Map-walker 相关模块设计与实现 1

包含模块：datab，log，router
运行环境：python 3.5
日期：2016/6/3

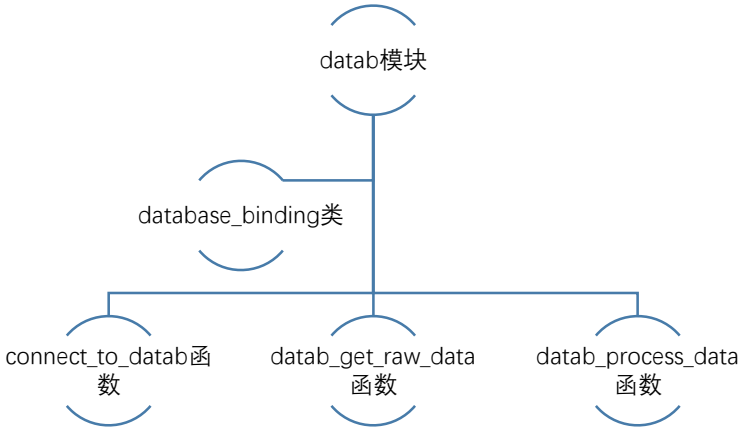
➢ datab 模块

■ 概述

该模块内通过一系列函数实现了对数据库的连接、读取以及处理等基础操作，将这些函数在一个类内进行统一调用，由 core 模块进行实例化后便可或得经过处理的数据库信息，方便后期核心算法调用。

模块内经后期改动，对每一条直接路径采用 router_path 类保存，该类内包含了该路径的所有信息，经过处理数据库后可得到一个二维列表，列表中每一项是一个包含有这两个城市之间所有直达路径类的 list。

■ 函数与类具体设计



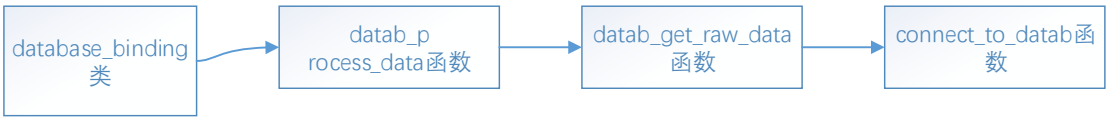
整个模块内函数均如上图所示，下面我将对各个函数及类一一做具体说明。

- connect_to_datab 函数
  - 传入参数：无
  - 功能：连接到项目根目录下 data/data.db 数据库文件
  - 返回值：指定数据库对象
  - 使用样例：sql = connect_to_datab()
- datab_get_raw_data 函数
  - 传入参数：(数据库对象)
  - 功能：从数据库中执行查询语句获得指定飞机、火车、汽车三个表的内容矩阵
  - 返回值：(飞机数据矩阵， 火车数据矩阵， 汽车数据矩阵)
  - 使用样例：(raw_data_flight，raw_data_train，raw_data_bus) = datab_get_raw_data(sql)
- datab_process_data 函数
  - 传入参数：(飞机、火车或汽车三个原生数据矩阵)
  - 功能 :对于三个原生数据矩阵中任意两个城市之间的直接路径生成一个 router_path 类并混合成为一个 list 保存到二维列表中。
  - 返回值：(输入矩阵处理过后的二维列表)
  - 使用样例：data_flight = datab_process_data(raw_data_flight，raw_data_train，raw_data_bus)
- database_binding 类
  - 传入参数：无

- 包含函数功能：__init__初始化函数用于组织及调用上述各个数据库处理函数，从而获得综合有三种交通方式的二维 list。
- 使用样例：database=database_binding()

■ 调用说明及样例

该模块内所有函数调用均在 database_binding 类的__init__初始化函数中进行，交叉引用如下图所示：



■ 存在的问题及后期规划

目前数据库处理方面缺乏差错处理等，后期还在继续完善中。

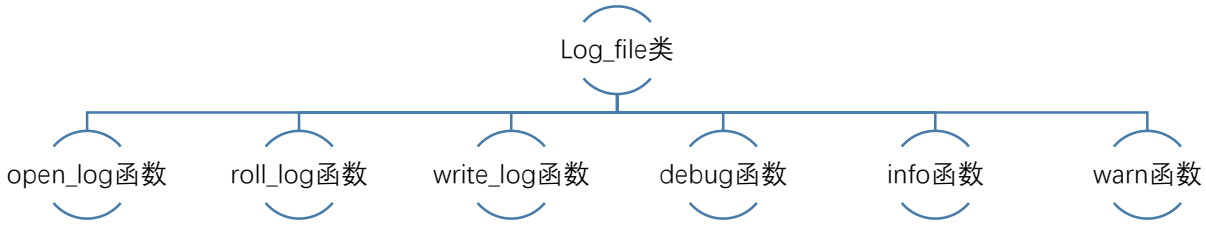
➢ Log 模块

■ 概述

该模块旨在实现日志文件的写入及存储管理，其中日志文件分为两种：log.txt 和 log_test.txt，第一个为常规的日志包含数据库的读写信息、客户端请求以及程序运行中的常规事件；第二个为调试日志，其中将包含更加详细地信息，服务端运行过程中的任何一次函数调用以及数据都将被记录在案，方便出现问题后开发者及时处理。

