**数据库主从复制**

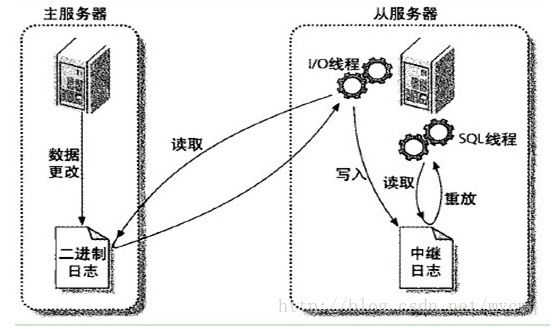
提示：红色字体应根据实际情况做出修改

# 作用

MySQL数据库自身提供的主从复制功能可以方便的实现**数据的多处自动备份**，实现**数据库的拓展**。多个数据备份不仅可以**加强数据的安全性**，通过实现读写分离还能进一步**提升数据库的负载性能**。

# 原理

MySQL之间数据复制的基础是**二进制日志文件**（binary log filej）。一台MySQL数据库一旦启用二进制日志后，其作为**Master**，它的数据库中所有操作都会以“事件”的方式记录在二进制日志中，其它数据库作为**Slave**通过**一个I/O线程**与Master保持通信，并监控Master的二进制日志文件的变化，一旦发现Master二进制日志文件发生变化，则会把变化复制到自己的**中继日志**中，然后Slave的**一个SQL线程**就会把相关的“事件”执行到自己的数据库中，以此实现数据库和主数据库的一致性，也就实现了主从复制。



# 实现

## 准备工作

主从数据库版本最好一致(MariaDB-10.1.20-linux-x86\_64)

主数据库Master：192.168.152.139 linux

从数据库Slave： 192.168.152.140 linux

## 配置Master

### 修改mysql配置【开启日志文件，设置server-id】

#### 找到配置文件my.cnf

**[shell ]# whereis my.cnf**



#### 打开配置文件my.cnf

**[shell ]# vim /etc/my.cnf**

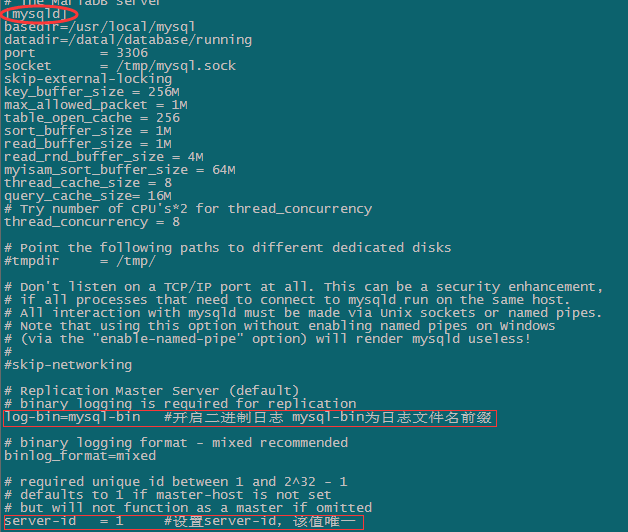


#### 在[mysqld]部分修改（添加）：

[mysqld]

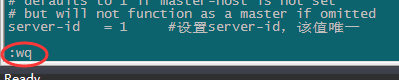
log-bin=mysql-bin #开启二进制日志，mysql-bin为日志文件名前缀

