



**Argentina  
programa  
4.0**



Ministerio de Economía  
**Argentina**

Secretaría de  
Economía del Conocimiento

***primero  
la gente***

# CLASE 11: MongoDB II

## Agenda de hoy

- A. Buscar en una Colección
- B. Tipos de búsqueda
  - a. exacta
  - b. parcial
  - c. específicas
  - d. operadores lógicos
- C. Manipular datos
  - a. ordenamiento
  - b. agregar
  - c. modificar
  - d. eliminar
- D. Prácticas



## Buscar en un documento

MongoDB, al igual que el resto de las bases de datos que existen en el mercado del software, cuenta con un mecanismo de búsqueda integrado.

El mismo, es accesible a través de diferentes comandos, los cuales se aplican sobre un campo y el valor específico que deseamos utilizar como búsqueda.

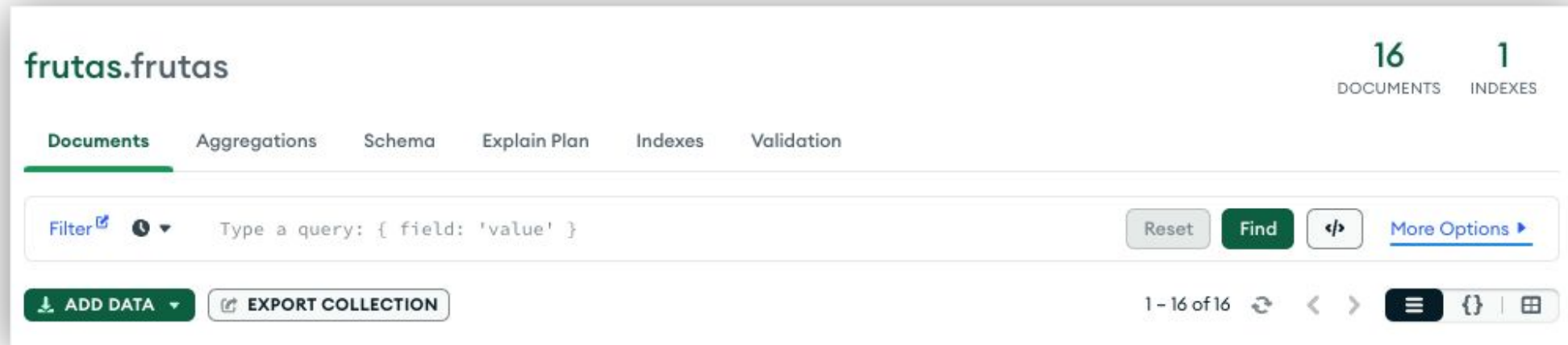


## Buscar en un documento

Si has trabajado con otras bases de datos, como SQL (*MySQL, SQL Server, Access, Oracle, etc*), seguramente encontrarás una equivalencia en los comandos que utiliza MongoDB para la búsqueda y filtrado de información.



# El concepto de base de datos



MongoDB Atlas cuenta con un panel de búsqueda donde encontraremos un set de herramientas bastante completo para poder ser asertivos con el proceso de búsqueda y filtrado de información.

# El concepto de base de datos

frutas.frutas

16 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find  [Less Options ▼](#)

Project { field: 0 }

Sort { field: -1 } or [['field', -1]] MaxTimeMS 60000

Collation { locale: 'simple' } Skip 0 Limit 0

ADD DATA EXPORT COLLECTION

1 - 16 of 16

Este panel puede ampliarse para ver en detalle las diferentes combinaciones que podemos aplicar a una búsqueda: *filtrar, ordenar, limitar el resultado, etcétera*.

# **Tipos de busqueda**



# Tipos de búsqueda

MongoDB cuenta con una amplia variedad de parámetros para filtrar datos en una base de datos. **Algunos de los más comunes son:**

Operador	Descripción
<b>\$eq</b>	Devuelve los documentos que tengan un valor igual al valor especificado.
<b>\$in</b>	Devuelve los documentos que tengan un valor que coincida con alguno de los valores especificados en un array.
<b>\$gt</b>	Permite buscar documentos con valores mayores a un número dado.
<b>\$gte</b>	Permite buscar documentos con valores mayores o iguales a un número dado.
<b>\$lt</b>	Permite buscar documentos con valores menores a un número dado.
<b>\$lte</b>	Permite buscar documentos con valores menores o iguales a un número dado.

# Búsqueda exacta

# Búsqueda exacta

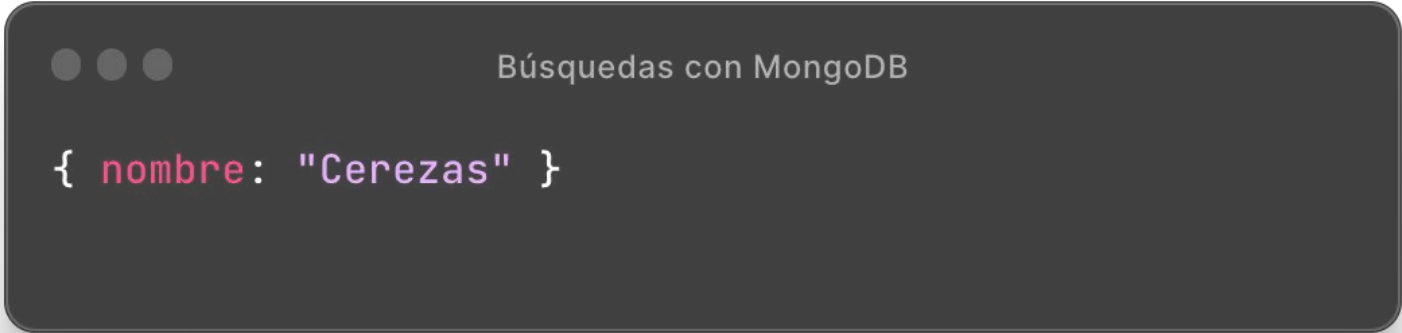
Comencemos a experimentar las diferentes formas de buscar información entre los documentos de MongoDB.

La primera forma que veremos en acción es realizar una búsqueda exacta. Pasaremos un valor específico y MongoDB nos retornará la o las ocurrencias que coincidan con dicha búsqueda.



# Búsqueda exacta

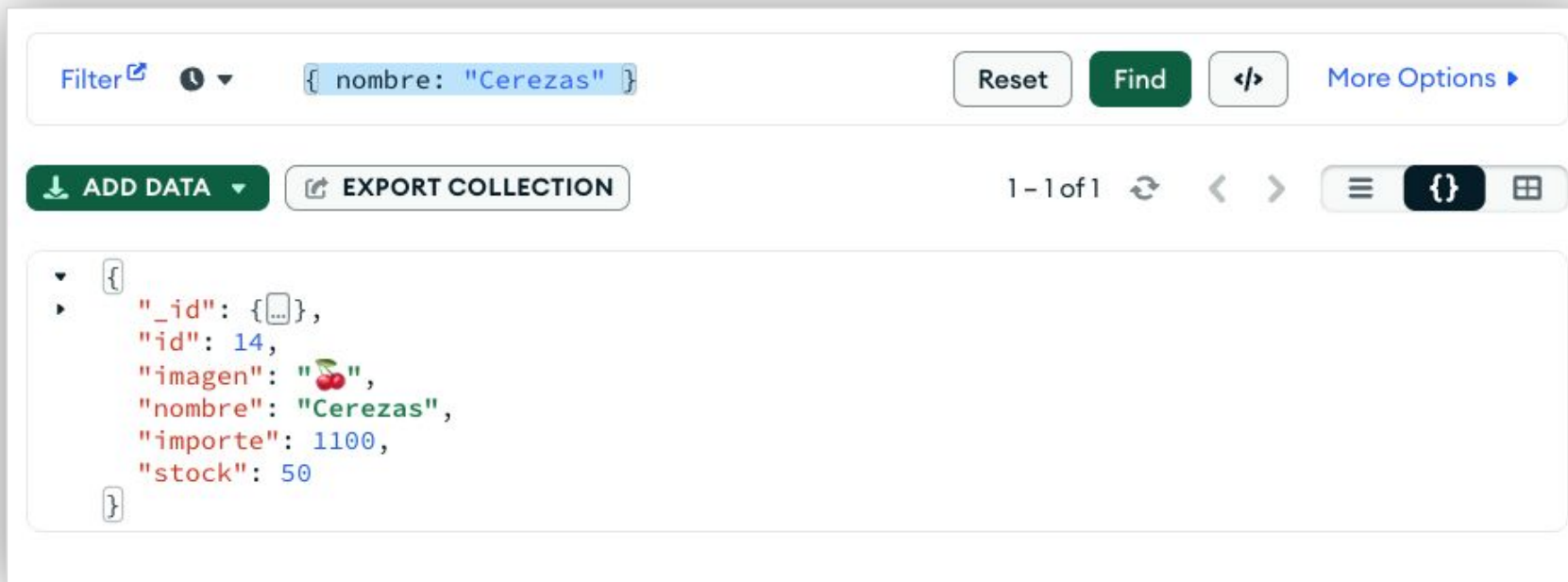
En el campo disponible al lado del apartado Filter, escribimos la siguiente estructura:



```
{ nombre: "Cerezas" }
```

Como resultado, debemos obtener un solo objeto con toda la información asociada a éste.

