

Bases de Datos NoSQL: Más Allá de lo Relacional

Las bases de datos NoSQL (Not Only SQL) representan un conjunto de sistemas gestores diseñados para superar las limitaciones de los modelos relacionales tradicionales. Nacieron como respuesta a las necesidades de aplicaciones web a gran escala, redes sociales, sistemas en tiempo real, Big Data e Internet of Things (IoT).

Estos sistemas no están restringidos por el modelo clásico de tablas, filas y columnas, sino que utilizan estructuras más flexibles como documentos, pares clave-valor, columnas anchas y grafos, permitiendo mayor escalabilidad y rendimiento en entornos modernos y distribuidos.

R por Kibernum Capacitación S.A.





Preguntas de Activación de Contenido

1. ¿Qué limitaciones crees que pueden tener las bases de datos relacionales al manejar grandes volúmenes de datos no estructurados?
2. ¿Has trabajado antes con datos en formato JSON u otro tipo de datos semiestructurados? ¿En qué contexto?
3. ¿Qué entiendes por “esquema laxo” en una base de datos y cómo crees que eso influye en el desarrollo ágil?

Ventajas de los Sistemas NoSQL

Variedad de Datos

Permiten almacenar estructuras flexibles como JSON, BSON, listas, mapas y más. Ideal para datos semiestructurados, heterogéneos y en constante cambio.

Escalabilidad

Diseñados para escalar horizontalmente (agregando más nodos) en lugar de verticalmente, permitiendo escalar con bajo coste, mayor tolerancia a fallos y distribución geográfica.

Desempeño

Más rápidos en operaciones específicas como lectura/escritura masiva, almacenamiento de grandes volúmenes y operaciones en tiempo real.

Esquema Flexible

No exigen un esquema fijo, permitiendo mayor flexibilidad en desarrollo ágil, menor rigidez ante cambios y evolución de modelos sin afectar datos existentes.



Bases de Datos Clave-Valor

```
# En Redis  
SET usuario:1 "Juan Pérez"  
GET usuario:1
```

Características

Las bases de datos clave-valor representan el modelo más simple de NoSQL. Cada dato se almacena como un par clave => valor, similar a un diccionario o mapa en programación.

Este modelo destaca por su simplicidad y eficiencia, permitiendo acceso ultrarrápido a los datos cuando se conoce la clave exacta.

Casos de Uso

- Manejo de sesiones de usuario
- Sistemas de caché
- Contadores y rankings
- Almacenamiento de preferencias

Ejemplos de motores: Redis y Amazon DynamoDB (en modo clave-valor).

Bases de Datos Orientadas a Documentos



Estructura Flexible

Almacenan datos como documentos JSON o BSON, permitiendo estructuras anidadas y complejas que representan objetos del mundo real.



Consultas Potentes

Ofrecen capacidades avanzadas de consulta sobre el contenido de los documentos, incluyendo búsquedas en campos anidados.



Integración Natural

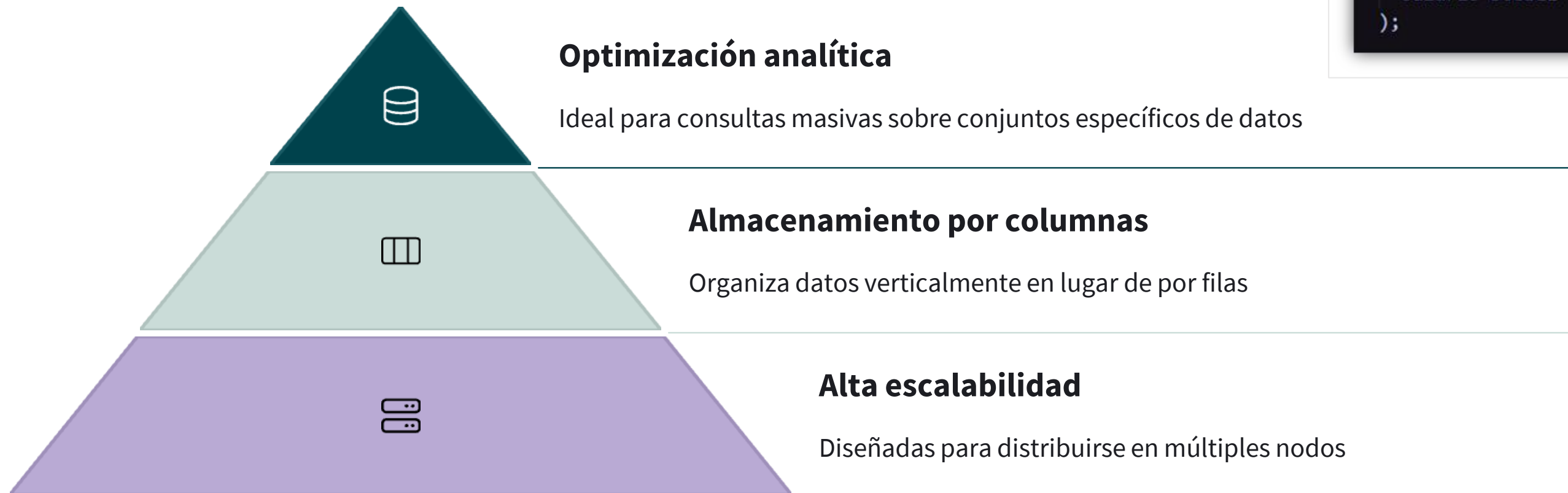
Se integran fácilmente con lenguajes de programación modernos al usar formatos como JSON, reduciendo la impedancia objeto-relacional.

