

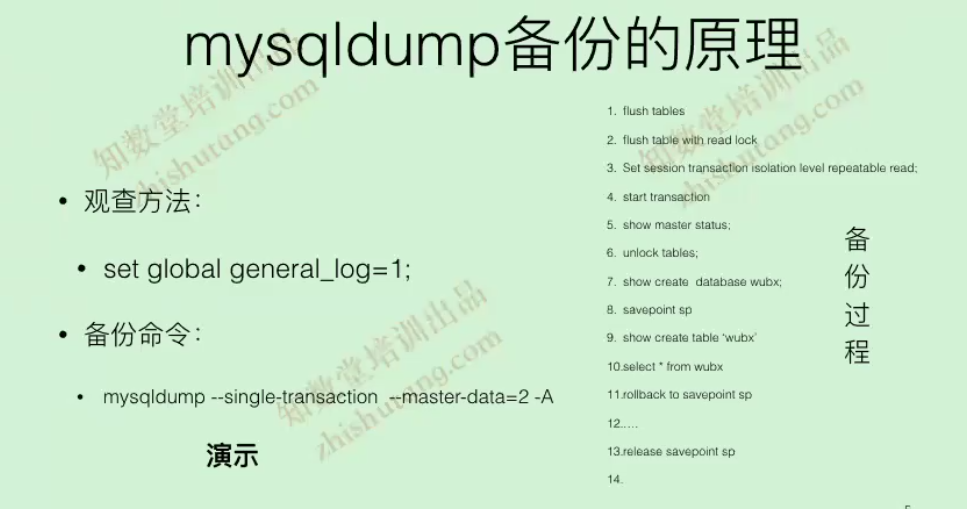


什么场景下使用什么备份，

经常需要传输数据或者给大数据平台可能使用mydumper比较多，mysqldump和mydumper实现原理一样，但是后者实现了并行备份。

增量备份建议使用binlog





Mysqldump原理：

通过sql和mysql进行交互，可以记录日志。

备份的时候一定要使用—single-transaction --master-data=2的参数，为什么要做这么做？

如果不适用参数无法一致性恢复。



不使用参数备份时什么样的流程？

假设一个库有10张表，mysqldump备份的时候使一张表一张表来备份的，备份每张表都需要时间，这是前提。

如果不适用参数，再备份table1的同时table2/3/4..10可能还在有数据变更，当备份完table1再备份table2，备份table1发生再时刻1，但是开始备份table2的时候是时刻2，同时table2可能发生数据变更，这样就不能保证数据一致性了。就不是一个完整一致性的数据库备份。

如果这个时候使用没有使用参数的备份文件来恢复数据库，可以恢复，但是无法保证一致性。

那么怎么可以做一个一致性的快照（备份）？

Mysql是支持事务的，可以开启一个事务，保存一个point，然后备份table1，备份完table1之后rollback到point，再备份table2，以此类推，直至数据库的表全部备份完毕。



Mysqldump 也是采用类似步骤来实现一致性备份；

1. flush tables；目的就是为了防止大事务再卡，备份时间和大事务时间进行错开。
2. flush table with read lock；FTWRL 做了这个动作，整个数据库是只读状态，会话无法写入数据，创造一个数据库相对静止状态。
3. set session transaction isolation level repeatable read ；做了这个操作就是把数据库的事务隔离级别改为RR操作。RR事务隔离级别可以保证一致性读。
4. start transaction 开启事务
5. show master status 获取当前状态，再GTID环境中还会获取gtid的状态
6. unlock tables ；做完这个操作，整个数据库恢复到可写状态
7. show create database xxx
8. savepoint sp ；创建一个保存点
9. ~10）备份每一张表

