1.mysql主从复制(主从同步及主备)及分区技术(插件技术)

2.用户授权和bin-log日志(排错其他做准备)

lamp架构:Linux--apache--mysql--php/python

集群(冗余) 主从(负载均衡)

3.给从服务器设置授权用户

主从复制需要进行的配置.

主服务器: 开启二进制日志

配置唯一的server-id

获得master二进制日志文件名及位置

创建一个用于slave和master通信的用户账号

从服务器

配置唯一的server-id

使用master分配的用户账号去读master二进制文件

启用slave服务

创建一个用户,并赋予其所有master所有库和表的权限,刷新权限

show master status;

执行同步sql语句,

CREATE USER 'repl'@'123.57.44.85' IDENTIFIED BY 'slavepass';#创建用户

GRANT REPLICATION SLAVE ON \*.\* TO 'repl'@'123.57.44.85';#分配权限

flush privileges; #刷新权限

创建完用户后,需要获取master二进制文件的日志名称和位置

show master status;

执行同步sql语句,

CHANGE MASTER TO

MASTER\_HOST='192.168.28.133',

MASTER\_USER='liutao',

MASTER\_PASSWORD='root',

MASTER\_LOG\_FILE='mysql-bin.000001',

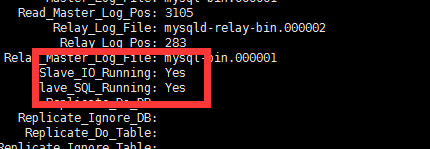
MASTER\_LOG\_POS=4259;

启动slave同步进程.

