数据库的备份

# 物理备份和恢复

**对于MyISAM存储引擎**：使用flush tables with read lock(锁住所有的表，并将数据文件刷新到磁盘。)，然后将物理文件直接复制保存。

**对于innoDB存储引擎**：对于InnoDB存储引擎，需要关闭服务器(刷新脏页)然后备份物理文件。

# 二进制日志的备份

## 二进制日志格式

|  |
| --- |
| # at 3118932  #160512 17:30:35 server id 1 end\_log\_pos 3119031 CRC32 0x8091cbba Query thread\_id=163075 exec\_time=0 error\_code=0  SET TIMESTAMP=1463045435/\*!\*/;  BEGIN  /\*!\*/; |

#第一行包含日志文件内偏移字节值

#第二行：事件日期时间，使用他们来产生set timestamp语句；原服务器id（防止复制之间无限循环）；end\_log\_pos（下一个时间的偏移字节量）；时间类型（Query）;原服务器执行事件线程id（thread\_id=163075）；exec\_time(语句的时间戳和写入二进制日志的时间差)；error\_code（原服务器上事件产生的错误代码）

## 二进制日志备份恢复

#完全恢复，先关闭二进制日志：SET sql\_log\_bin=0;备份完了之后后附二进制日志 SET sql\_log\_bin=1;

mysqlbinlog mysql-bin.00001|mysql -uroot –p

#基于时间点的恢复

如果误删了一张表，使用完全恢复是没有用的，因为日志里同样也保留着删除的sql语句，所以我们需要恢复到误操作前的状态，然后跳过误操作的语句。

假如我在20:00误删了一张表，可以使用以下语句恢复：

mysqlbinlog --stop-date='2012-06-05 19:59:59' /var/log/mysql-bin.000001 | mysql -uroot -p

跳过误删除的时间点，再执行：

mysqlbinlog --start-date='2012-06-05 20:01:00' /var/log/mysql-bin.000001 | mysql -uroot –p

#基于位置点的恢复

基于位置点的恢复可以得到更为精确的数据，通过grep找到语句对应的位置

mysqlbinlog --database=… | grep -B3 -i 'drop table test'

mysqlbinlog --stop-position='889107' /var/lib/mysql/mysql-bin.000001|mysql -uroot -p

mysqlbinlog --start-position='889189' /var/lib/mysql/mysql-bin.000001|mysql -uroot -p

# LVM快照实现备份恢复

注意：确保事务日志和数据文件必须在同一卷上

|  |
| --- |
| #!/bin/bash  . /etc/profile  TIMESTAMP=`date +%Y%m%d%H%M%S`  HOSTNAME="localhost"  USERNAME="root"  PASSWORD="123"  SNAP\_SIZE=1G  SNAP\_MYSQL=SNAP\_MYSQL\_LV  MOUNT\_POINT=/mnt  EXEC\_MySQL="FLUSH TABLES;  FLUSH TABLES WITH READ LOCK;  FLUSH LOGS;  \! mysql -u$USERNAME -p$PASSWORD -h$HOSTNAME -e 'SHOW MASTER STATUS;' >/root/binlog.txt  \! lvcreate --snapshot --size=$SNAP\_SIZE --name $SNAP\_MYSQL /dev/MySQL\_VG/MYSQL\_LV  UNLOCK TABLES;"  echo "$EXEC\_MySQL" | mysql -u$USERNAME -p$PASSWORD -h$HOSTNAME  mount /dev/MySQL\_VG/$SNAP\_MYSQL $MOUNT\_POINT  cd /root  tar -zcvf ${TIMESTAMP}.tgz $MOUNT\_POINT  umount $MOUNT\_POINT  lvremove -f /dev/MySQL\_VG/$SNAP\_MYSQL |

