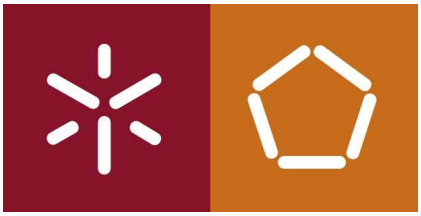

Universidade do Minho
Mestrado Integrado em Engenharia Informática
4ºano - 1º Semestre

Base de Dados NoSQL

Ficha de Exercícios 04

Grupo 01




a83899 – André Loureiro Morais
a76089 - Etienne da Silva Filipe Amado da Costa
a83732 – Gonçalo Rodrigues Pinto
a85954 – Luís Mário Macedo Ribeiro

18 de Novembro de 2020

Após instalar o MongoDB, utilizou-se a *mongoshell* para executar os seguintes exercícios…

1. Customers

[1] Listar todas as bases de dados após a instalação do container com a imagem do MongoDB.

```
> show dbs
```

[2] Criar uma base de dados denominada "*customers*".

```
> use customers
```

[3] Verificar a criação da base de dados.

```
> db
```

[4] Criar uma coleção denominada "customers".

```
> db.createCollection("customers")
```

[5] Validar a criação da coleção.

```
> show collections
```

[6] Criar um cliente com os seguintes características: **first_name: "John", last_name:"Doe", age: 30**

```
> db.customers.insert( { first_name: 'John', last_name: 'Doe', age: 30 } )
```

[7] Introduzir 2 clientes na coleção criada com as seguintes características: **first_name: "Steven", last_name:"Williams", gender: "male" ; first_name: "Mary", last_name: "Troy", age: 19**

```
> db.customers.insert( { first_name: 'Steven', last_name:'Williams', gender: 'male'})
> db.customers.insert( { first_name: 'Mary', last_name: 'Troy', age: 19})
```

[8] Introduzir mais um cliente com as seguintes características: **first_name: "Ric", last_name: "Foe", address: {street: "4 main st", city: "Boston"}**

```
> db.customers.insert( { first_name: 'Ric', last_name: 'Foe', address: {street: '4 main st', city: 'Boston'} } )
```

[9] Criar um cliente com as seguintes características: **first_name: "Ana", last_name: "Durant", degree:["phD","Msc"], address: {street: "4 Square Garden", city: "New York"}, age: 32**

```
> db.customers.insert( { first_name: 'Ana', last_name: 'Durant', degree: ['phD','Msc'], address: {street: '4 Square Garden', city: 'New York'}, age: 32 })
```

[10] Criar um cliente com as seguintes características: **first_name: “Natalia”, last_name:”Will”, age: 44, gender: “female”**

> db.customers.insert({ first_name: 'Natalia', last_name:'Will', age: 44, gender: 'female' })

[11] Listar todos os clientes.

> db.customers.find()

[12] Listar todos os clientes usando a função pretty().

> db.customers.find().pretty()

[13] Efetuar uma atualização ao cliente ‘Ric’, colocar idade 45.

> db.customers.update({first_name: 'Ric'},{$set : {age:45}})

[14] Encontrar todos os clientes que tenham ‘Will’ no último nome.

> db.customers.find({last_name: {$regex: 'Will', $options: "$i"}})

[15] Efetuar uma atualização ao cliente ‘Steven’, colocar idade 35.

> db.customers.update({first_name: 'Steven'},{$set : {age:35}})

16] Verificar se a idade da cliente ‘Ana’ é superior a 30 e se sim aumentar a idade em 10 anos.

> db.customers.update({first_name: 'Ana',age: {$gt: 30}},{$inc: {age:10}})

[17] O cliente ‘Ric’ quer que a sua idade seja removida da base de dados.

> db.customers.update({first_name: 'Ric'},{$unset : {age:''}})

[18] Procurar um cliente com o primeiro nome: "Jimmy" e atualizar, ou criar, caso não exista, com as seguintes características.: **first_name: "Jimmy", last_name: "Connors", age: 25, gender: male**

> db.customers.update({first_name: 'Jimy'},{$set: {first_name: 'Jimmy', last_name: 'Connors', age: 25, gender: 'male'}},{upsert:true})

[19] Procurar todos os clientes com idade superior ou igual a 25.

> db.customers.find({age: {$gt: 25}})

[20] Procurar todos os clientes sexo masculino.

> db.customers.find({gender: 'male'})

[21] Apagar o cliente cujo primeiro nome é "Mary".

> db.customers.remove({first_name: 'Mary'})

[22] Encontrar os clientes com o nome "Ana" ou "Ric".

> db.customers.find({$or :[{first_name: 'Ana'},{first_name: 'Ric'}]})

---

2. restaurants.json

Após importar o ficheiro "restaurants.json" utilizando o Compass, foi possível responder ás seguintes questões:

[1] liste todos os documentos na coleção *restaurants* .

> db.restaurants.find()

[2] liste apenas os campos *restaurante_id* , *name* , *borough* e *cuisine* para todos os documentos na coleção.

> db.restaurants.aggregate({ $project:{address: 0, grades:0}})

[3] liste os campos *restaurante_id* , *name* , *borough* e *cuisine* para todos os documentos na coleção, mas que exclua o campo *_id* .