Los principales motores de este tipo incluyen MongoDB, Couchbase y Amazon DocumentDB. Son ideales para aplicaciones web, sistemas de usuarios y plataformas de comercio electrónico donde la flexibilidad del esquema es crucial.

```
{  
  "nombre": "Laura",  
  "correo": "laura@gmail.com",  
  "direccion": {  
    "ciudad": "Valparaíso",  
    "pais": "Chile"  
  }  
}
```

Bases de Datos Orientadas a Columnas

```
-- En Cassandra (CQL)  
CREATE TABLE empleados (  
  id UUID PRIMARY KEY,  
  nombre TEXT,  
  salario DOUBLE  
);
```



A diferencia de las bases de datos relacionales que almacenan datos por filas, las orientadas a columnas agrupan los datos por columnas. Por ejemplo, en lugar de almacenar [1, "Ana", 23] y [2, "Luis", 45] como filas, almacenan [1, 2], ["Ana", "Luis"] y [23, 45] como columnas separadas.

Motores como Apache Cassandra, HBase y ScyllaDB son ejemplos de este tipo. Son especialmente útiles para Big Data, Data Warehouses y sistemas de reporting donde se analizan grandes volúmenes de datos.

Bases de Datos Orientadas a Grafos

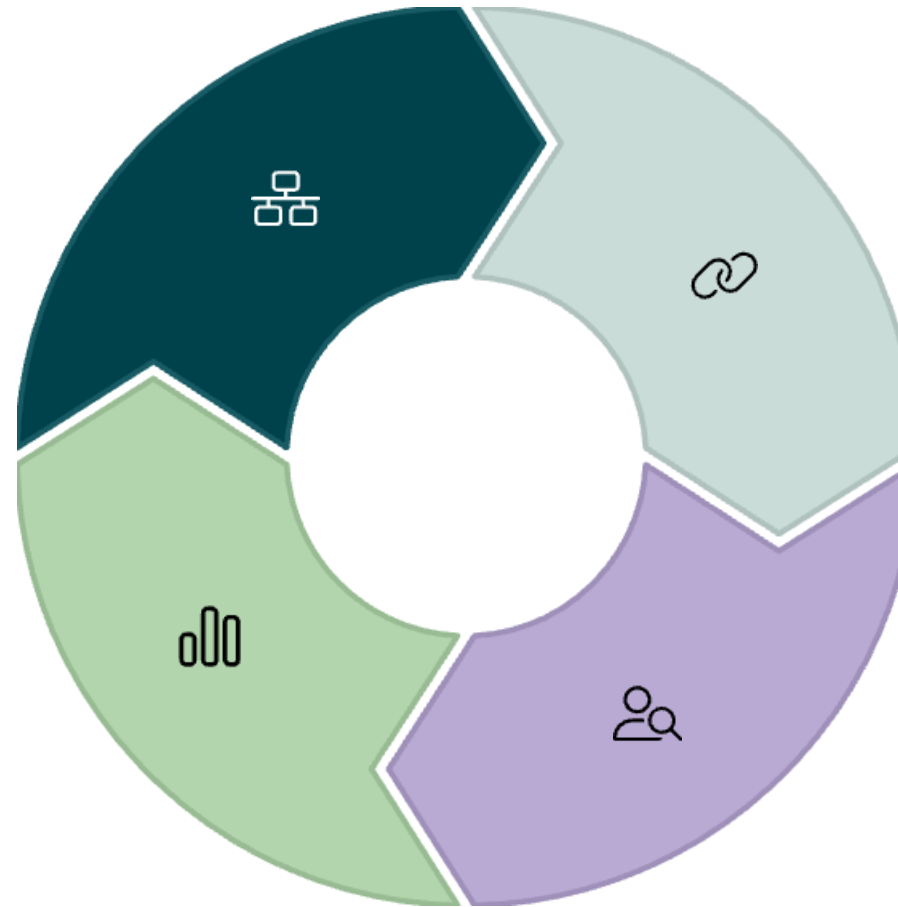
```
// En Neo4j
CREATE (a:Persona {nombre: "Carlos"})
CREATE (b:Persona {nombre: "Ana"})
CREATE (a)-[:AMIGO_DE]->(b)
```

Nodos

Representan entidades como personas, productos o conceptos

Análisis

Facilitan el descubrimiento de insights basados en conexiones



Relaciones

Conectan nodos con propiedades y direccionalidad

Consultas

Permiten recorrer el grafo para encontrar patrones complejos

Las bases de datos orientadas a grafos están optimizadas para representar y consultar relaciones complejas entre entidades. Son perfectas para casos donde las conexiones entre los datos son tan importantes como los datos mismos.

