

NIE-PDB: **Advanced Database Systems**

<http://www.ksi.mff.cuni.cz/~svoboda/courses/231-NIE-PDB/>

Practical Class 4

MongoDB

Martin Svoboda

martin.svoboda@fit.cvut.cz

28. 11. 2023

Charles University, Faculty of Mathematics and Physics

Czech Technical University in Prague, Faculty of Information Technology

First Steps

Connect to our NoSQL server

- SSH / PuTTY and SFTP / WinSCP
- **nosql.kti.in.fit.cvut.cz:42222**

Start mongo shell

- `mongosh -u login -p password`
 - Password option can be omitted, you will then be prompted

Switch to your database

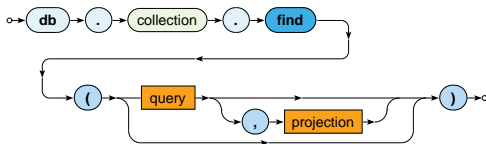
- use `login`

Insert sample data into your database

- See `/home/PDB/mongodb/data.js`

Find Operation

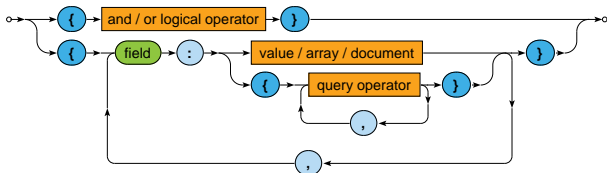
Selects documents from a given collection



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

Selection

Query parameter describes the documents we are interested in

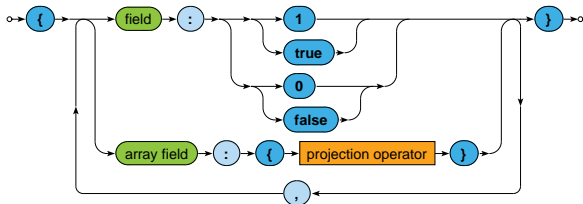


Selection operators

- `$eq`, `$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,...

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 Hřebejk***
- 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 starred 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

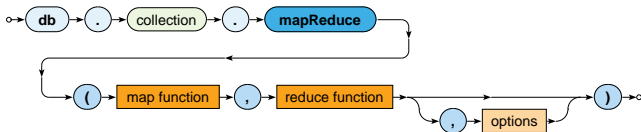
Exercise 9

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

MapReduce

Executes a **MapReduce** job on a selected collection



- Parameters

- **Map**: JavaScript implementation of the Map function
- **Reduce**: JavaScript implementation of the Reduce function
- **Options**

MapReduce

Map function

- Current document is accessible via `this`
- `emit(key, value)` is used for emissions

Reduce function

- Intermediate key and values are provided as arguments
- Reduced value is published via `return`

Options

- `query`: only matching documents are considered
- `sort`: they are processed in a specific order
- `limit`: at most a given number of them is processed
- `out`: output is stored into a given collection

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**
 - `if (this.movies instanceof Array) {...}`
 - `for (var i = 0; i < this.movies.length; i++)`
`{...this.movies[i] ...}`
 - `Array.sum(values)`
 - Use `out: { inline: 1 }` option once again

References

Documentation

- <https://docs.mongodb.com/v6.0/>