



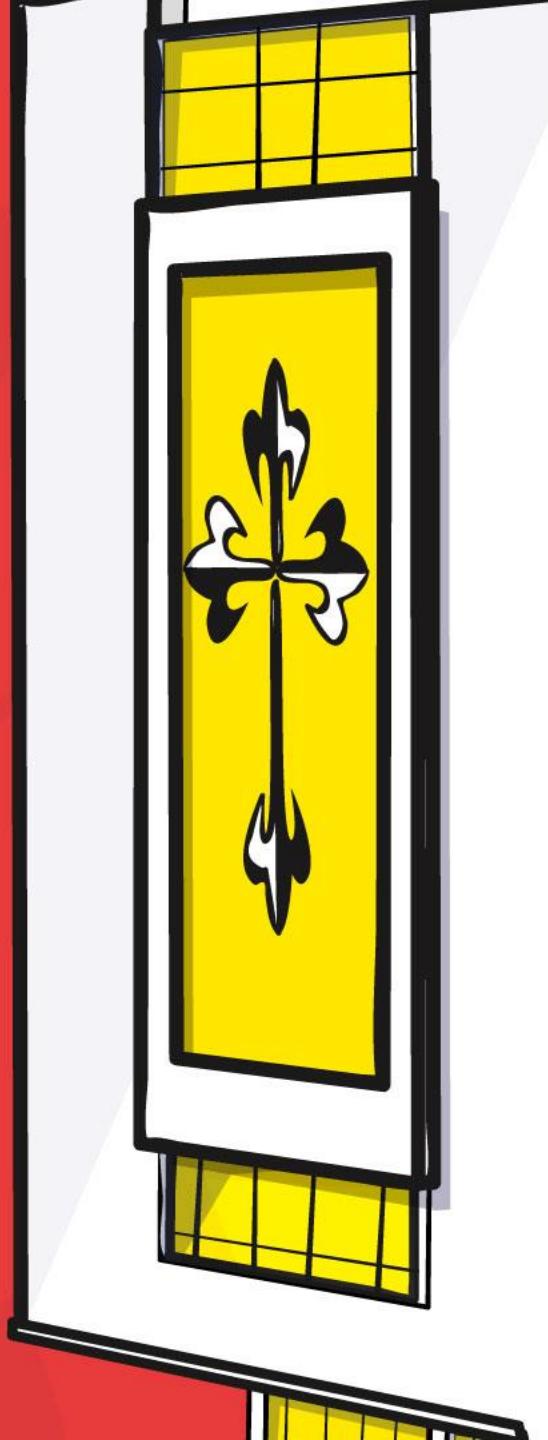
SANTOTO
Seccional Tunja

www.ustatunja.edu.co



NoSQL

Facultad de Ingeniería de Sistemas



Temas a desarrollar:

- ✓ *Tendencias actuales*
- ✓ *Bases de datos semi- estructuradas*
- ✓ *CRUD con MongoDB*



TIPOS DE DATOS



Datos no estructurados 85%

- Provienen de la Web, Redes Sociales e Internet de las Cosas, en formato texto, audio, video, imagen.



Datos estructurados 15%

- Son internos de compañías: clientes, productos, transacciones, etc.



¿Cuál es la tendencia?

ESCALAMIENTO

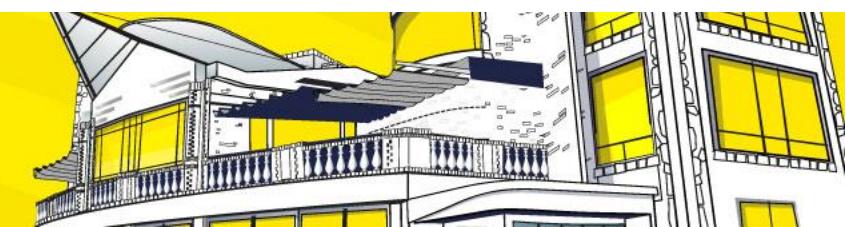
VERTICAL SCALING

Increase size of instance
(RAM, CPU etc.)



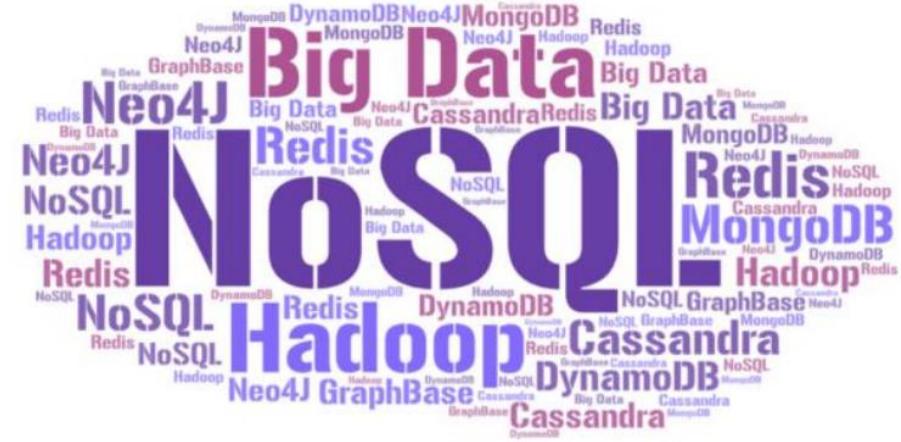
HORIZONTAL SCALING

(Add more instances)



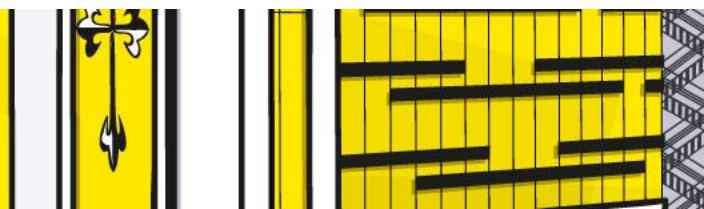
NOSQL

NoSQL o "no sólo SQL", esto hace referencia a que dichas bases de datos **difieren** del modelo clásico **RDBMS**, estas son especialmente útil cuando una empresa necesita acceder y analizar grandes cantidades de datos o su enfoque arquitectural esta diseñado para distribuir los centros de datos y sus esquemas de datos no son fijos.



