actibyti] a netmind social project

MongoDB

Autor: Ricardo Ahumada

GOBIERNO DE ESPAÑA
MINISTERIO DE ENERGÍA, TURISMO Y AGENDA DIGITAL
red.es
ESTRATEGIA DE EMPRENDIMIENTO Y EMPLEO JOVEN garantía juvenil
Agenda Digital para España
UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"

MongoDB

- **MongoDB** (https://www.mongodb.com/) es una base de datos orientada a documentos. Es decir que en lugar de guardar los datos en registros, guarda los datos en documentos.
- Esta base de datos open source toma su nombre del inglés humongous (gigantesco), y forma parte de la familia de sistemas NoSQL.
- Estos documentos son almacenados en BSON, que es una representación binaria de JSON.
- Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que **no es necesario seguir un esquema**.
- Los documentos de una misma **colección** - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

MongoDB: ejemplos de documentos

- Imaginemos que tenemos una colección a la que llamamos Personas.
- Podríamos almacenar documentos con estructuras diferentes para la misma colección:

```
{
 Nombre: "Pedro",
 Apellidos: "Martínez Campo",
 Edad: 22,
 Aficiones: ["fútbol","tenis","ciclismo"],
 Amigos: [
  {
   Nombre:"María",
   Edad:22
  },
  {
   Nombre:"Luis",
   Edad:28
  }
 ]
}
```

```
{
 Nombre: "Luis",
 Estudios: "Administración y Dirección de Empresas",
 Amigos:12
}
```

Cuando usar MongoDB

MongoDB es una buena alternativa para:

- Prototipos y aplicaciones simples
- Hacer la transición de front a back
- Aplicaciones con mucha carga de escritura
- Agregado de datos a un nivel medio/alto
- Aplicaciones con datos muy heterogéneos
- Enormes colecciones de datos (sharding)
- Almacenar ficheros (sharding)

Cuando no usar MongoDB

- **Mongo no puede hacer JOINs!**
- El lenguaje de consulta menos potente que SQL
- **No tiene transacciones!**
- La velocidad baja al subir la seguridad (escritura)
- Ten cuidado:
  - Es muy fácil empezar con MongoDB
  - Si tu app crece mucho… vas a necesitar JOINs

¿Quién usa MongoDB?

- En la sección **customers** del site (https://www.mongodb.com/) de MongoDB destacan las compañías que han depositado su confianza en una base de datos como MongoDB. Se dividen por secciones según el caso de uso:
  - Single View
  - Internet of Things
  - Mobile
  - Real-Time Analytics
  - Personalization
  - Content Management
  - Catalog
- Empresas como MetLife, Bosch, The Weather Channel, La ciudad de Chicago, Expedia, Forbes, son ejemplos de empresas que usan esta base de datos.

Instalando MongoDB

Instala MongoDB

- Descarga la última versión estable del Communitu Server desde https://www.mongodb.com/download-center?jmp=nav#community
- Instala MongoDB siguiendo los pasos del proceso
- Crea una carpeta donde quieras que se localice los archivos de tu base de datos.
  - Por ejemplo c:\mongodb\data

Inicia MongoDB

- Abre un terminal en la carpeta **bin** de instalación de mongo
  - Para iniciar mongo se sigue la siguiente orden:
    - mongod –dbpath <path> –nojournal
  - Por ejemplo:
    - ./mongod –dbpath "c:\mongodb\data " –nojournal

Accede con el cli de MongoDB

- Abre otro terminal de consola en la carpeta **bin** de instalación de mongo

- Ejecuta
  - mongo.exe
- Ya puedes comenzar a usar MongoDb

Usando MongoDB

- En el sitio de MongoDB tienes información sobre el uso de base de datos y colecciones usando el cli de MongoDB:
  - https://docs.mongodb.com/manual/core/databases-and-collections/

- Crea una base de datos
  - use myNewDB
  - db.myNewCollection1.insertOne({ x: 1 })