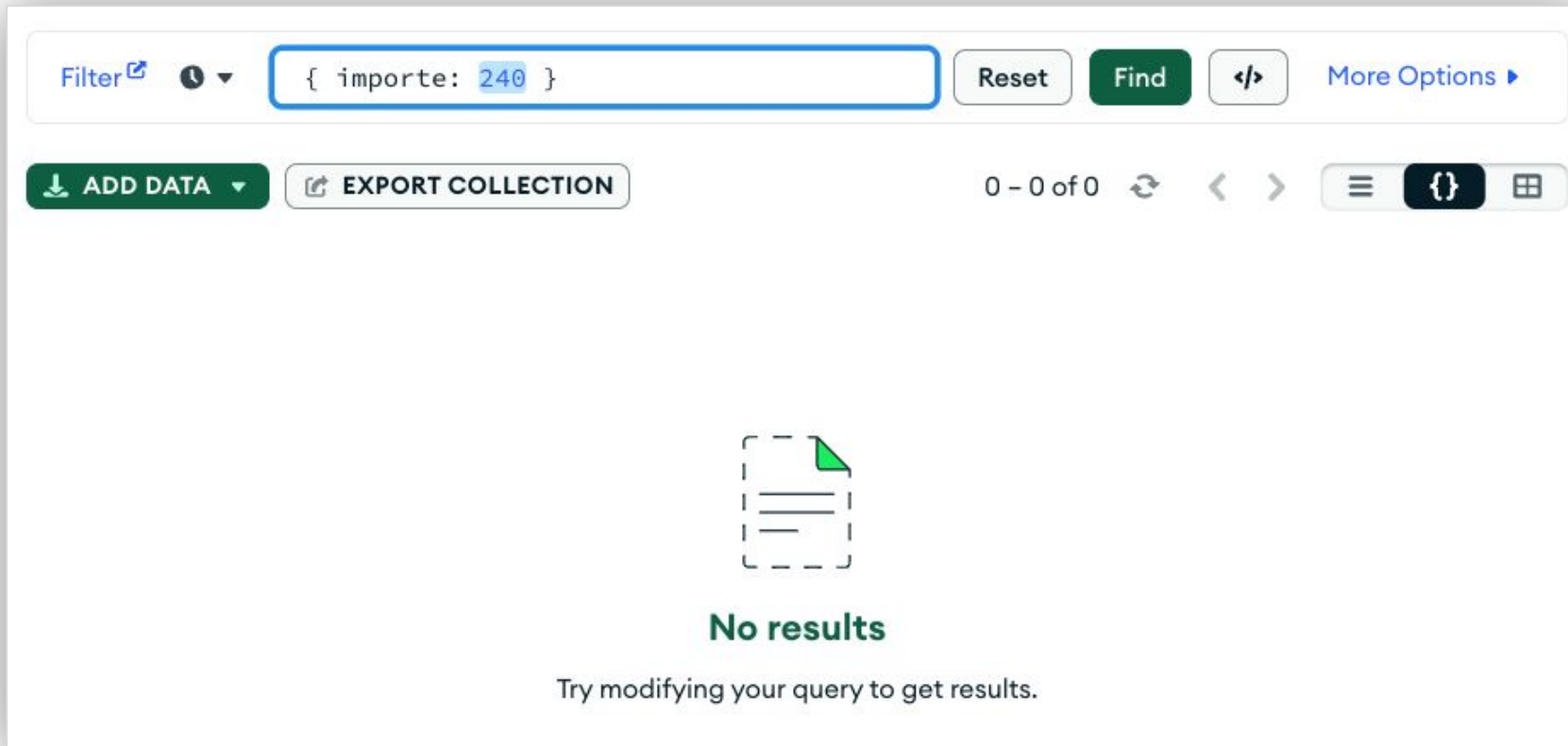
# Búsqueda exacta



De esta forma podemos observar la respuesta de MongoDB, visualizando el objeto que coincide con nuestra búsqueda especificada.

**Toda búsqueda puede realizarse por cualquiera de sus campos.**

# Búsqueda exacta



Si no existen coincidencias con la búsqueda establecida, MongoDB nos devuelve un mensaje de que no halló resultados.

# Búsqueda exacta

The screenshot shows a MongoDB query interface. At the top, a filter bar contains the text `{ importe: 270 }`. To the right of the filter bar are buttons for `Reset`, `Find`, a code icon, and `More Options`. Below the filter bar, there are buttons for `ADD DATA` and `EXPORT COLLECTION`. On the right side, it shows `1 - 2 of 2` results with navigation arrows and icons for list, JSON, and grid views. The results are displayed in a code editor with syntax highlighting. The first result is for a document with `id: 2` and `nombre: "ManzanaR"`. The second result is for a document with `id: 7` and `nombre: "Cocos"`. Both documents have `importe: 270` and `stock: 50`.

```
{
  "_id": {...},
  "id": 2,
  "imagen": "🍏",
  "nombre": "ManzanaR",
  "importe": 270,
  "stock": 50
}
```

```
{
  "_id": {...},
  "id": 7,
  "imagen": "🥥",
  "nombre": "Cocos",
  "importe": 270,
  "stock": 50
}
```

Cuando encuentre más de una ocurrencia, mostrará todas ellas en el panel de resultados, tal como vemos en este otro ejemplo.



## BB.DD. no SQL

Como vimos en el ejemplo inicial, la búsqueda se puede estructurar tanto de forma directa (*imagen superior*), como también utilizando la simbología propia de MongoDB (*imagen inferior*).

**Ambos casos nos llevarán al mismo resultado.**

Búsquedas con MongoDB

```
{ nombre: "Cerezas" }
```

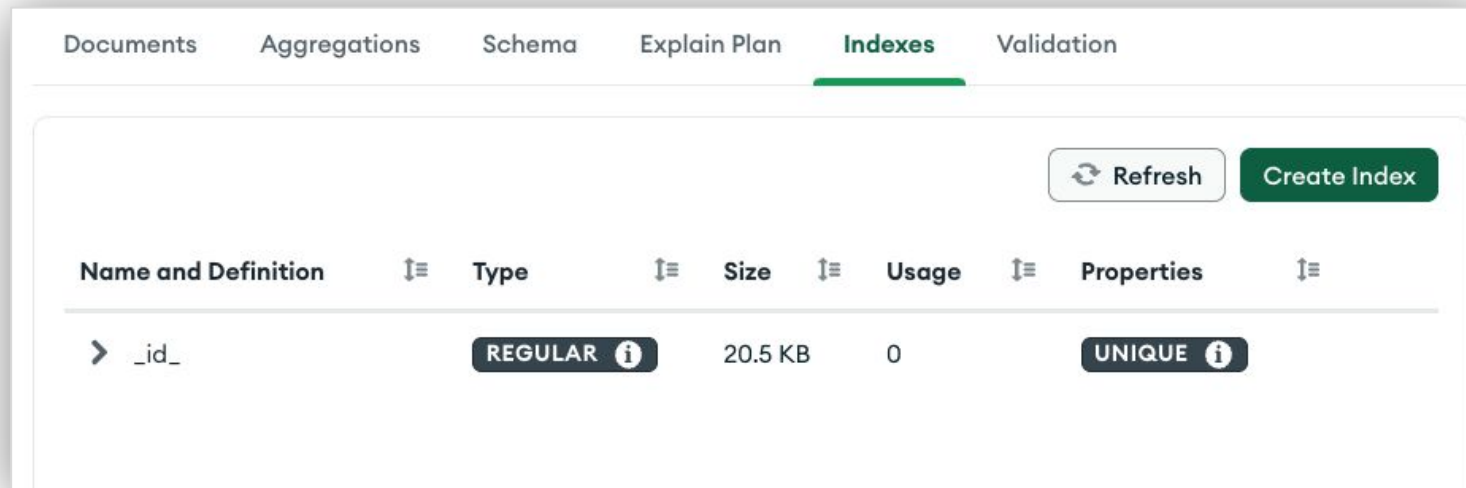
Búsqueda exacta

```
{ nombre: { $eq: "Cerezas" } }
```



