



Implementación y gestión de bases de datos

CRUD

Operaciones CRUD

Guardar datos en MongoDB

La base de datos MongoDB dispone de los siguientes métodos para realizar la inserción de datos:

- `db.collection.insertOne()`

Este método es implementado para insertar un solo documento dentro de la colección, en caso de no existir dicha colección el código construirá una.

Sintaxis:

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

- `db.collection.insertMany()`

Este método es implementado para insertar varios documentos en la colección.

Sintaxis:

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

- `db.collection.insert()`

Este método es multifuncional, es usado para insertar un documento o varios documentos dentro de la colección.

Insertando varios documentos en la colección

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

Consultas

En MongoDB también se puede realizar una búsqueda de todos los documentos dentro de una colección por medio del método `Find()`, como en el caso de SQL sería un `SELECT * FROM`.

- `db.collection.findOne()`

Este método se implementa para retornar la búsqueda de un único documento.

Sintaxis:

```
db.collection_name.findOne()
```

- `db.collection.find().pretty()`

Este método se implementa para que el resultado retorne un formato de fácil lectura.

Sintaxis:

```
db.collection_name.find(<query>).pretty()
```

- Condición AND en MongoDB

Este operador `$and` en MongoDB permite la comparación de dos documentos y retorna el resultado cuando las condiciones coinciden.

Sintaxis:

```
db.collection_name.find({ $and: [ {key_name_1:value_detail_1},  
{key_name_2:value_detail_2} ] })
```

Modificación de datos en MongoDB

La base de datos MongoDB dispone de los siguientes métodos para realizar la modificación de datos:

- `db.collection.updateOne()`

Este método es usado para modificar un único documento dentro de la colección.

```
> db.item_inventory.updateOne(
  { item_name: "journal" },
  {
    $set: { item_quatity: 30 }
  }
)
```

■ db.collection.updateMany()

Este método se implementa para modificar todos los documentos que coindicen con el filtro aplicado.

Sintaxis:

```
> db.item_inventory.updateMany(
  { item_quatity: { $lt: 50 } },
  {
    $set: { item_size.uom: "in" }
  }
)

> db.item_inventory.find( {} )
```

Eliminación de datos en MongoDB

MongoDb nos brinda dos métodos para la eliminación de datos dentro de una colección; a continuación, se explica cada uno de los métodos:

■ db.collection_name.deleteOne()

Este método es implementado para eliminar un solo documento de la colección.

Sintaxis:

```
db.collection.deleteOne({ field_1: value_1, .. })
```

■ db.collection_name.deleteMany()

Este método se implementa para la eliminación de todos los documentos dentro de la colección, en caso de aplicar un filtro solo eliminara los documentos que coincidan con la condición.

Sintaxis:

```
db.collection_name.deleteMany({ field_1: value_1, .. }) // elimina según el filtro
db.collection_name.deleteMany({}) // elimina todos los documentos
```

CRUD

Se presenta a continuación operaciones que se pueden trabajar en MongoDB.

Para iniciar la creación de operaciones en mongoDB se debe construir un nuevo documento por medio de los comandos:

- db.collection_name.insertOne()
- db.collection_name.insertMany()

En el siguiente ejemplo se construye una colección por nombre "user_detail" se ingresan algunos datos:

Figura 1.
Operación de inserción en BD

```
db.users_detail.insertOne(           ← collection
{
    username: "Suresh",            ← field: value
    Age      : 40,                ← field: value
    Status   : "pending"          ← field: value
}
)
```



Nota: adaptado de CloudDuggu. (s/f).

<https://www.cloudduggu.com/mongodb/crud-operations/>

Operaciones de lectura en MongoDB

Las diferentes operaciones de lectura en MongoDB son implementadas para leer las colecciones alojadas en la base de datos.

A continuación, la sintaxis para lectura de una colección:

Figura 2.
Método `find()` en MongoDB

```
db.users_detail.find(
  { user_age: { $gt: 40 } },
  { username: 5, user_address: 5 }
).limit(10)
```



Nota: adaptado de CloudDuggu. (s/t).

<https://www.cloudduggu.com/mongodb/crud-operations/>

Operaciones de actualización en MongoDB

Las operaciones de actualización se implementan para modificar los datos dentro de las colecciones.

Los siguientes métodos son los implementados para realizar las respectivas actualizaciones:

- db.collection_name.updateOne()
- db.collection_name.updateMany()
- db.collection_name.replaceOne()

Figura 3.

Operaciones de actualización en MongoDB

```
db.users_detail.updateMany(           ← collection
    { user_age: { $lt: 40 } },
    { $set: { status: "selected" }})   ← update filter
                                         ← update action
```

Nota: adaptado de CloudDuggu. (s/f).
<https://www.cloudduggu.com/mongodb/crud-operations/>

Operaciones de eliminación en MongoDB

Las operaciones de eliminación se implementan para eliminar los datos dentro de las colecciones. Los siguientes métodos son los implementados para realizar la eliminación de los datos en las colecciones:

- db.collection.deleteOne()
- db.collection.deleteMany()

Figura 4.

