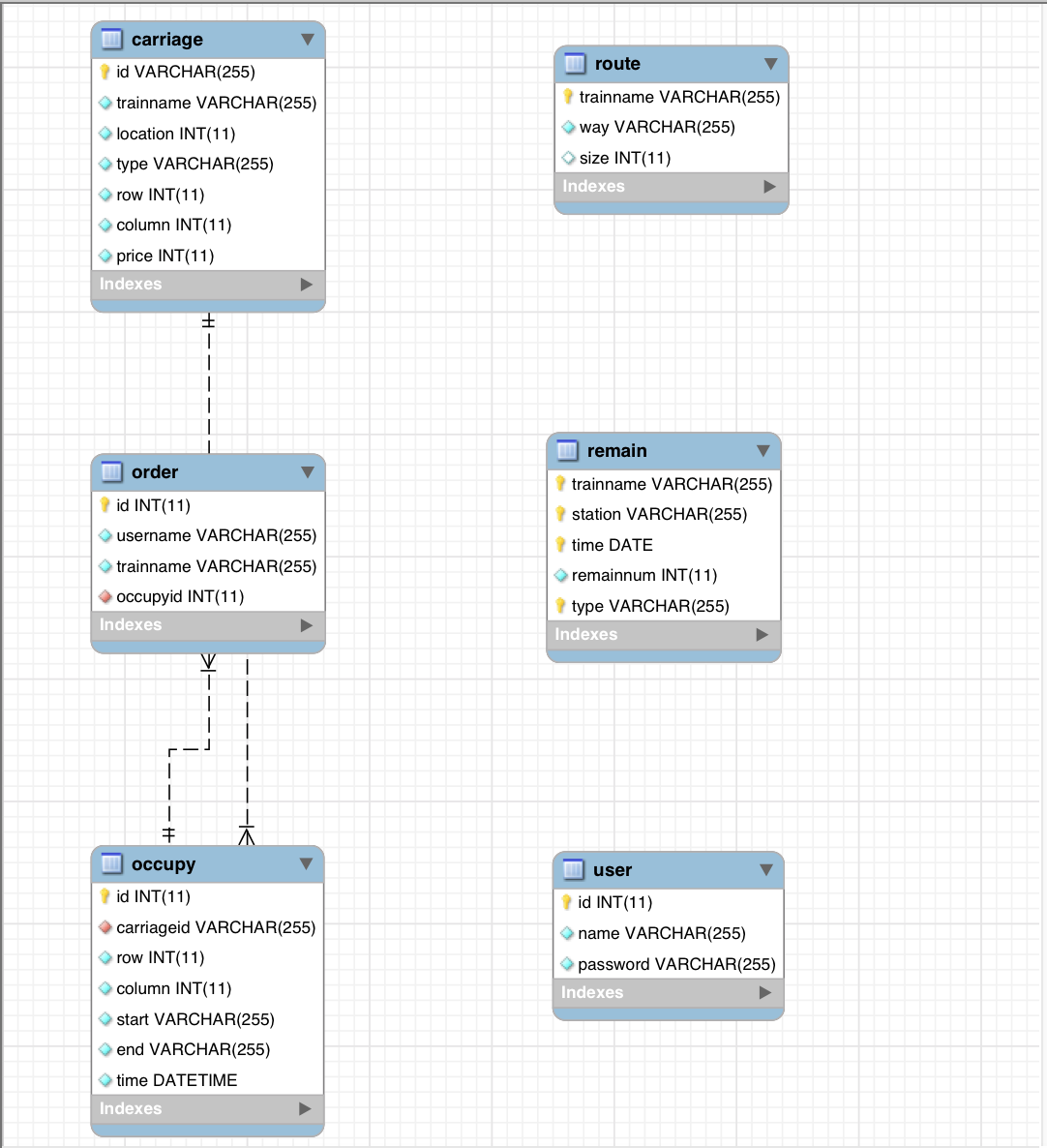
文件说明：

ticket.sql中是建表语句，homework.sql中是加好了初始测试数据的数据库导出文件，可直接导入（不然还要调用JDBC里丑陋的初始化代码）。java包中util包中JDBC为mysql部分的连接数据库、初始化方法，Mongodb中是Mongodb的。MysqlStart中是mysql部分的具体查看余票、购票的方法。test包中即对应的测试方法，每个类按名对应着两种数据库的测试方法。CRUDTest里是对高并发的测试