# Índices

# Índices



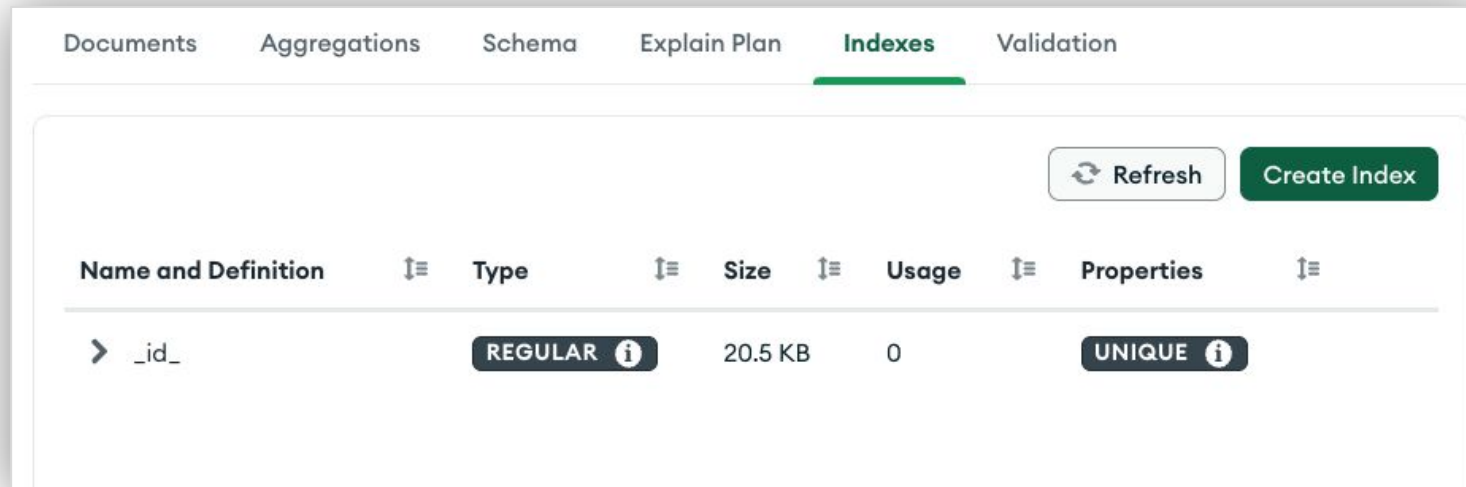
The screenshot shows the 'Indexes' tab in a MongoDB management interface. At the top, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes' (which is selected and highlighted with a green underline), and 'Validation'. Below the tabs, there are two buttons: 'Refresh' (with a circular arrow icon) and 'Create Index' (in a green box). A table below displays the index information. The table has columns: 'Name and Definition', 'Type', 'Size', 'Usage', and 'Properties'. Each column has a sort icon (three horizontal lines) to its left. The table contains one row for the index '\_id\_'. The 'Type' column shows 'REGULAR' with an information icon. The 'Size' column shows '20.5 KB'. The 'Usage' column shows '0'. The 'Properties' column shows 'UNIQUE' with an information icon.

Name and Definition	Type	Size	Usage	Properties
> _id_	REGULAR ⓘ	20.5 KB	0	UNIQUE ⓘ

Por defecto, MongoDB se ocupa de **crear un índice general** para cada colección basado siempre en el **ObjectId** que define con un código unívoco basado en el identificador único universal: **UUID**.

Esto podemos verlo reflejado en el apartado Indexes. Y, en este mismo apartado, podemos sumar otros índices que consideremos necesarios tener.

# Índices



Documents	Aggregations	Schema	Explain Plan	Indexes	Validation
					<button>Refresh</button> <button>Create Index</button>
Name and Definition	Type	Size	Usage	Properties	
> _id_	REGULAR ⓘ	20.5 KB	0	UNIQUE ⓘ	

Sabiendo que nuestro campo id y nuestro campo **nombre** tienen valores unívocos en la Colección, podemos agregar los mismos como índices adicionales, pulsando el botón **Create Index**.

Contar con varios índices ayudará a MongoDB a responder con resultados de una forma más dinámica y rápida.

# Índices

Aunque también debemos tener en cuenta que cada nuevo índice, ocupa un espacio adicional en el motor de la base de datos.

Esto lo podemos ver específicamente en la columna **Size**, de este mismo apartado.

En Colecciones pequeñas no es significativo pero, si estamos ante una Colección de cientos o miles de registros, mientras más índices agregamos más espacio ocupará la indexación en el motor de bb.dd.

Name and Definition	Type	Size
> _id_	REGULAR ⓘ	36.9 KB
> id_1	REGULAR ⓘ	20.5 KB
> nombre_1	REGULAR ⓘ	20.5 KB

# Búsqueda parcial

## Búsqueda parcial {\$gt}



Las búsquedas parciales o aproximadas nos permiten dar con documentos que cumplan una condición similar a la planteada en el parámetro de búsqueda.






```
{ importe: {$gt: 600} }
```

En este ejemplo, queremos visualizar productos cuyo importe sea mayor de 600. **[\$gt = greater than = mayor a]**

# Búsqueda parcial {\$gt}

Filter   { importe: {\$gt: 600 } } Reset Find

 ADD DATA  EXPORT COLLECTION 1 - 3 of 3 

```
{
  "_id": {...},
  "id": 10,
  "imagen": "🍷",
  "nombre": "Arandanos",
  "importe": 650,
  "stock": 50
}
```

```
{
  "_id": {...},
  "id": 14,
  "imagen": "🍒",
  "nombre": "Cerezas",
  "importe": 1100,
  "stock": 50
}
```

```
{
  "_id": {...},
  "id": 12,
  "imagen": "🍇",
  "nombre": "Uvas",
  "importe": 700,
  "stock": 50
}
```

De esta forma, el resultado de la búsqueda cuenta con una probabilidad de que sea múltiple y una baja probabilidad de que sea un único valor el resultante.

Es aplicable tanto a valores numéricos como también a cadenas de texto.

## Búsqueda parcial {\$gte}


En nuestra consulta anterior, la búsqueda sobre el campo **importe** definía como **valores** resultantes todos aquellos que sean **mayores a 600**. Si queremos que los valores de 600 se incluyan en las ocurrencias, debemos utilizar este otro operador.

```
Búsqueda parcial  
  
{ importe: {$gte: 600 } }
```

**[\$gte = greater than or equal to = mayor o igual a]**



# Búsqueda parcial {\$gte}

Filter  { importe: {\$gte: 600 } } Reset Find

ADD DATA EXPORT COLLECTION 1 - 4 of 4

```
{
  "_id": {...},
  "id": 10,
  "imagen": "🍷",
  "nombre": "Arandanos",
  "importe": 650,
  "stock": 50
}
```

```
{
  "_id": {...},
  "id": 14,
  "imagen": "🍒",
  "nombre": "Cerezas",
  "importe": 1100,
  "stock": 50
}
```

```
{
  "_id": {...},
  "id": 12,
  "imagen": "🍇",
  "nombre": "Uvas",
  "importe": 700,
  "stock": 50
}
```

De esta manera podemos concentrar el resultado de la búsqueda con una precisión mayor a la ofrecida por el operador **\$gte**.

## Búsqueda parcial {\$lt}

También podemos realizar búsquedas sobre campos que posean valores menores a un determinado parámetro.



Veamos en este caso todos aquellos productos que poseen un precio **menor a 300**.






```
{ importe: {$lt: 300 } }
```

**[\$lt = less than = menor a]**

# Búsqueda parcial `{>}`

Filter   `{ importe: {>: 600 } }` Reset Find

 ADD DATA  EXPORT COLLECTION 1 - 4 of 4 

```
{
  "_id": {},
  "id": 10,
  "imagen": "🍷",
  "nombre": "Arandanos",
  "importe": 650,
  "stock": 50
}
```

```
{
  "_id": {},
  "id": 14,
  "imagen": "🍒",
  "nombre": "Cerezas",
  "importe": 1100,
  "stock": 50
}
```

```
{
  "_id": {},
  "id": 12,
  "imagen": "🍇",
  "nombre": "Uvas",
  "importe": 700,
  "stock": 50
}
```

De esta manera podemos concentrar el resultado de la búsqueda con una precisión mayor a la ofrecida por el operador `>`.

