MySQL数据库备份（linux）

标签： mysql linux

2017年04月06日 15:26:02 538人阅读 评论(0) 收藏 举报

为了保障数据的安全，需要定期对数据进行备份。备份的方式有很多种，效果也不一样。一旦数据库中的数据出现了错误，就需要使用备份好的数据进行还原恢复。从而将损失降到最低。下面我们来了解一下MySQL常见的有三种备份恢复方式：

1、利用Mysqldump+二进制日志实现备份
2、利用LVM快照+二进制日志实现备份
3、使用Xtrabackup备份

一：实验环境介绍：

系统介绍：CentOS6.4_X64
数据库版本：mysql-5.5.33

二：基于Mysqldump命令实现备份恢复

2.1、思路概念

Mysqldump是一个逻辑备份命令；意思就是将数据库中的数据备份成一个文本文件；也可以说是将表的结构和数据存储在文本文件中。

Mysqldump命令的工作原理很简单，它先查出需要备份的表的结构，再在文本文件中生成一个CREATE语句。然后，将表中的所有记录转换为一条INSTERT语句。这些CREATE语句和INSTERT语句都是还原时使用的。还原数据时就可以使用其中的CREATE语句来创建表。使用其中的INSERT语句来还原数据。它可以实现整个服务器备份，也可以实现单个或部分数据库、单个或部分表、表中的某些行、存储过程、存储函数、触发器的备份；并且能自动记录备份时刻的二进制日志文件及相应的位置。对于InnoDB存储引擎来讲支持基于单事务模式实现热备，对于MyISAM则最多支持温备。

2.2、备份策略

Mysqldump全备+二进制日志增备

2.3、过程实现

（1）Mysqldump全备
由于Mysql数据库默认的为MyISAM存储引擎所以只有使用温备（备份同时仅支持读请求）进行，所以我们要为所有数据库添加读锁

?

```
[root@stu18 ~]
#mysqldump -uroot -pmypass --lock-all-tables --master-data=2 --events --routines--all-databases > /zhao/database_`date +%F`.sql
```

解析：–lock-all-tables表示为所有表施加读锁；–master-data=2表示在备份文件中记录当前二进制日志的位置；–events表示备份数据的同时备份时间调度器代码；–routines表示备份数据的同时备份存储过程和存储函数；–all-databases表示备份所有库。

?

```
[root@stu18 zhao]
# less database_2013-08-13.sql
```

```
--
#表示注释项
-- Position to start replication or point-in-time recovery from
--
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=14203;
#这里表示当前处于mysql-bin.000001这个二进制日志中，事件为14203这是通过--master-data=2产生的
--
-- Current Database: `hellodb`
--
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `hellodb` /*!40100 DEFAULT CHARACTER SET utf8
*/;
```

（2）二进制全备

方法一： 导出二进制日志文件内容

?

```
[root@stu18 data]
# mysqlbinlog mysql-bin.000001 >/zhao/binlog_`date +%F`.sql
```

方法二：滚动日志复制文件

?

```
mysql> flush logs;
#滚动日志
[root@stu18 data]
# cp mysql-bin.000001 /zhao/mysql-bin.000001 #复制导出二进制文件
```

（3）二进制增备
首先添加数据信息

?

```
mysql> use hellodb;
mysql> INSERT INTO students(Name,Age,Gender,ClassID,TeacherID) values ('Yang kang',22,
'M',3,3);
```

然后二进制增备

?

```
[root@stu18 data]
# mysqlbinlog --start-position=14203 --stop-position=14527 mysql-bin.000001 > /zhao/bin
log_`date +%F_%H`.sql
```

解析：–start-position=14203是上次全备之后的二进制事件位置；–stop-position=14527最近一天的二进制事件位置。

2.4、模拟数据库损坏，实现恢复工作

?

```
mysql> DROP DATABASE hellodb;
#删除数据库
#############下面这些过程要在离线状态下执行#############
mysql> SET sql_log_bin=0;
#先关闭二进制日志
mysql> flush logs;
#滚动日志
[root@stu18 ~]
# mysql -uroot -pmypass < /zhao/database_2013-08-13.sql #导入数据库备份文件
[root@stu18 ~]
# mysql -uroot -pmypass < /zhao/binlog_2013-08-13_19.sql #导入增量备份文件
[root@stu18 ~]
# mysql -uroot –pmypass #登录查看，恢复完成
```

```
14 mysql> SET sql_log_bin=1;
```

