Consultar / Buscar documentos

1. db.movies.find({actors:"Brad Pitt"})
2. db.movies.find({$and: [{year: {$gt: 2000}}, {year: {$lt: 2010}}]})
3. db.movies.find({synopsis: /Bilbo/}) → con expresión regular
4. db.movies.find({$and: [{synopsis: /Bilbo/}, {synopsis: {$not:/Gandalf/}}]})
5. db.movies.find({$or: [{synopsis: /dwarves/}, {synopsis: /hobbit/}]})

Actualizar Documentos

1. db.movies.update({title:"The Hobbit: An Unexpected Journey"},{$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}})
2. db.movies.update({title:"Pulp Fiction"},{$push:{actors:"Samuel L. Jackson"}})

// $set te crea el campo en caso de no estar

// con $unset eliminas el campo indicado

// $rename renombras el campo

Eliminar Documentos

1. db.movies.deleteOne({"title": "Avatar"})

Ejemplo con lookup;

```
db.pruebas.aggregate( [{$group:{ _id:{nombre:"$nombre"},totalDist:{$sum:'$distKm'} }},
{$sort:{totalDist:-1}}, {$limit:1},
{$lookup:{
        from:'gustos', // de la coleccion gustos
        localField:'_id.nombre',
        foreignField: 'nombre',
        as: 'susGustos' }},
{$unwind:'$susGustos'}, {$project: { _id:0, nombre:'$_id.nombre', totalDist:1,  aficiones: '$susGustos.aficiones'  }} ])
```

TEORÍA

Comparación:

- $eq: //Igual que -- {<nameCampo>:{$eq:<value>}}
- $ne: //Distinto de | no es igual que -- {<nameCampo>:{$ne:<value>}}
- $gt: //valores mayores que -- {<nameCampo>:{$gt:<value>}}
- $gte: //Valores mayores o iguales que -- {<nameCampo>:{$gte:<value>}}
- $lt: //Valores menores que -- {<nameCampo>:{$lt:<value>}}
- $lte: //Valores menores o iguales que -- {<nameCampo>:{$lte:<value>}}
- $in: //Valores que se encuentran en un array de elementos -- {<nameCampo>:{$in:[<value1>,..., <valueN>]}}

- $nin: //Obtiene valores que NO se encuentran en un array de elementos -- {<nameCampo>:{$nin:[<value1>,..., <valueN>]}}

Operadores de elementos

- $exists: //Comprueba si el campo existe -- {<nameCampo>: {$exists: true | false}}
- $type: //Comprueba el tipo de dato del campo -- { y: { $type: "value" } }

```
// busqueda por texto:
    //carácter comodín
        db.collection.find({'campoBusqueda':/valor/},{})

    //operador $regex
        db.collection.find({'campoBusqueda': { $regex: value} },{} )

// operaciones con array
    // mostrar n valores de un array (-n para contar desde el final) y [n,m] para un intervao
        db.collection.find( { 'nameCampo': {$slice: n} } )
    // igualdad de todo el array
        db.collection.find( { 'nameCampo': [value_a0, value_a1, value_a2] } )

    //encontrar matriz con los elementos que indiquemos sin tener encuenta el orden
        db.collection.find( { 'nameCampo': { $all: ["value1", "value2"] } } )

    // Consulta de un elemento de matriz que cumple varios criterios
        db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )

    // Consulta de un elemento por la posición del índice de matriz
        // todos los documentos en los que el segundo elemento [.1] de la matriz dim_cmes mayor que 25
        db.inventory.find( { "dim_cm.1": { $gt: 25 } } )

    // Consultar una matriz por longitud de matriz [$size: nº de elementos]
        db.inventory.find( { "tags": { $size: 3 } } )

    // Añadir un elemento al array por la derecha
        db.collection.update[One|Many]( {filtro} , { $push : { 'nameCampo' : new_value } } )

    // Eliminar un elemento del array por la izquierda (primer elemento)
        db.collection.updateOne( { filtro}, { $pop: { 'nameCampo': -1 } } )

    // Eliminar un elemento del array por la dereche (ultimo elemento)
        db.collection.updateOne( { filtro }, { $pop: { 'nameCampo': 1 } } )
```

