

# MongoDB Fundamentos

Módulo 3

# Cursores en MongoDB

# Cursores en MongoDB

## Acceder a datos desde un cursor

Las operaciones de lectura que devuelven varios documentos no traen inmediatamente todos los valores que coinciden con la consulta. Debido a que una consulta puede coincidir potencialmente con conjuntos muy grandes de documentos, estas operaciones se basan en un objeto denominado **cursor**. Un **cursor** recupera los documentos por lotes para reducir tanto el consumo de memoria como el uso del ancho de banda de la red.

Los cursores son altamente configurables y ofrecen múltiples **paradigmas de interacción** para diferentes casos de uso.

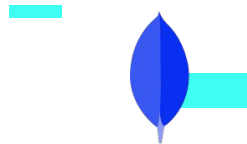


Las siguientes funciones devuelven cursores directamente:

- `Collection.find()`
- `Collection.aggregate()`
- `Collection.listIndexes()`
- `Db.aggregate()`
- `Db.listCollections()`

En MongoDB, cuando se realiza una consulta (**`.find()`**) sobre una colección, se obtiene como resultado un cursor de documentos. Para ver los documentos que contiene, es necesario recorrerlo.

Esto se puede realizar con la ayuda de los métodos presentes en el cursor como: **`forEach`**, **`hasNext`**, **`next`** y **`toArray`**.

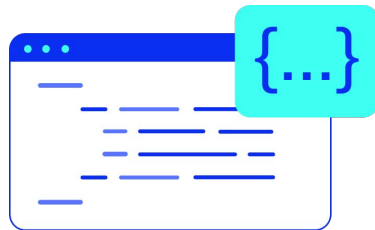


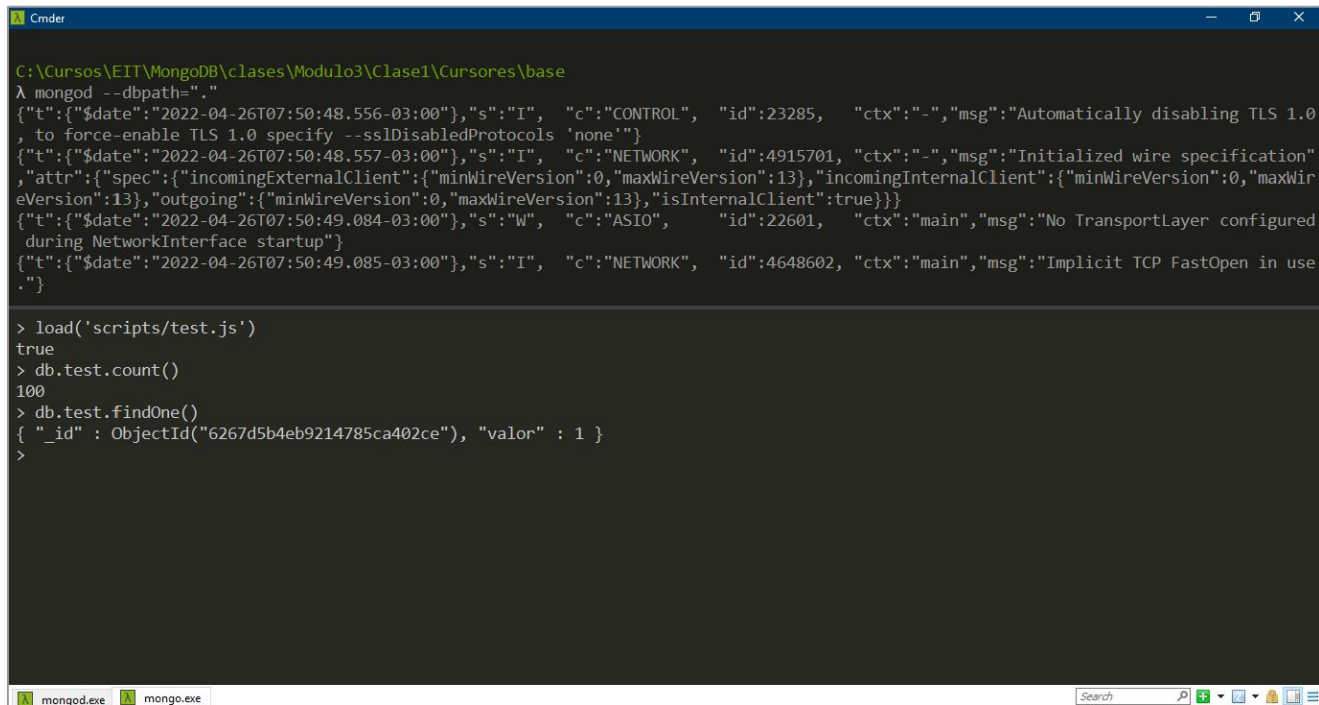
## Cursores y sus métodos en MongoDB

Se crea el siguiente script para realizar una inserción de 100 documentos de test:

```
for(var i=1; i<=100; i++) {  
    db.test.insert({valor: i})  
}
```

Se ejecuta en una consola de MongoDB para revisar la cantidad de documentos que contiene la colección **test**. Con un **findOne()** se verifica el primer documento creado.