server-id=1 #设置server-id，该值唯一



#### 保存配置文件

:wq



### 重启mysql

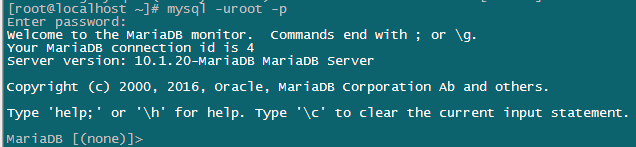
若没有修改MySQL配置文件，则无需重启。

**[shell ]# service mysqld restart**



### 创建用户账号

#### 登录数据库



#### 创建用户

\*该账号给Slave使用（用于同步）

\*用户名：SlaveName 密码：slavepass

\*从服务器IP：192.168.152.140

**mysql > CREATE USER ‘SlaveName’@’192.168.152.140’ IDENTIFIED BY “slavepass”;**



#### 分配权限

\*分配的权限：REPLICATION SLAVE（复制权限）

\*可操作的数据库：\*.\*（所有数据库）

\*用户名：SlaveName

\*密码：slavepass

\*所要授权的远程IP地址：192.168.152.140

**mysql > GRANT REPLICATION SLAVE ON \*.\* TO 'SlaveName'@'192.168.152.140' IDENTIFIED BY "slavepass";**



#### 刷新权限

**mysql > FLUSH PRIVILEGES;**



### 数据库备份

备份指定数据库：

**[shell]# mysqldump -uroot -p -R --master-data=2 --flush-logs --databases ForwardSystem > backup.sql**



**默认操作：**

--add-drop-table(创建表时先判断表是否存在，若存在则删除表)；

--add-locks(往表中导入数据前上锁，导入完成后解锁)；

--create-options(在创建表时包括所有MySQL特性选项)；

--quick(不缓冲查询，直接导出到标准输出)；

--extended-insert(使用具有多个VALUES列的INSERT语法，这样使得导出文件跟小，并加速导入时的速度)；

--set-charset(添加SET NAMES default\_character\_set到输出文件)；

--disable-keys(对于每个表，用/\*!40000 ALTER TABLE tbl\_name DISABLE KEYS \*/;和/\*!40000 ALTER TABLE tbl\_name ENABLE KEYS \*/;语句引用INSERT语句。这样可以更快地导入dump出来的文件，因为它是在插入所有行后创建索引的。该选项只适合MyISAM表)；

--triggers(导出触发器);

**-u：**用户名root

**-p：**密码

**-R：**导出存储过程以及自定义函数

**--master-data：**该选项将设置全局读锁，锁定所有数据库中的所有表，保证数据的一致性；并且将二进制日志文件的当前位置（Position）和文件名追加到输出文件中。

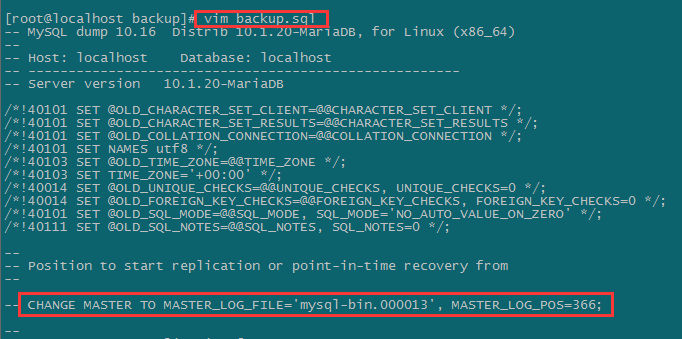
**--flush-logs：**刷新二进制日志，也就是重新生成一个二进制日志文件。

**--databases：**要备份的数据库名ForwardSystem；**若导出全部数据库，该项改为--all-databases**

**backup.sql：**备份文件名

### 记录Master状态

打开刚生成的备份文件backup.sql，记录MASTER\_LOG\_FILE和MASTER\_LOG\_POS



\*二进制文件名：mysql-bin.0000013

\*位置：366

## 配置Slave

### 3.1. 修改mysql配置

同样找到my.cnf配置文件，添加server-id（该值必须唯一）

[mysqld]