Operaciones de eliminación en MongoDB

```
db.users_detail.deleteMany(           ← collection
    { status: "selected" })          ← delete filter
```

Nota: adaptado de CloudDuggu. (s/f).
<https://www.cloudduggu.com/mongodb/crud-operations/>

Guardar datos en MongoDB

En la base de datos MongoDB dispone de los siguientes métodos para realizar la inserción de datos:

1. db.collection.insertOne()

Este método es implementado para insertar un solo documento dentro de la colección, en caso de no existir dicha colección el código construirá una.

Sintaxis:

Figura 5.

Guardar datos en MongoDB

Ejemplo: para el ejemplo nos ubicaremos en la base de datos mongodb_test, e insertaremos un documento en la colección item_inventory:

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

Figura 6.

Ejemplo guardar datos en MongoDB

```
> use mongodb_test  
> db.item_inventory.insertOne(  
    { item_name: "canvas", item_quantity: 115, item_tags: ["cotton"], item_size: { height: 29, weight: 36.5, uom: "cm" } }  
)
```

Para encontrar el documento ingresado en la colección, se realiza por medio del siguiente método:

```
> db.item_inventory.find( { item_name: "canvas" } )
```

2. db.collection.insertMany()

Este método es implementado para insertar varios documentos en la colección.

Sintaxis:

Figura 7.

Insertar varios documentos en MongoDB

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

Ejemplo: por medio de los siguientes comandos se insertan varios documentos en la colección item_inventory de la base de datos mongodb_test:

Figura 8.

Ejemplo insertar varios documentos en MongoDB

```
> db.item_inventory.insertMany(
[ { item_name: "journal", item_quantity: 25, item_tags: ["blank", "red"], item_size: { height: 14, weight: 21, uom: "cm" } },
{ item_name: "mat", item_quantity: 85, item_tags: ["gray"], item_size: { height: 27.9, weight: 35.5, uom: "cm" } },
{ item_name: "laptop", item_quantity: 100, item_tags: ["black"], item_size: { height: 10, weight: 2, uom: "cm" } },
{ item_name: "mouse", item_quantity: 110, item_tags: ["white"], item_size: { height: 3, weight: 4, uom: "cm" } },
{ item_name: "mousepad", item_quantity: 25, item_tags: ["gel", "blue"], item_size: { height: 19, weight: 22.85, uom: "cm" } }
] )
```

Por medio del método find() observamos todos los documentos dentro de la colección:

```
> db.item_inventory.find( {} )
```

3. db.collection.insert()

Este método es multifuncional, es usado para insertar un documento o varios documentos dentro de la colección.

Sintaxis:

Figura 9.

Método insert() en MongoDB

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

Ejemplo: insertar un documento dentro de la colección:

Figura 10.

Ejemplo método insert() en MongoDB

```
> db.products_detail.insert( { item_name: "box", item_quantity: 100 } )
```

Insertando varios documentos dentro de la colección:

Figura 11.

Ejemplo 2 método *insert()* en MongoDB

```
> db.products_detail.insert(
[ { item_name: "pencil", item_quantity: 60, item_type: "no.2" },
  { item_name: "pen", item_quantity: 30 },
  { item_name: "eraser", item_quantity: 65 },
  { item_name: "notebook", item_quantity: 100 },
  { item_name: "pen", item_quantity: 110 }
] )
```

Modificación de datos en MongoDB

En la base de datos MongoDB dispone de los siguientes métodos para realizar la modificación de datos:

4. db.collection.updateOne()

Este método es usado para modificar un único documento dentro de la colección.

Sintaxis:

Figura 12.

Modificar un elemento en MongoDB

```
db.collection_name.updateOne(<filter>, <update>, <options>)
```

Tomando como referencia los valores insertados en el método *Insert()*, actualizaremos el siguiente ítem por nombre "item_name: "journal"" y usaremos el comando *\$set* para asignar el valor de "item_quantity: 30". Una vez ejecutado el comando, evidenciamos si el valor fue modificado por medio del método *find()*.

Figura 13.

Ejemplo modificar un elemento en MongoDB

```
> db.item_inventory.updateOne(
  { item_name: "journal" },
  {
    $set: { item_quantity : 30 }
  }
)

> db.item_inventory.find( { item_name: "journal" } )
```

Figura 14.

Ejecución del método `updateOne` en MongoDB

```
> db.item_inventory.updateOne(
...   { item_name: "journal" },
...   {
...     $set: { item_quantity : 30 }
...   }
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.item_inventory.find( { item_name: "journal" } )
{ "_id": ObjectId("60d3008f180cb8e0c00ebb40"), "item_name": "journal", "item_quantity": 30, "item_tags": "ed"], "item_size": { "height": 14, "weight": 21, "uom": "cm" } }
> █
```

5. `db.collection.updateMany()`

Este método se implementa para modificar todos los documentos que coindicen con el filtro aplicado.

Sintaxis:

```
db.collection_name.updateMany(filter, update, options)
```

Ejemplo:

Figura 15.

