# 5.2.1数据库恢复事项

提示：

数据恢复和字符集关联很大，如果字符集不正确会导致恢复的数据乱码mysql命令以及source命令恢复数据库的原理就是把文件的SQL语句，在数据行的过程

5.2.2 利用source命令恢复数据

进入mysql数据库控制台，mysql-uroot-p登录后

mysql>use

然后使用source没拿过来，后面参数为脚本文件（如这里用到的.sql）

mysql>source oldboy\_db.sql#这个文件是系统路径，默认是登录mysql前的

mysql> truncate table test;

Query OK, 0 rows affected (0.00 sec)

mysql> source /opt/bak\_2015-12-18.sql

mysql> select \* from test;

+----+---------+

| id | name |

+----+---------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

+----+---------+

5 rows in set (0.00 sec)

提示：

source数据恢复和字符集关联很大，如果字符集不正确会导致恢复到数据乱码

utf8数据库，那么恢复的文件格式需要为utf8没有签名格式

# 5.2.3利用mysql命令恢复（标准）

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "use oldboy;drop table student;show tables"

[root@db02 3306]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock < /opt/bak\_2015-12-18.sql

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "use oldboy;show tables;"

+------------------+

| Tables\_in\_oldboy |

+------------------+

| student |

+------------------+

# 5.2.4针对压缩备份数据

方法一：

gzip –d /opt/oldboy\_bak.sql.gz 删除源文件解压  
mysql –uroot –p’oldboy’</opt/oldboy\_bak.sql 导入

方法二：

gunzip < oldboy\_bak.sql >/opt/oldboy\_bak.sql.gz  
mysql –uroot –p123456 </opt/oldboy\_bak.sql.gz  
或者

gunzip <oldboy\_bak.sql.gz|mysql –uroot –p123456

恢复总结：

1. source命令
2. mysql命令

# 5.3实现和mysql非交互式对话

查看完整的线程

[root@db02 3306]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show processlist;"

+----+------+-----------+------+---------+------+-------+------------------+

| Id | User | Host | db | Command | Time | State | Info |

+----+------+-----------+------+---------+------+-------+------------------+

| 27 | root | localhost | NULL | Sleep | 761 | | NULL |

| 57 | root | localhost | NULL | Query | 0 | NULL | show processlist |

+----+------+-----------+------+---------+------+-------+------------------+

[root@db02 3306]#

企业案例：mysql sleep线程过多的问题案例：

mysql> show processlist;

+----+------+-----------+------+---------+------+-------+------------------+

| Id | User | Host | db | Command | Time | State | Info |

+----+------+-----------+------+---------+------+-------+------------------+

| 27 | root | localhost | NULL | Query | 0 | NULL | show processlist |

+----+------+-----------+------+---------+------+-------+------------------+

1 row in set (0.00 sec)

mysql>kill 27 27是线程的ID号

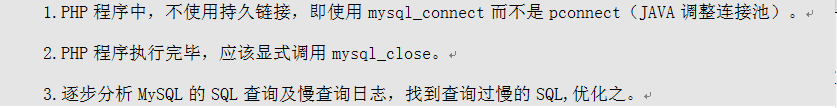
1. 配置文件里修改：

[mysqld]

interactive\_timeout = 120 此参数设置后wait\_timeout自动生效。

wait\_timeout = 120

1. 其他方法：



# 5.3.3利用mysql –e参数查看mysql变量及性能状态

[root@db02 backup]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show variables like '%server\_id%';"  
+---------------+-------+  
| Variable\_name | Value |  
+---------------+-------+  
| server\_id     | 1     |  
+---------------+-------+  
[root@db02 backup]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show variables like '%log\_bin%';"    
+---------------------------------+-------+  
| Variable\_name                   | Value |  
+---------------------------------+-------+  
| log\_bin                         | ON    |  
| log\_bin\_trust\_function\_creators | OFF   |  
| sql\_log\_bin                     | ON    |  
+---------------------------------+-------+

