02 CRUD Operaciones de lectura

Método find()

https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find

Sintaxis

db.collection.find(query, projection)

Parameter	Type	Description
query	document	Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
projection	document	Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter.

Select All Documents in a Collection

db.collection.find()
db.collection.find({})

Specify Equality Condition

To specify equality conditions, use <field>:<value> expressions in the query filter document:

{ <field1>: <value1>, ... }

Práctica

```
> db.clientes.insert({nombre:'Luisa',apellidos:'Pérez',dni: '07967545D'})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:'José',apellidos:'Gómez',dni: '88967967F'})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:'José',apellidos:'López',dni: '44531123J'})
WriteResult({ "nInserted" : 1 })

>db.clientes.find({nombre: 'José'})
{ "_id" : ObjectId("5e25e8a08621cff1393f70b8"), "nombre" : "José", "apellidos" :
"Gómez", "dni" : "88967967F" }
{ "_id" : ObjectId("5e25e8c38621cff1393f70b9"), "nombre" : "José", "apellidos" :
"López", "dni" : "44531123J" }

> db.clientes.find({_id: "5e25e8c38621cff1393f70b9"}) // no devuelve resultado
porque _id es una instancia de ObjectId

> db.clientes.find({_id: ObjectId("5e25e8c38621cff1393f70b9")})
{ "_id" : ObjectId("5e25e8c38621cff1393f70b9"), "nombre" : "José", "apellidos" :
"López", "dni" : "44531123J" }

> db.clientes.find({nombre:"José",apellidos:"Gómez"})
{ "_id" : ObjectId("5e25e8a08621cff1393f70b8"), "nombre" : "José", "apellidos" :
"Gómez", "dni" : "88967967F" }
```

Specify Conditions Using Query Operators

A query filter document can use the query operators to specify conditions in the following form:

{ <field1>: { <operator1>: <value1> }, ... }

Práctica

```
> db.clientes.insert({nombre:"Lucía", apellidos:"Pérez", clases: ["aerobic","zumba"]})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:"Sergio", apellidos:"González", clases:
["padel","zumba"]})
WriteResult({ "nInserted" : 1 })

> db.clientes.find({clases: { $in: [ "zumba", "aerobic" ] }})
{ "_id" : ObjectId("5e25eb401d4ba0ea8964cdca"), "nombre" : "Lucía", "apellidos" :
"Pérez", "clases" : [ "aerobic", "zumba" ] }
{ "_id" : ObjectId("5e25eb5a1d4ba0ea8964cdcb"), "nombre" : "Sergio", "apellidos" :
"González", "clases" : [ "padel", "zumba" ] }
```

Specify AND Conditions

A compound query can specify conditions for more than one field in the collection's documents. Implicitly, a logical AND conjunction connects the clauses of a compound query so that the query selects the documents in the collection that match all the conditions.

{ <field1>: <value1>, <field2>: <value2>,... }

Práctica

```
> db.clientes.insert({nombre:'Luisa',apellidos:'Gutierrez', edad: 30})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:Jorge,apellidos:'Martínez', edad: 22})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:'Jorge',apellidos:'López', edad: 18})
WriteResult({ "nInserted" : 1 })

> db.clientes.find({edad: {$gt: 20}, nombre: "Jorge"})
{ "_id" : ObjectId("5e25ed0d1d4ba0ea8964cdcf"), "nombre" : "Jorge", "apellidos" :
"Martínez", "edad" : 22 }
```

```
> db.clientes.find({edad: {$gt: 20}, edad: {$lt: 30}})
{ "_id" : ObjectId("5e25ecd91d4ba0ea8964cdce"), "nombre" : "Jorge", "apellidos" :
"López", "edad" : 18 }
{ "_id" : ObjectId("5e25ed0d1d4ba0ea8964cdcf"), "nombre" : "Jorge", "apellidos" :
"Martínez", "edad" : 22 }
```

Specify OR Conditions

Using the $or operator, you can specify a compound query that joins each clause with a logical OR conjunction so that the query selects the documents in the collection that match at least one condition.

{ $or: [{<field1>: <value1>, <field2>: <value2>...}, {<field3>: <value3>...}, ...] }

Práctica

```
> db.clientes.find({$or: [{nombre: "José"},{edad: {$gte: 20}}]})
{ "_id" : ObjectId("5e25e8a08621cff1393f70b8"), "nombre" : "José", "apellidos" :
"Gómez", "dni" : "88967967F" }
{ "_id" : ObjectId("5e25e8c38621cff1393f70b9"), "nombre" : "José", "apellidos" :
"López", "dni" : "44531123J" }
{ "_id" : ObjectId("5e25ec9b1d4ba0ea8964cdcc"), "nombre" : "Luisa", "apellidos" :
"Gutierrez", "edad" : 30 }
{ "_id" : ObjectId("5e25ecb11d4ba0ea8964cdcd"), "nombre" : "Javier", "apellidos" :
"Martínez", "edad" : 22 }
{ "_id" : ObjectId("5e25ed0d1d4ba0ea8964cdcf"), "nombre" : "Jorge", "apellidos" :
"Martínez", "edad" : 22 }
```

