MEDAC
Instituto Oficial de Formación Profesional

ACCESO A DATOS
TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Explotación de bases de datos No-SQL

13

ÍNDICE

OBJETIVOS

Aprender cómo podemos gestionar índices en MongoDB.

Estudiar la gestión de roles y usuarios en MongoDB.

Conocer alguna forma de exportación e importación en MongoDB.

/ 1. Introducción y contextualización práctica

En este nuevo tema sobra bases de datos NoSQL, dedicaremos algún tiempo en explicar qué es un índice, cómo utilizarlo y los diferentes tipos de índices que encontramos en el sistema de almacenamiento de datos, ya conocido del tema anterior, MongoDB.

Aprenderemos también a gestionar los diferentes roles y usuarios en esta base de datos.

Estudiaremos finalmente, una herramienta visual que nos facilitará las operaciones de exportación e importación en la base de datos MongoDB.

A continuación, vamos a plantear un caso práctico a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y Resolución del caso práctico.

Fig. 1. NoSQL.

Audio Intro. "Operaciones MongoDB"

https://bit.ly/3lK5gGF

/ 2. Concepto y características principales de los índices

El concepto de índice que maneja MongoDB es el mismo que el que se maneja en otros repositorios de información.

Audio 1. "Concepto índice"
https://bit.ly/2KbdQAi

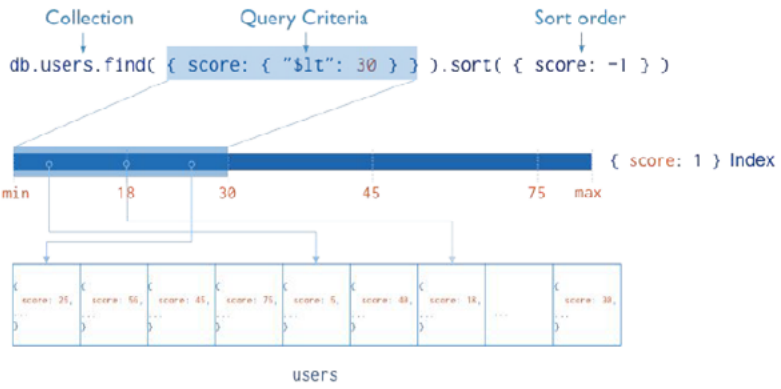

Fig. 2. Esquema índice.

Características de los índices:

- Ofrecen grandes mejoras en las operaciones de lectura.
- Pueden penalizar las operaciones de escritura.
- Se almacenan en memoria.
- Por defecto se crea un índice único sobre el campo _id.
- No sólo se indican los campos del índice sino el orden del mismo.
- Los operadores de negación no utilizan índices.
- Los índices los emplean las querys y las "agregaciones".

A continuación, veremos la sintaxis para la creación de un índice en MongoDB:

```
db.collection.createIndex(
{ <field_1>: <index type>,
<field_2>: <index type> ... }
{<options>})
```

Código. 1. Sintaxis índice.

/ 3. Tipologías de los índices en MongoDB

Los índices en MongoDB pueden ser:

- **Monoclave**: Sólo indexan por un campo de los documentos de la colección.

```
db.collection.createIndex( { <field>: <-1|1> } )
```

Código. 2. Sintaxis índices monoclave.

Características:

- **Sólo afectan a un campo de búsqueda**. Indicar -1 o 1 muestra si el índice es ascendente o descendente.
- **Índices sobre entidades embebidas**: sólo se produce un acierto en el índice si las entidades son exactamente iguales, incluyendo orden.
- **Compuestos**: Indexan por varios campos de los documentos de la colección. El -1 indica ordenación descendente y la 1 ordenación ascendente.

```
db.collection.createIndex( { <field_1>: <-
1|1>,<field_2>: <-1|1> } )
```

Código. 3. Sintaxis indices compuestos.

- ***Hashed***: Indexan de forma clave-valor. Son índices en los que un elemento del índice hace referencia un campo que tiene valores de array. Pueden ser índices de un único campo o compuestos. Se crean tantas entradas del índice como valores contenga el array. Solo uno de los elementos del índice puede tener valores de array.
- ***Text***: Índices de texto.
- **Geoespaciales**: Indexan por coordenadas espaciales
- ***Unique***: Sólo puede haber una correspondencia entre entrada del índice y documento. Indica que cada valor del índice debe corresponder con un único valor. Se puede utilizar tanto para índices de un solo campo o multicampo.

```
db.collection.createIndex({ <field>: <index_type>
}, {unique: <true|false>} )
```

Código. 4. Sintaxis índice únicos.

