INGENIERÍA UNIVERSIDAD DE LOS ANDES MÉRIDA VENEZUELA | DEPARTAMENTO DE COMPUTACIÓN ESCUELA DE SISTEMAS

Laboratorio de Base de Datos
Práctica Nro. 10, Bases de datos orientada a documentos

Prof. Solazver Solé
Preps. Victor Albornoz, Yenifer Ramírez
Semestre B-2018

1. Introducción

Las bases de datos *NoSql (Not Only SQL)* han ganado popularidad, debido a que han logrado resolver las deficiencias de las bases de datos relaciones a un bajo precio. *MongoDB* es un sistema gestor de bases de datos basado en documentos diseñada para aplicaciones modernas. Esta funciona bajo el concepto de colecciones y documentos, ofreciendo un gran rendimiento, escalabilidad y flexibilidad.

- A la misma velocidad que el hardware, las aplicaciones y los datos han obtenido un cambio masivo durante los últimos años. No obstante las técnicas de manejos de datos han prevalecido.
- Toda la información se almacena de forma diaria, en los dispositivos móviles, correo, audio, videos, web logs, redes sociales, redes de difusión de contenido, sensores y demás información.
- Toda esta información sirve para comprender diferentes tipos de fenómenos, como el comportamiento de las masas, las tendencias sociales, los intereses de un cliente por un producto particular para hacer una estrategia de marketing enfocada entre otros.
- A toda esa información se les denomina *datos no estructurados*[1].

[1]No poseen un formato estricto

1.1. Datos no estructurados

- Estadísticas muestran que cerca del 80-90 % de los datos en las organizaciones no es información estructurada.
- La cantidad de datos no estructurados incrementa cada día de manera significante.
- Las bases de datos relaciones no fueron construidas para soportar de manera eficiente estos datos, en términos de rendimiento.
- Para manejar el gran volumen de datos no estructurados, existen una variedad enormes de *herramientas open-source*.
- Las bases de datos *NoSQL (Not only SQL)* están creciendo en popularidad en los últimos días. Principalmente porque su orientación fue hecha para trabajar de manera eficiente con herramientas analíticas y para almacenar grandes cantidades de información.

1.2. Sistemas de Base de Datos Distribuidos

- La mayoria de los sistemas gestores de de bases de datos NoSQL son sistemas de almacenamiento distribuidos, enfocados en:
  - Escalabilidad
  - Disponibilidad, Replicación y consistencia eventual.
  - Modelos de replicación.
  - Posicionamiento horizontal de archivos *Sharding of Files*.
  - Alto desempeño de acceso a datos *(hashing o clave-objeto)*.
- Los sistemas NoSQl se caracterizan por hacer énfasis en la **disponibilidad**, por lo que **la replicación de los datos es inherente** en estos sistemas.
- Otra característica importante de los sistemas NoSQL es la **escalabilidad**.
- Tipos de escabilidad en sistemas distribuidos:
  - **Horizontal**: el sistema distribuido es expandido mediante la **adición de mas nodos** para el almacenamiento de información y el procesamiento de datos.
  - **Vertical**: el sistema distribuido se expande mediante la **adición de mayor capacidad** de almacenamiento y procesamiento **a los nodos existentes**.

1.3. Sistemas NoSQL Populares

- **BigTable:** desarrollado por Google, utilizados en aplicaciones como: Gmail, Google Maps. Se le considera orientado o basado en columnas.
- **DynamoDB:** desarrollado por Amazon, esta disponible a través de los servicios de la nube de Amazon. Se le considera orientado o basado en datos de clave-valor *(key-value data)*.
- **Cassandra:** desarrollado por Facebook, posee una versión open-source conocida como Apache Cassandra. Se le considera basado en columnas y datos de clave-valor.
- **MongoDB:** desarrollado al inicio por 10gen Inc, se le considera un sistema basado en documentos.

