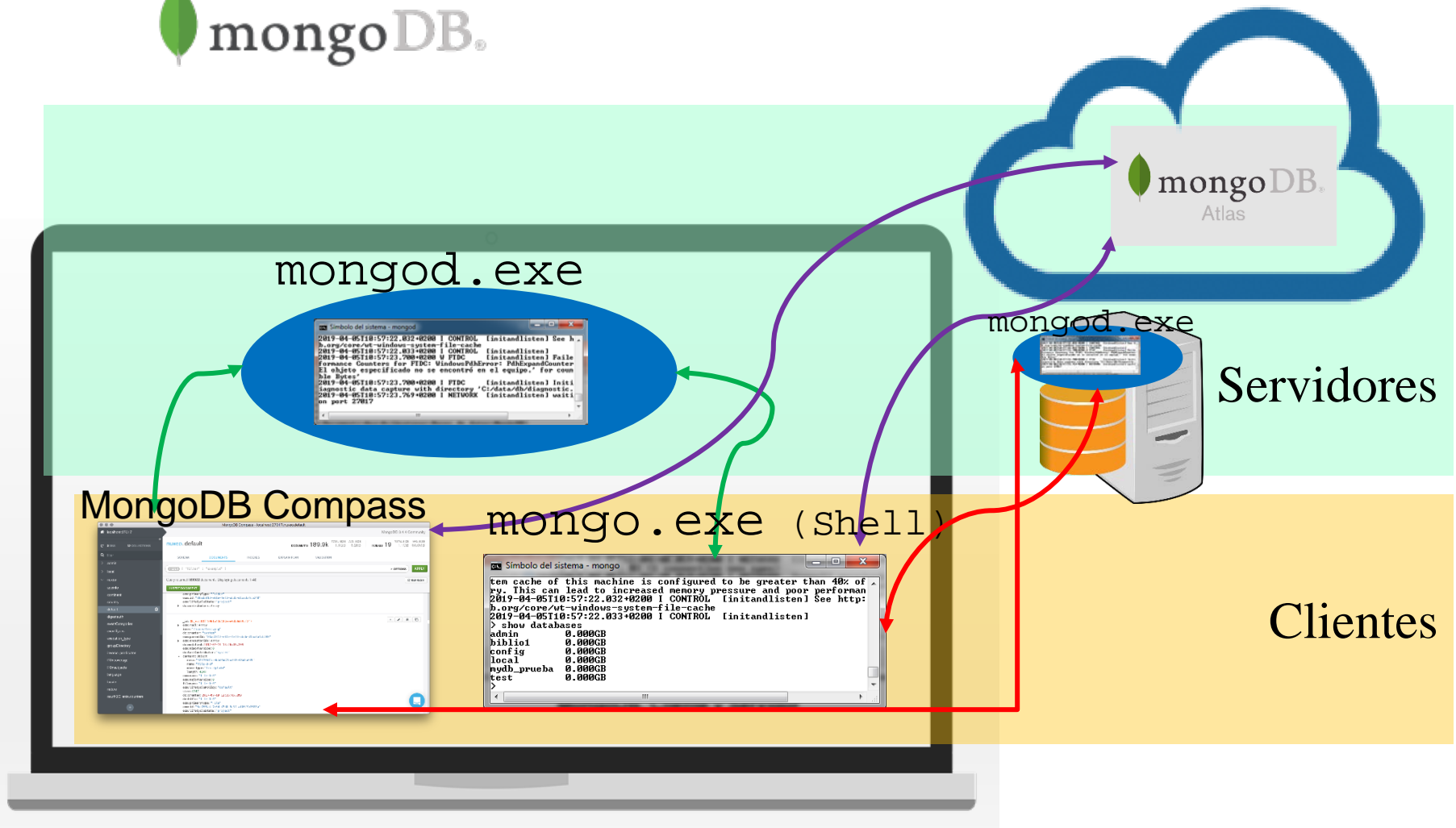


Introducción a MongoDB. Características

- Sistema de almacenamiento orientado a documentos
- Alta velocidad de respuesta
- Alta escalabilidad
- Permite almacenar datos semiestructurados
- Desde la versión 4.0 incluye soporte transaccional
- Lenguaje propio de consultas.
- Almacena documentos **BSON** (*Binay JSON*): representación binaria del formato **JSON** (*JavaScript Object Notation*)