## Búsqueda parcial {\$lte}

De igual forma que antes, si en el parámetro de búsqueda deseamos incluir el valor indicado como posible valor resultante, entonces debemos utilizar la búsqueda que sea menor o igual a. De esta forma, conseguiremos incluir el parámetro especificado dentro de los resultados de búsqueda.



```
Búsqueda parcial
```

```
{ importe: {$lte: 300 } }
```

**[\$lte = less than or equal to = menor o igual a]**

## Búsqueda parcial {\$in}

Y si necesitamos dar con un set de datos que cumplimenten valores específicos (*estrictos*), entonces utilizamos el operador **IN**.

Como parámetro, incluimos un array de elementos que contengan esos valores específicos que deseamos visualizar.

```
    Búsqueda parcial  
  
{ importe: {$in: [200, 220, 250, 270] } }
```

**[\$in = into = incluya a]**

# Búsqueda parcial {\$in}

Filter   { importe: {\$lt: 300 } } Reset Find

ADD DATA EXPORT COLLECTION 1 - 8 of 8

```
{
  "_id": {},
  "id": 1,
  "imagen": "🍌",
  "nombre": "Bananas",
  "importe": 220,
  "stock": 50
}
```

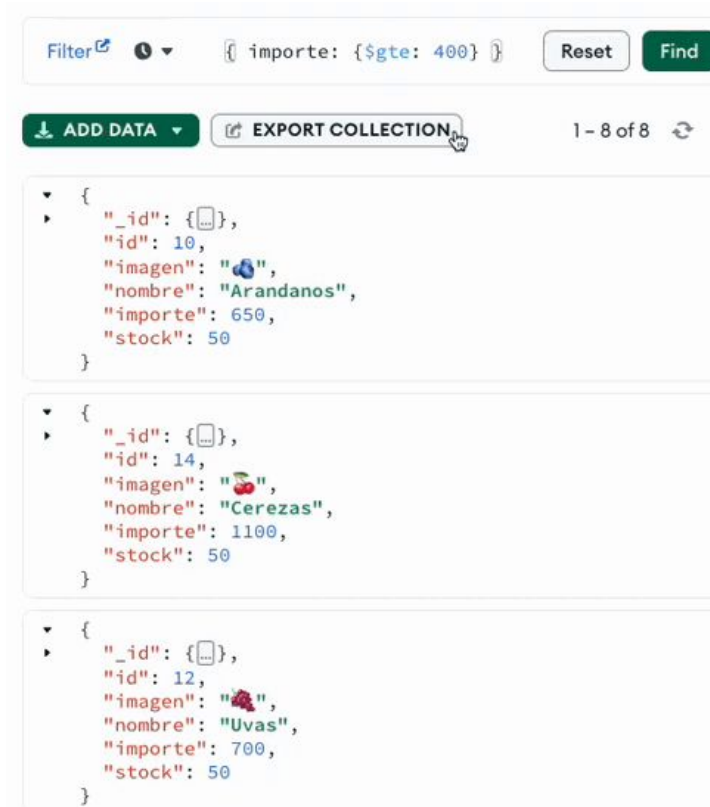
```
{
  "_id": {},
  "id": 6,
  "imagen": "🍅",
  "nombre": "Tomates",
  "importe": 140,
  "stock": 50
}
```

```
{
  "_id": {},
  "id": 8,
  "imagen": "🍉",
  "nombre": "Sandias",
  "importe": 200,
  "stock": 50
}
```

Como resultado de la consulta veremos sólo aquellos documentos que coincidan con los valores indicados en el array de elementos.

Si alguno falta, no veremos ningún mensaje de advertencia.

# Búsqueda parcial {\$gte, \$lte}



Y si deseamos una búsqueda específica de valores que se encuentren entre rango específico, recurrimos a los operadores combinados **mayor o igual a y menor o igual a**.

# **Busquedas específicas**



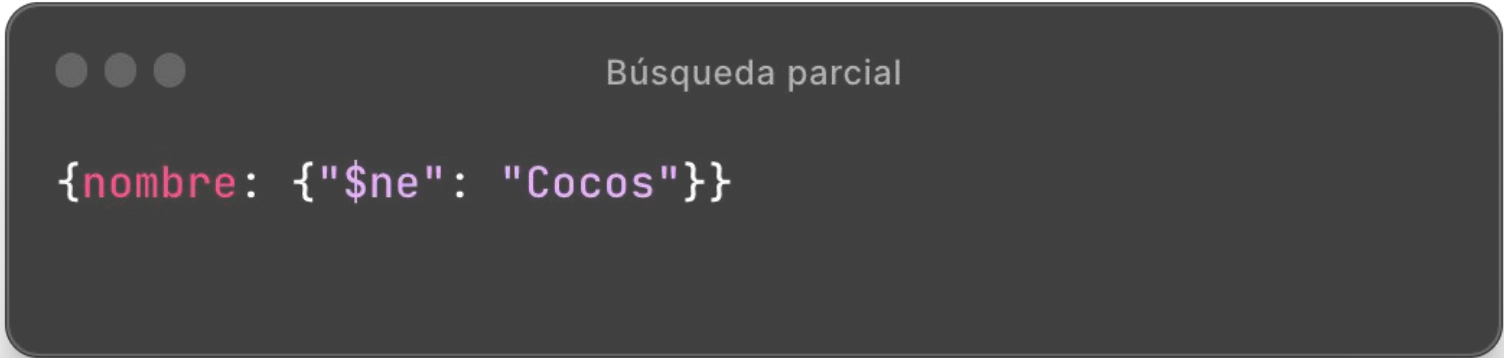
## Búsquedas específicas

Existen también operadores para búsquedas específicas, como ser las búsquedas de negación, utilización de expresiones regulares en lugar de valores, y para identificar si existe un campo específico y/o con un tipo de valor específico. Todo esto apunta a un nivel superior de búsquedas ideal para cuando trabajamos con colecciones que poseen estructuras complejas.

Operador	Descripción
<b>\$ne</b>	Devuelve los documentos que no tengan el valor especificado.
<b>\$nin</b>	Devuelve los documentos que no tengan un valor que coincida con alguno de los valores especificados en un array.
<b>\$regex</b>	Permite buscar documentos que contengan una expresión regular especificada en un campo.
<b>\$exists</b>	Permite buscar documentos que contengan o no un campo específico.
<b>\$type</b>	Permite buscar documentos que contengan un campo específico con un tipo de datos específico.

## Búsqueda parcial {\$ne}

Podemos obviar en los resultados, determinados documentos que posean una condición específica. Para ello, podemos excluirllos en la consulta resultante utilizando un valor estricto junto al operador **\$ne**.



```
{nombre: {'$ne': 'Cocos'}}
```

**[\$ne = not equal to = no sea igual a]**

## Búsqueda parcial {\$nin}

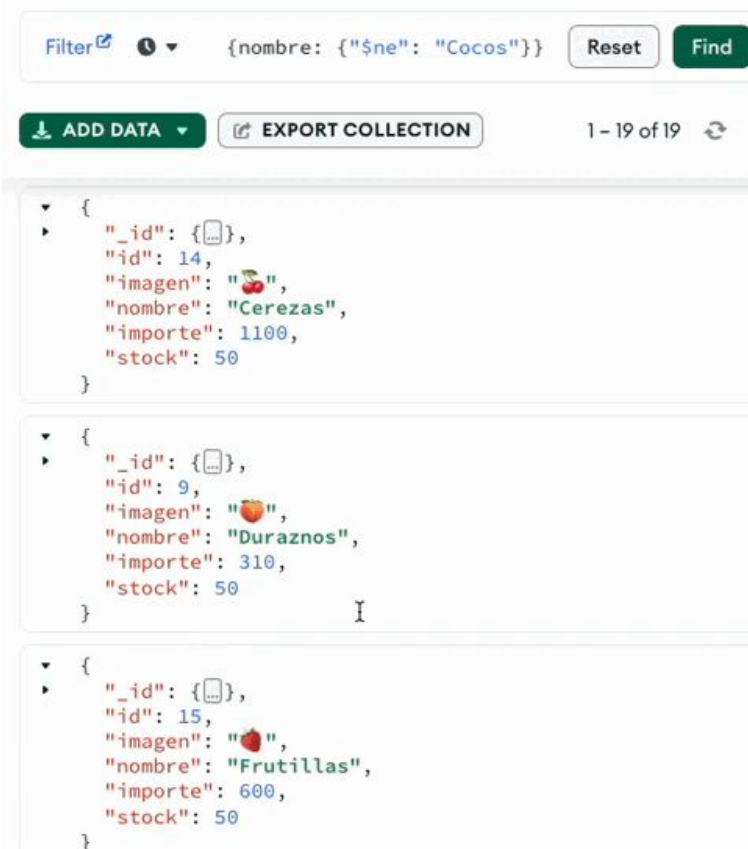
También podemos omitir una serie de documentos que no se encuentren en un rango específico. Esto lo podemos realizar con el operador **\$nin** en conjunto con un array de elementos que especifiquen qué debemos dejar afuera del resultado.

```
Búsqueda parcial

{nombre: {"$nin": ["ManzanaR", "ManzanaV"]}}
```

