MongoDB Proyecto

MIguel Ángel Martín Serrano

I.E.S Gonzalo Nazareno

21 Marzo, 2014

Contenido

Descripción breve del proyecto

Partimos de la base de que no tenemos ningún conocimiento sobre base de datos no relacionales, excepto su existencia, y elegimos el sistema gestor de base de datos no relacional más popular "MongoDB", para llevar a cabo distintos objetivos.

Aquí vamos a explicar desde un punto de vista técnico en el que se tienen ya ciertos conocimientos de base de datos relacionales.

Vamos a poner en funcionamiento un entorno, más cocnretamente una aplicación basa en python, para usar la base de datos mongo.

Además vamos a ver cómo, una vez comprendido mongo, podemos optimizar su funcionamiento y un mantenimiento óptimo.

Todo lo aquí descrito se va a llevar a cabo desde un sistema **Linux** más concretamente **Debian 7 Whezzy** en un entorno de pruebas nunca de producción (Aunque se va a tratar cómo si de producción se tratara para hacerlo lo más real posible).

Definiciones

¿Que es MongoDB?

MongoDB (de la palabra en inglés "**humongous**" que significa enorme) es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formatoBSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

¿Por que Mongo db? Ventajas y desventajas

A grosso modo, y de manera muy resumida, mongo ofrece mayor velocidad y mejor escalabilidad que un sistema tradicional de base de datos de tipo relacional, pero no cumple **ACID:** Atomicidad, Consistencia, Aislamiento y Durabilidad.

Actualmente los portales web más visitados y famosos del mundo cómo Facebook, utilizan este tipo de base de datos por la cantidad tan grande de datos que deben de servir en el menor tiempo posible.

Vamos entonces a empezar con mongoDB, y comenzamos por la instalación.

Conociendo MongoDB

Modelado de datos

Una de las partes más dificiles es crear una base de datos no relacional desde cero, para ello lo principal es tener claro los tipos de datos:

Tipos de datos

A continuación detallamos, algunos de los tipos de datos que soporta mongo:

- **Integer** – Números enteros.
- **Double** – Números con decimales.
- **Boolean** – Booleanos verdaderos o falsos.
- **Date** – Fechas.
- **Timestamp** – Estampillas de tiempo.
- **Null** – Valor nulo.
- **Array** – Arreglos de otros tipos de dato.
- **Object** – Otros documentos embebidos.
- **ObjectID** – Identificadores únicos creados por MongoDB al crear documentos sin especificar valores para el campo _id.
- **Data Binaria** – Punteros a archivos binarios.
- **Javascript** – Código y funciones Javascript.
- **String** – Cadenas de caracteres.

Podemos guardar los diferentes datos, según un patrón, los dos más comúnes son:

Embeber

Este patrón se enfoca en incrustar documentos uno dentro de otro con la finalidad de hacerlo parte del mismo registro y que la relación sea directa.

Referenciar

Este patrón busca imitar el comportamiento de las claves foráneas para relacionar datos que deben estar en colecciones diferentes.

