Clase 11: MongoDB colecciones y registros

Sitio:	Centro de E-Learning - UTN.BA	Imprimido por:	Nelson Brian Avila Solano
Curso:	Curso de Backend Developer - Turno Noche	Día:	Tuesday, 9 de December de 2025, 19:52
Libro:	Clase 11: MongoDB colecciones y registros		

Descripción

Objetivos

- Entender la importancia de los índices en MongoDB y cómo utilizar diferentes tipos de índices para mejorar la eficiencia de las consultas.
- Aprender a crear índices en MongoDB utilizando tanto la consola como Mongo Compass.
- Utilizar proyecciones en MongoDB para obtener diferentes vistas de los datos en una colección.
- Hacer uso de agregaciones en MongoDB para obtener información útil de los datos almacenados, como el promedio de los precios de los productos, la cantidad total de productos en stock, los clientes que han realizado más pedidos, etc.

Tabla de contenidos

1. Introducción

Node.js es un entorno de ejecución de JavaScript que permite desarrollar aplicaciones del lado del servidor de manera eficiente y escalable. Su modelo basado en eventos y operaciones asincrónicas lo hace ideal para manejar múltiples conexiones simultáneamente. Además, cuenta con npm, el gestor de paquetes más grande del mundo, que facilita la instalación y gestión de librerías para distintos proyectos.

En esta unidad, exploraremos los conceptos básicos de Node.js, desde su instalación y uso en la terminal hasta la creación de servidores y la gestión de dependencias con npm. También introduciremos Express, un framework que simplifica el desarrollo de aplicaciones web, y Nodemon, una herramienta útil para agilizar el proceso de desarrollo.

2. MONGO

MONGO

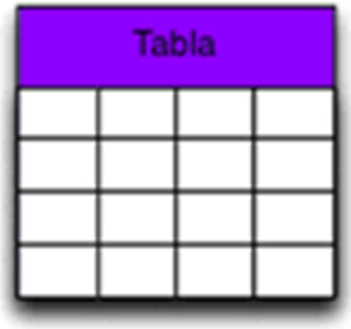
MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto.

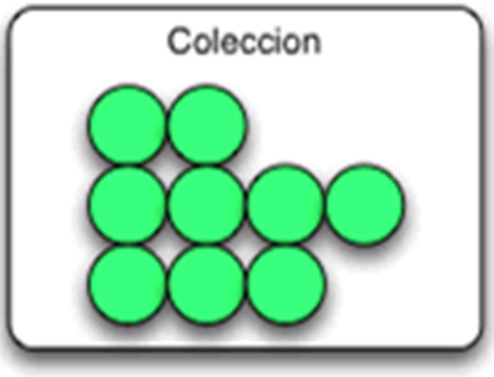
En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

MongoDB y Colecciones

Una colección en MongoDB es muy similar a una Tabla de una base de datos. La tabla almacena registros (filas) mientras que las colecciones almacenan documentos.

Base de datos y colecciones



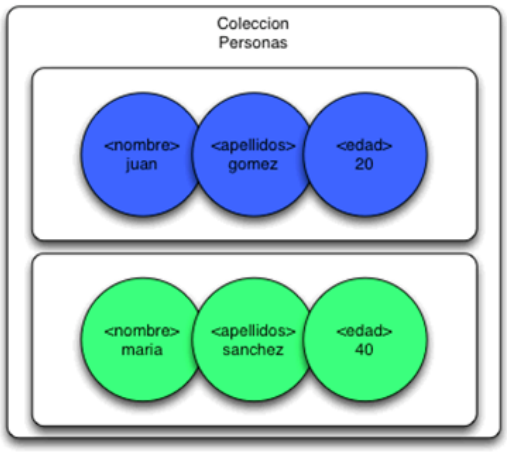


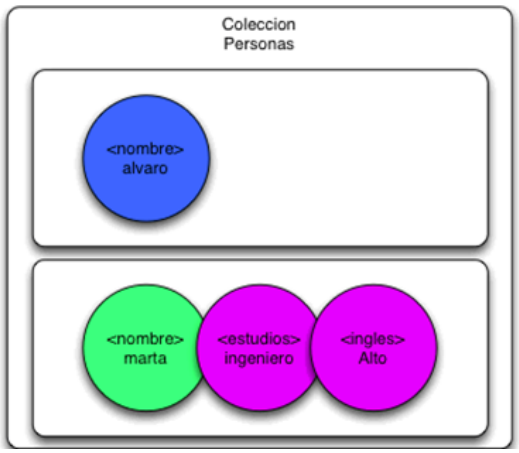
MongoDB y Documentos

Un documento está compuesto por claves y valores (key,value) y cada documento puede tener variaciones importantes con el anterior dentro de una colección.

