# MyCat

## 1.Mycat简介

### 1.1什么是mycat?

Mycat 是数据库中间件，连接在java应用程序和数据库之间。

### 1.2为什么要用mycat？

1. java程序与数据库低耦合
2. 降低高访问高并发对数据库的压力
3. 保证读写请求数据一致

### 1.3 Mycat的官网？

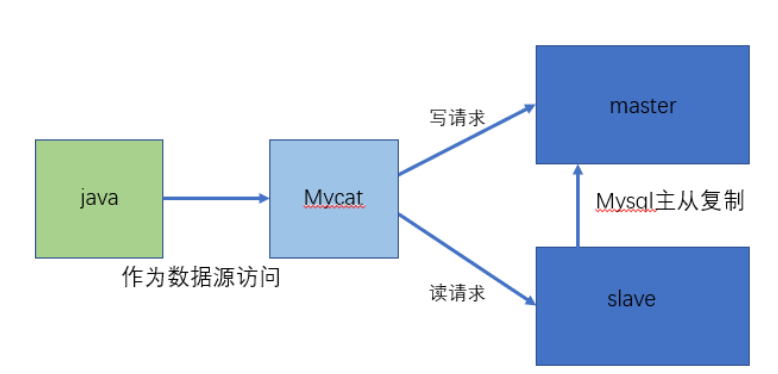
<http://www.mycat.org.cn/>

### 1.4 软件下载地址？

<http://dl.mycat.org.cn/>

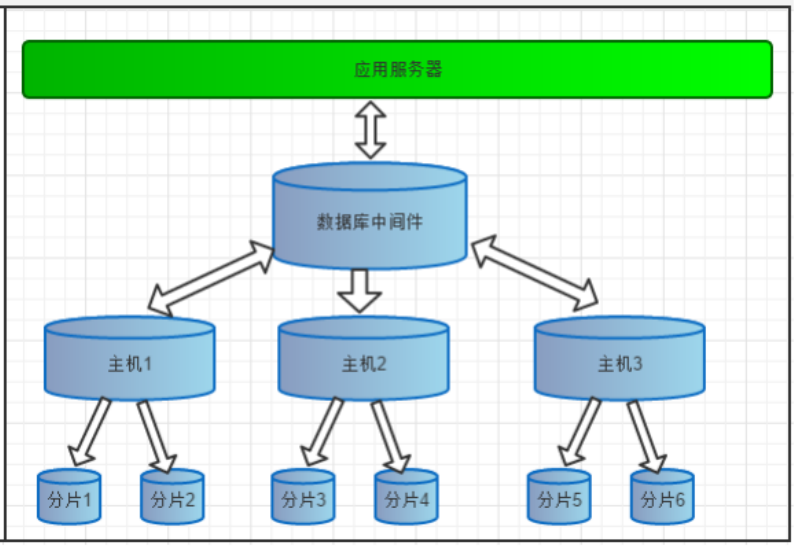
### 1.5 mycat能干什么？

1. 读写分离

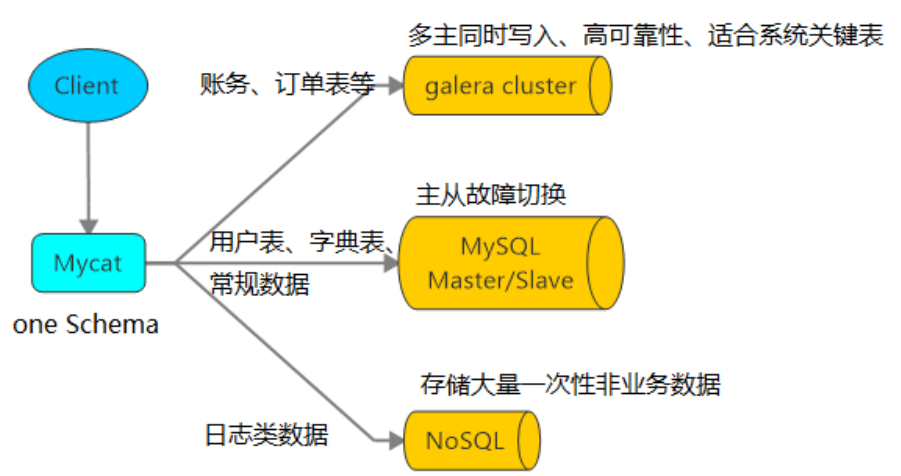


1. 数据分片

垂直拆分（分库）、水平拆分（分表）、垂直+水平拆分（分库分表）

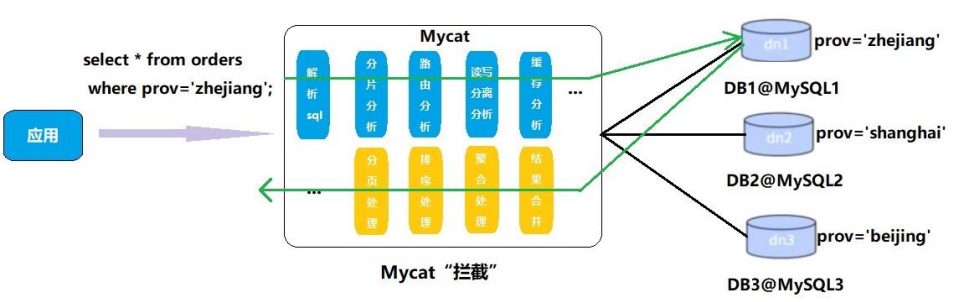


1. 数据源整合



### 1.6 Mycat的原理？

Mycat 的原理中最重要的一个动词是“拦截”，它拦截了用户发送过来的 SQL 语句，首先对 SQL 语句做了一些特定的分析：如分片分析、路由分析、读写分离分析、缓存分析等，然后将此 SQL 发 往后端的真实数据库，并将返回的结果做适当的处理，最终再返回给用户。



## 2.安装启动

### 2.1安装

1. 获取安装包，安装包放在 /opt 目录下



1. 解压安装包

tar –zxvf Mycat-server-1.6.7.1-release-20190627191042-linux.tar.gz

1. 解压后的文件 拷贝到 /usr/local/目录下

cp –r mycat /usr/local/

1. 进入到 mycat/conf/目录

三个配置文件

* schema.xml:定义逻辑库，表、分片节点等内容
* rule.xml:定义分片规则
* server.xml:定义用户以及系统相关变量。如端口等等

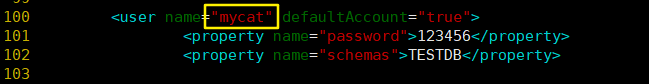
### 2.2启动

1. 在mycat/conf/ 修改配置文件server.xml

vim server.xml

