modelo de datos flexible

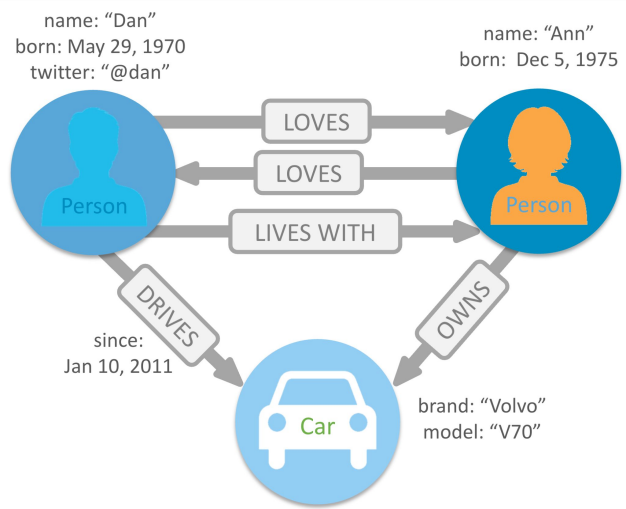
Sin lenguaje de consulta

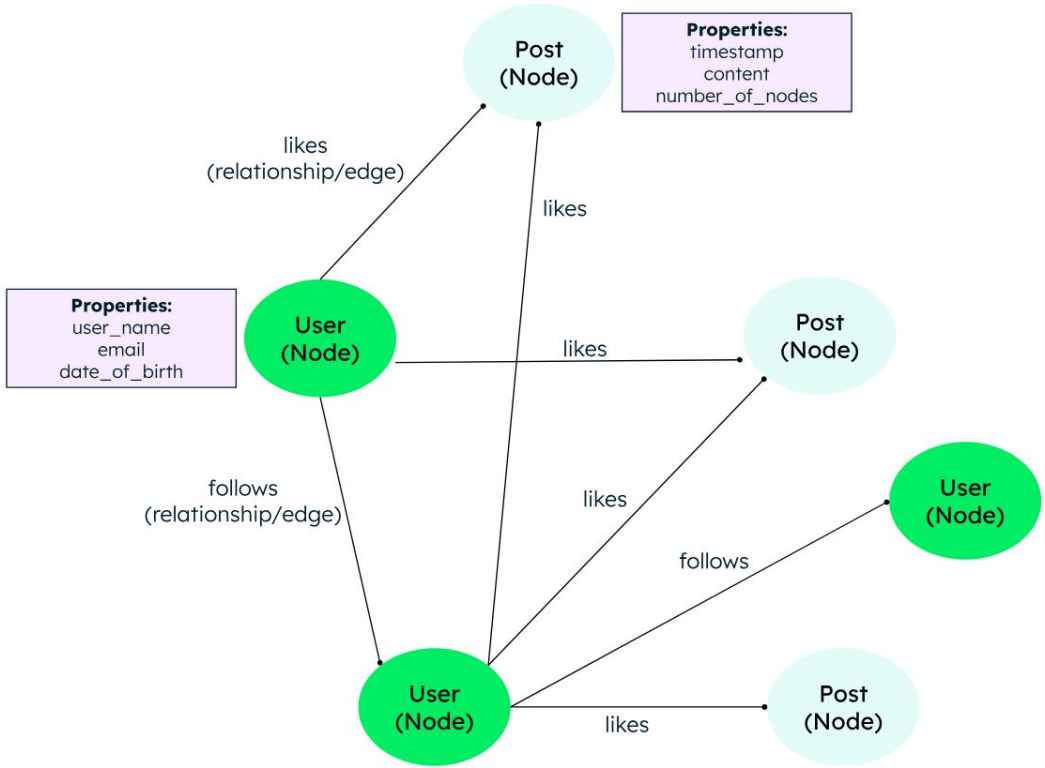
Compatibilidad con tipos de datos complejos



Tipos: grafos

Los elementos de datos se almacenan como **nodos**, **aristas** y **propiedades**. Los nodos suelen almacenar información sobre personas, lugares y cosas (como sustantivos), mientras que las aristas almacenan información sobre las relaciones entre los nodos.