MongoDB. Cliente y Servidor



MongoDB. Modelo de documentos

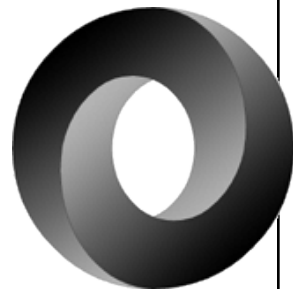
- En MongoDB una Base de Datos está formada por un conjunto de colecciones
- Una colección está formada por un conjunto de documentos
- Un documento es un objeto JSON que contiene información sobre algún elemento de interés

MONGODB	RELACIONAL
Base de Datos	Base de Datos
Colección	Tabla
Documento	Fila o tupla
Campo	Columna

Formato JSON

- Permite definir objetos complejos fácilmente mediante texto plano como un conjunto de pares *nombre : valor*
- Un **objeto** se define entre llaves { } que contienen una sucesión de pares *nombre : valor* separados por comas

```
{ "CODIGO" : 35 , "NOMBRE" : "FERNANDO" ,  
  "APELLIDO" : "DE ROJAS" , "ANO_NAC" : 1470 ,  
  "ANO_FALL" : 1541 , "NACION" : "ESPAÑA" }
```



- Un **nombre** es un *string* (cadena de caracteres escrita entre dobles comillas “ “, aunque éstas son obviables)
- Un **valor** puede ser un *string*, un número, un objeto, un array o los literales `true`, `false` y `null`
- Especificaciones del formato: <https://www.json.org/json-es.html>

Formato JSON

- Un **array** es una colección ordenada de valores separados por comas y escrita entre corchetes []

```
{ "CODIGO": 98, "NOMBRE": "MARIO", "APELLIDO": "VARGAS  
LLOSA", "ANO_NAC": 1936,  
  "NACION": [ "ESPAÑA", "PERU" ] ,  
  "LIBROS": [ "5024176", "5025160" ] }
```

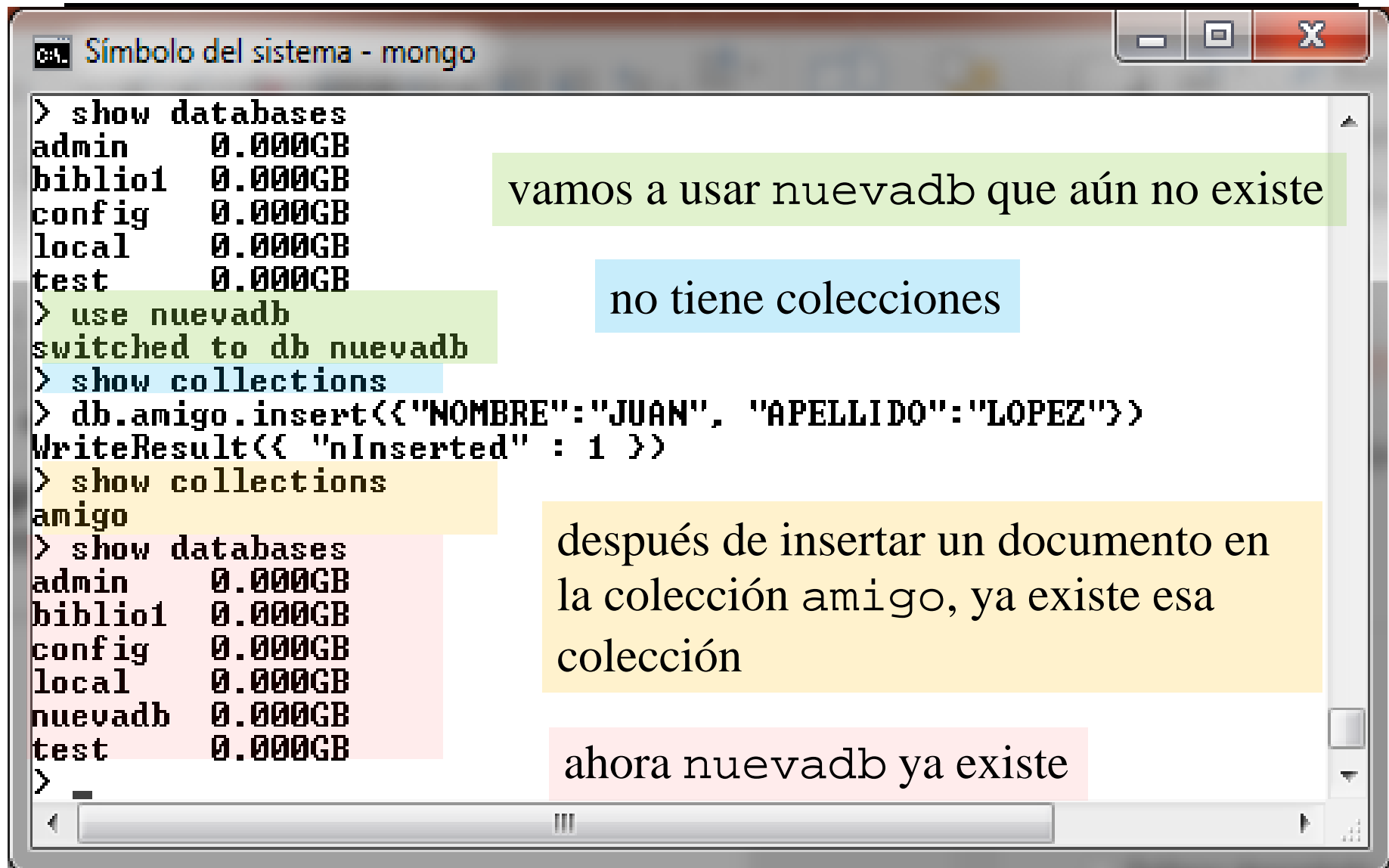
```
-----  
{ "CODIGO": 98, "NOMBRE": "MARIO", "APELLIDO": "VARGAS  
LLOSA", "ANO_NAC": 1936,  
  "NACION": [ "ESPAÑA", "PERU" ] ,  
  "LIBROS": [  
    { "ISBN": "5024176", "TITULO": "EL SUEÑO DEL CELTA" },  
    { "ISBN": "5025160", "TITULO": "LITUMA EN LOS ANDES" } ] }
```

