

**PRACTICAFINAL - MongoDB**

**Nombre**: Marco Antonio Roca Montenegro.

**Institución**: Universidad Privada Boliviana.

**Materia**: Bases de datos Avanzadas.

**Docente**: Msc. Carlos Aníbarro.

**Fecha:** 26/06/2023.

Contenido

[INTRODUCCIÓN 3](#_Toc138623434)

[Iniciar 3](#_Toc138623435)

[COMMANDOS – DB: ROCA\_MARCO\_PRACTICAS 4](#_Toc138623436)

[Obtener todos los documentos 4](#_Toc138623437)

[Obtener documentos con writer igual a “Quentin Tarantino” 5](#_Toc138623438)

[Actores igual a “Brad Pitt” 7](#_Toc138623439)

[Documentos con FRANCHISE igual a “The Hobbit” 9](#_Toc138623440)

[Obtener todas las películas de los 90’s 10](#_Toc138623441)

[Obtener películas estrenadas entre 2000 y 2010 11](#_Toc138623442)

[Películas estrenadas en el 2022 11](#_Toc138623443)

[Actualizar documentos 12](#_Toc138623444)

[Agregar sinopsis a "The Hobbit: An Unexpected Journey" 12](#_Toc138623445)

[Agregar sinopsis a "The Hobbit: The Desolation of Smaug" 13](#_Toc138623446)

[Agregar un actor llamado "Samuel L. Jackson" a la película "Pulp Fiction" 14](#_Toc138623447)

[Agregar actores a la película Eternals 14](#_Toc138623448)

[Agregar a MORBIUS, que fue dirigida por “Daniel Espinosa” 16](#_Toc138623449)

[Agregue a 5 films un campo llamado “Rating” 16](#_Toc138623450)

[Búsqueda por texto 18](#_Toc138623451)

[Synopsis que contenga la palabra “Bilbo” 18](#_Toc138623452)

[Synopsis que contenga la palabra “Bilbo” 18](#_Toc138623453)

[Synopsis que contenga la palabra “Bilbo” y no “Gandalf” 19](#_Toc138623454)

[Synopsis que contenga “dwarves” o “hobbit” 19](#_Toc138623455)

[Synopsis que contengan “gold” y “dragon” 20](#_Toc138623456)

[Eliminar documentos 20](#_Toc138623457)

[Eliminar la película "Pee Wee Herman's Big Adventure" y “Morbius” 20](#_Toc138623458)

[PARTE 2 21](#_Toc138623459)

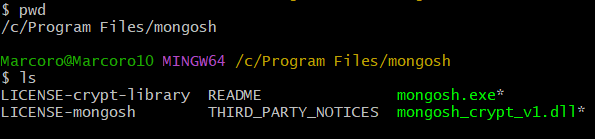
# INTRODUCCIÓN

El presente documento contiene las pruebas realizadas con respecto a los procedimientos y funciones implementados dentro de la base de datos ROCA\_MARCO\_PRACTICA, realizado en MongoDb 🡪 mongoSh V: 1.10.0.

# Iniciar

Los commandos para ingresar correctamente a la base de datos mongo son:

1. Estar en la ruta donde se encuentre “mongosh.exe”.



1. Inicializar mongosh, posteriormente verificar que exista la base de datos.



1. Ingresamos y verificamos que existan los datos.

* use ROCA\_MARCO\_PRACTICA



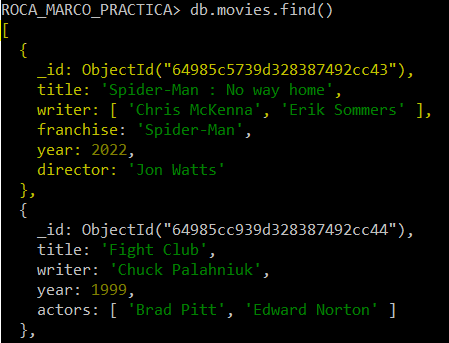
# COMMANDOS – DB: ROCA\_MARCO\_PRACTICAS

## Obtener todos los documentos

El comando para obtener todos los documentos de una colección es el siguiente:

* **db.movies.find()**

Como resultado obtenemos un array, el cual nos muestra los elementos que contiene nuestra colección. Como no hemos definido un “***\_id***”, se les otorga el identificador por defecto. Cada “fila” se la identifica por que se encuentran entre “***{ }*** “ (llaves) y separados por “ ***,*** “ (comas).



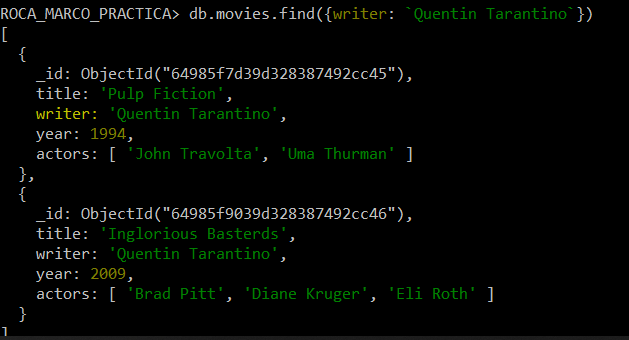
## Obtener documentos con writer igual a “Quentin Tarantino”

Se utiliza de igual manera la función “***find()***”, voy a dejar 2 ejemplos.

