

# PRÁCTICA 4

1.- El dump de tu BD.

2.-Un documento que describa en cada apartado:

- la solución,
- el resultado después de la ejecución y una
- breve descripción de lo que haces
- Incluye también lista de dudas que te han surgido y no has resuelto al terminar la práctica

Preparación para trabajar con MongoDB:

- Repasar y ejecutar el archivo *empezando-con-MongoDB.pdf* (no entregues nada)
- Objetivo: Queremos hacer una Red Social de datos de Hobbies o Aficiones. Para ello empezamos con la BD de las aficiones de los alumnos de la clase del año pasado. Luego añades tus aficiones y haces consultas para descubrir quienes tienen gustos parecidos, planear eventos por grupos afines, etc.

APARTADO 1.- Utilizar una BD de Aficiones *AficionesBD* para relacionar personas como en una Red Social

a) Crear tus propios contenidos de la BD. Puedes hacer antes el apartado b) para usarlo como modelo:

1. Escoge una de tus aficiones favoritas.
2. Escoge tus componentes favoritos de esa afición (*cada componente es un documento distinto*): si es música pues tus grupos favoritos, si es cocinar pues tus recetas favoritas, si es esquiar pues tus estaciones favoritas
3. En un editor de texto, escribe la información de cada componente en un formato que se pueda importar en MongoDB de acuerdo a las normas indicadas más abajo. Puedes ver el formato en el fichero *aficionesrestaurantes.json*

*json*

*Normas para Formato de los Componentes de Aficiones:*

- Campos Obligatorios:
    - o *Tema*: sobre el que es tu *Afición*. EJ: música, esquí, cine, cocina, futbol
    - o *Apodo*: no uses tu nombre real porque los datos van a ser públicos en las wikis
    - o *Nombre*: del componente. EJ: Nombre equipo, nombre de grupo, título de película o de libro
    - o *Puntuación*: de ese componente según tu gusto (máximo: 10)
    - o *Precio*: lo que cuesta disfrutar de eso. EJ: precio si es un CD, una película, del concierto de un grupo, etc.
  - Campos específicos tuyos sobre tu componente: Características propias de esa afición que te interesen representar.
    - o Si son películas: género, director, etc.
- b) Cargar los datos de Aficiones del año pasado: Para ello sigue estos pasos
- Una vez arrancada MongoDB, como se indica en *Empezando-con-MongoDB.pdf* :
  - En la shell de Robomongo teclea: use prac1617 //para crear esa BD

- En la ventana (CMD) del entorno Mongo: (en tu portatil recuerda poner el path correcto)

mongorestore

--collection aficiones --db prac1617 A:\...dump-para-empezar-prac\aficiones

- En la shell de Robomongo

db.aficiones.count(); // Debes tener 89 documentos

c) Carga los datos de tu Afición, insertando cada componente (cada documento)

IMPORTANTE: Si tu afición es una de las que ya están creadas: usa el mismo nombre en el campo "Tema".

APARTADO 2.- Trabajando con la colección *Aficiones* en MongoDB

a) De cada *Tema*:

1. Obtener los nombres de los componentes mejor valorados (puntuaciones >= 9)

```
db.aficiones.find({"Puntuacion" :{$gte: 9}}, {"Nombre":1, _id:0});
```

Base de datos "aficiones" find (encontrar) donde puntuación sea >= (greater than or equal) que 9 y sacar el nombre. El id no.

2. Calcula cuanto te gastarías si vas a todos los temas mejor valorados.

```
db.aficiones.aggregate([
  { $match: { "Puntuacion":{$gte:9} } },
  { $group: { _id: "$Tema",
total: { $sum: "$Precio" },
count: { $sum: 1 } } },
{$sort: { _id:1}}]);
```

“aggregate” te deja hacer los “match” y el resto. “match” que coincida Puntuación con  $\geq 9$  y agrupar por Tema. Muestra cuantos temas hay en el “count” y el precio total de cada tema en “total”.

```
MongoDB Enterprise > db.aficiones.aggregate([
... { $match: { "Puntuacion":{$gte:9} } },
... { $group: { _id: "$Tema",
... total: { $sum: "$Precio" },
... count: { $sum: 1 } } },
... {$sort: { _id:1}}]);
{ "_id" : "Baloncesto", "total" : 135000000, "count" : 3 }
{ "_id" : "Fútbol", "total" : 100003300, "count" : 5 }
{ "_id" : "Libros", "total" : 28.4, "count" : 3 }
{ "_id" : "Música", "total" : 50.97, "count" : 4 }
{ "_id" : "Rugby", "total" : 605, "count" : 2 }
{ "_id" : "Series", "total" : 9.95, "count" : 1 }
{ "_id" : "Videojuegos", "total" : 179.8, "count" : 4 }
```