**[\$nin = not into = no estén entre estos valores]**

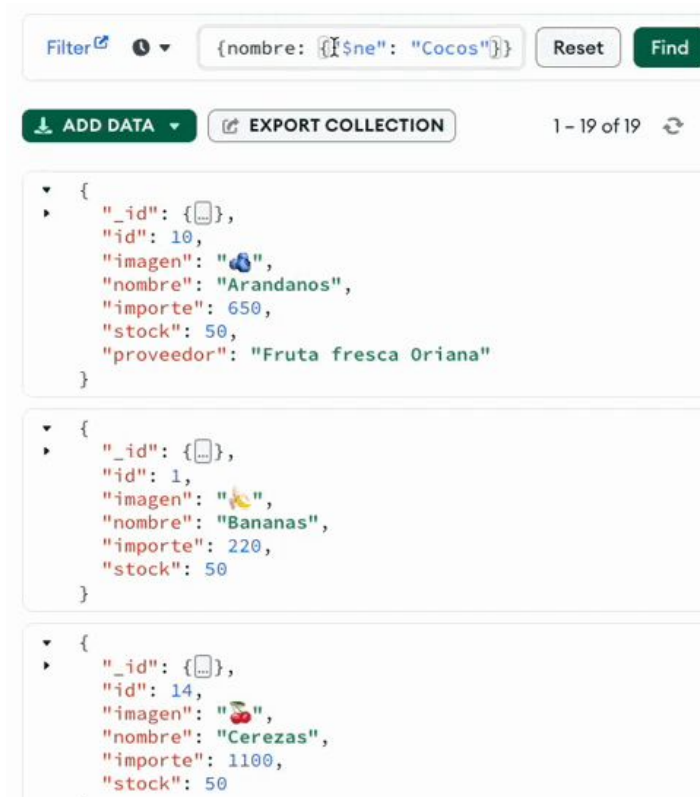
# Búsqueda parcial {\$ne}



La estructura de documentos de MongoDB puede contener en determinadas situaciones campos adicionales que otros tantos documentos de la misma colección, no.

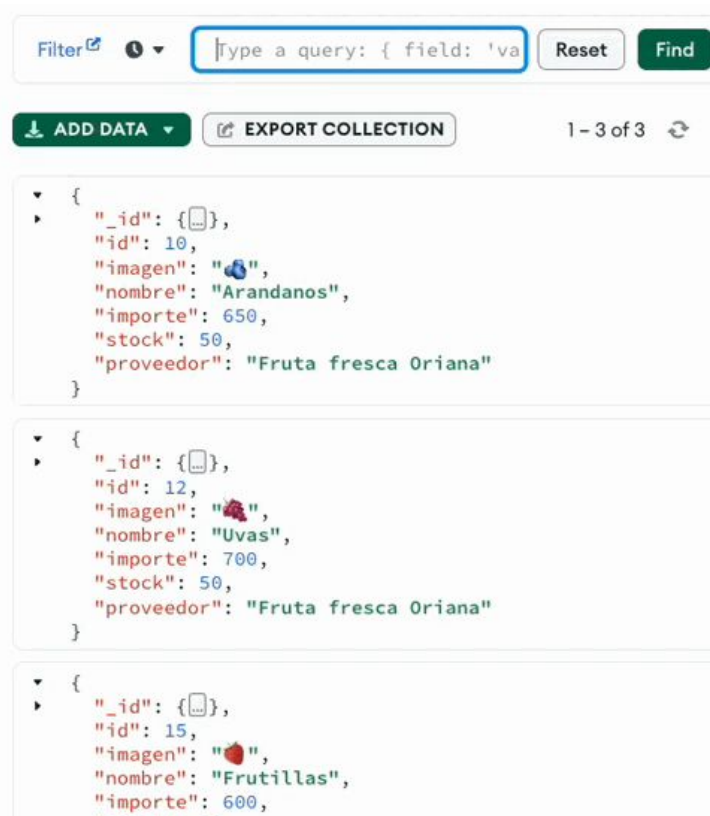
Esto es normal en la estructura de bases de datos NoSQL, y justamente imposible de encontrar este tipo de situaciones en bases de datos relacionales / SQL.

# Búsqueda parcial {\$exists}



En este caso, podemos utilizar el operador **\$exists** el cual nos permitirá visualizar todos aquellos documentos que posean esta propiedad o campo determinado.

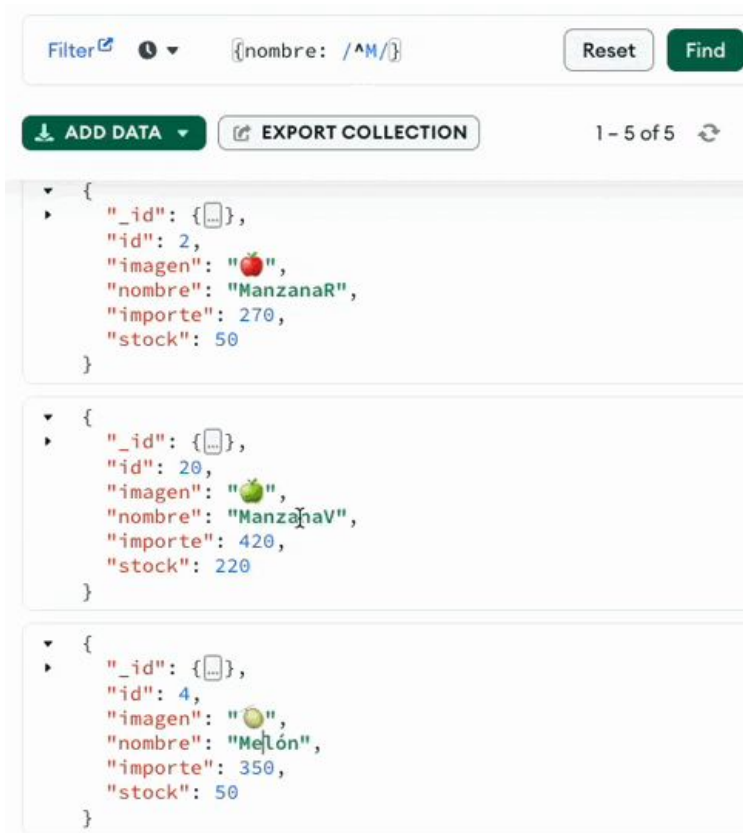
# Búsqueda parcial {\$regex}



Las expresiones regulares pueden implementarse en la búsqueda de datos específicos en una colección de MongoDB.

Por ejemplo, podemos utilizarlas para definir un patrón de búsqueda y obtener todos aquellos documentos cuyo campo nombre comience con una letra específica.

# Búsqueda parcial {\$regex}



El parámetro a buscar se encierra entre los caracteres `//` y anteponiendo el carácter `^` indicamos que el parámetro debe comenzar que lo que luego definimos.

Si queremos buscar documentos con un parámetro que se encuentren en cualquier parte de un nombre, definimos el mismo directamente entre las barras `//`.

# Expresiones Regulares

**El mundo de las Expresiones Regulares es muy amplio y completo. No es indispensable conocerlo pero sí ayuda mucho en tareas de mediana a alta complejidad.**

**Lo bueno de aprender las expresiones regulares es que son transversales a cualquier lenguaje de programación, de marcado, bases de datos relacionales y NoSQL, como también para algunas tareas de administración de infraestructuras IT.**



# Operadores lógicos

# Operadores lógicos

En cada una de las búsquedas, podemos sumar dos o más campos para así obtener un resultado más preciso. Para realizar esto último, podemos integrar los operadores lógicos:

Operador	Descripción
<b>\$and</b>	Permite combinar múltiples condiciones de búsqueda con una operación lógica "AND".
<b>\$or</b>	Permite combinar múltiples condiciones de búsqueda con una operación lógica "OR".
<b>\$not</b>	Permite invertir (o negar) una condición de búsqueda.



