Índice

Referencias:

Curso básico https://openwebinars.net/accounts/login/?next=/academia/portada/mongodb/
Curso avanzado https://openwebinars.net/academia/aprende/mongodb-2017/
Tutorial https://www.tutorialspoint.com/mongodb/index.htm
Manual https://docs.mongodb.com/manual/

Características de los índices

● Estructuras de datos especiales gestionadas por MongoDB.
● Almacenan una pequeña porción de los datos de la colección de una manera rápida y fácil de acceder.
● El índice almacena el valor de un campo específico o de un conjunto de campos, ordenados por el valor de dichos campos.
● Permiten optimizar la ejecución de las consultas que afecta a la velocidad con la que estas se ejecutan y devuelven el resultado.
● Agregar un nuevo índice afecta a las operaciones de escritura. Para colecciones que manejan gran número de operaciones de escritura, los índices afectan un poco el rendimiento puesto que con cada inserción de documentos se deben actualizar los índices.
● Sin los índices, MongoDB debe hacer un análisis de la colección, es decir, leer todos los documentos en una colección, con el fin de encontrar los documentos que cumplan con los criterios de consulta.
● En definitiva, mejoran el rendimiento que de la base de datos en relación a la ejecución de consultas, apoyan la resolución eficiente de las queries.

Análisis del rendimiento de las querys

```
>db.COLLECTION_NAME.find({<fiter>}).explain(“executionStats”)
```

Al agregar el **explain(“executionStats”)** obtendremos información relacionada a la ejecución de la consulta. Si prestamos atención a la llave “executionStats” tendremos dos valores importantes:

- “executionTimeMillis”: tiempo total de la ejecución de la consulta en milisegundos.
- “totalDocsExamined”: número total de documentos analizados.

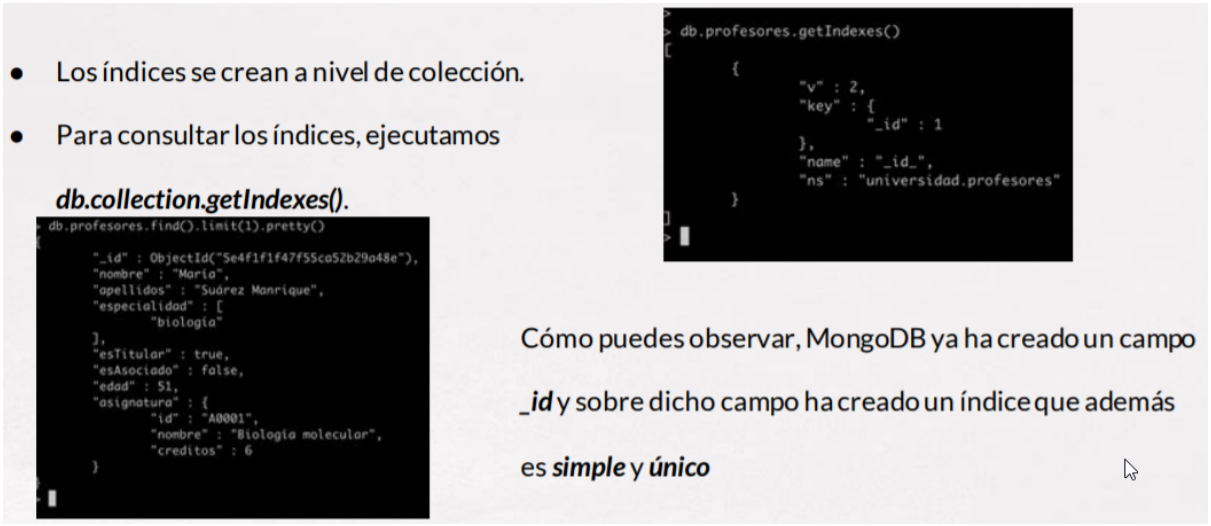
Para ver la diferencia usando un índice o no:
>db.collection.find({key:value}).hint({key:orden}).explain(“executionStats”)
>db.collection.find({key:value}).hint({$natural:orden}).explain(“executionStats”)

