**CLASE COMPLETA Y GUÍA DE USO DE MONGODB (Shell, Compass y Atlas)**

**📚 Clase: Introducción a MongoDB (Bases de datos NoSQL)**

**🎯 Objetivos de aprendizaje**

* Comprender qué es una base de datos NoSQL y cuándo conviene usarla.
* Conocer la estructura de datos en MongoDB.
* Aprender operaciones CRUD.
* Conocer la arquitectura de MongoDB (Replica Set, Sharding).
* Realizar consultas y manipulación de datos tanto por consola (Shell), como desde Compass y Atlas.

**✅ Parte 1: Teórica didáctica**

**Introducción a las Bases de Datos NoSQL**

**Explicación sencilla:**

* 1. **¿Qué es una base de datos relacional?**
* Son las que usan tablas (como Excel).
* Se usan relaciones con claves primarias y foráneas.
* Ej: bases de datos de usuarios, bibliotecas, bancos.

**1.2 ¿Qué es una base de datos NoSQL?**

* No usan tablas tradicionales.
* Guardan datos en **otros formatos**, como documentos JSON, grafos, clave-valor, etc.
* Ideales para datos muy variados y/o en gran volumen.
* No significa "sin SQL", sino que usa otra estructura.

**🔑 1. Clave/Valor (Key/Value)**

Es el modelo más simple. Cada dato se almacena como un **par clave-valor**, donde la clave es única y el valor es lo que se quiere guardar.

📦 **Ejemplo** (como si fuera un diccionario en Python):

{

"usuario123": "Juan Pérez",

"usuario124": "María López"

}

🧠 Imaginá que "usuario123" es la clave, y "Juan Pérez" es el valor asociado. Es como un índice rápido para acceder a un valor.

🛠️ Bases que usan este modelo: Redis, DynamoDB (de Amazon).

**🧾 2. Documento (por ejemplo JSON)**

Aquí los datos se guardan como **documentos estructurados**, muy comúnmente en formato **JSON** (JavaScript Object Notation).

📦 **Ejemplo** de un documento JSON:

{

"id": "123",

"nombre": "Lucía",

"edad": 28,

"dirección": {

"calle": "Av. Colón",

"ciudad": "Mar del Plata"

}

}

🧠 Este formato es muy flexible: podés guardar datos anidados, listas, objetos, etc.

🛠️ Bases que usan este modelo: MongoDB, CouchDB.

**🌐 3. Grafo**

Este modelo representa los datos como **nodos (vértices)** y **relaciones (aristas o bordes)**. Es útil cuando querés representar **conexiones entre entidades**, como redes sociales.

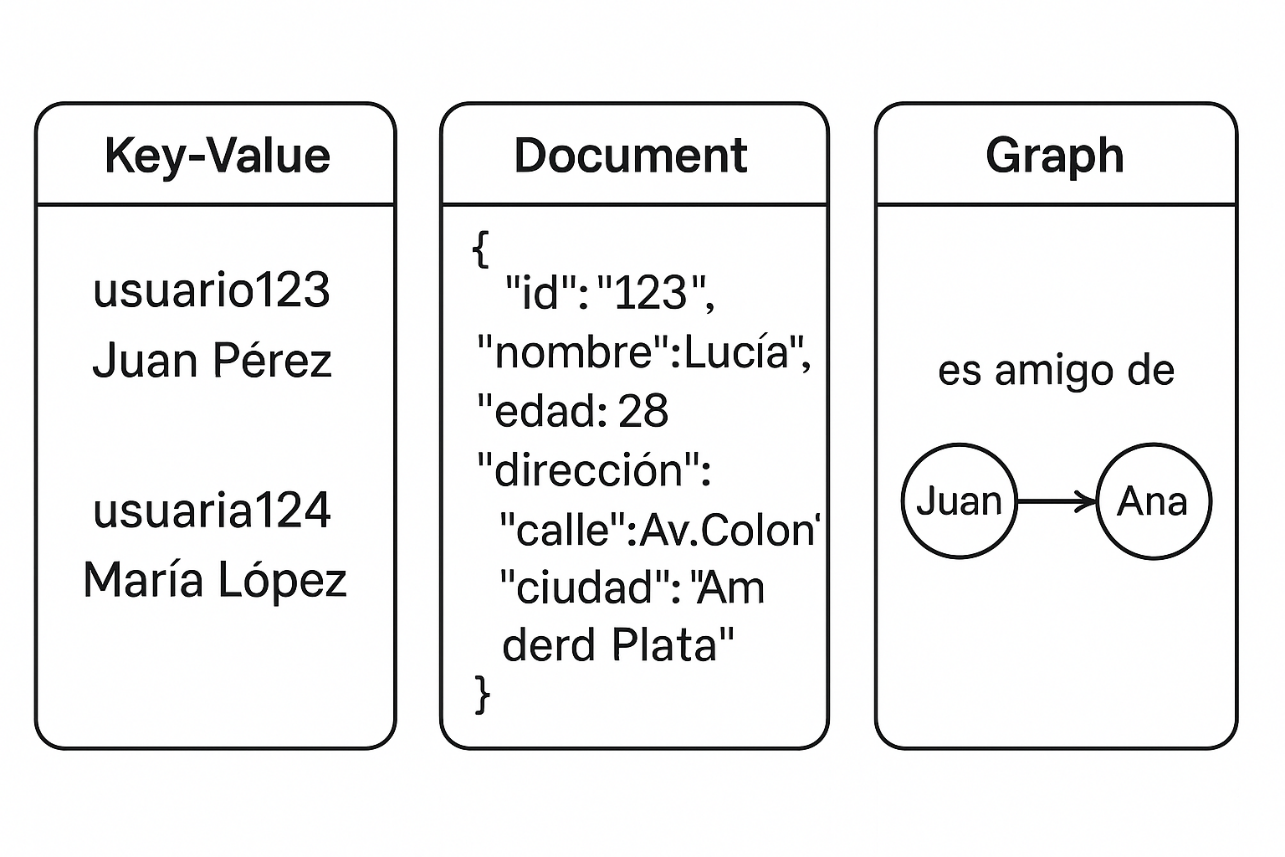
📦 **Ejemplo** (con nodos y relaciones):

