Distributed Database Design and Implementation

In this experiment, we configure MongoDB's replica sets and sharding, then perform CRUD operations to test data distribution, high availability, and fault tolerance in a distributed database environment.  
  
const { MongoClient } = require('mongodb');

// Function to set up the database and perform operations

async function run() {

const uri = "mongodb://localhost:27025"; // Connect to the mongos router instance

const client = new MongoClient(uri);

try {

await client.connect();

console.log("Connected to MongoDB via mongos router on port 27025");

const adminDb = client.db("admin");

// Step 1: Enable Sharding on Database

console.log("Enabling sharding on 'myDatabase'...");

await adminDb.command({ enableSharding: "myDatabase" });

console.log("Sharding enabled on 'myDatabase'.");

// Step 2: Shard the Collection

console.log("Sharding collection 'myDatabase.myCollection' on 'userId'...");

await adminDb.command({

shardCollection: "myDatabase.myCollection",

key: { userId: 1 }

});

console.log("Sharding enabled for 'myDatabase.myCollection'.");

// Step 3: Insert Data

const db = client.db("myDatabase");

const collection = db.collection("myCollection");

console.log("Inserting documents...");

await collection.insertMany([

{ userId: 1, name: "Alice", age: 25, location: "Region1" },

{ userId: 2, name: "Bob", age: 30, location: "Region2" },

{ userId: 3, name: "Charlie", age: 28, location: "Region3" }

]);

console.log("Documents inserted.");

// Step 4: Query Data

console.log("Querying all documents...");

const docs = await collection.find().toArray();

console.log("Documents in 'myCollection':", docs);

// Step 5: Update Data

console.log("Updating document with userId 1...");

await collection.updateOne(

{ userId: 1 },

{ $set: { age: 26 } }

);

console.log("Document updated.");

// Step 6: Delete Data

console.log("Deleting document with userId 2...");

await collection.deleteOne({ userId: 2 });

console.log("Document deleted.");

// Step 7: Check Sharding Status (using `sh.status()` isn't available in Node.js)

console.log("Sharding status can be checked in the MongoDB shell using `sh.status()`.");

} finally {

await client.close();

}

}

run().catch(console.dir);  
  
  
**output:**