

BASES DE DADOS
Módulo III

MongoDB

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A
WEB

Agenda

- ❖ MongoDB Structure
- ❖ MongoDB Syntax
- ❖ *MongoDB CRUD operations*

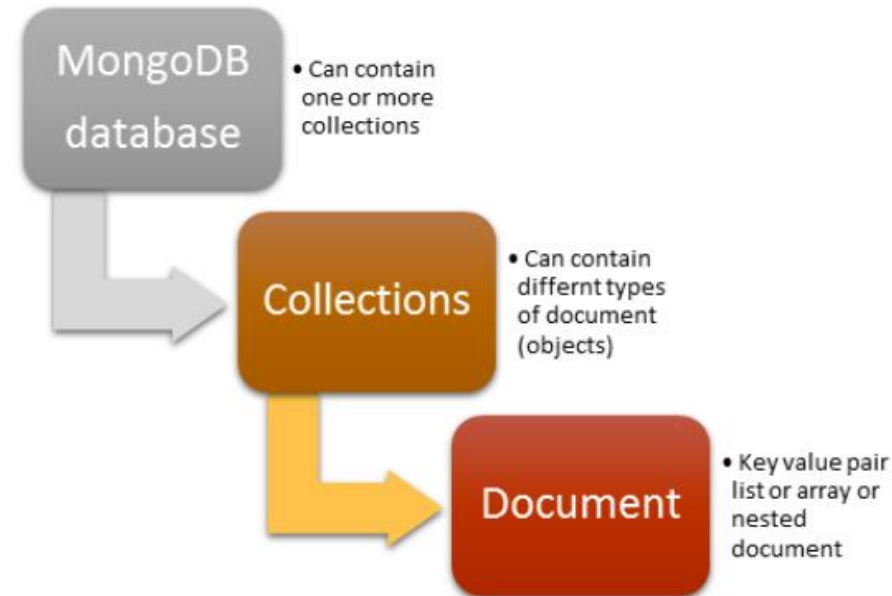


C	• insert()
R	• find()
U	• update()
D	• remove()

MongoDB

MongoDB Sctucture

MySQL	MongoDB
<ul style="list-style-type: none">• Database• Table• Rows• Column• Joins• Primary Key	<ul style="list-style-type: none">• Database• Collection• Document• Field (document field)• Embebed documents• Primary Key (default id_key provided by MongoDB)



MongoDB

MongoDB Sctucture



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

← field: value
← field: value
← field: value
← field: value

```
{  
  name: "al",  
  age: 18,  
  status: "D",  
  groups: [ "politics", "news" ]  
}
```

Collection

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

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 **documents**, such as:

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


MongoDB

MongoDB – Based on documents, 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

```
> 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"
      }
    ]
  }
})
```