nombre	apellidos	edad
juan	gomez	20
maria	sanchez	40

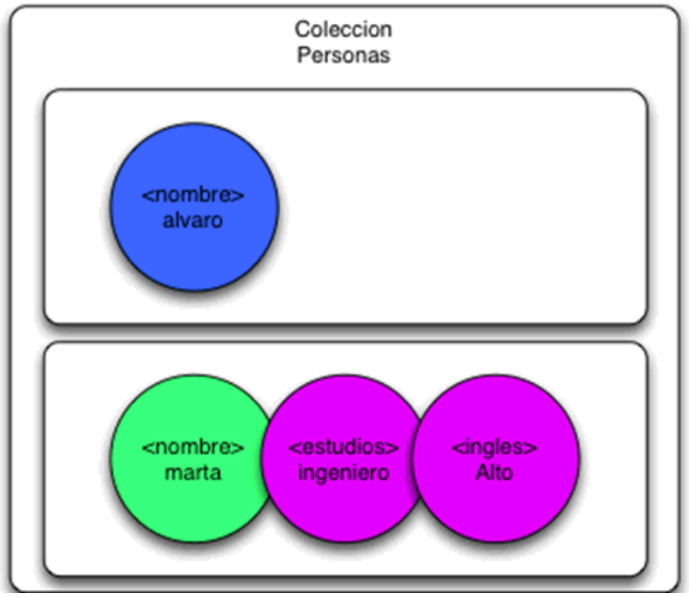
Relacional





NO-Relacional

El primer documento solo dispone de un campo . Mientras el segundo dispone de tres pero estos son diferentes a los anteriores. Esto es lo que en NoSQL se denomina Schema Free o libre y que aporta una gran flexibilidad a la hora de trabajar. Aunque también puede acabar siendo un poco caótico



NO-Relacional

3. Índices, agregaciones y proyecciones con MongoDB

Índices, agregaciones y proyecciones con MongoDB

MongoDB y Documentos

Las colecciones pueden tener índices que nos permiten buscar y ordenar los documentos de manera más eficiente. Para crear un índice en una colección, podemos usar el comando createIndex. Por ejemplo, para crear un índice en la colección "mi_coleccion" para el campo "edad", podemos usar el siguiente comando:

```
db.mi_coleccion.createIndex({ "edad": 1 })
```

Índice

Esto creará un índice ascendente en el campo "edad" de la colección "mi_coleccion". Ahora, cuando realizamos una consulta en esta colección que involucra el campo "edad", MongoDB utilizará este índice para acelerar la búsqueda.

Agregaciones

Las agregaciones son una herramienta poderosa para procesar y transformar datos en MongoDB. En lugar de buscar documentos individuales en una colección, las agregaciones permiten trabajar con conjuntos de documentos y realizar operaciones complejas, como la combinación de datos, la agregación y la filtración de datos.

También podemos utilizar agregaciones para realizar operaciones de consulta más complejas. Las agregaciones son una forma flexible y poderosa de procesar los datos en MongoDB. Por ejemplo, para obtener el promedio de edad de todos los documentos en la colección "mi_coleccion", podemos usar la siguiente agregación:

```
db.mi_coleccion.aggregate([
  { $group: { _id: null, avgAge: { $avg: "$edad" } } }
])
```

Este comando nos devolverá un documento con el promedio de edad de todos los documentos en la colección "mi_coleccion".

```
db.personajes.aggregate([
  { $group: { _id: null, avgAge: { $avg: "$edad" } } }
])
```

$avg: operador que calcula el promedio

Este comando nos devolverá un documento con el promedio de edad de todos los documentos en la colección "mi_coleccion".

```
db.personajes.aggregate([
  { $group: { _id: null, maxAge: { $max: "$edad" } } }
])
```

```
db.personajes.aggregate([
{ $group: { _id: null, maxAge: { $max: "$edad" } } },
{ $lookup: { from: “personajes”, localField: “maxAge”, foreingFiel:
“edad”, as: “personaje” }},
{ $project: {_id: 0, “personaje.nombre”: 1, maxAge: 1 }}
])
```

Operadores

$group: agrupa documento de una colección

$lookup: busca elementos de la misma colección (personajes)

$project: campos que queremos mantener

Operadores de comparación

● $cmp: compara dos valores y devuelve un número entero como resultado. Devuelve -1 si el primer valor es menor que el segundo, 0 si son iguales y 1 si el primer valor es mayor que el segundo.

