TEMA 13: Explotación de bases de datos No-SQL

CASO PRÁCTICO

1. Colecciones en archivos .json para importar colecciones una vez creada la base de datos.

factories

```
[
 {
  "metro": { "city": "New York", "state": "NY" },
  "name": "Giant Factory"
 },
 {
  "metro": { "city": "Los Angeles", "state": "CA" },
  "name": "Mega Factory"
 }
]
```

vehiculo

```
[
 {
  "vehiculo": { "marca": "Renault", "modelo": "c3" },
  "ruedas": null
 },
 {
  "vehiculo": { "marca": "Toyota", "modelo": "Corolla" },
  "ruedas": 4
 }
```

]

2. Una vez cargados, abre la poweShell de MongoDB.

- Selecciona la base de datos (en mi caso c1)
    - use c1;

Si la base de datos no existe, MongoDB la creará automáticamente en cuanto insertes documentos o crees índices.

3. Crear índice monoclave en la colección factories:

db.factories.createIndex({ metro: 1 });

La sintaxis **db.factories.createIndex({ metro: 1 });** crea un índice monoclave en la colección factories para el campo metro en orden ascendente.

db.factories: Se refiere a la colección llamada factories dentro de la base de datos activa.

.createIndex: Es el método que crea un índice en la colección para optimizar las operaciones de consulta.

{ metro: 1 }: Define el índice a crear:

- La clave del objeto (metro) indica el campo sobre el cual se creará el índice.
- El valor 1 indica que el índice será en orden ascendente. Si quisieras un índice en orden descendente, usarías -1 en su lugar.

Un índice en MongoDB permite que las búsquedas sean más rápidas y eficientes. En este caso:

- Si consultas documentos por el campo metro (por ejemplo, db.factories.find({ metro: "New York" })), MongoDB utilizará este índice para buscar de manera más rápida que si no existiera.

- Orden ascendente significa que los valores en el índice se organizarán de menor a mayor (alfabéticamente o numéricamente, dependiendo del tipo de datos).

Ventajas del índice

- Mejora el rendimiento de las consultas que filtran o clasifican datos por el campo metro.
- Reduce el tiempo de búsqueda al evitar un escaneo completo de la colección.
- Consideraciones

Si la colección contiene muchos documentos, el índice puede ocupar espacio adicional en disco y memoria.

El índice debe actualizarse cada vez que insertes, actualices o elimines un documento en la colección.

4. Crear índice compuesto en la colección factories:

```
db.factories.createIndex({ metro: 1, name: -1 });
```

La sintaxis **db.factories.createIndex({ metro: 1, name: -1 });** crea un índice compuesto en la colección factories, utilizando los campos metro y name.

{ metro: 1, name: -1 }: Este objeto define los campos y el orden del índice:

- metro: 1: Indica que el campo metro se indexará en orden ascendente.
- name: -1: Indica que el campo name se indexará en orden descendente.

Índice compuesto:

Un índice compuesto permite ordenar y buscar de manera eficiente usando más de un campo.

En este caso, el índice se aplicará primero al campo metro y luego, dentro de los valores de metro, ordenará por name en orden descendente.

Este índice optimiza consultas que involucran ambos campos, como:

db.factories.find({ metro: "New York" }).sort({ name: -1 });

MongoDB usará el índice para encontrar rápidamente documentos con metro = "New York" y ordenarlos por name en orden descendente.

- También es útil para búsquedas parciales, como:
- Consultas que solo filtran por metro: db.factories.find({ metro: "New York" }).
- Pero no optimiza consultas que solo utilicen name sin incluir metro.

Ventajas

- Mejora el rendimiento de las consultas combinadas (filtros y ordenaciones) que involucran los campos metro y name.
- Permite que las consultas sean más rápidas y eficientes en bases de datos grandes.

Orden del índice:

- MongoDB respeta el orden de los campos en el índice.
- En este caso, las consultas deben empezar filtrando por metro para aprovechar el índice.

Espacio:

- Los índices compuestos ocupan más espacio en disco y memoria, pero proporcionan mayor flexibilidad y rendimiento.

Actualizaciones:

- Los índices se actualizan automáticamente cuando se insertan, modifican o eliminan documentos en la colección, lo que puede impactar el rendimiento de las operaciones de escritura.

5. Crear índice sparse en la colección vehiculos:

```
db.vehiculos.createIndex({ vehiculo: 1 }, { sparse: true });
```

La **sintaxis db.vehiculos.createIndex({ vehiculo: 1 }, { sparse: true });** crea un índice sparse (disperso) en la colección vehiculos sobre el campo vehiculo en orden ascendente.

{ vehiculo: 1 }: Indica que se creará un índice sobre el campo vehiculo en orden ascendente (de menor a mayor).

{ sparse: true }: Especifica que el índice será de tipo sparse.

Los índices sparse solo incluyen documentos en los que el campo especificado existe y no tiene el valor null.

- Este índice se utiliza para consultas que filtran por el campo vehiculo.
- Los índices sparse ignoran documentos donde el campo vehiculo no está presente o su valor es null.
- Esto reduce el tamaño del índice y mejora el rendimiento en consultas donde solo se necesitan documentos con valores válidos en vehiculo.
- Este índice se utiliza para consultas que filtran por el campo vehiculo.
- Los índices sparse ignoran documentos donde el campo vehiculo no está presente o su valor es null.
- Esto reduce el tamaño del índice y mejora el rendimiento en consultas donde solo se necesitan documentos con valores válidos en vehiculo.

6. Puedes verificar los índices creados en cada colección con:

Para la colección factories:

```
db.factories.getIndexes();
```

Para la colección vehiculos:

```
db.vehiculos.getIndexes();
```

Esto mostrará los índices existentes, incluyendo los que acabas de crear.

CASO PRÁCTICO 2

1. Preparar el archivo empleado.csv

Nombre,Apellidos,Edad,Dirección

Luis,Gómez,30,Calle A

Ana,Pérez,25,Calle B

Juan,Martínez,35,Calle C

2. Abrir MongoDB Compass

- Inicia MongoDB Compass e inicia sesión en tu base de datos.
- Selecciona la base de datos llamada Empleados o créala si no existe:
- Haz clic en "Create Database".
- Introduce Empleados como nombre de la base de datos y un nombre cualquiera para una colección inicial

3. Importar el archivo .csv

- Ve a la colección donde deseas importar los datos o crea una nueva colección
- Haz clic en el botón "Import Data" en la esquina superior derecha.
- En la ventana emergente:
- Selecciona la opción "CSV".
- Haz clic en "Select File" y elige el archivo empleados.csv.

4. Excluir el campo Dirección

Una vez que cargues el archivo .csv, MongoDB Compass mostrará una vista previa de los datos y todos los campos detectados.

- Busca el campo Dirección en la lista de campos.
- Desmarca la casilla junto a Dirección para excluirlo de la importación.

5. Configurar otras opciones (opcional)

- Marca la opción "Ignore empty Strings" si deseas que se omitan los valores vacíos en los campos.
- Asegúrate de que los tipos de datos (string, number, etc.) estén configurados correctamente para cada campo.