# 背景

火车票智能推荐，包括上车补票、接续换乘、机火换乘。做智能推荐需要依赖火车线路数据作支持，火车线路数据不能直接获取，需要曲线救国，通过车次数据查询车次对应的途径站，依据途径站拼装线路数据。

# 基础数据

## 城市数据

城市数据的作用在于，在机火引流、机火换乘的时候，能够通过火车站精确的找到该火车站所属的城市，然后查询该城市下能够到达目的地航线。由此，机票业务线与火车票业务线城市数据一定要统一（可延后）。

## 车站数据

### 表结构

已有表 train\_station，其结构如下：

|  |  |
| --- | --- |
| id | 主键 |
| station | 站名 |
| code | 三字码 |
| city | 车站所属城市名 |
| quanpin | 全拼，eg：北京=beijing |
| jianpin | 简拼，eg：北京=bj |
| created | 创建时间 |
| modified | 到达时间 |
| ~~address~~ | ~~车站地址~~ |
| lbs | 用于地图定位 |
| ~~hot~~ | ~~热度~~ |

hot应用场景，前台搜索车站时，展示部分常用的车站。

address暂无应用场景。

### 车站获取方式：

* 途牛接口：3.12车站三字码接口
* 12306接口：<https://kyfw.12306.cn/otn/resources/js/framework/station_name.js>
* ~~携程查询接口：（目前手里只有查询余票接口）~~

车站变更频率低，建议一天更新一次

## 途径站（时刻表）数据

用于存储一个车次从始发站到终点站所经过的所有车站，包括到站时间、出站时间、停靠实现、站序等。

### 表结构

设计结构如下

|  |  |
| --- | --- |
| train\_no | 车次 |
| station\_code | 车站三字码 |
| station\_name | 车站名 |
| station\_no | 站序 |
| departure\_time | 出发时间 HH:mm |
| arrival\_time | 到站时间HH:mm |
| stop\_time | 停靠时长，字符串，n分钟 |

### 数据获取方式

* 供货商（途牛）接口：3.7途径站查询接口
* 12306接口：<https://kyfw.12306.cn/otn/queryTrainInfo/query?leftTicketDTO.train_no=240000G1010M&leftTicketDTO.train_date=2019-05-21&rand_code=>
* ~~携程查询接口：（目前手里只有查询余票接口）~~

途径站变更频率低，建议一周更新一次

## 车次数据（线路数据）

该数据落库采用穷举法（n\*(n-1)/2），n代表一个车次的所有站点。

假设有一趟列车：G101，其途径站为 北京（始发站）-上海-南京-广州（终点站）。

则数据落库为：

~~2019-10-01-G101-北京-上海~~

G101-北京-上海

G101-北京-（上海） -南京

G101-北京-（上海-南京） -广州

G101-上海-南京

G101-上海-（南京） -广州

G101-南京-广州

依据数据计算得出，每日车次在1w左右。按照平均每个车次30个站点。

A:不存预售期日历数据，允许出现误差：总数据量为：435w。

~~B:存预售期日历数据，但不能保证完全不会变动：总数据量为：1.3亿（30天预售期）和3.9亿（60天预售期）~~。

### 表结构

其数据结构如下：

|  |  |
| --- | --- |
| train\_no | 车次 |
| from\_station\_code | 出发站三字码 |
| from\_station\_name | 出发站名 |
| to\_station\_code | 到达站三字码 |
| to\_station\_code | 到达站名 |
| from\_time | 出站时间 HH：mm |
| to\_time | 到站时间 HH：mm |
| run\_time | 耗时，以分钟为单位 int |
| mileage | 里程，int |
| yn | 0:禁用，1:启用 |

