Bancos orientados a documentos

João Pedro Schmitt

1

Modelos semi-estruturados

- Não existe separação entre o modelo e os dados;
- Vantagens:
  - Armazena informações não modeláveis por um esquema (Docx);
  - Flexível para ver dados estruturados em formatos não estruturados;
  - O esquema é atualizado facilmente;
- Desvantagens:
  - Performance;
  - Confiança;

2

Diferenças

- Bancos de documentos X bancos chave-valor;
  - Chave-valor: se apoia na chave para realizar buscas;
  - Documento: mantém um arquivo de metadados dos valores para realizar buscas;
- Bancos de documentos X bancos grafos;
  - Grafos, adicionam uma nova camada de associação entre as entidades;
- Bancos de documentos X bancos relacionais;
  - Relacional: dados segmentados em múltiplas tabelas;
  - Documentos: dados em um único documento;

3

Estrutura dos dados

- Armazenamento em documentos únicos;
- Codificadores/Decodificadores;
  - Converter o dado de entrada e saída;
  - Estruturar o documento para persistência;
- Codificações de formatos suportados:
  - XML;
  - YAML;
  - JSON;
  - BSON;

4

XML

- *eXtensible Markup Language*;
- Projeto para ser legível para humano e máquina;

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <name>Foo</name>
    <age>15</age>
    <height>1.78</height>
    <accountMoney>1.7E8</accountMoney>
    <male>true</male>
    <items>
        <element>Keys</element>
        <element>Glasses</element>
    </items>
    <address>
        <city>San Francisco</city>
        <state>CA</state>
    </address>
</root>
```

5

JSON

- *Java Script Object Notation*;
- Se baseia em duas estruturas:
  - Chave/valor (Ex: struct, hash tables)
  - Lista de valores (Ex: array, vector)
- Baseado na linguagem JavaScript;

```json
{
    "name": "Foo",
    "age": 15,
    "height": 1.78,
    "accountMoney": 1.7e8,
    "male": true,
    "items": ["Keys", "Glasses"],
    "address": {
        "city": "San Francisco",
        "state": "CA"
    }
}
```

6

BSON

- *Binary JSON*;
- Versão do JSON que permite dados binários;
- Permite o tipo *BinDataType*;
- Codificação usada pelo MongoDB;

```
{"hello":       // "\x16\x00\x00\x00\x02hello\x00
"world"}        //  \x06\x00\x00\x00world\x00\x00"

{"BSON":        // "\x31\x00\x00\x00\x04BSON\x00\x26\x00
["awesome",     //  \x00\x00\x020\x00\x08\x00\x00
5.05,1986]}     //  \x00awesome\x00\x011\x00\x33\x33\x33\x33\x33\x33
                //  \x14\x40\x102\x00\x00\xc2\x07\x00\x00
                //  \x00\x00"
```

7

Campos dinâmicos

- Não requer que todos documentos compartilhem a mesma estrutura;
- Permite adicionar campos dinâmicamentes;