修改用户名，默认是root， 改成name=”mycat”，与Mysql做区分。

TESTDB为mycat的逻辑库，对应了对个数据库



2、修改配置文件 schema.xml

vim schema.xml

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">  <mycat:schema xmlns:mycat="http://io.mycat/">  <!-- dataNode="dn1" 默认的数据节点 -->  <schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">  </schema>  <dataNode name="dn1" dataHost="host1" database="testdb" />  <dataHost name="host1" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <!-- 心跳检测，有数据返回说明正常 -->  <heartbeat>select user()</heartbeat>  <!--写主机配置-->  <writeHost host="hostM1" url="192.168.64.130:3306" user="root" password="123456">  <!-- can have multi read hosts -->  <!—读机配置-->  <readHost host="hostS1" url="192.168.64.131:3306" user="root" password="密码xxx " />  </writeHost>  </dataHost>  </mycat:schema> |

3、验证数据访问数据库的情况

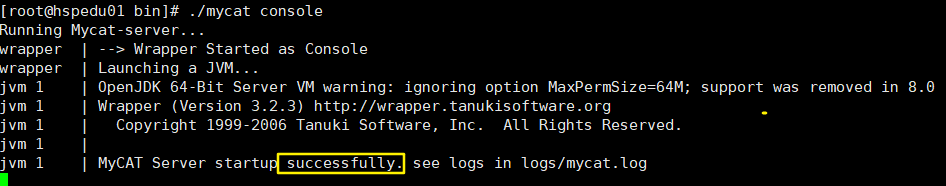
Mycat 作为数据库中间件要和数据库部署在不同机器上，所以要验证远程访问情况。

|  |
| --- |
| mysql -uroot –p123456 -h192.168.64.130 -P3306  mysql -uroot –p123456 -h192.168.64.131 -P3306  #如远程访问报错，请建对应用户  https://www.cnblogs.com/withLevi/p/16005877.html |