* Nodo 1: Juan
* Nodo 2: Ana
* Relación: Juan **es amigo de** Ana

🧠 Así se vería en forma textual:

(Juan) ---es amigo de---> (Ana)

🛠️ Bases que usan este modelo: Neo4j, Amazon Neptune.



1️⃣ Almacenes de datos de documentos

Guardan datos en documentos estructurados, usualmente en JSON, BSON o XML. Son flexibles y permiten guardar datos anidados.

📦 Ejemplo:

{

"id": "u123",

"nombre": "Lucía",

"edad": 25,

"intereses": ["lectura", "viajes"]

}

🛠️ Ejemplo de base de datos: MongoDB, CouchDB.

2️⃣ Almacenes de datos en columnas

Guardan los datos por columnas en lugar de filas, lo que mejora el rendimiento en análisis de grandes volúmenes de datos.

📦 Ejemplo (simplificado):

| ID | Nombre | Edad |

|----|--------|------|

| 01 | Ana | 30 |

| 02 | Luis | 25 |

Pero internamente se almacena así:

"Nombre": ["Ana", "Luis"],

"Edad": [30, 25]

🛠️ Ejemplo de base de datos: Apache Cassandra, HBase.

3️⃣ Almacenes de datos de clave/valor

Guardan los datos como un par clave → valor. Muy simples y rápidos para búsquedas directas.

📦 Ejemplo:

"usuario123": "Juan Pérez"

🛠️ Ejemplo de base de datos: Redis, Amazon DynamoDB.

4️⃣ Almacenes de datos de grafos

Representan los datos en forma de nodos y relaciones. Perfectos para redes sociales, rutas, relaciones entre cosas.

📦 Ejemplo:

(Juan) ---es amigo de---> (Ana)

🛠️ Ejemplo de base de datos: Neo4j, ArangoDB.

5️⃣ Almacenes de datos de series temporales

Especiales para guardar datos que cambian con el tiempo, como temperatura, sensores, métricas, etc.

📦 Ejemplo:

{

"sensor\_id": "temp01",

"valores": [

{"tiempo": "2025-04-08T10:00", "valor": 22.5},

{"tiempo": "2025-04-08T11:00", "valor": 23.1}

]

}

🛠️ Ejemplo de base de datos: InfluxDB, TimescaleDB.

6️⃣ Almacenes de datos de objetos

Se usan para guardar objetos tal como se representan en la programación orientada a objetos. Cada objeto se almacena como una entidad completa.

📦 Ejemplo en pseudocódigo:

Objeto Persona {

nombre = "Carlos"

edad = 40

}

🛠️ Ejemplo de base de datos: db4o, ObjectDB.