server-id=140 #设置server-id，必须唯一



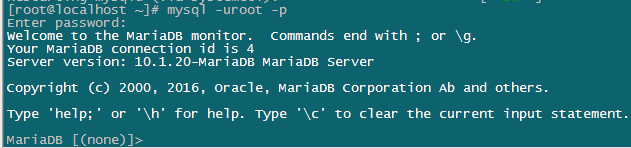
### 3.2. 重启mysql

**[shell ]# service mysqld restart**



### 3.3. 设置Slave

#### 3.3.1. 登录MySQL



#### 3.3.2. 导入备份

将Master生成的备份文件backup.sql导入Slave：

**mysql > source backup.sql;**



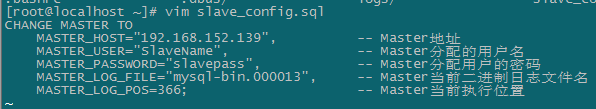
#### 3.3.3. 关闭Slave同步进程

**mysql > stop slave;**



#### 3.3.4. 配置Slave

**[shell ]# vim slave\_config.sql**



**mysql > source slave\_config.sql;**



#### 3.3.5. 开启Slave同步进程

**mysql > start slave;**



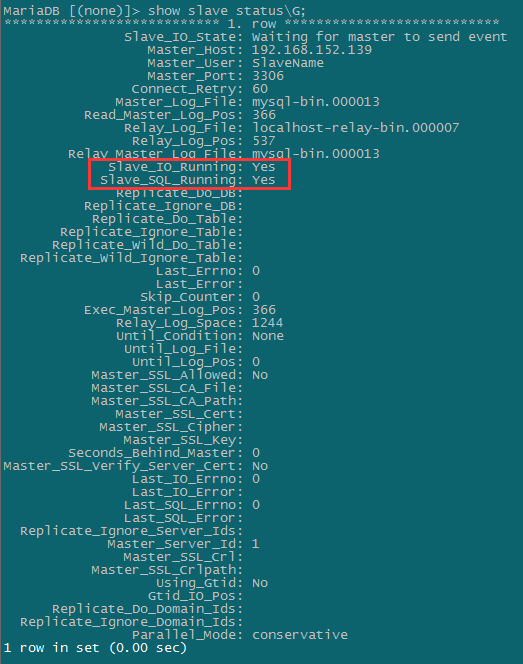
#### 3.3.6. 查看Slave状态

**mysql > show slave status\G;**

\*Slave\_IO\_Running：监控Master二进制文件变化的进程

\*Slave\_SQL\_Running：同步Master变化的进程

当Slave\_IO\_Running和Slave\_SQL\_Running都为YES时就表示主从同步设置成功。



## 注意事项

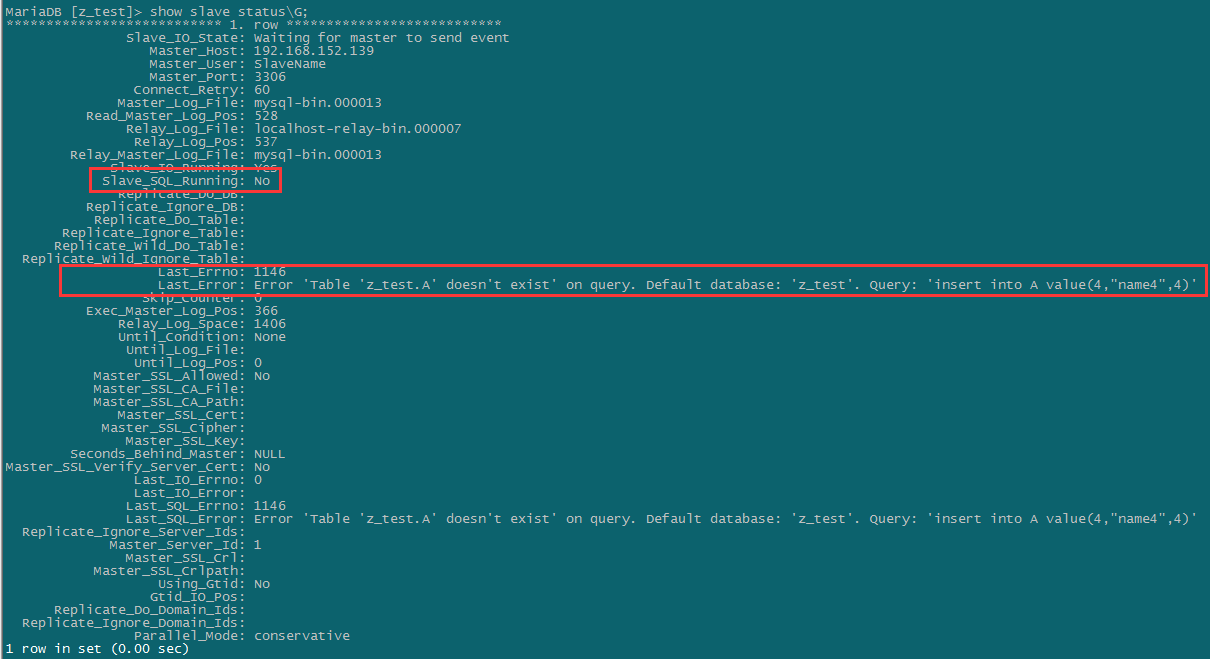
1. Master数据库每次重启或执行FLUSH LOGS命令时，都会重新生成一个二进制日志文件，这时，Slave的配置就需要跟着修改。
2. Master和Slave配置文件中的server-id需要不同。
3. Master的数据操作请通过sql完成，否则将造成主从库数据不一致（导致主从复制失败）。
4. 主从复制出错的原因，基本上都是主库和从库数据不一致。所以，针对核心表，可以定制自动数据校验脚本，定期进行数据校验。

