

# 一、MySQL集群

## 1. 数据库集群的必要性

### 1-1. 单节点数据库的弊病

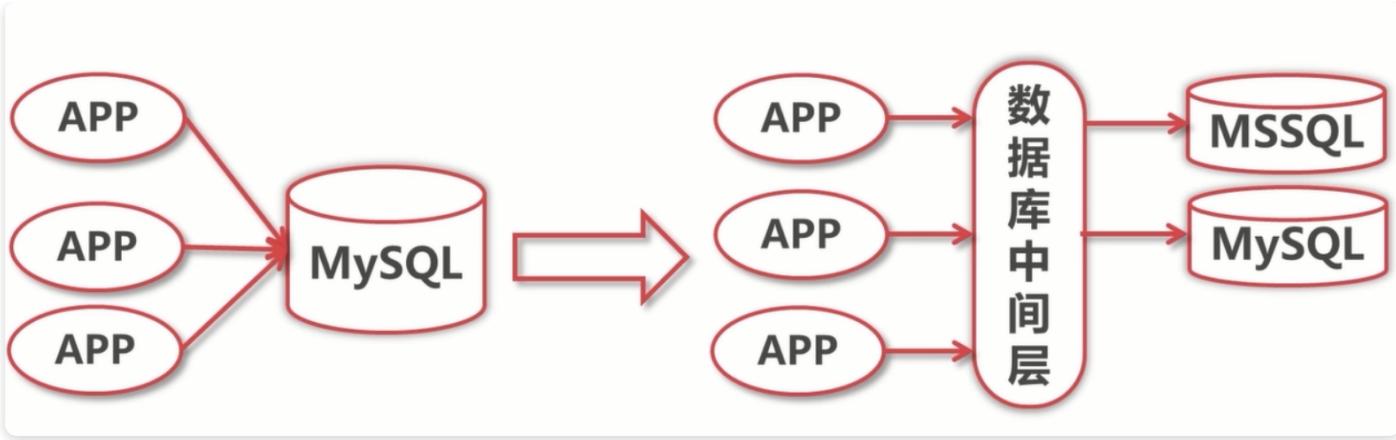
- 单节点的数据库无法满足性能上的要求
- 单节点的数据库没有冗余设计，无法满足高可用

## 2. MyCAT

### 2-1. 什么是MyCAT?

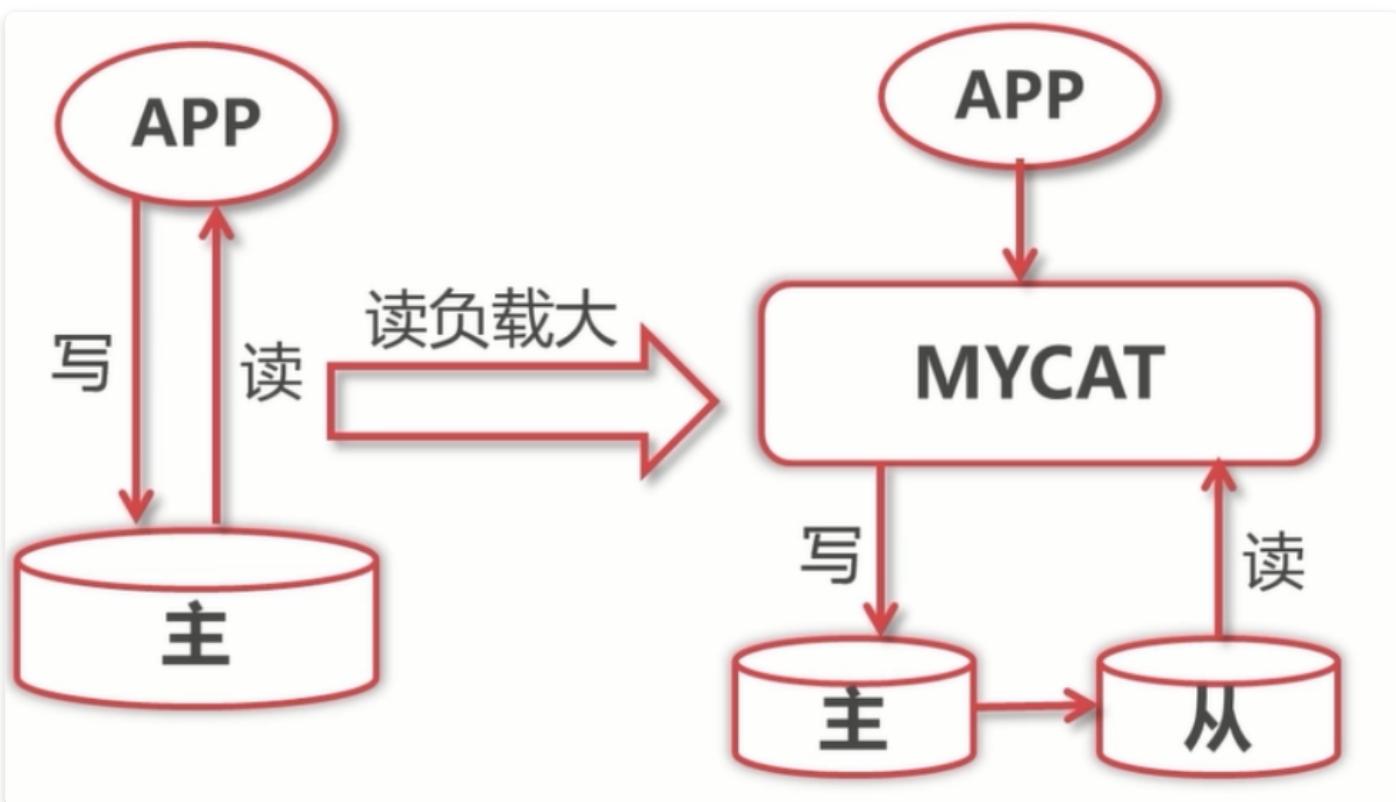
- MyCAT相当于MySQL的Server层
- MySQL相当于MyCAT的存储层
- MyCAT不存储数据，所有数据存储在MySQL中
- MyCAT是一个数据库中间层
- 可以实现对后端数据库的分库分表和读写分离
- 对前端应用隐藏了后端数据库的存储逻辑