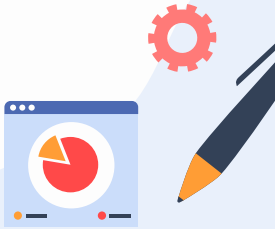



Post
(Node)
Properties:
timestamp
content
number_of_nodes
likes
(relationship/edge)
likes
Properties:
user_name
email
date_of_birth
User
(Node)
likes
Post
(Node)
follows
(relationship/edge)
likes
User
(Node)
follows
User
(Node)
Post
(Node)
likes

Cuándo utilizar una base de grafos

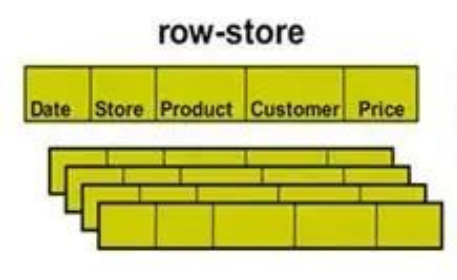
- Datos conectados: redes sociales con diferentes tipos de conexiones entre los usuarios.
- Enrutamiento, entrega o servicios basados en la posición.
- Motores de recomendación.

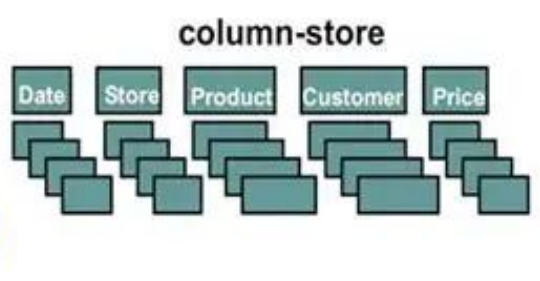
No se recomienda su uso cuando necesitemos modificar todos o un subconjunto de entidades, modificar una propiedad en todos los nodos es una operación compleja.



Tipos: basada en columnas

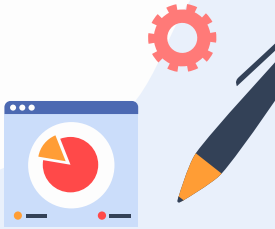
Almacenan los datos en tablas, filas y columnas **dinámicas**.
A diferencia de las bases de datos SQL tradicionales, los almacenes de columnas anchas son **flexibles**, las filas pueden tener diferentes conjuntos de columnas.
Los usuarios pueden acceder solo a las columnas que necesitan sin tener que acceder a la fila completa.





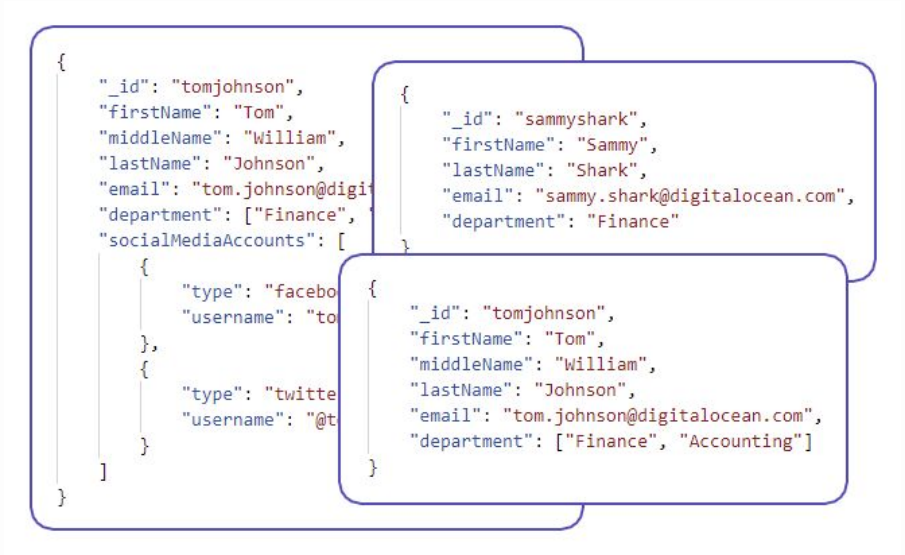
Cuándo utilizar una base basada en columnas

- Sistemas de flujo de eventos: para almacenar estados de las aplicaciones o errores de las mismas.
- Gestores de Contenido, plataformas: podemos almacenar entradas, etiquetas, categorías, enlaces. Los comentarios se pueden almacenar en la misma fila o en otra base de datos.
- Contadores: para poder almacenar las visitas de cada persona a cada apartado de un sitio web.
- Aplicaciones que sólo necesitan consultar los datos por un único valor.