1. Buscar ***writer*** con el nombre escrito “tal cual”

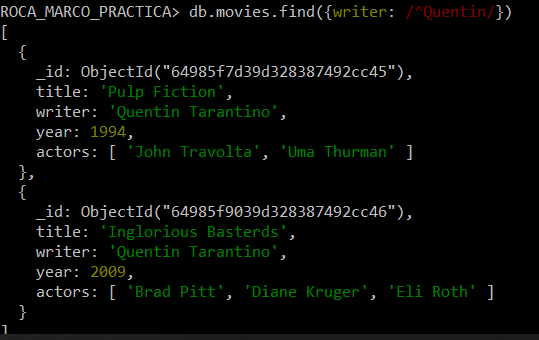
En este caso dentro de la función debemos de especificar en “***FORMATO JSON”***, entre “***{ }****”* y dentro de ellas la **LLAVE**y **VALOR** que queremos encontrar.

* db.movies.find({writer: `Quentin Tarantino`})



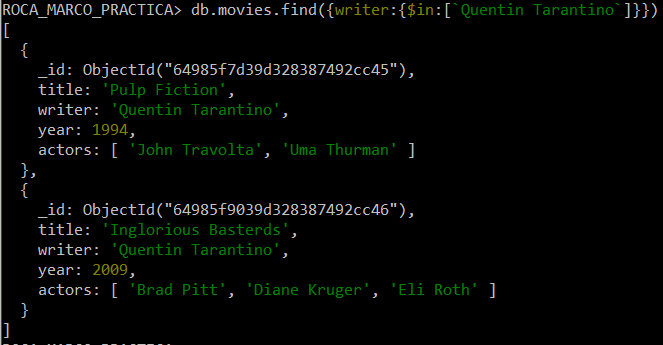
1. Buscar ***writer*** con una expresión regular que empiece por “***Quentin***”

* db.movies.find({writer: /^Quentin/})



1. Buscar ***writer*** que coincida dentro del ***Array***. El utilizar el operador ***$in*** tiene mayor utilizad cuando queremos buscar entre 2 coincidencias de valores dentro de los arrays.

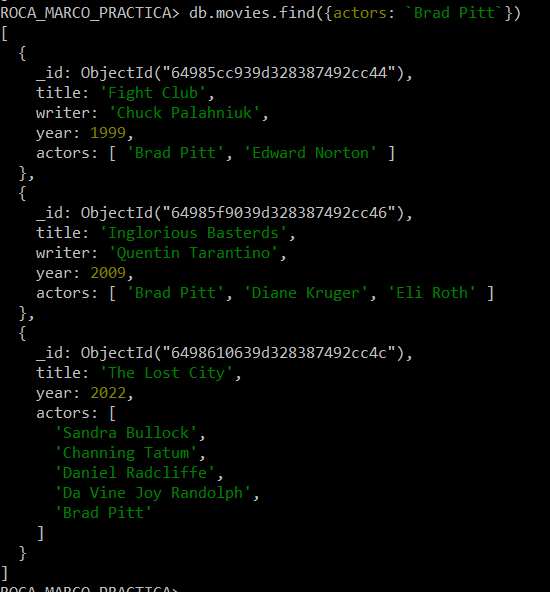
* db.movies.find({writer:{$in:[`Quentin Tarantino`]}})

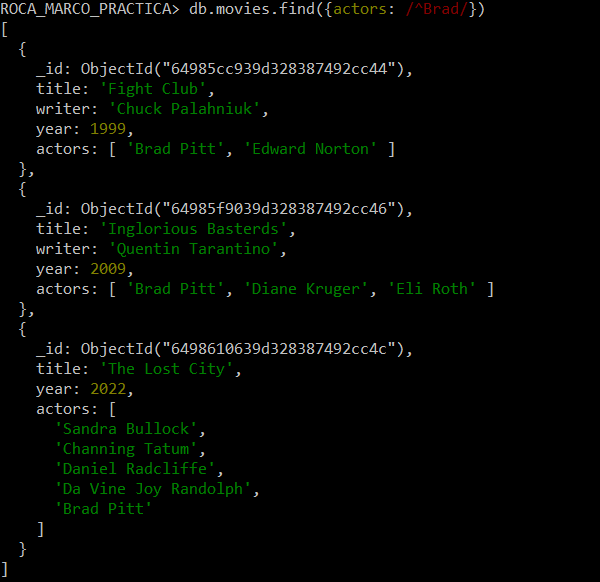


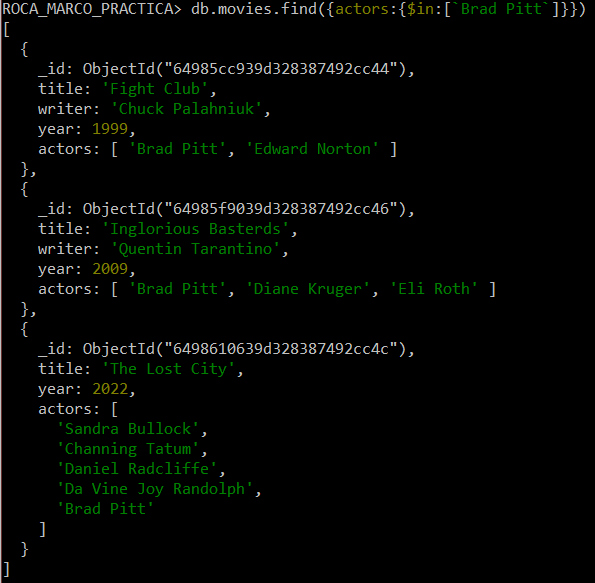
## Actores igual a “Brad Pitt”

* db.movies.find({actors: `Brad Pitt`})
* db.movies.find({actors: /^Brad/})
* db.movies.find({actors:{$in:[`Brad Pitt`]}})

Son dos comandos los cuales buscan dentro de la **LLAVE** (la cual es un array de elementros) el **VALOR** Brad Pitt y el segundo el**VALOR** que comience con Brad.

