B4M36DS2, BE4M36DS2: **Database Systems 2**

https://cw.fel.cvut.cz/wiki/courses/b4m36ds2/

Practical Class 7

MongoDB

Yuliia Prokop
prokoyul@fel.cvut.cz

11. 11. 2024

Authors: Martin Svoboda
(martin.svoboda@matfyz.cuni.cz),
Yuliia Prokop

Czech Technical University in Prague, Faculty of Electrical Engineering

First steps

Connect to our NoSQL server

- SSH / PuTTY and SFTP / WinSCP
- nosql.felk.cvut.cz

Start mongo shell

mongosh --port 42222 -u login -p password database

database=login, password – from the first email

Switch to your database

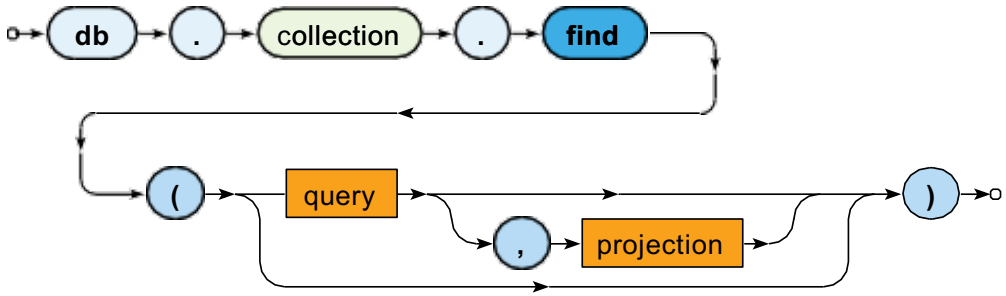
- **use** login

Insert sample data into your database

- Empty your collections first
- See /home/DS2/mongodb/data.js

Find Operation

Selects documents from a given collection



- Parameters
  - **Query**: description of documents to be selected
  - **Projection**: fields to be included / excluded in the result

Exercise 1

Express the following MongoDB query

- **Find movies filmed between years *2000* and *2005* such that they have a director specified**
- Return movie identifier only
- Order the result by ratings in descending order and then by years in ascending order

Exercise 1 - Solution

```
db.movies.find(
    {
        year: { $gte: 2000, $lte: 2005 },
        director: { $exists: 1 }
    },
    { _id: 1 }
).sort(
    { rating: -1, year: 1 }
);
/*INCORRECT*/
db.movies.find(
    {
        year: { $gte: 2000 },
        year: { $lte: 2005 },
        director: { $exists: 1 }
    },
    { _id: 1 }
).sort(    { rating: -1, year: 1 });
```

Exercise 2

Express the following MongoDB query

- **Find actors who stared in *Samotari* or *Medvidek* movies**
- Return actor identifier only
- Propose two different approaches

Exercise 2 - Solution

```
db.actors.find(
        { movies: { $in: [ "medvidek", "samotari" ] } },
        { _id: 1 }
);

db.actors.find(
    {
        $or: [
            { movies: "medvidek" },
            { movies: "samotari" }
        ]
    },
    { _id: 1 }
);
```

Exercise 3

Express the following MongoDB query

- **Find actors who played in both *Samotari* and *Medvidek***
- Return actor identifier only
- Propose two different approaches

Exercise 3 - Solution

```
db.actors.find(
        { movies: { $all: [ "medvidek", "samotari" ] } },
        { _id: 1 }
);

db.actors.find(
        { $and: [ { movies: "medvidek" }, { movies: "samotari" } ]  },
        { _id: 1 }
);
```

Exercise 3 - Solution

```
db.actors.find(
    { $and: [
            { movies: { $eq: "medvidek" } },
            { movies: { $eq: "samotari" } }
    ] },
    { _id: 1 }
);

db.actors.find(
    { $and: [
            { movies: { $in: [ "medvidek" ] } },
            { movies: { $in: [ "samotari" ] } }
    ] },
    { _id: 1 }
);
```

Exercise 4

Express the following MongoDB query

- **Find movies with Czech title equal to *Vratne lahve***
- Return movie title only
- Note that there are two means how movie titles are defined

