MongoDB: Base de Datos NoSQL Orientada a Documentos

MongoDB es una base de datos NoSQL orientada a documentos que almacena datos en formato JSON (internamente como BSON). A diferencia de las bases de datos relacionales, MongoDB utiliza colecciones y documentos en lugar de tablas y filas, ofreciendo una estructura flexible, jerárquica y semiestructurada.

Esta presentación explorará las características fundamentales de MongoDB, su estructura, modelado de datos, operaciones CRUD y casos de uso prácticos, proporcionando una visión completa de esta potente herramienta para el desarrollo de aplicaciones modernas.

por Kibernum Capacitación S.A.

Preguntas de Activación de Contenido

1) ¿Qué diferencias recuerdas entre una base de datos relacional y una base NoSQL?
2) ¿Alguna vez has trabajado con estructuras de tipo JSON en tus desarrollos? ¿Cómo fue tu experiencia?
3) ¿Qué crees que significa que una base de datos esté "orientada a documentos"?

Características Principales de MongoDB

Modelo Flexible

Permite documentos con estructuras distintas dentro de la misma colección, adaptándose a datos cambiantes y con estructura no fija.

Escalabilidad Horizontal

Utiliza replicación y particionado (sharding) de manera sencilla, ideal para aplicaciones que requieren alta disponibilidad.

Alta Velocidad

Especialmente eficiente en operaciones de escritura, optimizando el rendimiento de aplicaciones con alta demanda.

Integración con JavaScript

Formato familiar para desarrolladores (JSON) y fácil integración con Node.js, ideal para aplicaciones web modernas.

Limitaciones de MongoDB

Menor Consistencia ACID

Menos rígido que PostgreSQL en términos de transacciones, aunque ha mejorado significativamente desde las versiones 4.x, ofreciendo mayor consistencia que otras bases NoSQL.

Curva de Aprendizaje

El diseño de colecciones puede resultar desafiante para desarrolladores provenientes del mundo SQL, requiriendo un cambio de mentalidad en el modelado de datos.

Redundancia de Datos

A menudo se duplican datos para optimizar lecturas, lo que puede aumentar el espacio de almacenamiento necesario y complicar las actualizaciones.

Falta de Integridad Referencial

No dispone de claves foráneas como en los sistemas de gestión de bases de datos relacionales, dificultando el mantenimiento de relaciones entre datos.

Casos de Uso Ideales

Aplicaciones Web

Perfecta para aplicaciones con estructuras de datos cambiantes que requieren adaptabilidad y evolución constante del esquema.

E-commerce

Ideal para catálogos de productos donde cada ítem puede tener atributos distintos (ropa, electrónica, libros) sin necesidad de cambiar el esquema global.