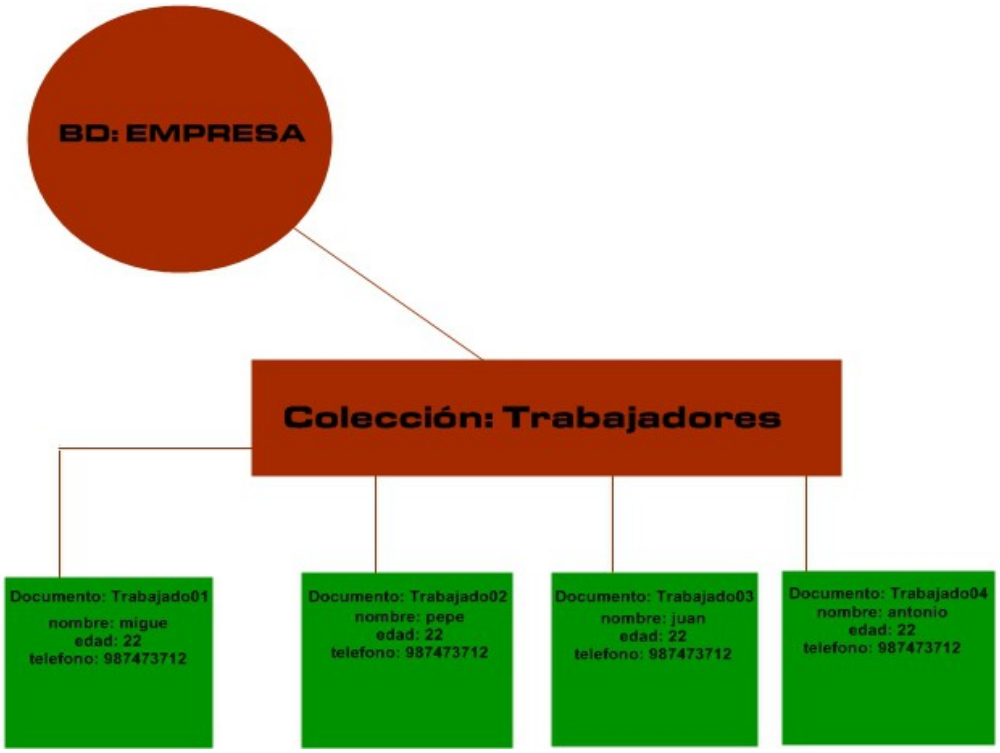
Estructura de Mongo

Lo principal sería comprender cómo funciona mongo, es decir la estructura que sigue a la hora de guardar o alojar los datos.

Mongo, puede tener varias **base de datos**, dentro de estas base de datos va a guardar **colecciones** y las colecciones no son más que un **conjunto de documentos**, los cuales guardan información a cerca de algo en concreto, con el formato clave-valor.

Un ejemplo básico, supongamos que tenemos una base de datos llamada: EMPRESA, dentro de esta tenemos una colección llamada TRABAJADORES, y dentro de esta colección tenemos un documento llamado TRABAJADOR01, y dentro de estos documentos los valores para las claves deseadas:

A continuación se detalla mejor en la imagen:



Instalación y gestión básica

Dado que estamos en un entorno Debian, MongoDB, esta en los repositorio de Debian actualmente, así que bastaría con ejecutar:

```
apt-get install mongo-db
```

Ahora ya tenemos mongo instalado en nuestra máquina. Mongo es un servicio más cómo puede serlo MySQL o Apache, así que una de las maneras de Arrancar, parar, reiniciar o ver el estado de el servicio es :

```
service mongodb [ start | stop | restart | status ]
```

Bueno, ahora que conocemos a mongo como servicio, vamos a usarlo, simplemente ejecutando en la consola "**mongo**", accederemos a mongo y a las bases de datos que tengamos, si hemos utilizado mysql, el equivalente a esto sería ejecutar en la terminal "**mysql -p -u {usuario_actual}"**, y cuándo ejecutamos mongo desde la línea de comandos, da igual el usuario que estemos usando en ese momento, en el ejemplo entramos cómo 'root', y se conecta a la base de datos por defecto, test, si no queremos que esto suceda tendriamos que arrancar mongo con otro parametros que veremos má adelante:

```
root@debian:/home/usuario# mongo
MongoDB shell version: 2.0.6
connecting to: test
>
```

En lo que a la gestión de usuarios se refiere, mongo por defecto no trae usuarios y permite el acceso a todos a todas las base de datos, para cambiar esto, una de las formas que utilizado es, editando el fichero **/etc/mongodb.conf** y descomentar la línea:

```
auth:True
```

Y luego hemos creado el usuario para la base de datos empresa de la siguiente manera, hemos accedido como root a mongo:

```
use empresa
db.adduser('migue','admin')
```

Y ya deberíamos de poder acceder a la base de datos empresa con este usuario y no anonimamente, si hubiéramos creado el usuario con el parametro 'True' tendría solo permisos de lectura.