Specify AND as well as OR Conditions

{ <field1>: <value1>, $or: [{<field2>: <value2>, <field3>: <value3>...}, {<field4>: <value4>...}, ...] }

```
> db.clientes.find({
... apellidos: "Gómez",
... $or: [{edad: {$gt: 20}}, {dni:"88967967F"}]
... })
{ "_id" : ObjectId("5e25e8a08621cff1393f70b8"), "nombre" : "José", "apellidos" :
"Gómez", "dni" : "88967967F" }
```

Match an Embedded/Nested Document

To specify an equality condition on a field that is an embedded/nested document, use the query filter document { <field>: <value> } where <value> is the document to match.

Práctica

```
> db.clientes.insert({nombre:"Cecilia", apellidos:"Sánchez", domicilio: {calle: "Gran Vía, 80", cp: 28003, localidad: "Madrid"}})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:"Carlos", apellidos:"Pérez", domicilio: {calle: "Alcalá, 90", cp: 28004, localidad: "Madrid"}})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:"Inés", apellidos:"Pérez", domicilio: {calle: "Burgos, 10", cp: 28901, localidad: "Getafe"}})
WriteResult({ "nInserted" : 1 })

> db.clientes.find({domicilio: {calle: "Alcalá, 90", cp: 28004, localidad: "Madrid"}})

> db.clientes.find({domicilio: {calle: "Alcalá, 90", localidad: "Madrid", cp: 28004}}) // No devuelve el registro puesto que el orden de los campos no es el mismo
```

Query on Nested Field

To specify a query condition on fields in an embedded/nested document, use dot notation ("field.nestedField"). When querying using dot notation, the field and nested field must be inside quotation marks.

Specify Equality Match on a Nested Field

Práctica

```
> db.clientes.find({"domicilio.localidad": "Madrid"})
```

Specify Match using Query Operator

Práctica

```
> db.clientes.find({"domicilio.cp": {$gte: 28004}})
```

Specify AND Condition

Práctica

```
> db.clientes.find({"domicilio.cp": {$gte: 28004}, apellidos: "Pérez",
"domicilio.localidad":"Getafe"})
```

Match an Array

To specify equality condition on an array, use the query document { <field>: <value> } where <value> is the exact array to match, including the order of the elements.

Práctica

```
> db.clientes.find({clases: ["aerobic","zumba"]})
{ "_id" : ObjectId("5e25eb401d4ba0ea8964cdca"), "nombre" : "Lucía", "apellidos" :
"Pérez", "clases" : [ "aerobic", "zumba" ] }
> db.clientes.find({clases: ["zumba","aerobic"]}) // No devuelve nada al no tener el
mismo orden de elementos del array
```

If, instead, you wish to find an array that contains the elements without regard to order or other elements in the array, use the $all operator.

Práctica

```
> db.clientes.insert({nombre:"Juan José", apellidos:"Pérez", clases:
["padel","esgrima","pesas"]})

> db.clientes.find({clases: {$all: ["pesas","padel","esgrima"]}})
> db.clientes.find({clases: {$all: ["pesas","esgrima"]}}) // También devuelve el
registro anterior porque contiene todos los elementos del array de la consulta
> db.clientes.find({clases: {$all: ["pesas","esgrima","natación"]}}) // No devuelve el
registro porque no incluyen uno de los elementos
```

Query an Array for an Element

To query if the array field contains at least one element with the specified value, use the filter { <field>: <value> } where <value> is the element value.

Práctica

> db.clientes.find({clases: "padel"}) // Cualquier documento cuyo array clases contenga al menos un elemento como el especificado en la consulta

> db.clientes.insert({nombre:"María", apellidos:"García", puntuaciones: [100, 120, 44]})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre:"Fernando", apellidos:"García", puntuaciones: [60, 90, 70]})
WriteResult({ "nInserted" : 1 })

> db.clientes.find({puntuaciones: {$lte: 50}}) // Devuelve cualquier documento cuyo array clases contenga al menos un elemento que cumpla la condición especificada

Specify Multiple Conditions for Array Elements

When specifying compound conditions on array elements, you can specify the query such that either a single array element meets these condition or any combination of array elements meets the conditions.

Query an Array with Compound Filter Conditions on the Array Elements

Práctica

> db.clientes.find({puntuaciones: {$gte: 50, $lt: 75}}) // Los elementos del array, uno o una combinación de varios, deben cumplir ambas condiciones especificadas en la consulta

Query for an Array Element that Meets Multiple Criteria

Práctica

> db.clientes.find({puntuaciones: {$elemMatch: {$gte: 50, $lt: 75}}}) // Al menos un elemento del array debe cumplir todas las condiciones especificados

Query for an Element by the Array Index Position

Using dot notation, you can specify query conditions for an element at a particular index or position of the array. The array uses zero-based indexing. When querying using dot notation, the field and nested field must be inside quotation marks.

