

# Consultas MongoDB: Função Find

# Agenda

- Consultas simples
- Consultas em documentos embutidos
- Consultas em arrays
- Consultas em arrays de documentos

## db.collection.find()

- **Seleciona** documentos de uma coleção e devolve um **cursor** para os documentos selecionados.
- Devolve: Um **cursor** para os documentos que correspondem aos **critérios** da consulta.

# db.collection.find()

- db.collection.find( <query>, <projection>, <options> )
  - <query>: Opcional. Especifica o filtro de **seleção** utilizando operadores de consulta. Para devolver todos os documentos em uma coleção, utiliza-se um **documento vazio** ({}).
  - <projection>: Opcional. Especifica os **campos** a serem **devolvidos** nos documentos que correspondem ao filtro de consulta. Para devolver todos os campos nos documentos correspondentes, omitimos este parâmetro.
  - <Options>: Opcional. Especifica mais opções para a consulta. Estas opções **modificam** o **comportamento** da consulta e a forma como os resultados são devolvidos.

# db.collection.find()

- Exemplo:

```
db.users.find(  
  { username : "rajiv"}, // query  
  { age : 1 }, // projection  
  { limit : 1 } // options  
)
```

- **Projeção:**
  - Determina quais os campos que são devolvidos nos documentos correspondentes. O parâmetro `projection` obtém um documento com a seguinte estrutura:
    - { <field1>: <value>, <field2>: <value> ... }
- **<field>: <1 or 0>:** 1 especifica a inclusão do campo, 0 a exclusão do campo.
- **"<field>.\$": <1 or true>:** Utiliza o operador de projeção de array `$` para devolver o primeiro elemento que corresponde à condição de consulta no campo de array.

# db.collection.find()

- Opções (algumas)
  - `allowDiskUse`
    - Se a operação exigir mais de **100 megabytes** de memória, é possível recorrer à escrita em disco - `cursor.allowDiskUse()`.
  - **Limit**: Define um **limite** de documentos devolvidos no conjunto de resultados.
  - **Ignorar**: Quantos documentos devem ser **ignorados** antes de devolver o primeiro documento no conjunto de resultados.
  - **Sort**: A **ordem** dos documentos devolvidos no conjunto de resultados. Os campos especificados na classificação devem ter um **índice**.

# db.collection.find()

- Exemplos:

```
db.customers.find().sort({birthdate: -1}).limit(1)
```

```
db.customers.find().skip(50).count()
```

```
db.customers.find().sort({birthdate: 1}).skip(1).limit(1)
```



## db.collection.find()

- O método `find()` devolve um `cursor` para os resultados.
- Em mongoshell, se o cursor devolvido não for atribuído a uma variável utilizando a palavra-chave `var`, o cursor será automaticamente `iterado` para aceder até os primeiros `20` documentos que corresponderem à consulta.
- Para iterar `manualmente` sobre os resultados, atribuímos o cursor devolvido a uma variável com a palavra-chave `var`.

- Utilizar a variável `myCursor` para iterar sobre o cursor e `imprimir` os documentos correspondentes:

```
var myCursor = db.bios.find( );  
myCursor
```

- O exemplo a seguir utiliza o método do cursor `next()` para aceder aos documentos:

```
var myCursor = db.bios.find( );  
  
var myDocument = myCursor.hasNext() ? myCursor.next() : null;  
  
if (myDocument) {  
    var myName = myDocument.name;  
    print (tojson(myName));  
}
```

- Para imprimir, podemos utilizar o método `printjson()`:

```
if (myDocument) {  
    var myName = myDocument.name;  
    printjson(myName);  
}
```

- Utilizar o método de cursor `forEach()` para iterar o cursor e aceder aos documentos:

```
var myCursor = db.bios.find( );  
  
myCursor.forEach(printjson);
```

# db.collection.find()

- Podemos também criar funções **anónimas** e alterar os dados (como o tipo de dados de um campo) e armazenar numa outra coleção:

```
var data = db.salesDetails.find()
data.forEach(
  function(item) {
    item.OrderDate = new ISODate(item.OrderDate);
    db.sales.save(item);
  }
)
```