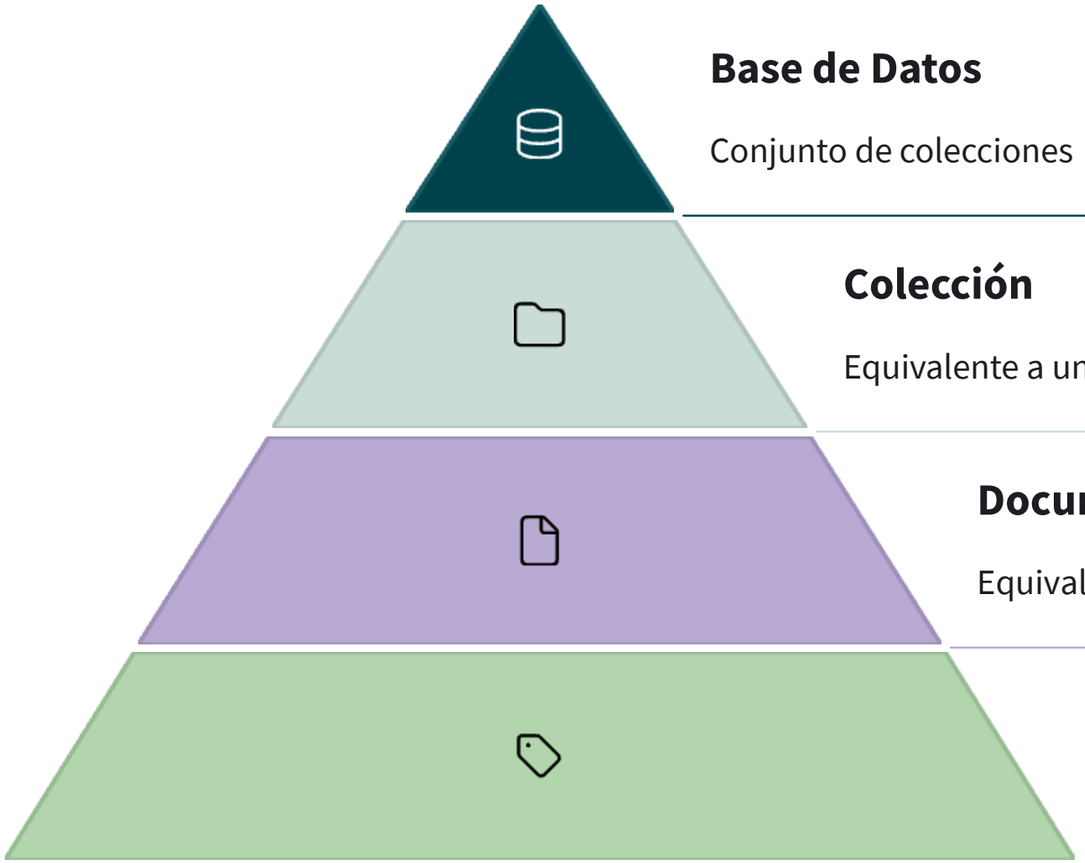
Apps Móviles

Excelente para gestionar perfiles de usuarios con diferentes características y preferencias que evolucionan con el tiempo.

Análisis de Datos

Eficiente para registro de eventos, logs y datos semiestructurados que requieren procesamiento y análisis rápido.

Estructura de MongoDB



Base de Datos

Conjunto de colecciones

Colección

Equivalente a una tabla en RDBMS

Documento

Equivalente a una fila, en formato JSON

Campo

Equivalente a una columna

MongoDB organiza la información en una jerarquía clara que facilita la gestión de datos. Cada base de datos contiene múltiples colecciones, que a su vez almacenan documentos individuales. Los documentos contienen campos que almacenan los valores específicos, todo ello en un formato JSON que resulta natural para los desarrolladores.

Ejemplo de Documento JSON en MongoDB

```
{
  "nombre": "Laura",
  "edad": 32,
  "email": "laura@gmail.com",
  "direccion": {
    "ciudad": "Santiago",
    "pais": "Chile"
  },
  "intereses": ["música", "viajes", "lectura"]
}
```

Campos Simples

Los campos como "nombre", "edad" y "email" almacenan valores directos de diferentes tipos (texto, número).

Subdocumentos

El campo "direccion" contiene un objeto anidado con sus propios campos (ciudad, país), creando una estructura jerárquica.

Arrays

El campo "intereses" almacena una lista de valores, permitiendo asociar múltiples elementos a un solo campo del documento.

Tipos de Datos en MongoDB

Tipo	Ejemplo
String	"nombre": "Miguel"
Number	"edad": 35
Boolean	"activo": true
Array	"cursos": ["BD", "Redes"]
Object	"direccion": {"ciudad": "Lima"}
Null	"telefono": null
Date	"fecha": ISODate("2024-12-01")
ObjectId	_id: ObjectId("...")

Operaciones CRUD: CREATE

insertOne()

```
db.estudiantes.insertOne({
  nombre: "Ana",
  edad: 28,
  carrera: "Ingeniería"
})
```

Agrega un solo documento a la colección especificada. Útil para insertar registros individuales.

insertMany()

```
db.estudiantes.insertMany([
  { nombre: "Pedro", edad: 30 },
  { nombre: "Carla", edad: 24 }
])
```

Inserta múltiples documentos en una sola operación, mejorando el rendimiento cuando se necesita agregar varios registros simultáneamente.

Operaciones CRUD: READ

```
// Todos los documentos
db.estudiantes.find();

// Filtrar por condición
db.estudiantes.find({ edad: { $gt: 25 } });

// Buscar uno solo
db.estudiantes.findOne({ nombre: "Ana" });
```

Consulta Simple

db.estudiantes.find() - Recupera todos los documentos de la colección sin aplicar filtros.

Consulta con Filtros

db.estudiantes.find({ edad: { $gt: 25 } }) - Filtra documentos donde la edad es mayor a 25 usando el operador $gt.

Búsqueda de Documento Único

db.estudiantes.findOne({ nombre: "Ana" }) - Devuelve solo el primer documento que coincide con el criterio especificado.