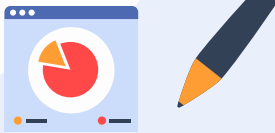
Tipos: documental

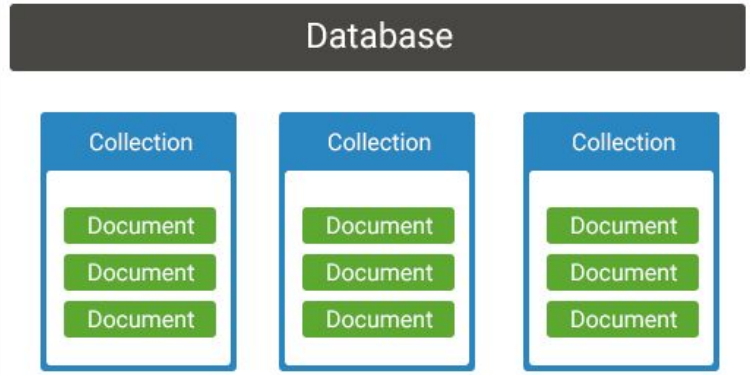
- Almacena los datos en documentos en formato JSON, BSON o XML.
- Brinda mayor **flexibilidad** ya que no es necesario que los esquemas de datos coincidan entre documentos.
- Es adecuado para datos semiestructurados y no estructurados.



Cuándo utilizar una base documental

- Propósito general, ofrece flexibilidad lo que permite consultar cualquier campo y modelar de manera natural o similar a la POO.
- Sistemas de gestión de contenidos y los perfiles de usuario. Al almacenar los documentos mediante JSON, facilita la estructura de datos para guardar los comentarios, registros de usuarios, etc...
- Analíticas Web, datos en Tiempo Real: permite modificar partes de un documento, e insertar nuevos atributos a un documento cuando se necesita una nueva métrica.

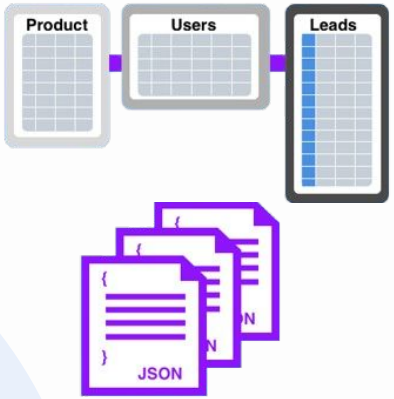





```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
              phone: "123-456-7890",
              email: "xyz@example.com"
            },
  access: {
              level: 5,
              group: "dev"
            }
}
```

Embedded sub-document

Embedded sub-document



03

SQL vs NoSQL

SQL

- Base de datos estructurada y segmentada.
- Tipo y validez de datos muy importante.
- Escritura y modificaciones recurrentes sobre elementos específicos.
- Búsquedas complejas.

NoSQL

- Bases de datos sin esquemas específicos.
- Múltiples búsquedas de lectura que se puedan recuperar de una vez sin combinaciones.
- Grandes conjuntos de datos.
- Datos distribuídos.

NoSQL: Ventajas

- **Flexibilidad**: ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo.
- **Escalabilidad**: están diseñadas para escalar usando clústeres distribuidos de hardware en lugar de escalar añadiendo servidores caros y sólidos.
- **Alto rendimiento**: están optimizadas para modelos de datos específicos (como documentos, clave-valor y grafos) y patrones de acceso que permiten un mayor rendimiento que el intento de lograr una funcionalidad similar con bases de datos relacionales.
- **Altamente funcional**: proporcionan APIs altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

NoSQL: Desventajas

- **Limitaciones en consultas complejas.**
- **Complejidad en el modelado de datos:** hay que elegir cuidadosamente el modelo y optimizar el rendimiento, lo que puede ser más complejo que en las bases de datos relacionales.
- **Menor madurez y soporte**: Las bases de datos NoSQL son relativamente nuevas en comparación con las bases de datos SQL, lo que puede resultar en menos herramientas, documentación y soporte disponibles.
- **Inconsistencia de datos.**

