POLITÉCNICO
DO PORTO
ESCOLA
SUPERIOR
DE MEDIA ARTES
E DESIGN

P.PORTO

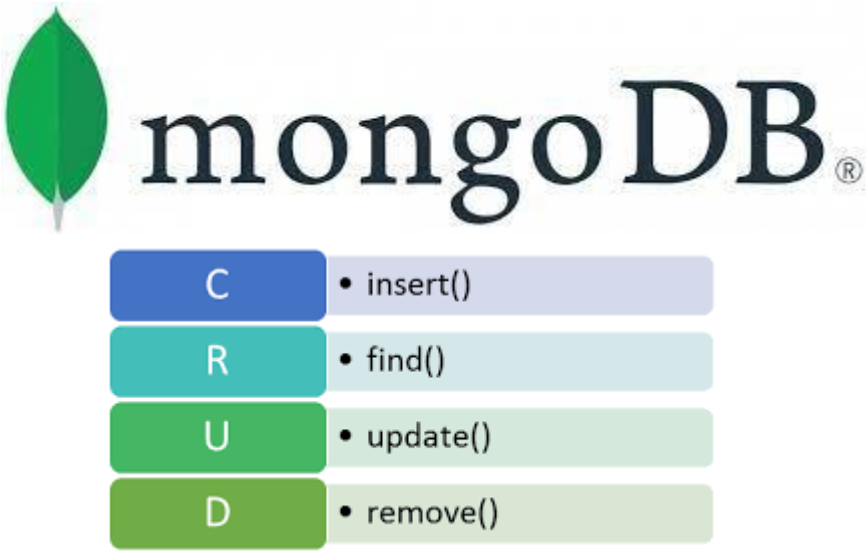
BASES DE DADOS
Módulo III

MongoDB

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

Agenda

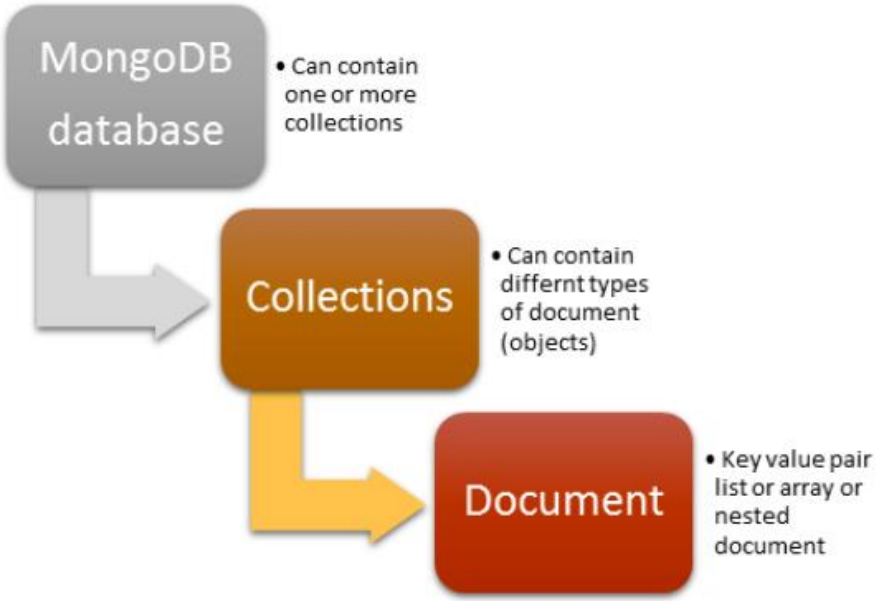
- **MongoDB Sctucture**
- **MongoBD Syntax**
- ***MongoBD CRUD operations***



MongoDB

MongoDB Sctucture

MySQL	MongoDB
• Database • Table • Rows • Column • Joins • Primary Key	• Database • Collection • Document • Field (document field) • Embebed documents • Primary Key (default id_key provided by MongoDB)



MongoDB

MongoDB Sctucture


```
{
  name: "sue",                    ← field: value
  age: 26,                        ← field: value
  status: "A",                    ← field: value
  groups: [ "news", "sports" ]    ← field: value
}
```

MongoDB

MongoDB

- It refers to the object-oriented paradigm
- It does not support joins but can represent hierarchical data structures
- It does not have schemas which means that each collection can contain different types of objects
- easily scalable (because it does not have a DB schema)

mongoDB

{ name: mongo, type: DB }

MongoDB

MongoDB

- Each object is called a **document** and is represented by a **JSON structure** (JavaScript Object Notation)

- A document field may include:
  - literal values,
  - arrays,
  - embedded documents
  - Arrays of documents

MongoDB

MongoDB

- Primary key is automatically set by the **_id** field

  Documents always have a unique **_id** field that is a kind of a primary key (unique identifier)

- Documents that share a collection, usually share the same structure, although this is not mandatory

MongoDB

MongoDB – Based on d**ocuments, such as:**

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_salary: "2150"
employee_depart: Array
    0: "sales"
    1: "Accounting"
employee_adress: Object
    street: "Rua da Bela Vista"
    number: "180"
    city: "Porto"
employee_evaluation: Array
    0: Object
        year: "2018"
        score: "4"
    1: Object
        year: "2019"
        score: "5"
```

array

Embedded document

Array of objects (documents)

MongoDB

MongoDB – Based on d**ocuments, such as:**

Insert to Collection

VIEW {} ≡