Operaciones CRUD: UPDATE

updateOne()

```
db.estudiantes.updateOne(
  { nombre: "Ana" },
  { $set: { edad: 29 } }
)
```

Actualiza un solo documento que coincida con el filtro especificado. El operador $set modifica los campos indicados sin afectar al resto del documento.

updateMany()

```
db.estudiantes.updateMany(
  {},
  { $set: { activo: true } }
)
```

Actualiza todos los documentos que coincidan con el filtro. En este ejemplo, se agrega el campo "activo" con valor true a todos los documentos de la colección.

Operaciones CRUD: DELETE

```
db.estudiantes.deleteOne({ nombre: "Ana" });
db.estudiantes.deleteMany({ edad: { $lt: 25 } });
```

deleteOne()

Elimina el primer documento que coincide con el filtro

deleteMany()

Elimina todos los documentos que coinciden con el filtro

Precaución

Las operaciones de eliminación son permanentes

Las operaciones de eliminación en MongoDB permiten eliminar documentos de forma selectiva o masiva según criterios específicos. Es importante utilizar estas operaciones con precaución, ya que los datos eliminados no pueden recuperarse a menos que se disponga de una copia de seguridad.

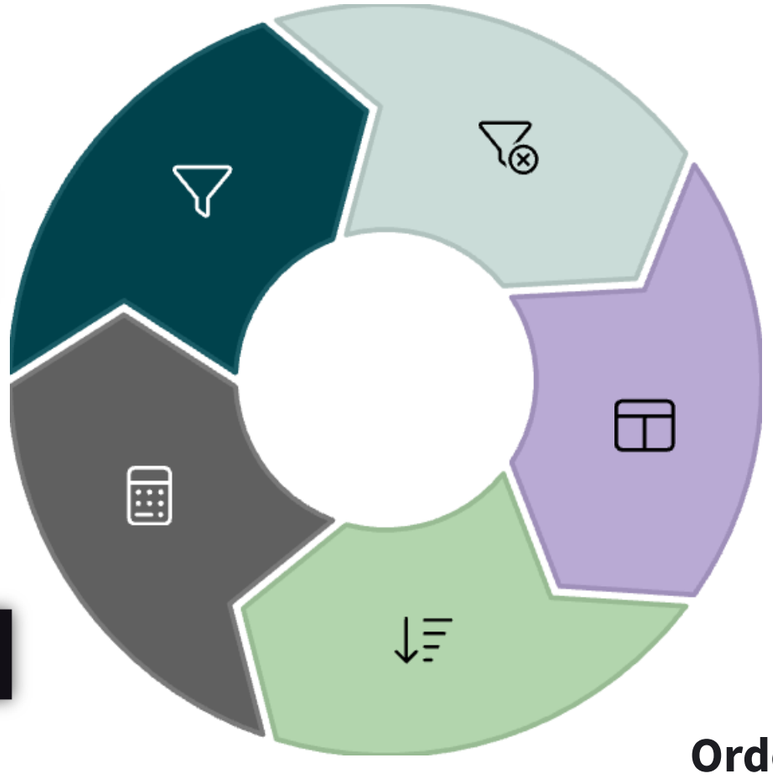
Operadores de Query en MongoDB

Operador	Función	Ejemplo
$gt	Mayor que	{ edad: { $gt: 30 } }
$lt	Menor que	{ edad: { $lt: 30 } }
$eq	Igual	{ nombre: { $eq: "Ana" } }
$in	Dentro de un array	{ nombre: { $in: ["Ana", "Pedro"] } }
$and	Y lógico	{ $and: [condición1, condición2] }
$or	O lógico	{ $or: [{edad: 25}, {nombre: "Carla"}] }
$exists	Campo existe	{ email: { $exists: true } }
$regex	Coincidencia con expresión regular	{ nombre: { $regex: "^M" } }

Tipos de Queries Avanzadas

Búsqueda Simple

```
db.estudiantes.find({ nombre: "Pedro" });
```



Múltiples Condiciones

```
db.estudiantes.find({ edad: { $gte: 25, $lte: 30 } });
```

Proyección de Campos

```
// Solo muestra el nombre y edad, excluyendo _id
db.estudiantes.find(
  { edad: { $gt: 25 } },
  { _id: 0, nombre: 1, edad: 1 }
);
```

Contar Resultados

```
db.estudiantes.countDocuments({ edad: { $gt: 25 } });
```

Ordenar Resultados

```
db.estudiantes.find().sort({ edad: -1 }); // descendente
```

Recomendaciones de modelado

- ✓ **Embebe documentos** si los datos están fuertemente relacionados (ej: dirección dentro de usuario).
- ✓ **Referencias manuales** si los datos son reutilizables o muy grandes (ej: usuario y sus cursos).
- ✓ Modela tus colecciones según **las consultas más frecuentes**, no pensando como SQL.
- ✓ Aprovecha los índices en campos consultados frecuentemente.

