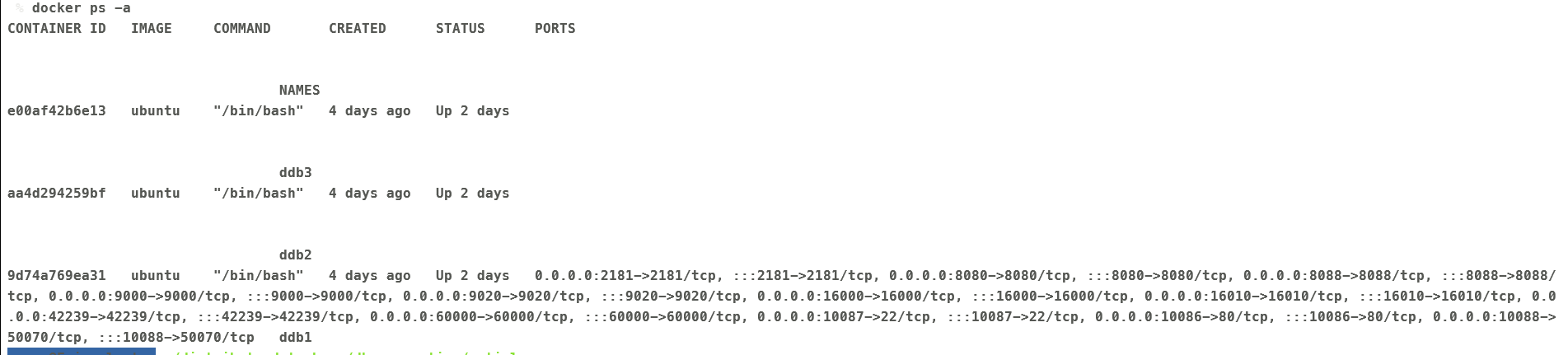
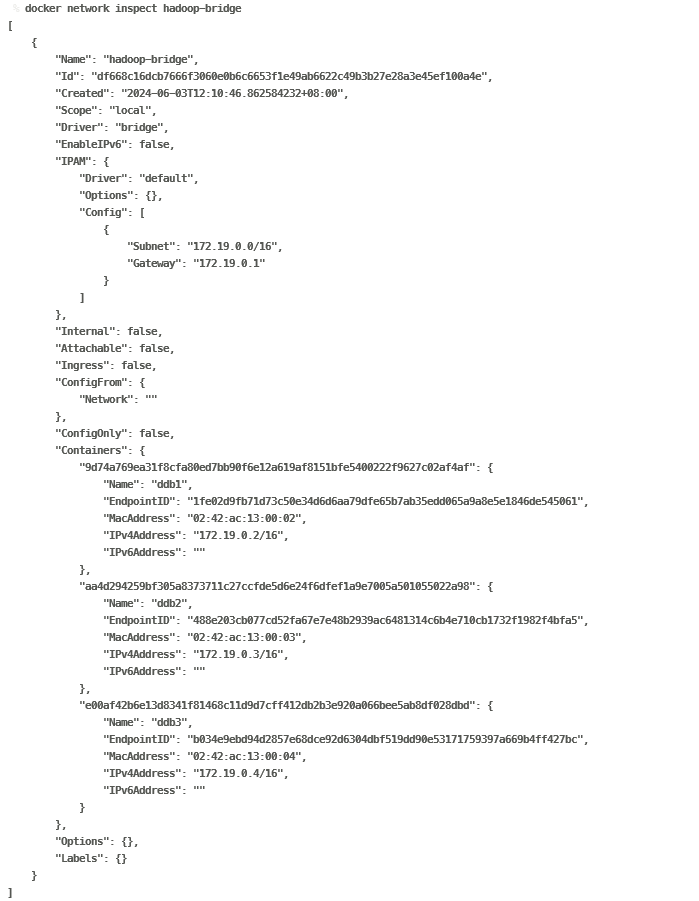
# Hadoop部署情况

我总共用了三个节点，分别是ddb1,ddb2,ddb3的三个docker container



他们的网络情况如下

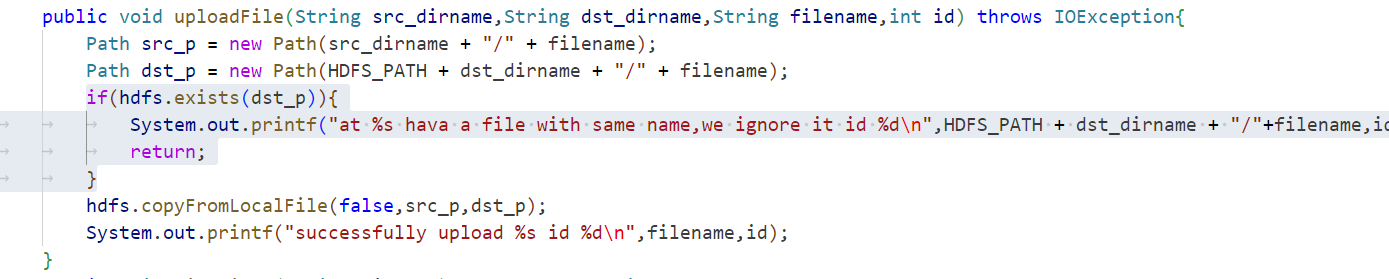


然后使用这三个分别作为hadoop1，hadoop2（同时也是DBMS1存放的节点），hadoop3（同时也是DBMS2存放的节点）

# 代码运行结果

在系统启动的时候，会初始化的将所有的文件都读入hadoop分布式文件系统中，并在两个节点中生成两个名为ddb的数据库，当然实际读入操作非常耗时，因此，在实际使用的时候，可以按照实际，查找是否已经upload该文件，如果已经upload，就不做处理。





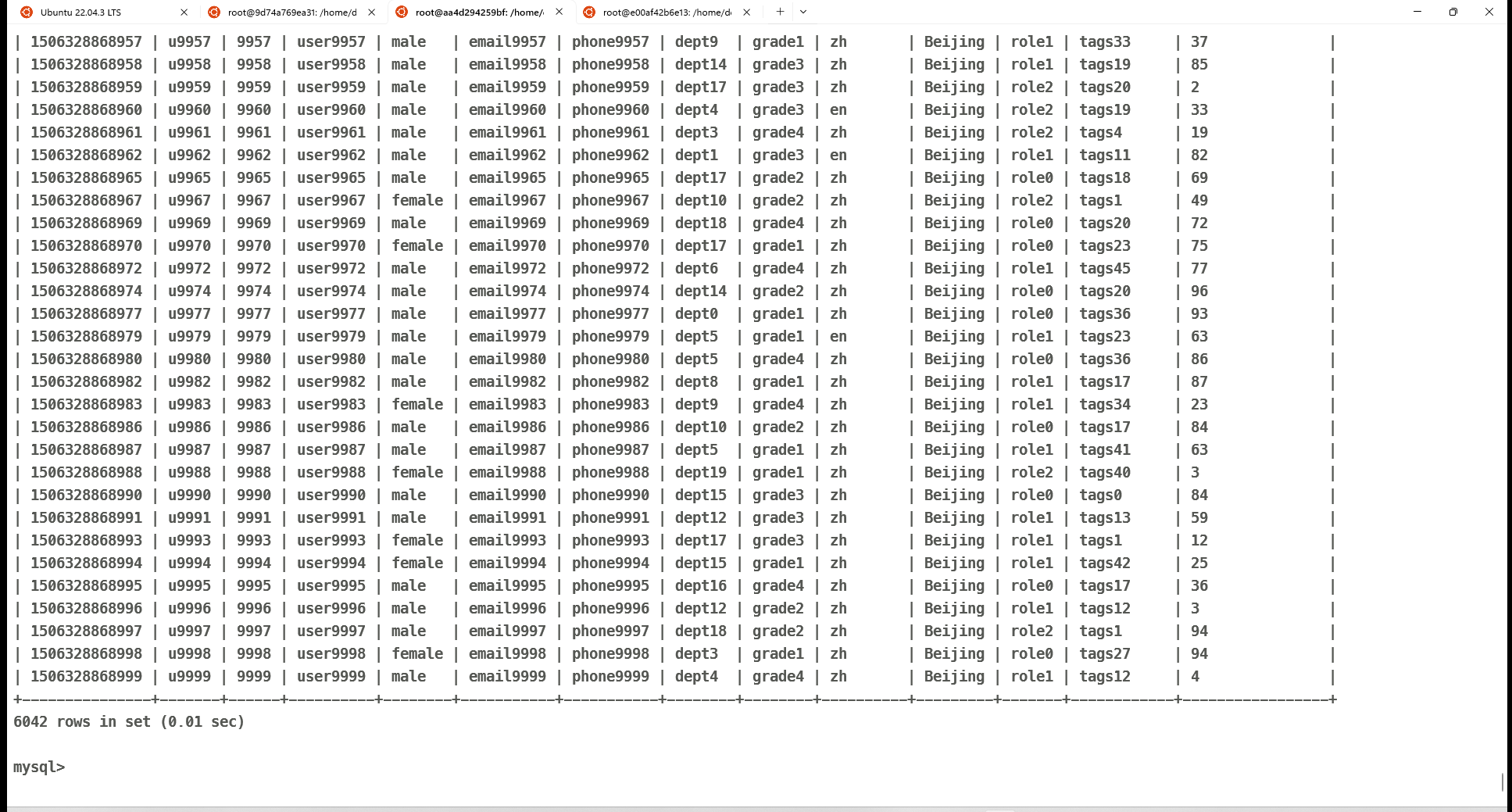
首先需要bulk insert user，article以及read三个表格，demo操作如下：