```
1  {
2      "_id": {
3          "$oid": "5ffc3687cf724c5863ca5444"
4      },
5      "employee_id": "01",
6      "employee_name": "Eduardo Albuquerque",
7      "employee_salary": "2150",
8      "employee_depart": ["sales", "Accounting"],
9      "employee_adress": {
10         "street": "Rua da Bela Vista",
11         "number": "180",
12         "city": "Porto"
13     },
14     "employee_evaluation": [{
15         "year": "2018",
16         "score": "4"
17     }, {
18         "year": "2019",
19         "score": "5"
20     }]
21 }
```

MongoDB

MongoDB - **Relationships**

- Relationships in MongoDB represent how various documents are logically related to each other.
- Relationships can be modeled by **Embedded** and **Referenced approaches**.
- Such relationships can be either 1:1, 1:N, N:1 or N:N.

MongoDB

MongoDB - **Relationships**

Let's consider the case of storing addresses for users. In this case, one user can have multiple addresses, making this a 1:N relationship

- **Embedded approach**

address attribute is an array of objects

```
> db.users.insert({
        {
                "_id":ObjectId("52ffc33cd85242f436000001"),
                "contact": "987654321",
                "dob": "01-01-1991",
                "name": "Tom Benzamin",
                "address": [
                        {
                                "building": "22 A, Indiana Apt",
                                "pincode": 123456,
                                "city": "Los Angeles",
                                "state": "California"
                        },
                        {
                                "building": "170 A, Acropolis Apt",
                                "pincode": 456789,
                                "city": "Chicago",
                                "state": "Illinois"
                        }
                ]
        }
})
```

MongoDB

MongoDB - **Relationships**

This approach maintains all the related data in a single document, which makes it easy to retrieve and maintain.

- **Embedded approach**

```
> db.users.insert({
        {
                "_id":ObjectId("52ffc33cd85242f436000001"),
                "contact": "987654321",
                "dob": "01-01-1991",
                "name": "Tom Benzamin",
                "address": [
                        {
                                "building": "22 A, Indiana Apt",
                                "pincode": 123456,
                                "city": "Los Angeles",
                                "state": "California"
                        },
                        {
                                "building": "170 A, Acropolis Apt",
                                "pincode": 456789,
                                "city": "Chicago",
                                "state": "Illinois"
                        }
                ]
        }
})
```

MongoDB

MongoDB - **Relationships**

- **Referenced approach**

  This is the approach of designing normalized relationship.
  In this approach, both the user and address documents will be maintained separately but the user document will contain a field that will reference the address document's **id** field

```
{
   "_id":ObjectId("52ffc33cd85242f436000001"),
   "contact": "987654321",
   "dob": "01-01-1991",
   "name": "Tom Benzamin",
   "address_ids": [
      ObjectId("52ffc4a5d85242602e000000"),
      ObjectId("52ffc4a5d85242602e000001")
   ]
}
```

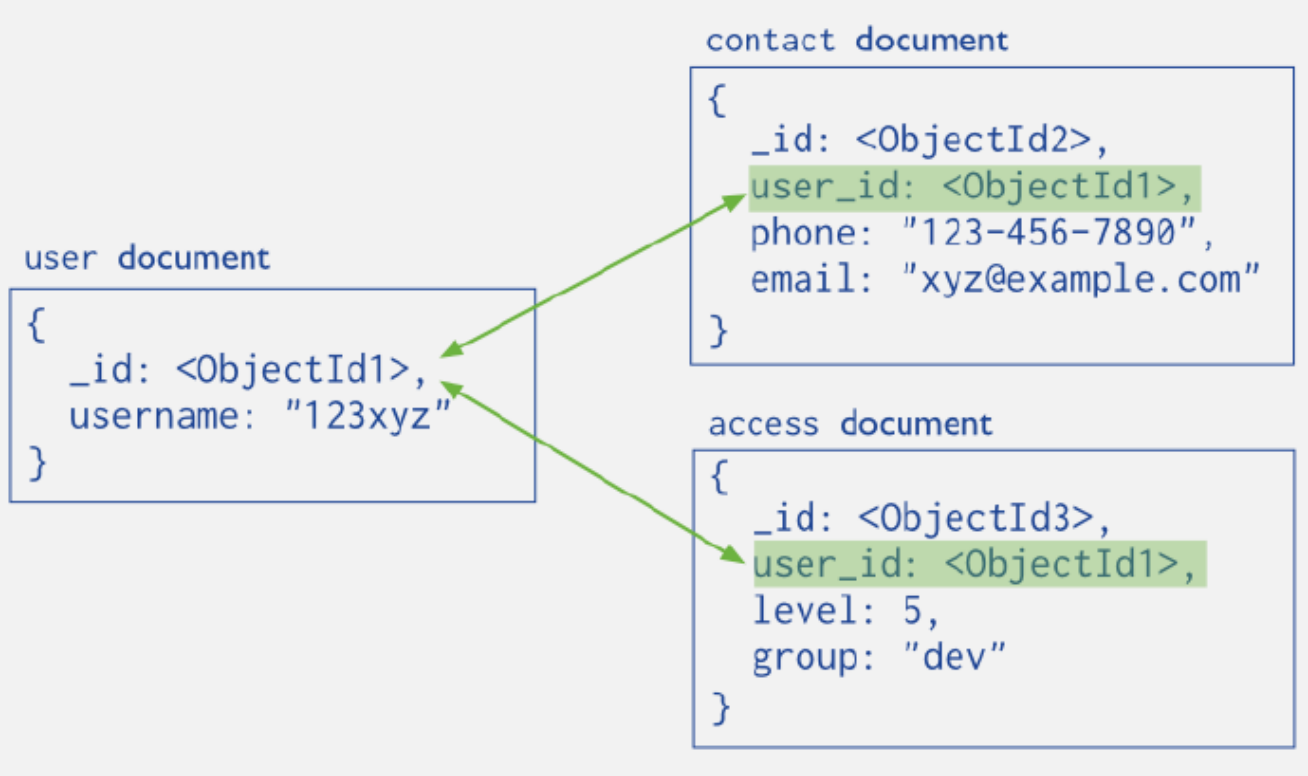
```
{
   "_id":ObjectId("52ffc4a5d85242602e000000"),
   "building": "22 A, Indiana Apt",
   "pincode": 123456,
   "city": "Los Angeles",
   "state": "California"
}
```

