Manual MongoDB

Iker Martín

ÍNDICE

0.- Contexto

Este manual está desarrollado con el único objetivo de recopilar las consultas o comandos más relevantes para el lenguaje de bases de datos NoSQL.

Esta documentación no está pensada para el aprendizaje autónomo sin bases previas a dicho lenguaje.

No es un documento oficial, por lo que es posible que contenga algún error o inexactitud a la hora de especificar algún comando o línea de código.

1.- Mostrar documentos

Mostrar todos los documentos de la colección especificada.

```
db.users.find()
```

2.- Composición del Object ID

Cuando se crea un documento, se le asigna un ObjectId con la siguiente información:

1.- Timestamp (fecha actual)

2.- Identificador para el servidor

3.- PID (ID del proceso)

4.- AutoIncrement

3.- Insertar documentos

3.1.- Insert One

Insertar un único documento

```
db.users.insertOne({
        nombre: “Pedro”,
        edad: 18,
        email: “pedro@gmail.com”
}
```

3.2.- Insert Many

Insertar varios documentos a la vez.

```
db.users.insertMany([
        {
                nombre: “María”,
                edad: 18,
                email: “maria@gmail.com”
        },
        {
                nombre: “Aitor”,
                edad: 19,
                email: “aitor@gmail.com”
        }
])
```

4.- Modificar documentos

4.1.- Save

Si el objeto no existe se crea. En cambio, si el objeto existe se actualiza.

```
db.users.save({
        nombre: “Mikel”,
        edad: 25,
        email: “mikel@gmail.com”
}
```

4.2.- Update One

Modificar un documento, especificando tanto la condición de búsqueda como el campo a modificar.

```
db.users.updateOne(
        { nombre: “Mikel” },
        { $set: { edad: 30 } }
)
```

4.3.- Update Many

Modificar varios documentos, especificando el criterio de búsqueda.

```
db.users.updateMany(
        { nombre: “Mikel” },
        { $set: { edad: 30 } }
)
```

```
db.users.update(
        { nombre: “Mikel” },
        { $set: { edad: 30 } },
        { multi: true }
)
```

4.4.- Replace One

Sustituye un documento por otro totalmente nuevo, siempre y cuando cumpla con los criterios de búsqueda.

```
db.users.replaceOne(
        { nombre: “Mikel” },
        { nombre: “Aitor”, edad: 19 }
)
```

5.- Eliminar documentos

5.1.- Delete One

Elimina el primer resultado que coincide con la condición especificada.

```
db.users.deleteOne({
        nombre: “Aitor”
 })
```

5.2.- Delete Many

Elimina los resultados que coinciden con la condición especificada.

```
db.users.deleteMany({
        nombre: “Pedro”
 })
```

6.- Filtrando por condiciones

Se utiliza find para realizar consultas de búsqueda.

Find devuelve un cursor.

```
db.users.find({ nombre: “María” })
```

En cambio, findOne devuelve un documento.

```
db.users.findOne({ nombre: “María” })
```

Se puede poner **count()** al final, para mostrar el número de resultados.

7.- Expresiones de comparación

7.1.- Mayor que

Compara un valor numérico mediante la expresión “mayor que”.

```
db.users.find({ edad: { $gt: 10} })
```

7.2.- Menor que

Compara un valor numérico mediante la expresión “menor que”.

```
db.users.find({ edad: { $lt: 10} })
```

7.3.- Mayor o igual que

Compara un valor numérico mediante la expresión “mayor o igual”.

```
db.users.find({ edad: { $gte: 10} })
```

7.4.- Menor o igual que

Compara un valor numérico mediante la expresión “menor o igual”.

```
db.users.find({ edad: { $lte: 10} })
```

7.5.- Diferente a / no es igual

Buscar los resultados cuando un campo concreto “no es igual” a un valor concreto.

```
db.users.find({
        edad: { $ne: 18 }
 })
```

8.- Expresiones de búsqueda

8.1.- Expresión $and

Varias condiciones en una sola búsqueda. Todas se deben cumplir.

```
db.users.find({
        $and: [
                { edad: {$gt: 10} },
                { estado: “Activo” }
        ]
})
```

8.2.- Expresión $or

Buscar los resultados cuando una de las condiciones se cumple. Este caso se utiliza cuando hay pocos valores a revisar.

```
db.users.find({
        $or: [
                { edad: 27 },
                { edad: 18 }
        ]
 })
```

8.3.- Expresión $in

Permite buscar sobre un listado.

```
db.users.find({
        edad: { $in: [27, 18, 11] }
})
```

8.4.- Expresión $exists

Busca los resultados cuando estos tienen el campo especificado, independientemente de su valor.

```
db.users.find({
        estado: { $exists: true }
})
```

8.5.- Expresión $type

Busca los resultados cuando el campo especificado coincide con el tipo especificado.

```
db.users.find({
        estado: { $type: “string” }
})
```

9.- Cursor sort

Muestra todos los resultados, ordenados de mayor a menos por un campo especificado.

```
db.users.find.sort({
        edad: -1
})
```

Orden ascendente: 1

Orden descendente: -1

10.- Cursor limit

Limita el número total de resultados mostrados en una búsqueda.

```
db.users.find.sort({
        edad: -1
}).limit(1)
```

11.- Expresiones regulares

Busca los resultados que tengan ".com" al final del email.

```
db.users.find.sort({ email: /.com$/ })
```

Busca los resultados que tengan "user" al inicio del email.

```
db.users.find.sort({ email: /^user/ })
```

Busca los resultados que tengan "@" dentro del email.

```
db.users.find.sort({ email: /@/ })
```

12.- Proyecciones

Además de mostrar los resultados de la consulta, también permite indicar que campos se van a mostrar y ocultar.

```
db.users.find({ }, { _id: 0,  nombre: 1, edad: 1 })
```

Ocultar campo: 0

Mostrar campo: 1