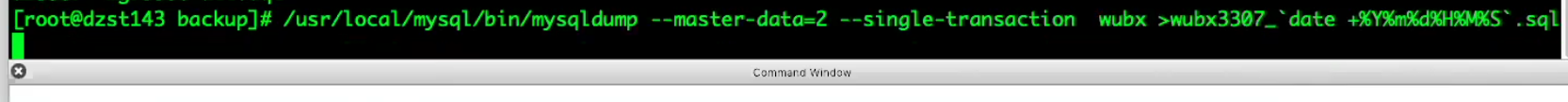
11）rollback to savepoint sp；回滚到保存点

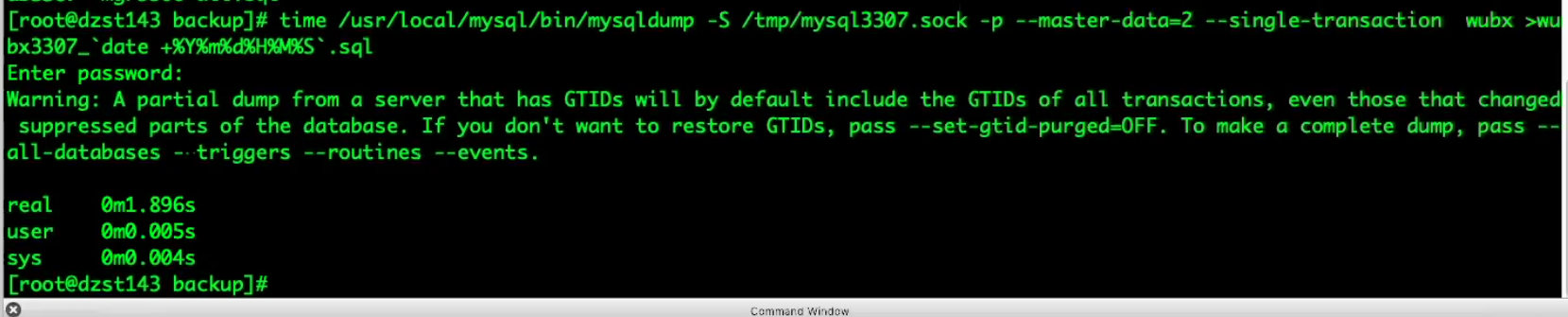
12）备份下一张表语句

13）release savepoint sp

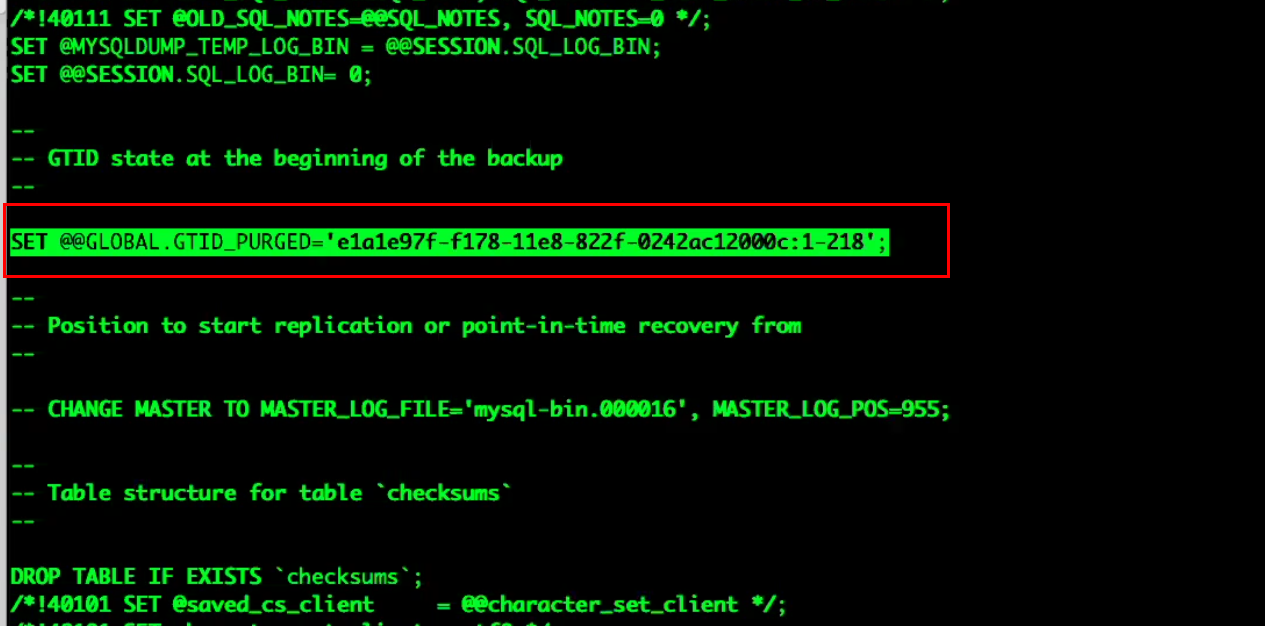
备份时间很长，假如从早上3点开始备份，备份了一个半小时，到4点30才完成备份，那么备份的数据是3点的还是4点半的？是3点的数据

