Base de datos ll

Queries complejas

En MySQL existen operadores y funciones que nos permiten crear consultas más específicas, como SUM, MAX, AVG, LIKE y BETWEEN.
En MongoDB también usamos operadores y funciones, que les permiten realizar consultas complejas sobre sus datos. Las buenas prácticas nos guían para incluir, en una sola entidad, todos o casi todos los datos que necesitamos. Entonces, la mayoría de las veces, pudimos usar una sola consulta por _id, que es la forma más eficiente de recuperar datos, reduciendo la cantidad de solicitudes y garantizando el rendimiento.
Los operadores y las funciones juegan un papel clave en este contexto. Pueden encontrar más información en el sitio web oficial haciendo clic aqui.
Los invitamos a ver algunos ejemplos:

Funciones de agregación

$group

La función $group es equivalente al group by de MySQL.
En este ejemplo, usamos la agregación para mostrar la cantidad de películas por género:

```
db.filmes.aggregate([{$group: {_id: "$genero",Total: {$sum: 1}}}])
{ _id: 'Drama', Total: 3 }
{ _id: 'Suspenso', Total: 1 }
{ _id: 'Infantil', Total: 2 }
{ _id: 'Aventura', Total: 3 }
{ _id: 'Animación', Total: 4 }
{ _id: 'Ciencia Ficción', Total: 5 }
{ _id: 'Comedia', Total: 1 }
```

Pero con MongoDB, podemos hacer más que eso.
¿Qué tal si creamos una consulta que devuelva el género, la cantidad de películas y el nombre de las películas?
Usemos la función de agregador $group y agreguemos un ingrediente más a esta fórmula: $push.
Probemos la siguiente consulta:

```
db.filmes.aggregate([ {$group:{ _id:"$genero", total : {$sum:1}, acervo: {$push: "$titulo"} } }])
{ _id: 'Drama',
  total: 3,
  acervo: [ 'Titanic', 'A vida é bela', 'I am Sam' ] }
{ _id: 'Infantil',
  total: 2,
  acervo: [ 'El Rey Leon', 'Hotel Transylvania' ] }
{ _id: 'Ciencia Ficción',
  total: 5,
```

```
  acervo:
   [ 'Avatar',
     'Star Wars: Episodio VI',
     'Star Wars: Episodio VII',
     'Jurassic Park',
     'Transformers: el lado oscuro de la Luna' ] }
{ _id: 'Animación',
  total: 4,
  acervo: [ 'Buscando a Nemo', 'Toy Story', 'Toy Story 2', 'Divertidamente' ] }
{ _id: 'Aventura',
  total: 3,
  acervo:
   [ 'Harry Potter y la Pedra Filosofal',
     'Harry Potter y la Cámara Secreta',
     'Big' ] }
{ _id: 'Comedia', total: 1, acervo: [ 'Mi pobre angelito' ] }
{ _id: 'Suspenso',
  total: 1,
  acervo: [ 'Harry Potter y las Reliquias de la Muerte - Parte 2' ] }
```

El operador $push agregó una matriz de todos los valores resultantes del campo de título a nuestra consulta. Y el resultado fue sorprendente.

Los operadores de actualización le permiten modificar o agregar datos en su base de datos. Son los siguientes:

$addToSet: agrega elementos distintos a una matriz.

$pop: elimina el primer o el último elemento de una matriz.

$pull: elimina todos los elementos de la matriz que coinciden con una consulta específica.

$push: agrega un elemento a una matriz.

$pullAll: elimina todos los valores coincidentes de una matriz.

$ función de búsqueda

Esta función es equivalente a LEFT JOIN de MySQL. Realiza un vínculo entre dos colecciones de una misma base de datos. Para cada documento de entrada —documentos de la colección desde donde llamamos a la función aggregate—, la función $lookup agrega un nuevo valor por cada documento que cumpla con la condición de unión que definimos indicando el nombre de la colección, los campos que vamos a vincular y el alias donde vamos a devolver los valores.