address attribute
is an array of objects



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")
  ]
}
```

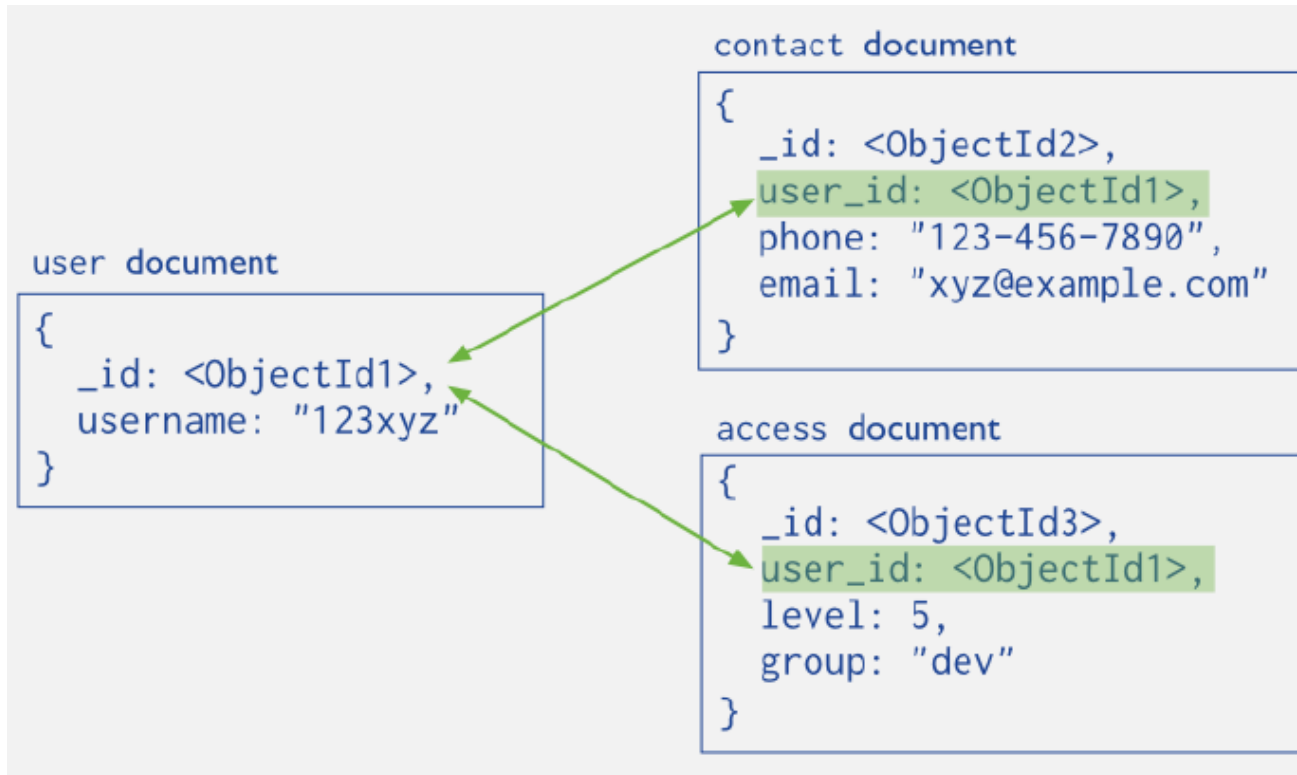
```
{
  "_id": ObjectId("52ffc4a5d85242602e000000"),
  "building": "22 A, Indiana Apt",
  "pincode": 123456,
  "city": "Los Angeles",
  "state": "California"
}
```

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

↑

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

```
> 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

{}

≡

```
1 ▾ db.employees.insert(  
2 ▾ {  
3   "employee_id": "11",  
4   "employee_name": "Ricardo Santos",  
5   "employee_salary": "1980",  
6 ▾  "employee_depart": [  
7     "Sales",  
8     "Logistics"  
9   ],  
10 ▾ "employee_adress": {  
11   "street": "Rua dos Banhos",  
12   "number": "981",  
13   "city": "Vila do Conde"  
14 }  
15 })
```

MongoDB

C

• insert()

► **db.collection_name.i**

To insert multiple documents
in a single query.

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

```
>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
      }
    ]
  }
])
```

MongoDB

C

• insert()

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

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

```
1 ▾ db.employees.insertOne(  
2 ▾ {  
3   "employee_id": "01",  
4   "employee_name": "Eduardo Albuquerque",  
5   "employee_salary": "2150",  
6 ▾   "employee_depart": [  
7     "sales",  
8     "Accounting"  
9   ],  
10 ▾   "employee_adress": {  
11     "street": "Rua da Bela Vista",  
12     "number": "180",  
13     "city": "Porto"  
14   }  
15 }  
15 })|
```

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.**



```
1 ▾ db.employees.insertMany([
2 ▾ {
3   "employee_id": "01",
4   "employee_name": "Eduardo Albuquerque",
5   "employee_salary": "2150",
6 ▾   "employee_depart": [
7     "sales",
8     "Accounting"
9   ],
10 ▾   "employee_adress": {
11     "street": "Rua da Bela Vista",
12     "number": "180",
13     "city": "Porto"
14   }
15 },
16 ▾ {
17   "employee_id": "02",
18   "employee_name": "Liliana Ferreira",
19   "employee_salary": "1850",
20 ▾   "employee_depart": [
21     "sales",
22     "Production"
23   ]
24 }
```