MongoDB. Comandos básicos

Comando	Acción
<code>show databases</code>	Lista las bases de datos
<code>db</code>	Muestra la base de datos actual
<code>show collections</code>	Muestra las colecciones de la base de datos actual
<code>use <i>nombrebd</i></code>	Selecciona una base de datos para utilizarla

- No es necesario definir las bases de datos ni las colecciones
- Ambas se crean si no existen cuando se usan
- Las colecciones, en principio, no deben ajustarse a ninguna estructura predefinida

MongoDB. Comandos básicos



```

> show databases
admin      0.000GB
biblio1    0.000GB
config     0.000GB
local      0.000GB
test       0.000GB
> use nuevadb
switched to db nuevadb
> show collections
> db.amigo.insert(<<"NOMBRE":"JUAN", "APELLIDO":"LOPEZ">>)
WriteResult(< "nInserted" : 1 >)
> show collections
amigo
> show databases
admin      0.000GB
biblio1    0.000GB
config     0.000GB
local      0.000GB
nuevadb    0.000GB
test       0.000GB
>

```

vamos a usar nuevadb que aún no existe

no tiene colecciones

después de insertar un documento en la colección amigo, ya existe esa colección

ahora nuevadb ya existe

Operaciones CRUD. Escritura

INSERCIÓN DE DOCUMENTOS

```
db.nombrecoleccion.insert(documento)
```

Inserta el documento en la colección indicada. Si la colección no existe en la base de datos en uso, la crea

Todo documento tiene un campo **_id** con valor único en la colección. Si no se especifica, MongoDB lo crea automáticamente

EJEMPLO:

```
db.AMIGO.insert( { "NOMBRE": "JOSE", "APELLIDO": "PEREZ",  
"EDAD": 23 } )
```

Inserta el siguiente documento en la colección AMIGO:

```
{ "_id" : ObjectId("5cab31dbfbbf7fff42d2995f"),  
"NOMBRE" : "JOSE", "APELLIDO" : "PEREZ", "EDAD" : 23 }
```


Operaciones CRUD. Escritura

Campo `_id`

Es un campo obligatorio y con valor único que se asigna a cada objeto de una colección

- Se puede crear de forma manual
- Si no se crea manualmente lo hace automáticamente
- Automáticamente se obtiene con `ObjectId ()` que devuelve un número hexadecimal de 12 bytes
 - los 4 primeros son una marca de tiempo,
 - los 3 siguientes la identificación de la máquina
 - los 2 siguientes identifican el proceso
 - los 3 últimos un contador empezando en un número aleatorio