1. mysql数据库表的建立与说明
2. 模型：

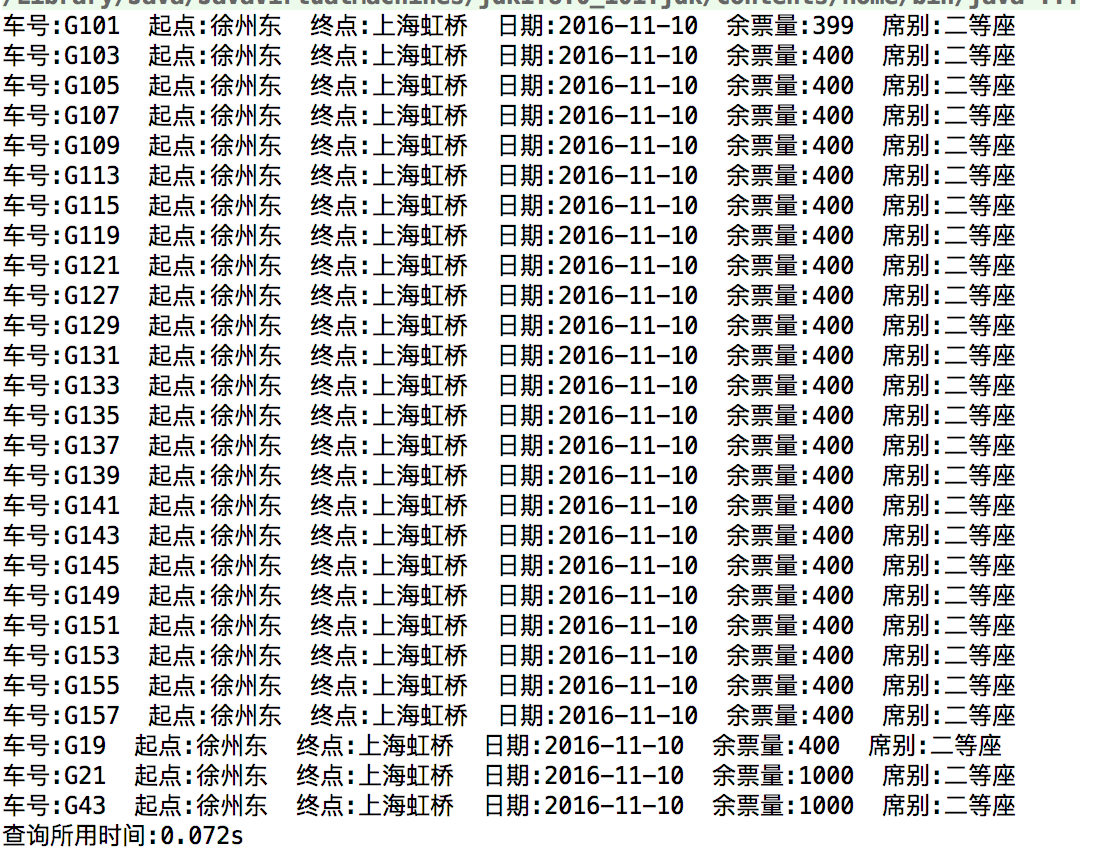


mongodb部分的表设计与mysql部分相同

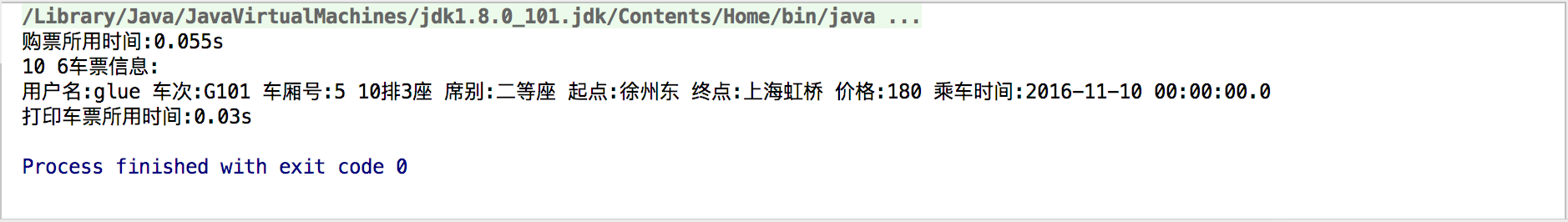
1. 说明：
2. carriage为车厢，字段分别为车厢名、所在车号、在火车的位置（即坐车时第几车厢），席别类型，行数，列数，基础票价（这里假设一班车一种坐席站台两两间价格固定，这个存储的就是两站之间的价格，总票价就用它\*两站点之间的站数）
3. occupy为已分配的座位信息，字段分别为id，车厢名，排号，座位号，起点站，终点站，乘车时间
4. order为订单信息，字段分别为id，用户名，火车号，座位号（外键，即occupy表中存的座位具体信息）
5. remain为剩余票量信息，字段为火车号，站台名（为啥这样设计以及这个表的设计思路在后文会解释），日期，剩余票数，席别类型，version（for乐观锁使用）
6. remaining为remain的第一版设计可以忽略不看了这里就不解释了2333
7. route为路线信息，即所给txt中的信息，字段为火车名，路线，火车车厢数
8. user为用户（这个表其实并没有被用到），字段为用户名和密码
9. 测试数据初始化部分
10. route表的初始化即导入txt，然后车厢数设计的是前100辆为8车，后面的为16车
11. carriage的初始化方法在JDBC.initCarriage()，代码写的很丑很羞愧所以希望助教直接导入homework.sql就好了2333。大致初始化的思路是分8车厢与16车厢，8车厢的列车2个商务舱（2\*5），2个一等座舱（4\*15），4个二等座舱（5\*20），无座4\*15，共600张车票；16车厢的列车2个商务舱（2\*5），4个一等座舱（4\*15），10个二等座舱（5\*20），无座10\*14，共1800张车票。然后价格大致是商务座90~100一站等等（代码中有体现）
