MEDAC
Instituto Oficial de Formación Profesional

ACCESO A DATOS
TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Operaciones con bases de datos No-SQL

12

ÍNDICE

OBJETIVOS

Entender qué es MongoDb.

Realizar la instalación de una base de datos NoSql.

Aprender a usar la base de datos MongoDb.

Estudiar el significado y usabilidad de operaciones CRUD.

/ 1. Introducción y contextualización práctica

En el siguiente capítulo nos adentraremos de lleno en el mundo de una base de datos NoSQL. Para ello aprenderemos qué es, y para qué se usa nuestra base de datos MongoDB.

Realizaremos una guía de instalación de la propia base de datos siguiendo los diferentes pasos.

Una vez instalada nuestra base de datos aprenderemos y practicaremos las diferentes operaciones, prestando especial atención a las operaciones CRUD.

Planteamiento del caso práctico inicial

A continuación, vamos a plantear un caso práctico a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y Resolución del caso práctico.

Fig. 1. Base de datos mongoDB

Audio intro. "Top 5 Operaciones MongoDB"

https://bit.ly/38Rs8jR

/ 2. Base de datos MongoDB

Para realizar un primer acercamiento a esta base de datos, podríamos destacar las siguientes **características**:

- Es una base de datos NoSQL **orientada a documentos**.
- La estructura de los documentos es en **formato JSON**. Internamente los documentos son almacenados en formato BSON.
- Al tener documentos más ricos, se reduce el I/O (*input/output*) sobre la base de datos. No existe la necesidad de hacer ***Joins***.
- Permite operaciones **CRUD** (*Create, Read, Update, Delete*) con una sintaxis parecida a JavaScript.
- Proporciona replicación y alta disponibilidad a través de ***Replica Sets***.
- También dispone de balanceo y escalado horizontal usando **Sharding**. El balanceo de los datos se realiza automáticamente.

Audio 1. "Top 5 Sharding"
https://bit.ly/3kH52yP

- Ofrece un mecanismo de procesamiento masivo de datos a través de **operaciones de agregación**.
- El grueso de los datos reside en **memoria**, por lo que las lecturas y escrituras son muy rápidas.
- Permite también crear colecciones de tipo circular de tamaño fijo, y mantiene el orden según se han ido insertando los datos (***Capped Collections***).
- Tienen múltiples motores de almacenamiento diferentes, ***nmap, WiredTiger, In-memory y Encrypted***.
- Es enorme, forma parte de ***Big Data***.

Fig. 2. Logo MongoDb

2.1. MongoDB el teorema de CAP

El teorema de CAP o también de *Brewer*, sostiene, que en sistemas distribuidos **no** es posible **garantizar al completo la consistencia, la disponibilidad y tolerancia**.

- **Consistencia**: La información que obtenemos a la hora de realizar una consulta debe ser siempre la misma.
- **Disponibilidad**: Todos los clientes deben de poder realizar las operaciones de lectura y escritura, aunque un nodo se haya caído.
- **Tolerancia a particiones**: Los sistemas, como vimos en la unidad anterior, pueden estar divididos en particiones distribuidas en distintos puntos. Esta propiedad consiste en, a pesar de esta división, asegurar el funcionamiento.

Según el teorema de CAP, podemos clasificar todas las bases de datos según las características comentadas anteriormente. **No todas cumplen los mismos puntos del teorema**.

- **Las bases de datos más cercanas al vértice AP**: aseguran la disponibilidad y tolerancia a particiones, pero no la consistencia, al menos en su totalidad. Algunas de ellas a través de la replicación y verificación, sí consiguen parte de consistencia.
- **Aquellas que su vértice este más cercano a CP**: estarán del lado de la consistencia y tolerancia a particiones. Sacrifican la disponibilidad al replicar los datos a través de nodos.
- Por último, **las bases de datos más cercanas al vértice CA**: poseerán más consistencia y disponibilidad dejando un poco de lado la tolerancia a particiones. Este problema se solucionará replicando los datos.

