

OPERACIONES CRUD

Vamos a escribir operaciones CRUD (Create, Read, Update y Delete) para MongoDB que realicen lo mismo que su equivalente en SQL estándar.

1. CREAR Y MODIFICAR COLECCIONES.

1.1)

```
CREATE TABLE people (  
  id MEDIUMINT NOT NULL AUTO_INCREMENT,  
  user_id Varchar(30),  
  age Number,  
  status char(1),  
  PRIMARY KEY (id)  
);
```

Solución:

```
use CursoMongoDB // Selecciona la base de datos si ya existe o la crea si aún  
no existe
```

```
> admin  
authenticated  
switched to db CursoMongoDB
```

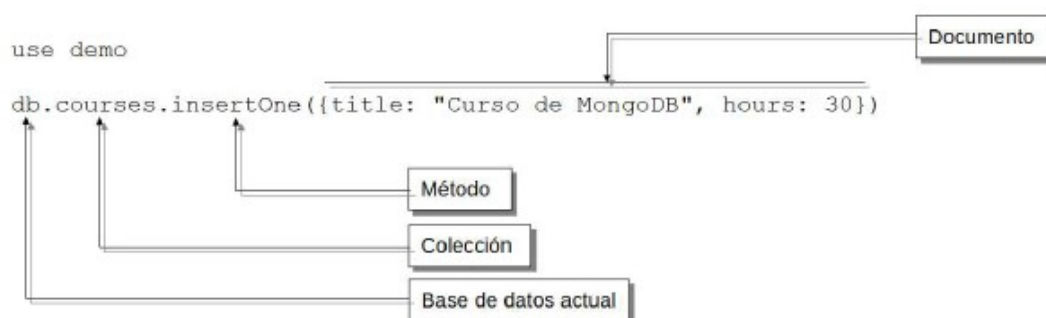
Para crear la colección podemos hacerlo de las dos siguientes maneras:

```
db.createCollection("people") // Crea la colección
```

```
{  
  "ok" : 1.0  
}
```

O también insertando un documento en la colección:

En MongoDB se pueden crear bases de datos y colecciones en modo perezoso ya que se crean de forma implícita una vez que se almacenan datos en ellas.



```
insertOne({document})
```

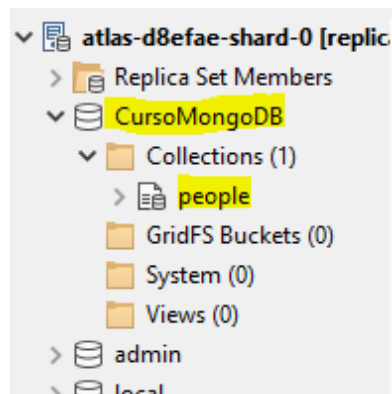
Este método inserta un documento único en la colección.

El *campo _id* es obligatorio en todos los documentos, y debe ser único. Si no lo especificamos, MongoDB lo creará automáticamente.

El *_id* creado por MongoDB es del tipo ObjectId y tiene 12 bytes de longitud: 4 para la marca de tiempo de creación del documento, en segundos, 5 bytes de un valor random generado una vez por proceso y 3 bytes para el valor de un contador, inicializado a un valor aleatorio.

```
db.people.insertOne({user_id:"bcd000", age:50, status:"J"})
```

```
{
  "_id" : ObjectId("662fc487efef71a5eb81060e"),
  "user_id" : "bcd000",
  "age" : NumberInt(50),
  "status" : "J"
}
```



En estos casos el *_id* se genera automáticamente. No existe el *auto_increment* en MongoDB.