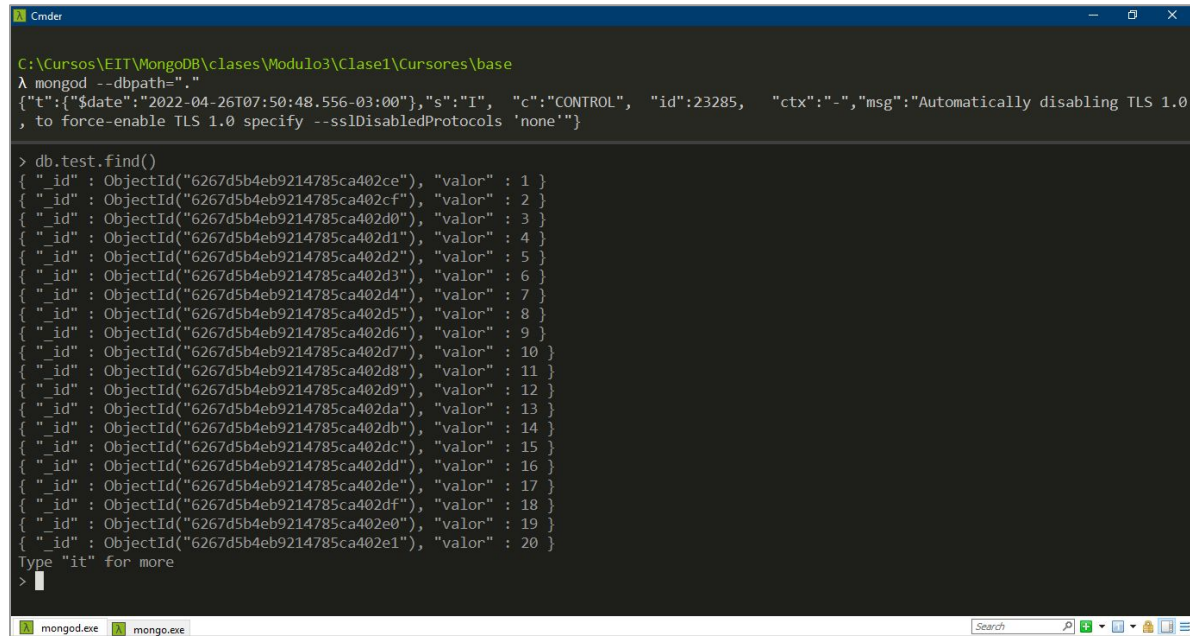




```
C:\Cursos\EIT\MongoDB\clases\Modulo3\Clase1\Cursores\base
λ mongod --dbpath="."
{"t":{"$date":"2022-04-26T07:50:48.556-03:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"-", "msg":"Automatically disabling TLS 1.0
, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2022-04-26T07:50:48.557-03:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"-", "msg":"Initialized wire specification
", "attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":13},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isInternalClient":true}}}
{"t":{"$date":"2022-04-26T07:50:49.084-03:00"},"s":"W",  "c":"ASIO",      "id":22601,   "ctx":"main", "msg":"No TransportLayer configured
during NetworkInterface startup"}
{"t":{"$date":"2022-04-26T07:50:49.085-03:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"main", "msg":"Implicit TCP FastOpen in use
."}

> load('scripts/test.js')
true
> db.test.count()
100
> db.test.findOne()
{ "_id" : ObjectId("6267d5b4eb9214785ca402ce"), "valor" : 1 }
>
```

Luego se realiza una consulta con **find()** sobre la misma consola:



```
C:\Cursos\EIT\MongoDB\clases\Modulo3\Clase1\Cursores\base
λ mongod --dbpath="."
{"t":{"$date":"2022-04-26T07:50:48.556-03:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"-", "msg":"Automatically disabling TLS 1.0
, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}

> db.test.find()
{ "_id" : ObjectId("6267d5b4eb9214785ca402ce"), "valor" : 1 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402cf"), "valor" : 2 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d0"), "valor" : 3 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d1"), "valor" : 4 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d2"), "valor" : 5 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d3"), "valor" : 6 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d4"), "valor" : 7 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d5"), "valor" : 8 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d6"), "valor" : 9 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d7"), "valor" : 10 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d8"), "valor" : 11 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d9"), "valor" : 12 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402da"), "valor" : 13 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402db"), "valor" : 14 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402dc"), "valor" : 15 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402dd"), "valor" : 16 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402de"), "valor" : 17 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402df"), "valor" : 18 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402e0"), "valor" : 19 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402e1"), "valor" : 20 }
Type "it" for more
>
```

Cada vez que se llama al método **find** de una colección, **retorna un objeto de la clase Cursor**.