Estrategias de indexación

- Cree índices para respaldar sus consultas
  Un índice admite una consulta cuando el índice contiene todos los campos escaneados por la consulta. La creación de índices que admitan consultas da como resultado un rendimiento de consultas mucho mayor.
- Utilice índices para ordenar los resultados de la consulta
  Para admitir consultas eficientes, utilice las estrategias aquí cuando especifique el orden secuencial y el orden de clasificación de los campos de índice.
- Asegúrese de que los índices encajen en la RAM
  Cuando su índice cabe en la RAM, el sistema puede evitar leer el índice del disco y usted obtiene el procesamiento más rápido.
- Cree consultas que aseguren la selectividad
  La selectividad es la capacidad de una consulta para limitar los resultados mediante el índice. La selectividad permite a MongoDB usar el índice para una mayor parte del trabajo asociado con el cumplimiento de la consulta.

Consultar los índices existentes

```
>db.COLLECTION_NAME.getIndexes()
```



Crear índices

Al crear el índice sólo examinamos los documentos necesarios para obtener los resultados esperados y no toda la colección, lo cual representa una mejora considerable al rendimiento de nuestra aplicación.

La operación de creación de índices es muy costosa por lo que se debe hacer con cautela, analizando los realmente necesarios.

`>db.collection.createIndex(keys:orden, options)`
(1 para ordenarlo ascendentemente y -1 para ordenarlo descendentemente)

Opciones de creación de los índices

unique (booleano)

- Crea un índice único para que la colección no acepte la inserción o actualización de documentos donde el valor de la clave del índice coincide con un valor existente en el índice.
- Especifique true para crear un índice único. El valor predeterminado es false.
- Se define sobre índices que queremos que funcionen como una primary key.
- No puede aplicarse sobre una colección que ya contenga datos que violen la norma (no pueden repetirse ni ser nulos).
- Podrán insertarse documentos en la colección si no contienen el campo al que el índice se refiera.
- Pueden crearse índices únicos simples o compuestos.
  >db.collection.createIndex({“email”:1},{unique:true})
  >db.collection.createIndex({“nombre”:1, “apellidos”:1},{unique:true})

name (string)

El nombre del índice. Si no se especifica, MongoDB genera un nombre de índice concatenando los nombres de los campos indexados y el orden de clasificación.

sparse (booleano)

Permite filtrar los resultados que se obtienen buscando por índice y no mostrando aquellos que no tienen el campo.
Si true, el índice solo hace referencia a documentos con el campo especificado. Estos índices utilizan menos espacio pero se comportan de manera diferente en algunas situaciones (en particular, en los tipos). El valor predeterminado es false.

Ejemplo:

```
{ "_id" : "1", "user_id" : "antonio", "horas" : 16}
{ "_id" : "2", "user_id" : "juan", "horas" : 12}
{ "_id" : "3", "user_id" : "pablo"}

db.developers.ensureIndex( { horas : 1 }, { sparse : true } )

db.developers.find( { horas : { $lt : 16 } } )

db.developers.find().sort({ horas : -1 })
db.developers.find().sort({ horas : -1 }).hint({ horas : 1 })
```

partialFilterExpression (documento)

Si se especifica, el índice solo hace referencia a documentos que coinciden con la expresión de filtro. Solo se indexa la parte de la muestra que cumpla la condición especificada.

Una expresión de filtro puede incluir:

- expresiones de igualdad (es decir, field: value o utilizando el $eq operador),
- $exists: true expresión,
- $gt, $gte, $lt, $lte Expresiones,
- $type expresiones,
- $and operador en el nivel superior solamente

Es compatible con la opción sparse.