Si quisiéramos **hacerlo autoincremental** la solución podría ser crear previamente una colección que llevara los contadores de los *_id* y mediante una función buscar ese valor y actualizarlo para futuras actualizaciones. Sería así:

Primero crear la colección de contadores iniciándolo en 0:

```
db.contador.insertOne({coleccion:"people", contador_id:0})
```

A continuación crear una función que permita leer ese valor y actualizarlo:

```
function siguienteid (namecoleccion){
  var salida=db.contador.findAndModify(({
    query:{coleccion:namecoleccion}, //criterio de búsqueda
    update:{$inc:{contador_id:1}} //incrementa el contador de la
coleccion en 1
  }))
  return salida.contador_id;
}
```

Ahora ya podríamos insertar con el `_id` autoincremental:

```
{
  "_id" : ObjectId("662fc487efef71a5eb81060e"),
  "user_id" : "bcd000",
  "age" : NumberInt(50),
  "status" : "J"
}
{
  "_id" : NumberInt(1),
  "user_id" : NumberInt(1),
  "age" : NumberInt(22),
  "status" : "J"
}
```

```
db.people.insertOne({_id:siguienteid("people"), user_id:1, age:22, status:"J"})
```

1.2)

```
ALTER TABLE people ADD join_date DATETIME;
```

Solución:

No existe algo similar a ALTER ya que no hay una estructura estática para todos los documentos.

Podemos hacerlo de dos maneras:

1.- Utilizando \$set.

En este caso añadimos un nuevo campo `join_date` tipo DATETIME. Lo que hace es modificar todos los documentos con \$set join_date. No existe el campo y lo crea, y le pone la fecha del sistema.

```
db.people.updateMany({}, {$set: {join_date: new Date()}})
```

Existen dos métodos para actualizar documentos:

```
updateOne({query},{update})
```

Este método actualiza el primer documento que cumpla con los criterios de la consulta.

```
updateMany({query},{update})
```

Este método actualiza todos los documentos que cumplan con los criterios de la consulta.

2.- Insertando nuevos documentos con el campo a insertar en la definición de la colección.

```
db.people.insertOne({user_id:"bcd0010", age:50, status:"A", join_date: new Date()})
```

```
{
  "_id" : ObjectId("662fca98efef71a5eb81061c"),
  "user_id" : "bcd000",
  "age" : NumberInt(50),
  "status" : "J"
}
{
  "_id" : NumberInt(2),
  "user_id" : NumberInt(1),
  "age" : NumberInt(22),
  "status" : "J"
}
{
  "_id" : ObjectId("662fcaaefef71a5eb81061d"),
  "user_id" : "bcd0010",
  "age" : NumberInt(50),
  "status" : "A",
  "join_date" : ISODate("2024-04-29T16:28:30.693+0000")
}
```

1.3)

ALTER TABLE people DROP COLUMN join_date;

Solución:

```
db.people.updateMany(
  {}, // Sin filtro para aplicar a todos los documentos
  {$unset: {join_date: ""}} // Elimina el campo join_date
);
```

1.4)

DROP TABLE people;

Solución:

```
db.people.drop()
```

2. INSERTAR.

Los ejercicios se resuelven sin el campo _id autoincremental.

2.1)

```
INSERT INTO people(user_id, age, status)
VALUES ("bcd001", 45, "A");
```

Solución:

```
db.people.insertOne({user_id:"bcd001", age:45, status:"A"})
```

Insertamos más documentos para luego jugar con ellos:

```
insertMany([{doc1}, {doc2}... {docN}])
```

Esté método recibe un array de uno o más documentos y los inserta en la colección.

```
db.people.insertMany([
  {user_id:"bcd001", age:45, status:"A"},
  {user_id:"fgbcd001", age:45, status:"A"},
  {user_id:"00bcd001", age:15, status:"D"},
  {user_id:"bcd003", age:50, status:"A"},
  {user_id:"bcd004", age:50, status:"S"},
  {user_id:"bcd005", age:44, status:"S"},
  {user_id:"bcd006", age:18, status:"A"},
  {user_id:"bcd007", age:70, status:"S"},
  {user_id:"bcd008", age:24, status:"J"},
  {user_id:"bcd009", age:58, status:"D"}])
```

3. CONSULTAR.

```
findOne({query}, {projection})
```

Este método recupera el primer documento de la colección que cumpla con los criterios de la consulta.

```
find({query}, {projection})
```

Este método recupera todos los documentos de la colección que cumplan los criterios de la consulta.

