Modern Database Concepts
Prof. Dr. Florian Heinz
florian.heinz@oth-regensburg.de
Exercise Sheet 9

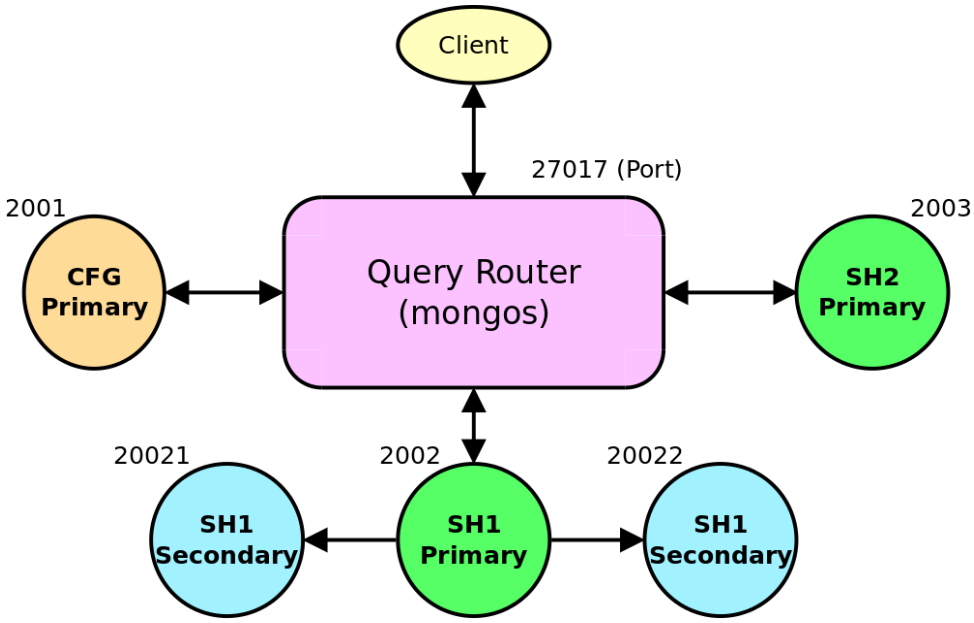
OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Exercise 1: Install and start a local MongoDB cluster

On the CIP pool hosts, boot Ubuntu, start a shell (e.g. Q-Terminal)

Download the MongoDB zip archive from Grips and unpack it

Set up a cluster with the following topology:



- Create data directories `cfg, sh1a, sh1b, sh1c, sh2` for each MongoDB node
- Start the 5 **mongod** processes, each in a separate terminal tab. You have to specify the type (*--configsvr* or *--shardsvr*), the *--dbpath*, the name of the replication set *--replSet* (cfg, sh1 or sh2) and the TCP port number *--port* as command line arguments
- Initialize the three replica sets by connecting to the three primaries with **mongosh --port <portnr>**, initiate them, and add the secondaries to the replica set "sh1" (see documentation or slides)
- Start the query router **mongos** and specify the configuration replica set with *--configdb* (see hint below)
- Connect to the query router with **mongosh** and add the replica sets **sh1** and **sh2** as shards (command: sh.addShard(<replicaSet>))

Hint: A replica set is always specified by its name and a comma-separated list with hostname:port of the replica set members, e.g. a replication set with name **rs1**:
`rs1/localhost:1234,localhost:1235,...`

Modern Database Concepts
Prof. Dr. Florian Heinz
florian.heinz@oth-regensburg.de
Exercise Sheet 9

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Exercise 2: Cluster replication

Create a database "oth"

Create a collection "students" and insert 5 items of the following structure (use two different values for semester, four different course names (each student should visit 2-3 courses) and unique values for **matrnr** and **cpEarned**):

```
{
    "matrnr" : 12345,
    "personal": {
        "name": "Anna",
        "age": 27
    },
    "study": {
        "semester": 3,
        "cpEarned": 40
    },
    "courses": ["databases", "programming"]
}
```

Create an index on the field **matrnr** that is unique

This collection is not sharded. Find out where it resides (*sh.status()* can help)

If this collection resides on **sh1**, move it to **sh2** (Look up the command *movePrimary*")

If this collection resides on **sh2**, move it to **sh1** (yes, do that also if you just moved it to **sh2** in the previous step)

Can you have one collection of database "oth" on **sh1** and another one on **sh2**? How (or why not)?

In another terminal, connect to one of the secondaries of replication set **sh1** and try to read from the collection **students**. If it doesn't work, use *db.getMongo().setReadPref("secondary")* to enable that.

Check the status of the replication set with *rs.status()*. Who is primary, who are the secondaries? Now, stop the primary by pressing `CTRL + C` in the corresponding terminal window and check the replication set status again. Who is the new primary now? Finally, start the old primary again (in the terminal, where you just stopped it). What happens now to the cluster state, is it elected as primary again?

Modern Database Concepts
Prof. Dr. Florian Heinz
florian.heinz@oth-regensburg.de
Exercise Sheet 9

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Exercise 3: MongoDB Queries

Query the collection **students** for a specific name. You are only interested in the semester of this student and the courses (s)he visits

Print out all entries of **students** and sort it by **semester** ascending and **cpEarned** descending.

One semester has passed. Increment the **semester** property for all students (with a single command) (use updateMany and the $inc operation)

Use the aggregation pipeline to match all people older that a certain **age** (choose a value that matches only some students) and sort them by their **matrnr**

Extend the pipeline to emit a document with a single property **studentname** containing the name of the student (and the _id field, that is always added). Use only the **$project** stage for this.

Now lets find out the total and average credit points per semester. You can use the **$group** phase of the aggregation pipeline for this. Write the results into a collection **creditpoints** in the **$out** phase.

In the next step, unwind the **courses** array. After that, group by each course and create an array with the matrnr of all students, that visit the course (use the **$push** aggregator).

Create another collection **courses** with 4 items and the following structure (use the course names that you also used in the collection **students**):

```
{
    "course": "databases",
    "lecturer": "Prof. Dr. Ache",
    "room": "K221"
}
```

Now, again, unwind the **courses** field in the **students** collection and join them with the **courses** collection. You can do that in the $lookup stage of the pipeline, where you have to specify the parameters **from**, **as**, **localField** and **foreignfield**

Modern Database Concepts
Prof. Dr. Florian Heinz
florian.heinz@oth-regensburg.de
Exercise Sheet 9

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Exercise 4: Sharding

As a preparatory step, set the chunk size to 1 MB:

```
use config
db.settings.updateOne({_id:"chunksize"},{$set:{_id:"chunksize",value:1}},{upsert:true})
```

After that, create a database **webapp** and enable sharding on it (function *sh.enableSharding()*)

Now, create a collection **users** and activate sharding for it (function *sh.shardCollection()*). Use the field **id** as the shard key

Use the **mongoimport** tool to import the file **users.json** into the collection **users**, which you can download from Grips

Check, if the collection was sharded (*sh.status()*, *db.users.getShardDistribution()*)

Do a simple lookup on the collection by the field **id**. View the execution plan with **.explain()**. After that, do a lookup by **name** and again view the execution plan. What is the difference? And why?

View the execution plan that is used, when you try to find all documents with id greater than 8000. What would you expect with hash partitioning here?

Group the documents by firstname and language and calculate the sum of logins for each group. Also view the execution plan for this query and try to understand, what happens.

Use the aggregation pipeline to add a field **numcolors** to the gui subdocument, that contains the number of distinct colors in the **colors** array. Do not only display it, but alter the collection permanently. You can use the **$merge** phase for this.