Gestores de BD NoSQL

Aponte Roldán, Sigfredo (2016054468)), Gonzales Cave, Angel Gabriel (2017057861)), Pacora Silva, Jorge Carlos (xxxxxxxxxx)), Quispe Mamani, José Luis (xxxxxxxxxx))

Tacna, Perú

Abstract

EDITAR

1. Resumen

EDITAR

2. Introducción

EDITAR

- A
- B
- C

3. Marco Teórico

3.1. ***A***

3.1.1. ***A.1***

EDITAR [**?**]

3.1.2. ***A.2***

3.2. ***Base de datos NoSQL orientadas a Documentos***

Una base de datos orientada a documentos está diseñada para gestionar información orientada a documentos o datos semiestructurados. Tienen un grado de complejidad y flexibilidad superior a las bases de datos clave – valor. A diferencia de las conocidas bases de datos relacionales con su definición de "tabla", los sistemas documentales están diseñados entorno a la definición abstracta de un "documento". Un documento es la unidad principal de almacenamiento de este tipo de base de datos, y toda la información que aquí se almacena, se hace en formato de documento.

Este tipo de Base de Datos NoSQL almacena la información como un documento, generalmente utilizando para ello una estructura simple como JSON o XML y donde se utiliza una clave única para cada registro. Este tipo de implementación permite, además de realizar búsquedas por clave – valor, realizar consultas más avanzadas sobre el contenido del documento. [1]

3.2.1. ***Ventajas***

- Almacenar y recuperar todos los datos relacionados como una sola unidad puede entregar ventajas enormes en el rendimiento y la escalabilidad.

- Los gestores no tienen que hacer operaciones complejas como las uniones para encontrar los datos que normalmente están relacionados, ya que todo se encuentra en un mismo lugar.

[1]

3.3. ***MongoDB***

MongoDB es una base de datos NoSQL Open Source orientada a documentos y ha sido diseñada con la idea de que fuera fácil tanto desarrollar para ella como de ser administrada. Un documento es una estructura de datos compuesta de pares de campos y valores. Además, estos objetos son muy similares a lo que serían objetos en notación JSON.
Utilizar documentos tiene ciertas ventajas:

- Los documentos son tipos de datos nativos en muchos lenguajes de programación.

- Los documentos embebidos y los arrays minimizan la necesidad de realizar joins muy pesados.
- Tener un esquema dinámico es la base del polimorfismo.

[3]

3.3.1. ***Características generales***

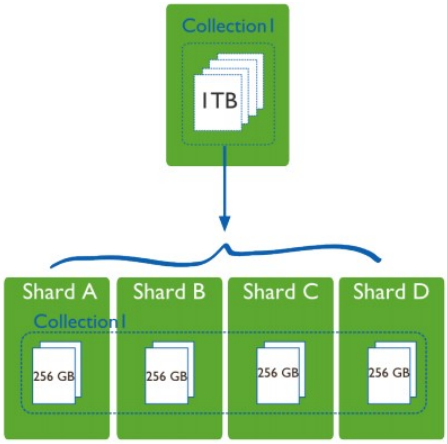
- **Indexado**
  MongoDB soporta índices secundarios, permitiendo una buena aceleración en muchas querys, pudiendo indexar cualquier campo de la base de datos.
- **Agregación**
  MongoDB viene con muchas funciones predefinidas que facilitan en gran medida el tratamiento de las colecciones almacenadas.
- **Replicación y Balanceo de Carga**
  MongoDB utiliza un sistema de replicación empleando la arquitectura maestro – esclavo con el resto de máquinas. Además, hace uso de un tipo de particionado horizontal llamado sharding. Estas dos herramientas de escalado hacen de mongo una herramienta poderosa cuando se trata de crecer mucho en poco tiempo.

  Mientras que la replicación busca realizar copias de los datos, repartirlos por nuevos hosts, permitiendo ofrecer alta disponibilidad, balanceo de carga y copias automáticas de seguridad, el sharding busca particionar esas tablas que, previsiblemente, se van haciendo más grandes y cada vez son más costosas de replicar.

[3]

3.3.2. ***Modelo de datos***

MongoDB es un tipo de base de datos de esquema flexible. Esto significa que ha de ser el usuario el que el que determina y declara el esquema de cada una de las tablas justo en el momento de realizar las inserciones. Por un lado se tiene la flexibilidad de añadir campos cuando son necesarios sin haberlos tenido que haberlos definido de antemano, o obviar algunos en los casos en los que no sean necesarios, pero a la vez se obliga al programador a ser mucho más metódico en su trabajo, puesto que el sistema no avisará de un posible



error o despiste en el código. Las bases de datos en MongoDB residen en un host que puede alojar varias bases de datos de forma simultánea almacenadas de forma independiente. Cada una de estas bases de datos puede contener una o más colecciones, a partir de las cuales se almacenarán los objetos o documentos.

3.3.3. ***BSON***

BSON es un formato binario que se utiliza para almacenar la información en MongoDB. BSON es la codificación binaria del formato JSON. Esta codificación ha sido elegida porque presenta ciertas ventajas a la hora de almacenar los datos, como la eficiencia o la compresión. Básicamente BSON y JSON son los formatos con los que trabaja MongoDB: JSON es el formato con el que se presenta la información a los usuarios y a las aplicaciones y BSON el formato que utiliza MongoDB de forma interna.

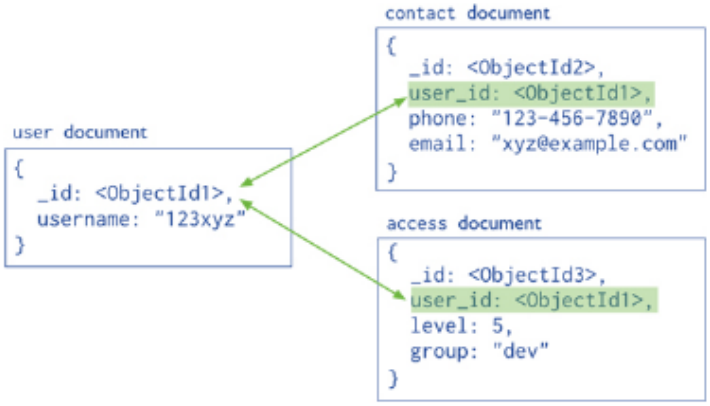
3.3.4. ***Estructura del documento***

El reto principal que tienen los documentos como objetos en MongoDB es el de realizar un buen diseño de los mismos para que sean capaces de representar lo más ampliamente el mundo real. Para ello existen 2 herramientas que permiten a las aplicaciones representar las relaciones entre los datos: las referencias y los datos embebidos. Una de las decisiones a tomar a la hora de implementar una aplicación que utilice MongoDB como capa de backend de aplicación es la de referenciar o embeber los documentos que se necesite.

3.3.5. ***Referencias***

Las referencias almacenan relaciones entre los datos mediante enlaces entre documentos. Los modelos de datos normalizados utilizan relaciones referenciales entre los documentos. Se usarán modelos de datos normalizados cuando:

- Embeber documentos produzca una duplicación de la información y los costes de mantenimiento sean mayores que las ganancias en lecturas.
- Sea necesario representar un modelo "varios a varios" con cierta complejidad.
- Para modelar conjuntos de datos de mucho tamaño.



[3]

3.3.6. ***Datos embebidos***

Los documentos embebidos representan relaciones entre los datos almacenando la información relacionada bajo el mismo documento. Los documentos permiten embeber estructuras documentales como subdocumentos dentro de un campo o de un array. En general, se utilizarán modelos embebidos cuando:

- Existan relaciones contenidas entre dos entidades. Estos es, que dos entidades independientes sean habitualmente accedidas a través de sólo una de ellas. Por ejemplo, Persona y Dirección. Cuando únicamente se

permitan búsquedas por persona para obtener su dirección. En ese caso, es preferible tener el objeto Dirección embebido dentro del de Persona.

- Cuando exista una relación "uno a varios" entre entidades. En este caso, la parte de "varios" suele ir embebida dentro de la de "uno". Los motivos son similares al anterior caso.

- En general, en ambos se trata de minimizar en la medida de los posible las operaciones de la base de datos. Embebiendo la información se consigue leer o escribir todos los datos simultáneamente y de forma secuencial.

[3]

4. Análisis (Aplicación)

4.1. ***Mongo DB***

Una de las razones principales porr la que los desarroladores usan MongoDB es debido a su modelo de datos intuitivo. MongoDB almacena su información en documentos en lugar de filas. ¿Qué es un documento? Aqui hay un ejemplo:

4.1.1. ***Esquema Básico***

Los productos y las categorías son los pilares de cualquier sitio de comercio electrónico. Los productos, en un modelo RDBMS(Sistema de gestión de bases de datos relacionales) normalizado, tienden a requerir una gran cantidad de tablas. Hay una mesa para información básica del producto, como el nombre y el código, pero habrá otras tablas para Relacionar la información de envío y los historiales de precios. Este esquema multitarea será facilitado por la capacidad del RDBMS para unir tablas. Modelar un producto en MongoDB debería ser menos complicado. Debido a que las colecciones no aplican un esquema, cualquier documento del producto tendrá espacio para cualquier atributos dinámicos que necesita el producto. Mediante el uso de matrices en su documento, puede normalmente condensa una representación RDBMS multitable en una sola colección MongoDB. [ecured] Más concretamente, la

