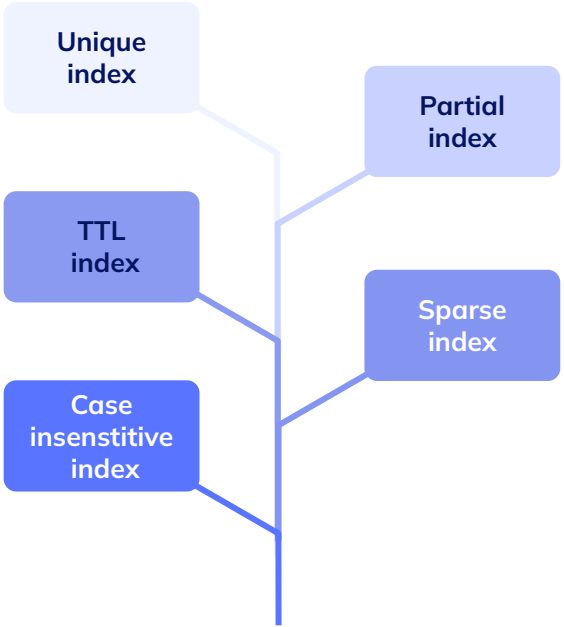
MongoDB Fundamentos

Módulo 3

Opciones para indexar



1. **Índice único (*Unique index)***

Cada documento presente en la colección Mongo contiene un índice predeterminado llamado "`_id`". Se crea una identificación de objeto al crear el documento si no hay valores de índice presentes.

Ejemplo:

```
db.collection.createIndex ({nombre: 1} , { unique: true })
```

Las restricciones únicas se pueden imponer también en el índice compuesto, índice multiciclo e índice de texto.

```
> db.personas.createIndex({nombre:1,apellidos:1},{unique:true})
{
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "createdCollectionAutomatically" : true,
 "ok" : 1
}

> db.personas.insert({nombre:"bertoldo",apellidos:"cacaseno"})
WriteResult({ "nInserted" : 1 })
> db.personas.insert({nombre:"herminia",apellidos:"cacaseno"})
WriteResult({ "nInserted" : 1 })
> db.personas.insert({nombre:"bertoldo",apellidos:"chirimoyo"})
WriteResult({ "nInserted" : 1 })
> db.personas.insert({nombre:"bertoldo",apellidos:"cacaseno"})
```

...

```
WriteResult({
   "nInserted" : 0,
   "writeError" : {
      "code" : 11000,
      "errmsg" : "E11000 duplicate key error collection:
test.personas index:  nombre_1_apellidos_1 dup key: { nombre:
\"berto
ldo\", apellidos: \"cacaseno\" }"
   }
})
```

En el último **`insert`** se produce el error, al querer insertar un documento repetido.

2. **Índice parcial (*Partial index)***

Indexa documentos de una colección en función de **un filtro o expresión específicos.** Dado que los índices parciales **indexan solo un subgrupo**, los requisitos de espacio de recolección (memoria) son menores y esto también resulta en un rendimiento reducido.

Ejemplo:

```
db.pupils.createIndex ( {nombre: 1}, {
partialFilterExpression: { edad: {$ gt: 5}} )
```

Filtrar expresiones compatibles:

- **`$ gt`**, **`$ gte`**, **`$ lt`**, **`$ lte`** (mayor que, mayor que o igual, menor que y menor que o igual).
- operadores tipo **`$`**
- **`$exists`**: operación verdadera.
- Operador de igualdad (**`$eq`**)
- Lógica y / u operaciones.

3. **Índice TTL (*TTL index)***

Los índices TTL son **índices especiales de un solo campo** que MongoDB puede usar **para eliminar automáticamente documentos de una colección**, después de una cierta cantidad de tiempo o en un horario específico.

La caducidad de los datos es útil para ciertos tipos de información, como datos de eventos generados por máquinas, registros e información de sesiones que solo **necesitan persistir en una base de datos durante un período de tiempo finito**.