Tabla resumen

Acción	Método
Insertar	insertOne, insertMany
Leer	find, findOne
Actualizar	updateOne, updateMany
Eliminar	deleteOne, deleteMany
Operadores	$gt, $lt, $in, $regex, $and, $or

Actividad Práctica: Guiada: Registro y Gestión de Estudiantes en MongoDB

Objetivo

Aplicar los conocimientos adquiridos sobre MongoDB para diseñar, insertar, consultar, actualizar y eliminar documentos en una colección que almacene datos de estudiantes, utilizando el modelo documental y operaciones CRUD con Mongo Shell o una herramienta como Compass.

Paso a Paso Detallado

1 Crear una base de datos y una colección

Tarea: Abre el shell de MongoDB e ingresa los siguientes comandos:

```
use academia
db.createCollection("estudiantes")
```

Actividad Práctica: Guiada: Registro y Gestión de Estudiantes en MongoDB

2 Insertar documentos de ejemplo

📌 **Tarea:** Inserta tres documentos en la colección estudiantes con los siguientes datos:

•Campos requeridos:

•nombre (string)

•edad (número)

•carrera (string)

•activo (booleano)

•cursos (array de strings)

•direccion (subdocumento con ciudad y país)

```
db.estudiantes.insertMany([
  {
    nombre: "Ana",
    edad: 22,
    carrera: "Ingeniería",
    activo: true,
    cursos: ["Bases de Datos", "Redes"],
    direccion: { ciudad: "Santiago", pais: "Chile" }
  },
  {
    nombre: "Luis",
    edad: 25,
    carrera: "Diseño",
    activo: false,
    cursos: ["UX", "HTML"],
    direccion: { ciudad: "Lima", pais: "Perú" }
  },
  {
    nombre: "Valeria",
    edad: 24,
    carrera: "Informática",
    activo: true,
    cursos: ["JavaScript", "MongoDB"],
    direccion: { ciudad: "Quito", pais: "Ecuador" }
  }
])
```

Actividad Práctica: Guiada: Registro y Gestión de Estudiantes en MongoDB

3 Realizar consultas básicas

Tarea: Ejecuta las siguientes búsquedas y analiza los resultados:

```
// Ver todos los estudiantes
db.estudiantes.find()

// Filtrar por estudiantes activos
db.estudiantes.find({ activo: true })

// Buscar por país
db.estudiantes.find({ "direccion.pais": "Perú" })

// Buscar estudiantes que tengan el curso "MongoDB"
db.estudiantes.find({ cursos: "MongoDB" })
```

Actividad Práctica: Guiada: Registro y Gestión de Estudiantes en MongoDB

4 Actualizar documentos

Tarea: Realiza estas dos actualizaciones:

```
// Cambiar la edad de Ana a 23
db.estudiantes.updateOne({ nombre: "Ana" }, { $set: { edad: 23 } })

// Marcar todos los estudiantes como "activo: true"
db.estudiantes.updateMany({}, { $set: { activo: true } })
```

Actividad Práctica: Guiada: Registro y Gestión de Estudiantes en MongoDB

5 Eliminar documentos

Tarea: Realiza estas eliminaciones:

```
// Eliminar al estudiante llamado Luis
db.estudiantes.deleteOne({ nombre: "Luis" })

// Eliminar a todos los estudiantes menores de 23 años
db.estudiantes.deleteMany({ edad: { $lt: 23 } })
```

Resultado Esperado

- Uso correcto de comandos CRUD en MongoDB.
- Comprensión del modelo documental, arrays y subdocumentos.
- Capacidad para filtrar, actualizar y eliminar registros en una colección.

Enlace a Material Complementario

📺 **Video en español – MongoDB desde cero (curso para principiantes)**
🔗 https://www.youtube.com/watch?v=DPdAfgmkNuE
Este video explica qué es MongoDB, cómo funciona su modelo orientado a documentos y cómo realizar operaciones básicas paso a paso.

📘 **Documentación oficial de MongoDB (en español):**
🔗 https://www.mongodb.com/docs/manual/?lang=es

Preguntas de Reflexión Final

1) ¿Qué ventajas observaste al trabajar con documentos en formato JSON dentro de MongoDB?
2) ¿Qué precauciones crees que debes tener al diseñar colecciones que contienen subdocumentos y arrays?
3) ¿Cómo influye el uso de operadores como $gt, $in, $regex, etc., en la forma de consultar información en MongoDB?