4、启动程序

* 控制台启动 ：去 mycat/bin 目录下执行 ./mycat console
* 后台启动 ：去 mycat/bin 目录下 ./mycat start

为了能第一时间看到启动日志，方便定位问题，我们选择控制台启动。



### 登录

1. 登录mycat后台管理窗口

此登录方式用于管理维护 Mycat

mysql -umycat -p--default\_auth=mysql\_native\_password -P 9066 -h 192.168.140.128

1. 登录mycat数据管理窗口

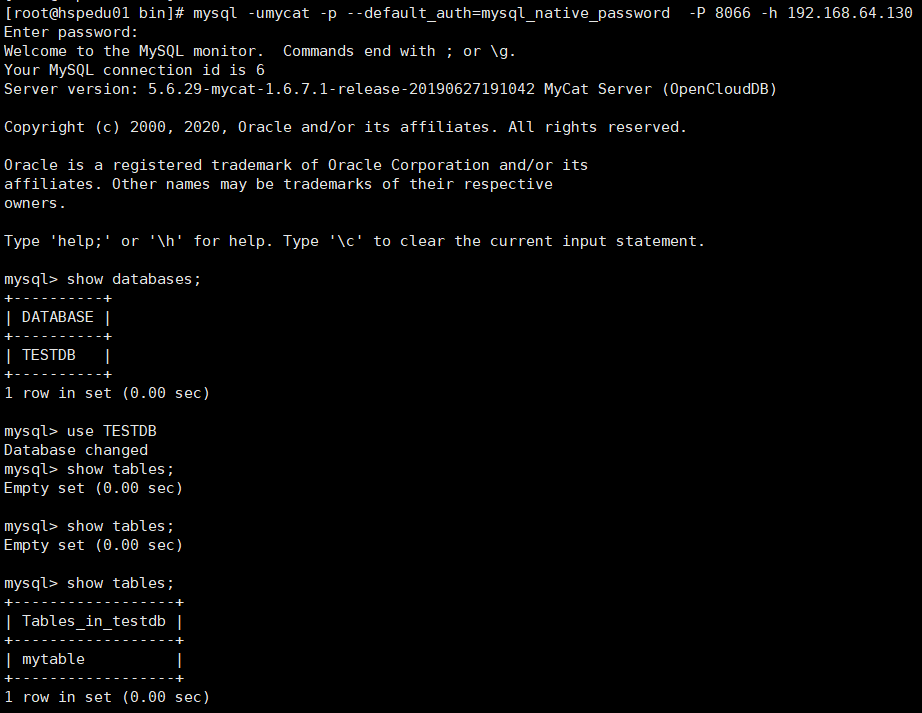
此登录方式用于通过mycat 查询数据，我们选择这种方式访问mycat

mysql -umycat -p --default\_auth=mysql\_native\_password -P 8066 -h 192.168.64.130

再输入密码：123456

用此语句会报密码错误

// mysql -umycat -p123456 -P 8066 -h 192.168.64.130



## 3．搭建读写分离

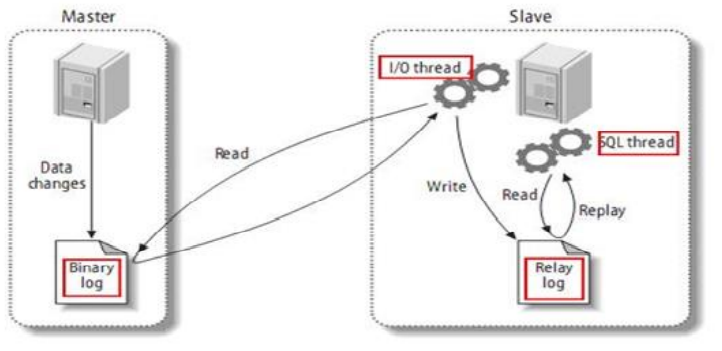
我们通过 Mycat 和 MySQL 的主从复制配合搭建数据库的读写分离，实现 MySQL 的高可用性。

