# Tipos de Bases NoSQL

Las bases de datos NoSQL ofrecen una alternativa al modelo relacional tradicional, diseñadas para manejar diferentes tipos de datos y casos de uso. Aquí te presento una descripción de los principales tipos de bases de datos NoSQL: documentales, de grafos, clave-valor y orientadas a columnas.

## Bases de Datos Documentales:

Las bases de datos documentales almacenan datos en forma de documentos, que suelen ser representados en formatos como JSON o BSON. Cada documento puede tener una estructura flexible y puede contener diferentes tipos de datos anidados. Este modelo es ideal para aplicaciones web, sistemas de gestión de contenido y aplicaciones que manejan datos semi-estructurados o no estructurados.

Ejemplo: **MongoDB**, Couchbase, CouchDB.

## Bases de Datos de Grafos:

Las bases de datos de grafos almacenan datos en forma de nodos (entidades) y relaciones (conexiones) entre ellos. Este modelo es ideal para representar datos interconectados, **como redes sociales**, sistemas de recomendación, análisis de redes, y cualquier otro escenario donde las relaciones entre entidades sean críticas para el análisis y la consulta.

Ejemplo: **Neo4j**, Amazon Neptune, JanusGraph.

## Bases de Datos Clave-Valor:

Las bases de datos clave-valor almacenan datos en forma de pares clave-valor, donde cada valor está asociado con una clave única. Son extremadamente simples y rápidas, lo que las hace ideales para casos de uso que requieren una alta velocidad de lectura y escritura con una estructura de datos básica.

Ejemplo: **Redis**, Amazon DynamoDB, Riak.

## Bases de Datos Orientadas a Columnas:

Las bases de datos orientadas a columnas almacenan datos en forma de columnas en lugar de filas. Esto permite una recuperación eficiente de datos para consultas que implican un subconjunto de columnas. Son ideales para aplicaciones que requieren análisis de datos en tiempo real, almacenamiento de series temporales y sistemas de gestión de bases de datos distribuidas.

Ejemplo: **Apache** **Cassandra**, Google Bigtable, ScyllaDB.

# ESCALAMIENTO

Las bases de datos NoSQL tienen la característica de tener gran facilidad de escalamiento.

El escalamiento en bases de datos NoSQL es un aspecto fundamental y diferenciador en comparación con las bases de datos relacionales tradicionales. El escalamiento en NoSQL se puede realizar de dos formas principales: escalamiento vertical y escalamiento horizontal.

## Escalamiento Vertical:

El escalamiento vertical, también conocido como escalamiento hacia arriba, implica aumentar la capacidad de una sola instancia de hardware, como agregar más CPU, RAM o almacenamiento a un servidor. En el contexto de las bases de datos NoSQL, esto puede significar aumentar la potencia de procesamiento o la capacidad de almacenamiento de un único servidor o nodo de base de datos.

Sin embargo, el escalamiento vertical tiene limitaciones en términos de cuánto puede crecer una instancia de hardware y puede resultar costoso y difícil de mantener a medida que la demanda de datos y la carga de trabajo aumentan.

## Escalamiento Horizontal:

El escalamiento horizontal, también conocido como escalamiento hacia fuera, implica distribuir los datos y la carga de trabajo a través de múltiples nodos o servidores. En lugar de aumentar la potencia de un solo servidor, se agregan más servidores a la infraestructura para distribuir la carga.

En las bases de datos NoSQL, el escalamiento horizontal se logra mediante técnicas como la fragmentación (sharding) y la replicación.

La fragmentación implica dividir los datos en fragmentos más pequeños y distribuirlos entre varios servidores, lo que permite distribuir la carga de manera más uniforme y facilita el manejo de grandes volúmenes de datos.

La replicación implica crear copias redundantes de los datos en varios nodos para garantizar la disponibilidad y la tolerancia a fallos.

El escalamiento horizontal proporciona una mayor flexibilidad y capacidad de crecimiento en comparación con el escalamiento vertical. Permite a las bases de datos NoSQL crecer de manera más eficiente y manejar cargas de trabajo crecientes sin comprometer el rendimiento o la disponibilidad.

## Consideraciones de Escalamiento en NoSQL:

Diseño de Datos: Es importante diseñar los esquemas de datos y las estrategias de fragmentación de manera adecuada para garantizar una distribución equilibrada de la carga y un rendimiento óptimo.

Disponibilidad y Tolerancia a Fallos: El escalamiento horizontal puede mejorar la disponibilidad y la tolerancia a fallos al distribuir los datos en múltiples nodos, lo que reduce el impacto de los fallos individuales.