Si no se asigna ese valor a una variable en el shell de MongoDB, se muestran los primeros 20 documentos recuperados y se pide que se confirme, mediante **it**, si se desea ver los 20 que siguen.

Lo mismo sucede si se almacena la consulta en una variable cursor y se invoca en la consola. Veamos la siguiente slide.



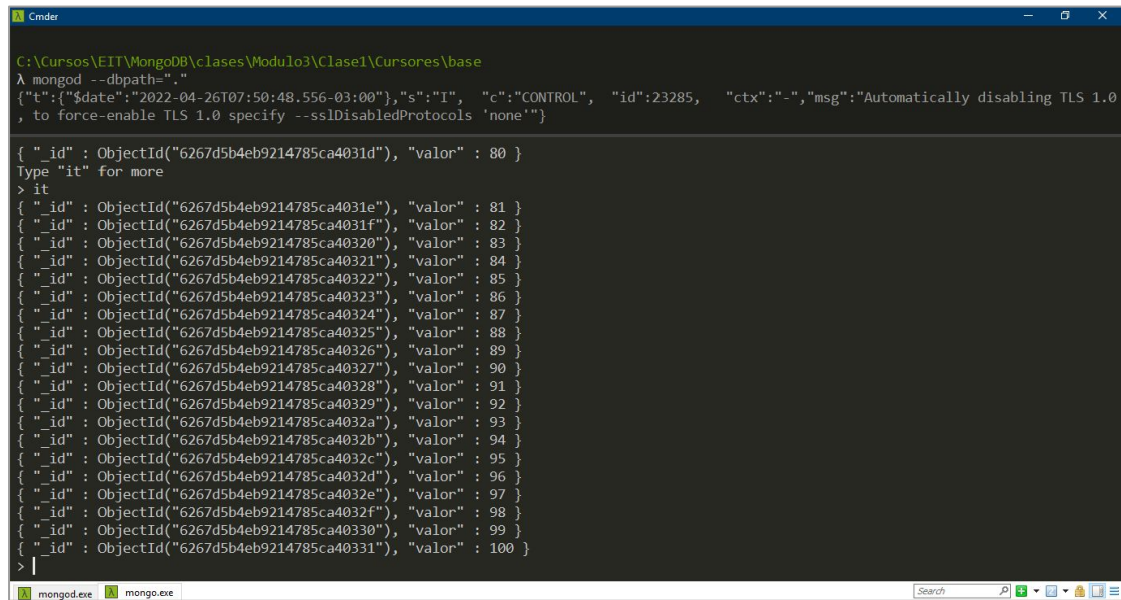


```
C:\Cursos\EIT\MongoDB\clases\Modulo3\Clase1\Cursore\base
λ mongod --dbpath="."
{"t":{"$date":"2022-04-26T07:50:48.556-03:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"-", "msg":"Automatically disabling TLS 1.0
, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}

> var cursor = db.test.find()
> cursor
{ "_id" : ObjectId("6267d5b4eb9214785ca402ce"), "valor" : 1 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402cf"), "valor" : 2 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d0"), "valor" : 3 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d1"), "valor" : 4 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d2"), "valor" : 5 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d3"), "valor" : 6 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d4"), "valor" : 7 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d5"), "valor" : 8 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d6"), "valor" : 9 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d7"), "valor" : 10 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d8"), "valor" : 11 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402d9"), "valor" : 12 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402da"), "valor" : 13 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402db"), "valor" : 14 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402dc"), "valor" : 15 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402dd"), "valor" : 16 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402de"), "valor" : 17 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402df"), "valor" : 18 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402e0"), "valor" : 19 }
{ "_id" : ObjectId("6267d5b4eb9214785ca402e1"), "valor" : 20 }
Type "it" for more
>
```

Luego de llamar cuatro veces a **it**, se ven todos los resultados. De esta forma se “agota el cursor”, ya no tiene más datos para ofrecer.

