Centos7安装mysql8.0.34

# Centos7安装

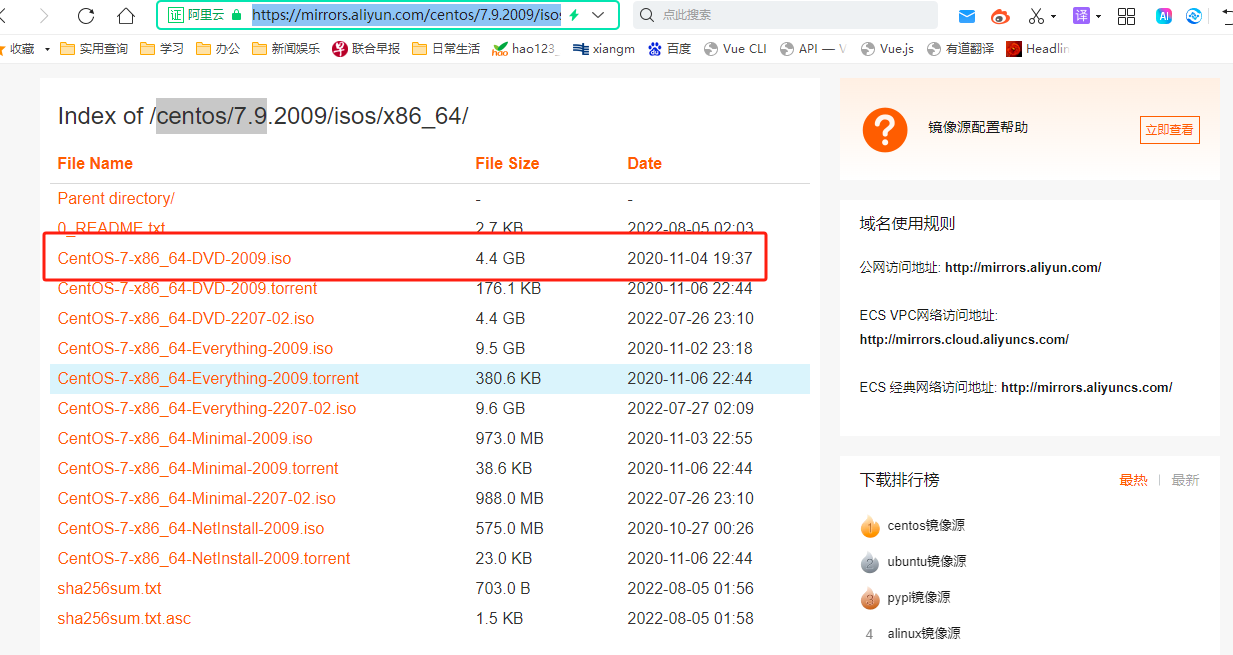
我的Centos7版本为**centos/7.9.2009**

下载地址

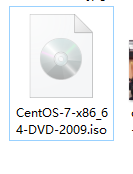
<https://developer.aliyun.com/mirror/>

具体地址

<https://mirrors.aliyun.com/centos/7.9.2009/isos/x86_64/?spm=a2c6h.25603864.0.0.4487f5adVOxgf4>



下载完成后



然后虚拟机安装就可以

# 下载安装文件

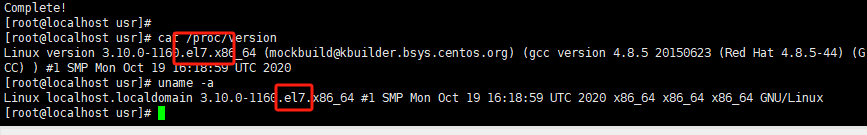
参考：https://blog.csdn.net/lht631935612/article/details/132020522

## 2.1查看centos版本

cat /proc/version

或者

uname -a



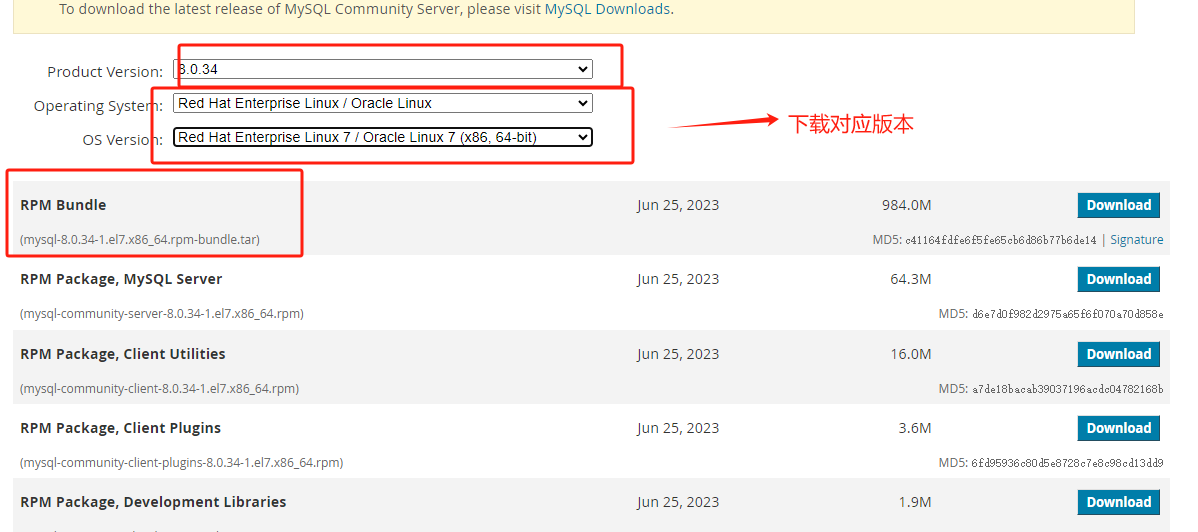
我机器是el7版本，安装是要下载el7版本

## 2.2根据contos版本下载

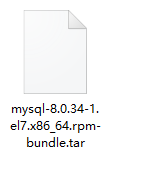
下载地址：<https://dev.mysql.com/downloads/mysql/>  
下载时，要注意自己系统的版本，我的Linux版本是el7，要下载对应的版本。

具体下载地址是

https://downloads.mysql.com/archives/community/

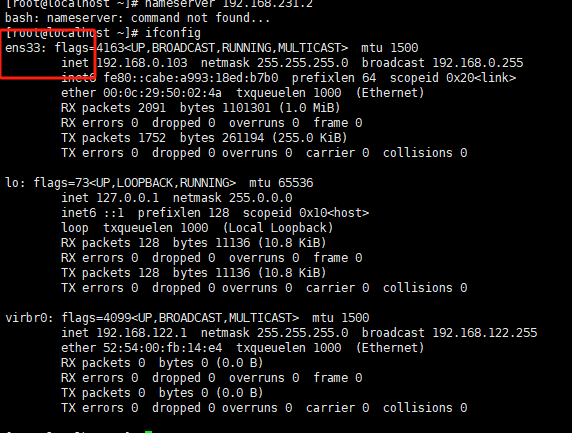


下载后如下：



## 2.3 设置centos7网络

修改文件 /etc/sysconfig/network-scripts/ifcfg-ens33 文件，如下所示。修改的配置文件 ifcfg-ens33 要对应系统上正在使用的网卡，可以使用 ifconfig 查看下当前的 IP 对应的是哪个网卡。



vim /etc/sysconfig/network-scripts/ifcfg-ens33

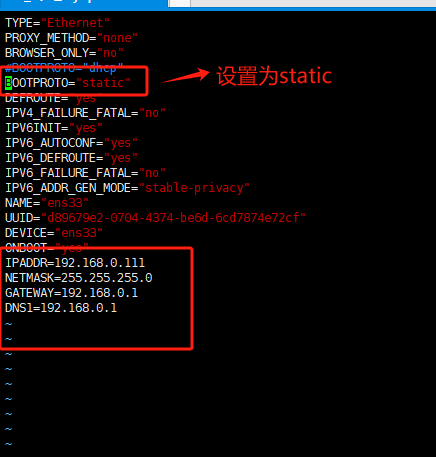
BOOTPROTO="static"

IPADDR=192.168.0.111

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=192.168.0.1

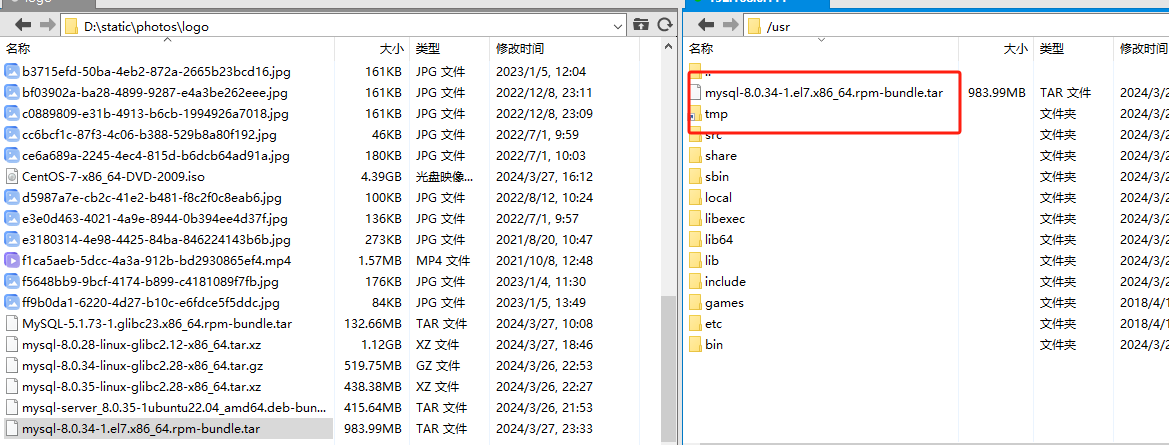


重启网络

systemctl restart network

# 安装

## 3.1上传文件到服务器



## 3.2卸载原有mysql依赖包

rpm -qa | grep mysql

rpm -qa | grep MySQL

yum remove mysql mysql-\*

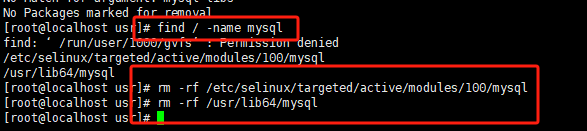
yum remove mysql MySQL-\*

卸载

rpm -e \*\* 注意是rpm -qa检查出来的名称

find / -name mysql

删除查找出来的文件



还有安装前要卸载linux自带的数据库

一般在安装完liunx时系统会默认自带一个mariadb的数据库，需要先把其卸载才能安装最的。

卸载命令如下：

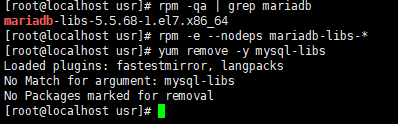
yum remove -y mysql-libs

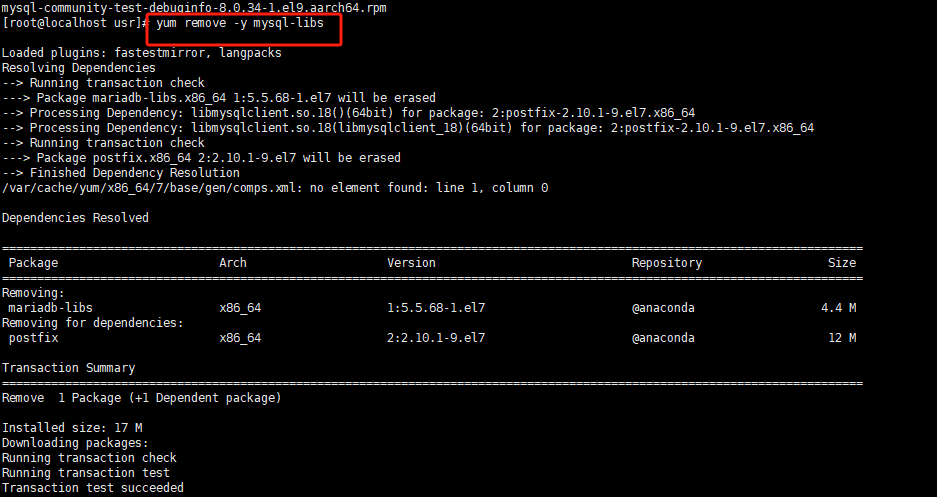
## 3.2卸载mariadb

rpm -qa | grep mariadb

rpm -e --nodeps mariadb-libs-\*

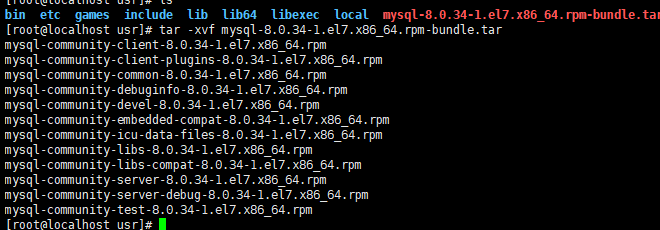
yum remove -y mysql-libs





## 3.3解压

tar -xvf mysql-8.0.34-1.el7.x86\_64.rpm-bundle.tar



## 3.4安装

分别依次安装下列6个文件

rpm -ivh mysql-community-common-8.0.34-1.el7.x86\_64.rpm

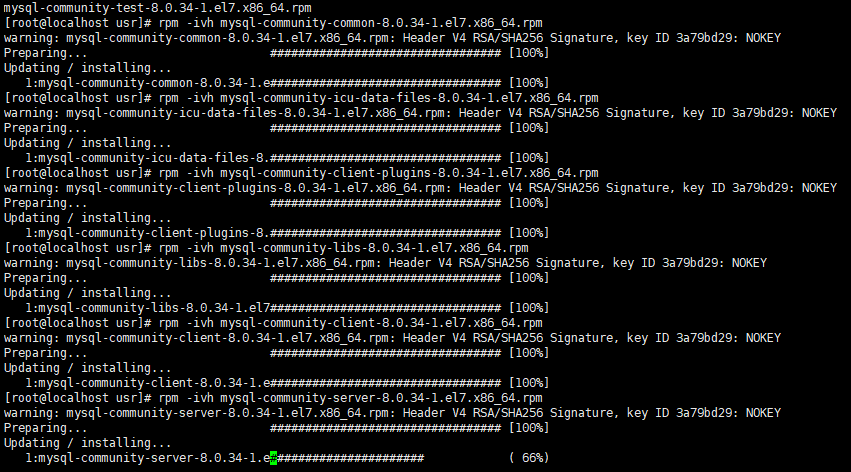
rpm -ivh mysql-community-icu-data-files-8.0.34-1.el7.x86\_64.rpm

