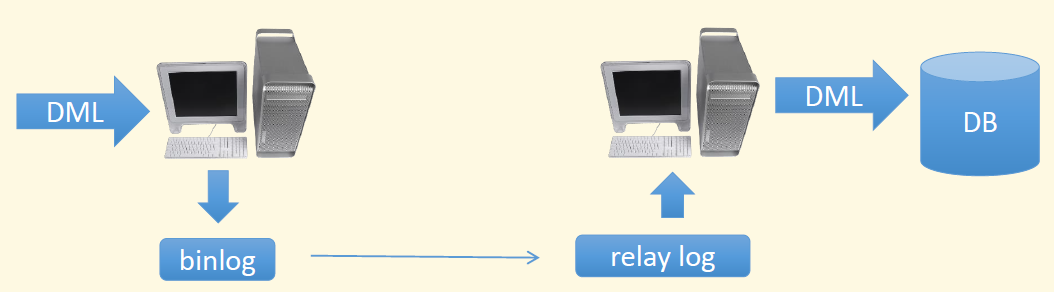
# **一：MySQL Replication**

## **什么是MySQL Replication**

* Replication可以实现将数据从一台数据库服务器（master）复制到一或多台数据库服务器（slave）
* 默认情况下属于异步复制，无需维持长连接
* 通过配置，可以复制所有的库或者几个库，甚至库中的一些表
* 是MySQL内建的，本身自带的

## Replication的原理

简单的说就是master将数据库的改变写入二进制日志，slave同步这些二进制日志，并根据这些二进制日志进行数据操作



DML:SQL操作语句，update, insert，delete

Relay log :中继日志

## Replication的作用

**1、Fail Over 故障切换**

**2、Backup Server 备份服务，无法对SQL语句执行产生的故障恢复，有限的备份**

**3、High Performance高性能，可以多台slave,实现读写分离**

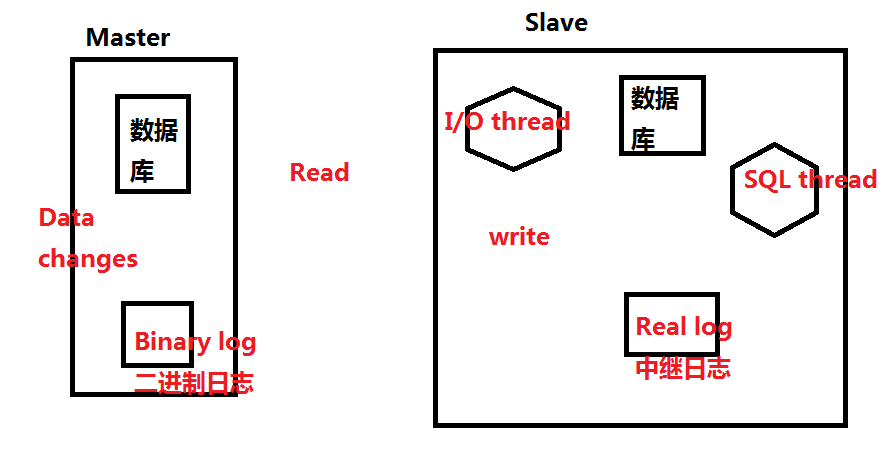
## Replication如何工作

**整体上来说，复制有3个步骤：**

**(1) master将改变记录到二进制日志(binary log)中（这些记录叫做二进制日志事件，binary log events）；**

**(2) slave将master的binary log events拷贝到它的中继日志(relay log)；**

**(3) slave重做中继日志中的事件，修改salve上的数据。**

****

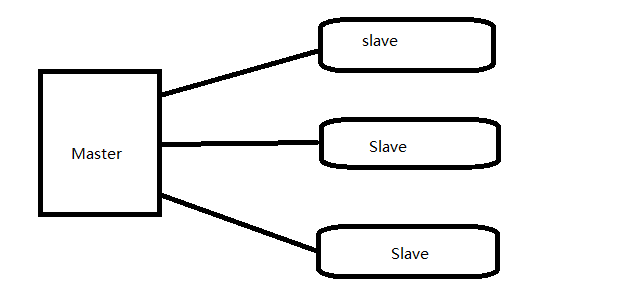
****

**mysql主从复制中：**

**第一步：master记录二进制日志。在每个事务更新数据完成之前，master在二进制日志记录这些改变。MySQL将事务写入二进制日志，即使事务中的语句都是交叉执行的。在事件写入二进制日志完成后，master通知存储引擎提交事务。  
第二步：slave将master的binary log拷贝到它自己的中继日志。首先，slave开始一个工作线程——I/O线程。I/O线程在master上打开一个普通的连接，然后开始binlog dump process。Binlog dump process从master的二进制日志中读取事件，如果已经执行完master产生的所有文件，它会睡眠并等待master产生新的事件。I/O线程将这些事件写入中继日志。  
 第三步：SQL slave thread（SQL从线程）处理该过程的最后一步。SQL线程从中继日志读取事件，并重新执行其中的事件而更新slave的数据，使其与master中的数据一致。**