我们将搭建：一主一从、双主双从两种读写分离模式。

### 3.1搭建一主一从

#### 3.1.1搭建 MySQL 数据库主从复制

##### 3.1.1.1 MySQL 主从复制原理

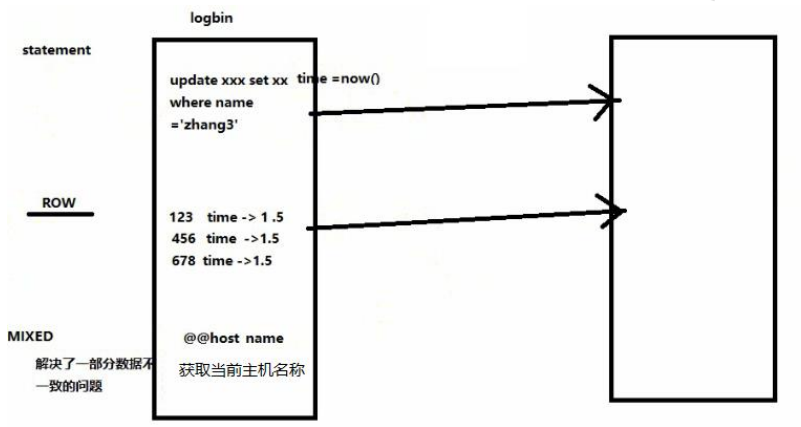


* 主从同步过程中主服务器有一个工作线程I/O dump thread，从服务器有两个工作线程I/O thread和SQL thread。
* 主库把外界接收的SQL（update、insert、delete）请求记录到自己的binlog日志中，
* 从库的I/O thread去请求主库的binlog日志，并将binlog日志写到中继日志中
* 从库创建一个SQL线程，从中继日志中读取内容，写到自己的从库中

##### 3.1.1.2主机配置(192.168.64.130)

|  |
| --- |
| 修改配置文件：vim /etc/my.cnf  #主服务器唯一ID  server-id=130  #启用二进制日志  log-bin=/usr/local/mysql/data/mysql-bin  # 设置不要复制的数据库(可设置多个)  binlog-ignore-db=mysql  binlog-ignore-db=information\_schema  #设置需要复制的数据库  binlog-do-db=需要复制的主数据库名字  #设置logbin格式  binlog\_format=STATEMENT |

##### 3.1.1.3 binlog 日志三种格式



* statement格式，将所有写操作的日志复制到从机，当更新语句中有now()时，会导致时间不一致
* row 格式，记录每一行的改变。占用空间大，假如行太多，执行效率会很低
* mixed 格式，它是前面两种格式的混合。意思是MySQL自己会判断这条SQL语句是否会引起主备不一致，是的话就会使用row，否则就用statement格式 。缺点就是识别不了@@host name

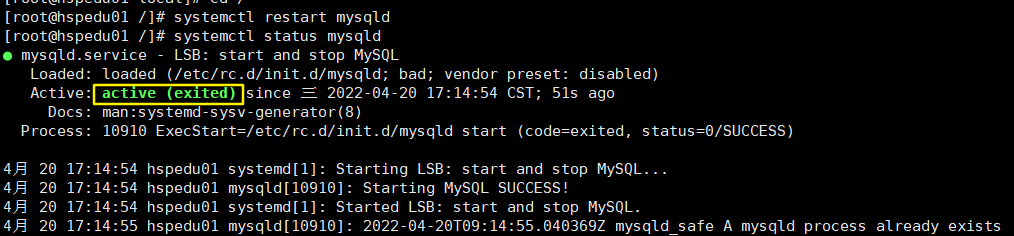
##### 3.1.1.4从机配置(192.168.64.131)

|  |
| --- |
| 修改配置文件：vim /etc/my.cnf  #从服务器唯一ID  server-id=131  #启用中继日志  relay-log=mysql-relay |

