Gabriel González Jorge Módulo: AED 2ºDAM B

Comandos de MongoDB:

<table>
<tr><td>Mostrar bases de datos en el sistema:</td><td>Show database;
<pre>1 show database;</pre></td></tr>
<tr><td>Ver la base de datos actual (Puede ser ficticia o no)*:
*Ficticia: acuñamos ese termino si esta en memoria RAM y no en el disco de almacenamiento(SSD o HDD)</td><td>db;
<pre>db;</pre></td></tr>
<tr><td>Crear una base “ficticia”:</td><td>use (nombre de la base de datos);
<pre>use prueba;</pre></td></tr>
<tr><td>Mostrarlas colecciones:</td><td>show collections;
<pre>show collections;</pre></td></tr>
<tr><td rowspan="2">Insertar documento:

Existen 4 formas</td><td>1.Deprecated—Desaconsejada:

Se pueden introducir varios documentos:

nombre del documento{
datos del documento
};
db.(nombre de la colección).Insert(nombre del documento);

<pre>user1={
    'name':'user1',
    'age':22,
    'height':180,
    };

db.people.insert(user1);</pre></td></tr>
<tr><td>2. Insert individual (buena práctica-recomendable):

Solo se puede Introducir un documentos

nombre del documento{
datos del documento
};

db.(nombre de la colección).InsertOne(nombre del documento);

<pre>user2={
    'name':'user2',
    'age':23,
    'height':185,
    };

db.people.insertOne(user1);</pre></td></tr>
</table>

3. Insert de muchos (buena práctica-recomendable):

nombre del documentos={
datos del documento}
db.(nombre de la colección).InsertMany(nombre del documentos, nombre del documento2);

```
/* Forma 1 */
user3={
    'name':'user3',
    'age':13,
    'height':155,
    };
    user4={
        'name':'user4',
        'age':15,
        'height':157,
        };

db.people.insertMany(user3,user4);
```

db.(nombre de la colección).InsertMany([nombre del documentos={
datos del documento
},
nombre del documentos2={
datos del documento2
}])

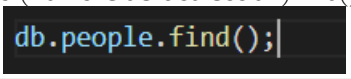
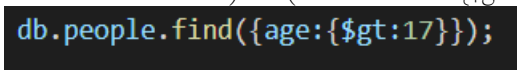
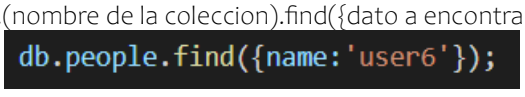
```
/* Forma 2 */
db.users.insertMany([
    user4={
        'name':'user3',
        'age':13,
        'height':155,
    },
    user5={
        'name':'user4',
        'age':15,
        'height':157,
    }
])
```

<table>
<tr><td></td><td>4.Insert con Save(Mala praxis)*:

nombre del documentos={
datos del documento
}

db.(nombre de la colección).save(nombre del documento)

*-Se verifica que el documento exista, de no ser así se crea
*-Se verifica que el documento exista, en caso de que exista lo actualiza(update)

 
</td></tr>
<tr><td>Visualizar la colección:</td><td>db.(nombre de lacolección).find();
 
</td></tr>
<tr><td rowspan="5">Operadores Relacionales:</td><td>gt = mayor que (>)</td></tr>
<tr><td>gte = mayor igual que(>=)</td></tr>
<tr><td>lt = menor que (<)</td></tr>
<tr><td>lte = menor igual que(=<)</td></tr>
<tr><td>ne= no igual que (!=)</td></tr>
<tr><td rowspan="6">Operadores lógicos:</td><td>or= O</td></tr>
<tr><td>in= en, se usa para buscar en una lista</td></tr>
<tr><td>exists=existe, valor boolean(true o false)</td></tr>
<tr><td>/$/= like para final de cadena</td></tr>
<tr><td>/^/= like para Inicio de cadena</td></tr>
<tr><td>/carácter/=like %a% para encontrar una carácter especifico</td></tr>
<tr><td>Encontrar un dato >:</td><td>inc=incremeta</td></tr>
<tr><td>Encontrar un dato:</td><td>db.(nombre de la colección).find(DatoABuscar: {$gt: dato});
 
</td></tr>
<tr><td>Encontrar la cantidad de una librería:</td><td>db.(nombre de la coleccion).find({dato a encontrar});
 
</td></tr>
<tr><td></td><td>db.(nombre de la colección).find(DatoABuscar: {$gt: dato}).count();</td></tr>
</table>