```
{
   "_id":ObjectId ("52ffc4a5d85242602e000001"),
   "building": "170 A, Acropolis Apt",
   "pincode": 456789,
   "city": "Chicago",
   "state": "Illinois"
}
```

MongoDB

MongoDB - **Relationships**

- **Referenced approach**



MongoDB

MongoDB

MongoDB

MongoDB

- CRUD operations (**create, read, update delete**) in MongoDB

database.collection.operation(*arguments*)

C	• insert()
R	• find()
U	• update()
D	• remove()

```
> db.user.insert({
    first: "John",
    last : "Doe",
    age: 39
})
```

```
> db.user.find ()
{
    "_id" : ObjectId("51..."),
    "first" : "John",
    "last" : "Doe",
    "age" : 39
}
```

```
> db.user.update(
{"_id" : ObjectId("51...")},
{
    $set: {
    age: 40,
    salary: 7000}
    }
)
```

```
> db.user.remove({
    "first": /^J/
})
```

MongoDB

C • insert()

- **db.*collection_name*.insert({*document*});**

```
doc = {
    name: 'xyx',
    class: '12th',
    subjects: ['physics', 'chemisrty', 'maths', 'english', 'computer'],
    address: {
        house_no: '123',
        sector: '50',
        city: 'noida'
    }
}
db.mycol.insert(doc);
```

MongoDB

C • insert()

- **db.*collection_name*.insert({*document*});**

Insert to Collection

VIEW {}

```
db.employees.insert(
{
  "employee_id": "11",
  "employee_name": "Ricardo Santos",
  "employee_salary": "1980",
  "employe_depart": [
    "Sales",
    "Logistics"
  ],
  "employee_adress": {
    "street": "Rua dos Banhos",
    "number": "981",
    "city": "Vila do Conde"
  }
})
```

MongoDB

C • insert()

- **db.*collection_name*.i**

```
>db.post.insert([
   {
      title: 'MongoDB Overview',
      description: 'MongoDB is no sql database',
      by: 'tutorials point',
      url: 'http://www.tutorialspoint.com',
      tags: ['mongodb', 'database', 'NoSQL'],
      likes: 100
   },

   {
      title: 'NoSQL Database',
      description: "NoSQL database doesn't have tables",
      by: 'tutorials point',
      url: 'http://www.tutorialspoint.com',
      tags: ['mongodb', 'database', 'NoSQL'],
      likes: 20,
      comments: [
         {
            user:'user1',
            message: 'My first comment',
            dateCreated: new Date(2013,11,10,2,35),
            like: 0
         }
      ]
   }
])
```

To **insert multiple documents** in a single query.

You can pass an **array of documents** in the Insert command.

MongoDB

C • insert()

- **db.*collection_name*.insertOne({*document*});**

If you need to insert **only one document** into a collection you can use this method.

```
db.employees.insertOne(
{
  "employee_id": "01",
  "employee_name": "Eduardo Albuquerque",
  "employee_salary": "2150",
  "employee_depart": [
    "sales",
    "Accounting"
  ],
  "employee_adress": {
    "street": "Rua da Bela Vista",
    "number": "180",
    "city": "Porto"
  }
})
```

MongoDB

C • insert()

- **db.*collection_name*.insertMany([{*documents*}]);**

You can insert multiple documents using the insertMany method. To this method **you need to pass an array of documents**.