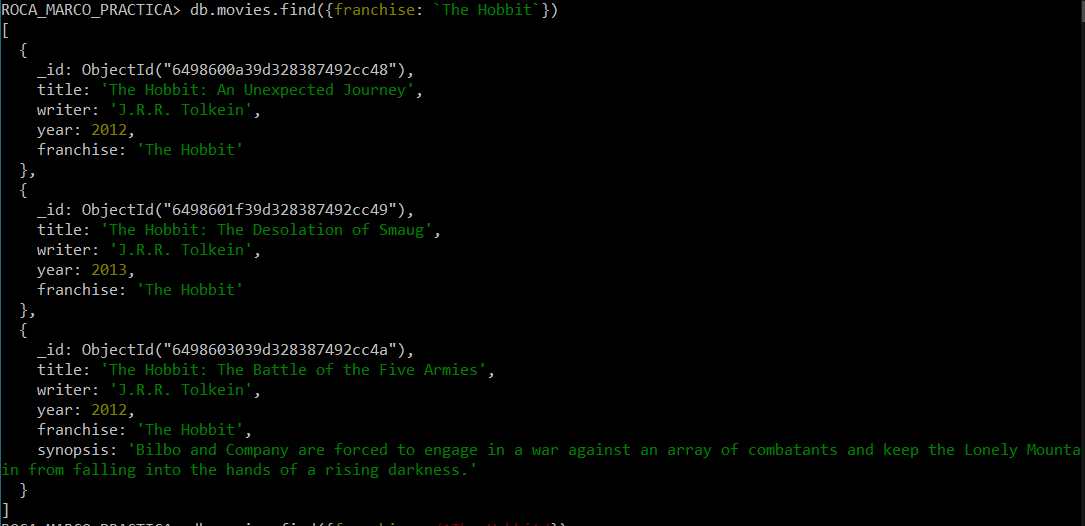


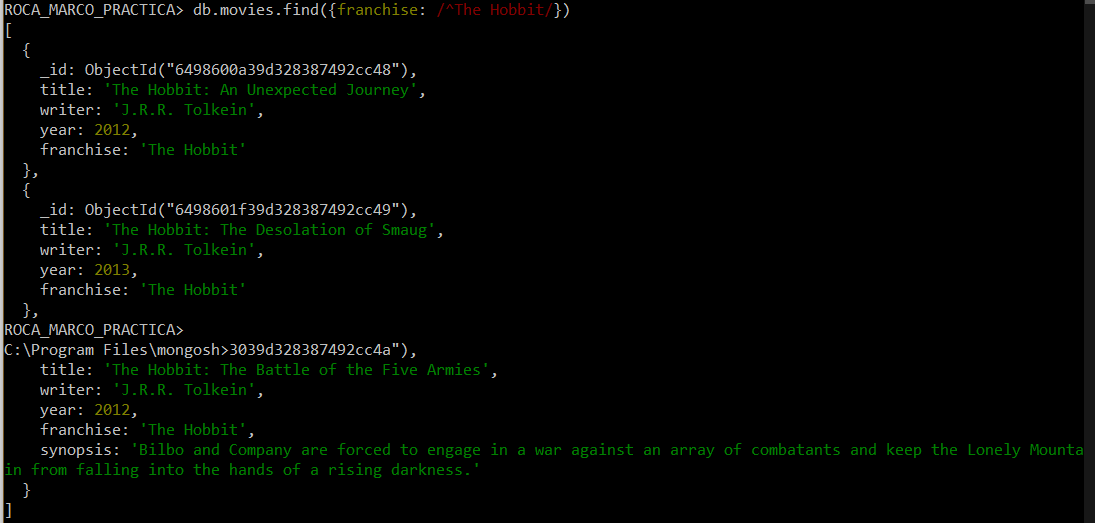


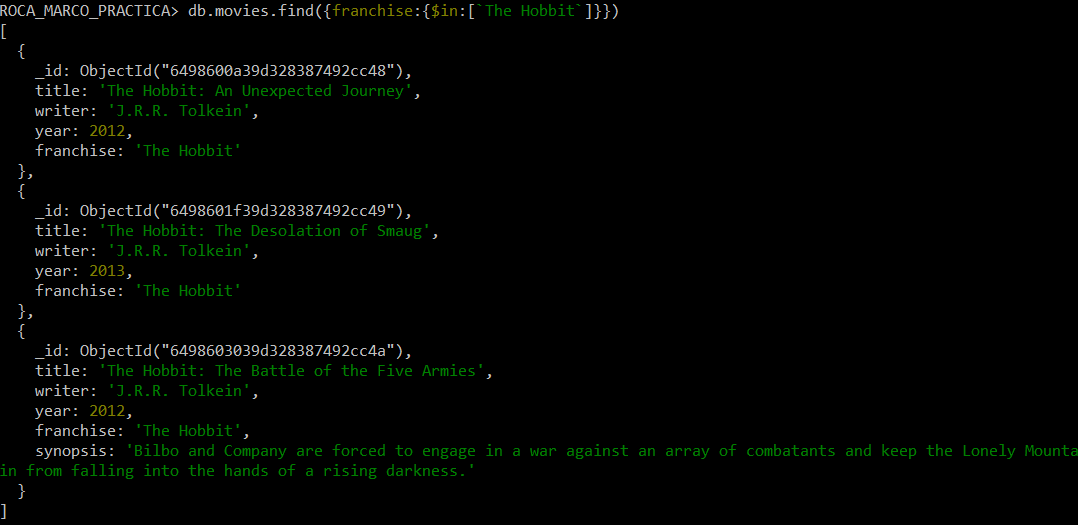


## Documentos con FRANCHISE igual a “The Hobbit”

* db.movies.find({franchise: `The Hobbit`})
* db.movies.find({franchise: /^The Hobbit/})
* db.movies.find({franchise:{$in:[`The Hobbit`]}})