Opción dos:

```
db.aficiones.aggregate([
{$match:{"Puntuacion": {$gt:9}}},
{ $group:{$ _id: null, PrecioTotal : { $sum :
"$Precio"}}});
```

No agrupa por grupos, simplemente muestra el precio total en “PrecioTotal”. A diferencia del anterior, comparamos solo los que sean  $>$  que 9.

```
MongoDB Enterprise > db.aficiones.aggregate([
... {$match:{"Puntuacion": {$gt:9}}},
... { $group:{$ _id: null, PrecioTotal : { $sum : "$Precio"}}});
{ "_id" : null, "PrecioTotal" : 135003302.88 }
```

3. Obtener los componentes valorados para cada uno de los valores (10,9,8,7,6 y 5) por separado

```
db.aficiones.aggregate(  
  {$match : { $or : [ {Puntuacion : {$eq : 10}},  
    {Puntuacion : {$eq : 9}},  
    {Puntuacion : {$eq : 8}},  
    {Puntuacion : {$eq : 7}},  
    {Puntuacion : {$eq : 6}},  
    {Puntuacion : {$eq : 5}}] }},  
  {$group : { _id : "$Puntuacion", componente : {$push :  
    "$Tema" } } } );
```

En este caso cogemos las puntuaciones que sean iguales a 10, 9, 8, 7, 6 y 5 y los agrupamos por esos valores. Luego de cada grupo cogemos una lista de todos los temas.

```
MongoDB Enterprise > db.aficiones.aggregate(  
... {$match : { $or : [ {Puntuacion : {$eq : 10}},  
... {Puntuacion : {$eq : 9}},  
... {Puntuacion : {$eq : 8}},  
... {Puntuacion : {$eq : 7}},  
... {Puntuacion : {$eq : 6}},  
... {Puntuacion : {$eq : 5}}] }},  
... {$group : { _id : "$Puntuacion", componente : {$push : "$Tema" } } });  
{ "_id" : 5, "componente" : [ "Música", "Rugby", "Fútbol" ] }  
{ "_id" : 10, "componente" : [ "Fútbol", "Música", "Rugby", "Series", "Fútbol", "Baloncesto", "Rugby", "Videojuegos", "F  
útbol", "Baloncesto", "Música" ] }  
{ "_id" : 6, "componente" : [ "Libros", "Videojuegos", "Música", "Libros" ] }  
{ "_id" : 7, "componente" : [ "Rugby", "Rugby", "Videojuegos", "Libros", "Rugby", "Videojuegos", "Videojuegos", "Fútbol"  
] }  
{ "_id" : 9, "componente" : [ "Videojuegos", "Libros", "Fútbol", "Música", "Videojuegos", "Música", "Fútbol", "Videojueg  
os", "Libros", "Libros" ] }  
{ "_id" : 8, "componente" : [ "Libros", "Videojuegos", "Libros", "Fútbol", "Fútbol", "Videojuegos", "Fútbol", "Fútbol",  
"Música", "Libros", "Rugby", "Libros" ] }
```

4. Lista de Apodos con esa afición (Tema)

```
db.aficiones.aggregate(  
  {$group : { _id : "$Tema",  
    Apodos : {$addToSet : "$Apodo" } } } );
```

Agrupamos por tema y en la variable “Apodo” añadimos todos los apodos que coincidan con ese tema. addToSet es como el push pero permite repeticiones.

```
MongoDB Enterprise > db.aficiones.aggregate(  
... {$group : { _id : "$Tema",  
... Apodos : {$addToSet : "$Apodo" } } });  
{ "_id" : "Series", "Apodos" : [ "Tyrion Lanister" ] }  
{ "_id" : "MotoGP", "Apodos" : [ "MGP" ] }  
{ "_id" : "Videojuegos", "Apodos" : [ "Giorgio" ] }  
{ "_id" : "Rugby", "Apodos" : [ "Infor" ] }  
{ "_id" : "Baloncesto", "Apodos" : [ "NBA" ] }  
{ "_id" : "Fútbol", "Apodos" : [ "soFIFA", "Mike" ] }  
{ "_id" : "Música", "Apodos" : [ "dios" ] }  
{ "_id" : "Ajedrez", "Apodos" : [ "Bobby" ] }  
{ "_id" : "Libros", "Apodos" : [ "Ceor", "Turre" ] }
```

b) Lista los Apodos, Componente y Tema en los que coincide al menos un componente (el nombre) del mismo Tema.

```
db.aficiones.aggregate([{"$group" : {"_id": "$Nombre",  
"apodos": {"$addToSet": "$Apodo"}, "temas": {"$first":  
"$Tema"}, "count": {"$sum": 1}}}, {"$match": {"count":  
{"$gte": 2}}}] );
```