● $eq: compara dos valores y devuelve true si son equivalentes.

● $gt: compara dos valores y devuelve true si el primero es más grande que el segundo.

● $gte: compara dos valores y devuelve true si el primero es igual o más grande que el segundo.

● $lt: compara dos valores y devuelve true si el primero es menor que el segundo.

● $lte: compara dos valores y devuelve true si el primero es igual o menor que el segundo.

● $ne: compara dos valores y devuelve true si los valores no son equivalentes

Operadores de lógicos

● $and

● $or

● $nor ● $not

Operadores de matrices

● $size

● $all

● $elemMatch

● $not

Proyecciones

Las proyecciones se utilizan para seleccionar y devolver un subconjunto de campos de los documentos en una colección. Esto puede ser útil para reducir el tamaño de los resultados de la consulta y hacer que las consultas sean más eficientes.

Las proyecciones se especifican como el segundo argumento de los métodos find(), findOne() y aggregate(), y pueden incluir o excluir campos específicos.

Por ejemplo, supongamos que tenemos la colección "Estudiantes" que contiene documentos que representan a los estudiantes de Hogwarts. Cada documento tiene campos como "nombre", "casa", "edad", "género", entre otros. Si queremos obtener solo los nombres y las casas de todos los estudiantes, podemos usar una proyección para excluir los demás campos. Para ello, podemos usar el método find() junto con el operador de proyección "$project":

```
db.personajes.find(
  {},   { _id: 0, nombre: 1,
casa: 1 }
)
```

En este ejemplo, el primer argumento de find() especifica un filtro vacío, lo que significa que se devolverán todos los documentos de la colección. El segundo argumento de find() especifica una proyección que excluye el campo "_id" y devuelve solo los campos "nombre" y "casa".

Podemos también utilizar operadores de agregación para proyectar información más compleja. Por ejemplo, si queremos obtener el nombre de los estudiantes y el número de hechizos que han realizado, podemos utilizar una consulta con agregación junto con el operador $project y $size:

```
db.Estudiantes.aggregate([   {
        $project: {       _id: 0,       nombre: 1,
    hechizos_realizados: { $size: "$hechizos" }
    }
      }
    ])
```

La consulta con agregación utiliza el operador $project para excluir el campo "_id" y devolver solo los campos "nombre" y "hechizos_realizados". El campo "hechizos_realizados" se calcula utilizando el operador $size, que devuelve el número de elementos en un array.

4. Índices, agregaciones y proyecciones MongoDB Compass

Índices, agregaciones y proyecciones MongoDB Compass

MongoDB Compass

En MongoDB Compass, podemos realizar estas operaciones de manera visual e interactiva, sin tener que escribir comandos manualmente.

Paso a paso: Crear un índice

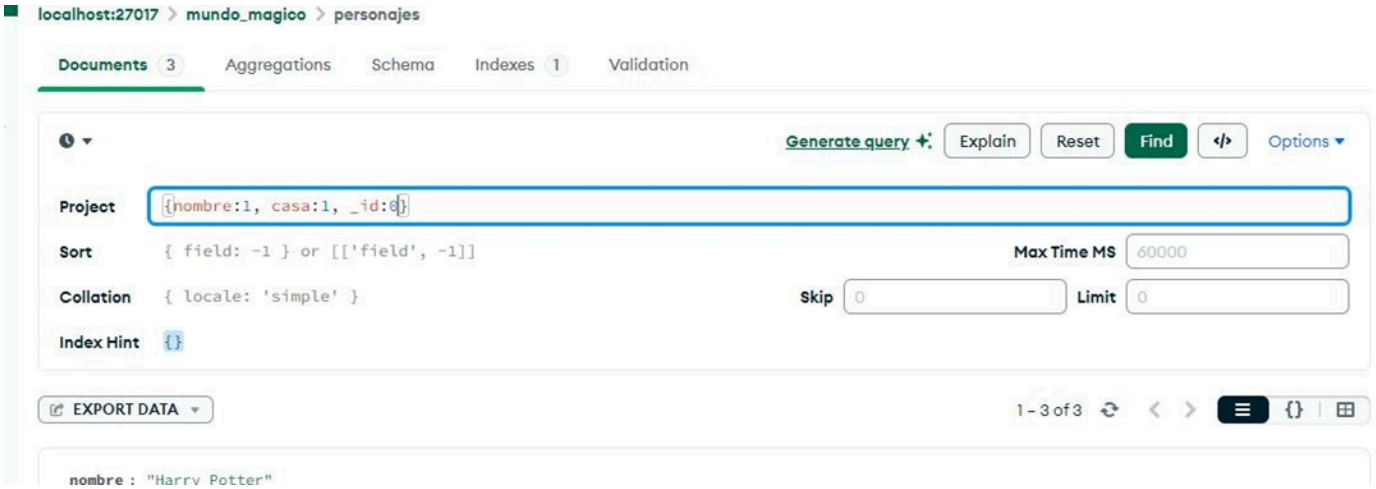
Crear un índice:

1. Abre MongoDB Compass y selecciona la base de datos y la colección en la que deseas crear el índice.
2. Haz clic en la pestaña "**Indexes**" en la parte superior de la pantalla.
3. Haz clic en el botón "**Create Index**" en la parte inferior de la pantalla.
4. En el campo "**Field**", ingresa el nombre del campo para el que deseas crear el índice (por ejemplo, "edad").
5. En el campo "**Order**", selecciona "*Ascending*" para crear un índice ascendente o "*Descending*" para crear un índice descendente.
6. Haz clic en "**Create**" para crear el índice.

Proyecciones

Realizar una Proyección:

1. Haz clic en la opción "Options" en la barra de herramientas superior.



{ nombre: 1, casa: 1, _id: 0 }: Esto especifica qué campos incluir (nombre y casa) y cuáles excluir (_id).

Agregación

Realizar una agregación:

1. Haz clic en la pestaña "*Aggregations*" en la parte superior de la pantalla.
2. Haz clic en el botón "*Add Stage*" para agregar una etapa a la agregación.
3. Selecciona la etapa que deseas agregar (por ejemplo, "$group" para agrupar los documentos por un campo).
4. En la sección de configuración de la etapa, ingresa la información necesaria para la operación (por ejemplo, el nombre del campo por el que deseas agrupar).
5. Haz clic en "Apply" para aplicar la etapa a la agregación.
6. Continúa agregando etapas según sea necesario para completar la agregación.
7. Haz clic en "Run" para ejecutar la agregación.

Realizar una agregación:

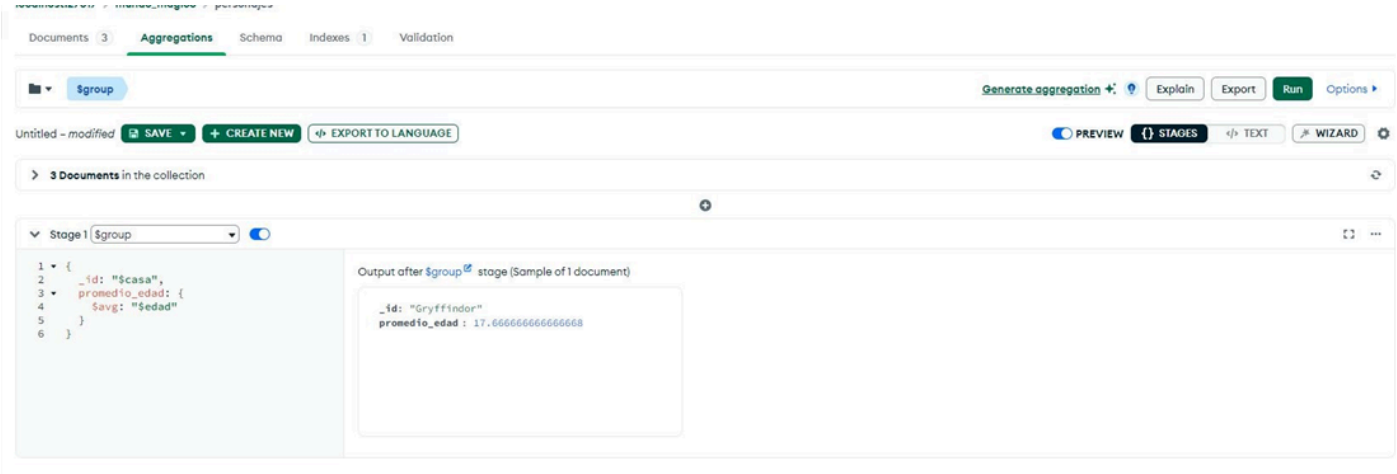
1. Haz clic en la pestaña "***Aggregations***" en la parte superior de la pantalla.
2. Haz clic en el botón "***Add Stage***" para agregar una etapa a la agregación.