Para buscar nuevamente con **find()**, se debe recargar el cursor.



```
C:\Cursos\EIT\MongoDB\clases\Modulo3\Clase1\Cursores\base
λ mongo --dbpath="."
{"t":{"$date":"2022-04-26T07:50:48.556-03:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"-", "msg":"Automatically disabling TLS 1.0
, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}

{ "_id" : ObjectId("6267d5b4eb9214785ca4031d"), "valor" : 80 }
Type "it" for more
> it
{ "_id" : ObjectId("6267d5b4eb9214785ca4031e"), "valor" : 81 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4031f"), "valor" : 82 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40320"), "valor" : 83 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40321"), "valor" : 84 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40322"), "valor" : 85 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40323"), "valor" : 86 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40324"), "valor" : 87 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40325"), "valor" : 88 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40326"), "valor" : 89 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40327"), "valor" : 90 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40328"), "valor" : 91 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40329"), "valor" : 92 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032a"), "valor" : 93 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032b"), "valor" : 94 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032c"), "valor" : 95 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032d"), "valor" : 96 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032e"), "valor" : 97 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032f"), "valor" : 98 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40330"), "valor" : 99 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40331"), "valor" : 100 }
>
```

## Método forEach del cursor

Mediante el método **forEach** es posible recorrer todo el contenido del cursor en forma secuencial hasta agotarlo. Para realizar otra lectura, se debe recargar el cursor.

Se crea el script que vemos a la derecha:

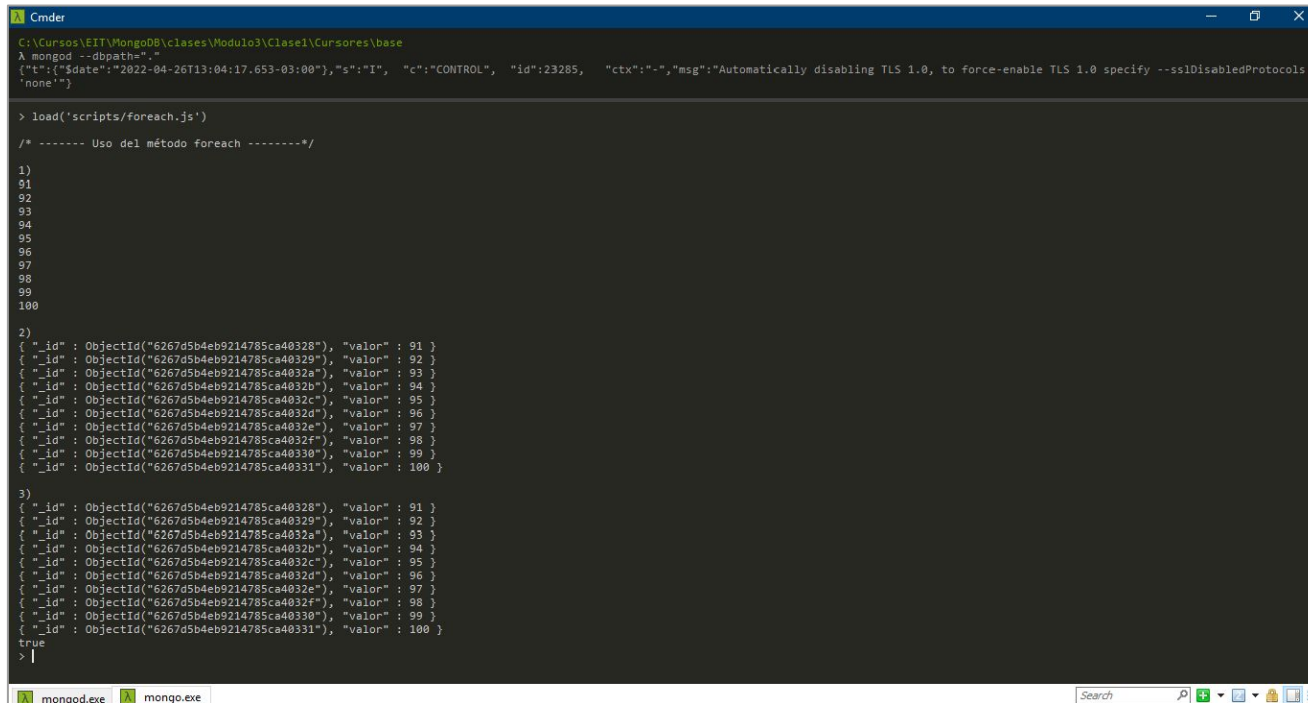
```
print('\n/* ----- Uso del método foreach
-----*/')

print('\n1')
var cursor = db.test.find({ valor: {$gt:90} })
cursor.forEach(function(d) { print(d.valor) });

print('\n2')
var cursor = db.test.find({ valor: {$gt:90} })
cursor.forEach(function(d) { printjson(d) });

print('\n3')
var cursor = db.test.find({ valor: {$gt:90} })
cursor.forEach(function(d) { print(tojson(d)) });
```