Atualiza ou insere o documento



# db.collection.find()

- Vamos considerar o dataset: **primerdataset**;

- Exemplo:

```
{
  "_id": {
    "$oid": "61dd7a655946af462c8bd2e4"
  },
  "address": {
    "building": "1007",
    "coord": [
      -73.856077,
      40.848447
    ],
    "street": "Morris Park Ave",
    "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

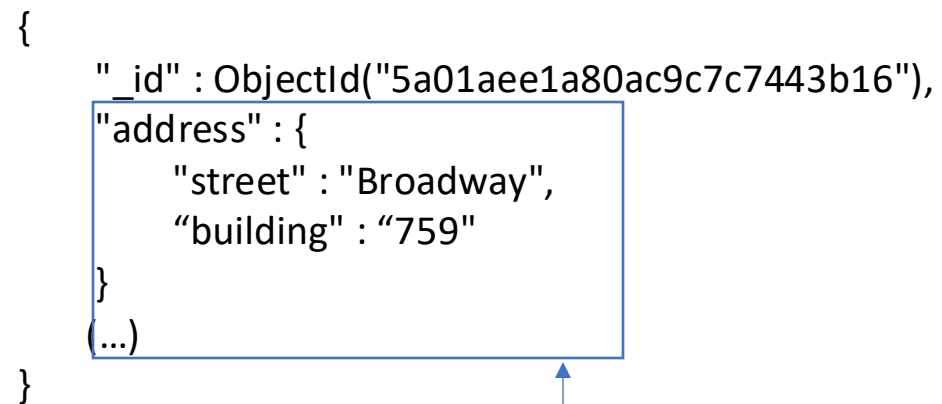
- Mostrar todos os restaurantes de cozinha: “Bakery” ou “American”:

```
db.restaurants.find(  
  {  
    cuisine:  
      {$in: ["Bakery", "American"]}  
  }  
)
```

# Consultas de documentos

- Selecionar **pelo documento** (completo):

```
{
  "_id" : ObjectId("5a01aee1a80ac9c7c7443b16"),
  "address" : {
    "street" : "Broadway",
    "building" : "759"
  }
  (...)
}
```



Documento

Restaurantes da rua (street): "Broadway" e edifício: "759":

```
db.restaurants.find({
  "address":{
    "street" : "Broadway"
    "building":"759"
  })
```

# Consultas de documentos

- Selecionar por um **campo embutido** no documento:

```
{
  "_id" : ObjectId("5a01aee1a80ac9c7c7443b16"),
  "address" : {
    "street" : "Broadway",
    "building" : "79"
  }
  (...)
}
```

Restaurantes da rua (street): "Broadway":

```
db.restaurants.find( {"address.street": "Broadway" } )
```



# Consultas de documentos

- Selecionar pelo campo **embutido** no documento com **múltiplas condições**:

```
{  
  "id" : ObjectId("5a01aee1a80ac9c7c7443b16"),  
  "address" : {  
    "building" : "1560",  
    "coord" : [  
      -73.98527039999999,  
      40.7589099  
    ],  
    "street" : "Broadway",  
    "zipcode" : "10036"  
  }  
}  
(...)
```

Restaurantes da rua (street): "Broadway" e edifício (building): "1560"

```
db.restaurants.find({"address.street": "Broadway",  
  "address.building": "1560"}).pretty()
```

# Consultas em arrays

- Consultar um **array exato**, incluindo a **ordem** dos elementos:

```
db.restaurants.find({"address.coord": [73.98527039999999, 40.7589099] })
```

- Devolver todos os documentos em que o conteúdo do campo: coord é um array com exatamente dois elementos pela ordem indicada.

# Consultas em arrays

- No exemplo anterior, a **ordem** dos elementos do array tem de ser seguida (o que faz sentido para a consulta apresentada);
- Se a ordem for **indiferente**, poderá utilizar o operador **\$all** para garantir que o array possui os dois elementos:

```
db.restaurants.find(  
  {  
    "address.coord":  
    {  
      $all: [-73.98527039999999, 40.7589099]  
    }  
  }  
)
```

# Consultas em arrays

- Consultar um array que tem **pelo menos** um elemento com o valor especificado:

```
db.restaurants.find({"address.coord":40.7589099 })
```

- Também é possível realizar consultas por **intervalo** de valores:

- > - gt
- < - lt
- >= gte
- <= lte

```
db.restaurants.find({"address.coord":{"$gt:40}})
```

```
db.restaurants.find(  
    {  
        "address.coord":  
            {$gt:40, $lt:60}  
    }  
)
```

# Consultas em arrays

- Consultar um elemento do array que satisfaz **múltiplos critérios**;
- No exemplo do último slide procuramos um valor maior do que 40 e menor que 60 mas **não necessariamente o mesmo elemento**!

- O operador **\$elemMatch** especifica **múltiplos** critérios nos elementos do array de forma a que exista **pelo menos um** elemento que satisfaça esses critérios:

```
db.restaurants.find({  
  "address.coord":  
    {  
      $elemMatch:  
        {$gt:40, $lt:60}  
    }  
})
```

# Consultas em arrays

- Consultar um elemento pelo **índice**:

```
db.restaurants.find({"address.coord.0":{"$gt:40}})
```

- Consultar pelo tamanho:

```
db.restaurants.find({"address.coord":{"$size:2}})
```

# Consultas em arrays de documentos

```
{
  "_id" : ObjectId("5a01aeel1a80ac9c7c7443b16"),
  "borough" : "Manhattan",
  "cuisine" : "Delicatessen",
  "grades" : [
    {
      "date" : ISODate("2014-01-21T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
  ],
  "name" : "Bully'S Deli",
  "restaurant_id" : "40361708"
}
```



# Consultas em arrays de documentos

- Consultar pelo documento **exato**:

```
db.restaurants.find(  
  {"grades":  
    {  
      date: ISODate("2011-06-07T00:00:00Z"),  
      grade: "A",  
      score: 9  
    }  
  }  
) .pretty()
```

# Consultas em arrays de documentos

- Consulta por um campo num **array** de documentos:

```
db.restaurants.find(  
  {  
    "grades.score":  
    {  
      $lte:100  
    }  
  }  
) .pretty()
```

# Consultas em arrays de documentos

- Consultar pelo índice:

```
db.restaurants.find(  
  {"grades.0.score":  
    {$lte:100}  
}  
) .pretty()
```

# Consultas em arrays de documentos

- Consulta por **múltiplas** condições:

```
db.restaurants.find(  
  {"grades":  
    {$elemMatch:  
      {score:  
        {$gt:75, $lte:100}  
      }  
    }  
  }  
) .pretty()
```

# Projeção em arrays com/sem documentos embutidos

- Projetar campos específicos de documentos **embutidos/arrays** de documentos embutidos:

```
db.restaurants.find({}, {name:1, "grades.score":1}).pretty()
```

# Projeção em arrays com/sem documentos embutidos

- Projetar elementos do array:

- Último elemento:

```
db.restaurants.find({}, {name:1, "grades":{$slice:-1}}).pretty()
```

- Últimos 3 elementos:

```
db.restaurants.find({}, {name:1, "grades":{$slice:3}}).pretty()
```

- Primeiros 3 elementos

```
db.restaurants.find({}, {name:1, "grades":{$slice:-3}}).pretty()
```

# Consultas por valores Null ou keys em falta

- Inserir documentos com valores nulos:

```
db.inventory.insertMany([
  { _id: 1, item: null },
  { _id: 2 }
])
```

- Selecionar com valores null:

```
db.inventory.find( { item: null } )
```

# Consultas por valores Null ou keys em falta

- Verificação de **existência**:

```
db.inventory.find(  
    {  
        item :  
            { $exists: false }  
    }  
)
```



## Bibliografia/referências

- <https://www.json.org/json-en.html>
- <https://docs.mongodb.com/manual/>
- <https://www.mongodb.com/pt-br/docs/manual/tutorial/query-documents/>
- Phaltankar, A. Ahsan, J., Harrison, M., Nedov, L. (2020). MongoDB Fundamentals: A hands-on guide to using MongoDB and Atlas in the real world. Packt Publishing

# Consultas MongoDB: Função Find