这种备份方式恢复简单，但是恢复还原之后索引会出现错误需要重建，而且备份结果会占据很大的空间，请酌情使用。

PS：mysqldump常用命令小结

备份MySQL数据库的命令

?

```
1 mysqldump -hhostname -uusername -ppassword databasename > backupfile.sql
```

备份MySQL数据库为带删除表的格式

备份MySQL数据库为带删除表的格式，能够让该备份覆盖已有数据库而不需要手动删除原有数据库。

?

```
1 mysqldump --add-drop-table -uusername -ppassword databasename > backupfile.sql
```

直接将MySQL数据库压缩备份

?

```
1 mysqldump -hhostname -uusername -ppassword databasename | gzip > backupfile.sql.gz
```

备份MySQL数据库某个(些)表

?

```
1 mysqldump -hhostname -uusername -ppassword databasename specific_table1 specific_table2
> backupfile.sql
```

同时备份多个MySQL数据库

?

```
1 mysqldump -hhostname -uusername -ppassword –databases databasename1 databasename2 datab
asename3 > multibackupfile.sql
```

仅仅备份数据库结构

?

```
1 mysqldump –no-data –databases databasename1 databasename2 databasename3 > structureback
upfile.sql
```

备份服务器上所有数据库

?

```
1 mysqldump –all-databases > allbackupfile.sql
```

还原MySQL数据库的命令

?

```
1 mysql -hhostname -uusername -ppassword databasename < backupfile.sql
```

还原压缩的MySQL数据库

?

```
1 gunzip < backupfile.sql.gz | mysql -uusername -ppassword databasename
```

将数据库转移到新服务器

?

```
mysqldump -uusername -ppassword databasename | mysql –host=*.*.*.* -C databasename
```

三、基于LVM快照实现备份恢复

3.1、思路明细

（1）LVM这种备份方式要求Mysql的数据保存在逻辑卷上
（2）需要给Mysql服务器施加读锁（mysql>FLUSH TABLES WITH READLOCK;），这里不可直接退出服务器
（3）另起终端为数据所在的卷创建快照（lvcreate），保证事务日志和数据文件必须在同一卷上（分别创建可能会导致数据文件和事务日志不一致，从而可能导致无法正常恢复）

3.2、备份策略

LVM快照全备+二进制日志增备（对于即时点恢复还要恢复至后续的二进制位置）

3.3、前提条件

（1）创建逻辑卷及挂载逻辑卷，此过程在此就不做演示了

（2）初始化mysql将其数据目录指向/mydata/data

?

```
[root@stu18 ~]
# cd /usr/local/mysql/
[root@stu18 mysql]
# scripts/mysql_install_db --user=mysql --datadir=/mydata/data
```

（3）编辑查看配置文件，重启服务

?

```
[root@stu18 mysql]
# vim /etc/my.cnf
datadir = /mydata/data
#查看此项是否定义数据目录位置
sync_binlog=1
#添加此项，每个事务提交时候，把事务日志从缓存区写到日志文件中，并且刷新日志文件的数据到磁盘上；
[root@stu18 mysql]
# service mysqld start
```

3.4、过程展示

（1）确保事务日志和数据文件必须在同一卷上

?

```
[root@stu18 ~]
# ls /mydata/data/
hellodb  myclass   mysql-bin.000003 stu18.magedu.com.err
ibdata1  mysql     mysql-bin.000004 stu18.magedu.com.pid
ib_logfile0 mysql-bin.000001 mysql-bin.index  student
ib_logfile1 mysql-bin.000002 performance_schema test
```

解析：其中ib_logfile0与ib_logfile1是日志文件
（2）施加全局锁并滚动日志

?

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> FLUSH LOGS;
```

（3）查看并保存当前正在使用的二进制日志及当前执行二进制日志位置（非常重要）

?

```
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File     | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000004 |  187 |     |     |
+------------------+----------+--------------+------------------+
[root@stu18 zhao]
# mysql -uroot -pmypass -e 'SHOW MASTER STATUS;' >/zhao/lvmback-2013-08-14/binlog.txt
```

（4）创建快照卷

?

```
[root@stu18 zhao]
# lvcreate -L 100M -s -p r -n mydata-lvm /dev/vg1/mydata
```

（5）立即切换终端释放锁

?

```
mysql> UNLOCK TABLES;
```

（6）备份数据

?

```
[root@stu18 data]
# cp -a * /zhao/lvmback-2013-08-14/
```

（7）二进制实现增量备份

?