读取user表：

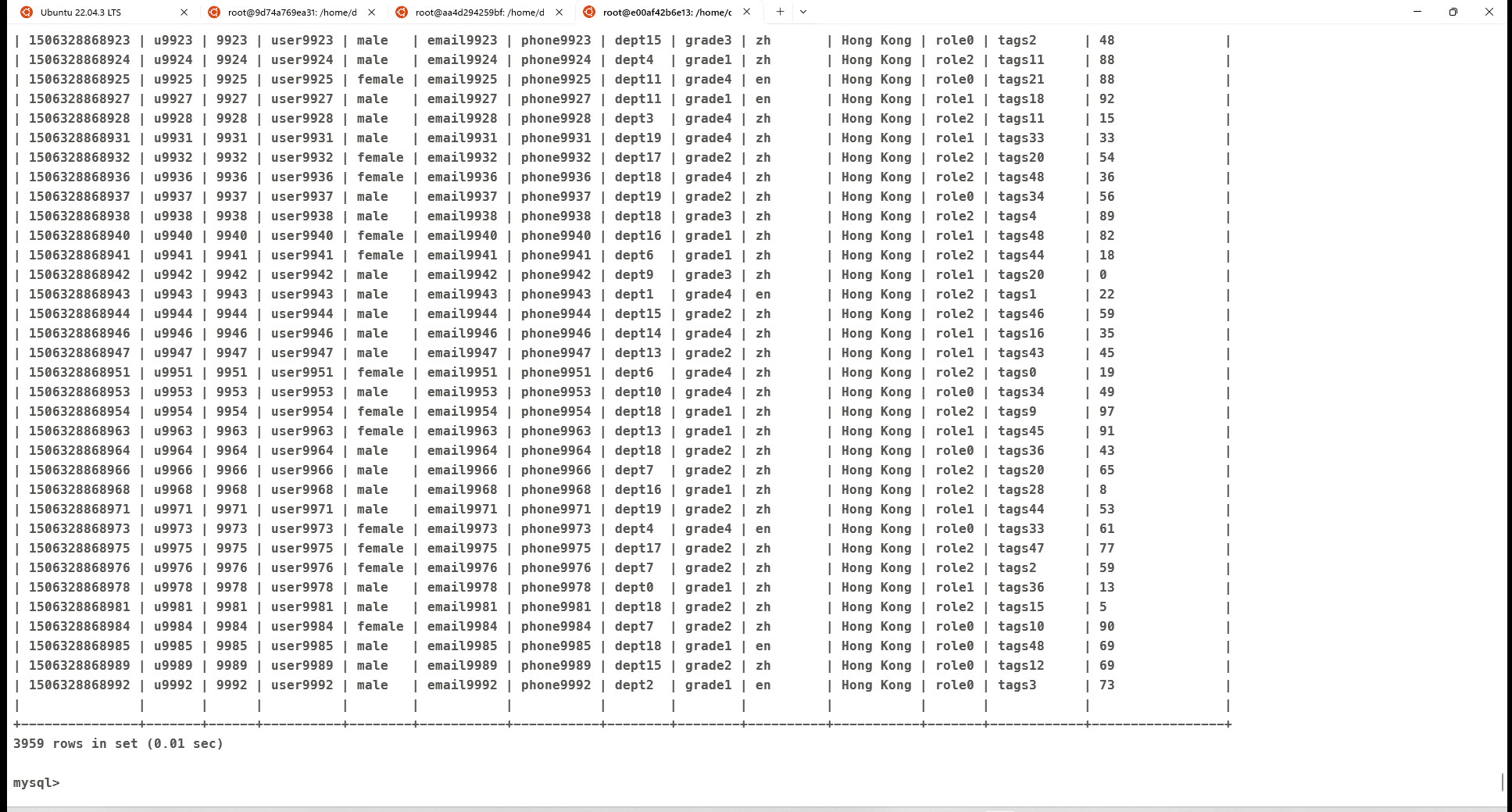
输入命令bulk user ../db-generation/user.sql



操作完成后，在DBMS的两个节点可以查看到

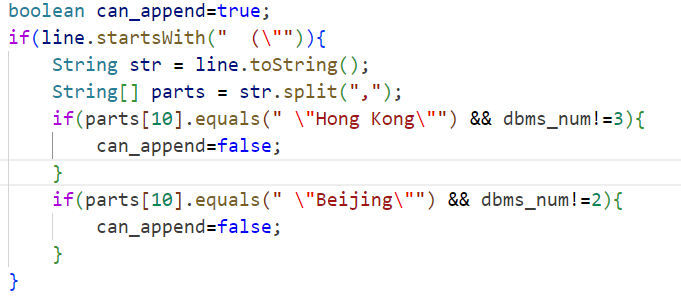


在节点1中，region一列均为Beijing



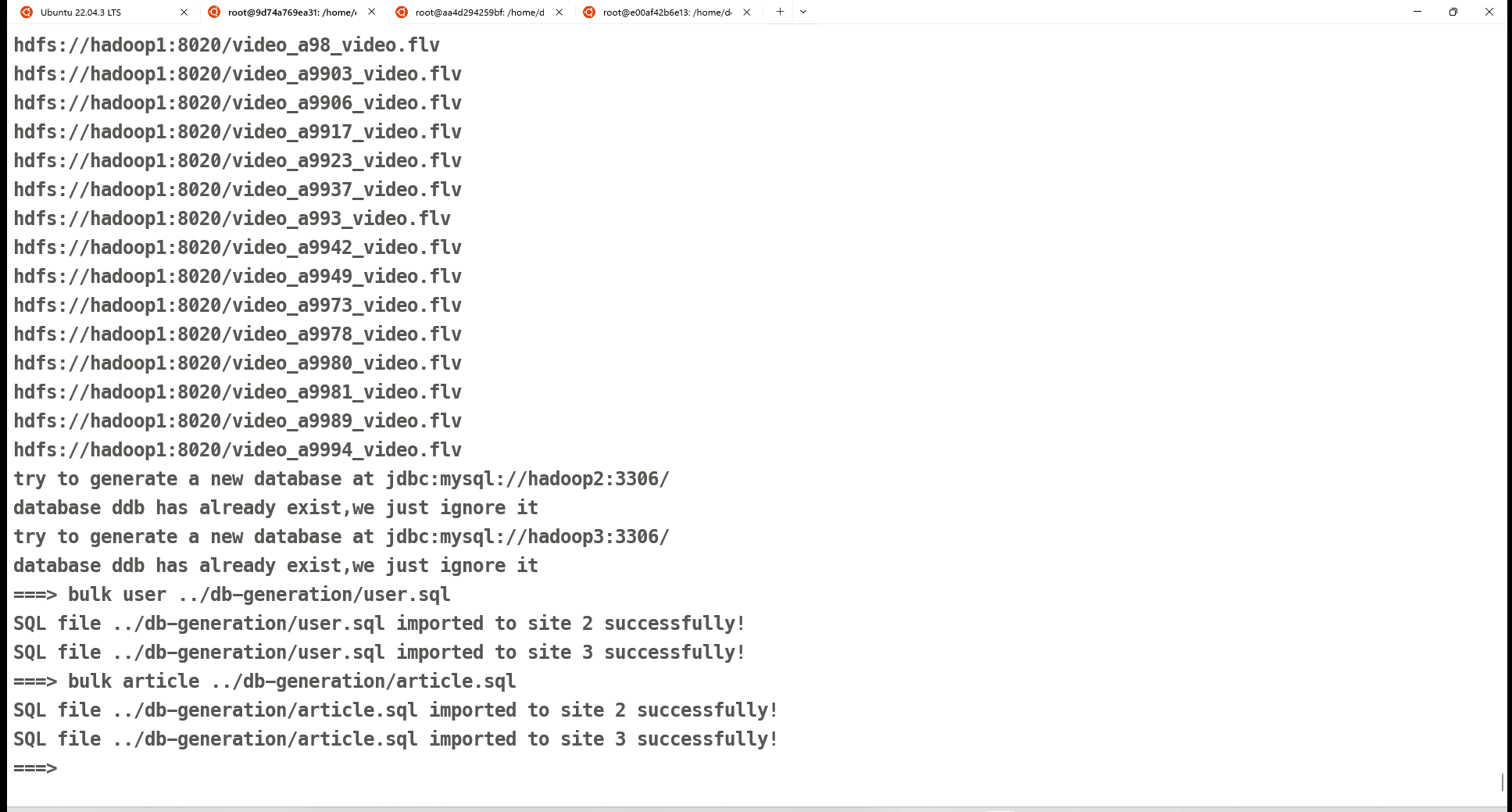
在节点2中，region一列均为Hong Kong

对于使用分片的方法，我在bulk从User.sql文件批量读入的时候，加以判断

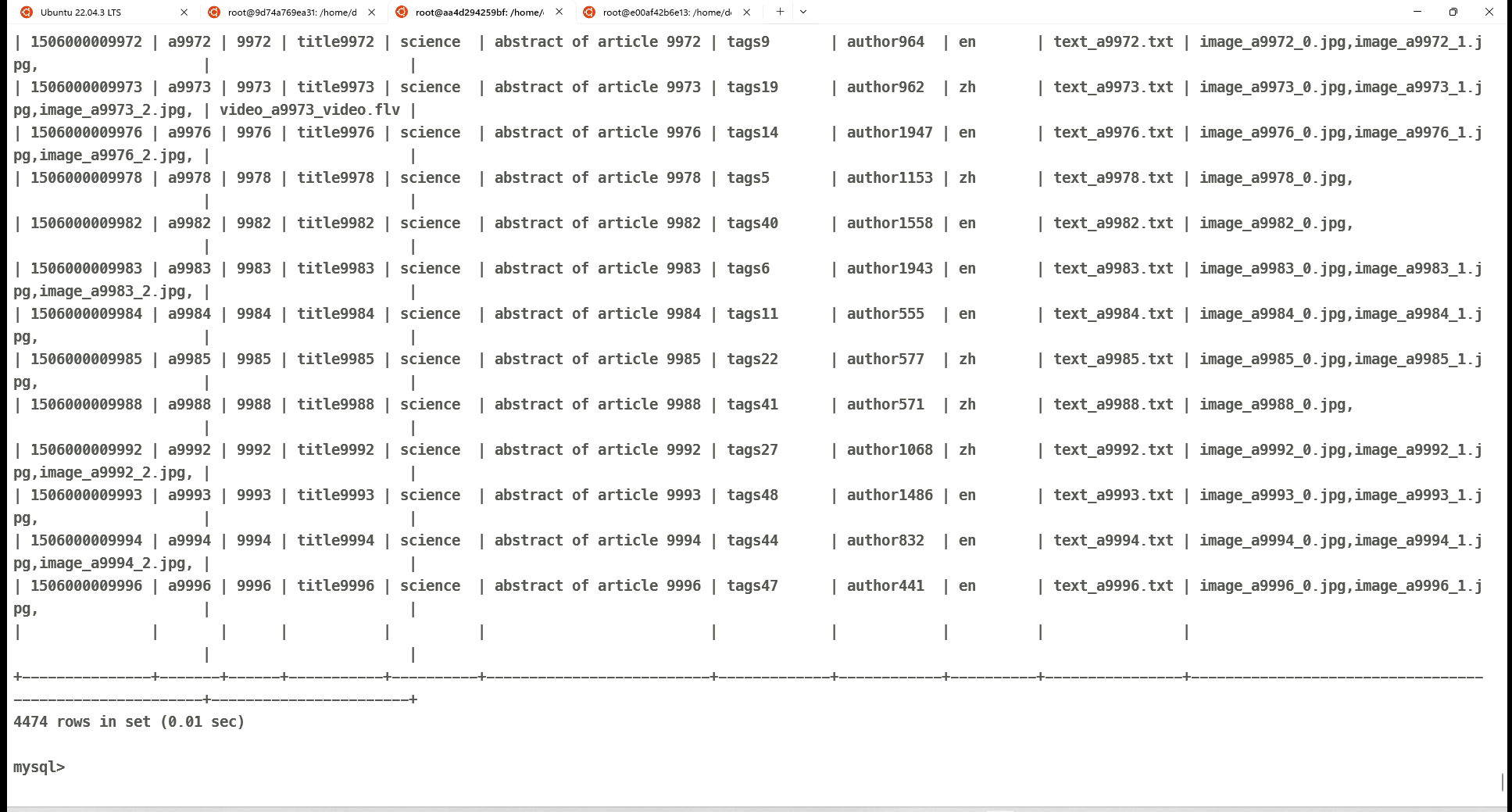


实现方法为查找对应的词条，并且查看是否是目标插入的数据库。

同样的输入bulk article ../db-generation/article.sql