#记录二进制日志

mysql -uroot -pmypass -e 'SHOW MASTER STATUS;' >/tmp /binlog.txt

#根据二进制日志实现增量备份

mysqlbinlog --start-position=187 mysql-bin.000004 | mysql -u … -p …

注意：一般使用复制+快照的方式进行数据库备份

# 逻辑备份和恢复

## mysqldump的使用

|  |
| --- |
| #导出所有数据库,该命令会导出包括系统数据库在内的所有数据库  mysqldump -uroot -proot --all-databases >/tmp/all.sql  #导出db1、db2两个数据库的所有数据  mysqldump -uroot -proot --databases db1 db2 >/tmp/user.sql  #导出db1中的a1、a2表,注意导出指定表只能针对一个数据库进行导出，且导出的内容中和导出数据库也不一样，导出指定表的导出文本中没有创建数据库的判断语句，只有删除表-创建表-导入数据  mysqldump -uroot -proot --databases db1 --tables a1 a2 >/tmp/db1.sql  #条件导出，导出db1表a1中id=1的数据条件导出只能导出单个表  mysqldump -uroot -proot --databases db1 --tables a1 --where='id=1' >/tmp/a1.sql  #生成新的binlog文件,-F  mysqldump -uroot -proot --databases db1 -F >/tmp/db1.sql  #只导出表结构不导出数据，–no-data(--no-create-info转储数据)  mysqldump -uroot -proot --no-data --databases db1 >/tmp/db1.sql  #跨服务器导出导入数据,将h1服务器中的db1数据库的所有数据导入到h2中的db2数据库中，db2的数据库必须存在否则会报错,加上-C参数可以启用压缩传递。  mysqldump --host=h1 -uroot -proot --databases db1 |mysql --host=h2 -uroot -proot db2  mysqldump --host=192.168.80.137 -uroot -proot -C --databases test |mysql --host=192.168.80.133 -uroot -proot test  #将主库的binlog位置和文件名追加到导出数据的文件中，–dump-slave该参数在在从服务器上执行，相当于执行show slave status。当设置为1时，将会以CHANGE MASTER命令输出到数据文件；设置为2时，会在change前加上注释。该选项将会打开–lock-all-tables，除非–single-transaction被指定。在执行完后会自动关闭--lock-tables选项。–dump-slave默认是1  #将当前服务器的binlog的位置和文件名追加到输出文件，–master-data改参数和--dump-slave方法一样，只是它是记录的是当前服务器的binlog，相当于执行show master status。  mysqldump -uroot -proot --dump-slave=1 --database db1 >/tmp/db1.sql  mysqldump -uroot -proot --dump-slave=2 --database db1 >/tmp/db1.sql  #--opt,默认启动等同于--add-drop-table, --add-locks, --create-options, --quick, extended-insert, --lock-tables, --set-charset, ==disable-keys 该选项默认开启, 可以用–skip-opt禁用.  mysqldump -uroot -p --host=localhost --all-databases --opt  #保证导出的一致性状态–single-transaction，该选项在导出数据之前提交一个 #BEGIN SQL语句，BEGIN 不会阻塞任何应用程序且能保证导出时数据库的一致性##状态。它只适用于多版本存储引擎（它不显示加锁通过判断版本来对比数据），仅InnoDB。本选项和–lock-tables 选项是互斥的，  #因为LOCK TABLES 会使任何挂起的事务隐含提交。要想导出大表的话，应结合使用--quick 选项，--quick, -q不缓冲查询，直接导出到标准输出。默认为打开状态，使用--skip-quick取消该选项。  #--lock-tables, -l 开始导出前，锁定所有表。用READ LOCAL锁定表以允许MyISAM表并行插入。对于支持事务的表例如InnoDB和BDB，  #--single-transaction是一个更好的选择，因为它根本不需要锁定表。请注意当导出多个数据库时。--lock-tables分别为每个数据库锁定表。因此，该选项不能保证导出文件中的表在数据库之间的逻辑一致性。不同数据库表的导出状态可以完全不同。  mysqldump -h10.161.219.44 -uk12\_tkdb -p123456 --master-data=2 --single-transaction k12\_tiku > /usr/local/src/dump.sql  #导出存储过程和自定义函数，触发器和事件--routines, --triggers,--events  #(默认转储触发器，--skip-triggers来启用（其它两个同样可以使用--skip-...）)  mysqldump -uroot -p --host=localhost --all-databases –routines |
| #恢复整个库数据库  mysql < dump.sql  #恢复指定表  sed -e '/./{H;$!d;}' -e 'x;/CREATE TABLE `tb-name`/!d;q' dump.sql  grep 'INSERT INTO `tb-name`' dump.sql |

## 符号分隔符文件备份恢复

# fields terminated：字段分隔符

# optionally enclosed by '\'' ：字符串类型标识

# lines terminated：行分隔符

select \* into outfile '/tmp/t1.txt' fields terminated by ',' optionally enclosed by '\'' lines terminated by '\n' from user;

注意：只能备份到运行MySQL服务器的机器上的文件中；运行MySQL的系统用户必须由文件目录的写权限。

#导入数据

LOAD DATA [LOW\_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file\_name.txt' [REPLACE | IGNORE] INTO TABLE tbl\_name [FIELDS [TERMINATED BY '\t'] [[OPTIONALLY] ENCLOSED BY ''] [ESCAPED BY '\\' ] ] [LINES TERMINATED BY '\n'] [IGNORE number LINES] [(col\_name,...)]

