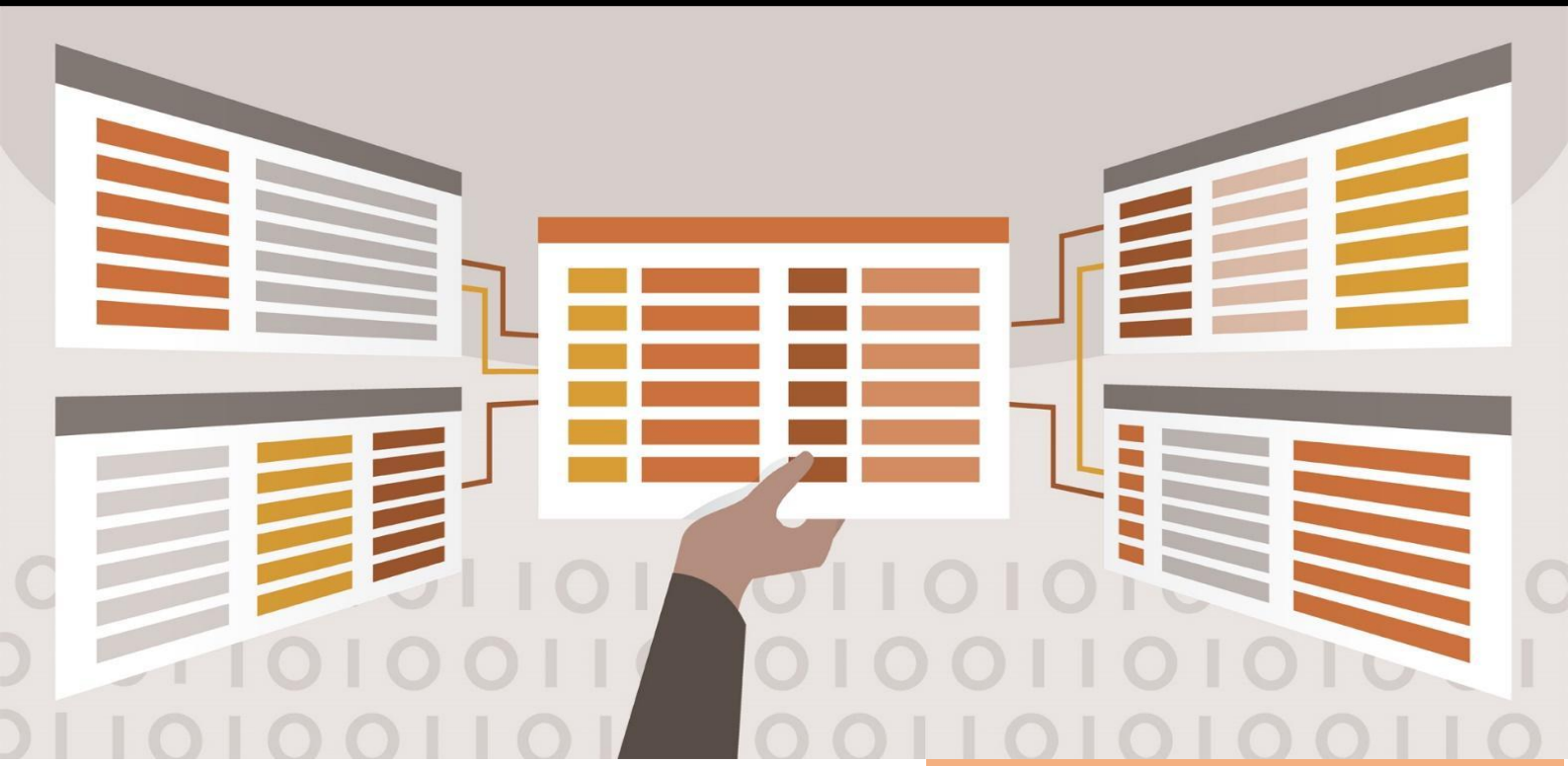
salesianos

2025

Bases de datos no relacionales



Jesus A. Gómez Rodríguez
Pablo Romo Grilo
Colegio Salesiano “San Ignacio”
5-2-2025