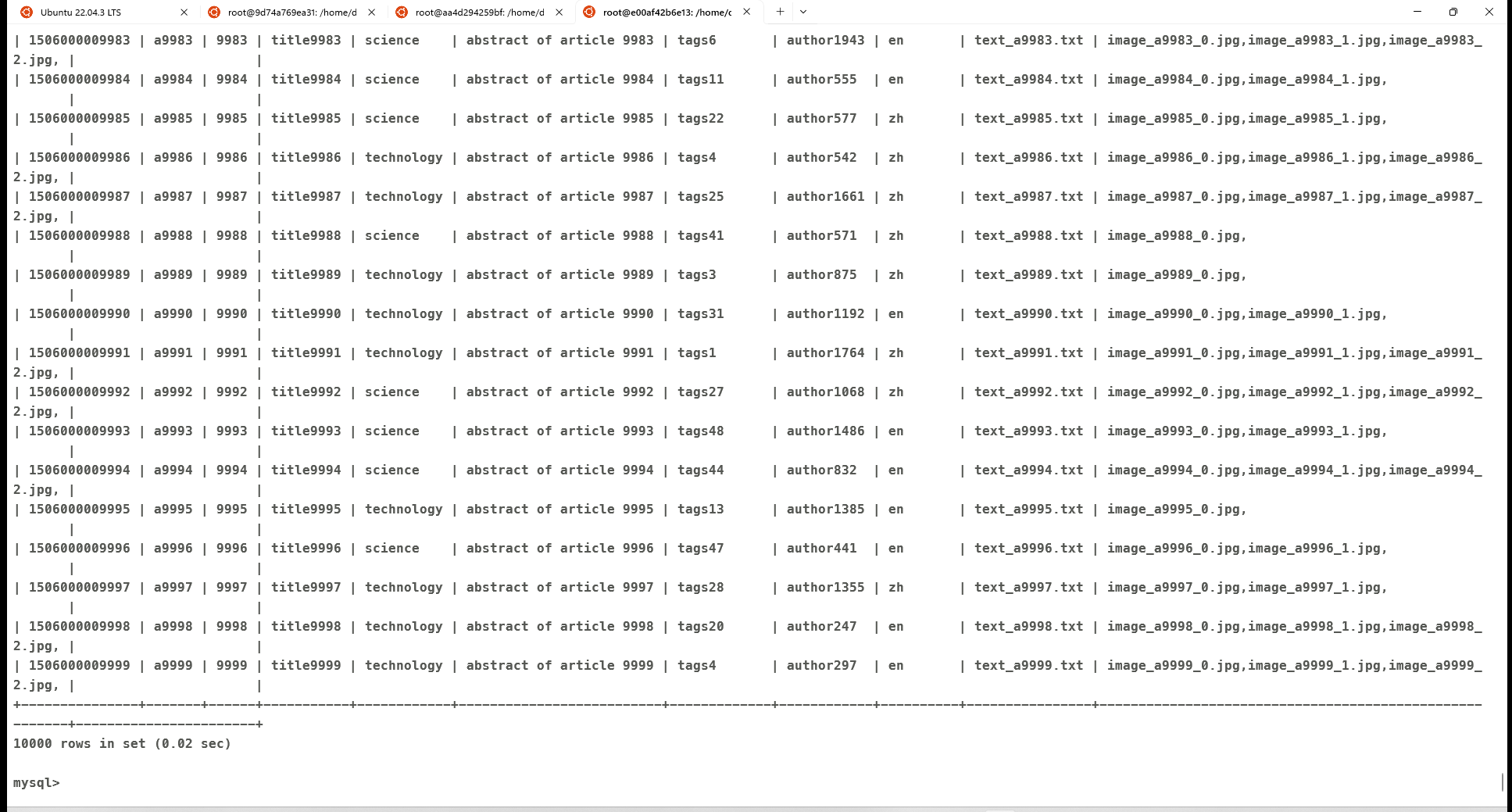


分别查看结果



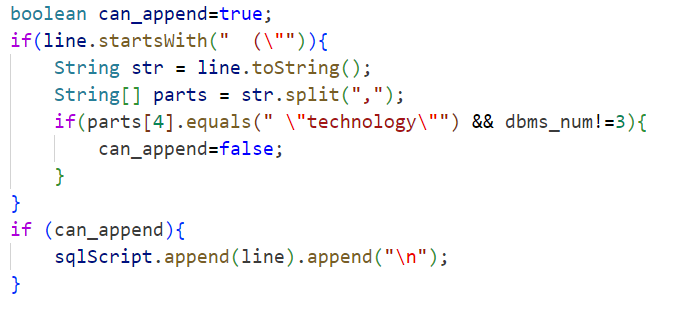
在节点1中只有category为science的记录

而在节点2中

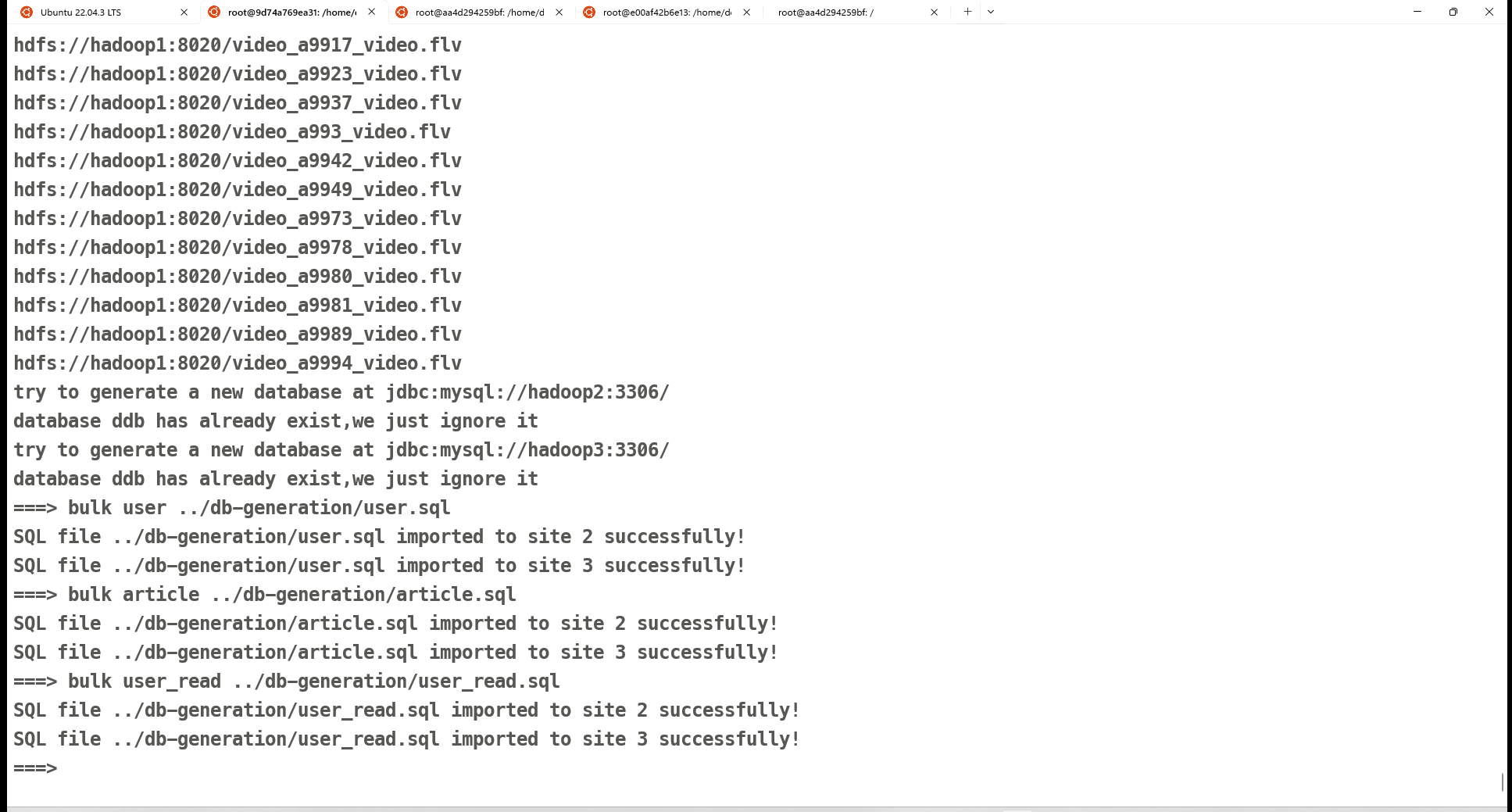


Category包括science和technology，有全部10000条记录。

