# redis存储方案设计：

typedef struct TrainsInfo

{

char group[30]; //车站组代码

char groupend[30]; //终点站 --------新增字段

int trainsno; //车组号

int srvno; //服务号

int direction; //列车运行方向（规约传入上行为1，下行为2，未知状态0）

int state; //状态（0为无车，1为到站，2为离站）

int platlocation; //站台类型，0表示上行站台，1表示下行站台

int jumpsta; //跳停标志，1表示跳停，0表示正常

int arrivet; //本站到达时间（时间戳）

int departuret; //本站出发时间

}TrainsInfo;

站台提供的列车数据中车站组代码、车组号、方向、本站到达时间可以确定唯一的一条数据，保证存历史库和实时记点界面更新不会错误；

redis存储方案，从起点站到终点站，车组号为a的列车，在这段时间内是不可能重复的，那么redis中首先以组代码、车组号、方向字段作为判断唯一一条数据的标识(可以查看下方的数据结构hash ttr:hash:组代码:车组号:方向)，列车a到达站点m时，存入redis中组代码、终点站、列车号、服务号、方向、状态、本站到达时间，以组代码、车组号、方向字段为标识的字段存入处理数据队列，

然后离站时存入本站出发时间，并再次把此条数据的组代码、车组号、方向字段为标识的字段存入处理数据队列；

em\_ttr处理 等待处理数据队列时，从第一条开始读，先查找前站，获取前站出发时间，并根据前站的组代码、列车号、方向、本站到达时间去更新前站的后站到达时间，然后删除hash ttr:hash:组代码:车组号:方向，防止本天内数据标识重复错误；

处理完前站，再处理本站数据，根据当前的数据和拿到的前站出发时间，建立一个新的hash，键名为: ttr:hash:组号:车组号:方向:本站到达时间；就这样先处理前站数据，删除前站的临时内存，然后处理本站的数据，新建立hash数据，直到本站判断是终点站，那么删除本站的临时内存数据。

这种方案引起的问题：

1. 判断一辆列车从起始站到终点站的标识从原模型：组代码、车组号、服务号转变成当前模型：组代码、车组号、方向、本站到达时间，那么假如申通需要增加列车a从起始站到终点站的所有数据，原数据模型可以满足，新模型由于加上了本站到达时间作为判别标志，不能满足此种需求；
2. 站点m发出了列车a无车状态，即列车a还未到站，此条数据显示界面本站到达时间是0，列车a到站时，会创建新的hash，界面上便会残留一条列车a的无车信息；

需要更改的程序：规约使用的redis写入库，em\_ttr，实时记点插件，以及可能需要修改的列车记点历史插件；

**规约存入redis初始数据**

哈希hash数据：

键名：ttr:hash:组代码:车组号:方向

域：组代码、终点站、列车号、服务号、方向、状态、本站到达时间、本站出发时间

列表list（未处理的记点列表）

键名：ttr:list:data

存储值：ttr:hash:组代码:车组号:方向

**ttr服务处理：**

对列表list的处理：

先根据hash‘ttr:hash:组代码:车组号:方向’更新另一个hash‘ttr:hash:组号:车组号:方向:本站到达时间’即前站的后站到达时间，然后删除hash‘ttr:hash:组代码:车组号:方向’；

接着创建本站的hash‘ttr:hash:组号:车组号:方向:本站到达时间’，然后根据hash‘ttr:hash:组代码:车组号:方向’中的‘终点站’判断本站如果是终点站，则删除本站的hash‘ttr:hash:组代码:车组号:方向’。

根据上面列表list处理后生成---

有序集ZSET

键名: ttr:zset:组号

存储值：ttr:hash:组号:车组号:方向:本站到达时间

哈希hash数据：

键名：ttr:hash:组号:车组号:方向:本站到达时间

域：组号、车组号、服务号、方向、状态、本站到达时间、本站出发时间

增加终点站：

char groupend[30]; //终点站 --------新增字段 ，目的是为了终点站是物理意义上起点站和终点站之间的站台，如从a到c(终点站),然后到停车区，然后接着从d到h，此时redis中c站的内存块需要提前删除，防止起始站d站有前站出发时间