Consistencia y Coherencia: Al escalar horizontalmente, es fundamental considerar cómo mantener la consistencia y la coherencia de los datos entre los nodos y las réplicas para garantizar la integridad de los datos.

Monitoreo y Gestión: El escalamiento horizontal puede complicar la gestión y el monitoreo de la infraestructura, por lo que es importante contar con herramientas y procesos adecuados para administrar y supervisar los clústeres de bases de datos distribuidos.

# Replicación

La replicación en MongoDB es un mecanismo fundamental que garantiza la disponibilidad de los datos y proporciona tolerancia a fallos en entornos distribuidos. MongoDB utiliza un enfoque de replicación llamado "replica sets" (conjuntos de réplicas), que es un conjunto de nodos MongoDB que mantienen los mismos datos.

## Características Clave de la Replicación en MongoDB:

### Tolerancia a Fallos:

La replicación en MongoDB permite que los datos estén disponibles incluso en caso de fallo de un nodo o un servidor. Si un nodo primario falla, otro nodo secundario puede tomar su lugar automáticamente, lo que garantiza la continuidad del servicio.

### Alta Disponibilidad:

Los replica sets de MongoDB están diseñados para garantizar la alta disponibilidad de los datos. Con múltiples nodos que contienen copias idénticas de los datos, los clientes pueden dirigirse a cualquier nodo disponible para realizar operaciones de lectura y escritura.

### Auto-recuperación:

Cuando un nodo primario falla, el conjunto de réplicas de MongoDB realiza automáticamente una elección de un nuevo nodo primario entre los nodos secundarios disponibles. Este proceso garantiza que el sistema pueda recuperarse sin intervención manual.

### Lecturas Escalables:

Los nodos secundarios en un conjunto de réplicas pueden utilizarse para operaciones de lectura, lo que distribuye la carga de lectura y mejora el rendimiento del sistema en general.

### Seguridad y Durabilidad:

MongoDB replica los datos de manera asíncrona y de forma segura a través de la red, lo que garantiza la integridad y la durabilidad de los datos replicados.

### **Topologías Flexibles:**

Los replica sets de MongoDB admiten diferentes topologías, incluidas configuraciones con múltiples nodos secundarios y nodos de arbitro, lo que permite adaptarse a diversos requisitos de disponibilidad y rendimiento.

## Proceso de Replicación en MongoDB:

### **Nodo Primario (Primary):**

El nodo primario es responsable de aceptar todas las operaciones de escritura y coordinar las operaciones de lectura. Es el único nodo al que los clientes envían operaciones de escritura.

### **Nodos Secundarios (Secondaries):**

Los nodos secundarios mantienen copias sincronizadas de los datos del nodo primario. Pueden aceptar operaciones de lectura y proporcionar redundancia de datos para mejorar la disponibilidad y la tolerancia a fallos.

### **Elección Automática de Nodo Primario:**

Si el nodo primario falla, los nodos secundarios participan en una elección automática para seleccionar un nuevo nodo primario. Este proceso se realiza de forma automática y transparente para los clientes.

La replicación en MongoDB es una característica clave que garantiza la disponibilidad, la tolerancia a fallos y la escalabilidad del sistema en entornos distribuidos. Al utilizar conjuntos de réplicas, MongoDB proporciona una infraestructura robusta y confiable para aplicaciones críticas que requieren alta disponibilidad y rendimiento constante.

# Documentos en MongoDB:

Un documento en MongoDB es un conjunto de datos almacenados en un formato similar a JSON (BSON, Binary JSON). Cada documento consiste en pares de clave-valor que representan los campos y sus valores correspondientes. Los documentos pueden contener datos estructurados y no estructurados, lo que significa que no están sujetos a un esquema rígido y pueden tener una estructura flexible.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

# Colecciones en MongoDB:

Una colección en MongoDB es un grupo de documentos almacenados en la base de datos. A diferencia de las tablas en las bases de datos relacionales, las colecciones no tienen un esquema fijo y pueden contener documentos con estructuras diferentes. Las colecciones pueden considerarse como análogas a las tablas en las bases de datos relacionales, pero sin la necesidad de definir una estructura de tabla fija.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Por ejemplo, considera una colección llamada usuarios que almacena los documentos de usuarios como el ejemplo mencionado anteriormente. La colección usuarios puede contener documentos con diferentes campos y estructuras, y MongoDB no impone restricciones sobre la uniformidad de la estructura de los documentos dentro de la colección.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En MongoDB, las operaciones se realizan a nivel de colección y documento. Puedes insertar, actualizar, eliminar y consultar documentos dentro de una colección utilizando el lenguaje de consulta de MongoDB.

# Creación base de datos

Mongo Atlas

Mongo Compass

Mongo VsCode