3. Selecciona la etapa que deseas agregar (por ejemplo, "$group" para agrupar los documentos por un campo).

4. En la sección de configuración de la etapa, ingresa la información necesaria para la operación (por ejemplo, el nombre del campo por el que deseas agrupar).

5. Haz clic en "**Apply**" para aplicar la etapa a la agregación.

6. Haz clic en "**Run**" para ejecutar la agregación.



$group agrupa los documentos por el campo casa.

"$avg" calcula el promedio de la edad ($edad) para cada grupo de casa.

```
{
 _id: null,
 suma_edades: {
  $sum: "$edad"
 }
}
```

$group agrupa los documentos por el campo casa.

"$sum" suma el campo de edad ($edad) para cada grupo de casa.

```
{
      "_id": "$casa",
      "promedio_edad": { "$avg": "$edad" },
      "suma_edad": { "$sum": "$edad" },
      "numero_hechizos": { "$sum": { "$size": "$hechizos" } }
    }
```

- **_id:** "$casa": Esto agrupa los documentos por el campo casa, es decir, todos los documentos que tienen el mismo valor para casa se agruparán juntos.
- **promedio_edad**: *{ "$avg": "$edad" }:* Utiliza el operador **$avg** para calcular el promedio de los valores en el campo edad para cada grupo (casa). $edad hace referencia al campo edad de cada documento dentro del grupo.
- suma_edad: *{ "$sum": "$edad" }:* Utiliza el operador **$sum** para sumar los valores en el campo edad para cada grupo (casa). $edad también hace referencia al campo edad de cada documento dentro del grupo.
- numero_hechizos: *{ "$sum": { "$size": "$hechizos" } }:* Utiliza el operador **$size** dentro de **$sum** para contar el número de elementos en el campo hechizos para cada documento dentro del grupo (casa). $hechizos es un array, y $size devuelve el número de elementos en ese array.

5. Ejemplo de la clase

Ejemplo de la clase

Base de datos: Clientes - usuarios

Práctica 1

1: Crear y poblar una base de datos

Crea una base de datos llamada tienda.

Dentro de tienda, crea una colección llamada productos.

Inserta al menos 5 documentos en productos con la siguiente estructura:

{

"nombre": "Producto X",

"precio": 100.50,

"categoria": "",

"stock": 20

}

Varía los valores de los campos nombre, precio, categoría, y stock.

2: Consultas

Realiza una consulta que muestre todos los productos con un precio menor a $50.000 Realiza una consulta para contar cuántos productos tienen un stock mayor a 20.

Listar Productos por Categoría y contar cuántos productos hay en cada categoría.

Calcular el precio promedio de los productos por categoría.

Práctica 2

Supongamos que tienes una base de datos que almacena información de ventas para una tienda. Cada documento en la colección "ventas" tiene los siguientes campos:

▪ fecha (string): la fecha de la venta en formato "YYYY-MM-DD".

▪ producto (string): el nombre del producto vendido.

▪ cantidad (number): la cantidad de unidades vendidas.

▪ precio_unitario (number): el precio unitario del producto.

▪ total (number): el total de la venta (cantidad x precio_unitario).

▪ categoria (string): la categoría del producto.

▪ vendedor (string): el nombre del vendedor que realizó la venta.

}

Tu tarea es:

▪ realizar una agregación que calcule el total de ventas por día para un rango de fechas específico y devuelva los resultados ordenados por fecha en orden ascendente.

▪ realizar una agregación que: calcule el total de ventas por día y por vendedor, calcule el total de ventas por categoría de producto y devuelva el total de unidades vendidas por día para cada producto.

▪ devuelva los resultados en un formato que incluya: fecha, total de ventas, total de unidades vendidas, vendedor y categoría