CARACTERÍSTICAS DE NoSQL

- No requiere un esquema de datos fijo
- Velocidad en el manejo de grandes volúmenes de datos
- Transacciones tolerantes a fallos
- Faciles de usar en clusters y balanceadores de carga convencionales
- Se ejecutan en máquinas con pocos recursos



TEOREMA CAP

TEOREMA CAP de Eric Brewer: Consistency –Availability -Partition

Consistencia – Disponibilidad – Tolerancia a partición

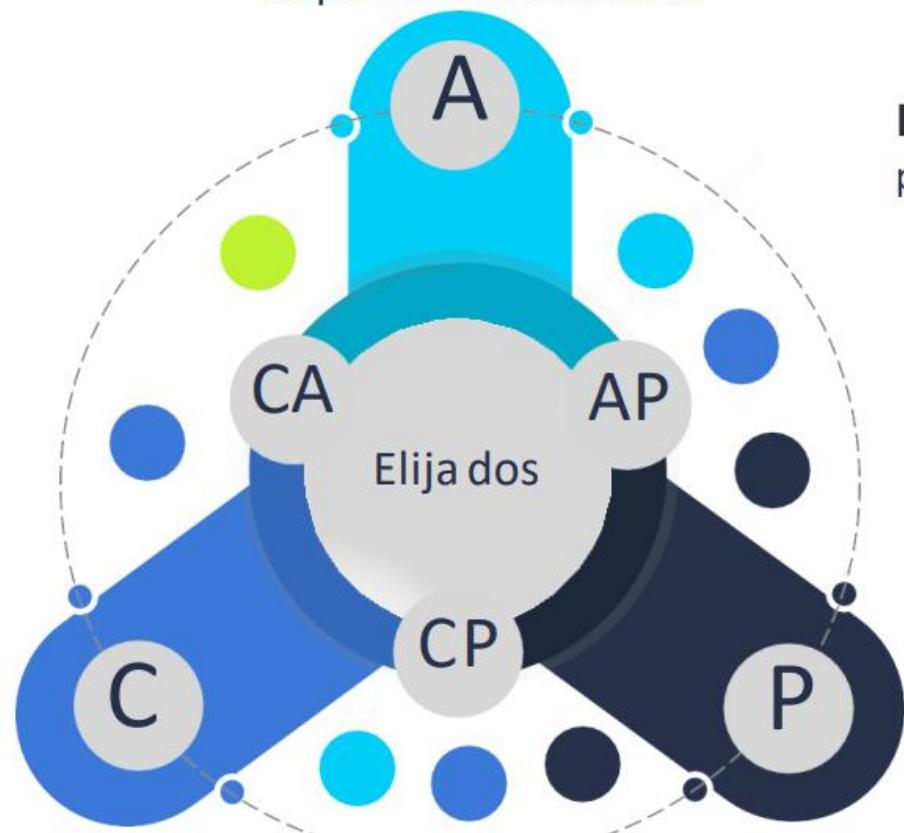
El teorema establece que hay **tres requisitos** espaciales básicos en el diseño de aplicaciones para una arquitectura distribuida.

Modelos de datos que aplican:

- RDBMS
- Clave-Valor
- Orientado a Columnas
- Orientado a Documentos

Consistencia:

Todos los clientes siempre tienen la misma vista de los datos



Disponibilidad: Cada cliente puede siempre leer y escribir

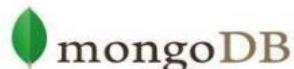
Tolerancia apartición: el sistema funciona bien a través de particiones físicas de la red.

CATEGORÍAS DE BASES DE DATOS NOSQL

Columnares



Documentales



Clave-valor



Orientada a Grafos

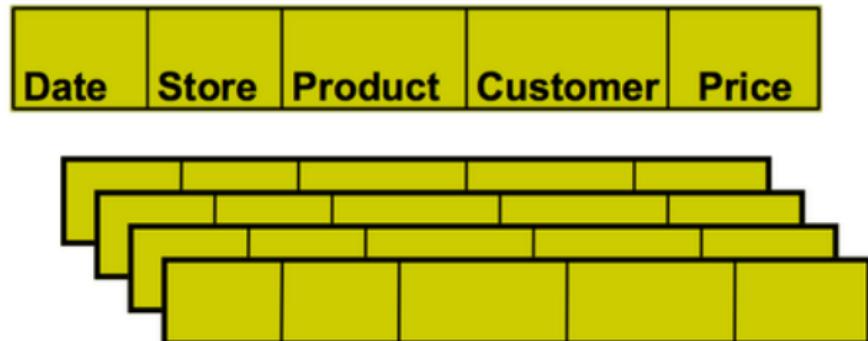


<https://db-engines.com/en/ranking>

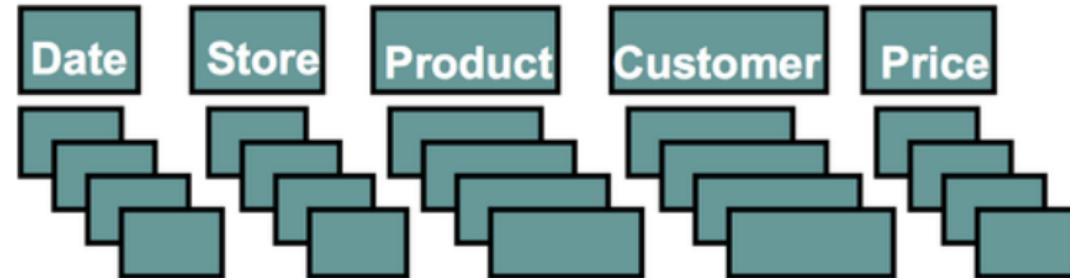
BASES DE DATOS COLUMNARES

Las bases de datos, en lugar de estar **estructuradas** por filas, están estructuradas **por columnas**. Al tratarse de una sola dimensión, hace más eficiente la recuperación de la información. Sin embargo, puede tener alguna desventaja de rendimiento para la inserción

