Bases de datos no relacionales

NoSQL es un término que significa “Not Only SQL” y se refiere a un conjunto de sistemas gestores de bases de datos no relacionales, diseñados para trabajar con grandes volúmenes de datos, datos semiestructurados o no estructurados, y para proporcionar alta escalabilidad y rendimiento en entornos modernos, distribuidos y en tiempo real.

No están restringidas por el modelo relacional clásico de tablas, filas y columnas. En cambio, usan estructuras como documentos, pares clave-valor, columnas anchas y grafos.

por Kibernum Capacitación S.A.

¿Por qué nació NoSQL?

La necesidad de NoSQL surgió con el crecimiento de:

- Aplicaciones web a gran escala
- Redes sociales
- Sistemas en tiempo real
- Big Data
- Internet of Things (IoT)

Los sistemas tradicionales (RDBMS) comenzaron a tener limitaciones para escalar horizontalmente, almacenar datos variados y ofrecer desempeño constante ante millones de transacciones.

Ventajas de NoSQL

1. Variedad de datos
2. Escalabilidad
3. Desempeño
4. Definición laxa del esquema

1 Variedad de datos

Las bases NoSQL permiten almacenar estructuras más flexibles, como:

- JSON, BSON (MongoDB)
- Listas y mapas (Redis)
- Columnas anchas (Cassandra)
- Nodos y relaciones (Neo4j)

Esto es ideal para datos:

- Semiestructurados
- Heterogéneos
- En constante cambio

2 Escalabilidad

Los RDBMS escalan verticalmente (mejorando hardware del servidor), mientras que NoSQL se diseñó para escalar horizontalmente (agregando más nodos o servidores).

Esto permite:

- Escalar con bajo costo
- Mayor tolerancia a fallos
- Distribución geográfica

3 Desempeño

NoSQL suele ser más rápido en operaciones específicas como:

- Lectura/escritura masiva
- Almacenamiento de grandes volúmenes
- Operaciones en tiempo real

Esto se logra sacrificando parte de la consistencia estricta (recordemos BASE vs ACID).

4 Definición laxa del esquema

Las bases de datos NoSQL no exigen un esquema fijo. Por ejemplo, en MongoDB puedes insertar un documento con una estructura diferente a otro.
Lo que permite:

- Mayor flexibilidad en el desarrollo ágil
- Menor rigidez en cambios de requisitos
- Evolución de modelos sin afectar datos existentes

Tipos de Bases de Datos NoSQL

NoSQL no es una sola tecnología, sino una familia de modelos de datos. A continuación, te detallo los principales:

- Key-Value (clave-valor)
- Document-Oriented (orientadas a documentos)
- Column-Oriented (orientadas a columnas)
- Graph-Oriented (orientadas a grafos)
- In-Memory (almacenamiento en memoria)

Key-Value y Document-Oriented

Key-Value (clave-valor)

Descripción: Cada dato se almacena como un par clave => valor.

Ejemplo: "session123" => "usuario=juan;rol=admin"

Usos comunes:

- Manejo de sesiones
- Almacenamiento en caché
- Contadores, rankings

Ejemplos de motores:

- Redis
- Amazon DynamoDB (modo clave-valor)

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

Document-Oriented (orientadas a documentos)

Descripción: Los datos se almacenan como **documentos JSON** o similares, que representan objetos del mundo real.

Ejemplo:

Document-Oriented (orientadas a documentos):

Usos comunes:

- Aplicaciones web
- Sistemas de usuarios
- E-commerce

Ejemplos de motores:

- MongoDB
- Couchbase
- Amazon DocumentDB

```
// En MongoDB
db.usuarios.insertOne({
  nombre: "Laura",
  correo: "laura@gmail.com"
});
```

Column-Oriented (orientadas a columnas)

Organizan los datos por columnas, en lugar de por filas. Son ideales para consultas analíticas masivas.