```
db.employees.insertMany([
{
  "employee_id": "01",
  "employee_name": "Eduardo Albuquerque",
  "employee_salary": "2150",
  "employee_depart": [
    "sales",
    "Accounting"
  ],
  "employee_adress": {
    "street": "Rua da Bela Vista",
    "number": "180",
    "city": "Porto"
  }
},
{
"employee_id": "02",
  "employee_name": "Liliana Ferreira",
  "employee_salary": "1850",
  "employee_depart": [
    "sales",
    "Production"
```

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(*query, projection*)**

  Selects documents in a collection and returns a cursor of documents

  To query data from MongoDB collection, you need to use the **find** method.

Parameter	Type	Description
query	document	Criteria to use in the query When not set, returns all the documents in the collection
projection	document	Specifies the fields to be returned in query When not set, returns all fields on the document Optional argument

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

- Query selectors
  - Comparation ($eq, $ne, $gt, $gte, $lt, $lte)
  - Logical ($or, $and, $not, $in, $all)
  - Element ($exists)

More information about query selector in:

https://docs.mongodb.com/manual/reference/operator/query/

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

Query

- All queries address a **single collection**
- All queries **return** a cursor with the **corresponding documents**
- Queries **return all attributes** in a document (**by default**!)

- Projection
- **Projections** allow you to limit the **number of attributes** returned
- **Projections** allow you to **specify the list of attributes to return**, or attributes to be excluded

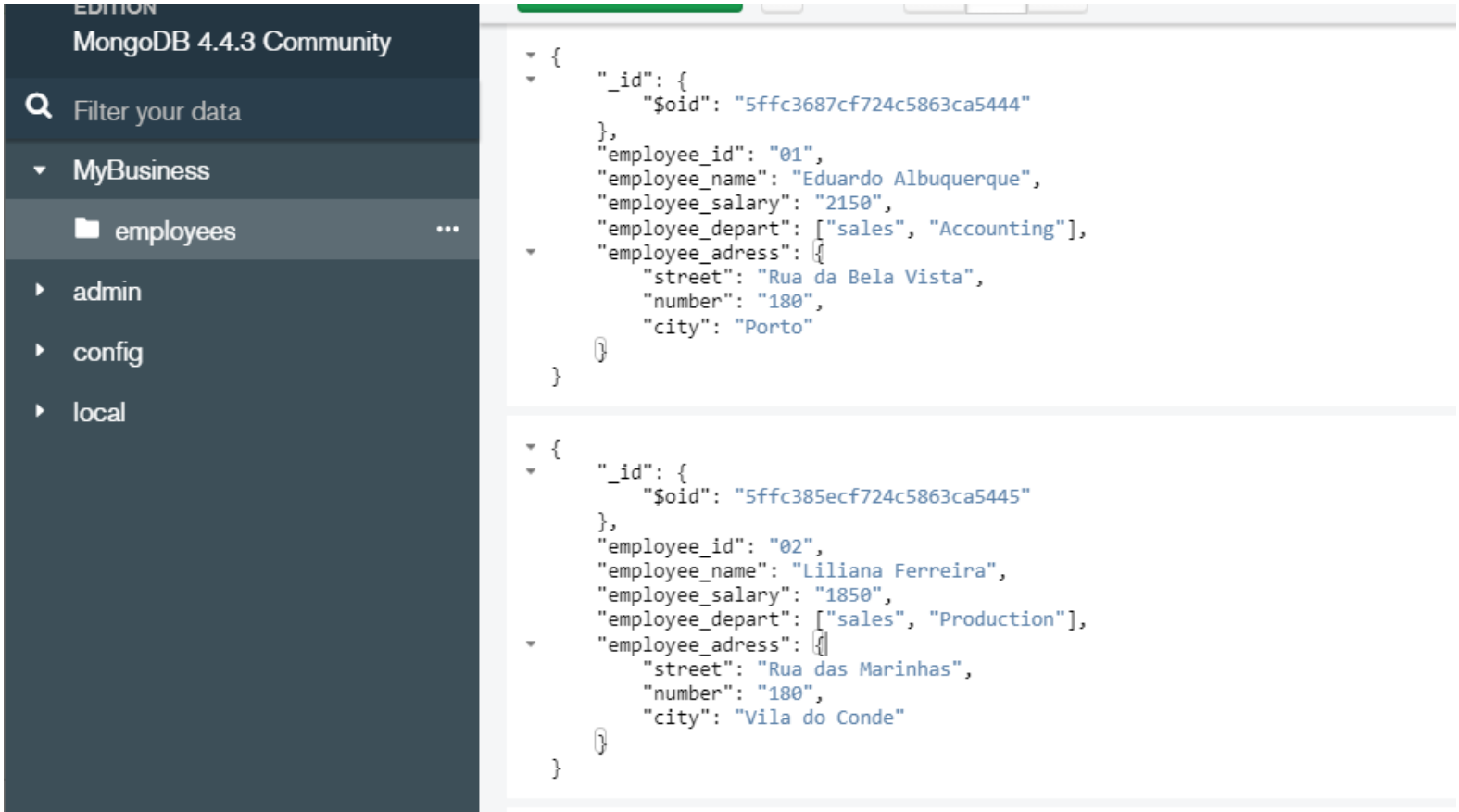
MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

  sample: employee's collection – 11 documents, such as:



MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_name": "Eduardo Albuquerque"}

ADD DATA VIEW


```
{
    "_id": {
        "$oid": "5ffc3687cf724c5863ca5444"
    },
    "employee_id": "01",
    "employee_name": "Eduardo Albuquerque",
    "employee_salary": "2150",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {
        "street": "Rua da Bela Vista",
        "number": "180",
        "city": "Porto"
    }
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_adress.city": "Porto"} OPTIONS FIND RESET

ADD DATA VIEW Displaying documents 1 - 4 of 4


```
{
    "_id": {
        "$oid": "5ffc3687cf724c5863ca5444"
    },
    "employee_id": "01",
    "employee_name": "Eduardo Albuquerque",
    "employee_salary": "2150",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {
        "street": "Rua da Bela Vista",
        "number": "180",
        "city": "Porto"
    }
}
```

```
{
    "_id": {
        "$oid": "5ffc3896cf724c5863ca5446"
    },
    "employee_id": "03",
    "employee_name": "Maria Mariazinha",
    "employee_salary": "1950",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {
        "street": "Rua da Boavista",
        "number": "170",
        "city": "Porto"
    }
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_depart": "sales"} OPTIONS FIND RE

ADD DATA VIEW Displaying documents 1 - 7 of 7