Al ejecutarlo, se obtiene:



```
C:\Cursos\EIT\MongoDB\clases\Modulo3\Clase1\Cursos\base
λ mongod --dbpath="."
{"t":{"sdate":"2022-04-26T13:04:17.653-03:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}

> load('scripts/foreach.js')

/* ----- Uso del método foreach ----- */

1)
91
92
93
94
95
96
97
98
99
100

2)
{ "_id" : ObjectId("6267d5b4eb9214785ca40328"), "valor" : 91 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40329"), "valor" : 92 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032a"), "valor" : 93 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032b"), "valor" : 94 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032c"), "valor" : 95 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032d"), "valor" : 96 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032e"), "valor" : 97 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032f"), "valor" : 98 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40330"), "valor" : 99 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40331"), "valor" : 100 }

3)
{ "_id" : ObjectId("6267d5b4eb9214785ca40328"), "valor" : 91 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40329"), "valor" : 92 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032a"), "valor" : 93 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032b"), "valor" : 94 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032c"), "valor" : 95 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032d"), "valor" : 96 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032e"), "valor" : 97 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032f"), "valor" : 98 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40330"), "valor" : 99 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40331"), "valor" : 100 }
true
> |
```

## Métodos hasNext y Next

Se usan en forma combinada para acceder al contenido de un cursor en forma secuencial hasta agotarlo.

### `.hasNext()`

Es un método que permite saber si quedan documentos por recorrer en el cursor.



## .next()

Es un método que va moviendo el cursor y permite iterar por el cursor de documentos. Se crea el siguiente script:

```
print('\n/* ----- Uso de los métodos hasNext y next -----*/')
print('\n')

var cursor = db.test.find({ valor: {$gt:90} })

while(cursor.hasNext()) {
    printjson(cursor.next())
}
```

Veamos en la próxima slide qué obtenemos al ejecutarlo en Mongo Shell.

```
cmdr
C:\Cursos\EIT\MongoDB\clases\Modulo3\Clase1\Cursos\base
λ mongod --dbpath="."

> load('scripts/next.js')

/* ----- Uso de los métodos hasNext y next -----*/

{ "_id" : ObjectId("6267d5b4eb9214785ca40328"), "valor" : 91 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40329"), "valor" : 92 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032a"), "valor" : 93 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032b"), "valor" : 94 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032c"), "valor" : 95 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032d"), "valor" : 96 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032e"), "valor" : 97 }
{ "_id" : ObjectId("6267d5b4eb9214785ca4032f"), "valor" : 98 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40330"), "valor" : 99 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40331"), "valor" : 100 }
true
> |
```

## Método toArray

Si ahora se quiere acceder en forma aleatoria al contenido de un cursor, es posible utilizar el método **toArray** de los cursores. Si se indexa directamente el cursor, se hará uso del **toArray**

en forma implícita. En estos casos, con **[1]** accedemos al segundo documento almacenado en el cursor. Creamos el siguiente script:

```
print('\n/* ----- Uso del método toArray ----- */')
print('\n')

var cursor = db.test.find({ valor: { $gt: 90 } })

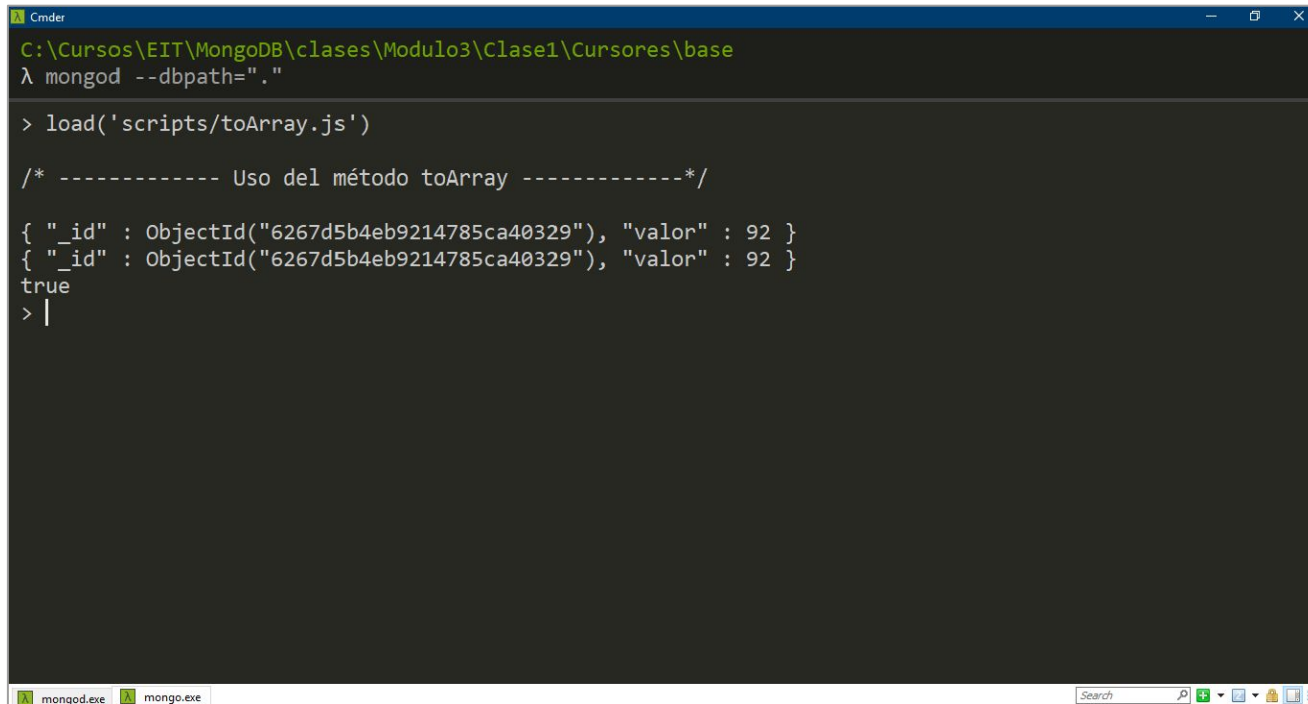
var documentArray = cursor.toArray()
var document = documentArray[1]
printjson(document)

var document = cursor[1]
printjson(document)
```