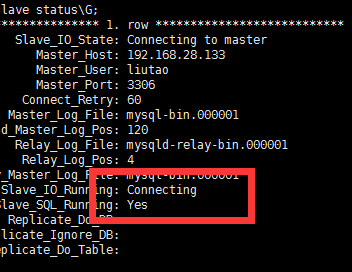
start slave; 开启同步数据

查看slave状态,

当slave\_io\_running和slave\_sql\_running都为yes表示主从同步设置成功了



mysql5.6有设置uuid,必须不一致,在 /var/lib/mysql/data/auto.cnf中



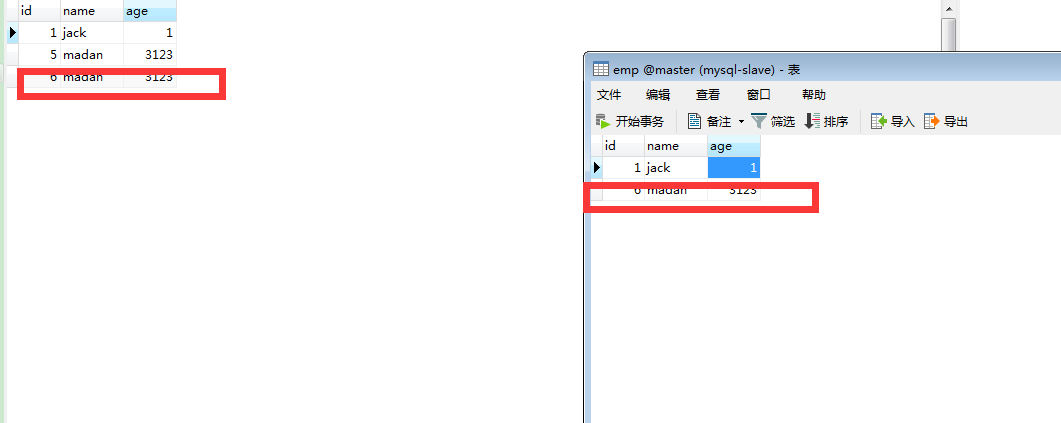
此处,io\_running为connecting 表示未连接上,

接下来需要验证是否同步

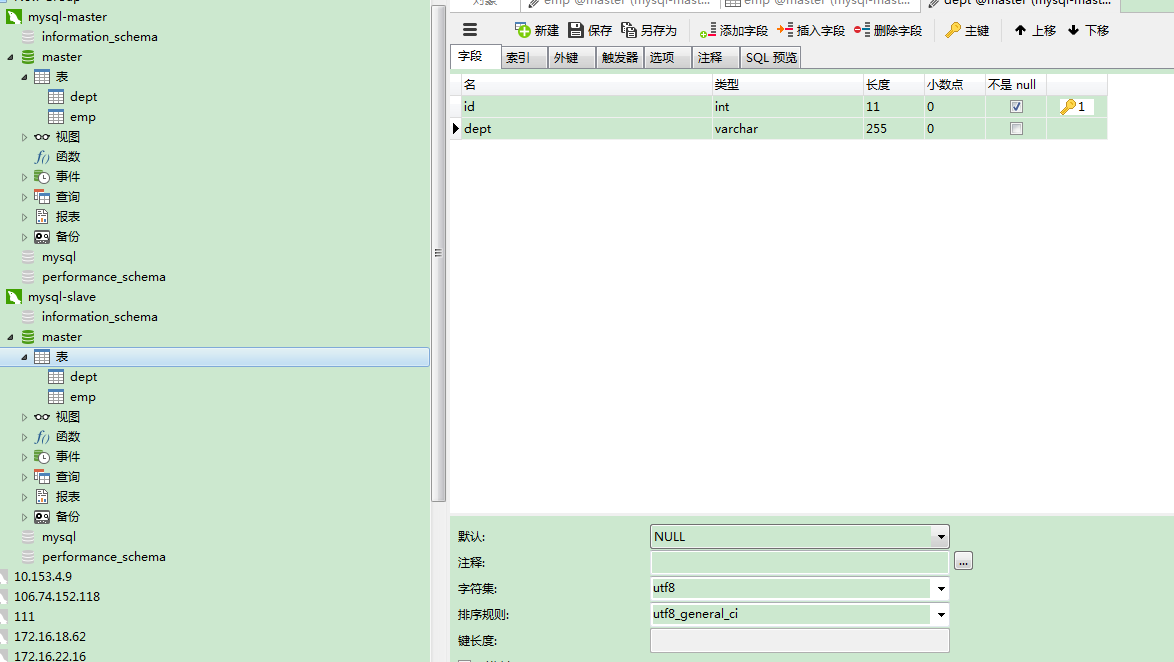
grant all privileges on \*.\* to root@'%' identified by "root";

flush privileges;

可以配置那些数据库需要同步,那些数据库不需要同步



完成数据的同步



新创建的表的同步

同步表的问题

==================================================

存储过程生成1w条数据

**DROP PROCEDURE IF EXISTS proc\_initData;**

**DELIMITER $**

**CREATE PROCEDURE proc\_initData()**

**BEGIN**

**DECLARE i INT DEFAULT 15;**

**WHILE i<=10000 DO**

**INSERT INTO emp VALUES(i,"name",i);**

**SET i = i+1;**

**END WHILE;**

**END $**

**CALL proc\_initData();**

分表和分区

分表:按照一定规则,将一张表分解成多张不同的表,将用户订单表根据时间分成多个表,分区与分表的区别在于;分区从逻辑上只有一张表,分表则是将一张表切分成多张表.

mysql数据库中的数据是以文件的形式存储在磁盘的,一张表对应着三个文件,

frm 存放表结构

myd 存放表数据

myi 存放表数据

如果文件很大,我们在一个磁盘切分,或者在多个磁盘上存储数据.

--------------------------------------------------

分区的两种方式:

横向分区:横着来分区,用merge来分表,并没有改变表的结构

纵向分区:竖着分,将某些不常用的字段单独提取出来.

mysql提供的分区属于横向分区,

mysql5.1 以上支持分区

查看是否支持分区

show variables like "%part%";

或者命令 显示插件 show plugins;



--=--=--=--=--=--=-=-=-=-=-=-=-=--=-=-==-=-=--=-=-=-=-==-=-

分区技术

range 分区: 这种模式允许将数据划分不同范围,可以将通过年份划分成若干区.

create table id\_range(

id int (11) not null primary key,

money int(11) unsigned not null

) partition by range(id)(

partition t1000 values less than(1000),

partition t2000 values less than(2000),

partition t3000 values less than(3000),

partition t4000 values less than(4000),

partition t5000 values less than(5000),

partition t6000 values less than(6000),

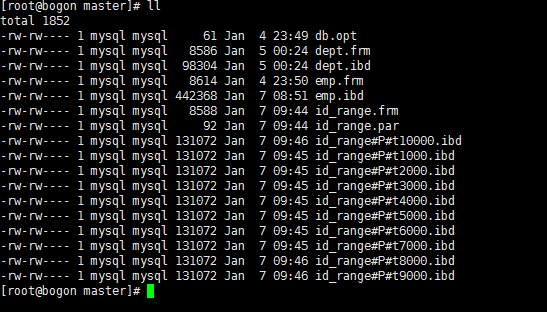
partition t7000 values less than(7000),

partition t8000 values less than(8000),

partition t9000 values less than(9000),

partition t10000 values less than(10000),

);