```
{
    "_id": {
        "$oid": "5ffc3687cf724c5863ca5444"
    },
    "employee_id": "01",
    "employee_name": "Eduardo Albuquerque",
    "employee_salary": "2150",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {...}
}
```

```
{
    "_id": {
        "$oid": "5ffc385ecf724c5863ca5445"
    },
    "employee_id": "02",
    "employee_name": "Liliana Ferreira",
    "employee_salary": "1850",
    "employee_depart": ["sales", "Production"],
    "employee_adress": {...}
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_depart": "Computing"} OPTIONS FIND

ADD DATA VIEW {} Displaying documents 1 - 3 of 3


```
{
    "_id": {
        "$oid": "5ffc38f3cf724c5863ca5448"
    },
    "employee_id": "05",
    "employee_name": "Carla Maria Cardoso",
    "employee_salary": "2350",
    "employee_depart": ["sales", "Computing"],
    "employee_adress": {...}
}

{
    "_id": {
        "$oid": "5ffc394ccf724c5863ca544a"
    },
    "employee_id": "06",
    "employee_name": "Fátima Maria Cardoso",
    "employee_salary": "2350",
    "employee_depart": ["Computing"],
    "employee_adress": {...}
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Restrictions on queries - **operators**:

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

FILTER {"employee_salary": {$gt: "1500"}}

ADD DATA VIEW


```
{
    "_id": {
        "$oid": "5ffc3687cf724c5863ca5444"
    },
    "employee_id": "01",
    "employee_name": "Eduardo Albuquerque",
    "employee_salary": "2150",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {...}
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Restrictions on queries - **operators**:

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

FILTER {"employee_salary": {$gte: "2350"}}

ADD DATA VIEW


```
{
    "_id": {
        "$oid": "5ffc38f3cf724c5863ca5448"
    },
    "employee_id": "05",
    "employee_name": "Carla Maria Cardoso",
    "employee_salary": "2350",
    "employee_depart": ["sales", "Computing"],
    "employee_adress": {...}
}
```

```
{
    "_id": {
        "$oid": "5ffc394ccf724c5863ca544a"
    },
    "employee_id": "06",
    "employee_name": "Fátima Maria Cardoso",
    "employee_salary": "2350",
    "employee_depart": ["Computing"],
    "employee_adress": {...}
}
```

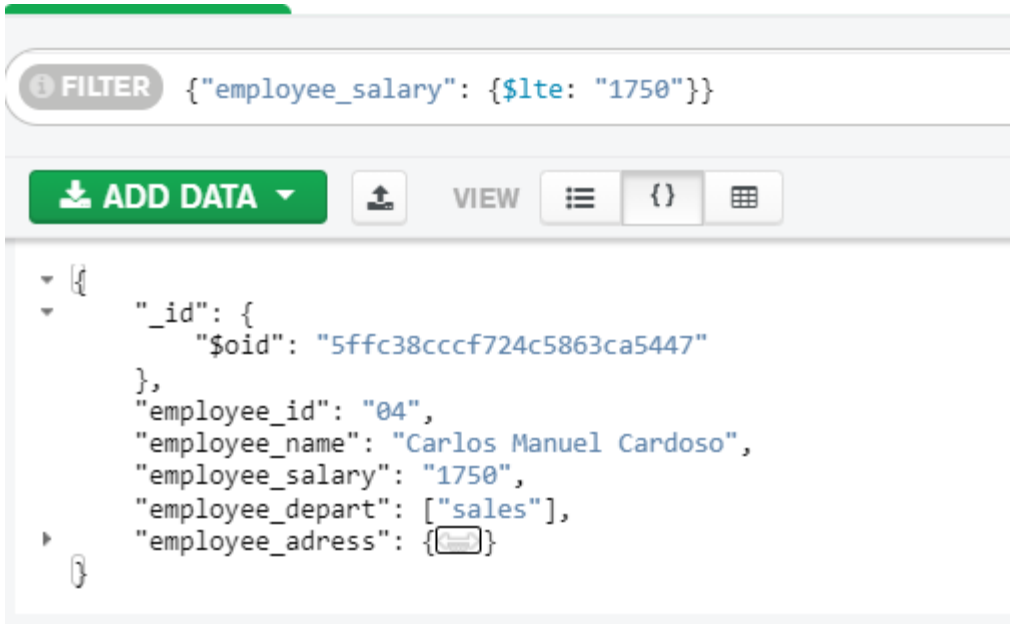
MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Restrictions on queries - **operators**:

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



MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Restrictions o

- $gt
- $gte
- $lt
- $lte

FILTER {"employee_salary": {$gt: "2000", $lt: "2300"}}

ADD DATA VIEW


```
{
    "_id": {
        "$oid": "5ffc3687cf724c5863ca5444"
    },
    "employee_id": "01",
    "employee_name": "Eduardo Albuquerque",
    "employee_salary": "2150",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {...}
}

{
    "_id": {
        "$oid": "5ffc39abcf724c5863ca544c"
    },
    "employee_id": "10",
    "employee_name": "Fernando Carlos Ferreira",
    "employee_salary": "2150",
    "employee_depart": ["sales\"", "Production ", "Logistics"],
    "employee_adress": {...}
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Restrictions on queries

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

FILTER {"employee_salary": {$gte: "2100", $lte: "2200"}}

ADD DATA VIEW {}


```
{
    "_id": {
        "$oid": "5ffc3687cf724c5863ca5444"
    },
    "employee_id": "01",
    "employee_name": "Eduardo Albuquerque",
    "employee_salary": "2150",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {...}
}
```

```
{
    "_id": {
        "$oid": "5ffc39abcf724c5863ca544c"
    },
    "employee_id": "10",
    "employee_name": "Fernando Carlos Ferreira",
    "employee_salary": "2150",
    "employee_depart": ["sales\"", "Production ", "Logistics"],
    "employee_adress": {...}
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Logical operator: **$and**

To query documents based on the AND condition, you need to use $and keyword.