## Replication常见方案：

### 1、One master and Muti salve 一主多备

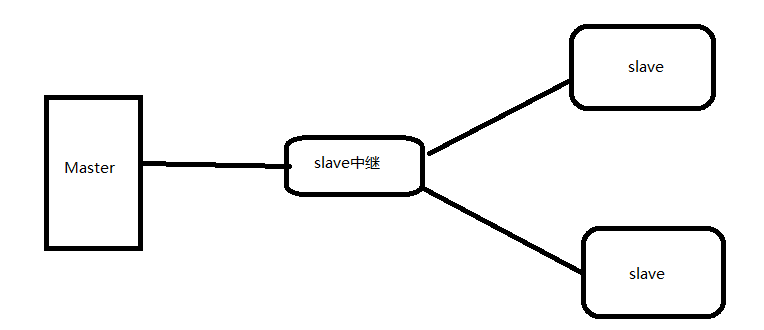


**一般用来做读写分离的，master写，其他slave读，这种架构最大问题I/O压力集中**

**在Master上<多台同步影响IO>**

### 2、M-S-S

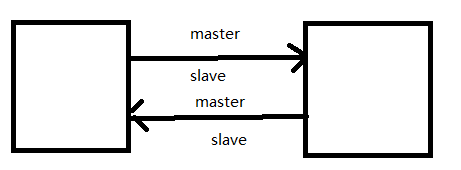
**使用一台slave作为中继，分担Master的压力，slave中继需要开启bin-log,并配置log-slave-updates**



**Slave中继可使用Black-hole存储引擎，不会把数据存储到磁盘，只记录二进制日志**

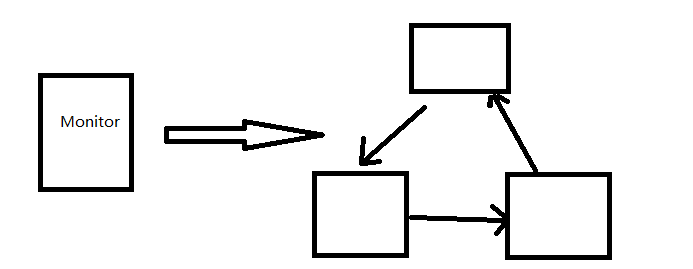
### 3、M-M 双主互备 (互为主从)

**很多人误以为这样可以做到MySQL负载均衡，实际没什么好处，每个服务器需要做同样的同步更新，破坏了事物的隔离性和数据的一致性**



### 4、M-M-M

**监控三台机器互相做对方的master**



**天生的缺陷：复制延迟，slave上同步要慢于master,如果大并发的情况那延迟更严重**

**Mysql在5.6已经自身可以实现fail over故障切换**

### 5、One slave Muti master 一从对多主

**好处：节省成本，将多个master数据自动化整合**

**缺陷：对库和表数据的修改较多**

# **二：部署MySQL主从同步 <M-S>**

## **环境准备：**

|  |  |  |  |
| --- | --- | --- | --- |
| **主机名** | **IP** | **系统/MySQL版本** | **角色** |
| **xuegod63** | **10.10.10.63** | **CentOS7.4/5.7.20** | **Master** |
| **xuegod64** | **10.10.10.64** | **CentOS7.4/5.7.20** | **slave** |

**模式：C/S**

**端口：3306**

## **配置主数据库服务器XUEGOD63**

**创建需要同步的数据库：**

**mysql> create database HA;**

**mysql> use HA;**

**mysql> create table T1(id int,name varchar(20));**

**service mysqld stop**

**配置my.cnf:**

**vim /etc/my.cnf**

**log-bin=mysql-bin-master #启用二进制日志**

**server-id=1 #本机数据库ID 标示**

**binlog-do-db=HA #可以被从服务器复制的库, 二进制需要同步的数据库名**

**binlog-ignore-db=mysql #不可以被从服务器复制的库**

**重启mysql ( 如果重启卡死现象，kill掉再启动)**

**Systemctl restart mysqld**

**授权：**

**mysql> grant replication slave on \*.\* to** [**slave@10.10.10.64**](mailto:slave@10.10.10.64) **identified by "123456";**

**查看状态信息：**

**mysql> show master status;**

**+-------------------------+----------+--------------+------------------+**

**| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |**

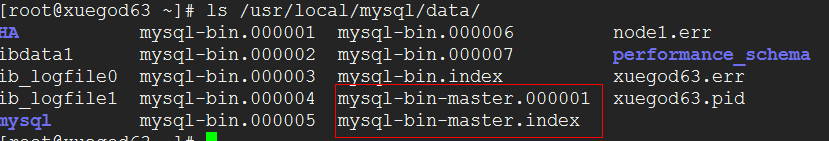
**+-------------------------+----------+--------------+------------------+**