### 数据获取方式：

* 根据车次数据和途径站，进行线路数据组装。
* 不以预售期日历为维度存储，则不需要全量更新，只需要增量更新、启/禁用即可。

新增车次线路或者暂停车次线路可以通过12306公告查看：<https://www.12306.cn/mormhweb/zxdt/index_zxdt.html>

后期可以针对这个公告做一个监控机制，当12306发布新的公告时，将公告内容通过邮件推送道相关人员处理。

# 上车补票

上车补票，调研了两个竞品，详细说明如下。

针对智行和去哪两个app进行比对发现

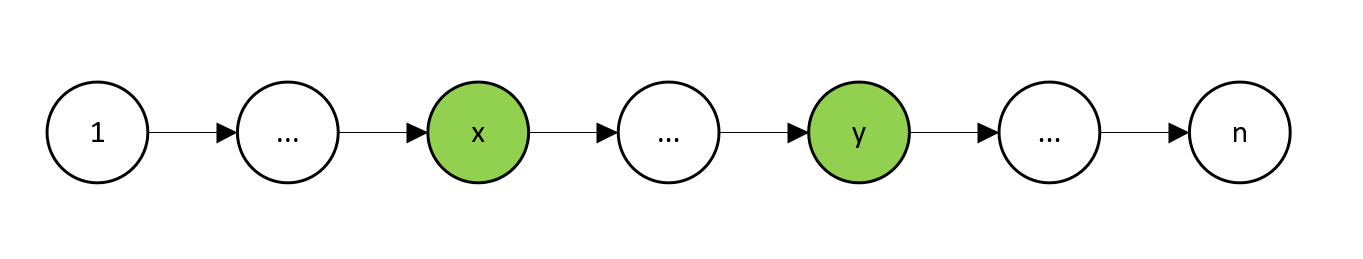
* 智行根据站站推荐上车补票方案，去哪根据车次推荐上车补票方案。



eg：现有车次：G101-北京(1)-石家庄(2)-上海(3)-南京(4)-广州(5) –海口(6)，乘客订票上海-广州。现有推荐方案如下：

* 头多尾多，晚上早下：北京-海口。头尾多买（北京-上海，广州-海口）
* 头多尾平，晚上：北京-广州。多买2站，北京-上海
* 头多尾少，晚上：北京-南京。多买2站，北京-上海；补1站，南京-广州
* ~~头多尾无，北京-石家庄 or 北京-上海，乘客在上海上车，所以实际情况汇总补不了车票~~
* 头平尾多，早下：上海-海口。多买一站，广州-海口
* ~~头平尾平，上车补票推荐，本事在无票的情况下进行，所以不存在这种情况~~
* 头平尾少，补票：上海-南京，补一站票，南京-广州（可以补多站）
* ~~头少尾平、头少尾多：头少无法上车~~。

以一个车次30站，站站查询共有10个车次结果，如下图：n=30



计算公式：nx-xx-1，其中：

n为该次列车总共站台数

x为乘客选取的出发站在途径站中的站序。

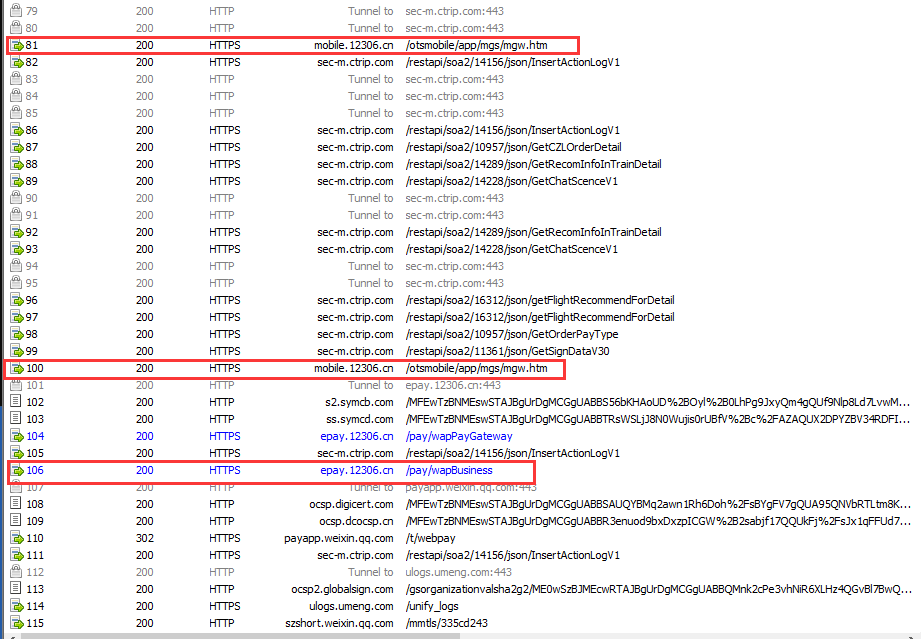
以每个车次30个车站为准，最少路线方案55次，最多路线方案224次。在此基础上\*10个车次，分别为550和2240。可见，在站站查询中推荐上车补票比在单车次页面推荐上车补票的查询量要大。

为何计算出来路线之后，还要在此查询余票呢？

因为，余票在12306是实时的，给用户推荐方案的前提是有票，需要根据计算得出的路线，在此查询余票，过滤掉无余票的路线，将有余票的路线呈现给用户。

经过对两个app多个日期的查询结果和12306测试结果，发现现象如下：

1. 平常时，出现上车补票推荐几率低于高峰时，同样的高峰时无座率也高于平常时，是否和无票相关？
2. 平常时，无票车次可能出现，也可能不出现上车补票推荐。
3. 并不是全部的无票车次都会有上车补票推荐。
4. 智行和去哪在同一个车次的推荐方案上9成相似（都被携程收购了(⊙o⊙)）。
5. 智行和去哪为同步推荐，途牛为异步推荐。
6. 同时发现智行在支付时直接唤起的是铁路12306清算中心的支付方，而其他的app均是本公司的支付方，智行是不是单单对12306的app进行了包裹？如下图：