# Operador lógico {\$and}

El operador \$and nos permite combinar una búsqueda donde necesitamos que se cumplan dos parámetros, con exactitud o aproximación.



```
Operadores lógicos

{
  $and: [
    { importe: { $gte: 200 } },
    { nombre: /an/ }
  ]
}
```

**[\$and = and = este valor Y este otro]**



# Operador lógico {\$and}

Filter Type a query: { field: 'va' Reset Find

ADD DATA EXPORT COLLECTION 1 - 20 of 20

```
{
  "_id": {...},
  "id": 1,
  "imagen": "🍌",
  "nombre": "Bananas",
  "importe": 220,
  "stock": 50
}
```

```
{
  "_id": {...},
  "id": 10,
  "imagen": "🍷",
  "nombre": "Arandanos",
  "importe": 650,
  "stock": 50,
  "proveedor": "Fruta fresca Oriana"
}
```

```
{
  "_id": {...},
  "id": 14,
  "imagen": "🍒",
  "nombre": "Cerezas",
  "importe": 1100,
  "stock": 50
}
```

En este ejemplo expresamos dos condiciones específicas:

- que el **importe** sea **mayor o igual a 200**

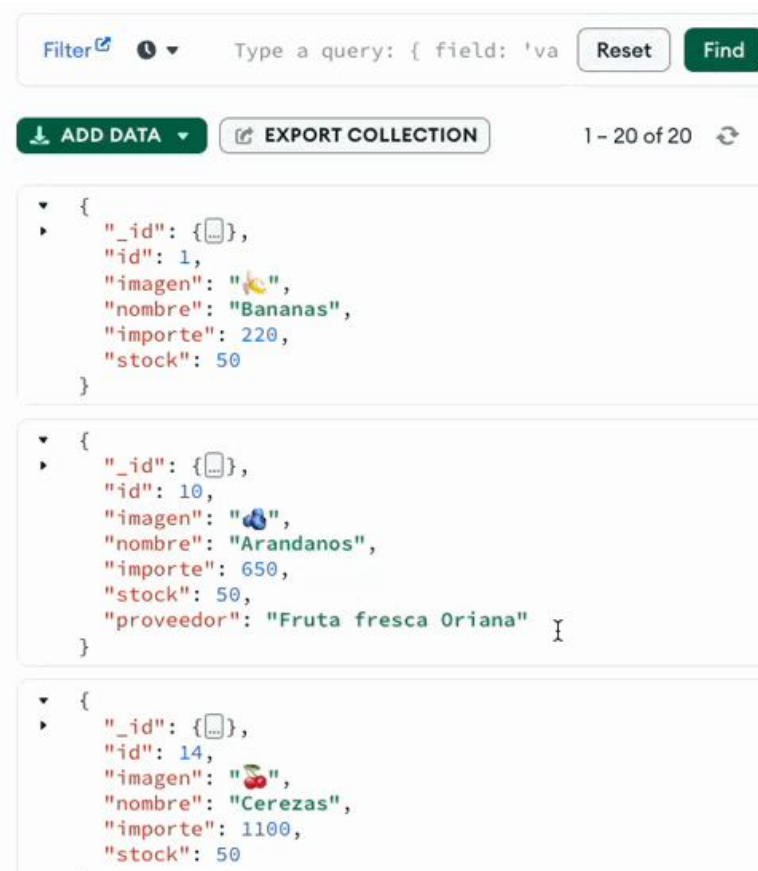
Y

- que el **nombre** contenga **/na/**

Tengamos presente la estructura utilizada para aplicar el operador en cuestión.



# Operador lógico {\$or}



En este ejemplo expresamos dos condiciones específicas:

- que el **importe** sea **mayor o igual a 200**
- Ó
- que el **nombre** sea mayor o igual a **“P”**

Como no existen importes con dicha cifra o superiores, **OR** provoca el cortocircuito e irá a buscar todos los documentos cuyo nombre comience o sea mayor a **“P”**.



# Operador lógico {\$nor}

Filter Type a query: { field: 'va' Reset Find

ADD DATA EXPORT COLLECTION 1 - 20 of 20

```
{
  "_id": {...},
  "id": 1,
  "imagen": "🍌",
  "nombre": "Bananas",
  "importe": 220,
  "stock": 50
}
```

```
{
  "_id": {...},
  "id": 10,
  "imagen": "🍷",
  "nombre": "Arandanos",
  "importe": 650,
  "stock": 50,
  "proveedor": "Fruta fresca Oriana"
}
```

```
{
  "_id": {...},
  "id": 14,
  "imagen": "🍒",
  "nombre": "Cerezas",
  "importe": 1100,
  "stock": 50
}
```

Por último, veamos cómo actúa el operador lógico **\$nor**.

Supongamos que queremos obtener todas las frutas que no sean ni *Naranjas* ni *Bananas*; **\$nor** permite excluir aquellos documentos que contengan alguno de estos dos valores en el campo **nombre**.

# **Manipular datos**

# Ordenamiento



ADD DATA ▾

EXPORT COLLECTION

1 - 20 of 20

```
  ▶ {
    "_id": {...},
    "id": 1,
    "imagen": "🍌",
    "nombre": "Bananas",
    "importe": 220,
    "stock": 50
  }
```

```
  ▼ {
    ▶ {
      "_id": {...},
      "id": 10,
      "imagen": "🍷",
      "nombre": "Arandanos",
      "importe": 650,
      "stock": 50,
      "proveedor": "Fruta fresca Oriana"
    }
  }
```

```
  ▼ {
    ▶ {
      "_id": {...},
      "id": 14,
      "imagen": "🍒",
      "nombre": "Cerezas",
      "importe": 1100,
      "stock": 50
    }
  }
```

```
  ▼ {
    ▶ {
      "_id": {...},
      "id": 4,
      "imagen": "🍈",
      "nombre": "Melón"
```

## Ordenamiento

MongoDB define el ordenamiento predeterminado de los objetos que conforman una colección, a través de su campo **\_id**.

Si apreciamos la estructura de nuestro ejemplo gráfico, veremos que el campo **id** propio de esta **Colección**, no es tenido para nada en cuenta en el ordenamiento de estos documentos.

```

  ▶ {
    "_id": {...},
    "id": 1,
    "imagen": "🍌",
    "nombre": "Bananas",
    "importe": 220,
    "stock": 50
  }

```

```

  ▼ {
    ▶ {
      "_id": {...},
      "id": 10,
      "imagen": "🍷",
      "nombre": "Arandanos",
      "importe": 650,
      "stock": 50,
      "proveedor": "Fruta fresca Oriana"
    }
  }

```

```

  ▼ {
    ▶ {
      "_id": {...},
      "id": 14,
      "imagen": "🍒",
      "nombre": "Cerezas",
      "importe": 1100,
      "stock": 50
    }
  }

```

```

  ▼ {
    ▶ {
      "_id": {...},
      "id": 4,
      "imagen": "🍈",
      "nombre": "Melón"
    }
  }

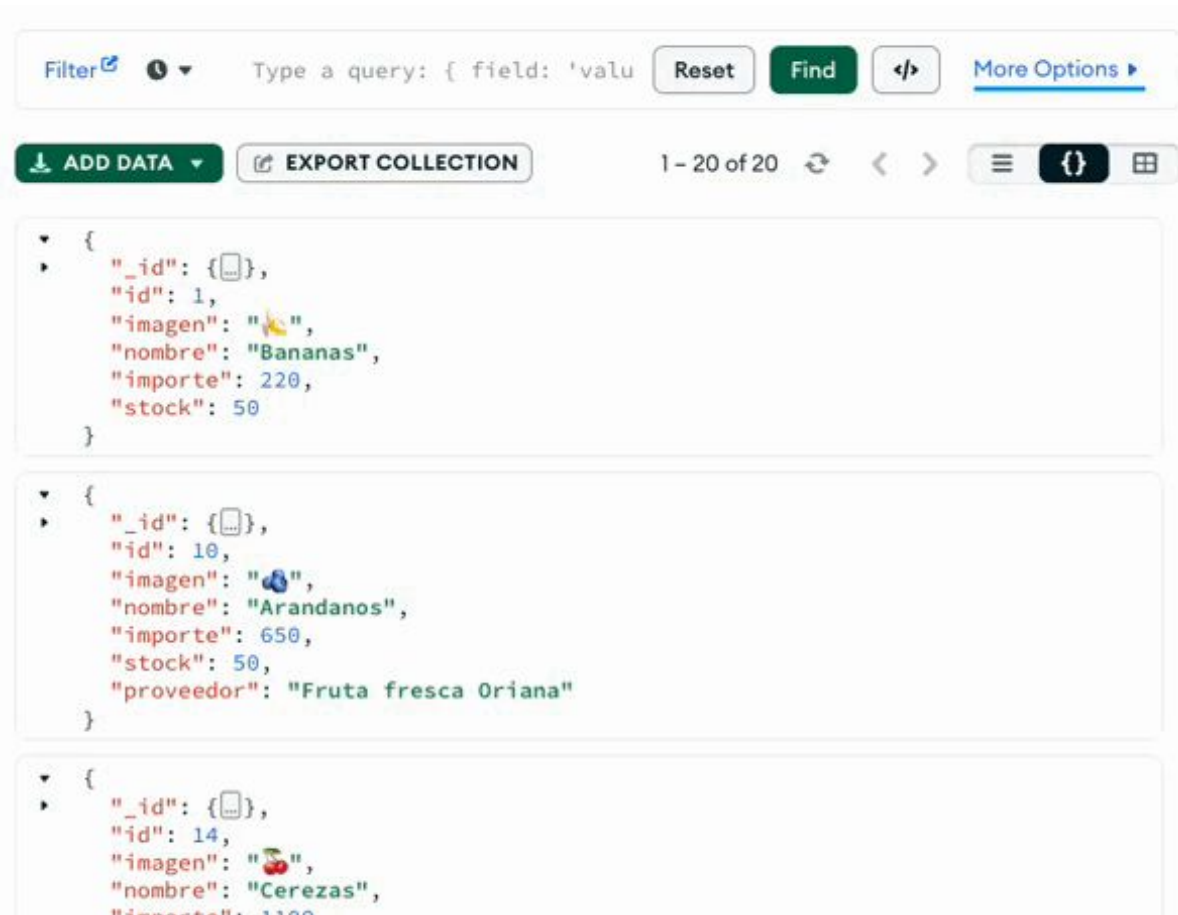
```