row-store



column-store



BASES DE DATOS COLUMNARES

Almacenamiento de Datos

Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura



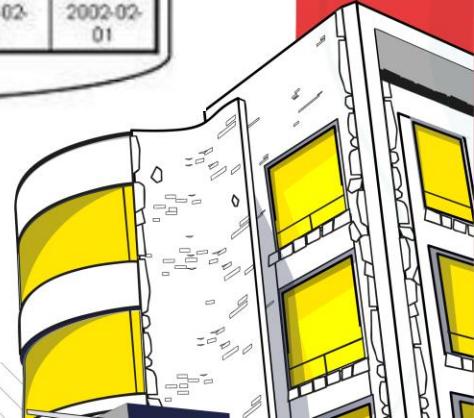
Row-Oriented Database

1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue



Column-Oriented Database

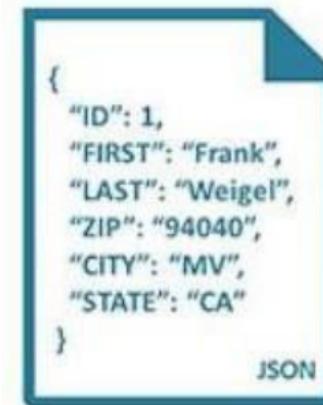
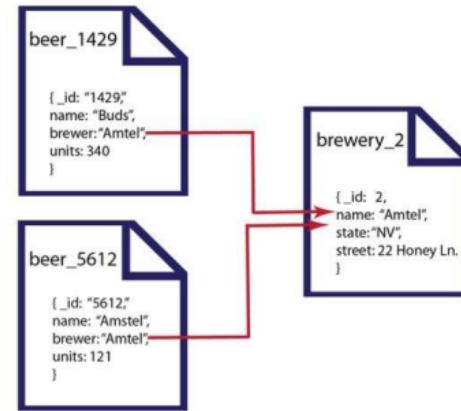
1	2	3	4	5
1	1	1	2	2
2001-01-01	2002-02-01	2002-02-01	2002-02-01	2002-02-01



BASES DE DATOS DOCUMENTALES

El documento encapsula la información en algún formato o codificación estándar: XML, YAML, JSON o BSON e incluso formatos binarios como PDF, documentos de Microsoft Office, etc.

```
{  
    FirstName: "Jonathan",  
    Address: "15 Wanamassa Point Road",  
    Children: [  
        {Name: "Michael", Age: 10},  
        {Name: "Jennifer", Age: 8},  
        {Name: "Samantha", Age: 5},  
        {Name: "Elena", Age: 2}  
    ]  
}
```



The diagram shows a JSON document with the following structure:

```
{  
    "ID": 1,  
    "FIRST": "Frank",  
    "LAST": "Weigel",  
    "ZIP": "94040",  
    "CITY": "MV",  
    "STATE": "CA"  
}
```

Below the JSON document is the word "JSON". To the right of the JSON document is a table titled "User Info" with four columns: KEY, First, Last, and ZIP_id. The table has four rows, each representing a user:

KEY	First	Last	ZIP_id
1	Frank	Weigel	2
2	Ali	Dodson	2
3	Mark	Azad	2
4	Steve	Yen	3

BASES DE DATOS DE CLAVE - VALOR

Este tipo de motores de bases de datos almacenan información en «diccionarios». Podemos imaginar una BD clave–valor como una colección de vectores asociativos.

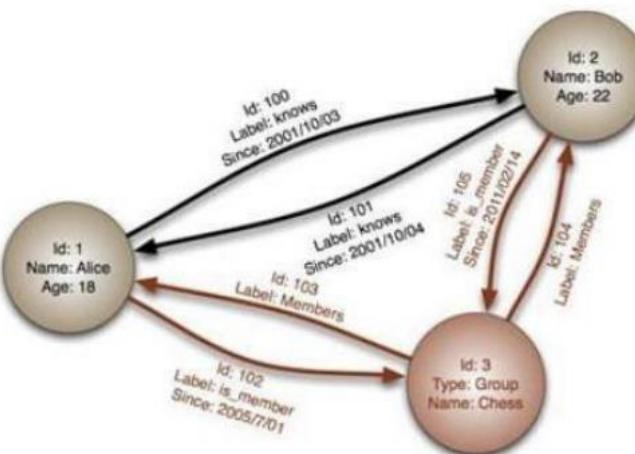
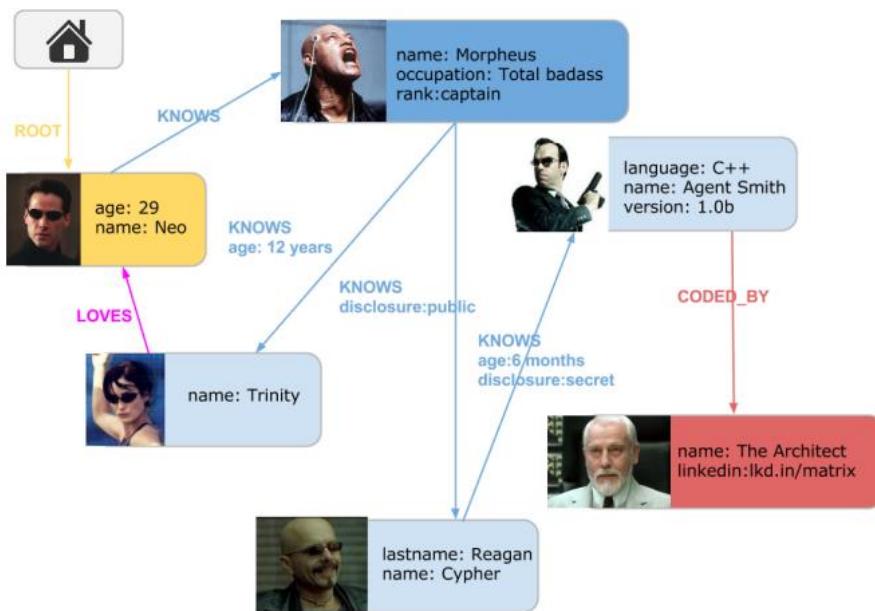
```
{  
    "Great Expectations": "John",  
    "Pride and Prejudice": "Alice",  
    "Wuthering Heights": "Alice"  
}
```

Utilizadas en:

Washington Post, AdRoll,
Scopely, WeatherBug

BASES DE DATOS ORIENTADAS A GRAFOS

La información se representa como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda usar teoría de grafos para recorrer la base de datos ya que esta puede describir atributos de los nodos (entidades) y las aristas (relaciones).



