MongoDB

Manuel Molino Milla

13 de noviembre de 2017

Índice

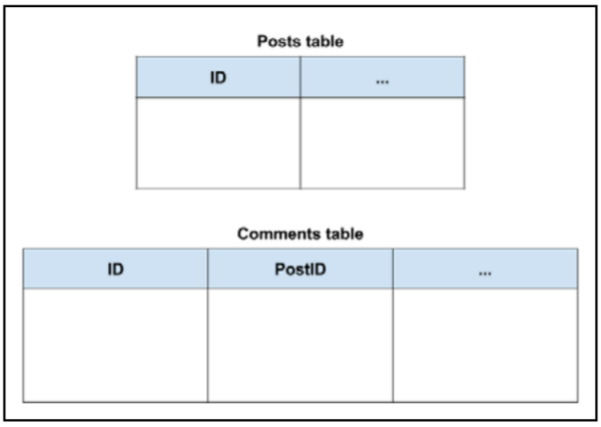
1. Introducción

1.1. NoSQL

- Es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales).
- No usan SQL como lenguaje principal de consultas.
- Los datos almacenados no requieren estructuras fijas como tablas.
- Los sistemas de bases de datos NoSQL crecieron con las principales redes sociales.
- Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares.
- Estas compañías se dieron cuenta de que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso.

1.2. SQL/NoSQL

En las bases de datos relacionales, los datos se almacenan en diferentes tablas, generelmente conectadas usando claves primarias y foráneas.
Usando lenguaje SQL insertas, recuperas, borras o actulizas datos.
En NoSQL se denominan base de datos orientadas a documentos, almacenados en formatos estándar tales como JSON o XML.
Ejemplo de BD SQL:



Ejemplo de BD NoSQL:

```
{
   "title": "First Blog Post",
   "comments": [......, ......]
}
```

Otro problema es la escalabilidad, *¿qué ocurre si añadimos una nueva propiedad?*

1.3. BD NoSQL

- MongoDB.
- Cassandra.
- Redis.
- CouchDB.

2. MongoDB

2.1. Introduccion

- Es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.
- MongoDB guarda estructuras de datos en documentos similares a JSON
- MongoDB utiliza una especificación llamada BSON (Binary JSON)
- El desarrollo de MongoDB empezó en octubre de 2007 por la compañía de software *10gen*.
- Los drivers para los lenguajes de programación están bajo la licencia de Apache.
- MongoDB está escrito en C++, aunque las consultas se hacen pasando objetos JSON como parámetro.

2.2. ¿Dónde se puede utilizar MongoDB?

- Cualquier aplicación que necesite almacenar datos semi estructurados puede usar MongoDB.
- Es el caso de las típicas aplicaciones CRUD de muchos de los desarrollos web actuales.
- MongoDB es especialmente útil en entornos que requieran escalabilidad.

2.3. ¿Dónde no se debe usar MongoDB?

- En esta base de datos no existen las transacciones. Solo garantiza operaciones atómicas a nivel de documento.Si las transacciones son algo indispensable en nuestro desarrollo, deberemos pensar en otro sistema.
- Tampoco existen los JOINS. Para consultar datos relacionados en dos o más colecciones, tenemos que hacer más de una consulta. En general, si nuestros datos pueden ser estructurados en tablas, y necesitamos las relaciones, es mejor que optemos por un RDBMS clásico.

3. Instalación

Instalación mongodb

4. Manejo MongoDB

4.1. Database

Un registro en MongoDB es un documento.

El cual es un estructura de datos compuesto por un campo y un valor, estos documentos son similares a los objetos JSON. Los valores de los campos pueden incluir otros documentos, arrays o arrays de documentos.

Los documentos son almacenados en colecciones:

Y las colecciones en base de datos

En MongoDB, las bases de datos almacenan colecciones de documentos.

Cada servidor mongo puede almacenar muchas bases de datos, por defecto se conecta a la base de datos denominada *test*
Para crear una nueva base de datos, en la shell de mongo hacemos:

use bd

Podemos entrar directamente con:

mongo bd

Podemos conocer las bases de datos del servidor con:

show dbs

Existen restricciones al nombre de las bases de datos

4.1.1. Documentos MongoDB

Los datos se almacenan en documentos *BSON* que es una representación binaria de documentos *JSON*


```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "
                   Turingery" ],
               views : NumberLong(1250000)
             }
```

En cuanto a tipos de datos
Restricciones en el nombre de los campos:

- El campo _id está reservado para us de *primary key*;
- El campo no puede empezar por $.
- Tampoco puede contener el punto (.)
- Ni ningún caracter nulo.

4.2. MongoDB CRUD

Las operaciones CRUD son *create, read, update y delete*

4.2.1. Create

Tenemos el método:

- db.collection.insert()

Insertando un registro

```
db.inventory.insert(
   { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w:
       35.5, uom: "cm" } }
)
```

4.2.2. Read

db.collection.find()


```
db.inventory.find( {} )
SELECT * FROM inventory

db.inventory.find( { status: "D" } )
SELECT * FROM inventory WHERE status = "D"

db.inventory.find( { status: { $in: [ "A", "D" ] } } )
SELECT * FROM inventory WHERE status in ("A", "D")

db.inventory.find( { status: "A", qty: { $lt: 30 } } )
SELECT * FROM inventory WHERE status = "A" AND qty < 30

db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ]
     } )
SELECT * FROM inventory WHERE status = "A" OR qty < 30

db.inventory.find( {
     status: "A",
     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
} )
SELECT * FROM inventory WHERE status = "A" AND ( qty < 30 OR item
    LIKE "p%")

db.inventory.find( { status: "A" }, { item: 1, status: 1 } )
SELECT _id, item, status from inventory WHERE status = "A"
```

```
db.inventory.find( { status: "A" }, { item: 1, status: 1, _id: 0 }
    )
SELECT item, status from inventory WHERE status = "A"
```

4.2.3. Update

- db.inventory.update()

```
db.users.update(
   { name: "xyz" },
   { name: "mee", age: 25, type: 1, status: "A", favorites: { "
       artist": "Matisse", food: "mango" } }
)
```

La sintáxis mas compleja es:

```
db.collection.update(
   <query>,
   <update>,
   {
     upsert: <boolean>,
     multi: <boolean>,
     writeConcern: <document>
   }
)
```

Donde:

query Es el criterio de selección.

update El modificador a aplicar.

upsert En caso de la opción *true*, si no hay ningún documento que cumpla las codificiones de búsqueda, se crea un nuevo documento.

multi En el caso de la opción *true* se modifica NO solo el primero que cuple la condición de búsqueda, se modifican TODOS.

Ejemplo:

```
db.people.update(
   { name: "Andy" },
   {
      name: "Andy",
      rating: 1,
      score: 1
   },
   { upsert: true }
)

db.collection.update( { "_id.name": "Robert Frost", "_id.uid": 0 },
   { "categories": ["poet", "playwright"] },
   { upsert: true } )


db.books.update(
   { _id: 1 },
   {
     $inc: { stock: 5 },
     $set: {
       item: "ABC123",
       "info.publisher": "2222",
       tags: [ "software" ],
       "ratings.1": { by: "xyz", rating: 3 }
     }
   }
)

db.mycol.update({'title':'MongoDB Overview'},
   {$set:{'title':'New MongoDB Tutorial'}},{multi:true})
```

El modificador *$inc* es de incremento y *$set* para la nueva condición

4.2.4. Delete

Borrado de una colección completa:

- db.collection.remove().

Borrado de un documento:

```
db.users.remove( { status: "D" }, 1)
```

4.2.5. Ejemplos de relacion entre SQL y NoSQL

Creacion de tablas:

```
CREATE TABLE people (
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id Varchar(30),
    age Number,
    status char(1),
    PRIMARY KEY (id)
)

db.people.insert( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )

db.createCollection("people") (se crea la coleccion vacia)
```

Añadimos una nueva columna a la tabla:

```
ALTER TABLE people
ADD join_date DATETIME

db.people.update(
  { },
  { $set: {join_date : new Date() }},
  {multi: true}
)
```

Borrado de una columna:

```
ALTER TABLE people
DROP COLUMN join_date

db.people.update(
  { },
  { $unset: {join_date : '' }},
  {multi: true}
)
```

Creación de índice:

```
CREATE INDEX idx_user_id_asc
ON people(user_id)

db.people.createIndex( { user_id: 1 } )

CREATE INDEX
       idx_user_id_asc_age_desc
ON people(user_id, age DESC)

db.people.createIndex( { user_id: 1, age: -1 } )
```

Borrado de una tabla

```
DROP TABLE people

db.people.drop()
```

Tutorial de tutorialpoins

4.2.6. Ejercicio

- Busca informacion de como importar una coleccion a una base de datos de *mongodb* a partir de un fichero conteniendo un array *json* (Usa el fichero *personas.json*)
- Incorpora un documento con tus datos personales.
- Busca todos los documentos relacionados con objetos que tenga el mismvo valor que tu edad.
- Busca todos los documentos que correspondan a persona mayores de edad.
- Ahora todos los documentos que correspondan a persona menores de edad y sexo femenino.
- Igual que antes y además que su nombre debe comenzar por A
- Igual que antes y además que su nombre debe comenzar por A o por F.
- Cambia de sexo al docuento que corresponde a tus datos personales.
- Cambia a sexo *Male* a todos los documentos que corresponde a menores de edad.
- Añade una nueva columna, denominada fecha y cuyo valor sea la fecha actual.
- Busca información de como exportar los datos de la colección anterior teniendo en cuenta:
  - Queremos guardar 100 documentos.
  - Para aquellos que sean mayores de edad.
  - Empezando por el registro 200