Índice

1. Introducción a las bases de datos no relacionadas

A. Definición de bases de datos no relacionadas

Son sistemas de gestión de bases de datos que no siguen el modelo relacional clásico basado en tablas con relaciones fijas entre ellas. En cambio, utilizan diferentes modelos de datos, como documentos, clave-valor, columnares o basados en grafos, para almacenar y acceder a la información. Estas bases de datos son diseñadas para abordar ciertos desafíos o requisitos específicos, como la gestión de grandes volúmenes de datos no estructurados, la escalabilidad horizontal, la flexibilidad de esquema y el rendimiento optimizado para ciertos tipos de consultas.

Las bases de datos no relacionales ofrecen alternativas flexibles y especializadas para abordar necesidades específicas, a menudo en entornos donde las bases de datos relacionales pueden no ser la solución más eficiente.

B. Ventajas y desventajas de las bases de datos no relacionales

➢ **VENTAJAS**

1. Flexibilidad de Esquema

Las bases de datos NoSQL permiten un esquema flexible, lo que significa que pueden manejar datos no estructurados o semiestructurados sin requerir un esquema predefinido.

Esto es útil cuando se trabaja con datos que pueden cambiar con el tiempo, como en el desarrollo ágil de software.

2. Escalabilidad Horizontal

Muchas bases de datos NoSQL están diseñadas para escalar horizontalmente, lo que significa que pueden manejar grandes volúmenes de datos distribuyendo la carga entre múltiples servidores o nodos.

Es útil para aplicaciones web con crecimiento rápido.

3. Alto rendimiento en operaciones específicas

Algunas bases de datos NoSQL están optimizadas para operaciones específicas, como lecturas o escrituras rápidas, lo que las hace eficientes para ciertos casos de uso.

Bases de datos clave-valor para operaciones rápidas de lectura y escritura.

4. Modelos específicos

Las bases de datos NoSQL ofrecen modelos de datos específicos (documentos, clave-valor, columnares, grafos), lo que permite elegir el modelo más adecuado para un caso de uso particular.

Bases de datos de documentos para almacenar datos semiestructurados.

➢ **DESVENTAJAS**

5. Complejidad en consultas

Para algunas bases de datos NoSQL, realizar consultas complejas puede ser más complicado debido a la falta de un lenguaje de consulta estandarizado como SQL en las bases de datos relacionales.

Realizar operaciones de agregación puede ser más desafiante.

6. Menos madurez y ecosistemas

En comparación con las bases de datos relacionales, algunas bases de datos NoSQL y sus ecosistemas pueden ser menos maduros y estandarizados

Menos herramientas de administración, menos estándares de consulta..

7. Menos transacciones ACID:

Algunas bases de datos NoSQL sacrifican las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) en favor de un rendimiento más alto, lo que puede ser una limitación en ciertos casos.

En entornos donde la integridad de los datos es crítica, esto puede ser una desventaja.

8. Menor soporte para relaciones complejas

Si un conjunto de datos tiene relaciones complejas que necesitan ser gestionadas de manera rigurosa, las bases de datos NoSQL pueden no ser la mejor opción.

Situaciones donde las relaciones complejas entre tablas son fundamentales.

C. Comparación

	RELACIONALES	**NO RELACIONALES**
ESTRUCTURA	Utilizan tablas para organizar la información, con filas que representan registros individuales y columnas que representan atributos.	Utilizan diferentes modelos de almacenamiento, como documentos, grafos, clave-valor o columnares, para organizar la información.
ESQUEMA	Tienen un esquema fijo y predefinido que define la estructura de los datos.	Tienen un esquema dinámico que permite almacenar datos de forma flexible, sin una estructura predefinida.
CONSULTAS	Utilizan el lenguaje SQL para realizar consultas y manipulación de datos.	Utilizan diferentes lenguajes o interfaces para realizar consultas, dependiendo del modelo de almacenamiento.