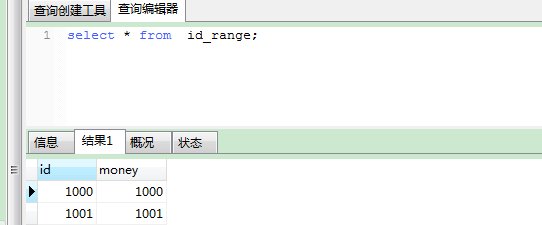


正常使用innodb 无法显示 \*.MYD 和 \*.MYI文件,需要将数据库引擎改为myISAM

适用的场合:

1.需要删除分区上的旧数据时,只需要删除分区即可, 简单使用

alter table id\_range drop partition t1000;



直接删除掉某个分区,数据直接删除

分区的理解:分区只是在数据库的底层上进行了切分,在使用或者可视化查看并没有直接的显示,在创建分区时,本身就是根据某个条件创建的,如果涉及到这个条件会直接先查询分区信息

2.想要使用一个包含有日期或者时间值,或包含从其他一些级数开始增长的值得列,作为分区的条件.

3.经常使用用于分区的那个列作为条件的查询

查询分区的信息.

SELECT \* FROM id\_range PARTITION (t10000);

list分区:这种模式允许系统通过预定义的列表的值来对数据进行分割.

create table t\_list(

　　a int(11),

　　b int(11)

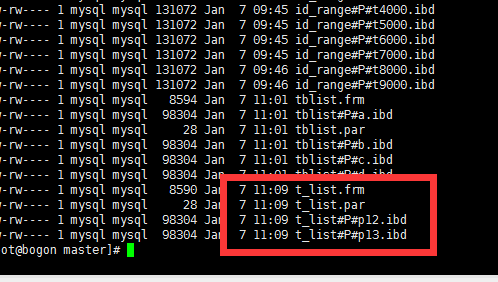
　　)

partition by list(b) (

　　partition p0 values in (1,3,5,7,9),

　　partition p1 values in (2,4,6,8,0)

);



//此处不清楚为什么关键字没有变色,

存在空格无法将数据作为分区字段的问题

CREATE TABLE tblist (

id INT NOT NULL,

store\_id INT

)

PARTITION BY LIST(store\_id) (

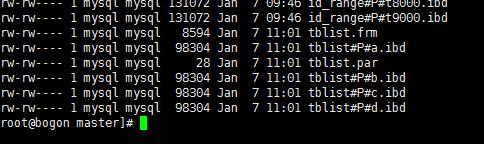
PARTITION a VALUES IN (1,5,6),

PARTITION b VALUES IN (2,7,8),

PARTITION c VALUES IN (3,9,10),

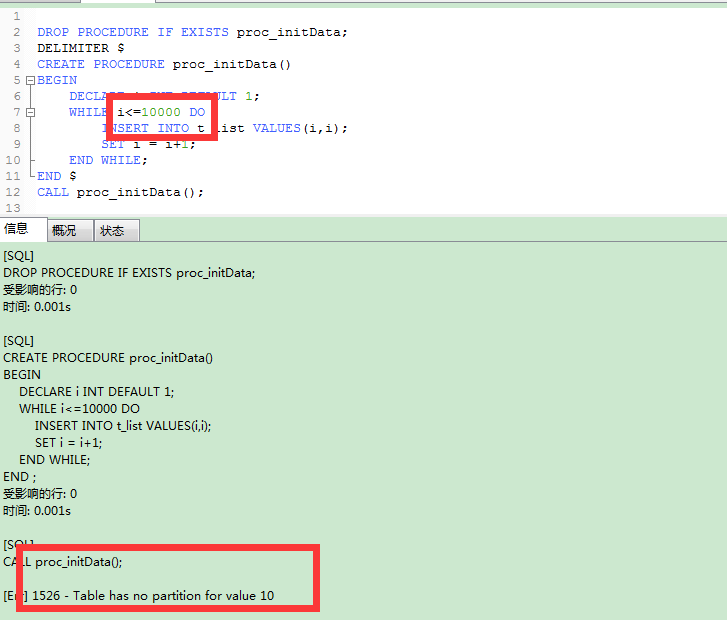
PARTITION d VALUES IN (4,11,12)

