OBJETIVO

En esta guía aprenderemos a instalar y configurar el motor de Base NOSQL MongoDB, para que a futuro puedas implementarlo en tus aplicaciones.

NOSQL y SQL

Las bases de datos SQL son bases de datos formadas por un conjunto de tablas que están vinculadas relacionalmente en función de claves comunes a cada una. Tienen un esquema predefinido y rígido (Helland, 2016).

Por otro lado, las bases de datos NoSQL modelan los datos de una manera que excluye las relaciones tabulares proporcionadas en las bases de datos relacionales, lo que permite a los desarrolladores utilizar esquemas de datos más flexibles (IBM, 2019).

Esto significa que las bases de datos NoSQL son mejores con datos semiestructurados y no estructurados y generalmente son más flexibles que las bases de datos SQL. Sin embargo, no ofrecen la misma integridad de datos que imponen las bases de datos SQL (Srinivasa y Hiriyannaiah, 2018).

Las bases de datos SQL tienen su propio lenguaje estándar para consultar datos, SQL (Structured Query Language). SQL es conocido como una excelente opción para consultas complejas y es un estándar internacional que permite a los desarrolladores trabajar solo con un idioma (Harrington, 2002).

Por otro lado, las consultas NoSQL se centran en colecciones de documentos. Como el esquema de la base de datos es diferente para cada base de datos, la sintaxis para consultar datos también es diferente para cada una.

Escalabilidad

Las bases de datos SQL se escalan verticalmente, lo que significa que los desarrolladores tendrán que ampliar la capacidad del servidor para permitir la escalabilidad. De hecho, las bases de datos SQL han sido diseñadas para ejecutarse en un solo servidor para una mejor integridad de los datos, lo que significa que son más difíciles de escalar (Watt, 2014).

Las bases de datos NoSQL, por otro lado, pueden escalar horizontalmente, lo que significa que se pueden agregar más servidores para alimentar la base de datos cuando escala. Esto le da a NoSQL una enorme ventaja en comparación con SQL (IBM, 2019).

El hecho de que las bases de datos NoSQL puedan escalar horizontalmente se debe a la falta de estructura de los datos. Como las bases de datos NoSQL necesitan menos estructura que SQL, cada objeto es independiente y autónomo. Por lo tanto, cada objeto se almacena en servidores diferentes y no necesita estar conectado. Para las bases de datos SQL, este no es el caso, ya que cada columna y fila de la tabla deben estar vinculadas (Srinivasa y Hiriyannaiah, 2018).

MONGODB

MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en

registros, guarda los datos en documentos. Estos documentos son almacenados en formato BSON, que es una representación binaria de JSON.

MongoDB frece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado.

Conceptos del entorno MongoDB

Colección

Una colección es un conjunto de documentos. Podemos verlo como si fuera una tabla en una base de datos relacional.

Documento

Un documento puede ser comparado con una fila o registro de una Base de Datos SQL. En Mongodb un documento es un conjunto de datos estructurados, que contiene pares de clave/valor y se almacena en formato BSON (JSON Binario).

Field

Field es el equivalente a una columna de una tabla de Base de Datos SQL.

Documentos Embebidos

Es la forma de comunicar dos colecciones, equivalente a los SQL joins de una Base de Datos Relacional.

Instalación de MongoDB

MongoDB se puede ejecutar de dos formas:

- Instalando en nuestro computador mediante el instalador oficial
- desde MongoDB Atlas que es un servicio alojado en la nube.

Cualquiera sea la forma que ejecutemos MongoDB necesitaremos instalar adicionalmente una herramienta para administrar nuestros datos. La herramienta oficial es MongoDB compass aunque tambien puedes usar otros como [https://robomongo.org/|Studio 3T]

Instalación via Docker

Para instalar mongodb via docker tenemos que descargar la imagen:

```
docker pull mongodb/mongodb-community-server:latest
```

y ejecutar el comando:

```
docker run --name mongodb -p 27017:27017 -d mongodb/mongodb-community-server:latest
```

Conexión a la Base de Datos MongoDB

Para poder conectarnos a MongoDB necesitamos abrir el programa MongoDB Compass y proporcionar la cadena de conexión.

