# **MongoDB Fundamentos**

## **Motores NoSQL**

NoSQL es una nueva forma de diseñar soluciones de base de datos a escala de Internet. No es un producto o tecnología, sino un término que define un conjunto de tecnologías de base de datos que no se basan en los principios tradicionales de los motores Relacionales.

En los últimos años con el auge de las aplicaciones web se recopilan cada vez más datos y cada vez más usuarios desean acceder a estos datos al mismo tiempo, lo que significa que la escalabilidad y el rendimiento se han convertido en un gran reto para las bases de datos relacionales basadas en esquemas.

Existen varios sistemas de bases de datos de código abierto siendo los más populares Hypertable, Cassandra, MongoDB, DynamoDB, Hbase y Redis.

### **Características**

* Alta escalabilidad
* Fácil gestion y administración
* Bajo costo
* Modelos de datos flexibles

## **MongoDB**

Es una base de datos **orientada a documentos.** Esta diseñado para trabajar con documentos sin necesidad de tener columnas predefinidas o tipos de datos, lo que hace que el modelo de datos sea extremadamente flexible.

Estos documentos se almacenan en un formato llamado JSON binario (también conocido como BSON). Los documentos pueden tener diferentes esquemas, lo que significa que el esquema puede cambiar a medida que la aplicación evoluciona. MongoDB está construido para escalabilidad, rendimiento y alta disponibilidad.

### **¿Cómo es un documento JSON?**

Un documento JSON siempre comienza y termina con llaves. Los campos y sus valores están separados por comas, con el nombre de campo siendo siempre un string y el valor de cualquier tipo de data que va de string, number, date, array, otro documento JSON, etc.

No proporciona soporte para operaciones JOIN, pero permite al usuario almacenar todos los datos relevante juntos en un solo documento.

MongoDB no proporciona soporte para transacciones de la misma manera que SQL. Sin embargo, garantiza la atomicidad a nivel de documento ya que utiliza un operador de aislamiento para operaciones de escritura que afectan a varios documentos.

### **¿Por qué usar MongoDB?**

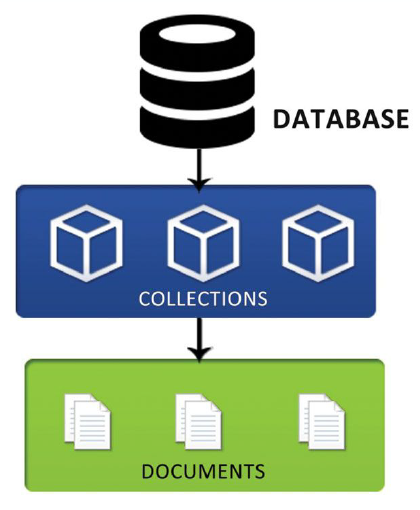
* Almacenamiento orientado a documentos.
* Índice sobre cualquier atributo.
* Replicación y alta disponibilidad.
* Auto-sharding
* Consultas enriquecidas.

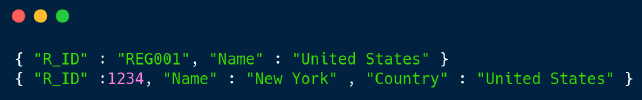
### **¿Dónde utilizar MongoDB?**

* Big Data
* Gestión de contenido y entrega
* Infraestructura móvil y social
* Gestión de datos de usuario
* Centro de datos

## **Modelado de Datos en MongoDB**

Una implementación de MongoDB puede tener muchas bases de datos. Cada base de datos es un conjunto de colecciones, las cuales no necesitan conocer con anterioridad el esquema (modelo de datos) y el mismo es **dinámico**. Cada colección puede tener múltiples documentos.

Esquema dinámico significa que los documentos dentro de la misma colección pueden tener conjuntos iguales o diferentes de campos o estructura, e incluso campos comunes pueden almacenar diferentes tipos de valores en los documentos.

Ejemplo:

Los documentos de una misma colección pueden tener esquemas completamente diferentes. Le corresponde a nuestra aplicación almacenar documentos en una colección particular juntos o para tener múltiples colecciones.

**Las colecciones son similares al concepto de tablas en RDBMS.**

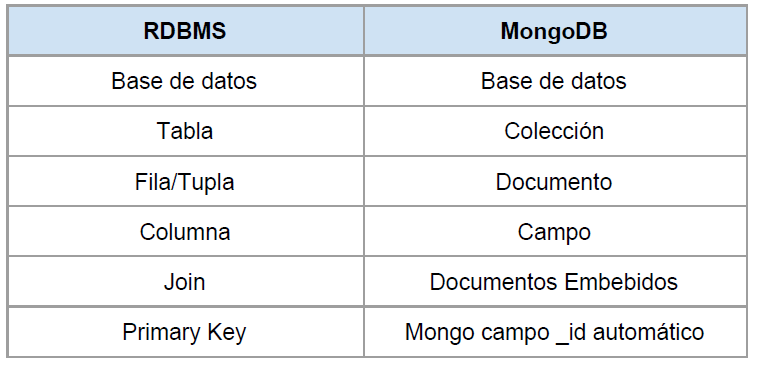
### **Identificador \_id**

Hemos visto que MongoDB almacena datos en documentos. Los documentos se componen de pares clave-valor. Una colección debe tener una clave que identifique de forma única cada documento. A esto nos referimos como identificador \_id en MongoDB. Si no se especifica un valor para una clave, se generará automáticamente un valor único por MongoDB. Este valor clave es inmutable y puede ser de cualquier tipo de datos, excepto los arrays.