3.1)

```
SELECT *
FROM people;
```

Solución:

```
db.people.find({})
```

3.2)

```
SELECT id, user_id, status
FROM people;
```

Solución:

El primer parámetro sería lo que busca y el segundo el proyect donde indicamos que campos visualizamos.

Los campos de un documento se pueden proyectar incluyendo:

```
{<field1>: 1, <field2>: 1... <fieldN>: 1}
```



O excluyendo:

```
{<field1>: 0, <field2>: 0... <fieldN>: 0}
```

El `_id` siempre se visualiza a menos que se indique de forma implícita, si así se desea (`_id:0`).

```
db.people.find({}, {user_id:1, status:1})
```

3.3)

```
SELECT user_id, status  
FROM people;
```

Solución:

```
db.people.find({}, {_id:0, user_id:1, status:1})
```

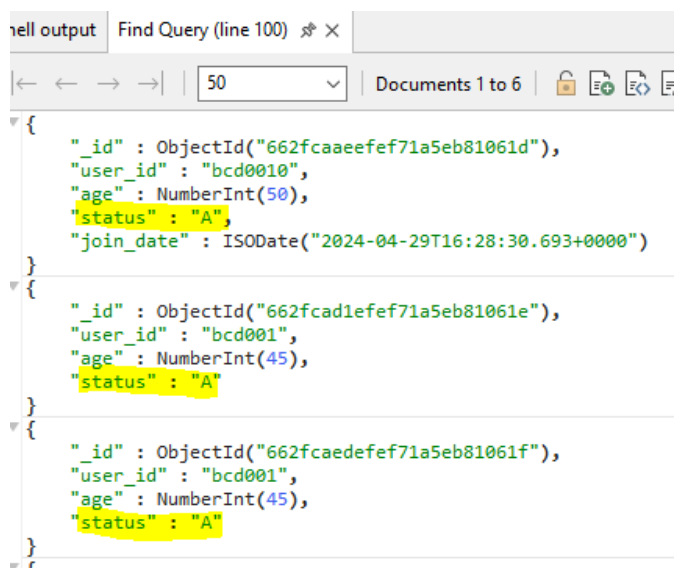
```
{  
  "_id" : ObjectId("662fca98efef71a5eb81061c"),  
  "user_id" : "bcd000",  
  "status" : "J"  
}  
  
{  
  "_id" : NumberInt(2),  
  "user_id" : NumberInt(1),  
  "status" : "J"  
}
```

3.4)

```
SELECT *  
FROM people  
WHERE status = "A";
```

Solución:

```
db.people.find({status: "A"})  
db.people.find({status:{$eq:"A"}})
```



The screenshot shows a MongoDB query result in a web interface. The title bar indicates 'Find Query (line 100)'. The interface includes navigation arrows, a line number '50', and a document count 'Documents 1 to 6'. The query results are displayed as a JSON array of three documents, each with the following fields: `_id` (ObjectId), `user_id` (string), `age` (NumberInt), `status` (string, highlighted as 'A'), and `join_date` (ISODate). The first document has `user_id` 'bcd0010' and `age` 50. The second document has `user_id` 'bcd001' and `age` 45. The third document has `user_id` 'bcd001' and `age` 45.

```
{  
  "_id" : ObjectId("662fcaaeefef71a5eb81061d"),  
  "user_id" : "bcd0010",  
  "age" : NumberInt(50),  
  "status" : "A",  
  "join_date" : ISODate("2024-04-29T16:28:30.693+0000")  
}  
  
{  
  "_id" : ObjectId("662fcad1efef71a5eb81061e"),  
  "user_id" : "bcd001",  
  "age" : NumberInt(45),  
  "status" : "A"  
}  
  
{  
  "_id" : ObjectId("662fcaedefef71a5eb81061f"),  
  "user_id" : "bcd001",  
  "age" : NumberInt(45),  
  "status" : "A"  
}
```

3.5)

```
SELECT user_id, status
FROM people
WHERE status = "A";
```

Solución:

```
db.people.find({status:"A"},{_id:0, user_id:1, status:1})
db.people.find({status:{$eq: "A"}},{_id:0, user_id:1, status:1})
```

```
{
  "user_id" : "bcd0010",
  "status" : "A"
}
{
  "user_id" : "bcd001",
  "status" : "A"
}
{
  "user_id" : "bcd001".
```