1. 一个有趣的现象，12306显示余票10张（冷门车次），我分两次购票8张，理应还剩余票2张，但是在次之后的2分钟内的查询余票结果均为10张，当再次购票时告知余票不足，随后再次查询余票，显示2张。缓存时间稍长，购票时余票不足可以认为是一个缓存纠错的行为。

思考：

1. 对于用户行为的查询（站站查询，单车次查询）依旧按照正常流程，先查询缓存，缓存没有查询供应商，而后更新缓存。
2. 余票数据缓存时长加长，推荐方案以缓存数据为准，允许出现误差，发现误差后更新缓存？
3. 设置一些热门车次，推荐方案只推荐热门车次？
4. 补票n站，多买n站，对这个n进行最大值的限制？
5. 考虑去哪，进行单个车次推荐？或者途牛一样异步推荐（用户点击上车补票才进行推荐）？

另一个小发现，对于抢票，是不是可以推进’ 跨站抢票’,类似上车补票，只是一抢票形式展示。

另：架构师提出，方案需要缓存命中率支持，目前系统并没有对余票缓存命中率这一块进行监控，而Redis info命令显示的命中率是这个库的命中率，不适用。通过查询日志进行佐证，粗略的乐观的估计命中率低于10%

# 接续换乘（中转）

接续换乘12306PC存在这样的接口，app暂不支持。使用接续换乘查询的前提是登录账号。如下图



那么供应商就一定可以爬取这样的接口数据。我们可以推动供应商提供这样的功能接口，如果供应商无法实现，则需要自己通过算法实现多条线路的换乘方案，可以参考《多目标铁路旅客乘车方案优化模型及算法研究》，铁总研究的一片文章。

或者可以结合上车补票的方案，进行计算路线，之后批量查询。

# 机火换乘

产品说的先不做，需要考虑的一点是：要给乘客预留出来火车站到机场的时间。

# 保险数据

统计从18年12月至今，购买保险的票数、购保成功的金额。

保险状态为：出保成功

划账状态为：正向已结算

|  |  |  |
| --- | --- | --- |
| 月份 | 购保总票数（张） | 购保总金额（元） |
| 18/11 | 2103 | 26377 |
| 18/12 | 4994 | 61049 |
| 19/01 | 7399 | 87082 |
| 19/02 | 5011 | 54768 |
| 19/03 | 4568 | 50038 |
| 19/04 | 33306 | 341368 |
| 19/05 | 12421 | 126347 |

# B端问题

1. 无抢票功能
2. 爬虫通信的核心应用RRN还存在一些问题尚未解决，如内存持续增长，目前开发人员已经离职。
3. ADSL服务器资源不足，目前市场上的ADSL服务器资源已经被瓜分的差不多，留给我们的优质服务器不是很多，以往经验150台中，较好的机器有40台左右。
4. ADSL需要外采，并且缺乏一种完善的监控机制，厂商会有停机维护的现象。需要人工定时查看服务器运行情况。
5. **目前爬虫机器人基于12306PC版本实现，在非高峰时期订票效率 20s – 40s**。
6. PC版本高峰期，订票效率低下 1分钟-5分钟，而且12306 PC有瘫痪的情况。
7. 需要一个兜底的供应商，并且这个供应商要服务稳定。（灾备方案）
8. **目前爬虫核心服务 RRN Server单机运行，存在单点故障的问题。**
9. PC版本验证码打码是一个额外的花费，需要接入专门的供应商（小黑鱼）。
10. **账号目前可用在60w-70w，目前12306在持续封禁长时间未修改密码的账号，每日两位数的消耗（10）**。
11. 需要人工运营成本。
12. APP版本需要破解。

# 落库流程

## 车次/经停站数据落库



## 路线数据落库



# 附件

## B端机器人功能文档



## RRN设计文档



## 供应商接口文档



其他接口

<https://doc.oschina.net/open_api_v1>

https://www.juhe.cn/docs/api/id/22  
https://www.juhe.cn/docs/api/id/173  
http://elife.qianmi.com/esj/2016-08-23/85.html?t=1

其中：

* 途牛在用
* 航天华有已经停止合作
* 携程具体情况未定
* 其他属于未接入