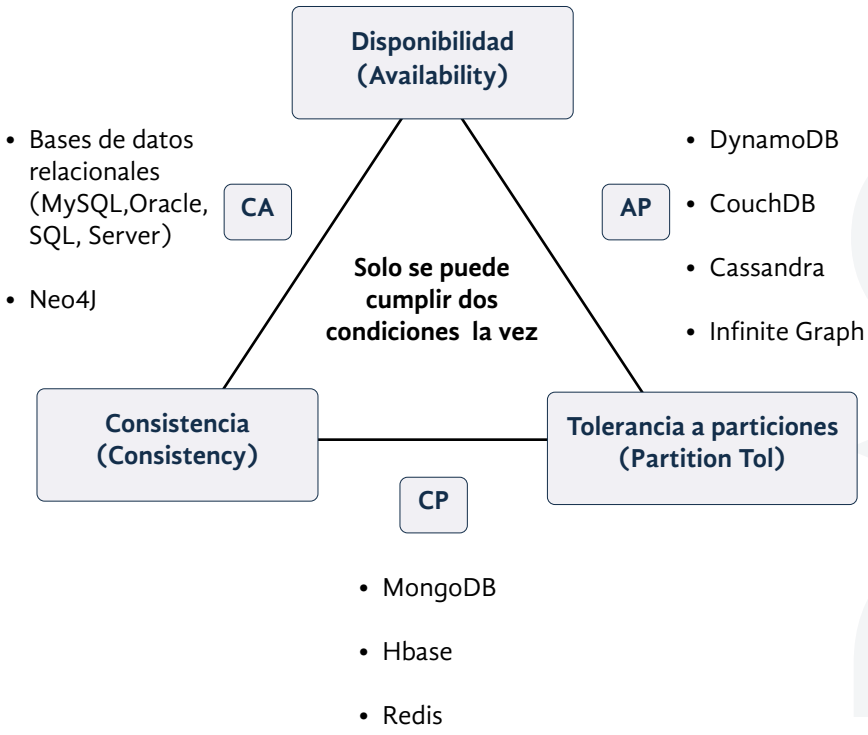

Fig. 3. Teorema de CAP. Fuente

Según el teorema CAP podemos encontrar a MongoDB en el vértice CP.

2.2. Casos de uso y documentos

a. Casos de uso y términos

MongoDB es un producto de propósito general y es muy útil para múltiples casos de uso tales como:

- CMS, Aplicaciones móviles.
- *Gaming.*
- *E-commerce.*
- *Bussiness inteligence.*

- *Analytics*.
- Proyectos *Big Data*.
- Web Caché.

A continuación, vamos a exponer algunos términos utilizados en MongoDB y su correspondencia en el mundo relacional

Relacional	MongoDB
Base de datos	Base de datos
Tabla	Colección
Fila	Documento
Índice	Índice
Insert	*Insert*
Select	*Find*
Update	*Update*
Delete	*Remove*

Tabla 1. Comparativa relacional – MongoDB

Video 1. "Comparativa relacional y mongo"
https://bit.ly/32Uluoa

b. Documentos

Veamos algunas características relacionadas con el concepto de documento:

- Cada entrada de una colección es un documento.
- Son estructuras de datos compuestas por campos de clave/valor.
- Los documentos tienen una estructura similar a objetos JSON.
- Los documentos se corresponden con tipos de datos nativos en los lenguajes de programación.
- En un documento es posible embeber otros documentos o *arrays*.
- Se tiene un esquema dinámico que permite polimorfismo de manera fluida.
- Las operaciones de escritura son solo atómicas a nivel de documento.

A continuación, mostraremos un ejemplo:

```
{
“_id” : ObjectId(“5457a502e308f720d8999e97”),
“Nombre”: “Francisco”,
“Apellidos” : “Fernandez Rioja”,
“Edad” : 30,
“Aficiones”: “{ “Comics” : null,
            “Deportes”: [“squash”, “natación”]
           },
“Empresa” : “XXXSA”,
“Cargo” : “MongoDB DBA”,
“Tecnologías” : [“Openstack” , “Openshift”,”MongoDB”],
“Proyectos” : “{ “Openstack”: [“Cliente1”,”Cliente2”],
             “Openshift”: [“Cliente4”]
            }
}
```

Código 1. Ejemplo documento Mongo

/ 3. Caso práctico 1: “Instalación MongoDB”

Planteamiento: Se quiere enlazar una aplicación web a una base de datos MongoDB, para ello antes deberemos instalar la base de datos. Se requiere instalar en un servidor Linux.