**LOAD DATA INFILE** 语句：以非常高的速度从一个文本文件中读取记录行并插入到一个表中。

**LOCAL**：如果 LOCAL 关键词被指定，文件从客户端主机读取。如果 LOCAL 没有被指定，文件必须位于服务器上。由于安全性的原因，当读取位于服务器端的文本文件时，文件必须处于数据库目录或可被所有人读取的地方。同时，为了对服务器端的文件使用 LOAD DATA INFILE，你必须在服务器主机上有 FILE 权限。路径搜索：绝对路径或者使用local（相对当前目录）或者本服务器相对数据库数据目录。

**LOW\_PRIORITY**：LOAD DATA 语句的执行将会被延迟，直到没有其它的客户端正在读取表。

**REPLACE 和 IGNORE 关键词**：控制对与现有的记录在唯一键值上重复的记录的处理。如果你指定 REPLACE，新的记录行将替换有相同唯一键值的现有记录行。如果你指定 IGNORE，将跳过与现有的记录行在唯一键值上重复的输入记录行。如果你没有指定任何一个选项，当重复键值出现时，将会发生一个错误，文本文件的剩余部分也将被忽略。

**FIELDS 子句**，它的子句 (TERMINATED BY、[OPTIONALLY] ENCLOSED BY 和 ESCAPED BY) 也是可选的，不过，你必须至少指定它们中的一个。如果你没有指定一个 FIELDS 子句，缺省的相同于如果你这样写：

|  |
| --- |
| FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\' |

OPTIONALLY ENCLOSED BY ：字符仅被作用于包围 CHAR 和 VARCHAR 字段。注意，在一个字段值中出现的 ENCLOSED BY 字符，通过用 ESCAPED BY 字符作为其前缀对其转义。同时也要注意，如果你指定一个空的 ESCAPED BY 值，可能会产生不能被 LOAD DATA INFILE 正确读出的输出文件。

ESCAPED BY ：控制如何写入或读出特殊字符(**对特殊字符进行转义**)。如果 FIELDS ESCAPED BY 字符不是空的，它将被用于做为下列输出字符的前缀，如果 FIELDS ESCAPED BY 字符为空，没有字符被转义。

**IGNORE number LINES**：忽略前面number行。

**[(col\_or\_user\_var\_name,...)][set col\_name=expr...]**：指定列导入

|  |
| --- |
| 如果一个行有很少的字段，没有输入字段的列将被设置为缺省值。  一个空的字段值不同于字段值丢失的解释：对于字符串类型，列被设置为空字符串;对于数字类型，列被设置为 0。对于日期和时间类型，列被设置为适合列类型的“零”值。  如果输入的记录行有太多的字段，多余的字段将被忽略，并增加警告的数目。  如果输入 ENUM 或 SET 列，你不能以 INSERT 语句的形式为其设置数字值。所有的 ENUM 和 SET 必须以字符串指定！ |
| 举例：  LOAD DATA INFILE 'data\_lexicon/data41.txt'  REPLACE INTO TABLE `assessment`.`lexicon2` CHARACTER SET utf8 FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '"' LINES TERMINATED BY '\r\n' IGNORE 1 LINES (`participle` ); |

# 使用xtrabackup实现备份恢复

xtrabackup是一个对InnoDB做数据备份的工具，支持在线热备份（备份时不影响数据读写），是商业备份工具InnoDB Hotbackup的一个很好的替代品。Xtrabackup有两个主要的工具：xtrabackup、innobackupex

* xtrabackup只能备份InnoDB和XtraDB两种数据表，而不能备份MyISAM数据表
* innobackupex-1.5.1则封装了xtrabackup，是一个脚本封装，所以能同时备份处理innodb和myisam，但处理myisam时需要加一个读锁

相比mysqldump，xtrabackup备份和恢复的速度快多了。

## 安装xtrabackup

|  |
| --- |
| #准备阶段  yum -y install cmake gcc gcc-c++ libaio libaio-devel automake autoconf bison libtool ncurses-devel libgcrypt-devel libev-devel libcurl-devel vim-common  #下载安装xtrabackup  cd /usr/local/src  cd percona-xtrabackup  git checkout 2.4  cmake -DBUILD\_CONFIG=xtrabackup\_release -DWITH\_MAN\_PAGES=OFF -DDOWNLOAD\_BOOST=1 -DWITH\_BOOST=/usr/local/boost&& make -j4  make install  或者  tar zxvf percona-xtrabackup-2.3.4-Linux-x86\_64.tar.gz  cp /usr/local/src/ percona-xtrabackup-2.3.4-Linux-x86\_64/bin/\* /usr/local/bin/  或者  rpm -ivh http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.noarch.rpm  yum install percona-xtrabackup |
| 安装XtraBackup后，其实会有几个工具：   * innobackupex：这个是其实是下面三个工具的一个perl脚本封装，可以备份MyISAM, InnoDB, XtraDB表。但在处理Myisam时需要加一个读锁。 * xtrabackup：一个由C编译而来的二进制文件，只能备份InnoDB和XtraDB数据。 * xbcrypt：用来加密或解密备份的数据。 * xbstream：用来解压或压缩xbstream格式的压缩文件。 |
| 常用参数：   * --user= #指定数据库备份用户 * --password= #指定数据库备份用户密码 * --port= #指定数据库端口 * --host= #指定备份主机 * --socket= #指定socket文件路径 * --databases= #备份指定数据库,多个空格隔开，如--databases=”dbname1 dbname2″,不加备份所有库 * --defaults-file= #指定my.cnf配置文件 * --apply-log #日志回滚 * --no-timestamp 创建备份时不自动生成时间目录，可以自定义备份目录名例如： /backups/mysql/base * --incremental= #增量备份，后跟增量备份路径 * --incremental-basedir= #增量备份，指上次增量备份路径 * --incremental-dir=DIRECTORY 增量备份存贮的目录路径 * --redo-only #合并全备和增量备份数据文件, 配合--incremental-dir 增量备份目录使用。 * --copy-back #将备份数据复制到数据库，数据库目录要为空 * --no-timestamp #生成备份文件不以时间戳为目录名 * --stream= #指定流的格式做备份,--stream=tar,将备份文件归档 * --remote-host=user@ip DST\_DIR #备份到远程主机 * --force-non-empty-directories 如果是特定库备份还原，不需要删掉整个mysql目录，只是特定库的及相关文件就可以，还原时加上此参数就不会报错。 |

## 完整备份与恢复

|  |
| --- |
| #完整备份,ulimit -n 65535  innobackupex --user=root --password=123 --socket=/data/mysql/data/mysql.sock ---host=127.0.0.1 /mnt  #备份恢复  innobackupex --defaults-file=/etc/mysql/my.cnf --copy-back /home/loongtao/mysql\_backup/2015-02-08\_11-56-48/ |
| #备份文件说明  # ls 2015-02-08\_11-56-48  backup-my.cnf：记录innobackup使用到mysql参数  xtrabackup\_binary：备份中用到的可执行文件  xtrabackup\_checkpoints：记录备份的类型、开始和结束的日志序列号  xtrabackup\_logfile：备份中会开启一个log copy线程，用来监控innodb日志文件（ib\_logfile），如果修改就会复制到这个文件 |

## 完整备份+增量备份与恢复

|  |
| --- |
| #完全备份后位置是：/mysql\_backup/2015-02-08\_11-56-48  innobackupex --user=bak --password='bak2015' /mysql\_backup  #增量备份，指定前一次备份的目录   innobackupex --user=bak --password='bak2015' --incremental /data1/mysql\_backup --incremental-basedir=/mysql\_backup/2015-02-08\_11-56-48  innobackupex --user=bak --password='bak2015' --incremental /data1/mysql\_backup --incremental-basedir=/mysql\_backup/2015-02-08\_12-16-06 |
|  |
| #备份恢复步骤：将增量备份1、增量备份2…合并到完整备份，拼接成完整备份，将新的完整备份以拷贝的形式到数据库空目录（rm /var/lib/mysql/\* -rf）  #xtrabackup把备份过程中可能有尚未提交的事务或已经提交但未同步数据文件的事务，写到xtrabackup\_logfile文件，所以要先通过这个日志文件回滚，把未完成的事务同步到备份文件，保证数据文件处于一致性。  innobackup --apply-log --redo-only 2015-02-08\_11-56-48  #合并第一个增量备份  innobackupex --apply-log --redo-only /mysql\_backup/2015-02-08\_11-56-48/ --incremental-dir=mysql\_backup/2015-02-08\_12-16-06  #合并第二个增量备份  innobackupex --apply-log --redo-only /mysql\_backup/2015-02-08\_11-56-48/ --incremental-dir=mysql\_backup/2015-02-08\_16-06-53  #恢复完整备份（这时2015-02-08\_11-56-48完整备份已经包含所有增量备份，可以通过查看checkpoints来核实）  innobackupex --defaults-file=/etc/mysql/my.cnf --copy-back /mysql\_backup/2015-02-08\_11-56-48/  #修改恢复数据文件权限，并启动MySQL,查看数据库恢复情况  chown -R mysql.mysql /var/lib/mysql |