Práctica

```
> db.clientes.find({"clases.0":"padel"})
```

Query an Array by Array Length

Práctica

```
> db.clientes.find({"clases":{$size: 3}})
```

Query for a Document Nested in an Array

Equality matches on the whole embedded/nested document require an exact match of the specified document, including the field order.

Práctica

```
> db.clientes.insert({nombre: "Carlos", apellidos: "García", direcciones: [{calle: "Alcalá, 40", cp: 28001, localidad: "Madrid"},{calle:"Zamora, 13", cp: 34005, localidad: "Vigo"}]})
WriteResult({ "nInserted" : 1 })
> db.clientes.insert({nombre: "Susana", apellidos: "Gómez", direcciones: [{calle: "Alcalá, 60", cp: 28001, localidad: "Madrid"},{calle:"Fuencarral, 20", cp: 28002, localidad: "Madrid"}]})
WriteResult({ "nInserted" : 1 })

> db.clientes.find({direcciones: [{calle: "Alcalá, 40", cp: 28001, localidad: "Madrid"},{calle:"Zamora, 13", cp: 34005, localidad: "Vigo"}]})
```

Specify a Query Condition on a Field in an Array of Documents

Specify a Query Condition on a Field Embedded in an Array of Documents

If you do not know the index position of the document nested in the array, concatenate the name of the array field, with a dot (.) and the name of the field in the nested document.

Práctica

> db.clientes.find({"direcciones.localidad":"Madrid"}) // cualquier documento que en su array de documentos direcciones contenga al menos un elemento con el campo localidad igual a “Madrid”

Use the Array Index to Query for a Field in the Embedded Document

Using dot notation, you can specify query conditions for field in a document at a particular index or position of the array. The array uses zero-based indexing.

Práctica

> db.clientes.find({"direcciones.1.localidad":"Madrid"}) // cualquier documento que en su array de documentos direcciones contiene en el segundo elemento el campo localidad igual a “Madrid”.

Specify Multiple Conditions for Array of Documents

When specifying conditions on more than one field nested in an array of documents, you can specify the query such that either a single document meets these condition or any combination of documents (including a single document) in the array meets the conditions.

A Single Nested Document Meets Multiple Query Conditions on Nested Fields

Use $elemMatch operator to specify multiple criteria on an array of embedded documents such that at least one embedded document satisfies all the specified criteria.

Práctica

```
> db.monitores.insert({nombre: "Pedro", apellidos: "López", actividades: [{clase: "aerobic", turno: "mañana", homologado: "false"},{clase: "aerobic", turno: "tarde"},{clase:"zumba", turno: "mañana", homologado: true}]})
WriteResult({ "nInserted" : 1 })
> db.monitores.insert({nombre: "María", apellidos: "Pérez", actividades: [{clase: "aerobic", turno: "tarde", homologado: true},{clase:"zumba", turno: "tarde", homologado: false}]})
WriteResult({ "nInserted" : 1 })
```

```
> db.monitores.find({actividades: {$elemMatch: {clase: "aerobic", turno: "mañana"}}}) // cualquier documento que en su array direcciones contenga el campo clase con valor “aerobic” y el campo turno con valor “mañana” en el mismo sub-documento.
```

Combination of Elements Satisfies the Criteria

If the compound query conditions on an array field do not use the $elemMatch operator, the query selects those documents whose array contains any combination of elements that satisfies the conditions.

Práctica

```
> db.monitores.find({"actividades.clase": "aerobic", "actividades.homologado": true}) // cualquier documento que en su array direcciones contenga el campo clase con valor “aerobic” y el campo homologado con valor true aunque sea en diferentes documentos
```

Project Fields to Return from Query

By default, queries in MongoDB return all fields in matching documents. To limit the amount of data that MongoDB sends to applications, you can include a projection document to specify or restrict fields to return.

Return All Fields in Matching Documents

If you do not specify a projection document, the db.collection.find() method returns all fields in the matching documents.

Return the Specified Fields and the _id Field Only

A projection can explicitly include several fields by setting the <field> to 1 in the projection document. By default, the _id fields return in the matching documents.

Práctica

```
> db.clientes.find({},{nombre: 1, apellidos: 1})
{ "_id" : ObjectId("5e25e8838621cff1393f70b7"), "nombre" : "Luisa", "apellidos" : "Pérez" }
{ "_id" : ObjectId("5e25e8a08621cff1393f70b8"), "nombre" : "José", "apellidos" : "Gómez" }
{ "_id" : ObjectId("5e25e8c38621cff1393f70b9"), "nombre" : "José", "apellidos" : "López" }
{ "_id" : ObjectId("5e25eb401d4ba0ea8964cdca"), "nombre" : "Lucía", "apellidos" : "Pérez" }
{ "_id" : ObjectId("5e25eb5a1d4ba0ea8964cdcb"), "nombre" : "Sergio", "apellidos" : "González" }
{ "_id" : ObjectId("5e25ec9b1d4ba0ea8964cdcc"), "nombre" : "Luisa", "apellidos" : "Gutierrez" }
{ "_id" : ObjectId("5e25ecb11d4ba0ea8964cdcd"), "nombre" : "Javier", "apellidos" : "Martínez" }
{ "_id" : ObjectId("5e25ecd91d4ba0ea8964cdce"), "nombre" : "Jorge", "apellidos" : "López" }
{ "_id" : ObjectId("5e25ed0d1d4ba0ea8964cdcf"), "nombre" : "Jorge", "apellidos" : "Martínez" }
{ "_id" : ObjectId("5e26169a1d4ba0ea8964cdd0"), "nombre" : "Cecilia", "apellidos" : "Sánchez" }
{ "_id" : ObjectId("5e2616d51d4ba0ea8964cdd1"), "nombre" : "Carlos", "apellidos" : "Pérez" }
```