// Añadir varios los elementos especificados al Array
db.collection.update({ filtro }, { $pushAll: { 'nameCampo': [value1, value2, value3] } })

// Eliminar un elemento cualquiera del array especificando su valor.
db.profiles.updateOne({ filtro }, { $pull: { votes: { $gte: 6 } } })

// Eliminar todos los elementos coincidentes del Array.
db.collection.update({ filtro }, { $pullAll: { 'nameCampo': [value1, value2, value3] } })

// Añadir un elemento al array únicamente si no existe ya.
db.inventory.updateOne({ _id: 1 }, { $addToSet: { 'nameCampo': value } })

¿Cuándo embeber? → Pequeños subdocumentos, sin grandes cambios, debe existir una consistencia, se necesitan siempre en la consulta por su lectura rápida.

¿Cuándo referenciar? → Documentos grandes, muchos cambios, que se necesite escritura rápida.

¿Para qué sirve un índice? → Optimiza las búsquedas (se puede ver con .explain("executionStats"))

consulta de índices existentes → db.collection.getIndexes()

índice simple → db.collection.createIndex({'nameCampo':1},{unique:true})

índices compuestos → db.collection.createIndex({'nameCampo':1 , 'nameCampo':1})

el índice filtra si los documente tienen o no el campo → db.collection.createIndex({'nameCampo':1 , 'nameCampo':1}, {sparse:true})

eliminar índices → db.collection.dropIndex('nombre') o db.collection.dropIndexes()

índices de texto → db.collection.createIndex({ key: "text" }) >> db.collection.find({$text:{$search:"palabra"}})

Agregaciones:

- $project // donde indicamos que campos mostrar : 1 o noMOstrar:0
  db.collection.aggregate([{ $project : { 'campo' : 1 , 'campo2' : 0 } }])
- $match //Filtra el flujo de documentos para permitir que solo los documentos coincidentes pasen sin modificar a la siguiente etapa de canalización.
  db.collection.aggregate([{ $match : { 'campoACoincidir' : "davalueve" } }])
- $limit //Pasa los primeros n documentos sin modificar a la canalización donde n es el límite especificado.
  db.collection.aggregate([{ $limit: numero }])
- $skip //Omite los primeros n documentos donde n es el número de omisión especificado
  db.collection.aggregate([{ $skip : numero }]);
- $unwind //Deconstruye un campo de matriz a partir de los documentos de entrada para generar un documento para cada elemento.
  db.collection.aggregate([{ $unwind : '$campoArray' }]);

- $group //separa los documentos en grupos según una "clave de grupo". El resultado es un documento para cada clave de grupo única. value puede ser el contenido de un campo: $nameCampo

  db.collection.aggregate([{ $group : { _id : "$property_type" }, {'nameCampo': {$acumulador:value}} }])

- $sort // Reordena el flujo de documentos mediante una clave de clasificación especificada.

  db.collection.aggregate([{ $sort : { 'campoPtoReferencia' : 1 } }])

- $lookup // Realiza una unión externa a otra colección en la misma base de datos para filtrar los documentos de la colección "referenciada" para su procesamiento.

  db.collection.aggregate([{$lookup: {
  from: <collection to join>, //[coleccion que queremos unir]
  localField: <field from the input documents>, //campo de nuestra coleccion actual
  foreignField: <field from the documents of the "from" collection>, //campo de la coleccion que queremos unir
  as: <output array field> //nombre de la salida de datos.
  } }])

- $out // -- Escribe los documentos resultantes de la canalización de agregación en una colección

  db.collection.aggregate([
  { $group : { _id : "$nameCampo", campoInteres: { $push: "$nameCampo" } } }, //query de interes
  { $out : "nameNewCollection" } // crea una colección con los datos -- realizar find()])

Acumuladores:

$sum // devuelve una suma de valores numéricos
$avg // Devuelve un promedio de valores numéricos.
$first // Devuelve un valor del primer documento para cada grupo.
$last // Devuelve un valor del último documento para cada grupo
$max //Devuelve el valor de expresión más alto para cada grupo
$push // Devuelve una matriz de valores de expresión para cada grupo. -- repeticiones
$addToSet // Devuelve una matriz de valores de expresión únicos para cada grupo. -- no rept
$stdDevPop // Devuelve la desviación estándar de la población de los valores de entrada.
$stdDevSamp // Devuelve la desviación estándar de la muestra de los valores de entrada.
$cmp // Devuelve: 0 si los dos valores son equivalentes, 1 si el primer valor es mayor que el //segundo y -1 si el primer valor es menor que el segundo
$eq // Devuelve true si los valores son equivalentes.
$gt // Devuelve true si el primer valor es mayor que el segundo.
$gte //Devuelve true si el primer valor es mayor o igual que el segundo.
$lt //Devuelve true si el primer valor es menor que el segundo.

$lte // Devuelve true si el primer valor es menor o igual que el segundo.

$ne // Devuelve true si los valores no son equivalentes.

$in // Devuelve true si un valor está contenido en un array.

$nin // Devuelve true si un valor no está contenido en un array.

$cond((condicion), result if true, result if false) // Realiza una cosa u otra dependiendo

//de si se cumple o no la condición especificada

$and // devuelve true solo cuando todas sus expresiones se evalúan como true.

$or // Devuelve true cuando cualquiera de sus expresiones se evalúa como true.

$not // Devuelve el valor booleano opuesto a la expresión de su argumento.

// -- $project

$add //Realiza la suma de un array de números.

$divide // Divide dos números.

$mod // A partir de dos números calcula el resto producido al dividir el primero entre el segundo.

//$mod:[2,0] -- indica si es par o no

$multiply //Multiplica dos números.

$substract //A partir de dos números realiza la resta.

// --- string

$concat //Concatena cualquier número de cadenas.

$substr //Devuelve una subcadena de una cadena, comenzando en una posición de índice

//especificada hasta una longitud especificada.

$toLower // Convierte una cadena a minúsculas.

$toUpper // Convierte una cadena en mayúsculas.

$strcasecmp // Realiza una comparación de cadenas no distingue entre mayúsculas y minúsculas

//devuelve: 0 si dos cadenas son equivalentes, 1 si la primera cadena es mayor que la

//segunda y -1 si la primera cadena es menor que la segunda.

// --- array

$arrayElemAt //Devuelve el elemento en el índice de matriz especificado.

$concatArrays //Concatena matrices para devolver la matriz concatenada.

$filter //Selecciona un subconjunto de la matriz para devolver una matriz con solo los elementos

//que coinciden con la condición del filtro.

$isArray // Determina si el operando es una matriz. Devuelve un booleano.

$size //Devuelve el número de elementos de la matriz. Acepta una sola expresión como argumento.

$slice // Devuelve un subconjunto de una matriz.

// --- date

$dayOfYear //Devuelve el día del año para una fecha como un número entre 1 y 366 (año bisiesto).

$dayOfMonth //Devuelve el día del mes para una fecha como un número entre 1 y 31.

$dayOfWeek //Devuelve el día de la semana para una fecha como un número entre 1 (domingo) y 7

//(sábado).

$year // Devuelve el año de una fecha como un número (por ejemplo, 2014).

$month // Devuelve el mes de una fecha como un número entre 1 (enero) y 12 (diciembre).

$week // Devuelve el número de semana de una fecha como un número entre 0 (la semana parcial

//que precede al primer domingo del año) y 53 (año bisiesto).

$hour //Devuelve la hora de una fecha como un número entre 0 y 23.

$minute //Devuelve el minuto de una fecha como un número entre 0 y 59.

$second //Devuelve los segundos de una fecha como un número entre 0 y 60 (segundos intercalares).

$millisecond //Devuelve los milisegundos de una fecha como un número entre 0 y 999.