## 备份文件归档压缩

|  |
| --- |
| #归档并发送到备份服务器  innobackupex --databases=test --user=bak --password='bak2015' --stream=tar /mysql\_backup 2>/mysql\_backup/bak.log |ssh root@192.168.18.251 "cat - > /mysql\_backup/`date +%F`.tar"  解压：tar -ixvf date +%F.tar  #归档备份  innobackupex --databases=test --user=bak --password='bak2015' --stream=tar /mysql\_backup > /mysql\_backup/`date +%F`.tar  解压：tar -ixvf date +%F.tar  #压缩归档备份  innobackupex --databases=test --user=bak --password='bak2015' --stream=tar /mysql\_backup |gzip >/mysql\_backup/`date +%F`.tar.gz  解压：tar -izxvf date +%F.tar.gz |

## 自动备份还原脚本

|  |
| --- |
| #backup.sh  #!/bin/sh  #  # 第一次执行它的时候它会检查是否有完全备份,否则先创建一个全库备份  # 当你再次运行它的时候，它会根据脚本中的设定来基于之前的全库备份进行增量备份  #ocpyang@126.com  INNOBACKUPEX\_PATH=innobackupex #INNOBACKUPEX的命令  INNOBACKUPEXFULL=/usr/bin/$INNOBACKUPEX\_PATH #INNOBACKUPEX的命令路径  #MySQL目标服务器以及用户名和密码  MYSQL\_CMD="--host=192.168.2.188 --user=system --password=password --port=3306"  MYSQL\_UP="--user=system --password=password -ppassword" #mysql的用户名和密码  TMPLOG="/tmp/innobackupex.$$.log"  MY\_CNF=/usr/local/mysql/my.cnf #mysql的配置文件  MYSQL=/usr/bin/mysql  MYSQL\_ADMIN=/usr/bin/mysqladmin  BACKUP\_DIR=/backup # 备份的主目录  FULLBACKUP\_DIR=$BACKUP\_DIR/full # 全库备份的目录  INCRBACKUP\_DIR=$BACKUP\_DIR/incre # 增量备份的目录  FULLBACKUP\_INTERVAL=86400 # 全库备份的间隔周期，时间：秒  KEEP\_FULLBACKUP=1 # 至少保留几个全库备份  logfiledate=backup.`date +%Y%m%d%H%M`.txt  #开始时间  STARTED\_TIME=`date +%s`    #############################################################################  # 显示错误并退出  #############################################################################  error()  {  echo "$1" 1>&2  exit 1  }    # 检查执行环境  if [ ! -x $INNOBACKUPEXFULL ]; then  error "$INNOBACKUPEXFULL未安装或未链接到/usr/bin."  fi    if [ ! -d $BACKUP\_DIR ]; then  error "备份目标文件夹:$BACKUP\_DIR不存在."  fi    if [ -z "`$MYSQL\_ADMIN $MYSQL\_UP status | grep 'Uptime'`" ] ; then  error "MySQL 没有启动运行."  fi      if ! `echo 'exit' | $MYSQL -s $MYSQL\_CMD` ; then  error "提供的数据库用户名或密码不正确!"  fi    # 备份的头部信息  echo "----------------------------"  echo  echo "$0: MySQL备份脚本"  echo "开始于: `date +%F' '%T' '%w`"  echo    #新建全备和差异备份的目录  mkdir -p $FULLBACKUP\_DIR  mkdir -p $INCRBACKUP\_DIR    #查找最新的完全备份  LATEST\_FULL\_BACKUP=`find $FULLBACKUP\_DIR -mindepth 1 -maxdepth 1 -type d -printf "%P\n" | sort -nr | head -1`    # 查找最近修改的最新备份  LATEST\_FULL\_BACKUP\_CREATED\_TIME=`stat -c %Y $FULLBACKUP\_DIR/$LATEST\_FULL\_BACKUP`    #如果全备有效进行增量备份否则执行完全备份  if [ "$LATEST\_FULL\_BACKUP" -a `expr $LATEST\_FULL\_BACKUP\_CREATED\_TIME + $FULLBACKUP\_INTERVAL + 5` -ge $STARTED\_TIME ] ; then  # 如果最新的全备未过期则以最新的全备文件名命名在增量备份目录下新建目录  echo -e "完全备份$LATEST\_FULL\_BACKUP未过期,将根据$LATEST\_FULL\_BACKUP名字作为增量备份目录命名"  echo " "  NEW\_INCRDIR=$INCRBACKUP\_DIR/$LATEST\_FULL\_BACKUP  mkdir -p $NEW\_INCRDIR    # 查找最新的增量备份是否存在.指定一个备份的路径作为增量备份的基础  LATEST\_INCR\_BACKUP=`find $NEW\_INCRDIR -mindepth 1 -maxdepth 1 -type d | sort -nr | head -1`  if [ ! $LATEST\_INCR\_BACKUP ] ; then  INCRBASEDIR=$FULLBACKUP\_DIR/$LATEST\_FULL\_BACKUP  echo -e "增量备份将以$INCRBASEDIR作为备份基础"  echo " "  else  INCRBASEDIR=$LATEST\_INCR\_BACKUP  echo -e "增量备份将以$INCRBASEDIR作为备份基础"  echo " "  fi    echo "使用$INCRBASEDIR作为基础做新的增量备份."  $INNOBACKUPEXFULL --defaults-file=$MY\_CNF --use-memory=4G $MYSQL\_CMD --incremental $NEW\_INCRDIR --incremental-basedir $INCRBASEDIR > $TMPLOG 2>&1  else  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  echo -e "正在执行全新的完全备份...请稍等..."  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  $INNOBACKUPEXFULL --defaults-file=$MY\_CNF --use-memory=4G $MYSQL\_CMD $FULLBACKUP\_DIR > $TMPLOG 2>&1  fi  #保留一份备份的详细日志  cat $TMPLOG>/backup/$logfiledate    if [ -z "`tail -1 $TMPLOG | grep 'innobackupex: completed OK!'`" ] ; then  echo "$INNOBACKUPEX命令执行失败:"; echo  echo -e "---------- $INNOBACKUPEX\_PATH错误 ----------"  cat $TMPLOG  rm -f $TMPLOG  exit 1  fi    THISBACKUP=`awk -- "/Backup created in directory/ { split( \\\$0, p, \"'\" ) ; print p[2] }" $TMPLOG`  rm -f $TMPLOG    echo -n "数据库成功备份到:$THISBACKUP"  echo    # 提示应该保留的备份文件起点  LATEST\_FULL\_BACKUP=`find $FULLBACKUP\_DIR -mindepth 1 -maxdepth 1 -type d -printf "%P\n" | sort -nr | head -1`  echo -e "必须保留$KEEP\_FULLBACKUP份全备和全备$LATEST\_FULL\_BACKUP以后的所有增量备份."  #删除过期的全备  echo -e "寻找过期的全备文件并删除">>/backup/$logfiledate  for efile in $(/usr/bin/find $FULLBACKUP\_DIR/ -mtime +6)  do  if [ -d $efile ]; then  rm -rf $efile  echo -e "删除过期全备文件:$efile" >>/backup/$logfiledate  elif [ -f $efile ]; then  rm -rf $file  echo -e "删除过期全备文件:$efile" >>/backup/$logfiledate  fi;    done  if [ $? -eq "0" ];then  echo  echo -e "未找到可以删除的过期全备文件"  fi    echo  echo "完成于: `date +%F' '%T' '%w`"  exit 0 |
| #!/bin/sh  #  # 使用方法：  # ./restore.sh /你备份文件的全路径  #ocpyang@126.com  INNOBACKUPEX=innobackupex  INNOBACKUPEX\_PATH=/usr/bin/$INNOBACKUPEX  TMP\_LOG="/var/log/restore.$$.log"  MY\_CNF=/usr/local/MySQL/my.cnf  BACKUP\_DIR=/backup # 你的备份主目录  FULLBACKUP\_DIR=$BACKUP\_DIR/full # 全库备份的目录  INCRBACKUP\_DIR=$BACKUP\_DIR/incre # 增量备份的目录  MEMORY=4096M # 还原的时候使用的内存限制数  ERRORLOG=`grep -i "^log-error" $MY\_CNF |cut -d = -f 2`  MYSQLD\_SAFE=/usr/local/mysql/bin/mysqld\_safe  MYSQL\_PORT=3306  #############################################################################  #显示错误  #############################################################################  error()  {  echo "$1" 1>&2  exit 1  }    #############################################################################  # 检查innobackupex错误输出  #############################################################################  check\_innobackupex\_fail()  {  if [ -z "`tail -2 $TMP\_LOG | grep 'completed OK!'`" ] ; then  echo "$INNOBACKUPEX命令执行失败:"; echo  echo "---------- $INNOBACKUPEX的错误输出 ----------"  cat $TMP\_LOG  #保留一份备份的详细日志  logfiledate=restore.`date +%Y%m%d%H%M`.txt  cat $TMP\_LOG>/backup/$logfiledate  rm -f $TMP\_LOG  exit 1  fi  }      # 选项检测  if [ ! -x $INNOBACKUPEX\_PATH ]; then  error "$INNOBACKUPEX\_PATH在指定路径不存在,请确认是否安装或核实链接是否正确."  fi    if [ ! -d $BACKUP\_DIR ]; then  error "备份目录$BACKUP\_DIR不存在."  fi    if [ $# != 1 ] ; then  error "使用方法: $0 使用还原目录的绝对路径"  fi    if [ ! -d $1 ]; then  error "还原到:$1不存在."  fi    # Some info output  echo "----------------------------"  echo  echo "$0: MySQL还原脚本"  START\_RESTORE\_TIME=`date +%F' '%T' '%w`  echo "数据库还原开始于: $START\_RESTORE\_TIME"  echo    PARENT\_DIR=`dirname $1`    if [ $PARENT\_DIR = $FULLBACKUP\_DIR ]; then  FULLBACKUP=$1  echo "还原`basename $FULLBACKUP`"  echo  else  if [ `dirname $PARENT\_DIR` = $INCRBACKUP\_DIR ]; then  INCR=`basename $1`  FULL=`basename $PARENT\_DIR`  FULLBACKUP=$FULLBACKUP\_DIR/$FULL  if [ ! -d $FULLBACKUP ]; then  error "全备:$FULLBACKUP不存在."  