Operaciones CRUD. Lectura

LECTURA DE DOCUMENTOS

```
db.nombrecoleccion.find()
```

```
db.nombrecoleccion.find(filtro, campos)
```

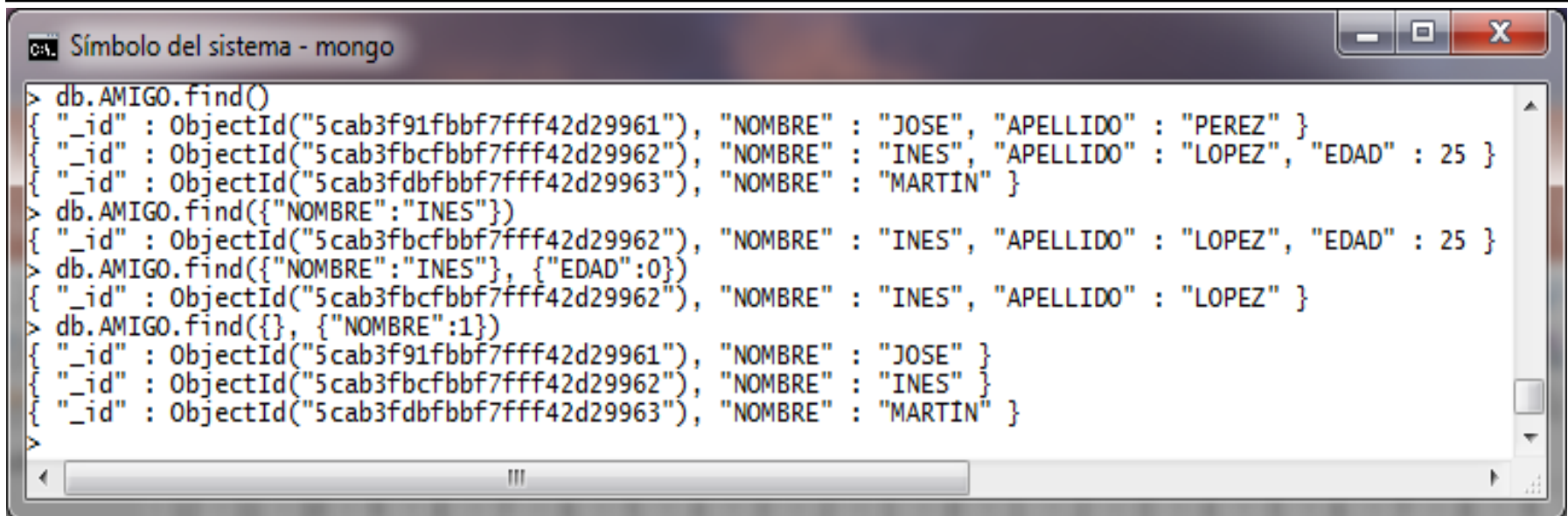
filtro: condición de búsqueda, (pares *nombre:valor* a buscar).

Si se omite (`{ }`) devuelve todos.

campos :campos a mostrar/omitir en la salida.

`{ nombre_campo1:1 }` → muestra el campo

`{ nombre_campo2:0 }` → omite el campo



```
> db.AMIGO.find()
{ "_id" : ObjectId("5cab3f91fbbf7fff42d29961"), "NOMBRE" : "JOSE", "APELLIDO" : "PEREZ" }
{ "_id" : ObjectId("5cab3fbcfbbf7fff42d29962"), "NOMBRE" : "INES", "APELLIDO" : "LOPEZ", "EDAD" : 25 }
{ "_id" : ObjectId("5cab3fdbfbbf7fff42d29963"), "NOMBRE" : "MARTIN" }
> db.AMIGO.find({"NOMBRE":"INES"})
{ "_id" : ObjectId("5cab3fbcfbbf7fff42d29962"), "NOMBRE" : "INES", "APELLIDO" : "LOPEZ", "EDAD" : 25 }
> db.AMIGO.find({"NOMBRE":"INES"}, {"EDAD":0})
{ "_id" : ObjectId("5cab3fbcfbbf7fff42d29962"), "NOMBRE" : "INES", "APELLIDO" : "LOPEZ" }
> db.AMIGO.find({}, {"NOMBRE":1})
{ "_id" : ObjectId("5cab3f91fbbf7fff42d29961"), "NOMBRE" : "JOSE" }
{ "_id" : ObjectId("5cab3fbcfbbf7fff42d29962"), "NOMBRE" : "INES" }
{ "_id" : ObjectId("5cab3fdbfbbf7fff42d29963"), "NOMBRE" : "MARTIN" }
```