3.6)

```
SELECT *
FROM people
WHERE status != "A";
```

Solución:

```
db.people.find({status: {$ne: "A"}})
```

```
{
  "_id" : ObjectId("662fca98efef71a5eb81061c"),
  "user_id" : "bcd000",
  "age" : NumberInt(50),
  "status" : "J"
}
{
  "_id" : NumberInt(2),
  "user_id" : NumberInt(1),
  "age" : NumberInt(22),
  "status" : "J"
}
{
  "_id" : ObjectId("662fcaedefef71a5eb810621"),
  "user_id" : "00bcd001",
  "age" : NumberInt(15),
  "status" : "D"
}
```

3.7)

```
SELECT *
FROM people
WHERE status = "A" AND age = 50;
```

Solución:

```
db.people.find({status: "A", age: 45})
db.people.find({$and:[{status: "A", age: 45}]})
```

```
{
  "_id" : ObjectId("662fcad1efef71a5eb81061e"),
  "user_id" : "bcd001",
  "age" : NumberInt(45),
  "status" : "A"
}
{
  "_id" : ObjectId("662fcaedefef71a5eb81061f"),
  "user_id" : "bcd001",
  "age" : NumberInt(45),
  "status" : "A"
}
{
  "_id" : ObjectId("662fcaedefef71a5eb810620"),
  "user_id" : "fgbcd001",
  "age" : NumberInt(45),
  "status" : "A"
}
```

3.8)

```
SELECT *  
FROM people  
WHERE status = "A" OR age = 50;
```

Solución:

```
db.people.find({$or:[{status: "A"}, {age: 50}]})
```

```
{  
  "_id" : ObjectId("662fca98efef71a5eb81061c"),  
  "user_id" : "bcd000",  
  "age" : NumberInt(50),  
  "status" : "J"  
}  
{  
  "_id" : ObjectId("662fcaaefef71a5eb81061d"),  
  "user_id" : "bcd0010",  
  "age" : NumberInt(50),  
  "status" : "A",  
  "join_date" : ISODate("2024-04-29T16:28:30.693+0000")  
}  
{  
  "_id" : ObjectId("662fcad1efef71a5eb81061e"),  
  "user_id" : "bcd001",  
  "age" : NumberInt(45),  
  "status" : "A"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb81061f"),  
  "user_id" : "bcd001".  
}
```

3.9)

```
SELECT *  
FROM people  
WHERE age > 25;
```

Solución:

```
db.people.find({age: {$gt: 25}})
```

```
{  
  "_id" : ObjectId("662fca98efef71a5eb81061c"),  
  "user_id" : "bcd000",  
  "age" : NumberInt(50),  
  "status" : "J"  
}  
{  
  "_id" : ObjectId("662fcaaefef71a5eb81061d"),  
  "user_id" : "bcd0010",  
  "age" : NumberInt(50),  
  "status" : "A",  
  "join_date" : ISODate("2024-04-29T16:28:30.693+0000")  
}  
{  
  "_id" : ObjectId("662fcad1efef71a5eb81061e"),  
  "user_id" : "bcd001",  
  "age" : NumberInt(45),  
  "status" : "A"  
}
```

3.10)

```
SELECT *  
FROM people  
WHERE age < 25;
```

Solución:

```
db.people.find({age: {$lt: 25}})
```

```
{  
  "_id" : NumberInt(2),  
  "user_id" : NumberInt(1),  
  "age" : NumberInt(22),  
  "status" : "J"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810621"),  
  "user_id" : "00bcd001",  
  "age" : NumberInt(15),  
  "status" : "D"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810625").  
}
```


3.11)

```
SELECT *  
FROM people  
WHERE age > 25 AND age <= 50;
```

Solución:

```
db.people.find({age: {$gt: 25, $lte: 50}})  
db.people.find({$and:[{age:{$gt:25}},{age:{$lte:50}}]})
```

```
{  
  "user_id" : "fgbcd001",  
  "age" : NumberInt(45),  
  "status" : "A"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810622"),  
  "user_id" : "bcd003",  
  "age" : NumberInt(50),  
  "status" : "A"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810623"),  
  "user_id" : "bcd004",  
  "age" : NumberInt(50),  
  "status" : "S"  
}  
{"_id" : ObjectId("662fcaedefef71a5eb810624")}
```