Ya hemos visto cómo acceder, es hora de ver la gestión de las base de datos, crear,borrar, listar y comandos más interesantes etc..

```
> db
test
```

Con esta instrucción vemos la base de datos que estamos usando ahora mismo

```
> use pruebas
switched to db pruebas
```

Este comando cambia a la base de datos especificada en este caso pruebas, y si no existe la crea.

```
> show dbs
local     (empty)
pruebas
```

Con este comando podemos ver las base de datos disponibles para el usuario

Si necesitamos ayuda, bastaría con ejecutar el comando "**help**" dentro de mongo o "**db.help()**" para obtener ayuda a cerca de los metodos de base de datos.

Una vez que estamos dentro de una base de datos, vamos a ver cómo gestionamos la base de datos en cuestion.

Equivalencias y diferencias base de datos relacional

A continuación mostramos algunas posibles equivalencias, lo que en una base de datos relacional sería una relacion 1-1 en mongo se podría definir como un documento dentro de otro así: (**embeber**)

```
Persona = {
  nombre :  'Jonathan',
  apellido :  'Wiesel',
  genero : 'M',
  documentos : {
               pasaporte  : 'D123456V7',
               licencia :  '34567651-2342',
               seguro_social  :  'V-543523452'
               }
}
```

Lo equivalente a una relación 1-n:

```
Persona = {
    nombre      :   'Jonathan',
    apellido    :   'Wiesel',
    genero      :   'M'
}

Documentosdb.insertPersonales = {
    pasaporte       :   'D123456V7',
    licencia        :   '34567651-2342',
    seguro_social   :   'V-543523452'
}
```

Ahora supongamos otro caso una relacion 1-*, es decir una documentos puede estar relacionado con varios:

```
Direccion1 = {
    _id             :   1,
    pais            :   'Venezuela',
    estado          :   'Distrito Capital',
    ciudad          :   'Caracas'
    urbanizacion    :   'La Florida',
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...
}

Direccion2 = {
    _id             :   2,
    pais            :   'Estados Unidos',
    estado          :   'Florida',
    ciudad          :   'Miami'
    urbanizacion    :   'Aventura',
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...
}

Persona = {
    nombre      :   'Jonathan',
    apellido    :   'Wiesel',
    genero      :   'M',
```

```
    direcciones :   [1,2]
}
```

Otra forma de hacer lo anterior:

```
Direccion1 = {
    _id             :   1,
    pais            :   'Venezuela',
    estado          :   'Distrito Capital',
    ciudad          :   'Caracas'
    urbanizacion    :   'La Florida',
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...,
    persona_id      :   1
}

Direccion2 = {
    _id             :   2,
    pais            :   'Estados Unidos',
    estado          :   'Florida',
    ciudad          :   'Miami'
    urbanizacion    :   'Aventura',
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...,
    persona_id      :   1
}

Persona = {
    _id         :   1
    nombre      :   'Jonathan',
    apellido    :   'Wiesel',
    genero      :   'M'
}
```

Relacion *-*:

```
Direccion1 = {
    _id             :   1,
    pais            :   'Venezuela',
    estado          :   'Distrito Capital',
    ciudad          :   'Caracas'
    urbanizacion    :   'La Florida',
```

```
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...,
    personas        :   [1000]
}

Direccion2 = {
    _id             :   2,
    pais            :   'Estados Unidos',
    estado          :   'Florida',
    ciudad          :   'Miami'
    urbanizacion    :   'Aventura',
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...,
    personas        :   [1000,1001]
}

Persona1 = {
    _id         :   1000,
    nombre      :   'Jonathan',
    apellido    :   'Wiesel',
    genero      :   'M',
    direcciones :   [1,2]
}

Persona2 = {
    _id         :   1001,
    nombre      :   'Carlos',
    apellido    :   'Cerqueira',
    genero      :   'M',
    direcciones :   [2]
}
```