rpm -ivh mysql-community-client-plugins-8.0.34-1.el7.x86\_64.rpm

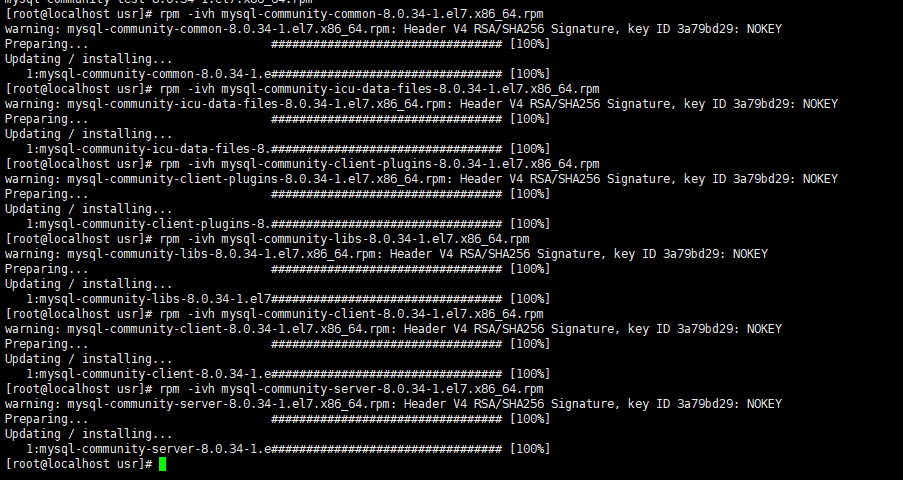
rpm -ivh mysql-community-libs-8.0.34-1.el7.x86\_64.rpm

rpm -ivh mysql-community-client-8.0.34-1.el7.x86\_64.rpm

rpm -ivh mysql-community-server-8.0.34-1.el7.x86\_64.rpm



完成如下



**另外安装mysql-community-libs-compat php依赖文件，也可以在7.4以后安装）**

**注意：如果需要搭载php使用，需要安装7.7，因为mysql-community-libs-compat是php的依赖。；如果不安装php，则无需安装7.7的依赖。**

rpm -ivh mysql-community-libs-compat-8.0.34-1.el7.x86\_64.rpm

## 3.5启动和重启

**启动**

systemctl start mysqld.service

或者

systemctl start mysqld



**重启**

systemctl restart mysqld.service

或者

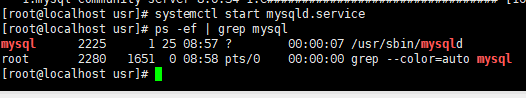
systemctl restart mysqld

**查看状态**

systemctl status mysqld

## 3.6查看启动进程

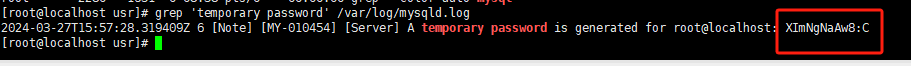
ps -ef | grep mysql



## 3.7查看初始密码

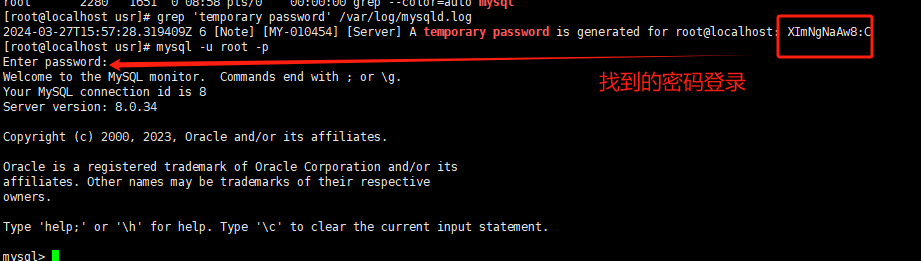
在日志里查找root密码（启动MySQL服务后才能有此日志，如果从来没有启动过，是不会有这个日志的）

grep 'temporary password' /var/log/mysqld.log



## 3.8修改初始密码

mysql -u root –p

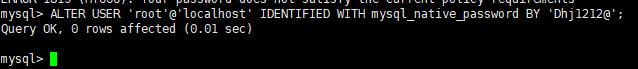


在数据库登录状态下修改密码命令

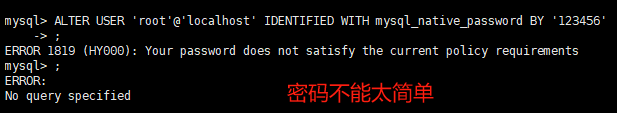
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'new password';

例如：

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'Dhj1212@';



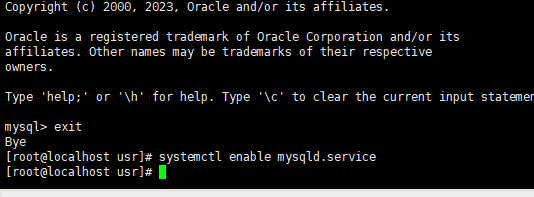
注意新密码不要太简单，因为设置了mysql\_native\_password策略



## 3.9设置开机启动服务

如果在mysql,退出mysql，执行

systemctl enable mysqld.service



## 3.10登录mysql

本地登录：

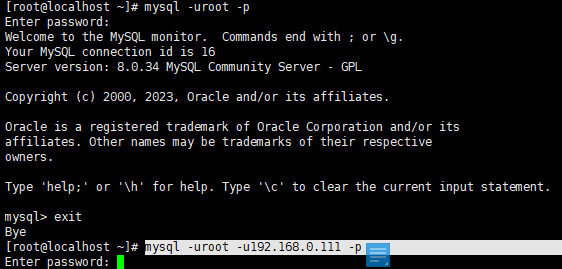
mysql -uroot –p

远程登录

mysql -uroot -u192.168.0.111 –p

退出

Exit

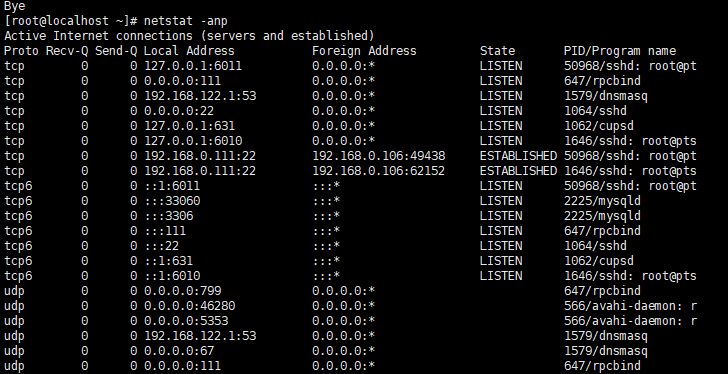


# 访问

## 4.1开放端口

### 4.1.1 查询已开放的端口

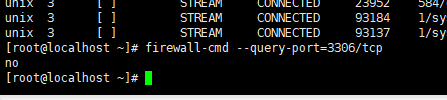
netstat –anp



### 4.1.2 检查指定的端口是否打开

firewall-cmd --query-port=3306/tcp

提示 yes，表示开启；no表示未开启。



### 4.1.3 开放端口

命令：firewall-cmd --add-port=123/tcp --permanent

firewall-cmd --add-port=3306/tcp --permanent

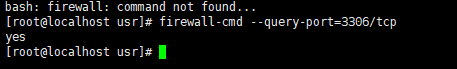


### 4.1.4 重载添加的端口并检查

firewall-cmd --reload

检查开放是否成功

firewall-cmd --query-port=3306/tcp

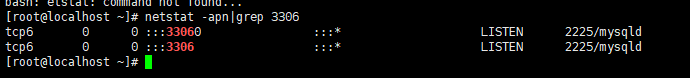


### 4.1.5 移除指定端口

firewall-cmd --permanent --remove-port=123/tcp

### 4.1.6 查看正在使用或者监听端口

netstat -apn|grep 3306

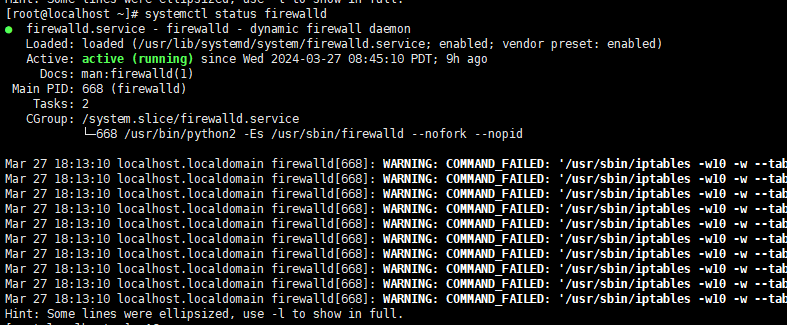


## 4.2开放防火墙

如果不想端口管理，可以关闭防火墙

### 4.2.1 查看防火墙状态

systemctl status firewalld



### 4.2.2 开启防火墙

命令：systemctl start firewalld

或者

命令：service firewalld start

**如遇到无法开启**

1.先执行

systemctl unmask firewalld.service

2.后执行

systemctl start firewalld.service

3.手动添加

vi /etc/sysconfig/iptables

添加如下代码

-A INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT 或者

-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT

可以参照22端口设置

4.重启

service iptables restart

### 4.2.3 关闭防火墙

命令：systemctl stop firewalld

### 4.2.4 禁用防火墙

sudo systemctl disable firewalld

这将从系统中删除防火墙服务的启动脚本，使其不会在系统启动时自动启动。

### 4.2.4 永久禁用防火墙

如果您希望永久禁用防火墙，您可以通过编辑 /etc/systemd/system/multi-user.target.wants/firewalld.service 文件来实现。执行以下命令打开该文件：

sudo vim /etc/systemd/system/multi-user.target.wants/firewalld.service

在打开的文件中，将 "enabled=1" 改为 "enabled=0"，然后保存并关闭文件。这将确保防火墙不会在系统重启时自动启动。

## 4.3Mysql授权远程访问

### 4.3.1 修改用户身份验证插件

MySQL 8.0默认使用caching\_sha2\_password身份验证插件，为了兼容更多旧版客户端，你可以将root用户的认证方式更改为mysql\_native\_password插件：

命令如下：（如果修改密码的时候已经设置了mysql\_native\_password就不需要设置）

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'your\_strong\_password';

比如root密码设置为Facai1212@：

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY ' Facai1212@;

### 4.3.2 允许任意IP访问

授权之前必须先登入mysql,下同

select User,authentication\_string,Host from mysql.user;

#更新指定用户主机（root用户为例）

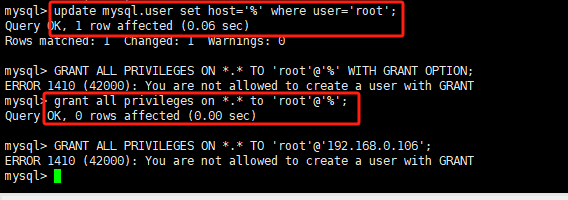
update mysql.user set host='%' where user='root';

#授权

grant all privileges on \*.\* to 'root'@'%';

-- 刷新Mysql用户权限，注意必须要刷新才能使修改生效

FLUSH PRIVILEGES;



### 4.3.3 特定IP访问

8.0后不能通过grant all privileges直接给指定IP授权，因为在8.0以后，这个特性已被移除，官方文档如下

原文：Using GRANT to modify account properties other than privilege assignments. This includes  
authentication, SSL, and resource-limit properties. Instead, establish such properties at account-creation  
time with CREATE USER or modify them afterward with ALTER USER.  
译文：使用grant修改账户权限分配以外的账户属性。包括认证，SSL，和资源限制配置等。取而代之的是创建用户create user或者创建后修改alter user的方式。

要给用户授予指定IP，必须重新创建用户

在登录mysql后，切换到mysql库，我们给2个主机（192.168.0.106，192.168.0.111）root用户授权

**mysql> use mysql;**

**给192.168.0.106授权**

创建用户【'root'@'ipaddress1'】并设置密码模式和密码【'password'】

CREATE USER 'root'@'192.168.0.106' IDENTIFIED WITH mysql\_native\_password BY 'Dhj1212@';

-- 给账号【'root'@'ipaddress1'】赋予所有权限

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'192.168.0.106';

-- 刷新Mysql用户权限，注意必须要刷新才能使修改生效

FLUSH PRIVILEGES;

**给192.168.0.111授权**

创建用户【'root'@'ipaddress2'】并设置密码模式和密码【'password'】

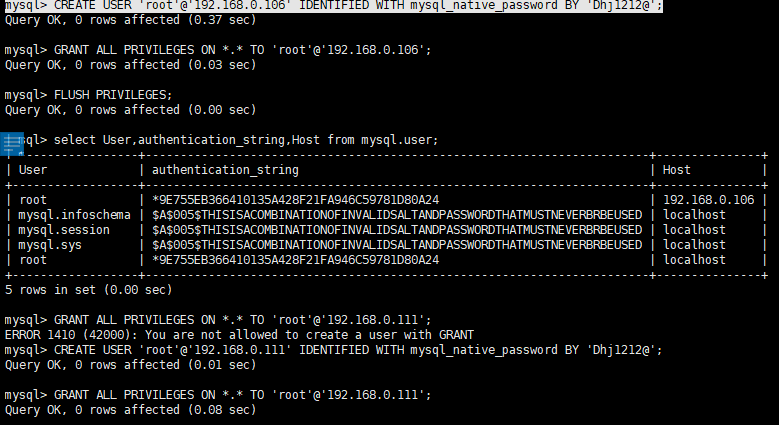
CREATE USER 'root'@'192.168.0.111' IDENTIFIED WITH mysql\_native\_password BY 'Dhj1212@';

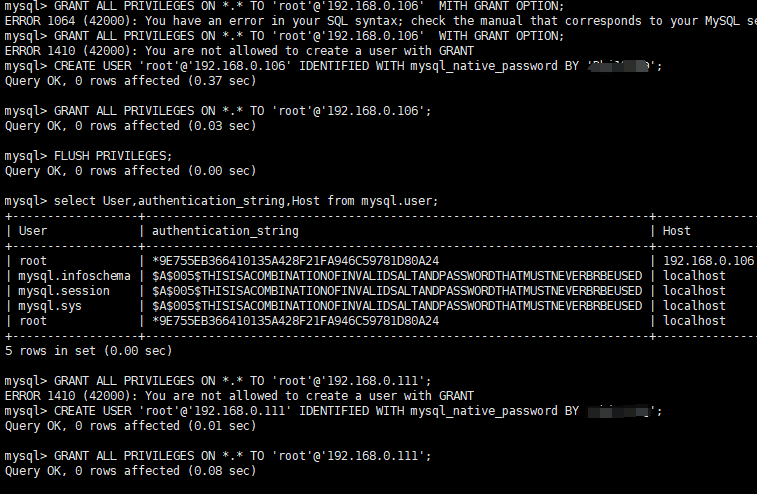
-- 给账号【'root'@'ipaddress2'】赋予所有权限

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'192.168.0.111';

