Thomas Amorsen, Rasmus Dalgaard & Markus Agnsgaard
1/5-2023

MongoDB SP Assignment questions

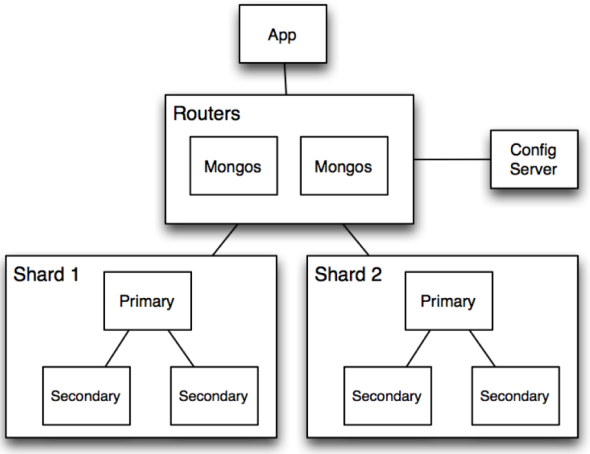
What is sharding in MongoDB

Sharding er en metode til at skalere ens MongoDB database horisontalt. Sharding gør det muligt at opdele databasens data over flere servere og øger derved databasens evne til at lagre større mængder af data.
Ny data bliver uddelegeret enten med en range-based eller en hash-based strategi for opdeling af data.

What are the different components required to implement sharding

1. Shards
   Shards er de individuelle DB instanser der hver især skal indeholde et subset af den samlede data. Shards kan have deres egne replicas for øget tilgængelighed og integritet.
2. Konfigurerings servere
   Disse servere lagrer Meta-data omkring den sharded cluster, som f.eks. shard nøgler og deres lokationer.
3. Router
   Routeren sørger for at dirigere forespørgsler til den korrekte shard på baggrund af shard nøglen.

Explain architecture of sharding in mongoDB?



Fra toppen i billedet er der en applikation der interagerer med MongoDB databasen. Når et read/write request bliver sendt til DB er der 1 eller flere routers der sørger for at omdirigere

et request til den korrekte server ud fra en shard nøgle gemt på konfigurations serveren. Hvert shard består af en primary og et vilkårligt antal secondary/replica servere. Primary server er den server hvor alt ny data bliver nedskrevet til, herefter opdaterer den alle secondary servere for at holde alle replicas opdaterede med den nyeste data. Hvis det skulle ske at primary server af en grund lukker ned finder konfigurations serveren ud af hvilken secondary servere der skal overtage som primary

Provide implementation of map and reduce function

"what are the top 10 hashtags used in the given tweets"

```
db.tweeter.aggregate([
  {
    $unwind: "$entities.hashtags"
  },
  {
    $group: {
      _id: "$entities.hashtags.text",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { count: -1 }
  },
    $limit: 10
])
```

Provide execution command for running MapReduce or the aggregate way of doing the same

Frontend blev lavet med et andet datasæt der indeholder students i stedet.

```
async function test() {
 try {
   await client.connect();
   console.log("Connected to MongoDB cluster");
   const database = client.db("School");
   const collection = database.collection("students");

   const pipeline = [
     { $unwind: "$scores" },
     { $match: { "scores.type": "exam" } },
     { $sort: { "scores.score": -1 } },
     {
       $group: {
```

```
          _id: null,
          topScore: { $push: "$scores.score" },
        },
      },
      {
        $project: {
          _id: 0,
          topScore: { $slice: ["$topScore", 10] },
        },
      },
    ];

    const result = await collection.aggregate(pipeline).toArray();

    console.log(result);
  } catch (error) {
    console.error("Error connecting to MongoDB cluster", error);
  } finally {
    await client.close();
  }
}
```

Provide top 10 recorded out of the sorted result.

(hint: use sort on the result returned by MapReduce or the aggregate way of doing the same)

```
{
  _id: 'FCBLive',
  count: 27
}
{
  _id: 'AngularJS',
  count: 21
}
{
  _id: 'nodejs',
  count: 20
}
{
  _id: 'LFC',
  count: 19
}
{
  _id: 'EspanyolFCB',
  count: 18
}
{
```

```
  _id: 'webinar',
  count: 16
}
{
  _id: 'IWCI',
  count: 16
}
{
  _id: 'javascript',
  count: 14
}
{
  _id: 'GlobalMoms',
  count: 14
}
{
  _id: 'RedBizUK',
  count: 12
}
```