Operaciones CRUD. Lectura

ORDENACIÓN

```
db.nombrecoleccion.find(    ).sort(criterios)  
criterios: columnas de ordenación, (pares campo:1/-1)  
(1 → ascendente, -1 → descendente)
```

CONTAR RESULTADOS

```
db.nombrecoleccion.find(    ).count( )
```

```
> db.AMIGO.find().sort({"APELLIDO":1})  
{ "_id" : ObjectId("5cab3fdbfbbf7fff42d29963"), "NOMBRE" : "MARTÍN" }  
{ "_id" : ObjectId("5cab3fbcfbbf7fff42d29962"), "NOMBRE" : "INES", "APELLIDO" : "LOPEZ", "EDAD" : 25 }  
{ "_id" : ObjectId("5cab3f91fbbf7fff42d29961"), "NOMBRE" : "JOSE", "APELLIDO" : "PEREZ" }  
> db.AMIGO.find().count()  
3  
>
```

Operaciones CRUD. Lectura

OPERADORES CONDICIONALES

OPERADOR	ACCIÓN	EJEMPLO	EQUIVALENTE SQL
\$eq \$ne	= ≠	{ "EDAD": { \$ne: 20 } }	EDAD <> 20
\$gt \$gte	> >=	{ "EDAD": { \$gt: 20 } }	EDAD > 20
\$lt \$lte	< <=	{ "EDAD": { \$lte: 20 } }	EDAD <= 20
\$exists	true si el campo está definido	{ "EDAD": { \$exists: true } } { "EDAD": { \$exists: false } }	No hay equivalente SQL

```
> db.AMIGO.find({"EDAD":{"$gt:20}}, {"NOMBRE":1, "EDAD":1, "_id":0})
{"NOMBRE" : "MICAELA", "EDAD" : 33 }
{"NOMBRE" : "MARIA", "EDAD" : 27 }
{"NOMBRE" : "NURIA", "EDAD" : 24 }
> db.AMIGO.find({"EDAD":{"$exists:false}}, {"_id":0})
{"NOMBRE" : "TOMAS", "APELLIDO" : "TORONZO", "CIUDAD" : "MADRID", "TELF" : 9483729563 }
{"NOMBRE" : "RAMON", "APELLIDO" : "RAMIREZ", "CIUDAD" : "AVILA" }
{"NOMBRE" : "LOLA", "APELLIDO" : "LUCAS", "CIUDAD" : "AVILA", "TELF" : 938475623 }
{"NOMBRE" : "ANDRES", "APELLIDO" : "ANDRADE", "CIUDAD" : "MADRID" }
{"NOMBRE" : "JULIO", "APELLIDO" : "JALON", "CIUDAD" : "MADRID" }
{"NOMBRE" : "PAULA", "APELLIDO" : "POLO", "TELF" : 276859123 }
{"NOMBRE" : "CARLOS", "APELLIDO" : "CARROZA", "CIUDAD" : null }
```

Operaciones CRUD. Lectura

OPERADORES LÓGICOS

OPERADOR	ACCIÓN	EJEMPLO	EQUIVALENTE SQL
\$or	OR	{ \$or: [{ "EDAD" : 20 } , { "EDAD" : 25 }] }	EDAD=20 OR EDAD=25
\$and	AND	{ \$and: [{ "EDAD" : 20 } , { "NOMBRE" : "LUCIA" }] } { "EDAD" : 20 , "NOMBRE" : "LUCIA" }	EDAD=20 AND NOMBRE= ' LUCIA '
\$not	NOT	{ "EDAD" : { \$not : { \$eq : 20 } } }	NOT EDAD=20
\$in	IN	{ "EDAD" : { \$in : [20 , 25 , 30] } }	EDAD IN (20 , 25 , 30)
\$nin	NOT IN	{ "EDAD" : { \$nin : [18 , 20] } }	EDAD NOT IN (18 , 20)

NOTA. El operador \$and tiene el mismo efecto que no usar operador: en cualquier caso se recuperan los documentos que cumplen todas las condiciones

```

> db.AMIGO.find({$and:[{"EDAD":18},{ "NOMBRE":"OSCAR"}]}, {"_id":0}).sort({"EDAD":-1})
{ "NOMBRE" : "OSCAR", "APELLIDO" : "OROZCO", "EDAD" : 18 }
> db.AMIGO.find({"EDAD":18,"NOMBRE":"OSCAR"}, {"_id":0}).sort({"EDAD":-1})
{ "NOMBRE" : "OSCAR", "APELLIDO" : "OROZCO", "EDAD" : 18 }