mysql> show status like "%select%"; select状态

+------------------------+-------+

| Variable\_name | Value |

+------------------------+-------+

| Com\_insert\_select | 0 |

| Com\_replace\_select | 0 |

| Com\_select | 1 |

| Select\_full\_join | 0 |

| Select\_full\_range\_join | 0 |

| Select\_range | 0 |

| Select\_range\_check | 0 |

| Select\_scan | 4 |

+------------------------+-------+  
[root@db02 backup]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show variables like '%slow\_log%';"  
[root@db02 backup]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show variables like '%slow%';"      
+---------------------+---------------------+  
| Variable\_name       | Value               |  
+---------------------+---------------------+  
| log\_slow\_queries    | ON                  |  
| slow\_launch\_time    | 2                   |  
| slow\_query\_log      | ON                  |  
| slow\_query\_log\_file | /data/3306/slow.log |

直接进数据库命令行敲

show variables like '%log\_bin%';

软件MySQLreport

小结：

myisa引擎索引缓存

即时生效，重启失效：

set global key\_buffer\_size = 1024\*1024\*32;

配置文件：

/etc/my.cnf,key\_buffer\_size = 32M

mysql> set global key\_buffer\_size = 1024\*1024\*32；  
Query OK, 0 rows affected (0.01 sec)  
mysql> show variables like '%key\_buffer%';        
+-----------------+----------+  
| Variable\_name   | Value    |  
+-----------------+----------+  
| key\_buffer\_size | 33554432 |  
+-----------------+----------+  
1 row in set (0.00 sec)

# 5.3.4不重启数据库修改数据库参数

方法一：

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "show variables;"|grep key\_buffer

key\_buffer\_size 16777216

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "set global key\_buffer\_size = 1024\*1024\*32;"

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "show variables;"|grep key\_buffer

key\_buffer\_size 33554432

方法二：

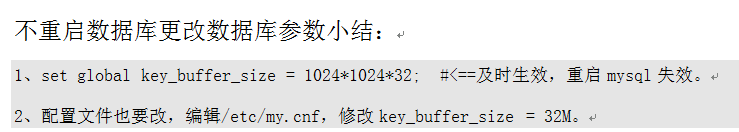
[root@db02 ~]# sed -n '47p' /data/3306/my.cnf

key\_buffer\_size = 16M

[root@db02 ~]# sed -i 's#key\_buffer\_size = 16M#key\_buffer\_size = 32M#g' /data/3306/my.cnf

[root@db02 ~]# sed -n '47p' /data/3306/my.cnf key\_buffer\_size = 32M

[root@db02 ~]#



# 5.3.5生产场景重要命令小结

1.show processlist; 查看数据库执行的SQL语句看不全

2.show full proseceelist;完整显示SQL语句

3.set global key\_buffer\_size = 1024\*1024\*32;"不重启数据库调整数据库参数直接生效重启后失效

4.show variables; 查看数据库配置参数例如：my.cnf里参数的生效情况

5.kill id  杀掉线程的命令，id为线程号（mysql下）

6.show status  查看当前会话的数据库状态信息。

7.show global status 查看整个数据库的运行状态信息。很重要，要分心并要做好监控

8.show engine innodb status 显示innodb疫情的性能状态（高级dba才会关注）

计算一天之内：数据库有多个insert，select

1. 定时任务每天0点show global status;按天取出对比
2. 按天分析binlog日志，获取数据库不同语句的频率

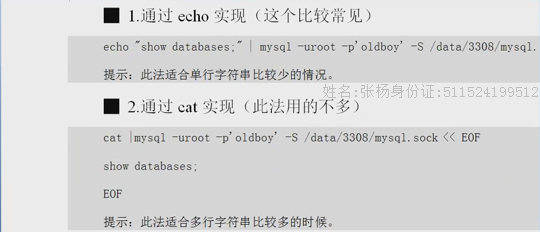
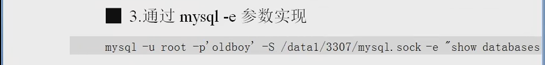
mysqladmin的相关命令：