Agrupamos por nombre, mostramos solo los apodos únicos y el primer tema. Luego si el contador es  $\geq 2$ , lo mostramos por pantalla.

```
MongoDB Enterprise > db.aficiones.aggregate([{"$group" : {"_id": "$Nombre", "apodos": {"$addToSet": "  
$Apodo"}, "temas": {"$first": "$Tema"}, "count": {"$sum": 1}}}, {"$match": {"count": {"$gte": 2}}}] );  
{ "_id" : "Mikhail", "apodos" : [ "Bobby" ], "temas" : "Ajedrez", "count" : 2 }  
{ "_id" : null, "apodos" : [ "MGP" ], "temas" : "MotoGP", "count" : 11 }  
MongoDB Enterprise > _
```

c) Repite la búsqueda anterior para puntuaciones intermedias: más de 4 y menos de 9. Muestra la puntuación también.

```
db.aficiones.aggregate([{"$match": {"$and": [{Puntuacion:  
{"$gt":4}}, {Puntuacion: {"$lt":9}}]}], {"$group" : {"_id":  
"$Nombre", "apodos": {"$addToSet": "$Apodo"}, "temas":  
{"$first": "$Tema"}, "count": {"$sum": 1}}}, {"$match":  
{"count": {"$gte": 2}}}] );
```

No hay más de 2.

d) Describe al menos cuatro consultas interesantes para ti, escribe el código y ejecútalas.

Hecho.

e) Obtener todos los componentes de tu colección clasificados por tema. Para imprimirlos de una manera ordenada, usa un cursor que llama a una función sin nombre (definida dentro del cursor). Esa función imprime cada ítem que, previamente hemos transformado a json (así le damos un formato más legible).

```
db.aficiones.aggregate(  
  { $group: {  
    _id: { tema: "$Tema" },  
    count: { $sum: 1 },  
    docs: { $push: "$Nombre" }  
  }}).forEach( function(myDoc) { print(myDoc);});
```

Lo agrupamos por temas, decimos cuantos objetos hay en cada tema. Y una lista de nombres por cada tema. Por cada documento que me devuelve, lo imprimo por consola.

```
MongoDB Enterprise > db.aficiones.aggregate( { $group: { _id: { tema: "$Tema" }, count: { $sum: 1 }, docs: { $push: "$Nombre" } }}).forEach( function(myDoc) { print(myDoc);});  
[object BSON]  
[object BSON]  
[object BSON]  
[object BSON]  
[object BSON]  
[object BSON]  
[object BSON]  
[object BSON]  
[object BSON]
```

f) Rebaja un 10% al precio de todos los componentes peor valorados (puntuación < 7). Y en la misma actualización añades el atributo *Descuento* a todas las aficiones. El valor se lo asignas tú de acuerdo a esta regla: cuanto mayor puntuación, menor % de descuento.

```
db.aficiones.find().forEach(function (myDoc) {  
  var descuento = myDoc.Precio * 0.10;  
  var porcentaje = (myDoc.Puntuacion/10) * 10;  
  if (myDoc.Puntuacion < 7){  
    myDoc.Precio = myDoc.Precio - descuento;  
  }  
  myDoc.Descuento = porcentaje;  
  db.aficiones.save(myDoc);  
});
```

```
MongoDB Enterprise > db.aficiones.find().forEach( function (myDoc) { var descuento = myDoc.Precio * 0.10; var porcentaje = (myDoc.Puntuacion/10) * 10 ; if (myDoc.Puntuacion < 7 ){ myDoc.Precio = myDoc.Precio-descuento;} myDoc.Descuento = porcentaje; db.Aficiones.save(myDoc); });  
MongoDB Enterprise > _
```

g) Crea una colección *PorNivel* donde vas a crear cuatro *niveles de calidad calculados* de acuerdo a esta fórmula:  $(\text{puntuación} * 100) / \text{precio}$ .