```
// Document 1
{
    "name": "Foo",
    "age": 18
}

// Document 2
{
    "name": "Bar",
    "height": 1.78,
    "weight": 78
}
```

8

Características

- Chaves únicas, cada documento deve possuir um;
- Recuperação, uso de metadados para realizar as buscas;
- Atualização, substituição completa do documento ou modificações parciais;

9

Ranking das bases de documentos

Rank			DBMS	Database Model	Score		
May 2019	Apr 2019	May 2018			May 2019	Apr 2019	May 2018
1.	1.	1.	MongoDB	Document	408.07	+6.10	+65.96
2.	2.	2.	Amazon DynamoDB	Multi-model	55.93	-0.08	+11.74
3.	3.	3.	Couchbase	Document	34.67	-1.61	+2.26
4.	4.	↑ 5.	Microsoft Azure Cosmos DB	Multi-model	27.59	+1.32	+10.06
5.	5.	↓ 4.	CouchDB	Document	19.11	-1.32	-0.31
6.	6.	6.	MarkLogic	Multi-model	14.05	-0.42	+3.66
7.	7.	7.	Firebase Realtime Database	Document	11.28	+0.28	+4.79
8.	8.	8.	OrientDB	Multi-model	6.37	+0.18	+1.12
9.	↑ 11.	↑ 16.	Google Cloud Firestore	Document	4.99	+0.68	+2.76
10.	↑ 12.	↑ 11.	ArangoDB	Multi-model	4.79	+0.50	+1.09

10

MongoDB

11

MongoDB

- Base de dados orientada a documentos;
- Community Edition: https://github.com/mongodb/mongo/
- Enterprise Edition. Oferece suporte, autenticação LDAP e Kerberos, criptografia e auditoria;
- Alta disponibilidade: failover automatico e redundância;
- Escalabilidade horizontal: sharding e zonas;

12

Instalação

- Download do site do MongoDB
  - https://fastdl.mongodb.org/win32/mongodb-win32-x86_64-2008plus-ssl-4.0.9.zip;
- Descompactar localmente;
- Executáveis principais: “C:\programas\mongodb\bin”;
  - mongod - servidor da base de dados;
  - mongos - roteador dos shardings;
  - mongo - JavaScript shell iterativo;
- Executáveis auxiliares: mongodump, mongorestore, mongoexport, mongoimport, mongofiles, mongostat, bsondump, mongoreplay, mongotop, install_compass;

13

Incialização

- Listar todos comandos disponíveis:
  - C:\programas\mongodb\bin>mongod.exe --help
- Criar as pastas data\db em C:\programas\mongodb\data\db
- Iniciar o banco na pasta definida:
  - C:\programas\mongodb\bin>mongod.exe --dbpath C:\programas\mongodb\data\db --bind_ip 0.0.0.0 --port 27017
- Abrir uma nova CMD e iniciar o shell:
  - C:\programas\mongodb\bin>mongo.exe
- Mostrar todas bases: show dbs

14

Operações de CRUD

15

Inserção

- Inserir vários registros:

```
db.inventory.insertMany([
  { item: "journal", qty: 25, status: "A", size: { h: 14, w: 21, uom: "cm" }, tags: [ "blank", "red" ] },
  { item: "notebook", qty: 50, status: "A", size: { h: 8.5, w: 11, uom: "in" }, tags: [ "red", "blank" ] },
  { item: "paper", qty: 100, status: "D", size: { h: 8.5, w: 11, uom: "in" }, tags: [ "red", "blank", "plain" ] },
  { item: "planner", qty: 75, status: "D", size: { h: 22.85, w: 30, uom: "cm" }, tags: [ "blank", "red" ] },
  { item: "postcard", qty: 45, status: "A", size: { h: 10, w: 15.25, uom: "cm" }, tags: [ "blue" ] }
]);
```

- Inserir um registro:

```
db.inventory.insertOne({ item: "book", qty: 5, status: "A", size: { h: 9.5, w: 10, uom: "in" }, tags: [ "green", "blank" ] })
```

16

Update / Delete

- Atualização de documentos [coleção | filtro | atualização]:
  - db.inventory.update({ item: "book" }, { item: "ebook" })
- Deletar vários documentos [coleção | filtro]:
  - db.inventory.deleteMany({ item: "ebook" })

17

Seleção

- Buscar todos documentos:
  - db.inventory.find({})
- Buscar documento por um valor (exato):
  - db.inventory.find({ status: "D" })
- Buscar por níveis diferentes (exato):
  - db.inventory.find({ size: { h: 14, w: 21, uom: "cm" }})
- Buscar baseada em elementos de listas:
  - db.inventory.find({ tags: "red" })

18

Filtros

- Múltiplas propriedades são implicitamente conjunções (AND):
  - db.inventory.find({ status: "A", qty: { $lt: 30 } })
- Operador OR:
  - db.inventory.find({ $or: [{ status: "A" }, { qty: { $lt: 30 } }] })
- Documentos embarcados:
  - db.inventory.find({ "size.h": { $lt: 15 } })
- Busca em lista (exato em mesma ordem e exato em qualquer ordem)
  - db.inventory.find({ tags: ["red", "blank"] })
  - db.inventory.find({ tags: { $all: ["red", "blank"] } })
- Lista de operadores: https://docs.mongodb.com/manual/reference/operator/query/

19

Projeção

- Campos a serem projetados [coleção | filtro | projeção]:
  - db.inventory.find({ status: "A" }, { item: 1, status: 1 })
- Ignorar campos:
  - db.inventory.find({ status: "A" }, { item: 1, status: 1, _id: 0 })
- Projetar campos de documentos embarcados:
  - db.inventory.find({ status: "A" }, { item: 1, status: 1, "size.uom": 1 })
- Projetar última posição de arrays:
  - db.inventory.find({ status: "A" }, { item: 1, status: 1, "tags": { $slice: -1 }})

20

Cursores

- Iterar por um cursor:

```
var myCursor = db.inventory.find({});
while (myCursor.hasNext()) {
  print(tojson(myCursor.next()));
}
```

- Obter array de um cursor:

```
var myCursor = db.inventory.find({});
var documentArray = myCursor.toArray();
var myDocument = documentArray[3];
printjson(myDocument);
```

21

Buscas textuais

- Dados:

```
db.stores.insert([
        { _id: 1, name: "Java Hut", description: "Coffee and cakes" },
        { _id: 2, name: "Burger Buns", description: "Gourmet hamburgers" },
        { _id: 3, name: "Coffee Shop", description: "Just coffee" },
        { _id: 4, name: "Clothes Clothes Clothes", description: "Discount clothing" },
        { _id: 5, name: "Java Shopping", description: "Indonesian goods" }
])
```

- Necessário um índice para o campo:
  - `db.stores.createIndex( { name: "text", description: "text" } )`
- Ou lógico por palavras tokenizando os textos por espaço e pontuações:
  - `db.stores.find( { $text: { $search: "java coffee shop" } } )`
- Excluir termo:
  - `db.stores.find( { $text: { $search: "java shop -coffee" } } )`

22

Buscas Geoespaciais

- Formato GeoJSON: <field>: { type: <GeoJSON type> , coordinates: <coordinates> }

```
db.places.insert({name:"Central Park",loc:{type:"Point",coordinates:[-73.97,40.77]},category:"Parks"});
db.places.insert({name:"Roosevelt Park",loc:{type:"Point",coordinates:[-73.9928,40.7193]},category:"Parks"});
db.places.insert({name:"Polo Grounds",loc:{type:"Point",coordinates:[-73.9375,40.8303]},category:"Stad."});
```

- Criar um índice de suporte das buscas geoespaciais:
  - db.places.createIndex({ loc: "2dsphere" })
- Buscar pontos ao menos 1000 e no máximo 5000 metros de distância:

```
db.places.find({ loc: {
      $near: {
      $geometry: { type: "Point",  coordinates: [ -73.9667, 40.78 ] },
      $minDistance: 1000,
      $maxDistance: 5000
   }
}})
```

23

Agregação

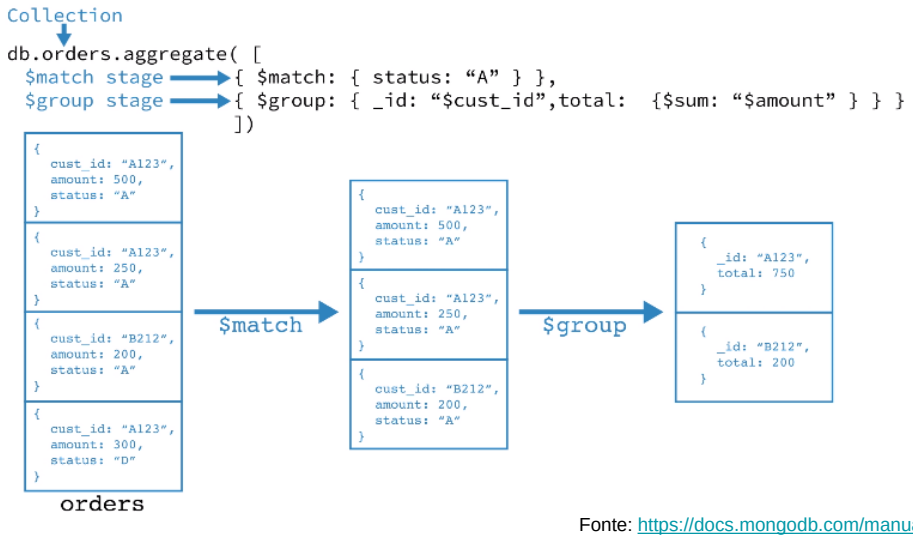
24

Agregação

- Consolidar registros da base de dados; Combinando diferentes esquemas, campos, entre outros.
- Três tipos de agregação:
  - Pipeline de agregação;
  - Map-reduce;

25

Aggregation



Fonte: https://docs.mongodb.com/manual/aggregation/

26

Aggregation

- Dados:

```
db.orders.insertMany([
        { cust_id: "A123", amount: 500, status: "A" },
        { cust_id: "A123", amount: 250, status: "A" },
        { cust_id: "B212", amount: 200, status: "A" },
        { cust_id: "A123", amount: 300, status: "D" }
])
```

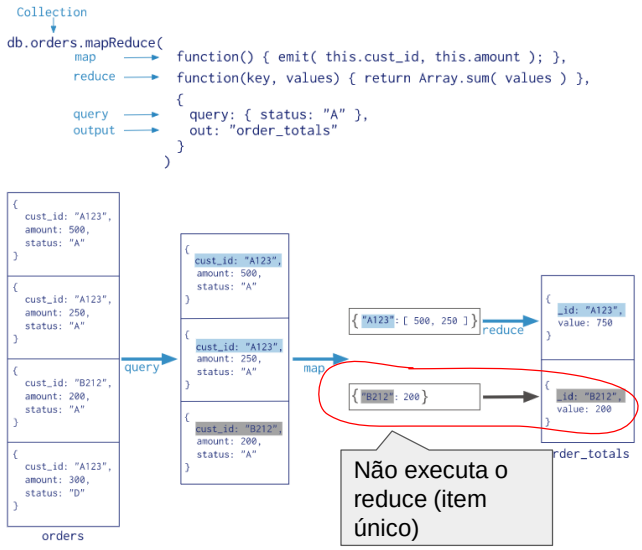
- Agregação - query 1:

```
db.orders.aggregate([
  { $match: { status: "A" } },
  { $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
  { $project: { summary: { $concat: ["$_id", " = ", { $substr : ["$total", 0, -1] } ] } } },
  { $sort: { "_id": 1 } }
])
```

27

Map-reduce

- Função map - agrupa os valores do campo “amount” por ID;
- Reduce -> sumariza os valores para cada entrada (somente para entradas com mais de um item);



28

Map-reduce

- Dados:

```
db.orders.insertMany([
        { cust_id: "A123", amount: 500, status: "A" },
        { cust_id: "A123", amount: 250, status: "A" },
        { cust_id: "B212", amount: 200, status: "A" },
        { cust_id: "A123", amount: 300, status: "D" }
])
```

- Map-reduce:

```
db.orders.mapReduce(
        function() { emit(this.cust_id, this.amount); },
        function(key, values) { return Array.sum(values); },
        {
                query: { status: "A" },
                out: "order_totals"
        }
).convertToSingleObject();
```

29

MongoDB - Java (SpringBoot)

30

Configurando projeto Java - Maven - SpringBoot

- Java 1.8+
- Baixar o maven: https://maven.apache.org/download.cgi
- Extrair na pasta C:\programas\maven
- Abrir o CMD e setar a variável de ambiente:
  - set MVN_HOME=C:\programas\maven\bin
- Testar: mvn -v
- Criar diretório para o projeto (ex: C:\projects\spring-boot-mongo);

31

pom.xml