```
mysql> use hellodb;
#指定默认数据库
Database changed
mysql> CREATE TABLE testtb (id int,name CHAR(10));
#创建表
Query OK, 0 rows affected (0.35 sec)
mysql> INSERT INTO testtb VALUES (1,'tom');
#添加数据
Query OK, 1 row affected (0.09 sec)
[root@stu18 data]
# mysqlbinlog --start-position=187 mysql-bin.000004 > /zhao/lvmlogbin_2013-08-14/binlo
g.sql   #日志实现增量备份
```

（8）模拟数据库崩溃

?

```
[root@stu18 ~]
# service mysqld stop
[root@stu18 ~]
# cd /mydata/data/
[root@stu18 data]
# rm -rf *
```

（9）恢复数据

?

```
[root@stu18 ~]
# cp /zhao/lvmback-2013-08-14/* /mydata/data/ -a   #完全备份恢复
[root@stu18 ~]
# cd /mydata/data/    #查看恢复数据内容
[root@stu18 data]
# chown -R mysql.mysql * #更改属主属组
```

```
[root@stu18 data]
# service mysqld start  #启动服务
[root@stu18 data]
# mysql -uroot –pmypass #登录测试
```

?

```
mysql> SHOW DATABASES;
#查看数据完整性，无测试表testtd使用二进制恢复
mysql> SET sql_log_bin=0
#关闭二进制日志
mysql> source /zhao/lvmlogbin_2013-08-14/binlog.sql;
#二进制恢复
mysql> SHOW TABLES;
#查看恢复结果
+-------------------+
| Tables_in_hellodb |
+-------------------+
| classes   |
| coc    |
| courses   |
| scores   |
| students   |
| teachers   |
| testtb   |
| toc    |
+-------------------+
mysql> SET sql_log_bin=1;
#开启二进制日志
```

此工具是接近于热备的方式实现的，并且用此方法来备份恢复数据速度是非常快的。

四:基于xtrabackup来实现备份恢复

4.1、优势特性

完全以热备的形式进行，能够实现快速可靠地完全备份和部分备份，支持增量备份，支持时间点还原，备份过程中不会打扰到事务操作，能够实现网络传输和压缩功能从而有效的节约磁盘空间，备份完成后可自动验证数据是否可用，恢复速度较快等等。更多优势特性请参考http://www.percona.com/software/percona-xtrabackup

注意：以上这些优势特性只能在InnoDB引擎上完美实现，而在MyISAM存储引擎上依然最多只能使用温备的形式进行并且还不支持增量备份。
另外Xtrabackup更多的高级功能还依赖于Mysql数据库对于InnoDB实现了单独的表空间，否则也就没有办法实现单表导入导出查看方式如下：

?

```
mysql> SHOW GLOBAL VARIABLES LIKE 'innodb_file%';
+--------------------------+----------+
| Variable_name    | Value |
+--------------------------+----------+
| innodb_file_format   | Antelope |
| innodb_file_format_check | ON |
| innodb_file_format_max | Antelope |
| innodb_file_per_table | ON |
+--------------------------+----------+
```

其中的innodb_file_per_table为ON则表示实现了单表单空间。若为OFF则需要使用mysqldump全备然后更改配置文件删除原来的数据文件并重新初始化服务器最后将数据重新导入。所以建议以后在安装Mysql服务器时将其选项默认设置成1即可（innodb_file_per_table = 1）。单表单空间的数据显示形式为：

?

```
[root@stu18 hellodb]
# ls
classes.frm coc.MYD  courses.MYI scores.MYI teachers.frm testtb.ibd
```

```
classes.MYD coc.MYI  db.opt  students.frm teachers.MYD toc.frm
classes.MYI courses.frm scores.frm students.MYD teachers.MYI toc.MYD
coc.frm  courses.MYD scores.MYD students.MYI testtb.frm toc.MYI
```

4.2、安装Xtrabackup

下载percona-xtrabackup最新的版本为2.1.4（percona-xtrabackup-2.1.4-656.rhel6.x86_64.rpm）安装：

?

```
[root@stu18 ~]
# rpm -ivh percona-xtrabackup-2.1.4-656.rhel6.x86_64.rpm
```

若有错误无法安装请安装perl-DBD-mysql依赖包

?

```
[root@stu18 ~]
# yum -y install perl-DBD-mysql
```

注意：不同的环境依赖的关系包可能有多个，请依照提示进行配置

4.3、完全备份

使用innobakupex备份时，其会调用xtrabackup备份所有的InnoDB表，复制所有关于表结构定义的相关文件(.frm)、以及MyISAM、MERGE、CSV和ARCHIVE表的相关文件，同时还会备份触发器和数据库配置信息相关的文件。这些文件会被保存至一个以时间命令的目录中。完全备份命令如下：

?