- ***Sparse***: No indexa campos con valores nulos. Es un índice de valores únicos que admite valores nulos. Los valores nulos no se introducen en el índice. Una búsqueda que utilice este índice no devolverá documentos que no estén en el índice

```
db.collection.createIndex( { <field>:
<index_type> }, {sparse: <true|false>} )
```

Código. 5. Sintaxis.

- ***Background***: Más lento, pero no bloquea a los lectores/escritores.
- **TTL**: Elimina documentos de la colección cuando pasa cierto tiempo o cuando alcanza la fecha de expiración.
- ***Partial***: Indexa aquellos documentos que cumplan una condición.

/ 4. Operaciones con los índices

En MongoDB podemos destacar las siguientes operaciones con los índices:

- ***Covered query***: Es una consulta que se resuelve contra un índice sin consultar datos de la colección.

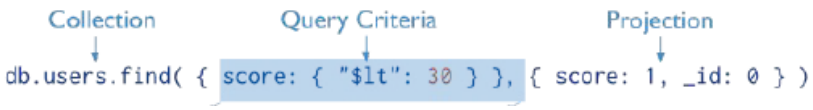

Fig. 3 Ejemplo covered query.

- **Borrado de índice**: El borrado de índice lo realizaríamos de la siguiente forma.

```
db.accounts.dropIndex( { “tax-id”: 1 } )
```

Código. 6. Sintaxis borrado índice.

- **Regeneración índice**: Para volver a crear un índice ya creado previamente usaremos:

```
db.accounts.reIndex()
```

Código. 7. Sixtaxis Reindex.

- **Listado de índices**: Para listar los índices asociados a una colección usaremos:

```
db.accounts.getIndexes()
```

Código. 8. Sintaxis listado ínidices.

Fig 4. Operaciones con los índices.

Video 1. “Documentación oficial”
https://bit.ly/3oQsuMA

/ 5. Caso práctico 1: "Creación índices"

Planteamiento: Se requiere realizar la creación de diferentes índices para las siguientes colecciones:

1. Para la colección de documentos ***factories***, escribir un índice de tipo monoclave al campo **metro**:

```
{
metro: { city: "New York", state: "NY" },
name: "Giant Factory"
}
```

2. Para la misma colección anterior se requiere crear un índice compuesto para poder mostrar con el campo **metro** ascendente y con el campo ***name*** descendente.

3. Para la colección **vehículos**, se ha insertado el siguiente documento:

```
{
vehiculo: { marca: "Renault", modelo: "c3" },
ruedas: null
}
```

Se requiere crear un índice de tipo *sparse* que ordene por el campo *vehículo* y el segundo campo si es nulo, sea ***true***.

Nudo: Una vez estudiados los diferentes tipos de índices veremos cómo podemos resolver los diferentes ejemplos

Desenlace:

1. Para el primer ejercicio la creación del índice de tipo monoclave será el siguiente:

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

2. Para el segundo ejercicio realizaremos un índice compuesto:

```
db.factories.createIndex({a:1, b:-1})
```

3. Para el último ejemplo realizaremos un índice *sparse* o nulo:

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

/ 6. Creación de usuarios y roles

Mongo DB emplea el **control de acceso basado en roles (RBAC)**, para determinar el acceso de los usuarios. A un usuario se le otorgan uno o más roles que determinan el acceso o los privilegios del usuario, a los recursos de nuestra base de datos MongoDB, y a las acciones que dicho usuario puede realizar.

Un usuario debe tener solo el conjunto mínimo de privilegios necesarios para garantizar un **sistema de privilegios mínimos**. Cada aplicación y usuario de un sistema Mongo debe asignarse a un usuario distinto.

Este aislamiento de acceso facilita la revocación del acceso y el mantenimiento continuo del usuario. Para crear un usuario en Mongo DB podemos ejecutar el comando ***"db.createUser()"***.