Al ejecutarlo en Mongo Shell, se obtiene:



```
C:\Cursos\EIT\MongoDB\clases\Modulo3\Clase1\Cursores\base
λ mongod --dbpath="."

> load('scripts/toArray.js')

/* ----- Uso del método toArray -----*/

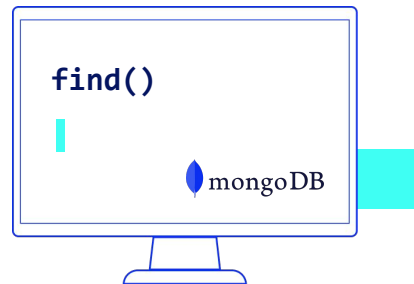
{ "_id" : ObjectId("6267d5b4eb9214785ca40329"), "valor" : 92 }
{ "_id" : ObjectId("6267d5b4eb9214785ca40329"), "valor" : 92 }
true
> |
```

## Otros métodos del cursor

Existen otros métodos que permiten ser aplicados a las consultas **find** como:

**sort**, **pretty**, **limit**, **skip**, **count**, **size** y **noCursorTimeout** entre otros.

Se creará una colección **personas** con documentos de test para demostrar esos métodos. Para esto, se crea y ejecuta el script de la próxima pantalla llamado **personas.js** en la consola Mongo Shell.



```
db = db.getSiblingDB("test"); // use test
db.personas.drop();
db.personas.insertMany([
  { "id": "1", "nombre": "Sigmund", "edad": 63, "acceso": true, "_id": "1" },
  { "id": "2", "nombre": "Lulu", "edad": 48, "acceso": false, "_id": "2" },
  { "id": "3", "nombre": "Katarina", "edad": 94, "acceso": true, "_id": "3" },
  { "id": "4", "nombre": "Nedra", "edad": 12, "acceso": false, "_id": "4" },
  { "id": "5", "nombre": "Shaina", "edad": 70, "acceso": false, "_id": "5" },
  { "id": "6", "nombre": "Retha", "edad": 59, "acceso": false, "_id": "6" },
  { "id": "7", "nombre": "Salvador", "edad": 70, "acceso": false, "_id": "7" },
  { "id": "8", "nombre": "Ilene", "edad": 84, "acceso": false, "_id": "8" },
  { "id": "9", "nombre": "Michelle", "edad": 93, "acceso": false, "_id": "9" },
  { "id": "10", "nombre": "Veronica", "edad": 64, "acceso": false, "_id": "10" },
  { "id": "11", "nombre": "Christophe", "edad": 80, "acceso": false, "_id": "11" },
  { "id": "12", "nombre": "Sarah", "edad": 76, "acceso": false, "_id": "12" },
  { "id": "13", "nombre": "Greg", "edad": 16, "acceso": false, "_id": "13" },
  { "id": "14", "nombre": "Piper", "edad": 18, "acceso": false, "_id": "14" },
```

...

```
{ "id": "15", "nombre": "Mavis", "edad": 94, "acceso": false, "_id": "15" },  
{ "id": "16", "nombre": "Gunner", "edad": 30, "acceso": false, "_id": "16" },  
{ "id": "17", "nombre": "Maxie", "edad": 65, "acceso": false, "_id": "17" },  
{ "id": "18", "nombre": "Alexie", "edad": 91, "acceso": false, "_id": "18" },  
{ "id": "19", "nombre": "Arthur", "edad": 3, "acceso": false, "_id": "19" }  
])
```

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

Luego, mediante otros scripts, se probarán de manera sistemática cada uno de los métodos citados al principio.