-- 刷新Mysql用户权限，注意必须要刷新才能使修改生效

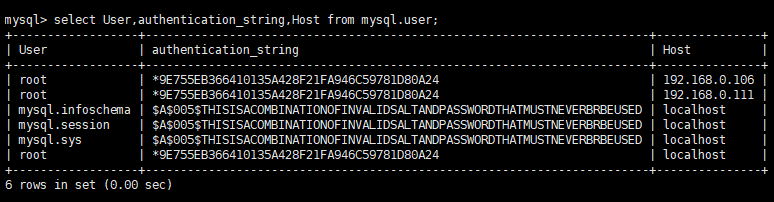
FLUSH PRIVILEGES;





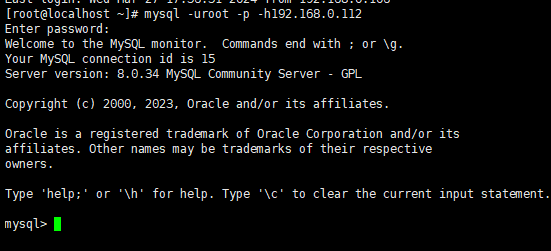
然后再查看用户信息

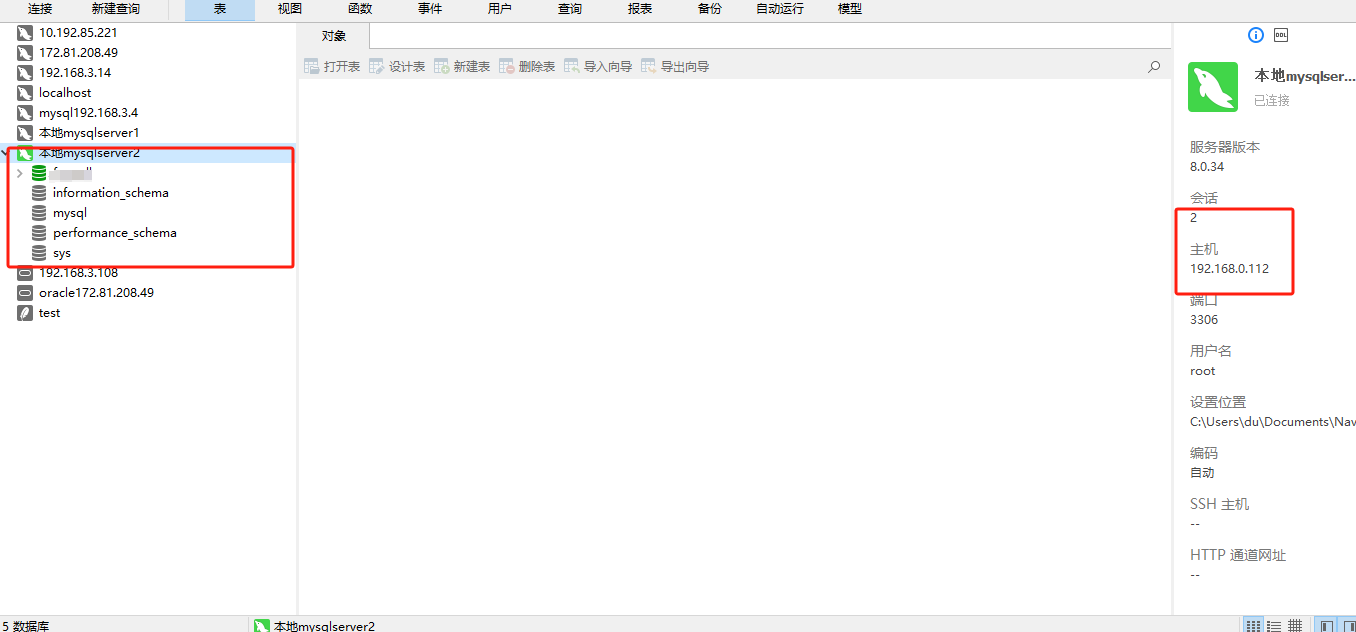
select User,authentication\_string,Host from mysql.user;



最后在另外一台主机测试

mysql -uroot -p -h192.168.0.112





**如果本机不能访问**

-- 创建用户【'root'@'localhost'】并设置密码模式和密码【'password'】

CREATE USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password';

-- 给账号【root'@'localhost'】赋予所有权限

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'localhost';

-- 刷新Mysql用户权限，注意必须要刷新才能使修改生效

FLUSH PRIVILEGES;

### 4.3.4 刷新权限

授权以后，flush才生效！

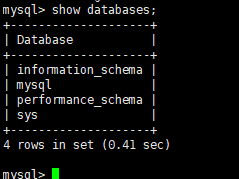
flush privileges;

# 数据库操作

## 5.1创建数据库

### 5.1.1 查看当前数据库

mysql> show databases;

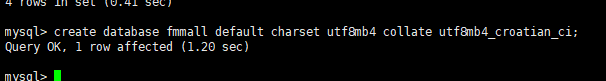


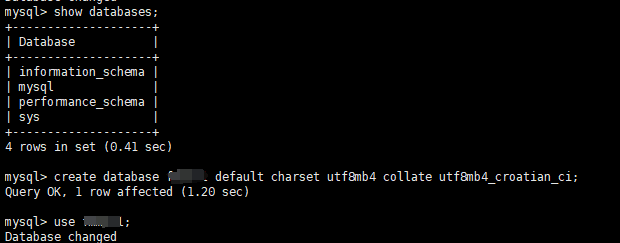
### 5.1.2 查看当前数据库

-- 创建名为test\_db的数据库，字符集设置为utf8mb4，字符集排序规则为utf8mb4\_croatian\_ci

create database test\_db default charset utf8mb4 collate utf8mb4\_croatian\_ci;

mysql> create database fmmall default charset utf8mb4 collate utf8mb4\_croatian\_ci;





### 5.1.3 切换数据库

mysql> use test\_db;

### 5.1.4 导入数据库文件

-- 导入/root/test/目录下的test\_db.sql文件

source /root/test/test\_db.sql;

我的

mysql> source /usr/fmmall2.sql;

### 5.1.5 查看数据库表中的表名称

use 数据库

show tables;

### 5.1.6 显示当前库中所有表的基本信息(所在库、表名、使用的引擎、数据行数、数据大小、表备注等)

select \* from information\_schema.`TABLES` t WHERE t.TABLE\_SCHEMA = (select database());

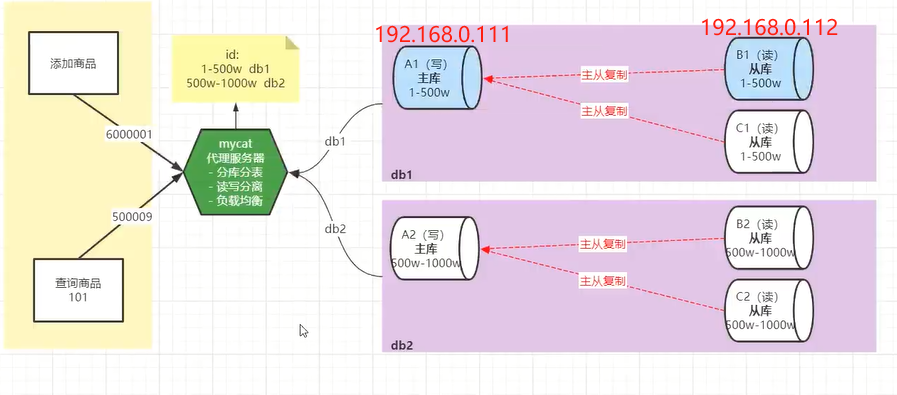
# MySql8.0主从复制配置

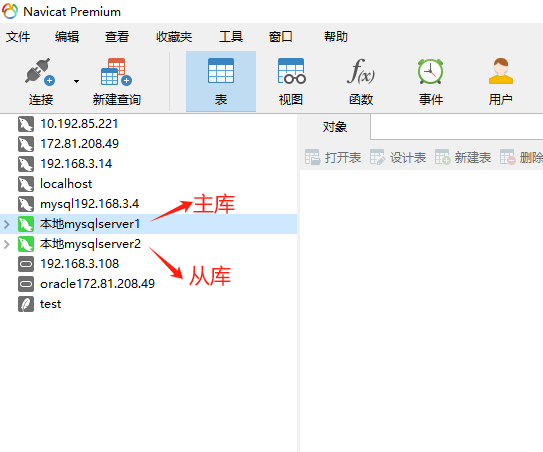
**当前演示IP：**

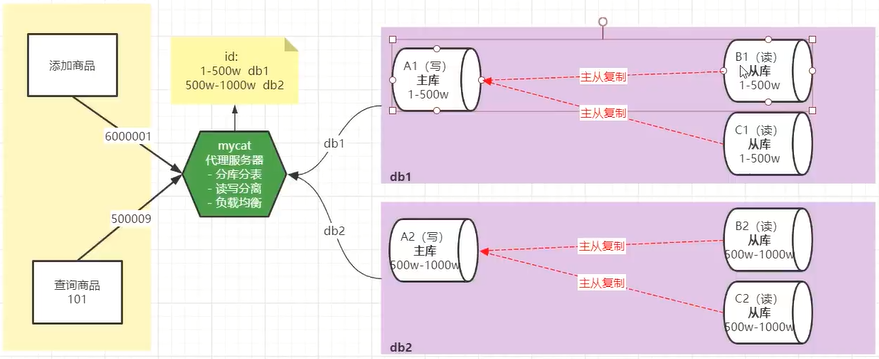
**主库IP：192.168.0.111（mysqlserver1）Master**

**从库IP：192.168.0.112（mysqlserver2）Slave**

故名思意，至少需要两台MySQL服务器，一台主(写)服务器(Master)，一台从(读)服务器(Slave)。写在主服务器上的数据会自动同步







## 6.1准备

### 6.1.1 条件

1.centos7中安装两个相同版本的mysql8，保证root用户是可以正常登陆的，

2.保证两台linux的网络是通的。切记将主mysql所在的linux的防火墙关闭(命令为：systemctl disable firewalld)。检查两台linux的网络是否是相通，可以通过ping对方的ip,看是否可以ping的通。例如: linux1的ip为192.168.0.111，linux2的ip为192.168.0.112，在linux1中ping 192.168.0.112，ping的通，就说明两台linux的网络是通的

3.创建同名的数据库

## 6.2设计思路

我们准备**两组**主存复制服务器群，分别为一台主库，两台从库；也就是说，总共需要2台主库服务器，4台从库服务器，一台mycat代理服务器

**第一组：**

**主库1：192.168.0.111**

**从库：192.168.0.112，192.168.0.113**

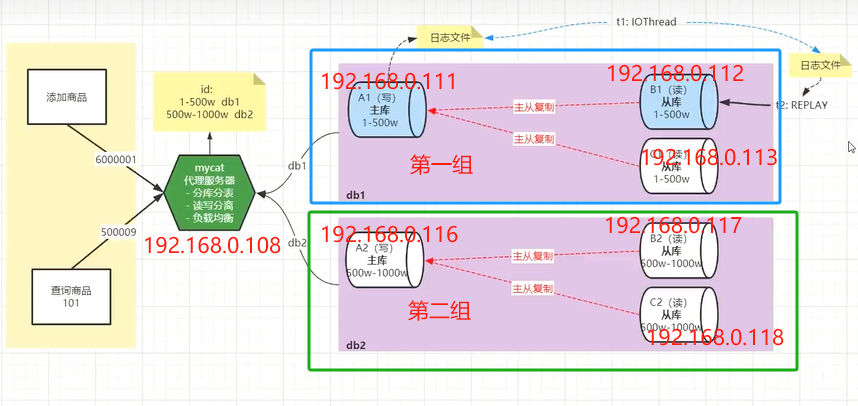
**第二组：**

**主库2：192.168.0.116**

**从库：192.168.0.117，192.168.0.118**

**mycat代理服务器（代理两台主库）：**

**192.168.0.108 （暂时用本机192.168.0.106 windows10 替代测试，linux同理；具体见下章的【MyCat读写分离配置】）**



## 6.3第一组主从复制配置

**主库1：192.168.0.111**

**从库：192.168.0.112，192.168.0.113**

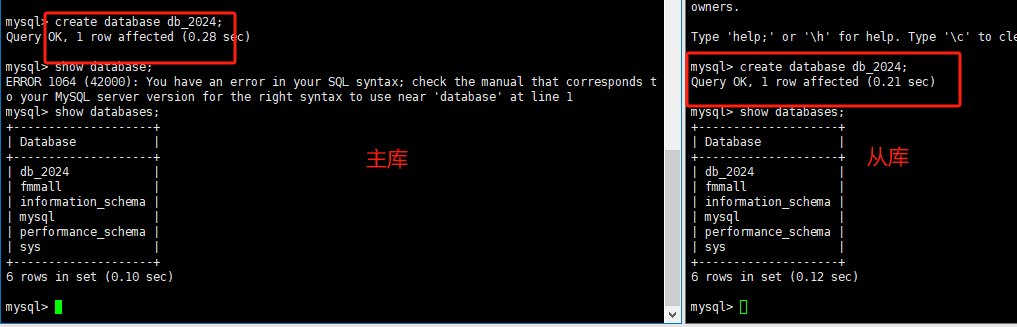
### 6.3.1 创建主从复制的数据库

**给主库1创建数据库 db\_2024**

create database db\_2024;

**给从库（192.168.0.112和192.168.0.113）创建数据库 db\_2024**

create database db\_2024;



