141250149吴秦月

1. mysql部分

1.1数据库表

第一张表列车表，记录了车次，时间，乘客上车的车站，乘客下车的车站，这两站之间剩余的商务座票，这两站之间剩余的一等座票，这两站之间剩余的二等座票，这两站之间剩余的无座票，这两站之间经过的所有车站。车次，时间，乘客上车的车站，乘客下车的车站确定了唯一的一张车票。

```
String trainDrop = "drop table if exists train;";
String trainCreate = "create table train" + "( "
        + "carNumber varchar(255) not null,"
        + "time datetime not null,"
        + "start varchar(255) not null,"
        + "end varchar(255) not null,"
        + "businessRemain int(3) not null,"
        + "firstRemain int(3) not null,"
        + "secondRemain int(3) not null,"
        + "noSeatRemain int(3) not null,"
        + "route varchar(255) not null,"
        + "primary key(carNumber,time,start,end)"
        + " ) default charset=utf8;";
```

第二张表线路表，记录了车次，该车次经过的所有站，上网查了高铁同一车次线路一样，这里也认为一个车次线路一样，不考虑会临时改线路 。

```
String routeDrop = "drop table if exists route";
String routeCreate="create table route(" +
        "carNumber varchar(255) not null," +
        "route varchar(255) not null," +
        "primary key(carNumber)" +
        ")default charset=utf8";
```

第三张表座位表。车次，时间，乘客上车的车站，乘客下车的车站，车厢号，座位号，座位类型（商务座，一等座，二等座，无座）

```
String seatDrop = "drop table if exists seat;";
String seatCreate = "create table seat (" +
        "carNumber varchar(255) not null,"
        + "time datetime not null,"
        + "start varchar(255) not null,"
        + "end varchar(255) not null," +
        "carriageNo int(2) not null," +
        "seatNo int(2) not null," +
        "type enum('商务座','一等座','二等座','无座') not null, " +
        "primary key(carNumber,time,start,end,seatNo,carriageNo)" +
        ")default charset=utf8;";
```

1.2查询余票

车次，时间，乘客上车的车站，乘客下车的车站确定了唯一的一张车票。
所以先让用户输入这些，然后就可以通过搜索数据库得到相应的四种票的余票，当没有查找到是则输出没有符合类型的票

```
public static void getRemain() {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String time = "";
    String start = "";
    String end = "";
    String route = "";
    System.out.println("欢迎来到购票系统!");
    System.out.println("请输入购票日期");
    try {
        time = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("请输入您想在哪一站开始乘车");
    try {
        start = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("请输入您想在哪一站下车");
    try {
        end = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        PreparedStatement ps = conn.prepareStatement
                ("select * from train where start like ? and end like ? and time=?");

        ps.setString(1, "%" + start + "%");
        ps.setString(2, "%" + end + "%");
        DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
        java.sql.Date sDate = java.sql.Date.valueOf(time);
        ps.setDate(3, sDate);
        ResultSet rs = ps.executeQuery();
        conn.commit();
```

```
            conn.commit();
            int i = 0;
            while (rs.next()) {
                i++;
                System.out.println(rs.getString(1) + "  "
                        + rs.getDate(2) + " 出发站:" + rs.getString(3) +
                        " 下车站:" + rs.getString(4) +
                        " 商务座剩余:" +
                        rs.getString(5) + " 一等座剩余:" +
                        rs.getString(6) + " 二等座剩余:" +
                        rs.getString(7) + " 无座剩余:" +
                        rs.getString(8));

            }
            if (i == 0) {
                System.out.println("没有符合类型的车票");
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
```

1.3购票和打印车票

首先通过查询余票知道自己想要的票还有木有步骤如1.2查询余票
然后请求用户输入想要的座位类型和购票数量，我原本想的是购票需要通过查询数据库的表来分配座位的，但是这样太慢了，所以我把分配座位完全通过逻辑算法来实现，提高了效率。
假设车厢是8节，一节200个位置（其实多少都无所谓，只是这里为了叙述方便）
我把车厢的第一节分配给商务座（200张），二、三节分配给一等座（400张），四到八节分配给二等座（1000）张，无座票（200张）只在二等车厢。
首先 我用remain纪录用户想买的票还剩几张，num纪录用户想要购买几张票，当用户需要商务座时，车厢号肯定为1，座位号为：总座位数-（remain－1）%总座位数，由于每卖出一张票，相应的remain都会减一这个算法可以保证不会卖出重复的票，
再来看一等座，由于有两节车厢，所以我判断算出的座位号如果大于200，则座位号减200，车厢号加一，二等座和无座同上。
我认为有一个难点就是卖了车票之后更新余票了，举个例子有一辆北京－济南－南京－上海的车，卖了一张济南到上海的票，那么对应类型的票不仅济南到上海要减1，只要包含济南到南京或南京到上海的票都要减1，所以我的数据库表里还包含了经过的所有车站，卖了票后，就拿出来一一比较是否要减一，这样就可以保证不会卖超出数量的票，缺点是当数量很大的时候可能会非常的慢，但是我实在是想不出别的又好又快的方法了😭

