后台管理项目多数据源管理方案实战

图灵：楼兰

整个电商管理后台本质上是一个访问频率比较低的CRUD管理系统，所以本身不需要考虑微服务拆分的事情。接入微服务体系也只是为了能够调用其他的微服务。

但是，在做电商管理项目时，我们为了简化业务流程，在做后端管理系统时，并没有严格按照微服务原则进行数据隔离，而是让管理系统直接访问所有的业务数据。但是，这也带来了新的问题，怎么让管理系统可以同时管理多个数据库的数据？

> 大型项目中，这种多数据源的访问是很常见的场景，大家想想，ShardingSphere框架适合用来做这种跨库的数据源切换吗？

一、电商后台项目需要访问的数据源说明

整个电商项目的数据库设计情况：

库名	表前缀	说明
tl_mall_goods	pms_	商品相关表
tl_mall_normal	cms_	其他辅助功能相关表
tl_mall_promotion	sms_	促销相关表
tl_mall_user	ums_	用户管理相关表
tl_mall_cart	oms_cart_	购物车相关表
tl_mall_order	oms_	订单相关表

这些不同库中的很多基础数据，除了购物车模块外，都需要由电商管理后台进行统一管理。所以，对于电商管理系统，会采用多数据源管理的方式，尽量快速的完成基础数据维护。其中，订单库由于进行了分库分表，管理比较复杂，所以电商管理后台不会直接访问订单相关的表，而是通过微服务的方式调用订单模块的相关功能来间接管理订单。

二、电商后台使用MyBatis-plus快速访问多个数据源

电商后台项目使用的MyBatis-plus框架访问数据库。对于MyBatis和MyBatis-plus框架，这里就不多做介绍了。而我们这个电商后台管理项目，与常见的一些普通的管理系统的最大区别，在于这个电商项目管理数据的方式更为直接粗暴，直接跨多个数据库管理的后台数据。这里分享三种常用的多数据源管理方案：

> 示例参见：springboot_dynamicdatasource

1、使用Spring提供的AbstractRoutingDataSource

这种方式的核心是使用Spring提供的AbstractRoutingDataSource抽象类，注入多个数据源。

```
@Component
@Primary   // 将该Bean设置为主要注入Bean
public class DynamicDataSource extends AbstractRoutingDataSource {
    // 当前使用的数据源标识
    public static ThreadLocal<String> name=new ThreadLocal<>();
    // 写库
    @Autowired
    DataSource dataSource1;
    // 读库
    @Autowired
    DataSource dataSource2;
    // 返回当前数据源标识
    @Override
    protected Object determineCurrentLookupKey() {
        return name.get();
    }
    @Override
    public void afterPropertiesSet() {
        // 为targetDataSources初始化所有数据源
        Map<Object, Object> targetDataSources=new HashMap<>();
        targetDataSources.put("W",dataSource1);
        targetDataSources.put("R",dataSource2);
        super.setTargetDataSources(targetDataSources);
        // 为defaultTargetDataSource 设置默认的数据源
        super.setDefaultTargetDataSource(dataSource1);
        super.afterPropertiesSet();
    }
}
```

将自己实现的DynamicDataSource注册成为默认的DataSource实例后，只需要在每次使用DataSource时，提前改变一下其中的name标识，就可以快速切换数据源。

```
@Component
@Aspect
public class DynamicDataSourceAspect implements Ordered {
    // 在每个访问数据库的方法执行前执行。
    @Before("within(com.tuling.dynamic.datasource.service.impl.*) && @annotation(wr)")
    public void before(JoinPoint point, WR wr){
        String name = wr.value();
        DynamicDataSource.name.set(name);
        System.out.println(name);
    }
    @Override
    public int getOrder() {
        return 0;

    }
```

```
}
```

> 完整示例参见01-dynamic_datasource模块。

2、使用MyBatis注册多个SqlSessionFactory

如果使用MyBatis框架，要注册多个数据源的话，就需要将MyBatis底层的DataSource、SqlSessionFactory、DataSourceTransactionManager这些核心对象一并进行手动注册。例如：

```
@Configuration
@MapperScan(basePackages = "com.tuling.datasource.dynamic.mybatis.mapper.r",
        sqlSessionFactoryRef="rSqlSessionFactory")
public class RMyBatisConfig {
    @Bean
    @ConfigurationProperties(prefix = "spring.datasource.datasource2")
    public DataSource dataSource2() {
        // 底层会自动拿到spring.datasource中的配置， 创建一个DruidDataSource
        return DruidDataSourceBuilder.create().build();
    }
    @Bean
    @Primary
    public SqlSessionFactory rSqlSessionFactory()
            throws Exception {
        final SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource2());
        // 指定主库对应的mapper.xml文件
        /*sessionFactory.setMapperLocations(new PathMatchingResourcePatternResolver()
                .getResources("classpath:mapper/r/*.xml"));*/
        return sessionFactory.getObject();
    }
    @Bean
    public DataSourceTransactionManager rTransactionManager(){
        DataSourceTransactionManager dataSourceTransactionManager = new
DataSourceTransactionManager();
        dataSourceTransactionManager.setDataSource(dataSource2());
        return dataSourceTransactionManager;
    }

    @Bean
    public TransactionTemplate rTransactionTemplate(){
        return new TransactionTemplate(rTransactionManager());
    }
}
```

这样就完成了读库的注册。而读库与写库之间，就可以通过指定不同的Mapper和XML文件的地址来进行区分。

> 完整示例02-dynamic-mybatis模块。

3、使用dynamic-datasource框架

dynamic-datasource是MyBaits-plus作者设计的一个多数据源开源方案。使用这个框架需要引入对应的pom依赖

```
<dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>dynamic-datasource-spring-boot-starter</artifactId>
            <version>3.5.0</version>
        </dependency>
```

这样就可以在SpringBoot的配置文件中直接配置多个数据源。

```
spring:
  datasource:
    dynamic:
      #设置默认的数据源或者数据源组,默认值即为master
      primary: master
      #严格匹配数据源,默认false. true未匹配到指定数据源时抛异常,false使用默认数据源
      strict: false
      datasource:
        master:
          url: jdbc:mysql://127.0.0.1:3306/datasource1?
serverTimezone=UTC&useUnicode=true&characterEncoding=UTF8&useSSL=false
          username: root
          password: 123456
          initial-size: 1
          min-idle: 1
          max-active: 20
          test-on-borrow: true
          driver-class-name: com.mysql.cj.jdbc.Driver
        slave_1:
          url: jdbc:mysql://127.0.0.1:3306/datasource2?
serverTimezone=UTC&useUnicode=true&characterEncoding=UTF8&useSSL=false
          username: root
          password: 123456
          initial-size: 1
          min-idle: 1
          max-active: 20
          test-on-borrow: true
          driver-class-name: com.mysql.cj.jdbc.Driver
```

这样就配置完成了master和slave_1两个数据库。

接下来在使用时，只要在对应的方法或者类上添加@DS注解即可。例如

```
@Service
public class FrendImplService implements FrendService {

    @Autowired
    FrendMapper frendMapper;
    @Override
    @DS("slave")  // 从库，  如果按照下划线命名方式配置多个  ，  可以指定前缀即可（组名）
    public List<Frend> list() {
        return frendMapper.list();
    }

    @Override
```

```
    @DS("master")
    public void save(Frend frend) {
        frendMapper.save(frend);
    }
    @DS("master")
    @DSTransactional
    public void saveAll(){
        // 执行多数据源的操作
    }
}
```

> 完整示例查看03-dynamic_datasource_framework模块。

当前电商管理后台采用了第三种方式来进行多数据源的管理。

> oms订单数据除外。因为订单相关数据已经完成了分库分表，不能直接查。

有道云笔记链接：https://note.youdao.com/s/KEiy5RP7