	RELACIONALES	NO RELACIONALES
ESCALABILIDAD	Pueden tener dificultades para escalar horizontalmente debido a su estructura rígida.	Suelen ser más fáciles de escalar horizontalmente debido a su flexibilidad en la estructura de datos.
TRANSACCIONES	Ofrecen soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).	Algunas bases de datos no relacionales ofrecen soporte para transacciones ACID, pero otras priorizan la disponibilidad y la tolerancia a fallos sobre la consistencia.

En resumen, las bases de datos relacionales tienden a ser más estructuradas y adecuadas para aplicaciones con requisitos de integridad de datos y transacciones complejas, mientras que las bases de datos no relacionales son más flexibles y escalables, lo que las hace adecuadas para aplicaciones con grandes volúmenes de datos y requisitos de rendimiento.

2. Tipos de bases de datos No Relacionales

A. Bases de datos orientadas a documentos

Son sistemas de gestión de bases de datos que almacenan, recuperan y gestionan información en formato de documentos, como JSON o XML, en lugar de utilizar un esquema fijo como en las bases de datos relacionales.

Estas bases de datos son altamente flexibles, ya que cada documento puede tener su propio esquema y no se requiere que todos los documentos en una colección tengan la misma estructura. Esto las hace ideales para el almacenamiento de datos no estructurados o semiestructurados, comunes en aplicaciones de inteligencia artificial y big data.

Algunas de las bases de datos no relacionales orientadas a documentos más populares incluyen MongoDB, Couchbase y Amazon DocumentDB. Estas bases de datos ofrecen capacidades de escalabilidad horizontal, permitiendo manejar grandes volúmenes de datos distribuidos en múltiples servidores.

B. Bases de datos de grafos

Son sistemas de gestión de bases de datos diseñados específicamente para almacenar y consultar datos en forma de grafos. En un grafo, los datos se representan como nodos (vértices) y relaciones (bordes) entre esos nodos.

Estas bases de datos son ideales para modelar y consultar relaciones complejas entre entidades, lo que las hace muy adecuadas para aplicaciones de inteligencia artificial, redes sociales, análisis de redes, recomendaciones y sistemas de recomendación, entre otros.

Las bases de datos de grafos permiten consultas eficientes para encontrar patrones, conexiones y rutas en grandes conjuntos de datos interconectados. Algunas de las bases de datos de grafos más conocidas incluyen Neo4j, Amazon Neptune y Microsoft Azure Cosmos DB.

C. Bases de datos clave-valor

Son sistemas de gestión de bases de datos que almacenan datos en una estructura simple de pares clave-valor. Cada dato se almacena asociado a una clave única, lo que permite un acceso rápido y eficiente a los datos.

Estas bases de datos son altamente escalables y eficientes, lo que las hace ideales para aplicaciones que requieren un alto rendimiento en la lectura y escritura de datos, como aplicaciones web, sistemas de almacenamiento en caché, y aplicaciones de Internet de las cosas (IoT).

Algunas de las bases de datos no relacionales de datos clave-valor más conocidas incluyen Redis, Amazon DynamoDB y Apache Cassandra. Estas bases de datos ofrecen capacidades de almacenamiento distribuido y tolerancia a fallos, lo que las hace adecuadas para entornos de alta disponibilidad y rendimiento.

D. Bases de datos de columna amplia

Son sistemas de gestión de bases de datos que almacenan datos de forma orientada a columnas en lugar de filas, como lo hacen las bases de datos relacionales.

Estas bases de datos están optimizadas para consultas analíticas y de agregación, lo que las hace ideales para aplicaciones de inteligencia empresarial, análisis de big data y procesamiento de datos en tiempo real. Almacenan los datos de cada columna de manera continua, lo que permite un acceso rápido y eficiente a grandes volúmenes de datos.

