

## 1. Wstęp

Pobrałem jeopardy questions z moodle, dokonując kilku modyfikacji, które ułatwiły mi zadanie:

a) plik JEOPARDY\_QUESTIONS1.json przepuściłem przez krótki skrypt w języku Python:

```
file_name = "JEOPARDY_QUESTIONS1.json"
with open(file_name, "r+") as f:
    file_content = f.read()
    fixed_content = file_content[1:-1].replace("}, {"", "}}\n{")

with open("JEOPARDY_QUESTIONS.json", "w+") as f:
    f.write(fixed_content)
```

Skrypt ten zamienił wejście JEOPARDY\_QUESTIONS1.json na wyjście

JEOPARDY\_QUESTIONS.json wg mapowania:

f: [x1,x2,...,xn] => x1\n x2\n... xn \n

gdzie xi oznacza dokument.

MongoDB nie pozwalało bowiem na załadowanie pliku o długości linii przekraczającej pewne maksimum. W naszym przypadku błąd powodowała pierwsza (i jedyna) linia pliku traktująca kolekcję jako listę dokumentów. Przydzielenie każdemu elementowi listy osobnej linijki pliku rozwiązało problem.

b) Zamieniłem typ dwóm polom, air\_date ze string na ISODate, oraz show\_number ze string na int:

```
db.question.find().forEach( function (x) {
    x.air_date = new ISODate(x.air_date); // convert field to string
    db.question.save(x);
});
db.question.find().forEach( function (x) {
    x.show_number = new NumberInt(x.show_number); // convert field to string
    db.question.save(x);
});
```

Dzięki temu umożliwiłem operacje sortowania po liczbach całkowitych i datach, nie będąc zdany na to, że są zapisane jako ciąg znaków (sortowanie stringów odbywa się bowiem za pomocą porządku leksykograficznego, więc np. 50 występuje zaraz po 3000, co istotnie psuje rozwiązanie).

W zadaniu jestem zainteresowany otrzymaniem odpowiedzi na trzy pytania /problemy:

a) Jaki jest dokument o największym show\_number taki, że jego show\_number jest większy od 6290 - a jeśli istnieje kilka dokumentów o największym show\_number, chcę wiedzieć który z nich posiada najmlodsza (najpóźniejszą) air\_date [proste zapytanie]

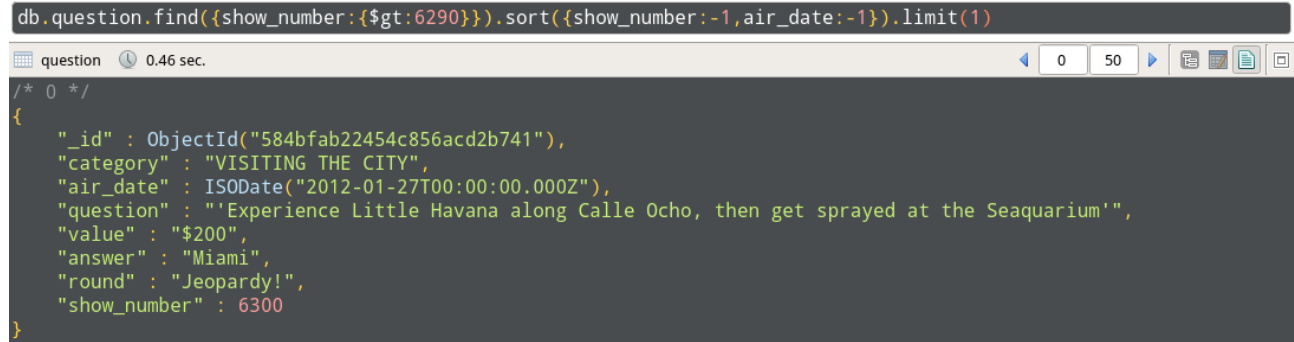
b) Ile jest dokumentów o największym show\_number, które spełniają: round = "Double Jeopardy!", oraz show\_number >= 6290 [zapytanie uwzględniające agregację pipeline'ową]

c) Po pierwsze: ile jest pytań z kategorii "EVERYBODY TALKS ABOUT IT...", po drugie: ile jest pytań których show\_number jest większe od 5000, po trzecie: ile jest pytań, których air\_date jest późniejsza niż 31 grudzień 2004 r. Odpowiedź ma być zawarta w jednym pliku. [mechanizm mapReduce]