Si la tabla en las dos tablas relacionales tiene algun campo mas este sería el equivalente:

```
Direccion1 = {
    _id             :   1,
    pais            :   'Venezuela',
    estado          :   'Distrito Capital',
    ciudad          :   'Caracas'
    urbanizacion    :   'La Florida',
    avenida         :   ...,
    edificio        :   ...,
```

```
    piso            :   ...,
    apartamento     :   ...,
    personas        :   [1000]
}

Direccion2 = {
    _id             :   2,
    pais            :   'Estados Unidos',
    estado          :   'Florida',
    ciudad          :   'Miami'
    urbanizacion    :   'Aventura',
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...,
    personas        :   [1000,1001]
}

Persona1 = {
    _id         :   1000,
    nombre      :   'Jonathan',
    apellido    :   'Wiesel',
    genero      :   'M',
    direcciones :   [{
            direccion_id    :   1,
            viveAqui        :   true
        },{
            direccion_id    :   2,
            viveAqui        :   false
        }]
}

Persona2 = {
    _id         :   1001,
    nombre      :   'Carlos',
    apellido    :   'Cerqueira',
    genero      :   'M',
    direcciones :   [{
            direccion_id    :   2,
            viveAqui        :   true
        }]
}
```

Algunos ejemplos:

```
db.trabajadores.find({"Nombre": "Inma"})
```

Esto sería al equivalente en una base de datos relacional a un select con where

nombre = Inma, es decir sacar todos los datos de los ficheros con el atributo “Nombre” con valor “Inma”.

```
db.trabajadores.insert({Nombre: "Jose Alejandro", FechaNac:
"26/03/2014", Direccion: "C/Aviacion nº30", Uid: "3"})
```

Esto sería equivalente a un insert en una base datos relacional un nuevo registro.(Aquí es dónde esta una de las grandes diferencias, en una base de datos relacional la estructura de la tabla esta previamente definida, en mongo vamos definiendo la estrucutra del documento según vamos insertando el registro).

```
db.trabajadores.update({"Uid": "1"}, {"$set":{"Apellidos":"Martin
Serrano"}})
```

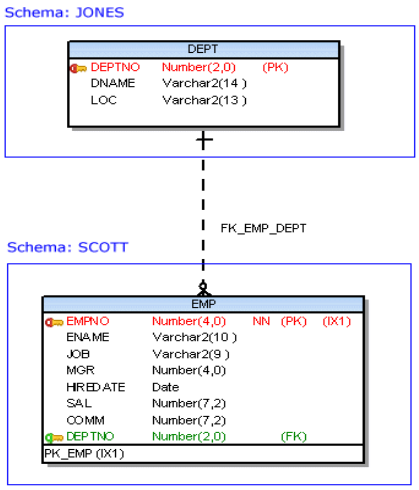
Este sería el equivalente a hacer un update en una base de datos relacional. Modificar el usuario con el Uid con valor 1, el apellido por el mostrado.

```
db.trabajadores.remove({"Uid": "2"})
```

Esto sería el equivalente a “delete” en una base de datos relacional, elimnar a el usuario con Uid = 2

Esquema SQL en MongoDB

Ahora vamos a ver cómo pasar o migrar un sencillo esquema desde SQL a MongoDB:



El esquema de scott es muy familiar y simple con el que todos empezamos a utilizar oracle, así que veo una buena idea para implementarlo como el primer esquema a utilizar en mongodb y además es simple.

En primer lugar se trata de una relación 1-n, entre dos tablas (Trabajadores y departamentos) en la cuál se migra la clave primaria de departamentos (id) a la tabla de los trabajadores.