Exercise 4 - Solution

```
db.movies.find(
    { $or: [
            { title: "Vratne lahve" },
            { "title.cs": "Vratne lahve" }
    ] },
    { title: 1, _id: 0 }
);
db.movies.find(
    { $or: [
            { title: { $eq: "Vratne lahve" } },
            { "title.cs": { $eq: "Vratne lahve" } }
    ] },
    { title: 1, _id: 0 }
);
```

Exercise 5

Express the following MongoDB query

- **Find movies that have a *Czech Lion* award from *2005***
- Return movie identifier and all awards

Exercise 5 - Solution

- **Find movies that have a *Czech Lion* award from *2005***
- Return movie identifier and all awards

```
/*INCORRECT*/
db.movies.find(
        { awards: { type: "Czech Lion", year: 2005 } },
        { _id: 1, awards: 1 }
);

/*INCORRECT*/
db.movies.find(
        { awards: { $in : [ { type: "Czech Lion", year: 2005 } ]} },
        { _id: 1, awards: 1 }
);
```

Exercise 5 - Solution

```
/*INCORRECT*/
db.movies.find(
        { "awards.type": "Czech Lion", "awards.year": 2005 },
        { _id: 1, awards: 1 }
);

/*INCORRECT*/
db.movies.find(
        { $and: [ { "awards.type": "Czech Lion" },
        { "awards.year": 2005 } ] }, { _id: 1, awards: 1 }
);

db.movies.find(
        { awards: { $elemMatch: { type: "Czech Lion", year: 2005 } } },
        { _id: 1, awards: 1 }
);
```

Exercise 5a

Express the following MongoDB query

- **Find movies that have a *Czech Lion* award from years between *2000 and 2007***
- Return movie identifier and all awards

Exercise 5a - Solution

```
db.movies.find(
    {
        awards: {
            $elemMatch: {
                type: "Czech Lion",
                year: {$gte: 2000, $lte: 2007 }
            }
        }
    },
    { _id: 1, awards: 1 }
);
```

```
[
  { _id: 'stesti', awards: [ { type: 'Czech Lion', year: 2005 } ] }
]
```

Exercise 6

Express the following MongoDB query

- **Find movies that are *comedies* and *dramas* at the same time**
  or
  have a rating *80* or more
- Return movie identifier and at most 2 countries

Exercise 6 - Solution

```
db.movies.find(
	{
		$or: [
			{ genres: { $all: [ "comedy", "drama" ] } },
			{ rating: { $gte: 80 } }
		]
	},
	{ _id: 1, country: { $slice: 2 } }
);

db.movies.find(
	{
		$or: [
			{ $and: [ { genres: "comedy" }, { genres: "drama" } ] },
			{ rating: { $gte: 80 } }
		]
	},
	{ _id: 1, country: { $slice: 2 } }
);
```

Exercise 7

Express the following MongoDB query

- **Find all users whose interests are "football" and "diving"**

Exercise 7 - Solution

```
db.users.find({ "interests" : ["football", "diving"] })
```

```
[
  {
    _id: 1,
    registration_date: ISODate("2016-12-13T18:55:16.981Z"),
    interests: [ 'football', 'diving' ],
    name: 'Roland',
    surname: 'Blažek',
    birthdate: ISODate("2002-02-24T23:53:07.663Z"),
    home: { city: 'Prague' },
    friends: [ 5, 8, 17, 20 ]
  }
]
```

Exercise 7 - Solution

```
db. users.find(
      {  interests : { $all: ["football", "diving"] } }
);

db. users.find(
                  { $and: [ {interests : "football"}, {interests : "diving" } ] }
 );
```

```
[
  {
    _id: 1,
    registration_date: ISODate("2016-12-13T18:55:16.981Z"),
    interests: [ 'football', 'diving' ],
    name: 'Roland', surname: 'Blažek',
    birthdate: ISODate("2002-02-24T23:53:07.663Z"),
    home: { city: 'Prague' },
    friends: [ 5, 8, 17, 20 ]
  },
  {
    _id: 10,
    registration_date: ISODate("2011-05-27T18:55:16.981Z"),
    interests: [ 'diving', 'football' ],
    name: 'Xenie', surname: 'Kratochvílová',
    birthdate: ISODate("1997-08-24T23:53:07.663Z"),
    home: { city: 'Prague', address: 'Karlovo namesti 17' },
    friends: [ 17 ]
  }
]
```