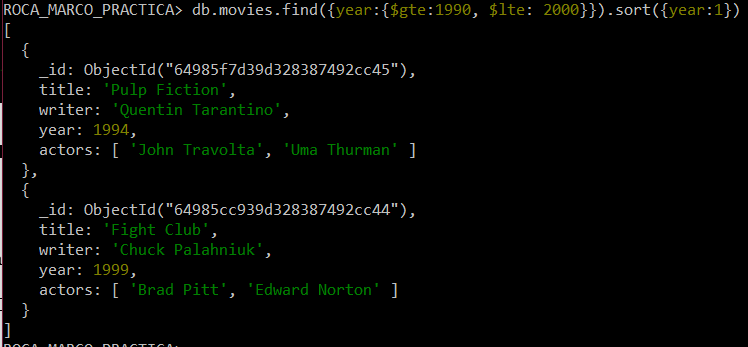




## Obtener todas las películas de los 90’s

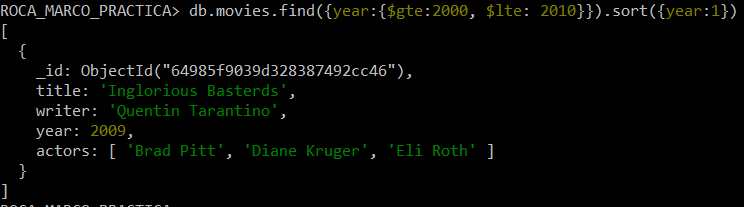
* db.movies.find({year:{$gte:1990, $lte: 2000}}).sort({year:1})

El siguiente comando busca en la **LLAVE** “***year***” valores: MAYORIGUALES a 1990, MENORIGUALES a 2000. Adicionamos el método ***sort: 1*** para ordenarlo de forma Ascendente.



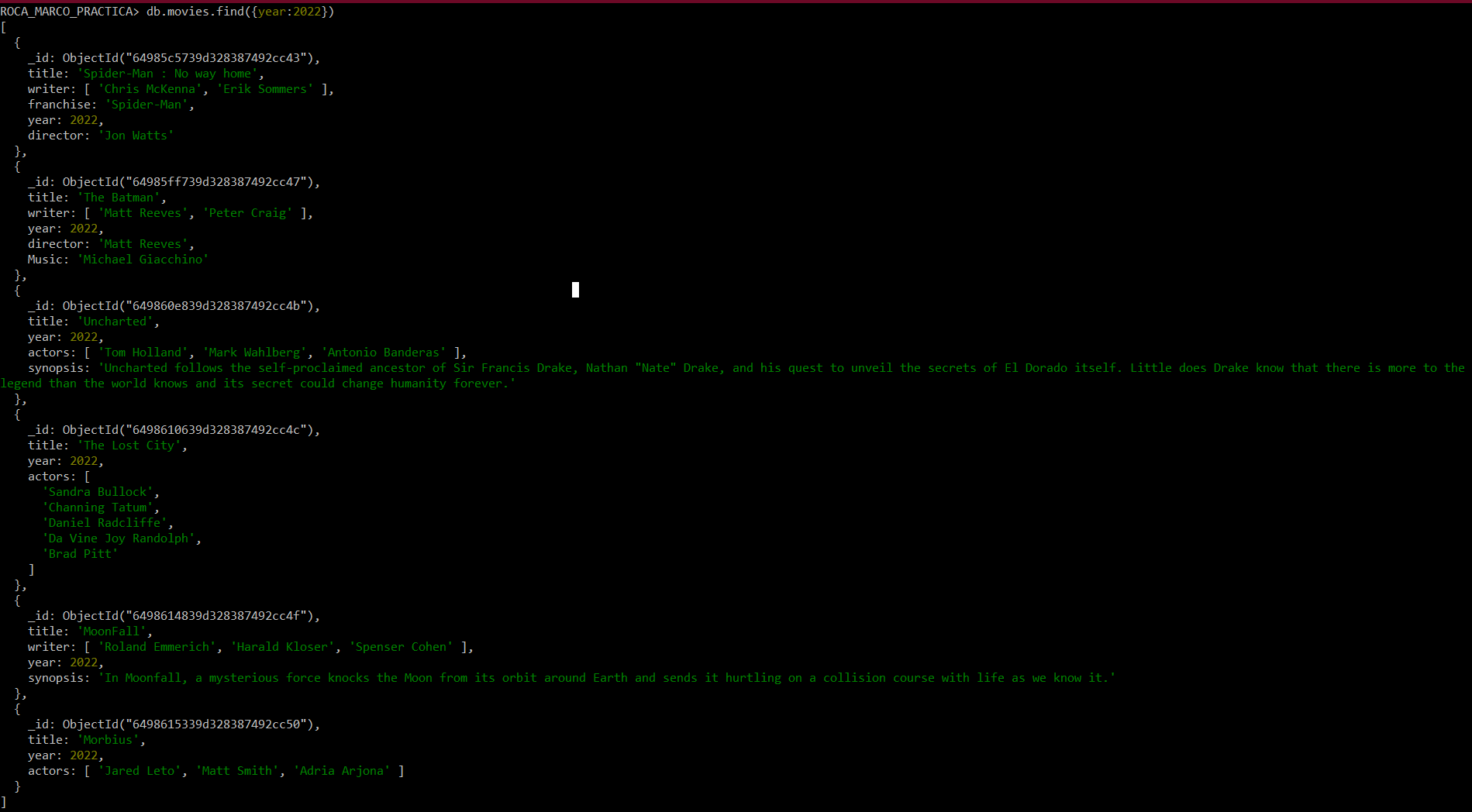
## Obtener películas estrenadas entre 2000 y 2010