{ "_id" : ObjectId("5e2619561d4ba0ea8964cdd2"), "nombre" : "Inés", "apellidos" : "Pérez" }
{ "_id" : ObjectId("5e261b701d4ba0ea8964cdd3"), "nombre" : "Juan José", "apellidos" : "Pérez" }
{ "_id" : ObjectId("5e261d2d1d4ba0ea8964cdd4"), "nombre" : "María", "apellidos" : "García" }
{ "_id" : ObjectId("5e261d5e1d4ba0ea8964cdd5"), "nombre" : "María", "apellidos" : "García" }
{ "_id" : ObjectId("5e261d9c1d4ba0ea8964cdd6"), "nombre" : "Fernando", "apellidos" : "García" }
{ "_id" : ObjectId("5e2622201d4ba0ea8964cdd7"), "nombre" : "Carlos", "apellidos" : "García" }
{ "_id" : ObjectId("5e26ea931d4ba0ea8964cdd8"), "nombre" : "Susana", "apellidos" : "Gómez" }

Suppress _id Field

You can remove the _id field from the results by setting it to 0 in the projection. With the exception of the _id field, you cannot combine inclusion and exclusion statements in projection documents.

Práctica

```
> db.clientes.find({},{nombre: 1, apellidos: 1, _id: 0})
{ "nombre" : "Luisa", "apellidos" : "Pérez" }
{ "nombre" : "José", "apellidos" : "Gómez" }
{ "nombre" : "José", "apellidos" : "López" }
{ "nombre" : "Lucía", "apellidos" : "Pérez" }
{ "nombre" : "Sergio", "apellidos" : "González" }
{ "nombre" : "Luisa", "apellidos" : "Gutierrez" }
{ "nombre" : "Javier", "apellidos" : "Martínez" }
{ "nombre" : "Jorge", "apellidos" : "López" }
{ "nombre" : "Jorge", "apellidos" : "Martínez" }
{ "nombre" : "Cecilia", "apellidos" : "Sánchez" }
{ "nombre" : "Carlos", "apellidos" : "Pérez" }
{ "nombre" : "Inés", "apellidos" : "Pérez" }
{ "nombre" : "Juan José", "apellidos" : "Pérez" }
{ "nombre" : "María", "apellidos" : "García" }
{ "nombre" : "María", "apellidos" : "García" }
{ "nombre" : "Fernando", "apellidos" : "García" }
{ "nombre" : "Carlos", "apellidos" : "García" }
{ "nombre" : "Susana", "apellidos" : "Gómez" }
```

Return All But the Excluded Fields

Instead of listing the fields to return in the matching document, you can use a projection to exclude specific fields.

Práctica

```
> db.clientes.find({},{domicilio: 0, direcciones: 0, _id: 0})
```

Return Specific Fields in Embedded Documents

You can return specific fields in an embedded document. Use the dot notation to refer to the embedded field and set to 1 in the projection document.

Práctica

```
> db.clientes.find({},{nombre: 1,"domicilio.calle": 1, _id: 0})
{ "nombre" : "Luisa" }
{ "nombre" : "José" }
{ "nombre" : "José" }
{ "nombre" : "Lucía" }
{ "nombre" : "Sergio" }
{ "nombre" : "Luisa" }
{ "nombre" : "Javier" }
{ "nombre" : "Jorge" }
{ "nombre" : "Jorge" }
{ "nombre" : "Cecilia", "domicilio" : { "calle" : "Gran Vía, 80" } }
{ "nombre" : "Carlos", "domicilio" : { "calle" : "Alcalá, 90" } }
{ "nombre" : "Inés", "domicilio" : { "calle" : "Burgos, 10" } }
```

Suppress Specific Fields in Embedded Documents

You can suppress specific fields in an embedded document. Use the dot notation to refer to the embedded field in the projection document and set to 0.