### 2-2. 什么是数据库中间层及其作用?



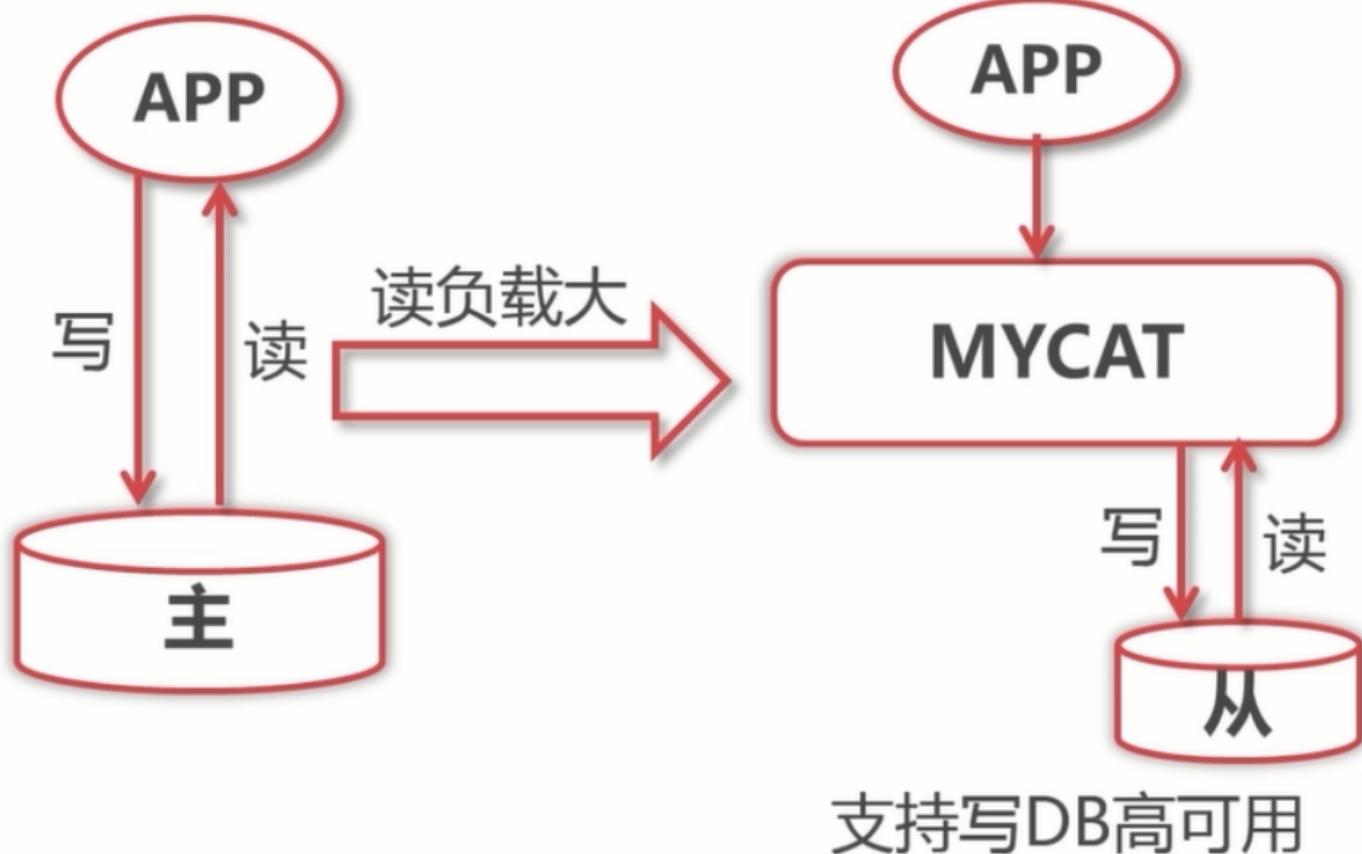
enter description here

## 2-2-1. 实现后端数据库的读写分离及负载均衡



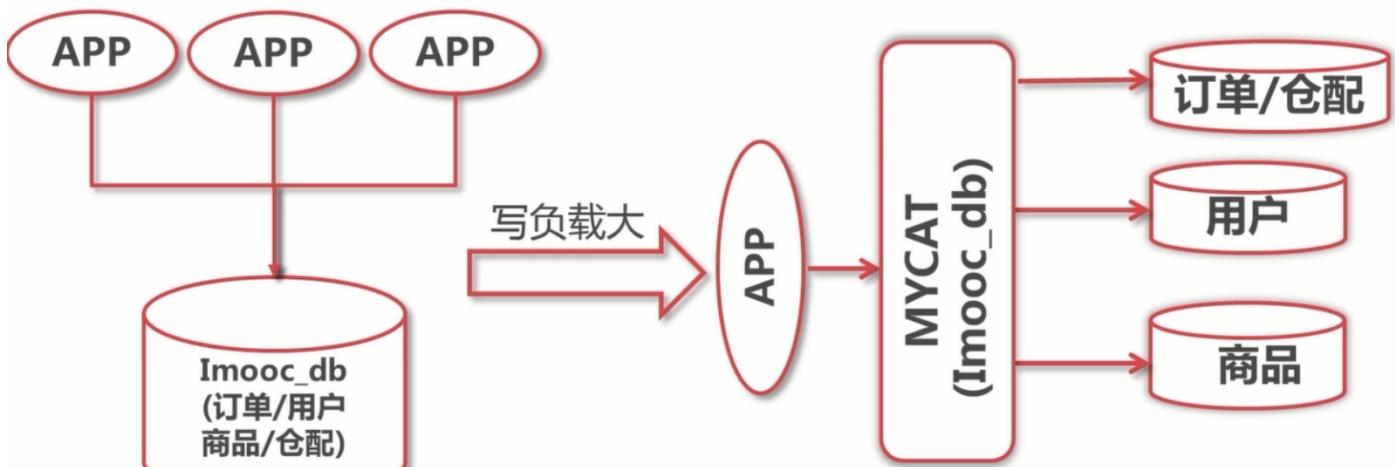
enter description here

当写数据库发生宕机，MyCAT会将写数据库踢出集群(只适用一主一从)



