Trabajo Práctico – Introducción a NoSQL

Parte 1 – Conceptos y contexto

1. Definición y origen

- Explica qué significa el término NoSQL y por qué surgió.

El término NoSQL, acrónimo de “no solo SQL”, se refiere a las bases de datos no relacionales que almacenan datos en un formato no tabular, en lugar de hacerlo en tablas relacionales basadas en reglas, como lo hacen las bases de datos relacionales. Las bases de datos NoSQL usan un modelo de esquema flexible que admite una amplia variedad de datos no estructurados, como documentos, pares clave-valor, columnas amplias y gráficos.

Las bases de datos NoSQL surgieron a finales de la década de 2000 a medida que el costo del almacenamiento disminuyó significativamente. Las bases de datos NoSQL permiten a los desarrolladores almacenar cantidades ingentes de datos no estructurados, lo que les da mucha flexibilidad.

Debido al crecimiento exponencial de la digitalización, las compañías ahora recopilan la mayor cantidad posible de datos no estructurados. Ser capaz de analizar y derivar información en tiempo real procesable de tales big data, Las compañías necesitan soluciones modernas que vayan más allá del simple almacenamiento. Las empresas necesitan una plataforma que pueda fácilmente escalar, transformar, y visualizar datos; crear cuadros de mando, reportes y gráficos; y trabajar con AI herramientas de business intelligence para acelerar la productividad de su negocio. Debido a su naturaleza flexible y distribuida, las bases de datos NoSQL (por ejemplo, MongoDB) brillan en estas tareas.

- ¿Qué problemas o limitaciones de las bases de datos relacionales motivaron la aparición de NoSQL?

Las bases de datos NoSQL surgieron para abordar las limitaciones de las bases de datos relacionales, especialmente en cuanto a escalabilidad, flexibilidad en el manejo de datos no estructurados y rendimiento en entornos distribuidos. Las bases de datos relacionales, con su rigidez en la estructura de datos y su dificultad para escalar horizontalmente, se vieron superadas por las necesidades de las aplicaciones modernas que manejan grandes volúmenes de datos y requieren alta disponibilidad y flexibilidad.

- Diferencia entre el significado inicial de “NoSQL” y el más aceptado actualmente (“Not Only SQL”).

El término inicialmente se asoció con bases de datos que no utilizaban el lenguaje SQL, ahora se refiere a bases de datos que ofrecen flexibilidad más allá de las bases de datos relacionales tradicionales.

2. Clasificación de bases de datos NoSQL

- Describe las cuatro categorías principales:
    - Documentos
    - Clave-valor
    - Columnares
    - Grafos
- Para cada categoría:
    - Explica cómo almacena la información.
    - Da dos ejemplos reales de bases de datos que pertenezcan a ese tipo.
    - Indica un caso de uso ideal.

Bases de datos orientadas a documentos

Una base de datos orientada a documentos almacena datos en documentos similares a objetos JSON (JavaScript Object Notation). Cada documento contiene pares de campos y valores. Por lo general, los valores pueden ser de varios tipos, incluidos elementos como cadenas, números, booleanos, matrices o incluso otros objetos. Una base de datos de documentos ofrece un modelo de datos flexible, muy adecuado para conjuntos de datos semiestructurados y, por lo general, no estructurados. También admiten estructuras anidadas, lo que facilita la representación de relaciones complejas o datos jerárquicos.

Caso de uso ideal: Son ideales para aplicaciones que requieren flexibilidad en el esquema de datos, como sistemas de gestión de contenido (CMS), aplicaciones móviles, y gestión de perfiles de usuarios.

Ejemplos de bases de datos de documentos son MongoDB y Couchbase.

Bases de datos clave-valor

Las bases de datos de clave-valor recopilan, recuperan y almacenan datos como agrupaciones de pares clave-valor. Esto significa que cada registro de datos está representado por una clave única y un valor asociado. La clave se emplea para recuperar el valor correspondiente de la base de datos. Por ejemplo, en una base de datos clave-valor de diseño de interiores, una clave podría ser " color " y el valor podría ser "púrpura”.

Caso de uso ideal: Excelente para el almacenamiento en caché, sesiones de usuario, gestión de sesiones en tiempo real, y aplicaciones que requieren acceso rápido a datos.

Ejemplos: AWS y ScyllaDB.

Bases de datos de familias de columnas

Las bases de datos de familias de columnas organizan los datos en columnas en lugar de filas, lo que resulta útil cuando se trabaja con conjuntos de datos amplios que son escasos en profundidad. De hecho, las tiendas familiares de columnas a veces se denominan "tiendas de columna ancha". En los almacenes de familias de columnas, cada fila tiene un conjunto diferente de columnas, y las columnas se agrupan en "familias". Estos modelos de datos son útiles cuando se trabaja con conjuntos de datos a gran escala que se benefician del escalado horizontal para optimizar el rendimiento.

Caso de uso ideal: Plataformas de streaming como Netflix que gestionan grandes volúmenes de datos de usuarios, aplicaciones de análisis en tiempo real de datos de sensores IoT, y sistemas de detección de fraude que analizan grandes cantidades de datos transaccionales.