### 6.3.2 配置主库（192.168.0.111）

**1、修改配置文件（主库192.168.0.111操作）**

vim /etc/my.cnf

给最后一行加如下代码

#id要确保在局域网环境里是唯一的，必须是整数

server-id=1

##1代表只读，0代表可读写

#read-only=0

##忽略的数据,指不需要同步的数据库，如需要请把以下的#注释符号删掉即可

##lbinlog-ignore-db=mysql

##开启日志记录

log-bin=mysql-bin

##指定同步的数据库，如需要请把以下的#注释符号删掉即可，注释掉表示全库同步，我们现在只同步db\_2024

binlog-do-db=db\_2024



**2、重启mysql服务**

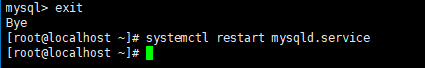
**systemctl restart mysqld.service**

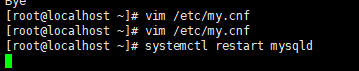
**或者**

**systemctl restart mysqld**

**或者**

service mysqld restart



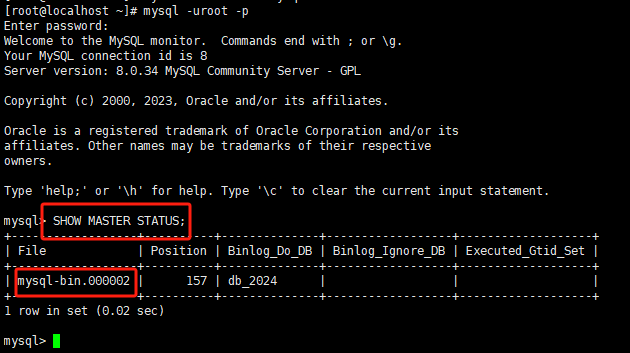


**3、检查主库配置**

登录mysql

mysql> SHOW MASTER STATUS;

如图，说明配置成功。



### 6.3.3 配置主库1的从库1（192.168.0.112）

指定当前数据库以谁为主库。

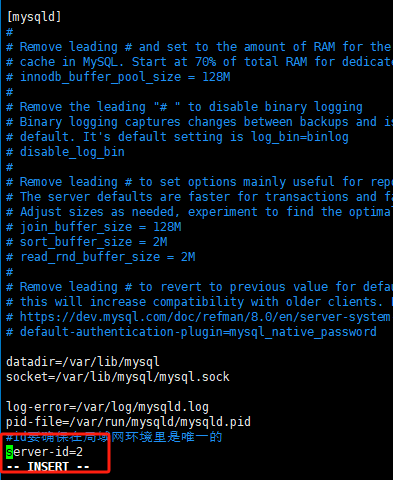
**1、修改配置文件（从库192.168.0.112操作）**

vim /etc/my.cnf

加入如下代码

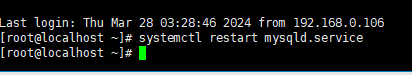
#id要确保在局域网环境里是唯一的

server-id=2



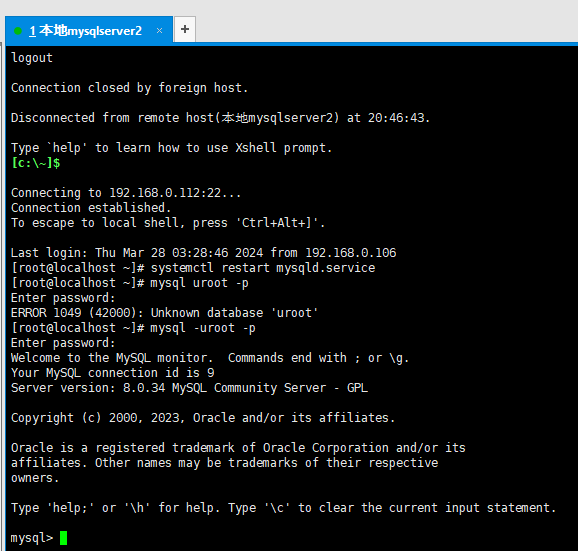
**2、重启mysql服务（从库192.168.0.112操作）**

**systemctl restart mysqld.service**



### 6.3.4 开启主从复制（192.168.0.112）

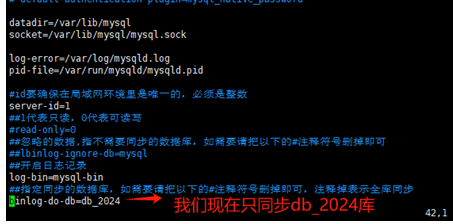
**1、进入mysql的主库1从库1（在从库1服务器 192.168.0.112）**



**2、切换到数据库（单独主从复制db\_2024库需操作）**

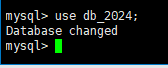
如果我们在主库中配置了只同步db\_2024

binlog-do-db=db\_2024



**切换到db\_2024**

use db\_2024;



否则如果全库复制，没有配置binlog-do-db=db\_2024，则切**换到mysql**

use mysql;

**3、配置主从复制（设置主库信息）**

登录mysql后执行

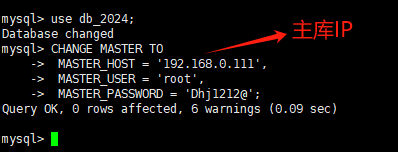
**配的简单（不建议）**

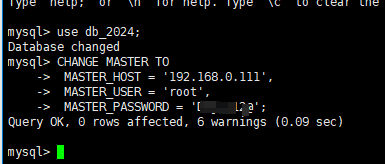
CHANGE MASTER TO

MASTER\_HOST = '192.168.0.111',

MASTER\_USER = 'root',

MASTER\_PASSWORD = 'Dhj1212@';





**配的更详细点（采用这种）**

CHANGE MASTER TO

MASTER\_HOST = '192.168.0.111',

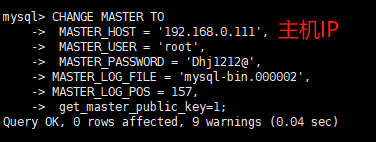
MASTER\_USER = 'root',

MASTER\_PASSWORD = 'Dhj1212@',

MASTER\_LOG\_FILE = 'mysql-bin.000002',

MASTER\_LOG\_POS = 157,

get\_master\_public\_key=1;



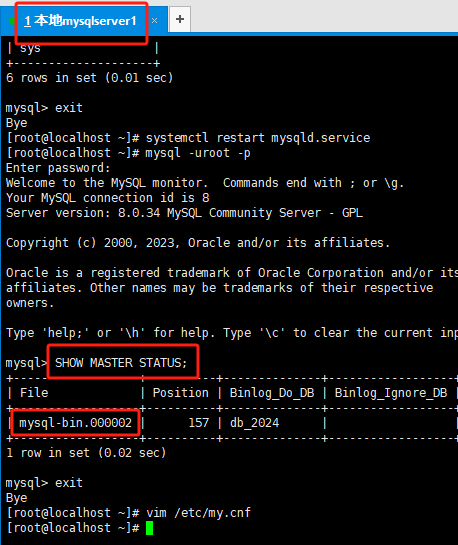
其中：

#主服务器数据库上的file值(不能有空格)

MASTER\_LOG\_FILE = 'mysql-bin.000002',

登录主库1（192.168.0.111），进入mysql，use db\_2024,执行如下指令查询

MASTER\_LOG\_FILE在在主服务上通过命令



#主服务器数据库的Position值

MASTER\_LOG\_POS = 157,

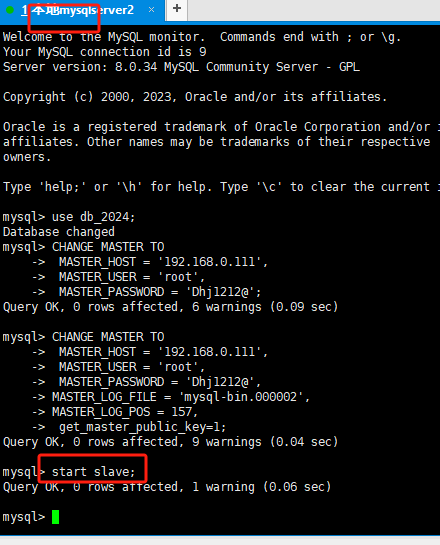
get\_master\_public\_key=1;

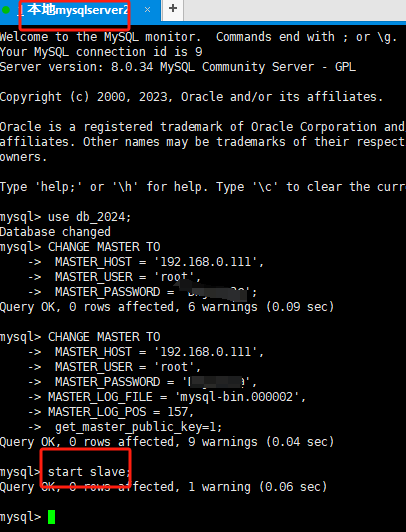
### 6.3.5启动主从服务

在主库1从服务上（192.168.0.112）登录mysql，启动主从服务

#开启同步

start slave;

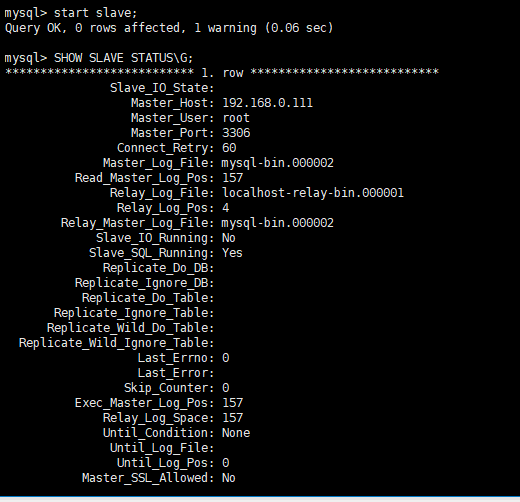




### 6.3.6查看主从服务状态

SHOW SLAVE STATUS\G;

mysql> SHOW SLAVE STATUS\G;

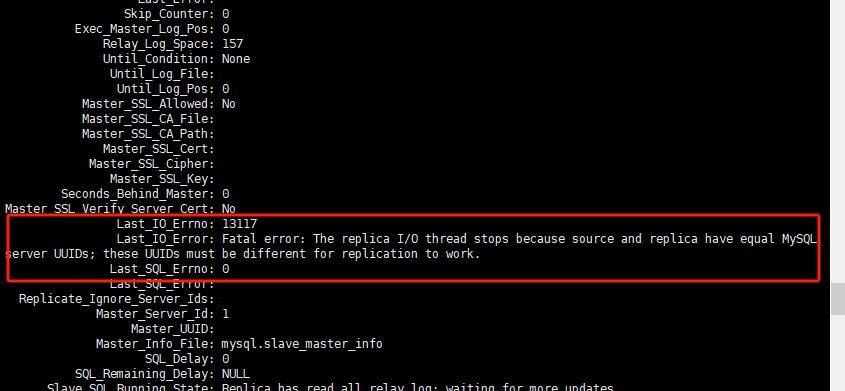


检查输出中的 Slave\_IO\_Running 和 Slave\_SQL\_Running 字段，确保两个字段的值都是 **Yes**，表示主从复制已经成功配置。

**错误：**

发现Slave\_IO\_Running 为NO，在看看错误信息

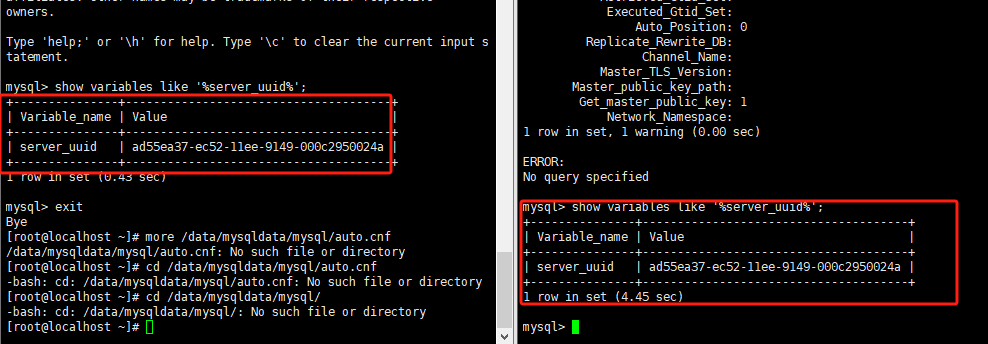
Last\_IO\_Error: Fatal error: The replica I/O thread stops because source and replica have equal MySQL server UUIDs; these UUIDs must be different for replication to work.



**原因：**知道了是UUID一致的问题，但我们还是要确定下UUID的值，分别在主机和从机的mysql下执行show variables like '%server\_uuid%';命令查看uuid值。发现server\_uuid是一致的

mysql> show variables like '%server\_uuid%';

**发现果然一致**



而为什么我这里的UUID一样呢:为了能够使用，我使用了VMware克降了两台虚拟机作为主机和从机，所以就报了此错。

**解决：**

解决方法就是找到主机和从机的auto.cnf文件修改uuid值或删除auto.cnf这这个文件

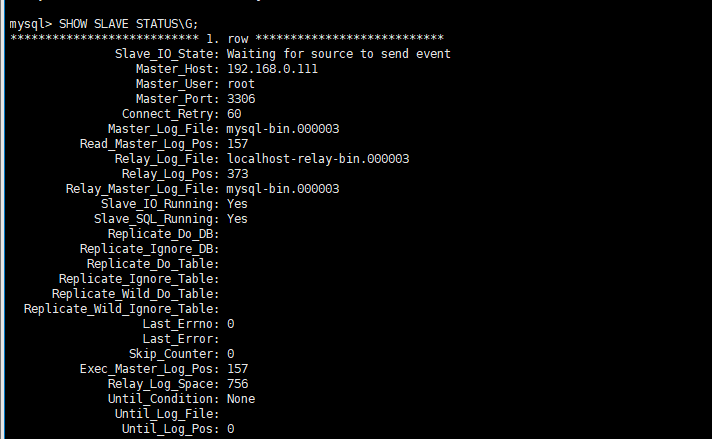
我们只修改从机，

删除以下文件即可。

rm –rf /var/lib/mysql/auto.cnf

然后重启mysql服务

再看结果



### 6.3.7停止主从服务

stop slave;



### 6.3.8测试

1.在主库中的db\_2004中创建一张表，比如创建user1

mysql> use db\_2024

mysql> CREATE TABLE `user1` (

-> `user\_id` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '用户ID',

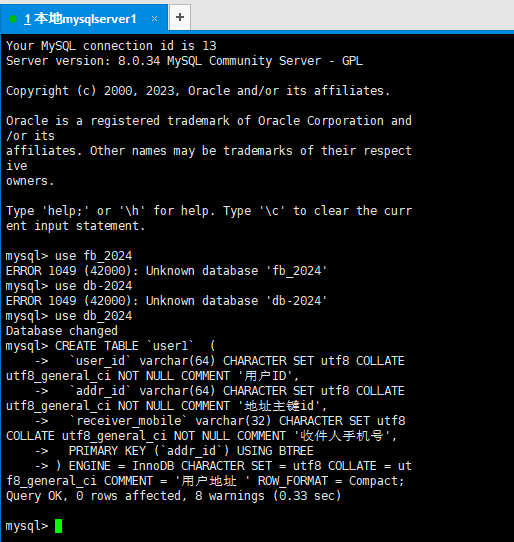
-> `addr\_id` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '地址主键id',