1. Prueba a ajustar tú la ecuación para que sea realista y consiga valores distintos que estén dentro de 4 intervalos.

Escoge esos cuatro intervalos de valores, p.e.: entre 0 y 30, más de 30 y menos de 50, etc.

La colección *PorNivel* tendrá los campos:

*NomCal*: Nombre de intervalo de calidad

*Componentes*: un array/vector que contenga, como elementos, los componentes de la colección *Aficiones* que correspondan a ese intervalo. Además, cada elemento del array, debe tener el *valor de calidad calculado* del componente.

2. Carga en la colección *PorNivel* todos los componentes de la colección *Aficiones* que correspondan.

3. Imprime su contenido formateado que se pueda leer bien, para comprobar que es correcto.

4. Consulta *PorNivel* para obtener los elementos más baratos: su nombre, su precio y su *NomCal*

5. Elimina las 2 aficiones más caras de cada intervalo.

```
db.createCollection("PorNivel",{capped:true, size:
100000,max:5, autoIndexId:true});
```

```
db.PorNivel.insert({NomCal: "NivelUno", Componentes: []});
db.PorNivel.insert({NomCal: "NivelDos", Componentes: []});
db.PorNivel.insert({NomCal: "NivelTres", Componentes: []});
db.PorNivel.insert({NomCal: "NivelCuatro", Componentes:
[]});
```

```

db.aficiones.find().forEach(function(doc) {
    var valor=(doc.Puntuacion * 100) / doc.Precio;
    if (valor <= 20) {
        db.PorNivel.update(
            { NomCal: "NivelUno" },
            { $push: { Componentes: { name: doc.Nombre,
valor: valor } } }
        )
    }
    else if (valor <= 30) {
        db.PorNivel.update(
            { NomCal: "NivelDos" },
            { $push: { Componentes: { name: doc.Nombre,
valor: valor } } }
        )
    }
    else if (valor <= 40) {
        db.PorNivel.update(
            { NomCal: "NivelTres" },
            { $push: { Componentes: { name: doc.Nombre,
valor: valor } } }
        )
    }
    else if (valor <= 50) {
        db.PorNivel.update(
            { NomCal: "NivelCuatro" },
            { $push: { Componentes: { name: doc.Nombre,
valor: valor } } }
        )
    }
});

```

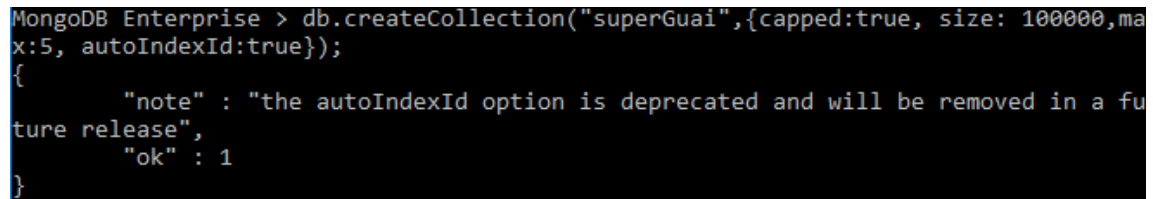


### APARTADO 3. Usando Colecciones limitadas (capped)

Queremos mantener en una colección *superGuai*, los 5 mejores componentes de la colección *Aficiones*. Para ello hacemos lo siguiente:

a) Crear dicha colección.

```
db.createCollection("superGuai",{capped:true, size: 100000,max:5, autoIndexId:true});
```



```
MongoDB Enterprise > db.createCollection("superGuai",{capped:true, size: 100000,max:5, autoIndexId:true});
{
  "note" : "the autoIndexId option is deprecated and will be removed in a future release",
  "ok" : 1
}
```

b) Crea las operaciones necesarias para poner los 5 mejores elementos de acuerdo al criterio de calidad del apartado anterior (apartado 2.g).

c) Inserta un elemento a mano.

```
db.superGuai.insert({NomCal: "NivelUno", Componentes: []});
```

d) Lista todos los componentes para comprobar que mantiene los último cinco introducidos.

```
db.superGuai.find();
```

### APARTADO 4.- EXTRA

Deseamos introducir elementos compuestos, ej.: como en un equipo de futbol si incluimos cada jugador con sus datos personales. Y queremos hacer muchas consultas sobre esos elementos compuestos. ¿Conviene normalizar o desnormalizar? ¿Cómo debería quedar la representación de la colección?

Convendría desnormalizar ya que incluye objetos dentro de objetos. Objetos compuestos.