Ejemplos: Apache, Cassandra y HBase.

Bases de datos de grafos

Las bases de datos de grafos almacenan datos en nodos y bordes. Los nodos suelen almacenar información sobre personas, lugares y cosas, mientras que las aristas almacenan información sobre las relaciones entre los nodos. Las bases de datos de grafos son excelentes herramientas para consultar estructuras de grafos (por ejemplo, redes sociales, jerarquías).

Caso de uso ideal: Ideales para aplicaciones que involucran relaciones complejas entre datos, como redes sociales, sistemas de recomendación y detección de fraudes.

Ejemplos: Neo4j, AWS y Kibana.

3. Comparativa con bases relacionales

- Enumera cinco diferencias clave (modelado de datos, escalabilidad, consultas, consistencia, flexibilidad de esquema, etc.).

Modelado de datos

NoSQL: Los modelos de datos varían según el tipo de base de datos NoSQL utilizada, por ejemplo, clave-valor, documento, gráfico y columna ancha, lo que hace que el modelo sea adecuado para datos semiestructurados y no estructurados.

RDBMS: RDBMS emplea una estructura de datos tabular, con datos representados como un conjunto de filas y columnas, lo que hace que el modelo sea adecuado para datos estructurados.

Esquema

NoSQL: Proporciona un esquema flexible en el que cada conjunto de pares de documentos/columna, columna y valor clave puede contener diferentes tipos de datos. Es más fácil cambiar el esquema, si es necesario, debido a la flexibilidad.

RDBMS: Este es un esquema fijo donde cada fila debe contener los mismos tipos de columna predefinidos. Es difícil cambiar el esquema una vez que se almacenan los datos.

Lenguaje de consulta

NoSQL: Varía según el tipo de base de datos NoSQL empleada. Por ejemplo, MongoDB tiene MQL y Neo4J usa Cypher.

RDBMS: Emplea el lenguaje de consulta estructurado (SQL).

Escalabilidad

NoSQL: NoSQL está diseñado para el escalado vertical y horizontal.

RDBMS: RDBMS está diseñado para escalamiento vertical. Sin embargo, puede ampliar las capacidades limitadas para el escalado horizontal.

Relaciones de datos

NoSQL: Las relaciones pueden ser anidadas, explícitas o implícitas.

RDBMS: Las relaciones se definen a través de claves foráneas y se accede a ellas mediante uniones.

Tipo de transacción

NoSQL: Las transacciones son ACID- o compatible con BASE.

RDBMS: Las transacciones cumplen con ACID.

Rendimiento

NoSQL: NoSQL es adecuado para procesamiento en tiempo real, análisis de big data y entornos distribuidos.

RDBMS: RDBMS es adecuado para cargas de trabajo de gran lectura y transacciones.

Consistencia de datos

NoSQL: Esto ofrece una alta consistencia de datos.

RDBMS: Esto ofrece consistencia eventual, en la mayoría de los casos.

Computación distribuida

Una de las principales razones para introducir NoSQL fue la computación distribuida, y las bases de datos NoSQL admiten el almacenamiento de datos distribuido, el escalado vertical y horizontal a través de la fragmentación, la replicación y la agrupación en clústeres.

RDBMS: RDBMS soporta la computación distribuida a través de clustering y replicación. Sin embargo, es menos escalable y flexible, ya que no está diseñado tradicionalmente para admitir la arquitectura distribuida.

Tolerancia a fallos

NoSQL: NoSQL tiene tolerancia a fallos incorporada y alta disponibilidad debido a la replicación de datos.

RDBMS: RDBMS emplea mecanismos de replicación, copia de seguridad y recuperación. Sin embargo, dado que están diseñados para estos, es posible que sea necesario implementar medidas adicionales, como mecanismos de recuperación ante desastres, durante el desarrollo de la aplicación.

Partición de datos

NoSQL: Se realiza mediante la partición y la replicación.

RDBMS: Admite particiones basadas en tabla y poda de particiones.

Mapeo de datos a objetos

NoSQL: NoSQL almacena los datos de diversas maneras, por ejemplo, como documentos JSON, almacenes de columnas anchos o pares de clave y valor. Proporciona abstracción a través de los marcos ODM (mapeo de datos de objetos) para trabajar con datos NoSQL de manera orientada a objetos.

RDBMS: RDBMS se basa más en la asignación de datos a objetos para que haya una integración perfecta entre las columnas de la base de datos y el código de la aplicación orientado a objetos.

- Explica brevemente qué es el modelo ACID y cómo se relaciona con el modelo BASE usado en muchas NoSQL.

El modelo **ACID** (Atomicidad, Consistencia, Aislamiento, Durabilidad) es un conjunto de propiedades que garantizan la fiabilidad de las transacciones en bases de datos, especialmente en bases de datos relacionales.

Este modelo prioriza la precisión y la integridad, ideal para entornos críticos (bancos, contabilidad, ERP).

El modelo **BASE** (Basically Available, Soft-state, Eventually Consistent) surge en bases de datos NoSQL, donde la prioridad suele ser la disponibilidad y escalabilidad antes que la consistencia estricta.