Exercise 8

Express the following MongoDB query

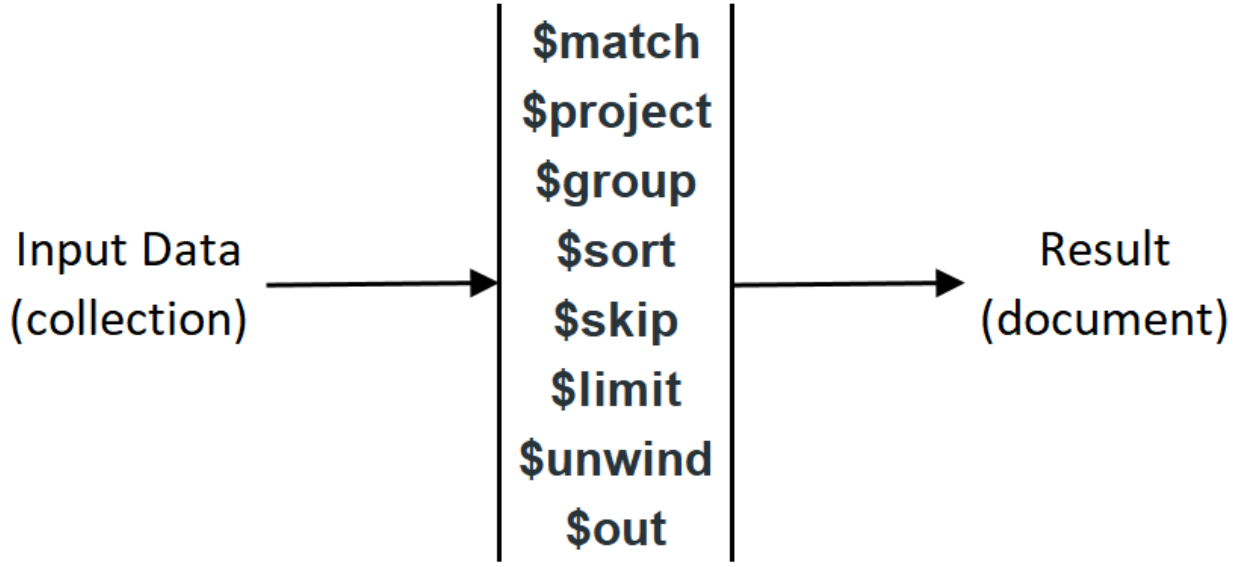
- **Find *Prague* and *Brno* residents born in the *last century***

Exercise 8 - Solution

```
db.users.find({
        "home.city": {$in: ["Prague", "Brno"]},
        "birthdate": {$lt: new Date("2000-01-01")}
});
```

```
[
 {
  _id: 4,
  registration_date: ISODate("2014-07-12T18:55:16.981Z"),
  interests: [ 'football', 'architecture' ], name: 'Vladimír', surname: 'Poláček',
  birthdate: ISODate("1996-01-24T23:53:07.663Z"),
  home: { city: 'Brno' }, friends: [ 6, 7, 14, 18, 19 ]
 },
 {
  _id: 10,
  registration_date: ISODate("2011-05-27T18:55:16.981Z"),
  interests: [ 'diving', 'football' ], name: 'Xenie',  surname: 'Kratochvílová',
  birthdate: ISODate("1997-08-24T23:53:07.663Z"),
  home: { city: 'Prague', address: 'Karlovo namesti 17' }, friends: [ 17 ]
 },
 {
  _id: 15,
  registration_date: ISODate("2009-05-27T18:55:16.981Z"),
  interests: [ 'basketball' ], name: 'Simona', surname: 'Ševčíková',
  birthdate: ISODate("1998-12-24T23:53:07.663Z"),
  home: { city: 'Prague', address: 'Vysehradska 425/31' }, friends: [ 6, 17 ]
 }
]
```

Aggregation pipeline




```
db.collection.aggregate( [ { <stage> }, ... ] )
```

Exercise 9

Express the following MongoDB query

- **Find the maximal number of friends.**

Exercise 9 - Solution

Express the following MongoDB query

- **Find the maximal number of friends.**

```
db.users.aggregate([
    { $addFields: { num_friends: { $size: "$friends" } } },
    { $group: { _id: null, max_friends: { $max: "$num_friends" }}}
])
```

```
[ { _id: 'num_friends', max_friends: 6 } ]
```

Exercise 10

Express the following MongoDB query

- **Find the number of users who are interested in both soccer and parties**

Exercise 10

Express the following MongoDB query

- **Find the number of users who are interested in both soccer and parties**

```
db.users.aggregate([
        { $match: { "interests" : { $all: ["football", "party"] } } },
        { $count: "football_and_party_fans" }
])
```

```
[ { football_and_party_fans: 1 } ]
```

Exercise 11

Express the following MongoDB query

- **Find top three cities for vaccations**

Exercise 11 - Solution

Express the following MongoDB query

- **Find top three cities for vaccations**

```
db.checkins.aggregate([
        { $group : { _id : "$name", count : { $sum: 1 } } },
        { $sort : { count : -1 } },
        { $limit : 3 }
])
```

```
[
  { _id: 'Prague', count: 2 },
  { _id: 'Brno', count: 2 },
  { _id: 'Liberec', count: 2 }
]
```

Exercise 12

Express the following MongoDB query

- **Find users who have friends with _id 1 and 2 and sort them by registration date in descending order.**

Exercise 12 - Solution

```
db.users.aggregate([
        { $match: { "friends" : { $all: [1, 2] } } },
        { $sort: { registration_date: -1 } }
])
```

```
[
  {
    _id: 17,
    registration_date: ISODate("2020-03-27T18:55:16.981Z"),
    interests: [ 'basketball', 'design' ],
    name: 'Miloš',
    surname: 'Beran',
    birthdate: ISODate("2001-02-24T23:53:07.663Z"),
    home: { city: 'Brno', address: 'Lipová 24' },
    friends: [ 1, 2, 4, 10, 15, 20 ]
  },
  {
    _id: 20,
    registration_date: ISODate("2018-08-27T18:55:16.981Z"),
    interests: [ 'coding', 'party' ],
    name: 'Vlastislav',
    surname: 'Hladík',
    birthdate: ISODate("1998-09-24T23:53:07.663Z"),
    friends: [ 1, 2, 5, 11, 17, 18 ]
  }
]
```

Exercise 13

Express the following MongoDB query

- **Find vacation places, which are popular in winter.**

Exercise 13 - Solution

Express the following MongoDB query

- **Find vacation places, which are popular in winter.**

```
db.checkins.aggregate([
        { $addFields : { "winter": { $month : "$timestamp" }}},
        { $match : {"winter" : { $in : [12, 1, 2] } } } ,
        { $group : {_id : "$name", count : { $sum : 1 }}},
        { $sort : {count: -1}}
])
```

```
[ { _id: 'Liberec', count: 2 }, { _id: 'Pec', count: 1 } ]
```

Exercise 14

Express the following MongoDB query

- **Find users who have more than 3 friends and sort them by the number of friends in descending order.**

Exercise 14 - Solution

Express the following MongoDB query

- **Find users who have more than 3 friends and sort them by the number of friends in descending order.**

```
db.users.aggregate([
    { $project: { name: 1, num_of_friends: { $size: "$friends" } } },
    { $match: { num_of_friends: { $gt: 3 } } },
    { $sort: { num_of_friends: -1 } }
])
```

```
[
  { _id: 17, name: 'Miloš', num_of_friends: 6 },
  { _id: 20, name: 'Vlastislav', num_of_friends: 6 },
  { _id: 2, name: 'Vilém', num_of_friends: 5 },
  { _id: 4, name: 'Vladimír', num_of_friends: 5 },
  { _id: 1, name: 'Roland', num_of_friends: 4 },
  { _id: 6, name: 'Jolana', num_of_friends: 4 },
  { _id: 8, name: 'Miriam', num_of_friends: 4 },
  { _id: 13, name: 'Vilém', num_of_friends: 4 }
]
```

Exercise 15

Express the following MongoDB query

- **Find the average age of users.**

Exercise 15 - Solution

Express the following MongoDB query

- **Find the average age of users.**

```
db.users.aggregate([
      { $project: { age: { $subtract: [ new Date(), "$birthdate" ] } } },
      { $group: { _id: null, avg_age: { $avg: "$age" } } },
      { $project: {_id: 0, avg_age: 1} }
])
```

```
[ { avg_age: 714050442934 } ]
```

Exercise 15 - Solution

Express the following MongoDB query

- **Find the average age of users.**

```
db.users.aggregate([
{ $project:
        { age: { $divide: [ { $subtract: [new Date(), "$birthdate"] },
        (365.25 * 24 * 60 * 60 * 1000)] } }
},
{ $group:
        { _id: null, avg_age: { $avg: "$age" } } },
        { $project: { _id: 0, avg_age: 1 }
}
])
```

```
[ { avg_age: 23.59297867008264 } ]
```

Exercise 16

Express the following MongoDB query

- **Find users who registered after 2015 and sort them in descending order by registration date.**

Exercise 16 - Solution

Express the following MongoDB query

- **Find users who registered after 2015 and sort them in descending order by registration date.**

```
db.users.aggregate([
        { $match: { "registration_date" : {
                            $gt : ISODate("2015-01-01T00:00:00.000Z") } } },
        { $sort: { registration_date: -1 } }
])
```

Exercise 17

Express the following MongoDB query

- **Find the number of users who registered in each year.**

Exercise 17 - Solution

Express the following MongoDB query

- **Find the number of users who registered in each year.**

```
db.users.aggregate([
        { $project: { year: { $year: "$registration_date" } } },
        { $group: { _id: "$year", count: { $sum: 1 } } }
])
```

```
[
  { _id: 2008, count: 1 },
  { _id: 2012, count: 3 },
  { _id: 2021, count: 1 },
  { _id: 2018, count: 1 },
  { _id: 2020, count: 1 },
  { _id: 2017, count: 1 },
  { _id: 2014, count: 2 },
  { _id: 2009, count: 1 },
  { _id: 2010, count: 3 },
  { _id: 2015, count: 3 },
  { _id: 2011, count: 1 },
  { _id: 2016, count: 2 }
]
```

Exercise 18

Express the following MongoDB query

- **Find users registered after 2015, group them by city of residence, and create a list of names for each city.**

Exercise 18 - Solution

Express the following MongoDB query

- **Find users registered after 2015, group them by city of residence, and create a list of names for each city.**

```
db.users.aggregate([
    { $match: { "registration_date": { $gt: ISODate("2015-01-01T00:00:00.000Z")}}},
    { $group: { _id: "$home.city",
                users: { $push: { name: "$name", surname: "$surname" } } } },
    { $replaceRoot: { newRoot: { city: "$_id", users: "$users" } } }
])
```

```
[
  { city: null,  users: [ { name: 'Dalimil', surname: 'Stejskal' }, { name: 'Štěpán', surname: 'Martínek' },
                          { name: 'Vilém', surname: 'Hanák' }, { name: 'Vlastislav', surname: 'Hladík' }  ] },
  { city: 'Brno', users: [ { name: 'Miloš', surname: 'Beran' }, { name: 'Bohdan', surname: 'Moravec' }  ] },
  { city: 'Prague', users: [ { name: 'Roland', surname: 'Blažek' }, { name: 'Miloslav', surname: 'Blažek' } ] },
  { city: 'Plzen', users: [ { name: 'Roland' } ] }
]
```

Exercise 19

Express the following MongoDB query

- **Who has the most friends?**
- **If several users have an equal number of friends, find them all.**

Exercise 19 - Solution

Express the following MongoDB query

- **Who has the most friends?**

```
db.users.aggregate([
        { $addFields : { num_friends : { $size : "$friends"}}},
        { $sort : { num_friends : -1}},
        { $project : { name : 1, num_friends : 1 }},
        { $limit : 1 }
])
```

```
[ { _id: 17, name: 'Miloš', num_friends: 6 } ]
```