## Ordenamiento

Más allá de esto, contamos con una opción de ordenamiento que puede ser definida en base a nuestra necesidad. El mecanismo de orden al cual responde MongoDB es similar al que utiliza JavaScript para ordenar un **array de objetos**: el método **.sort()**.

El mismo se basa en el paradigma clásico de ordenamiento burbuja, que vimos oportunamente en la clase dedicada a las funciones de orden superior.

# Ordenamiento

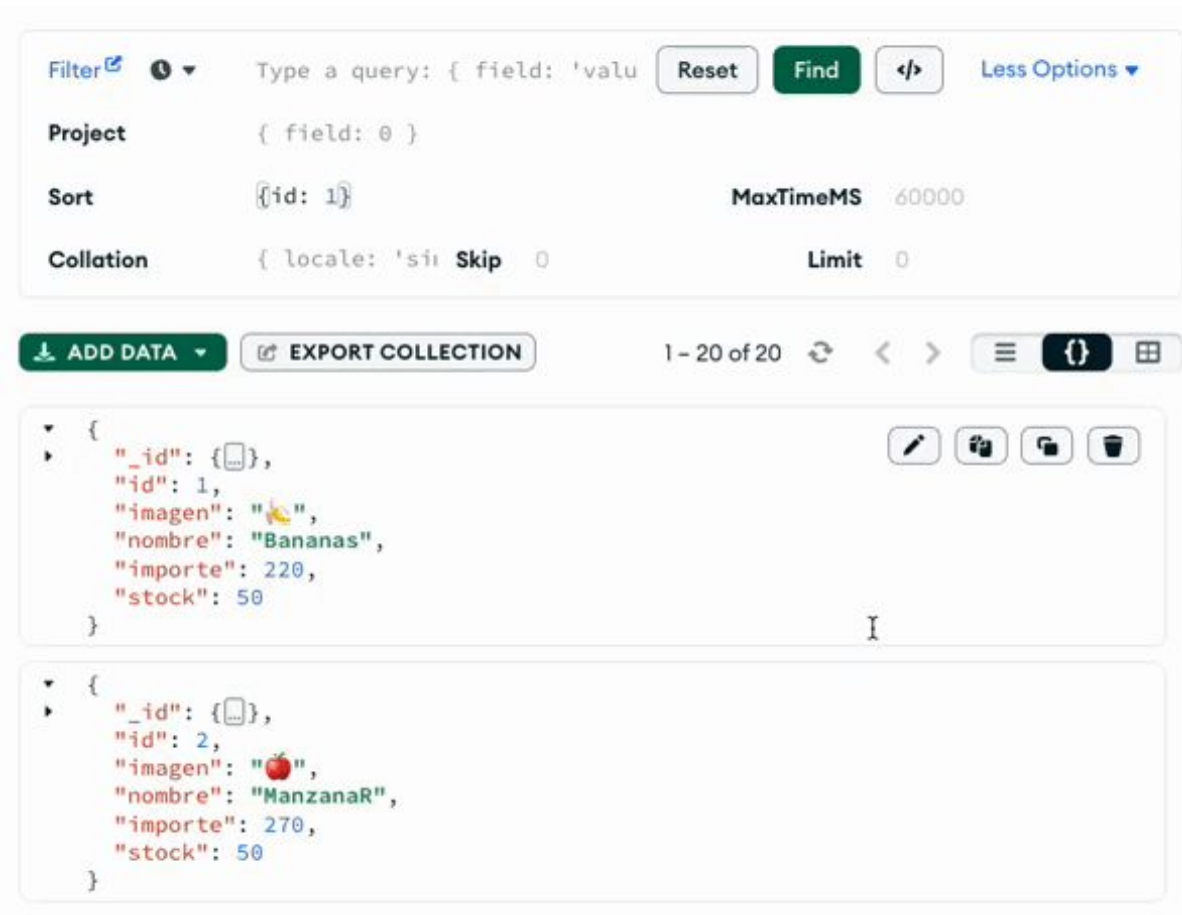


Entonces, si deseamos ordenar por el campo **id** que ya existía dentro de esa Colección previo a ser importada a MongoDB, podemos definirlo a través del panel:

- **More Options**
- **Sort**

Y definir el ordenamiento ascendente indicando el valor **1** como parámetro al campo **id**.

# Ordenamiento



The screenshot shows the MongoDB Compass interface. At the top, there's a 'Filter' section with a query: `{ field: 'valu' }`. Below it, the 'Project' section shows `{ field: 0 }`. The 'Sort' section shows `{ id: 1 }` and 'MaxTimeMS' set to 60000. The 'Collation' section shows `{ locale: 'si' }` and 'Skip' set to 0. The 'Limit' is set to 0. Below the query fields, there are buttons for 'ADD DATA' and 'EXPORT COLLECTION'. The main area displays two documents in a list view, sorted by 'importe' in descending order. The first document has 'importe': 220 and 'stock': 50. The second document has 'importe': 270 and 'stock': 50.

```
{
  "_id": { },
  "id": 1,
  "imagen": "🍌",
  "nombre": "Bananas",
  "importe": 220,
  "stock": 50
}
```

```
{
  "_id": { },
  "id": 2,
  "imagen": "🍏",
  "nombre": "ManzanaR",
  "importe": 270,
  "stock": 50
}
```

Y, al igual que con el método **.sort()**, el ordenamiento descendente lo definimos con el parámetro **-1**.