Includes an array of criteria that must be matched in the query

```
>db.mycol.find({ $and: [ {<key1>:<value1>}, { <key2>:<value2>} ] })
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Logical operator: **$and**

To query documents based on the AND condition, you need to use $and keyword.

FILTER {$and: [{ "employee_depart": 'Accounting'},{ "employee_depart": 'sales'}]}

OPTIONS

FIN

ADD DATA

VIEW

Displaying documents 1 - 3 of 3

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_salary: "2150"
employee_depart: Array
    0: "sales"
    1: "Accounting"
employee_adress: Object
```

```
_id: ObjectId("5ffc3896cf724c5863ca5446")
employee_id: "03"
employee_name: "Maria Mariazinha"
employee_salary: "1950"
employee_depart: Array
    0: "sales"
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Logical operator: **$and**

To query documents based on the AND condition, you need to use $and keyword.

```
FILTER {$and: [{ "employee_salary": { $gt: '2000' }},{ "employee_salary": { $lt: '2200' } }]}
```

ADD DATA | VIEW | Displaying docu

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_salary: "2150"
> employee_depart: Array
> employee_adress: Object
```

```
> _id: ObjectId("5ffc39abcf724c5863ca544c")
  employee_id: "10"
  employee_name: "Fernando Carlos Ferreira"
  employee_salary: "2150"
> employee_depart: Array
> employee_adress: Object
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Logical operator: **$and**

To query documents based on the AND condition, you need to use $and keyword.

```
FILTER {"employee_salary": {$gte: '2100',$lte: '2200'}}
```

ADD DATA | VIEW | {} | Displa

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_salary: "2150"
> employee_depart: Array
> employee_adress: Object
```

```
_id: ObjectId("5ffc39abcf724c5863ca544c")
employee_id: "10"
employee_name: "Fernando Carlos Ferreira"
employee_salary: "2150"
> employee_depart: Array
> employee_adress: Object
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Logical operator: **$or**

includes an array of criteria that must be matched in the query

```
>db.mycol.find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Logical operator: **$or**

FILTER {$or: [{"employee_adress.city": "Porto"}, {"employee_adress.city": "Matosinhos"}]}

ADD DATA VIEW {} Disp


```
{
    "_id": {
        "$oid": "5ffc3687cf724c5863ca5444"
    },
    "employee_id": "01",
    "employee_name": "Eduardo Albuquerque",
    "employee_salary": "2150",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {
        "street": "Rua da Bela Vista",
        "number": "180",
        "city": "Porto"
    }
}

{
    "_id": {
        "$oid": "5ffc3896cf724c5863ca5446"
    },
    "employee_id": "03",
    "employee_name": "Maria Mariazinha",
    "employee_salary": "1950",
    "employee_depart": ["sales", "Accounting"],
    "employee_adress": {
        "street": "Rua da Boavista",
        "number": "170",
        "city": "Porto"
    }
}
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**

Logical operator: **$or**

FILTER {$or: [{ employee_depart: 'Accounting'},{ employee_depart: 'Production'}]}

ADD DATA VIEW {} Dis

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_salary: "2150"
employee_depart: Array
    0: "sales"
    1: "Accounting"
employee_adress: Object
```

```
_id: ObjectId("5ffc385ecf724c5863ca5445")
employee_id: "02"
employee_name: "Liliana Ferreira"
employee_salary: "1850"
employee_depart: Array
    0: "sales"
    1: "Production"
employee_adress: Object
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**
- **$in** to search for a list of values in a given field

```
FILTER {"employee_depart": {$in: [ 'sales', 'Computing', 'Production']}}
```

ADD DATA | VIEW | {} | Displaying

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_salary: "2150"
employee_depart: Array
    0: "sales"
    1: "Accounting"
employee_adress: Object
```

```
_id: ObjectId("5ffc385ecf724c5863ca5445")
employee_id: "02"
employee_name: "Liliana Ferreira"
employee_salary: "1850"
employee_depart: Array
    0: "sales"
    1: "Production"
```

it is necessary to find at least one of the values

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**
- **$in** to search for a list of values in a given field
- To find at least <u>one of the list values</u>

```
FILTER {"employee_adress.city": {$in: [ 'Vila do Conde', 'Matosinhos']}}
```

ADD DATA | VIEW | {} | Display

```
_id: ObjectId("5ffc385ecf724c5863ca5445")
employee_id: "02"
employee_name: "Liliana Ferreira"
employee_salary: "1850"
> employee_depart: Array
v employee_adress: Object
    street: "Rua das Marinhas"
    number: "180"
    city: "Vila do Conde"
```

MongoDB

R • find()

- **db.*collection_name*.find(query, projection)**
- **$all** to search for **all** the values in a list field

