

# **Introducción a MongoDB y Operaciones CRUD**

Bases de Datos 2024

# Introducción a MongoDB

# MongoDB

MongoDB es una base de datos de **documentos** con un modelo de datos flexible

## Documento

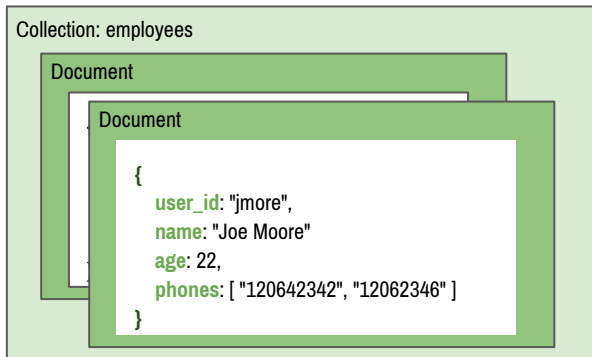
```
{
  "_id": "153499",
  "name": "Joe Moore"
  ← field: value
  "age": 22,
  "phones": [ "120642342", "12062346" ] ← arreglo
  "address": {
    "street": "Joe",
    "phones": Moore" ← documento anidado
    "state": "Moore"
  }
}
```

- MongoDB almacena los datos como **documentos** similares a JSON (documentos **BSON**)
- Un documento es una estructura de datos compuesta de **pares campo-valor**.
- Cada **campo** debe estar entre comillas
- El **valor** de cada campo puede ser cualquiera de los [tipos de datos BSON](#)

# Colecciones y bases de datos

- MongoDB almacena los documentos en **colecciones**
- Una **base de datos** almacena **una o más** colecciones de documentos

## Colección



```
> use employees
switched to db employees
> db.createCollection("employees")
> db.employees.insertOne(
  {"user_id": "jmore", "name": "Joe Moore", "age": 22}
)
...
> db.employees.find()
[
  {
    _id: ObjectId("635098ad6ef3e1db925dbb3e"),
    user_id: 'jmore',
    name: 'Joe Moore',
    age: 22
  }
]
```

# Documentos BSON

BSON es la representación binaria de documentos JSON aunque tiene más tipos de datos que JSON

## Documento BSON

```
// bios_collection
{
  _id: ObjectId("5099803df3f4948bd2f98391"),
  name: {
    first: "Alan",
    last: "Turing"
  },
  birth: new Date('Jun 23, 1912'),
  death: new Date('Jun 07, 1954'),
  contribs: [
    "Turing machine",
    "Turing test",
    "Turingery"
  ],
  views : NumberLong(1250000)
}
```

## El campo \_id

- Reservado y actúa como la **clave primaria**
- Es inmutable
- Si al insertar el documento se omite el campo \_id, se genera un **ObjectId** para este campo

# MongoDB Shell (mongosh y mongo)

## Comandos básicos

- `show dbs`
  - Lista todas las bases de datos en el servidor
- `use <db>`
  - Cambia la base de datos actual a <db>
- `db`
  - Variable que representa la base de datos actual luego de ejecutar el comando `use`
- `show collections`
  - Lista todas las colecciones de la base de datos actual
- `db.createCollection(name, <options>)`
  - Crea una nueva colección
- `db.<collection>.help()`
  - Muestra ayuda sobre los métodos de la colección

# Operaciones CRUD

# Operaciones CRUD

Permiten crear (Create), leer (Read), actualizar (Update) y eliminar (Delete) documentos

Operaciones CRUD usando MQL (MongoDB Query Language)

- `db.<collection>.insertOne( <document> )`
- `db.<collection>.insertMany( [ <doc1>, ... , <docN> ] )`
- `db.<collection>.findOne( <query filter>, <projection> )`
- `db.<collection>.find( <query filter>, <projection> )`
- `db.<collection>.updateOne( <query filter>, <update>, <options> )`
- `db.<collection>.updateMany( <query filter>, <update>, <options> )`
- `db.<collection>.deleteOne( <query filter> )`
- `db.<collection>.deleteMany( <query filter> )`