mysqladmin password oldboy123 #🡸 设置密码  
mysqladmin -uroot -poldboy123 password oldboy #🡸 修改密码  
mysqladmin -uroot -poldboy123 status #🡸 查看状态  
mysqladmin -uroot -poldboy123 -S /data/3306/mysql.sock -i 1 status #🡸show glabal  
mysqladmin -uroot -poldboy123 flush-logs #🡸刷新  
mysqladmin -uroot -poldboy123 processlist #🡸慢查询  
mysqladmin -uroot -poldboy123 extended-status  
mysqladmin -uroot -poldboy123 processlist -i 1 ç #🡸实时跟踪。  
watch mysqladmin -uroot -poldboy123 processlist  
mysqladmin -uroot -p'oldboy' -S /data/3306/mysql.sock shutdown #🡸关闭mysql  
mysqladmin -uroot -p'oldboy' -S /data/3306/mysql.sock variables #🡸show variables

# 5.4mysql生产环境案例

5.4.1mysql数据库批量插入数据shell脚本实现

5.4.2不登录数据库执行mysql命令小结



# 5.5mysql工具 mysqlbinlog

mysqlbinlog工具的作用是解析二进制mysql内容

按照位置截取：

mysqlbinlog mysqlbin.000020 --start-position=365 --stop-position=456 -r pos.sql



mysqlbinlog命令：

1、把binlog解析为SQL语句（包含位置和时间点）。  
2、-d参数根据指定库查分binlog（单表通过关键字过滤）。  
3、截取部分binlog：位置参数：--start-position =365 –stop-position=456，精确定位取部分内容。  
4、截取部分binlog：时间参数：--start-datetime=’2014-10-16 17:14:15’ –stop-datetime=’2014-10-16 17:15:15’，模糊取部分内容，会丢数据。  
5、-r文件名，相当于重定向 “>文件名”  
6、解析ROW级别binlog日志的方法：  
mysqlbinlog –base64-output=decode-rows-v mysql-bin.000016  
mysqlbinlog –base64-output=”decode-rows” –verbose mysql-bin.000004

课后作业：总结如下命令  
mysqldump 备份命令  
mysql登录及恢复命令  
mysqlbinlog解析binlog命令  
mysqladmin管理mysql命令  
source恢复数据命令

# mysql完善恢复

生产工作中一般常用是一主多从的数据库架构，常见的备份方案是在某一个不对外的务的从库上开启binlog，然后实施定时全备份和实时增量备份。  
什么是增量备份？  
 利用二进制日志和全备进行的恢复过程，被称为增量恢复。  
  
什么情况下才需要数据库增量恢复？  
1） 主或者从库宕机（硬件损坏），是否需要增量恢复呢？  
答：不需要增量恢复，主机宕机，只需要把其中一个同步最快的从库切换为主库即可。  
   主机宕机，只要选择更更新最快的从库（master.info，或5.5半同步机制），提升为主库。  
   从库宕机，直接不用就好了（一般为LVS负载均衡）。或者正常恢复  
2） 人为操作数据库SQL语句破坏主库是否需要增量恢复？  
在数据库主库内部命令行误操作，会导致所有的数据库（包括主从库）数据丢失，例如：在主库执行了drop database oldboy ；这样的删除语句，这时候的从库也会执行这个drop database oldboy语句。从而导致所有数据库上的Oldboy数据库的数据丢失，这样的场景是需要增量恢复的。  
3） 只有一个主库是否需要增量恢复？  
 如果公司里只有一个主库的情况，首先应该多定时全量备份（一天每天一次）及增量备份（每隔1-10分钟对binlog日志做切割后备份到其他的服务器上，或者本地的其他硬盘里）或者写到网路文件系统（备份服务器）里。  
如果不允许数据丢失，最好的办法是做从库，通过DRBD（基于磁盘块的）同步。  
正常情况：  
主从同步：除了分担读写压力外，还可以防止物理设备损坏数据丢失的恢复。  
从库备份：在从库进行全量和增量方式的备份，可以防止人为对主库的误操作导致数据丢失。确保备份的从库实时和主库是同步状态。  
  