2. Características del modelo de datos y lenguajes de consultas en sistemas NoSQL

- **No requiere esquema**, dado que los sistemas NoSQL están orientados al manejo de datos semi-estructurados, los datos son almacenados sin la necesidad de un esquema.

  Esto tiene como beneficio flexibilidad para almacenar datos no estructurados, pero delega cualquier tipo de restricción a las aplicaciones que usan la base de datos.

  Entre los lenguajes que permiten describir datos semi-estructurados tenemos: *XML, JSON*.
- **Lenguajes de consulta menos poderosos**, la mayoría de aplicaciones NoSQL no requiere un lenguaje de consultas poderoso como SQL, ya que las consultas por lo general requieren un solo objeto en un archivo basándose en una `clave-id`.
- **Sistema de versiones**, los sistemas NoSQL permiten hacer múltiples versiones los ítems de datos.

3. Modelado de datos en Mongo DB

- El modelo de datos tradicional (relacional), es descrito en función de entidades, atributos y relaciones entre entidades, los datos en MongoDB son almacenados en documentos *JSON*. En los cuales los datos puedes ser embebidos.
- En los sistemas SQL se necesita la creación de un esquema, en MongoDB los datos son almacenados en documentos sin un esquema predefinido.

RDBMS	MongoDB
Tabla	Colección
Tupla/Fila	Documento
Columna	Campo
Clave primaria	Campo id por omisión en el documento
Indice	Indice

3.1. JSON

JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de *JavaScript* aunque hoy, debido a su amplia adopción como alternativa a *XML*, se considera un formato de lenguaje independiente.

- Esta compuesto de claves-valores (`Key:Value`).
- Los campos se separan usando comas.
- Soporta tipos de datos como cadenas, números, arreglos, objetos de datos.
- Los manejadores de MongoDB envian y reciben paquetes *BSON*.
- Del lado de la aplicación MongoDB mapea los paquetes BSON en JSON.
- Características de BSON:
  - **Liviano** ->Mínimo espacio.
  - **Traversable** ->Writing, reading, indexing.
  - **Eficiente** ->Codificación/Decodificación.

```
{
" headline " : " Venezuela. " ,
" date " : " 2019 -07 -05 t22 :35:21.908 Z " ,
" views " : 1132 ,
" autor " : {
" name " : " Base de Datos" ,
" title " : " Escuela de Ingeniera de sistemas "
},
" published " : true ,
" tags " :[ " estudios " ," pais " ," universidad "
   ]
}
```

4. Instalación MongoDB

Para la instalación en un sistema operativo basado en Debian debe ejecutar los siguientes comandos:

```
usuario@pc:~$ sudo aptitude install mongodb-org
```

```
usuario@pc:~$ sudo aptitude install mongodb-org-server
```

```
usuario@pc:~$ sudo aptitude install mongodb-org-mongos
```

```
usuario@pc:~$ sudo aptitude installmongodb-org-shell mongodb-org-tools
```

Para verificar la instalación puedes ejecutar los siguientes comandos:

```
usuario@pc:~$ sudo service mongod start
```

```
usuario@pc:~$ cat varlogmongodbmongod.log
```

En este punto debe aparecer un mensaje como el siguiente :

```
2019 -07 -05 T00 :14:32.197 -0400 I NETWORK
[ thread1 ] waiting for connections on port 27017
```

En este punto el servidor de Mongo estará funcionando y esperando conexiones por el puerto que se muestra en el comando anterior.

Para conectarse desde el terminal usamos el comando:

```
usuario@pc:~$ mongo
```

Los siguientes comandos nos serán de utilidad para realizar pruebas sobre los elementos de la base de datos:

- Mostrar las bases de datos:

```
> show dbs
```

- Usar una base de datos particular:

```
> use coleccion
```

- Una vez seleccionada la base de datos podemos empezar a ver información sobre las colecciones, con:

```
> db.documento.find ()
```

5. Creación de una base de datos y una colección a partir de un archivo BSON

Para crear una nueva base de datos, debemos estar en el cliente terminal de mongodb (mongo).

