01 CRUD Operaciones de inserción

Método insert()

https://docs.mongodb.com/manual/reference/method/db.collection.insert/#db.collection.insert

Sintaxis

```
db.collection.insert(
   <document or array of documents>,
   {
     writeConcern: <document>,
     ordered: <boolean>
   }
)
```

Parameter	Type	Description
document	document or array	A document or array of documents to insert into the collection.
writeConcern	document	Optional. A document expressing the write concern. Omit to use the default write concern.
ordered	boolean	Optional. If true (default), perform an ordered insert of the documents in the array, and if an error occurs with one of documents, MongoDB will return without processing the remaining documents in the array. If false, perform an unordered insert, and if an error occurs with one of documents, continue

		processing the remaining documents in the array.

Comportamiento

Write Concern
The insert() method uses the insert command, which uses the default write concern. To specify a different write concern, include the write concern in the options parameter.

Create Collection
If the collection does not exist, then the insert() method will create the collection.

_id Field
If the document does not specify an _id field, then MongoDB will add the _id field and assign a unique ObjectId for the document before inserting. Most drivers create an ObjectId and insert the _id field, but the mongod will create and populate the _id if the driver or application does not.

If the document contains an _id field, the _id value must be unique within the collection to avoid duplicate key error.

Prácticas

Insertar documento en colección sin _id

```
>use app
>db.usuarios.insert({nombre:"Juan",apellidos:"López",edad:30})
>WriteResult({ "nInserted" : 1 })

>db.usuarios.find()
>{ "_id" : ObjectId("5c599d3664f23551fe1d2390"), "nombre" : "Juan", "apellidos" :
"López", "edad" : 30 }
```

Insertar documento en colección con _id

```
>db.usuarios.insert({_id: 02, nombre:"Pedro",apellidos:"Pérez",edad:22})
>WriteResult({ "nInserted" : 1 })

>db.usuarios.find()
```

>{ "_id" : ObjectId("5c599d3664f23551fe1d2390"), "nombre" : "Juan", "apellidos" : "López", "edad" : 30 } { "_id" : 2, "nombre" : "Pedro", "apellidos" : "Pérez", "edad" : 22 }

Insertar múltiples documentos

```
>db.usuarios.insert([
        {_id: 04, nombre:"Lucía",apellidos:"Pérez",edad:44},
        {_id: 03, nombre:"María",apellidos:"Rodríguez"}
])
>BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

> db.usuarios.find()
>{ "_id" : ObjectId("5c599d3664f23551fe1d2390"), "nombre" : "Juan", "apellidos" : "López", "edad" : 30 } { "_id" : 2, "nombre" : "Pedro", "apellidos" : "Pérez", "edad" : 22 } { "_id" : 4, "nombre" : "Lucía", "apellidos" : "Pérez", "edad" : 44 } { "_id" : 3, "nombre" : "María", "apellidos" : "Rodríguez" }

Insertar múltiples documentos con la opción ordered true (default)

```
>db.usuarios.insert([
        {_id: 10},
        {_id: 11},
        {_id: 11},
        {_id: 12}
        {_id: 13}
])
>BulkWriteResult({
        "writeErrors" : [
                { "index" : 2, "code" : 11000, "errmsg" : "E11000 duplicate key error
                collection: app.usuarios index: _id_ dup key: { : 11.0 }", "op" : {"_id" : 11 } }
                ], "writeConcernErrors" : [ ], "nInserted" : 2, "nUpserted" : 0,
                "nMatched" : 0, "nModified" : 0, "nRemoved" : 0, "upserted" : [ ]
})

> db.usuarios.find()
>{ "_id" : ObjectId("5c599d3664f23551fe1d2390"), "nombre" : "Juan", "apellidos" :
"López", "edad" : 30 } { "_id" : 2, "nombre" : "Pedro", "apellidos" : "Pérez", "edad" : 22 } {
"_id" : 4, "nombre" : "Lucía", "apellidos" : "Pérez", "edad" : 44 } { "_id" : 3, "nombre" :
"María", "apellidos" : "Rodríguez" } { "_id" : 10 } { "_id" : 11 }  // El documento con _id 12
no se inserta
```

Insertar múltiples documentos con la opción ordered false

```
>db.usuarios.insert([
{_id: 20},
{_id: 21},
{_id: 21},
{_id: 22}
],{ordered: false})

>BulkWriteResult({ "writeErrors" : [ { "index" : 2, "code" : 11000, "errmsg" : "E11000
duplicate key error collection: app.usuarios index: _id_ dup key: { : 21.0 }", "op" :
{"_id" : 21 } } ], "writeConcernErrors" : [ ], "nInserted" : 3, "nUpserted" : 0, "nMatched" :
0, "nModified" : 0, "nRemoved" : 0, "upserted" : [ ]})