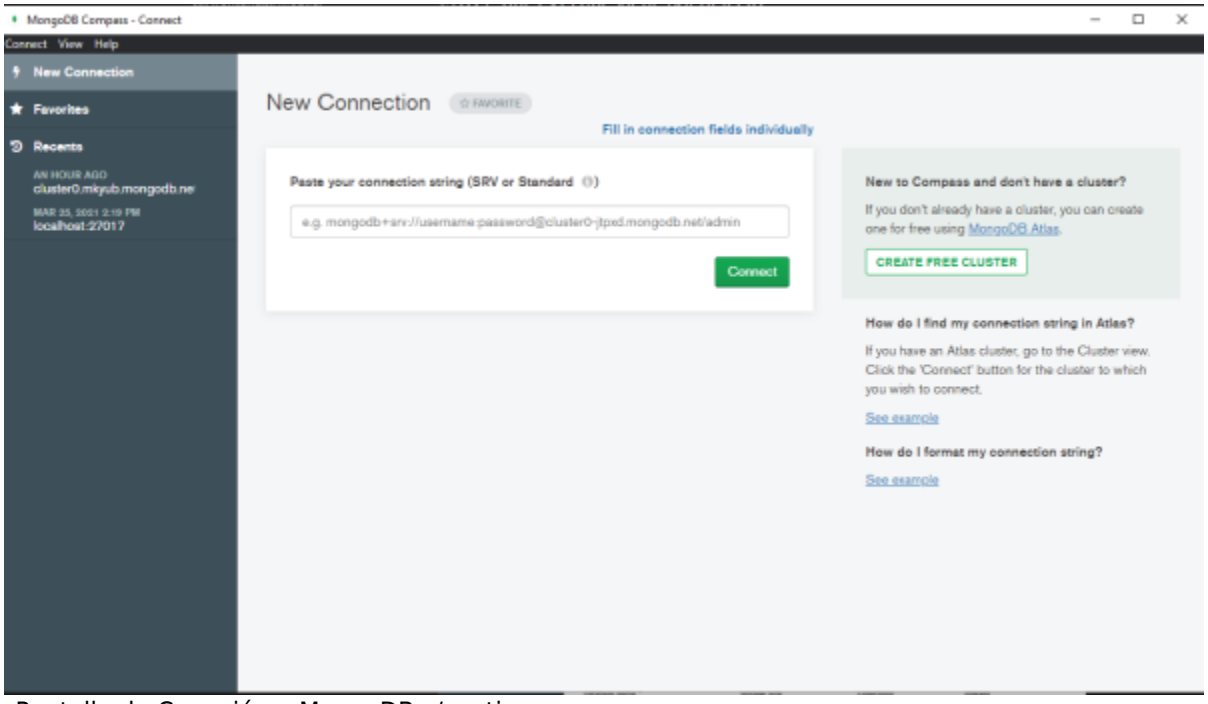


Figure 1: Pantalla de Conexión a MongoDB</caption

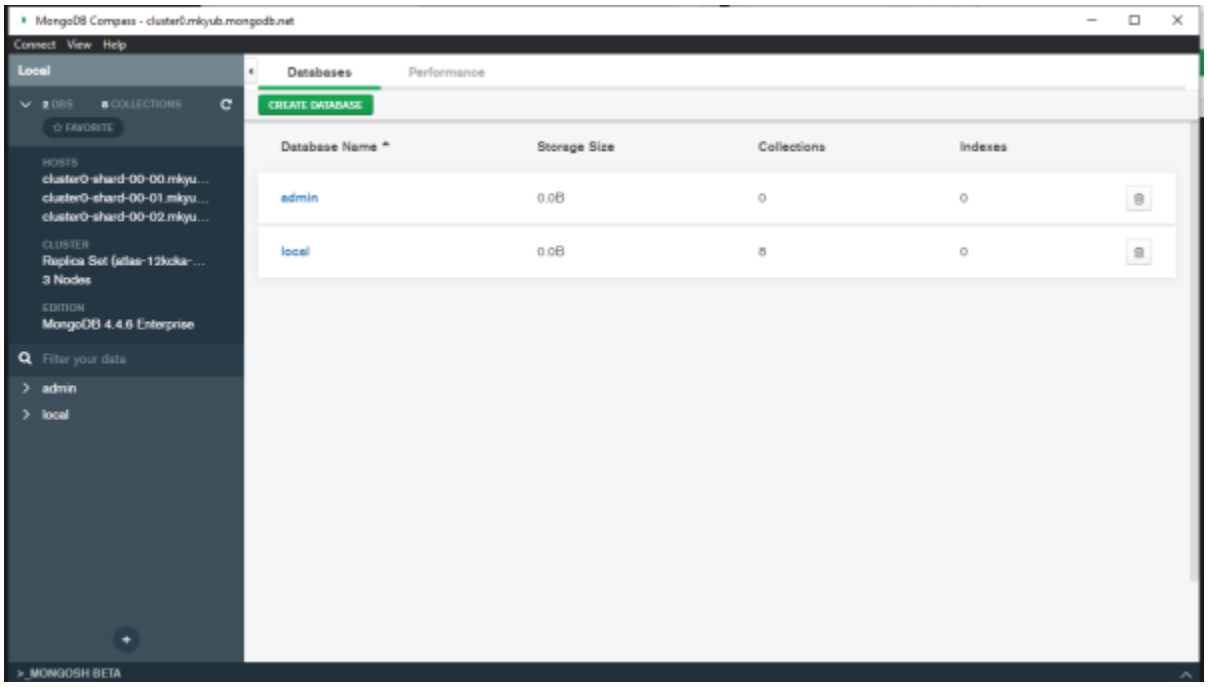
Una cadena de conexión a MongoDB esta compuesto generalmente de:

```
mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]
```

Para una conexión MongoDB Atlas, la cadena de conexión a la Base de Datos será de la siguiente forma:

```
mongodb+srv://teofilo:<password>@cluster0.mkyub.mongodb.net/test
```

Una vez conectado a nuestra Base de datos podremos ver nuestra lista de Bases de datos:



Pantalla Principal de MongoDB Compass
Figure 2:

Operaciones CRUD en MongoDB

Crear Base de Datos

En mongodb no existe un comando para crear una base de datos como tal, sin embargo debemos usar una base de datos y crear un documento dentro de ella.

```
show dbs
use sicrama
db.sicrama.frutas.insertOne({nombre:"naranja",color:"3"})
```

Crear Documentos

.insert() Para poder crear nuevos documentos MongoDB posee el método `.insert()`, `.save()` y en de una manera especial `.update()`. Por ejemplo si deseamos insertar dentro la colección `mcompras` los siguientes datos ejecutamos:

```
var doc1 = {"nrodoc":"16","montototal":1200}
db.mcompras.insert(doc1);
```

.save()

Tiene todas las funciones de .insert(), pero ademas permite actualizar un documento si ya existe el _id de dicho documento, en ese caso .insert() mostraría una excepción.

Operadores de comparación

`$eq`: Igual a (equivalente a = en SQL)

```
db.collection.find({ field: { $eq: value } })