-> `receiver\_mobile` varchar(32) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '收件人手机号',

-> PRIMARY KEY (`addr\_id`) USING BTREE

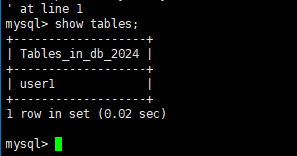
-> ) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '用户地址 ' ROW\_FORMAT = Compact;

**同理，可以创建多个从库。一旦开启了主库，创建的新的从库的数据也从主存数据库同步过来，老的数据也会同步。**

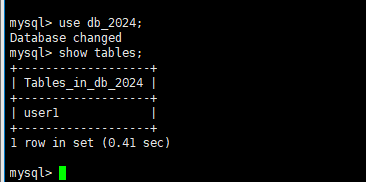


2. 在主库中的db\_200数据库中查看user1是否创建

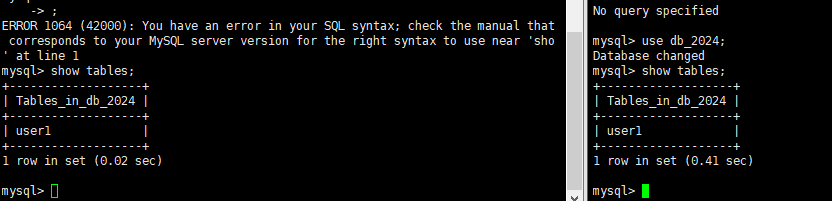
show tables;



3. 在从库中的db\_2004数据库中查看user1是否已经复制创建



发现已经复制创建。



4. 进一步测试，在从库中的db\_200数据库的user1表中插入一条数据

在主库插入两条数据

INSERT INTO `user1` VALUES ('1', '1', '13030303300');

INSERT INTO `user1` VALUES ('2', '22', '13653303346');



从库也随着插入两条数据

### 6.3.9 配置主库1的从库2（192.168.0.113）

**和配置存库1一致**

**1、修改配置文件（从库192.168.0.113操作）**

vim /etc/my.cnf

加入如下代码

#id要确保在局域网环境里是唯一的

server-id=3

**2、重启mysql服务（从库192.168.0.113操作）**

**systemctl restart mysqld.service**

**3.开启主存复制**

进入mysql从库2（192.168.0.113），use db\_2024;

use db\_2024;

mysql下执行

CHANGE MASTER TO

MASTER\_HOST = '192.168.0.111',

MASTER\_USER = 'root',

MASTER\_PASSWORD = 'Dhj1212@',

MASTER\_LOG\_FILE = 'mysql-bin.000002',

MASTER\_LOG\_POS = 157,

get\_master\_public\_key=1;

**3.启动主存复制**

在从库2（192.168.0.113）上登录mysql，在mysql下启动主从服务

start slave;

**3.查看状态**

mysql> SHOW SLAVE STATUS\G;

确保Slave\_IO\_Running 和 Slave\_SQL\_Running 字段，两个字段的值都是 **Yes**

**4.测试和从库一致。**

## 6.4第二组主从复制配置

**主库2：192.168.0.116**

**从库：192.168.0.117，192.168.0.118**

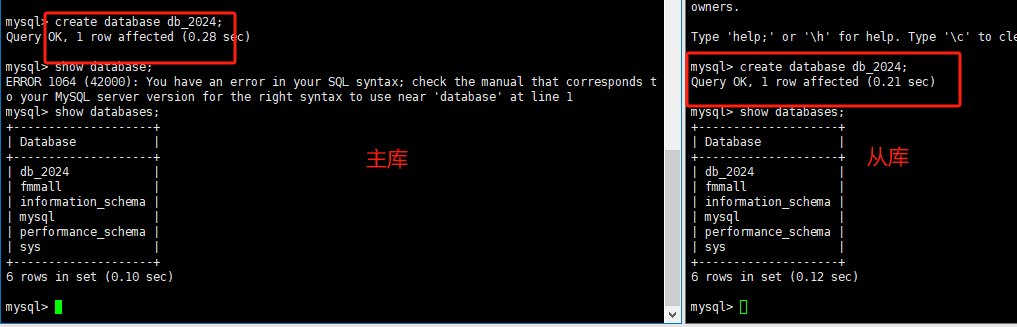
### 6.4.1 创建主从复制的数据库

**给主库2创建数据库 db\_2024**

create database db\_2024;

**给从库（192.168.0.117和192.168.0.118）创建数据库 db\_2024**

create database db\_2024;



### 6.4.2 配置主库（192.168.0.116）

**1、修改配置文件（主库192.168.0.116操作）**

vim /etc/my.cnf

给最后一行加如下代码

#id要确保在局域网环境里是唯一的，必须是整数

server-id=4

##1代表只读，0代表可读写

#read-only=0

##忽略的数据,指不需要同步的数据库，如需要请把以下的#注释符号删掉即可

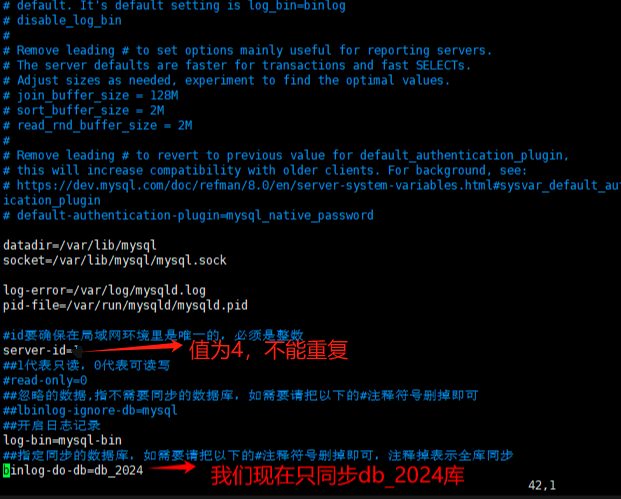
##lbinlog-ignore-db=mysql

##开启日志记录

log-bin=mysql-bin

##指定同步的数据库，如需要请把以下的#注释符号删掉即可，注释掉表示全库同步，我们现在只同步db\_2024

binlog-do-db=db\_2024



**2、重启mysql服务**

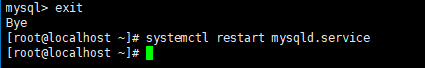
**systemctl restart mysqld.service**

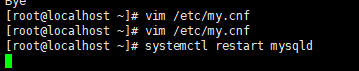
**或者**

**systemctl restart mysqld**

**或者**

service mysqld restart



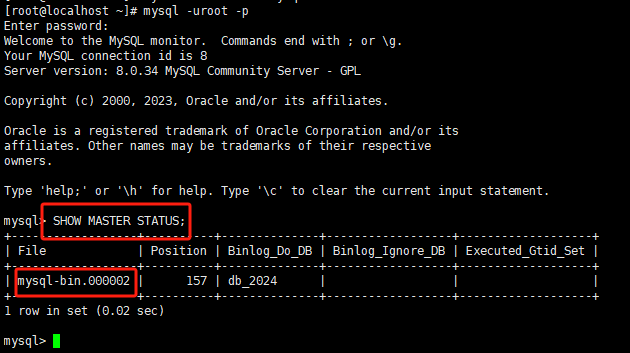


**3、检查主库配置**

登录mysql

mysql> SHOW MASTER STATUS;

如图，说明配置成功。



### 6.4.3 配置主库2的从库1（192.168.0.117）

指定当前数据库以谁为主库。

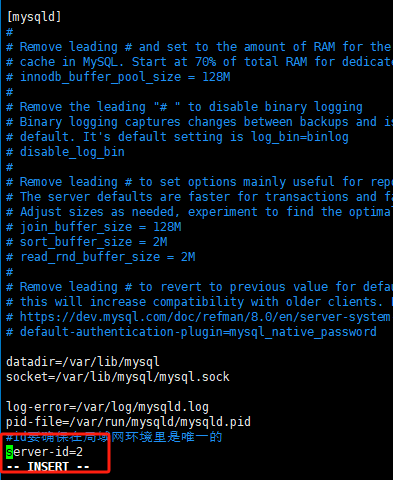
**1、修改配置文件（从库192.168.0.117操作）**

vim /etc/my.cnf

加入如下代码

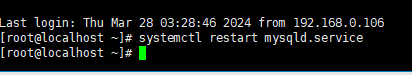
#id要确保在局域网环境里是唯一的

server-id=5



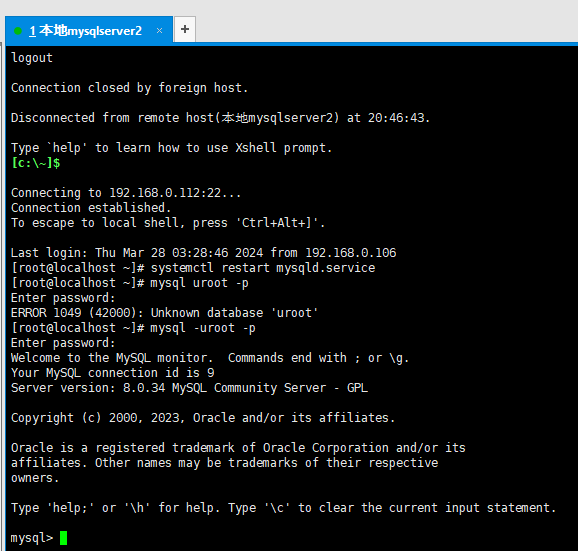
**2、重启mysql服务（从库192.168.0.117操作）**

**systemctl restart mysqld.service**



### 6.4.4 开启主从复制（192.168.0.117）

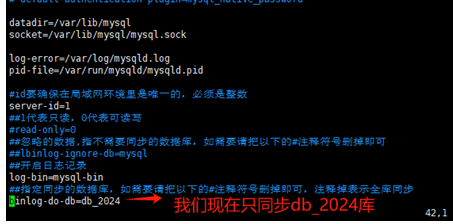
**1、进入mysql的主库2的从库1（在从库1服务器 192.168.0.116）**



**2、切换到数据库（单独主从复制db\_2024库需操作）**

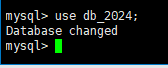
如果我们在主库中配置了只同步db\_2024

binlog-do-db=db\_2024



**切换到db\_2024**

use db\_2024;



否则如果全库复制，没有配置binlog-do-db=db\_2024，则切**换到mysql**

use mysql;

**3、配置主从复制（设置主库信息）**

登录主库2的从库1（192.168.0.117）mysql后执行

CHANGE MASTER TO

MASTER\_HOST = '192.168.0.116',

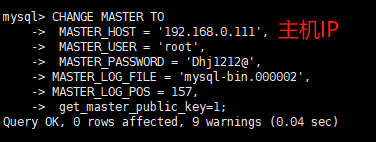
MASTER\_USER = 'root',

MASTER\_PASSWORD = 'Dhj1212@',

MASTER\_LOG\_FILE = 'mysql-bin.000002',

MASTER\_LOG\_POS = 157,

get\_master\_public\_key=1;



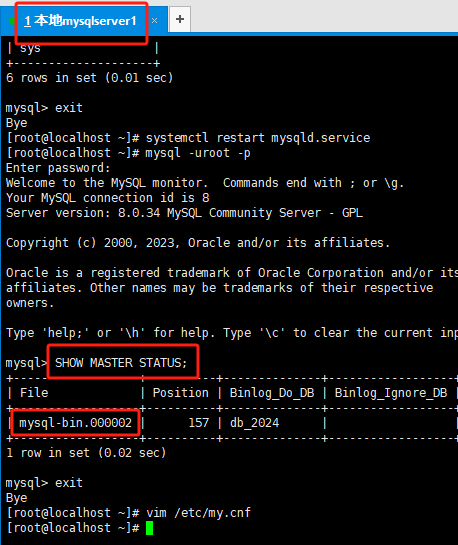
其中：

#主服务器数据库上的file值(不能有空格)

MASTER\_LOG\_FILE = 'mysql-bin.000002',

登录主库2（192.168.0.116），进入mysql，use db\_2024,执行如下指令查询

MASTER\_LOG\_FILE在在主服务上通过命令



#主服务器数据库的Position值

MASTER\_LOG\_POS = 157,

get\_master\_public\_key=1;

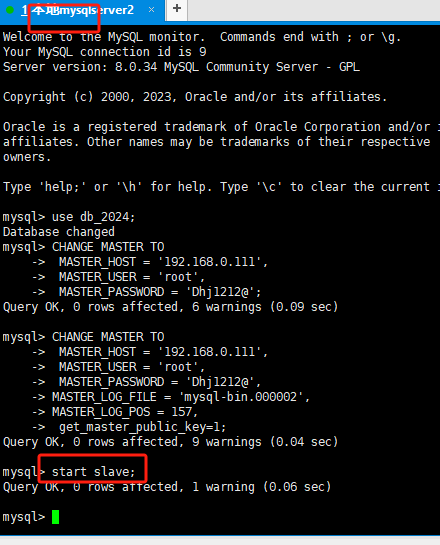
### 6.4.5启动主从服务

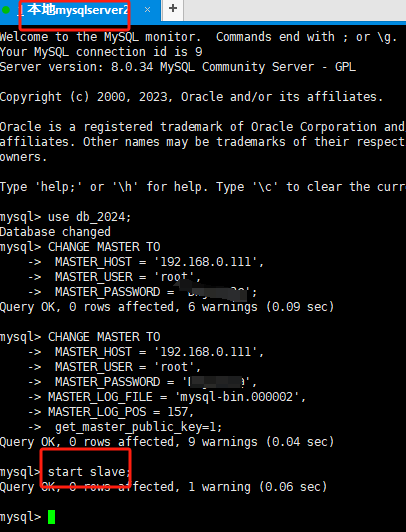
在主库2的从服务上（192.168.0.117）登录mysql，启动主从服务

#开启同步

start slave;

下图可参考：

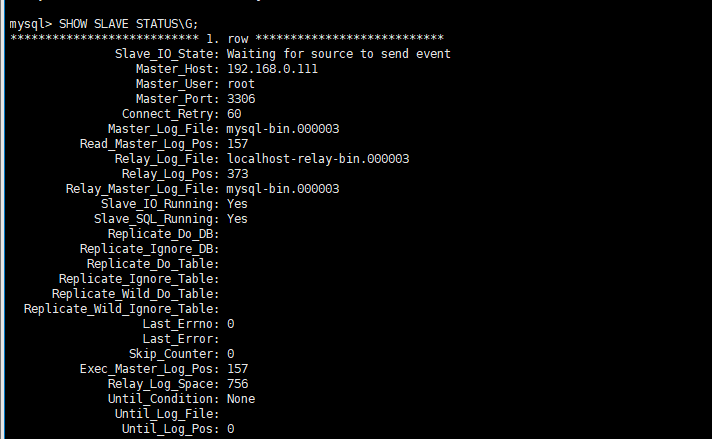




### 6.4.6查看主从服务状态

SHOW SLAVE STATUS\G;

mysql> SHOW SLAVE STATUS\G;