>db.usuarios.find()
>{ "_id" : ObjectId("5c599d3664f23551fe1d2390"), "nombre" : "Juan", "apellidos" :
"López", "edad" : 30 } { "_id" : 2, "nombre" : "Pedro", "apellidos" : "Pérez", "edad" : 22 } {
"_id" : 4, "nombre" : "Lucía", "apellidos" : "Pérez", "edad" : 44 } { "_id" : 3, "nombre" :
"María", "apellidos" : "Rodríguez" } { "_id" : 10 } { "_id" : 11 } { "_id" : 20 } { "_id" : 21 } { "_id"
: 22 } // se inserta aunque hubiese un error en el documento anterior
```

Tipos de dato para _id

Si no se especifica, se crea de tipo ObjectId (el manual no detalla si lo crea MongoDB o los drivers).

Cualquier tipo (incluso documentos), excepto arrays.

```
> db.usuarios.insert({_id: 2})
> db.usuarios.insert({_id: {zona: "sur", sector: 1}, nombre: "María"})
> db.usuarios..insert({_id: ["sur",2]})
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 2,
                "errmsg" : "can't use an array for _id"
        }
})
```

Tipo de dato Date

```
> db.pacientes.insert({nombre: "Juan", apellidos: "Pérez", createdAt: new Date()})
```

Orden de colocación de los campos

Método insertOne()

https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/

Sintaxis

```
db.collection.insertOne(
  <document>,
  {
    writeConcern: <document>
  }
)
```

Parameter	Type	Description
document	document	A document to insert into the collection.
writeConcern	document	Optional. A document expressing the write concern. Omit to use the default write concern.

Comportamiento

Explainability
insertOne() is not compatible with db.collection.explain(), use insert() instead.

Prácticas

```
> db.pacientes.insertOne({nombre: "Juan", apellidos: "Pérez", createdAt: new
Date()})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5e25de598621cff1393f70b6")
}
```

Método insertMany()

https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany

Sintaxis

```
db.collection.insertMany(
  [ <document 1> , <document 2>, ... ],
  {
    writeConcern: <document>,
    ordered: <boolean>
  }
)
```

Parameter	Type	Description
idem insert()		

Comportamiento

Order
By default documents are inserted in order. If ordered is set to false, documents are inserted in an unordered format and may be reordered by mongod to increase performance. Applications should not depend on ordering of inserts if using an unordered insertMany().

Executing an ordered list of operations on a sharded collection will generally be slower than executing an unordered list since with an ordered list, each operation must wait for the previous operation to finish.

Explainability
insertMany() is not compatible with db.collection.explain().

Prácticas

> db.pacientes.insertMany([{nombre: 'Laura', apellidos: 'Pérez', createdAt: new Date()}, {nombre: 'Juan', apellidos: 'López, createdAt: new Date()},{nombre: 'María', apellidos: 'Gómez', createdAt: new Date()}])
> 2020-01-20T18:18:45.242+0100 E QUERY   [thread1] SyntaxError: missing } after property list @(shell):1:157 // En este caso es un error sintáctico que anula toda la operación

Método save()

https://docs.mongodb.com/manual/reference/method/db.collection.save/#db.collection.save

Sintaxis

```
db.collection.save(
  <document>,
  {
    writeConcern: <document>
  }
)
```

Parameter	Type	Description
document	document	A document to save (create or update if exists) into the collection.
writeConcern	document	Optional. A document expressing the write concern. Omit to use the default write concern.

Comportamiento

Insert
If the document does not contain an _id field, then the save() method calls the insert() method. During the operation, the mongo shell will create an ObjectId and assign it to the _id field.

Update
If the document contains an _id field, then the save() method is equivalent to an update with the upsert option set to true and the query predicate on the _id field.

Prácticas

```
> db.pacientes.save({_id: 10, nombre: "Luis", apellidos: "López"})
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 10 })
> db.pacientes.save({_id: 10, nombre: "Luis", apellidos: "Pérez"})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Métodos adicionales de inserción

The following methods can also add new documents to a collection:

db.collection.update() when used with the upsert: true option.
db.collection.updateOne() when used with the upsert: true option.
db.collection.updateMany() when used with the upsert: true option.
db.collection.findAndModify() when used with the upsert: true option.
db.collection.findOneAndUpdate() when used with the upsert: true option.
db.collection.findOneAndReplace() when used with the upsert: true option.
db.collection.bulkWrite().