1.m同时购买多张票座位分配

又因为用num纪录了顾客想要购买的票数，所以如果remain>num,时，循环这个过程num次就可以买多张票.

```
System.out.println("请输入您想乘坐的车次");
String carno = "";
String seattype = "";
int num = 0;
try {
    carno = br.readLine();
    System.out.println("请输入您想要的座位类型:商务座,一等座,二等座,无座");
    seattype = br.readLine();
    System.out.println("请输入购票数量");
    num = Integer.parseInt(br.readLine());
} catch (IOException e) {
    e.printStackTrace();
}
```

```
        while (num > 0) {
            i++;
            if (seattype.equals("商务座")) {//商务座为第一节车厢,200个位置
                carriageNo = 1;
                seatno = 200 - (remain - 1) % 200;
            } else if (seattype.equals("一等座")) {//一等座为第2-3个车厢,400个位置
                carriageNo = 2;
                seatno = 400 - (remain - 1) % 400;
                if (seatno > 200) {
                    carriageNo = 3;
                    seatno = seatno - 200;
                }
            } else if (seattype.equals("二等座")) {//二等座为4-8车箱,1000个位置
                carriageNo = 4;
                seatno = 1000 - (remain - 1) % 1000;
                if (seatno > 200 && seatno <= 400) {
                    carriageNo = 5;
                    seatno = seatno - 200;
                } else if (seatno > 400 && seatno <= 600) {
                    carriageNo = 6;
                    seatno = seatno - 400;
                } else if (seatno > 600 && seatno <= 800) {
                    carriageNo = 7;
                    seatno = seatno - 600;
                } else if (seatno > 800 && seatno <= 1000) {
                    carriageNo = 8;
                    seatno = seatno - 800;
                }
            } else {//无座
                carriageNo = 5;
            }
```

```
            }
            if (remain < num) {
                System.out.println("没有符合类型的车票");
                return;
            }
```

```java
public static boolean isMinus(String route, String temproute) {
    //只要route中有一段在temproute中,那么temproute的票就要减一
    String[] splitroute = route.split("-");
    int i = 0;
    for (int j = 0; j < splitroute.length; j++) {
        if (temproute.contains(splitroute[j])) {
            i++;
        }
    }
    if (i >= 2) {
        return true;
    }
    return false;
}
```

1.4结果展示暨操作时间

```
开始建表,插入测试数据...
初始化数据库成功
导入数据成功
结束建表,插入测试数据! 时间 : 111 ms
欢迎来到购票系统!
请输入购票日期
2016-12-2
请输入您想在哪一站开始乘车
济南
请输入您想在哪一站下车
上海
开始查询余票...
G13  2016-12-02 出发站:济南西 下车站:上海虹桥 商务座剩余:200 一等座剩余:400 二等座剩余:1000 无座剩余:200
G7  2016-12-02 出发站:济南西 下车站:上海虹桥 商务座剩余:200 一等座剩余:400 二等座剩余:1000 无座剩余:200
结束查询余票!时间:39ms
请输入您想乘坐的车次
G7
请输入您想要的座位类型:商务座,一等座,二等座,无座
一等座
请输入购票数量
2
开始为您购票安排座位...
正在为您打印车票!
G7  2016-12-02 出发站:济南西 下车站:上海虹桥 车厢号:2 座位号:1 座位类型:一等座
正在为您打印车票!
G7  2016-12-02 出发站:济南西 下车站:上海虹桥 车厢号:2 座位号:2 座位类型:一等座
结束购票!时间:3ms

Process finished with exit code 0
```

整个查询余票和购票和打印车票的流程如图。

关于记录操作时间:

```java
long start = 0;
long end = 0;
System.out.println("开始建表,插入测试数据...");
start = System.currentTimeMillis();
```