## find con query y projection

```
print('\n/* ----- find query projection -----*/')
print('\n')

var cursor = db.personas
    .find(
        { edad: {$gt:50} },           // filtro query
        { nombre: 1, edad: 1, _id:0 } // projection -> 1: sale, 0: no sale
    )

// Represento la información devuelta por el cursor
while(cursor.hasNext()) print(tojson(cursor.next()))
```

El script busca todas las personas mayores a 50 años y expone los campos **nombre** y **edad** de la consulta.

```
> load('scripts/metodos.js')

/* ----- find query projection -----*/

{ "nombre" : "Sigmund", "edad" : 63 }
{ "nombre" : "Katarina", "edad" : 94 }
{ "nombre" : "Shaina", "edad" : 70 }
{ "nombre" : "Retha", "edad" : 59 }
{ "nombre" : "Salvador", "edad" : 70 }
{ "nombre" : "Ilene", "edad" : 84 }
{ "nombre" : "Michelle", "edad" : 93 }
{ "nombre" : "Veronica", "edad" : 64 }
{ "nombre" : "Christophe", "edad" : 80 }
{ "nombre" : "Sarah", "edad" : 76 }
{ "nombre" : "Mavis", "edad" : 94 }
{ "nombre" : "Maxie", "edad" : 65 }
{ "nombre" : "Alexie", "edad" : 91 }
true
> |
```

## Método sort

A partir del cursor que retorna el método **find**, se llama al método **sort** de la clase Cursor y, como condición, se indica el campo por el que se desea

ordenar (si se pasa un 1 se ordena en forma ascendente y si se pasa un -1 se ordena en forma descendente):

```
print('\n/* ----- Método sort -----*/')
print('\n')

var cursor = db.personas
    .find(
        { edad: {$gt:50} },           // filtro query
        { nombre: 1, edad: 1, _id:0 } // projection -> 1: sale, 0: no sale
    )
    .sort({edad: -1, nombre: 1})

// Represento la información devuelta por el cursor
while(cursor.hasNext()) print(tojson(cursor.next()))
```

Se obtiene:

```
> load('scripts/metodos.js')

/* ----- find query projection -----*/

{ "nombre" : "Katarina", "edad" : 94 }
{ "nombre" : "Mavis", "edad" : 94 }
{ "nombre" : "Michelle", "edad" : 93 }
{ "nombre" : "Alexie", "edad" : 91 }
{ "nombre" : "Ilene", "edad" : 84 }
{ "nombre" : "Christophe", "edad" : 80 }
{ "nombre" : "Sarah", "edad" : 76 }
{ "nombre" : "Salvador", "edad" : 70 }
{ "nombre" : "Shaina", "edad" : 70 }
{ "nombre" : "Maxie", "edad" : 65 }
{ "nombre" : "Veronica", "edad" : 64 }
{ "nombre" : "Sigmund", "edad" : 63 }
{ "nombre" : "Retha", "edad" : 59 }
true
>
```



## Método pretty

El método **sort** también retorna un Cursor con los datos ordenados, luego podemos llamar al método **pretty** a partir del Cursor devuelto por **sort**:

```
> db.personas.find({edad:{$gt:90}}).sort({edad:-1,nombre:1})
{ "_id" : "3", "id" : "3", "nombre" : "Katarina", "edad" : 94, "acceso" : true }
{ "_id" : "15", "id" : "15", "nombre" : "Mavis", "edad" : 94, "acceso" : false }
{ "_id" : "9", "id" : "9", "nombre" : "Michelle", "edad" : 93, "acceso" : false }
{ "_id" : "18", "id" : "18", "nombre" : "Alexie", "edad" : 91, "acceso" : false }
> db.personas.find({edad:{$gt:90}}).sort({edad:-1,nombre:1}).pretty()
{
  "_id" : "3",
  "id" : "3",
  "nombre" : "Katarina",
  "edad" : 94,
  "acceso" : true
}
{
  "_id" : "15",
  "id" : "15",
  "nombre" : "Mavis",
  "edad" : 94,
  "acceso" : false
}
{
  "_id" : "9",
  "id" : "9",
  "nombre" : "Michelle",
  "edad" : 93,
  "acceso" : false
}
{
  "_id" : "18",
  "id" : "18",
  "nombre" : "Alexie",
  "edad" : 91,
  "acceso" : false
}
> |
```

## Método limit

Otro método útil es: **limit**, que tiene por objetivo limitar a un determinado número de documentos a recuperar del Cursor. Por ejemplo, si la llamada

al método **find** retorna 4 documentos, se puede aplicar el método **limit** a dicho Cursor para que se limite a recuperar los dos primeros:

```
print('\n/* ----- Método limit -----*/')

var cursor = db.personas
    .find(
        { edad: {$gt:50} },           //filtro query
        { nombre: 1, edad: 1, _id:0 } // projection -> 1: sale, 0: no sale
    )
    .limit(4)

// Represento la información devuelta por el cursor
while(cursor.hasNext()) print(tojson(cursor.next()))
```