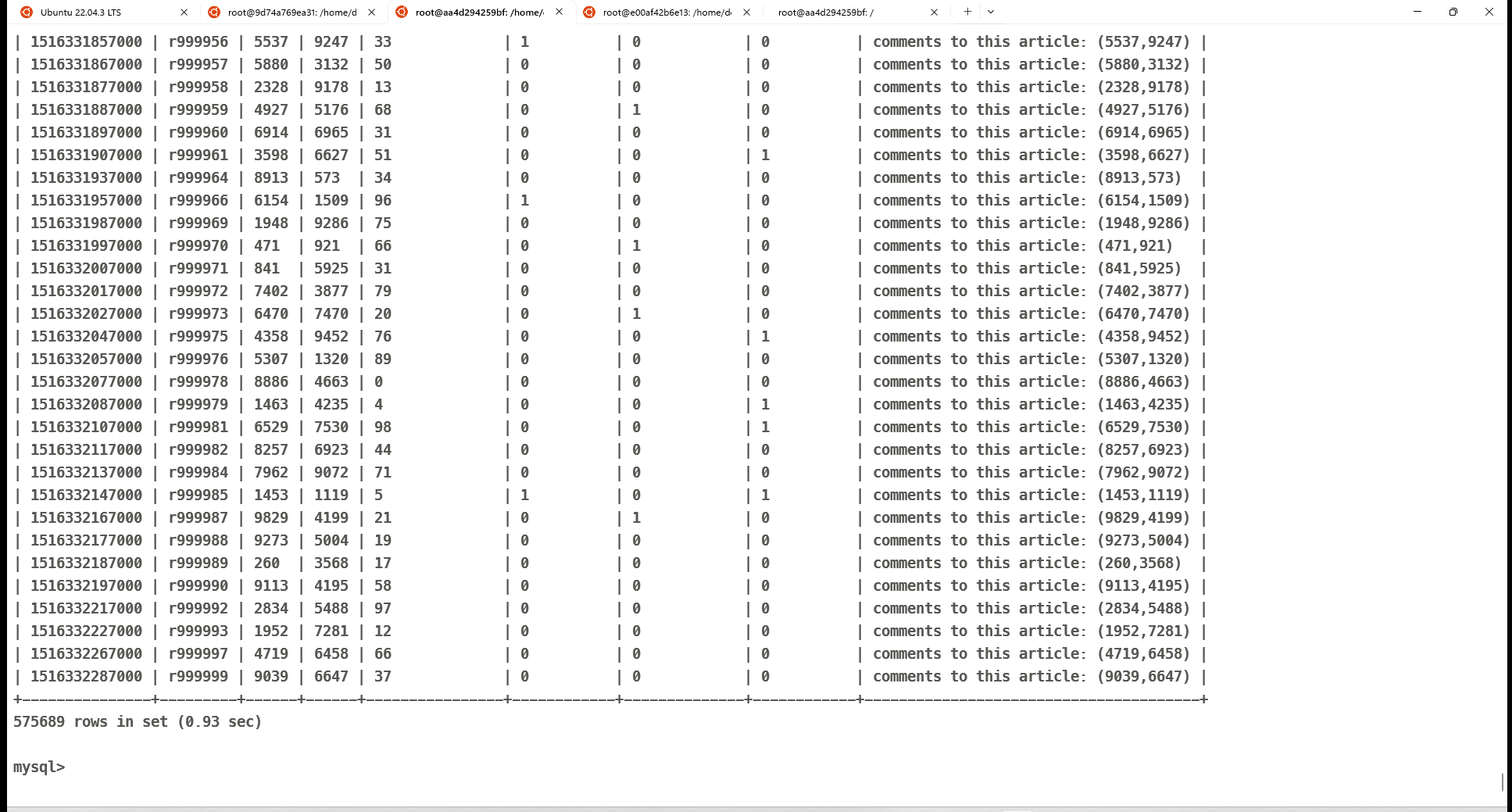
分片的方法也是类似的。



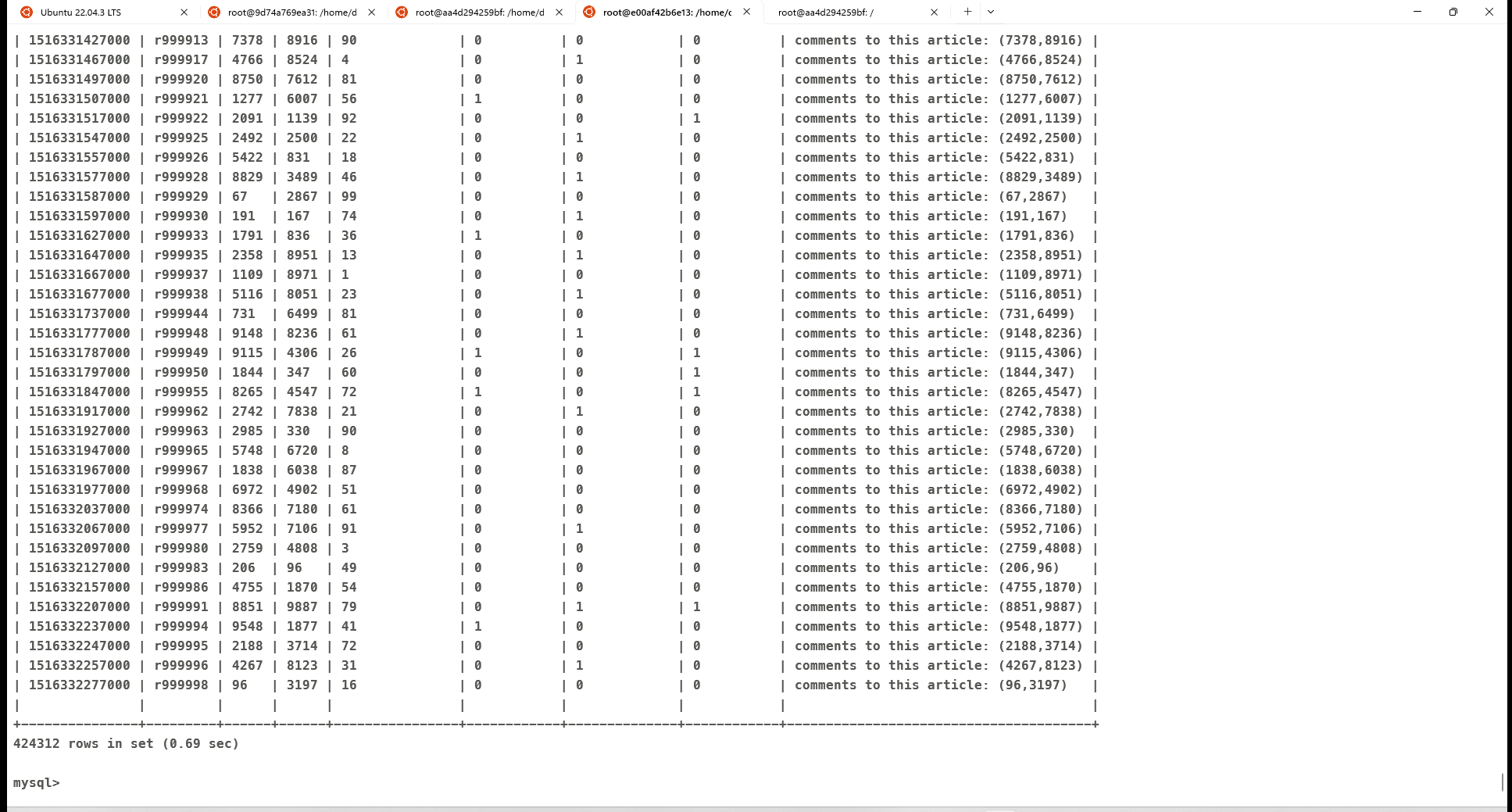
之后插入对应的user\_read表：



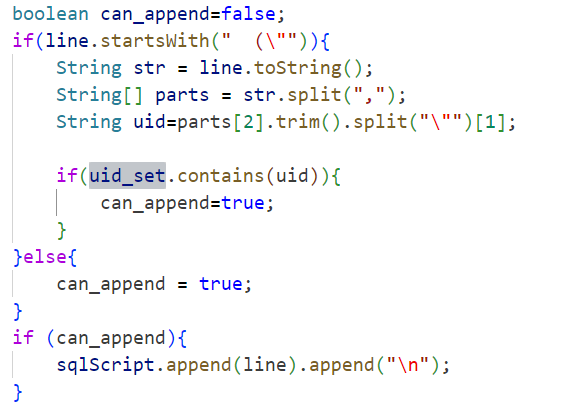
DBMS1的情况：根据user表分片，可以看到总共有575689条记录，



而DBMS2的情况，总共有424312条记录