我在查询数据库前纪录当前时间

```java
end = System.currentTimeMillis();
System.out.println("结束建表,插入测试数据! 时间 : "+(end  - start)+" ms");
```

在查完后又纪录当前时间，两个时间一减就是操作时间。

由图可见建表插数据用时111ms，查询余票用时39ms，购票和打印车票用时3ms。由此可见用算法而不需要用到数据库的座位安排策略非常的迅速。

1.5多线程卖票

关于多个人同时买票，我觉得查阅余票的时候不需要锁起来，因为即使一个人在查阅余票的时候另一个人买了票票数有变化，对查阅余票的人并没有什么很大的影响，最多就是他发现有票但是点进去买票的时候木有了，12306也是这个设计。所以我觉得只需要在买票的时候锁起来不让别人买就行，在multithread这个类里，我就模拟里两个线程同时买最后一张票。

为了模拟这个情况，我在数据库中插入2016-12-2 G3 南京南到上海虹桥的商务座，但是只有一张票

首先创建两个线程

```
multithread a = new multithread();
a.setName("顾客A");
multithread b = new multithread();
b.setName("顾客B");
a.start();
b.start();
```

屏幕快照 2016-11-03 上午9.42.3

然后将够票的方法锁起来，一次只能有一个线程买票

```
public static synchronized void getRemain(String name) {
```

两位顾客都想买2016-12-2 G3 南京南到上海虹桥的商务座，但是只有一张票，所以只有顾客A买到了，顾客B没没买到

```
G13 2016-12-2   南京南   上海虹桥   商务座剩余: 1
顾客A正在为您打印车票！
G13  2016-12-02 出发站:南京南 下车站:上海虹桥 车厢号:1 座位号:200 座位类型:商务座
顾客B没有符合类型的车票
```

1.n 测试数据

助教检查的时候可以把这三行前面的注释符去了，然后把这两行里的目录换成自己电脑里的目录，routes.txt,和data.txt我放在了上交的作业的src并行的目录

下，别忘了把数据库的用户名和密码改成自己的，最重要的是先在自己的电脑里创建一个叫cms的数据库。。。。

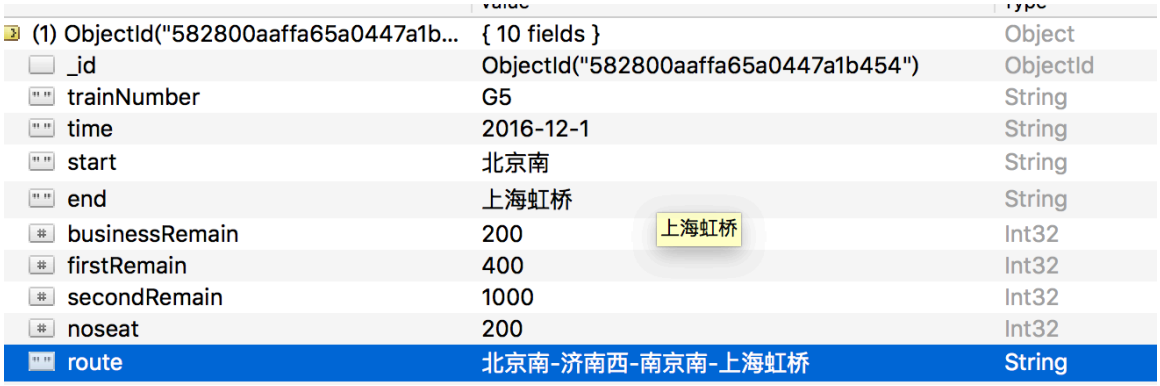
```
//          InitDB initDb = new InitDB(conn);
//          initDb.createTable();
//          initDb.insertData();
```

```
String insertRoute ="load data local infile '/Users/Alisa/Desktop/routes.txt'" +
        "into table route(carNumber,route);";
String inserttrain ="load data local infile '/Users/Alisa/Desktop/data.txt'" +
        "into table train(carNumber,time,start,end,businessRemain,firstRemain,secondRemain,noSeatRemain,route);";
Typo: In word 'inserttrain' more... (⌘F1)
```

2 mongodb 部分

2.1collection

列车集合，记录了车次，时间，乘客上车的车站，乘客下车的车站，这两站之间剩余的商务座票，这两站之间剩余的一等座票，这两站之间剩余的二等座票，这两站之间剩余的无座票，这两站之间经过的所有车站。