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
 - ▶ Comparison (`$eq`, `$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:

The screenshot shows the MongoDB 4.4.3 Community interface. On the left, the 'MyBusiness' database is selected, and the 'employees' collection is highlighted. The main area displays two sample documents from the collection:

```
{
  "_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": "5ffc385ecf724c5863ca5445"
  },
  "employee_id": "02",
  "employee_name": "Liliana Ferreira",
  "employee_salary": "1850",
  "employee_depart": ["sales", "Production"],
  "employee_adress": {
    "street": "Rua das Marinhas",
    "number": "180",
    "city": "Vila do Conde"
  }
}
```

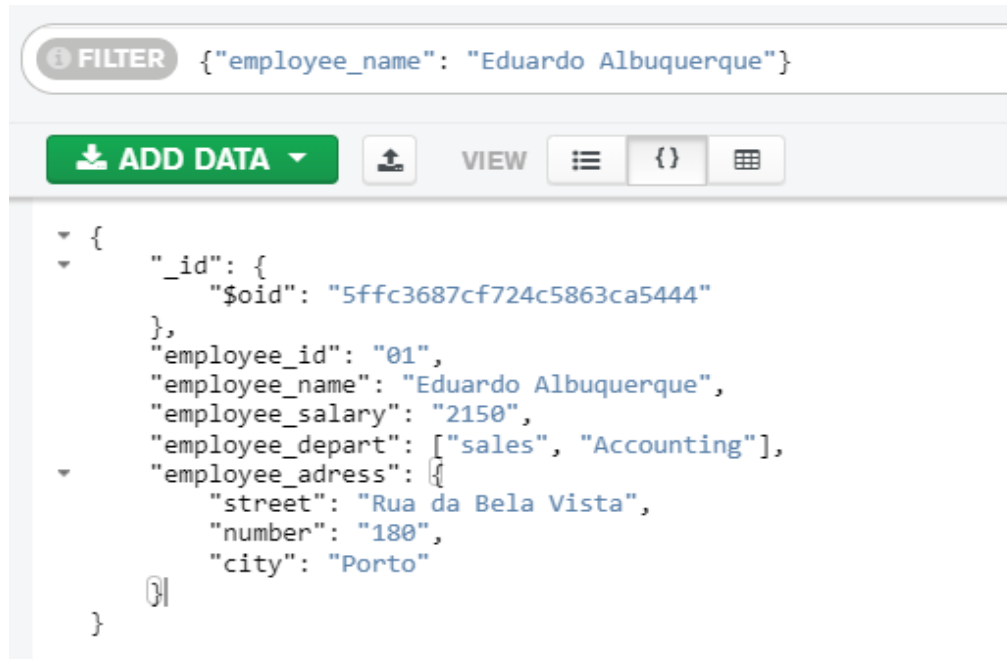
MongoDB

R

• find()