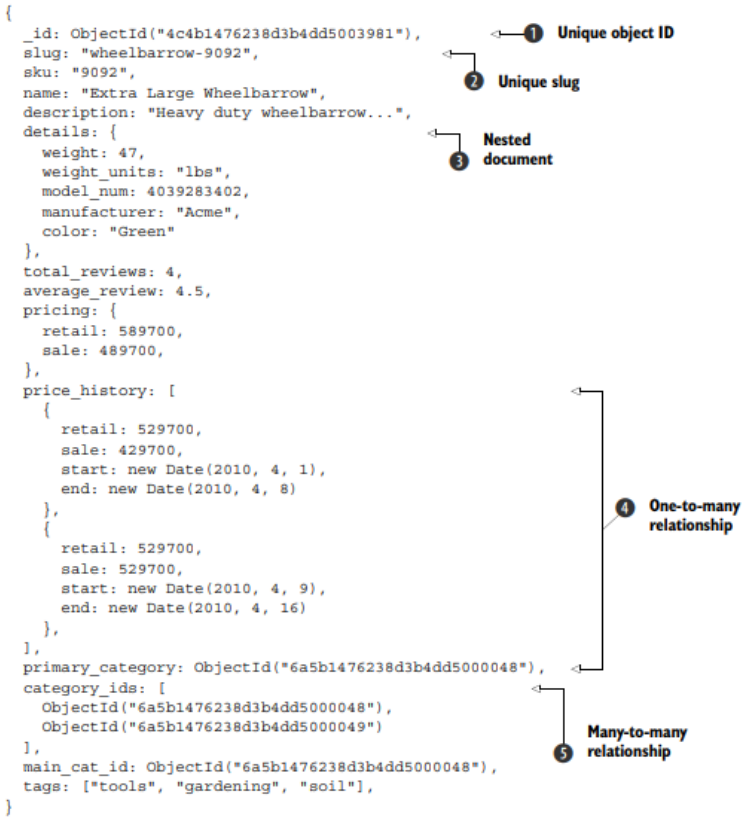


imagen muestra un producto de muestra de una tienda de jardinería. Es recomendable asignar este documento a una variable antes de insertarlo en la base de datos utilizando "db.products.insert (yourVariable)"para poder ejecutar las consultas discutidas durante las siguientes varias páginas.

4.1.2. ***Usuarios y ordenes***

Si observa cómo modela los usuarios y los pedidos, verá otra relación común: uno a muchos. Es decir, cada usuario tiene muchos pedidos. En un RDBMS(Sistema de gestión de bases de datos relacionales), usaría una clave foránea en su tabla de pedidos; aquí, la convención es similar.

4.1.3. ***Manejando Bases de Datos***

No existe una fomra explicita de vrear una base de datos en MongoDB. En cambio, una base de datos se crea automaticamente una vez que escribe en una coleecion en esa base de datos. A continuacion veremos un ejemplo:

Suponiendo que la base de datos no existe, la base de datos no se a creado en el disco luego de haber ejecutado el codigo mencionado. Todo lo que se

```
connection = Mongo::Client.new( [ '127.0.0.1:27017' ], :database => 'garden' )
db = connection.database
```

ha hecho es instanciar una instancia de la clase Mongo::DB, que representa una base de datos de MongoDB. Para crear nuestra base de datos se debera ejecutar el código siguiente: [4]

```
products = db['products']
products.insert_one({:name => "Extra Large Wheelbarrow"})
```

5. Conclusiones

- Conclusion 1 :
  Este artículo describe las limitaciones de las bases de datos tradicionales. y las ventajas de la base de datos NO SQL, análisis, objetivo de sus fortalezas y debilidades respectivamente, lo que ayudará al usuario a elegir entre una base de datos NO SQL y una base de datos tradicional.
- Conclusion 2 :
  B
- Conclusion 3 :
  C

Referencias

[1] Acens (2014). Bases de datos nosql. qué son y tipos que nos podemos encontrar. Recuperado de https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf. Accedido el 06-12-2019.

[ecured] ecured. Mongodb. Recuperado de ecured.cu. Accedido el 09-12-2019.

[3] Kyle, B. (2016). *MongoDB in Action*. Manning Publications Co.

[4] Raul, H. (2014). Bases de datos nosql: Arquitectura y ejemplos de aplicacion. Master's thesis, Universidad Carlos III, España. Accedido el 06-12-2019.