2.2 查询余票

车次，时间，乘客上车的车站，乘客下车的车站确定了唯一的一张车票。所以先让用户输入这些，然后就可以通过搜索数据库得到相应的四种票的余票，当没有查找到是则输出没有符合类型的票

```java
public static void getRemain(MongoCollection<Document> collection) {
    long start1 = 0;
    long end1 = 0;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String time = "";
    String start = "";
    String end = "";
    String route = "";
    System.out.println("欢迎来到购票系统!");
    System.out.println("请输入购票日期");
    try {
        time = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("请输入您想在哪一站开始乘车");
    try {
        start = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("请输入您想在哪一站下车");
    try {
        end = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("开始查询余票...");
    start1 = System.currentTimeMillis();
    Document doc = new Document();
    Pattern startP = Pattern.compile(".*" + start + ".*", CASE_INSENSITIVE);
    Pattern endP = Pattern.compile(".*" + end + ".*", CASE_INSENSITIVE);
    doc.put("start", startP);
    doc.put("end", endP);
    doc.put("time", time);
    doc.put("start", startP);
    doc.put("end", endP);
    doc.put("time", time);
    int i = 0;
    FindIterable<Document> findIterable = collection.find(doc);
    MongoCursor<Document> mongoCursor = findIterable.iterator();
    while (mongoCursor.hasNext()) {
        i++;
        Document temp = mongoCursor.next();
        System.out.println(temp.get("trainNumber") + " " + temp.get("time") + " 出发站:" + temp.get("start") +
                " 下车站:" + temp.get("end") +
                " 商务座剩余:" +
                temp.get("businessRemain") + " 一等座剩余:" +
                temp.get("firstRemain") + " 二等座剩余:" +
                temp.get("secondRemain") + " 无座剩余:" +
                temp.get("noseat"));
    }
    if (i == 0) {
        System.out.println("没有符合类型的车票");
    }


    end1 = System.currentTimeMillis();
    System.out.println("结束查询余票!时间:" + (end1 - start1) + "ms");
```

2.3购票和打印车票

思路同1.3，都是通过java实现，为了方便助教查看再写一遍。

首先通过查询余票知道自己想要的票还有木有步骤如1.2查询余票

然后请求用户输入想要的座位类型和购票数量，我原本想的是购票需要通过查询数据库的表来分配座位的，但是这样太慢了，所以我把分配座位完全通过逻辑算法来实现，提高了效率。
假设车厢是8节，一节200个位置（其实多少都无所谓，只是这里为了叙述方便）
我把车厢的第一节分配给商务座（200张），二、三节分配给一等座（400张），四到八节分配给二等座（1000）张，无座票（200张）只在二等车厢。
首先 我用remain纪录用户想买的票还剩几张，num纪录用户想要购买几张票，当用户需要商务座时，车厢号肯定为1，座位号为：总座位数-（remain－1）%总座位数，由于每卖出一张票，相应的remain都会减一这个算法可以保证不会卖出重复的票，
再来看一等座，由于有两节车厢，所以我判断算出的座位号如果大于200，则座位号减200，车厢号加一，二等座和无座同上。
我认为有一个难点就是卖了车票之后更新余票了，举个例子有一辆北京－济南－南京－上海的车，卖了一张济南到上海的票，那么对应类型的票不仅济南到上海要减1，只要包含济南到南京或南京到上海的票都要减1，所以我的数据库表里还包含了经过的所有车站，卖了票后，就拿出来一一比较是否要减一，这样就可以保证不会卖超出数量的票，缺点是当数量很大的时候可能会非常的慢，但是我实在是想不出别的又好又快的方法了😭