# CRUD - INSERT

## ➤ InsertOne

```
db.<collection>.insertOne(  
    <document>,  
    <options>  
)
```

## ➤ Ejemplo

```
db.movies.insertOne(  
    {  
        "title": "Star Trek II: The Wrath of Khan",  
        "year": 1982,  
        "type": "movie"  
    }  
)
```

## ➤ InsertMany

```
db.<collection>.insertMany(  
    [ <document1>, ... ,<documentN> ]  
    <options>  
)
```

## ➤ Ejemplo

```
db.movies.insertMany([  
    { "_id": "tt0796366", "title": "Star Trek", "year": 2009,  
      "type": "movie" },  
    { "_id": "tt1408101", "title": "Star Trek Into Darkness",  
      "year": 2013, "type": "movie" },  
    { "_id": "tt0117731",  
      "title": "Star Trek: First Contact", "year": 1996 }  
)
```

# CRUD - FIND

## ➤ Colección inventory

```
db.inventory.insertMany([  
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  
  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },  
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },  
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },  
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }  
]);
```

## ➤ Find

```
db.<collection>.find(  
  <query filter>,  
  <projection>  
)
```

- **query filter** especifica el filtro o criterio de selección mediante **operadores de selección**
- **projection** especifica los campos a devolver de los documentos que matchean con el filtro de selección.
- **find** retorna los documentos que matchean con el criterio de selección (el resultado es un **cursor**)
- Ejemplo

```
db.inventory.find( { "status": { "$eq": "A" } }, { "item": 1 } )
```

= {Azúcar sintáctico de \$eq}

```
db.inventory.find( { "status": "A" }, { "item": 1 } )
```

# CRUD - FIND - Projection

- El parámetro **projection** es opcional.

`{ <field1>: <value>, <field2>: <value> ... }`

- **<field>: <1 or true>** → especifica inclusión

`db.inventory.find( {}, { "item": 1, "qty": 1 } )`

- **<field>: <0 or false>** → especifica exclusión

`db.inventory.find( {}, { "qty": 0 } )`

- Por defecto se incluye el campo `_id`

`db.inventory.find( {}, { "item": 1, "qty": 1, "_id": 0 } )`

- Proyectar campos anidados con la notación `"."` ("field.nestedField")

`db.bios.find( {}, { "name.last": 1 } )`

- **0** usando la forma anidada

`db.bios.find( {}, { "name": { "last": 1 } } )`

- **<field>: <aggregation expression>**

Con el uso de expresiones de agregación se pueden proyectar nuevos campos o proyectar existentes con nuevos valores

# CRUD - FIND - Operadores de Selección

## ➤ Operadores de Comparación

```
db.<collection>.find(  
    { <field>: { <operator>: <value> }, ... }  
)
```

## ➤ Operadores

- `$eq`            `$nq`
- `$gt`            `$gte`
- `$lt`            `$lte`
- `$in`            `$nin`

## ➤ Ejemplos

```
db.inventory.find(  
    { qty: { $lt: 30 } }  
)
```

```
db.inventory.find(  
    { "status" : { $in: ["A", "D"] } }  
)
```

# CRUD - FIND - Operadores de Selección

## ➤ Operadores Logicos

```
db.<collection>.find(  
    { <operator>: [ { clause1 }, { clause2 }, ... ] }  
)
```

```
db.<collection>.find(  
    { <operator>: { clause } }  
)
```

## ➤ Operadores

- \$and
- \$or            \$nor
- \$not

## ➤ Ejemplos

```
db.inventory.find( { $and: [ { "status": "A" }, { "qty": { $lt: 30 } } ] } )
```

= {Azúcar sintáctico de \$and}

```
db.inventory.find( { "status": "A" , { "qty": { $lt: 30 } } } )
```

```
db.inventory.find(  
    { $or: [ { status: "A"}, { qty: { $lt: 30 } } ] }  
)
```

```
db.inventory.find( {  
    status: "A", $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]  
} )
```

# CRUD - FIND - Consulta en documentos anidados

- Matchear un documento anidado

```
db.inventory.find( { size: { h: 8.5, w: 11, uom: "in" } } )
```

- Especificar condiciones sobre campos anidados usando la notación "."

```
db.inventory.find( { "size.w": { $gte: 16 } } )
```

