Base de Datos MongoDB

Lectura sesión Base de Datos MongoDB

por Kibernum Capacitación S.A.

1. Introducción a MongoDB

MongoDB es una base de datos NoSQL orientada a documentos, en la que los datos se almacenan en formato JSON (internamente como BSON), lo que permite una estructura flexible, jerárquica y semiestructurada.
A diferencia de las bases de datos relacionales, no usa tablas ni filas, sino colecciones y documentos.

MongoDB fue diseñado para manejar datos:

- Que cambian con frecuencia.
- Que tienen una estructura no fija.
- Que requieren alta disponibilidad y escalabilidad horizontal.

Recuerda que, para avanzar, debes tener instalado MongoDB. Para ello revisa las instalaciones previas desde la plataforma o sigue las siguientes instrucciones.

Instalaciones previas

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

Ventajas de MongoDB

Ventaja	Descripción
Modelo flexible	Permite documentos con estructuras distintas dentro de la misma colección.
Escalabilidad horizontal	Usa replicación y particionado (sharding) fácilmente.
Alta velocidad	Especialmente para escritura.
Formato familiar	Los documentos se ven como objetos JSON, muy utilizados por desarrolladores frontend/backend.
Fácil integración con Node.js	Ideal para aplicaciones JavaScript.

Desventajas de MongoDB

Desventaja	Descripción
Menor consistencia ACID	Menos rígido que PostgreSQL, aunque ha mejorado desde versiones 4.x.
Curva de aprendizaje en modelado	El diseño de colecciones puede ser desafiante si vienes del mundo SQL.
Redundancia de datos	A menudo se duplican datos para optimizar lecturas.
Falta de integridad referencial directa	No hay claves foráneas como en los RDBMS.

Caso de uso: Base de datos Document-Oriented

MongoDB es ideal cuando necesitas:

- Aplicaciones web con estructuras cambiantes.
- Catálogos de productos (e-commerce).
- Perfiles de usuarios en apps móviles.
- Registro de eventos, logs, datos semiestructurados.

Por ejemplo: En un e-commerce, cada producto puede tener atributos distintos (ropa, electrónica, libros), y MongoDB permite manejar esas diferencias sin necesidad de cambiar el esquema global.

2. Estructura y Modelado de Datos

¿Cómo se estructura MongoDB?

- Base de datos (Database): Conjunto de colecciones.
- Colección (Collection): Equivalente a una tabla en RDBMS.
- Documento (Document): Equivalente a una fila, pero en formato JSON.
- Campo (Field): Equivalente a una columna.

Ejemplo de documento en MongoDB

Este es un documento JSON utilizado en MongoDB para representar la información de un usuario. MongoDB trabaja con documentos estructurados en formato BSON (una versión binaria de JSON), pero visual y programáticamente se escribe así.

Detalle de cada campo

Campo	Tipo de dato	Descripción
nombre	String	Contiene el nombre de la persona. En este caso, "Laura".
edad	Número (int)	Representa la edad exacta de la persona: 32.
email	String	Guarda el correo electrónico de la persona: "laura@gmail.com".
direccion	Documento (object)	Es un subdocumento con dos claves: ciudad y pais, que indican la ubicación.
intereses	Array de strings	Es una lista de intereses: "música", "viajes" y "lectura".

Crear base de datos y colección

En la terminal interactiva mongo:

```
// Selecciona o crea base de datos
use academia;

// Crea una colección (implícitamente al insertar el primer documento)
db.estudiantes.insertOne({
  nombre: "Miguel",
  carrera: "Informática",
  año: 2025
});
```

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("...")