Algunas de las bases de datos no relacionales de datos de columna ancha más conocidas incluyen Apache Cassandra, Apache HBase y Google Bigtable. Estas bases de datos ofrecen capacidades de escalabilidad horizontal y tolerancia a fallos, lo que las hace adecuadas para entornos de alto rendimiento y disponibilidad.

E. Bases de datos de tiempo real

Son sistemas de gestión de bases de datos diseñados para capturar, almacenar y procesar datos en tiempo real. Estas bases de datos están optimizadas para manejar flujos continuos de datos, como los generados por sensores, aplicaciones de IoT, registros de eventos y sistemas de monitorización.

Estas bases de datos son ideales para aplicaciones que requieren análisis en tiempo real, detección de anomalías, generación de informes en tiempo real y toma de decisiones basada en datos en tiempo real.

3. Ejemplos de bases de datos no relacionales

A. MongoDB

Es un sistema de gestión de bases de datos (DBMS) NoSQL que se destaca por su enfoque en bases de datos orientadas a documentos. Aquí hay una explicación general de lo que hace MongoDB y algunas de sus características clave.

➢ CARACTERÍSTICAS PRINCIPALES

1. **Base de Datos NoSQL**

   MongoDB es una base de datos NoSQL, lo que significa que no sigue el modelo relacional tradicional basado en tablas, sino que utiliza un modelo de documentos.

2. **Modelo de Documentos**

   En lugar de usar filas y columnas como en las bases de datos relacionales, MongoDB almacena datos de documentos BSON(Binary JSON), que son estructuras de datos flexibles en formato JSON.

3. **Escalabilidad Horizontal**

   MongoDB es altamente escalable horizontalmente, lo que permite agregar más servidores para manejar grandes volúmenes de datos y tráfico.

4. **Esquema Dinamico**

   No se requiere un esquema fijo, lo que significa que cada documento en una colección puede tener campos diferentes. Esto proporciona flexibilidad en la representación de datos.

5. **Índices**

   MongoDB admite índices para mejorar el rendimiento de las consultas y acelerar la recuperación de datos.

➢ USO COMÚN

1. **Aplicaciones**

   MongoDB es popular en aplicaciones web donde se necesita flexibilidad en el esquema de datos y escalabilidad.

2. **Big Data**

   Se utiliza en entornos de Big Data debido a su capacidad para manejar grandes volúmenes de datos distribuidos.

3. **Desarrollo Rápido**

   Es comúnmente elegido en entornos de desarrollo ágil y prototipado rápido debido a su flexibilidad de esquema.

4. **Análisis en Tiempo Real**

   Puede ser utilizado en sistemas que requieren análisis en tiempo real y procesamiento de datos en tiempo real.

B. Neo4j

Es un sistema de gestión de bases de datos de grafos, diseñado específicamente para almacenar, gestionar y consultar datos basados en relaciones. A diferencia de las bases de datos tradicionales basadas en tablas, Neo4j utiliza un modelo de datos de grafos para representar y almacenar información.

➢ **CARACTERÍSTICAS PRINCIPALES**

1. **Modelo de Datos de Grafos**

Neo4j organiza la información en nodos y relaciones. Los nodos representan entidades, y las relaciones conectan nodos, describiendo cómo están relacionados.

2. **Estructura de Grafo Nativa**

A diferencia de las bases de datos relacionales, que emulan relaciones a través de claves foráneas, Neo4j almacena relaciones directamente como parte de su estructura de datos nativa, lo que facilita y acelera las consultas relacionales.

3. **Consultas de Patrones(Cypher Query Language)**

Neo4j utiliza el lenguaje de consulta Cypher, que permite a los usuarios expresar patrones y relaciones en las consultas. Esto simplifica la expresión de consultas complejas en estructuras de grafo.

4. **Escalabilidad Horizontal y Vertical**

Neo4j es escalable tanto vertical como horizontalmente. Puede manejar grandes volúmenes de datos y crecer en capacidad añadiendo más recursos o distribuyendo la carga entre múltiples servidores.