```
# innobackupex --user=DBUSER--password=DBUSERPASS /path/to/BACKUP-DIR/
```

实现过程及说明：

?

```
<P>[root@stu18 ~]
# mkdir /innobackup    #创建备份文件目录
[root@stu18 ~]
# innobackupex --user=root --password=mypass /innobackup/ #完全备份
################如果执行正确其后输出的几行信息通常如下###############
xtrabackup: Transaction log of lsn (1604655) to (1604655) was copied.
#二进制日志的位置（lsn）
130814 07:04:55 innobackupex: All tables unlocked
innobackupex: Backup created in directory '/innobackup/2013-08-14_07-04-49'
#备份文件保存的位置
innobackupex: MySQL binlog position: filename 'mysql-bin.000003', position 538898
130814 07:04:55 innobackupex: Connection to database server closed
130814 07:04:55 innobackupex: completed</P>
```

OK!　　备份完成
切换至备份文件目录查看备份的数据信息及创建生成的文件：

?

```
<P>[root@stu18 ~]
# cd /innobackup/2013-08-14_07-04-49/
[root@stu18 2013-08-14_07-04-49]
# ls
backup-my.cnf myclass    student   xtrabackup_binlog_info
hellodb  mysql    test  xtrabackup_checkpoints
ibdata1  performance_schema xtrabackup_binary xtrabackup_logfile</P>
```

针对文件解析：

(1)xtrabackup_checkpoints —— 备份类型（如完全或增量）、备份状态（如是否已经为prepared状态）和LSN(日志序列号)范围信息；
每个InnoDB页(通常为16k大小)都会包含一个日志序列号，即LSN。LSN是整个数据库系统的系统版本号，每个页面相关的LSN能够表明此页面最近是如何发生改变的。
(2)xtrabackup_binlog_info —— mysql服务器当前正在使用的二进制日志文件及至备份这一刻为止二进制日志事件的位置。
(3)xtrabackup_binary —— 备份中用到的xtrabackup的可执行文件；
(4)backup-my.cnf —— 备份时用到的配置选项信息，也就是配置文件中关于mysqld的相关文件配置；
(5) xtrabackup_logfile —— 非文本文件是xtrabackup本身的日志文件；

4.4、准备一个完全备份

一般情况下，在备份完成后，数据尚且不能用于恢复操作，因为备份的数据中可能会包含尚未提交的事务或已经提交但尚未同步至数据文件中的事务。因此，此时数据文件仍处于不一致状态。“准备”的主要作用正是通过回滚未提交的事务及同步已经提交的事务至数据文件从而使得数据文件处于一致性状态。

innobakupex命令的–apply-log选项可用于实现上述功能。如下面的命令：

?

```
[root@stu18 ~]
# innobackupex -apply-log /innobackup/2013-08-14_07-04-49/
##############如果执行正确，其最后输出的几行信息通常如下#################
xtrabackup: starting shutdown with innodb_fast_shutdown = 1
130814 7:39:33 InnoDB: Starting shutdown...
130814 7:39:37 InnoDB: Shutdown completed; log sequence number 1606156
130814 07:39:37 innobackupex: completed OK!
```

4.5、模拟数据库崩溃实现完全恢复

（1）模拟崩溃

?

```
[root@stu18 ~]
# service mysqld stop
[root@stu18 ~]
# cd /mydata/data/
[root@stu18 data]
# rm -rf *
```

（2）从完全备份中恢复数据（谨记：在恢复数据之前千万不可初始化数据库和启动服务）
innobackupex命令的–copy-back选项用于执行恢复操作，其通过复制所有数据相关的文件至mysql服务器DATADIR目录中来执行恢复过程。innobackupex通过backup-my.cnf来获取DATADIR目录的相关信息。

?

```
[root@stu18 ~]
# innobackupex --copy-back /innobackup/2013-08-14_07-04-49/
##############如果执行正确，其最后输出的几行信息通常如下#################
innobackupex: Starting to copy InnoDB log files
innobackupex: in '/innobackup/2013-08-14_07-04-49'
innobackupex: back to original InnoDB log directory '/mydata/data'
innobackupex: Copying '/innobackup/2013-08-14_07-04-49/ib_logfile0' to '/mydata/data'
innobackupex: Copying '/innobackup/2013-08-14_07-04-49/ib_logfile1' to '/mydata/data'
innobackupex: Finished copying back files.
130814 07:58:22 innobackupex: completed OK!
```

（3）当数据恢复至数据目录以后，还需要确保所有数据文件的属主和属组均为正确的用户，如mysql，否则，在启动mysqld之前还需要事先修改数据文件的属主和属组。

?