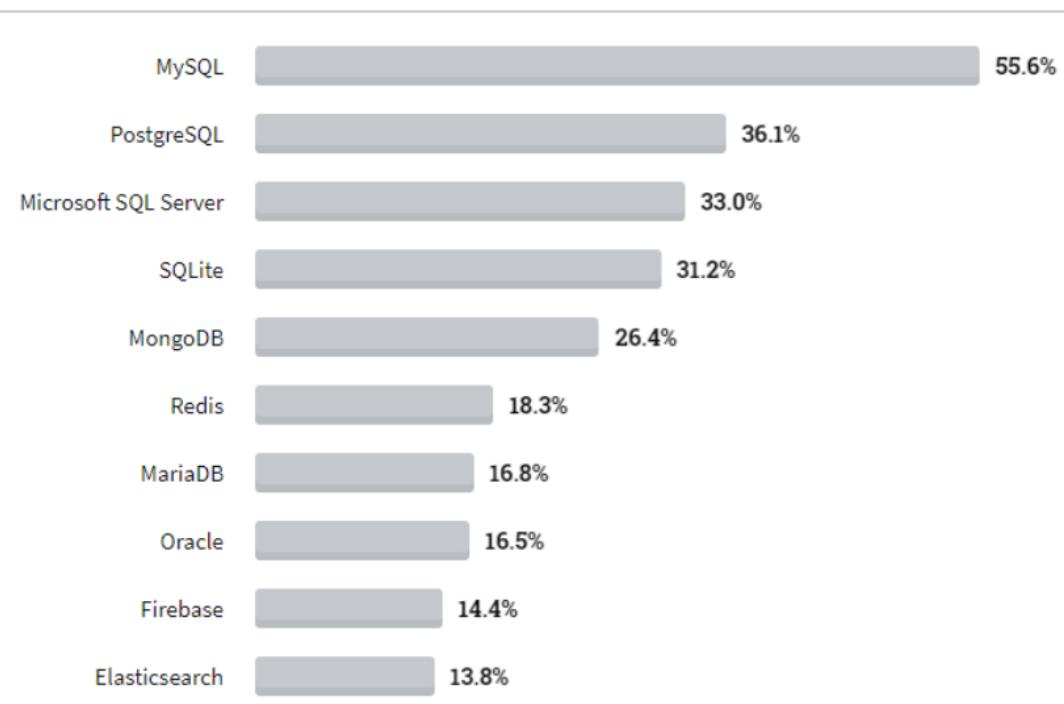
Utilizadas en:
HP, Infojobs, Cisco

MONGODB

- MongoDB (de la palabra en inglés “**humongous**” que significa enorme) es un sistema de base de datos NoSQL **orientado a documentos**
- MongoDB guarda estructuras de datos en documentos tipo **BSON** (*Binary JSON* (JSON Binario))
- Implementada en C++
- Uno de los motores NoSQL más populares
- URL: <http://www.mongodb.org/>

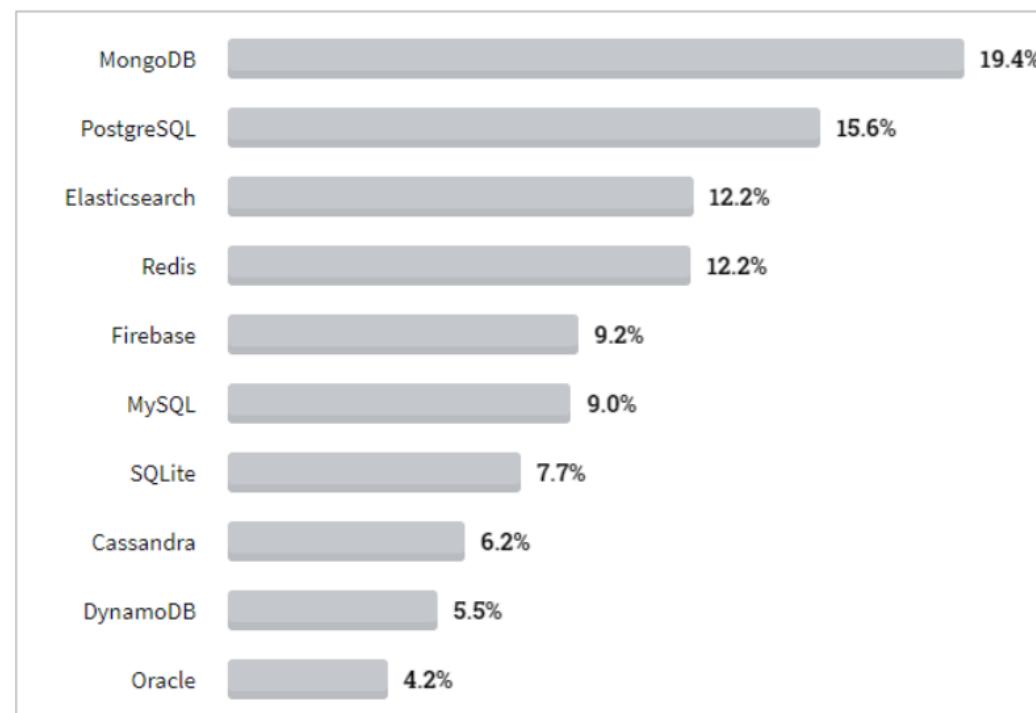


MONGODB



Bases de datos más utilizadas

Fuente: Stack Overflow Developer Survey (2020)



Bases de datos más “deseadas”

QUIÉNES UTILIZAN MONGODB

Google

facebook

CISCO

BuzzFeed

Expedia

eBay

BOSCH

craigslist

Adobe



SAP

GAP

IBM

McAfee

Telefónica

The
New York
Times

SANTOTO
Seccional Tunja

CONCEPTOS BÁSICOS

BD Relacional

Tabla

Fila

Columna

Joins

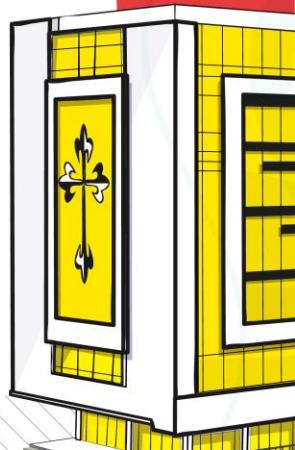
MongoDB

Colección

Documento

Campo

Documentos
embebidos, linking



CONCEPTOS BÁSICOS

Base de datos

- Contenedor físico de colecciones o documentos

Colecciones

- Agrupaciones de documentos, no todas las BD lo tienen

Documentos

- Unidad básica de datos en Json

Valores

- Atómicos
- Listas o arreglos
- Adjuntos (pdf, jpg, etc)