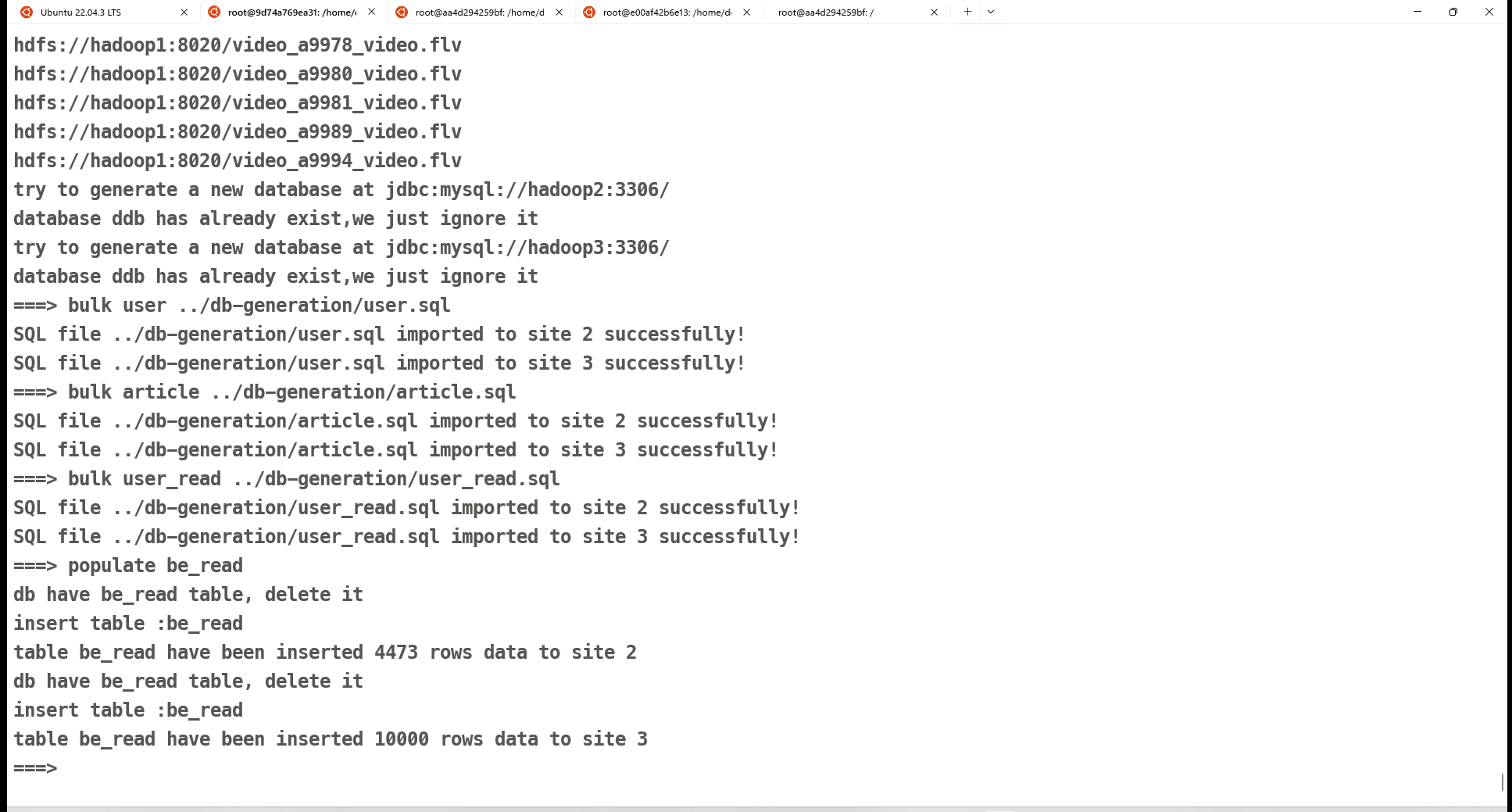


在对这部分的实现过程中，我查找了对应的user表，并记录为uid\_set，然后如果发现不包含对应的uid，说明他不在这个分片中，则不需要插入其中：



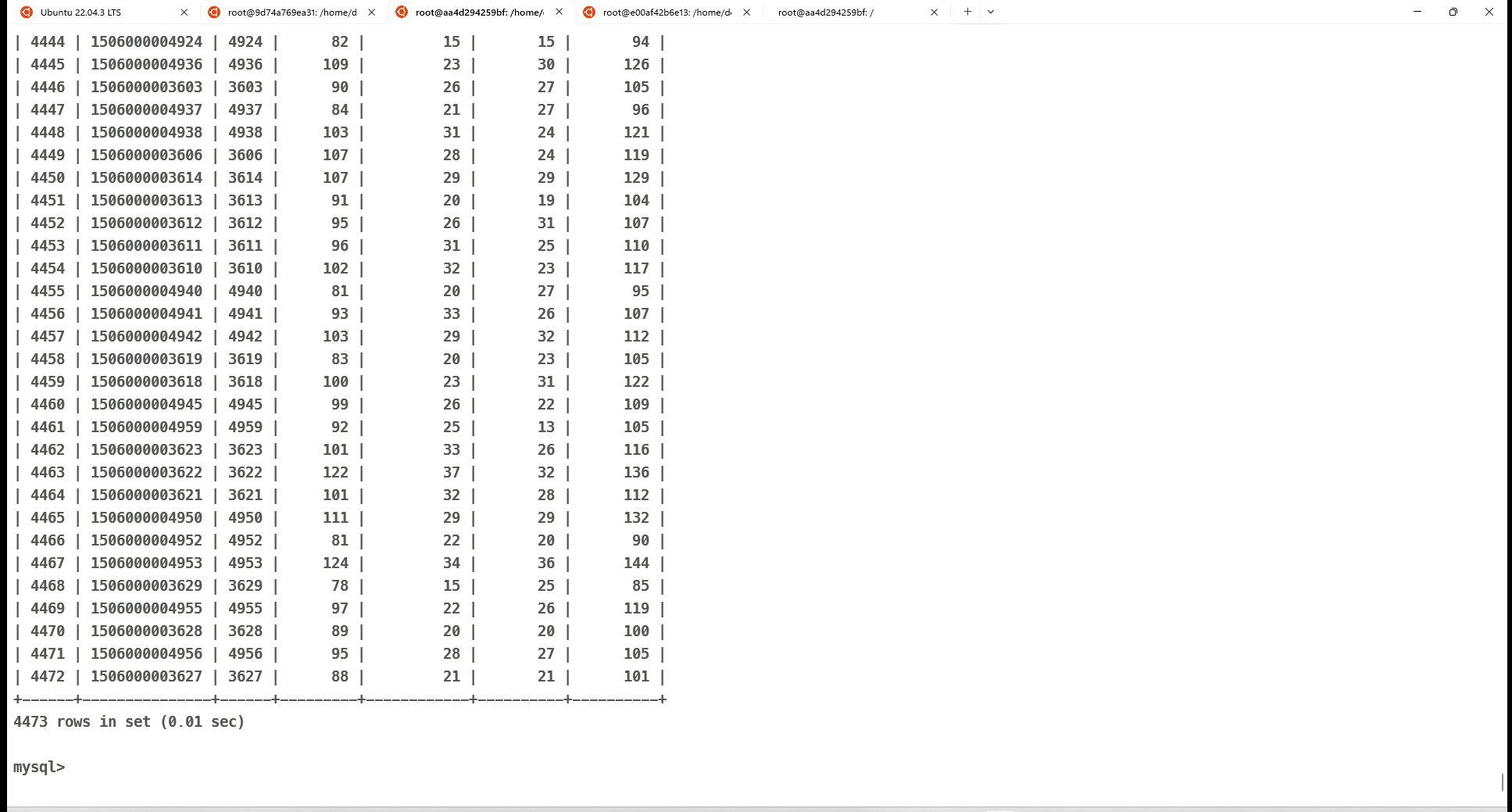
随后，根据对应的user表，article表以及user\_read表，生成be\_read表：