$dateToString //Devuelve la fecha como una cadena formateada

Otras consultas:

1) Seleccionar los documentos de tipo «CD», de manera que solo se muestre en dichos documentos los campos «Artista», «TituloCanción», y un nuevo campo «TitulosCanciones», que contenga un array con las canciones del disco.

db.multimedia.aggregate({ $match: { tipo:'CD'} }, { $project: { _id:0,artista:1, tituloCanciones:'$canciones.titulo' } })

2)Seleccionar todos los documentos de tipo «DVD» y calcular cuántas películas hay de cada año de estreno, mostrando el año de estreno y el número de películas de cada año

db.multimedia.aggregate({ $match: {tipo:'DVD' }}, { $group: {_id: '$estreno', numPeliculas: {$sum: 1 } } })

3) Seleccionar el documento sobre la película «Matrix» y crear un documento por cada uno de los actores que intervienen. En los documentos resultantes solo se mostrará el título y el actor.

db.multimedia.aggregate({$match:{titulo:'Matrix'}}, {$project: {_id:0, titulo:1,actores:1}},{$unwind:'$actores' })

4) Igual que la consulta anterior, pero se mostrará solo los tres últimos resultados ordenados por el nombre del actor

db.multimedia.aggregate({$match:{titulo:'Matrix'}}, {$project: {_id:0, titulo:1, actores:1 }}, { $unwind:'$actores' },{$sort:{actores:1}},{ $skip: 3 })

6) Agrupar los documentos por «Título», mostrando el título y el total que hay de cada grupo.

db.multimedia.aggregate({$group: { _id: '$titulo', cantidad: {$sum: 1}}})

7) Agrupar los documentos por «Título», mostrando el título y el total que hay de cada grupo. Pero que solo sean películas

db.multimedia.aggregate({$match:{tipo:'DVD'}}, {$group: { _id: '$titulo', cantidad: { $sum: 1}}})

incrementar en 5 unidades el valor de la clave «leídos».

db.productos.update({titulo:'Constantinopla'}, {$inc:{leidos:5 }})

5) Actualizar el documento referido al libro «Java para todos» de manera que se elimine la clave «editorial».

db.productos.update({titulo:'Java para todos'},{$unset:{editorial:''} })

6) Actualizar el documento referido al libro «Java para todos» añadiendo el autor «María Sancho» al array de autores.

db.productos.updateOne({titulo: 'Java para todos'},{ $push: { autor:'María Sancho' }})

7) Actualizar el documento referido a la película «Matrix» añadiendo al array «actores» los valores de «Antonio Banderas» y «Brad Pitt» en una única operación.

db.productos.updateOne({titulo: 'Matrix'},{ $push: { actores: { $each:['Antonio Banderas','Brad Pitt'] }}}
) // $each para poder especificar una lista de elementos

8) Actualizar el documento referido a la película «Matrix» añadiendo al array «actores» los valores «Joe Pantoliano», «Brad Pitt» y «Natalie Portman» en caso de que no se encuentren, y si se encuentran no se hace nada.

db.productos.updateOne({ titulo:'Matrix' }, { $addToSet: { actores: { $each: ['Joe Pantoliano', 'Brad Pitt' , 'Natalie Portman']}} })

9) Actualizar el documento referido a la película «Matrix» eliminando del array el primer y último actor.

db.productos.updateOne({titulo:'Matrix'},{$pop: { actores: -1 }})...

10) Actualizar el documento referido a la película «Matrix» añadiendo al array actores los valores «Joe Pantoliano» y «Antonio Banderas».

db.productos.updateOne({titulo:'Matrix'},{$pushAll: {actores: ['Antonio Bandera', 'Joe Pantoliano'] } })

db.productos.updateOne({titulo: 'Matrix'},{ $push: { actores: { $each:['Antonio Banderas','Joe Pantoliano'] }}})

11) Actualizar el documento referido a la película «Matrix» eliminado todas las apariciones en el array «actores» de los valores «Joe Pantoliano» y «Antonio Banderas».

db.productos.updateOne({ titulo:'Matrix' }, { $pullAll: { actores: ['Antonio Banderas','Joe Pantoliano'] } })

//12) Actualizar el documento referido al disco «Recuerdos» y añadir una nueva canción al array «canciones»:

```
db.productos.updateOne(
   {titulo: 'Recuerdos'},
   {$push:{
      canciones: {
         cancion: 5,
         titulo: 'El atardecer',
         longitud: '6:50'
      }
   } }
)
```

