Fundamentos de MongoDB Enterprise 4.4.1

Instructor: Carlos Carreño
Email: ccarrenovi@Gmail.com

Modulo 3 CRUD en Mongo DB

- ¿Qué es CRUD?
  - Create
    - Insert()
    - InsertMany()
    - .save()
  - Read
  - Find()
  - findOne()
  - Update
    - Update ()
  - Drop()

Que es CRUD?

- El CRUD describe las funciones elementales de una base de datos persistente. CRUD significa Crear, Recuperar/Leer, Actualizar y Eliminar.
- Operaciones de crear e insertar nuevos documentos a una colección.
- En MongoDB si la colección no existe actualmente, las operaciones de inserción crearán la colección.

Creación de base de datos

- La base de datos se crea con el comando use y al insertar un documento en alguna colección.

```
> use devmongodb;
> db.customer.insert({name:'Dwight',lastname:'Merriman',phone:'+1 345671',age:40,address:'19 Avenued, NY'});
```

Insertar documentos

- Para insertar documentos utiliza el comando db.<nombre de la colección>.insert({..}). Si la colección no existe, mongoDB la crea.

```
> db.customer.insert({name:'Kevin P.',lastname:'Ryan',phone:'+1 345678',age:40,address:'20 Renfer, Malibu'});
WriteResult({ "nInserted" : 1 })
> db.customer.insert({name:'Eliot',lastname:'Horowitz',phone:'+1 445673',age:42,address:'34 Bronze, NY'});
WriteResult({ "nInserted" : 1 })
```

Insertar varios documentos

- Utiliza la función insertMany

```
db.products.insertMany( [
      { _id: 10, item: "large box", qty: 20 },
      { _id: 11, item: "small box", qty: 55 },
      { _id: 12, item: "medium box", qty: 30 }
   ] );
```

Consultando datos de la colección

- Utiliza la función find(…)

```
use devmongodb
switched to db devmongodb
show collections
customer
 db.customer.find();
{ "_id" : ObjectId("5daa4d1f56aead7367e49658"), "name" : "Kevin P.", "lastname" : "Ryan", "phone" : "+1 345678", "age" : 40, "address" : "20 Renfer, Malibu" }
{ "_id" : ObjectId("5daa4d7656aead7367e49659"), "name" : "Eliot", "lastname" : "Horowitz", "phone" : "+1 445673", "age" : 42, "address" : "34 Bronze, NY" }
{ "_id" : ObjectId("5daa4db656aead7367e4965a"), "name" : "Dwight", "lastname" : "Merriman", "phone" : "+1 345671", "age" : 40, "address" : "19 Avenued, NY" }
```

Consultando datos con parámetros

- Utiliza la función find con uno o mas argumentos

```
db.customer.find({age:40});
```

{ "_id" : ObjectId("5daa4d1f56aead7367e49658"), "name" : "Kevin P.", "lastname" : "Ryan", "phone" : "+1 345678", "age" : 40, "address" : "20 Renfer, Malibu" }

{ "_id" : ObjectId("5daa4db656aead7367e4965a"), "name" : "Dwight", "lastname" : "Merriman", "phone" : "+1 345671", "age" : 40, "address" : "19 Avenued, NY" }

```
db.customer.find({age:{$gt:40}});
```

{ "_id" : ObjectId("5daa4d7656aead7367e49659"), "name" : "Eliot", "lastname" : "Horowitz", "phone" : "+1 445673", "age" : 42, "address" : "34 Bronze, NY" }

Operadores de comparación

- Operadores de comparación para diferentes tipos de datos en BSON

Name	Description
`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$nin`	Matches none of the values specified in an array.