);



分区使用了某个名字,就不能在一那个名字来分区了

list没有匹配的模糊分区,匹配的值必须在分区内存在



直接可以看到,确定完分区后,在表中插入没有匹配的数据则无法插入

因此,我觉得list分区不常用.

hash分区:这种模式允许通过对表的一个或多个列的hash key进行计算 ,最后通过这个哈市码不同数值对应的数据区域进行分区,可以建立一个对表主键进行分区的表.

主键的hash值就是uuid的不可重复的那个值的hash

CREATE TABLE employees (

id INT NOT NULL,

fname VARCHAR(30),

lname VARCHAR(30),

hired DATE NOT NULL DEFAULT '1970-01-01',

separated DATE NOT NULL DEFAULT '9999-12-31',

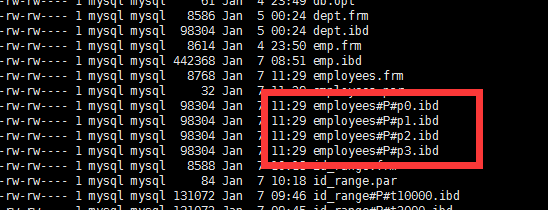
job\_code INT,

store\_id INT

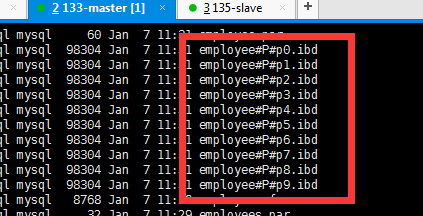
)

PARTITION BY HASH(store\_id)

PARTITIONS 4;



分区的名字是more的P后的分区数



具体是怎么根据hash值进行分区的不清楚

key分区:key分区和hash分区是一样的,只不过这里的可以是mysql系统根据自己的规则生成的.

CREATE TABLE tk (

col1 INT NOT NULL,

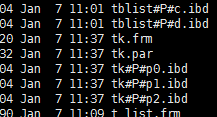
col2 CHAR(5),

col3 DATE

)

PARTITION BY LINEAR KEY (col1)

PARTITIONS 3;



子分区:子分区是分区表中每个分区的再次分割,子分区既可以使用hash分区,也可以使用key分区,这也被成为符复合分区.

一些规则:

1.如果一个分区创建了子分区,其他分区也要有子分区.

2.如果创建了分区,,每个分区的子分区数必须相同.

3.同一分区的子分区,名字不相同,不同分区的子分区名字可以相同(5.1.50版本不适用)\

CREATE TABLE IF NOT EXISTS `sub\_part` (

`news\_id` int(11) NOT NULL COMMENT '新闻ID',

`content` varchar(1000) NOT NULL DEFAULT '' COMMENT '新闻内容',

`u\_id` int(11) NOT NULL DEFAULT 0s COMMENT '来源IP',

`create\_time` DATE NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '时间'

) ENGINE=INNODB DEFAULT CHARSET=utf8

PARTITION BY RANGE(YEAR(create\_time))

SUBPARTITION BY HASH(TO\_DAYS(create\_time))(

PARTITION p0 VALUES LESS THAN (1990)(SUBPARTITION s0,SUBPARTITION s1,SUBPARTITION s2),

PARTITION p1 VALUES LESS THAN (2000)(SUBPARTITION s3,SUBPARTITION s4,SUBPARTITION good),

PARTITION p2 VALUES LESS THAN MAXVALUE(SUBPARTITION tank0,SUBPARTITION tank1,SUBPARTITION tank3)

);

================================================================

CREATE TABLE IF NOT EXISTS `sub\_part` (

id int(11) NOT NULL ,

`create\_time` DATE NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '时间'

) ENGINE=INNODB DEFAULT CHARSET=utf8

PARTITION BY RANGE(YEAR(create\_time))