TIPOS DE DATOS

- **String:** Cadenas de caracteres.
- **Integer:** Números enteros.
- **Double:** Números con decimales.
- **Boolean:** Booleanos verdaderos o falsos.
- **Date:** Fechas.
- **Timestamp:** Fecha / hora.
- **Array:** Arreglos de otros tipos de dato.
- **Object:** Otros documentos embebidos.
- **ObjectID:** Identificador de objeto (campo `_id`).
- **Data Binaria:** Punteros a archivos binarios.
- **Javascript:** código y funciones Javascript



EJEMPLO DE DOCUMENTO

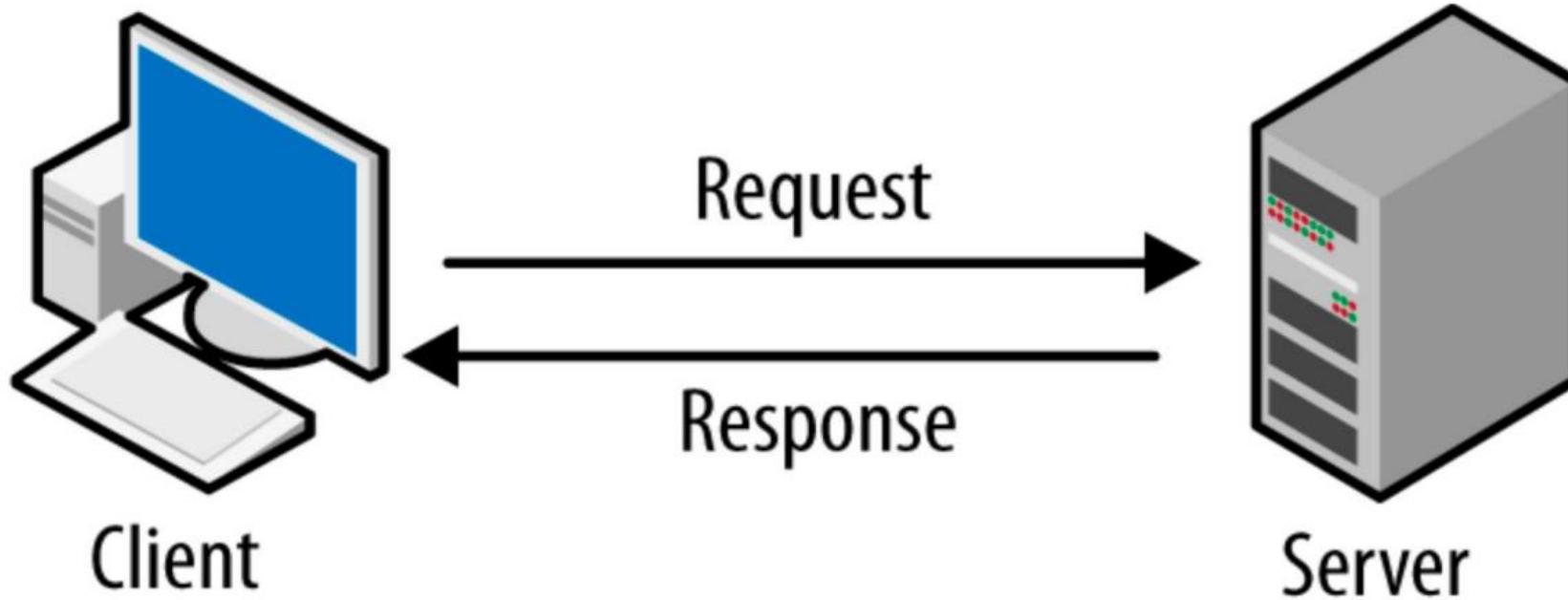
```
{  
    "_id": ObjectId("51c420ba77edcdc3ec709218"),  
    "nombre": "Manuel",  
    "apellidos": "Pérez",  
    "fecha_nacimiento": "1982-03-03",  
    "altura": 1.80,  
    "activo": true,  
    "intereses": ["fútbol", "tenis"],  
    "tarjeta_credito": null,  
    "dni": {  
        "numero": "465464646J",  
        "caducidad": "2021-10-21"  
    }  
}
```

- Un documento en MongoDB no puede ocupar más de 16 MB.
- No es necesario que los documentos tengan la misma estructura.

HERRAMIENTAS QUE UTILIZAREMOS

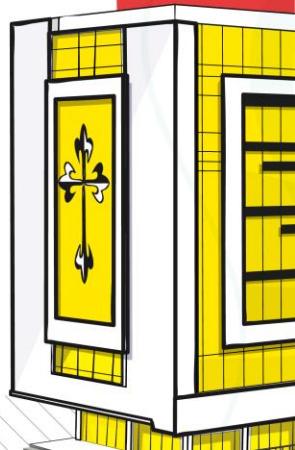


Robo 3T



OPERACIONES CRUD CON MONGODB

CRUD
CREATE
READ
UPDATE
DELETE



CRUD - CREATE

Añadir documentos a una colección

```
db.users.insert ( ← collection
  {
    name: "sue", ← field: value
    age: 26, ← field: value
    status: "A" ← field: value
  }
)
```

db.users.insertOne()
db.users.insertMany()

Si la colección no existe al momento de ejecutar el insert, será creada.

El campo `_id` es único, es el 'primary key' del documento, este se crea automáticamente, aunque puede ser seleccionado manualmente.

```
db.users.insertMany(
  [
    { name: "bob", age: 42, status: "A", },
    { name: "ahn", age: 22, status: "A", },
    { name: "xi", age: 34, status: "D", }
  ]
)
```

CRUD - READ

db.collection.find() : Muestra la colección con los documentos que fueron creados sobre ellos. La colección es mostrada en formato **JSON** (JavaScript Object Notation).

