MongoDB

MIguel Ángel Martín Serrano

¿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 groso 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.

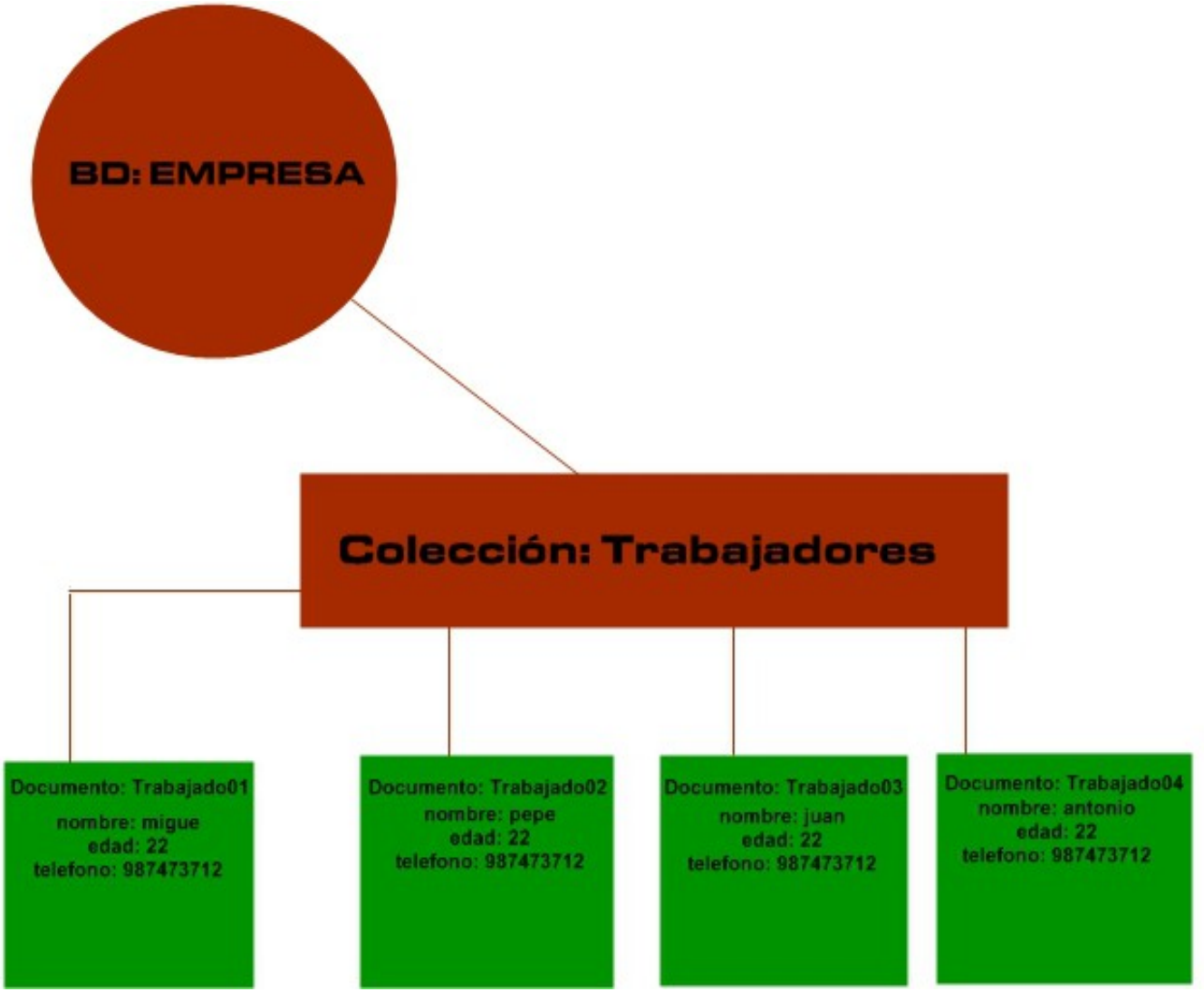
Estructura de Mongo

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:


BD: EMPRESA
Colección: Trabajadores
Documento: Trabajado01
nombre: migue
edad: 22
telefono: 987473712
Documento: Trabajado02
nombre: pepe
edad: 22
telefono: 987473712
Documento: Trabajado03
nombre: juan
edad: 22
telefono: 987473712
Documento: Trabajado04
nombre: antonio
edad: 22
telefono: 987473712

Tipos de datos

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.