MongoDB

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



The screenshot shows the MongoDB Compass web interface. At the top, there is a 'FILTER' button and a text input containing the query `{"employee_name": "Eduardo Albuquerque"}`. Below this, there is a green 'ADD DATA' button and a 'VIEW' section with icons for JSON, BSON, and GridFS. The main area displays a single document in JSON format, with some fields expanded to show their internal structure.

```
{
  "_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)**

The screenshot shows the MongoDB Compass interface. At the top, there is a filter bar with the text `{"employee_adress.city": "Porto"}`. To the right of the filter bar are buttons for **OPTIONS**, **FIND**, and **RESET**. Below the filter bar is a toolbar with **ADD DATA**, **VIEW**, and icons for list, JSON, and table views. The main area displays two JSON documents. The first document is for employee Eduardo Albuquerque, and the second is for Maria Mariazinha. Both documents include fields for employee_id, employee_name, employee_salary, employee_depart, and employee_adress (with street, number, and city).

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)**

The screenshot shows the MongoDB Compass web interface. At the top, there is a filter bar with the text `{"employee_depart": "Computing"}` and an **OPTIONS** button. Below the filter bar, there is a toolbar with a green **ADD DATA** button, a **VIEW** button, and icons for list, JSON, and grid views. The main area displays two documents. The first document is expanded, showing its fields: `_id`, `$oid`, `employee_id`, `employee_name`, `employee_salary`, `employee_depart`, and `employee_adress`. The second document is also expanded, showing similar fields but with a different `$oid` and `employee_name`.

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

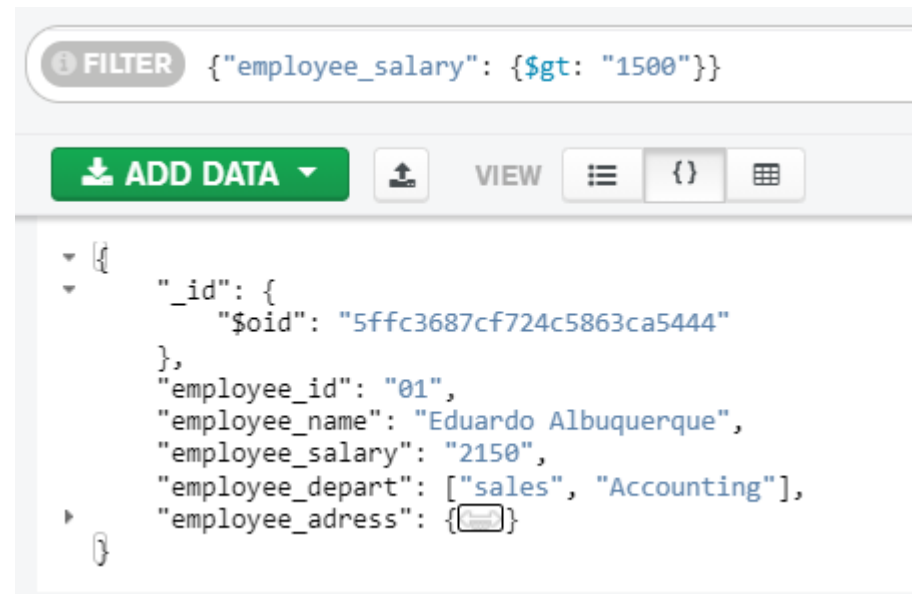
R

• find()

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

Restrictions on queries - **operators**:

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



MongoDB

R

• find()

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

Restrictions on queries - **operators**:

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

The screenshot shows the MongoDB Compass interface. At the top, there is a 'FILTER' button and a filter query: `{"employee_salary": {"$gte": "2350"}}`. Below the filter, there are buttons for 'ADD DATA', 'VIEW', and a dropdown menu. The main area displays two JSON documents. The first document is for an employee with ID '05', name 'Carla Maria Cardoso', salary '2350', and departments 'sales' and 'Computing'. The second document is for an employee with ID '06', name 'Fátima Maria Cardoso', salary '2350', and department 'Computing'. Both documents have an 'employee_adress' field with a placeholder icon.

```
{
  "_id": {
    "$oid": "5fffc38f3cf724c5863ca5448"
  },
  "employee_id": "05",
  "employee_name": "Carla Maria Cardoso",
  "employee_salary": "2350",
  "employee_depart": ["sales", "Computing"],
  "employee_adress": {}
}
```