3.12)

```
SELECT *  
FROM people  
WHERE user_id like "%bc%";
```

Solución:

```
db.people.find({user_id: {$regex: /.bc./}}) // bc solo está en el interior
```

```
{  
  "_id" : ObjectId("662fcaedefef71a5eb810620"),  
  "user_id" : "fgbcd001",  
  "age" : NumberInt(45),  
  "status" : "A"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810621"),  
  "user_id" : "00bcd001",  
  "age" : NumberInt(15),  
  "status" : "D"  
}
```

```
db.people.find({user_id: {$regex: "bc"}}) // tiene bc en algún sitio
```

3.13)

```
SELECT *  
FROM people  
WHERE user_id like "bc%";
```

Solución:

```
db.people.find({user_id: {$regex: "^bc*"}})  
db.people.find({user_id: {$regex: /^bc*/}})
```

```
db.people.find({user_id: {$regex: /^bc./}})
```

```
{
  "_id" : ObjectId("662fca98efef71a5eb81061c"),
  "user_id" : "bcd000",
  "age" : NumberInt(50),
  "status" : "J"
}
{
  "_id" : ObjectId("662fcaaeefef71a5eb81061d"),
  "user_id" : "bcd0010",
  "age" : NumberInt(50),
  "status" : "A",
  "join_date" : ISODate("2024-04-29T16:28:30.693+0000")
}
{
  "_id" : ObjectId("662fcad1efef71a5eb81061e"),
  "user_id" : "bcd001",
  "age" : NumberInt(45)
```

3.14)

```
SELECT *
FROM people
WHERE status = "A"
ORDER BY user_id ASC;
```

Solución:

```
db.people.find({status: "A"}).sort({user_id: 1})
```

3.15)

```
SELECT *
FROM people
WHERE status = "A"
ORDER BY user_id DESC;
```

Solución:

```
db.people.find({status: "A"}).sort({user_id: -1})
```

3.16)

```
SELECT COUNT(*) FROM people;
```

Solución:

```
db.people.countDocuments()
db.people.find({}).count()
```

3.17)

```
SELECT COUNT(user_id) FROM people;
```

Solución:

```
// vamos a insertar un documento sin user_id
db.people.insertOne({age:90,status:"L"})
```

```
{
  "age" : NumberInt(58),
  "status" : "D"
}
{
  "_id" : ObjectId("662fd0ffefef71a5eb810629"),
  "age" : NumberInt(90),
  "status" : "L"
}
```

```
db.people.countDocuments({ user_id: { $exists: true} })
db.people.find({user_id:{$exists:true}}).count()
```

14

3.18)

```
SELECT COUNT(*)
FROM people
WHERE age > 30;
```

Solución:

```
db.people.countDocuments({age: {$gt: 30}})
db.people.find({age:{$gt:30}}).count()
```

11

3.19)

```
SELECT DISTINCT(status)
FROM people;
```

Solución:

```
db.people.distinct("status")
```

```
[
  "A",
  "D",
  "J",
  "L",
  "S"
]
```

3.20)

```
SELECT *
FROM people
LIMIT 1;
```

Solución:

```
db.people.findOne()
db.people.find({}).limit(1)
```

```
{
  "_id" : ObjectId("662fca98efef71a5eb81061c"),
  "user_id" : "bcd000",
  "age" : NumberInt(50),
  "status" : "J"
}
```