**| mysql-bin-master.000001 | 259 | HA | mysql |**

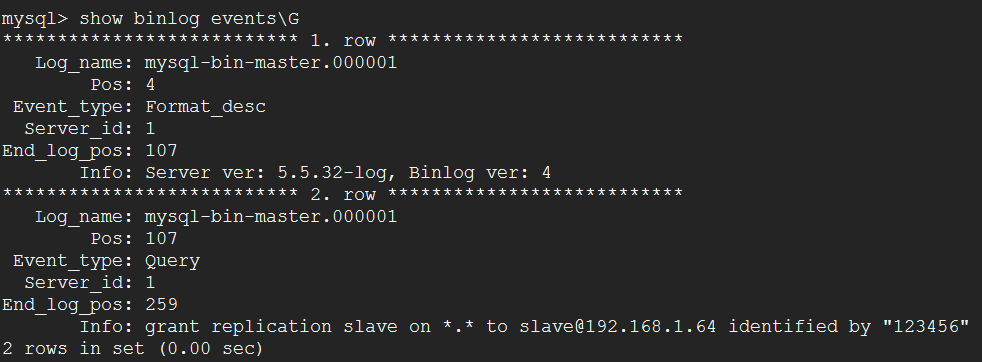
**+-------------------------+----------+--------------+------------------+**

**查看二进制日志：**

**ls /usr/local/mysql/data/**

****

**mysql> show binlog events\G**

****

**复制前要保证同步的数据库一致**

**mysqldump -uroot -p123456 HA >HA.sql #可以导出数据库**

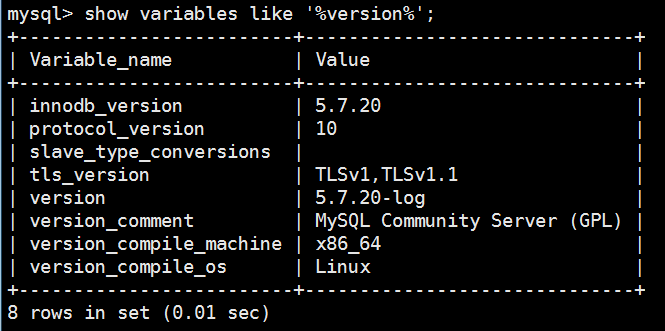
**将导出的数据库传给从服务器**

**方法：scp HA.sql 10.10.10.64:/root**

## **配置从数据库服务器XUEGOD64**

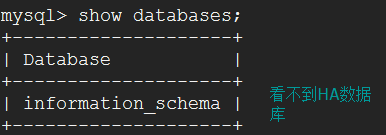
**两台数据库服务器mysql版本要一致**

**mysql> show variables like '%version%';**



**测试连接到主服务器是否成功**

**mysql -uslave -p123456 -h 10.10.10.63**

****

**只有复制的权限, 是看不到其他库的。正常**

**导入数据库，和主数据库服务器保持一致**

**mysql> create database HA;**

**mysql -uroot -p123456 HA<HA.sql**

**修改从服务器配置文件：**

**从服务器没必要开启bin-log日志**

**service mysqld stop**

**vim /etc/my.cnf**

**server-id = 2 #从服务器ID号，不要和主ID相同 ，如果设置多个从服务器，每个从服务器必须有一个唯一的server-id值，必须与主服务器的以及其它从服务器的不相同。可以认为server-id值类似于IP地址：这些ID值能唯一识别复制服务器群集中的每个服务器实例。**