```
# chown -R mysql:mysql /mydata/data/
```

（4）启动服务器登陆查看恢复完成。

?

```
[root@stu18 data]
# service mysqld start
```

注意：每一次恢复完成之后一定要重新做一次完全备份工作！！

4.6、使用innobackupex进行增量备份

说明：每个InnoDB的页面都会包含一个LSN信息，每当相关的数据发生改变，相关的页面的LSN就会自动增长。这正是InnoDB表可以进行增量备份的基础，即innobackupex通过备份上次完全备份之后发生改变的页面来实现。
第一次改动数据实现增量备份
实现增量备份可以使用下面的命令进行：

?

```
[root@stu18 data]
# innobackupex --user=root --password=mypass --incremental /innobackup --incremental-ba
sedir=/innobackup/2013-08-14_08-14-12/
```

其中，/innobackup指的是完全备份所在的目录，此命令执行结束后，innobackupex命令会在/backup目录中创建一个新的以时间命名的目录以存放所有的增量备份数据。–incremental-basedir是指向上一次完全备份所在的目录。

第二次改动数据进行增量备份：

?

```
[root@stu18 ~]
# innobackupex --user=root --password=mypass --incremental /innobackup --incremental-ba
sedir=/innobackup/2013-08-14_08-29-05/
```

第二次增量备份的执行命令和第一次大致相同，只有其–incremental-basedir应该指向上一次的增量备份所在的目录。

第三次改动数据还未进行增量备份

?

```
mysql> delete from coc where id=14;
```

4.7、使用innobackupex基于完全+增量+二进制日志恢复数据

（1）由于笔者这里将二进制日志和数据文件写在了同一个文件目录下所以在模拟数据库崩溃前必须先复制出二进制日志文件，所以建议看客们将数据目录和二进制目录分开存放，不要和笔者一样犯如此二的错误。方法如下：
前提是在刚刚建立服务器尚未启动服务器之前做如下操作；

?

```
mkdir /mybinlog
#建立一目录用于存放二进制日志
chown mysql:mysql /mybinlog
#更改权限
vim /etc/my.cnf
#修改配置文件
log-bin=/mybinlog/mysql-bin
#二进制日志目录及文件名前缀，添加之
```

好了言归正传复制二进制日志文件：

?

```
[root@stu18 data]
```

```
2 # cp mysql-bin.000001/innobackup/
```

（2）模拟服务器崩溃

?

```
1 [root@stu18 ~]
2 # service mysqld stop
3 [root@stu18 ~]
4 # cd /mydata/data/
5 [root@stu18 data]
6 # rm -rf *
```

（3）准备备份

首先注意“准备”增量备份与整理完全备份有着一些不同，尤其要注意的是：
1)需要在每个备份(包括完全和各个增量备份)上，将已经提交的事务进行“重放”。“重放”之后，所有的备份数据将合并到完全备份上。
2)基于所有的备份将未提交的事务进行“回滚”。

完全备份“准备”

?

```
1 [root@stu18 ~]
2 # innobackupex --apply-log --redo-only/innobackup/2013-08-14_08-14-12/
```

第一次增量备份“准备”也就是说将第一次增量备份合并到了完全备份中

?

```
1 [root@stu18 ~]
2 # innobackupex --apply-log--redo-only /innobackup/2013-08-14_08-14-12/--incremental-dir
=/innobackup/2013-08-14_08-29-05/
```

第二次增量备份“准备”也就是说将第二次增量备份也合并到了完全备份中

?

```
1 [root@stu18 ~]
2 # innobackupex --apply-log--redo-only /innobackup/2013-08-14_08-14-12/ --incremental-di
r=/innobackup/2013-08-14_09-08-39/
```

其中 –redo-only是只将已提交的事务同步到数据文件中，未提交的事务日志不在进行回滚了。

（4）恢复数据（基于innobackupex基于完全+增量）

?