##### 3.1.1.5主机、从机重启 MySQL 服务，让配置文件生效

systemctl restart mysqld 重启服务

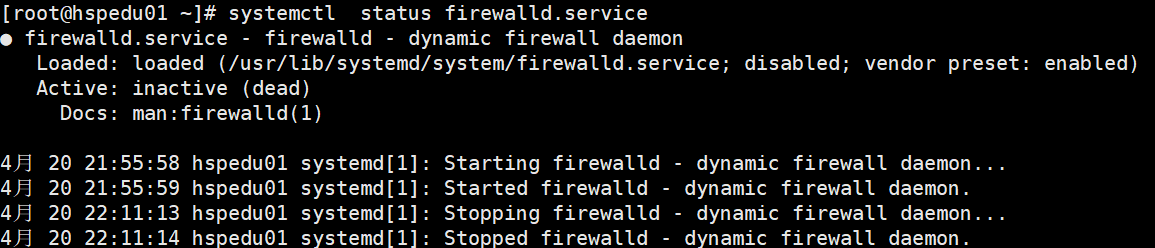
systemctl status mysqld 查询状态



##### 3.1.1.6主机从机都关闭防火墙

systemctl stop firewalld.service关闭防火墙；

systemctl status firewalld.service查看防火墙的状态；



##### 3.1.1.7在主机上建立帐户并授权 slave

|  |
| --- |
| #创建用户  create user 'slave'@'%' identified by'123456';    #设置用户权限  grant replication slave,replication client on \*.\* to 'slave'@'%';    刷新权限  flush privileges;    #查看用户权限  show grants for'slave'@'%';    #查询master的状态  show master status;    #记录下File和Position的值  #执行完此步骤后不要再操作主服务器MySQL，防止主服务器状态值变化 |

##### 3.1.1.8在从机salve上配置需要复制的主机

|  |
| --- |
| #停止复制    #重置master    #复制主机的命令 ，登录从机执行命令  CHANGE MASTER TO MASTER\_HOST='192.168.64.130 ',//主机的IP地址  MASTER\_USER='slave', //用户  MASTER\_PASSWORD='123456', //密码 123456  MASTER\_LOG\_FILE='mysql-bin.000011 ',MASTER\_LOG\_POS=840;//刚查到的主机的值    #启动从服务器复制功能  start slave ;  #查看从服务器状态  show slave status\G;    #下面两个参数都是Yes，则说明主从配置成功！  # Slave\_IO\_Running: Yes  # Slave\_SQL\_Running: Yes |

##### 3.1.1.9主机新建库、新建表、insert 记录，从机复制

|  |
| --- |
| 从机： |

##### 3.1.1.10、如何停止从服务复制功能

|  |
| --- |
| stop slave; |

##### 3.1.1.11如何重新配置主从

|  |
| --- |
| stop slave;  reset master; |

#### 3.1.2修改Mycat的配置文件 schema.xml

修改mycat的schema.xml 中<dataHost> balance属性，通过此属性配置读写分离的类型

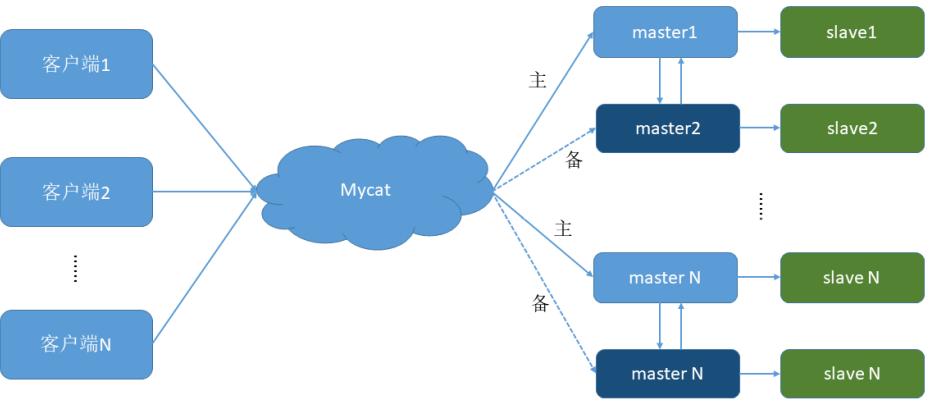
|  |
| --- |
| 负载均衡类型，目前的取值有4 种：  （1）balance="0", 不开启读写分离机制，所有读操作都发送到当前可用的 writeHost 上。  （2）balance="1"，全部的 readHost 与 stand by writeHost 参与 select 语句的负载均衡，简单的说，当双主双从 模式(M1->S1，M2->S2，并且 M1 与 M2 互为主备)，正常情况下，M2,S1,S2 都参与 select 语句的负载均衡。  （3）balance="2"，所有读操作都随机的在 writeHost、readhost 上分发。  （4）balance="3"，所有读请求随机的分发到 readhost 执行，writerHost 不负担读压力（单读单写）    为了演示效果，blance设置成2，一般设置成1或者3 |

