MONGO

https://bit.ly/2v5EaEv

MongoDB (del inglés humongous, "enorme") es un sistema de base de datos NoSQL orientado a documentos. Es la base dedatos no relacional más popular. Su gran fortaleza viene de su flexibilidad.

Instalación

Pre-requisitos:

```
$ sudo apt-get install libcurl4 openssl
```

Para obtener los binarios se puede correr el comando:

```
$ wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu1804-4.2.3.tgz
```

Y para descomprimir:

```
$ tar -xzvf mongodb-linux-x86_64-ubuntu1804-4.2.3.tgz
```

Instalación

Desde la nueva carpeta en *bin* se puede correr mongo, pero seguramente prefieren copiarlo al *path*.

```
$ cd mongodb-linux-x86_64-ubuntu1804-4.2.3/
$ sudo cp bin/* /usr/local/bin/
```

Instalación

Antes de correr la base de datos, se necesita crear un directorio en el que se va a guardar la información, sin cambiar configuraciones, mongo espera uno por defecto:

```
$ sudo mkdir -p /data/db
$ sudo chown `whoami` /data/db
```

OJO con esas últimas comillas, provocan que la terminal ejecute el comando, si quieren pueden escribir allí su nombre de usuario en lugar de esa instrucción.

Instalación

Instalación

Antes de correr la base de datos, se necesita crear un directorio en el que se va a guardar la información, sin cambiar configuraciones, mongo espera uno por defecto:

```
$ sudo mkdir -p /data/db
$ sudo chown `whoami` /data/db
```

OJO con esas últimas comillas, provocan que la terminal ejecute el comando, si quieren pueden escribir allí su nombre de usuario en lugar de esa instrucción.

Si tienen problemas para encontrar esa comilla especial, mejor simplemente escriban el nombre de su usuario, el efecto será el mismo.

mongo shell

También aquí tenemos un comando para comenzar el servidor:

$ mongod

Y otro para el cliente:

$ mongo

mongo shell

Primero lo primero, elegir/crear una base de datos:

```
mongo> use basePrueba;
```

A partir de ese momento "db" se va a referir a la base de datos que elegimos, y vamos a utilizarlo para prácticamente todos los comandos.

Para insertar un elemento, debemos elegir una colección en la que se va a añadir, no hace falta declararla, simplemente la usamos como si ya existiera:

```
mongo> db.coleccion.insertOne({llave:"valor"})
```

mongo shell

Para buscar elementos, se usa la función *find*. OJO, por defecto, *mongo shell* limitará el resultado a 20 elementos (aunque se pueden pedir más). Pero en los drivers, la consulta regresa iteradores.

mongo> db.coleccion.find()

Cuando no recibe ningún argumento, la función *find* regresa todos los elementos.

mongo shell

Así como se puede insertar un documento, se pueden insertar muchos, y recuerden que son documentos completos, no solo llave valor:

```
mongo> db.coleccion.insertMany([
   {
      nombre: "Alejandro" ,
      asistencia: 40 ,
      rol: "profesor" ,
   } , {
      nomre: "X" ,
      asistencia: 35 ,
      rol: "alumno" ,
   }
])
```

mongo shell

mongo shell

Así como se puede insertar un documento, se pueden insertar muchos, y recuerden que son documentos completos, no solo llave valor:

```
mongo> db.coleccion.insertMany([
  {
    nombre: "Alejandro" ,
    asistencia: 40 ,
    rol: "profesor" ,
  } , {
    nomre: "X" ,
    asistencia: 35 ,
    rol: "alumno" ,
  }
])
```

Pongan atención en la sintaxis, para agregar varios documentos al mismo tiempo, se debe usar una lista (con corchetes).

mongo shell

También es importante poder especificar las búsquedas, y aquí lo importante es recordar: Cada parámetro de la función es un documento, y cada criterio a buscar es la entrada de un documento:

```
mongo> db.coleccion.find(
  {
    nombre: "Alejandro"
  } , {
    nombre: 1 ,
    rol: 1
  }
)
```

```
SELCT nombre,rol FROM coleccion
  WHERE nombre = "Alejandro"
```

mongo shell

También es importante poder especificar las búsquedas, y aquí lo importante es recordar: Cada parámetro de la función es un documento, y cada criterio a buscar es la entrada de un documento:

```
mongo> db.coleccion.find(
  {
    nombre: "Alejandro"
  } , {
    nombre: 1 ,
    rol: 1
  }
)
```

```
SELCT nombre,rol FROM coleccion
  WHERE nombre = "Alejandro"
```