- Crea una colección
  - db.myNewCollection2.insertOne({ x: 1 })
  - db.myNewCollection3.createIndex({ y: 1 })

Ejemplo de carga de datos

- En el cli de MongoDB copia y pega:

```
db.nettuts.insert({
    first: 'matthew',
    last: 'setter',
    dob: '21/04/1978',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'developer',
    nationality: 'australian'
});

db.nettuts.insert({
    first: 'james',
    last: 'caan',
    dob: '26/03/1940',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'american'
});

db.nettuts.insert({
    first: 'arnold',
    last: 'schwarzenegger',
    dob: '03/06/1925',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'american'
});

db.nettuts.insert({
    first: 'tony',
    last: 'curtis',
    dob: '21/04/1978',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'developer',
    nationality: 'american'
});

db.nettuts.insert({
    first: 'jamie lee',
    last: 'curtis',
    dob: '22/11/1958',
    gender: 'f',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'american'
});

db.nettuts.insert({
    first: 'michael',
    last: 'caine',
    dob: '14/03/1933',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'english'
});

db.nettuts.insert({
    first: 'judi',
    last: 'dench',
    dob: '09/12/1934',
    gender: 'f',
    hair_colour: 'white',
    occupation: 'actress',
    nationality: 'english'
});
```

- Para recuperar los datos:

```
db.nettuts.find()
```

Documentos BSON

- Son objetos JSON con un ID de almacenamiento

```
{
  "_id" : ObjectId("524872a99c50880000000001"),
  "email" : "test@asdf.com",
  "password" : "asdf1234",
  "name" : "Test User",
  "date" : 1380479657300,
  "token" : "hm6ly43v.0o1or"
}
```

- Un documento puede contener arrays y otros documentos

```
{
  "_id" : ObjectId("5249a2e9b90687d56453b2f3"),
  "text" : "Soy un comentario",
  "user" : {
    "_id" : ObjectId("524872a99c50880000000001"),
    "nombre" : "Test User",
    "avatar" : "/img/as09a8sd09.jpg"
  },
  "tags" : [  "test",  "prueba" ]
}
```

Usando Mongo con Node.js / Instalación

- Node_redis, es el cliente de Redis para Node.js. Puede instalarlo en un proyecto node usando el npm:

```
npm install mongodb
```

- Una vez que haya instalado el modulo de mongodb ya estará listo para ser utilizado.

Conectarte al servidor

- En nuestra app.js nos podemos conectar indicado la url del servidor MongoDB y el nombre de la base de datos

```
var client = require('mongodb').MongoClient
  , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/<mydb>';

// Use connect method to connect to the server
client.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected successfully to server");

  db.close();
});
```

- En este caso assert sirve para verificar que no hay errores

Abrir una colección

- La BD está dividida en colecciones. Para abrir una colección:

```
var collection = client.then(function(db) {
                return db.collection("coleccion");
});
```

Colecciones

- Una colección es una agrupación de documentos:
  - Puede alojar cualquier documento (no impone estructura)
  - Puede alojar documentos con diferentes formas
  - Operaciones de consulta
  - Es donde se ponen los índices

Operaciones sobre una colección

- **collection.save**: guardar/actualizar un documento
- **collection.insert**: inserta un documento
- **collection.findOne**: recuperar un documento
- **collection.find**: recuperar varios documentos
- **collection.remove**: borrar uno o varios documentos
- **collection.drop**: elimina la colección
- **collection.rename**: cambia de nombre la colección
- **collection.count**: número de documentos

Colecciones desde Node.js

- Se realiza con el método collection:

```
client.then(function(db){
        var collection = db.collection("micoleccion");
        collection.insertAsync({ uno: 1, dos: 2})
        .then(function(){
                return collection.findOne({ uno: 1 })
                .toArrayAsync();
        });
});
```