```

Operaciones CRUD. Lectura

```

db.AMIGO.find({$or:[{"EDAD":18},{ "NOMBRE":"NURIA"}]}, {"_id":0})
"NOMBRE" : "MIGUEL", "APELLIDO" : "PONZANO", "CIUDAD" : "TOLEDO", "EDAD" : 18, "TELF" : 847365028 }
"NOMBRE" : "MARIA", "APELLIDO" : "LOZANO", "CIUDAD" : "MADRID", "TELF" : 776674338, "EDAD" : 18 }
"NOMBRE" : "NURIA", "APELLIDO" : "NARIZ", "CIUDAD" : "AVILA", "EDAD" : 24 }
"NOMBRE" : "OSCAR", "APELLIDO" : "OROZCO", "EDAD" : 18 }

db.AMIGO.find({"EDAD":{"$not":{"$gt":20}}}, {"_id":0})
"NOMBRE" : "MIGUEL", "APELLIDO" : "PONZANO", "CIUDAD" : "TOLEDO", "EDAD" : 18, "TELF" : 847365028 }
"NOMBRE" : "MARIA", "APELLIDO" : "LOZANO", "CIUDAD" : "MADRID", "TELF" : 776674338, "EDAD" : 18 }
"NOMBRE" : "TOMAS", "APELLIDO" : "TORONZO", "CIUDAD" : "MADRID", "TELF" : 948329563 }
"NOMBRE" : "RAMON", "APELLIDO" : "RAMIREZ", "CIUDAD" : "AVILA" }
"NOMBRE" : "LOLA", "APELLIDO" : "LUCAS", "CIUDAD" : "AVILA", "TELF" : 938475623 }
"NOMBRE" : "ANDRES", "APELLIDO" : "ANDRADE", "CIUDAD" : "MADRID" }
"NOMBRE" : "JULIO", "APELLIDO" : "JALON", "CIUDAD" : "MADRID" }
"NOMBRE" : "LUCIA", "APELLIDO" : "LUARCA", "EDAD" : 20 }
"NOMBRE" : "PAULA", "APELLIDO" : "POLO", "TELF" : 276859123 }
"NOMBRE" : "OSCAR", "APELLIDO" : "OROZCO", "EDAD" : 18 }
"NOMBRE" : "CARLOS", "APELLIDO" : "CARROZA", "CIUDAD" : null }

db.AMIGO.find({"EDAD":{"$lte":20}}, {"_id":0})
"NOMBRE" : "MIGUEL", "APELLIDO" : "PONZANO", "CIUDAD" : "TOLEDO", "EDAD" : 18, "TELF" : 847365028 }
"NOMBRE" : "MARIA", "APELLIDO" : "LOZANO", "CIUDAD" : "MADRID", "TELF" : 776674338, "EDAD" : 18 }
"NOMBRE" : "LUCIA", "APELLIDO" : "LUARCA", "EDAD" : 20 }
"NOMBRE" : "OSCAR", "APELLIDO" : "OROZCO", "EDAD" : 18 }

db.AMIGO.find({"CIUDAD":{"$in":["MADRID", "AVILA"]}}, {"_id":0})
"NOMBRE" : "MICAELA", "APELLIDO" : "PONZANO", "EDAD" : 33, "CIUDAD" : "MADRID", "TELF" : 215487123 }
"NOMBRE" : "MARIA", "APELLIDO" : "LOZANO", "CIUDAD" : "MADRID", "TELF" : 776674338, "EDAD" : 18 }
"NOMBRE" : "TOMAS", "APELLIDO" : "TORONZO", "CIUDAD" : "MADRID", "TELF" : 948329563 }
"NOMBRE" : "RAMON", "APELLIDO" : "RAMIREZ", "CIUDAD" : "AVILA" }
"NOMBRE" : "LOLA", "APELLIDO" : "LUCAS", "CIUDAD" : "AVILA", "TELF" : 938475623 }
"NOMBRE" : "ANDRES", "APELLIDO" : "ANDRADE", "CIUDAD" : "MADRID" }
"NOMBRE" : "NURIA", "APELLIDO" : "NARIZ", "CIUDAD" : "AVILA", "EDAD" : 24 }
"NOMBRE" : "JULIO", "APELLIDO" : "JALON", "CIUDAD" : "MADRID" }

db.AMIGO.find({"CIUDAD":{"$not":{"$in":["MADRID", "AVILA"]}}}, {"_id":0})
"NOMBRE" : "MIGUEL", "APELLIDO" : "PONZANO", "CIUDAD" : "TOLEDO", "EDAD" : 18, "TELF" : 847365028 }
"NOMBRE" : "MARIA", "APELLIDO" : "MILLAN", "CIUDAD" : "TOLEDO", "TELF" : 215498763, "EDAD" : 27 }
"NOMBRE" : "LUCIA", "APELLIDO" : "LUARCA", "EDAD" : 20 }
"NOMBRE" : "PAULA", "APELLIDO" : "POLO", "TELF" : 276859123 }
"NOMBRE" : "OSCAR", "APELLIDO" : "OROZCO", "EDAD" : 18 }
"NOMBRE" : "CARLOS", "APELLIDO" : "CARROZA", "CIUDAD" : null }