* db.movies.find({year:{$gte:2000, $lte: 2010}}).sort({year:1})



## Películas estrenadas en el 2022

* db.movies.find({year:2022})

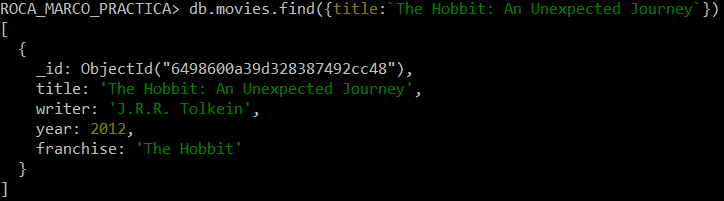


# Actualizar documentos

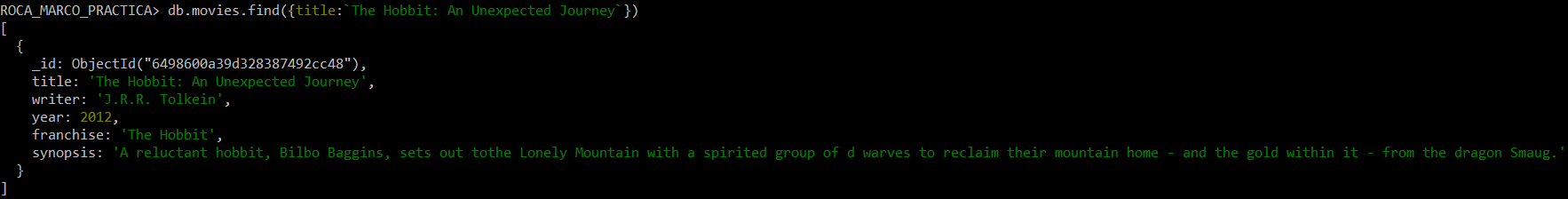
## Agregar sinopsis a "The Hobbit: An Unexpected Journey"

* db.movies.updateOne({title:'The Hobbit: An Unexpected Journey'},{$set: {synopsis: `A reluctant hobbit, Bilbo Baggins, sets out tothe Lonely Mountain with a spirited group of d warves to reclaim their mountain home - and the gold within it - from the dragon Smaug.`}})

Antes:



Después:

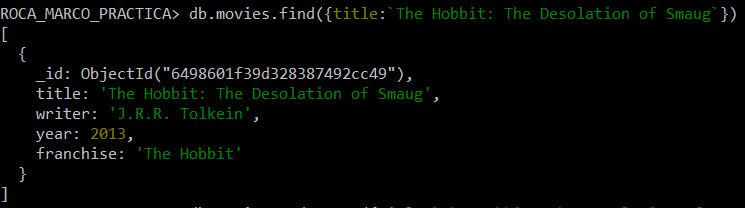


Para agregar elementos utilizamos ***update*** – En este hay que pasarle el parámetro de coincidencia (en este ejemplo es el ***title***) y luego ***$set***- como no existe la **LLAVE** la adiciona.

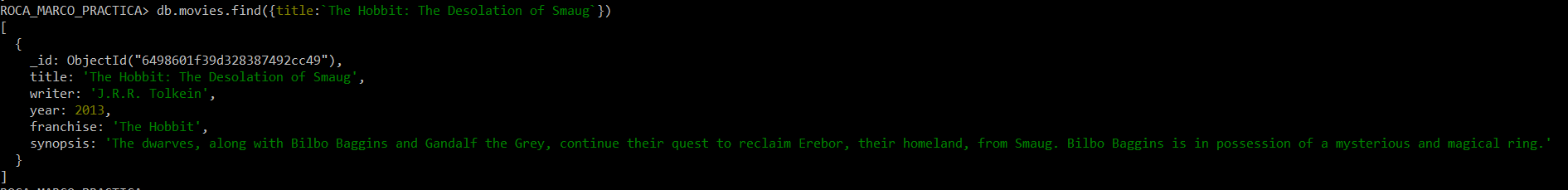
## Agregar sinopsis a "The Hobbit: The Desolation of Smaug"

* db.movies.updateOne({title:`The Hobbit: The Desolation of Smaug`},{$set: {synopsis: `The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring.`}})

Antes:



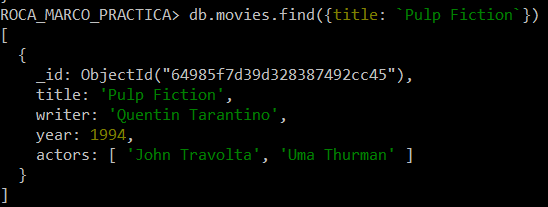
Después:



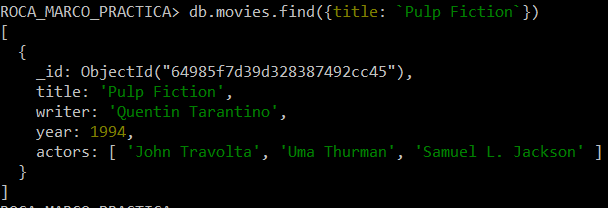
## Agregar un actor llamado "Samuel L. Jackson" a la película "Pulp Fiction"

* db.movies.updateOne({title:`Pulp Fiction`},{$push:{actors:`Samuel L. Jackson`}})

Antes:



Después:

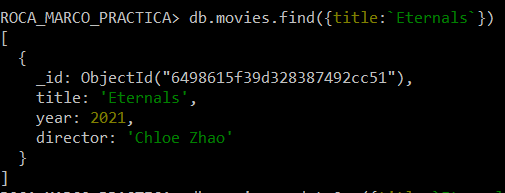


En este ejemplo se utiliza ***$push*** – El cual nos permite adicionar al final del array el nuevo dato.

## Agregar actores a la película Eternals

* db.movies.updateOne({title:`Eternals`},{$push:{actors:{$each: [`Harry Styles`, `Angelina Jolie`, `Kit Harington","Gemma Chan`, `Richard Madden`] }}})

Antes:



Después:

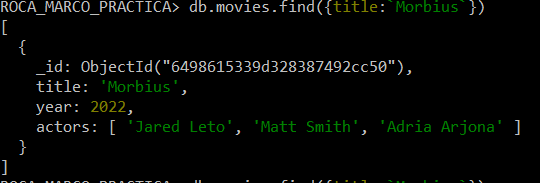


Al contrario de los anteriores ejemplos, en este debemos de insertar más de 1 valor, en este caso hay que utilizar “***$each***” para recorrer todos y cada unos de los valores.

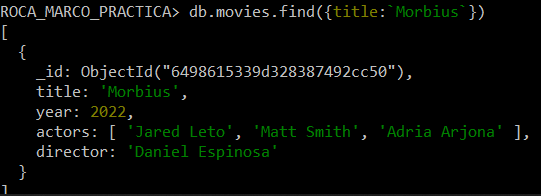
## Agregar a MORBIUS, que fue dirigida por “Daniel Espinosa”

* db.movies.updateOne({title:`Morbius`},{$set:{director:`Daniel Espinosa`}})

Antes:



Después:



## Agregue a 5 films un campo llamado “Rating”

* db.movies.updateOne({title:`Morbius`},{$set:{rating:3}})
* db.movies.updateOne({title:`The Lost City`},{$set:{rating:1}})
* db.movies.updateOne({title:`MoonFall`},{$set:{rating:2}})
* db.movies.updateOne({title:`Eternals`},{$set:{rating:3}})
* db.movies.updateOne({title:'Pee Wee Herman`s Big Adventure'},{$set:{rating:5}})

**Comando para verificar los que contengan “*rating*”**

* db.movies.find({rating:{$exists:true}})

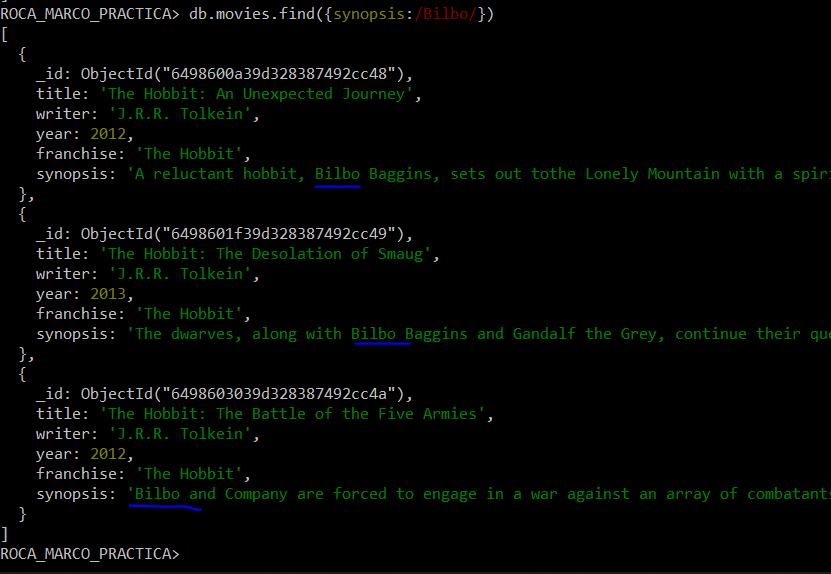


# Búsqueda por texto

## Synopsis que contenga la palabra “Bilbo”

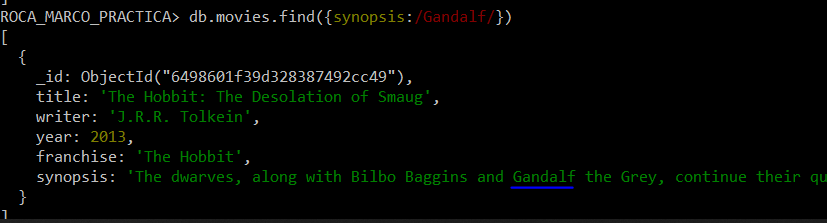
* db.movies.find({synopsis:/Bilbo/})

Esta expresión regular busca coincidencias de valor en cualquier parte de la cadena.