Almacenamiento de datos

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.

Gestión de usuarios

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, hemos editado 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')
```

Instalación y gestión básica

Instalar

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

Controlar el servicio

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

Acceder

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

Equivalencias y diferencias SQL

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'
      }
}
```

Equivalencias y diferencias SQL

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'
}
```

Equivalencias y diferencias SQL

Relacion *-*:

```
Direccion1 = {
    _id             :   1,
    pais            :   'Venezuela',
    estado          :   'Distrito Capital',
    ciudad          :   'Caracas'
    urbanizacion    :   'La Florida',
    avenida         :   ...,
    edificio        :   ...,
    piso            :   ...,
    apartamento     :   ...,
    personas        :   [1000]
}
```

Equivalencias y diferencias SQL

Relacion *-*:

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

Equivalencias y diferencias SQL

Relacion *-*:

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

Otras equivalencias SQL

Algunos ejemplos:

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

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

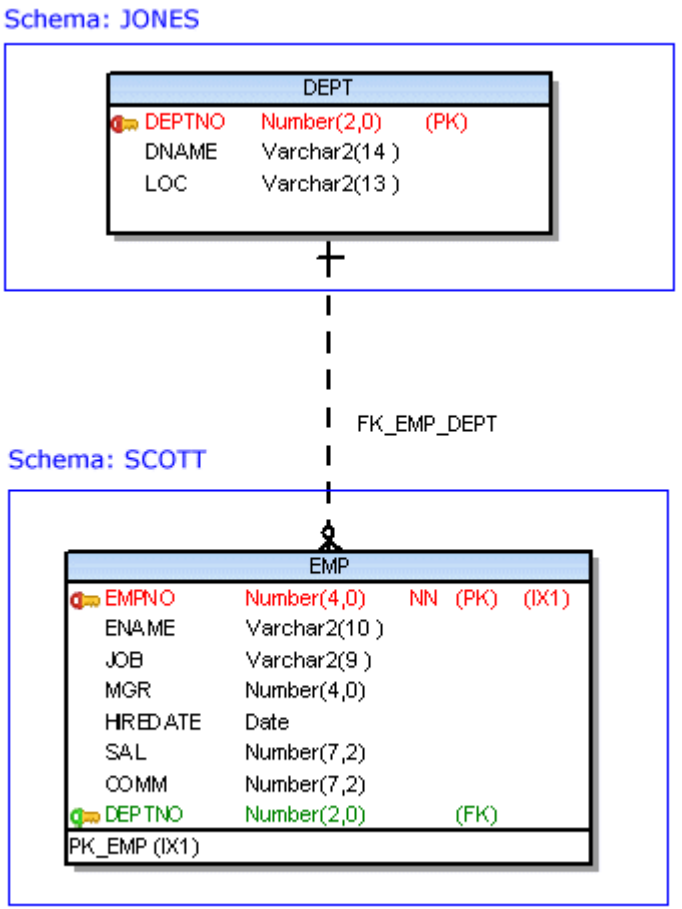
```
db.trabajadores.update({"Uid": "1"}, {"$set":{"Apellidos":"Martin Serrano"}})
```

Este sería el equivalente a hacer un update en una base de datos relacional.

```
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



Esquema SQL en MongoDB

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

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

Aplicación Python Web

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.

Ver aplicación:

https://github.com/miguelmartin/mongo

Mantenimiento MongoDB

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

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

Inconsistente:

```
mongodump -h localhost -d empresa -o empresadump -u migue -p admin
```

Restarurar

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

Replicación

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
```

Replicación

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")
```

Conclusión

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.

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.

En definitiva es un gestor muy potente, pero que requiere conocimientos claros, para poder llevar a cabo una uso correcto.

Fuentes

A continuación de muestran las fuentes de información utilizadas:

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/