3.21)

```
SELECT *  
FROM people  
LIMIT 5  
SKIP 10;
```

Solución:

```
db.people.find().skip(9).limit(5)
```

```
{  
  "_id" : ObjectId("662fcaedefef71a5eb810624"),  
  "user_id" : "bcd005",  
  "age" : NumberInt(44),  
  "status" : "S"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810625"),  
  "user_id" : "bcd006",  
  "age" : NumberInt(18),  
  "status" : "A"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810626"),  
  "user_id" : "bcd007",  
  "age" : NumberInt(70),  
  "status" : "S"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810627"),  
  "user_id" : "bcd008",  
  "age" : NumberInt(24),  
  "status" : "J"  
}  
{  
  "_id" : ObjectId("662fcaedefef71a5eb810628"),  
  "user_id" : "bcd009",  
  "age" : NumberInt(58),  
  "status" : "D"  
}
```

4. ACTUALIZACIÓN.

4.1)

```
UPDATE people  
SET status = "C"  
WHERE age > 25;
```

Solución:

Existen dos métodos para actualizar documentos:

```
updateOne({query},{update},{options})
```

Este método actualiza el primer documento que cumpla con los criterios de la consulta.

```
updateMany({query},{update},{options})
```

Este método actualiza todos los documentos que cumplan con los criterios de la consulta.

```
db.people.updateMany(
  {age: {$gt: 25}}, // Comprueba age > 25
  {$set: {status: "C"}} // Establece el campo status a "C"
```

```
{
  "_id" : ObjectId("662fca98efef71a5eb81061c"),
  "user_id" : "bcd000",
  "age" : NumberInt(50),
  "status" : "C"
}
{
  "_id" : ObjectId("662fcaaeefef71a5eb81061d"),
  "user_id" : "bcd0010",
  "age" : NumberInt(50),
  "status" : "C",
  "join_date" : ISODate("2024-04-29T16:28:30.693+0000")
}
{
  "_id" : ObjectId("662fcad1efef71a5eb81061e"),
  "user_id" : "bcd001",
  "age" : NumberInt(45),
  "status" : "C"
}
{
  "_id" : ObjectId("662fcaedefef71a5eb81061f").
```

4.2)

```
UPDATE people
SET age = age + 3
WHERE status = "A";
```

Solución:

```
db.people.find({status: "A"}) // Comprobamos cuantos tenemos con status A
```

```
{
  "_id" : ObjectId("662fcaedefef71a5eb810625"),
  "user_id" : "bcd006",
  "age" : NumberInt(18),
  "status" : "A"
}
```

```
db.people.updateMany(
  {status: "A"}, // Comprueba status = "A"
  {$inc: {age: 3}} // Incrementa la edad en 3
```

```
db.people.find({status: "A"}) // Comprobamos si hizo la actualización
```

```
{
  "_id" : ObjectId("662fcaedefef71a5eb810625"),
  "user_id" : "bcd006",
  "age" : NumberInt(21),
  "status" : "A"
}
```

5. ELIMINACIÓN.

5.1)

```
DELETE FROM people  
WHERE status = "D";
```

Solución:

Existen dos métodos para eliminar documentos:

```
deleteOne({query})
```

Este método elimina el primer documento que cumpla con los criterios de la consulta.

```
deleteMany({query})
```

Este método elimina todos los documentos que cumplan con los criterios de la consulta.

```
db.people.find({status: "A"}) // Comprobamos cuantos tenemos con status D
```

```
{  
  "_id" : ObjectId("662fcaedefef71a5eb810621"),  
  "user_id" : "00bcd001",  
  "age" : NumberInt(15),  
  "status" : "D"  
}
```

```
db.people.countDocuments() // comprobamos cuantos documentos tenemos
```

15

```
db.people.deleteMany({ status: "D" })
```

```
db.people.countDocuments() // comprobamos cuantos documentos tenemos
```

14

5.2)

```
DELETE FROM people;
```

Solución:

```
db.people.deleteMany({})
```

```
{  
  "acknowledged" : true,  
  "deletedCount" : 14.0  
}
```

```
db.people.countDocuments() // comprobamos cuantos documentos tenemos
```

0

6. MOSTRAR BASES DE DATOS Y LISTAR COLECCIONES.

```
show databases // Muestra las base de datos que existen en el servidor
```

```
use nombre_base_datos // Selecciona una base de datos existente o crea una  
nueva base de datos si no existe
```

```
show collections // Lista las colecciones de documentos que existen en la base  
de datos
```