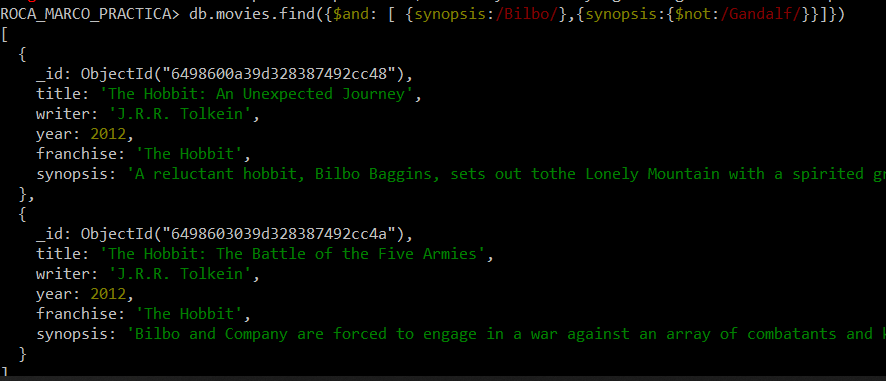
## Synopsis que contenga la palabra “Bilbo”

* db.movies.find({synopsis:/Gandalf/})



## Synopsis que contenga la palabra “Bilbo” y no “Gandalf”

* db.movies.find({$and: [ {synopsis:/Bilbo/},{synopsis:{$not:/Gandalf/}}]})

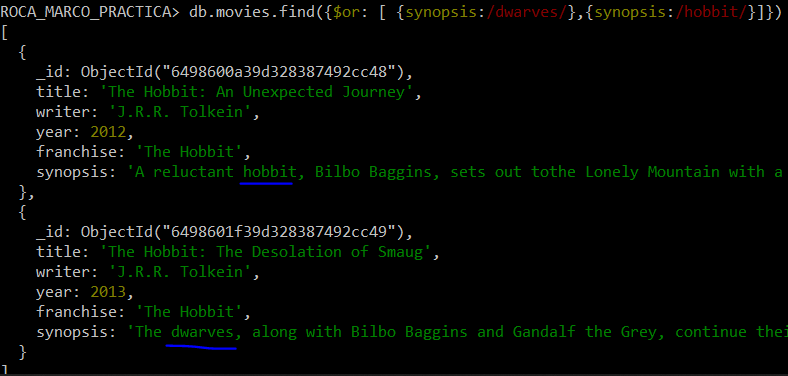


Utilizamos el operador ***$and*** para que se cumpla la condicion [ synopsis 🡪 Bilbo , synopsos 🡪 ***$not*** Gandalf].

El operador ***$not*** nega la condición de contener la palabra Gandalf dentro de la cadena de caracteres.

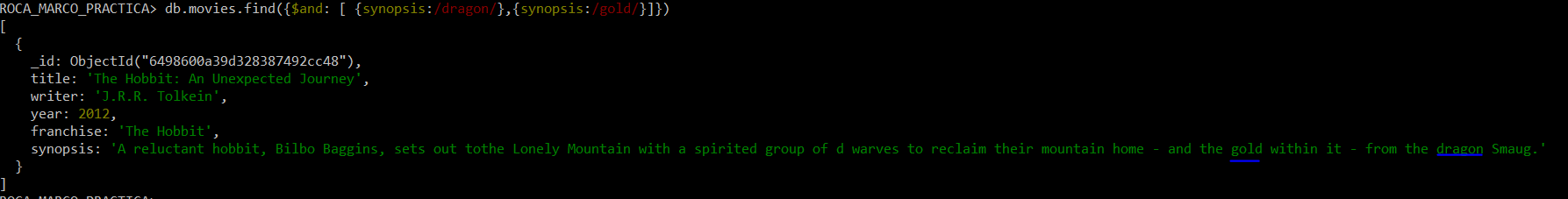
## Synopsis que contenga “dwarves” o “hobbit”

* db.movies.find({$or: [ {synopsis:/dwarves/},{synopsis:/hobbit/}]})



## Synopsis que contengan “gold” y “dragon”

* db.movies.find({$and: [ {synopsis:/dragon/},{synopsis:/gold/}]})



En este ejemplo necesariamente deben contener las palabras gold y dragon.

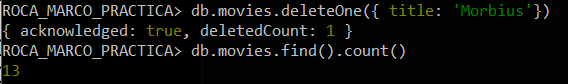
# Eliminar documentos

## Eliminar la película "Pee Wee Herman's Big Adventure" y “Morbius”

* db.movies.deleteOne({ title: ‘Pee Wee Herman`s Big Adventure’ })



* db.movies.deleteOne({ title: 'Morbius'})



# PARTE 2

Para realizar la práctica 2 tuve en mente el trabajar con una base de datos de migración. En esta ocasión opté por datos migratorios debido a las cantidades de personas reportadas por desaparecidas, las cuales pueden ser encontradas vivas años más tarde. ¿Porqué migración?, porque cuando viajé a Chile y Perú me hicieron llenar un formulario migratorio para poder ingresar al país. Eso quiere decir, que toda persona sea mayor de edad o no la cual migra (de manera voluntaria o forzada), debe de generar un registro con sus datos y la fecha de su residencia.