检查输出中的 Slave\_IO\_Running 和 Slave\_SQL\_Running 字段，确保两个字段的值都是 **Yes**，表示主从复制已经成功配置。

### 6.4.7测试

1.在主库2中的db\_2004中创建一张表，比如创建user1

mysql> use db\_2024

mysql> CREATE TABLE `user1` (

-> `user\_id` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '用户ID',

-> `addr\_id` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '地址主键id',

-> `receiver\_mobile` varchar(32) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '收件人手机号',

-> PRIMARY KEY (`addr\_id`) USING BTREE

-> ) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '用户地址 ' ROW\_FORMAT = Compact;

脚本如下：

CREATE TABLE `user1` (

`user\_id` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '用户ID',

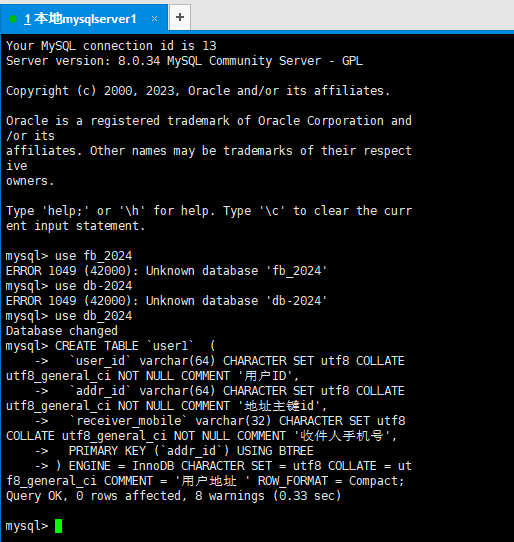
`addr\_id` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '地址主键id',

`receiver\_mobile` varchar(32) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '收件人手机号',

PRIMARY KEY (`addr\_id`) USING BTREE

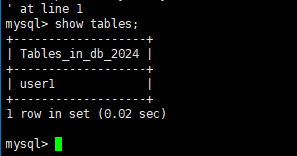
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '用户地址 ' ROW\_FORMAT = Compact;

**同理，可以创建多个从库。一旦开启了主库2，创建的新的从库（192.168.0.117，192.168.0.118）的数据也从主存数据库同步过来，老的数据也会同步。**

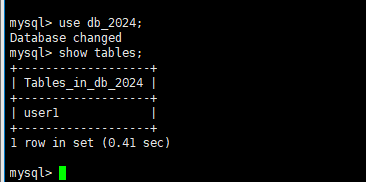


2. 在主库中的db\_200数据库中查看user1是否创建

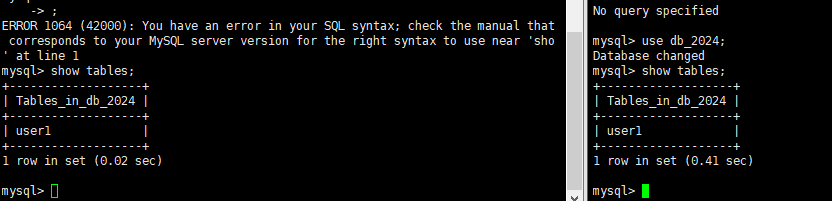
show tables;



3. 在从库中的db\_2004数据库中查看user1是否已经复制创建



发现已经复制创建。



4. 进一步测试，在从库中的db\_200数据库的user1表中插入一条数据

在主库插入两条数据

INSERT INTO `user1` VALUES ('1', '1', '13030303300');

INSERT INTO `user1` VALUES ('2', '22', '13653303346');



从库也随着插入两条数据

### 6.4.8 配置主库2的从库2（192.168.0.118）

**和配置从库（192.168.0.117）一致**

**1、修改配置文件（从库192.168.0.117操作）**

vim /etc/my.cnf

加入如下代码

#id要确保在局域网环境里是唯一的

server-id=6

**2、重启mysql服务（从库192.168.0.118操作）**

**systemctl restart mysqld.service**

**3.开启主存复制**

进入mysql从库2（192.168.0.118），use db\_2024;

use db\_2024;

mysql下执行

CHANGE MASTER TO

MASTER\_HOST = '192.168.0.111',

MASTER\_USER = 'root',

MASTER\_PASSWORD = 'Dhj1212@',

MASTER\_LOG\_FILE = 'mysql-bin.000002',

MASTER\_LOG\_POS = 157,

get\_master\_public\_key=1;

**3.启动主存复制**

在从库（192.168.0.118）上登录mysql，在mysql下启动主从服务

start slave;

**3.查看状态**

mysql> SHOW SLAVE STATUS\G;

确保Slave\_IO\_Running 和 Slave\_SQL\_Running 字段，两个字段的值都是 **Yes**

**4.测试和从库一致。**

# MyCat读写分离配置

**Mycat** 是 **采用java语言开发的开源的数据库中间件**，支持Windows和Linux运行环境

## 7.1下载

Mycat官网是<http://www.mycat.org.cn/>

下载地址是：

<http://dl.mycat.org.cn/2.0/>

下载后如下图所示：

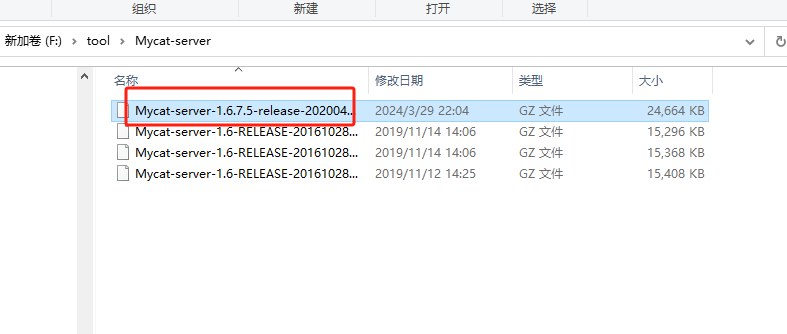
，这个版本在mysql8下没有测试成功，总是连接不上数据库，差不多数据，弃用。

我用的是Mycat-server-1.6.7.5-release-20200410174409-win.tar.gz这个版本，稳定。

环境：jdk1.8

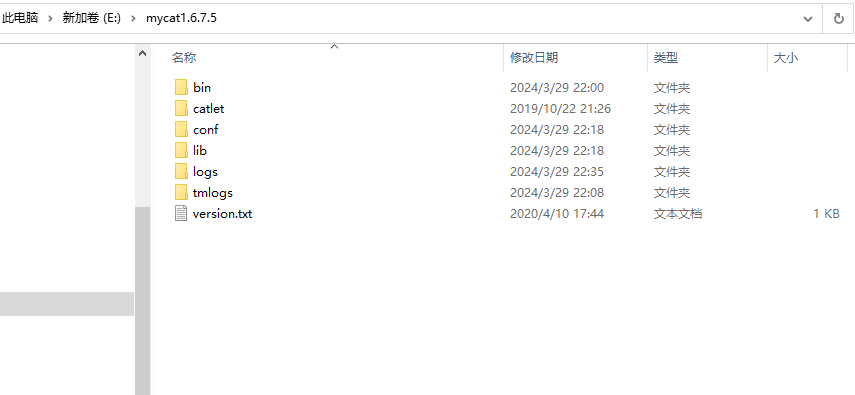
数据库：Mysql8.0.34

我用的是



## 7.2解压

下载成功后，解压到本地对应目录，我的目录结构如下：

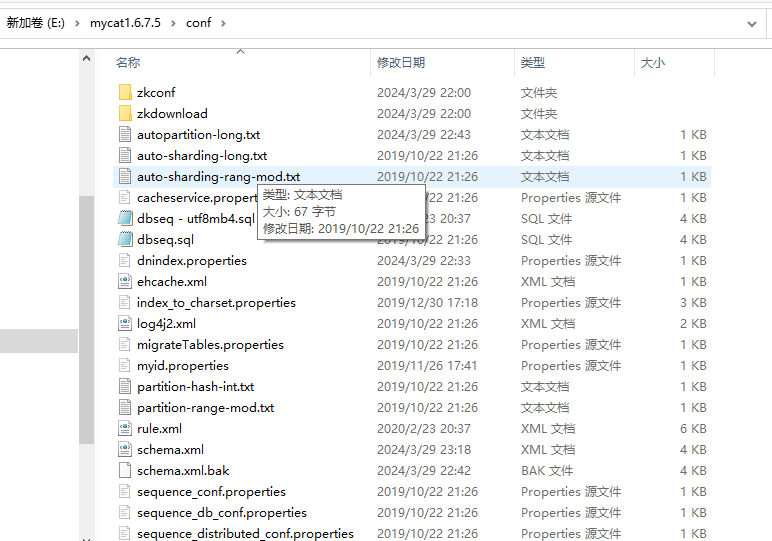


|  |  |
| --- | --- |
| **目录** | **说明** |
| bin | mycat命令，启动、重启、停止等 |
| catlet | catlet为Mycat的一个扩展功能 |
| conf | Mycat 配置信息,重点关注 |
| lib | Mycat引用的jar包，Mycat是java开发的 |
| logs | 日志文件，包括Mycat启动的日志和运行的日志。 |

## 7.3修改配置文件

配置文件的目录为mycat下的conf

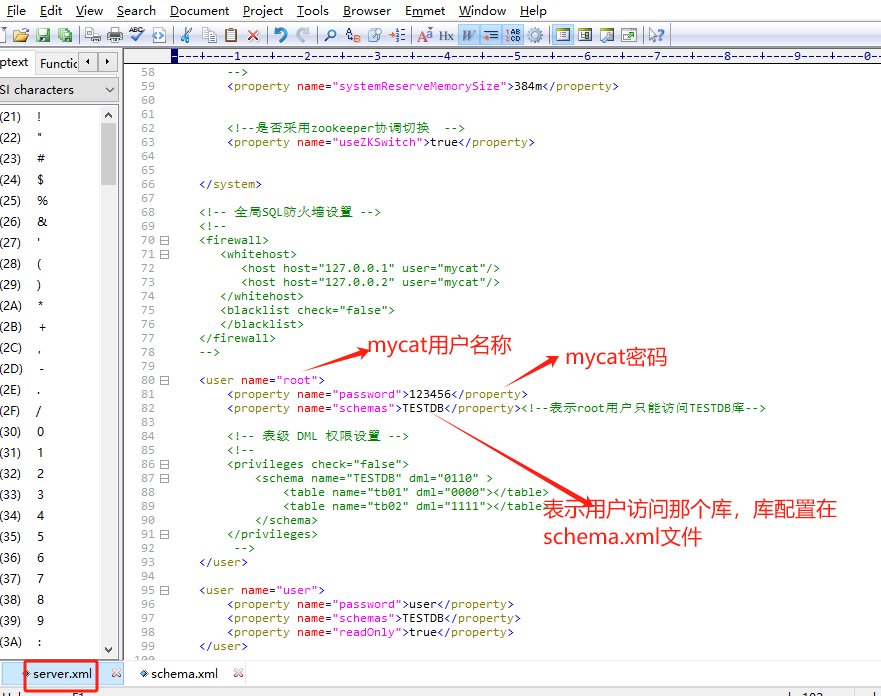
1. [schema](https://so.csdn.net/so/search?q=schema&spm=1001.2101.3001.7020).xml：定义逻辑库，表、[分片](https://so.csdn.net/so/search?q=%E5%88%86%E7%89%87&spm=1001.2101.3001.7020)节点等内容
2. rule.[xml](https://so.csdn.net/so/search?q=xml&spm=1001.2101.3001.7020)：定义分片规则
3. server.xml：定义用户以及系统相关变量，如端口等



### 7.3.1 conf/server.xml配置

1.配置用户

基本不需要配置



需要说明的是TESTDB库是虚拟的，不存在

### 7.3.2 conf/schema.xml配置

注意配置的时候最好不要用中文，防止启动失败

**1.配置数据库**

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

<!-- auto-sharding-long，对应rule.xml文件，配置规则。rule.xml中的auto-sharding-long对应本目录中 auto-sharding-long.txt规则文件 -->

<table name="user1" dataNode="dn1,dn2" rule="auto-sharding-long" /><!--对应 db\_2024库中user1表 ；-->

<!--TESTDB中的表，有多少表就创建多少条，也是虚拟的，需要dataNode指定数据库中的表-->

<table name="org1" dataNode="dn1,dn2" rule="sharding-by-intfile" autoIncrement="true" fetchStoreNodeByJdbc="true">

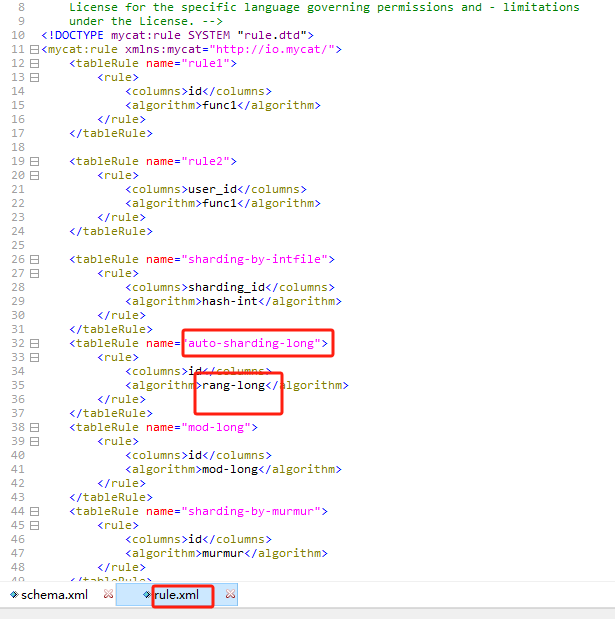
<!--<childTable name="customer\_addr" primaryKey="id" joinKey="customer\_id" parentKey="id"> </childTable>-->