## 常见问题

### 场景一：操作了在Slave中不存在的数据库/表

#### 错误现象

【ERROR：1146】



#### 错误原因

对数据的操作不是通过执行sql，而是通过copy/rm系统命令，因此该过程没有写入bin-log，Slave也就不会有相应的同步；或是在从库中删除了数据库/表。

#### 解决方案

在Slave中手动同步数据：

\*停止Slave

**mysql > stop slave;**



\*手动同步数据

**mysql > source A\_table.sql;**



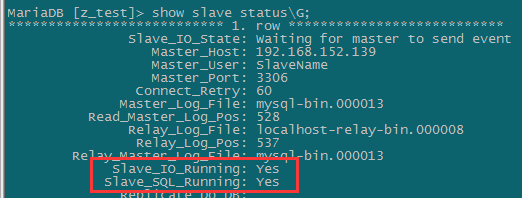
\*启动Slave

**mysql > start slave;**



\*查看Slave状态，当Slave\_IO\_Running和Slave\_SQL\_Running都为YES时就表示主从复制恢复正常。

**mysql > show slave status\G;**



之后，Master的数据操作请通过sql完成，避免该场景的出现。

### 场景二：主键重复，发生唯一性冲突

#### 错误现象

【ERROR：1062】



#### 错误原因

在Slave中已有该记录，又在Master上插入了同一条记录

#### 解决方案

##### 方法一：在Slave将重复的主键记录删除，然后重启Slave

\*出错表：A

\*出错主键：no=3

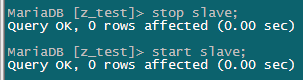
**mysql > DELETE FROM A where no=3;**



\*重启Slave

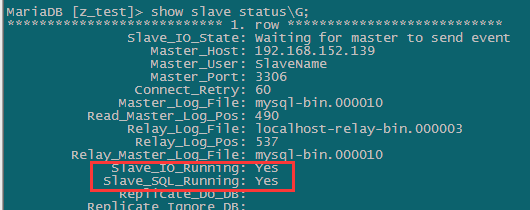
**mysql > stop slave;**

**mysql > start slave;**



\*查看Slave状态当Slave\_IO\_Running和Slave\_SQL\_Running都为YES时就表示主从复制恢复正常

**mysql > show slave status\G;**



##### 方法二：停止Slave，忽略一次错误，再开启Slave

\*停止Slave

**mysql > stop slave;**

\*忽略一次错误

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

\*开启Slave