FILTER {"employee_depart": {$all: ["sales", "Accounting"]}}

ADD DATA VIEW {}

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_salary: "2150"
employee_depart: Array
    0: "sales"
    1: "Accounting"
employee_adress: Object
```

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**
- **$ne not equal than**

FILTER {'employee_adress.city': {$ne: "Porto"} }

ADD DATA VIEW {} Displ

```
_id: ObjectId("5ffc385ecf724c5863ca5445")
employee_id: "02"
employee_name: "Liliana Ferreira"
employee_salary: "1850"
> employee_depart: Array
v employee_adress: Object
    street: "Rua das Marinhas"
    number: "180"
    city: "Vila do Conde"
```

MongoDB

R

MongoDB

- **db.*collection_name*.find(query, projecti**

pipelines: $...

Quick reference

Query - Quick Reference (full docs)

This query browser uses MongoDB Extended JSON in strict mode to encode queries and documents.

```
Field Constraint
  e.g. { "zip" : "94114" }

Dot Notation (reference)
  e.g. { "author.firstname" : "Jackie" }

Value in an Array
  e.g. { "nicknames": "Liz" }
       // "nicknames" array contains "Liz"

System-generated Object IDs (reference)
  e.g. { "_id": { "$oid":"4c32420f2f59bce80013371f" } }

Regular Expressions (reference)
  e.g. { "genre": { "$regex":"Hip.*Hop",
                    "$options":"i" } }

$or
  e.g. { "$or" : [ {"zip": 94114}, {"area": 415} ] }

$not (negation of $ operators)
  e.g. { "children" : { "$not" : {"$size":1 } } }

Conditional Operators

$gt  - greater than <value>, $lt - less than <value>,
$gte - greater than or equal to <value>,
$lte - less than or equal to <value>
  e.g. { "size" : {"$lt": 3} }

$ne - not equals <value>
  e.g. { "status" : {"$ne": "complete"} }

$mod - mod <divisor> is <value>
  syntax: { <field> : {"$mod":[<divisor>, <value>]} }
  e.g. { "size" : {"$mod": [10, 1]} }

$in - in <array>, $nin - not in <array>
  e.g. { "size" : {"$in": [1, 2, 3]} }

$exists - field present or missing
  e.g. { "nicknames" : {"$exists": false} }

$type - BSON type (reference) is <type number>
  e.g. { "name" : {"$type": 2} }
       // matches if "name" is a String

$all - array matches all of the elements in <array>
  e.g. { "nicknames": { "$all": ["Liz", "Beth"] } }

$size - array has specified # of elements
```

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

Project attributes to show or hide
Show: 1
Hide : 0

boolean value

FILTER {"employee_salary": {$gte: '2100',$lte: '2200'}}
OPTIONS
PROJECT {"employee_name": 1}
SORT { field: -1 } or [['field', -1]]
MAX TIME MS 60000
COLLATION { locale: 'simple' }
SKIP 0
LIMIT 0

VIEW

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_name: "Eduardo Albuquerque"

_id: ObjectId("5ffc39abcf724c5863ca544c")
employee_name: "Fernando Carlos Ferreira"
```

FILTER

```
{
 employee_salary: {
  $gte: '2100',
  $lte: '2200'
 }
}
```

PROJECT

```
{
 employee_name: 1
}
```

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

Project attributes to show or hide
Show: 1
Hide : 0

boolean value

FILTER `{"employee_salary": {$gte: '2100',$lte: '2200'}}`

OPTIONS

PROJECT `{"employee_id": 1,"employee_name": 1}`

SORT `{ field: -1 } or [['field', -1]]`

MAX TIME MS 60000

COLLATION `{ locale: 'simple' }`

SKIP 0

LIMIT 0

VIEW

Displaying documents 1 - 2

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
```

```
_id: ObjectId("5ffc39abcf724c5863ca544c")
employee_id: "10"
employee_name: "Fernando Carlos Ferreira"
```

```
{
 employee_salary: {
  $gte: '2100',
  $lte: '2200'
 }
}
```

PROJECT

```
{
 employee_id: 1,
 employee_name: 1
}
```

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

Project attributes to show or hide
Show: 1
Hide : 0

boolean value

FILTER {"employee_salary": {$gte: '2100',$lte: '2200'}}
OPTIONS
PROJECT {"employee_salary": 0}
SORT { field: -1 } or [['field', -1]]
MAX TIME MS 60000
COLLATION { locale: 'simple' }
SKIP 0
LIMIT 0

VIEW

```
_id: ObjectId("5ffc3687cf724c5863ca5444")
employee_id: "01"
employee_name: "Eduardo Albuquerque"
employee_depart: Array
  0: "sales"
  1: "Accounting"
employee_adress: Object
```

FILTER

```
{
 employee_salary: {
  $gte: '2100',
  $lte: '2200'
 }
}
```

PROJECT

```
{
 employee_salary: 0
}
```

MongoDB

R • find()

Project attributes to show or hide
Show: 1
Hide : 0

boolean value

MongoDB

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_depart": "Computing"} OPTIONS
PROJECT {"employee_name": 1}
SORT { field: -1 } or [['field', -1]] MAX TIME MS 60000
COLLATION { locale: 'simple' } SKIP 0 LIMIT 0

VIEW Displaying documents 1

```
_id: ObjectId("5ffc38f3cf724c5863ca5448")
employee_name: "Carla Maria Cardoso"