El equivalente en mongo sería el siguiente, dentro de la base de datos empresa, creamos una colección llamada sucursales:

```
{
   _id: "1",
   dname: "SALES",
   loc: "California"
}

Y luego en la colección trabajadores:

{
   _id: 3,
   name: "SCOTT",
   job: "SALESMAN",
   hiredate: ISODate("2010-09-24"),
   mgr: 216,
   sal: "1000",
   comm: "200",
   empno_id: "1"
}

{
   _id: 2,
   name: "MICHAEL",
   job: "TECHNIC",
   hiredate: ISODate("2010-09-24"),
   mgr: 216,
   sal: "1000",
   comm: "200",
   empno_id: "1"
}
```

Aplicación python web

Vamos a realizar una aplicación basada en el esquema anterior, tenemos una base de datos llamada "**Empresa**" y una colección llamada "**Trabajadores**". Vamos a realizar una aplicación web en pthon-bootle web para consultar, editar, insertar o borrar datos relacionados con esta base de datos:

Lo principal es crear la base de datos y luego la colección que vamos a utilizar y mas tarde algunos fichero de prueba.

Creamos la base de datos y creamos un nuevo registro y con ellos la colección:

```
Use empresa
```

```
db.trabajadores.insert({Nombre: "Jose Alejandro", FechaNac:
"26/03/2014", Direccion: "C/Aviacion nº30", Uid: "3"})
```

La aplicación la vamos a realizar en el lenguaje de programación python, utilizaremos la librería Pymongo, disponible en los repositorios de debian, asi que para instalarla bastará con:

```
apt-get install pymongo
```

Y tambien necesitamos la librería de bootle:

```
apt-get install python-bottle
```

Una vez instalado desde python la podemos utilizar haciendo un simple

```
import pymongo
```

Vamos con la estructura de la aplicación, vamos a tener varios ficheros en python/bottle:

```
usuario@debian:~/mongo$ ls
adduser    editar    inicio   README.md  template
confirmar  index.py  LICENSE  static
```

adduser: Fichero encargadao de añadir usuarios nuevos

editar: Fichero encargado de editar usuarios

inicio: Patanlla de inicio dónde nos autenticamos para acceder a la aplicación

template: Es la página principal una vez que nos autenticamos, de aquí sacamos un listado de todos los usuarios

confirmar: Aquí vamos una vez que hemos decidido borrar un usuario.

Index.py: Es el programa principal en el se ejecuta el framework.

Ya tenemos la base de datos, es hora de empezar a utilizar pymongo:

```
#Importamos las bibliotecas necesarias
import pymongo
from pymongo import Connection
conn = Connection()
db = conn.empresa
```

```
db.authenticate('migue','admin')
trabajadores = db.trabajadores
#Ya estamos desde python conectados a nuestra base de datos, con el usuario con el que estemos ejecutando, así que vamos a sacar todos los registros:
for trabajador in trabajadores.find():
        nombre = trabajador['Nombre']
        direccion = trabajador['Direccion']
        fechanac =  trabajador['FechaNac']
        uid = trabajador['Uid']
        apellidos = trabajador['Apellidos']
#Ahora vamos a ver como borrar un registro
trabajadores.remove({"Uid": 3})
#Como editar un registro donde %s es el valor
trabajadores.update({"Uid": %s}, {"$set":{"Apellidos": "%s", "Nombre": "%s", "FechaNac": "%s", "Direccion": "%s"}})
#Y finalmente como insertar un registro
registronuevo = {"Nombre": nombrenuevo, "Apellidos": apellidonuevo, "FechaNac": fechanacnuevo, "Direccion": direccionnuevo, "Uid": nuevouid }
trabajadores.insert(registronuevo)
```

Luego veremos más detalladamente cada fichero de la aplicación y funcionando.

https://github.com/miguelmartin/mongo

Mantenimiento de MongoDB

La creación de copias de seguridad es una de las tareas más típicas en el mantenimiento de una base de datos, en mongo tenemos la herramienta mongodump:

Podemos hacer backups consistentes (Base de datos parada) o inconsistentes (Base de datos funcionando)

Consistente:

```
mongodump --dbpath /var/lib/mongo -o dump_consistente
```

Inconsistente:

```
root@debian:/home/usuario/mongo# mongodump -h localhost -d empresa -o empresadump -u migue -p admin
```

```
connected to: localhost
DATABASE: empresa    to  empresadump/empresa
     empresa.system.indexes to
empresadump/empresa/system.indexes.bson
           3 objects
     empresa.empresa to empresadump/empresa/empresa.bson
           0 objects
     empresa.trabajadores to empresadump/empresa/trabajadores.bson
           9 objects
     empresa.system.users to empresadump/empresa/system.users.bson
           1 objects
```