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier



Práctica con MongoDB:

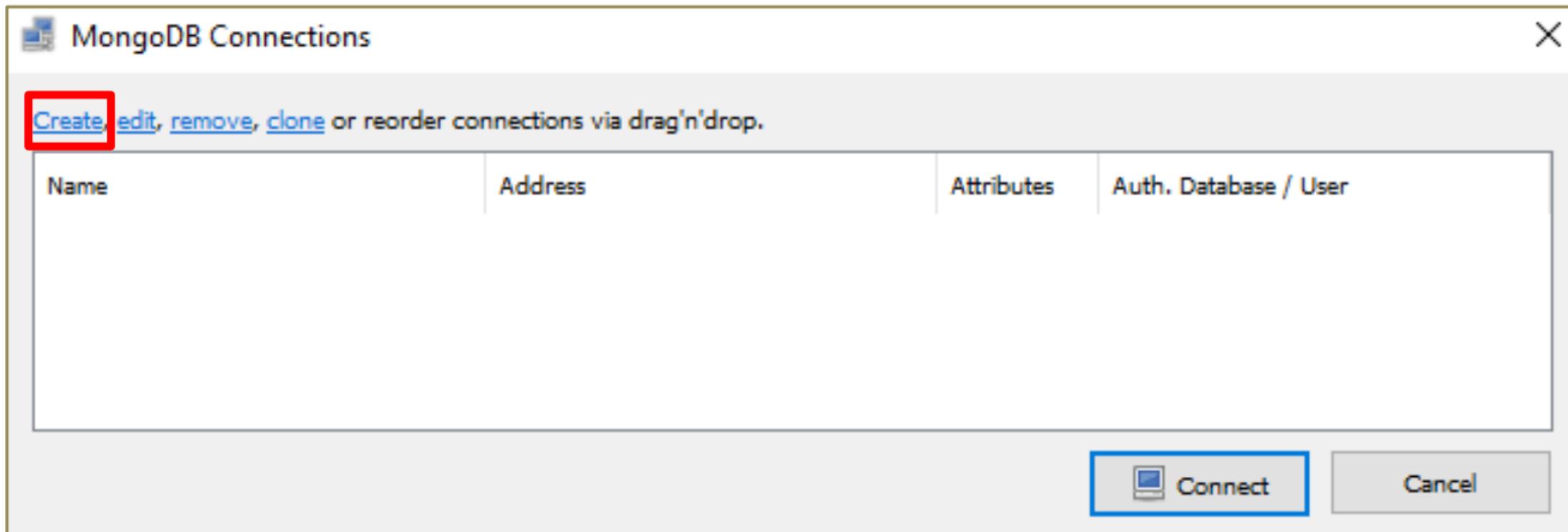
1. Descomprimir el archivo MongoDB.zip
2. La estructura de carpetas quedaría así:

Nombre	Fecha de modificación	Tipo	Tama
mongodb-win32-x86_64-2012plus-4.2.11	8/08/2022 11:50 p. m.	Carpeta de archivos	
robo3t-1.4.2-windows-x86_64-8650949	8/08/2022 11:48 p. m.	Carpeta de archivos	
Iniciar MongoDB.bat	25/11/2020 6:31 a. m.	Archivo por lotes de Windows	
Iniciar Robo3T.bat	25/11/2020 6:34 a. m.	Archivo por lotes de Windows	

3. Iniciar primero mongoDB.bat y luego Robo3T.bat

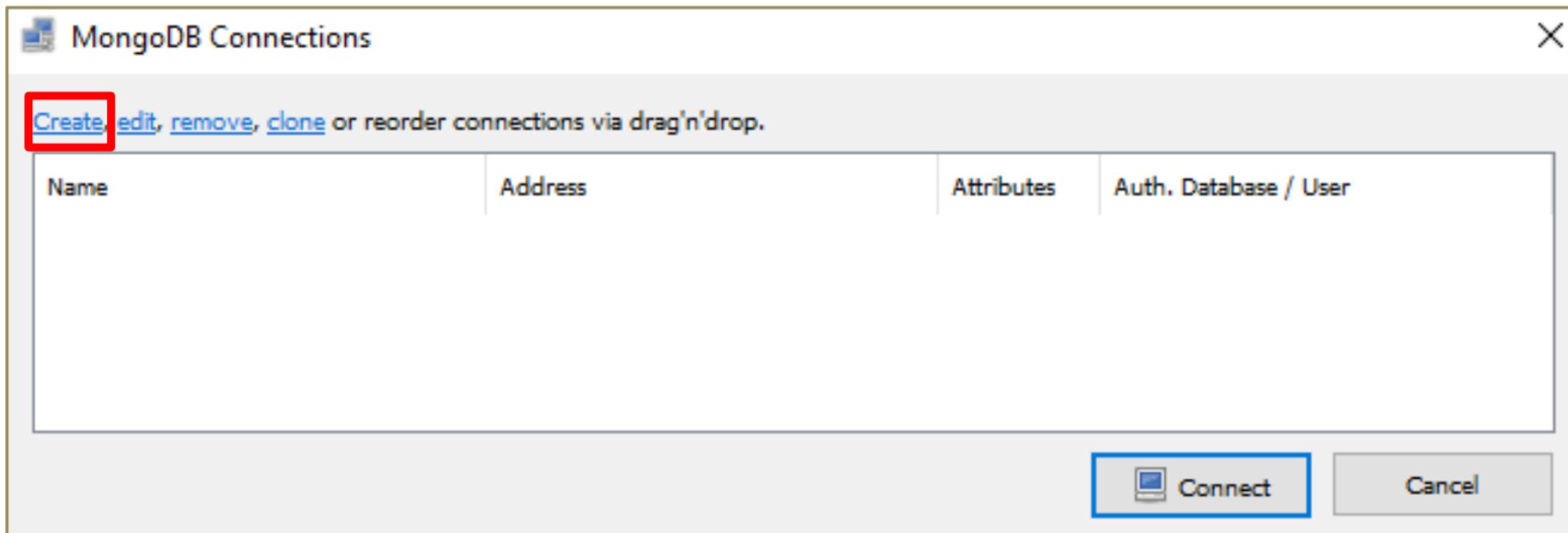
Práctica con MongoDB:

4. Al ejecutar Robo3T.bat aparece una ventana:



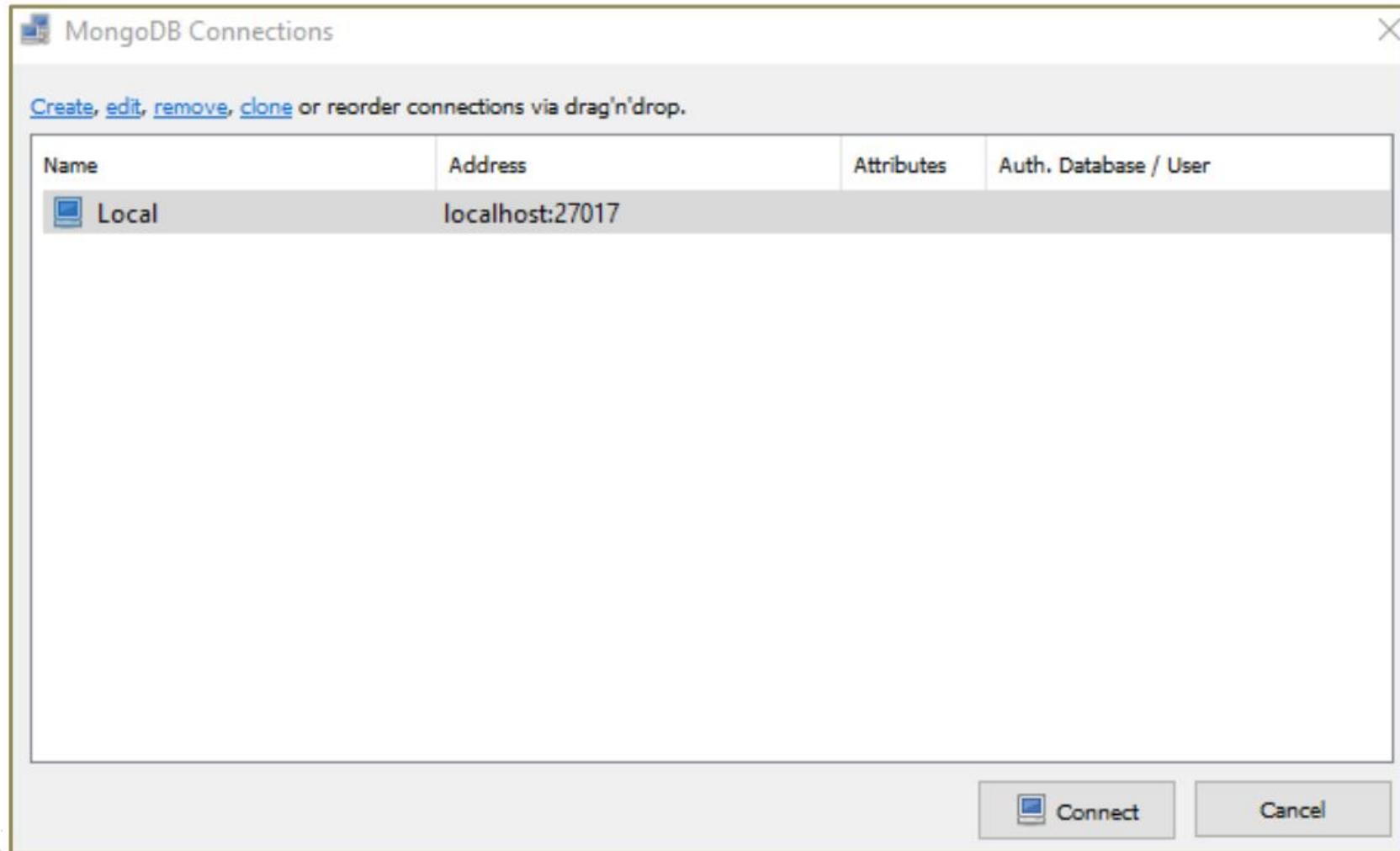
Práctica con MongoDB:

5. Creamos una conexión con el nombre Local:



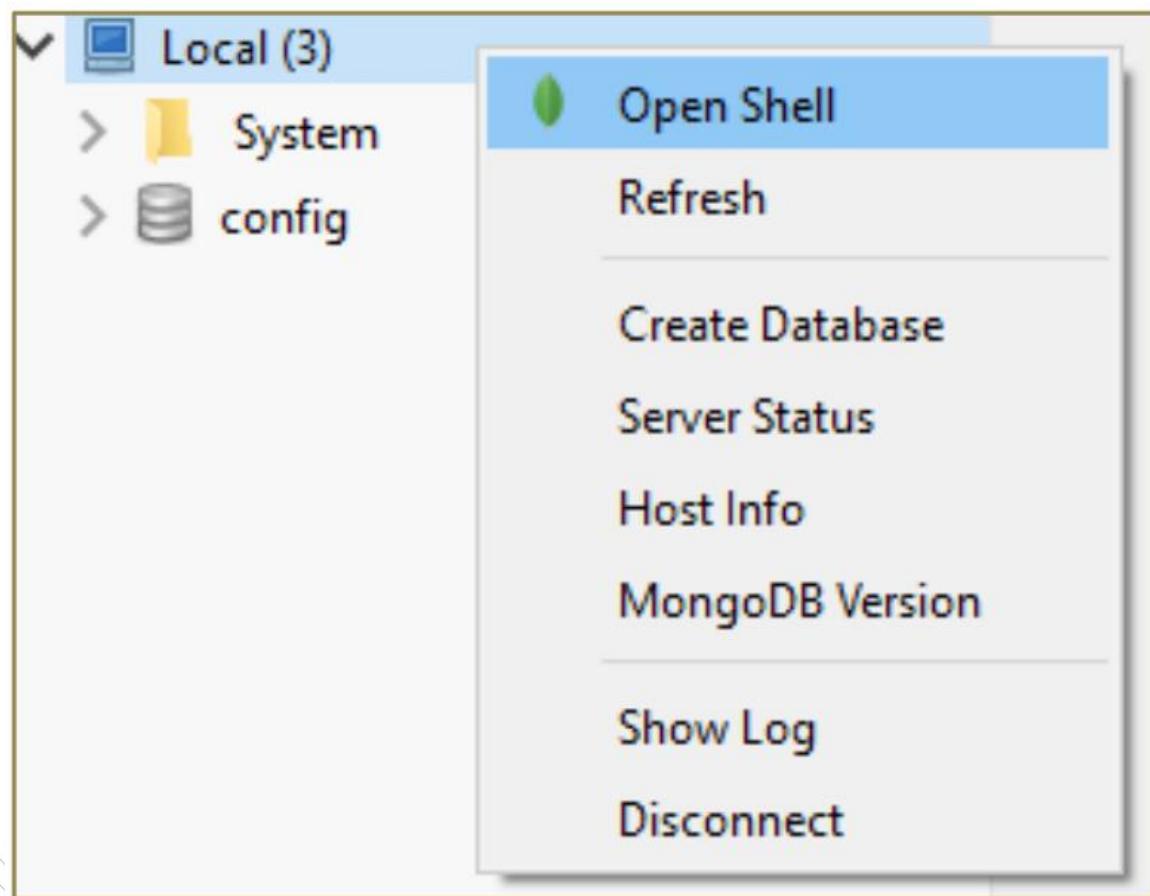
Práctica con MongoDB:

6. Seleccionamos la conexión y presionamos connect:



Práctica con MongoDB:

7. Debería mostrarse una pantalla como la siguiente, en la cual se observa la conexión creada. Para empezar a trabajar presionamos clic derecho sobre nuestra conexión >> Open Shell.