El siguiente comando crea una base de datos y una colección a la cual contendrá los archivos que se encuentra en archivo de extensión BSON:

```
> mongorestore -d db_name -c collection_name path/file.bson
```

En nuestro caso utilizaremos el siguiente comando, (el archivo **moviesScratch** esta adjunto dado en este laboratorio):

```
> mongorestore -d mi_db -c peliculas moviesScratch.bson
```

Para mostrar la información de esta colección:

```
> use mi_bd
```

```
> db.peliculas.find()
```

6. Insertar documentos

MongoDB proporciona distintas formas de insertar documentos, insertarUno, insertarMuchos (insertOne,insertMany), para nuestro ejemplo insertaremos en el terminal lo siguiente:

```
# Insertar un documento
db.peliculas.insertOne ({
" title " : " Guardianes de la galaxia 2 " ,
" year " : 2017 ,
" imdb " : " un codigo con formato " ,
" type " : " movie "
});
{
" acknowledged " : true ,
" insertedId " : ObjectId ( " 5 a 2 a 2 f e f 6 1 d a 5 d 4
    b a a 2 f b 4 b 3 " )
}

# Insertar varios documentos
db.peliculas.insertMany (
[
{
" _id " : " tt0084726 " ,
" title " : " Star Trek II : The Wrath of Khan " ,
" year " : 1982 ,
" type " : " movie "
},
{
" _id " : " tt0796366 " ,
" title " : " Star Trek " ,
" year " : 2009 ,
" type " : " movie "
}
],
{
" ordered " : false
```

```
}
);
{
" acknowledged " : true ,
" insertedIds " : [ " tt0084726 " , " tt0796366 " , ]
}
```

Para eliminar una colección utilizamos el comando `drop` como sigue:

```
> db.peliculas.drop()
```

7. Leyendo Documentos

Para esta sección utilizaremos el archivo **moviesDetails** BSON:

```
> mongorestore -d mi_db -c peliculasDetalles moviesDetails.bson
```

Para filtrar documentos según una clave el método principal es `find()`. Podemos buscar documentos y filtrar por clave-valor, de la siguiente manera:

```
> db.peliculasDetalles.find(director:Costa-Gavras)
```

Una función muy utilizada es `count()`, con la cual podemos contar la cantidad de registros que cuentan con cierta característica.

```
> db.peliculasDetalles.find(rated:null).count()
```

Para acceder objetos (data objets) utilizamos el operador punto ".". Por ejemplo, la siguiente consulta muestra todas las peliculas que han ganando un premio.

```
> db.peliculasDetalles.find(.awards.wins":1).pretty()
```

Para leer información filtrando por valores dentro de un arreglo. Existen 3 casos particulares: mediante un arreglo entero, mediante cualquier elemento de arreglo y mediante un elemento. especifico. La siguiente consulta encuentra un documento que contiene al array que se muestra exactamente igual y en orden:

```
> db.peliculasDetalles.find("writers":[.Ethan Coen","Joel Coen"]).pretty()
```

La siguiente consulta encuentra cualquier documento que tenga en el arreglo `actors` el nombre *Jeff Bridges*. Puede ser en cualquier posición:

```
> db.peliculasDetalles.find(actors: Jeff Bridges).pretty();
```

La siguiente consulta encuentra los documentos que contengan en la posicion 0 del arreglo de la clave especificada el nombre "Jeff Bridges".

```
> db.peliculasDetalles.find(actors.0 : Jeff Bridges).pretty();
```

MongoDB también cuenta con una operación muy importante conocida como proyección en el álgebra relacional. La proyección se realiza como sigue: Puedes usar exclusión o inclusión pero no ambas.

```
> db.peliculasDetalles.find(actors.0: Jeff Bridges,tomato:1, _id:0).pretty();
```

```
> db.peliculasDetalles.find(actors.0: Jeff Bridges,tomato:0, _id:0).pretty();
```

Nota: tenga en cuenta que los arreglos y búsquedas anidadas van entre comilla.