

LAB2: MONGODB

EXERCISE 1

After the login in cluster and the selection of the movieClients database using the command *use movieClients*, the command *show collections* showed two collections: *client* and *movie*.

```

user9@it:~
mongos> show collections
client
movie
mongos> |

```

EXERCISE 2

1. One shard key (*id=1*) and two chunks with two shards *rs0*: 1 and *rs1*: 1.

```

user9@it:~
{ "_id" : "movieclients", "primary" : "rs1", "partitioned" : true, "version" : { "uuid" : UUID("d7f82c1b-5a79-4f51-96b4-cb41bd89896b"), "lastMod" : 1 } }
  movieclients.movie
    shard key: { "_id" : 1 }
    unique: true
    balancing: false
    chunks:
      rs0      1
      rs1      1
    { "_id" : { "$minKey" : 1 } } --> { "_id" : ObjectId("6192373e7ac90092e1b55e79") } on : rs0 Tim
estamp(2, 0)
    { "_id" : ObjectId("6192373e7ac90092e1b55e79") } --> { "_id" : { "$maxKey" : 1 } } on : rs1 Tim

```

2. The *movie* collection has two shards *rs0* e *rs1*, two chunks and 42926 documents for each chunk.

```

user9@it:~
mongos> db.movie.getShardDistribution()

Shard rs0 at rs0/it.unige.dibris.lsccluster.node3:27017,it.unige.dibris.lsccluster.node4:27017,it.unige.dibris.lsccluster.node5:27017
data : 17.87MiB docs : 42926 chunks : 1
estimated data per chunk : 17.87MiB
estimated docs per chunk : 42926

Shard rs1 at rs1/it.unige.dibris.lsccluster.node6:27017,it.unige.dibris.lsccluster.node7:27017
data : 17.7MiB docs : 42926 chunks : 1
estimated data per chunk : 17.7MiB
estimated docs per chunk : 42926

Totals
data : 35.58MiB docs : 85852 chunks : 2
Shard rs0 contains 50.23% data, 50% docs in cluster, avg obj size on shard : 436B
Shard rs1 contains 49.76% data, 50% docs in cluster, avg obj size on shard : 432B

```

3. The *client* collection is not sharded, but it's replicated: the primary set is *rs1*.

```

user9@it:~
mongos> db.client.stats()
{
  "sharded" : false,
  "primary" : "rs1",
  "capped" : false,
  "wiredTiger" : {
    "metadata" : {
      "formatVersion" : 1
    }
  },

```

4. The shard containing the document is *rs0*.

```
user9@it:~  
mongos> db.movie.explain().find({"_id": "61911fa092b76d1c8ef99383"})  
{  
  "queryPlanner" : {  
    "mongosPlannerVersion" : 1,  
    "winningPlan" : {  
      "stage" : "SINGLE_SHARD",  
      "shards" : [  
        {  
          "shardName" : "rs0",  
          "connectionString" : "rs0/it.unige.dibris.lsccluster.node3:27017,it.unige.dibris.lsccluster.node4:27017,it.unige.dibris.lsccluster.node5:27017",  
          "serverInfo" : {  
            "host" : "it.unige.dibris.lsccluster.node3",  
            "port" : 27017,  
            "version" : "4.4.1",  
            "gitVersion" : "ad91a93a5a31e175f5cbf8c69561e788bbc55ce1"  
          },  
          "plannerVersion" : 1,  
          "namespace" : "movieclients.movie",  
          "indexFilterSet" : false,  
          "parsedQuery" : {  
            "_id" : {  
              "$eq" : "61911fa092b76d1c8ef99383"  
            }  
          },  
          "queryHash" : "A300CFDE",  
          "planCacheKey" : "A2B33459",  
          "winningPlan" : {  
            "stage" : "SHARDING_FILTER",  
            "inputStage" : {
```

EXERCISE 3

- Q1: `db.movie.find({"director":"Robert Altman"},{title:1})`
- Q2: `db.movie.find({$or: [{"director":"Robert Altman"}, {"director": "Ken Loach"}]})`
- Q3: `db.movie.find({"title":"Atlantis"},{actors:1})`. The actors field is an aggregate.
- Q4: `db.movie.find({"director":"Robert Altman", "year": {$gt:1990}})`
- Q5: `db.movie.find({"director":"Robert Altman", "title": {$in: ["Gang", "Aria"]}, {year:1})`
- Q6: `db.movie.find({"year": {$gte: 1990}, "year": {$lte: 1999}}, {director:1})`
- Q7: `db.client.find({"birthdate": {$gt: "2000-01-01"}}, {name:1, surname:1, birthdate:1})`
- Q8: `db.movie.find({"director":"Ken Loach"},{title:1, year:1}).sort({"title":1}).limit(4)`
- Q9: `db.movie.aggregate([{$group: {_id:"$director", count: {$sum:1}}})`
- Q10: `db.movie.aggregate([{$group: {_id:"$director", avgEvaluation: {$avg: "$eval"}}})`
- Q11a: `db.movie.find({"title": "Atlantis", "director": "August Blom"}, {recommeded_by:1})` or
- Q11b: `db.client.distinct("client_id", {"recommends.title": "Atlantis", "recommends.director": "August Blom"})`
- Q12: `db.movie.aggregate([{$match: {$and: [{title:"Atlantis"}, {director:"August Blom"}]}}, {$lookup: {from:"client", localField:"recommeded_by", foreignField:"client_id", as:"client_information"}}])`

EXERCISE 4

- Q1: `db.movie.find({"title": "Dracula"})`
- Q2: `db.movie.find({"director": "Ken Loach", genre: "Comedy"}, {title: 1})`
- Q3: `db.movie.find({"director":"Ken Loach"},{title:1, year:1}).sort({"year":1}).limit(4)`
- Q4: `db.client.aggregate([{$match: {birthdate: {$gt: "2000-01-01"}}, {$group: {_id: "client_id", count: {$sum:1}}})`
- Q5: `db.client.find({"recommends.title": "Atlantis"}, {name:1, surname:1})`
- Q6: `db.movie.aggregate([{$match: {$and: [{title:"Atlantis"}, {director:"August Blom"}]}}, {$lookup: {from:"client", localField:"recommeded_by", foreignField:"client_id", as:"client_information"}, {$match: {birthdate: {$gt: "2000-01-01"}}})`

(Q6 of Exercise 4 seems to not work properly, the one written in this report is our actual best attempt)