➢ **USO COMÚN**

1. **Redes Sociales y Recomendaciones**

Neo4j es adecuado para modelar y consultar relaciones sociales, lo que lo convierte en una opción popular para aplicaciones de redes sociales y sistemas de recomendación.

2. **Análisis de Fraude**

Se utiliza para detectar patrones de fraude y anomalías en datos conectados, donde las relaciones son esenciales para entender el comportamiento del fraude.

3. **Gestión de Conocimientos**

Neo4j es útil para modelar y analizar estructuras de conocimiento, como ontologías y taxonomías, para mejorar la gestión y la búsqueda del conocimiento.

4. **Análisis de Redes**

Es empleado en aplicaciones que requieren análisis de redes, como la identificación de influencers en redes sociales o la optimización de rutas en sistemas logísticos.

C. Redis

Es un sistema de almacenamiento en memoria de código abierto que se utiliza como base de datos, caché y sistema de mensajería. Su diseño está optimizado para el rendimiento y la velocidad, ya que almacena datos en la memoria principal (RAM), lo que permite un acceso rápido a la información.

➢ **CARACTERÍSTICAS PRINCIPALES**

1. **Almacenamiento en Memoria (In-Memory)**

Redis almacena todos sus datos en la memoria principal del servidor. Esto permite un acceso extremadamente rápido a los datos, ya que no hay necesidad de acceder a discos para la mayoría de las operaciones.

2. **Estructura de Datos**

Redis admite una variedad de estructuras de datos, incluyendo cadenas de texto, listas, conjuntos, conjuntos ordenados, mapas hash y bits. Cada tipo de dato tiene sus propias operaciones específicas.

3. **Cache de Alto Rendimiento**

Se utiliza comúnmente como una capa de caché en aplicaciones para almacenar datos temporalmente en la memoria y acelerar el acceso a información que se recupera o se calcula con frecuencia.

4. **Persistencia Opcional**

Aunque Redis es principalmente un sistema en memoria, ofrece opciones de persistencia para almacenar datos en disco, lo que permite recuperar los datos en caso de reinicio o fallo del servidor.

➢ **USO COMÚN**

1. **Caché de Datos**

Utilizado para almacenar datos en memoria para acelerar el acceso a información que se accede con frecuencia.

2. **Colas de Mensajes**

Empleado como sistema de mensajería para implementar colas de mensajes y gestionar la comunicación entre componentes de una aplicación.

3. **Almacenamiento Temporal de Sesiones**

Puede utilizarse para almacenar datos de sesión en aplicaciones web, mejorando la velocidad de acceso a información de sesión.

4. **Contador y Puntuaciones en Conjuntas Ordenados**

Utilizado para implementar contadores y clasificaciones basadas en puntuaciones en conjuntos ordenados.

D.Cassandra

Es un sistema de gestión de bases de datos distribuido y NoSQL, diseñado para proporcionar escalabilidad lineal, alta disponibilidad y tolerancia a fallos.

➢ CARACTERÍSTICAS PRINCIPALES

1. **Modelo de Datos NoSQL**

Cassandra utiliza un modelo de datos NoSQL basado en pares clave-valor, donde los datos se organizan en columnas y filas, y la clave primaria actúa como un índice de acceso rápido.

2. **Escalabilidad Horizontal**

Cassandra está diseñada para escalar horizontalmente de manera eficiente. Puede agregar más nodos al clúster para manejar mayores volúmenes de datos y tráfico, sin tener que cambiar el diseño de la base de datos.

3. **Alta Disponibilidad y Tolerancia a Fallos**

Cassandra está diseñada para garantizar la alta disponibilidad incluso en presencia de fallos. Utiliza replicación y particionamiento para distribuir datos entre nodos y garantizar que haya copias redundantes.

4. **Arquitectura Descentralizada**

No tiene un nodo centralizado y cada nodo en el clúster tiene el mismo rol. Esto mejora la escalabilidad y la resistencia a fallos.

