Certified Tech Developer
The Ultimate Degree

Base de datos II

# Comandos básicos

**db** - Exhibe el nombre de la base de datos que está siendo utilizada en el momento;

> db
mibase
>

**use** - Crea una base de datos:

> use clase
switched to db clase - Informa que ahora estamos usando la base de datos.

De forma predeterminada, el comando MongoDB viene con el nombre de base de datos "test". Supongamos que insertan un documento sin especificar la base de datos, se almacenará automáticamente en la base de datos "test" .

**db.dropDatabase( )** - Elimina una base de datos. Si no se selecciona un banco o una base de datos el test será eliminado.

> use mibase - Selecciona la base.
switched to db mibase - El sistema confirma la base seleccionada.
> db.dropDatabase() - Introduce el comando para eliminar la base.
{ "ok" : 1 } - El sistema informa que la base fue eliminada.
>

**db.createCollections** - Crea una colección. MongoDB es muy flexible, por lo tanto, si no se utiliza el comando, MongoDB crea la colección.

> use mibase – Selecciona la base.

```
switched to db mibase
> db.curso.insert({nombre: "MongoDB"}, {capítulo: "Comandos Básicos"})
— Crea la colección curso e inserta el documento.
WriteResult({ "nInserted" : 1 })
>
```

La sintaxis del comando es: **`db.createCollection(name, options)`**, donde name es el nombre de la colección que se quiere crear y options especifica las opciones sobre el tamaño de memoria e indexación.

Lista de opciones que podemos utilizar para crear una collection:

| Campo | Tipo | Descripción |
|---|---|---|
| capped | Boolean | (Opcional) Si es verdadero (true), habilita el *capped collection* - una colección de tamaño fijo establecido que sobrescribe las entradas más antiguas cuando se alcanza el tamaño máximo. Si especifican verdadero, precisan especificar el tamaño como parámetro. |
| autoIndexID | Boolean | (Opcional) Si es verdadero (true), automáticamente creará un índice para el campo _id. El valor predeterminado es falso. |
| size | number | (Opcional) Especifica un tamaño máximo en bytes para el capped collection limitada. Si el capped es verdadero, también se debe especificar este campo. |
| max | number | (Opcional) Especifica el número máximo de documentos permitidos en el *capped collection*. |

**insert( )** - Inserta un documento en una colección.

```
> db.curso.insert({_id: 123, título: 'Curso MongoDB', descripción: 'El curso de MongoDB tiene como prerrequisito el MySQL.', autor: 'Janete Ferreira', tags: ['mongodb', 'database', 'noSQL']})
WriteResult({ "nInserted" : 1 })
>
```

**find( )** - Consulta datos en las colecciones de MongoDB. Para exhibir los resultados de un modo formateado, pueden usar el método pretty( ).

```
> db.curso.find().pretty()
{ "_id" : ObjectId("61f35b43d0bb2f17ea8e6b55"), "nombre" : "MongoDB" }
{
        "_id" : 123,
        "título" : "Curso MongoDB",
          "descripción" : "El curso de MongoDB tiene como prerrequisito el MySQL.",
        "autor" : "Janete Ferreira",
        "tags" : [
                "mongodb",
                "database",
                "noSQL"
        ]
}
>
```

El **find()** también permite que hagan una proyección de los datos que desean exhibir. Por ejemplo, si sus documentos tienen dos campos y quieren exhibir solo uno, hagan lo siguiente:

```
> db.mycol.find({},{titulo: 1, _id: 0})
{ "título" : "MongoDB" }
{ "título" : "MySQL" }
{ "título" : "Cassandra" }
{ "título" : "SQL Server" }
```

>

Observen el ejemplo de arriba. Si no quieren mostrar el id, deben setearlo como 0.

**limit( )** - Limita los registros en el MongoDB. Este método acepta un tipo numérico como argumento, que es el número de documentos que desean que sea exhibido.

```
> db.mycol.find().limit(2)
{ "_id" : 83, "título" : "MongoDB" }
{ "_id" : 84, "título" : "MySQL" }
>
```

**sort( )** - Ordena documentos. Este método acepta un documento que contiene una lista de campos acompañada de su valor de clasificación. Para especificar el tipo de clasificación, se usa 1 para establecer el orden ascendente mientras que -1 se usa para establecer el orden descendente.

```
> db.mycol.find({}, {titulo: 1, _id: 0}).sort({título: -1})
{ "título" : "SQL Server" }
{ "título" : "MySQL" }
{ "título" : "MongoDB" }
{ "título" : "Cassandra" }
> db.mycol.find({}, {título: 1, _id: 0}).sort({título: 1})
{ "título" : "Cassandra" }
{ "título" : "MongoDB" }
{ "título" : "MySQL" }
{ "título" : "SQL Server" }
>
```

**skip( )** - Lista los registros a partir de la cantidad informada.

> db.mycol.find().skip(2) - Significa que la lista será exhibida a partir del 3° registro.



**Operadores Relacionais:**

$eq - equal - igual
$lt - low than - menor que
$lte - low than equal - menor o igual que
$gt - greater than - mayor que
$gte - greater than equal - mayor o igual que
$ne - not equal - diferente
$in - in - dentro de
$nin - not in - fuera de

**Ejemplo:**

db.mycol.find({_id : ({$lt : 86})})
{ _id: 83, título: 'MongoDB' }
{ _id: 84, título: 'MySQL' }
{ _id: 85, título: 'Cassandra' }