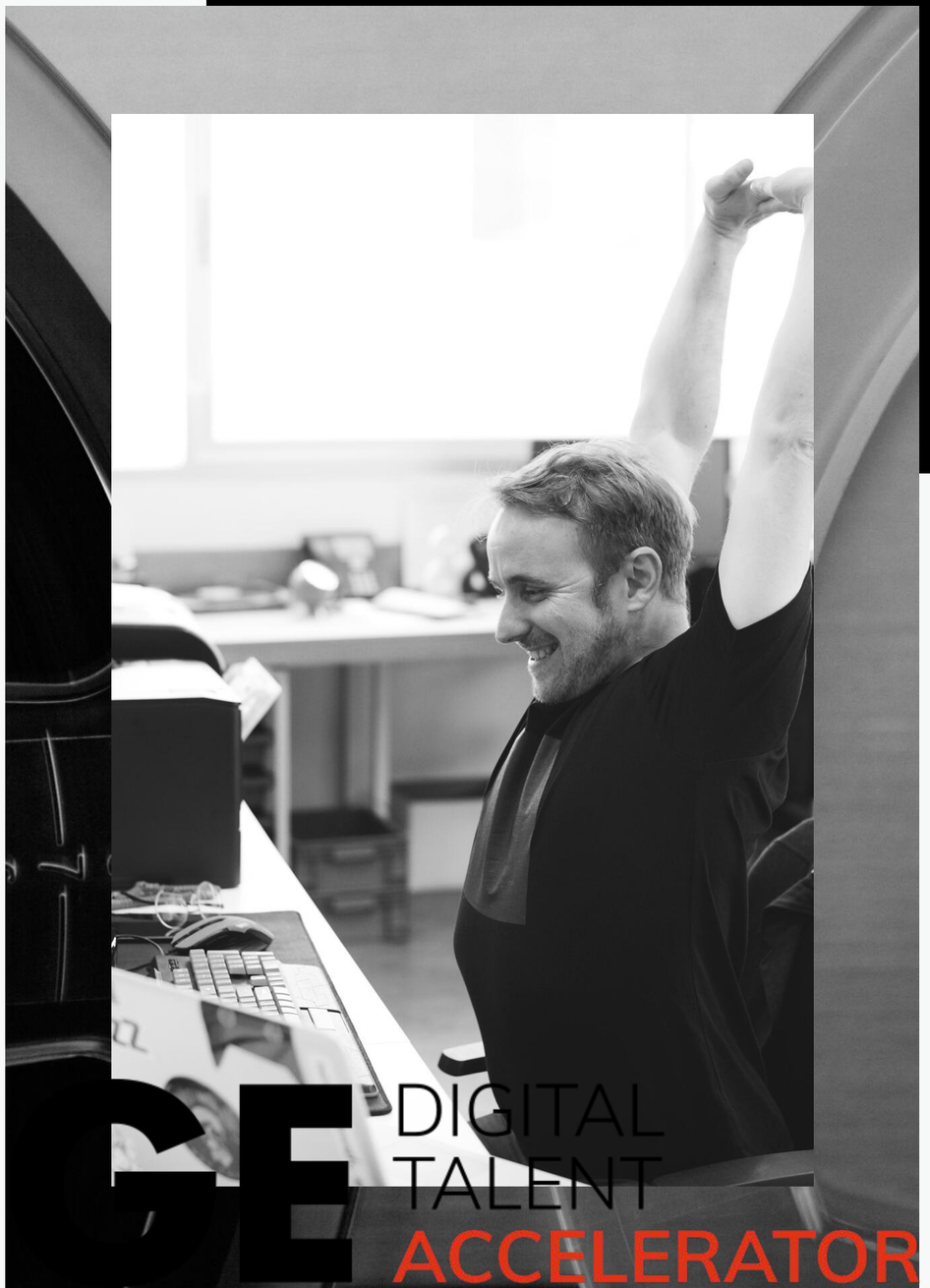
NoSQL & MongoDB

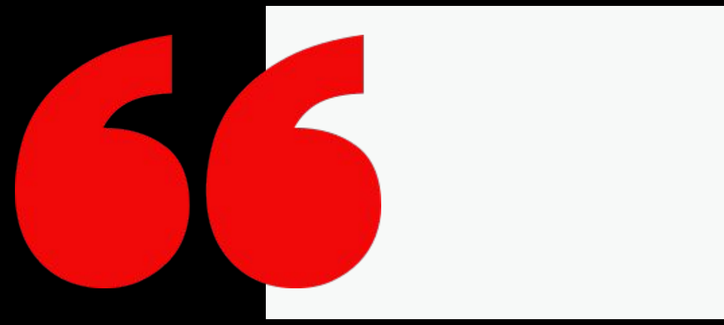
Fundación ONCE | Inserta | Cofinanciado por la Unión Europea | Ministerio de Trabajo y Economía Social | Fondos Europeos

Índice



THE BRIDGE DIGITAL TALENT ACCELERATOR

¿Que es NoSQL?



NoSQL

Las bases de datos **NoSQL** (también conocidas como "**no solo SQL**") **almacenan datos de manera diferente a las tablas relacionales**. Las bases de datos NoSQL vienen en una variedad de tipos según su modelo de datos.
Proporcionan **esquemas flexibles y se escalan fácilmente** con grandes cantidades de datos y una alta carga de usuarios.

MongoDB

Clasificada como la base de datos NoSQL más popular del mundo

MongoDB

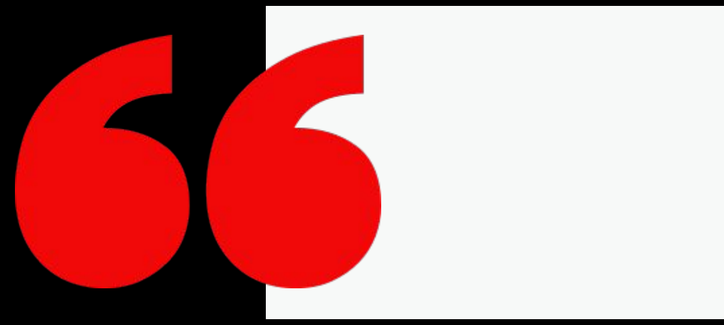
MongoDB es una **base de datos basada en documentos**, creada para desarrolladores de aplicaciones modernas y para la era de la nube.