Descripción de los tipos de datos

- **String**
  Es el tipo de dato más usado. Se utiliza para almacenar texto, como nombres, descripciones o correos electrónicos.
  Ejemplo: "nombre": "Miguel"
  El valor "Miguel" es una cadena de caracteres (texto).
- **Number**
  MongoDB permite representar números enteros y flotantes. Este tipo se usa para almacenar edades, precios, puntuaciones, entre otros.
  Ejemplo: "edad": 35
  El valor 35 es un número entero.
- **Boolean**
  Representa un valor lógico: true o false. Es útil para indicar estados como "activo", "verificado", o "disponible".
  Ejemplo: "activo": true
  Esto indica que el usuario está activo.
- **Array**
  Se utiliza para almacenar múltiples valores en una misma clave. Los elementos pueden ser del mismo tipo o variados, incluyendo otros objetos.
  Ejemplo: "cursos": ["BD", "Redes"]
  Aquí se está indicando que el usuario está inscrito en los cursos de "BD" y "Redes".
- **Object (Subdocumento)**
  Permite anidar estructuras dentro del documento principal, creando una jerarquía de datos. Esto es útil para representar direcciones, perfiles o cualquier agrupación de datos.
  Ejemplo: "direccion": {"ciudad": "Lima"}
  Este campo incluye un subdocumento con la clave "ciudad" y el valor "Lima".
- **Null**
  Se utiliza cuando no hay un valor asignado o se quiere indicar explícitamente que un campo está vacío.
  Ejemplo: "telefono": null
  Significa que no hay un número de teléfono disponible.
- **Date**
  MongoDB permite trabajar con fechas y horas mediante el tipo ISODate. Se utiliza para registrar momentos específicos como fechas de creación, actualización o vencimiento.
  Ejemplo: "fecha": ISODate("2024-12-01")
  Este valor representa una fecha exacta en formato estándar.
- **ObjectId**
  Es un identificador único que MongoDB genera automáticamente para cada documento. Sirve como clave primaria (_id) y garantiza que cada documento pueda ser identificado de manera única.
  Ejemplo: "_id": ObjectId("...")
  Aunque aparece como una cadena, internamente es un objeto binario de 12 bytes.

3. Manipulación y Consultas en MongoDB

CRUD en MongoDB

- CREATE: Insertar documentos
- READ: Consultar documentos
- UPDATE: Modificar datos
- DELETE: Eliminar documentos

CREATE: Insertar documentos

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


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

Este código se muestra dos comandos de inserción de documentos en MongoDB:

- insertOne() agrega un solo documento a la colección estudiantes. En el ejemplo, se inserta a Ana con edad 28 y carrera “Ingeniería”.
- insertMany() agrega varios documentos a la vez. Aquí se insertan los datos de Pedro (30 años) y Carla (24 años).

Ambos comandos se usan para registrar nuevos datos en una colección de forma estructurada y eficiente.

READ: Consultar documentos

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

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

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

En este código se muestran tres formas de consultar documentos en la colección estudiantes de MongoDB:

- db.estudiantes.find(); Recupera todos los documentos de la colección.
- db.estudiantes.find({ edad: { $gt: 25 } }); Filtra los documentos donde la edad es mayor a 25 usando el operador $gt (greater than).
- db.estudiantes.findOne({ nombre: "Ana" }); Busca y devuelve el primer documento que tenga el nombre "Ana".

Estas consultas permiten explorar los datos almacenados aplicando filtros o trayendo resultados específicos.

UPDATE: Modificar datos

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

// Agregar un campo nuevo
db.estudiantes.updateMany(
  {},
  { $set: { activo: true } }
);
```

En este código se muestran dos formas de actualizar documentos en la colección estudiantes de MongoDB:

- db.estudiantes.updateOne({ nombre: "Ana" }, { $set: { edad: 29 } }); Actualiza un solo documento donde el nombre sea "Ana", cambiando su edad a 29.
- db.estudiantes.updateMany({}, { $set: { activo: true } }); Agrega el campo activo con valor true a todos los documentos de la colección.

Ambos comandos utilizan el operador $set para establecer o modificar valores dentro de los documentos.

DELETE: Eliminar documentos

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

En este código se muestran dos formas de eliminar documentos en la colección estudiantes de MongoDB:

- db.estudiantes.deleteOne({ nombre: "Ana" }); Elimina el primer documento que tenga el nombre "Ana".
- db.estudiantes.deleteMany({ edad: { $lt: 25 } }); Elimina todos los documentos donde la edad sea menor que 25, usando el operador $lt (less than).

Estas operaciones permiten gestionar la limpieza y mantenimiento de los datos almacenados en una colección.

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 en MongoDB

- 🔸 Búsqueda simple:

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

- 🔸 Búsqueda por 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 }
);
```

- 🔸 Ordenar resultados:

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

🔸 Contar resultados:

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

Recomendaciones de modelado y tabla resumen

- 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.

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

Conclusión y Actividad Práctica Guiada

Conclusión

MongoDB es una base de datos moderna, ideal para manejar datos en aplicaciones ágiles y escalables. Su enfoque orientado a documentos permite desarrollar sin esquemas rígidos, adaptándose a necesidades cambiantes.

Es especialmente valiosa para desarrolladores web y móviles, donde JSON ya es formato nativo de trabajo.

El dominio de MongoDB es fundamental en proyectos donde la velocidad, la flexibilidad y la facilidad de escalar son más importantes que la estructura rígida o relaciones complejas entre tablas.

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")
```

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" }
  }
])
```

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" })
```

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 } })
```

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.