#### 3.1.3启动mycat，验证读写分离

|  |
| --- |
| 1. 在写主机数据库表mytable插入带系统变量的数据insert into mytbl values (1,@@hostname); 造成主从主机数据不一致     在从机查询：     1. 在Mycat里查询mytable表,可以看到查询语句在主从两个主机间切换 |
|  |

### 3.2搭建双主双从

一个主机 m1 用于处理所有写请求，它的从机 s1 和另一台主机 m2 还有它的从机 s2 负责所有读请求。当 m1 主机宕机后，m2 主机负责写请求，m1、m2 互为备机（例如：当m1宕机后，m2成了主机；m1重启后，m1成了m2的备机）。架构图如下



|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 角色 | IP地址 | 机器名 |
| 1 | Matser1 | 192.168.64.128 | host128 |
| 2 | Slave1 | 192.168.64.129 | host129 |
| 3 | Matser2 | 192.168.64.130 | host130 |
| 4 | Slave2 | 192.168.64.131 | host131 |

#### 3.2.1搭建 MySQL 数据库主从复制（双主双从）

##### 3.2.1.1双主机配置

Master1（host128）配置

|  |
| --- |
| 修改配置文件：vim /etc/my.cnf  #主服务器唯一ID  server-id=128  #启用二进制日志  log-bin=mysql-bin  # 设置不要复制的数据库(可设置多个)  binlog-ignore-db=mysql  binlog-ignore-db=information\_schema  #设置需要复制的数据库  binlog-do-db=需要复制的主数据库名字  #设置logbin格式  binlog\_format=STATEMENT  # 在作为从数据库的时候，有写入操作也要更新二进制日志文件  **log-slave-updates**  #表示自增长字段每次递增的量，指自增字段的起始值，其默认值是1，取值范围是1 .. 65535  auto-increment-increment=2  # 表示自增长字段从哪个数开始，指字段一次递增多少，他的取值范围是1 .. 65535 auto-increment-offset=1 |

Master2（host130）配置

|  |
| --- |
| 修改配置文件：vim /etc/my.cnf  #主服务器唯一ID  server-id=130  #启用二进制日志  log-bin=mysql-bin  # 设置不要复制的数据库(可设置多个)  binlog-ignore-db=mysql  binlog-ignore-db=information\_schema  #设置需要复制的数据库  binlog-do-db=需要复制的主数据库名字  #设置logbin格式  binlog\_format=STATEMENT  # 在作为从数据库的时候，有写入操作也要更新二进制日志文件  **log-slave-updates**  #表示自增长字段每次递增的量，指自增字段的起始值，其默认值是1，取值范围是1 .. 65535  auto-increment-increment=2  # 表示自增长字段从哪个数开始，指字段一次递增多少，他的取值范围是1 .. 65535  auto-increment-offset=2 |

##### 3.2.1.2双从机配置

Slave1（host129）配置

|  |
| --- |
| 修改配置文件：vim /etc/my.cnf  #从服务器唯一ID  server-id=131  #启用中继日志  relay-log=mysql-relay |

Slave2（host131）配置

|  |
| --- |
| 修改配置文件：vim /etc/my.cnf  #从服务器唯一ID  server-id=133  #启用中继日志  relay-log=mysql-relay |

##### 3.2.1.3双主机、双从机重启 mysql 服务

#重启mysql

systemctl restart mysqld

#查看mysql状态

systemctl status mysqld

|  |
| --- |
|  |

##### 3.2.1.4主机从机都关闭防火墙

#关闭防火墙

systemctl stop firewalld.service

#查看防火墙的状态

systemctl status firewalld.service

|  |
| --- |
|  |

##### 3.2.1.5在两台主机上建立帐户并授权 slave

|  |
| --- |
| #主机上建立账户并授权 slave  #创建用户  create user 'slave'@'%' identified by'123456';  #设置用户权限  grant replication slave,replication client on \*.\* to 'slave'@'%';  刷新权限  flush privileges;  #查看用户权限  show grants for'slave'@'%';      #查询master1状态  show master status;    #查询master2状态  show master status;    分别记录下File和Position的值  #执行完此步骤后不要再操作主服务器MYSQL，防止主服务器状态值变化 |