Ejemplo:

```
db.delanteros.createIndex({
      nombre : 1, equipo : 1 },{
      partialFilterExpression : { goles : { $gt : 10 } }})
```

¿Qué ocurre si filtramos para delanteros con más de 12 goles?
¿Qué ocurre si filtramos para delanteros con menos de 15 goles?
¿Si buscamos delanteros del Betis con más de 12 goles usamos el índice?
¿Si buscamos a Messi del Barcelona usamos el índice?

Si queremos además que sea único, la condición de ser único sólo debe cumplirse para aquellos documentos que cumplan con el filtro parcial.

Ejemplo:

```
db.delanteros.createIndex({
      nombre : 1 },{
      unique : true,
      partialFilterExpression : { goles : { $gt : 10 } }})

{ "_id" : "1", "nombre" : "Rubén", "goles" : 16}
{ "_id" : "2", "nombre" : "Lionel", "goles" : 12}
{ "_id" : "3", "nombre" : "Cristiano", "goles" : 11}
```

¿Qué ocurre si introducimos a Rubén con 19 goles?
¿Qué ocurre si introducimos a Cristiano con 10 goles?
¿Qué ocurre si introducimos a Rubén sin campos goles?

expireAfterSeconds (entero)

Especifica un valor, en segundos, como TTL para controlar cuánto tiempo MongoDB retiene los documentos de esta colección y el propio motor de mongo eliminará el documento transcurrido ese tiempo desde que se indexó.

```
>db.collection.createIndex(keys:orden, options)
```

Ejemplo:
db.collection.createIndex({ "Key": 1 }, { expireAfterSeconds: 3600 })
Cada documento que se inserte con una key se borrará después de una hora.

Subdocumentos

Se puede crear un índice sobre un subdocumento. Incluso se pueden mezclar las dos opciones:

Ejemplo:
db.cuentas.createIndex({"cuentas.saldo":1})
db.cuentas.explain().find({"cuentas.saldo":{$gt:1500}})

Eliminar índices

Si un índice no se usa mucho es mejor eliminarlo.

Podemos eliminar un índice en particular `>db.COLLECTION_NAME.dropIndex({KEY_1})`

Podemos eliminar todos los índices (menos el del campo _id)
`>db.COLLECTION_NAME.dropIndexes()`

Tipos de índices

- _id: Se genera automáticamente por parte de Mongo para cualquier documento.
- Simples: Se definen sobre un único campo de cualquier documento.
- Compuesto: Se definen sobre varios campos de un documento.
- Multillave: Se emplean cuando el campo a indexar pertenece a un subdocumento dentro de un array del documento padre.
- Texto: Se definen para indexar el contenido de los campos de tipo string.
- Otros menos comunes:
    - Geoespacial: Se definen para indexar campos relativos a coordenadas de tipo GeoJSON.
    - Hash: Se emplean para almacenar los valores hasheados del campo a indexar cuando aplicamos sharding sobre las bases de datos.

Índices simples

Estos índices se aplican a un solo campo de nuestra colección.

db.collection.createIndex(<key and index type specification>, <options>

)

Como ejemplo, hemos creado un índice simple sobre el campo ***edad*** y cuyos valores estarán ordenados descendentemente (**-1**).

db.collection.dropIndex('edad_-1')

Podemos borrar un índice o todos los que hayamos creado:

db.profesores.dropIndex('edad_-1')

db.profesores.dropIndexes()

Índices compuestos

Estos índices se aplican varios campo de nuestra colección.

db.collection.createIndex(<key and index type specification>, <options>

)

Como ejemplo, hemos creado un índice compuesto sobre el campo ***matrícula*** con orden ascendente (**1**) y ***edad*** con orden descendente (**-1**).

db.profesores.createIndex({matricula: 1, edad: -1})

El índice compuesto consta de { matricula: 1, edad: -1 }, el índice ordena primero por matricula y luego, dentro de cada valor de matricula, ordena por edad. El orden es importante y afecta al rendimiento.