Eliminar documentos

- Eliminar un documento dado un _id

```
db.customer.remove({_id:ObjectId("5daa4d7656aead7367e49659")});
WriteResult({ "nRemoved" : 1 })
db.customer.find();
{ "_id" : ObjectId("5daa4d1f56aead7367e49658"), "name" : "Kevin P.", "lastname" : "Ryan", "phone" : "+1 345678", "age" : 40, "address" : "20 Renfer, Malibu" }
{ "_id" : ObjectId("5daa4db656aead7367e4965a"), "name" : "Dwight", "lastname" : "Merriman", "phone" : "+1 345671", "age" : 40, "address" : "19 Avenued, NY" }
```

Eliminar documentos

- Eliminar documentos mediante una comparación

```
db.customer.remove({age:40});
WriteResult({ "nRemoved" : 2 })
db.customer.find();
{ "_id" : ObjectId("5daa538356aead7367e4965b"), "name" : "Eliot", "lastname" : "Horowitz", "phone" : "+1 445673", "age" : 42, "address" : "34 Bronze, NY" }
```

Editar documentos

- Para editar documentos usa la función update

```
db.customer.update({age:42},{phone:'+1 345678'});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
db.customer.find();
{ "_id" : ObjectId("5daa538356aead7367e4965b"), "phone" : "+1 345678" }
{ "_id" : ObjectId("5daa53bb56aead7367e4965c"), "name" : "Kevin P.", "lastname" : "Ryan", "phone" : "+1 345678", "age" : 40, "address" : "20 Renfer, Malibu" }
{ "_id" : ObjectId("5daa53bf56aead7367e4965d"), "name" : "Dwight", "lastname" : "Merriman", "phone" : "+1 345671", "age" : 40, "address" : "19 Avenued, NY" }
```

Editar documentos y agregar campos

- Para editar documentos y agregar campos utiliza update y $set

```
db.customer.update({age:40},{$set:{country:'USA',hobby:'Tennis'}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
db.customer.find();
{ "_id" : ObjectId("5daa53bb56aead7367e4965c"), "name" : "Kevin P.", "lastname" : "Ryan", "phone" : "+1 345678", "age" : 40, "address" : "20 Renfer, Malibu", "country" : "USA", "hobby" : "Tennis" }
{ "_id" : ObjectId("5daa53bf56aead7367e4965d"), "name" : "Dwight", "lastname" : "Merriman", "phone" : "+1 345671", "age" : 40, "address" : "19 Avenued, NY" }
{ "_id" : ObjectId("5daa583256aead7367e49661"), "name" : "Eliot", "lastname" : "Horowitz", "phone" : "+1 445673", "age" : 42, "address" : "34 Bronze, NY" }
```

Nota: Observa que el comando solo actualiza la primera coincidencia.

Actualizando varios documentos

- Utiliza la función updateMany

```
{ "_id" : 1, "name" : "Central Perk Cafe", "places" : 3 }
{ "_id" : 2, "name" : "Rock A Feller Bar and Grill", "places" : 2 }
{ "_id" : 3, "name" : "Empire State Sub", "places" : 5 }
{ "_id" : 4, "name" : "Pizza Rat's Pizzaria", "places" : 8 }
```

```
db.restaurant.updateMany(
    { places: { $gt: 4 } },
    { $set: { "Review" : true } }
 );
```

Actualización masiva de documentos

- Se puede utilizar update para la actualización masiva, para ello la propiedad multi debe tener el valor de true

```
db.products.update({stock:20},
        {
          $set:{stock:40}
         },
         {multi:true}
);
```

Función pretty()

- Por defecto mongoDB, muestra los datos como una lista, para mejorar el formato de salida puedes usar la función pretty()

```
db.customer.find().pretty();
{
        "_id" : ObjectId("5daa53bb56aead7367e4965c"),
        "name" : "Kevin P.",
        "lastname" : "Ryan",
        "phone" : "+1 345678",
        "age" : 40,
        "address" : "20 Renfer, Malibu",
        "country" : "USA",
        "hobby" : "Tennis"
}
```

Documentos embebidos con arrays

- Un documento puede contener varios documentos embebidos, para este propósito se utiliza un array en BSON se representa por los corchetes []. Ejemplo: En los siguientes documentos, el documento **comments** esta embebido en los documentos de **products**

Products documents:

{name:'hard disc SDD 480 GB',stock:20,price:128.50, **comments**:[{uid:1,text:"buen producto"},{uid:2,text:"llego incompleto"}]},
{name:'processor core i7 8th 2.3 GHZ',stock:10,price:1250.0, **comments**:[{uid:1,text:"relacion precio producto bueno"}]},
{name:'Video card GTI 1050 4GB',stock:15,price:800.50},
{name:'Monitor LG HDMI full hd',stock:10,price:1300.0, **comments**:[{uid:1,text:"buena resolucion"}]}