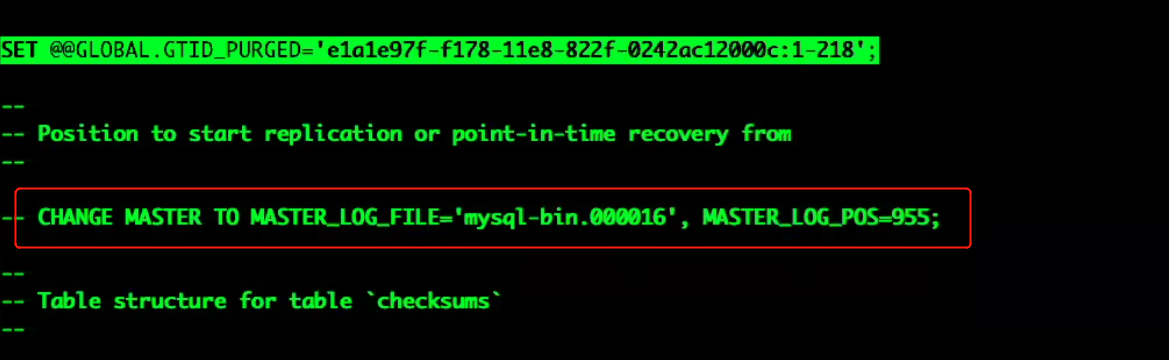
这里是MVCC 来保证的数据是3点的而不是4点半的





查看备份文件





因为备份没有创建数据库的语句，所以再恢复之前需要手动创建库。

