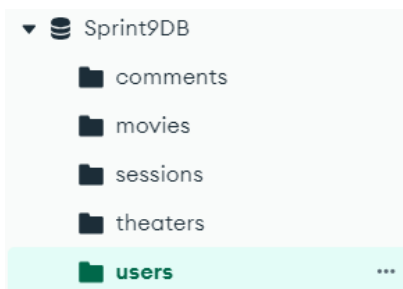


Tasca S9.01. Consultas con MongoDB

Nivel 1

Crea una base de datos con MongoDB utilizando como colecciones los archivos adjuntos.

Primero vamos a crear la base de datos en MongoDB con “Compass” e importaremos los archivos JSON.



Comprobamos que los datos se han importado correctamente a cada colección:

Por ejemplo, en comments:

```
▶ {
  _id: ObjectId('5a9427648b0beebeb69579cc'),
  name: "Andrea Le",
  email: "andrea_le@fakegmail.com",
  movie_id: ObjectId('573a1390f29313caabcd418c'),
  text: "Rem officiis eaque repellendus amet eos doloribus. Porro dolor volupta...",
  date: 2012-03-26T23:20:16.000+00:00
}
```

```
{
  _id: ObjectId('5a9427648b0beebeb69579cf'),
  name: "Greg Powell",
  email: "greg_powell@fakegmail.com",
  movie_id: ObjectId('573a1390f29313caabcd41b1'),
  text: "Tenetur dolorum molestiae ea. Eligendi praesentium unde quod porro. Co...",
  date: 1987-02-10T00:29:36.000+00:00
}
```

- Ejercicio 1

- Muestra los dos primeros comentarios que hay en la base de datos.

Hecho con “Compass”: Buscamos en la tabla ‘comments’ y limitamos a dos:

The screenshot shows the MongoDB Compass interface. The query editor on the left contains the following JSON query:

```
{  
  "Project": { "field": 0 },  
  "Sort": { "field": -1 } or [ [ 'field', -1 ] ],  
  "Collation": { "locale": 'simple' },  
  "Index Hint": { "field": -1 }  
}
```

On the right, the 'Find' button is highlighted. Below the query editor, there are input fields for 'Skip' (0) and 'Limit' (2). The 'Max Time MS' is set to 60000. Below the query editor, there are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. The results pane on the right shows two documents:

```
{  
  "_id": ObjectId('5a9427648b0beebe69579cc'),  
  "name": "Andrea Le",  
  "email": "andrea_le@fakegmail.com",  
  "movie_id": ObjectId('573a1390f29313caabcd418c'),  
  "text": "Rem officiiis eaque repellendus amet eos doloribus. Porro dolor volupta..",  
  "date": 2012-03-26T23:20:16.000+00:00  
},  
{  
  "_id": ObjectId('5a9427648b0beebe69579cf'),  
  "name": "Greg Powell",  
  "email": "greg_powell@fakegmail.com",  
  "movie_id": ObjectId('573a1390f29313caabcd41b1'),  
  "text": "Tenetur dolorum molestiae ea. Eligendi praesentium unde quod porro. Co..",  
  "date": 1987-02-10T00:29:36.000+00:00  
}
```

En consola:

```
> db.comments.find().limit(2)  
< {  
  _id: ObjectId('5a9427648b0beebe69579cc'),  
  name: 'Andrea Le',  
  email: 'andrea_le@fakegmail.com',  
  movie_id: ObjectId('573a1390f29313caabcd418c'),  
  text: 'Rem officiiis eaque repellendus amet eos doloribus. Porro dolor voluptatum voluptates neque culpa molestias. Voluptate unde nulla temporibus ullam.',  
  date: 2012-03-26T23:20:16.000Z  
}  
{  
  _id: ObjectId('5a9427648b0beebe69579cf'),  
  name: 'Greg Powell',  
  email: 'greg_powell@fakegmail.com',  
  movie_id: ObjectId('573a1390f29313caabcd41b1'),  
  text: 'Tenetur dolorum molestiae ea. Eligendi praesentium unde quod porro. Commodi nisi sit placeat rerum vero cupiditate neque. Dolorum nihil vero animi.',  
  date: 1987-02-10T00:29:36.000Z  
}  
Spring9DB >
```

- ¿Cuántos usuarios tenemos registrados?