命令populate be\_read

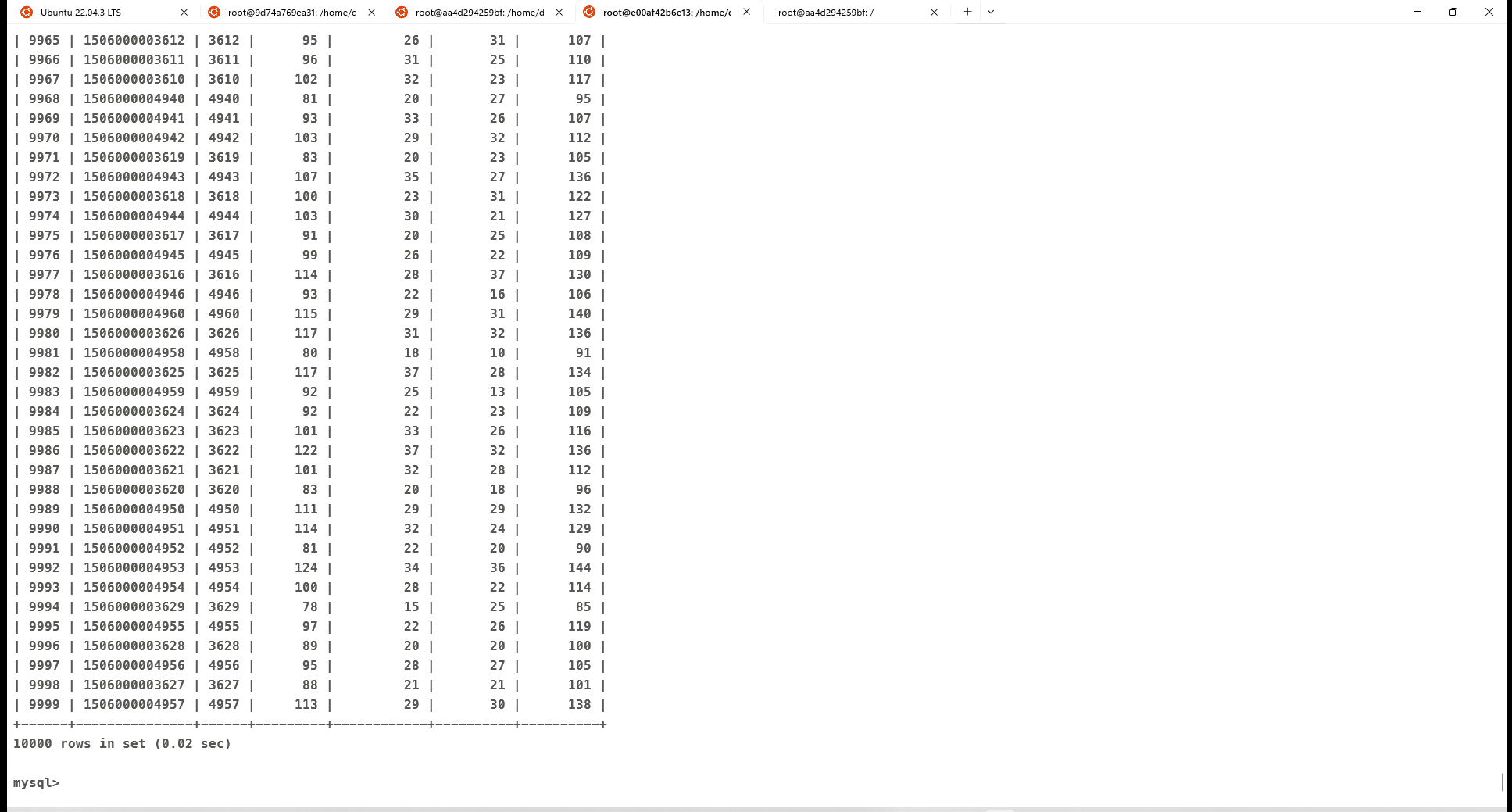


使用select id,timestamp,aid,readNum,commentNum,agreeNum,shareNum from be\_read;查找对应字段（因为整个表太长了，一行放不下，非常不清晰，只能选择部分字段打印了）