**master-host=192.168.1.63 #指定主服务器IP地址**

**master-user=slave #指定定在主服务器上可以进行同步的用户名**

**master-password=123456 #密码**

**#master-port=3306**

**master-connect-retry=60 #断点重新连接时间**

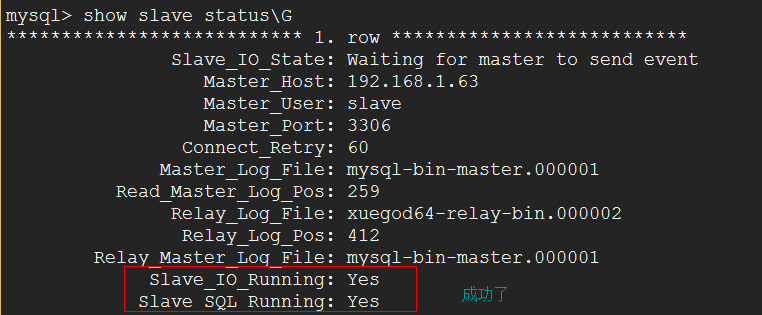
**文本框是mysql 5.1的配置**

**mysql>stop slave; #停止slave**

**mysql> change master to master\_host='10.10.10.63',master\_user='slave',master\_password='123456';**

**mysql> start slave; #启动slave**

**mysql> show slave status\G 查看状态**

****

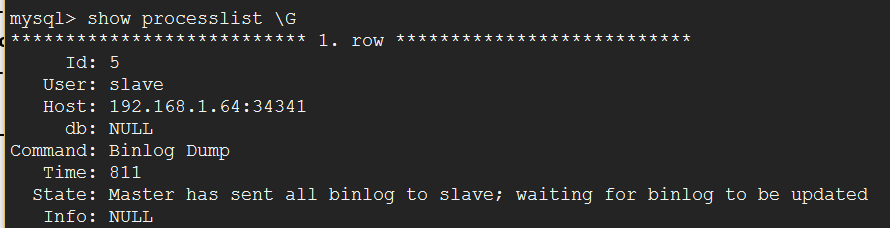
**Slave\_IO\_Running ：一个负责与主机的io通信**

**Slave\_SQL\_Running：负责自己的slave mysql进程**

**两个为YES 就成功了！**

**再到主服务器上查看状态：**

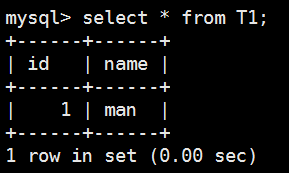
**mysql> show processlist \G**

****

## **插入数据测试同步：**

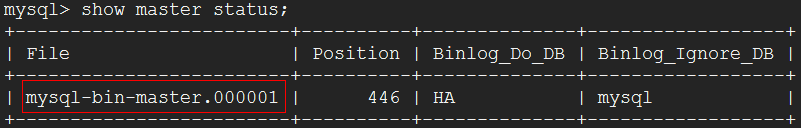
**mysql> insert into T1 values (1,'man');**

**从数据库上查看：**



## **排错：**

**如果遇到主从不同步，看一下主从bin-log的位置，然后再同步。**

****

**在主服务器上看二进制日志事件列表**

**mysql> show binlog events \G**

**从服务器执行MySQL命令下：**

**mysql> stop slave； #先停止slave服务**

**mysql> change master to master\_log\_file='mysql-bin-master.000001',master\_log\_pos=1164;**

**#根据上面主服务器的show master status的结果，进行从服务器的二进制数据库记录回归，达到同步的效果**

**mysql>slave start; #启动从服务器同步服务**

**mysql> show slave status\G; #用show slave status\G;看一下从服务器的同步情况**

**Slave\_IO\_Running: Yes**

**Slave\_SQL\_Running: Yes**

**如果都是yes，那代表已经在同步**

**重启从服务器，再查看状态：**

**停止从服务器slave stop;**

**开启从服务器slave start;**

**排错思路：**

**1、二进制日志没有开启**

**2、IPTABLES 没有放开端口**

**3、对应的主机 IP地址写错了**

**SQL线程出错**

**1、主从服务器数据库结构不统一**

**出错后，数据少，可以手动解决创建插入，再更新slave状态。**

**注：如果主上误删除了。那么从上也就误删除了。 #因此主上要定期做mysqldump备份。**

# **三：部署MySQL主主双向主从复制 M-M(恢复快照再实验)**

**通过mysql主主：进行mysql 双向同步数据库HA的配置**

**mysql主：服务端：XUEGOD63 IP：10.10.10.63**

**mysql主：服务端：XUEGOD64 IP：10.10.10.64**

**如果没有恢复快照要先清空一下二进制日志，mysql> reset master**

**在上面主从的基础上进行配置**

## **配置XUEGOD63**

**它有两种身份：**

**身份1： XUEGOD64的主。 身份2： XUEGOD64的从。**

**vim /etc/my.cnf**

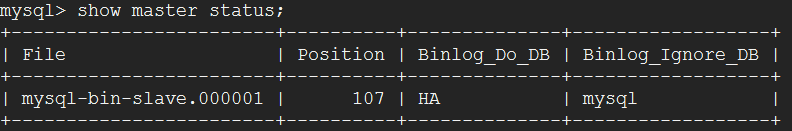
**server-id = 1**

**log-bin=mysql-bin-master**

**binlog-do-db=HA**

**binlog-ignore-db=mysql #避免同步mysql用户 相关配置**

**service mysqld restart**

****

**授权：**

**mysql> grant replication slave on \*.\* to slave@'10.10.10.%' identified by '123456';**

如果不符合密码要求修改如下（5.7版本）

mysql> set global validate\_password\_policy=**0**; #定义复杂度

mysql> set global validate\_password\_length=**1**; #定义长度 默认是8

关闭密码强度审计，在my.cnf添加validate-password=off

grant replication slave on \*.\* to slave@'10.10.10.%' identified by '123456';

**mysql> flush privileges; #记得刷新授权表**

## **配置XUEGOD64**

**5.1的配置**

**master-host=192.168.1.64 #指定主服务器IP地址**

**master-user=slave64 #指定定在主服务器上可以进行同步的用户名**

**master-password=123456 #密码**

**#master-port=3306 可以不写**

**replicate-do-db=HA #要复制的数据库**

**master-connect-retry=60 #断点重新连接时间**

**vim /etc/my.cnf**

**server-id = 2**

**log-bin=mysql-bin-slave**

**binlog-do-db=HA**

**binlog-ignore-db=mysql**

**先测试一下从帐号登陆是否正常？**

**[root@xuegod64 ~]# mysql -uslave -p123456 -h 10.10.10.63**

**然后作主授权给从xuegod63,**

**[root@xuegod64 ~]# grant replication slave on \*.\* to slave@'10.10.10.63' identified by '123456';**

**指定哪台是xuegod64的master**

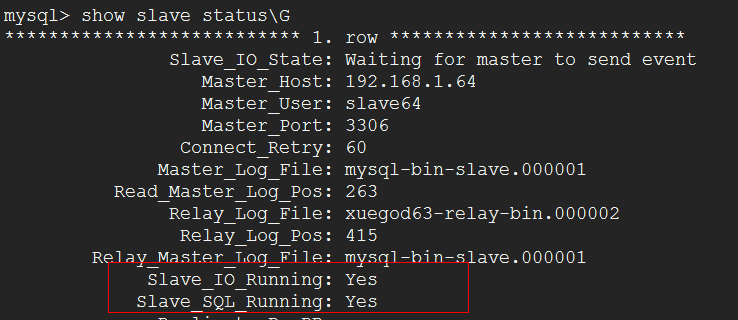
**[root@XUEGOD64 ~]# mysql -uroot -p123456**