Compass:

En agregaciones hacemos un \$count:

+

▼ Stage 1

\$count

☑

```

1  ▾ /**
2    * Provide the field name for the count.
3    */
4    'TotalUsers'

```

Output after [\\$count](#) stage (Sample of 1 document)

TotalUsers : 185

Consola:

```

> db.users.aggregate([
  {$count: "TotalUsers"}
])
< {
  TotalUsers: 185
}

```

¿Cuántos cines hay en el estado de California?

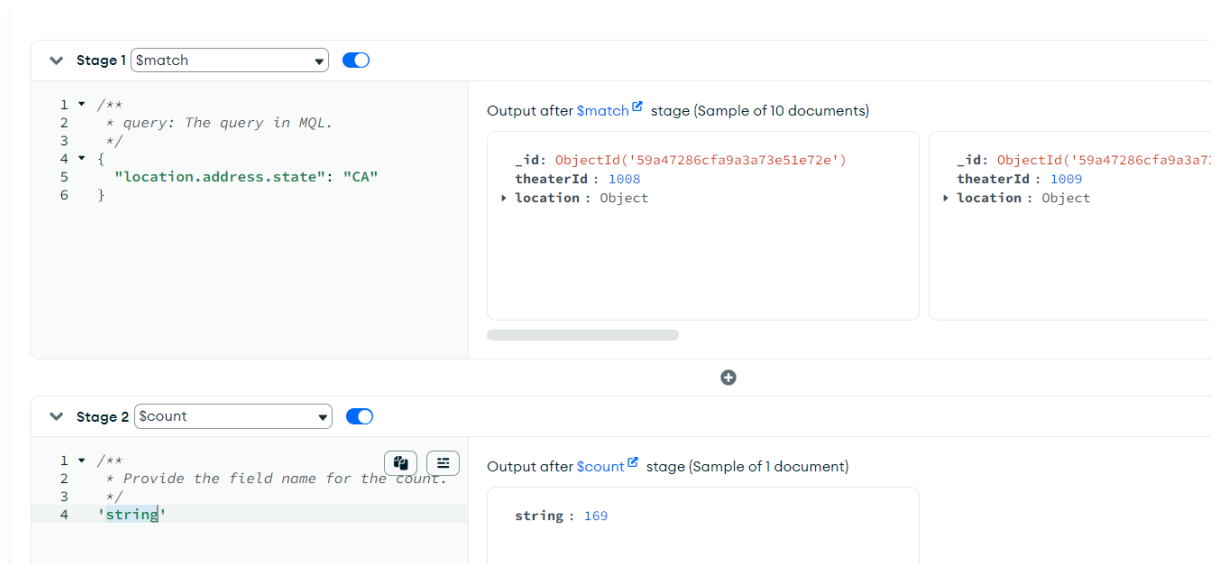
En este ejercicio primero hay que fijarse bien como están definidos los datos en “Theaters”

```

  _id: ObjectId('59a47286cfa9a3a73e51e73b')
  theaterId : 101
  ▾ location : Object
    ▾ address : Object
      street1 : "25422 El Paseo"
      city : "Mission Viejo"
      state : "CA"
      zipcode : "92691"
    ► geo : Object

```

En compas buscamos primero los cines que pertenecen al estado de California “CA” y luego contamos:



The screenshot shows the MongoDB Compass interface with two aggregation stages:

- Stage 1:** \$match. The query is `{ "location.address.state": "CA" }`. The output shows two sample documents with IDs and theater IDs.
- Stage 2:** \$count. The field name for the count is `totalTheaters`. The output shows a count of 169.

En consola:

```
db.theaters.aggregate([
  { $match: { "location.address.state": "CA" } },
  { $count: "totalTheaters" }
])
< {
  totalTheaters: 169
}
```

- ¿Quién fue el primer usuario en registrarse?

Como no tenemos un campo que nos defina la fecha de registro. El campo “_id” contiene un timestamp, con lo cual podemos estimar que los users con id más bajos serán los primeros registros.