fi  echo "还原$FULL到增量$INCR"  echo  echo "Prepare完整备份集..........."  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  $INNOBACKUPEX\_PATH --defaults-file=$MY\_CNF --apply-log --redo-only --use-memory=$MEMORY $FULLBACKUP > $TMP\_LOG 2>&1  check\_innobackupex\_fail  #判断lsn  check\_incre\_file=`find $PARENT\_DIR -mindepth 1 -maxdepth 1 -type d -printf "%P\n " | sort -nr |grep -v '^$' | head -n 1`  check\_incre\_lastlsn=${PARENT\_DIR}/${check\_incre\_file}/xtrabackup\_checkpoints  fetch\_incre\_lastlsn=`grep -i "^last\_lsn" ${check\_incre\_lastlsn} |cut -d = -f 2`  check\_full\_file=`find $FULLBACKUP/ -mindepth 1 -maxdepth 1 -type d -printf "%P\n " | sort -nr |grep -v '^$' | head -n 1`  check\_full\_lastlsn=$FULLBACKUP/${check\_full\_file}/xtrabackup\_checkpoints  fetch\_full\_lastlsn=`grep -i "^last\_lsn" ${check\_incre\_lastlsn} |cut -d = -f 2`    # Prepare增量备份集,即将增量备份应用到全备目录中  for i in `find $PARENT\_DIR -mindepth 1 -maxdepth 1 -type d -printf "%P\n" | sort -n`;  do  ######判断LSN  if [ "${fetch\_incre\_lastlsn}"="${fetch\_full\_lastlsn}" ];then  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  echo "LSN不需要prepare!"  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  echo  break  else  ######判断LSN  echo "Prepare增量备份集$i........"  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  $INNOBACKUPEX\_PATH --defaults-file=$MY\_CNF --apply-log --redo-only --use-memory=$MEMORY $FULLBACKUP --incremental-dir=$PARENT\_DIR/$i > $TMP\_LOG 2>&1  check\_innobackupex\_fail  if [ $INCR = $i ]; then  break  fi  fi  ######判断LSN  done    else  error "未知的备份类型"  fi  fi  echo "prepare全备集，回滚那些未提交的事务..........."  $INNOBACKUPEX\_PATH --defaults-file=$MY\_CNF --apply-log --use-memory=$MEMORY $FULLBACKUP > $TMP\_LOG 2>&1  check\_innobackupex\_fail  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  echo "1.数据库还原中 ...请稍等"  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  $INNOBACKUPEX\_PATH --defaults-file=$MY\_CNF --copy-back $FULLBACKUP > $TMP\_LOG 2>&1  check\_innobackupex\_fail    rm -f $TMP\_LOG  echo "2.恭喜,还原成功!."  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  #修改目录权限  echo "修改mysql目录的权限."  mysqlcnf="/usr/local/mysql/my.cnf"  mysqldatadir=`grep -i "^basedir" $mysqlcnf |cut -d = -f 2`  `echo 'chown -R mysql:mysql' ${mysqldatadir}`  echo "3.权限修改成功!"  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  #自动启动mysql  INIT\_NUM=1  if [ ! -x $MYSQLD\_SAFE ]; then  echo "mysql安装时启动文件未安装到$MYSQLD\_SAFE或无执行权限"  exit 1 #0是执行成功,1是执行不成功  else  echo "启动本机mysql端口为:$MYSQL\_PORT的服务"  $MYSQLD\_SAFE --defaults-file=$MY\_CNF > /dev/null &  while [ $INIT\_NUM -le 6 ]  do  PORTNUM=`netstat -lnt|grep ${MYSQL\_PORT}|wc -l`  echo "mysql启动中....请稍等..."  sleep 5  if [ $PORTNUM = 1 ];  then  echo "mysql \*\*\*\*启动成功\*\*\*\*"  exit 0  fi  INIT\_NUM=$(($INIT\_NUM +1))  done  echo -e "mysql启动失败或启动时间过长,请检查错误日志`echo 'cat ' ${ERRORLOG}`"  echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  exit 0  fi    END\_RESTORE\_TIME=`date +%F' '%T' '%w`  echo "数据库还原完成于: $END\_RESTORE\_TIME"  exit 0 |