```
└── src
    └── main
        └── java
            └── hello
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XI
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.springframework</groupId>
    <artifactId>gs-accessing-data-mongodb</artifactId>
    <version>0.1.0</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.4.RELEASE</version>
    </parent>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

32

src/main/java/hello/Customer.java

```java
package hello;

import org.springframework.data.annotation.Id;


public class Customer {

    @Id
    public String id;

    public String firstName;
    public String lastName;

    public Customer() {}

    public Customer(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return String.format(
                "Customer[id=%s, firstName='%s', lastName='%s']",
                id, firstName, lastName);
    }

}
```

33

src/main/java/hello/CustomerRepository.java

```java
package hello;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;

public interface CustomerRepository extends MongoRepository<Customer, String> {

    public Customer findByFirstName(String firstName);
    public List<Customer> findByLastName(String lastName);

}
```

34

src/main/java/hello/Application.java

```java
package hello;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application implements CommandLineRunner {

	@Autowired
	private CustomerRepository repository;

	public static void main(String[] args) {
		SpringApplication.run(Application.class, args);
	}

	@Override
	public void run(String... args) throws Exception {

		repository.deleteAll();

		// save a couple of customers
		repository.save(new Customer("Alice", "Smith"));
		repository.save(new Customer("Bob", "Smith"));

		// fetch all customers
		System.out.println("Customers found with findAll():");
		System.out.println("-------------------------------");
		for (Customer customer : repository.findAll()) {
			System.out.println(customer);
		}
		System.out.println();

		// fetch an individual customer
		System.out.println("Customer found with findByFirstName('Alice'):");
		System.out.println("--------------------------------");
		System.out.println(repository.findByFirstName("Alice"));

		System.out.println("Customers found with findByLastName('Smith'):");
		System.out.println("--------------------------------");
		for (Customer customer : repository.findByLastName("Smith")) {
			System.out.println(customer);
		}

	}

}
```

35

Executar

- Iniciar mongodb localmente
- Executar o comando: mvn spring-boot:run

```
Customers found with findAll():
-------------------------------
Customer[id=5cec7b9c264b1d3fac9b5e73, firstName='Alice', lastName='Smith']
Customer[id=5cec7b9c264b1d3fac9b5e74, firstName='Bob', lastName='Smith']

Customer found with findByFirstName('Alice'):
--------------------------------
Customer[id=5cec7b9c264b1d3fac9b5e73, firstName='Alice', lastName='Smith']
Customers found with findByLastName('Smith'):
--------------------------------
Customer[id=5cec7b9c264b1d3fac9b5e73, firstName='Alice', lastName='Smith']
Customer[id=5cec7b9c264b1d3fac9b5e74, firstName='Bob', lastName='Smith']
[INFO]
```

36

Exercícios

37

Prática

- Clonar repositório: https://github.com/schmittjoaopedro/udesc;
- Terminar de programar métodos em aberto na classe: <dir>/src/main/java/com/github/schmittjoaopedro/App.java

```
: ================ Creating orders
: Created order A123
: Created order A124
: ================ Counting orders
: ================ Updating order
: ================ Finding all orders
: ================ Finding all orders from Jaraguß do Sul
: ================ Finding all orders with items price greater than 55.0
: ================ Show order description
: ================ Using cursors with flux
: ================ Finishing application
```

38

Referências

- https://en.wikipedia.org/wiki/Document-oriented_database
- https://www.json.org/
- https://www.w3schools.com/xml/
- https://db-engines.com/en/ranking/document+store
- https://en.wikipedia.org/wiki/Semi-structured_model
- https://docs.mongodb.com/manual
- https://spring.io/guides/gs/accessing-data-mongodb/

39