_id: ObjectId("5ffc394ccf724c5863ca544a")
employee_name: "Fátima Maria Cardoso"

_id: ObjectId("5ffc3975cf724c5863ca544b")
employee_name: "Vítor Manuel Marques"
```

FILTER

```
{
 employee_depart: 'Computing'
}
```

PROJECT

```
{
 employee_name: 1
}
```

MongoDB

R • find()

To sort a cursor of documents:
Ascending: 1
Descending : -1

MongoDB

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_depart": "Computing"} OPTIONS
PROJECT {"employee_name": 1}
SORT {"employee_name": -1} MAX TIME MS 60000
COLLATION { locale: 'simple' } SKIP 0 LIMIT 0

VIEW Displaying documents 1 -

```
_id: ObjectId("5ffc3975cf724c5863ca544b")
employee_name: "Vítor Manuel Marques"

_id: ObjectId("5ffc394ccf724c5863ca544a")
employee_name: "Fátima Maria Cardoso"

_id: ObjectId("5ffc38f3cf724c5863ca5448")
employee_name: "Carla Maria Cardoso"
```

FILTER
```
{
 employee_depart: 'Computing'
}
```

PROJECT
```
{
 employee_name: 1
}
```

SORT
```
{
 employee_name: -1
}
```

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_depart": "Computing"} OPTIONS
PROJECT {"employee_id": 1, "employee_name": 1}
SORT {"employee_name": 1} MAX TIME MS 60000
COLLATION { locale: 'simple' } SKIP 0 LIMIT 0

VIEW Displaying documents 1

```
_id: ObjectId("5ffc38f3cf724c5863ca5448")
employee_id: "05"
employee_name: "Carla Maria Cardoso"

_id: ObjectId("5ffc394ccf724c5863ca544a")
employee_id: "06"
employee_name: "Fátima Maria Cardoso"

_id: ObjectId("5ffc3975cf724c5863ca544b")
employee_id: "07"
employee_name: "Vítor Manuel Marques"
```

FILTER

```
{
 employee_depart: 'Computing'
}
```

PROJECT

```
{
 employee_id: 1,
 employee_name: 1
}
```

SORT

```
{
 employee_name: 1
}
```

MongoDB

R • find()

MongoDB

- **db.*collection_name*.find(query, projection)**

FILTER {"employee_depart": "Computing"} OPTIONS
PROJECT { field: 0 }
SORT {"employee_salary": -1, "employee_name": 1} MAX TIME MS 60000
COLLATION { locale: 'simple' } SKIP 0 LIMIT 0

ADD DATA VIEW Displaying documents 1

```
_id: ObjectId("5ffc38f3cf724c5863ca5448")
employee_id: "05"
employee_name: "Carla Maria Cardoso"
employee_salary: "2350"
> employee_depart: Array
> employee_adress: Object

_id: ObjectId("5ffc394ccf724c5863ca544a")
employee_id: "06"
employee_name: "Fátima Maria Cardoso"
employee_salary: "2350"
> employee_depart: Array
> employee_adress: Object
```

FILTER

```
{
 employee_depart: 'Computing'
}
```

SORT

```
{
 employee_salary: -1,
 employee_name: 1
}
```

MongoDB

U • update()

MongoDB

- **db.*collection_name*.update({criteria}, {value updated})**

Other methods:

MongoDB provides the following methods for updating documents in a collection:

`db.collection.updateOne()`	Updates at most a single document that match a specified filter even though multiple documents may match the specified filter.
`db.collection.updateMany()`	Update all documents that match a specified filter.
`db.collection.replaceOne()`	Replaces at most a single document that match a specified filter even though multiple documents may match the specified filter.

MongoDB

U • update()

- **db.*collection_name*.update({*criteria*}, {*value updated*})**
  The update method updates the values in the existing document

$set allows modifying the content of a particular field

$unset allows you to remove a field from a document

```
db.employee.update(
  { "employee_id": 2},
  {
    $set: { "employee_salary ": "2500" }
  } )
```

Update salary to 2500

MongoDB

U • update()

MongoDB

- **db.*collection_name*.update({criteria}, {value updated})**

- $set allows modifying the content of a particular field
- $unset allows you to remove a field from a document

```
db.employee.update(
  { "employee_id": 2},
  {
    $set: { "employee_depart.0 ": “Computing” }
  } )
```

Updates the first occurrence in array list

MongoDB

U • update()

MongoDB

- **db.*collection_name*.update({criteria}, {value updated})**

- $set allows modifying the contents of a particular field
- $unset allows you to remove a field from a document

```
db.employee.update(
  { "employee_id": 2},
  {
    $unset: { "employee_depart ": 1 }
  } )
```

Removes second occurrence in the array list
(the first occurrence has index 0)

MongoDB

U • update()

MongoDB

- **db.*collection_name*.update({criteria}, {value updated})**

$ push to add a field to a document array

$ pop do remove a value from the array

```
db.employee.update(
  { "employee_id": 2},
  {
    $push: { "employee_depart ": “Production” }
  } )
```

adds the *Production* department at the end of the array field

MongoDB

U • update()

MongoDB

- **db.*collection_name*.update({criteria}, {value updated})**

$ pop: 1 remove the value that is most right (the last array element)

$ pop: -1 remove the value that is most left (the first array element)

Production

```
db.employee.update(
  { "employee_id": 2},
  {
    $pop: { "employee_depart ": 1 }
  } )
```