➢ USO COMÚN

1. **Aplicaciones Web a Escala**

Cassandra es ampliamente utilizado en aplicaciones web que requieren escalabilidad horizontal y manejo de grandes volúmenes de datos, como en redes sociales y comercio electrónico.

2. **Sistemas de Registro y Monitores**

Se utiliza para almacenar datos de registro y métricas en sistemas de monitoreo debido a su capacidad para manejar flujos de datos en tiempo real.

3. **Catálogos de Productos y Servicios**

Es adecuado para sistemas que gestionan catálogos de productos y servicios, donde se requiere un rápido acceso y actualizaciones frecuentes.

4. Aplicaciones con Altas Demandas de Escritura y Lectura

Cassandra se destaca en aplicaciones con altas demandas de escritura y lectura concurrentes, ya que su diseño distribuido y descentralizado permite un rendimiento eficiente.

4. Conclusión

En resumen, las bases de datos no relacionales han surgido como una solución flexible y escalable para gestionar grandes volúmenes de datos en entornos informáticos modernos. A diferencia de las bases de datos relacionales tradicionales, ofrecen una estructura más dinámica y adaptable, lo que las hace ideales para aplicaciones donde la velocidad y la escalabilidad son críticas, como en el ámbito de la web y las aplicaciones móviles.

5. Expectativas del proyecto

Este proyecto tiene como objetivo desarrollar una aplicación que almacene y gestione libros utilizando **MongoDB** como base de datos y **Atlas** como gestión de la misma. Se espera que, al finalizar el desarrollo, se logre una aplicación funcional, escalable y eficiente, aprovechando las ventajas de las bases de datos no relacionales.

Comprender el funcionamiento de MongoDB

- Explorar la arquitectura y los principios de **MongoDB**, comprendiendo su modelo de almacenamiento basado en documentos.
- Aprender a configurar y gestionar **MongoDB Atlas**, aprovechando sus capacidades de escalabilidad y seguridad.

Diseñar una base de datos óptima para almacenar libros

- Definir una estructura eficiente para los libros utilizando documentos JSON en **MongoDB**.
- Implementar esquemas adecuados para manejar información relevante como título, autor, género, fecha de publicación y estado del libro.
- Evaluar el uso de **colecciones, índices y agregaciones** para mejorar la eficiencia de consultas.

Desarrollar una aplicación que interactúe con MongoDB Atlas

- Implementar una API o interfaz que permita agregar, consultar, actualizar y eliminar libros en la base de datos.
- Utilizar **librerías y frameworks** adecuados (por ejemplo, Node.js con Mongoose o Python con PyMongo) para conectar la aplicación con **MongoDB Atlas**.
- Garantizar una interacción eficiente y segura con la base de datos, aplicando buenas prácticas de diseño.

Implementar autenticación y control de acceso

- Configurar mecanismos de autenticación y seguridad en **MongoDB Atlas** para proteger la base de datos.
- Aplicar permisos y roles adecuados para restringir el acceso a la información sensible.

6. Bibliografía

https://www.stackscale.com/es/blog/bases-de-datos-nosql/#Caracteristicas_y_ventajas_principales
https://thedataschools.com/que-es/nosql/
https://www.ibm.com/es-es/topics/nosql-databasesa
https://openwebinars.net/blog/sql-vs-nosql-comparativa-para-elegir-correctamente/
https://openwebinars.net/blog/que-es-mongodb/
https://neo4j.com/es/producto/#:~:text=Neo4j%20Graph%20Data%20Science%20es,cr%C3%ADticas%20y%20mejorar%20las%20predicciones.
https://www.enmilocalfunciona.io/conoces-neo4j-o-sabes-de-que-va/
https://openwebinars.net/blog/mongodb-vs-redis/
https://www.ibm.com/es-es/topics/redis
https://proximahost.es/blog/redis-almacen-datos-memoria/
https://openwebinars.net/blog/que-es-apache-cassandra/
https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/cassandra-base-de-datos-agilidad-y-rendimiento-a-prueba-de-fallos