ACCESO A DATOS

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

MongoDB
Instalación
Operaciones

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**. BSON son los binarios de JSON.

• 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**.

• 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 o Colecciones limitadas).

• Tienen **múltiples motores de almacenamiento** diferentes, importante

- **nmap**,
- **WiredTiger**,
- **In-memory**
- **Encrypted**.

• Es **enorme**, forma parte de **Big Data**.

Sharding (fragmentación)

El sharding es una expresión que en su propia traducción nos hace referencia a un **particionado de disco**. Concretamente significa **fragmentación**. Es una **técnica** que utilizan bases de datos como Mongo DB para **gestionar automáticamente** la **carga** en sus servidores. Van distribuyendo los datos en diferentes "**shards**" o fragmentos, es decir, conjuntos de servidores que **guardan los diferentes datos**.

MongoDB el teorema de CAP (importante)

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**. Debe de dar siempre el mismo resultado.
- 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 (Availability Partition tolerance):
  - ➔ 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.
    - **Dynamo DB**, **Couch DB**, **Cassandra**, **Infinite Graph**.
- Aquellas que su vértice este más cercano a CP (Consistency Partition tolerance):
  - ➔ estarán del lado de la **consistencia y tolerancia** a particiones.
  - ➔ **Sacrifican la disponibilidad** al replicar los datos a través de nodos.
    - **MongoDB, Hbase**, **Redis**.
- Por último, las bases de datos más cercanas al vértice CA (Consistency Availability):
  - ➔ poseerán más **consistencia y disponibilidad**
  - ➔ **dejando** un poco de lado la **tolerancia a particiones**.
    - Este problema se solucionará **replicando** los datos.
    - **BBDD relacionales** (**MySQL, Oracle, SQL Server**), **Neo 4J**.

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

Casos de uso y términos

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

- **CMS** (Content Management System), **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:

Comparativa relacional – MongoDB

Relacional → MongoDB

Base de datos → Base de datos

Tabla → Colección

Fila → Documento

Índice → Índice

Insert → Insert

Select → Find

Update → Update

Delete → Remove

Comparativa relacional y mongo

Estamos haciendo una introducción básica de la base de datos MongoDB. En la tabla anterior comparamos las bases de datos relacionales y bases de datos orientadas a objetos que trabajan con documentos.

Hemos visto que este tipo de base de datos, como es MongoDB, casi siempre (o, digamos, a nivel comercial), vamos a ver su objetivo en salidas como:

- gaming,
- aplicaciones de e-commerce,
- algunas de analíticas,
- de business intelligence,
- aplicaciones que están conectadas a nivel de compras con business intelligence como hemos dicho
- y, sobre todo proyectos de Big Data; aplicaciones que muevan ficheros de información muy grandes y que tengan la facilidad de almacenarse bien con este sistema.

Pero algo que venimos a ver aquí en la tabla es que, si vemos la diferencia entre las bases de datos relacionales y MongoDB, pues todo parte de un origen y realmente hay diferencias, pero con los conocimientos previos que solemos tener de las bases de datos relacionales, no nos va a costar mucho adaptarnos a MongoDB, por ejemplo. Ya veréis.

Vamos a ver a continuación que, cuando en una **tabla** de una base de datos relacional (o sea, cuando en una base de datos relacional tenemos una tabla), aquí tendremos su símil, sería una **colección**. Porque en MongoDB tenemos una base de datos y dentro de esa base de datos tenemos diferentes colecciones. Dentro de la colección tenemos **documentos** y, en la base de datos relacional, tendremos **filas**.

El resto de cosas, como podemos ver, índices seguimos teniendo igual. Los inserts se hacen muy parecidos. En la base de datos relacional, tenemos el **select** y aquí tenemos el **find**, con algunas especificaciones que veremos más adelante. Update, tenemos update, y **delete**, en vez de delete, tenemos **remove**.

Es una forma muy fácil de aprender en MongoDB porque, digamos, que los conceptos de la base de datos relacional ya los tenemos, entonces adaptarnos un poco a lo que nos pide esta base de datos orientada a objetos u **orientada a colecciones**, en este caso.