> db.restaurants.aggregate({ $project:{_id:0, address: 0, grades:0}})

[4] liste os campos *restaurante_id* , *name* , *borough* e *zipcode* para todos os documentos na coleção, mas que exclua o campo *_id* .

> db.restaurants.aggregate({$project:{_id:0, restaurant_id:1, name:1, borough:1, "address.zipcode":1}})

[5] liste os restaurantes que estão localizados no bairro (*borough*) "Bronx".

> db.restaurants.find({borough:'Bronx'})

[6] liste os primeiros 5 restaurantes que estão localizados no bairro (*borough*) "Bronx".

> db.restaurants.find({borough:'Bronx'}).limit(5)

[7] liste os 5 restaurantes após dos primeiros 5 (do 6º ao 10º) que estão localizados no bairro (*borough*) "Bronx".

> db.restaurants.find({borough:'Bronx'}).skip(5).limit(5)

[8] liste todos os restaurantes que têm pelo menos uma pontuação (*score*) maior que 90.

> db.restaurants.find({"grades.score": {$gt: 90}})

[9] liste todos os restaurantes que têm uma pontuação (*score*) maior que 80 mas menor que 100.

> db.restaurants.find({$and :[{"grades.score": {$gt: 80}}, {"grades.score": {$lt: 100}}]})

[10] liste todos os restaurantes que estão localizados numa latitude (*coord.0*) menor que -95.754168.

> db.restaurants.find({"address.coord.0": {$lt: -95.754168}})

[11] recorrendo à mongo shell atualize todos os restaurantes que possuam a cozinha "American " para "American".

> db.restaurants.update ({cuisine:'American '}, {$set: {cuisine:'American'}}, {multi:true})

[12] liste todos os restaurantes cujo tipo de cozinha (*cuisine*) não seja "American", que a sua pontuação (*score*) seja maior que 70 e a latitude (*address .coord.0*) menor que -65.754168, utilizando o operador $and.

> db.restaurants.find({$and :[{"grades.score": {$gt: 70}}, {"address.coord.0": {$lt: -65.754168}}, {cuisine:{$ne: 'American'}}]})

[13] liste todos os restaurantes cujo tipo de cozinha (*cuisine*) não seja "American", que a sua pontuação (*score*) seja maior que 70 e a latitude (*address .coord.0*) menor que -65.754168.

> db.restaurants.find({"cuisine":{$ne: "American"}, "grades.score" : { $gt : 70} , "address.coord.0" : {$lt: -65.754168}})

[14] liste todos os restaurantes cujo tipo de cozinha (*cuisine*) não seja do tipo "American" e que tenham atingido uma classificação (*grade*) de "A" mas que não pertençam ao bairro (*borough*) de "Brooklyn". Deverá ser apresentada de acordo com o tipo de cozinha (*cuisine*) em ordem descendente.

> db.restaurants.find({$and :[{"grades.grade": 'A'}, {cuisine:{$ne: 'American'}}, {borough:{$ne: 'Brookly'}}]}).sort({cuisine: -1})

[15] liste todos os restaurantes que pertençam ao bairro (*borough*) "Bronx" e cujo tipo de cozinha (*cuisine*) seja quer "American" quer "Chinese".

> db.restaurants.find({$and :[{borough:'Bronx'}, {$or :[{cuisine: 'American'},{cuisine: 'Chinese'}]}]})

[16] liste todos os restaurantes cujas coordenadas (*address.coord*) sejam do tipo double (type: 1).

> db.restaurants.find({"address.coord" : {$type : "double"}})

[17] liste todos os restaurantes que contenham informação da rua (address.street).

> db.restaurants.find({"address.street": {$exists: true}})

[18] liste todos os restaurantes de forma ascendente pelo tipo de cozinha (*cuisine*) e descendente pelo bairro (*borough*).

> db.restaurants.find().sort({cuisine: 1},{borough: -1 })

[19] liste o restaurant_id, name, address e localização geográfica (*coord*) para os restaurantes cujo segundo elemento do array da localização geográfica (*coord*) seja maior que 42 e até 52.

> db.restaurants.aggregate([{$project:{_id:0, restaurant_id:1,name:1, address:1}}, {$match:{$and :[{"address.coord.1": {$gt: 42}}, {"address.coord.1": {$lt: 52}}]}}])

[20] liste os restaurantes (*restaurante_id* , *name* , *borough* , *cuisine*) que não conseguiram uma pontuação (score) maior que 10.

> db.restaurants.aggregate([{$project:{_id:0, restaurant_id:1,name:1, borough:1, cuisine:1}}, {$match: {"grades.score": {$lt: 10}} }])

[21] liste todos os restaurantes (*restaurante_id* , *name* , *borough* e *cuisine*) que não pertencem ao bairro (*borough*) de "Staten Islan", ou "Queens" ou "Bronx" ou "Brooklin".

> db.restaurants.aggregate([{$project:{_id:0, restaurant_id:1,name:1, borough:1, cuisine:1}}, {$match: {$or :[{borough:{$ne: 'Staten Islan'}},{borough:{$ne: 'Queens'}},{borough:{$ne: 'Bronx'}},{borough:{$ne: 'Brooklin'}}]} }])