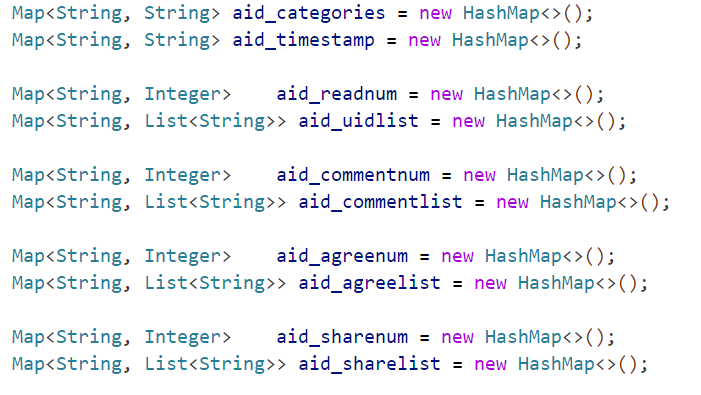
在DBMS1中，有4473条记录



而因为be\_read表和article表，采用了相同的分片的方法，因此有全部记录。

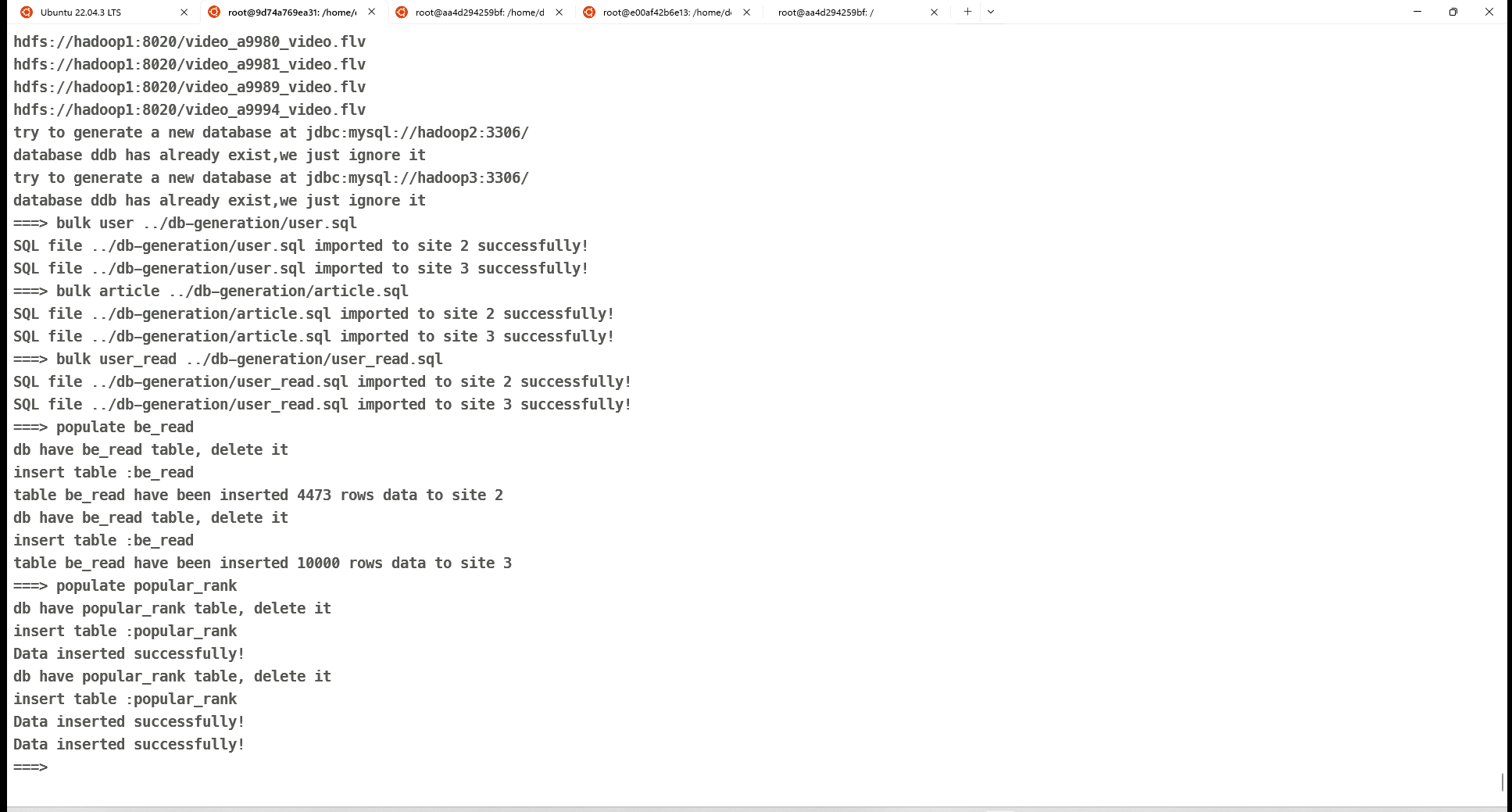


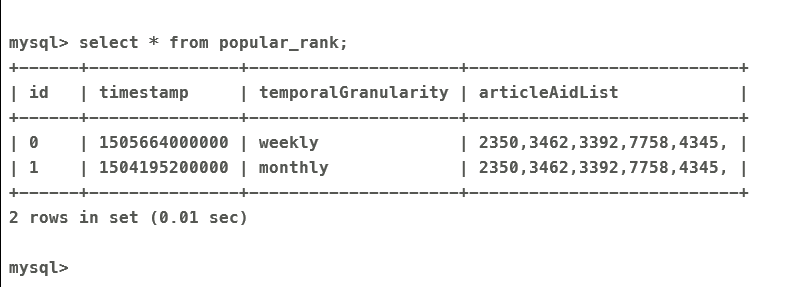
在这一步的实现中，我查找并维护了众多的hashmap，用于记录对于某一篇文章read，comment，agree，share的uid列表：

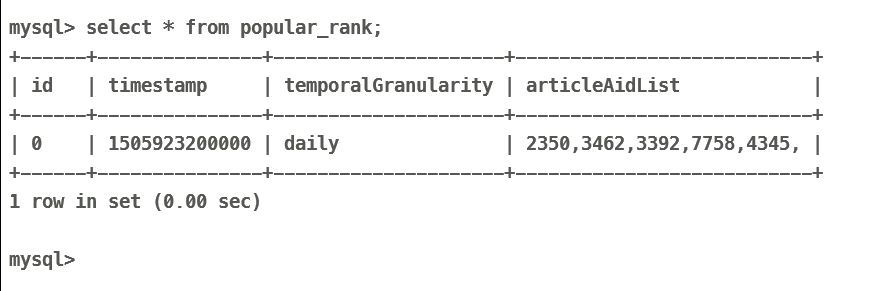


同时，由于他放在不同的分片中，我还综合了对两个数据库的查询，将结果存放到这些map中，最后组合成一条记录并插入。

最后是生成对应的popular\_rank表，使用命令populate popular\_rank

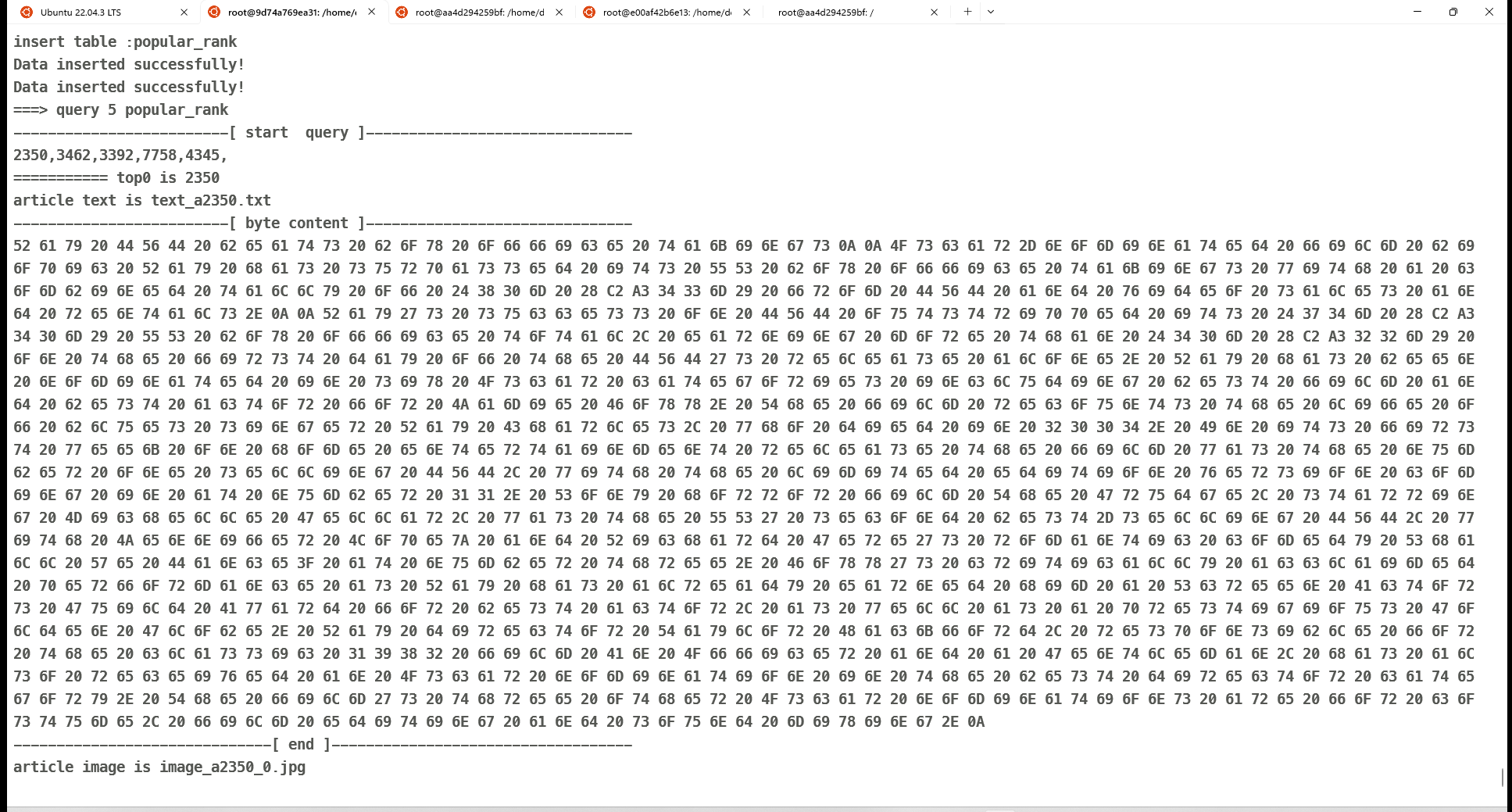






在这一步，分别放入不同的分片中去。并且计算出来最popular的5条记录。

最后是查找top5的article以及具体的文章内容



如图所示，在实际代码中，我只把第一个text的内容用二进制的方式打印出来了，（因为image和video太多了）

除此之外，我还在代码中实现了使用redis作为缓存



在上述打印过程中，会先去通过jedis调用redis，查找是否存在对应的键值的内容，如果不存在，则从hadoop读取并以二进制方式插入对应的内容。

# DBMS系统监控

我使用了innotop作为监控

结果如下