**mysql> stop slave;**

**mysql> change master to master\_host='10.10.10.63' master\_user='slave',master\_password='123456';**

**mysql> start slave;**

**mysql> show slave status\G**

****

**再回来配置XUEGOD63**

**先测试一下从帐号登陆是否正常？**

**[root@xuegod63~]# mysql -uslave -p123456 -h 10.10.10.64 #可以登陆**

**指定哪台是xuegod63的master**

**[root@XUEGOD63 ~]# mysql -uroot -p123456**

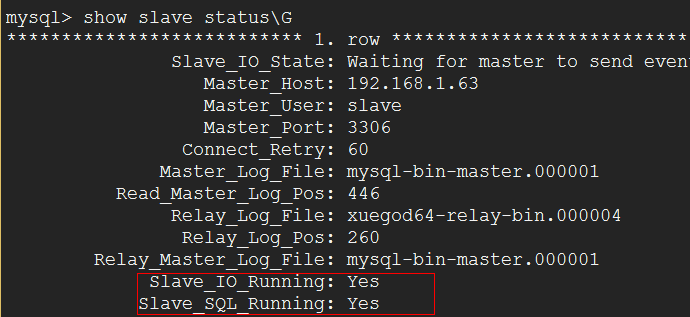
**mysql> change master to master\_host='10.10.10.64',master\_user='slave',master\_password='123456';**