**mysql > start slave;**

### 场景三：删除或更新数据失败

（数据库版本：mariadb-10.1.20-linux-x86\_64，验证出现此问题不报错）

#### 错误现象

【ERROR】1032：删除或更新数据，从库找不到记录

#### 错误原因

在Master上删除或更新一条记录，而Slave上找不到该记录

#### 解决方案

##### 更新数据

在从库添加相同的数据，再开启复制恢复SQL线程。

##### 删除数据

由于Master要删除一条记录，而Slave上找不到所以报错，这种情况主库都删除该记录了，从库可以直接跳过。

\*停止Slave

**mysql > stop slave;**

\*跳过一次错误

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

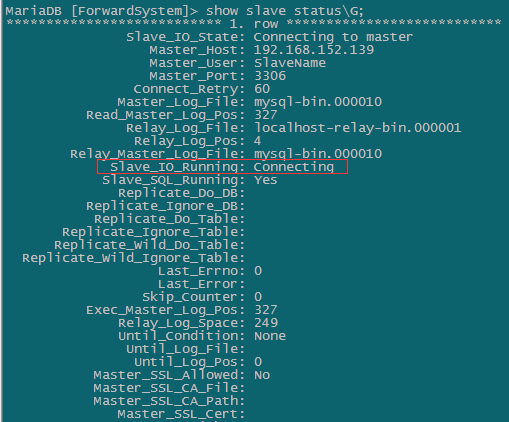
\*启动Slave

**mysql > start slave;**

### 场景四：Slave连接Master失败

#### 错误现象

**Slave\_IO\_Running: Connecting**



#### 错误原因

连接不上Master，多数是因为有防火墙限制。

#### 解决方案

关闭Master的防火墙

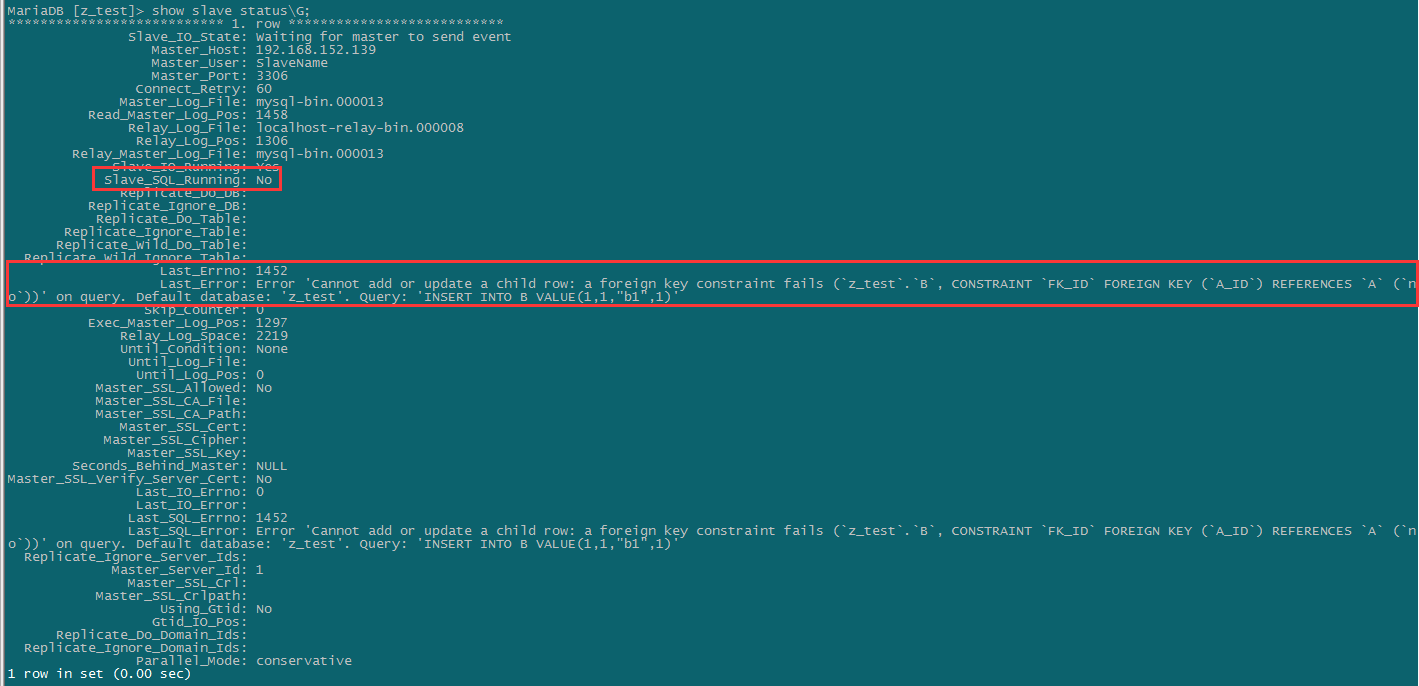
**[shell ]# iptables –F**



### 场景五：无法在有外键的表插入参考主键没有的数据

#### 错误现象

【Error：1452】



#### 错误原因

由于B.A\_ID字段(外键)参考了A.no字段(主键)，当要在B表插入数据时，如果A表的主键没有对应的数据，则无法插入。

#### 解决方案

根据从库的报错信息，在主库查询参考表的数据，手动插入到从库，再重启Slave。

##### 1. Slave操作

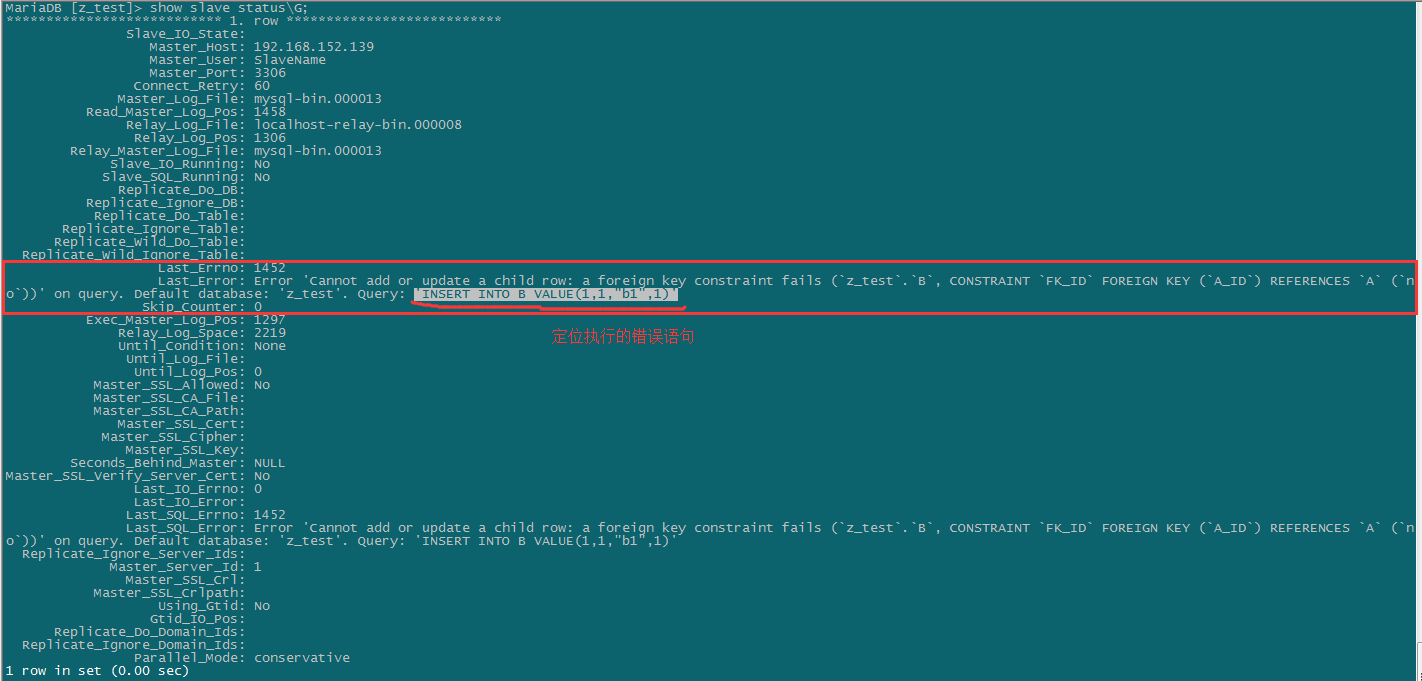
停止Slave

**mysql > stop slave;**



查看出错位置

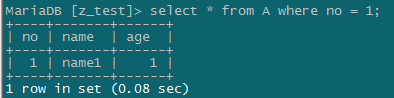
**mysql > show slave status\G;**



##### Master操作

根据Slave报错信息查询参照表A的信息

**mysql > select \* from A where no = 1;**



##### Slave操作

将Master的查询结果插入到Slave的参照表中