```
1 [root@stu18 ~]
2 # innobackupex --copy-back/innobackup/2013-08-14_08-14-12/
```

（5）更改属组属主

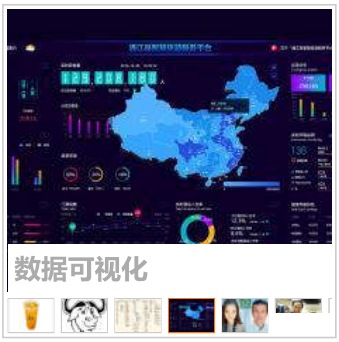
?

```
1 [root@stu18 ~]
2 # cd /mydata/data/
3 [root@stu18 data]
4 # chown -R mysql:mysql *
```

（6）启动查看

?

```
1 [root@stu18 ~]
2 # mysql -uroot -pmypas
3 mysql> select * from coc;
```




```
+----+---------+----------+
| ID | ClassID | CourseID |
+----+---------+----------+
| 1|  1 |  2 |
| 2|  1 |  5 |
| 3|  2 |  2 |
| 4|  2 |  6 |
| 5|  3 |  1 |
| 6|  3 |  7 |
| 7|  4 |  5 |
| 8|  4 |  2 |
| 9|  5 |  1 |
| 10 |  5 |  9 |
| 11 |  6 |  3 |
| 12 |  6 |  4 |
| 13 |  7 |  4 |
| 14 |  7 |  3 |
+----+---------+----------+
14 rows in set (0.00 sec)
```

结果显示数据正确完整，但是第三次的改动信息未生效。

（7）基于二进制日志实现数据恢复
查看最后一次增量备份二进制日志所在的位置：

?

```
[root@stu18 data]
# cd /innobackup/2013-08-14_09-08-39/
[root@stu18 2013-08-14_09-08-39]
# cat xtrabackup_binlog_info
mysql-bin.000001 780
```

查看二进制日志文件将未备份数据的二进制日志导出

?

```
[root@stu18 innobackup]
# mysqlbinlog mysql-bin.000001
# at 780
#130814 9:20:19 server id 1 end_log_pos 851 Query thread_id=7 exec_time=0 error_code=0
SET TIMESTAMP=1376443219/*!*/;
BEGIN
/*!*/;
# at 851
#130814 9:20:19 server id 1 end_log_pos 944 Query thread_id=7 exec_time=0 error_code=0
SET TIMESTAMP=1376443219/*!*/;
delete from coc where id=14
/*!*/;
# at 944
#130814 9:20:19 server id 1 end_log_pos 1016 Query thread_id=7 exec_time=0 error_code=
0
SET TIMESTAMP=1376443219/*!*/;
COMMIT
/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
[root@stu18 innobackup]
# mysqlbinlog --start-position=780 mysql-bin.000001 > ./all.sql  #导出数据
```

```
2
2
2
3
2
4
```

恢复数据

?

```
[root@stu18 ~]
# mysql -uroot -pmypass
mysql> SET SQL_LOG_BIN=0;
#关闭二进制日志
mysql> source /innobackup/all.sql
#导入数据
mysql> SET SQL_LOG_BIN=1;
#开启二进制日志
mysql> select * from coc;
#查看数据，恢复完成
+----+---------+----------+
| ID | ClassID | CourseID |
+----+---------+----------+
| 1 |  1 |  2 |
| 2 |  1 |  5 |
| 3 |  2 |  2 |
| 4 |  2 |  6 |
| 5 |  3 |  1 |
| 6 |  3 |  7 |
| 7 |  4 |  5 |
| 8 |  4 |  2 |
| 9 |  5 |  1 |
| 10 |  5 |  9 |
| 11 |  6 |  3 |
| 12 |  6 |  4 |
| 13 |  7 |  4 |
+----+---------+----------+
13 rows in set (0.00 sec)
```

这种备份恢复方式完全以热备的形式实现完全备份和增量备份和二进制日志还原数据，并且恢复速度也很快，是最佳的备份恢复方式！！

总结：以上三种备份恢复都是可以基于二进制日志文件进行的，因而体现出了二进制日志的重要性，从而映射出了日志的重要性；所以学习查看使用日志文件是学习Mysql的重中之重！

您还没有登录,请[登录]或[注册]

浅谈MySQL数据库备份的几种方法 English0523 2016年06月08日 15:19 10688

mysql常见的备份方式有:mysqldump、mysqlhotcopy、BACKUP TABLE 、SELECT INTO OUTFILE，又或者备份二进制日志（binlog），还可以是直接拷贝数据...

MySQL 单个数据库备份还原 kk185800961 2017年07月10日 10:06 1779

数据库备份还原 #单个数据库备份及压缩 mysqldump -uroot -pmysql --opt --databases --routines --events --flush-logs --s...