## 2. Wykonanie zapytań przy użyciu narzędzia Robomongo:

### a) proste zapytanie

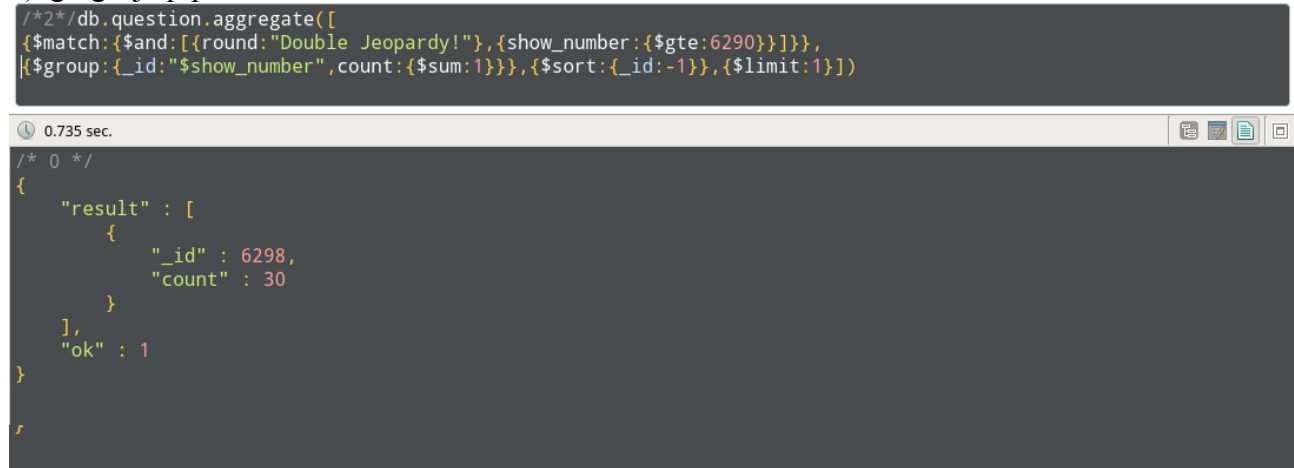
```
db.question.find({show_number:{$gt:6290}}).sort({show_number:-1,air_date:-1}).limit(1)
```



The screenshot shows the Robomongo interface with a query window. The query is `db.question.find({show_number:{$gt:6290}}).sort({show_number:-1,air_date:-1}).limit(1)`. The execution time is 0.46 sec. The result is a single document with the following fields: `_id` (ObjectId), `category` ("VISITING THE CITY"), `air_date` (ISODate), `question` ("Experience Little Havana along Calle Ocho, then get sprayed at the Seaquarium"), `value` ("200"), `answer` ("Miami"), `round` ("Jeopardy!"), and `show_number` (6300).

### b) agregacja pipeline'owa

```
/*2*/db.question.aggregate([
{$match:{$and:[{round:"Double Jeopardy!"},{show_number:{$gte:6290}}]}},
{$group: {_id:"$show_number",count:{$sum:1}}},{sort: {_id:-1}},{limit:1}])
```