Agregando un documento embebido

- Para agregar un nuevo documento embebido utiliza la operación $push

```
db.products.update(
 { "_id" : ObjectId("5dab225a81e628f3a93ce363")},
 {
  $push:{
    comments:{uid:4,text:"excelente producto"}
   }
 }
);
```

Incrementar datos con $inc

- Se puede incrementar un campo numérico, para ello utiliza la operación $inc

```
db.products.update(
  { "_id" : ObjectId("5dab225a81e628f3a93ce363")},
  {
    $inc:{
       stock:+10
     }
  }
);
```

Ordenar resultados de la consulta

- Utiliza la función sort

```
db.products.find().sort({stock:-1}).pretty();
```

1 ordena de manera ***ascendente***

-1 ordena de manera ***descendente***

Consultas con filtros tipo LIKE

- Para los patrones de las expresiones regulares MongoDB utiliza "***Perl Compatible Regular Expressions***" (PCRE). De esta forma tendremos las siguientes similitudes con los patrones LIKE.

cadena%	/^cadena/	Que empiecen
%cadena%	/cadena/	Que contengan
%cadena	/cadena$/	Que terminen

```
db.products.find({"comments.text":/buen/}).pretty();
```

Consultas tipo Between

- Para realizar tipo de consultas between usamos los operadores de comparación

```
db.products.find({stock:{$gt:15,$lt:30}}).pretty();
```

Recordar:

$gt	>	mayor que
$lt	<	menor que
$gte	>=	mayor igual que
$lte	<=	menor igual que
$eq	==	igual que
$ne	!=	diferente que

Ocultar un campo en el resultado

- Puedes ocultar un campo en el resultado con el valor 0
- Ejemplo: La siguiente operación de búsqueda oculta el campo comments.

```
db.products.find({stock:{$gt:15,$lt:30}},{comments:0}).pretty();
```

Contando los resultados

- Para contar los documentos del resultado utiliza la función count()

```
db.products.find({stock:{$eq:20}}).count();
```

Creando un documento embebido

- Utiliza la función update y la operación $set

- Ejemplo: Se crea el documento ***provider*** en el documento ***products***

```
db.products.update({"_id" : ObjectId("5dab225a81e628f3a93ce362")},
    {
     $set:{provider:{country:"China",phone:"+86 3456789"}}
    }
);
```

Agregando una propiedad al documento embebido

- Utiliza la función update y el operador $set

```
db.products.update({"provider.country":'China'},
        {
            $set:{"provider.email":'leeyuang@aliexpress.com'}
        }
);
```

Eliminando una propiedad al documento embebido

- Utiliza la función update y el operador $unset

```
db.products.update({"provider.country":'China'},
        {
         $unset:{"provider.email":''}
        }
);
```

Eliminando una propiedad de un documento embebido en un Array

- Ejemplo 1: Agregando la propiedad

```
db.products.update({"stock":10},
                   {$push:{ comments:{uid:3,text:"llego bien",tag:"normal"} }
});
```

- Ejemplo 2: Eliminando la propiedad

```
db.products.update({"stock":10},
                   {$pull:{ comments:{tag:"normal"} } },
{multi:true});
```

Eliminando una propiedad de un documento embebido en un array con el operador $in

- Ejemplo 3: Elimina un documento embebido del array que contenga determinada palabra o lista de palabras

```
db.products.update({"stock":40},
                   {$pull:{ comments:{text:{$in:[/excelente/]}} } },
{multi:true});

db.products.update({"stock":40},
                   {$pull:{ comments:{text:{$in:[/excelente/,/resolucion/]}} } },
{multi:true});
```

Eliminando la colección

- Utiliza la función drop()

```
db.customer.drop()
```

Borrar la base de datos

- Para eliminar una base de datos usa la función dropDatabase()

```
use devmongodb;
switched to db devmongodb
db.dropDatabase();
{ "dropped" : "devmongodb", "ok" : 1 }
```

Practica

- Practica 2 PARTE 2

Referencias

- https://es.wikipedia.org/wiki/MongoDB
- http://www.cantabriatic.com/introduccion-a-mongodb/
- https://api.mongodb.com/wiki/current/Starting%20and%20Stopping%20Mongo.html
- http://lineadecodigo.com/mongodb/consultas-mongodb-like/