Sintaxis método `updateMany` en MongoDB

```
> db.item_inventory.updateMany(
  { "item_quantity": { $lt: 50 } },
  {
    $set: { "item_size.uom": "in" }
  }
)

> db.item_inventory.find( { } )
```

6. `db.collection.replaceOne()`

Este método se implementa para modificar un solo documento según si la condición del filtro.

Sintaxis:

```
db.collection_name.replaceOne(filter, replacement, options)
```

Ejemplo:

Figura 16.

Sintaxis método `replaceOne` en MongoDB

```
> db.item_inventory.replaceOne(
  { item_name: "laptop" },
  { item_name: "laptop", item_size: [ { height: 10, weight: 2, uom: "cm" }, { height: 20, weight: 5, uom: "im" } ] }
)

> db.item_inventory.find( { item_name: "laptop" } )
```

Eliminación de datos en MongoDB

MongoDB nos brinda dos métodos para la eliminación de datos dentro de una colección, a continuación, se explica cada uno de los métodos:

7. db.collection_name.deleteOne()

Este método es implementado para eliminar un solo documento de la colección.

Sintaxis:

```
db.collection.deleteOne({ field_1: value_1, .. })
```

Ejemplo:

Figura 17.

Ejemplo método deleteOne en MongoDB

```
> use mongodb_test  
> db.item_inventory.find( { } )  
> db.item_inventory.deleteOne( { item_name: "journal" } )
```

8. db.collection_name.deleteMany()

Este método se implementa para la eliminación de todos los documentos dentro de la colección, en caso de aplicar un filtro solo eliminará los documentos que coincidan con la condición.

Sintaxis:

Figura 18.

Método deleteMany en MongoDB

```
db.collection_name.deleteMany( { field1: value1, .. } ) // elimina según el filtro  
db.collection_name.deleteMany( {} ) //elimina todos los documentos
```

Ejemplo:

Figura 19.

Ejemplo del método deleteMany en MongoDB

```
> db.item_inventory.deleteMany( { item_name: "mouse" } )  
> db.item_inventory.deleteMany( { } )
```

Consultas

En MongoDB también podemos realizar una búsqueda de todos los documentos dentro de una colección por medio del método `Find()`, como en el caso de SQL sería un `SELECT * FROM`.

Ejemplo:

```
> db.inventory_records.find( {} )
```

En SQL sería su equivalente:

`SELECT * from inventory_records;`

9. `db.collection.findOne()`

Este método se implementa para retornar la búsqueda de un único documento.

Sintaxis:

`db.collection_name.findOne()`

Ejemplo:

```
> db.inventory_records.findOne( { item_name: "canvas" } )
```

Figura 20.

Método buscar en MongoDB

```
> db.inventory_records.findOne( { item_name: "canvas" } )
{
    "_id" : ObjectId("60d54ac71267a85070f02b7d"),
    "item_name" : "canvas",
    "item_qty" : 110,
    "item_size" : {
        "height" : 29,
        "weight" : 36.5,
        "uom" : "cm"
    },
    "status" : "A"
}
```

10. `db.collection.find().pretty()`

Este método se implementa para que el resultado retorne un formato de fácil lectura.

Sintaxis:

`db.collection_name.find(<query>).pretty()`

Ejemplo:

```
> db.inventory_records.find().pretty()
```

Figura 21.
Método pretty en MongoDB

```
> db.inventory_records.find().pretty()
{
    "_id" : ObjectId("60d54ac71267a85070f02b7d"),
    "item_name" : "canvas",
    "item_qty" : 110,
    "item_size" : {
        "height" : 29,
        "weight" : 36.5,
        "uom" : "cm"
    },
    "status" : "A"
}
{
    "_id" : ObjectId("60d54ac71267a85070f02b7e"),
    "item_name" : "jurnal",
    "item_qty" : 35,
    "item_size" : {
        "height" : 15,
        "weight" : 22,
        "uom" : "cm"
    },
    "status" : "A"
}
```

11. Condición AND en MongoDB

Este operador \$and en MongoDB permite la comparación de dos documentos y retorna el resultado cuando las condiciones coinciden.

Sintaxis:

```
db.collection_name.find({ $and: [ {key_name_1:value_detail_1}, {key_name_2:value_detail_2} ] })
```

Ejemplo:

```
> db.inventory_records.find({$and:[ {"status":"A"}, {item_qty: { $lt: 30}} ]})
```

12. Condición OR en MongoDB

Este operador \$or en MongoDB permite la comparación de dos documentos y retorna el resultado al menos una de las condiciones coincide.

Sintaxis:

```
db.collection_name.find({ $or: [ {keyname_1:valuedetail_1}, {keyname_2:valuedetail_2} ] })
```

Ejemplo:

```
> db.inventory_records.find({$or:[ {"status":"D"}, {item_qty: { $lt: 30}} ]})
```

REFERENCIAS BIBLIOGRÁFICAS

CloudDuggu. (s/f). Operaciones CRUD de MongoBD. CloudDuggu.
<https://www.cloudduggu.com/mongodb/crud-operations/>