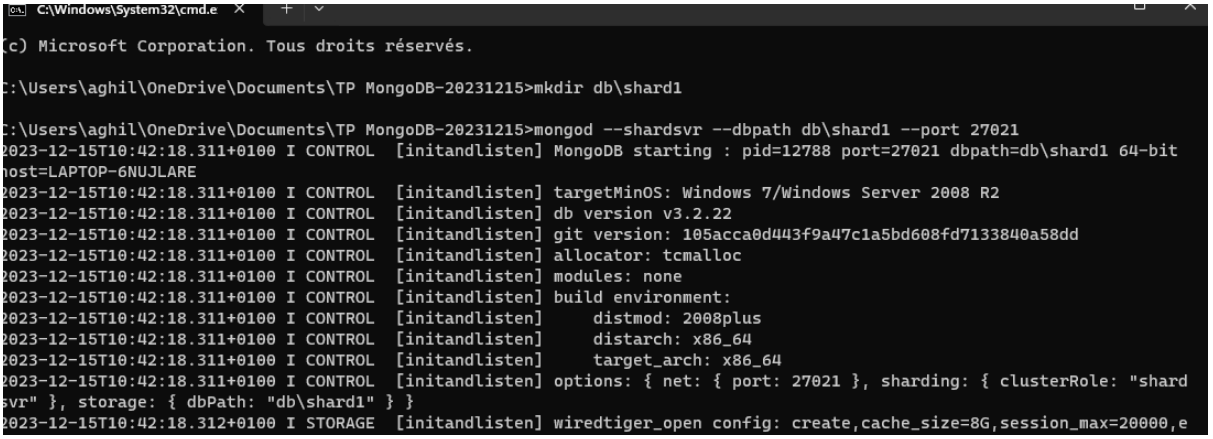
TP Mongo DB

1.1: Start a MongoDB instance:



1.2 : Using a new shell, connect to the MongoDB instance:

1.3 : Create the database mydb and the database collection cities:

1.4 : Verify the existence of the database (mydb) and the database collection (cities):

```
> show dbs
local  0.000GB
mydb   0.000GB
> show collection
2023-12-15T10:49:27.423+0100 E QUERY    [thread1] Er
shellHelper.show@src/mongo/shell/utils.js:885:11
shellHelper@src/mongo/shell/utils.js:671:15
@(shellhelp2):1:1

> show collections
cities
>
```

2.1 : Populating and querying a database:

- import the content of the file cities.txt into mydb.cities collection.

```
C:\Users\aghil\OneDrive\Documents\TP MongoDB-20231215>mongoimport --host localhost:27021 --db mydb --collection cities --file cities.txt
2023-12-15T10:55:30.216+0100    connected to: localhost:27021
2023-12-15T10:55:30.414+0100    imported 29353 documents
```

2.2: verify the existence of data in the database by issuing some queries:

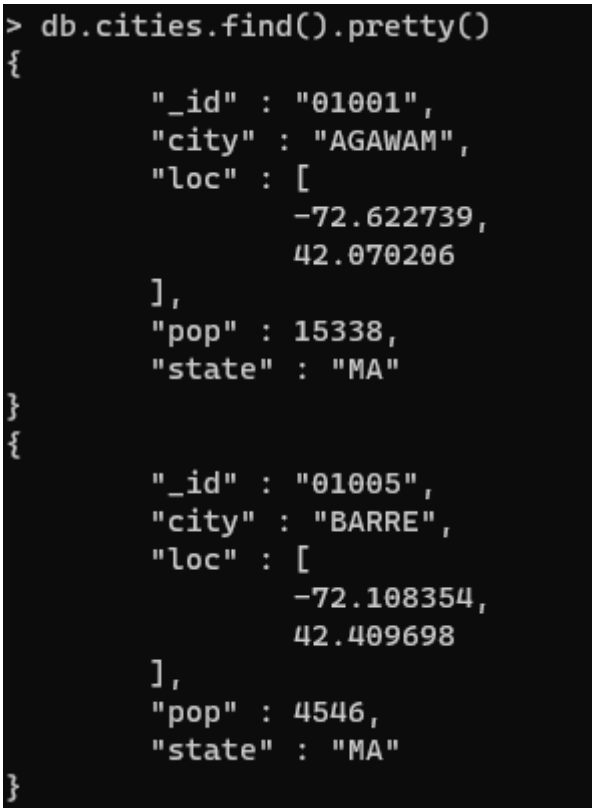
```
> db.cities.find()
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
{ "_id" : "01005", "city" : "BARRE", "loc" : [ -72.108354, 42.409698 ], "pop" : 4546, "state" : "MA" }
{ "_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240, "state" : "MA" }
{ "_id" : "01007", "city" : "BELCHERTOWN", "loc" : [ -72.410953, 42.275103 ], "pop" : 10579, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state" : "MA" }
{ "_id" : "01010", "city" : "BRIMFIELD", "loc" : [ -72.188455, 42.116543 ], "pop" : 3706, "state" : "MA" }
{ "_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396, "state" : "MA" }
{ "_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" : "MA" }
{ "_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177, "state" : "MA" }
{ "_id" : "01022", "city" : "WESTOVER AFB", "loc" : [ -72.558657, 42.196672 ], "pop" : 1764, "state" : "MA" }
{ "_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495, "state" : "MA" }
{ "_id" : "01026", "city" : "CUMMINGTON", "loc" : [ -72.905767, 42.435296 ], "pop" : 1484, "state" : "MA" }
{ "_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864, "state" : "MA" }
{ "_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" : 13367, "state" : "MA" }
{ "_id" : "01030", "city" : "FEEDING HILLS", "loc" : [ -72.675077, 42.07182 ], "pop" : 11985, "state" : "MA" }
{ "_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385, "state" : "MA" }
{ "_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" : "MA" }
{ "_id" : "01034", "city" : "TOLLAND", "loc" : [ -72.908793, 42.070234 ], "pop" : 1652, "state" : "MA" }
{ "_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" : "MA" }
{ "_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" : "MA" }
Type "it" for more
> db.cities.count()
29353
>
```

Q1. Describe in natural language the content of the database collection.

The cities collection in the database consists of documents, each representing a city in the United States. Key fields in each document include:

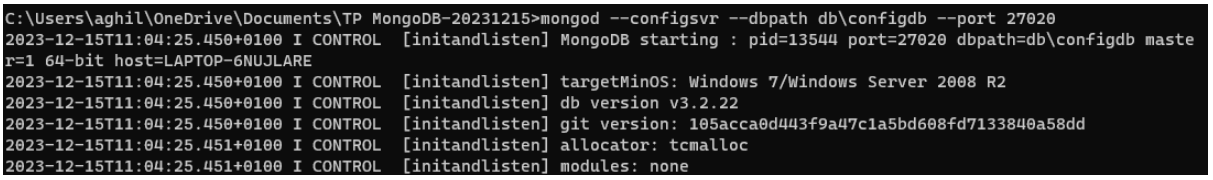
- **Document Identifier (_id):** A unique code for each city, such as "01001".
- **City Name (city):** The name of the city, e.g., "AGAWAM".

- **Location (loc):** Geographical coordinates (latitude and longitude) of the city, like [-72.622739, 42.070206] for AGAWAM.
- **Population (pop):** The population number, such as 15,338 for AGAWAM.
- **State (state):** The state abbreviation where the city is located, like "MA" for Massachusetts.



- How many cities are there in the database?
  There are **29353** cities in the database.

3.1 : Starting a config server instance:



3.2 : Starting a query router instance :

```
C:\Users\aghil\OneDrive\Documents\TP MongoDB-20231215>mongos --configdb localhost:27020 --port 27019
2023-12-15T11:06:09.137+0100 W SHARDING [main] Running a sharded cluster with fewer than 3 config servers should only be
 done for testing purposes and is not recommended for production.
2023-12-15T11:06:09.142+0100 I SHARDING [mongosMain] MongoS version 3.2.22 starting: pid=12364 port=27019 64-bit host=LA
PTOP-6NUJLARE (--help for usage)
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain] db version v3.2.22
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain] git version: 105acca0d443f9a47c1a5bd608fd7133840a58dd
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain] allocator: tcmalloc
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain] modules: none
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain] build environment:
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain]     distmod: 2008plus
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain]     distarch: x86_64
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain]     target_arch: x86_64
2023-12-15T11:06:09.142+0100 I CONTROL  [mongosMain] options: { net: { port: 27019 }, sharding: { configDB: "localhost:2
7020" } }
```