```

Operaciones CRUD. Lectura

Ejemplo de combinación de operadores `$or` y `$and`. Buscar los documentos correspondientes a los nombres y apellidos Julio Jalón y Paula Polo.

Simbolo del sistema - mongo

```
> db.AMIGO.find({$or:[{"NOMBRE":"JULIO", "APELLIDO":"JALON"}, {"NOMBRE":"PAULA", "APELLIDO":"POLO"}]}, {"_id":0})
"NOMBRE" : "JULIO", "APELLIDO" : "JALON", "CIUDAD" : "MADRID" }
"NOMBRE" : "PAULA", "APELLIDO" : "POLO", "TELF" : 276859123 }

> db.AMIGO.find({$or:[{$and:[{"NOMBRE":"JULIO"}, {"APELLIDO":"JALON"}]}, {$and:[{"NOMBRE":"PAULA"}, {"APELLIDO":"POLO"}]}]}, {"_id":0})
"NOMBRE" : "JULIO", "APELLIDO" : "JALON", "CIUDAD" : "MADRID" }
"NOMBRE" : "PAULA", "APELLIDO" : "POLO", "TELF" : 276859123 }
```

En la primera sentencia no se incluye el operador `$and` explícitamente.

En la segunda sentencia sí aparece explícitamente el comando `$and`.

El resultado es el mismo en ambos casos.

Operaciones CRUD. Lectura

Consultas sobre arrays. Se pueden poner condiciones sobre los valores de un campo array

```
> db.AMIGO.insert({"NOMBRE":"PABLO", "NACIONALIDAD":["ESPAÑA", "FRANCIA"]})
WriteResult({ "nInserted" : 1 })
> db.AMIGO.insert({"NOMBRE":"PEDRO", NACIONALIDAD:"ITALIA"})
WriteResult({ "nInserted" : 1 })
>
> db.AMIGO.find({"NACIONALIDAD":"ESPAÑA"})
{ "_id" : ObjectId("5cbcfe40b114a6ef99806f0c"), "NOMBRE" : "PABLO", "NACIONALIDAD" : [ "ESPAÑA", "FRANCIA" ] }
>
```

Consultas sobre subdocumentos.

El valor de un campo puede ser un documento (subdocumento). Nos referimos al valor de un campo del subdocumento como:

`CampoDocumento.CampoSubdocumento`

```
>
> db.AMIGO.insert({"NOMBRE":"MARIA", "DIRECCION":{"CALLE":"PEZ", "NUM":13, "POBLACION":"MEDINA"}})
WriteResult({ "nInserted" : 1 })
> db.AMIGO.find({"DIRECCION.POBLACION":"MEDINA"})
{ "_id" : ObjectId("5cbd003cb114a6ef99806f0e"), "NOMBRE" : "MARIA", "DIRECCION" : { "CALLE" : "PEZ", "NUM" : 13
, "POBLACION" : "MEDINA" } }
```