</table> <!--对应db\_2024库中org1表-->

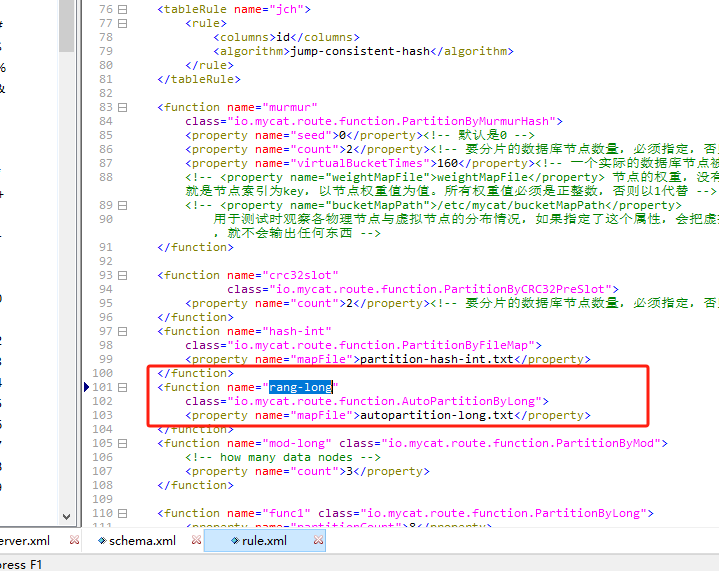
</schema>

**注意：**

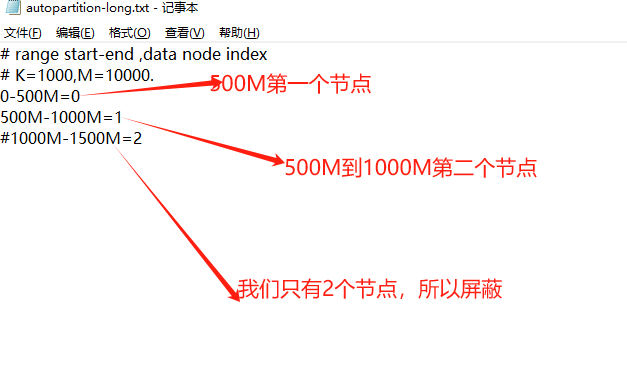
rule="auto-sharding-long" 的值对应的是rule.xml中的id，最终对应的是auto-sharding-long.txt



<algorithm>rang-long</algorithm> 对应的是autopartition-long.txt，如下图



而且autopartition-long.txt中的服务节点数要和schema.xml节点中配置的节点数相同



**2.配置mycat节点**

我们暂时配置两个主库（192.168.0.111和192.168.0.116；其中192.168.0.116由于条件关系暂时由本机mysql代替），而且如果配置了rule="auto-sharding-long"，autopartition-long.txt也必须是两个节点

数据库是主库的db\_2024

<dataNode name="dn1" dataHost="db1" database="db\_2024" />

<dataNode name="dn2" dataHost="db2" database="db\_2024" />

**3.配置mysql连接**

配置两个mysql连接，我们使用jdbc连接

配置如下：

<dataHost name="db1" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="jdbc:mysql://192.168.0.111:3306" user="root"

password="Dhj1212@">

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

<dataHost name="db2" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

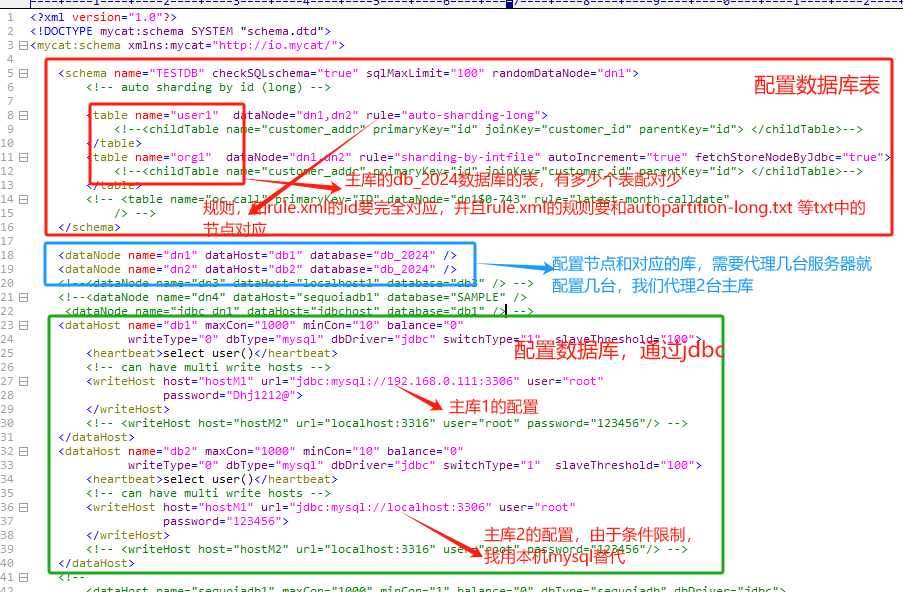
<writeHost host="hostM1" url="jdbc:mysql://localhost:3306" user="root"

password="123456">

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>整体配置完成后如下：



**完整代码如下**

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<schema name="TESTDB" checkSQLschema="true" sqlMaxLimit="100" randomDataNode="dn1">

<!-- auto sharding by id (long) -->

<table name="user1" dataNode="dn1,dn2" rule="auto-sharding-long">

<!--<childTable name="customer\_addr" primaryKey="id" joinKey="customer\_id" parentKey="id"> </childTable>-->

</table>

<table name="org1" dataNode="dn1,dn2" rule="sharding-by-intfile" autoIncrement="true" fetchStoreNodeByJdbc="true">

<!--<childTable name="customer\_addr" primaryKey="id" joinKey="customer\_id" parentKey="id"> </childTable>-->

</table>

<!-- <table name="oc\_call" primaryKey="ID" dataNode="dn1$0-743" rule="latest-month-calldate"

/> -->

</schema>

<dataNode name="dn1" dataHost="db1" database="db\_2024" />

<dataNode name="dn2" dataHost="db2" database="db\_2024" />

<!--<dataNode name="dn3" dataHost="localhost1" database="db3" /> -->

<!--<dataNode name="dn4" dataHost="sequoiadb1" database="SAMPLE" />

<dataNode name="jdbc\_dn1" dataHost="jdbchost" database="db1" /> -->

<dataHost name="db1" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="jdbc:mysql://192.168.0.111:3306" user="root"

password="Dhj1212@">

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

<dataHost name="db2" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="jdbc:mysql://localhost:3306" user="root"

password="123456">

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

<!--

<dataHost name="sequoiadb1" maxCon="1000" minCon="1" balance="0" dbType="sequoiadb" dbDriver="jdbc">

<heartbeat> </heartbeat>

<writeHost host="hostM1" url="sequoiadb://1426587161.dbaas.sequoialab.net:11920/SAMPLE" user="jifeng" password="jifeng"></writeHost>

</dataHost>

<dataHost name="oracle1" maxCon="1000" minCon="1" balance="0" writeType="0" dbType="oracle" dbDriver="jdbc"> <heartbeat>select 1 from dual</heartbeat>

<connectionInitSql>alter session set nls\_date\_format='yyyy-mm-dd hh24:mi:ss'</connectionInitSql>

<writeHost host="hostM1" url="jdbc:oracle:thin:@127.0.0.1:1521:nange" user="base" password="123456" > </writeHost> </dataHost>

<dataHost name="jdbchost" maxCon="1000" minCon="1" balance="0" writeType="0" dbType="mongodb" dbDriver="jdbc">

<heartbeat>select user()</heartbeat>

<writeHost host="hostM" url="mongodb://192.168.0.99/test" user="admin" password="123456" ></writeHost> </dataHost>

<dataHost name="sparksql" maxCon="1000" minCon="1" balance="0" dbType="spark" dbDriver="jdbc">

<heartbeat> </heartbeat>

<writeHost host="hostM1" url="jdbc:hive2://feng01:10000" user="jifeng" password="jifeng"></writeHost> </dataHost> -->

<!-- <dataHost name="jdbchost" maxCon="1000" minCon="10" balance="0" dbType="mysql"

dbDriver="jdbc"> <heartbeat>select user()</heartbeat> <writeHost host="hostM1"

url="jdbc:mysql://localhost:3306" user="root" password="123456"> </writeHost>

</dataHost> -->

</mycat:schema>

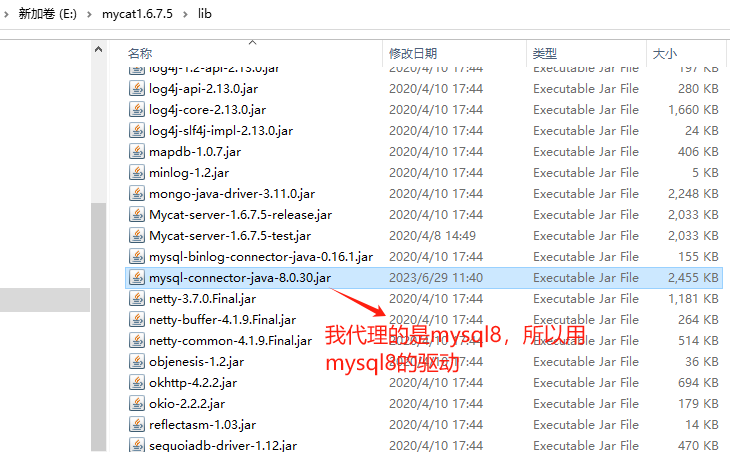
### 7.3.3 conf/rule.xml配置

规则配置文件

rule.xml关联的文件autopartition-long.txt

### 7.3.4 替换mysql驱动

由于使用jdbc连接mysql，如果使用的mysql8，则把mysql8的驱动放在lib目录下，并删除mysql5的驱动

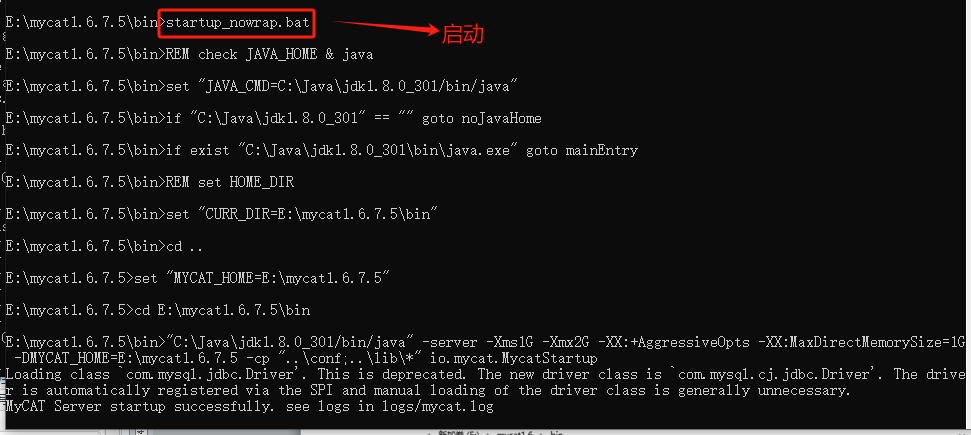


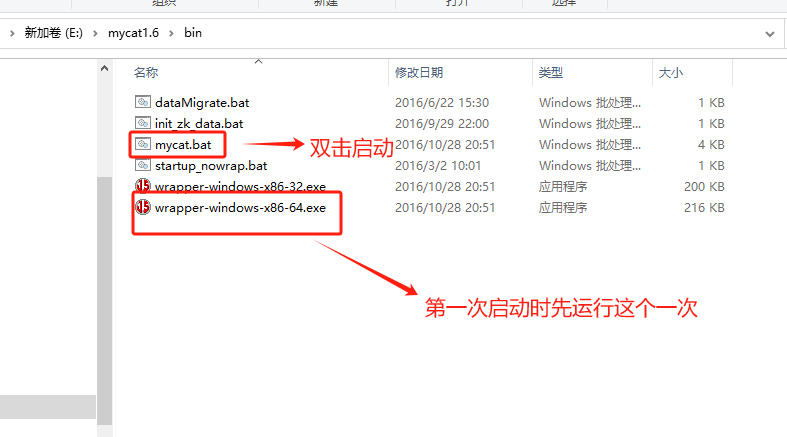
## 7.4启动mycat

找到mycat解压目录下的bin目录，bin目录下有一个叫startup\_nowrap.bat的文件，这就是mycat的启动文件。

双击mycat.bat启动，第一次启动时先运行wrapper-windows-x86-64.exe

然后开一个cmd，执行startup\_nowrap.bat



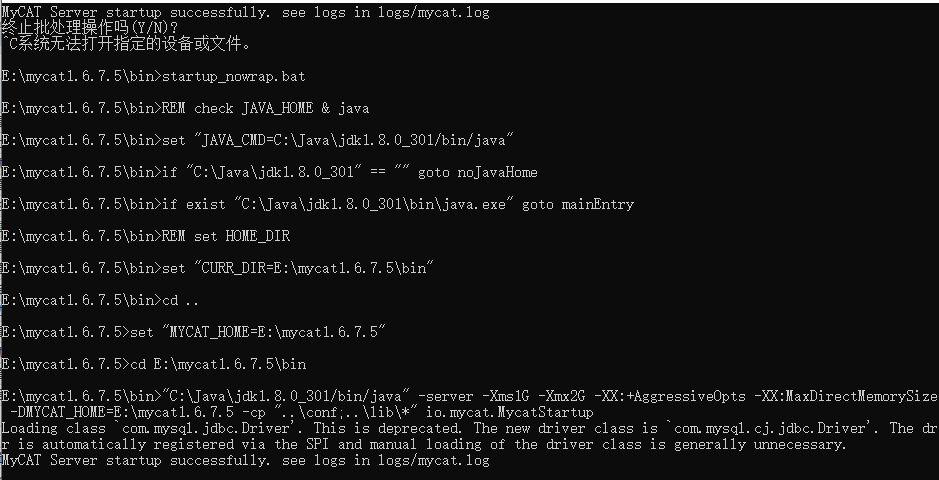


或者startup\_nowrap.bat，console下运行

windows：管理员启动cmd  
**启动命令**：mycat.bat start  
**关闭命令**：mycat.bat stop  
**查看状态**：mycat.bat status  
**重启命令**：mycat.bat restart

**前台运行命令**：mycat console

双击 bin/startup\_nowrap.bat 即运行（如果出现闪退，使用cmd命令运行，排查异常），默认端口为 8066 端口，端口可通过配置 conf/server.xml 配置



## 7.5 Navicat 连接mycat

端口：8066

用户名：root

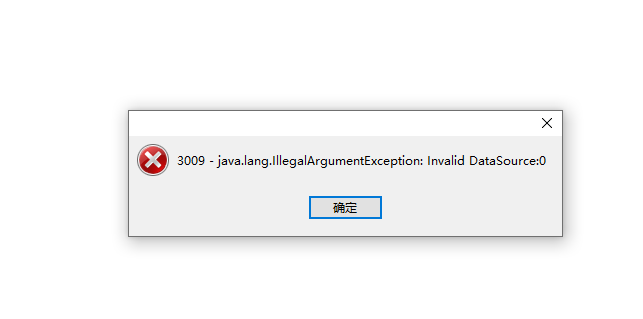
密码：123456

都可以在conf/server.xml 下配置

新建一个mysql连接

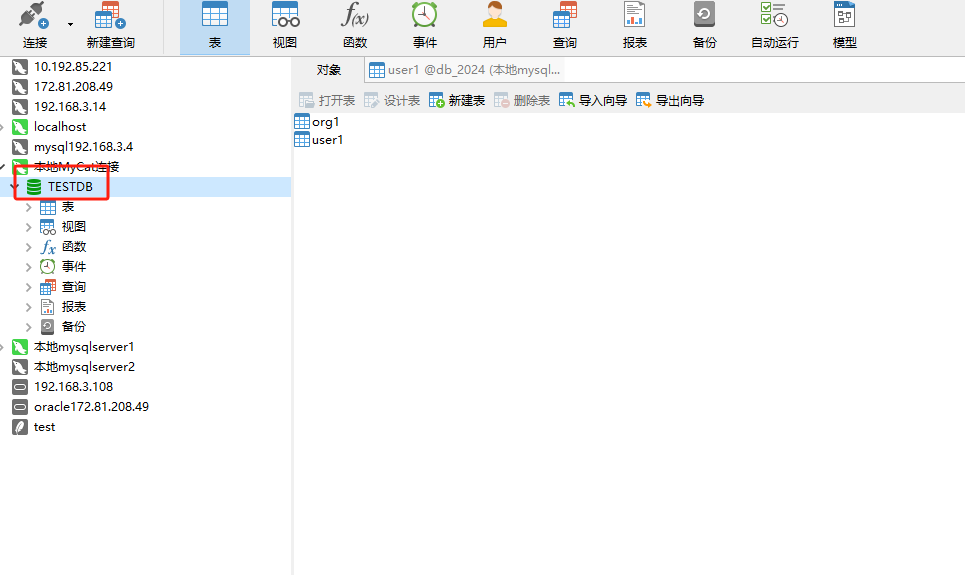


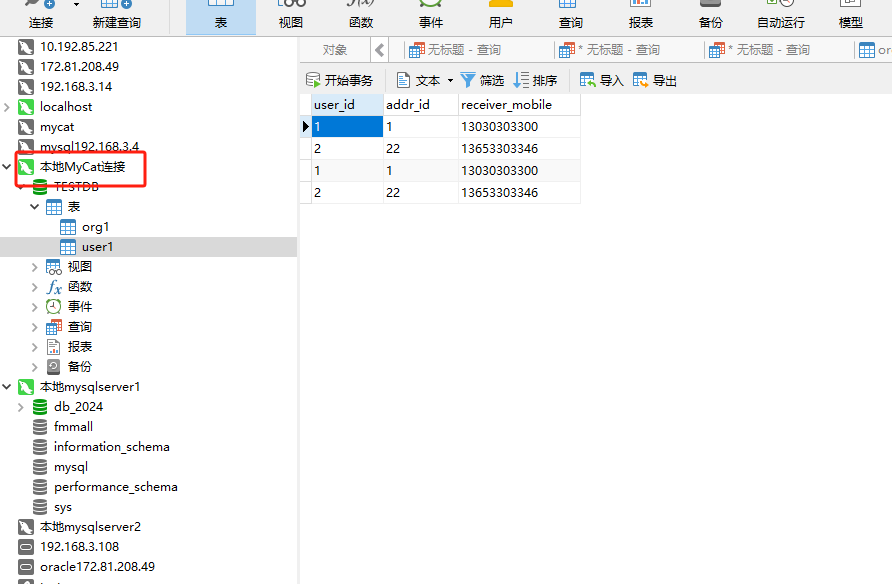
登录报如下错误



配置文件schema.xml中数据库配置异常或者无法连接。

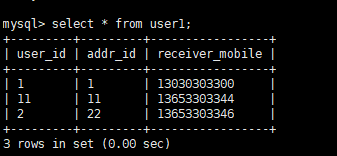
连接成功如下：





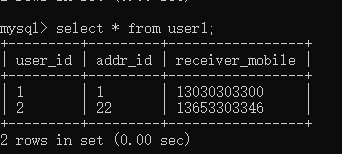
## 7.6 配置完成以后初步测试

**主库1** db\_2024库数据：



有三条数据

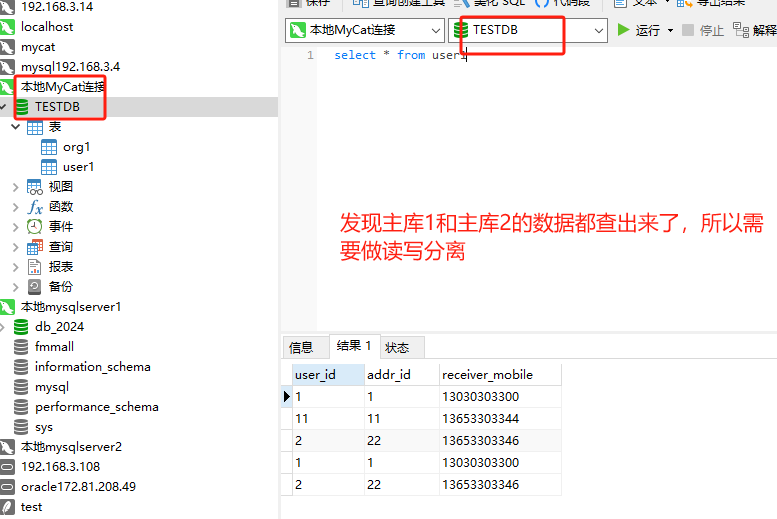
**主库2** db\_2024库数据：



有2条数据

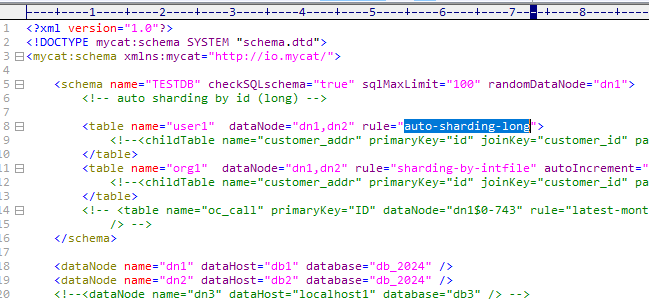
**登录mycat查询**

发现主库1的数据和主库2的数据都查出来了



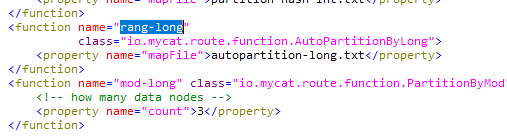
## 7.7 数据插入分库规则测试

由于我们use1表在schema.xml的规则是rule="auto-sharding-long"，rule="auto-sharding-long"

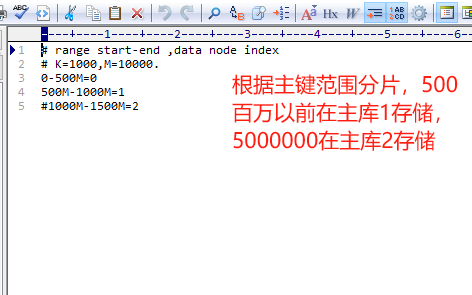
在规则rule.xml的配置是<algorithm>rang-long</algorithm>



rang-long的属性对应的是rule.xml的autopartition-long.txt文件规定，表user1字段id对应的值小于500万数据，数据插入主库1（192.168.0.111），500到1000万插入主库2（192.168.0.116）

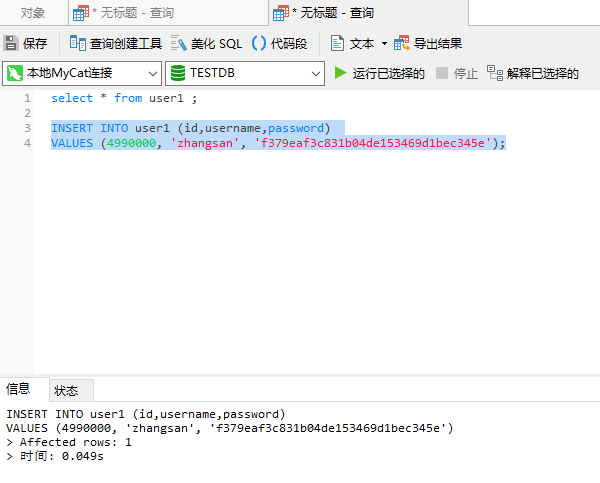


打开autopartition-long.txt

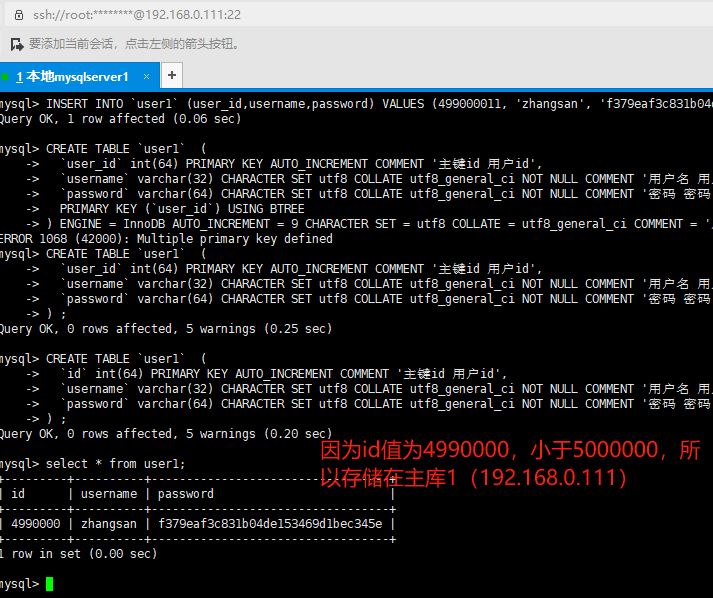


测试流程如下：

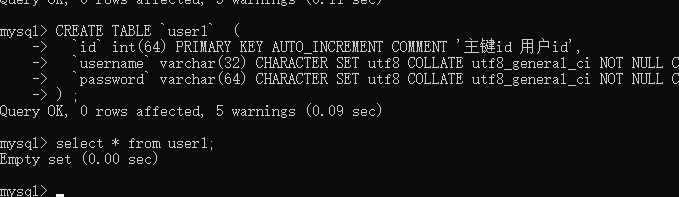
1.在mycat向user1插入一条数据，id小于500万，根据rang-long的属性对应的是rule.xml的autopartition-long.txt规则，应该写入主库1（192.168.0.111）中



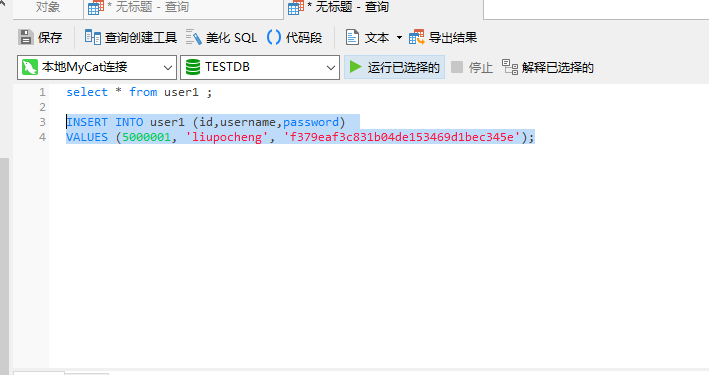
2.查询主库1数据，数据已经插入



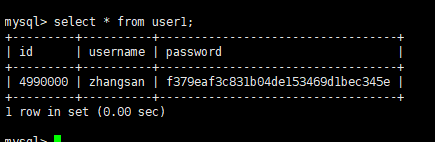
3.查询主库2数据，数据没有插入



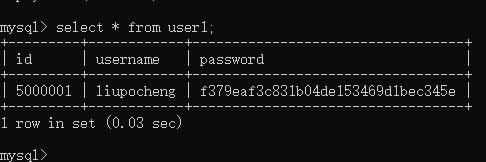
4. 在mycat向user1插入一条数据，id大于500万，根据规则，数据会存放在主库2中



5.查询主库1数据，数据5000001没有插入



5.查询主库2数据，数据5000001插入



## 7.8 读写分离配置

### 7.8.1 配置schema.xml

1.配置schema.xml，配置主库和存库读写关系，

根据前期的规划，主库（192.168.0.111，192.168.0.116）用于写数据，从库分别用于读数据（192.168.0.113、192.168.0.114；192.168.0.117、192.168.0.118，）

给主库中分别加入从库，用于读数据

<dataHost name="db1" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="jdbc:mysql://192.168.0.111:3306" user="root"

password="Dhj1212@">

<readHost host="hosts1" url="jdbc:mysql://192.168.0.112:3306" user="root" password="Admin@123"/>

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

<dataHost name="db2" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="jdbc:mysql://localhost:3306" user="root"

password="123456">

<readHost host="hosts3" url="jdbc:mysql://192.168.0.117:3306" user="root" password="Admin@123"/>

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>



2.配置开启读写分离

根据dataHost 标签属性：

Balance属性决定负载均衡方式

负载均衡类型，目前的取值有3种：

1）、balance=“0”，所有读操作都发送到当前可用的 writeHost上。

2）、balance=“1”，所有读操作都随机的发送到 readHost。

3）、balance=“2”，所有读操作都随机的在 writeHost、readhost上分发。

4）、balance=“3”，所有写操作都在 writeHost、读操作在readhost上分发。

所以配置schema.xml的<dataHost>节点

**balance=3**

**配置如下：**

<dataHost name="db1" maxCon="1000" minCon="10" balance="3"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="jdbc:mysql://192.168.0.111:3306" user="root"

password="Dhj1212@">

<readHost host="hosts1" url="jdbc:mysql://192.168.0.112:3306" user="root" password="Admin@123"/>

<readHost host="hosts2" url="jdbc:mysql://192.168.0.113:3306" user="root" password="Admin@123"/>

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

<dataHost name="db2" maxCon="1000" minCon="10" balance="3"

writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="jdbc:mysql://localhost:3306" user="root"

password="123456">

<readHost host="hosts3" url="jdbc:mysql://192.168.0.117:3306" user="root" password="Admin@123"/>

<readHost host="hosts4" url="jdbc:mysql://192.168.0.118:3306" user="root" password="Admin@123"/>

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>



### 7.8.2 重启mycat

关掉窗口，在启动

### 7.8.3 mycat中测试