Para crear un índice TTL, se utiliza el método **`createIndex()`** en un campo cuyo valor sea una fecha y especifique la opción **`expireAfterSeconds`**.

Se puede usar el índice TTL de dos formas, que se detallan a continuación.

1. *Especificando la opción **`expireAfterSeconds`** con el valor TTL deseado en segundos y la fecha dentro del campo indexado en el documento:*

```
db.log.createIndex( { "lastModifiedDate": 1 }, { expireAfterSeconds: 10000 } )
```

```
db.log.insert({ "lastModifiedDate": new Date(), ... : ...., })
```

Este documento se eliminará automáticamente a los 10000 segundos de haber sido insertado.

2. Especificando la **opción `expireAfterSeconds`** con el valor TTL en 0 (cero) y la fecha futura de expiración **(formato YYYY/MM/DD HH:MM:SS)** dentro del campo indexado en el documento:

```
db.log.createIndex( { "lastModifiedDate": 1 }, { expireAfterSeconds: 0 } )
```

```
db.log.insert({ "lastModifiedDate": new Date("YYYY/MM/DD HH:MM:SS"), ... : ...., })
```

Este documento se eliminará automáticamente cuando se llegue a la fecha futura.

4. **Índice disperso (*Spare index)***

Los índices *spare (*“dispersos”) **indexan solo los documentos que contienen el valor de campo del índice.** Ignora todos los demás documentos que no contienen el campo. Por defecto, los índices no dispersos contienen todos los documentos en las colecciones, con un valor nulo como el valor para aquellos campos que no están presentes.

Ejemplo:

```
db.pupil.createIndex( {"age": 1},
{sparse: true} )
```

El índice no indexa documentos que no contienen el campo de edad.

Vamos a ver un ejemplo práctico respecto al uso de este índice.

Se crea el siguiente script:

```
db = db.getSiblingDB("test"); // use test
db.scores.drop();
db.scores.insertMany([
  { userid : "newbie" },
  { userid : "abby", score : 82 },
  { userid : "nina", score : 90 }
])
```

Se ejecuta en la consola de Mongo Shell y se verifica la colección ordenada en forma descendente por el campo **score**.

```
> load('scripts/spare.js')
true

> db.scores.find().sort({score: -1})
{ "_id" : ObjectId("626d44e3e4c3d772e3c263af"), "userid" : "nina", "score" : 90 }
{ "_id" : ObjectId("626d44e3e4c3d772e3c263ae"), "userid" : "abby", "score" : 82 }
{ "_id" : ObjectId("626d44e3e4c3d772e3c263ad"), "userid" : "newbie" }
>
```

Ahora se crea un índice spare al campo **score**

```
db.scores.createIndex( { score: 1 } , { sparse: true } )
```

```
> db.scores.createIndex( { score: 1 } , { sparse: true } )
{
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "createdCollectionAutomatically" : false,
 "ok" : 1
}
```

Los documentos en los que no exista el campo **score** no serán indexados. Si se realiza nuevamente la misma consulta, se verá que el índice **spare** no elimina los datos completos en una consulta normal. Los documentos que no contienen el campo **score** no están indexados.

```
> db.scores.find().sort({score: -1})
{ "_id" : ObjectId("626d44e3e4c3d772e3c263af"), "userid" : "nina", "score" : 90 }
{ "_id" : ObjectId("626d44e3e4c3d772e3c263ae"), "userid" : "abby", "score" : 82 }
{ "_id" : ObjectId("626d44e3e4c3d772e3c263ad"), "userid" : "newbie" }
>
```

Si se desea incluir de esta búsqueda sólo los documentos indexados con spare se utiliza al final el método **hint()**

```
> db.scores.find().sort({score: -1}).hint({ score : 1}) { "_id" :
ObjectId("626d44e3e4c3d772e3c263af"), "userid" : "nina", "score" : 90 } { "_id" :
ObjectId("626d44e3e4c3d772e3c263ae"), "userid" : "abby", "score" : 82 }
>
```

Los documentos en los que no exista el campo **score** no serán indexados.

