ACCESO A DATOS

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

MongoDB
Índices
Roles y usuarios
Exportación e importación

Índices en MongoDB

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

Concepto índice

Un índice permite que una **consulta devuelva información** de forma **más efectiva**. Los índices están relacionados a tablas, o en el caso de MongoDB, a **colecciones específicas o campos** de las mismas con una o más **claves definidas**.

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:

Sintaxis índice

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

Tipologías de los índices en MongoDB

Los índices en MongoDB pueden ser:

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

Sintaxis índices monoclave

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

Características:

- Sólo afectan a un campo de búsqueda.

  Indicar:

  -1 si el índice es ascendente,

  1 si el índice es 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.

Sintaxis indices compuestos

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

Hashed: Indexan de forma clave-valor.

Son índices en los que un elemento del índice hace referencia a 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 (o de tipo único): 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.

Sintaxis índice únicos

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

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:

Sintaxis sparse

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

Background: Más **lento**, pero **no bloquea a los lectores/escritores**.

TTL: **Elimina documentos** de la colección cuando:

- pasa cierto tiempo,
- alcanza la fecha de expiración.

Partial: Indexa aquellos documentos **que cumplan una condición**.

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**. (Como ya está guardado en memoria, consultamos directamente esos datos, sin consultar los datos de la colección).

Ejemplo covered query

db.users.find(**{ score: { "$lt": 30} }**, { score: 1, _id: 0 })

• Borrado de índice: El borrado de índice lo realizaríamos de la siguiente forma:

Sintaxis borrado índice

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

• Regeneración índice: Para volver a crear un índice ya creado previamente usaremos. (la explicación del profesor fue: Regenera todos los índices de la colección):

Sixtaxis Reindex

db.accounts.reIndex()

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

Sintaxis listado índices

db.accounts.getIndexes()

Documentación oficial

En primer lugar, quiero presentarles la página de la documentación oficial web de MongoDB:

https://www.mongodb.com/docs/

Aquí encontramos un árbol que abarca toda la documentación, y gran parte de la teoría que hemos utilizado se basa en esta documentación oficial, la cual recomiendo completamente por su exhaustividad. Aunque está en inglés, podemos traducirla fácilmente, lo que también nos resulta útil para practicar el idioma.

Como podemos observar, en el tema anterior exploramos el shell, cubriendo la mayor parte o la totalidad de la teoría sobre su uso, los requisitos necesarios, cómo iniciar los servicios de Mongo y cómo configurar el shell. Contamos con una abundancia de información relacionada con el tema actual, incluyendo índices. Aquí encontramos explicaciones gráficas sobre qué es un índice, los tipos de índices que encontraremos, y cómo crear un índice, aspecto que abordaremos en esta unidad.

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

Además, disponemos de diversos tipos de índices para explorar, acompañados ocasionalmente de ejemplos gráficos. Esta web de mongodb.com/docs se presenta como una herramienta valiosa para contrastar y ampliar la información, ya que todo lo que se presenta aquí constituye la documentación oficial sobre el código de Mongo.

Simplemente quiero mostrarles esta herramienta. Aquí pueden encontrar secciones sobre introducción e instalación, lo que les permite contrastar lo que hemos estado viendo y aprender nuevas cosas, como transacciones, agregaciones y la operación CRUD, que también se trató en el tema pasado. Les dejo aquí la herramienta para que le echen un vistazo y la utilicen como refuerzo conceptual.

Ejemplo de creación índices

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**.

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

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})

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:

Sintaxis create user

```
{
    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>"
}
```

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**. (Se suele usar para que los empleados no se conecten fuera de la oficina).

• 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**.

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 una de estas configuraciones:

- "—auth".
- "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 especifica 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**.

Sintaxis rolesInfo

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

Ejemplo de 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:

Acceso con privilegios

mongo --port 27017 -u user -p '1234' --authenticationDatabase 'admin'

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

Creación role

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

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

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**.

1. El primer paso sería **conectarnos a nuestra base de datos** mongoDB como hemos estudiado previamente.
2. Después haríamos clic en el botón "**Add Data**" y seleccionaríamos la opción "**Import File**".
3. La aplicación nos mostrará un cuadro de diálogo donde deberemos de indicar la **ruta del fichero** que queremos introducir a nuestra base de datos.
4. 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**.

   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.
5. 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**.

Instalación Compass

Abordaremos los documentos de Mongo y exploraremos la documentación oficial.