Aquí la instrucción de SQL es la equivalente de la consulta de mongo. Si se fijan, tiene las mismas partes pero el orden y la sintaxis es distinta. El primer documento es el filtro, lo que hay que buscar. El segundo documento es la información que se quiere obtener (el SELECT ...).

mongo shell

Por supuesto, hay mas condiciones que se pueden buscar:

```
mongo> db.coleccion.find(
   {
      nombre: { $in : [ "X" , "Y" ] }
   }
)
```

```
SELCT * FROM coleccion WHERE nombre in ("X",""Y")
```

mongo shell

mongo shell

Por supuesto, hay mas condiciones que se pueden buscar:

```
mongo> db.coleccion.find(
  {
    nombre: { $in : [ "X" , "Y" ] }
  }
)
```

```
SELCT * FROM coleccion WHERE nombre in ("X",""Y")
```

La instrucción se compone de un documento dentro de un documento, es decir, se tiene un documento de búsqueda para encontrar "nombre = ... ". Pero en lugar de darle un valor, se le da un nuevo documento, la llave de ese documento es la instrucción, y el valor está relacionado con esa instrucción. El símbolo "$" es el que le indica a mongo que esa palabra clave es una instrucción.

mongo shell

OR:

```
mongo> db.coleccion.find(
   {
      $or: [
            { nombre : "Alejandro"} ,
            { rol: "profesor"}
      ]
   }
)
```

```
SELCT * FROM coleccion WHERE nombre = "Alejandro"
  OR rol = "profesor"
```

mongo shell

mongo shell

OR:

```
mongo> db.coleccion.find(
  {
    $or: [
          { nombre : "Alejandro"} ,
          { rol: "profesor"}
    ]
  }
)
```

```
SELCT * FROM coleccion WHERE nombre = "Alejandro"
  OR rol = "profesor"
```

Aquí ocurre lo mismo, solo que la instrucción se usa directamente, no necesita estar dentro de una búsqueda inicial.

mongo shell

Rangos:

```
mongo> db.coleccion.find(
   {
      asistencia : { $lt : 30 } ,
   }
)
```

```
SELCT * FROM coleccion WHERE asistencia < 30
```

mongo shell

Rangos:

```
mongo> db.coleccion.find(
  {
    asistencia : { $lt : 30 } ,
  }
)
```

```
SELCT * FROM coleccion WHERE asistencia < 30
```

Existen muchas instrucciones, pueden buscar todo un catálogo en la documentación de mongo, las comparaciones como menor que y mayor que son de las mas comunes.

mongo shell

```
mongo> db.coleccion.find(
   {
      nombre: "Alejandro" ,
      $or: [
            { asistencia : { $gt : 30 } } ,
            { rol: "profesor"}
      ]
   }
)
```

Intenten encontrar el equivalente de esta consulta (la respuesta en la siguiente diapositiva).

mongo shell

```
mongo> db.coleccion.find(
   {
      nombre: "Alejandro" ,
      $or: [
            { asistencia : { $gt : 30 } } ,
            { rol: "profesor"}
      ]
   }
)
```

Intenten encontrar el equivalente de esta consulta (la respuesta en la siguiente diapositiva).

```
SELCT * FROM coleccion WHERE nombre = "Alejandro"
  AND ( rol = "profesor" OR asistencia > 30 )
```

mongo shell

Sin duda ya todos notaron el `"_id":ObjectId("...")`. Cada que se agrega un nuevo valor, se le asigna automáticamente un ID. Así que se necesita algo diferente para cambiar un valor.

```
mongo> db.coleccion.updateOne(
   { apellido: "Y" } ,
   { $set : { apellido : "Z" } }
)
```

mongo shell

O muchos.

```
mongo> db.coleccion.updateMany(
   { asistencia: {$lt:40} } ,
   { $set : { exento : false } }
)
```

mongo shell

Borrar documentos funciona igual.

```
mongo> db.coleccion.deleteOne(
   {
      llave: "valor"
   }
)

mongo> db.coleccion.deleteMany( )
```