**mysql> start slave;**

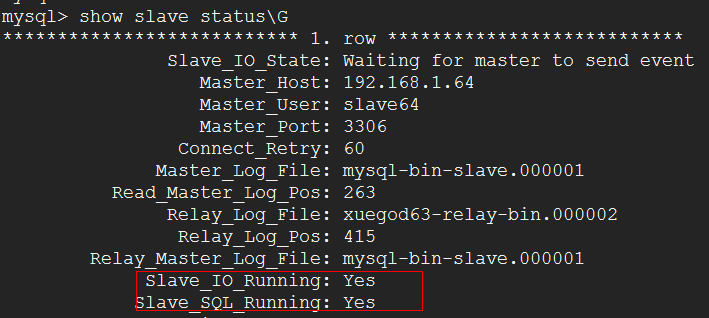
**mysql> show slave status\G**

## **检查：**

**XUEGOD63上查看slave状态**

****

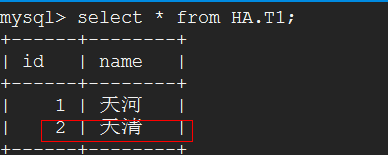
**XUEGOD64上查看slave状态**

****

**插入数据测试：**

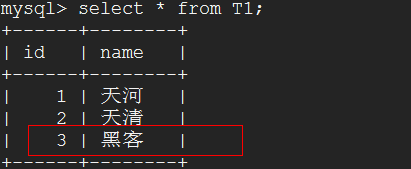
**在XUEGOD63上插入数据，XUEGOD64上查看**

**mysql> insert into T1 values(2,'天清');**

****

**在XUEGOD64上插入数据，XUEGOD63上查看**

**mysql> insert into T1 values(3,'黑客');**

****

**注意：这种M-M架构没有什么好处，他往往给我们产生一种实现了负载均衡的错觉**

# **六：部署M-S-S模型**

**环境：**

**XUEGOD68 master mysql5.7.20 10.10.10.68**

**XUEGOD69 slave中继 mysql5. 7.20 10.10.10.69**

**XUEGOD70 slave mysql5. 7.20 10.10.10.70**

## **部署master---->XUEGOD68:**

**在主服务上授权用户：**

**mysql>** **grant replication slave on \*.\* to repl@'10.10.10.%' identified by '123456';**

**mysql> flush privileges;**

**[root@xuegod68 ~]# vim /etc/my.cnf #修改配置 增加以下选项**

**server-id=1**

**binlog-do-db=HA**

**log-bin=mysql-bin-master**

**binlog-ignore-db=mysql**

**sync-binlog=1**

**binlog-format=row**

**重启服务**

**[root@xuegod68 ~]# systemctl restart mysqld**

**导出主服务器HA库完整备份， 拷贝到 中继服务器 和slave服务器**

**[root@xuegod68 ~]# mysqldump -uroot -p123456 -B HA>ha.sql**

**[root@xuegod68 ~]# scp ha.sql 10.10.10.69:/root**

**[root@xuegod68 ~]# scp ha.sql 10.10.10.70:/root**

## **部署slave中继------> XUEGOD69**

**导入数据库ha.sql**

**[root@xuegod69 ~]# mysql –uroot –p123456 <ha.sql**

**配置my.cnf**

**[root@xuegod69 ~]#vim /etc/my.cnf**

**server-id= 2**

**#修改主配置文件也要开启bin-log：**

**log-bin=mysql-bin-slave1**

**log-slave-updates=1 #把它从relay-log当中读取出来的二进制日志并且这本机上执行的操作也记录这自己的二进制日志里面，这样才能使第三台slave通过中继slave读取到相应数据变化**

**binlog-format=row**

**[root@xuegod69 ~]# systemctl restart mysqld**

**授权：**

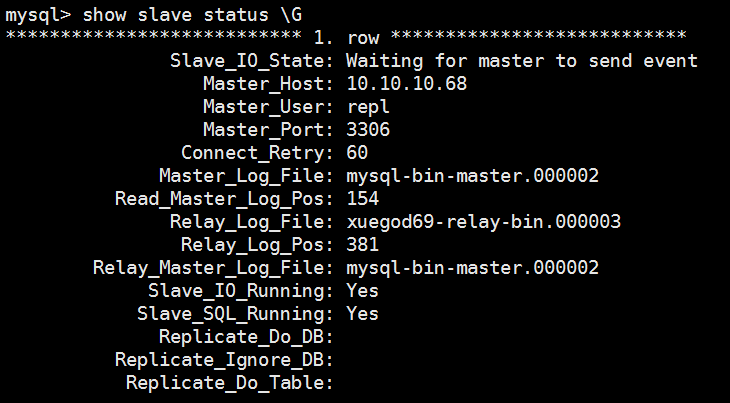