Práctica

```
> db.clientes.find({},{"domicilio.calle": 0, _id: 0})
{ "nombre" : "Luisa", "apellidos" : "Pérez", "dni" : "07967545D" }
{ "nombre" : "José", "apellidos" : "Gómez", "dni" : "88967967F" }
{ "nombre" : "José", "apellidos" : "López", "dni" : "44531123J" }
{ "nombre" : "Lucía", "apellidos" : "Pérez", "clases" : [ "aerobic", "zumba" ] }
{ "nombre" : "Sergio", "apellidos" : "González", "clases" : [ "padel", "zumba" ] }
{ "nombre" : "Luisa", "apellidos" : "Gutierrez", "edad" : 30 }
{ "nombre" : "Javier", "apellidos" : "Martínez", "edad" : 22 }
{ "nombre" : "Jorge", "apellidos" : "López", "edad" : 18 }
{ "nombre" : "Jorge", "apellidos" : "Martínez", "edad" : 22 }
{ "nombre" : "Cecilia", "apellidos" : "Sánchez", "domicilio" : { "cp" : 28003, "localidad" :
"Madrid" } }
{ "nombre" : "Carlos", "apellidos" : "Pérez", "domicilio" : { "cp" : 28004, "localidad" :
"Madrid" } }
{ "nombre" : "Inés", "apellidos" : "Pérez", "domicilio" : { "cp" : 28901, "localidad" :
"Getafe" } }
```

Projection on Embedded Documents in an Array

Use dot notation to project specific fields inside documents embedded in an array.

Práctica

```
> db.clientes.find({},{"direcciones.localidad": 1, _id: 0})
{ }
{ }
{ }
{ "direcciones" : [ { "localidad" : "Madrid" }, { "localidad" : "Vigo" } ] }
{ "direcciones" : [ { "localidad" : "Madrid" }, { "localidad" : "Madrid" } ] }
```

Project Specific Array Elements in the Returned Array

For fields that contain arrays, MongoDB provides the following projection operators for manipulating arrays: $elemMatch, $slice, and $.

$elemMatch, $slice, and $ are the only way to project specific elements to include in the returned array. For instance, you cannot project specific array elements using the array index; e.g. { "instock.0": 1 } projection will not project the array with the first element.

Práctica

```
> db.monitores.find({},{actividades: {$elemMatch:{clase: "zumba"}}})
{ "_id" : ObjectId("5e26ef5b1d4ba0ea8964cddc"), "actividades" : [ { "clase" : "zumba", "turno" : "tarde", "homologado" : false } ] }
{ "_id" : ObjectId("5e26efaf1d4ba0ea8964cddd"), "actividades" : [ { "clase" : "zumba", "turno" : "mañana", "homologado" : true } ] }


> db.clientes.find({},{"clases.0": 1}) // Aunque no arroja error, no proyecta el primer elemento del array clases.
```

Query for Null or Missing Fields

Different query operators in MongoDB treat null values differently.

Equality Filter

The { item : null } query matches documents that either contain the item field whose value is null or that do not contain the item field.

Práctica

```
> db.monitores.insert({nombre:"Sergio", apellidos:"Pérez"})
WriteResult({ "nInserted" : 1 })
> db.monitores.insert({nombre:"Sara", apellidos:"González", actividades: null})
WriteResult({ "nInserted" : 1 })
```

> db.monitores.find({actividades: null})
{ "_id" : ObjectId("5e2723271d4ba0ea8964cdde"), "nombre" : "Sergio", "apellidos" : "Pérez" }
{ "_id" : ObjectId("5e2723431d4ba0ea8964cddf"), "nombre" : "Sara", "apellidos" : "González", "actividades" : null } // devuelve tanto los documentos con campo actividades con valor null y también los que no tienen ese campo

Type Check

The { item : { $type: 10 } } query matches only documents that contain the item field whose value is null; i.e. the value of the item field is of BSON Type Null (type number 10).

Práctica

> db.monitores.find({actividades: {$type: 10}})
{ "_id" : ObjectId("5e2723431d4ba0ea8964cddf"), "nombre" : "Sara", "apellidos" : "González", "actividades" : null } // devuelve sólo los documentos con campo actividades con valor null

Existence Check

The { item : { $exists: true } } query matches all documents that contain the item field. Set to false, matches all documents that not contain the field.

Práctica

> db.monitores.find({actividades: {$exists: false}})
{ "_id" : ObjectId("5e2723271d4ba0ea8964cdde"), "nombre" : "Sergio", "apellidos" : "Pérez" } //devuelve los documentos que no tienen el campo actividades

Método findOne()

https://docs.mongodb.com/manual/reference/method/db.collection.findOne/index.html

Sintaxis

db.collection.findOne(query, projection)

Parameter	Type	Description
query	document	Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
projection	document	Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter.

Comportamiento

The findOne Result Document
You cannot apply cursor methods to the result of findOne() because a single document is returned

Type Bracketing

MongoDB treats some data types as equivalent for comparison purposes. For instance, numeric types undergo conversion before comparison. For most data types, however, comparison operators only perform comparisons on documents where the BSON type of the target field matches the type of the query operand.

Práctica

```
> db.clases.insert({titulo:"zumba", descripcion:"lorem...", capacidad: 20})
WriteResult({ "nInserted" : 1 })
> db.clases.insert({titulo:"aerobic", descripcion:"lorem...", capacidad: "15"})
WriteResult({ "nInserted" : 1 })
> db.clases.insert({titulo:"maratón", descripcion:"lorem...", capacidad: "sin limite"})
WriteResult({ "nInserted" : 1 })
```

Métodos adicionales de lectura

In aggregation pipeline, the $match pipeline stage provides access to MongoDB queries.