**mysql > insert into A value(1,"name1",1);**



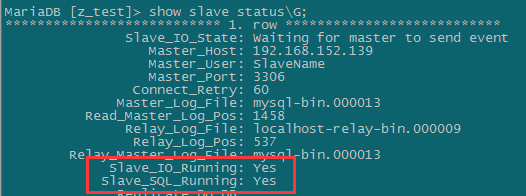
开启Slave

**mysql > start slave;**



查看Slave状态当Slave\_IO\_Running和Slave\_SQL\_Running都为YES时就表示主从复制恢复正常

**mysql > show slave status\G;**



### 其它

#### 配置文件my.cnf中主从server-id一致

**报错：**Last\_IO\_Error: Fatal error: The slave I/O thread stops because master and slave have equal MySQL server ids; these ids must be different for replication to work (or the –replicate-same-server-id option must be used on slave but this does not always make sense; please check the manual before using it).

**分析：**在mysql主从配置中，每台mysql数据库的my.cnf中的server-id必须是唯一，但有时候可能因为粗心配成了相同的数值，也有可能mysql没有加载到my.cnf文件中的server-id。

**解决：**修改从库配置文件中的server-id，使之与主库不一致；重启主从复制。

#### 主库重启（数据库服务器宕机）

**报错：**Last\_IO\_Error: Got fatal error 1236 from master when reading data from binary log: ‘binlog truncated in the middle of event; consider out of disk space on master; the first event ‘mysql-bin.001989’ at 9179, the last event read from ‘./mysql-bin.001989’ at 9179, the last byte read from ‘./mysql-bin.001989’ at 9179.’