# CRUD - FIND - Consulta en arreglo

## ➤ Colección food

```
db.food.insertMany([  
  { _id: 1, fruits: ["apple", "banana", "mango"], por_sizes: [2, 3, 5] },  
  { _id: 2, fruits: ["apple", "lemon", "orange"], por_sizes: [1, 5] },  
  { _id: 3, fruits: ["cherry", "banana"], por_sizes: [1, 2] }  
])
```

## ➤ Matchear un arreglo

```
db.food.find( { "fruits": ["cherry", "banana"] } )
```

## ➤ Operadores de consulta de arreglos

```
db.<collection>.find(  
  { <array field>: { <operator>: <value> }, ... }  
)
```

## ➤ Operadores

- **\$all**: matchea si el campo arreglo contiene todos los elementos especificados en value
- **\$elemMatch**: matchea si al menos un elemento en el campo arreglo cumple todas las condiciones especificadas
- **\$size**: matchea si el campo arreglo es del largo especificado

## ➤ Ejemplos

```
db.food.find( { fruits: { $all: ["apple", "banana"] } } )
```

```
db.food.find( {  
  por_sizes: { $elemMatch: { $gt: 2, $lte: 4 } }  
} )
```

```
db.food.find( { por_sizes: { $size: 3 } } )
```

# CRUD - FIND - Consulta en arreglo de documentos

## ➤ Colección survey

```
db.survey.insertMany([
    { _id: 1, results: [ { product: "abc", score: 10 }, { product: "xyz", score: 5 } ] },
    { _id: 2, results: [ { product: "abc", score: 8 }, { product: "xyz", score: 7 } ] },
    { _id: 3, results: [ { product: "abc", score: 7 }, { product: "xyz", score: 8 } ] }
])
```

## ➤ Ejemplos:

```
db.survey.find( { "results.product": "xyz", "results.score": { $gt: 7 } } )
```

```
db.survey.find( {
    results: {
        $elemMatch: { product: "xyz", score: { $gt: 7 } }
    }
} )
```



# CRUD - FIND - Consulta por nulos o campos ausentes

## ➤ Colección inventory

```
db.inventory.insertMany([  
  { _id: 1, item: null },  
  { _id: 2 }  
])
```

## ➤ Matchear por null o campo ausente

```
db.inventory.find( { item: null } )
```

## ➤ Matchear por tipo

```
db.inventory.find( { item: { $type: 10 } } )
```

## ➤ Matchear por existencia

```
db.inventory.find( { item: { $exists: false } } )
```

# CRUD - FIND - Métodos del cursor - SORT SKIP LIMIT

## ➤ Sort, Skip, y Limit

```
db.<collection>.find(  
    <query filter>,  
    <projection>  
)  
.sort(  
    { <field1>: <1 or -1>, <field2>: <1 or -1> ... }  
)  
.skip(  
    <offset>  
)  
.limit(<number>)
```

## ➤ Más métodos del cursor

## ➤ Ejemplo

//Listar los 3 items con mayor cantidad dentro de los inventarios con estado "A". Listar en orden alfabético si los items tiene la misma cantidad.

```
db.inventory.find(  
    { "status": "A" },  
    { "item": 1, "qty": 1 }  
)  
.sort(  
    { qty: -1, item: 1 }  
)  
.skip(  
    0  
)  
.limit(3)
```

# CRUD - UPDATE - DELETE

## ➤ Update y Operadores de Actualización

```
db.inventory.updateOne(
  { item: "paper" },
  {
    $set: { "size.uom": "cm", status: "P" },
    $currentDate: { lastModified: true }
  }
)

db.food.updateMany(
  { _id: { $in: [1, 3] } },
  {
    $addToSet: { fruits: { $each: ["cherry", "pear"] } }
  }
)
```

## ➤ Más operadores de actualización

## ➤ Upsert (Update + Insert)

```
db.analytics.updateOne(
  { url: "/blog" },
  { $inc: { pageviews : 1 } },
  { upsert: true }
)
```

## ➤ Delete

```
db.inventory.deleteMany( { status: "A" } )
```

# Temas a estudiar

- Próxima clase
  - Pipeline de agregación
  - Vistas
- Referencias
  - [Operaciones CRUD](#)