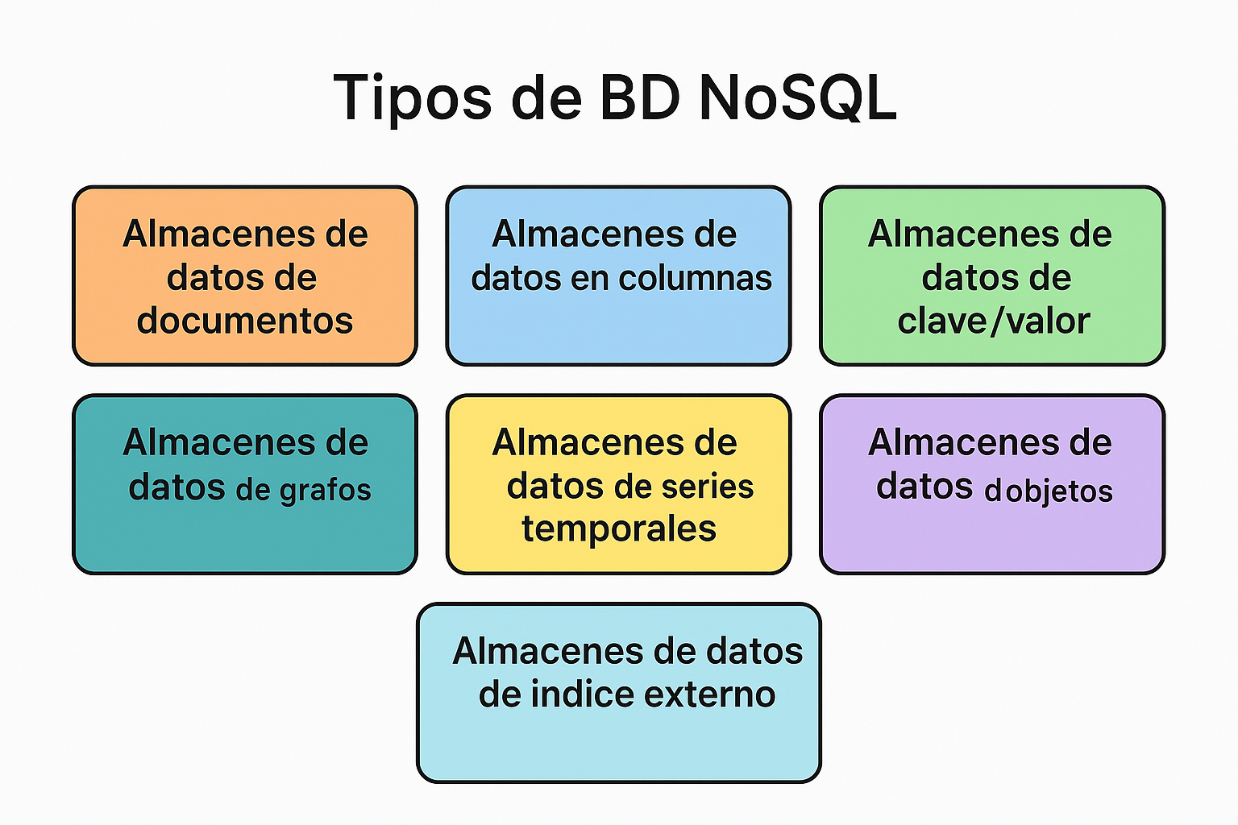
7️⃣ Almacenes de datos de índice externo

