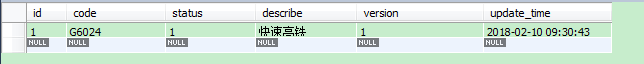
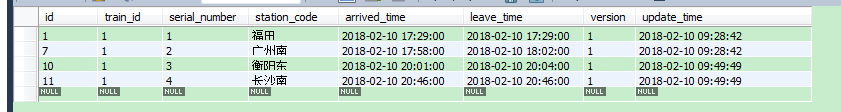
1. 数据结构分析
2. 静态数据
   1. 车次基本信息(train\_info)

说明:工作人员后台录入。



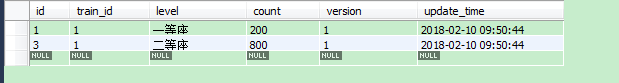
* 1. 车次经停信息(train\_station)

说明: 工作人员后台录入。



* 1. 车次座位信息(train\_seat\_info)

说明: 工作人员后台录入。



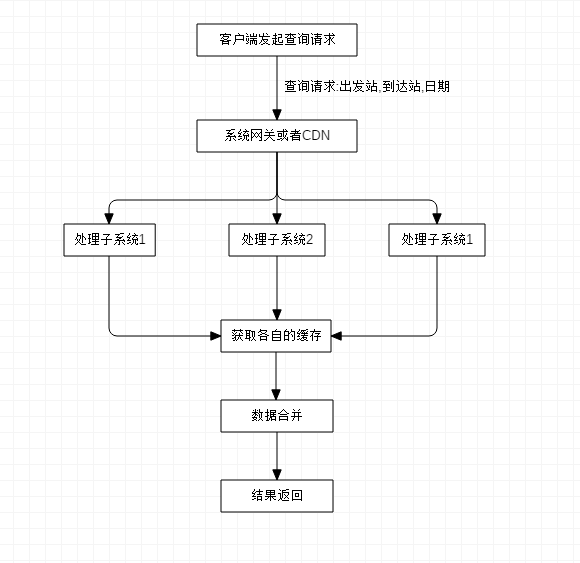
1. 动态数据(主要性能瓶颈数据)
   1. 车次座位库存信息(train\_seat\_stock\_20180210)

说明:

1. 具备以天为周期和顺序性。
2. 因为12306有30天预售,所以表数据可以通过定时job根据以上表数据信息提前生成，并且在缓存中保存一份提高查询性能。
3. 因为库存数据需要做写操作,所以以每车次每天生成一张库存表，这样分表可以降低锁的粒度来提高并发。



1. 用户流程分析
   1. 查询车次

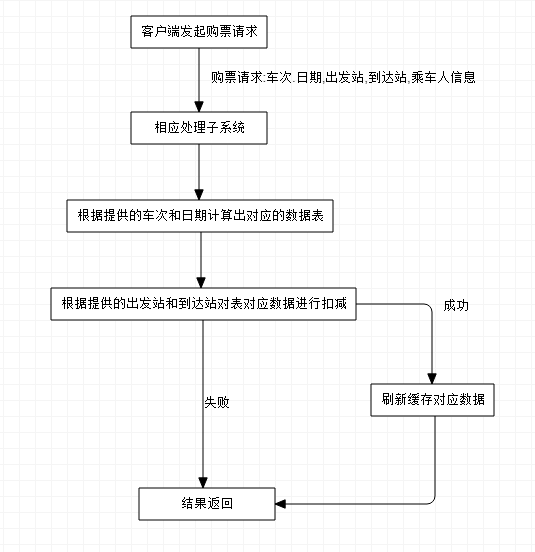


流程步骤:

1. 客户端发送请求(出发日期，出发站，到达站)。
2. CDN或者系统网关根据提供的起始站(获取起始站相关的所有车次)提供相应的处理系统路由分发。
3. 处理子系统根据 车次 日期 出发站 到达站 4个维度作为索引查询缓存库存数据。redis.mget(车次+日期, 出发站+到达站)
4. 将各处理子系统返回结果合并。
5. 客户端展示。

说明:

1. 非核心容错这里不作讨论。
2. CDN或者网关获取车次路由分发容错，需要处理。
3. 不需要保证缓存数据的绝对实时，这个对于查询来说是可接受的。
4. 因为查询并发非常大，不需要查询缓存未命中时再查询数据库这个流程，因为车票销售有预售期，有定时job生成车票库存信息数据表，并刷新缓存，所以此流程缓存100%命中，如果未命中则系统异常(需要在定时job那里处理)。
   1. 购买车票



流程步骤:

1. 客户端发起购票请求(日期,车次,出发站,到达站,座位,乘车人信息)。
2. 根据车次可以路由到对应处理子系统。
3. 根据日期车次计算出购票请求所需要操作的库存数据表。
4. 对库存表里与请求的数据路途冲突的记录字段ticket\_count-n 是否成功(ticket\_count约束>=0)。冲突条件如下:
   1. 库存出发站序号>= 请求出发站序号 并且 库存出发站序号<请求到达站序号) 。
   2. 库存到达站序号>= 请求出发站序号 并且 库存到达站序号<=请求到达站序号。
5. 扣除成功,刷新缓存对应的数据或者失败结果返回。

说明:

1. 非核心容错这里不作讨论。