Índices Multikey

MongoDB utiliza índices de varias claves para indexar el contenido almacenado en arrays. Si indexa un campo que contiene un valor de array, MongoDB crea entradas de índice independientes para cada elemento del array. Estos índices de varias claves permiten que las consultas seleccionen documentos que contienen arrays haciendo coincidir el elemento o elementos de los arrays.

Ejemplo: índice sobre tags.

```
{
 title:"MongoDB: base de datos NoSQL",
 tags: ["MongoDB","10gen","tutorial"]
}
```

Si creamos un índice sobre el campo tags, podremos buscar documentos con las querys {tags:“MongoDB”}, {tags:“10gen”} o {tags:“tutorial”}.

Si el array contiene documentos también podemos generar índices

```
{
 title:"MongoDB: base de datos NoSQL",
 tags:
 [
  {id:1222,text:"MongoDB"},
  {id:1223,text:"10gen"},
  {id:1234,text:"tutorial"}
 ]
}
```

Podemos crear el siguiente índice

```
> db.blogPosts.createIndex({"tags.text":1})
```

Lo que nos permitirá realizar consultas utilizando dicho índice:

```
> db.blogPosts.find({"tags.text":"MongoDB"})
```

También podemos crear índices compuestos que contengan índices multikey, pero hay que tener en cuenta que solo uno de los campos en el índice compuesto podrá ser un array. Por ejemplo, pensemos en documentos con el siguiente formato

```
{
 title:"MongoDB: base de datos NoSQL",
 authors:["Pedro J","Roman K", "Luis"],
 tags: ["MongoDB","10gen", "tutorial"]
}
```

Está permitido crear un índice compuesto para {title:1,authors:1}, pero no crear un índice para {authors:1,tags:1} ya que MongoDB tendría que hacer el producto cartesiano de ambos arrays lo que podría dar lugar a un índice imposible de mantener.

MongoDB determina automáticamente si se debe crear un índice de varias claves si el campo indexado contiene un valor de array; no es necesario especificar explícitamente el tipo de múltiples claves.

Es poco eficiente.

Índices de Texto

MongoDB proporciona índices de texto para admitir consultas de búsqueda de texto sobre contenido de cadenas. Los índices text pueden incluir cualquier campo cuyo valor sea una cadena o un array de elementos de cadena.

Una colección puede tener como máximo un índice text.

```
db.collection.createIndex( { key: "text" } )
```

El índice no tiene en cuenta mayúsculas (Berto y berto son consideradas iguales) y también intenta omitir repeticiones con plurales (en esto es menos hábil, ya que toma las reglas del inglés).

Ejemplo:
Las consultas basadas en el índice usan también find, pero tienen una estructura distinta.
>db.frases.find({$text:{$search:"amor"}})

La siguiente consulta muestra las frases que tiene o bien amor o bien odio
>db.frases.find({$text:{$search:"amor odio"}})

Consejos sobre índices

- Cuantos más índices --> consultas más rápidas
- Cuantos más índices --> escrituras más lentas
- En el caso de importaciones masivas es más rápido crear los índices al final
- Índices tan grandes que no quepan en memoria pueden ser de menos utilidad. Esta información la podemos encontrar con el comando db.coleccion.stats() o directamente db.coleccion.totalIndexSize()
- Dada una consulta si hay que elegir:
    - El primer candidato para indexar es alguna clave que contenga una igualdad, si la hay
    - El segundo es alguna clave incluida en un sort
    - Finalmente las claves que impliquen un rango (mayor o menor)

- Hay que ser cuidadosos al crear índices, hacerlo solo para consultas realmente repetidas.

EJERCICIO

Sobre la colección coches_profesores realiza las siguientes acciones:

- Modifica sus documentos añadiendo las siguientes propiedades:
    - email
    - clave
    - salario
- Crea un índice simple sobre el campo salario.
- Crea un índice compuesto sobre los campos email y clave. Elige el orden sobre cada campo como desees.
- Elimina los índices (uno a uno por su nombre o todos de una vez)