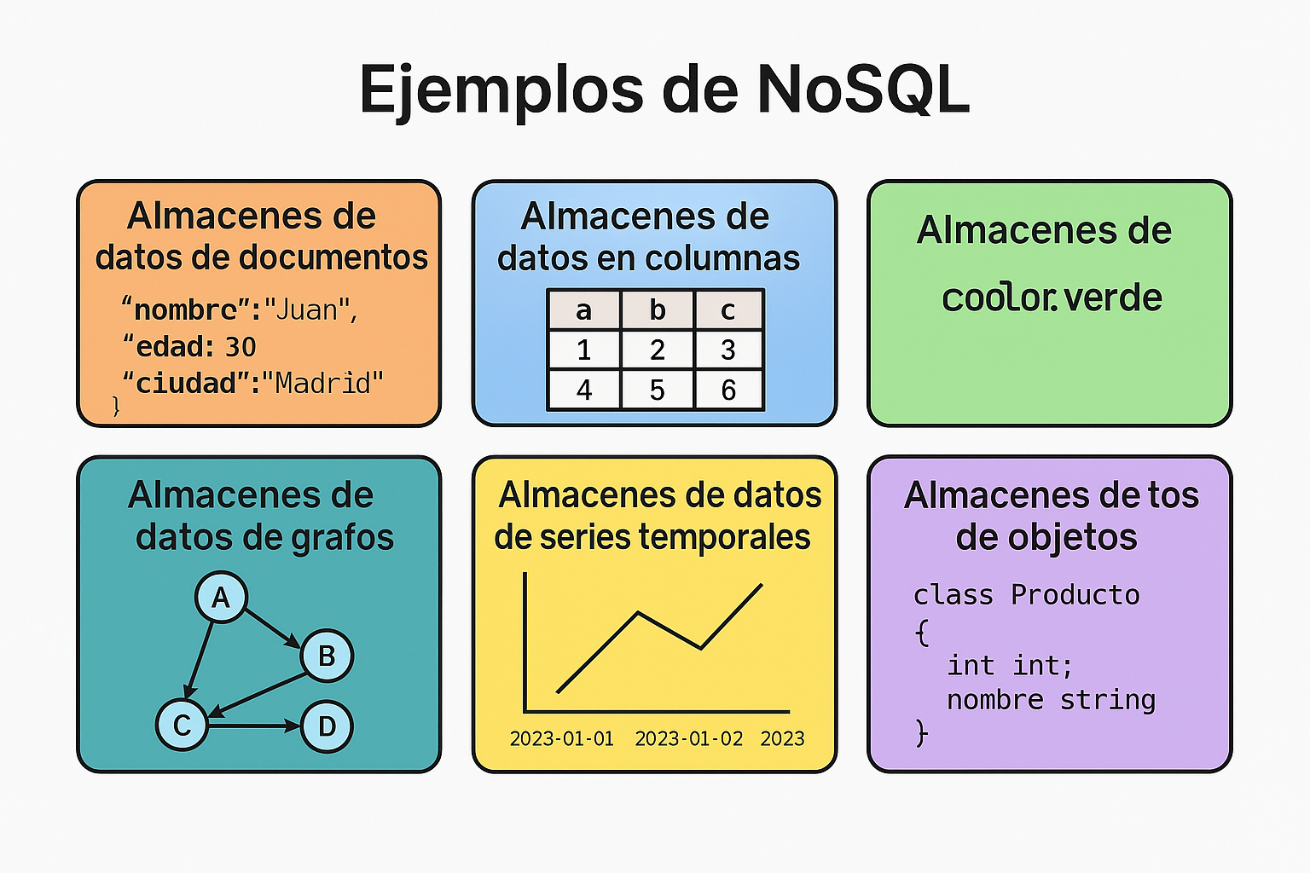
Estos almacenes no guardan los datos directamente, sino que mantienen índices para hacer más eficiente la búsqueda de datos en otros sistemas o archivos.

📦 Ejemplo: Un índice que apunta a la posición de documentos en un archivo grande:

"juan\_perez" → ubicación: byte 10523 del archivo X

🛠️ Ejemplo de base de datos: Apache Solr, Elasticsearch.





**1.3 Tipos de BD NoSQL**

* Documentales → MongoDB
* Columnas → Cassandra
* Clave/valor → Redis
* Grafos → Neo4j
* Series temporales → InfluxDB
* Objetos → MinIO
* Índice externo → ElasticSearch

📌 **Apoyate en las diapositivas de la presentación BD2\_01\_INTRO.pdf**​

**1.4 Relacional vs NoSQL**

|  |  |  |
| --- | --- | --- |
| **Característica** | **Relacional (SQL)** | **No Relacional (NoSQL)** |
| Estructura | Tablas | Documentos JSON |
| Relaciones | Fuertes | Débiles o embebidas |
| Escalabilidad | Vertical | Horizontal |
| Consulta | SQL | JSON/API |

**1.5 MongoDB y documentos JSON**

{

"titulo": "El aleph",

"autor": "Borges",

"editorial": ["Planeta", "Siglo XXI"],

"precio": 20,

"cantidad": 50

}

**1.5 Arquitectura de MongoDB**

* **Replica Set**: redundancia y alta disponibilidad.
* **Sharding**: escalamiento horizontal.
* **Mongos y Config Server**: distribuyen las consultas.