存储管理则是该模块中一个亮点，考虑到服务端运行时可能会接受到大量地客户端请求以及长时间地运行情况，若不进行日志存储管理，单个地日志文件将变得越来越多臃肿，对于查看日志造成了十分大的障碍。因此该模块将在初始化时对于日志文件进行检查，若文件超过一定行数就将对当前日志进行备份为.bak 文件，同时清空该日志，之后再进行正常写入工作。

■ 函数与类具体设计



- ◆ __init__函数
  - 传入参数：无
  - 功能：实现类成员变量初始化并调用 open_log 函数打开日志文件
- ◆ open_log 函数
  - 传入参数：self
  - 功能：实现打开文件根目录下/data/log.txt 文件并判断文件行数
  - 返回值或类成员变量：修改类成员变量—日志文件对象
  - 使用样例：self.open_log()
- ◆ roll_log 函数

- 传入参数：self
- 功能：将当前日志文件重命名后转移到指定目录
- 返回值：无
- 使用样例：self.roll_log()

◆ write_log 函数

- 传入参数：(self, mode, fmt, *msg)，其中 mode 用于选择写入模式，fmt 用于指定输入的格式，msg 可以输入一个 list。
- 功能：实现将指定消息按照固定格式写入日志文件中
- 返回值：无
- 使用样例：self.write_log('DEBUG', '%d %s', 21, 'hello')

◆ debug 函数

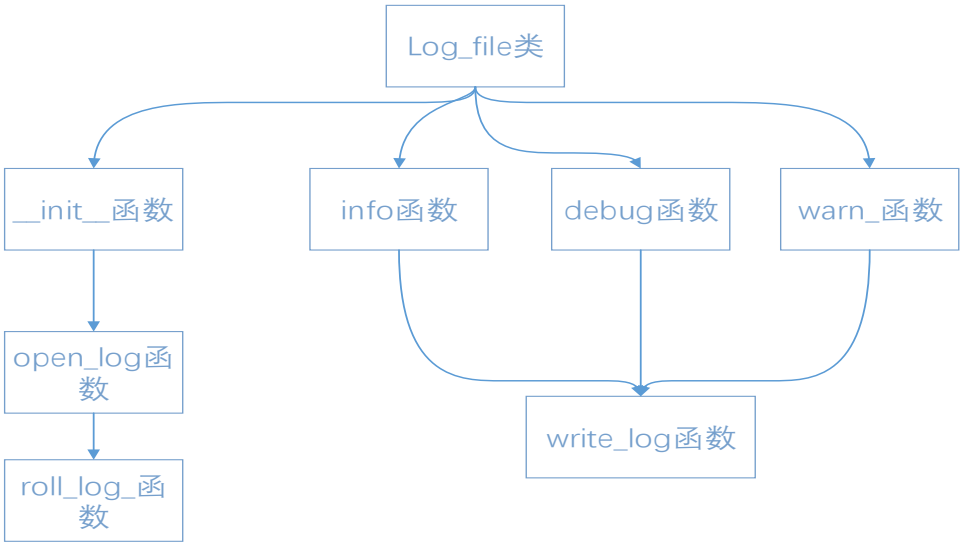
- 传入参数：(self, fmt, *msg)
- 功能：实现 write_log 模式中的 debug 模式，写入日志文件后改行会有 debug 标记
- 返回值：无
- 使用样例：self.debug('%d %s', 21, 'hello')

◆ info 函数

- 传入参数：(self, fmt, *msg)
- 功能：实现 write_log 模式中的 info 模式，写入日志文件后改行会有 info 标记
- 返回值：无
- 使用样例：self.info('%d %s', 21, 'hello')

- warn 函数
  - 传入参数：(self, fmt, *msg)
  - 功能：实现 write_log 模式中的 warn 模式，写入日志文件后改行会有 warn 标记
  - 返回值：无
  - 使用样例：self.warn(‘%d %s’ , 21, ‘hello’)

■ 调用说明及样例



该类中实例化一个 Log_file 类的对象 log_file，因此可以通过在其他模块中直接引用 log_file 对象进行日志文件的写入。

■ 存在的问题及后期规划

目前程序中只加入了普通日志文件的写入，还较为基础，后期规划中将会引入 log_test 的详细设计以及对于 log_test 的自动分析从而使得系统可以简单判断服务器出现的问题。