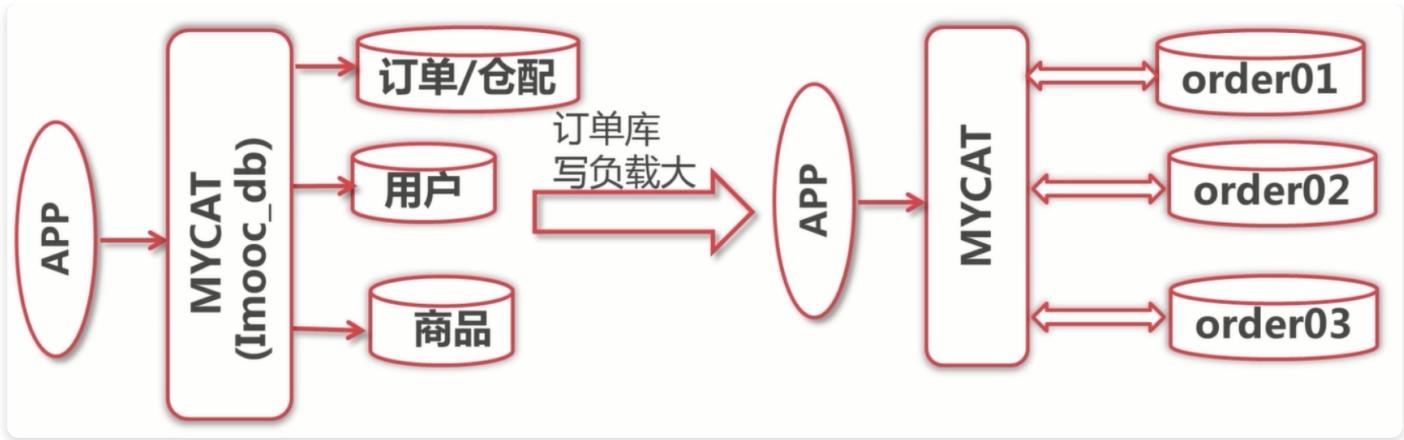
## 2-2-2. 对业务数据库进行垂直切分

数据库Imooc\_db集群写负载过大时，将集群中的表进行拆分，按模块分库



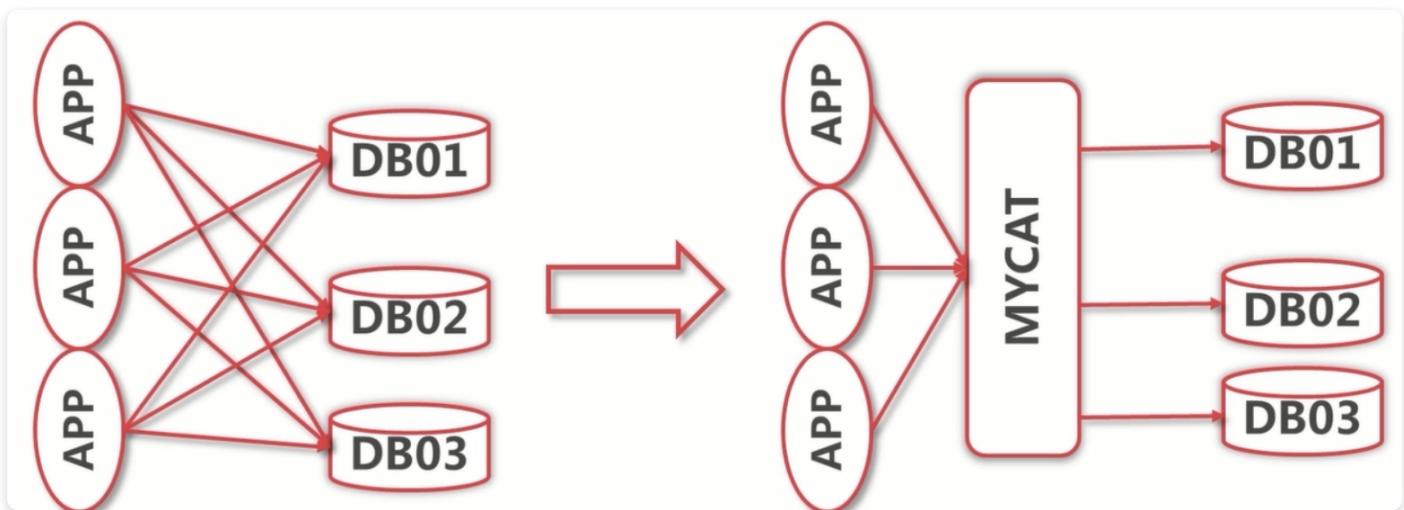
## 2-2-3. 对业务数据库进行水平切分





enter description here

## 2-2-4. 控制数据库连接的数量



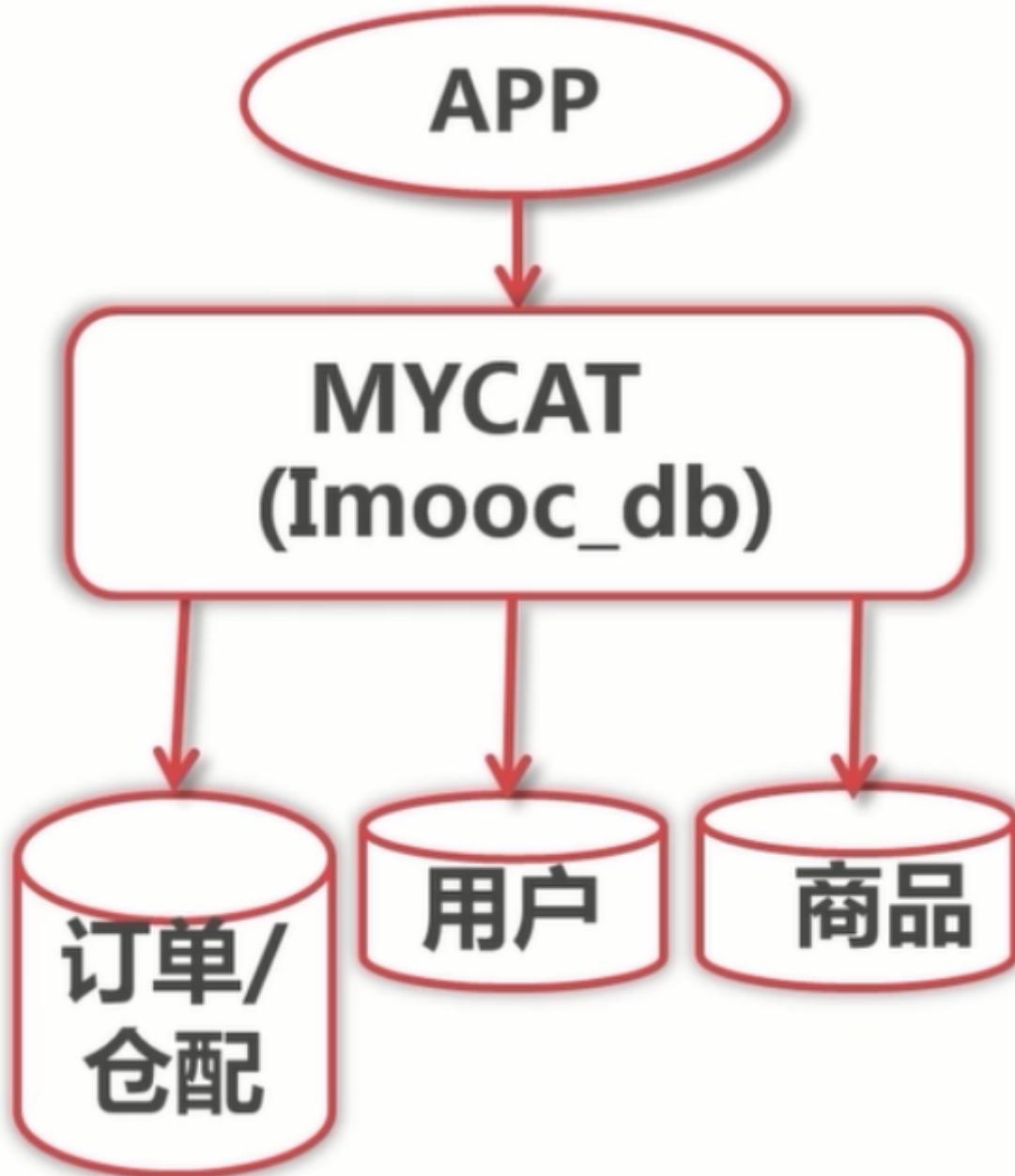
enter description here

## 2-3. MyCAT的基本元素

### 2-3-1. 逻辑库

- 对应用来说相当于MySQL中的数据库
- 逻辑库对应后端多个物理数据库

- 逻辑库并不保存数据



## 2-3-2. 逻辑表

- 对应用来说相当于MySQL中的数据表
- 逻辑库对应后端多个物理数据库中的表
- 逻辑表并不保存数据

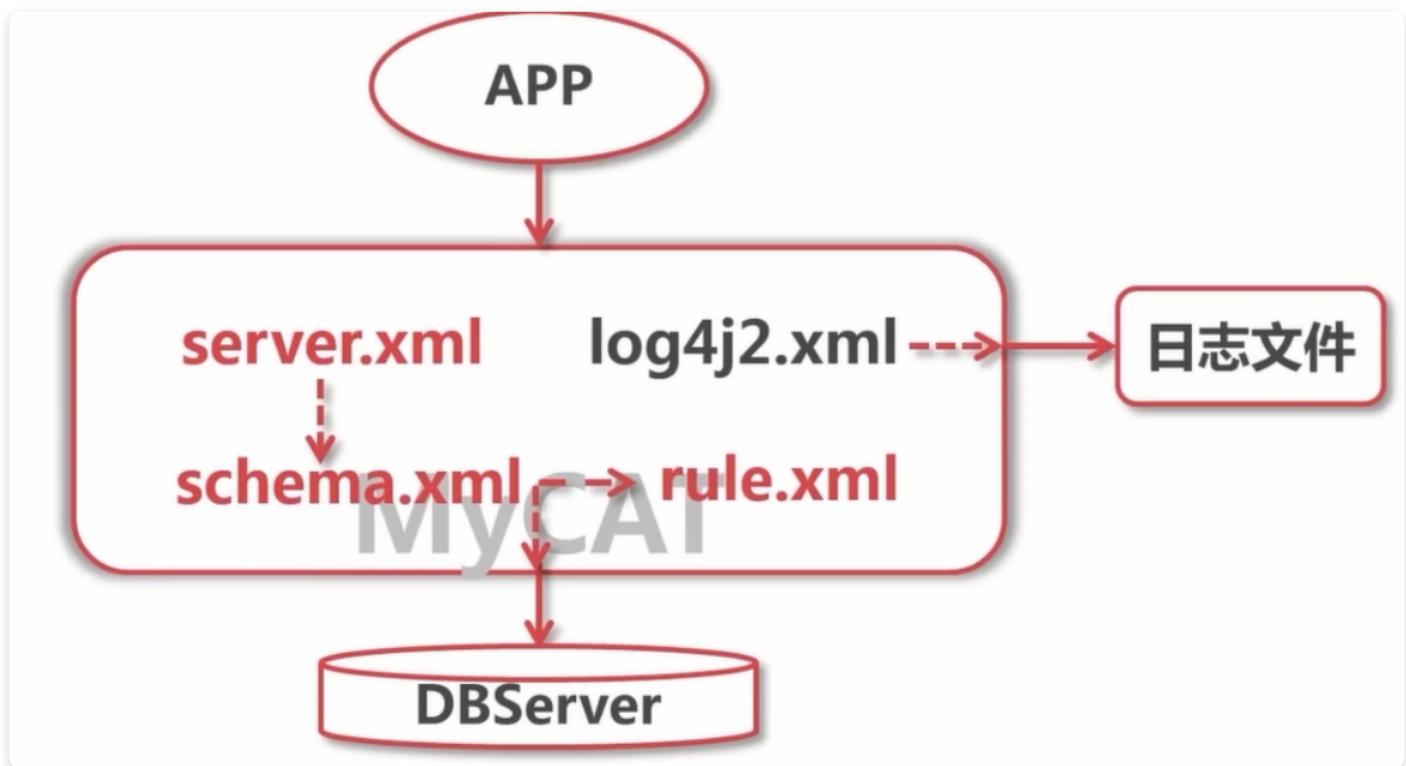
### 2-3-2-1. 逻辑表的类别

- 分片表与非分片表按是否被分片划分(分片表就是具有相同表结构，但是存储在不同数据库中)
- 全局表，在所有分片都存在的表
- ER关系表，按ER关系进行分片的表

## 2-4. MySQL核心配置

常用配置文件间的关系

- `server.xml`--对系统参数和用户权限进行配置
- `schema.xml`--对逻辑库和逻辑表进行配置
- `rule.xml`--逻辑表中涉及水平切分进行规则配置
- `log4j2.xml`--配置日志格式



### 2-4-1. `server.xml`

`server.xml`用于配置系统相关参数、用户访问权限以及SQL防火墙及SQL拦截功能

```

<!-- <system>配置MyCAT系统参数-->
<system>
<!-- key-value -->
<property name = "${key} ${value}"></property>
</system>

<system>
<property name="serverPort">8066</property> <property
name="managerPort">9066</property>
</system>

<!-- <user>配置MyCAT的访问用户及权限 -->
<user name="user">

```

```

<property name="password">user</property>
<!-- 用户可访问的数据库 -->
<property name="schemas">TESTDB</property>
<property name="readOnly">true</property>
</user>

<!-- 多数据库访问设置-->
<property name="schemas">db1, db2, db3...</property>

<!-- 表级DML权限设置 -->
<!-- 数字代表: insert、updata、seect、delete-->
<privileges check="false">
    <schema name="TESTDB" dml="0110" >
        <table name="tb01" dml="0000"></table>
        <table name="tb02" dml="1111"></table>
    </schema>
</privileges>

```

## 2-4-2. log4j2.xml

- 配置输出日志的格式
- 配置输出日志的级别

```

<!-- %5p表示输出的日志级别 -->
<!-- %t日志中记录线程名称 -->
<!-- %m输出代码中提及的消息 -->
<!-- %n是换行符 -->
<PatternLayout>
    <Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %5p [%t] (%1) -
    %m%n</Pattern>
</PatternLayout>

<!-- 日志级别 All < Trace < Debug < Info < Warn < Error < Fatal < OFF -->
<asyncRoot level="info" includeLocation="true">

```

## 2-4-3. rule.xml

- 配置水平分片的分片规则**
- 配置分片规则所对应的分片函数

```

<tableRule name="rule1"><!-- rule1是分片函数(不可重复) -->
    <rule>
        <columns>id</columns><!-- 分片列, MyCAT对分片列进行运算, 最终根据
结果看如何分片 -->
        <algorithm>func1</algorithm><!-- 分片算法, 取自<function>的name属
性 -->
    </rule>

```

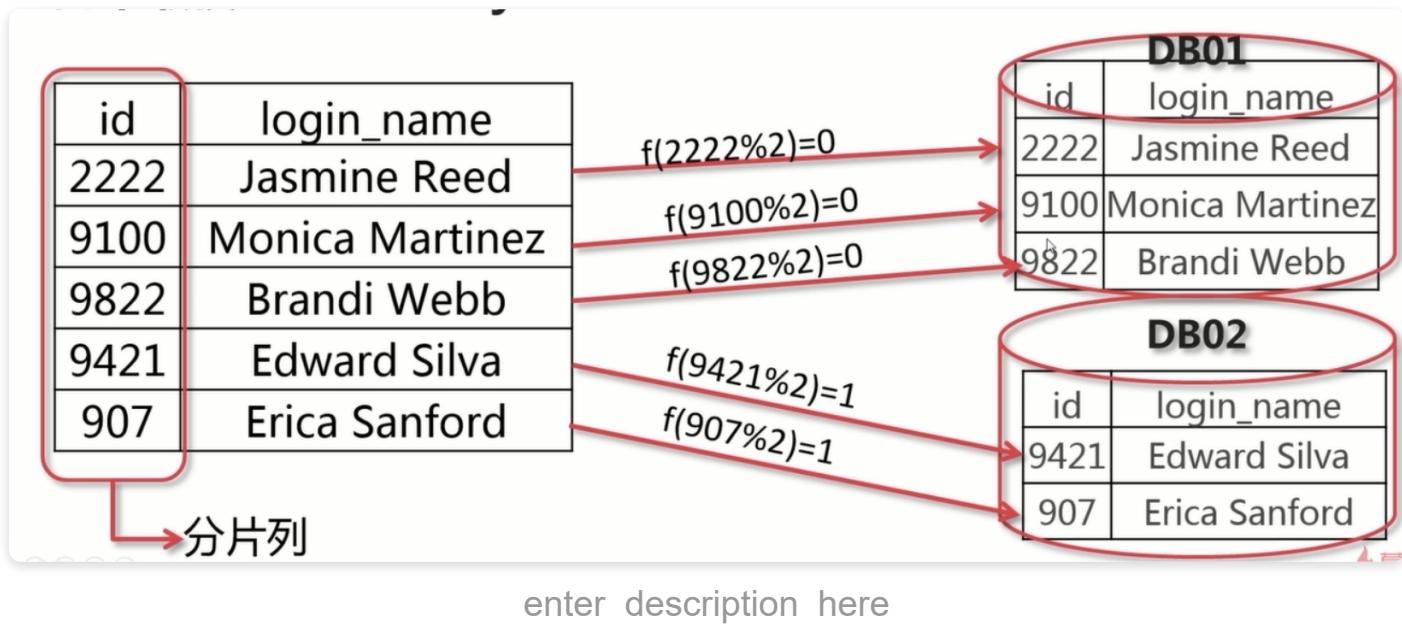
```
</tableRule>

<function name="murmur"
    class="io.mycat.route.function.PartitionByMurmurHash"><!-- Java类 -->
    <property name="seed">0</property><!-- 默认是0 -->
    <property name="count">2</property><!-- 要分片的数据库节点数量，必须指定，否则没法分片 -->
    <property name="virtualBucketTimes">160</property><!-- 一个实际的数据库节点被映射为这么多虚拟节点，默认是160倍，也就是虚拟节点数是物理节点数的160倍 -->
    <!-- <property name="weightMapFile">weightMapFile</property> 节点的权重，没有指定权重的节点默认是1。以properties文件的格式填写，以从0开始到count-1的整数值也就是节点索引为key，以节点权重值为值。所有权重值必须是正整数，否则以1代替 -->
    <!-- <property name="bucketMapPath">/etc/mycat/bucketMapPath</property>
        用于测试时观察各物理节点与虚拟节点的分布情况，如果指定了这个属性，会把虚拟节点的murmur hash值与物理节点的映射按行输出到这个文件，没有默认值，如果不指定，就不会输出任何东西 -->
</function>
```

## 2-4-3-1. 常用的分片算法

### 2-4-3-1-1. 简单取模-PartitionByMod

- 可以用于分片列为整数类型的表
- 分片列 mod 分片基数(count参数的值，就是分成几片)
- 类全名：io.mycat.route.function.PartitionByMod

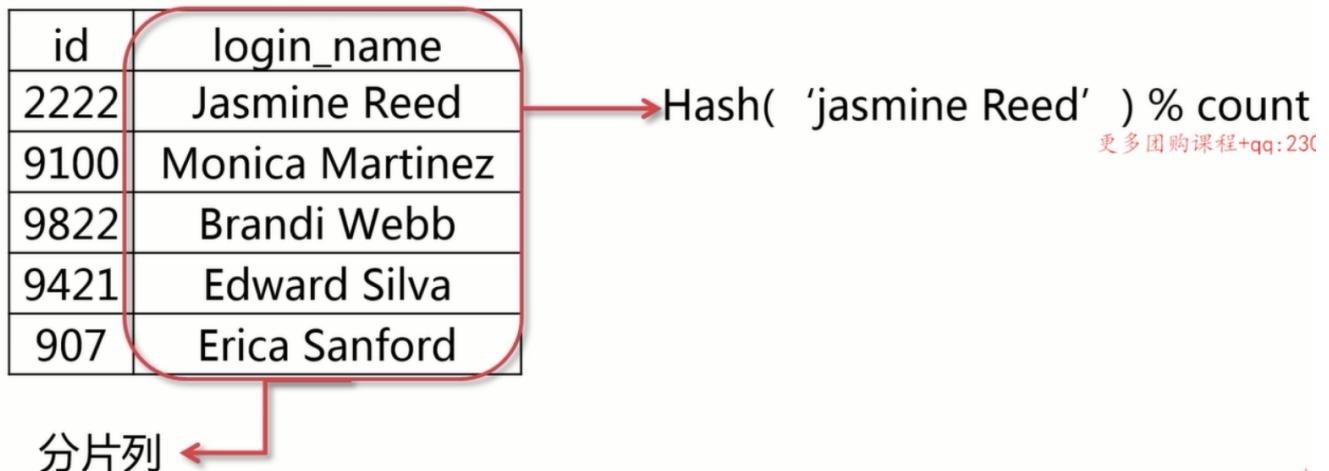


```
<tableRule name="customer_login">
    <rule>
        <columns>id</columns>
        <algorithm>mod_long</algorithm>
    </rule>
</tableRule>
```

```
<function name="mod-long" class="io.mycat.route.function.PartitionByMod">
    <!-- how many data nodes -->
    <property name="count">3</property><!-- 分片数-->
</function>
```

## 2-4-3-1-2. 哈希取模-PartitionByHashMod

- 可以用于多种数据类型如字符串、日期等
- hash(分片列) mod 分片基数



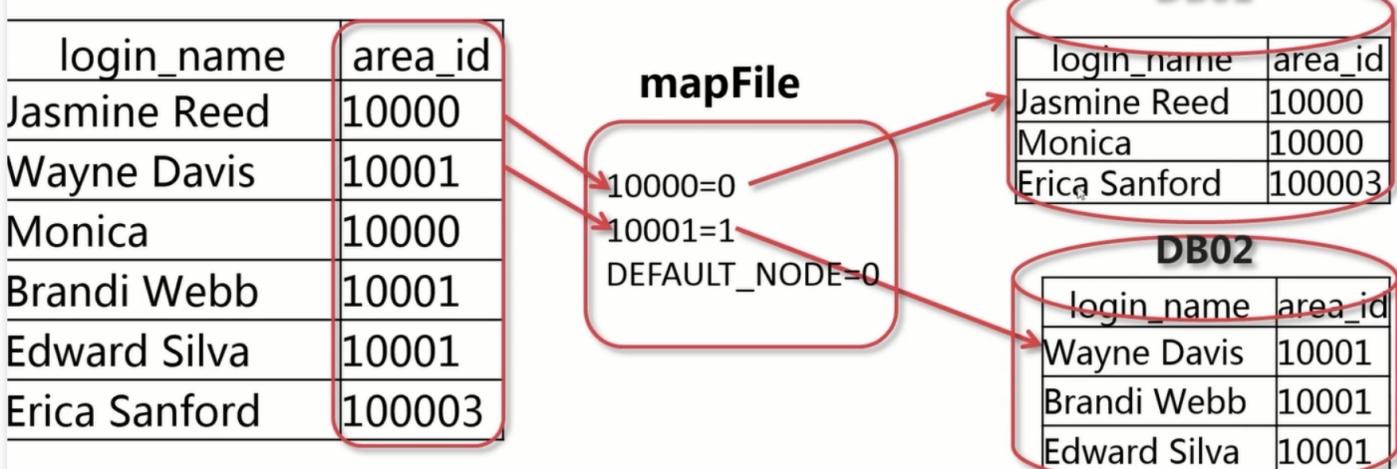
```
<tableRule name="customer_login">
    <rule>
        <columns>login_name</columns><!-- 分片字段可以为非整数 -->
        <algorithm>mod_long</algorithm>
    </rule>
</tableRule>

<function name="mod-long" class="io.mycat.route.function.PartitionByHashMod">
    <!-- how many data nodes -->
    <property name="count">3</property><!-- 分片数-->
</function>
```

## 2-4-3-1-3. 分片枚举-PartitionByFileMap

- 可以根据可能的枚举值指定数据存储的位置
- \$MYCAT/conf目录下增加MapFile配置枚举值同节点的对应关系
- 类全名：io.mycat.route.function.PartitionByFileMap

## 枚举分片-PartitionByFileMap



enter description here

```

<function name="xxx"
    class="io.mycat.route.function.PartitionByFileMap">
    <property name="mapFile">partition-hash-int.txt</property><!-- 配置文件-->
    <property name="type">0</property><!-- type表示枚举类型，0为Integer；非0为String-->
    <property name="defaultNode">0</property><!-- >=0:启用默认节点；<0不启用默认节点-->
</function>

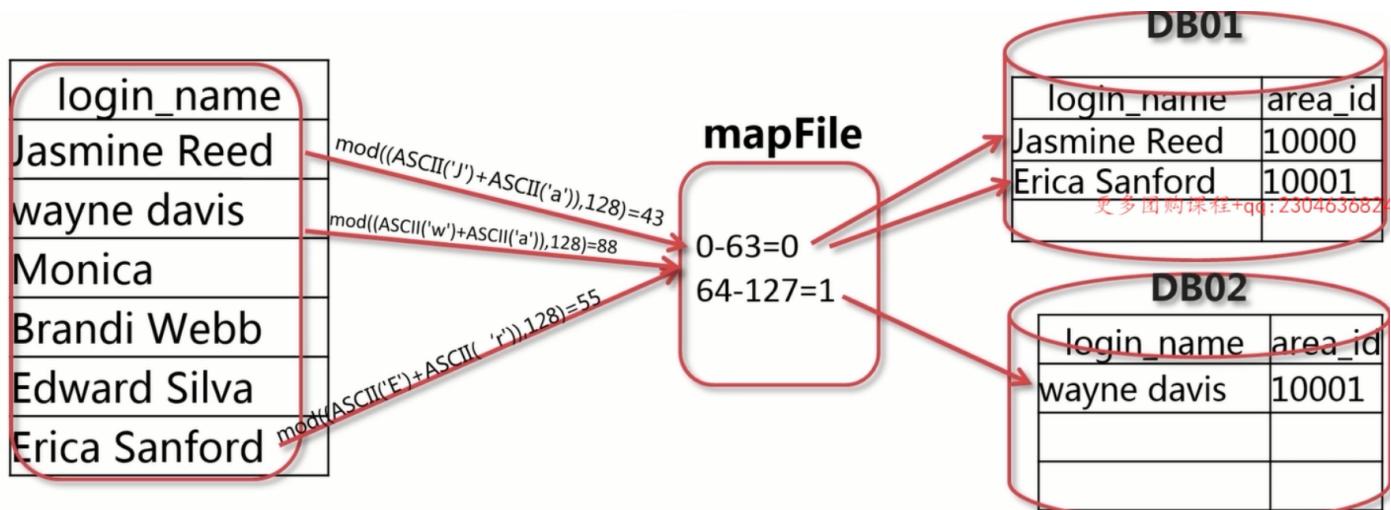
```

## 2-4-3-1-4. 字符串范围取模分片

- 可以根据指定字符串的前N个字符确定存储位置
- \$MYCAT/conf 增加MapFile配置取模范围同节点的对应关系
- 类全名：io.mycat.route.function.PartitionByPrefixPattern



enter description here



```
<function name="xxx"
    class="io.mycat.route.function.PartitionByPrefixPattern">
    <property name="patternValue">128</property>
    <property name="prefixLength">2</property>
    <property name="mapFile">prefix-partition-pattern.txt</property>
</function>
```

## 2-4-4. schema.xml

逻辑库和逻辑表相当于MySQL中的视图，而视图是不存储数据的

- 配置逻辑库及逻辑表
- 配置逻辑表所存储的数据节点
- 配置数据节点所对应的物理数据库服务器信息

```
<!-- <schema>定义逻辑库-->
<!-- sqlMaxLimit代表返回结果集行数, -1表示不限制-->
<!-- checkSQLSchema属性判断是否检查发给MyCAT的SQL是否包含库名-->
<schema name="TESTDB" checkSQLSchema="false" sqlMaxLimit="100"></schema>
```

schema.xml中会定义逻辑库和物理库的对应关系，但是不定义逻辑表和物理表的对应关系，所以逻辑表的名字必须和物理表的名字一致

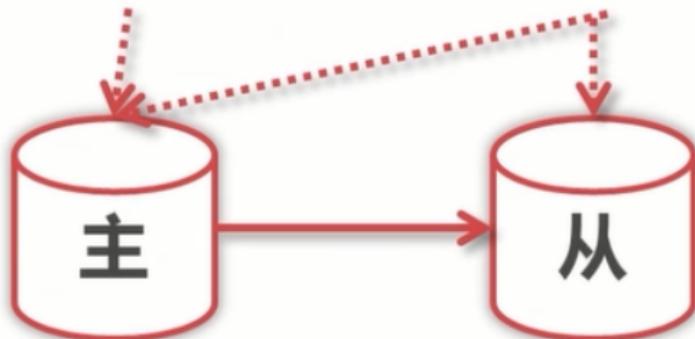
```
<!-- <table>定义逻辑表-->
<!-- dataNode表示物理节点, 顺序表示索引, 从0开始计-->
<!-- primaryKey表示主键-->
<!-- rule对应了逻辑表分片规则, 对应rule.xml中的<tableRule>-->
<table name="company" primaryKey="ID" type="global" dataNode="dn1, dn2, dn3"
rule="auto-sharding-long" />
```

```
<!-- <dataNode>定义逻辑表存储的物理数据库 -->
<!-- database属性定义物理数据库名称 -->
<!-- datahost属性定义分片所在的物理主机 -->
<dataNode name="dn1" dataHost="localhost1" database="db1" />
```

dataHost定义后端数据库主机信息，代表一组数据库服务器

## <dataHost>

<writeHost/> <ReadHost/>



</dataHost>

```
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
          writeType="0" dbType="mysql" dbDriver="native" switchType="1"
          slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <!-- can have multi write hosts -->
    <writeHost host="hostM1" url="localhost:3306" user="root"
               password="123456">
        <!-- can have multi read hosts -->
        <readHost host="hostS2" url="192.168.1.200:3306" user="root"
                  password="xxx" />
    </writeHost>
    <writeHost host="hostS1" url="localhost:3316" user="root"
               password="123456" />
    <!-- <writeHost host="hostM2" url="localhost:3316" user="root"
          password="123456"/> -->
</dataHost>
```

## 2-4-4-1. dataHost标签

- balance
  - 0: 不开启读写分离机制
  - 1: 全部的readHost与stand by writeHost参与select语句的负载均衡(多主多从)
  - 2: 全部的readHost与writeHost都参与select语句的负载均衡
  - 3: 全部的readHost参与select语句的负载均衡(一主多从)
- writeType
  - 0: 所有的写请求都由writeHost配置的第一台主机完成, 除非第一台主机宕机
  - 1: 写请求随机发送到writeHost配置的主机上(一般不建议随机发送, 除非是PXC方案集群)
- dbDriver
  - native: MySQL原生驱动
  - jdbc: 其他关系数据库驱动
- switchType(决定写数据库如何高可用切换)
  - 1: 自动切换
  - -1: 关闭自动切换功能

## 2-4-4-2. heartHost子标签

heartHost用于判断后端数据库能否正常提供服务

```
<!-- 返回数据库使用用户, 用来判断数据库能否正常提供服务-->
<heartbeat>select user()</heartbeat>
```

## 2-4-4-3. writeHost及readHost子标签

- 定义一组主从数据库, < readHost > 依赖于 < writeHost >
- user和password为后端数据库中的用户

```
<writeHost host="hostM1" url="localhost:3306" user="root"
           password="123456">
    <!-- can have multi read hosts -->
    <readHost host="hostS2" url="192.168.1.200:3306" user="root"
              password="xxx" />
</writeHost>
```

```

<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

<schema name="imooc_db" checkSQLschema="false" sqlMaxLimit="100">

    <table name="customer_login" primaryKey="customer_id" dataNode="logindb01,logindb02" rule="customer_login"/>
</schema>

<dataNode name="logindb01" dataHost="mysql0103" database="login_db01" />
<dataNode name="logindb02" dataHost="mysql0103" database="login_db02" />

<dataHost name="mysql0103" maxCon="1000" minCon="10" balance="1"
          writeType="0" dbType="mysql" dbDriver="native" switchType="1">
    <heartbeat>select user()</heartbeat>
    <writeHost host="192.168.1.3" url="192.168.1.3:3306" user="im_mycat" password="123456">
        <readHost host="192.168.1.4" url="192.168.1.4:3306" user="im_user" password="123456" />
    </writeHost>
    <writeHost host="192.168.1.4" url="192.168.1.4:3306" user="im_user" password="123456" />
</dataHost>

</mycat:schema>

```

enter description here

## 2-4-4-4. 小结

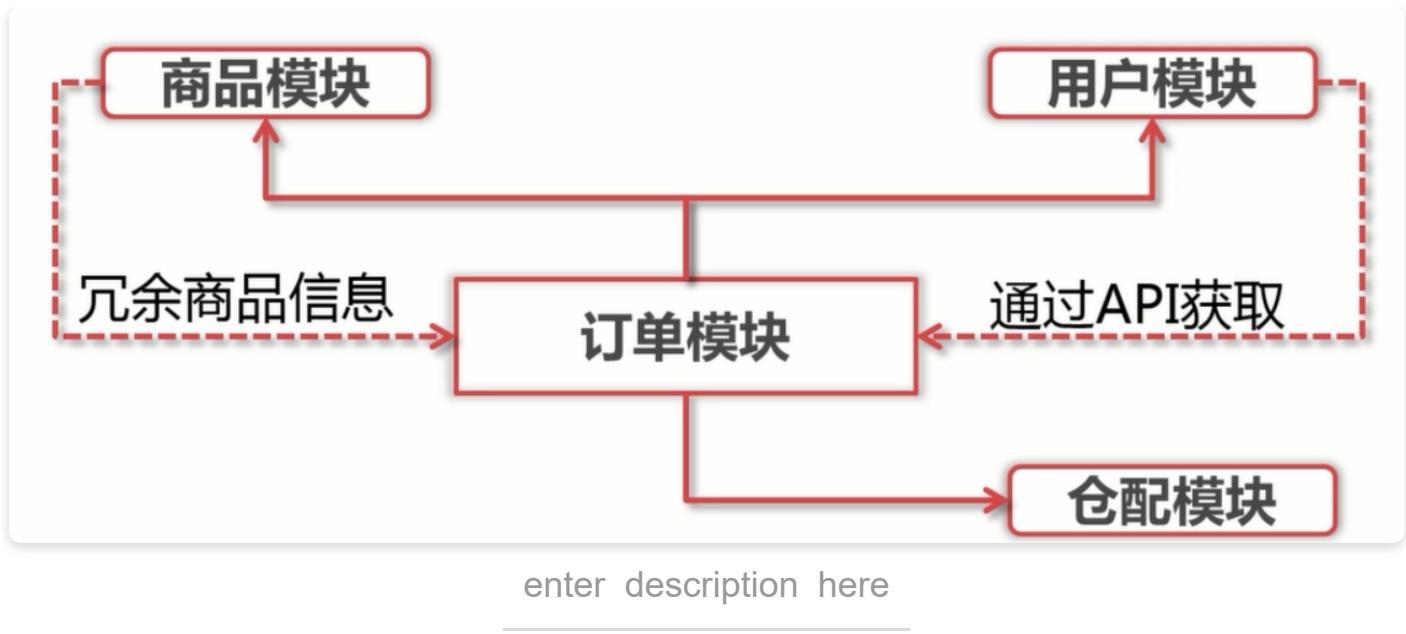
- 分片节点：大数据表进行数据切分后，每个表分片所在的数据库就是分片节点，狭义的理解可以认为一个DB实例就是一个节点
- 分片主机：数据切分后，每个分片节点(dataNode)不一定都会独占一个机器，同一个机器上面可以有多个分片数据库，这样一个或多个分片节点所在的机器就是节点主机。为了规避单节点主机的并发数限制，尽量将读写压力高的分片节点均衡的放在不同的节点主机(读写分离 )

## 2-5. 垂直分库

垂直分库的步骤

1. 收集分析业务模块间的关系
2. 复制数据库到其他实例
3. 配置MyCAT垂直分库
4. 通过MyCAT访问DB
5. 删除原库中已迁移的表

### 2-5-1. 收集分析业务模块间的关系



## 2-5-2. 复制数据库到其他实例

1. 备份原数据库并记录相关事务点
2. 在原数据库中建立复制用户
3. 在新的实例上恢复备份数据库
4. 在新实例上配置复制链路
5. 在新实例上启动复制

如何改变复制链路的数据库名？

当我们主机的数据库名为imooc\_db，我们希望从机的数据库名为order\_db时，我们需要对复制链路进行过滤

```
change replication filter replicate_rewrite_db=((imooc_db,order_db));
```

## 2-5-3. 配置MyCAT垂直分库

前面我们已经配置复制数据库到其他主机了(imooc\_db已经被垂直拆分为三个库)，接下来配置MyCAT

主机名	IP	角色	数据库
		MyCAT	imooc_db
node1	192.168.1.2	MYSQL	
node2	192.168.1.3	MYSQL	order_db
node3	192.168.1.4	MYSQL	product_db
node4	192.168.1.5	MYSQL	customer_db

- 使用schema.xml配置逻辑库
- 使用server.xml配置系统变量及用户权限
- 由于没有用到水平分配因此不需要配置rule.xml

```

<schema name="imooc_db" checkSQLschema="false" sqlMaxLimit="100">
    <!-- 配置逻辑表 -->
    <table name="order_master" primaryKey="order_id" dataNode="ordb" />
    <table name="order_detail" primaryKey="order_detail_id"
dataNode="ordb" />
    ...
</schema>

<dataNode name="ordb" dataHost="mysql110103" database="order_db" />
<dataNode name="prodb" dataHost="mysql110104" database="product_db" />
<dataNode name="custdb" dataHost="mysql110105" database="customer_db" />

<!-- dataHost可以是一组主机，因为一个数据库可以有多个实例，而一组主机里面可以有写机
也可以有读机，其中读机依赖于写机，因为需要进行主从复制，而读机为从机-->
<dataHost name="mysql110103" maxCon="1000" minCon="10" balance="0"
           writeType="0" dbType="mysql" dbDriver="native" switchType="1"
slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <!-- can have multi write hosts 这里的密码是连接mysql的-->
    <writeHost host="192.168.1.3" url="192.168.1.3:3306" user="root"
               password="123456">
</dataHost>

<dataHost name="mysql110104" maxCon="1000" minCon="10" balance="0"
           writeType="0" dbType="mysql" dbDriver="native" switchType="1"
slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <!-- can have multi write hosts -->
    <writeHost host="192.168.1.4" url="192.168.1.4:3306" user="root"
               password="123456">
</dataHost>

<dataHost name="mysql110105" maxCon="1000" minCon="10" balance="0"
           writeType="0" dbType="mysql" dbDriver="native" switchType="1"
slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <!-- can have multi write hosts -->

```

```
<writeHost host="192.168.1.5" url="192.168.1.5:3306" user="root"  
password="123456">  
</dataHost>  
<!-- 这里的密码是后端连接mycat的, 数据库是逻辑数据库-->  
<user name="root" defaultAccount="true">  
    <property name="password">123456</property>  
    <property name="schemas">imooc_db</property>  
</user>
```

接下来直接 mycat start 再连接即可 mysql -h xxx.xx..xx:8806

## 2-5-4. 跨分片查询

多个表如果不在同个节点中，查询需要关联时就会产生错误。

解决

运用MyCAT在多个节点维护全局表

```
# 备份需要经常关联的表  
mysqldump -uroot -p order_db region_info > region_info;  
# 拷贝到其他节点  
scp region_info root@(ip地址)  
# 编辑MyCAT文件 配置为全局表  
<table name="region_info" primaryKey="order_id" dataNode="ordb, prodb, custdb"  
type="global" />
```