Hemos observado varias herramientas visuales de gestión, pero Compass es la opción principal proporcionada por Mongo, ya que es desarrollada por ellos. Pueden acceder y descargar el software directamente desde la siguiente URL.

https://www.mongodb.com/es/products/tools/compass

Esta herramienta ha sido utilizada en los ejemplos de importación y exportación de colecciones que hemos estado aprendiendo.

La instalación es sencilla: descarguen el software desde la URL proporcionada y seleccionen la versión adecuada para su sistema operativo (Windows, Mac o Linux). Elijan entre alguna versión beta o la última estable. Sigan los pasos de instalación y consulten la documentación para obtener información detallada sobre la aplicación.

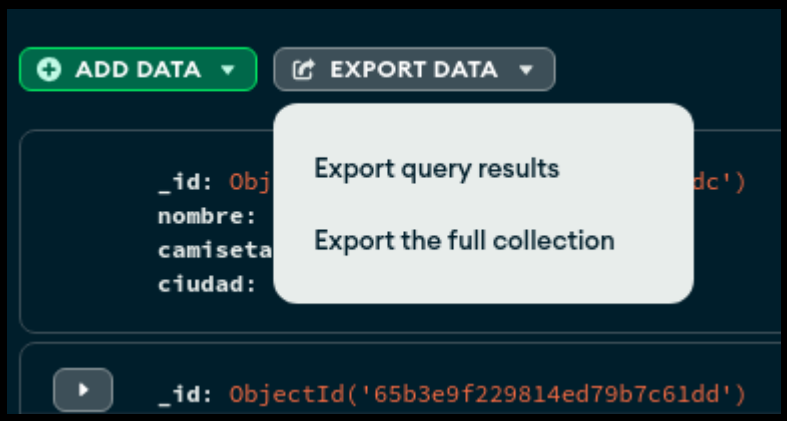
Al conectarnos a la base de datos, la documentación nos guía en el proceso de conexión y las interacciones con la información disponible. La página se presenta como un tutorial completo que cubre desde el inicio hasta el final de su funcionamiento. Aunque nos hemos centrado en la importación y exportación de datos, Compass es una herramienta potente que complementa al Shell. Es una excelente opción para aquellos que pueden sentir cierta aprensión hacia el Shell o simplemente prefieren una interfaz más visual.

Además de la gestión de datos, la aplicación ofrece funciones como importación-exportación, la cual hemos explorado en este tema. Les recomiendo explorar a fondo esta herramienta, ya que puede ser de gran utilidad para gestionar su base de datos Mongo. Aquí les dejo la herramienta para que la estudien y le saquen el máximo provecho.

Exportación de datos

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

1. El primer paso sería **conectarnos a nuestra base de datos** MongoDB y navegar hasta encontrar la información, **colección**/es que deseemos exportar.
2. 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:



   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”.

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

   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”.

4. Básicamente en esta parte deberemos de seleccionar el formato del fichero que queremos exportar:

   JSON o **CSV** son las opciones disponibles.

5. Hacemos clic en Export.

Ejemplo de importación de fichero .csv

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.

Para realizar la importación solicitada usaremos la herramienta visual ya utilizada, Compass.

1. En primer lugar, nos **conectaremos a la base de datos** donde vamos a importar la colección deseada.
2. Haremos clic sobre el botón de “Add Data” y justo después a la opción “Import File”.
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”.
4. 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.
5. La implementación de esto es tan sencilla como desmarcar dicho campo, y ese campo no será importado.
6. Hacemos clic en Import y tendremos nuestra colección nueva importada en la base de datos Empleados.

Ejemplo de creación de usuario y rol

Se requiere crear un **nuevo usuario** en la **base de datos “Maths”**, en MongoDB, con las características de **role “readwrite”** para tener los siguientes permisos:

- *collStats*
- *convertToCapped*
- *createCollection*
- *dbHash*
- *dbStats*
- *dropCollection*
- *createIndex*
- *dropIndex*
- *find*
- *insert*
- *killCursors*
- *listIndexes*
- *listCollections*
- *remove*
- *renameCollectionSameDB*
- *update*

Para ello deberemos de ejecutar el comando en la base de datos:

db.createUser()

y dentro añadiremos el siguiente fragmento:

Ejemplo creación usuario

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

De esta forma estaremos:

1. creando el usuario “vGonzalez”
2. con la contraseña indicada en pwd,
3. y además añadiéndole el rol “readWrite” sobre la base de datos anteriormente mencionada, “Maths”.

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