**里程修正设计文档**

# 一、里程修正类组成部分

## 1、需要进行里程修正的原因

假设A，B是公里标，A为35km, B为36km，在正常状态下，B = A + 1000米，也就是 36km = 35km + 1000米。数学上也是正确的。

但是现在因为某种原因，使得AB之间的实际距离不再是1000米，可能比1000米长（比如AB之间把一段直线铁轨改造成了曲线铁轨），那么，AB之间就有一个长链。此时，可以修改原先涉及到的所有的工程图纸，把里程重新丈量并在图纸上标注正确，同时也要把现场的里程标（一根已经埋好了的柱子）重新调整。但这样的话工作量太大了，因此，就设置了长短链表，这样，原先的工程图纸不用动，现场的已经做好了的里程标也不用改造。做某种计算的时候只需要把长短链表考虑进去即可。

## 2、里程修正实体类

### 2.1长短链设计的结构

长短链类【LongChain】

该类主要操作的是InnerDB数据库中的长短链表。

其中ExtraLength计算是根据长短链的类型判断的，

对于一个长链点，长出的里程为：ExtraLength=米数-1000；

对于一个短链点，缺少的里程为：ExtraLength=米数；

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | LineCode | string | 线路编号 |
| 2 | Dir | int | 行别 |
| 3 | Km | float | 公里标 |
| 4 | ExtraLength | float | 长出或短出的里程 |

### 2.2用户手动标记点的设计

用户手动标记类【UserMarkedPoint】

用于存储用户在图层上面的操作，记录用户的操作，会存储到idf文件中的IndexOri表中。这个表按顺序记录用户的操作，无需排序。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | ID | string | ID |
| 2 | FilePointer | long | 文件流指针 |
| 3 | UserSetMileage | float | 设置的标准点里程(米) |

### 2.3里程修正数据的设计

里程修正类【MileStoneFixData】

该类主要是用来做里程校正的表的一个实体类的对应，对应数据库中的表为IndexSta表，

用户标注的点之间的点的数目也可以计算出来，那么，用户标注的点之间的采样点的采样间距就为：设某段用户标注点之间的实际里程为L， 期间采样点数为M：采样间距 = L / (M-1)

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | ID | string | ID |
| 2 | MarkedStartPoint | UserMarkedPoint | 标记的起始点 |
| 3 | MarkedEndPoint | UserMarkedPoint | 标记的终止点 |
|  | Chains | List< LongChain> | 标记点之间的长短链 |
|  | RealDistance | float | 标记点之间的实际距离，单位为米 |
|  | SamplePointCount | long | 标记点之间的采样点个数 |
|  | SampleRate | float | 两个采样点之间的距离 |

# 二、里程修正类操作设计

## 1、长短链操作类

长短链操作类【LongChainTable】

该类主要是读取长短链表中的配置信息数据，为里程修正提供一些基本信息。

根据线路编号、增减里程以及行别获取长短链的集合，也可以根据标注的里程获取之间的长短链集合。

## 2、用户标记操作类

用户标记操作类【UserFixedTable】

该类主要是记录用户的操作，存储到IndexOri表中，这个表按顺序记录用户的操作。包括以下几种操作：

读取用户操作信息

根据增减里程读取用户操作记录，如果是增里程升序，如果是减里程降序。

删除所有用户操作信息

删除清除IndexOri表中的所有数据。

删除指定的用户标记点

根据传入的标记点的ID，删除指定的标记点信息。

保存一个用户标记点

将用户在图层上的操作保存在IndexOri表中，包括某个采样点在cit文件中的偏移位置（用户在点击屏幕上的一个点，会映射到cit中的某个点，和用户标注的新里程。

保存多个用户标记点

按照保存用户标记点的信息保存多个。

## 3、里程修正操作类

里程修正操作类【MilestoneFix】

该类是进行里程修正计算的最主要的方法。

**1、获取长短链信息集合，以及用户的标记点信息。**

用户的标记点的个数必须大于等于2个，才能进行下一步的操作。

**2、计算两个标记点的实际距离以及标记点之间的采样个数。**

两个标记点的实际里程是标记点结束里程减去标记点的开始里程加上长链长出的里程和短链短出的里程。

两个标记点之间的采用个数是调用cit操作类中的获取采样点个数的方法获取到采样点个数。

**3、计算标记点的采样间距和平均采样间距。**

采样间距为标注点之间的实际里程除以采样点的个数减一。即设某段用户标注点之间的实际里程为L， 期间采样点数为M：采样间距 = L / (M-1)

平均采样间距为把用户标注点两两之间的采样间距求平均值。

**4、计算起始里程**

（1）计算的起始里程中的标记终止点（这里标记为A）为上面步骤中获取修正数据的第一个点的标记起始点

（2）计算的起始里程中的采样点个数为（这里标记为B）：

a获取里程修正数据中的第一个数据的标记点位置

b使用cit操作类中获取指定标记点的位置的采样点个数函数，获取标记点位置的采样点个数

（3）计算的起始里程中的距离（这里标记为C）为步骤三中的平均采样间距

（4）计算的起始里程中的标记开始点：

偏移位置为使用cit操作类获取cit文件的第一个点的偏移位置；

设置的标准里程为A-（B-1）\*C

（5）计算的起始里程中的标记点之间的实际距离为步骤一减去步骤四

（6）将这条记录插入到里程修正数据的第一个位置

**5、计算结束里程**

（1）计算的起始里程中的标记开始点（这里标记为A）为上面步骤中获取修正数据的最后一个点的标记终止点

（2）计算的起始里程中的采样点个数为（这里标记为B）：

a获取里程修正数据中的最后一个数据的标记点位置

b使用cit操作类中获取指定标记点的位置的采样点个数函数，获取标记点位置的采样点个数

（3）计算的起始里程中的距离（这里标记为C）为步骤三中的平均采样间距

（4）计算的起始里程中的标记终止点：

偏移位置为使用cit操作类获取cit文件的最后一个点的偏移位置；

设置的标准里程为A+（B-1）\*C

（5）计算的起始里程中的标记点之间的实际距离为步骤四减去步骤一

（6）将这条记录插入到里程修正数据的最后一个位置