### **Algunas consideraciones al diseñar el esquema en MongoDB**

* Diseña tu esquema de acuerdo a los requerimientos del usuario.
* Combine los objetos en un solo documento si los usará juntos. De lo contrario, sepárelos (pero asegúrese de que no haya necesidad de uniones).
* Duplique los datos (pero limitado).
* Optimice su esquema para los casos de uso más frecuentes.

## **RDBMS vs. MongoDB**



## **Componentes**

* Mongod (motor de base de datos)
* Mongo (cliente de línea de comandos)
* Mongo Compass (cliente gráfico)

Debemos ejecutar “mongod” para levantar el motor, el administrador de los datos. Cuando logremos levantarlo podemos conectarnos con “mongo” el cliente de línea de comandos para realizar operaciones en la base de datos.

## **Mongo Shell (Línea de comandos)**

Mongo Shell viene con la distribución estándar de MongoDB. Ofrece un entorno JavaScript con acceso completo al lenguaje y las funciones estándar.

Para ejecutarlo simplemente escribir “mongo” en una terminal.

**Comandos básicos**

show dbs //Muestra todas las bases de datos

use <dbname> //Creo una base nueva, si existe la usa

db.help() //Todos los comandos del objecto db

db.stats(); //Estadisticas de la base actual

db.hostInfo(); //Datos del host

## **Exportar e Importar Datos**

Su principal uso es el resguardo o backup de la información, pero también se utilizan para generar entornos de trabajo aislados de los productivos obteniendo datos reales para que el desarrollador pueda trabajar sin comprometer la operación del sitio o app.

//Importar desde JSON

mongoimport --db dbName --collection collectionName --file fileName.json -–jsonArray

mongoexport -d dbName -c collName --out fileName //Exporta un JSON

mongodump --db dbName //Backup en dir dump BSON

mongorestore //Restore desde dump

## **Comandos CRUD (ABM)**

Los comandos CRUD (Create, Read, Update, Delete) nos permiten manipular los datos de una colección.

**Crear bases de datos**

use db //Creo una base nueva o si existe la uso

**Crear colecciones**

db.createCollection(collName) //Crear colecciones.

show collections //Muestra las colecciones de una base.

**Crear Documentos en colecciones**

Las operaciones de crear o insertar agregan nuevos documentos a una colección. Si la colección no existe actualmente, las operaciones de inserción crearán la colección.

Métodos para insertar documentos en una colección:

db.col.insert() //Inserto un documento

db.col.insert([]) //Inserto múltiples documentos

db.col.save() //Otra sintaxis, más usada desde código

**Consultar Documentos**

db.col.find({criterio}) //Resultados

db.col.find({}).pretty() //Resultados con formato

db.col.find({$or:[{ "userId" :6},{ "userId" :7}]})

db.col.find({$and:[{ "userId" :6},{ "id" :53}]}) //AND

db.col.find({$and:[{ "userId" :6},{ "title" : /error/ }]}) //AND-LIKE

db.col.find({ "id" :{$gte:60}, "userId" :6}) //AND Mayor o igual que

**Proyección**

db.col.find({},{ "website" :1, "\_id" :0}) //Selecciono qué atributos mostrar

**Límite**

db.col.find({}).limit(n)

**Orden**

db.col.find({}).sort({ "website" :-1}); //Orden DESC

**Editar documentos**

db.col.update({criterio},{data}) //Actualiza un doc

db.col.update({criterio},{data},{multi: **true** }). //Actualiza multiples doc

**Reemplazar documentos**

db.col.save({ObjectId},{nuevo doc}) //Reemplaza todo el doc

**Borrar documentos**

db.col.remove({criterio}) //Elimina un elemento

## **Agregación**

Las operaciones de agregación procesan los registros de datos y devuelven los resultados calculados. MongoDB proporciona tres formas de realizar la agregación: el canal de agregación, la función de reducción de mapas y los métodos de agregación de propósito único.

### **Función aggregate()**

Nos permite filtrar un conjunto de datos y luego efectuar operaciones de agregación como $sum, $avg, $min, $max.

db.col.aggregate([

{$match:{key:value}},

{$group:{\_id, operator}}

]);

### **MapReduce**

En general, las operaciones de reducción de mapa tienen dos fases: una etapa que procesa cada documento y emite uno o más objetos para cada documento de entrada, y la fase de reducción que combina la salida de la operación de mapa.

Map-Reduce puede especificar una condición de consulta para seleccionar los documentos de entrada, así como clasificar y limitar los resultados. No es tan eficiente como el canal de agregación, esta pensada para un procesamiento batch.

### **Agregación de propósito único**

Todas estas operaciones agregan documentos de una sola colección. Si bien estas operaciones proporcionan un acceso simple a los procesos de agregación comunes, carecen de la flexibilidad y las capacidad de la canalización de agregación y reducen el mapa.

db.collection.count()

db.collection.distinct()

## **Indexación**

Los índices soportan la resolución eficiente de consultas. Sin índices, MongoDB debe escanear cada documento de una colección para seleccionar aquellos documentos que coincidan con la declaración de consulta.

Los índices son estructuras de datos especiales, que almacenan una pequeña parte del conjunto de datos en una forma fácil de recorrer. El índice almacena el valor de una campo específico o conjunto de campos.

db.col.ensureIndex({ campo: 1});

//Crea un índice en el campo con orden creciente

## **Ejecutar Javascript desde el motor**

MongoDB incluye un motor de Javascript el cual nos permite ejecutar código JS directo en MongoDB o podemos incluir un archivo externo.

### **Ejecutar código JS desde un archivo externo**

load(path) //Ejecuto un archivo externo