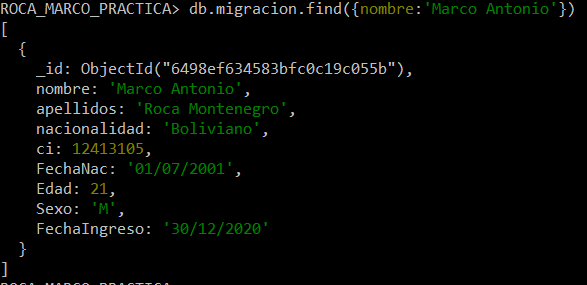
En un mundo utópico, donde los funcionarios y encargados de migración en Bolivia realicen sus funciones de manera efectiva… se podría contar con una base de datos compartida con los organismos judiciales, y en caso de activarse una alerta de desaparición fronteriza (o se tenga información de que estuvo por los países aledaños), buscar y obtener información de manera rápida para reportar si la persona ingresó al país o se encuentra aún dentro de nuestro territorio.

## Comandos

### Búsquedas simples:

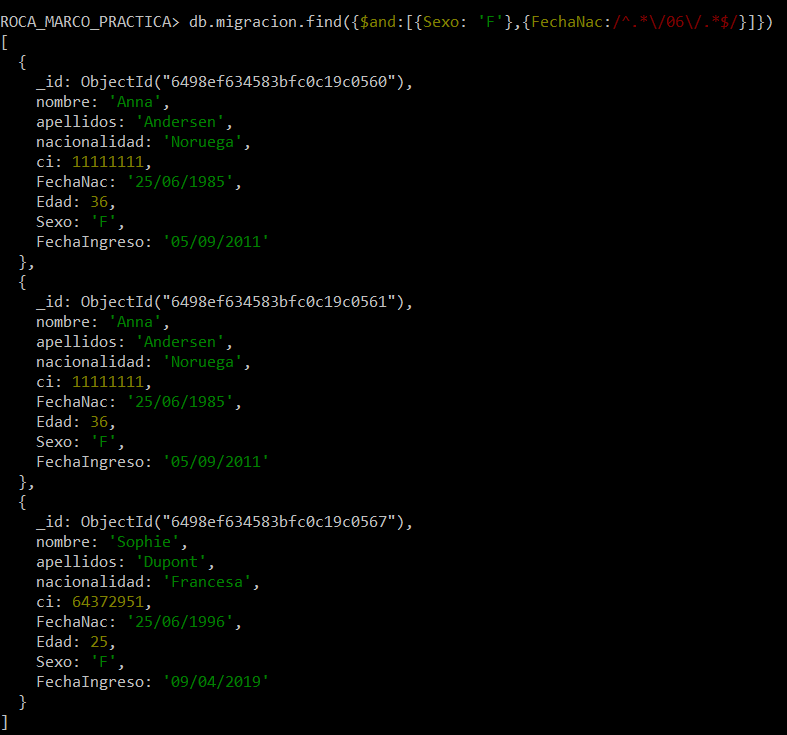
En caso de buscar a una persona por su Nombre.

* db.migracion.find({nombre:'Marco Antonio'})



O por parámetros: que sexo sea ‘F’ y además que hayan nacido en el mes ‘06’.

* db.migracion.find({$and:[{Sexo: 'F'},{FechaNac:/^.\*\/06\/.\*$/}]})



### Comandos con expresiones regulares

Listar la cantidad de todos aquellos que sus apellidos NO contengan López.

* db.migracion.find({apellidos:{$not:/López/}}).count()



O encontrar a aquellas personas que su nombre termine en ‘na’ y sean de sexo ‘F’.

* db.migracion.find({$and:[{nombre:/na$/},{Sexo:{$not:/M/}}]})



### Columnas indexadas

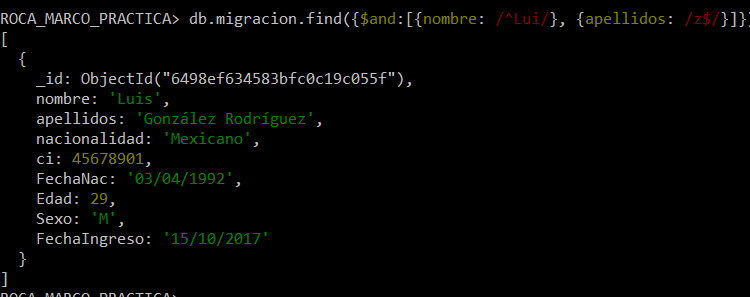
Si esta base de datos estuviese siendo utilizada a la hora de activar una alerta de persona desaparecida los campos más importantes/utilizados en las búsquedas serían: Nombre, Apellido y FechaIngreso. Con un índice múltiple y de caracteres podemos buscar fácilmente las coincidencias. Esto facilitaría en la búsqueda de coincidencia de nombres y también en patrones de fechas comparando mes, año o ambos.

**Cabe recalcar que el orden en que se colocan los índices son muy importantes.**

* db.migracion.ensureIndex({nombre: 1, apellidos: 1, FechaIngreso: 1})

Ejemplo: cuyo nombre empieze con “Lui”, y apellido termine en Z.

* db.migracion.find({$and:[{nombre: /^Lui/}, {apellidos: /z$/}]})



### Comando update

Puede darse el caso de haber registrado mal el carnet de identidad a la hora de transcribir el dato con el formato numérico. De igual forma con las fechas (tanto de Nacimiento como de Ingreso).

Entonces bien, para cambiar el dato del carnet de identidad de Isabel.

db.migracion.updateOne({ nombre: `Marco Antonio` }, { $set:{fechaNac: `01/12/2003`}})

Para cambiar la fecha de nacimiento de Marco Antonio.

* db.migracion.updateOne({ nombre: `Marco Antonio` }, { $set:{fechaNac: `01/12/2003`}})