Operaciones CRUD. Borrado

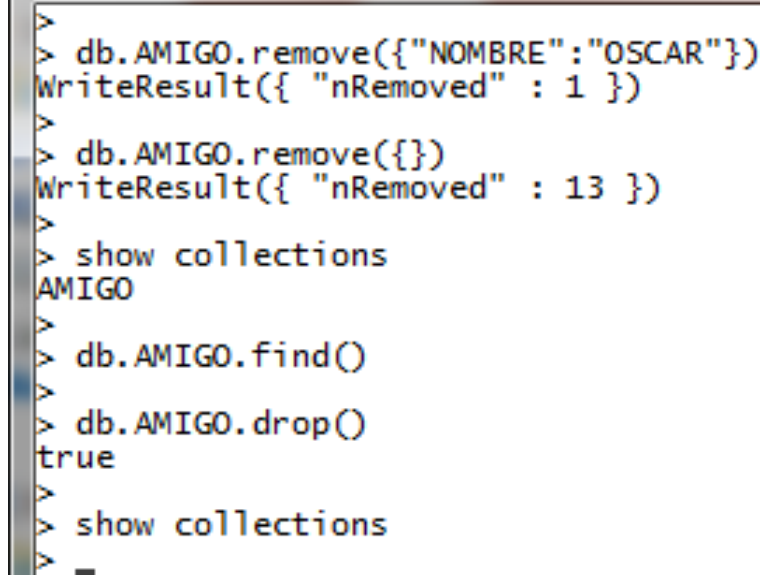
BORRADO DE DOCUMENTOS

`db.nombrecoleccion.remove(filtro)`

`filtro`: condición de borrado, (pares *nombre:valor* de los documentos a eliminar). Si se omite (`{ }`) se eliminan todos los documentos de la colección.

`db.nombrecoleccion.drop()`

Elimina la colección.



```
> db.AMIGO.remove({"NOMBRE":"OSCAR"})
WriteResult({ "nRemoved" : 1 })
> db.AMIGO.remove({})
WriteResult({ "nRemoved" : 13 })
> show collections
AMIGO
> db.AMIGO.find()
> db.AMIGO.drop()
true
> show collections
>
```

Operaciones CRUD. Modificación

MODIFICACIÓN DE DOCUMENTOS

`db.nombrecoleccion.update(filtro, documento)`

filtro: condición de modificación, (pares *nombre:valor* de los documentos a modificar). Si se omite (`{ }`) se modifican todos los documentos de la colección.

documento: nuevos pares del documento modificado. Los valores de los campos no incluidos desaparecerán.

► Operadores de modificación:

- `$set` añade nuevas propiedades un documento. Si la propiedad ya existe, cambia su valor
- `$unset` elimina propiedades de un documento:
- `$inc` incrementa en una cantidad el valor de un campo
- `$rename` cambia el nombre de un campo

Operaciones CRUD. Modificación

```
db.AMIGO.update( {NOMBRE : "Mara" } ,  
                  { $set: {EDAD : 24} } ) ;  
db.AMIGO.update( {NOMBRE : "Rosa" } ,  
                  { $unset: {EDAD : 27} } ) ;  
db.AMIGO.update( {NOMBRE : "Rosa" } ,  
                  { $inc: {EDAD : 3} } ) ;  
db.AMIGO.update( {NOMBRE : "Rosa" } ,  
                  { $rename: {EDAD: "AGE" , NOMBRE : "NAME" } } ) ;
```

Operaciones CRUD. Modificación

► Operadores de modificación para arrays:

- `$push` añade un elemento a un array
- `$addToSet` añade un elemento a un array solo si no existe
- `$pop` elimina el primer o el último elemento de un array (Con -1 se elimina el primero, con otro valor se elimina el último)
- `$pull` elimina los elementos del array que cumplan el filtro

```
> db.AMIGO.insert({"NOMBRE":"PABLO", "NACIONALIDAD":["ESPAÑA", "FRANCIA"]})
WriteResult({ "nInserted" : 1 })
> db.AMIGO.update({"NOMBRE":"PABLO"}, {$push:{"NACIONALIDAD":"ITALIA"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.AMIGO.find({"NOMBRE":"PABLO"})
{ "_id" : ObjectId("5cbd04efb114a6ef99806f15"), "NOMBRE" : "PABLO", "NACIONALIDAD" : [ "ESPAÑA", "FRANCIA", "ITALIA" ] }
> db.AMIGO.update({"NOMBRE":"PABLO"}, {$pop:{"NACIONALIDAD":-1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.AMIGO.find({"NOMBRE":"PABLO"})
{ "_id" : ObjectId("5cbd04efb114a6ef99806f15"), "NOMBRE" : "PABLO", "NACIONALIDAD" : [ "FRANCIA", "ITALIA" ] }
```