```

Ejemplo:

```
db.users.find({ age: { $eq: 30 } })
```

`$ne`: No igual a (equivalente a <> o != en SQL)

```
db.collection.find({ field: { $ne: value } })
```

Ejemplo:

```
db.users.find({ age: { $ne: 30 } })
```

`$gt`: Mayor que (equivalente a > en SQL)

```
db.collection.find({ field: { $gt: value } })
```

Ejemplo:

```
db.users.find({ age: { $gt: 25 } })
```

`$gte`: Mayor o igual que (equivalente a >= en SQL)

```
db.collection.find({ field: { $gte: value } })
```

Ejemplo:

```
db.users.find({ age: { $gte: 25 } })
```

`$lt`: Menor que (equivalente a < en SQL)

```
db.collection.find({ field: { $lt: value } })
```

Ejemplo:

```
db.users.find({ age: { $lt: 30 } })
```

`$lte`: Menor o igual que (equivalente a ⇐ en SQL)

```
db.collection.find({ field: { $lte: value } })
```

```
db.users.find({ age: { $lte: 30 } })
```

`$in`: En un conjunto de valores (equivalente a IN en SQL)

```
db.collection.find({ field: { $in: [value1, value2, ...] } })
```

Ejemplo:

```
db.users.find({ age: { $in: [25, 30, 35] } })
```

`$nin:` No en un conjunto de valores (equivalente a NOT IN en SQL)

```
db.collection.find({ field: { $nin: [value1, value2, ...] } })
```

Ejemplo:

```
db.users.find({ age: { $nin: [25, 30, 35] } })
```

Listar y Buscar Documentos

.find() Para leer datos MongoDB utiliza dos métodos el primer método es `.find()` que nos permite leer todos los documentos de una colección y `.findOne()` que nos permite leer solo uno.