Nudo: En la web de MongoDB encontraremos los enlaces de descarga en diferentes formatos. Seguiremos las instrucciones que nos indica en sus tutoriales oficiales:

https://docs.mongodb.com/guides/server/install/

Desenlace:

1. En primer lugar, nos descargaremos los ficheros binarios desde MongoDB Download Center.
2. Extraeremos los ficheros descargados con:

```
tar -xvzf <tgz file>
```

3. Copiaremos la carpeta que acabamos de extraer a la localización en la que hayamos elegido ejecutar MongoDB con el comando “cp”
4. Cargaremos en la variable de entorno PATH nuestra instalación. Los ficheros binarios en MongoDB se encuentran en la carpeta *bin*. Añadiremos a nuestro fichero. *bashrc* la siguiente línea:

```
export PATH=<mongodb-install-directory>/bin:$PATH
```

5. Antes de arrancar MongoDB podemos crear un directorio donde el proceso "*mongod*" escribirá los datos. Por defecto, el proceso usa la ruta */data/db*. Si creamos otro directorio distinto deberemos de especificarlo en el arranque. Para crear el directorio por defecto:

```
mkdir -p /data/db
```

6. Antes de ejecutar el proceso habría que asegurarse que el usuario con el que se va a ejecutar "mongod" tiene los permisos de escritura en el directorio.

7. Para arrancar la base de datos MongoDB con la ruta por defecto mencionada anteriormente, habría que ejecutar el proceso "mongod":

```
mongod
```

Si no se usa el directorio por defecto, habría que aplicar:

```
mongod –dbpath <directorio>
```

8. Finalmente, habría que verificar que MongoDB ha arrancado correctamente, comprobando que se ha mostrado la siguiente línea:

```
[initandlisten] waiting for connections on port 27017
```

/ 4. Ficheros binarios de MongoDB

El paquete de MongoDB contiene algunos ficheros binarios. Se usan para arrancar el servidor de la base de datos y para acceder al *shell* de la misma. Algunos de ellos son:

- **mongod**: Es el servicio principal de MondoDB. Maneja los accesos a los datos, las peticiones de datos, y ejecuta tareas de mantenimiento en *background*. Su fichero de configuración es "mongod.conf".
- **mongo**: Es la *shell* interactiva de MongoDb. Aporta un entorno funcional completo para ser usado con la base de datos.
- mongos: Es un servicio propio del modo de despliegue *Shard*. Su función es la de enrutar las peticiones de la capa de aplicación y determinar la ubicación de los datos en los diferentes *shards* del despliegue.
- **mongodump**: Es una utilidad para crear un *export* binario del contenido de una base de datos. Podemos considerar MongoDB como una herramienta más para realizar copias de seguridad. Podremos usar esta herramienta contra "mongod" o "mongos" teniendo en cuenta que "mongod" podrá estar arrancado o parado indistintamente.
- **mongorestore**: En conjunción con *mongodump*, se utiliza para restaurar los respaldos realizados con *mongodump*.
- **mongooplog**: Es una herramienta que permite hacer "*polling*" del *oplog* de un servidor remoto, y aplicarlo sobre el servidor local. Esta utilidad la podemos usar para realizar cierta clase de migraciones en tiempo real, donde se requiere que el servidor fuente se mantenga Online y en funcionamiento

Fig. 4. Herramientas MongoDB

/ 5. Herramientas

a. Herramientas exportación / importación

A continuación, vamos a ver las herramientas que disponemos para la exportación e importación de datos:

- **bsondump**: Convierte ficheros BSON a algún formato legible por humanos, incluido a JSON. Se trata de una herramienta de análisis, en ningún caso debe ser utilizada para otro tipo de actividades.
- **mongoexport**: Utilidad que permite exportar los datos de una instancia de MongoDB en formato JSON o CSV. En conjunción con *mongoimport* son útiles para hacer *backup* de una parte bien definida de los datos de la BBDD MongoDB o para casos concretos de inserción de datos.
- **mongoimport**: Utilidad que permite importar los datos de una instancia de MongoDB desde ficheros con formato JSON o CSV.
- **mongofiles**: Utilidad que permite manejar ficheros en una instancia de MongoDB con objetos *GridFS*, desde la línea de comandos. En *Replica* Set sólo podrá leer desde el primario.

b. Herramientas análisis

También disponemos de algunas herramientas para el análisis de *performace* y actividad:

- **mongoperf**: Utilidad para comprobar el *performance* I/O de forma independiente a MongoDB.
- **mongostat**: Utilidad que proporciona una rápida visión del estado actual de los servicios *mongod* y mongos. Es similar a la utilidad *vmstat*.
- **mongotop**: Proporciona un método para trazar el tiempo que una instancia de MongoDB emplea en las operaciones de Lectura/ Escritura de datos. Proporciona estadísticas a nivel de colección, por defecto, cada segundo.