12. remain的初始化方法在JDBC.initRemain()，对于剩余票的思路在下文解释，大概就是每班车的每个站点都要存每种的余票，然后各种循环（因为存了一周的余票情况，所以多了一层index为7的循环）
13. 设计思路
14. 暂不设置区间售票策略
15. 查询余票问题：对于每辆车的每个站，都存本站的余票，比如一辆车的路线为1-2-3-4,则4个站点都存着各类余票，若买了1-2-3的车票，则1，2的余票量都减一，余票问题靠remain表负责
16. 出票问题（座位分配问题）：这是我觉得最复杂的地方，感觉怎么存都不方便。。。我设计的是occupy这个表，存放的是已经分配的座位情况，有座位号和起始点。座位分配时是随机先找一个座位，然后去看是否与occupy中已有的冲突（具体代码可参见MysqlStarter的isConflict方法），如果occupy中当天没有这个座位号的存储情况，或者要乘坐的起点在存储的终点之后或者要下车的终点在存储的起点之前，则这个座位都是可以被分配的。我觉得自己的实现还很不好，只能考虑单张随机分配的问题，没有考虑多张票的连续分配。。。
17. 具体测试及实现方法说明

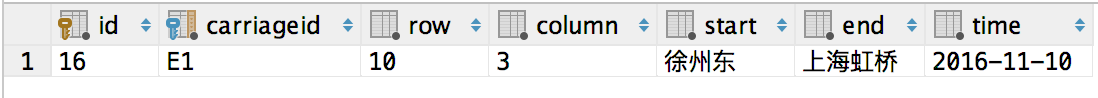
（测试用数据均在test包中）

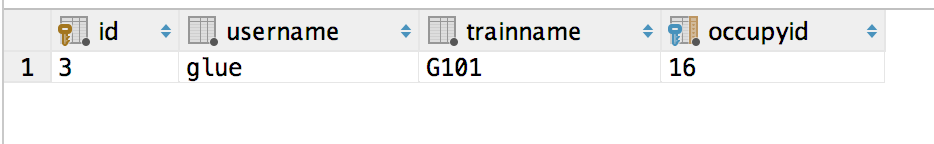
1. 查询余票：实现的方法是在MysqlStart的selectTicket方法中，主要思路是先从route表中找到符合的车号，然后对每个车号从remain表中查询余票情况，测试数据为s.selectTicket("徐州东","上海虹桥","二等座",d);测试结果见下图