小结：一般由人为误操作（或者程序）逻辑的方式在数据库执行SQL语句等误操作时，才需要增量恢复，因为此时，所有从库也都执行了误操作。

# 2 mysql增量恢复的必备条件

开启MySQL log-bin日志功能

MySQL数据库开启了log-bin参数记录binlog日志功能如下：

grep log-bin /data/3306/my.cnf

log-bin = /data/3306/mysql-bin

提示：主库和备份的从库都要开启binlog记录功能。

小结：

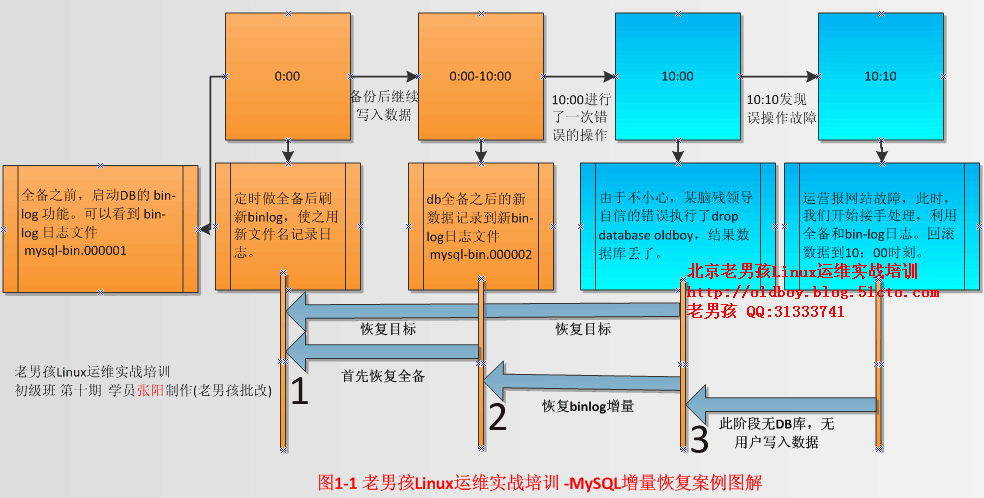
增量恢复的条件：

存在一份全备加上全备之后的时刻的所有增量binlog文件备份。

小结：增量恢复的条件：

存在一份全备加上全备之后的时刻到出问题时刻的所有增量binlog文件备份。

**mysql增量备份**



**解决上表问题：**

进入查看表test

mysql> select \* from test;

+----+---------+

| id | name |

+----+---------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

+----+---------+

5 rows in set (0.00 sec)

备份

mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -B -F -R -x --master-data=2 oldboy|gzip >/server/backup/oldboy\_$(date +%F).sql.gz

插入数据查看

mysql> insert into test values(6,'gongli')；

Query OK, 1 row affected (0.00 sec)

mysql> select \* from test;

+----+---------+

| id | name |

+----+---------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

| 6 | gongli |

+----+---------+

6 rows in set (0.00 sec)

删除库

mysql> drop database oldboy;

Query OK, 2 rows affected (0.05 sec)

进入备份目录

gzip -d oldboy\_2015-12-22.sql.gz

grep CHANGE oldboy\_2015-12-22.sql

cp /data/3306/mysql-bin.000015 .

mysqlbinlog -d oldboy mysql-bin.000015 >15.sql

vim 15.sql

删除drop命令行

恢复全备

mysql -uroot -poldboy123 -S /data/3306/mysql.sock </server/backup/oldboy\_2015-12-22.sql

查看

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| bingbing |

| gongli |

| mysql |

| oldboy |

| oldboy\_gbk |

| performance\_schema |

+--------------------+

7 rows in set (0.00 sec)

mysql> use oldboy

mysql> select \* from test;

+----+---------+

| id | name |

+----+---------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

+----+---------+

5 rows in set (0.00 sec)

恢复增量

mysql -uroot -poldboy123 -S /data/3306/mysql.sock </server/backup/15.sql

mysql> select \* from test;

+----+---------+

| id | name |

+----+---------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

| 6 | gongli |

+----+---------+

6 rows in set (0.00 sec)