// 13) Actualizar el documento referido al disco «Recuerdos» de manera que la canción «El atardecer» tenga asignado el número
//3 en vez de 5.

```
db.productos.updateOne(
   {titulo:'Recuerdos'},
   {$set:{'canciones.2.cancion':3 }}
)
```

//buscamos en el array cancines le tercer objetos y el campo cancion

// 14) Renombrar el nombre de la colección «media» a «multimedia».

```
db.productos.renameCollection('multimedia')
```

// 15) Actualizar el documento referido al disco «Recuerdos» de manera que en una sola operación se cambia el nombre del artista a «Los piratillas» y se muestre el documento resultante.

```
db.multimedia.findOneAndUpdate( {titulo:'Recuerdos'} ,{$set:{artista:'Los piratillos'}} )
```

```
//5) Recuperar todos los documentos que sean de tipo «libro» y editorial «Anaya»
mostrando solo el array «capítulos».

db.multimedia.find(
   {$and:[{tipo:'libro'},{editorial:'Anaya'}]},
   {_id:0, capitulos:1}
)

//6) Recuperar todos los documentos referidos a canciones que tengan una canción que se
denomine «Pajaritos».

db.multimedia.find(
   {'canciones.titulo':'Pajaritos'}
)

// 7) Recuperar todos los documentos en los que «Timoteo Marino» es autor de un libro.

db.multimedia.find(
   {autor:{$all:['Timoteo Marino']}}
)

// 11) Recuperar todos los documentos de la colección «media» ordenados de manera
decreciente por el campo «tipo». Recuperar
// solo dos resultados y saltarse los dos primeros resultados.

db.multimedia.find().sort({tipo:-1}).limit(2).skip(2)

//13) consultas:
 //Recuperar los documentos sobre películas cuya fecha de estreno sea mayor que 2000.
En los resultados no mostrar el array de actores.
   db.multimedia.find(
      {estreno:{$gt:2000}},
      {actores:0}
   )

   // Recuperar los documentos sobre películas cuya fecha de estreno sea mayor o igual
que 1999. En los resultados no mostrar el array de actores.
   db.multimedia.find(
      {estreno:{$gte:1999}},
      {actores:0}
   )
   // Recuperar los documentos sobre películas cuya fecha de estreno sea menor que 1999.
En los resultados no mostrar el array de actores.
   db.multimedia.find(
      {estreno:{$lt:1999}},
      {actores:0}
   )
```

```
   // Recuperar los documentos sobre películas cuya fecha de estreno sea menor o igual que 1999. En los resultados no mostrar el array de actores.
   db.multimedia.find(
      {estreno:{$lte:1999}},
      {actores:0}
   )
   // Recuperar los documentos sobre películas cuya fecha de estreno sea mayor o igual que 1999 y menor que 2010.En los resultados no mostrar el array de actores
   db.multimedia.find(
      {estreno:{$gte:1999,$lt:2010}},
      {actores:0}
   )

//14) Recuperar todos los documentos sobre libros de manera que el autor no sea «Camilo José Cela».

db.multimedia.find(
   {tipo:'libro',autor:{$ne:'Camilo José Cela'}}
)

// 15) Recuperar todos los documentos sobre películas que se hayan estrenado en alguno de los años 1999, 2005 y 2006. En los resultados no mostrar el array de actores.

db.multimedia.find(
   {tipo:'DVD', estreno:{$in:[1999, 2005,2006]}}
)

// 16) Recuperar todos los documentos sobre películas que no se hayan estrenado en los años 1999, 2005 y 2006. En los resultados no mostrar el array de actores.

db.multimedia.find(
   {tipo:'DVD', estreno:{$nin:[1999, 2005,2006]}}
)

// 17) Recuperar todos los documentos sobre películas que se hayan estrenado en los años 1999 y 2005 exactamente. En los resultados no mostrar el array de actores.

db.multimedia.find(
      {tipo:'DVD', estreno:{$all:[1999, 2005]}},
      {actores:0}
