============================================

bd

.\mongod --dbpath \Users\"Charles Antunes"\Documents\Alura\meus-projetos-alura\projetos\_aulas\29-mongo\_db\db

============================================

bin

C:\Users\Charles Antunes\Documents\Alura\meus-projetos-alura\projetos\_aulas\29-mongo\_db\mongodb-win32-x86\_64-2008plus-ssl-3.4.24\bin>

============================================

executando o mongo

./mongo

**======================================CREATE=========================================**

* criar tabela

db.createCollection("alunos")

obs: inserir dados (observe é a mesma estrutura de estânciar um Objeto no js).

**==================================INSERT ==============================**

* inserindo um aluno.

db.alunos.insert(

{

"nome" : "Charles",

"data\_nascimento" : new Date(1994,02,26)

}

)

==========================================================================================

* inserindo mais um aluno.

db.alunos.insert(

{

"nome" : "Charles",

"data\_nascimento" : new Date(1994,02,26),

"curso" : {

"nome":"Sistemas de Informação",

},

"notas" : [10.0,7.5,8.0,5.0],

"habilidades" : [

{

"nome" : "Inglês",

"nível" : "Intermediario"

},

{

"nome" : "futebol",

"nível" : "Craque"

}

]

}

)

**=================================== REMOVER ==========================================**

* removendo um aluno (copia o id do id que aparece na pesquisa)

db.alunos.remove(

{

"\_id" : ObjectId("64615a7e6ccef7a716355508")

}

)

**================================== CONSULTAR - FIND() ===================================**

* consultar todos aluno:

db.alunos.find()

===================================================================================

* consultar todos alunos de forma legível:

db.alunos.find().pretty()

==================”” aspas na propriedade é opcional, vamos tirar ela ==========================

* consultar um aluno em específico de 2 forma, com aspas ou sem na propriedade

db.alunos.find(

{

nome : "Charles",

}

).pretty()

================= Ou ==============

db.alunos.find( { "nome" : "Felipe" } )

========================================================================================

Liste todos os alunos com o nome Felipe e com a data de nascimento de 26/02/1994

db.alunos.find(

{

"nome" : "Felipe",

"data\_nascimento" : new Date(1994, 02, 26)

}

)

============================================

* consultar o aluno por propriedade específica

db.alunos.find( { “habilidade.nome” : ”ingles” } ).pretty()

====================================operador $or ========================================

* consultar alunos de tabelas diferentes que tenham um dos dois cursos: Sistemas... ou Engenha.....

db.alunos.find()(

$or : [

{curso.nome: “Sistemas de Informação”},

{curso.nome: “Engenharia Química”}

],

“nome”: Daniela

).pretty()

Obs: repare que o $or só estão comparando o que está no array, o “nome” fora será chamado independente

======================= operador $in =====================

db.alunos.find()(

“curso.nome” : { $in : [“Sistemas de Informação” , “Engenharia Química”] }

).pretty()

Obs: $in é parecido com $or, ele faz uma pesquisa com uma comparação se está dentro de um ou do outro.

========================================================================================

* mais exemplos de $or e $in

Busque todos os alunos dos cursos de Sistemas de informação ou de Ciências da computação

db.alunos.find(

{ $or : [

{"curso.nome": "Sistemas de informação"},

{"curso.nome": "Ciências da computação"}

]

}

).pretty()

Ou

db.alunos.find(

{

"curso.nome": { $in : [

"Sistemas de informação",

"Ciências da computação"

]

}

}

).pretty()

`

**=======================================UPDATE=======================================**

* alterar **um valor** em Sistema que faltou o ‘s’ para Sistemas

Db.alunos.update(

{“curso.nome” : “Sistema de Informação”},

{

$set : {“curso.nome” : “Sistemas de Informação”}

}

)

Obs: O update só altera uma propriedade por vez, se quiser alterar outra, será valida a primeira

========================================================================================

* alterar **vários valores** em Sistema que faltou o ‘s’ para Sistemas e informação colocando o ‘i’ em maiúsculo.

Db.alunos.update(

{“curso.nome” : “Sistema de Informação”},

{

$set : {“curso.nome” : “Sistemas de Informação”}

},

{

multi : true

}

)

Obs: foi adicionado o multi : true para alterar vários valores ou se quiser apenas um.

========================================================================================

* Atualizar e adicionar uma nota 8.5 no campo notas (**forma não otimizada**)

db.alunos.update(

{ “\_id" : ObjectId("64615a7e6ccef7a716355508") } ,

{

$set : { “notas” : [10, 9 , 4.5 , 8.5 ] }

}

)

========================================================================================

* Atualizar e adicionar uma nota 8.5 no campo notas (**forma otimizada**)

db.alunos.update(

{ “\_id" : ObjectId("64615a7e6ccef7a716355508") } ,

{

$push : { notas : 8.5 }

}

)

Obs: $push indica que sera atualizado apenas um elemento.