Si el usuario es creado no devolverá nada, si el usuario existe previamente, nos devolverá un error de duplicado. Veamos a continuación la sintaxis:

```
{
   user: "<name>",
   pwd: passwordPrompt(),     // Or  "<cleartext password>"
   customData: { <any information> },
   roles: [
   { role: "<role>", db: "<database>" } | "<role>",
    ...
   ],
   authenticationRestrictions: [
      {
        clientSource: ["<IP>" | "<CIDR range>", ...],
        serverAddress: ["<IP>" | "<CIDR range>", ...]
      },
      ...
   ],
   mechanisms: [ "<SCRAM-SHA-1|SCRAM-SHA-256>", ... ],
   passwordDigestor: "<server|client>"
}
```

Código.9. Sinxtaxix create user.

En el código anterior, podemos observar:

- ***User***: *String* que indica el nombre del nuevo usuario
- ***Pwd***: *String* donde indicaremos la contraseña.
- ***CustomData***: Documento opcional. En este campo indicaremos información adicional como puede ser el nombre completo del usuario o el ID del empleado.
- **Roles**: En este array indicaremos los permisos que queramos otorgarle al nuevo usuario. Vacío si no lo queremos dotar de permisos.
- ***authenticationRestrictions***: Es un array opcional. Se establecerá un rango de direcciones IP desde las cuales dicho usuario se podrá conectar.
- ***Mechanisms***: array opcional que especifica si dicho usuario tiene permisos para crear credenciales SCRAM.
- ***passwordDigestor***: *String* opcional que indica si el servidor o el cliente traducen la contraseña.

6.1. Acceso de control basado en roles (RBAC)

Como hemos adelantado en el apartado anterior, MongoDB emplea el control de acceso basado en roles (RBAC) para controlar el acceso a un sistema MongoDB, además, a un usuario se le otorgan uno o más roles que determinarán el acceso del usuario a los recursos y operaciones de la base de datos, lo que quiere decir, que fuera de las asignaciones de funciones, el usuario no tiene acceso al sistema.

MongoDB no habilita el control de acceso basado en roles por defecto. **Se puede habilitar la autorización mediante la configuración "*--auth*" o a través de la configuración "*security.authorization*"**. Si se habilita la autenticación interna también se habilita la del cliente. Una vez que se habilita el control de acceso, los usuarios deben autenticarse. Un rol otorga privilegios para realizar acciones específicas sobre nuestros recursos. Cada privilegio se especifica explícitamente en el rol o se hereda de otro rol. **Un privilegio** se aplica sobre un recurso especificado, y sobre las acciones permitidas de ese mismo.

Un recurso puede ser una base de datos, una colección, un conjunto de ellas, o incluso el clúster. Si el recurso es el clúster, las acciones afiliadas afectan al estado del sistema en lugar de a una base de datos o colección determinada. Una acción específica la operación permitida sobre el recurso. Podemos usar el comando "***rolesInfo***" para visualizar los privilegios de un rol con el parámetro "showPrivileges" y "showBuiltinRoles" a *true*.

```
{
   rolesInfo: { role: <name>, db: <db> },
   showPrivileges: <Boolean>,
   showBuiltinRoles: <Boolean>,
   comment: <any>
}
```

Código. 10 Sintaxis rolesInfo.

6.2. Ejemplo creación de rol

A continuación, veremos un ejemplo de cómo crear un rol que provee privilegios para correr dos bases de datos.

- En primer lugar, **nos conectaremos a nuestra base de datos** MongoDB con los privilegios pertinentes.

```
mongo --port 27017 -u user -p '1234' --authenticatio
nDatabase 'admin'
```

Código. 11. Acceso con privilegios.

- A continuación, **crearemos un nuevo rol** para administrar las operaciones actuales. Crearemos un rol llamado "manRole":

```
use admin
db.createRole(
   {
      role: "manRole",
      privileges: [
        { resource: { cluster: true }, actions: [ "killop",
"inprog" ] },
        { resource: { db: "", collection: "" }, actions: [
"killCursors" ] }
      ],
      roles: []
   }
)
```

Código. 12 creación role.

En la operación anterior podemos observar cómo hemos creado un rol que garantiza **permisos para hacer "kill"** de cualquier operación.

En la figura nos ilustra la creación rol de permisos en MongolDB.

Fig. 5 Creación rol. Permisos.

/ 7. Importación de datos

A continuación, procederemos a estudiar el proceso por el cual realizaremos el proceso de **importación de datos a través de** una herramienta visual llamada ***Compass***.