**分析：**由报错可看出是由于从库的二进制日志文件位置与主库的不一致导致的。

**解决：**[重新配置从库Slave](#_3.3.4._配置Slave)。

#### 从库表字段、不够长等

**报错：**【Last\_Errno：1264】字段不够长；【Last\_Errno：1054】从库字段丢失

**解决：**修改表结构，重启主从。

#### binlog pos记录不一致

**报错：**【Last\_Errno：1593】【Last\_Errno：1594】

**分析：**SLAVE在宕机，或者非法关机造成中继日志损坏，同步停掉；主数据库突然停止或问题终止，更改了mysql-bin.xxxx日志，slave服务器找不到这个文件。

**解决：**[在主库上找到同步的binlog和pos点](#_记录Master状态)，然后[重新配置Slave](#_3.3.4._配置Slave)，重启主从。

#### 隐藏问题：主从时区不一致导致主从数据不一致

#### 临时表太大导致磁盘写满

**报错：**【Last\_Errno：3】

**分析：**1.tmp目录不可写，或磁盘没有空间；2.tmp还有空间，但是原表太大，所以查询时生成的临时表过大，因此出错。

**解决：**1.确认/tmp目录可写入，同时磁盘未写满；2.修改socket目录到空间较大的分区，在重启实例。

## 修复主从数据不一致

使用pt-table-checksum和pt-table-sync彻底修复数据不一致。

pt工具包安装：<https://www.cnblogs.com/zishengY/p/6852280.html>

### 在从库停止主从复制

**mysql > stop slave;**



### 在主库创建校验账号

**GRANT UPDATE,INSERT,DELETE,SELECT,PROCESS,SUPER,REPLICATION SLAVE ON \*.\* TO 'checksums'@'192.168.152.139' IDENTIFIED BY 'checksums';**



**GRANT ALL ON test.\* TO 'checksums'@'192.168.152.139' IDENTIFIED BY 'checksums';**



### 在主库创建校验信息表

**mysql > use test;**

**mysql > CREATE TABLE IF NOT EXISTS checksums(**

**db char(64) NOT NULL,**

**tbl char(64) NOT NULL,**

**chunk int NOT NULL,**

**chunk\_time float NULL,**

**chunk\_index varchar(200) NULL,**

**lower\_boundary text NULL,**

**upper\_boundary text NULL,**

**this\_crc char(40) NOT NULL,**

**this\_cnt int NOT NULL,**

**master\_crc char(40) NULL,**

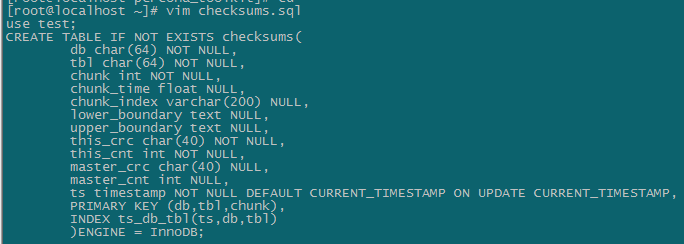
**master\_cnt int NULL,**

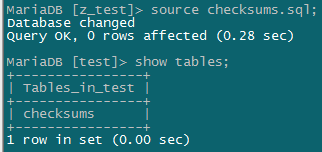
**ts timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,**

**PRIMARY KEY (db,tbl,chunk),**

**INDEX ts\_db\_tbl(ts,db,tbl)**

**)ENGINE = InnoDB;**





### 在从库创建校验账号，与主库创建的账号需一致

**GRANT UPDATE,INSERT,DELETE,SELECT,PROCESS,SUPER,REPLICATION SLAVE ON \*.\* TO 'checksums'@'192.168.152.139' IDENTIFIED BY 'checksums';**



### 在从库开启主从复制

**mysql > start slave;**



### 在主库用pt-table-checksum校验主从数据一致性

**参数意义：**

**--nocheck-binlong-format：**不检查复制的binlog模式，要是binlog模式是ROW，则会报错。

**--nocheck-replication-filters：**不检查复制过滤器。

**--replicate：**把checksum的信息写入到检验信息表中。

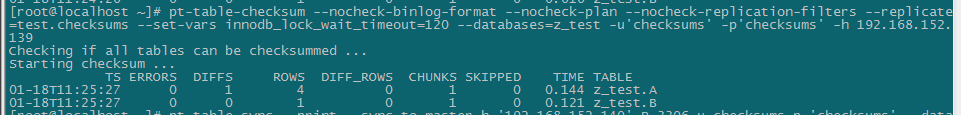
**--databases：**指定需要被检查的数据库，多个则用逗号隔开。

**u=checksums：**用户名。

**p=checksums：**密码。

**h=192.168.152.169：**主库IP地址。

**[shell]# pt-table-checksum --nocheck-binlog-format --nocheck-plan --nocheck-replication-filters --replicate=test.checksums --set-vars innodb\_lock\_wait\_timeout=120 --databases=z\_test -u'checksums' -p'checksums' -h 192.168.152.139**