```
{
  "_id": {
    "$oid": "5fffc394ccf724c5863ca544a"
  },
  "employee_id": "06",
  "employee_name": "Fátima Maria Cardoso",
  "employee_salary": "2350",
  "employee_depart": ["Computing"],
  "employee_adress": {}
}
```

MongoDB

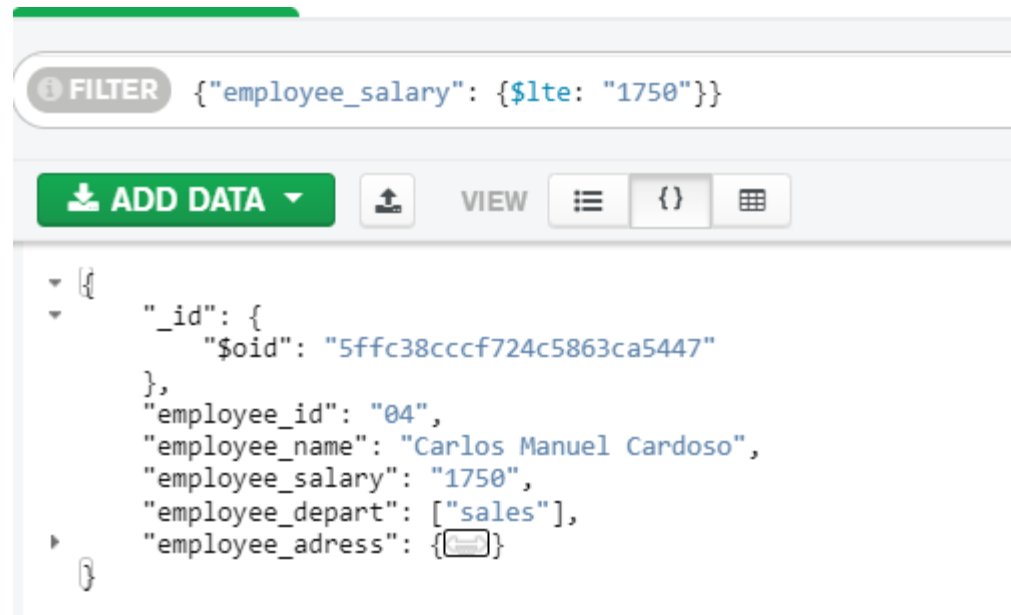
R

• 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 on

- ☐ `$gt`
- ☐ `$gte`
- ☐ `$lt`
- ☐ `$lte`

The screenshot shows the MongoDB Compass interface. At the top, a filter bar contains the query: `{"employee_salary": {"$gt": "2000", "$lt": "2300"}}`. Below the filter bar, there are buttons for "ADD DATA", "VIEW", and icons for list, JSON, and table views. The main area displays two JSON documents. The first document is for Eduardo Albuquerque with employee_id "01" and salary "2150". The second document is for Fernando Carlos Ferreira with employee_id "10" and salary "2150". Both documents have an "employee_depart" array and an "employee_adress" object.

```
{
  "_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` <=

The screenshot shows the MongoDB Compass interface. At the top, a 'FILTER' button is followed by the query: `{"employee_salary": {"$gte": "2100", "$lte": "2200"}}`. Below the filter, there are buttons for 'ADD DATA', 'VIEW', and icons for document, list, and table views. The main area displays two JSON documents. The first document is for 'Eduardo Albuquerque' with an employee_id of '01' and a salary of '2150'. The second document is for 'Fernando Carlos Ferreira' with an employee_id of '10' and a salary of '2150'. Both documents show their respective departments and addresses.

```
{
  "_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.

The screenshot shows the MongoDB Compass web interface. At the top, there is a 'FILTER' button and a query input field containing the following JSON query: `{ "$and": [{ "employee_salary": { "$gt": '2000' } }, { "employee_salary": { "$lt": '2200' } }] }`. To the right of the query field is a '► OF' button. Below the query field is a toolbar with a green 'ADD DATA' button, an upload icon, a 'VIEW' label, and three view mode icons (list, JSON, grid). On the right side of the toolbar, it says 'Displaying docu'. The main area displays two documents in a list view. Each document is shown with its fields and values, with expandable sections for arrays and objects.