==================骚气分割线=================

查看general.log，来查看mysqldump 和mysql交互的过程都做了什么。

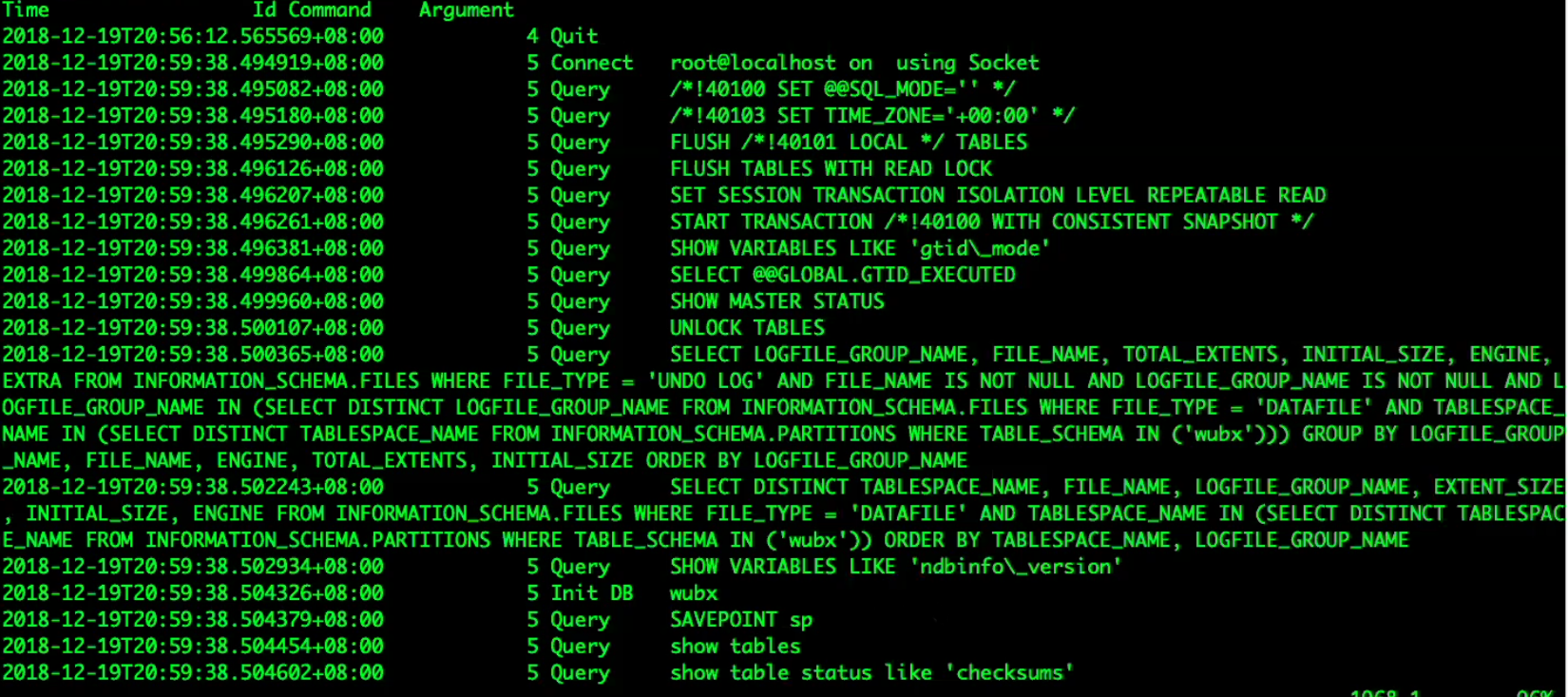


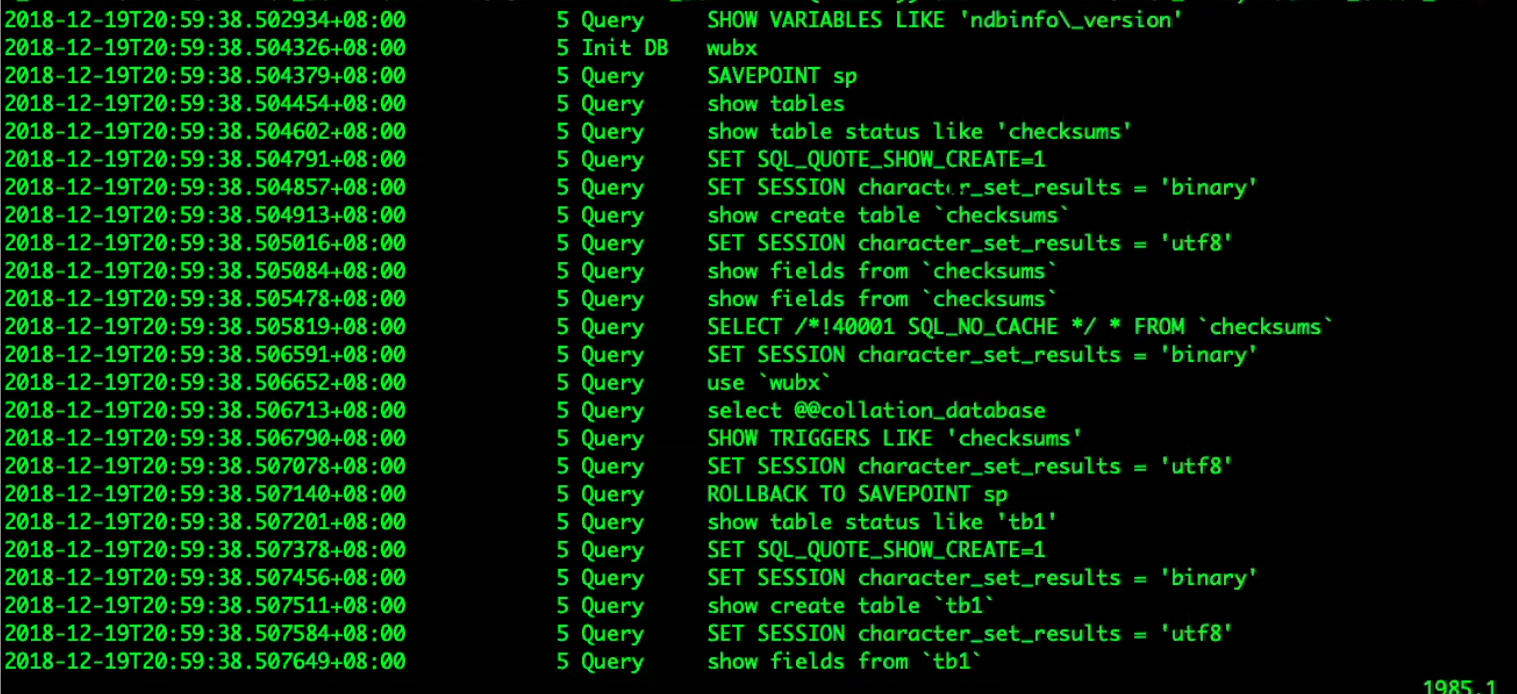