Fig. 5. Análisis y herramientas MongoDB

/ 6. Shell de MongoDB

Para interactuar con esta base de datos utilizamos la *shell* mongo, que básicamente:

- Es una *shell* interactiva en JavaScript.
- Permite ejecutar scripts escritos en JavaScript para manipulación de datos, ejecución de comandos en la base de datos, aplicación de índices, etc.
- La *shell* puede ser utilizada tanto para la visualización de datos, como para la administración de la base de datos, sea *Standalone*, *Replica Set o Sharding*.

Para trabajar con la *shell*, en primer lugar, nos conectamos a la instancia que hemos levantado anteriormente ejecutando el comando **mongo**.

Una vez obtenemos el *prompt*, ejecutaremos algunos comandos de ejemplo:

```
Show dbs
```

Código. 2. Show DB

Con este comando podremos ver las diferentes bases de datos que tenemos.

```
use miBD
```

Código. 3. Usar bd

Mediante use *miDB*, estaremos creando una base de datos y posicionándonos en ella.

```
db.createCollection("holaMundo")
```

Código. 4. Crear colección.

Con *createCollection*() creamos una colección nueva, a continuación, vamos a insertar datos.

```
db.holaMundo.insert({"Nombre" : "Ernesto", "Apellido" : "Perez", "Edad" : "45"})
```

Código. 5. Creando registro

De esta forma realizaremos inserciones en nuestra nueva colección creada previamente. Para ver todos los registros de nuestra colección:

```
db.holaMundo.find()
```

Código. 6. Viendo registros de una colección

6.1. Comandos

En la *shell* de Mongo podremos encontrar una serie de comandos que nos facilitarán la realización de alguna tarea o mostrar información sobre la base de datos. Vamos a revisar algunos de los más destacados.

Helper	**Descripción**
help	Muestra ayuda general.
db.help()	Muestra ayuda sobre los comandos de ejecutables sobre BBDD.
db.<collection>.help()	Muestra ayuda sobre comandos de ejecutables sobre colecciones.
Show dbs	Muestra las BBDD del servidor.
db	Devuelve el nombre de la BBDD donde nos encontramos posicionados.
show collections	Muestra las colecciones contenidas en la BBDD donde estamos posicionados.
use <db>	Nos posicionamos sobre la BBDD db.
show users	Muestra los usuarios sobre la BBDD actual.
load("<ruta_script>")	Carga en la sesión actual el script contenido en la ruta ruta_script.
it	Itera el cursor sobre el que se haya hecho una query.

Tabla 2. Algunos comandos útiles de la shell

Mira el ejemplo en este video:

Video 2. "Operaciones en el Shell"
https://bit.ly/3nzeExl

A parte de la *shell* de mongoDB tenemos otra opción para poder administrar la información de nuestra base de datos mongo. Existen varias herramientas gráficas que también nos permiten acceder a nuestra BBDD y administrarla de manera más intuitiva. Una de las más conocidas es **Robomongo**.

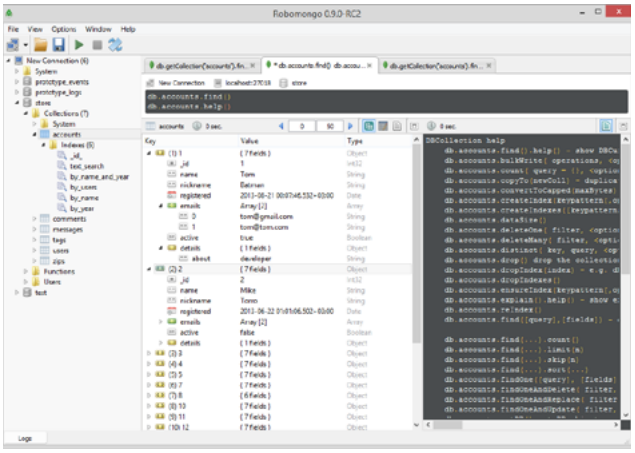

Fig. 6. Imagen RoboMongo

/ 7. Operaciones CRUD

Las operaciones CRUD no son más que las habituales operaciones de **Insertar, leer, actualizar y borrar**. De ahí sus siglas (*Create/Read/Update/Delete*).