- El primer paso sería **conectarnos a nuestra base de datos mongoDB** como hemos estudiado previamente.
- Después haríamos clic en el botón "***Add Data***" y seleccionaríamos la opción "***Import File***"
- La aplicación nos mostrará un cuadro de diálogo donde deberemos de indicar la **ruta del fichero** que queremos introducir a nuestra base de datos.

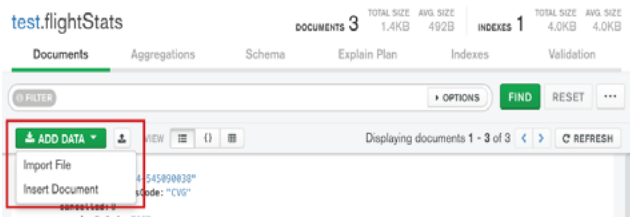

Fig. 6. Import data 1.

- Una vez elegida la ruta, seleccionaremos el **tipo de fichero** que vamos a introducir: JSON o CSV. Si importamos un fichero CSV deberemos especificar los campos que vamos a importar y los tipos de los mismos. El tipo de datos por defecto es ***String***:

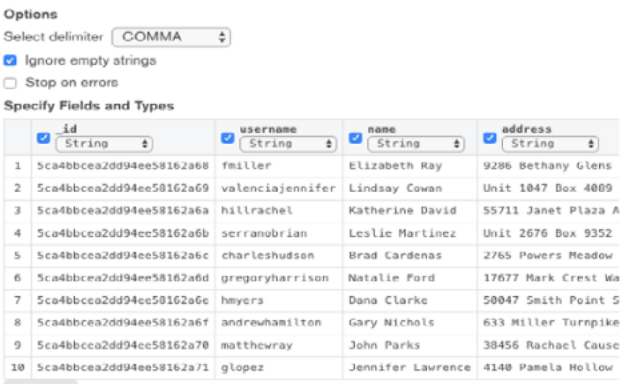

Fig. 7. Import data 2.

- Como vemos en la imagen superior, tendremos que **configurar** las opciones de **importación** acorde a nuestro caso. Si importamos un CSV tenemos que indicar cómo están delimitados los campos.
- Una **barra de progreso** mostrará el estado actual de la importación. Si ocurre algún error durante el proceso de importación, la barra se mostrará en color rojo, y aparecerá un mensaje en el cuadro de dialogo. Una vez finalizado el proceso, la aplicación mostrará los datos importados.

Video 2. "Instalación Compass"
https://bit.ly/2HcOxNb

/ 8. Exportación de datos

A continuación, procederemos a realizar la operación inversa. Realizaremos una **exportación de datos con la aplicación visual *Compass***.

- El primer paso sería **conectarnos a nuestra base de datos MongoDB y navegar** hasta encontrar la información, colección/es que deseemos exportar.

- Haremos Clic en el menú "***Collection***" de nuestra aplicación y a continuación en la opción "***Export Collection" y Compass*** nos mostrará el siguiente cuadro de dialogo:

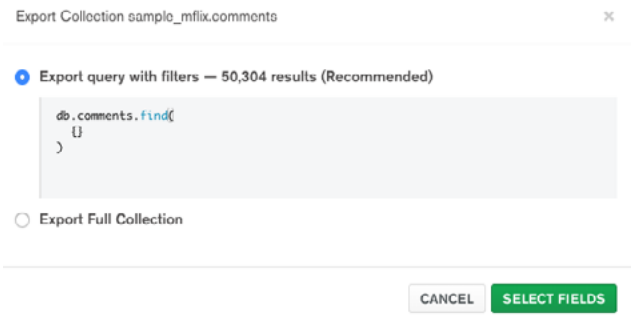

Fig.8 Export data 1.

El cuadro muestra la "***query***" por medio de la que se va a realizar la operación. Si queremos ignorar la "*query*" y exportar directamente la colección completa podemos seleccionar el radio ***button "Export Full Collection***" y hacer clic en "***Select Filelds***".

- En esta parte se nos muestra otro cuadro de diálogo donde **seleccionaremos los campos a exportar**.