Query and Projection Operators

Comparison Query Operators

$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

$eq

Specifies equality condition. The $eq operator matches documents where the value of a field equals the specified value.

{ <field>: { $eq: <value> } }

The $eq expression is equivalent to { field: <value> }.

$in

The $in operator selects the documents where the value of a field equals any value in the specified array.

{ field: { $in: [<value1>, <value2>, ... <valueN>] } }

If the field holds an array, then the $in operator selects the documents whose field holds an array that contains at least one element that matches a value in the specified array (e.g. <value1>, <value2>, etc.)

Práctica

```
> db.clientes.find({clases: {$in: ["zumba","pesas"]}})
{ "_id" : ObjectId("5e25eb401d4ba0ea8964cdca"), "nombre" : "Lucía", "apellidos" : "Pérez", "clases" : [ "aerobic", "zumba" ] }
{ "_id" : ObjectId("5e25eb5a1d4ba0ea8964cdcb"), "nombre" : "Sergio", "apellidos" : "González", "clases" : [ "padel", "zumba" ] }
{ "_id" : ObjectId("5e261b701d4ba0ea8964cdd3"), "nombre" : "Juan José", "apellidos" : "Pérez", "clases" : [ "padel", "esgrima", "pesas" ] }
```

The $in operator can specify matching values using regular expressions of the form /pattern/. You cannot use $regex operator expressions inside an $in

Práctica

```
> db.clientes.find({clases: {$in: ["zumba",/^p/]}})
```

$ne

$ne selects the documents where the value of the field is not equal to the specified value. This includes documents that do not contain the field.

{field: {$ne: value} }

Práctica

```
> db.clientes.insert({nombre:"Dummye"})
WriteResult({ "nInserted" : 1 })
> db.clientes.find({apellidos:{$ne: "Pérez"}},{nombre:1, apellidos:1, _id: 0})
{ "nombre" : "José", "apellidos" : "Gómez" }
{ "nombre" : "José", "apellidos" : "López" }
{ "nombre" : "Sergio", "apellidos" : "González" }
{ "nombre" : "Luisa", "apellidos" : "Gutierrez" }
{ "nombre" : "Javier", "apellidos" : "Martínez" }
{ "nombre" : "Jorge", "apellidos" : "López" }
{ "nombre" : "Jorge", "apellidos" : "Martínez" }
{ "nombre" : "Cecilia", "apellidos" : "Sánchez" }
{ "nombre" : "María", "apellidos" : "García" }
{ "nombre" : "María", "apellidos" : "García" }
{ "nombre" : "Fernando", "apellidos" : "García" }
{ "nombre" : "Carlos", "apellidos" : "García" }
{ "nombre" : "Susana", "apellidos" : "Gómez" }
{ "nombre" : "Dummye" }
```

$nin

$nin selects the documents where:

- the field value is not in the specified array or
- the field does not exist.

{ field: { $nin: [<value1>, <value2> ... <valueN>]} }

Práctica

```
> db.clientes.find({"direcciones.localidad": {$nin: ["Madrid"]}})
```

Logical Query Operators

$and Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not Inverts the effect of a query expression and returns documents that do not match the query expression.
$nor Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

$and

$and performs a logical AND operation on an array of one or more expressions (e.g. <expression1>, <expression2>, etc.) and selects the documents that satisfy all the expressions in the array. The $and operator uses short-circuit evaluation. If the first expression (e.g. <expression1>) evaluates to false, MongoDB will not evaluate the remaining expressions.

{ $and: [{ <expression1> }, { <expression2> } , ... , { <expressionN> }] }

MongoDB provides an implicit AND operation when specifying a comma separated list of expressions. Using an explicit AND with the $and operator is necessary when the same field or operator has to be specified in multiple expressions.

$not

$not performs a logical NOT operation on the specified <operator-expression> and selects the documents that do not match the <operator-expression>. This includes documents that do not contain the field.

{ field: { $not: { <operator-expression> } } }

Práctica

db.clientes.find({nombre: {$not: /^M/}})

$nor

$nor performs a logical NOR operation on an array of one or more query expression and selects the documents that fail all the query expressions in the array. The $nor has the following syntax:

{ $nor: [{ <expression1> }, { <expression2> }, ... { <expressionN> }] }

Práctica

db.clientes.find({$nor: [{nombre: /^M/},{nombre: /^J/}]}) // Devuelve los que su nombre no empiezan por M o los que no empiezan por J

Element Query Operators

$exists	Matches documents that have the specified field.
$type	Selects documents if a field is of the specified type.

Evaluation Query Operators

$expr Allows use of aggregation expressions within the query language.
$jsonSchema Validate documents against the given JSON Schema.
$mod Performs a modulo operation on the value of a field and selects documents with a specified result.
$regex Selects documents where values match a specified regular expression.
$text Performs text search.
$where Matches documents that satisfy a JavaScript expression.