En “Compas”:

Ordenamos por _id y limitamos a 1:

{}
 Project { field: 0 }
 Sort { _id :1}
 Collation { locale: 'simple' }
 Index Hint { field: -1 }

Generate query Explain Reset Find Options

Max Time MS
 Skip Limit

ADD DATA EXPORT DATA UPDATE DELETE

1 - 1 of 1

```

{
  "_id": ObjectId('59b99db4cfa9a34dcd7885b6'),
  "name": "Ned Stark",
  "email": "sean_bean@gameofthron.es",
  "password": "$2b$12$UREFwsRUoyF0CRqGNK0Lz00HM/jLhgUCNNIJ9RJAqMUQ74cr1J1Vu"
}
  
```

en consola:

```

> db.users.find().sort({_id: 1}).limit(1)
< {
  _id: ObjectId('59b99db4cfa9a34dcd7885b6'),
  name: 'Ned Stark',
  email: 'sean_bean@gameofthron.es',
  password: '$2b$12$UREFwsRUoyF0CRqGNK0Lz00HM/jLhgUCNNIJ9RJAqMUQ74cr1J1Vu'
}
  
```

- ¿Cuántas películas de comedia hay en la base de datos?

En “Compass”:

Primero agregamos por “genre” y le decimos que busque todas las películas que sean de comedia

Añadimos otro paso tipo \$Count” para que cuente cuantas son de comedia:

Untitled - modified **SAVE** **+ CREATE NEW** **EXPORT TO LANGUAGE** **PREVIEW** **STAGES** **TEXT**

Stage 1 **\$match**

```

1 /**
2  * query: The query in MQL.
3  */
4 {
5   genres: "Comedy"
6 }

```

Output after **\$match** stage (Sample of 10 documents)

```

{
  "_id": ObjectId("573a1390f29313caabcd4803"),
  "plot": "Cartoon figures announce, via comic strip balloons, that they will mov...",
  "genres": Array (3)
    0: "Animation"
    1: "Short"
    2: "Comedy"
  "runtime": 7
  "cast": Array (1)
}

```

Stage 2 **\$count**

```

1 /**
2  * Provide the field name for the count.
3  */
4 'PelículasComedy'

```

Output after **\$count** stage (Sample of 1 document)

```

{
  "PelículasComedy": 7024
}

```

En consola:

Podemos usar el countDocuments:

```

> db.movies.countDocuments({'genres':'Comedy'})
< 7024

```

Respuesta: Tenemos 7024 películas de comedia en la base de datos

- EJERCICIO 2

Muestra todos los documentos de las películas producidas en 1932, pero que el género sea drama o estén en francés.

En "Compass":

Filtramos por año, por género (Drama) y por lengua (French):

The screenshot shows the MongoDB Atlas interface. On the left, a query is written in MQL: `{ 'countries': 'USA', 'awards.wins': {'$gte': 5, '$lte': 9}, 'year': {'$gte': 2012, '$lte': 2014} }`. The right panel shows the output after the query, displaying two sample documents. The first document is for the movie 'The manager of the negative assets sector of Life magazine, Walter Mitty' (2013), and the second is for 'After their cave is destroyed, a caveman family must trek through an u...' (runtime 98, metacritic 55, rated PG).

En consola:

The screenshot shows a MongoDB shell console with the following command and output: `> db.movies.find({'countries': 'USA', 'awards.wins': {'$gte': 5, '$lte': 9}, 'year': {'$gte': 2012, '$lte': 2014}})`. The output shows a single document for the movie 'The manager of the negative assets sector of Life magazine, Walter Mitty' (2013), with a rating of 7.4, 211238 votes, and genres of Adventure, Comedy, and Drama.

Contamos cuantos hay:

The screenshot shows a MongoDB shell console with the following command and output: `> db.movies.countDocuments({'countries': 'USA', 'awards.wins': {'$gte': 5, '$lte': 9}, 'year': {'$gte': 2012, '$lte': 2014}})`. The output shows the count of 166 documents.

Hay 166 películas que cumplen los requisitos.