**mysql> stop slave;**

**mysql> change master to master\_host='10.10.10.68',master\_user='repl',master\_password='123456';**

**mysql> start slave;**

**查看中继服务的状态**

**mysql> show slave status \G**



**再授权一个用户给slave(xuegod70):**

**mysql> grant replication slave on \*.\* to 'repl'@'10.10.10.70' identified by '123456';**

**mysql> flush privileges;**

## **部署slave------>xuegod70**

**导入数据库ha.sql**

**[root@xuegod70 ~]# mysql –uroot –p123456 <ha.sql**

**配置my.cnf**

**server-id = 3**

**log-bin=mysql-bini-slave2**

**binlog-format=row**

**[root@xuegod70 ~]# systemctl restart mysqld 重启**

**指定slave中继服务作为xuegod70的主：**

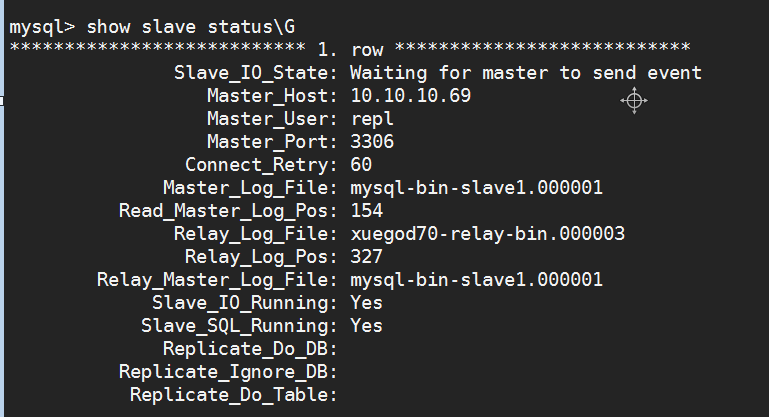
**mysql> stop slave;**

**mysql> change master to master\_host='10.10.10.69',master\_user='repl',master\_password='123456';**

**mysql> start slave;**

**查看从服务的状态**

**mysql> show slave status \G**

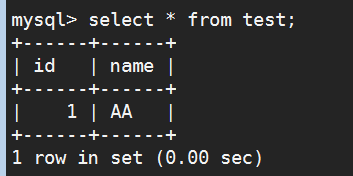


## **从master上插入数据测试：**

**mysql> insert into test values (1,'AA');**

**Query OK, 1 row affected (0.09 sec)**

**然后分别在slave中继，与slave上查看**



**为什么中继服务器也查到数据呢？ 因为是存储引擎的问题。**

**修改表的引擎：**

**mysql> alter table t1 ENGINE=blackhole; (先关闭日志记录再修改set sql\_log\_bin=off)**

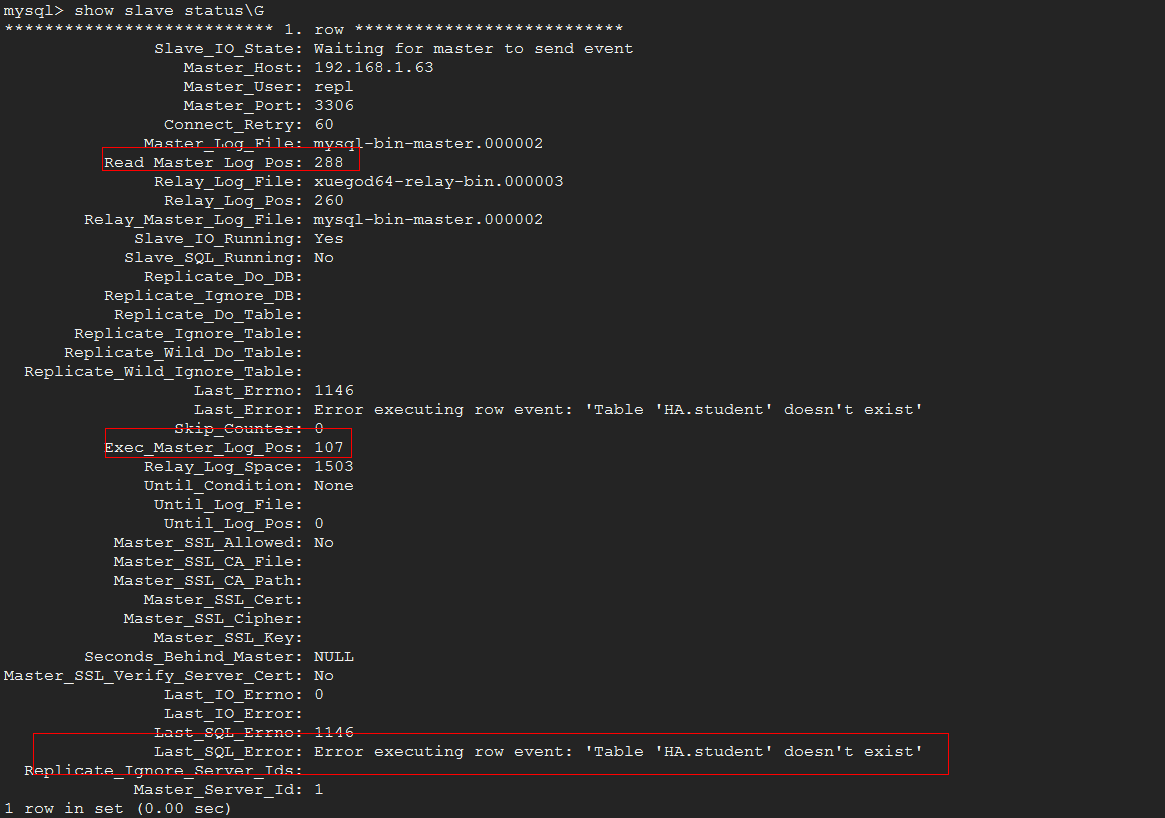
**你再测试！**

# **排错：**

## **错误1：**

**此方法可能失效，自行验证**

**mysql> show slave status\G**

****

**由结果可以看到：**

**Read\_Master\_Log\_Pos: 288**

**Exec\_Master\_Log\_Pos: 107**

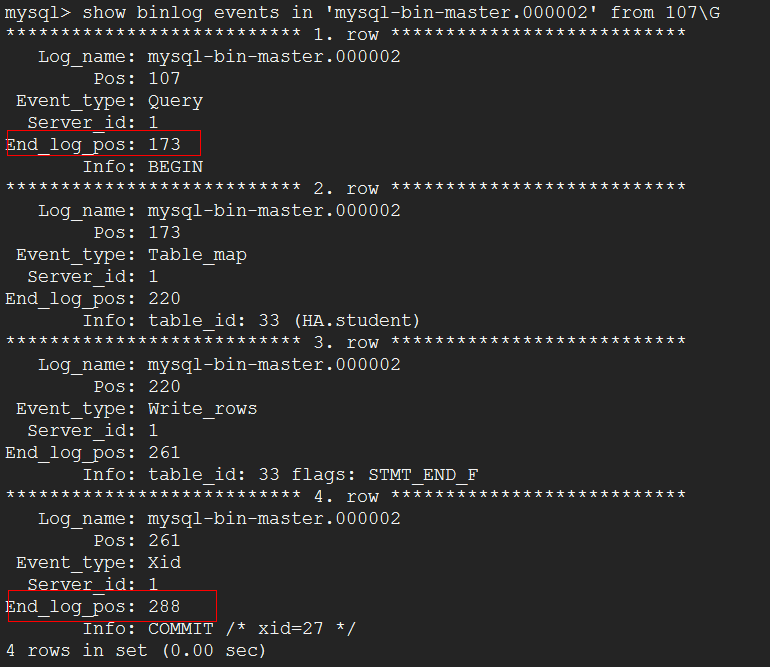
**Last\_SQL\_Errno: 1146**

**Last\_SQL\_Error: Error executing row event: 'Table 'HA.student' doesn't exist'**

**因为只对HA记录了binlog,当在mydb库操作其它数据库的表，但该表在slave上又不存在时就出错了。**

**到master上查看事件记录**

**mysql> show binlog events in 'mysql-bin-master.000002' from 107\G**

****

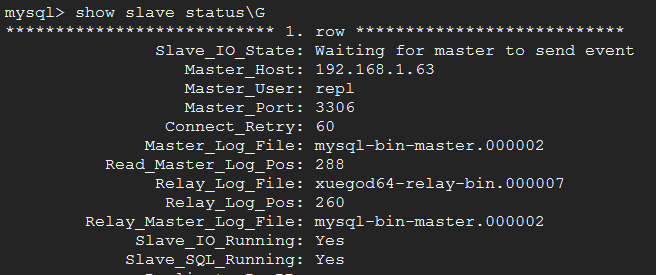