$regex

Provides regular expression capabilities for pattern matching strings in queries. MongoDB uses Perl compatible regular expressions (i.e. “PCRE”) version 8.42 with UTF-8 support.

To use $regex, use one of the following syntaxes:

{ <field>: { $regex: /pattern/, $options: '<options>' } }
{ <field>: { $regex: 'pattern', $options: '<options>' } }
{ <field>: { $regex: /pattern/<options> } }

In MongoDB, you can also use regular expression objects (i.e. /pattern/) to specify regular expressions:

{ <field>: /pattern/<options> }

$options
The following <options> are available for use with regular expression.

Option	Description	Syntax Restrictions
i	Case insensitivity to match upper and lower cases.	
m	For patterns that include anchors (i.e. ^ for the start, $ for the end), match at the beginning or end of each line for strings with multiline values. Without this option, these anchors match at beginning or end of the string.	
x	“Extended” capability to ignore all white space characters in the $regex pattern unless escaped or included in a character class. Additionally, it ignores characters in-between and including an un-escaped hash/pound (#) character and the	Requires $regex with $options syntax

	next new line, so that you may include comments in complicated patterns. This only applies to data characters; white space characters may never appear within special character sequences in a pattern. The x option does not affect the handling of the VT character (i.e. code 11).	
s	Allows the dot character (i.e. .) to match all characters including newline characters.	Requires $regex with $options syntax.

Práctica

```
> db.clientes.insert([
{nombre:'Luis',apellidos:'García Pérez'},
{nombre:'Pedro',apellidos:'Gutiérrez López'},
{nombre:'Carlos',apellidos:'López Gómez'},
{nombre:'Carlos',apellidos:'Pérez Góngora'},
{nombre:'Juan',apellidos:'Pérez \nGóngora'}])
```

```
>db.clientes.find({apellidos:{$regex: 'G'}}) // cualquier G mayúscula
```

```
>db.clientes..find({apellidos:{$regex: /G/}}) / /ídem al anterior
```

```
>db.clientes.find({apellidos:{$regex: /Gó/}}) // cualquier Gó
```

```
>db.clientes.find({apellidos:{$regex: /^Gu/}}) // Los que empiezan por Gu
```

```
>db.clientes.find({apellidos:{$regex: /ez$/}}) // Los que finalizan por ez
```

```
>db.clientes.find({apellidos:{$regex: /^gu/i}}) // case insesitive
```

```
>db.clientes.find({apellidos:{$regex: '^gu', $options: "i"}}) // ídem ant
```

>db.clientes.find({apellidos:{$regex: "^G", $options: "m"}}) // reconoce los saltos de línea

>db.clientes.find({apellidos:{$regex: "Gó m", $options: "x"}}) // omite los espacios en blancos en la expresión

> db.clientes.insert({nombre: "Karl", apellidos: "Friedrich Hieronymus, \nbarón de Münchhausen"})
WriteResult({ "nInserted" : 1 })

> db.clientes.find({apellidos: {$regex: 'ronymus, ba', $options: "m"}}) // no identifica los caracteres antes del salto de línea

> db.clientes.find({apellidos: {$regex: 'ronymus.*ba', $options: "s"}}) // con la opción s y .* se identifican los caracteres antes del salto de línea

Geospatial Query Operators

$geoIntersects Selects geometries that intersect with a GeoJSON geometry. The 2dsphere index supports $geoIntersects.
$geoWithin Selects geometries within a bounding GeoJSON geometry. The 2dsphere and 2d indexes support $geoWithin.
$near Returns geospatial objects in proximity to a point. Requires a geospatial index. The 2dsphere and 2d indexes support $near.
$nearSphere Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The 2dsphere and 2d indexes support $nearSphere.

$box Specifies a rectangular box using legacy coordinate pairs for $geoWithin queries. The 2d index supports $box.
$center Specifies a circle using legacy coordinate pairs to $geoWithin queries when using planar geometry. The 2d index supports $center.
$centerSphere Specifies a circle using either legacy coordinate pairs or GeoJSON format for $geoWithin queries when using spherical geometry. The 2dsphere and 2d indexes support $centerSphere.
$geometry Specifies a geometry in GeoJSON format to geospatial query operators.
$maxDistance Specifies a maximum distance to limit the results of $near and $nearSphere queries. The 2dsphere and 2d indexes support $maxDistance.
$minDistance Specifies a minimum distance to limit the results of $near and $nearSphere queries. For use with 2dsphere index only.
$polygon Specifies a polygon to using legacy coordinate pairs for $geoWithin queries. The 2d index supports $center.
$uniqueDocs Deprecated. Modifies a $geoWithin and $near queries to ensure that even if a document matches the query multiple times, the query returns the document once.

Array Query Operators

$all Matches arrays that contain all elements specified in the query.
$elemMatch Selects documents if element in the array field matches all the specified $elemMatch conditions.
$size Selects documents if the array field is a specified size.