04

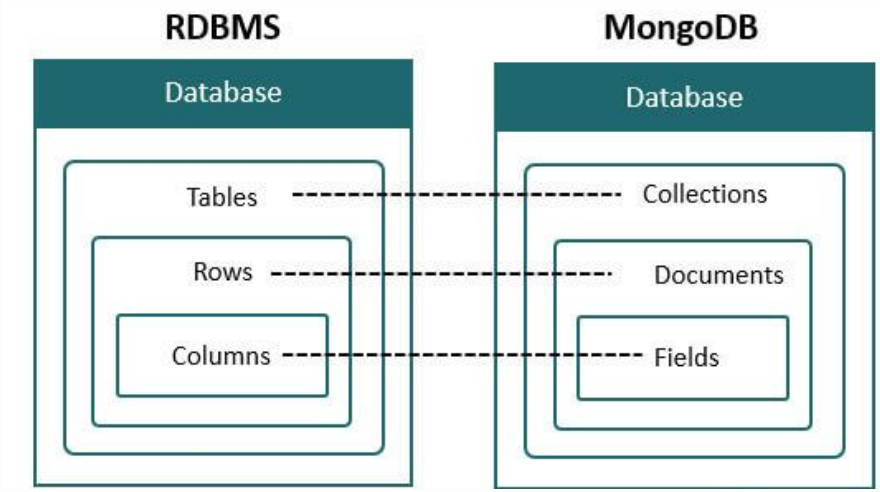
MongoDB

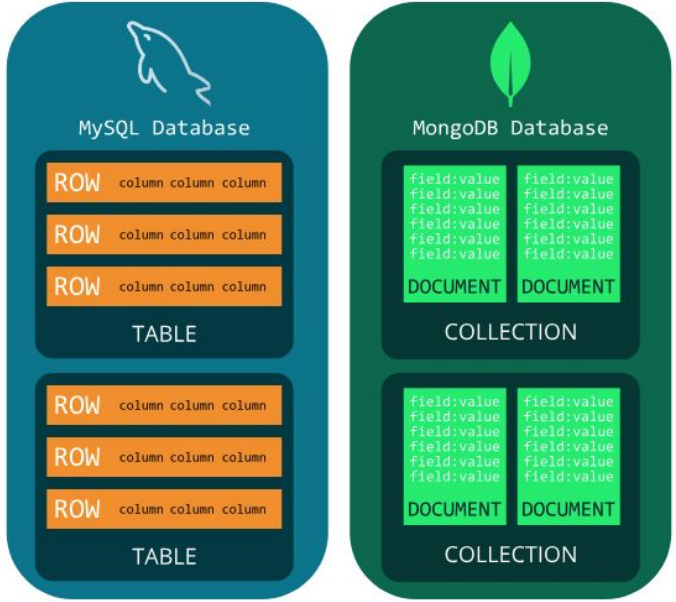
MongoDB: características

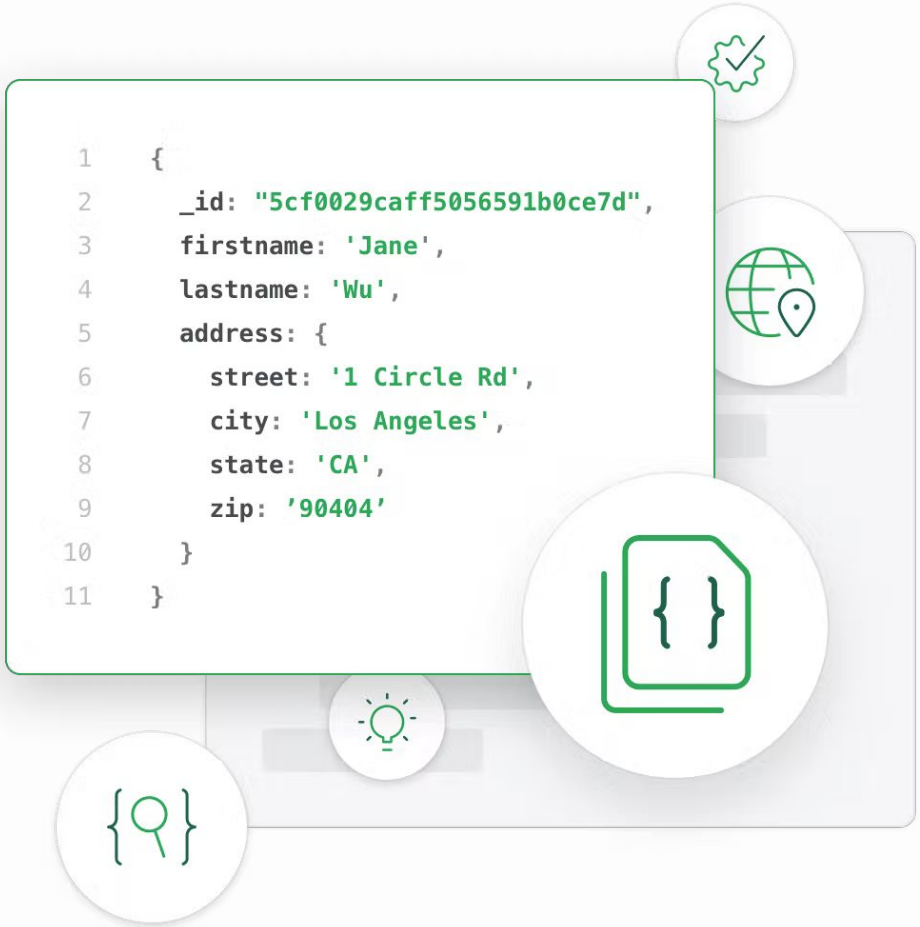
Es una de las bases de datos NoSQL más populares. Sigue un modelo de datos documental, donde los documentos se guardan en formato BSON.