Ejemplo:

```
db.actorPelícula.aggregate([
  {
    '$lookup': {
      'from': 'películas',  // colección externa
       'localField': 'película',  // nombre del campo actual para comparar con el campo de la colección externa
      'foreignField': 'título',  // nombre de campo de colección externa
      'as': 'género' // nombre del nuevo campo que recibirá los valores buscados.
        }
    }
  ]
```

)
El resultado es:

{ _id: 1, nombre: 'Sam', apellido: 'Worthington', película: 'Avatar',	**Datos de la colección actorPelícula** (colección actual)
género: [{ _id: 1, titulo: 'Avatar', puntaje: 7.9, premios: 3, fecha_lanzamiento: 2010-04-10T03:00:00.000Z, duración: 120, gênero: 'Ciencia Ficción' }] }	**El campo género fue el nombre que se le dio al nuevo campo que recibirá el resultado de la búsqueda.** Tengan en cuenta que el resultado de la búsqueda devuelve todos los datos de la colección externa. **Por lo tanto, el género recibe como matriz todos los datos de la colección de películas.**

Ahora, para capturar solo el campo de género de la matriz, usemos la función **$unwind**.
Función $unwind
Esta función deconstruye un campo de arreglo (array) y lo transforma en un objeto, permitiendo la captura del campo desde la tabla externa.

Ejemplo:

{$unwind: {
path: '$género'
}
	El campo de **género**, en este caso, es el nuevo campo que creamos para recibir el resultado de la consulta a la colección externa, como una matriz.
	$unwind destruye esta matriz y la transforma en un objeto, lo que facilita la captura del campo deseado.

{$group: { _id: '$película',	En esta nueva etapa, agrupamos nuestra consulta por películas y traemos el título de

actores: {
$sum: 1
},
elenco: {
$push: {
$concat: [
'$nombre',
' ',
'$apellido'
]
}
},
	la película, el número de actores y listamos todos los actores, concatenando el nombre y apellido.

género: {
$addToSet: '$genero.genero'
}
	Y finalmente, creamos un nuevo campo de **género,** recuperamos el **género** de la colección de películas y finalizamos nuestra consulta.

Luego, vean el resultado:

{ _id: 'Avatar',
actores: 3,
elenco: ['Sam Worthington', 'Zoe Saldana', 'Sigourney Weaver'],
género: ['Ciencia Ficción'] }
	// título de la película;
	// cantidad de actores de la película;
	// los nombres de los actores
	// y el género que buscamos de la colección películas.

¿No es extraordinario? Básicamente, hicimos una combinación izquierda y recuperamos los datos, usando funciones y operadores.

Exists

El operador $exists verifica si un determinado campo existe o no en la colección.

En el siguiente ejemplo, verificamos si existe el campo de género. Si lo hay, le pedimos que enumere las películas que no pertenecen a los géneros mencionados.

```
db.peliculas.find( { género: { $exists: true, $nin: [ "Ciencia Ficción", "Suspenso", "Drama", "Aventura", "Animación" ] } } )
{ _id: 10,
 título: 'El Rey León',
 puntaje: 9.1,
 premios: 3,
 fecha_lanzamiento: 2000-04-02T03:00:00.000Z,
 género: 'Infantil' }
{ _id: 15,
 título: 'Mi pobre angelito',
 puntaje: 3.2,
 premios: 1,
 fecha_lanzamiento: 1989-04-01T03:00:00.000Z,
 duración: 120,
```

género: 'Comedia' }
{ _id: 19,
título: 'Hotel Transylvania',
puntaje: 7.1,
premios: 1,
fecha_lanzamiento: 2012-04-05T03:00:00.000Z,
duración: 90,
género: 'Infantil' }

Like

Para listar películas cuyos géneros contienen %ra%, en MongoDB reemplazamos % con //

En el siguiente ejemplo, enumeramos todas las películas cuyo género comienza con la secuencia In:

```
db.películas.find({género: /^In/})
{ _id: 10,
título: 'El Rey León',
puntaje: 9.1,
premios: 3,
fecha_lanzamiento: 2000-04-02T03:00:00.000Z,
género: 'Infantil' }
{ _id: 19,
título: 'Hotel Transylvania',
puntaje: 7.1,
premios: 1,
fecha_lanzamiento: 2012-04-05T03:00:00.000Z,
duración: 90,
género: 'Infantil' }
```

Como resultado, sólo se incluyó el género "Infantil",

db.peliculas.find({género: /^In/}) es equivalente a:

```
SELECT * FROM filmes WHERE genero LIKE 'In%'
```

Si queremos listar todos los géneros que tienen la sílaba “ra” en alguna parte de la palabra, usamos: db.peliculas.find({"genre": /ra/}).

Esta consulta es equivalente a '%ra%'.

Ahora vamos a conocer algunos operadores: