**SỬ DỤNG MYSQL TRONG MICROSERVICES**

**1. Giới thiệu**

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ phổ biến, được sử dụng rộng rãi trong các ứng dụng Microservices để lưu trữ dữ liệu. Khi triển khai trong Microservices, MySQL có thể được sử dụng theo mô hình cơ sở dữ liệu riêng biệt cho từng dịch vụ hoặc chia sẻ một cơ sở dữ liệu chung.

Báo cáo này sẽ trình bày cách sử dụng MySQL trong Microservices, bao gồm cách kết nối, quản lý dữ liệu và tối ưu hóa hiệu suất.

**2. Cấu trúc và Kiến trúc MySQL trong Microservices**

**2.1. Mô hình Database trong Microservices**

Có hai mô hình phổ biến:

* **Database Per Service**: Mỗi dịch vụ có cơ sở dữ liệu riêng, đảm bảo tính độc lập.
* **Shared Database**: Nhiều dịch vụ chia sẻ cùng một cơ sở dữ liệu, phù hợp với hệ thống nhỏ nhưng có thể gây phụ thuộc cao.

**2.2. Ưu và nhược điểm của MySQL trong Microservices**

**Ưu điểm:**

* Dễ triển khai và sử dụng.
* Hỗ trợ ACID, đảm bảo tính nhất quán của dữ liệu.
* Cộng đồng lớn, tài liệu phong phú.

**Nhược điểm:**

* Chưa tối ưu cho kiến trúc phân tán.
* Có thể gặp vấn đề về performance khi scale lớn.

**3. Kết nối MySQL với Spring Boot Microservices**

**3.1. Cấu hình MySQL trong Spring Boot**

Thêm dependency trong pom.xml:

<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <scope>runtime</scope>  
</dependency>  
Cấu hình application.properties:  
spring.datasource.url=jdbc:mysql:*//localhost:3306/microservices\_db*  
spring.datasource.username=root  
spring.datasource.password=123456  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

**3.2. Tạo Entity và Repository**

@Entity  
public class Product {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String name;  
 private double price;  
   
 *// Getters and Setters*  
}  
@Repository  
public interface ProductRepository extends JpaRepository<Product, Long> {}

**4. Tối ưu hóa MySQL trong Microservices**

**4.1. Connection Pooling**

Sử dụng HikariCP để quản lý kết nối hiệu quả:

spring.datasource.hikari.maximum-pool-size=10

**4.2. Caching**

Dùng Redis hoặc Hibernate Second-Level Cache để giảm tải MySQL.

**4.3. Phân mảnh dữ liệu (Sharding)**

Chia cơ sở dữ liệu thành nhiều phần nhỏ để tối ưu truy vấn.