CRUD - READ:

1

```
db.usuarios.insert({ nombre:"Henry",
                     apellido:"Guio", usuario:"hguio", edad:26,
                     ciudad:"Tunja"
                   })
```

2

```
db.usuarios.find()
```

Consulta la BD

3

```
db.usuarios.findOne()
```

Consulta un solo documento

4

```
db.usuarios.find().pretty()
```

Devuelve el contenido mas legible

5

```
db.usuarios.find().count()
```

Cuenta los documentos



CRUD - READ:

6 db.usuarios.count()

Total de documentos en la colección

db.usuarios.insert({nombre:"Ali",apellido:"Kate",usuario:"akate",edad:40,ciudad:"Bogota"})
db.usuarios.insert({nombre:"Alan",apellido:"Brito",usuario:"abrito",edad:39,ciudad:"Bogota"})
db.usuarios.insert({nombre:"Too",apellido:"Mohahito",usuario:"tmohahito",edad:25,ciudad:"Osaka"})
db.usuarios.insert({nombre:"Suvan",apellido:"Strugen-Vahen",usuario:"sstrugen",edad:18,ciudad:"Berlin"})

7 db.usuarios.find({ ciudad: "Bogota" })

Consulta los documentos donde ciudad sea Bogota

8 db.usuarios.find({ ciudad: "Bogota", edad: 39 })

MongoDB aplica por defecto un AND entre cláusulas(es decir, la coma (,) es un operador AND)

9 db.usuarios.find({ \$or: [{ciudad: "Bogota" }, {ciudad: "Osaka"}] })

Para utilizar un operador OR usaremos \$or

OPERADORES

\$gt:

mayor que (o greater than en inglés). (>)

\$lt:

menor que (o lower than en inglés). (<)

\$gte:

mayor o igual que (o greather or equal than en inglés). (>=)

\$lte:

menor o igual que (o lower or equal than en inglés).(<=)

\$not:

no (negación o not en inglés).

\$in:

en, para buscar dentro de un array.

\$nin:

no en, para buscar algo que no se encuentre en un determinado array

\$ne:

no es igual a (o not equal to en inglés). (#)



CRUD - READ:

11 db.usuarios.find({ ciudad:{\$in:["Osaka","Berlin"]}})

Operador in

12 db.usuarios.find({edad:{\$gt:30}})

Usuarios con edad mayor de 30

CRUD - UPDATE

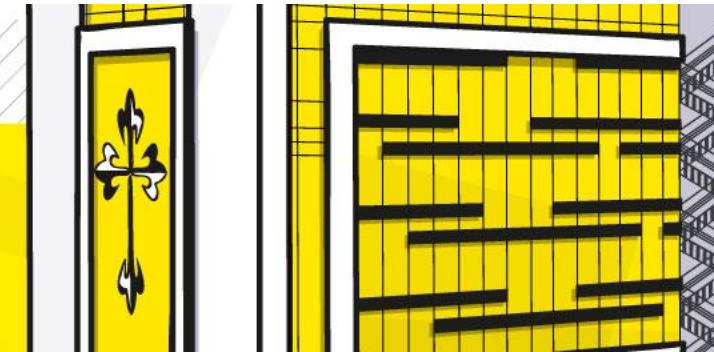
Actualizar un documento

```
db.users.update(  
  { age: { $gt: 18 } },  
  { $set: { status: "A" } },  
  { multi: true })
```

← collection
← update criteria
← update action
← update option

Esta operación modifica un documento existente, apunta a un documento en particular o a muchos que cumplan con las condiciones del criterio. Los criterios pueden ser:

<campo>: <valor> O
<campo>: { <operador>: <valor> }



CRUD - UPDATE

12

```
db.usuarios.update({usuario:"abrito"},{nombre:"Alan",apellido:"Brito",usuario:"abrito",edad:40,ciudad:"Bogota"})
```

\$set: que permite establecer un nuevo valor a un campo del documento. Si el campo no existe actualmente en el documento, será creado automáticamente

13

```
db.usuarios.update({edad:40, ciudad: "Bogota" }, { $set: { peso: 68.9 }})
```

14

```
db.usuarios.find().pretty()
```

15

```
db.usuarios.update({edad:25, ciudad: "Osaka" }, { $set: { edad: 33 }})
```



CRUD - DELETE

Esta operación eliminar el documento de la colección.

```
db.users.remove(  
    { status: "D" } )
```

← collection
← remove criteria

```
db.collection.remove( <query>,  
                      <justOne> )
```

El parámetro **justOne** permite indicar al método remove() si queremos eliminar solamente un documento de todos los coincidentes. En ese caso, se debe pasar el valor true o 1.



CRUD - DELETE

15 db.usuarios.remove({edad: { \$gt: 40} })

Elimina solamente el primer documento que encuentre que cumpla el criterio de consulta

16 db.usuarios.remove({edad: { \$gt: 20} }, true)

Eliminar todos los documentos de una colección si no se especifica una query o si ésta está vacía.
La colección sigue existiendo.

17 db.usuarios.remove({})

Si queremos eliminar todos los documentos de una colección, es recomendable utilizar el método drop(), que elimina la **colección completa**

18 db.usuarios.drop()



CÓDIGO JAVASCRIPT

- ✓ MongoDB posee un intérprete de Javascript que permite la ejecución de instrucciones en dicho lenguaje.
- ✓ Puede ser muy útil para automatizar ciertas tareas.

18

```
//Es posible trabajar con datos primitivos  
a =12  
print(a)
```

19

```
//Tambien podemos trabajar con documentos  
doc={"nombre":"Juan",apellido:"Perez"}  
db.prueba.insert(doc);
```

CÓDIGO JAVASCRIPT

20

```
//Podemos trabajar con bucles  
for(i =0;i <20;i++) {  
    db.prueba.insert({nombre:"nombre"+i,apellido:"apellido"+i});  
}
```

21

```
//Tambien con funciones  
function obtenerDocumento(nomPersona,apePersona) {  
    res ={nombre:nomPersona,apellido :apePersona};  
    return res;  
}  
db.prueba.insert(obtenerDocumento ("Ana", "Tang"));
```



SANTOTO
Seccional Tunja



SC4289-1

www.USTaTUNJa.edu.co

♥ ☰ ☳ @SANTOTOMaSTUNJa ─