	`db.people.find( {age:{$gte: 10}}).count();`
Encontrar a través de mas de un atributos:	db.(nombre de la colección).find({ $and: [{DatoABuscar: {$gt: dato}}, {DatoABuscar: {$lt: dato}}]}); `db.people.find({ $and: [{age: {$lte: 3}}, {name: true}]})`
Variables : Se pueden guardar cursores y/o consultas en variables, al ser un cursor se consumen una vez invocados	var (nombre de la variable)= lo que quieras poner; `var encontrarUser=db.people.find({name:true}).pretty();`
Proyecciones :	db.(nombre de la colección).find({ [{}//definimos las condiciones, {}//definimos los atributos que nos retorna por defecto el "_id" es true;]}); `db.people.find({},{"_id":0,"username": true});`
Añadir datos en un atributo ya existente:	db.(nombre de la colección).UpdateOne({condicion},$push{});
Filtrar con ElemMatch	db.(nombre de la colección).find({ atributo:{$elemMatch:{}},{}});
Cursor: Un cursor es un objeto que contiene ciertos métodos.	.count() devuelve la cantidad de documentos
	.sort() posibilita al profesional de ordenar los documentos según ponga los datos y ademas retorna un cursor
	.limit() nos da la posibilidad de limitar el numero de documentos a obtener y ademas retorna un cursor
	.skip() salta los documentos indicados y ademas retorna un cursor
	.pretty() nos devuelve los documentos en formato Json
	.forEach() para cada documento
Eliminar documentos : Si no ponemos condiciones MongoDB entiende que queremos eliminar todos los documentos	db.(nombre de la colección).remove(); `db.people.remove();`
Actualizar Documentos en la BD : Existen 6 formas	*1.-Poco eficiente :* var (nombre de la variable)=db.(nombre de la colección).findOne(); nombre de la variable.atributo a cambiar= asignación del nuevo dato;

db.(nombre de la colección).save(nombre de la variable);

```
/*Forma 1*/

var theuser1=db.people.findOne();

theuser1.age=23;

db.people.save(theuser1);
```

2.-Un poco mas eficiente (Deprecated—Desaconsejada):

Por defecto Update es sólo valido por un documento.

db.(nombre de la colección).update({}//definimos las condiciones del documento
,{$set: {}//definimos los cambios de los atributos con sus datos})

```
db.people.update({age:{e$exits:true}},{age: {$set:75}});
```

3.-Si quisiéramos actualizar + eliminar un atributo :

db.(nombre de la colección).update({},
{$unset: {<atributo>:true});

```
db.people.update({"_id": objectId("610c0b8d24c521bbdde1a251")},
                 {$unset:{eyeColor:true}});
```

4.-Actualizar múltiples documentos :

db.(nombre de la colección).update({},
{$set: {<atributo>:true},
{multi:true}
});

```
db.people.update({"_id": objectId("610c0b8d24c521bbdde1a251")},
                 {$set:{eyeColor:true}},{multi:true});
```

5.-Actualizar 1 documento de forma mucho más eficiente :

En caso de ser ambiguos o poco específicos mongoDB actualiza el primer valor que cumpla la condición.

	db.(nombre de la colección).updateOne({},{$set: {<atributo>: valor nuevo}); `db.people.updateOne({"_id":objectId("610c0b8d24c521bbdde1a251")},{$set: {age:23}});`
	6.-Actualizar varios documento de forma mucho más eficiente : db.(nombre de la colección).updateMany({},{$set: {<atributo>: valor nuevo}); `db.people.updateMany({eyeColor:{$exists:true}},{$set:{eyeColor:"brown"}});`
Para eliminar un dato o varios debemos ponerle un condicionante :	*db.(nombre de la colección).updateMany({},{&unset: {<atributo>:valor nuevo}})* `db.people.updateMany({eyeColor:{$exists:true}},{$unset:{eyeColor:"brown"}})`
Eliminar las colecciones :	db.(nombre de la colección).drop(); `db.people.drop();`
Eliminar la base de datos :	db.dropDatabase(); `db.dropDatabase();`
Trabajar con Documentos Embebidos Dot Notation: Para acceder a los atributos dentro de documentos con atributos embebidos se debe acceder mediante	db.(nombre de la colección).find({nombre del atributo.nombre del atributo embebido.atributo:dato}) `db.users.find({'adress.location.lat': 109});`
Actualizar	db.(nombre de la colección).updateMany({//condicion},{$set: { 'Atributo': {'atributo Embebido': dato}}}); `db.users.updateMany({'adress.zip':{$exists: true}},{$set:{'adress.zip':110}});`
Actualizar Un dato muy Especifico en una posición determinada:	db.(nombre de la colección).UpdateOne({<Atributo>:dato},{ $set:{ 'Atributo.posicion del dato': dato a reemplazar}}); `db.users.updateOne({username: 'user7'},{$set:{'adress.location':{lat: -180,long: 250}}});`
Actualizar un dato especifico en una posición Indeterminada: cuando no sabemos la posición de un dato especifico ponemos $ en lugar del numero.	db.(nombre de la colección).UpdateOne({<Atributo>:dato},{ $set:{ 'Atributo.$': dato a reemplazar}}); `db.users.updateOne({username: 'user4'},{$set: {'adress.$':{lat: -180,long: 250}}});`

Añadir un dato especifico:	db.(nombre de la colección).updateOne({condicion},{$push:{atributo:{datos de los atributos}}});
	
	`db.users.updateOne({username:'user3'},{$push:`
	`{Comment:`
	`{like:true, body:'estudio de las caracteristicas de las personas'}`
	`}`
	`});`

Hacer un Backup de la base de datos: Creamos una carpeta desde la terminal, una vez creada ingresamos en ella y una vez dentro hacemos el backup con el siguiente comando.	mongodump –db (nombre de la base de datos) `mongodump -db pruebaDam`
Reestablecer la Base datos desde el backup: Entramos a la carpeta donde hallamos hecho el backup, seguidamente usamos el siguiente comando:	mangorestore –db (nuevo nombre de la base de datos) dump/nombre de la base de datos antigua/ `mongorestore -db(pruebaDam2) dump/pruebaDam/`