Este modelo sacrifica consistencia inmediata a favor de rendimiento y tolerancia a fallos, útil para aplicaciones distribuidas masivas (redes sociales, e-commerce global, big data)

Comparacion:

ACID: Consistencia fuerte, más rígido, seguro, pero menos escalable.

Si se necesita integridad absoluta (ej. transacciones financieras) → ACID.

BASE: Consistencia eventual, más flexible, pensado para sistemas distribuidos de gran escala.

Si se prioriza rendimiento y disponibilidad global (ej. Amazon, Facebook) → BASE.

Parte 2 – Casos de uso y ejemplos reales

1. Elige cuatro empresas o proyectos que utilicen bases NoSQL.

2. Para cada una, explica:
- Qué tipo de NoSQL utilizan.
- Qué problema específico resuelven con ella.
- Qué ventajas obtienen en comparación con una base de datos relacional.

Netflix

Base NoSQL: Apache Cassandra.

Uso: Guardar el estado de reproducción de los usuarios, recomendaciones y gestión de contenido en múltiples regiones.

Razón: Necesitan disponibilidad global y baja latencia para millones de usuarios al mismo tiempo.

Amazon

Base NoSQL: DynamoDB (propia de AWS).

Uso: Manejo de carritos de compra, catálogos y eventos de alto volumen como Prime Day.

Razón: Escalabilidad automática y millones de transacciones por segundo.

Meta (Facebook)

Base NoSQL: Cassandra y RocksDB.

Uso: Mensajería de Facebook Messenger y almacenamiento de publicaciones/likes.

Razón: Distribución geográfica, tolerancia a fallos y consultas rápidas en billones de filas.

Uber

Base NoSQL: Riak y Cassandra.

Uso: Geolocalización de autos en tiempo real y matching entre conductor-pasajero.

Razón: Necesitan respuestas en milisegundos y tolerancia a fallos en múltiples regiones.

3. Incluye un ejemplo del mundo open source y uno de una empresa tecnológica grande (ej. Netflix, Amazon, Meta, etc.).

Redis

Tipo de NoSQL: Clave-valor en memoria.

Problema que resuelven: Proveer caché ultrarrápida, gestión de sesiones y colas de mensajes en aplicaciones web y microservicios.

Ventajas frente a relacional:

Velocidad: operaciones en microsegundos al estar en memoria.

Estructuras de datos avanzadas (listas, sets ordenados, hashes).

Reduce carga en bases relacionales al actuar como caché.

Ejemplo de uso: GitLab, WordPress y muchos frameworks (como Django y Rails) lo integran para cachear y mejorar rendimiento.

Parte 3 – Tendencias y futuro

1. Investiga y explica dos tendencias actuales en el uso de NoSQL (por ejemplo: integración con IA, bases multimodelo, edge databases, etc.).

Bases de datos multimodelo

Qué son: Bases que permiten almacenar y consultar datos bajo más de un paradigma NoSQL (ej. documentos, grafos, clave-valor) e incluso relacional en un mismo motor.

Ejemplos:

- ArangoDB y OrientDB permiten manejar documentos y grafos en la misma base.
- Cosmos DB (Microsoft Azure) soporta múltiples APIs: MongoDB (documentos), Cassandra (columnar), Gremlin (grafos).

Beneficio: Evita tener que mantener varias bases diferentes para distintos casos de uso. Esto simplifica la arquitectura y reduce costos de integración.

Aplicación: Una app de e-commerce puede usar un modelo de documentos para los productos, un grafo para recomendaciones y un esquema clave-valor para caché… todo dentro de la misma base.

Integración con Inteligencia Artificial (IA) y Machine Learning

Qué es: Uso de bases NoSQL para soportar modelos de IA/ML que necesitan procesar grandes volúmenes de datos no estructurados (texto, imágenes, logs, IoT).

Ejemplos:

- Vector databases como Pinecone, Weaviate o Milvus, especializadas en búsquedas semánticas para embeddings de IA.
- MongoDB Atlas integra soporte para consultas vectoriales, pensadas para modelos de lenguaje (LLMs).

Beneficio: Permite búsquedas más inteligentes (ejemplo: buscar por “similitud” en lugar de coincidencia exacta).

Aplicación: Un chatbot con IA puede buscar respuestas relevantes en un corpus de documentos usando embeddings guardados en una base vectorial NoSQL.

2. ¿Crees que las bases relacionales desaparecerán o convivirán con NoSQL en el futuro? Justifica tu respuesta con fundamentos técnicos y ejemplos.

No. Las bases relacionales (SQL) no desaparecerán porque siguen siendo la mejor opción para muchos escenarios críticos, pero NoSQL gana terreno donde se requieren escalabilidad masiva, flexibilidad de esquemas o datos no estructurados.

Incluso hay empresas que utilizan las 2 para distintas partes del sistema, llamado “Polyglot Persistence”, es decir, elegir la base más adecuada para cada parte del sistema.

Ejemplo: Uber usa MySQL (transacciones financieras) y Cassandra/Riak (geolocalización en tiempo real).