Introducción a las Bases de Datos NoSQL

¿Qué es una base de datos NoSQL?

Las bases de datos NoSQL (del inglés **Not Only SQL**) son sistemas de almacenamiento de datos que **no utilizan el modelo relacional tradicional**.
En lugar de almacenar los datos en **tablas con filas y columnas**, como sucede en SQL Server, las bases NoSQL utilizan **estructuras más flexibles**, tales como:

- Documentos (por ejemplo, en formato JSON)
- Pares clave-valor
- Columnas independientes
- Grafos

Importante: NoSQL no significa que estas bases "no usen SQL" en absoluto, sino que **no se basan exclusivamente en él** ni en la estructura rígida de tablas.

¿Por qué surgen las bases NoSQL?

Durante años, las bases de datos relacionales fueron la solución ideal para la mayoría de las aplicaciones.
Sin embargo, con la llegada de internet, las redes sociales, las apps móviles y los sistemas en la nube, **empezaron a surgir nuevos desafíos**:

Escenarios problemáticos en SQL tradicional

Cambios frecuentes en la estructura de los datos

En una base relacional como SQL Server, antes de insertar datos se debe definir una estructura fija (esquema).
Esto implica crear tablas con columnas predeterminadas, y todos los registros deben respetar esa estructura.

El problema surge cuando los datos no siempre tienen los mismos campos, algo común en sistemas modernos. Por ejemplo:

- Productos con características diferentes (una notebook vs. una remera).
- Formularios que se adaptan según el usuario o el contexto.

Modificar una tabla en SQL para agregar o quitar columnas implica cambios estructurales que pueden ser complejos y riesgosos si hay mucha información cargada.

En cambio, las bases NoSQL permiten almacenar datos con estructura flexible, sin necesidad de definir todas las columnas por adelantado.

Esto permite adaptarse fácilmente a contextos dinámicos donde los datos cambian con frecuencia.

Grandes volúmenes de información

Uno de los motivos principales por los que surgieron las bases NoSQL fue la necesidad de manejar enormes volúmenes de datos en tiempo real.

Sistemas modernos como Facebook, Amazon, Netflix, Twitter o Google reciben millones de solicitudes por segundo: nuevos usuarios que se registran, productos que se agregan al carrito, videos que se reproducen, mensajes que se envían, entre otros.

En una base de datos relacional tradicional (como SQL Server), el rendimiento comienza a degradarse cuando la cantidad de registros, conexiones o consultas simultáneas supera cierto límite físico o lógico.

¿Por qué sucede esto?

Las bases relacionales fueron pensadas para trabajar en un único servidor potente (escala vertical). Cuando la demanda crece, se necesita:

- Un procesador más rápido.
- Más memoria RAM.
- Mejor disco rígido o SSD.

Pero todo eso tiene un límite físico y económico. Y aún con el mejor hardware, la arquitectura monolítica relacional no está diseñada para repartir fácilmente la carga entre múltiples servidores.

¿Qué hace diferente a NoSQL?

Las bases NoSQL están pensadas desde su diseño para escalar horizontalmente:
Es decir, cuando hace falta más capacidad, simplemente se agregan nuevos servidores al sistema, y la base distribuye los datos entre ellos automáticamente. A esto se lo conoce como *sharding*.

Gracias a este enfoque, NoSQL puede sostener sistemas de gran escala con alta disponibilidad y rendimiento.

Baja performance en relaciones complejas

En bases relacionales, los datos se guardan en tablas separadas: una tabla de usuarios, otra de publicaciones, otra de comentarios, etc.
Para armar una vista completa (por ejemplo, un post con el nombre del autor y los comentarios), hay que hacer JOINs entre varias tablas.

Esto no es lento por sí mismo, pero a medida que los datos crecen y las relaciones se hacen más complejas, las consultas con JOINs pueden volverse más costosas en tiempo y recursos, especialmente en sistemas con alta demanda y estructuras profundamente conectadas.

Las bases NoSQL resuelven esto de una forma distinta:
En lugar de armar los datos en el momento de la consulta, los guardan ya preparados para ser leídos directamente.
No necesitan hacer uniones porque toda la información relacionada está en el mismo lugar, lista para usarse.

Esto evita operaciones costosas y mejora la eficiencia en entornos con muchos datos interrelacionados.

¿Qué solución propone NoSQL?

Las bases NoSQL surgieron para dar respuesta a estos problemas.
En lugar de seguir un único modelo, como el relacional, **existen distintos tipos de bases NoSQL**, cada uno con un enfoque particular.
Además, comparten ciertas características clave:

- **Flexibilidad:** no necesitan un esquema fijo.
- **Escalabilidad horizontal:** pueden crecer fácilmente en varios servidores.
- **Alto rendimiento:** optimizadas para operaciones simples, rápidas y masivas.
- **Desnormalización consciente:** permite evitar joins incorporando toda la información en un solo documento o registro.

Tipos de bases de datos NoSQL

A diferencia del mundo relacional, donde todas las bases se basan en tablas, en NoSQL existen **familias de bases**. A continuación las mencionamos y explicamos brevemente:

1. Bases de datos documentales

Guardan la información como documentos individuales, usualmente en formato **JSON** o similar.
Cada documento representa una entidad (como un producto, usuario o empleado) y puede tener una estructura única.

Ejemplo:

```
{
  "nombre": "Juan",
  "edad": 30,
  "puesto": "Desarrollador",
  "habilidades": ["JavaScript", "MongoDB"]
}
```

Ventajas:

- No requiere definir columnas fijas.
- Fácil de representar objetos del mundo real.

- Muy útil para APIs y aplicaciones web.

Ejemplo de base documental: **MongoDB**

2. Bases de datos clave-valor

Cada dato se guarda con una clave única.
Es como un diccionario: buscamos algo por su nombre clave y obtenemos el valor.

Ejemplo:

```
"usuario:1023" → "Juan Pérez"
```

Ventajas:

- Extremadamente rápidas.
- Sencillas de implementar.

Ejemplo de base clave-valor: **Redis**

3. Bases de datos columnares

En lugar de almacenar los datos por filas, como en SQL, los agrupan por **columnas**. Esto permite acceder más rápidamente a todos los valores de una columna en particular.

Ventajas:

- Alto rendimiento para consultas analíticas.
- Ideal para procesamiento masivo de datos.

Ejemplo de base columnar: **Apache Cassandra**

4. Bases de datos de grafos

Están diseñadas para manejar datos con relaciones muy conectadas.
Representan las entidades como **nodos**, y las relaciones entre ellas como **aristas (líneas)**.

Ejemplo:

```
Juan ——amigo de——> Pedro
     └—trabaja con—> Laura
```

Ventajas:

- Excelente rendimiento para recorrer relaciones.
- Se adapta naturalmente a redes sociales, rutas, sistemas de recomendación, etc.

Ejemplo de base de grafos: **Neo4j**