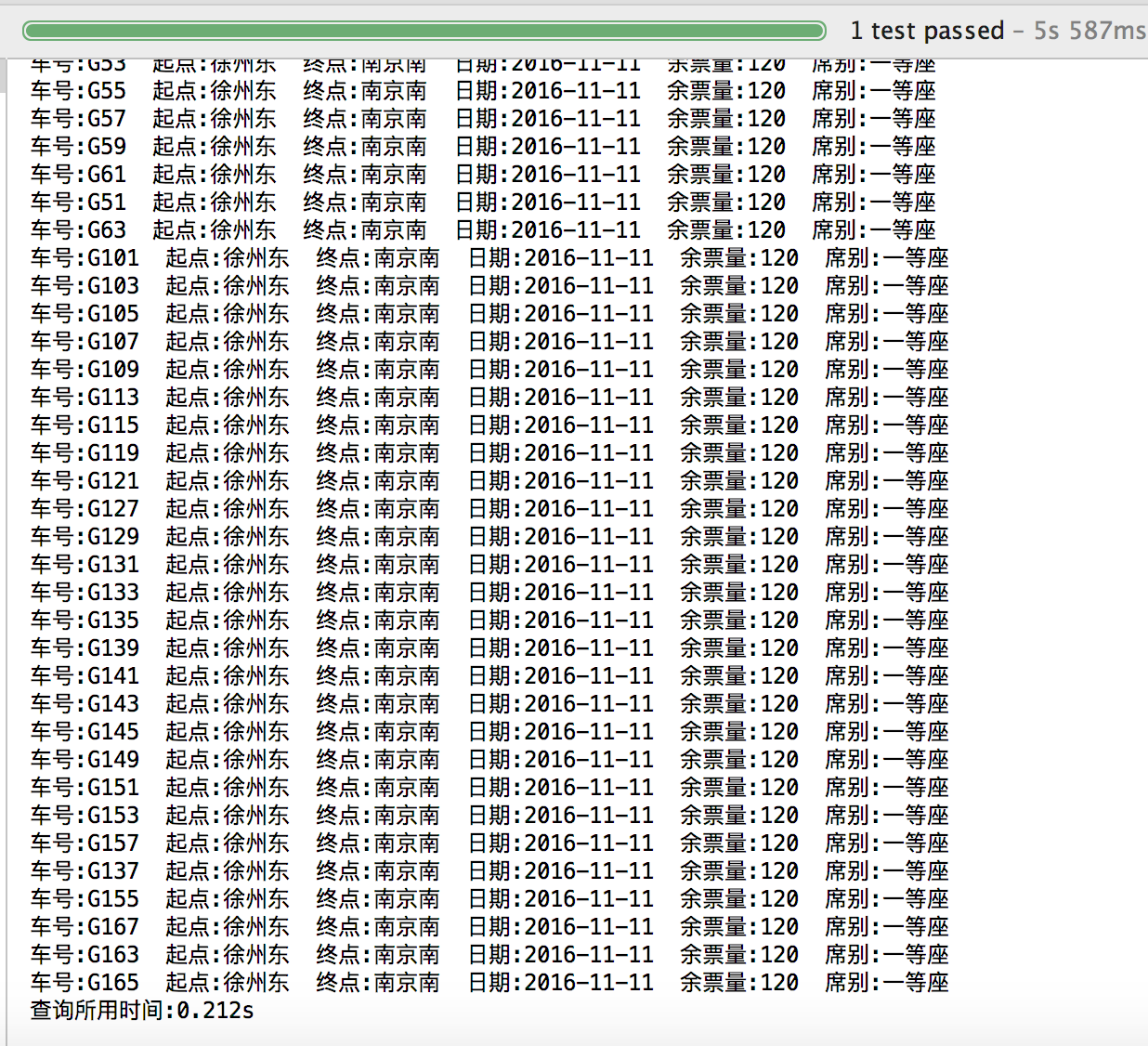
1. 购票：主要思路是先随机从要买的车次中生成一个座位然后与occupy表中已售出的票进行比对看是否冲突，如果冲突则重新生成。得到了合适的座位号后将其插入至occupy表中，然后将remain表中对应车次当天中途车站对应席别的车票都减一。然后调用打印车票的方法将车票信息打印出。测试方法为s.orderTicket("glue","G101","徐州东","上海虹桥","二等座",d);运行结果及occupy与remain表中的数据如下图展示。