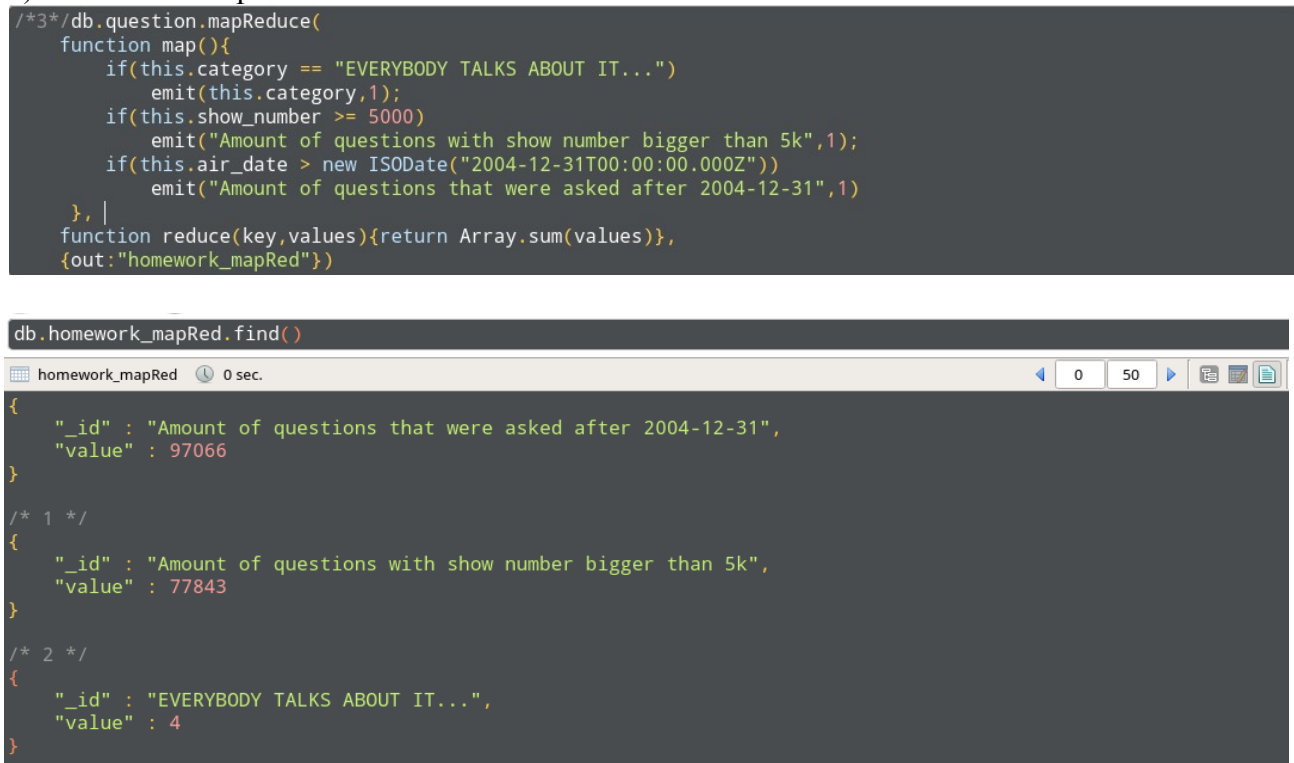


The screenshot shows the Robomongo interface with an aggregation pipeline query. The query is `/*2*/db.question.aggregate([{$match:{$and:[{round:"Double Jeopardy!"},{show_number:{$gte:6290}}]}},{$group: {_id:"$show_number",count:{$sum:1}}},{sort: {_id:-1}},{limit:1}])`. The execution time is 0.735 sec. The result is an array with one document: `{_id: 6298, count: 30}`. The `ok` field is 1.

### c) mechanizm mapReduce

```
/*3*/db.question.mapReduce(
function map(){
if(this.category == "EVERYBODY TALKS ABOUT IT...")
emit(this.category,1);
if(this.show_number >= 5000)
emit("Amount of questions with show number bigger than 5k",1);
if(this.air_date > new ISODate("2004-12-31T00:00:00.000Z"))
emit("Amount of questions that were asked after 2004-12-31",1)
}, |
function reduce(key,values){return Array.sum(values)},
{out:"homework_mapRed"})
```

```
db.homework_mapRed.find()
```



The screenshot shows the Robomongo interface with a mapReduce query. The map function emits three categories: "EVERYBODY TALKS ABOUT IT...", "Amount of questions with show number bigger than 5k", and "Amount of questions that were asked after 2004-12-31". The reduce function sums the values for each key. The output is stored in the `homework_mapRed` collection. The result of `db.homework_mapRed.find()` is an array of three documents: `{_id: "Amount of questions that were asked after 2004-12-31", value: 97066}`, `{_id: "Amount of questions with show number bigger than 5k", value: 77843}`, and `{_id: "EVERYBODY TALKS ABOUT IT...", value: 4}`.

### 3. Wykonanie zapytań w oparciu o interfejs programistyczny (Java)

#### a) proste zapytanie:

```
public void queryOne() {
    DBCollection question = db.getCollection("question");
    BasicDBObject sortFields = new BasicDBObject("show_number", -1);
    sortFields.put("air_date", -1);
    DBObject res = question.find(new BasicDBObject("show_number", new
        BasicDBObject("$gt", 6290))).sort(sortFields).limit(1).one();
    System.out.println(res);
}
```

Rezultat (nie screen, ponieważ linijka tekstu była za długa):

```
{
  "_id" : { "$oid" : "58405487e43ba670a05521e6" } ,
  "category" : "VISITING THE CITY" ,
  "air_date" : { "$date" : "2012-01-27T00:00:00.000Z" } ,
  "question" : "'Experience Little Havana along Calle Ocho, then get sprayed at
the Seaquarium'" ,
  "value" : "$200" ,
  "answer" : "Miami" , "round" : "Jeopardy!" ,
  "show_number" : 6300
}
```

#### b) agregacja pipeline'owa:

```
public void queryTwo() {
    DBCollection question = db.getCollection("question");
    List<DBObject> and_input = new ArrayList<DBObject>();
    and_input.add(new BasicDBObject("round", "Double Jeopardy!"));
    and_input.add(new BasicDBObject("show_number", new
BasicDBObject("$gte", 5000)));
    DBObject match = new BasicDBObject("$match", new
BasicDBObject("$and", and_input));
    DBObject groupFields = new BasicDBObject("_id", "$show_number");
    groupFields.put("count", new BasicDBObject("$sum", 1));
    DBObject group = new BasicDBObject("$group", groupFields);
    DBObject sort = new BasicDBObject("$sort", new BasicDBObject("_id", -
1));

    DBObject limit = new BasicDBObject("$limit", 1);
    List<DBObject> pipeline = Arrays.asList(match, group, sort, limit);
    AggregationOutput output = question.aggregate(pipeline);
    for (DBObject result : output.results()) {
        System.out.println(result);
    }
}
```

Rezultat:



### c) mechanizm mapReduce

```
public void queryThree() {
    DBCollection question = db.getCollection("question");
    String map = "    function map() {                if(this.category == \"EVERYBODY
TALKS ABOUT IT...\")                emit(this.category,1);
if(this.show_number >= 5000)                emit(\"Amount of questions with show
number bigger than 5k\",1);                if(this.air_date > new ISODate(\"2004-12-
31T00:00:00.000Z\"))                emit(\"Amount of questions that were asked after
2004-12-31\",1)                }";
    String reduce = "function reduce(key,values){return Array.sum(values)}";
    MapReduceCommand cmd = new
MapReduceCommand(question,map,reduce,null,MapReduceCommand.OutputType.INLINE,null
1);

    MapReduceOutput out = question.mapReduce(cmd);
    for(DBObject o : out.results()){
        System.out.println(o.toString());
    }
}
```

Rezultat:

