B4M36DS2, BE4M36DS2: **Database Systems 2**

https://cw.fel.cvut.cz/b231/courses/b4m36ds2/

Practical Class 9

MongoDB

Yuliia Prokop

prokoyul@fel.cvut.cz

20. 11. 2023

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

Czech Technical University in Prague, Faculty of Electrical Engineering

Mongo Shell

Connect to our NoSQL server

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

Start mongo shell

mongosh "mongodb://localhost:42222" -u login -p password

Switch to your database

- **use** login

Insert sample data into your database

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

Sample Data

Insert the following actors into your emptied collection

```
{ _id: "trojan",
  name: "Ivan Trojan", year: 1964,
  movies: [ "samotari", "medvidek" ] }
```

```
{ _id: "machacek",
  name: "Jiri Machacek", year: 1966,
  movies: [ "medvidek", "vratnelahve", "samotari" ] }
```

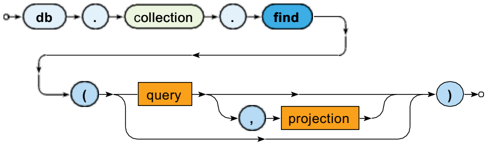
```
{ _id: "schneiderova",
  name: "Jitka Schneiderova", year: 1973,
  movies: [ "samotari" ] }
```

```
{ _id: "sverak",
  name: "Zdenek Sverak", year: 1936,
  movies: [ "vratnelahve" ] }
```

```
{ _id: "geislerova",
  name: "Anna Geislerova", year: 1976 }
```

Find Operation

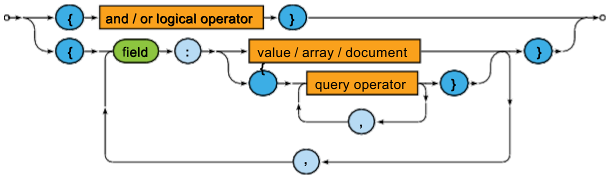
Selects documents from a given collection



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

Selection

Query parameter describes the documents we are interested in

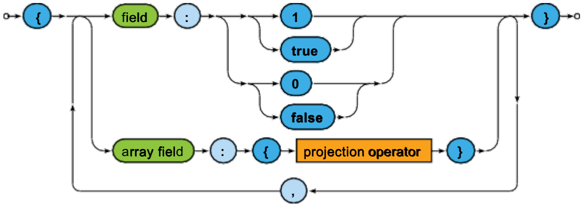


Selection operators

- $eq, $neq, $lt, $lte, $gte, $gt, $in, $nin
- $and, $or, $not
- $exists, $regex, $text
- …

Projection

Projection allows us to determine fields returned in the result



Projection operators

- $elemMatch, $slice,…

Querying

Execute and explain the meaning of **the following queries**

```
db.actors.find()
db.actors.find({ })
db.actors.find({ _id: "trojan" })
db.actors.find({ name: "Ivan Trojan", year: 1964 })
db.actors.find({ year: { $gte: 1960, $lte: 1980 } })
db.actors.find({ movies: { $exists: true } })
db.actors.find({ movies: "medvidek" })
db.actors.find({ movies: { $in: [ "medvidek", "pelisky" ] } })
db.actors.find({ movies: { $all: [ "medvidek", "pelisky" ] } })
```

Querying

Execute and explain the meaning of **the following queries**

```
db.actors.find({ $or: [ { year: 1964 }, { rating: { $gte: 3 } } ] })
```

```
db.actors.find({ rating: { $not: { $gte: 3 } } })
```

```
db.actors.find({ }, { name: 1, year: 1 })
```

```
db.actors.find({ }, { movies: 0, _id: 0 })
```

```
db.actors.find({ }, { name: 1, movies: { $slice: 2 }, _id: 0 })
```

```
db.actors.find().sort({ year: 1, name: -1 })
```

```
db.actors.find().sort({ name: 1 }).skip(1).limit(2)
```

```
db.actors.find().sort({ name: 1 }).limit(2).skip(1)
```

Index Structures

Motivation

- Full **collection scan** must be conducted when searching for documents **unless an appropriate index exists**

Primary index

- Unique index on values of the **_id field**
- Created automatically

Secondary indexes

- Created manually for values of a given key field / fields
- Always within just a single collection

Index Structures

Execute the following query and study its **execution plan**

```
db.actors.find({ movies: "medvidek" })
```

```
db.actors.find({ movies: "medvidek" }).explain()
```

Create a multikey index for movies of actors

```
db.actors.createIndex({ movies: 1 })
```

Examine the execution plan once again

MapReduce: Example

Count the number of movies filmed in each year, starting in *2005*

```
db.movies.mapReduce(
  function() {
    emit(this.year, 1);
  },
  function(key, values) {
    return Array.sum(values);
  },
  {
    query: { year: { $gte: 2005 } },
    sort: { year: 1 },
    out: "statistics"
  }
)
```

MapReduce

Implement and execute the following MapReduce jobs

- **Find a list of actors (their names sorted alphabetically) for each year (they were born)**
  - Only consider actors born in year 2000 or before
  - `values.sort()`
  - Use `out: { inline: 1 }` option
- **Calculate the overall number of actors for each movie**
  - `this.movies.forEach(function(m) { ..})`
  - `Array.sum(values)`
  - Use `out: { inline: 1 }` option once again

Exercise 1

Express the following MongoDB query

- **Find actors born in *1966* with first name *Jiri***

Exercise 2

Express the following MongoDB query

- **Find movies directed by *Jan Hrebejk***
- Note that the order of fields for first and last names can be arbitrary

Exercise 3

Express the following MongoDB query

- **Find actors with first name *Jiri* who played in *Medvidek* movie**
- Return names of these actors only

Exercise 4

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 5

Express the following MongoDB query

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

Exercise 6

Express the following MongoDB query

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

Exercise 7

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 8

Express the following MongoDB query

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

Exercise 9

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