Consulta de colecciones

- Dos operaciones fundamentales:
  - **findOne**: devuelve un documento (findOneAsync)
  - **find**: devuelve varios documentos en un cursor
- Tanto find como findOne aceptan criterios:
  - { clave: valor, clave: valor }
- Los operadores de búsqueda:
  - $gt / $gte: mayor/mayor o igual
  - $lt / $lte: menor/menor o igual
  - $ne: diferente
  - $in / $nin: en/no en array de valores

```
micol.findOne({ valor: {$in: [ 5, 15 ] }}, cb)
```

Consulta de colecciones

- Los operadores lógicos:
  - $or: se cumple alguna cláusula
  - $and: se cumplen todas las cláusulas
  - $nor: el resultado opuesto
  - $not: no se cumplen todas las cláusulas

```
micol.findOne({$or: [
                {valor: 5},
                {precio: {$gt: 15 }}
]}, callback)
```

Cursores: Operador find()

El operador find(…) devuelve un cursor

- Representa un conjunto de resultados
- cursor.count(callback): cantidad de documentos
- cursor.limit(n): limitar a n documentos
- cursor.skip(n): saltarse los n primeros documentos
- cursor.nextObject(callback): siguiente documento
- cursor.each(callback): para cada doc, en orden
- cursor.toArray(callback): convierte el cursor en array

Cursores: Operador sort()

cursor.sort(opciones, [callback])

- Ordenar los resultados
- Opciones del tipo:
  - [[“campo”, 1], [“otroCampo”, -1]]
  - 1 para ascendente, -1 para descendente

```
coleccion.find()
.sort([['a', -1]])
.nextObject(function(err, item) {
// ...
})
```

Añadir y modificar

- El operador más sencillo para modificar: save
  - Si el documento es nuevo (no tiene _id), lo inserta
  - Si el documento ya existe, lo modifica

```
db.micol.save({ nombre: "Test User" })
```

- insert(<documento o array>)
  - Inserta uno o varios documentos en la colección

```
db.micol.insert([
        { nombre: "Test User" },
        { nombre: "Test User 2" }
])
```

Insertar varios documentos

- Asimismo podemos insertar varios documentos con insertMany

```
var insertDocuments = function(db, callback) {
  // Get the documents collection
  var collection = db.collection('documents');
  // Insert some documents
  collection.insertMany([
    {a : 1}, {a : 2}, {a : 3}
  ], function(err, result) {
    assert.equal(err, null);
    assert.equal(3, result.result.n);
    assert.equal(3, result.ops.length);
    console.log("Inserted 3 documents into the collection");
    callback(result);
  });
}
```

Borrado de documentos

- remove(<patrón>)
  - Elimina los documentos que satisfagan la búsqueda

```
db.collection("customers").remove(myquery, function(err, obj) {
    if (err) throw err;
    console.log(obj.result.n + " document(s) deleted");
    db.close();
  });
```

Jugando con MongoDB

- Crea una web app Node que permita gestionar los datos de la base de datos **nettuts** (creada anteriormente)
- La App debe poder permiter consultar, añadir y borrar datos a través de un formulario

Persistencia de BananaTube con MongoDB

- Analiza en equipo si se podría usar MongoDB como base de datos de persistencia para BananaTube
- Que ventajas/desventajas presenta?
- Cubre todos los casos de uso posible?
- Compara estas conclusiones con Redis

BananaTube con Persistencia

- Decide en equipo qué base de datos usar para BananaTube
- Implementa la versión de API con persistencia

netmind

WeKnowIT

Barcelona

C. Almogàvers, 123
08018 Barcelona
Tel. 93 304.17.20
Fax. 93 304.17.22

Madrid

Plaza Carlos Trías Bertrán, 7
28020 Madrid
Tel. 91 442.77.03
Fax. 91 442.77.07

www.netmind.es

GOBIERNO DE ESPAÑA
MINISTERIO DE ENERGÍA, TURISMO Y AGENDA DIGITAL
red.es
ESTRATEGIA DE EMPRENDIMIENTO Y EMPLEO JOVEN garantía juvenil
Agenda Digital para España
UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"