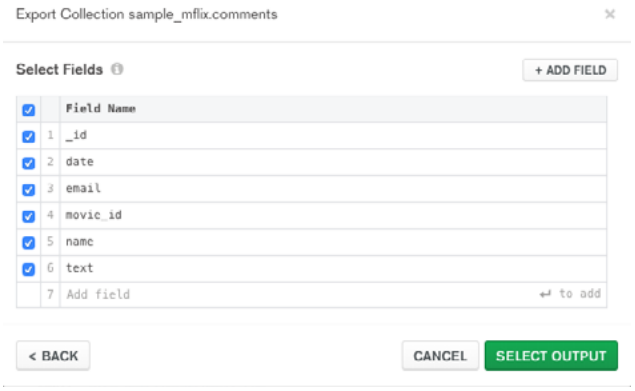

Fig. 9. Export Data 2.

Si nuestra aplicación no nos detecta algún campo podremos añadirlo manualmente con el botón de "*Add Field*". Si todo está correcto hacemos clic en "***Select Output***".

- Básicamente en esta parte deberemos de **seleccionar el formato del fichero** que queremos exportar: JSON o CSV son las opciones disponibles.

- Hacemos clic en ***Export***.

/ 9. Caso práctico 2: "Importación fichero csv"

Planteamiento: Disponemos de una base de datos MongoDB cuyo nombre es **Empleados**.

Se requiere importar a nuestra base de datos una colección de datos a través de un fichero .csv. Dicho fichero contiene la información personal de los empleados:

- Nombre
- Apellidos
- Edad
- Dirección

Un dato a tener en cuenta, es que no interesa disponer del campo "Dirección" en la colección que se importe en base de datos

Nudo: para realizar la importación solicitada usaremos la herramienta visual ya utilizada, *Compass*.

Desenlace:

- En primer lugar, nos conectaremos a la base de datos donde vamos a importar la colección deseada
- Haremos clic sobre el botón de "Add Data" y justo después a la opción "Import File".

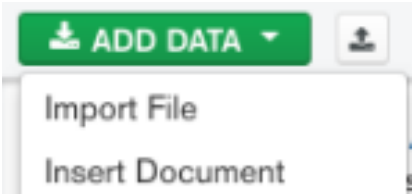

Fig. 10 Import Data 3.

- Después elegiremos la ruta del fichero .csv que nos han proporcionado, elegiremos la opción de csv, pulsaremos en "*Ignore empty Strings*" para ignorar los campos vacíos y haremos clic en "*Import*".
- Aquí se nos abrirá nuestro documento con los campos delimitados por el csv introducido. El ejercicio en este punto nos especifica que no se quiere disponer del campo "Dirección" en la colección final de la base de datos. La implementación de esto es tan sencilla como desmarcar dicho campo, y ese campo no será importado.
- Hacemos clic en *Import* y tendremos nuestra colección nueva importada en la base de datos **Empleados**.

/ 10. Resumen y resolución del caso práctico de la unidad

En este tema hemos realizado una inmersión a fondo en los diferentes **tipos de índices** y cómo crearlos en nuestra base de datos MongoDB. Sus características y peculiaridades.

También hemos estudiado los **roles** que podemos encontrar en la base de datos, los privilegios, y cómo otorgar a un usuario dichos privilegios.

Por último, hemos aprendido a usar una herramienta muy interesante llamada ***Compass***, la cual, podremos compaginar al Shell de Mongo para realizar operaciones de importación o exportación.

Resolución del caso práctico inicial

Según el enunciado del caso práctico inicial, se requiere crear un nuevo usuario con la otorgación a una serie de privilegios que coinciden con el role "*readwrite*" en Mongo para la base de datos "*Maths*".

Para ello deberemos de ejecutar el comando en la base de datos: db.createUser(), y dentro añadiremos el siguiente fragmento:

```
{
   user: "vGonzalez
   pwd: "rFgtR3456H",
   customData: { Este usuario tendrá privilegios sobr
e el rol readwrite },
   roles: [
      { role: "readWrite", db: "Maths" }
   ],
   mechanisms: [ "SCRAM-SHA-1"],
   passwordDigestor: "server
}
```

Código.13. Ejemplo creación usuario

De esta forma estaremos creando el usuario "vGonzalez" con la contraseña indicada en *pwd*, y además añadiéndole el rol "*readWrite*" sobre la base de datos anteriormente mencionada, "Maths".

/ 11. Webgrafía

https://docs.mongodb.com/manual/security/