# 检查和修复数据库中的表

## 使用check table检查表

CHECK TABLE tb1,tb2 [选项]

注意：检查的存储引擎有InnoDB，MyISAM，ARCHIVE，CSV表

以下选项适合MyISAM存储引擎，其他存储引擎忽略

* CHANGED：表正常关闭且上次未被更改，跳过表检查
* EXTENDED：执行扩展检查，以确保表完全一致
* FAST：表为正常关闭才检查
* MEDIUM：通过检查索引，扫描数据行的问题，并进行校验。默认值
* QUICK：只扫描索引，不扫描数据行。
* FOR UPGRADE：确定被检查的表是否与当前的MySQL版本兼容。

## 使用repair table修复表

REPAIR TABLE tb1,tb2 [选项]

注意：支持的存储引擎有MyISAM，ARCHIVE，CSV表

以下选项适合MyISAM存储引擎，其他存储引擎忽略

* EXTENDED：执行扩展修复，重建索引
* QUICK：尝试修复索引数据，不修复数据文件

## 使用mysqlcheck检查修复表

等效于check table，repair table两个命令

|  |
| --- |
| #检查数据库中的所有数据库的所有表  mysqlcheck --all-databases  ##检查数据库中的多个数据库  mysqlcheck --databases 库名1 库名2  #检查数据库中的所有表  mysqlcheck 数据库名  #检查数据库中的指定表  mysqlcheck 数据库名 表名1 表名2  常用选项：  #check相关  --check-only-changed  --extended  --fast  --medium-check  --quick  #repair相关  --repair  --repair -extended  --repair -quick |

## InnoDB的自动修复

Innodb存储引擎启动失败，解决方法：

第一步：设置变量

innodb\_force\_recovery=4(自动恢复策略)

innodb\_force\_load\_corrupted=on(表示表受损也要加载他们)

第二步：使用mysqldump转储文件的方式备份InnoDB表，然后修改配置，重新生成表空间和数据文件。