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:

Ejemplo documento Mongo

```
{
        "id": ObjectId("5457a502e308f720d8999e97"),
        "Nombre": "Francisco",
        "Apellidos": "Fernandez Rioja",
        "Edad": 30,
        "Aficiones": "{ "Comics" : null, "Deportes": ["squash","natacion"]
                        },
                        "Empresa": "XXXSA",
                        "Cargo": "MongoDB DBA",
                        "Tecnologias": ["Openstack", "Openshift", "MongoDB"],
                        "Proyectos": "{ "Openstack": ["Cliente1","Cliente2"],
                                        "Openshift": ["Cliente4" ]
                                    }
}
```

Instalación MongoDB

Intentaremos instalar **instalar en un servidor Linux**. (Ubuntu, Debian)

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/

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

Esta explicación es muy pobre...

Este parece que instala mongo con .deb y luego la shell para el comando mongo:

https://www.youtube.com/watch?v=L5bRKSBflwM en debian

Este instala en server debian, parece más completo pero no se si funcionará por lo del server debian: este creo que es el mejor, instalar antes debian server

https://www.youtube.com/watch?v=ls2mpE9W6FQ

En este se instala en windows, y se parece al temario, con la variable path

https://www.youtube.com/watch?v=_C6AuXNySqo

otra instalación que parece buena, y casi funciona:

https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-debian/

primero instalé esto:

FUNCIONÓ ESTA pero no conecta a la bbdd:

https://www.ochobitshacenunbyte.com/2015/01/12/mongodb-en-gnu-linux/

para manejo de mongo esta bien esto:

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/mongodb-en-debian/

Esta instalacion funciona:

https://medium.com/@hackthebox/installing-mongodb-7-0-community-edition-on-debian-12-bookworm-a-comprehensive-guide-19906ddb47ad

y cuando probelam con libssl1.1:

https://askubuntu.com/questions/1403619/mongodb-install-fails-on-ubuntu-22-04-depends-on-libssl1-1-but-it-is-not-insta

Una vez que está todo hecho:

sudo systemctl enable mongod --now

se queda parado activo...

disable para apagar

en otra terminal:

sudo systemctl start mongod

y comprobam,os:

sudo systemctl status mongod

y está activo!

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.

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.

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 esta**do 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**.

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

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

use miBD

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

db.**createCollection**(“holaMundo”)

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

db.holaMundo.**insert**({“Nombre” : “Ernesto”, “Apellido” : “Perez”, “Edad” : “45”})

De esta forma realizaremos **inserciones en nuestra nueva colección** creada previamente.

Para **ver todos los registros** de nuestra colección:

db.holaMundo.**find**()

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.

Operaciones en el Shell

En esta ocasión, abordaremos el tema de la shell de MongoDB, la cual desempeña un papel fundamental en la interacción con la base de datos. La shell es el entorno donde proporcionamos instrucciones para realizar diversas operaciones, como **escritura, lectura, actualización de datos**, entre otras. Además, nos brinda la capacidad de realizar importaciones y exportaciones de datos, así como otras operaciones diversas.

Una **alternativa gráfica** para administrar la información de la base de datos es **Robomongo**, una aplicación que ofrece una interfaz visual para este propósito. Sin embargo, la **verdadera potencia se encuentra en la shell**.

Dentro de la shell, encontramos diversos comandos esenciales. Entre ellos, el comando `help` nos proporciona asistencia general, mientras que `db.help` nos muestra información específica sobre los comandos disponibles en la base de datos. Si especificamos el nombre de una colección, como en `db.nombreColeccion.help`, obtenemos una **lista de comandos aplicables** a esa colección en particular.