2.m同时购买多张票座位分配

又因为用num纪录了顾客想要购买的票数，所以如果remain>num,时，循环这个过程num次就可以买多张票.

```
    System.out.println("请输入您想乘坐的车次");
    String carno = "";
    String seattype = "";
    int num = 0;
    try {
        carno = br.readLine();
        System.out.println("请输入您想要的座位类型:商务座,一等座,二等座,无座");
        seattype = br.readLine();
        System.out.println("请输入购票数量");
        num = Integer.parseInt(br.readLine());
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("开始为您购票安排座位...");
    start1 = System.currentTimeMillis();
    Document doc1 = new Document();
    doc1.put("start", startP);
    doc1.put("end", endP);
    doc1.put("time", time);
    doc1.put("trainNumber", carno);
    FindIterable<Document> findIterable1 = collection.find(doc1);
    MongoCursor<Document> mongoCursor1 = findIterable1.iterator();
    int remain = 0;
    int carriageNo = 0;
    int seatno = 0;
    i = 0;
    if (mongoCursor1.hasNext()) {
        Document temp = mongoCursor1.next();
        if (seattype.equals("商务座")) {//商务座为第一节车厢,200个位置
            remain = temp.getInteger("businessRemain");
        } else if (seattype.equals("一等座")) {//一等座为第2-3个车厢,400个位置
            remain = temp.getInteger("firstRemain");
        } else if (seattype.equals("二等座")) {//二等座为4-8车箱,1000个位置
            remain = temp.getInteger("secondRemain");
```

```
        } else if (seattype.equals("二等座")) {//二等座为4-8车箱,1000个位置
            remain = temp.getInteger("secondRemain");
        } else {//无座
            remain = temp.getInteger("noseat");
        }
        route = temp.getString("route");
        while (num > 0) {
            i++;
            if (seattype.equals("商务座")) {//商务座为第一节车厢,200个位置
                carriageNo = 1;
                seatno = 200 - (remain - 1) % 200;
            } else if (seattype.equals("一等座")) {//一等座为第2-3个车厢,400个位置
                carriageNo = 2;
                seatno = 400 - (remain - 1) % 400;
                if (seatno > 200) {
                    carriageNo = 3;
                    seatno = seatno - 200;
                }
            } else if (seattype.equals("二等座")) {//二等座为4-8车箱,1000个位置
                carriageNo = 4;
                seatno = 1000 - (remain - 1) % 1000;
                if (seatno > 200 && seatno <= 400) {
                    carriageNo = 5;
                    seatno = seatno - 200;
                } else if (seatno > 400 && seatno <= 600) {
                    carriageNo = 6;
                    seatno = seatno - 400;
                } else if (seatno > 600 && seatno <= 800) {
                    carriageNo = 7;
                    seatno = seatno - 600;
                } else if (seatno > 800 && seatno <= 1000) {
                    carriageNo = 8;
                    seatno = seatno - 800;
                }
            } else {//无座
```

```
                    seatno = seatno - 800;
                }
            } else {//无座
                carriageNo = 5;
            }
            if (remain < num) {
                System.out.println("没有符合类型的车票");
                return;
            }
            if (seatno != 0) {
                System.out.println("正在为您打印车票!");
                System.out.println(temp.get("trainNumber") + "  "
                        + temp.get("time") + " 出发站:" + temp.get("start") +
                        " 下车站:" + temp.get("end") +
                        " 车厢号:" +
                        carriageNo + " 座位号:" +
                        seatno + " 座位类型:" +
                        seattype);

            } else {
                System.out.println(temp.get("trainNumber") + "  "
                        + temp.get("time") + " 出发站:" + temp.get("start") +
                        " 下车站:" + temp.get("end") +
                        " 车厢号:" +
                        carriageNo + " 座位类型:" +
                        seattype);
            }
            num--;
            remain--;

        }
```

2.4结果展示暨操作时间

```
Connect to database successfully
欢迎来到购票系统!
请输入购票日期
2016-12-2
请输入您想在哪一站开始乘车
北京
请输入您想在哪一站下车
上海
开始查询余票...
G7 2016-12-2 出发站:北京南 下车站:上海虹桥 商务座剩余:200 一等座剩余:394 二等座剩余:1000 无座剩余:200
G13 2016-12-2 出发站:北京南 下车站:上海虹桥 商务座剩余:0 一等座剩余:400 二等座剩余:1000 无座剩余:200
结束查询余票!时间:46ms
请输入您想乘坐的车次
G7
请输入您想要的座位类型:商务座,一等座,二等座,无座
一等座
请输入购票数量
2
开始为您购票安排座位...
正在为您打印车票!
G7  2016-12-2 出发站:北京南 下车站:上海虹桥 车厢号:2 座位号:7 座位类型:一等座
正在为您打印车票!
G7  2016-12-2 出发站:北京南 下车站:上海虹桥 车厢号:2 座位号:8 座位类型:一等座
结束购票!时间:2ms
```