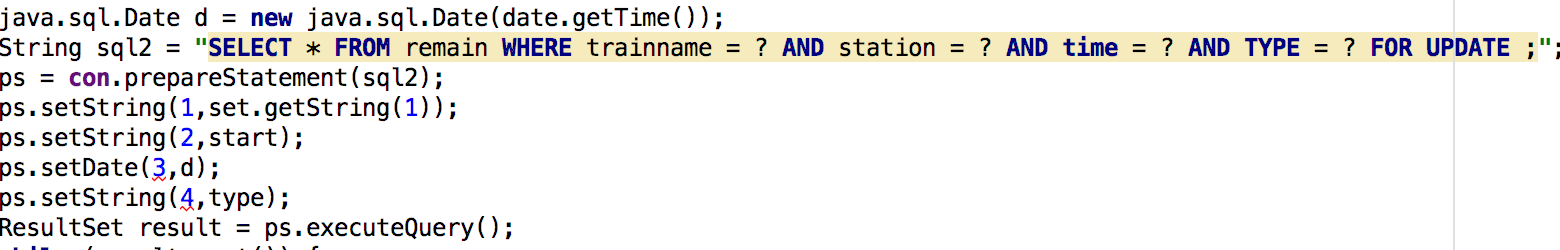


3.mongodb的方法思路与mysql部分相同，测试结果见下图

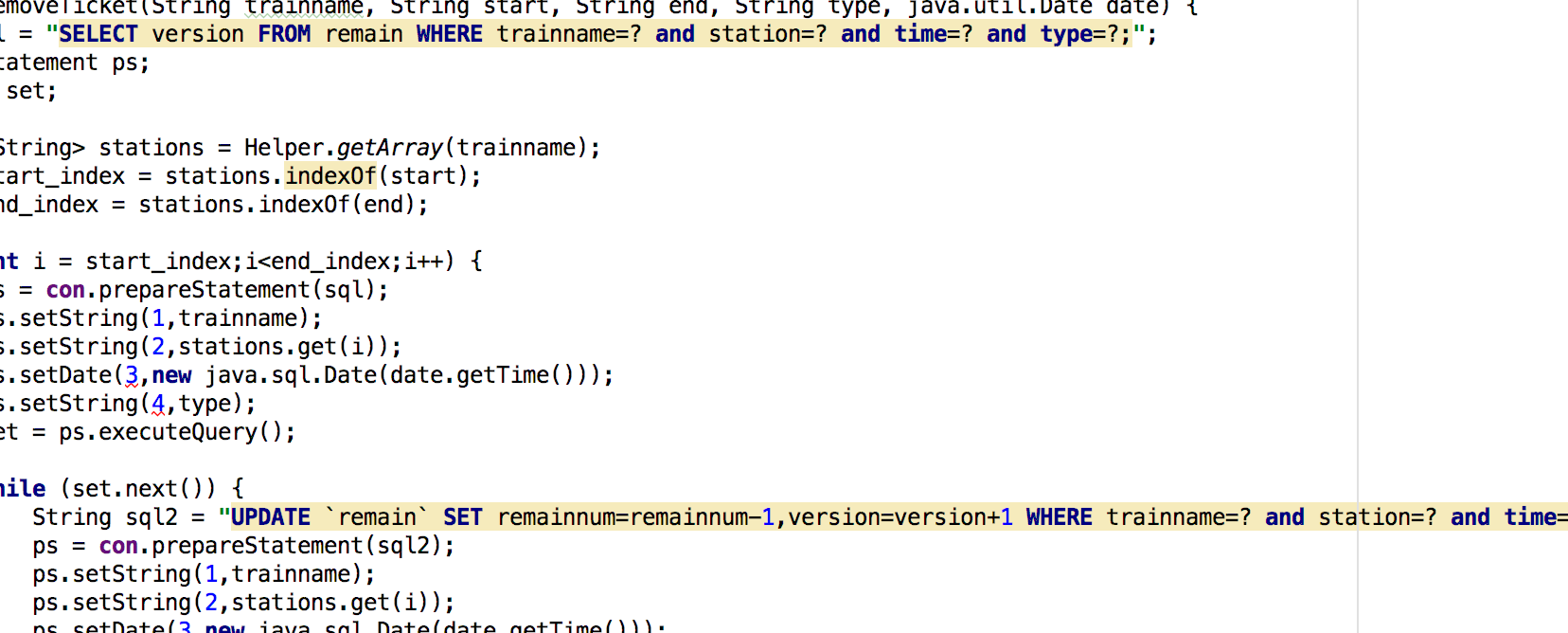




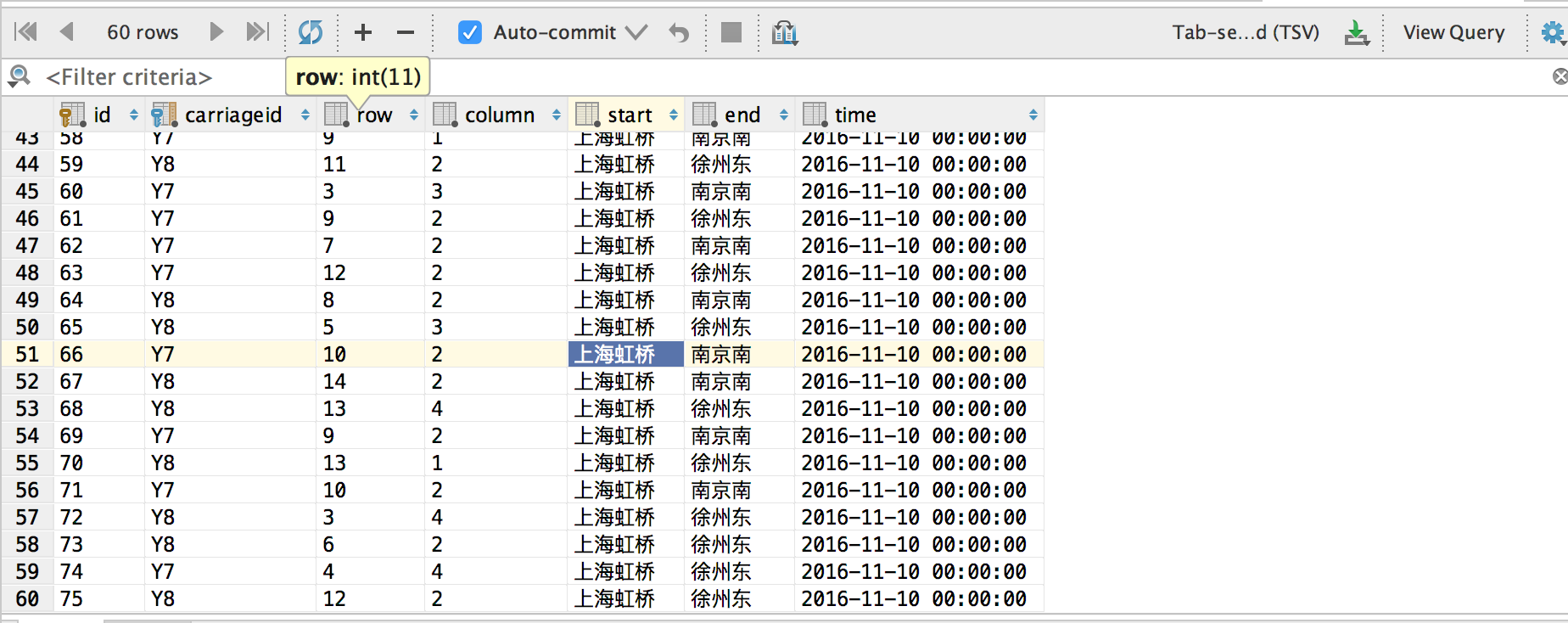
1. 优化
2. 优化思路一点是体现在occupy的表的设计中，最开始对于座位分配觉得太难实现了，开始想的是把两两站点每辆车的余票情况都存起来，后来觉得这样冗余太多也不好维护，后来想到可以换个思路不存所有余票然后买的时候从表中删除，而是反过来存已卖出去的票，这样剩余票数也好维护。
3. 优化方法二体现在高并发上，采用了悲观锁与乐观锁，均作用于remain这张表上。查询余票时采用悲观锁，方法见下图。



乐观锁用于更新余票数量上，加上了version这个字段，每次update时对version+1。（代码参见MysqlStart中removeticket的方法）



1. 高并发部分的测试是在CRUDTest中，测试结果如图（图中为分配的座位即occupy表的情况）



1. 对比mysql&mongodb

mongodb的概念是集合（collection），存储方式按照键值对key-value的形式存储。而mysql则是Resultset