##### 3.2.1.6在从机上配置需要复制的主机

|  |
| --- |
| #复制主机的命令  CHANGE MASTER TO MASTER\_HOST='主机的IP地址',  MASTER\_USER='slave',  MASTER\_PASSWORD='123456',  MASTER\_LOG\_FILE='mysql-bin.具体数字',MASTER\_LOG\_POS=具体值;  #Slava1 复制 Master1  CHANGE MASTER TO MASTER\_HOST=' 192.168.64.128',  MASTER\_USER='slave',  MASTER\_PASSWORD='123456',  MASTER\_LOG\_FILE='mysql-bin.000016',MASTER\_LOG\_POS=156;  #Slava2 复制 Master2  CHANGE MASTER TO MASTER\_HOST='192.168.64.130 ',  MASTER\_USER='slave',  MASTER\_PASSWORD='123456',  MASTER\_LOG\_FILE='mysql-bin.000004',MASTER\_LOG\_POS=156;  #启动两台从服务器复制功能  start slave;  #查看从服务器状态  show slave status\G;      #下面两个参数都是Yes，则说明主从配置成功！  # Slave\_IO\_Running: Yes  # Slave\_SQL\_Running: Yes |

##### 3.2.1.7 两个主机互相复制

|  |
| --- |
| #Master2 复制 Master1  CHANGE MASTER TO MASTER\_HOST=' 192.168.64.128',  MASTER\_USER='slave',  MASTER\_PASSWORD='123456',  MASTER\_LOG\_FILE='mysql-bin.000016',MASTER\_LOG\_POS=156;  #Master1 复制 Master2  CHANGE MASTER TO MASTER\_HOST='192.168.64.130 ',  MASTER\_USER='slave',  MASTER\_PASSWORD='123456',  MASTER\_LOG\_FILE='mysql-bin.000004',MASTER\_LOG\_POS=156;  #启动两台主服务器复制功能  start slave;  #查看从服务器状态  show slave status\G; |

#### 3.2.2 修改Mycat 的配置文件 schema.xml

修改mycat的schema.xml 中<dataHost> balance属性，通过此属性配置读写分离的类型

|  |
| --- |
| 负载均衡类型，目前的取值有4 种：  （1）balance="0", 不开启读写分离机制，所有读操作都发送到当前可用的 writeHost 上。  （2）balance="1"，全部的 readHost 与 stand by writeHost 参与 select 语句的负载均衡，简单的说，当双主双从 模式(M1->S1，M2->S2，并且 M1 与 M2 互为主备)，正常情况下，M2,S1,S2 都参与 select 语句的负载均衡。  （3）balance="2"，所有读操作都随机的在 writeHost、readhost 上分发。  （4）balance="3"，所有读请求随机的分发到 readhost 执行，writerHost 不负担读压力（单读单写）  为了双主双从读写分离balance设置为1    #balance="1": 全部的readHost与stand by writeHost参与select语句的负载均衡。  #writeType="0": 所有写操作发送到配置的第一个writeHost，第一个挂了切到还生存的第二个  #writeType="1"，所有写操作都随机的发送到配置的 writeHost，1.5 以后废弃不推荐  #writeHost ，重新启动后以切换后的为准，切换记录在配置文件中:dnindex.properties 。  #switchType="1": 1 默认值，自动切换。 -1 表示不自动切换 2 基于 MySQL 主从同步的状态决定是否切换。 |

#### 3.2.3启动Mycat验证读写分离

|  |
| --- |
| #启动Mycat  去 mycat/bin 目录下执行 ./mycat console  #在写主机Master1数据库表mytbl中插入带系统变量数据，造成主从数据不一致  INSERT INTO mytable VALUES(3,@@hostname);  #在Mycat里查询mytable表,可以看到查询语句在Master2、Slava1、Slava2主从三个主机间切换 |

