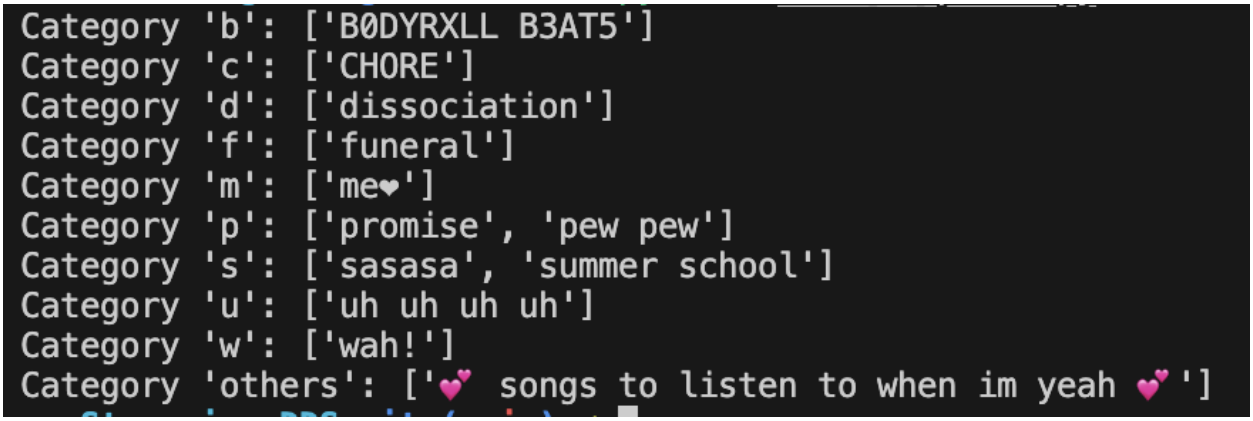
Part 2

Horizontal Fragmentation

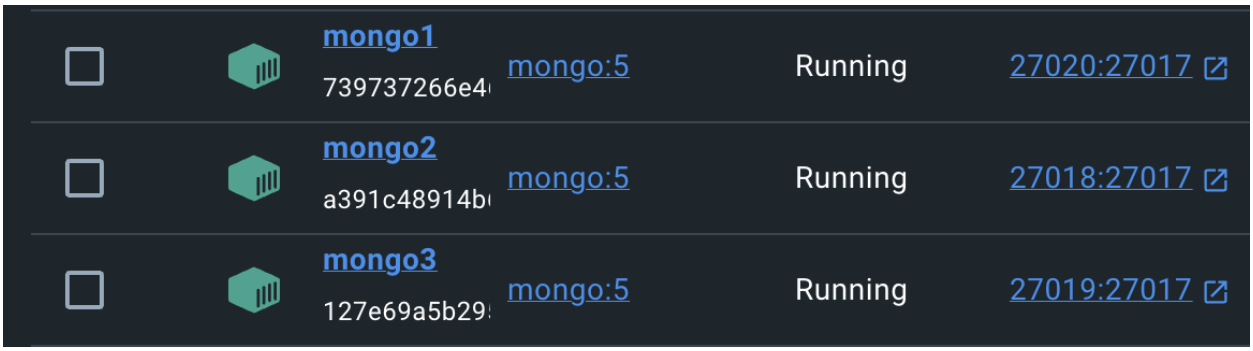
We have successfully completed the implementation of horizontal fragmentation in our database, a process that organizes our data more efficiently. This method involves dividing our music playlists into separate groups based on the first letter of each playlist name. For example, all playlists starting with 'A' are grouped together, then those starting with 'B', and so on. This approach makes it easier and faster to find and manage specific playlists, especially as our collection grows. During this process, we also created a special category for playlists that begin with non-letter characters like numbers or emojis. This ensures that every playlist is sorted appropriately.



Vertical Fragmentation

Vertical fragmentation, splitting a table into smaller ones based on columns, is not supported in MongoDB due to its design as a NoSQL database. Unlike traditional relational databases, MongoDB uses flexible, JSON-like documents instead of fixed tables and rows. This allows for efficient data management without needing to split data across multiple tables, making vertical fragmentation unnecessary in MongoDB's document-oriented structure.

Replica Sets



```
docker exec -it mongo1 mongosh --eval "rs.status()";
```

```
{ ok: 1 }
➜ Streaming-DDS git:(main) ✗ docker exec -it mongo1 mongosh --eval "rs.status()"
Current Mongosh Log ID: 655fd282e1c7333a644a22a2
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB: 5.0.22
Using Mongosh: 2.0.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/
```

```
{
set: 'myReplicaSet',
date: ISODate("2023-11-23T22:30:26.878Z"),
myState: 1,
term: Long("1"),
syncSourceHost: '',
syncSourceId: -1,
heartbeatIntervalMillis: Long("2000"),
majorityVoteCount: 2,
writeMajorityCount: 2,
votingMembersCount: 3,
writableVotingMembersCount: 3,
optimes: {
lastCommittedOpTime: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
lastCommittedWallTime: ISODate("2023-11-23T22:30:19.022Z"),
readConcernMajorityOpTime: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
appliedOpTime: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
durableOpTime: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
lastAppliedWallTime: ISODate("2023-11-23T22:30:19.022Z"),
lastDurableWallTime: ISODate("2023-11-23T22:30:19.022Z")
},
lastStableRecoveryTimestamp: Timestamp({ t: 1700778606, i: 1 }),
electionCandidateMetrics: {
lastElectionReason: 'electionTimeout',
lastElectionDate: ISODate("2023-11-23T22:30:17.501Z"),
electionTerm: Long("1"),
lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1700778606, i: 1 }), t: Long("-1") },
lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1700778606, i: 1 }), t: Long("-1") },
numVotesNeeded: 2,
priorityAtElection: 1,
electionTimeoutMillis: Long("10000"),
numCatchUpOps: Long("0"),
newTermStartDate: ISODate("2023-11-23T22:30:17.537Z"),
wMajorityWriteAvailabilityDate: ISODate("2023-11-23T22:30:18.982Z")
},
members: [
{
_id: 0,
name: 'mongo1:27017',
health: 1,
state: 1,
stateStr: 'PRIMARY',
uptime: 43,
optime: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
optimeDate: ISODate("2023-11-23T22:30:19.000Z"),
lastAppliedWallTime: ISODate("2023-11-23T22:30:19.022Z"),
lastDurableWallTime: ISODate("2023-11-23T22:30:19.022Z"),
```

```
syncSourceHost: '',
syncSourceId: -1,
infoMessage: '',
electionTime: Timestamp({ t: 1700778617, i: 1 }),
electionDate: ISODate("2023-11-23T22:30:17.000Z"),
configVersion: 1,
configTerm: 1,
self: true,
lastHeartbeatMessage: ''
},
{
_id: 1,
name: 'mongo2:27017',
health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 20,
optime: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
optimeDurable: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
optimeDate: ISODate("2023-11-23T22:30:19.000Z"),
optimeDurableDate: ISODate("2023-11-23T22:30:19.000Z"),
lastAppliedWallTime: ISODate("2023-11-23T22:30:19.022Z"),
lastDurableWallTime: ISODate("2023-11-23T22:30:19.022Z"),
lastHeartbeat: ISODate("2023-11-23T22:30:25.525Z"),
lastHeartbeatRecv: ISODate("2023-11-23T22:30:25.028Z"),
pingMs: Long("0"),
lastHeartbeatMessage: '',
syncSourceHost: 'mongo1:27017',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
},
{
_id: 2,
name: 'mongo3:27017',
health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 20,
optime: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
optimeDurable: { ts: Timestamp({ t: 1700778619, i: 5 }), t: Long("1") },
optimeDate: ISODate("2023-11-23T22:30:19.000Z"),
optimeDurableDate: ISODate("2023-11-23T22:30:19.000Z"),
lastAppliedWallTime: ISODate("2023-11-23T22:30:19.022Z"),
lastDurableWallTime: ISODate("2023-11-23T22:30:19.022Z"),
lastHeartbeat: ISODate("2023-11-23T22:30:25.525Z"),
lastHeartbeatRecv: ISODate("2023-11-23T22:30:25.028Z"),
pingMs: Long("0"),
lastHeartbeatMessage: '',
syncSourceHost: 'mongo1:27017',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
}
],
ok: 1,
'$clusterTime': {
clusterTime: Timestamp({ t: 1700778619, i: 5 }),
signature: {
hash: Binary.createFromBase64("AAAAAAAAAAAAAAAAAAAAAAAAAAA=", 0),
keyId: Long("0")
}
```

```
},
operationTime: Timestamp({ t: 1700778619, i: 5 })
}
```

We have successfully set up MongoDB replica sets, improving our database's reliability and backup capabilities. This setup includes several MongoDB servers working together. The main server handles all the data changes, while the others keep copies of this data. This approach ensures that our data is safe and always available, even if one server has problems. We carefully configured each server and tested the entire system to ensure it works well together. This successful setup shows our commitment to a strong and reliable data management system.

The replica set also allows for automatic switching to a backup server if the main one fails. This means our data remains accessible with minimal interruption. We regularly check the system's performance to quickly fix any issues.