Redis and MongoDB Integration

1. Introduction

This lab focuses on understanding and implementing NoSQL databases (Redis and MongoDB) within a Spring Boot application. The objective is to explore the differences between relational and NoSQL databases and to integrate and perform CRUD operations on Redis and MongoDB using Spring Data.

2. Objectives

- Comprehend the fundamental differences between relational and NoSQL databases.
- Integrate Redis and MongoDB with Spring Data for efficient data access and storage.
- Perform CRUD (Create, Read, Update, Delete) operations on Redis and MongoDB.
- Explore advanced features of Redis and MongoDB to enhance data management in a Hospital System context.

3. Implementation Details

3.1. Redis Integration

- **Setting Up Redis Connection:**
    - Configured a Redis server connection in the Spring Boot application.
    - Performed basic operations such as setting and getting key-value pairs, managing hashes, and handling lists.
    - Example

```java
redisTemplate.opsForValue().set("doctor:123", "Dr. Smith");
String doctor = redisTemplate.opsForValue().get("doctor:123");
```

Spring Data Redis:

- Integrated Spring Data Redis to interact with Redis using repositories.
- Implemented repository-based access for managing complex data structures, like hash maps, in Redis.

3.2. MongoDB Integration

- **Setting Up MongoDB Connection:**

    - Switched from a relational database setup to MongoDB to handle the storage of unstructured data.
    - Established a MongoDB connection and created collections to store documents (e.g., doctor records).

Spring Data MongoDB:

- Utilized Spring Data MongoDB for performing CRUD operations on documents.
- Example:

```java
@Repository
public interface DoctorMongoRepository extends MongoRepository<Doctor, String>
{
    List<Doctor> findBySpecialty(String specialty);
}
```

querying NoSQL Databases:

- Executed basic and advanced queries on both Redis and MongoDB to retrieve and manipulate data efficiently.
- Example MongoDB Query

```java
List<Doctor> doctors = doctorMongoRepository.findBySpecialty("Cardiology");
```

4. Comparison: Relational vs. NoSQL Data Modeling

- **Relational Databases:**
    - Data is stored in structured tables with predefined schemas.
    - Ideal for complex transactions and consistency across multiple related entities.
    - Example: Storing patient records in a relational database with foreign keys linking to appointments and doctors.
- **NoSQL Databases:**

- Data is stored in unstructured or semi-structured formats, such as key-value pairs or JSON-like documents.
- Offers flexibility in data modeling, making it suitable for handling large volumes of diverse data types.
- Example: Storing dynamic and varying doctor profiles in MongoDB, allowing for different fields across documents.

6. Conclusion

This lab provided practical experience in integrating and using NoSQL databases (Redis and MongoDB) in a Spring Boot application. The ability to perform CRUD operations and query data in a NoSQL context demonstrated the flexibility and performance benefits of these databases in a Hospital System