Ejemplo conceptual:

id	nombre	edad
1	Ana	23
2	Luis	45

Pero internamente lo almacena así:

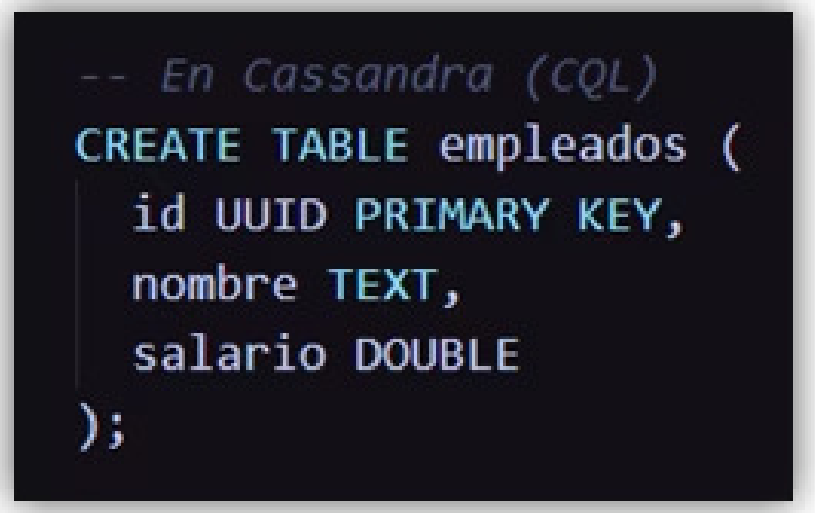
- Columna id: [1, 2]
- Columna nombre: [“Ana”, “Luis”]
- Columna edad: [23, 45]

Usos comunes:

- Big Data
- Data Warehouses
- Reporting

Ejemplos de motores:

- Apache Cassandra
- HBase
- ScyllaDB



Graph-Oriented (orientadas a grafos)

Descripción: Almacenan nodos (entidades) y relaciones entre ellos. Son excelentes para **representar conexiones** complejas.

Ejemplo:

- Carlos —[AMIGO_DE]→ Ana

Usos comunes:

- Redes sociales
- Recomendaciones
- Motores de búsqueda semántica

Ejemplos de motores:

- Neo4j
- ArangoDB
- Amazon Neptune

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

In-Memory (almacenamiento en memoria)

Descripción: Bases que almacenan datos en RAM para acceder a ellos de forma ultra rápida.

Usos comunes:

- Cachés
- Leaderboards
- Colas de tareas

Ejemplos de motores:

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

- Redis
- Memcached

Casos de Uso y Ejemplos 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

Reflexión final

Las bases de datos NoSQL no buscan reemplazar a las relacionales, sino complementarlas en contextos donde:

- El volumen de datos es grande
- La estructura varía con el tiempo
- Se requiere alta disponibilidad y escalabilidad

En un mundo dominado por aplicaciones móviles, sistemas distribuidos y Big Data, NoSQL se vuelve una herramienta poderosa que permite construir soluciones modernas, flexibles y eficientes.

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.

Paso a Paso Detallado

Instalación de MongoDB

- Para instalación de MongoDB para Window, ver el siguiente **link**
- Para instalación de MongoDB para Linux, ver el siguiente **link**

1. Abrir la terminal MongoDB

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

```
mongo
```

- En **Windows**: abre **CMD** o **PowerShell** y ejecuta: *Con esto ingresas al shell interactivo de MongoDB en ambos sistemas.*

```
mongo
```

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

```
use tienda;
```

3. Insertar un documento en la colección productos: Se inserta un documento JSON. MongoDB permite que otros documentos tengan estructuras diferentes.

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

4. Insertar múltiples documentos: Se insertan múltiples documentos con un solo comando.

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

5. Consultar todos los documentos: pretty() mejora la legibilidad del resultado.

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

6. Consultar documentos por criterio: Filtra los documentos que coincidan con la categoría indicada.

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

7. Actualizar un documento: Actualiza el campo stock del documento correspondiente.

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

8. Eliminar un documento: Elimina el documento cuyo nombre coincida.

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

R**esultado 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

🎥 **Material Complementario**

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

🗒️ Documentación oficial de MongoDB (modelo orientado a documentos)