)

// 18) Recuperar todos los documentos sobre libros que hayan sido escritos por Pepe Caballero e Isabel Sanz y que además entre los títulos de sus capítulos haya alguno que se denomine «Bucles».

db.multimedia.find(
   {
```

```
        autor:{$all:['Pepe Caballero','Isabel Sanz']},
        'capitulos.titulo':{$in:['Bucles']}
    }
)

// 19) Recuperar todos los documentos que tomen en la clave «Título» el valor «Recuerdos» o que tome en la clave «estreno» el valor «1999», y que tome en la clave tipo «DVD».

db.multimedia.find(
    {
        $and:[{tipo:'DVD'} , {$or:[{titulo:'Recuerdos'}, {estreno:1999}]}]
    }
)

// 20) Considerar el documento acerca de la película «Matrix» y recuperar del array de actores:
    // Los 3 primeros actores.
    db.multimedia.find(
        {titulo:'Matrix'},
        {actores:{$slice:3}}
    )
    // Los últimos 3 actores.
    db.multimedia.find(
        {titulo:'Matrix'},
        {actores:{$slice:-3}}
    )
    // 3 actores saltándose los 2 dos primeros actores.
    db.multimedia.find(
        {titulo:'Matrix'},
        {actores:{$slice:[2,3]}}
    )

    // 4 actores saltándose los 5 últimos actores
    db.multimedia.find(
        {titulo:'Matrix'},
        {actores:{$slice:[-5,4]}}
    )

// 21) Recuperar los documentos referidos a películas tales que en su campo «estreno» tenga un valor par. No mostrar el array actores.

db.multimedia.find(
    {tipo:'DVD' , estreno:{$mod:[2,0]}},
    {actores:0}
)

// 22) Recuperar los documentos referidos a películas tales que en su campo «estreno» tenga un valor impar. No mostrar el array actores.
```

```
db.multimedia.find(
   {tipo:'DVD' , estreno:{$mod:[2,1]}},
   {actores:0}
)

// 23) Recuperar todos los documentos referidos a canciones tales que el número de canciones es exactamente 2.

db.multimedia.find( {canciones:{$size:3}} )

// 24) Recuperar todos los documentos tales que tengan un array de actores.

db.multimedia.find( {actores:{$exists:true}} )

// 25) Recuperar todos los documentos tales que no tengan un array de actores.

db.multimedia.find( {actores:{$exists:false} } )

//26) Considerar la siguiente tabla que asigna a cada tipo de
//datos BSON con un valor numérico
//Recuperar todos los documentos que tienen un campo denominado «canciones» cuyo valor sea del tipo un documento embebido.

db.multimedia.find( {canciones:{$type: 3}} )

// 27)  Recuperar todos los documentos sobre discos en los que se dan exactamente las siguientes condiciones: existe una canción
// denominada «Pajaritos» y el número de canción es el 2.

db.multimedia.find(
   {
      tipo:'CD',
      $and:[{ 'canciones.titulo':'Pajaritos'},{'canciones.cancion':2}]
   }
)

db.multimedia.find(
   {
      canciones:{$elemMatch:{cancion:2, titulo:'Pajaritos'}}
   }
)

//28) Recuperar todos los documentos sobre discos en los que no se dan exactamente las siguientes condiciones: existe una
//canción denominada «Pajaritos» y el número de canción es el 2.

db.multimedia.find(
```

```
   {
      canciones:{$not:{$elemMatch:{cancion:2, titulo:'Pajaritos'}}}
   }
)

// 30) Encontrar los DVD que sean más antiguos que 1995.

db.multimedia.find(
   {tipo:'DVD', estreno:{$lt:1995}}
)

// 31) Encontrar todos los documentos en los que en el campo «Artista» aparece la palabra
«pira».

db.multimedia.find(
   {artista:/pira/}
)

// 32) Encontrar todos los documentos en los que en el campo «Artista» aparece la palabra
«piratas» y además tienen un campo «Titulo».
db.multimedia.find(
   {$and:[{artista:/pira/},{titulo:{$exists:true}}]}
)
```