Neo4j, ArangoDB y Amazon Neptune son ejemplos destacados. Se utilizan ampliamente en redes sociales, sistemas de recomendación y motores de búsqueda semántica.

Bases de Datos In-Memory

Almacenamiento en RAM

Todos los datos se mantienen en memoria principal, eliminando la latencia de acceso a disco y proporcionando tiempos de respuesta en microsegundos.

Estructuras optimizadas

Utilizan estructuras de datos especialmente diseñadas para maximizar el rendimiento en memoria, como tablas hash, listas enlazadas y árboles balanceados.

Persistencia opcional

Aunque operan principalmente en memoria, suelen ofrecer mecanismos de persistencia como snapshots o logs de operaciones para recuperación ante fallos.

Redis y Memcached son los ejemplos más populares de este tipo de bases de datos. Se utilizan principalmente como cachés, para gestionar leaderboards en tiempo real y como colas de tareas en sistemas distribuidos.

```
# Redis nuevamente como ejemplo  
INCR contador_visitas
```




Casos de Uso por Tipo de NoSQL

Caso de uso	Tipo de NoSQL	Motor recomendado
Manejo de sesiones	Key-Value	Redis
Tienda online	Document-Oriented	MongoDB
Analytics y métricas	Column-Oriented	Cassandra
Red social	Graph-Oriented	Neo4j
Alta velocidad (cache)	In-Memory	Redis / Memcached
Catálogos flexibles	Document-Oriented	Couchbase
Personalización de contenido	Graph-Oriented	ArangoDB
Replicación distribuida	Column-Oriented	ScyllaDB

Actividad Práctica Guiada: Introducción a MongoDB en Linux y Windows

Objetivo

Guiar al estudiante en el uso básico de una base de datos NoSQL orientada a documentos (MongoDB), aplicando comandos esenciales para insertar, consultar, actualizar y eliminar documentos desde el cliente interactivo en Linux o Windows.

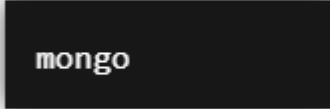
Instalación de MongoDB

- Para instalación de MongoDB para Window, ver el siguiente [link](#)
- Para instalación de MongoDB para Linux, ver el siguiente [link](#)

Paso a Paso Detallado


1. Abrir la terminal MongoDB

- En **Linux**: abre una terminal y ejecuta:



```
mongo
```

- En **Windows**: abre **CMD** o **PowerShell** y ejecuta:



```
mongo
```

Con esto ingresas al shell interactivo de MongoDB en ambos sistemas.

Actividad Práctica Guiada: Introducción a MongoDB en Linux y Windows

2. Crear o seleccionar una base de datos

```
use tienda;
```

Esto selecciona (o crea si no existe) una base de datos llamada tienda.

3. Insertar un documento en la colección productos

```
db.productos.insertOne({  
  nombre: "Zapatilla deportiva",  
  precio: 49990,  
  stock: 10,  
  categoria: "calzado"  
});
```

Se inserta un documento JSON. MongoDB permite que otros documentos tengan estructuras diferentes.

Actividad Práctica Guiada: Introducción a MongoDB en Linux y Windows

4. Insertar múltiples documentos

```
db.productos.insertMany([
  { nombre: "Polera básica", precio: 12990, stock: 25, categoria: "ropa" },
  { nombre: "Mochila escolar", precio: 29990, stock: 15, categoria: "accesorios" }
]);
```

Se insertan múltiples documentos con un solo comando.

5. Consultar todos los documentos

```
db.productos.find().pretty();
```

pretty() mejora la legibilidad del resultado.

Actividad Práctica Guiada: Introducción a MongoDB en Linux y Windows

6. Consultar documentos por criterio

```
db.productos.find({ categoria: "ropa" });
```

Filtra los documentos que coincidan con la categoría indicada.

7. Actualizar un documento

```
db.productos.updateOne(  
  { nombre: "Polera básica" },  
  { $set: { stock: 20 } }  
);
```

Actualiza el campo stock del documento correspondiente.

Actividad Práctica Guiada: Introducción a MongoDB en Linux y Windows

8. Eliminar un documento

```
db.productos.deleteOne({ nombre: "Mochila escolar" });
```

Elimina el documento cuyo nombre coincida.

Resultado Esperado

- Inserción exitosa de documentos con distintas estructuras.
- Comprensión del uso de colecciones y comandos básicos de MongoDB.
- Práctica de operaciones CRUD: Create, Read, Update, Delete.

Material Complementario



 https://www.youtube.com/watch?v=NLXCt_MECJQ

 [Documentación oficial de MongoDB \(modelo orientado a documentos\)](#)

Preguntas de Reflexión Final

- 1) ¿Cuál de los tipos de bases de datos NoSQL te pareció más interesante o útil? ¿Por qué?
- 2) ¿En qué tipos de proyectos reales considerarías usar una base NoSQL en lugar de una relacional?
- 3) ¿Cómo se relacionan las ventajas de escalabilidad y flexibilidad del modelo NoSQL con las exigencias actuales del desarrollo de software?