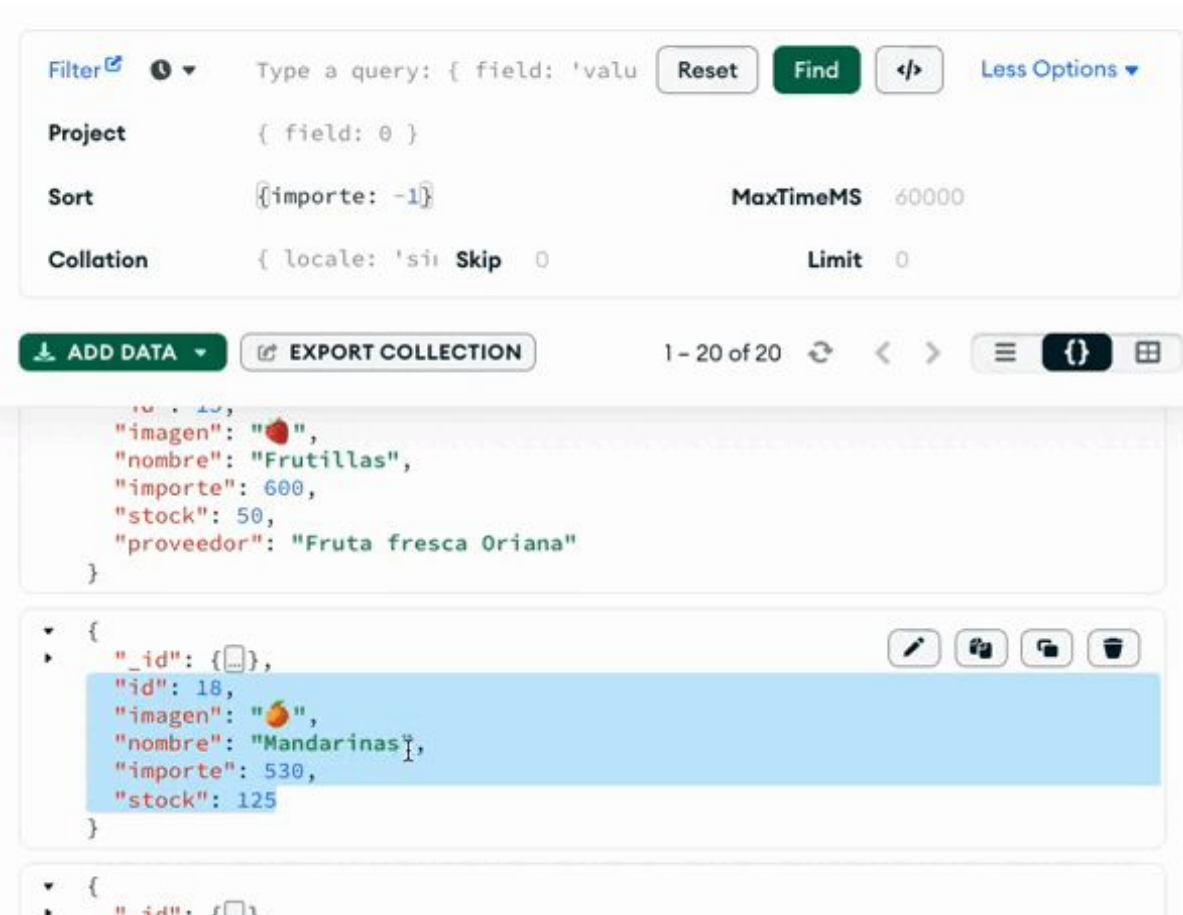
Veamos cómo impacta el ordenamiento de los documentos a través de su campo **importe**, de manera descendente.

**Agregar, Modificar, Eliminar**

# Agregar

Si deseamos agregar un nuevo documento a la Colección, podemos realizar esto utilizando el botón **Add Data > Document**.

En la ventana emergente completamos los campos que deseamos tener, junto con los valores apropiados. De esta forma, simple y práctica, ya tendremos un nuevo documento conformando la **Colección**.



# Modificar

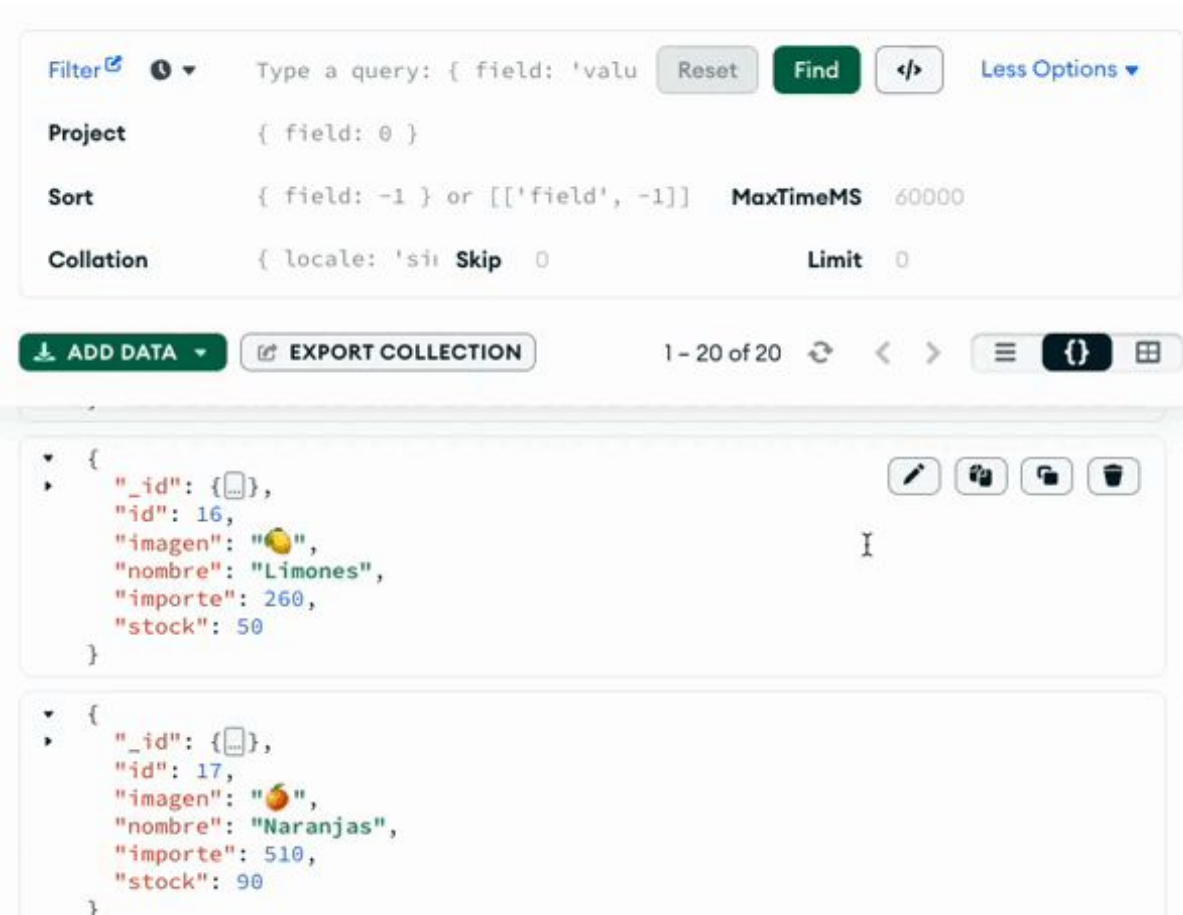
The screenshot shows the MongoDB Compass interface. At the top, there is a query bar with the text "Type a query: { field: 'valu" and buttons for "Reset", "Find", and "Less Options". Below the query bar, there are sections for "Project", "Sort", and "Collation". The "Project" section shows the query "{ field: 0 }". The "Sort" section shows the query "{ field: -1 } or [['field', -1]]" and a "MaxTimeMS" of 60000. The "Collation" section shows the query "{ locale: 'spa' Skip 0" and a "Limit" of 0. Below these sections, there are buttons for "ADD DATA" and "EXPORT COLLECTION". The main area displays a list of documents. The first document is for "Bananas" with an "id" of 1, "imagen" of a banana emoji, "importe" of 220, and "stock" of 50. The second document is for "Arandanos" with an "id" of 10, "imagen" of a blueberry emoji, "importe" of 650, "stock" of 50, and "proveedor" of "Fruta fresca Oriana".

```
{
  "_id": { },
  "id": 1,
  "imagen": "🍌",
  "nombre": "Bananas",
  "importe": 220,
  "stock": 50
}
```

```
{
  "_id": { },
  "id": 10,
  "imagen": "🫐",
  "nombre": "Arandanos",
  "importe": 650,
  "stock": 50,
  "proveedor": "Fruta fresca Oriana"
}
```

Si deseamos modificar algún dato de cualquier documento de la Colección actual, sólo debemos posicionar el mouse sobre el documento a modificar. Pulsamos el botón **Edit** de la barra flotante y, luego del cambio, confirmamos la actualización pulsando el botón **Replace**.

# Eliminar



De igual manera, podremos eliminar un documento ubicando el mismo y pulsando en el ícono **Delete** de la barra de herramientas flotante.

Luego nos queda confirmar la acción para su eliminación definitiva.



# Practicas

Utilizando la aplicación **Compass**, realiza las siguientes consultas sobre la Colección **'frutas'**:

1. Busca las frutas donde su **nombre comienza con 'Manz'** (*recuerda /regex/*)
2. Busca las frutas que tienen un **precio menor a 200**
3. Busca las frutas que tienen un **precio mayor o igual a 500**
4. Buscar las frutas **que tienen los nombres 'Arándanos', 'Cerezas', 'Frutillas'**
5. Buscar las frutas que tienen un **precio mayor o igual a 600** y su **stock superior a 50**
6. Repite el punto 2, agregando un **ordenamiento por el nombre**, de forma ascendente
7. Repite el punto 4, agregando un **ordenamiento por el nombre de forma descendente**
8. repite el punto 5, agregando un **ordenamiento por el id** de forma ascendente



# Muchas gracias.



Ministerio de Economía  
**Argentina**

Secretaría de  
Economía del Conocimiento

*primero  
la gente*