Suponiendo que tenemos la colección personas con los siguientes datos:

```
[
  {
    "nombres": "Willy Angel",
    "apellidos": "Tapia Saldaña",
    "ci": 1872971,
    "nrocelular": 74660196,
    "direccion": "Villa Montes",
    "nrorecibo": 25651,
    "edad":30
  },
  {
    "nombres": "Omar",
    "apellidos": "Franco Sanchez",
    "ci": 5059008,
    "nrocelular": 67202080,
    "direccion": "Palos Blancos",
    "nrorecibo": 25652,
    "edad":38
  },
  {
    "nombres": "Osvaldo",
    "apellidos": "Farfan Cuellar",
    "ci": 1868486,
    "nrocelular": 70214312,
    "direccion": "Santa Ana",
    "nrorecibo": 25653,
    "edad":29
```

```
  },
  {
    "nombres": "Jose Humberto",
    "apellidos": "Casso Estrada",
    "ci": 7169158,
    "nrocelular": 68693756,
    "direccion": "Tarija",
    "nrorecibo": 25654,
    "edad":30
  },
  {
    "nombres": "Edmundo",
    "apellidos": "Olguin Jaramillo",
    "ci": 1852452,
    "nrocelular": 72969932,
    "direccion": "Tarija",
    "nrorecibo": 25655,
    "edad":20
  }
]
```

La consulta `db.personas.find()` nos retornará todos los documentos de nuestra colección `personas`.

Si queremos obtener todas las personas que viven en Tarija. La consulta find sería de la siguiente forma:

```
db.personas.find({direccion:"Tarija"})
```

Por otro lado si queremos hacer una busqueda por coincidencia equivalente a una consulta SQL `select * from personas where direccion like '%tarija%'` podemos usar el operador:

```
db.personas.find({"direccion":/.tarija./})
```

Todas las personas cuya direccion comienza con *Tar*

```
db.personas.find({"direccion":/^Tar./})
```

```
db.personas.find({},{nombres:1,_id:0})
//select nombre from personas
```

Cadenas que empiezan con `john`

```
db.personas.find({ nombres: { $regex: /^John/, $options: 'i' } })
//select * from personas where nombres ilike 'john%'
```

Cadenas que terminan con `john`

```
db.personas.find({ nombre: { $regex: /John$/, $options: 'i' } })
//select * from personas where nombres ilike '%john'
```

Cadenas que coinciden con `john`

```
db.users.find({ name: { $regex: /John/, $options: 'i' } })
```

```
//select * from personas where nombres ilike '%john%'
```

Seleccionar documentos donde la edad es mayor que 25

```
db.personas.find({ edad: { $gt: 25 } })
```

Seleccionar documentos donde la edad este entre 25 y 30

```
db.personas.find({ age: { $gte: 25, $lte: 30 } })
```

Seleccionar documentos donde el campo `apellido` sea igual a: `copa, perez'`

```
db.personas.find({ apellido: { $all: ["Farfan Cuellar", "Caso Estrada"] } })
```

.findOne()

Devuelve solo el primer documento en orden natural en el disco.

Actualizar

.update() Actualizar es otra de nuestras funciones elementales, con ella podremos modificar información y en el caso de MongoDB se puede realizar hasta inserción de documentos con el método `.update()`, y además se puede utilizar la función `.save()`, en actualizaciones simples. La sintaxis de `.update` es:

```
db.collection.update (
<consulta - criterios>,
<documento_modificado >,
{upsert: true | false, multi : true | false }
)
```

Ejemplo:

```
db.personas.update({_id:ObjectId("60c3aea2cb05fa2c31a02fa8")},{$set:{"direccion":"Comunidad de Palos Blancos"}});
```

Eliminar

MongoDB utiliza la función `.remove()` para eliminar documentos y colecciones, en el caso de las colecciones es mejor utilizar `.drop()`