- Soporta **esquemas dinámicos**, los documentos de una misma colección pueden tener atributos diferentes.
- **No soporta joins pero sí índices**.
- Dentro de una instancia de MongoDB podemos tener **0 o más bases de datos**. Cada base de datos tendrá **0 o más colecciones**. Cada colección tiene **0 o más documentos**. Cada documento tiene 0 o más atributos, compuestos por **parejas de clave-valor**.
- Al realizar cualquier consulta, se **devuelve un cursor**, con el cual podemos contar, ordenar, limitar o saltar documentos.

Bases relacionales vs MongoDB







MongoDB	SQL
Base de Datos	Base de Datos
Colecciones	Tablas
Documentos	Filas (registros)
Campos	Columnas

Base de Datos	Tienda	
Colecciones	Usuarios	Pedidos
Documentos	{Nombre: 'Juan', edad: 25}	{...}
	{Nombre: 'Felipe'}	{...}

MongoDB

Tener en cuenta que:

- Es sensible a las mayúsculas.
- No asegura que el orden de los campos se respete.
- Es sensible a los tipos de los datos.
- Los campos no pueden empezar por “$”, “.” y el nombre _id queda reservado para la clave primaria.

Con http://jsonlint.com/ podemos validar que el json sea correcto.

MongoDB: cómo modelar datos relacionados

1:1

Hay que embeber un documento dentro del otro, como parte de un atributo.
Algunos aspectos que nos pueden llevar a no embeberlos son:

- La frecuencia de acceso. Si a uno de ellos se accede raramente, puede que convenga tenerlos separados para liberar memoria.
- El tamaño de los elementos. Si hay uno que es mucho más grande que el otro, o a uno lo modificamos muchas más veces que el otro, será mejor tenerlos en documentos separados.

MongoDB: cómo modelar datos relacionados

1:N o N:1

Si N son “muchos” es mejor usar una referencia a un documento (como una clave foránea en SQL).
Si N son “pocos” la mejor solución es crear un array dentro de una entidad embebiendo documentos de la otra entidad.

MongoDB: cómo modelar datos relacionados

N:M

Para este tipo de relaciones hay que establecer el tamaño de N y M. Si N como máximo vale 3 y M 500000, entonces deberíamos seguir un enfoque de embeber la N dentro de la M (One Way Embedding).

En cambio, si N vale 3 y M vale 5, entonces podemos hacer que ambos embeban al otro documento (Two Way Embedding)

use nombreBase

db. coleccion.find ({dni: 47872344, nombre : “Juan”}, {apellido: 1})

condiciones

db. coleccion.updateMany ({nombre: “Juan”} , {$set: {dni: 47872344}})

condición acción

db. coleccion.deleteMany ({edad: 17})

db. coleccion.insertOne ({nombre: “Juan”, edad: 17, dni: 47872344})

db. coleccion.insertMany ([{dni: 47872344}, {dni: 44872344}, ...])

Operadores básicos:

$gt: Mayor que
$lt: Menor que
$eq: Igual
$not
$or
$and
$elemMatch
$exists

.sort(campo: ± 1) Ordena asc o desc

.count() Cuenta la cantidad de documentos

.limit()

MongoDB shell

```
C:\Users\bisha>mongosh
Current Mongosh Log ID: 65d512ce4932abd79f29859c
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true
&serverSelectionTimeoutMS=2000&appName=mongosh+1.8.0
Using MongoDB:          7.0.5
Using Mongosh:          1.8.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-02-21T01:57:06.899+05:30: Access control is not enabled for the
database. Read and write access to data and configuration is unrestricte
d
------

test>
```

MongoDB Compass