```
{ "_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



Display

```
_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}  
    ]  
  }  
)<pre>.pretty()
```

MongoDB

R

• find()

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

Logical operator: **\$or**

The screenshot shows the MongoDB Compass interface. At the top, a filter bar contains the query: `{ $or: [{"employee_adress.city": "Porto"}, {"employee_adress.city": "Matosinhos"}]}`. Below the filter bar, there are buttons for "ADD DATA", "VIEW", and a "DISPLAY" button. The main area shows two JSON documents. The first document is for an employee named Eduardo Albuquerque, and the second is for Maria Mariazinha. Both documents have an "employee_adress" field with a "city" property.

```
{
  "_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

The screenshot shows the MongoDB Compass interface. At the top, there is a 'FILTER' button and a query input field containing: `{"employee_depart": {"$in": ['sales', 'Computing', 'Production']}}`. Below the query field, there is a toolbar with buttons for 'ADD DATA', 'VIEW', and icons for list, JSON, and table views. The 'VIEW' button is currently selected. The results pane shows two documents. The first document is expanded, showing its fields: `_id`, `employee_id`, `employee_name`, `employee_salary`, `employee_depart` (an array with 'sales' and 'Accounting'), and `employee_adress` (an object). The second document is collapsed, showing only its `_id` field.

```
{
  "_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")
}
```

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 one of the list values

The screenshot shows the MongoDB Compass interface. At the top, there is a 'FILTER' button and a query: `{"employee_adress.city": {"$in": ['Vila do Conde', 'Matosinhos']}}`. Below the filter bar, there is a green 'ADD DATA' button, an upload icon, and a 'VIEW' section with icons for list, JSON, and grid views. On the right, there is a 'Display' button. The main area shows a document with the following fields:

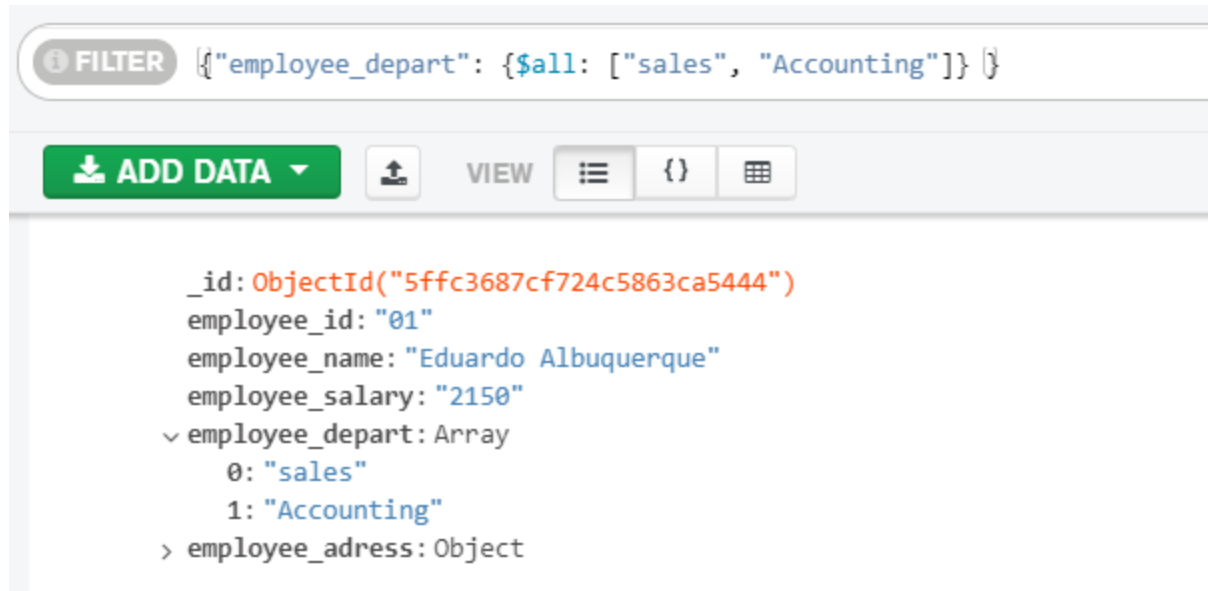
```
_id: ObjectId("5ffc385ecf724c5863ca5445")
employee_id: "02"
employee_name: "Liliana Ferreira"
employee_salary: "1850"
> employee_depart: Array
v employee_adress: Object
  street: "Rua das Marinhas"
  number: "180"
  city: "Vila do Conde"
```

MongoDB

R

• find()

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



The screenshot shows the MongoDB Compass interface. At the top, there is a 'FILTER' button and a query field containing `{"employee_depart": {"$all": ["sales", "Accounting"]}}`. Below the filter bar, there is a green 'ADD DATA' button and a 'VIEW' section with icons for list, JSON, and grid views. The main area displays a document in JSON format:

```
{
  "_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

The screenshot shows the MongoDB Compass interface. At the top, there is a filter bar with the text: `{'employee_adress.city': {'$ne': "Porto"}} {}`. Below the filter bar, there is a toolbar with buttons for "ADD DATA", "VIEW", and "Displ". The main area displays a document with the following fields:

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

MongoDB

MongoDB

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

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"
```

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

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

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()

MongoDB

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

Project attributes to show or hide

Show: **1**

Hide: **0**

boolean value

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()

MongoDB

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

To sort a cursor of documents:

Ascending: |

Descending : -|

The screenshot displays the MongoDB Compass web interface. At the top, there are tabs for FILTER, PROJECT, SORT, and COLLATION. The FILTER tab is active, showing the query `{"employee_depart": "Computing"}`. The PROJECT tab shows `{"employee_name": 1}`. The SORT tab shows `{"employee_name": -1}`. The COLLATION tab shows `{ locale: 'simple' }`. On the right, there are controls for MAX TIME MS (60000), SKIP (0), and LIMIT (0). Below the query builder, there are icons for VIEW, a list view icon, a JSON view icon, and a grid view icon. The text "Displaying documents 1 -" is visible. The main area shows three documents in a list view. Each document has an `_id` field (an ObjectId) and an `employee_name` field. The documents are: 1. `_id: ObjectId("5ffc3975cf724c5863ca544b")`, `employee_name: "Vitor Manuel Marques"`. 2. `_id: ObjectId("5ffc394ccf724c5863ca544a")`, `employee_name: "Fátima Maria Cardoso"`. 3. `_id: ObjectId("5ffc38f3cf724c5863ca5448")`, `employee_name: "Carla Maria Cardoso"`. On the right side, there are three panels showing the query, project, and sort specifications in JSON format.

```
db.collection_name.find({"employee_depart": "Computing"}, {"employee_name": 1}, {"employee_name": -1}, { locale: 'simple' })
```

Displaying documents 1 -

_id	employee_name
ObjectId("5ffc3975cf724c5863ca544b")	Vitor Manuel Marques
ObjectId("5ffc394ccf724c5863ca544a")	Fátima Maria Cardoso
ObjectId("5ffc38f3cf724c5863ca5448")	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)**

The screenshot shows the MongoDB Compass interface. At the top, there's a query editor with the following settings:

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

Below the query editor, there's a "VIEW" section with icons for JSON, Grid, and Table views. The "Displaying documents 1" text is visible. The results are shown in a table-like format with three documents:

Document
<pre>{ "_id": ObjectId("5ffc38f3cf724c5863ca5448"), "employee_id": "05", "employee_name": "Carla Maria Cardoso" }</pre>
<pre>{ "_id": ObjectId("5ffc394ccf724c5863ca544a"), "employee_id": "06", "employee_name": "Fátima Maria Cardoso" }</pre>
<pre>{ "_id": ObjectId("5ffc3975cf724c5863ca544b"), "employee_id": "07", "employee_name": "Vitor Manuel Marques" }</pre>

On the right side, there's a sidebar showing the query components:

- 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"}`

ⓘ PROJECT

`{ field: 0 }`

ⓘ SORT

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

ⓘ COLLATION

`{ locale: 'simple' }`

ⓘ MAX TIME MS

60000

ⓘ SKIP

0

ⓘ LIMIT

0

⌵ OPTIONS

⬇️ 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 }  
  })
```