Si se realiza, nuevamente, la misma consulta se verá que el índice **spare** no elimina los datos completos en una consulta normal, sólo que los documentos que no contienen el campo **score** no están indexados.

```
> db.scores.find().sort({score: -1})
{ "_id" : ObjectId("626d44e3e4c3d772e3c263af"), "userid" : "nina", "score" : 90 }
{ "_id" : ObjectId("626d44e3e4c3d772e3c263ae"), "userid" : "abby", "score" : 82 }
{ "_id" : ObjectId("626d44e3e4c3d772e3c263ad"), "userid" : "newbie" }
>
```

Para incluir de esta búsqueda sólo los documentos indexados con **spare**, se utiliza al final el método **hint()**

```
> db.scores.find().sort({score: -1}).hint({ score : 1})
{ "_id" : ObjectId("626d44e3e4c3d772e3c263af"), "userid" : "nina", "score" : 90 }
{ "_id" : ObjectId("626d44e3e4c3d772e3c263ae"), "userid" : "abby", "score" : 82 }
>
```

5. **Índice no sensible a mayúsculas y minúsculas**
   (*Case insensitive index)*

Los índices insensibles a mayúsculas y minúsculas se utilizan para admitir consultas que ejecutan comparaciones de cadenas sin ningún tipo de distinción entre mayúsculas y minúsculas.

Ejemplo:

```
db.collection.createIndex ({"clave": 1},
{collation: { locale: 'en_US', strength: 2 }}
)
```

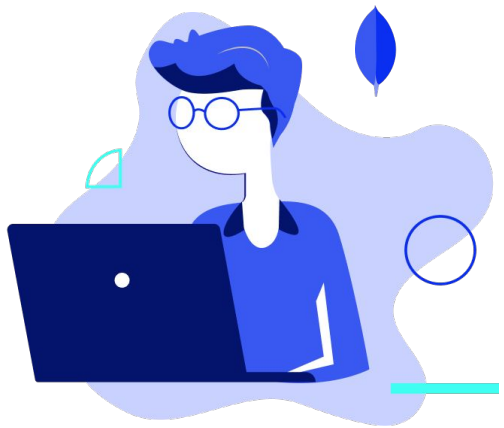
En el ejemplo, **`collation`** indica el **documento de clasificación,** se utiliza para especificar reglas de idioma, mayúsculas, y más detalles, **para la comparación de cadenas.**

Los documentos de clasificación consisten en:

```
{
   locale: <string>,
   caseLevel: <boolean>,
   caseFirst: <string>,
   strength: <int>,
   numericOrdering: <boolean>,
   alternate: <string>,
   maxVariable: <string>,
   backwards: <boolean>
}
```

Un ejemplo práctico respecto al uso de este índice. Se crea el siguiente script:

```
db = db.getSiblingDB("test"); // use test
db.fruit.drop();
db.fruit.insertMany([
  { type: "apple" },
  { type: "Apple" },
  { type: "APPLE" }
])
```



Se ejecuta en la consola de Mongo Shell y se crea un índice:

```
> load('scripts/case.js')
true

> db.fruit.createIndex( { type: 1}, { collation: { locale: 'en', strength: 2 } } )
{
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "createdCollectionAutomatically" : false,
 "ok" : 1
}
```

Cuando hacemos la consulta sin utilizar el índice, obtenemos un resultado:

```
> db.fruit.find( { type: "apple" } )
{ "_id" : ObjectId("626d4842e4c3d772e3c263b0"), "type" : "apple" }
```

Si se realiza, ahora, la misma consulta con el índice `Case Insensitive`, se obtienen los tres resultados:

```
> db.fruit.find( { type: "apple" } ).collation( { locale: 'en', strength: 2 } )
{ "_id" : ObjectId("626d4842e4c3d772e3c263b0"), "type" : "apple" }
{ "_id" : ObjectId("626d4842e4c3d772e3c263b1"), "type" : "Apple" }
{ "_id" : ObjectId("626d4842e4c3d772e3c263b2"), "type" : "APPLE" }
```

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!