3.3 : Adding a shard instance to the cluster

connect to the query router :

```
C:\Users\aghil\OneDrive\Documents\TP MongoDB-20231215>mongo --host localhost:27019
MongoDB shell version: 3.2.22
connecting to: localhost:27019/test
mongos>
```

3.4 : Add to the cluster the mongo instance containing the mydb database:

```
mongos> use admin
switched to db admin
mongos> db.runCommand( { addShard: "localhost:27021", name: "shard1" } )
{ "shardAdded" : "shard1", "ok" : 1 }
mongos>
```

3.5 : Verify the state of the cluster:

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("657c25114cae16f7d3076c67")
  }
  shards:
        {  "_id" : "shard1",  "host" : "localhost:27021" }
  active mongoses:
        "3.2.22" : 1
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "mydb",  "primary" : "shard1",  "partitioned" : false }
```

Q2 Which is the important information reported by this command? Refer to the explanation of the previous lecture:

La commande sh.status() fournit des informations importantes sur l'état du cluster partitionné :

Shard: affiche les détails de chaque fragment du cluster, y compris leurs noms et la plage de données qu'ils contiennent.

Database and Collection : montre comment les bases de données et les collections sont distribuées entre les partitions.
Chunk Information:indique comment les données sont partitionnées en fragments et distribuées entre les fragments. Cela inclut des détails sur la taille et le nombre de morceaux.

Balancer : fournit des informations sur le processus d'équilibrage, qui garantit que les données sont réparties uniformément entre les partitions.

Server Status: affiche l'état et les détails des serveurs de configuration.

Comprendre le résultat de **sh.status()** est crucial pour gérer et surveiller un environnement MongoDB partitionné, car il fournit des informations sur la distribution et l'état des données au sein du cluster.

4.1 : Sharding a database collection :
4.1.1 : Sharding a collection using ranges :

```
mongos> use mydb
switched to db mydb
mongos> db.createCollection("cities1")
{ "ok" : 0, "errmsg" : "collection already exists", "code" : 48 }
mongos> show collections # Verify collection existence
cities
cities1
```

4.1.2 : using as shard key the attribute state:

```
mongos> sh.enableSharding("mydb")
{ "ok" : 1 }
mongos> sh.shardCollection("mydb.cities1", { "state": 1} )
{ "collectionsharded" : "mydb.cities1", "ok" : 1 }
```

4.1.3 : Verify the number of chunks:

Q3 How many chunks did you create? Which are their associated ranges? Include a screen copy of the results of the command in your answer to support your answer.

- J'ai créé un chunk pour la collection cities1 dans mon cluster MongoDB.

Q4 How many chunks are there now? Which are their associated ranges? Which changes can you identify in particular? Include a screen copy of the results of the command in your answer to support your answer.

- **Nombre de Chunks**

Il y a maintenant 3 chunks dans la collection cities1 de la base de données mydb.

- **Plages Associées aux Chunks**

Les plages de ces chunks sont les suivantes :

- Premier Chunk :

Plage : de { "state" : { "$minKey" : 1 } } à { "state" : "MA" }
Emplacement : shard1
Ce chunk contient les documents où la valeur du champ state est inférieure à "MA".

- Deuxième Chunk :

Plage : de { "state" : "MA" } à { "state" : "RI" }
Emplacement : shard1
Ce chunk détient les documents où la valeur du champ state est supérieure ou égale à "MA" et inférieure à "RI".

- Troisième Chunk :

Plage : de { "state" : "RI" } à { "state" : { "$maxKey" : 1 } }
Emplacement : shard1

Ce chunk inclut les documents où la valeur du champ state est supérieure ou égale à "RI".

- **Changements Identifiés:**

Les changements notables identifiés dans le statut du sharding par rapport à l'état initial sont:

- Augmentation du Nombre de Chunks : Initialement, il n'y avait qu'un seul chunk. L'augmentation à trois chunks indique que les données dans la collection cities1 ont été partitionnées en plusieurs chunks basés sur la clé de shard state.

- Fractionnement des Chunks Basé sur la Valeur de state : Les nouvelles plages de chunks indiquent que les chunks ont été divisés en fonction des valeurs du champ state. Cela est conforme au modèle de clé de shard { "state" : 1 }.

- Tous les Chunks sont sur shard1 : Malgré l'augmentation du nombre de chunks, ils sont tous situés sur shard1, car il s'agit d'une configuration à shard unique.

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("657c25114cae16f7d3076c67")
  }
  shards:
        {  "_id" : "shard1",  "host" : "localhost:27021" }
  active mongoses:
        "3.2.22" : 1
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "mydb",  "primary" : "shard1",  "partitioned" : true }
                mydb.cities1
                        shard key: { "state" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  3
                        { "state" : { "$minKey" : 1 } } -->> { "state" : "MA" } on : shard1 Timestamp(1, 1)
                        { "state" : "MA" } -->> { "state" : "RI" } on : shard1 Timestamp(1, 2)
                        { "state" : "RI" } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(1, 3)

mongos>
```

5.1 : Sharding a collection using hash-based partitioning
5.1.1 : create the collection cities2:

```
mongos> db.createCollection("cities2")
{ "ok" : 1 }
mongos> show collections # Verify collection existence
cities
cities1
cities2
```

5.1.2 : Enable sharding on the collection mydb.cities2 :

```
mongos> sh.shardCollection("mydb.cities2", { "state": "hashed"} )
{ "collectionsharded" : "mydb.cities2", "ok" : 1 }
mongos>
```

Q5 How many chunks did you create? What differences do you see with respect to the same task in the range sharding strategy? Include a screen copy of the results of the command in your answer to support your answer.

- **Nombre de Chunks Créés : Y'a 5 Chunks au total.**

Pour mydb.cities1 : Il y a 3 chunks.
Pour mydb.cities2 : Vous avez créé 2 chunks.

Sharding par Plage pour cities1 :

La clé de shard est { "state" : 1 }.
Les chunks sont divisés en fonction des valeurs réelles de la clé de shard (state).
Les plages de chunks sont définies par des valeurs spécifiques (MA, RI).
Sharding Hashé pour cities2 :
La clé de shard est { "state" : "hashed" }.
Les chunks sont divisés en fonction d'une valeur de hachage des données de la clé de shard. Cela répartit plus uniformément les données, indépendamment de la distribution ordonnée des valeurs de clé.
Les plages de chunks sont basées sur des valeurs de hachage (NumberLong(0)), pas sur les valeurs réelles de la clé.

```
mongos> sh.shardCollection("mydb.cities2", { "state": "hashed"} )
{ "collectionsharded" : "mydb.cities2", "ok" : 1 }
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("657c25114cae16f7d3076c67")
  }
  shards:
        {  "_id" : "shard1",  "host" : "localhost:27021" }
  active mongoses:
        "3.2.22" : 1
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "mydb",  "primary" : "shard1",  "partitioned" : true }
                mydb.cities1
                        shard key: { "state" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  3
                        { "state" : { "$minKey" : 1 } } -->> { "state" : "MA" } on : shard1 Timestamp(1, 1)
                        { "state" : "MA" } -->> { "state" : "RI" } on : shard1 Timestamp(1, 2)
                        { "state" : "RI" } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(1, 3)
                mydb.cities2
                        shard key: { "state" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  2
                        { "state" : { "$minKey" : 1 } } -->> { "state" : NumberLong(0) } on : shard1 Timestamp(1, 1)
                        { "state" : NumberLong(0) } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(1, 2)
```

5.1.3 : Populate collection cities2:

Q6. How many chunks are there now? Include a screen copy of the results of the command in your answer to support your answer. Compare the result with respect to the range sharding. Include a screen copy of the results of the command in your answer to support your answer.

La collection `mydb.cities1`:

utilise un sharding par plage sur la clé `state` et contient **3 chunks**, ce qui est idéal pour des requêtes basées sur des intervalles de valeurs mais peut causer un déséquilibre si la clé de shard est mal choisie.

La collection `mydb.cities2`:

utilise un sharding hashé sur la même clé et a **4 chunks**, favorisant une distribution uniforme des données, utile lorsque les valeurs de la clé de shard ne sont pas uniformément réparties ou que les requêtes ne sont pas basées sur des plages.

```
mongos> db.cities.find().forEach(
...  function(d) { db.cities2.insert(d); }
... )
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("657c25114cae16f7d3076c67")
  }
  shards:
        {  "_id" : "shard1",  "host" : "localhost:27021" }
  active mongoses:
        "3.2.22" : 1
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "mydb",  "primary" : "shard1",  "partitioned" : true }
                mydb.cities1
                        shard key: { "state" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  3
                        { "state" : { "$minKey" : 1 } } -->> { "state" : "MA" } on : shard1 Timestamp(1, 1)
                        { "state" : "MA" } -->> { "state" : "RI" } on : shard1 Timestamp(1, 2)
                        { "state" : "RI" } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(1, 3)
                mydb.cities2
                        shard key: { "state" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  4
                        { "state" : { "$minKey" : 1 } } -->> { "state" : NumberLong(0) } on : shard1 Timestamp(1, 1)
                        { "state" : NumberLong(0) } -->> { "state" : NumberLong("3630192931154748514") } on : shard1 Timestamp(1, 3)

                        { "state" : NumberLong("3630192931154748514") } -->> { "state" : NumberLong("8213220138195528769") } on : sh
ard1 Timestamp(1, 4)
                        { "state" : NumberLong("8213220138195528769") } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(1
, 5)
```

5.1.4 : Adding shards to a cluster :

```
C:\Users\aghil\OneDrive\Documents\TP MongoDB-20231215>mongod --shardsvr --dbpath db\shard2 --port 27022
2023-12-15T13:36:37.056+0100 I CONTROL  [initandlisten] MongoDB starting : pid=13700 port=27022 dbpath=db\shard2 64-bit
host=LAPTOP-6NUJLARE
2023-12-15T13:36:37.056+0100 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2023-12-15T13:36:37.056+0100 I CONTROL  [initandlisten] db version v3.2.22
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten] git version: 105acca0d443f9a47c1a5bd608fd7133840a58dd
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten] allocator: tcmalloc
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten] modules: none
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten] build environment:
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten]     distmod: 2008plus
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten]     distarch: x86_64
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten]     target_arch: x86_64
2023-12-15T13:36:37.057+0100 I CONTROL  [initandlisten] options: { net: { port: 27022 }, sharding: { clusterRole: "shard
svr" }, storage: { dbPath: "db\shard2" } }
2023-12-15T13:36:37.058+0100 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=8G,session_max=20000,e
viction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,co
mpressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),verbose
```

5.2.1 : add the new mongo instance to the cluster: :

Q6. Draw the new configuration of the cluster and label each element (router, config server and shards) with its corresponding port as you defined in the previous tasks.

```
mongos> db.runCommand( { addShard: "localhost:27022", name: "shard2" } )
{ "shardAdded" : "shard2", "ok" : 1 }
mongos>
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("657c25114cae16f7d3076c67")
  }
  shards:
        {  "_id" : "shard1",  "host" : "localhost:27021" }
        {  "_id" : "shard2",  "host" : "localhost:27022" }
  active mongoses:
        "3.2.22" : 1
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                3 : Success
  databases:
        {  "_id" : "mydb",  "primary" : "shard1",  "partitioned" : true }
                mydb.cities1
                        shard key: { "state" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  2
                                shard2  1
                        { "state" : { "$minKey" : 1 } } -->> { "state" : "MA" } on : shard2 Timestamp(2, 0)
                        { "state" : "MA" } -->> { "state" : "RI" } on : shard1 Timestamp(2, 1)
                        { "state" : "RI" } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(1, 3)
                mydb.cities2
                        shard key: { "state" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  2
                                shard2  2
                        { "state" : { "$minKey" : 1 } } -->> { "state" : NumberLong(0) } on : shard2 Timestamp(2, 0)
                        { "state" : NumberLong(0) } -->> { "state" : NumberLong("3630192931154748514") } on : shard2 Timestamp(3, 0)

                        { "state" : NumberLong("3630192931154748514") } -->> { "state" : NumberLong("8213220138195528769") } on : sh
ard1 Timestamp(3, 1)
                        { "state" : NumberLong("8213220138195528769") } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(1
, 5)
```

le schema :

Composant	Type	Port	Description
mongos	Routeur	27017	Gère les requêtes, les achemine vers le shard approprié.
ConfigDBs	DB de Config	N/A	Stocke les métadonnées et les paramètres de configuration du cluster.
Shard 1	Shard	27021	Stocke une partie des données du cluster.
Shard 2	Shard	27022	Stocke une autre partie des données du cluster.

6.1 : Sharding using tagged shards :

```
mongos> use admin
switched to db admin
mongos> db.runCommand( { addShard: "localhost:27023", name: "shard3" } )
{ "shardAdded" : "shard3", "ok" : 1 }
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("657c25114cae16f7d3076c67")
  }
  shards:
        {  "_id" : "shard1",  "host" : "localhost:27021" }
        {  "_id" : "shard2",  "host" : "localhost:27022" }
        {  "_id" : "shard3",  "host" : "localhost:27023" }
  active mongoses:
        "3.2.22" : 1
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                5 : Success
  databases:
        {  "_id" : "mydb",  "primary" : "shard1",  "partitioned" : true }
                mydb.cities1
                        shard key: { "state" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  1
                                shard2  1
                                shard3  1
                        { "state" : { "$minKey" : 1 } } -->> { "state" : "MA" } on : shard2 Timestamp(2, 0)
                        { "state" : "MA" } -->> { "state" : "RI" } on : shard3 Timestamp(3, 0)
                        { "state" : "RI" } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(3, 1)
                mydb.cities2
                        shard key: { "state" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  1
                                shard2  2
                                shard3  1
                        { "state" : { "$minKey" : 1 } } -->> { "state" : NumberLong(0) } on : shard2 Timestamp(2, 0)
                        { "state" : NumberLong(0) } -->> { "state" : NumberLong("3630192931154748514") } on : shard2 Timestamp(3, 0)
                        { "state" : NumberLong("3630192931154748514") } -->> { "state" : NumberLong("8213220138195528769") } on : shard3 Timestamp(4, 0)
                        { "state" : NumberLong("8213220138195528769") } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(4, 1)
Démarrer
```

6.2 : Associate tags to shard instances:

```
mongos> sh.addShardTag("shard1", "CA")
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
mongos> sh.addShardTag("shard2", "NY")
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
mongos> sh.addShardTag("shard3", "Others")
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
mongos>
```

6.3 : Associate shard key ranges to tagged shards:

```
mongos> sh.addTagRange("mydb.cities3", { state: MinKey }, { state: "CA" }, "Others")
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : {
                "ns" : "mydb.cities3",
                "min" : {
                        "state" : { "$minKey" : 1 }
                }
        }
})
mongos> sh.addTagRange("mydb.cities3", { state: "CA" }, { state: "CA_" }, "CA")
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : {
                "ns" : "mydb.cities3",
                "min" : {
                        "state" : "CA"
                }
        }
})
mongos> sh.addTagRange("mydb.cities3", { state: "CA_" }, { state: "NY" }, "Others")
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : {
                "ns" : "mydb.cities3",
                "min" : {
                        "state" : "CA_"
                }
        }
})
```