Además, es una **base de datos NoSQL** orientada a documentos que se utiliza para el almacenamiento de datos de gran volumen. En lugar de usar tablas y filas como en las bases de datos relacionales tradicionales, **MongoDB hace uso de colecciones y documentos**.

Términos comunes utilizados en MongoDB

A continuación se muestran algunos de los términos comunes utilizados en MongoDB

- **_id**: este es un campo obligatorio en todos los documentos de MongoDB. El campo _id representa un **valor único** en el documento MongoDB. El campo **_id es como la clave principal del documento**.
- **Colección**: esta es una agrupación de documentos MongoDB. Una colección es el **equivalente a una tabla en MySQL**.
- **Documento**: un **registro en una colección de MongoDB** se denomina básicamente documento. El documento, a su vez, consta de nombre de campo y valores.
- **Campo**: un par de nombre y valor en un documento. Un documento tiene cero o más campos. Los campos son análogos a las **columnas en las bases de datos relacionales**.

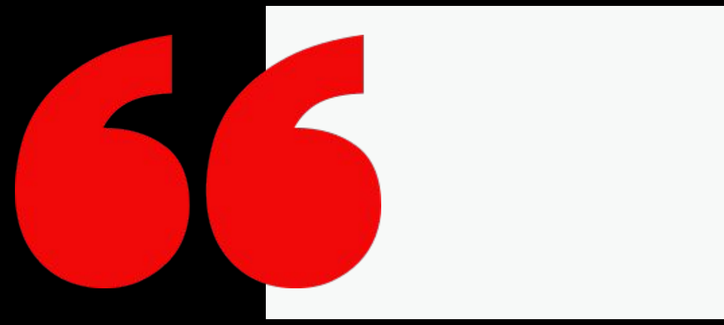


Como programador, piensas en objetos. MongoDB también lo hace.

MongoDB como base de datos de documentos

MongoDB es una **base de datos de documentos**, lo que significa que almacena datos en documentos similares a JSON. Creen que esta es la forma más natural de pensar en los datos y es mucho más expresiva y poderosa que el modelo tradicional de filas y columnas.

```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  },
  "hobbies": ["surfing", "coding"]
}
```



Empezando

Instalación, creación de base de datos, uso de base de datos y creación de una colección.

Instalación

MongoDB

https://www.mongodb.com/try/download/community

Mongosh

https://www.mongodb.com/try/download/shell

Empezando

Vamos a crear nuestra base de datos también podemos eliminarla si queremos.

```
mongosh
// Crear base de datos
use myFirstDB
// Eliminar base de datos
db.dropDatabase().myFirstDB
// Seleccionar base de datos
use myFirstDB
// Mostrar base de datos
show dbs
```

Crear colecciones

Vamos a crear nuestra colección de productos, también podemos eliminarla si queremos.

```
db.createCollection("products")

db.products.drop()
```

MongoDB CRUD

Insert, find, update & delete

Insert

INSERT inserta un nuevo documento en nuestra colección de productos.

```
db.products.insertOne({
  name: 'Portatil Apple MacBook',
  price: 889,
  category: 'Computers',
  date: Date()
})
```

Insert Many

Para insertar más de un producto, ejecuta lo siguiente:

```
db.products.insertMany([
   { name: "Portatil Asus", price: 998, category: "Computers",date: Date()},
   { name: "Nintendo Switch", price: 374.66, category: "Consoles",date: Date()},
   { name: "Breath of the Wild", price: 58.98, category: "Video games",date: Date()},
   { name: "Playstation 4", price: 449.95, category: "Consoles"}
])
```

Find

Podemos obtener todas las filas con la siguiente consulta:

```
db.products.find()
```

Hacer coincidir un documento incrustado/anidado

Para especificar una condición de igualdad en un campo que es un documento incrustado/anidado, utilice el documento de **filtro de consulta { <campo>: <valor> }** donde **<valor> es el documento que debe coincidir**.

Delete

También podemos eliminar todos los productos o eliminar solo uno.

```
// Eliminar todos los documentos
db.products.deleteMany({})
// Eliminar una fila
db.products.deleteOne({ name: 'Nintendo Switch' })
```

Encontrar documentos

Por ejemplo, la siguiente consulta selecciona todos los documentos en los que el campo categoría es igual a 'Computers'.

```
db.products.find({ category: 'Computers' })
```

Encontrar un documento

- **Devuelve un documento que cumple los criterios de consulta** especificados en la colección.
- **Si varios documentos satisfacen la consulta**, este método **devuelve el primer documento** según el orden natural que refleja el orden de los documentos en el disco.
- **Si ningún documento satisface la consulta, el método devuelve nulo**.

```
db.products.findOne({ category: 'Computers' })
```

Consultar campos específicos

Además, podemos consultar los campos específicos que queramos.

```
db.products.find({ name: 'Breath of the Wild' }, {
  name: 1,
  price: 1
})
```

And o Or

And o Or para el find()

```
db.products.find({ $or:[
{price :400},{price:374.66}
]})
```

Comandos MongoDB para el find()

Comandos MongoDB para el find(), los criterios de búsqueda son equivalentes a las cláusulas where de SQL:

```
db.products.find({price:{$lt:400}})   // where price < 400
db.products.find({price:{$lte:400}})  // where price<= 400
db.products.find({price:{$gt:410}})   // where price> 410
db.products.find({price:{$gte:410}})  // where price>=410
db.products.find({price:{$ne:500}})   // where price!=500
```

Actualizar un documento

- El método **db.collection.updateOne(query, update, options)** modifica el primer documento que cumpla los criterios de consulta de una colección.
- El método puede modificar campos específicos de un documento o reemplazar un documento existente por completo, según el parámetro de actualización.

```
db.products.updateOne({ name: 'Breath of the Wild' },
{ $set:{name: "Breath of the Wild", price: 58.99, category: "Video games",date: Date()}})
```

Actualizar un documento

Actualizamos todas las coincidencias:

```
db.products.updateMany({ name: 'Breath of the Wild' },
{ $set:{name: "Breath of the Wild", price: 500, category: "Video games",date:
Date()}})
```

Actualizar un campo

Con **$set** podemos actualizar un campo específico.

```
db.products.updateOne({ name: 'Breath of the Wild'
},
{
  $set: {
    price: 33
  }
})
```

Sort

Especifica en el parámetro de **sort** el campo o campos a ordenar y el valor 1 o -1 para especificar si es ascendente o descendente el orden.

```
# asc
db.products.find().sort({ price: 1 })
# desc
db.products.find().sort({ price: -1 })
```

Count

El método **db.collection.count()** no realiza la operación find() sino que **cuenta y devuelve el número de resultados** que coinciden con una consulta.

```
db.products.find().count()

db.products.find({ category: 'Computers' }).count()
```

Limit

Utilice el método **limit()** para especificar el número máximo de documentos que devolverá:

```
db.products.find().limit(2)
```

Encadenando

Encadenamos múltiples métodos de MongoDB.

```
db.products.find().sort({ price: -1}).limit(3)
```

Foreach

El siguiente ejemplo invoca el método **forEach()** en los productos devueltos por **find()** para imprimir el nombre de cada producto en la colección:

```
db.products.find().forEach((doc)=> {
  print("Product name: " + doc.name)
})
```

$inc

- Podemos incrementar un campo.

  En este caso, incrementaremos el precio en 5.

```
db.products.updateOne({ name:
'Breath of the Wild' },
{
  $inc: {
    price: 5
  }
})
```

$mul

- Podemos incrementar un campo con multiplicación. En este caso aumentamos un 50% el precio:

```
db.products.update({"name":"Playstat
ion 4"},
{
  $mul: {
  "price": 1.5
  }
});
```

$rename

- El operador **$rename** actualiza el nombre de un campo.

```
db.products.updateOne({ name: 'Playstation 4'
}, { $rename: { name: 'title'}})
```

Sub-documentos

- A diferencia de MySQL, con MongoDB si deseas añadir comentarios a tu colección de productos, puedes hacerlo dentro de la misma colección sin tener que crear una nueva colección.

```
db.products.updateOne({ name: 'Portatil Asus' },
{
  $set: {
    comments: [
      {
        body: 'Comment One',
        user: 'Mary Williams',
        date: Date()
      },
      {
        body: 'Comment Two',
        user: 'Harry White',
        date: Date()
      }
    ]
  }
})
```

$elemMatch

También podemos encontrar el producto por un elemento que está en un Array (**$elemMatch**) como podemos ver en el ejemplo.

```
db.products.find({
  comments: {
    $elemMatch: {
      user: 'Mary Williams'
      }
    }
  })
```

Operador $text

Utiliza el operador de consulta **$text** para realizar búsquedas de texto en una colección con un índice de texto.

Crear índices incrementa la velocidad a la hora de buscar y también a la hora de devolver resultados ya ordenados.

Sin los índices, MongoDB debe hacer un análisis de la colección, es decir, leer todos los documentos en una colección, con el fin de encontrar los documentos que cumplan con los criterios de consulta.

Text Index

- MongoDB proporciona índices de texto para admitir consultas de búsqueda de texto de string.
- Para realizar consultas de búsqueda de texto, debe tener un índice de texto en su colección.
- Por ejemplo, puede ejecutar lo siguiente en un shell mongo para permitir la búsqueda de texto sobre el campo de nombre:

```
db.products.createIndex({ name: 'text' })
```

$text

Por ejemplo, podríamos usar la siguiente consulta para buscar todos los productos que contengan "Nintendo S":

```
db.products.find({
  $text: {
    $search: "Nintendo S"
    }
})
```