➢ router 模块

■ 概述

该模块内实现了对三种策略的最优路径计算，其中最少花费和最少时间策略采用改造迪杰斯特拉算法，而限制时间最少花费策略采用有限广度优先搜索的方式进行。为方便计算，还设计了一些小函数方便转换时间。

■ 函数与类具体设计

- ◆ Translata 函数
  - 传入参数：一个 router_path 类的 list
  - 功能：转换类为列表便于在客户端和服务器传送数据
  - 返回值：list
  - 使用样例：无
- ◆ Transfer 函数
  - 传入参数：一个整数
  - 功能：转换时间长度为一个时间点的 datetime 类型
  - 返回值：datetime
  - 使用样例：无
- ◆ transfer_reverse 函数
  - 传入参数：datetime 类型的时间点
  - 功能：转换一个 datetime 类型到时间长度
  - 返回值：整数

- 使用样例：无

◆ minimal_cost_path 函数

- 传入参数 :(source, destination), 均为城市 id 号, 其中 destination 为一个城市 id 列表，代表到达多个城市
- 功能 :通过改造迪杰斯特拉算法计算两个城市间的最少花费路径, 当向已找到路径结点列表中加入一个城市时, 如果这个城市是终点城市列表中的一个, 那么无条件将到达其他所有城市的路径更改为经过该城市的路径。其余均还采用 dijkstra 算法。
- 返回值：一条经过城市路径的 list
- 使用样例：minimal_cost_path(1, [2, 3 , 9])

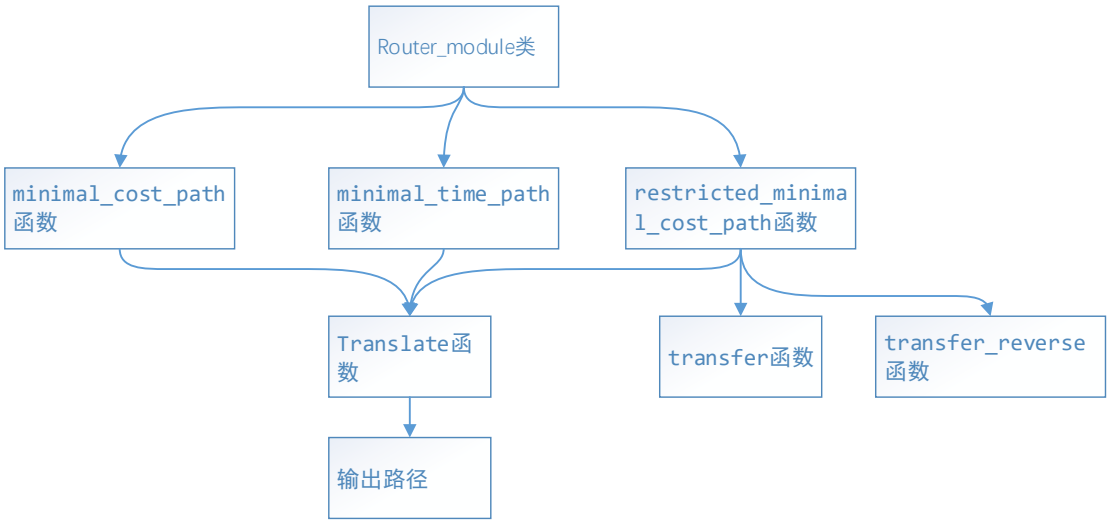
◆ minimal_time_path 函数

- 传入参数 :(source, destination), 均为城市 id 号, 其中 destination 为一个城市 id 列表，代表到达多个城市
- 功能：计算一个城市到其他终点城市的时间最少路径。此时，因为要考虑不同车次的发车时间不同可能造成的等车时间，因此，在 dijkstra 算法中增加一个 list 用于存储到达该城市时的时间偏移, 在计算到该城市时间时应先将该时间偏移转换为时间点与发车时间作比较，计算时间差加入到下次时间花费中。其余还采用和最少花费相同策略。
- 返回值：一条经过城市路径的 list
- 使用样例：minimal_ time_path(1, [2, 3 , 9])

◆ restricted_minimal_cost_path 函数

- 传入参数：(source, destination, restrict)，加入了一个 restrict 参数作为时间限制
- 功能：从起点城市开始创建一颗树，并同时进行 BFS 搜索，当从 queue 中取到的第一个节点显示找到所有城市时即停止搜索，表明找到路径。另外，在搜索过程中对于任何超过限制的节点停止扩展，优化了时间。
- 返回值：一条经过城市路径的 list
- 使用样例：restricted_minimal_cost_path(1, [2, 3], 500)

■ 调用说明及样例



■ 存在的问题及后期规划

目前，该模块内对于第三种策略，即限制时间最少花费策略，当终点城市较多时，计算时间较长，因为采用了搜索策略，造成一棵树的节点输呈指数形式增长，尽管采用了某些限制策略，但还是显得有些吃力，后期会对该策略进行进一步优化，尝试进行动态规划处理，可能效果会更好。