6.4 : Display cluster information:

```
sharding version: {
      "_id" : 1,
      "minCompatibleVersion" : 5,
      "currentVersion" : 6,
      "clusterId" : ObjectId("657c25114cae16f7d3076c67")
}
shards:
      {  "_id" : "shard1",  "host" : "localhost:27021",  "tags" : [ "CA" ] }
      {  "_id" : "shard2",  "host" : "localhost:27022",  "tags" : [ "NY" ] }
      {  "_id" : "shard3",  "host" : "localhost:27023",  "tags" : [ "Others" ] }
active mongoses:
      "3.2.22" : 1
balancer:
      Currently enabled:  yes
      Currently running:  no
      Failed balancer rounds in last 5 attempts:  0
      Migration Results for the last 24 hours:
            12 : Success
databases:
      {  "_id" : "mydb",  "primary" : "shard1",  "partitioned" : true }
            mydb.cities1
                  shard key: { "state" : 1 }
                  unique: false
                  balancing: true
                  chunks:
                        shard1  1
                        shard2  1
                        shard3  1
                  { "state" : { "$minKey" : 1 } } -->> { "state" : "MA" } on : shard2 Timestamp(2, 0)
                  { "state" : "MA" } -->> { "state" : "RI" } on : shard3 Timestamp(3, 0)
                  { "state" : "RI" } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(3, 1)
            mydb.cities2
                  shard key: { "state" : "hashed" }
                  unique: false
                  balancing: true
                  chunks:
                        shard1  1
                        shard2  2
                        shard3  1
                  { "state" : { "$minKey" : 1 } } -->> { "state" : NumberLong(0) } on : shard2 Timestamp(2, 0)
                  { "state" : NumberLong(0) } -->> { "state" : NumberLong("3630192931154748514") } on : shard2 Timestamp(3, 0)
                  { "state" : NumberLong("3630192931154748514") } -->> { "state" : NumberLong("8213220138195528769") } on : shard3 Timestamp(4, 0)
                  { "state" : NumberLong("8213220138195528769") } -->> { "state" : { "$maxKey" : 1 } } on : shard1 Timestamp(4, 1)
            mydb.cities3
                  shard key: { "state" : 1 }
                  unique: false
                  balancing: true
                  chunks:
                        shard1  1
                        shard2  1
                        shard3  5
                  { "state" : { "$minKey" : 1 } } -->> { "state" : "CA" } on : shard3 Timestamp(8, 1)
                  { "state" : "CA" } -->> { "state" : "CA_" } on : shard1 Timestamp(6, 0)
                  { "state" : "CA_" } -->> { "state" : "MA" } on : shard3 Timestamp(7, 0)
                  { "state" : "MA" } -->> { "state" : "NY" } on : shard3 Timestamp(3, 6)
                  { "state" : "NY" } -->> { "state" : "NY_" } on : shard2 Timestamp(8, 0)
                  { "state" : "NY_" } -->> { "state" : "RI" } on : shard3 Timestamp(3, 9)
                  { "state" : "RI" } -->> { "state" : { "$maxKey" : 1 } } on : shard3 Timestamp(4, 0)
                   tag: Others  { "state" : { "$minKey" : 1 } } -->> { "state" : "CA" }
                   tag: CA  { "state" : "CA" } -->> { "state" : "CA_" }
                   tag: Others  { "state" : "CA_" } -->> { "state" : "NY" }
                   tag: NY  { "state" : "NY" } -->> { "state" : "NY_" }
                   tag: Others  { "state" : "NY_" } -->> { "state" : { "$maxKey" : 1 } }
```

Q7. Analyze the results and explain the logic behind this tagging strategy. Connect to the shard that contains the data about California, and count the documents. Do the same operation with the other shards. Is the sharded data collection complete with respect to initial one?

L'ajout de tags aux shards dans un cluster MongoDB est une stratégie pour contrôler la distribution des données. Dans votre cas, vous avez affecté des tags spécifiques à chaque shard :

- shard1 a reçu le tag CA, indiquant qu'il devrait contenir des données pour l'État de Californie (CA).

- shard2 a reçu le tag NY, indiquant qu'il devrait contenir des données pour l'État de New York (NY).

- shard3 a reçu le tag Others, qui accueillera des données pour tous les autres États.

```
2023-12-15T14:31:45.632+0100 E QUERY    [thread1] SyntaxError: missing ; before statement @(shell):1:8

> use mydb
switched to db mydb
> db.cities3.count()
1516
> ^C
bye

C:\Users\aghil\OneDrive\Documents\TP MongoDB-20231215>mongo --host localhost:27022
MongoDB shell version: 3.2.22
connecting to: localhost:27022/test
> db.cities3.count()
0
> use mydb
switched to db mydb
> db.cities3.count()
1595
> ^C
bye

C:\Users\aghil\OneDrive\Documents\TP MongoDB-20231215>mongo --host localhost:27020
MongoDB shell version: 3.2.22
connecting to: localhost:27020/test
configsvr> use mydb
switched to db mydb
configsvr> db.cities3.count()
0
configsvr> ^C
bye

C:\Users\aghil\OneDrive\Documents\TP MongoDB-20231215>mongo --host localhost:27023
MongoDB shell version: 3.2.22
connecting to: localhost:27023/test
> use mydb
switched to db mydb
> db.cities3.count()
26242
```

Q8 :

1. Expliquez les raisons pour lesquelles vous avez choisi votre clé de partitionnement. Qui sont possibles implications lors de l'interrogation de données et lors de l'ajout/suppression de données ?

Cardinalité élevée: Une clé de shard avec une grande variété de valeurs aidera à une distribution plus uniforme des données.

Accès aux données: La clé doit correspondre aux modèles d'accès courants des requêtes pour optimiser les performances.

Écritures distribuées: Pour éviter les goulots d'étranglement, il est préférable que les écritures soient réparties entre les shards.