**由上面的结果可知，我们需要跳过两个事务173，288**

**然后到salve中继操作：**

**mysql> slave stop;**

**mysql> SET GLOBAL SQL\_SLAVE\_SKIP\_COUNTER = 2;  跳过一个事务**

**mysql> slave start;**

****

**这个搞定**

**或者：**

**它提供了一个session粒度的选项，通过关闭这个选项可以不让主库将打开这个选项或关闭连接前的SQL语句写入binlog。**

**set sql\_log\_bin=off;**

**mysql>alter table aaa add column xxx int default 1 after yyy;**

## **模拟故障：**

**由于历史遗留问题，MySQL主从库的表结构不一致，主库的某个表tableA比从库表tableA少了一个字段**

**当尝试在主库上更改表结构时，这行alter语句会随着binlog同步到从库，如果从库执行这行语句时出错，主从同步线程就会自动停止，那样只能人为手动处理错误，然后再启动slave上的主从同步线程。场景大概是下面这个样子：**

**先在从库添加这个字段：**

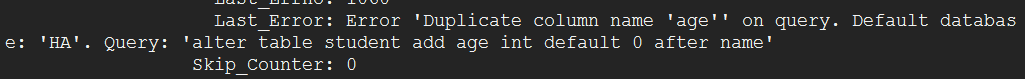
**mysql> alter table student add age int default 0 after name;**

**再在主库添加这个字段：**

**mysql> alter table student add age int default 0 after name;修改主库上的表结构，添加一个字段**

**从库会同步主库的，但是从库已经存在了这个字段**

**查看slave状态**

****

****

**解决方法1：**

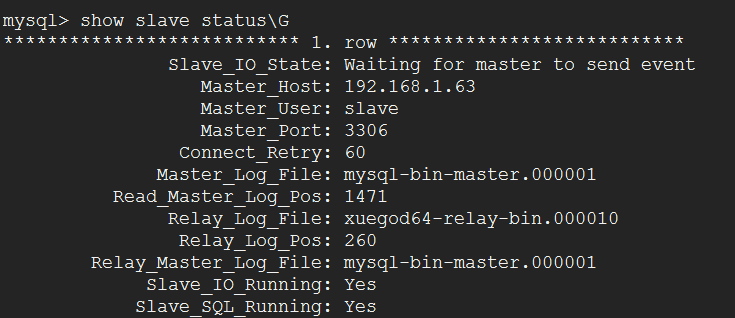
**跳过错误的事物**

**从库上执行：**

**mysql> stop slave;**

**set global sql\_slave\_skip\_counter=1;**

**mysql> start slave;**

**很多slave数据库的时候这样改太麻烦了**

**解决方法2:**

**slave比较少的时候还可以，但是当从库有几十台时，逐台去处理既费时又容易出错，怎样在主库这一侧一劳永逸地避免呢？**

**那很简单，我们不要让主库将alter语句记录到binlog中就行**

**我们直接这主库中关闭binlog记录**

**mysql> set sql\_log\_bin=off;**

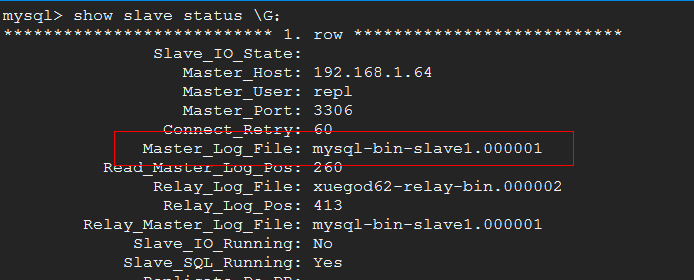
**然后我们再执行alter语句**

**mysql> alter table student add age int default 0 after name;**

**再开启bin-log**

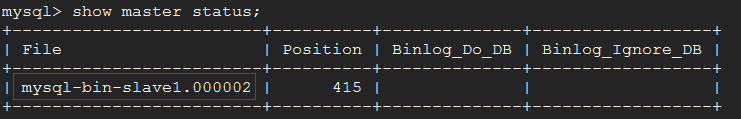
**mysql> set sql\_log\_bin=on;**

## **错误2：**

****

**这种要求对齐binlog**

**先到作为它的主上查看binlog**

****

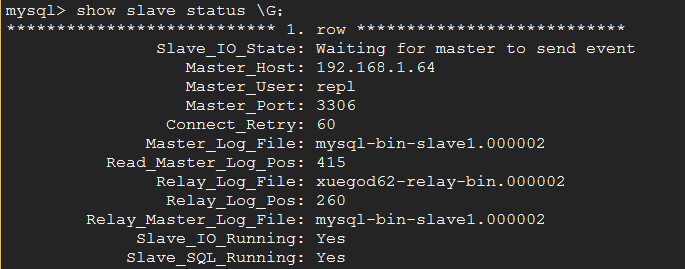
**Slave上不对应**

**Slave上操作：**

**mysql> stop slave;**

**mysql> change master to master\_host='10.10.10.65',master\_user='repl',master\_password='123456',master\_log\_file='mysql-bin-slave1.000002',master\_log\_pos=415;**

**mysql> start slave;**

****

**Ok,恢复正常**