Bitwise Query Operators

$bitsAllClear Matches numeric or binary values in which a set of bit positions all have a value of 0.
$bitsAllSet Matches numeric or binary values in which a set of bit positions all have a value of 1.
$bitsAnyClear Matches numeric or binary values in which any bit from a set of bit positions has a value of 0.
$bitsAnySet Matches numeric or binary values in which any bit from a set of bit positions has a value of 1.

$comment

The $comment query operator associates a comment to any expression taking a query predicate.

Because comments propagate to the profile log, adding a comment can make your profile data easier to interpret and trace.

The $comment operator has the form:

db.collection.find({ <query>, $comment: <comment> })

Práctica

> db.clientes.find({apellidos:{$regex: "Gó m", $options: "x"}, $comment: "Omite los espacios en blanco"})

Projection Operators

$ Projects the first element in an array that matches the query condition.
$elemMatch Projects the first element in an array that matches the specified $elemMatch condition.
$meta Projects the document's score assigned during $text operation.
$slice Limits the number of elements projected from an array. Supports skip and limit slices.

NOTE find() operations on views do not support the following projection operators:

$
$elemMatch
$slice
$meta

$ (projection)

The positional $ operator limits the contents of an <array> from the query results to contain only the first element matching the query document. To specify an array element to update, see the positional $ operator for updates.

Use $ in the projection document of the find() method or the findOne() method when you only need one particular array element in selected documents.

db.collection.find() operations on views do not support $ projection operator.

Sintaxis

db.collection.find({ <array>: <value> ... }, { "<array>.$": 1 })
db.collection.find({ <array.field>: <value> ...}, { "<array>.$": 1 })

Limitaciones:
• Sólo puede aparecer en un campo del doc de proyección
• Sólo puede aparecer un array en el documento de consulta
• La condición del array del documento de consulta debe ser simple

Práctica

```
> db.scores.insert({jugador:"Pepe", juego:"Tetris", puntos: [79, 102, 89, 101]})
WriteResult({ "nInserted" : 1 })
> db.scores.insert({jugador:"Laura", juego:"Tetris", puntos: [120, 99, 100, 120]})
WriteResult({ "nInserted" : 1 })
```

```
> db.scores.find({juego:"Tetris", puntos: {$gte: 100}}, {"puntos.$": 1})
{ "_id" : ObjectId("5e276a031d4ba0ea8964cdef"), "puntos" : [ 102 ] }
{ "_id" : ObjectId("5e276a1f1d4ba0ea8964cdf0"), "puntos" : [ 120 ] }
```

$elemMatch (projection)

The $elemMatch operator limits the contents of an <array> field from the query results to contain only the first element matching the $elemMatch condition.

Similar a $ pero se le pasa una expresión

Práctica

```
> db.games.insert({title:"Tetris", jugadores: [{nombre: "Pepe", maxPuntuacion: 98},{nombre: "Luisa", maxPuntuacion: 110}, {nombre: "John", maxPuntuacion: 105}]})
WriteResult({ "nInserted" : 1 })
> db.games.find({title: "Tetris"},{jugadores: {$elemMatch: {maxPuntuacion: {$gte: 100}}} })
{ "_id" : ObjectId("5e276d951d4ba0ea8964cdf1"), "jugadores" : [ { "nombre" : "Luisa", "maxPuntuacion" : 110 } ] }
```

$slice (projection)

The $slice operator controls the number of items of an array that a query returns.

db.collection.find() operations on views do not support $slice projection operator.

db.collection.find({ field: value }, { array: {$slice: count } });

Práctica

```
> db.scores.find({},{puntos: {$slice: 1}}) // devuelve el primer elemento del array
{ "_id" : ObjectId("5e276a031d4ba0ea8964cdef"), "jugador" : "Pepe", "juego" : "Tetris", "puntos" : [ 79 ] }
{ "_id" : ObjectId("5e276a1f1d4ba0ea8964cdf0"), "jugador" : "Laura", "juego" : "Tetris", "puntos" : [ 120 ] }

> db.scores.find({},{puntos: {$slice: -2}}) // devuelve los dos últimos
{ "_id" : ObjectId("5e276a031d4ba0ea8964cdef"), "jugador" : "Pepe", "juego" : "Tetris", "puntos" : [ 89, 101 ] }
{ "_id" : ObjectId("5e276a1f1d4ba0ea8964cdf0"), "jugador" : "Laura", "juego" : "Tetris", "puntos" : [ 100, 120 ] }
```

```
> db.scores.find({},{puntos: {$slice: [1,2]}}) // entre array como formato skip-limit: omite el primero y devuelve los dos siguientes
{ "_id" : ObjectId("5e276a031d4ba0ea8964cdef"), "jugador" : "Pepe", "juego" : "Tetris", "puntos" : [ 102, 89 ] }
{ "_id" : ObjectId("5e276a1f1d4ba0ea8964cdf0"), "jugador" : "Laura", "juego" : "Tetris", "puntos" : [ 99, 100 ] }
```