**参数说明：**

\*TS：完成检查的时间。

\*ERRORS：检查时候发生错误和警告的数量。

\*DIFFS：0表示一致，1表示不一致。

\*ROWS：表的行数。

\*CHUNKS：被划分到表中的块的数目。

\*SKIPPED：由于错误或警告或过大，则跳过块的数目。

\*TIME：执行的时间。

\*TABLE：被检查的表名。

### 在主库用pt-table-sync打印出修复不一致数据的SQL

如果有外键约束，修复数据应先从外键参考的字段所属表开始修复

**参数意义：**

**--print：**将不一致的数据打印到指定输出文件。

**--sync-to-master：**指定一个DSN，即从库的IP。

**h=192.168.152.140：**从库地址

**P=3306：**端口号

**u=checksums：**用户名

**p=checksums：**密码

**--databases=z\_test：**指定执行同步的数据库，多个用逗号隔开。

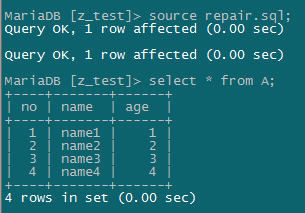
**repair.sql：**打印出来的输出文件。

**[shell]# pt-table-sync --print --sync-to-master h='192.168.152.140',P=3306,u=checksums,p='checksums' --databases=z\_test > repair.sql**



### 在从库执行修复数据的SQL

**mysql > source repair.sql;**



### 注意事项

\*测试需要一个既能登录主库，也能登录从库，而且还能同步数据库的账号。

\*只能指定一个host，必须为主库的IP。

\*在检查时会向表加S锁。

\*运行pt-table-checksum前需要在从库开启主从复制。

\*pt-table-sync不同步表结构、索引、或任何其它模式对象，所以在修复一致性之前需要保证他们的表存在。

\*该工具执行检查表动作，检查连接的账号需要有很高的权限，在一般权限上需要加SELECT、PROCESS、SUPER、REPLICATION SLAVE等权限。

\*pt-table-checksum配合pt-table-sync使用，在执行pt-table-sync数据同步之前，一定要执行pt-table-checksum命令检查。

\*执行pt-table-checksum时可能出现”Diffs cannot be detected because no slaves were found.Please read the --recurision-method documentation for information.”从字面意思上看是，主库找不到从库：

1.确认从库开启了主从复制。

2.在从库配置文件my.cnf中添加: report\_host=slave\_ip report\_port=slave\_port。

3.也可以在pt-table-checksum –recursion-method=hosts

## 主从复制出错万金油

### 方法一：忽略错误，继续同步

该方法适用于主从数据相差不大，或者要求数据可以不完全统一的情况，数据要求不严格的情况

#### 停止Slave

**mysql > STOP SLAVE;**

#### 跳过错误

**mysql > SET GLOBAL sql\_slave\_skip\_counter =1;**

\*后面的数字可变：

sql\_slave\_skip\_counter以event为单位skip，直到skip完第N个event所在的event group才停止。对于事务表，一个event group对应一个事务；对于非事务表，一个event group对应一条SQL语句。一个event group包含多个events。使用sql\_slave\_skip\_counter有可能跳过错误以外的其他events，导致主从数据不一致。

#### 启动Slave

**mysql > START SLAVE;**

#### 查看Slave状态

**mysql > SHOW SLAVE STATUS\G;**

当Slave\_IO\_Running和Slave\_SQL\_Running都为**YES**时就表示主从同步正常了。

### 方法二：重新做主从，完全同步

该方法适用于主从库数据相差较大，或者要求数据完全统一的情况。

#### Master操作

##### 锁表

此处锁定为只读状态，防止数据写入

**mysql > FLUSH TABLES WITH READ LOCK;**



##### 数据备份

把数据备份为.sql的文件

[shell ]# mysqldump -uroot -p --databases database\_name > mysql\_back.sql

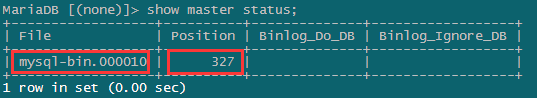
##### 记录Master状态

在主库执行：

**mysql > SHOW MASTER STATUS;**

\*记下File：mysql-bin.000010

\*记下Position：327



##### 解锁

**mysql > UNLOCK TABLES;**



#### 传输备份文件

将Master生成的备份文件mysql\_back.sql传输到Slave

#### Slave操作

##### 停止Slave同步进程

**mysql > STOP SLAVE;**

##### 清除Slave

删除所有旧的同步日志，使用新的日志重新开始

**mysql > RESET SLAVE;**



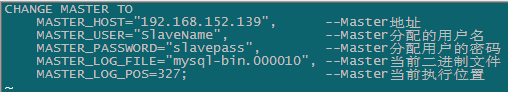
##### 导入备份数据

**mysql > source mysql\_back.sql;**



##### 配置Slave

**[shell ]# vim slave\_config.sql**



**mysql > source slave\_config.sql;**



##### 开启Slave同步进程

**mysql > START SLAVE;**

##### 查看Slave状态

**mysql > SHOW SLAVE STATUS\G;**

当Slave\_IO\_Running和Slave\_SQL\_Running都为**YES**时就表示主从同步设置成功。