模拟一台M1宕机

|  |
| --- |
| #停止数据库Master1  systemctl stop mysqld  systemctl status mysqld  #在Mycat里插入数据依然成功，Master2自动切换为写主机  INSERT INTO mytbl VALUES(3,@@hostname);  #启动数据库Master1  systemctl stop mysqld  systemctl status mysqld  #在Mycat里查询mytbl表,可以看到查询语句在Master1、Slava1、Slava2主从三个主机间切换  Master1、Master2 互做备机，负责写的主机宕机，备机切换负责写操作，保证数据库读写分离高可用性。 |

## 4.Mycat的安全设置

### 4.1 权限配置

#### 4.1.1 user 标签权限控制

目前 Mycat 对于中间件的连接控制并没有做太复杂的控制，目前只做了中间件逻辑库级别的读写权限控制。是通过 server.xml 的 user 标签进行配置。

|  |
| --- |
| #server.xml配置文件user部分 |

配置说明：

|  |  |
| --- | --- |
| 标签属性 | 说明 |
| name | 应用连接中间件逻辑库的用户名 |
| password | 该用户对应的密码 |
| TESTDB | 应用当前连接的逻辑库中所对应的逻辑表。schemas 中可以配置一个或多个 |
| readOnly | 应用连接中间件逻辑库所具有的权限。true 为只读，false 为读写都有，默认为 false |

#### 4.1.2 privileges 标签权限控制

* 在 user 标签下的 privileges 标签可以对逻辑库（schema）、表（table）进行精细化的 DML 权限控制。
* privileges 标签下的 check 属性，如为 true 开启权限检查，为 false 不开启，默认为 false。
* 由于 Mycat 一个用户的 schemas 属性可配置多个逻辑库（schema） ，所以 privileges 的下级节点 schema 节点同样可配置多个，对多库多表进行细粒度的 DML 权限控制。

|  |
| --- |
| #server.xml配置文件privileges部分  #配置orders表没有增删改查权限 |

配置说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DML 权限 | 增加（insert） | 更新（update） | 查询（select） | 删除（select） |
| 0000 | 禁止 | 禁止 | 禁止 | 禁止 |
| 0010 | 禁止 | 禁止 | 可以 | 禁止 |
| 1110 | 可以 | 禁止 | 禁止 | 禁止 |
| 1111 | 可以 | 可以 | 可以 | 可以 |

#### 4.1.3 sql拦截

* firewall 标签用来定义防火墙；
* firewall 下 whitehost 标签用来定义 IP 白名单 ，
* blacklist 用来定义 SQL 黑名单。

##### 4.1.3.1 白名单

可以通过设置白名单，实现某主机某用户可以访问 Mycat，而其他主机用户禁止访问。

|  |
| --- |
| #设置白名单  #server.xml配置文件firewall标签  #例如：配置只有192.168.140.128主机可以通过mycat用户访问    #重启Mycat后，192.168.140.128主机使用mycat用户访问 mysql -umycat -p123456 -h 192.168.140.128 -P 8066 |

##### 4.1.3.2 黑名单

可以通过设置黑名单，实现 Mycat 对具体 SQL 操作的拦截，如增删改查等操作的拦截。

|  |
| --- |
| #设置黑名单  #server.xml配置文件firewall标签  #配置禁止mycat用户进行删除操作    #重启Mycat后，192.168.140.128主机使用mycat用户访问 mysql -umycat -p123456 -h 192.168.140.128 -P 8066 |

可以设置的黑名单 SQL 拦截功能列表

|  |  |  |
| --- | --- | --- |
| 配置项 | 缺省值 | 描述 |
| selelctAllow | true | 是否允许执行 SELECT 语句 |
| deleteAllow | true | 是否允许执行 DELETE 语句 |
| updateAllow | true | 是否允许执行 UPDATE 语句 |
| insertAllow | true | 是否允许执行 INSERT 语句 |
| createTableAllow | true | 是否允许创建表 |
| setAllow | true | 是否允许使用 SET 语法 |
| alterTableAllow | true | 是否允许执行 Alter Table 语句 |
| dropTableAllow | true | 是否允许修改表 |
| commitAllow | true | 是否允许执行 commit 操作 |
| rollbackAllow | true | 是否允许执行 roll back 操作 |

参考：https://www.bilibili.com/video/BV1WJ411x7bD?spm\_id\_from=333.1007.top\_right\_bar\_window\_custom\_collection.content.click