1）图中的/\*!40100 xxxx \*/

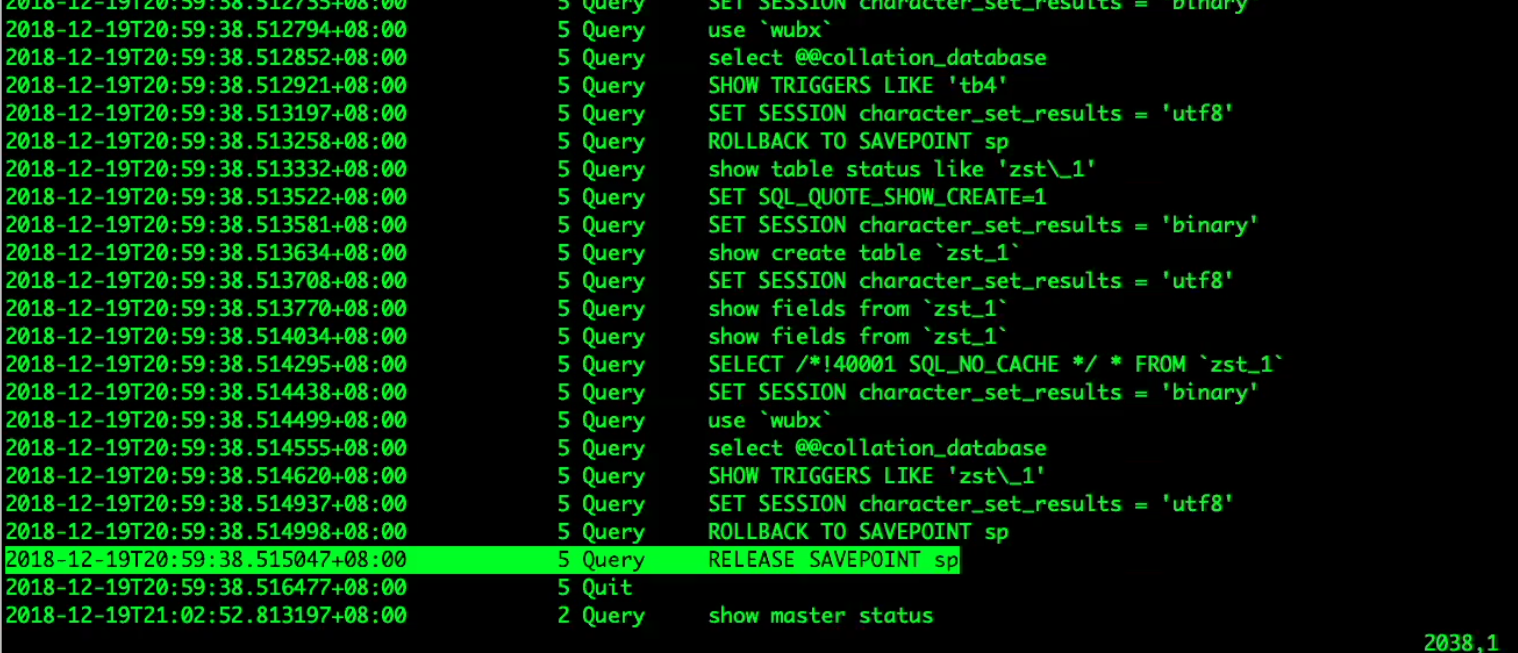
40100 是数据库的版本号，可以使用select @@version；来查看数据库的版本号；