**🧾 2. MongoDB: una base de datos documental**

**¿Qué es MongoDB?**

* Es una base de datos **NoSQL documental**.
* Guarda datos en **documentos JSON**.
* Es muy flexible, ideal para proyectos con datos cambiantes.

**Ejemplo de documento JSON:**

{

"title": "The Big New Movie",

"year": 2015,

"info": {

"plot": "Nothing happens at all.",

"rating": 0

}

}

🔁 Operaciones CRUD en MongoDB:

* **Create**: insertar documentos
* **Read**: buscar documentos por campos
* **Update**: modificar valores dentro de los documentos
* **Delete**: eliminar documentos

👩‍🏫 **Explicación con ejemplos prácticos** como si estuviéramos guardando películas o productos en una tienda online.

📌 **Basado en BD2\_03\_MODELO-RELACIONAL-MONGODB.pdf**​

**🧠 4. Arquitectura de MongoDB**

**Conceptos clave:**

* **Replica Set**: Conjunto de servidores MongoDB (1 primario y varios secundarios).
* Si el primario falla, uno secundario toma su lugar.
* **Heartbeat** y **Oplog** permiten sincronizar los datos entre servidores.
* **Limitaciones**: RAM, CPU, Disco, Red.

**Escalamiento:**

* **Vertical**: mejoro una sola máquina (más RAM, CPU).
* **Horizontal (Sharding)**: agrego más máquinas.
* **Mongos**: el "enrutador" que envía las consultas a cada servidor.

📌 **Usá las presentaciones de BD2\_02\_ARQUITECTURA.pdf para mostrar estos conceptos gráficamente**​

**✍️ 5. Actividad sugerida**

**Objetivo:** Reforzar los conceptos con un caso práctico.

📌 Propuesta:

1. Dividí la clase en grupos.
2. Cada grupo debe elegir entre dos casos:
   * Crear una base relacional para una **biblioteca escolar**.
   * Crear una base NoSQL (MongoDB) para una **tienda online con productos variados**.
3. En 10-15 min arman en una hoja o presentación cómo sería su estructura de datos.
4. Exponen brevemente sus ideas al grupo.

**📄 Parte 2: Actividad sugerida (15-30 min)**

**Caso práctico:**

* Grupo 1: Diseñar base relacional para una biblioteca.
* Grupo 2: Diseñar base NoSQL para una app de películas.

**🔧 GUÍA DE COMANDOS MONGODB**

**🔢 En MongoDB Shell**

**⚙️ CREAR/SELECCIONAR BASE DE DATOS**

use base1 # crea o selecciona base

**📂 CREAR DOCUMENTO (INSERT)**

db.libros.insertOne({titulo:'El aleph', autor:'Borges'})

db.libros.insertMany([{titulo:'Libro A'},{titulo:'Libro B'}])

**🗑️ BORRAR DOCUMENTOS/COLECCIONES**

db.libros.deleteOne({\_id:1})

db.libros.deleteMany({})

db.libros.drop() # borra colección

db.dropDatabase() # borra BD

**🔍 LEER DOCUMENTOS (FIND)**

db.libros.find() # todos

db.libros.find({precio: {$gt: 50}}) # filtro

**✏️ ACTUALIZAR DOCUMENTOS**

db.libros.updateOne({\_id: 1}, {$set: {precio: 100}})

**db.libros.updateOne({\_id: ObjectId('68003fd7ad3591e6fdb71237')},** {$set: {precio: 100}})

db.libros.updateMany({}, {$set: {stock: 10}})

**🛠️ OTROS COMANDOS**

db.libros.find().pretty()

db.libros.find().sort({titulo: 1})

db.libros.find().limit(5).skip(2)

**🔹 En Compass (GUI de escritorio)**

1. **Conectar** con URL local: mongodb://localhost:27017
2. **Crear base y colección** desde interfaz.
3. **Insertar documento**: Botón "Insert Document"
4. **Buscar** con filtros tipo:

{ "autor": "Borges" }

1. **Actualizar** desde el editor de documentos.
2. **Borrar** desde el botón "Delete"

**☁️ En MongoDB Atlas (Cloud)**

1. **Crear cuenta** en https://www.mongodb.com/cloud/atlas
2. **Crear clúster gratuito**
3. **Conectarse con Compass/Shell**:
   * Te da un string como:

mongodb+srv://usuario:clave@cluster.mongodb.net/test

1. **Trabajar como en Compass**