A continuación, algunos **comandos básicos** que podemos ejecutar en la shell:

1. `help`: Muestra la ayuda general.
2. `db.help`: Proporciona ayuda sobre los comandos disponibles en la base de datos.
3. `show dbs`: Lista las bases de datos disponibles.
4. `db`: Devuelve la **base de datos actual**.
5. `show collections`: Muestra la **cantidad de colecciones** en la base de datos actual.
6. `use db`: **Mueve o crea una base de datos** y **se sitúa** en ella.
7. `show user`: Muestra los usuarios de la base de datos actual.

Estos comandos permiten obtener información y desplazarse entre bases de datos. Para acceder a una guía completa de comandos, se recomienda visitar los documentos en `**docs.mongodb.com**`, donde el manual de referencia proporciona detalles sobre todas las colecciones y comandos disponibles.

Entre los comandos avanzados, encontramos ejemplos como `**db.collection.count**`, que cuenta la cantidad de documentos en una colección, o `**db.collection.createindex**`, que facilita la creación de **índices en la colección**.

En resumen, MongoDB ofrece una amplia variedad de opciones que pueden explorarse y aprenderse con detenimiento en la sección correspondiente del sitio web oficial de MongoDB.

A parte de la shell de mongoDB, como acabamos de decir, 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.

Compass es fácil y funciona.

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

Inserción de datos

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

Base de datos, punto, colección, punto, insert, abro paréntesis y abro corchete. A continuación el campo seguido de dos puntos, y el valor entre comillas dobles si es string, si es número es sin comillas. Se cierra el corchete y el paréntesis al terminar el insert.

Consulta de datos

Este es un ejemplo claro de cómo se consultarían datos en nuestra base de datos Mongo:

Base de datos, punto, colección, punto, find. Abrimos paréntesis y corchete, la query criteria se forma con el campo y dos puntos, seguido de el signo del dólar, el criterio o filtro gt y el valor (18). Añadimos una coma y la projection se forma igual, entre corchetes, y en el interior el campo y valor separados por dos puntos, cerrando a continuación el paréntesis del find, poniendo un punto y añadiendo un cursor modifier, que tendrá entre paréntesis un valor:

collection → db. users. Find(

query criteria → { age: { Sgt 18 } },

projection → { name: 1, address: 1 }

cursor modifier →). Limit(5)

Actualizar información

Podemos observar cómo se realizaría el Update:

Base de datos, punto, colección, punto, update, abre paréntesis (abre corchete y aquí va el update criteria, que será el campo seguido de dos puntos, abre corchete, signo de dólar con el criterio gt y seguido de dos puntos, y el valor (18), cerrando los dos corchetes, coma, y abriendo corchete para el update action, con el signo del dólar y set, dos puntos, abre corchete y otro campo (status) seguido de dos puntos y el valor (“A” entre comillas en este caso), cerrando los dos corchetes, y una coma, y terminamos abriendo corchete para el option update, que será: multi seguido de dos puntos y el valor a true. Cerramos el corchete del update option y el paréntesis del update de la colección.

collection → db.users.update(

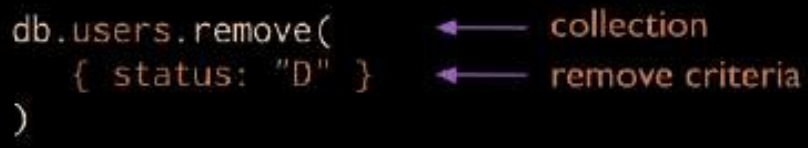
update criteria → { age: { $gt: 18 } },

update action → { $set: { status: "A" } },

update option → { multi: true }

)

Borrado de documentos



Base de datos, punto, colección, punto, remove y abre paréntesis. Ahora el remove criteria entre corchetes, que será un campo (status) seguido de dos puntos, y el valor entre comillas (“D”). Por último cerramos el paréntesis del remove.

collection → db.users. Remove(

remove criteria → { status: “A” }

)

Ejemplo de Insertar y borrar

Vamos a 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"})

Estudiadas las sentencias CRUD básicas para el manejo de datos realizaremos la inserción de datos con el **comando insert**.

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"})

Ejemplo de creación de colección Empleados y consulta en MongoDB

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.

https://docs.mongodb.com/

https://www.paradigmadigital.com/