A continuación, mostraremos una tabla con las distintas operaciones en las diferentes bases de datos y su equivalencia:

Operación	Mongo	SQL
Create	*Insert / save*	*INSERT*
Read	*Find / finOne*	*SELECT*
Update	*Update / findAndModify / save*	*UPDATE*
Delete	*Remove / drop*	*DELETE*

Tabla 3. Operaciones CRU.D

a. Inserción de datos

Fig. 7. Inserción de datos mongo

Todo documento insertado en MongoDB tiene un campo _id, que identifica unívocamente el documento. Genera valor por defecto.

b. **Consulta de datos**

Fig. 8. Consulta de datos

En la imagen superior tenemos un ejemplo claro de cómo se consultarían datos en nuestra base de datos Mongo.

c. **Actualizar información**

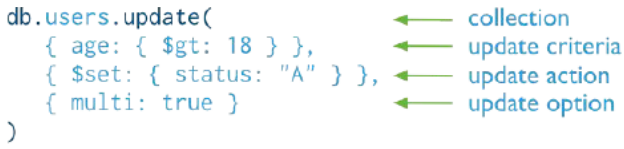

Fig. 9. Actualización de datos

En la imagen, podemos observar cómo se realizaría el Update.

d. **Borrado de documentos**

Fig. 10. Borrado de documentos

/ 8. Caso práctico 2: "Insertar y borrar"

Planteamiento: Se requiere insertar 2 documentos cuyo equipo sea:

- RealMadrid, blanca, Madrid.
- FCBarcelona, azulgrana, Barcelona.

La inserción se debe realizar en la colección "**equipos**" que tiene la siguiente estructura:

```
({"Nombre" : "X", "Camiseta" : "X", "Ciudad" : "X"})
```

Nudo: Una vez estudiadas las sentencias CRUD básicas para el manejo de datos realizaremos la inserción de datos con el comando ***insert.***

Desenlace: Suponiendo que estamos ya situados en la base de datos adecuada, deberemos recordar y aplicar, cual es la sentencia para agregar datos, en este caso ***insert***.

Para ello, en primer lugar, pondremos la palabra clave "db", a continuación, la colección donde vamos a insertar los datos, justo después la sentencia que en este caso es ***insert***, y, por último, agregaremos los datos.

Las dos inserciones quedarían de la siguiente forma:

```
db.equipos.insert({“Nombre” : “RealMadrid”, “Camiseta” : “blanca”, “Ciudad” : “Madrid”})
```

Código. 7 Insert 1

```
db.equipos.insert({“Nombre” : “FCBarcelona”, “Camiseta” : “azulgrana”, “Ciudad” : “Barcelona”})
```

Código. 8 Insert 2

/ 9. Resumen y resolución del caso práctico de la unidad

Una vez estudiado el tema, podemos asegurar que conocemos el **significado y principales características** de la base de datos **MongoDB**.

Hemos aprendido a **instalar** con el sistema operativo Linux, ésta base de datos por medio del paquete que contiene los ficheros binarios.

Hemos hecho uso también, de la ***shell* de MongoDB**, ejecutando los **comandos** más comunes, que deberemos tener presentes a la hora de manejar los datos.

Finalmente, hemos visto la diferencia y equivalencias de las **sentencias CRUD** en bases de datos relacionales, y en MongoDB.

Resolución del caso práctico inicial

Para llevar a cabo la implementación de las tareas que se nos plantean en el caso inicial, una vez levantado nuestro servicio *Mongod*, ejecutando el comando “mongo” sobre el *shell* de nuestra base de datos, realizaremos las siguientes operaciones:

1. **Use MiEmpresa**: cuando ejecutemos este comando, estaremos creando el almacén de datos y a su vez estaremos entrando en él. Podremos usar el comando **show dbs**, para comprobar que nuestra base de datos aparece entre las creadas.

2. **Diseñaremos la colección Empleados** de la siguiente forma:

   ```
   ({“Nombre”: “XXX”, “Edad” : “xxx”, “Antiguedad” : “xxx”, “Especialidad” : “xxx”})
   ```

   Una vez tenemos claro el modelo, el próximo paso será realizar la creación de la colección **Empleados**:

   ```
   db.createCollection(“Empleados”)
   ```

3. Por último, se requiere realizar una consulta en nuestra base de datos imaginando que hubiéramos insertado ya una cantidad de registros en dicha colección, nos dispondríamos a realizar la consulta con la palabra clave ***find***.

   ```
   db.Empleados.find({Edad: {$gt : 40}})
   ```

Con el comando ***find*** podremos realizar distintas consultas en la base de datos Mongo y directamente en nuestras colecciones.

Para mostrar aquellos empleados cuya edad sea superior a 40 años, podemos ver que se usa "$gt". En este caso hemos usado el operador ***Greater Than*** que nos mostrará los resultados mayores que la cantidad que indiquemos, en este caso 40.

/ 10. Bibliografía

10.1. Webgrafía

https://docs.mongodb.com/
https://www.paradigmadigital.com/