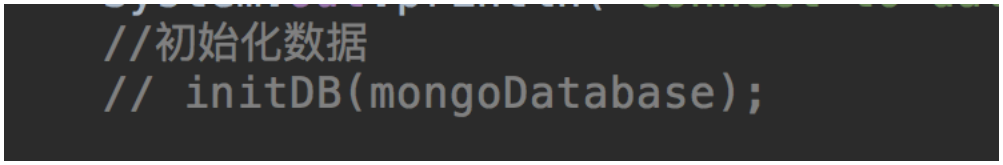
查询余票 46ms，打印车票 2ms。

2.5多线程卖票

思路和实现都同1.5

2.6测试数据

我在 java 程序中写了一个方法自动初始化 mongodb 和插入相应数据，检查时只要将注释去掉即可。



三、比较 mysql 和 mongodb

因为分配座位和打印车票是通过 java 实现的没有调用数据库所以比较这两个的时间没有意义，
一开始比较查询余票的时间都发现 mysql 比 mongodb 快，但当我增大数据时，发现 mongodb 比 mysql 快，数据越多 mongodb 越比 mysql 快。

	Mongodb	Mysql
数据库模型	非关系型	关系型
存储方式	虚拟内存 + 持久化	不同的引擎有不同的存储方式
查询语句	独特的 mongodb 查询方式	传统 sql 语句
架构特点	可以通过副本集，以及分片来实现高可用	常见有单点，m-s,mha,mmm,cluster 等架构方式
数据处理方式	基于内存，将热数据存在物理内存中，从而达到高速读写	不同的引擎拥有其自己的特点

成熟度	新兴数据库，成熟度较低	拥有较为成熟的体系，成熟度较高
广泛度	Nosql 数据库中，mongodb 是较为完善的 db 之一 使用人群也在不断增长	开源数据库的份额在不断增加，mysql 的份额也在持续增长

Mongodb 的优势：快速，高扩展性，自身的 failover 机制，json 的存储格式。
Mongodb 的劣势：应用经验少，非关系型的数据库模型可能会造成部分新使用者的不适应，锁机制导致队列堆积，无事务机制。

四、回顾设计方案,简单谈一谈感受以及优化思路

感受

首先，我从来没有吐槽过 12306，我一直觉得能写出 12306 的人十分地了不起。
因为我自己在 12306 买票的时候，每次都十分迅速而且也没有出过错误，所以可能不能理解为什么有人会吐槽 12306.
当我写完这份作业后，更加倾佩 12306 的程序员了，因为几乎每个老师上课的时候都说过，空间和时间是不能并存的，要想省空间就得花时间，要想省时间就得花空间，一开始我写这份作业的时候想得是选座位用数据库里的数据库实现，但是那样实在是太慢了，于是我就设计了算法用 java 来实现，但是这样就得在数据库里存很多额外的东西，而当我卖完票去更新数据库的时候，又有好多相关连的东东需要更新，总之就是十分钦佩 12306 的程序员能够写出又好又快的程序，帮助我们回家，好人有好报呀。

优化思路

1.我把分配座位完全通过逻辑算法来实现，提高了效率。
假设车厢是8节，一节200个位置（其实多少都无所谓，只是这里为了叙述方便）
我把车厢的第一节分配给商务座（200张），二、三节分配给一等座（400张），四到八节分配给二等座（1000）张，无座票（200张）只在二等车厢。
首先 我用remain纪录用户想买的票还剩几张，num纪录用户想要购买几张票，当用户需要商务座时，车厢号肯定为1，座位号为：总座位数-（remain－1）% 总座位数，由于每卖出一张票，相应的remain都会减一这个算法可以保证不会卖出重复的票，
再来看一等座，由于有两节车厢，所以我判断算出的座位号如果大于200，则座位号减200，车厢号加一，二等座和无座同上。

我认为有一个难点就是卖了车票之后更新余票了，举个例子有一辆北京－济南－南京－上海的车，卖了一张济南到上海的票，那么对应类型的票不仅济南到上海要减1，只要包含济南到南京或南京到上海的票都要减 1，所以我的数据库表里还包含了经过的所有车站，卖了票后，就拿出来一一比较是否要减一，这样就可以保证不会卖超出数量的票

2.由于主键默认有索引，而我每次用到的语句正好就是那几个联合主键，所以不需其它索引。

ps.助教是我见过最认真的助教，从来没有见过一个助教因为学生不认真写作业而生气，但是大部分同学都是认真写的呀，我觉得作业可以分成三类，一类是自己不想学随便写了交的，这种人自己不想好，你也不用为了他生气啊，一类是随手就能写好的大神，助教肯定会喜欢的，还有一类就是芸芸众生，比如说我，虽然我每天都在写数据库（真的！）但是助教你要知道，人与人的智商真的有差距，可能我每天都认真的学每天都认真的写就是不如大神一会会写的好，如果我这样写出来的作业被认为是不认真，我真的会想哭的。。。