Exercise 19 - Solution

- **If several users have an equal number of friends, find them all.**

```
// Step 1: Find the maximum number of friends
var maxFriends = db.users.aggregate([
      { $addFields : { num_friends : { $size : "$friends" }}},
      { $group: { _id: null, maxFriends: { $max: "$num_friends" }}}
]).toArray()[0].maxFriends;

// Step 2: Use the maxFriends value to find all users with that number of friends
db.users.aggregate([
      { $addFields : { num_friends : { $size : "$friends" }}},
      { $match : { num_friends : maxFriends }},
      { $project : { name : 1, num_friends : 1 }}
]);
```

```
[
  { _id: 17, name: 'Miloš', num_friends: 6 },
  { _id: 20, name: 'Vlastislav', num_friends: 6 }
]
```

Exercise 19 - Solution

- **If several users have an equal number of friends, find them all.**

PREFERABLE WAY!

```
db.users.aggregate([
        { $addFields: { num_friends: { $size: "$friends" }}},
        { $group: { _id: "$num_friends", users: { $push: "$$ROOT" }}},
        { $sort: { _id: -1 }},
        { $limit: 1 },
        { $unwind: "$users" },
        { $replaceRoot: { newRoot: "$users" }},
        { $project: { name: 1, num_friends: 1 }}
])
```

```
[
  { _id: 17, name: 'Miloš', num_friends: 6 },
  { _id: 20, name: 'Vlastislav', num_friends: 6 }
]
```

Join Collections

The $lookup stage lets you specify which collection you want to join with the current collection, and which fields that should match.

There are four required fields:

- from: The collection to use for lookup in the same database
- localField: The field in the primary collection that can be used as a unique identifier in the from collection.
- foreignField: The field in the from collection that can be used as a unique identifier in the primary collection.
- as: The name of the new field that will contain the matching documents from the from collection.

Exercise 20

Express the following MongoDB query

- **Find all check-ins for each user.**

Exercise 20 - Solution

Express the following MongoDB query

- **Find all check-ins for each user.**

```
db.users.aggregate([
    {$lookup: {
        from: "checkins",
        localField: "_id",
        foreignField: "user",
        as: "checkins"
    } }
])
```

```
{
  _id: 2,
  registration_date: ISODate("2010-06-25T18:55:16.981Z"),
  interests: [ 'diving', 'party' ],
  name: 'Vilém',
  surname: 'Linhart',
  birthdate: ISODate("2003-04-24T23:53:07.663Z"),
  friends: [ 3, 6, 16, 17, 20 ],
  checkins: [
    {
      _id: 21,
      timestamp: ISODate("2017-11-13T01:23:28.476Z"),
      location: {
        coordinates: [ 17.3051540639007, 45.7246604566228 ],
        type: 'Point'
      },
      user: 2,
      name: 'Prague'
    },
    {
      _id: 26,
      timestamp: ISODate("2019-01-17T01:23:28.476Z"),
      location: {
        coordinates: [ 27.3051540639007, 12.7246604566228 ],
        type: 'Point'
      },
      user: 2,
      name: 'Liberec'
    }
  ]
},
```

Exercise 21

Express the following MongoDB query

- **Who has the most check-ins?**

  Show top-3.

Exercise 21 - Solution

```
db.users.aggregate([{
        $lookup: {
                from: "checkins",
                localField: "_id",
                foreignField: "user",
                as: "checkins"
        }
},
        { $addFields: { num_checkins: { $size: "$checkins"} } },
        { $sort: { num_checkins: -1 } },
        { $limit: 3 }
])
```

Exercise 22

Express the following MongoDB query

- **Find all users interested in football and display the names of their friends.**

Exercise 22 - Solution

```
db.users.aggregate([
    { $match: { interests: "football" } },
    { $lookup: { from: "users", localField: "friends", foreignField: "_id", as:
    "friends_info" } },
    { $project: { name: 1, surname: 1, friends: "$friends_info.name"}}
])
```

```
{
  _id: 1,
  name: 'Roland',
  surname: 'Blažek',
  friends: [ 'Štěpán', 'Vlastislav', 'Miriam', 'Miloš' ]
},
{
  _id: 4,
  name: 'Vladimír',
  surname: 'Poláček',
  friends: [ 'Miloslav', 'Xenie', 'Jolana', 'Bohdan', 'Jolana' ]
},
{
  _id: 5,
  name: 'Štěpán',
  surname: 'Martínek',
  friends: [ 'Miloslav', 'Vlastislav', 'Roland' ]
},
{ _id: 9, name: 'Roland', friends: [ 'Xenie', 'Bohdan' ] },
```