/\*! 40100 代表的意思是 如果大于这个版本号就执行注释里面的内容

General.log里面的内容可以很好的看到mysqldump和mysql每一步的交互。可以把里面的语句拷贝出来自己去执行，可以更好的理解。



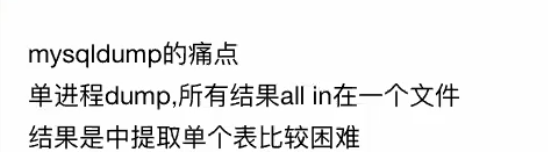




以上三张图可以很好的解释mysqldump备份的过程。

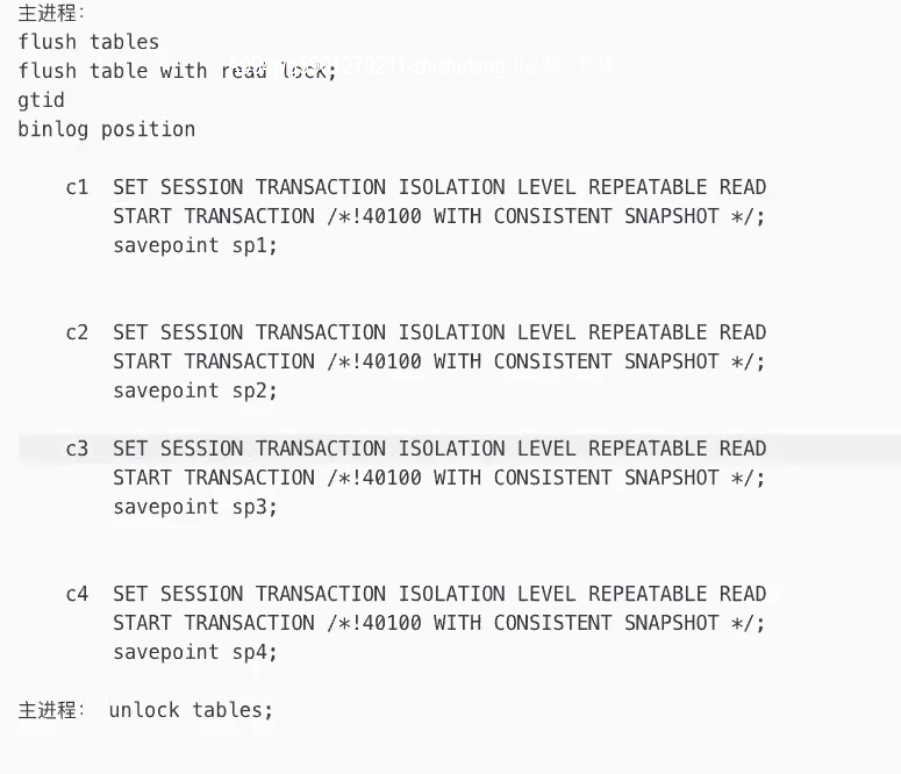






再实际工作中可以使用mysqldumper，这是一个开源工具。

Mysqldumper可以并行备份，mysqldumper 并行备份原理



再read lock状态进行多个链接，开启事务，创建保存点，目的就是创建一致性快照。

Mysqldumper 是mysqldump的增强。

=======================不怎么骚气的分割线====================

那么如果只需要一张表怎么做？

创建用户，创建库，再给用户权限的时候只给要备份表的权限即可。

恢复的时候一定会报错，但是使用-f即可。

不要使grep，一看就没有做过大项目。

=====================为下节课做准备==========================



LSN：每一个log都有一个唯一的序号





只要写到redo 就会有一个lsn号，每个lsn号会对应一个事务。

Last checkpoint at 持久化到datafile

Pages flushed up to 对应buffer pool

Log flushed up to 写到redo里面

Log sequence number 下一个事务的num，下一个lsn

掉电恢复：