Lab3说明文档（暂时只完成了Mysql部分）

141250034

范佳杰

1. Mysql(文件为12306.sql)

运行mysqlDBHelper里的main函数

1.1自定义售票策略

1.1.1用户可以购买某车次路线上的所有任意组合的区间票，然后根据此更新数据库，具体更新如下例说明：

某列车路线为A-B-C-D-E；

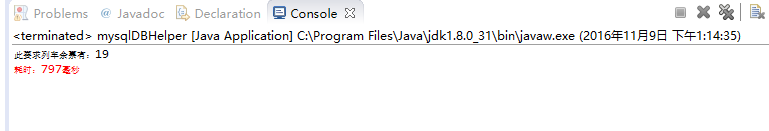
某人购买B到D的一等座车票，那么除了A-B，和D-E的此车次一等座余票不变化之外，这条路线上的其余区间票组合的一等座余票数减一。

1.1.2票价计算为每站10元再加上额外价格（商务座：120元，一等座：60元，二等座10元，无座：0元）公式为price=10\*station+extra\_price;

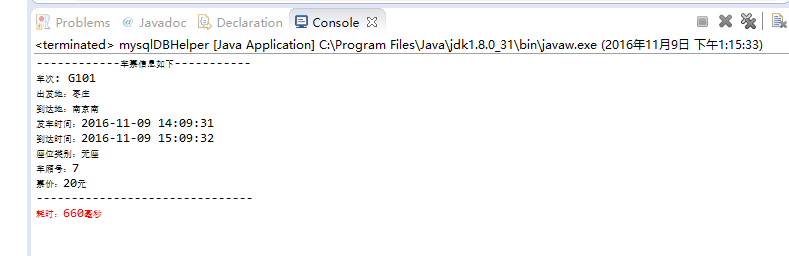
程序实现购票流程为：用户选择（车次，座位类别，出发地，到达地）请求购票，系统查询余票，提示无票或者安排车厢，座位，更新数据库，打印车票信息。

1.2操作运行时间截图

1.2.1查询余票



1.2.2购票并打印



1.3同时购买两、三张票的座位分配策略

假设同时买n张票，策略流程为：

1.系统先查询符合购票要求的车次的余票数total.

2.比较total和n的大小关系，若total<n,则提示用户无法满足此要求下的 n张同时购买的要求，建议更换要求（比如更换座位类型等）；若 total>=n,则为其分配车厢号和座位号（采取从后往前的顺序排位（除了无座）），以下是关键代码实现。

**private** **static** String[] *types* = {"商务座", "一等座", "二等座", "无座"};//座位类型

**private** **static** **int** *seatNum*=20;//每节车厢20个座位

**private** **static** **int**[] *carNum* = {2,2,4,1};//车厢数量组成(最后一个1表示有数量等于一个车厢的无座票，具体分配时是把这些无座的平均分配到二等座车厢中)

**int** car\_num=0;//车厢号

**int** seat\_num = 0;//座位号

**if**(type.equals("无座")){

car\_num=(total)%*carNum*[*2*]+1+*carNum*[0]+*carNum*[1];

}**else**{

**if**(total%*seatNum*==0){

car\_num=(total)/*seatNum*;

}**else**{

car\_num=(total/*seatNum*)+1;

}

**int** num=*getIndex*(type, *types*);

**for**(**int** i=0;i<num;i++){

car\_num+=*carNum*[i];

}

**if**(total%*seatNum*!=0){

seat\_num=total % *seatNum*;

}**else**{

seat\_num=total % *seatNum*+*seatNum*;

}

System.***out***.println("座位号：" + seat\_num);

}

System.***out***.println("车厢号："+ car\_num);

1.4如何保证插入数据的效率以及数据的一致性

使用事务处理，出错回滚，使用约束（比如建立外键约束）等

1.5避免一张票卖出两次

使用锁机制修改时加排他锁，直到事务提交后才释放，读取时加共享锁，这样一个事务读取数据的过程中，其他事务就不会修改该数据，只能读取，之后此事务如果有更新操作，那么会转换为排他锁，其他事务不能进来读写，避免一张票卖出两次。

1.6高并发时，怎样提高读写效率，并且尽量避免脏读幻读等

1.建立事务隔离级别 TRANSACTION\_SERIALIZABLE。

2.使用锁机制修改时加排他锁，直到事务提交后才释放，读取时加共享锁，这样一个事务读取数据的过程中，其他事务就不会修改该数据，只能读取，之后此事务如果有更新操作，那么会转换为排他锁，其他事务不能进来读写，防止了脏读问题。

3.采用范围锁RangeS RangeS\_S模式，锁定检索范围为只读，避免幻读。

4.减少锁持有的时间，提高读写效率。