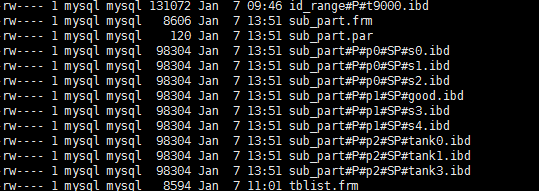
SUBPARTITION BY HASH(TO\_DAYS(create\_time))(

PARTITION p0 VALUES LESS THAN (1990)(SUBPARTITION s0,SUBPARTITION s1,SUBPARTITION s2),

PARTITION p1 VALUES LESS THAN (2000)(SUBPARTITION s3,SUBPARTITION s4,SUBPARTITION good),

PARTITION p2 VALUES LESS THAN MAXVALUE(SUBPARTITION tank0,SUBPARTITION tank1,SUBPARTITION tank3)

);



分区管理:

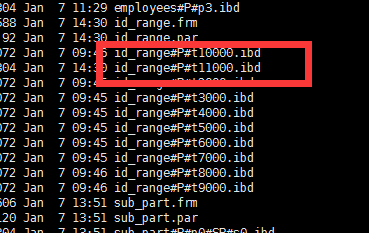
1.删除分区:

alter table t\_range drop partition 名称

2.新增分区

新增range分区:

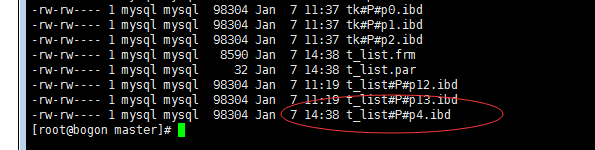
alter table id\_range add partition(partition t11000 values less than (11000));



新增range分区必须新增到最大的(只能) 最大值必须用括号包起来

新增list分区:

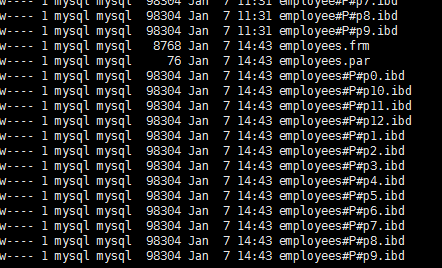
alter table t\_list add partition(partition p4 values in(25,26,27))



新增list分区,要保证分区的常量不能重复,需要做好备份,以防万一.

新增hash分区:

alter table employees add partition partitions 3;



新增hash分区,后面的数字是新增的数量,能一直新增.

新增key分区: 新增key和新增hash分区一致,

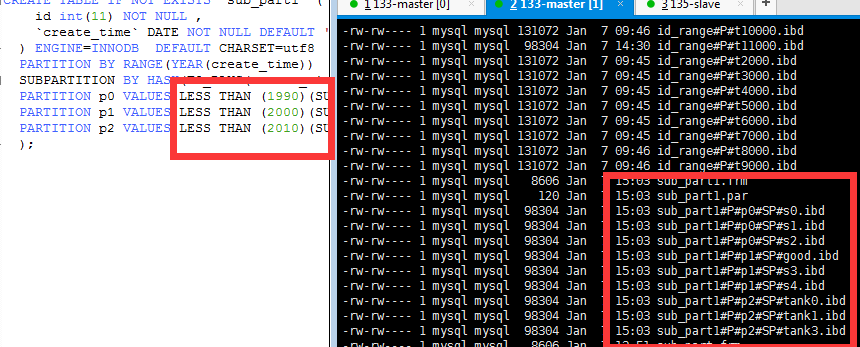
问题就是如果只是新增分区,不知道分区的规则,那新增分区直接等同于重新分区

新增子分区:

alter table sub\_part add partition(partition p3 values less than (2010)(

SUBPARTITION s1, SUBPARTITION s2,SUBPARTITION s3

));

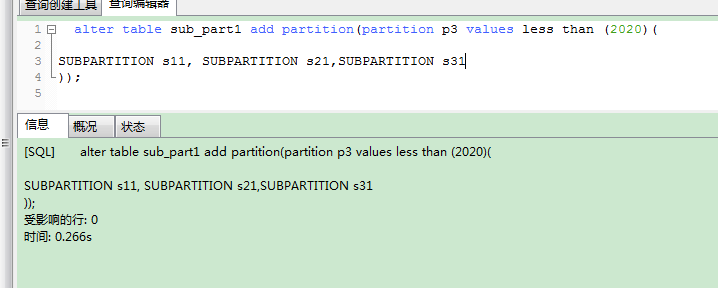


没有设置成maxvalue

出现问题:之前定义子分区时,设置分区是使用的时最大值,怎么才能添加新分区

range分区只能添加比之前大的分区,但是之前直接设置最后一个为maxvalue,所以不能添加新的子分区,

重新创建一个子分区,不设置最大值:

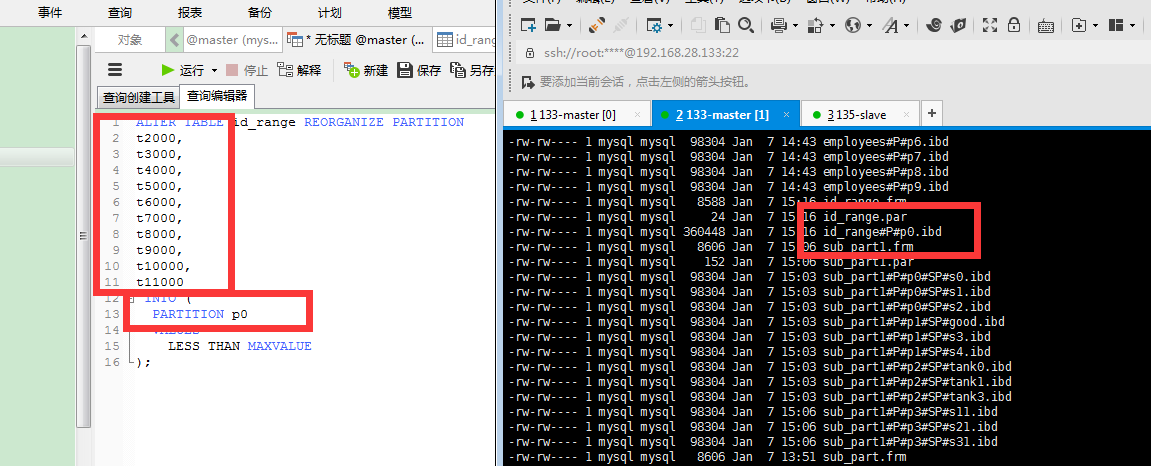


出现问题,不同分区的子分区的名称不能相同.

3.重新分区

range重新分区

ALTER TABLE user REORGANIZE PARTITION p0,p1,p2,p3,p4 INTO (PARTITION p0 VALUES LESS THAN MAXVALUE);



list重新分区:

ALTER TABLE t\_list REORGANIZE PARTITION

p12,p13,p4

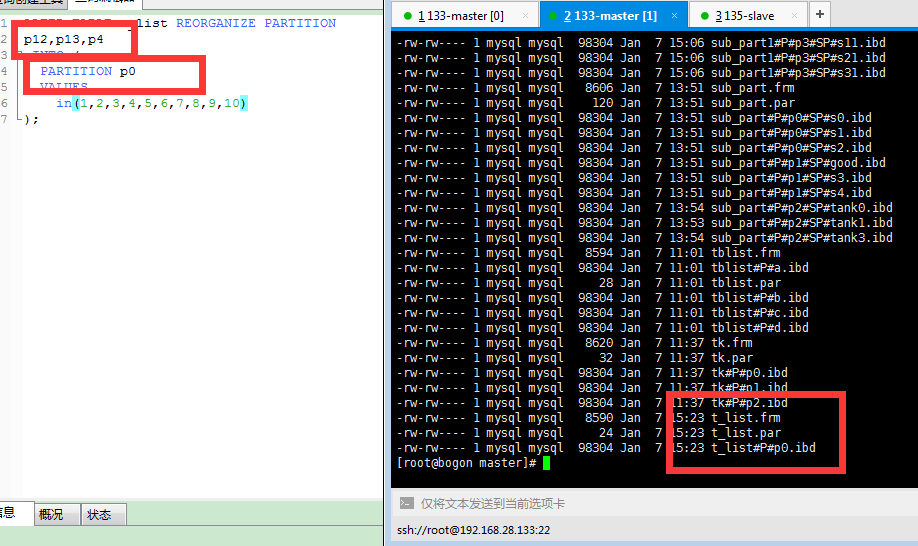
INTO (

PARTITION p0

VALUES

in(1,2,3,4,5,6,7,8,9,10)

);



子分区重新分区(子分区是各个分区的组合)

hash和key不支持重新分区