El resultado en la consola es:

```
> load('scripts/metodos.js')

/* ----- Método limit -----*/

{ "nombre" : "Sigmund", "edad" : 63 }
{ "nombre" : "Katarina", "edad" : 94 }
{ "nombre" : "Shaina", "edad" : 70 }
{ "nombre" : "Retha", "edad" : 59 }
true
>
```

Nuevamente, tener en cuenta que primero se pueden ordenar los datos del Cursor y llamar a partir de éste al método **limit**.



## Método skip

Otro método llamado **skip** nos permite saltar una determinada cantidad de documentos desde el principio del cursor. Se puede usar en conjunto con **limit**.

```
print('\n/* ----- Método skip -----*/')
print('\n')

var cursor = db.personas
    .find(
        { edad: { $gt: 50 } },           //filtro query
        { nombre: 1, edad: 1, _id: 0 }  // projection -> 1: sale, 0: no sale
    )
    .skip(1)
    .limit(4)

// Represento la información devuelta por el cursor
while(cursor.hasNext()) print(tojson(cursor.next()))
```

Consola:

```
> load('scripts/metodos.js')  
  
/* ----- Método skip -----*/  
  
{ "nombre" : "Katarina", "edad" : 94 }  
{ "nombre" : "Shaina", "edad" : 70 }  
{ "nombre" : "Retha", "edad" : 59 }  
{ "nombre" : "Salvador", "edad" : 70 }  
true  
> |
```



## Método count

```
print('\n/* ----- Método count -----*/')
print('\n')

var cantidad = db.personas
    .find(
        { edad: { $gt: 50 } },           //filtro query
        { nombre: 1, edad: 1, _id: 0 }  // projection -> 1: sale, 0: no sale
    )
    .skip(1)
    .limit(4)
    .count()

print('Cantidad de personas: ' + cantidad)
```

El método `count` devuelve la cantidad de documentos almacenados en el cursor sin considerar la acción de **`skip`** y **`limit`**.

### Consola

```
> load('scripts/metodos.js')  
  
/* ----- Método count -----*/  
  
Cantidad de personas: 13  
true  
>
```

## Método size

En cambio, el método **size** devuelve la cantidad de documentos almacenados en el cursor y considera el filtro por parte de **skip** y **limit**.

```
print('\n/* ----- Método size -----*/')
print('\n')

var cantidad = db.personas
    .find(
        { edad: {$gt:50} },           //filtro query
        { nombre: 1, edad: 1, _id:0 } // projection -> 1: sale, 0: no sale
    )
    .skip(1)
    .limit(4)
    .size()

print('Cantidad de personas: ' + cantidad)
```



## Consola

```
> load('scripts/metodos.js')  
  
/* ----- Método count -----*/  
  
Cantidad de personas: 4  
true  
>
```

## Método noCursorTimeout

Este método le indica al servidor que evite cerrar un cursor automáticamente después de un período de inactividad que se establece en 30

minutos. Puede funcionar en forma combinada con los otros métodos mencionados.

```
print('\n/* ----- Método noCursorTimeout -----*/')
print('\n')

var cursor = db.personas
    .find(
        { edad: { $gt: 50 } },           //filtro query
        { nombre: 1, edad: 1, _id: 0 }  // projection -> 1: sale, 0: no sale
    )
    .sort({edad: -1, nombre: 1})
    .noCursorTimeout() // permite que el cursor no se agote por timeout

// Represento la información devuelta por el cursor
while(cursor.hasNext()) print(tojson(cursor.next()))
```

Salida en consola:

```
> load('scripts/metodos.js')

/* ----- Método noCursorTimeout -----*/

{ "nombre" : "Katarina", "edad" : 94 }
{ "nombre" : "Mavis", "edad" : 94 }
{ "nombre" : "Michelle", "edad" : 93 }
{ "nombre" : "Alexie", "edad" : 91 }
{ "nombre" : "Ilene", "edad" : 84 }
{ "nombre" : "Christophe", "edad" : 80 }
{ "nombre" : "Sarah", "edad" : 76 }
{ "nombre" : "Salvador", "edad" : 70 }
{ "nombre" : "Shaina", "edad" : 70 }
{ "nombre" : "Maxie", "edad" : 65 }
{ "nombre" : "Veronica", "edad" : 64 }
{ "nombre" : "Sigmund", "edad" : 63 }
{ "nombre" : "Retha", "edad" : 59 }
true
> |
```

**¡Sigamos  
trabajando!**