Y para restaurar la copia:

```
mongorestore --host 192.168.1.2 --port 3017 --db empresa
--username migue --password admin --drop /backup/dump
```

Otra herramienta ndispensable como en cualquier servicio, para la monitorización y el mantenimiento es el log situado en **/var/log/mongodb/mongodb.log**

Replicación

Vamos a ver cómo replicar mongodb de una forma simple y sencilla.

En primer luegar tenemos que asegurarnos que en cada nodo podemos resolver correctamente el nombre del otro, si no tenemos servidor DNS una buena manera es modificando el fichero **/etc/hosts** :

```
ip_address     mongo_host.mongo.db
```

Una vez que estamos seguros de esto, modificamos el fichero **/etc/hostname** para darle el nombre correcto a el nodo en cuestión en el que estemos en ese momento.

Paramos mongo:

```
service mongodb stop
```

Y creamos el directorio

```
mkdir /mongo-metadata
```

Y modificamos el fichero **/etc/mongodb.conf**:

```
dpath=/mongo-metadata
```

Y nos aseguramos de que esta trabajando en el puerto correcto:

```
port = 27017
```

Añadimos el siguiente parametro con el siguiente valor:

```
replSet = rs0
```

Ponemos el valor de la variable fork a true:

```
fork = true
```

Y leemos de nuevo el fichero de configuración

```
mongod —config /etc/mongodb.conf
```

Ahora vamos a iniciar la replicación entramos en mongo con el comando mongo y iniciamos:

```
rs.initiate()
```

Y vemos la configuración:

```
rs.conf()
{
    "_id" : "rs0"
    "version" : 1,
    "members" : [
        {
            "_id" : 0,
            "host" "mongo0.example.com:27017"
        }
    ]
}
```

Y para añadir un nodo nuevo:

```
rs.add("mongo1.example.com")
```

Conclusiones

Despues de bastantes horas trabajando con mongodb, he llegado a algunas conclusiones :

En primer lugar no se trata de un gestor de base de datos tan disparado y desordenado cómo puede parecer a primera vista, si no, yo entiendo, que se deja mucha más libertad a el programador de la aplicación que utilice esta base de datos para moldearla a su aplicación(Solo le veo uso a este tipo de base de datos en la web, de ser en otra situación ya depende de el número de consultas que pueda llegar a tener).

Si tienes los conocimientos y la estructura de los datos clara y se ve claro que se puede hacer en este tipo de base de datos, es una gran ventaja en cuanto a

escalabilidad y rapidez.

Su mayor arma y más potente es la escalabilidad y flexibilidad, pero gran parte de esto es debido a que no cumple la normativa ACID, entonces es por ello que no tiene mucho sentido migrar, pasar o basar un diseño de base de datos no relacional en un diseño de una base de datos relacional, por que perderíamos gran parte de esta potencia. Poder replicar la información de la base de datos en varios nodos, es otra de las grandes ventajas respecto a las base de datos SQL, dado que esto se puede llevar a cabo también en SQL, pero no funciona de forma tan rápida, y en un situación con miles de peticiones se notaría bastante, dado que la diferencia de tiempo de respuesta entre una relacional y otra no relacional no es nada despreciable.

En definitiva es un gestor muy potente, pero que requiere conocimientos claros, para poder llevar a cabo una uso correcto y poder sacarle el 100% y todas las ventajas que tienes sobre SQL.

Fuentes:

http://codehero.co/mongodb-desde-cero-modelado-de-datos/

http://docs.mongodb.org/manual/reference/sql-comparison/

http://docs.mongodb.org/manual/reference/mongo-shell/

http://blog.jam.net.ve/2011/01/09/usos-basicos-de-mongodb-console/