========================================================================================

* Atualizar e adicionar **mais de uma** **nota** 8.5 e 3.5 no campo notas (**forma otimizada**)

db.alunos.update(

{ “\_id" : ObjectId("64615a7e6ccef7a716355508") } ,

{

$push : { notas : { $each : [ 8.5, 3.5 ] } }

}

)

Obs: $push junto com $each indica que serão atualizados e adicionados mais de um elemento.

========================================================================================

* Pesquisar quem tem nota 8.8

Db.alunos.find({

{

“nota” : 8.5

}

})

Obs: dessa forma será feita a busca por todos que tiverem nota = 8.5

========================================================================================

* Pesquisar quem tem nota 5

Db.alunos.find({

{

nota: {$gt : 5}

}

})

Obs: o termo $gt vem da palavra ‘greater than, que é ‘maior quê’. Pesquisa todos que sejam maior que 5

========================================================================================

* Pesquisar uma pessoa com nota maior que 5

Db.alunosfindOne({

Nota : {$gt : 5}

})

Obs: essa pesquisa retorna o primeiro do banco de dados.

========================================================================================

* Pesquisar todas as pessoas por nome em ordem alfabética

Db.alunos.find().sort({“nome” : 1}) ordem alfabética ascendente

Db.alunos.find().sort({“nome” : -1}) ordem alfabética descendente

Db.alunos.find().sort({“nome” : 1}).limit(3) ordem alf asc apenas os 3 primeiros

========================================================================================

* Localização Geográfica
* Adicionar um campo localização.

Db.alunos.update(

{ “ id " : ObjectId("64615a7e6ccef7a716355508") },

{

$set : {

Localização : {

Endereço : “Rua Sávia, 3332”,

Cidade : “cascavel”,

Coordinates : [-4.119655464963974, -38.23935270309448],

Type : “Point”

}

}

}

)

Obs: o importante é as coordenadas escrito em inglês e o tipo que é “Point” para achar o local exato.

========================================================================================

* Importando um array com vários nomes e suas localizações.

Mongoimport -c alunos –jasonArray < alunos.jason

========================================================================================

* Realizando uma pesquisa por a proximidade, busca é feita em array
* Primeiro precisar criar um índice informando qual o nome vai ser usado para localizar, no nosso caso é localizacao.

Db.alunos.createIndex({ localização : “2dsphere” })

* Agora fazer a pesquisa das coordenadas.

Db.alunos.aggregate([

{

$geoNear : {

near : {

coordinates : [-4.119655464963974, -38.23935270309448],

type : “Point”

},

distanceField : “distancia.calculada”,

Spherical : true

}

}

])

Obs: $geoNear e near = significa quem está mais próximo das coordinates e type

========================================================================================

* Realizando uma pesquisa por a proximidade, buscar apenas 3 pessoas e ignorar o primeiro que é a própria coordenada da pesquisa.

Db.alunos.aggregate([

{

$geoNear : {

near : {

coordinates : [-4.119655464963974, -38.23935270309448],

type : “Point”

},

distanceField : “distancia.calculada”,

spherical : true,

**num : 4**

}

},

{

**$skip : 1**

}

])

Obs: $skip é um comando de ignorar pela posição, no caso a primeira

===================== dados para exportar ====================

Link: <https://s3.amazonaws.com/caelum-online-public/MONGO/alunos.json>

ou

manualmente:

{

"nome" : "Guilherme",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5882133, -46.63235580000003]}

},{

"nome" : "Paulo",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5707855, -46.643499399999996]}

},

{

"nome" : "Ana",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5829461, -46.6315126]}

},

{

"nome" : "Carlos",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5834181, -46.6418552]}

},

{

"nome" : "Lúcia",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5702415, -46.635375]}

},

{

"nome" : "Stella",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5623743, -46.6478634]}

},

{

"nome" : "Daniela",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5690615, -46.6592789]}

},

{

"nome" : "Eduardo",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5552147, -46.6574764]}

},

{

"nome" : "Felipe",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5606041, -46.6663599]}

},

{

"nome" : "Marco",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5625317, -46.6686773]}

},

{

"nome" : "Mariana",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5617056, -46.662454600000004]}

},

{

"nome" : "Juliana",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5578111, -46.6656303]}

},

{

"nome" : "Adriana",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5653639, -46.661725]}

},

{

"nome" : "Roberto",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5549787, -46.6588497]}

},

{

"nome" : "Marcelo",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5640265, -46.6527128]}

},

{

"nome" : "Sofia",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5673307, -46.6529703]}

},

{

"nome" : "Sheila",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5672914, -46.6462326]}

},

{

"nome" : "William",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5803502, -46.6352892]}

},

{

"nome" : "Manoela",

"localizacao" : { "type" : "Point", "coordinates" : [-23.5831428, -46.6334867]}

}

]