

SUSTech CS307 Database Project2

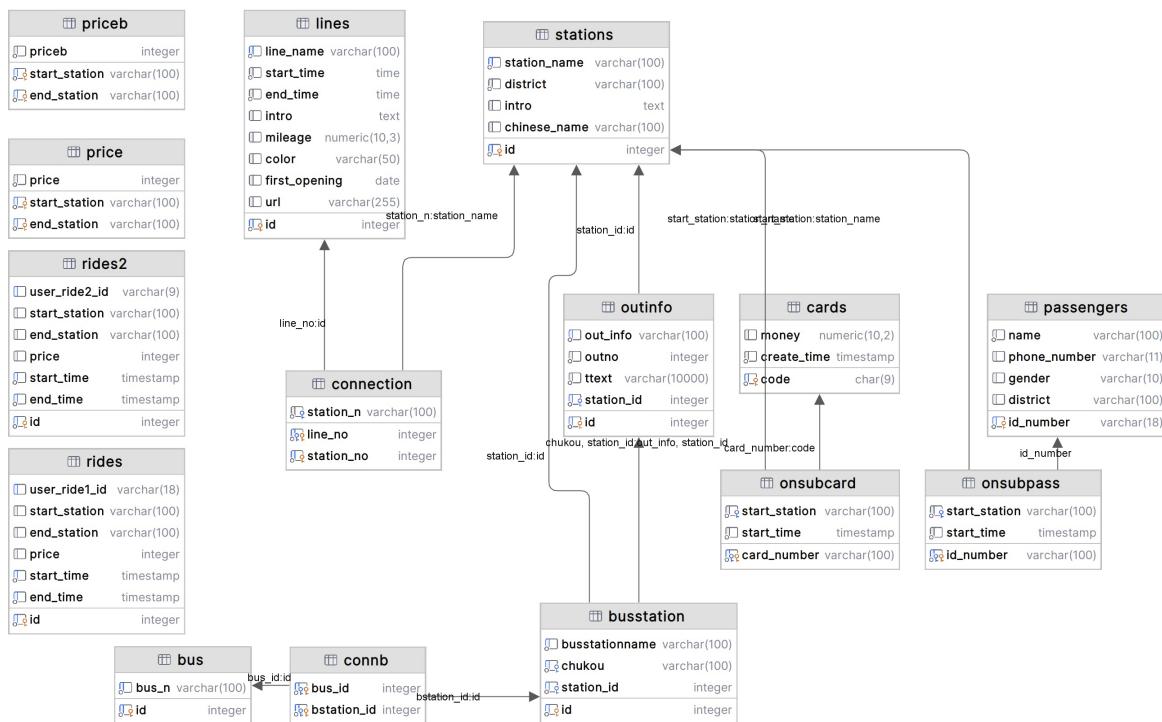
Spring 2024

Start at May 1, End at June 2 , 2024

12210651贾适萌 CS307 2024春 周三56班 贡献比100%

对Proj1数据库表做的改动:

1. 解析原表Price.xlsx, PriceB.xlsx (商务车厢) , 通过遍历 (x_station, y_station) , 得到对应的 priceOut.csv 和 priceOutB.csv, 转化脚本见 `PriceTrans.py`, `PriceTransB.py` , 并通过 `ImportPrice.py` 和 `ImportPriceB.py` 导入数据库表 price 和 priceB 。
2. 去掉大数据集 rides 和 rides2 的外键, 增加两个表 onSubCard 和 onSubPass, 存储正在地铁上的乘客和乘车卡, 在乘客下车后, 再把乘客/乘车卡的乘车记录存进 rides/rides2。
3. station 表增加一列车站状态。
4. 更改Connction表的主键, 从primary key (line_No, station_No)到primary key (line_No, station_n), 避免了update时的unique问题。



1 Basic Part 基础API设计

1.1 添加、修改、删除一个地铁站 *interface sAMD.java*

```

//增加一个地铁站
void add_a_station(String name, String district, String intro, String
chinese_name);
//修改一个地铁站
void modify_a_station(String nameI, String name, String district, String
intro, String chinese_name);
//删除一个地铁站
void delete_a_station(String name);

```

1.2 添加、修改、删除一个地铁线 *interface IAMD.java*

```

//增加一条地铁线
void add_a_line(String line_name, String start_time, String end_time, String
intro, String mileage, String color, String first_opening, String url);
//修改一条地铁线
void modify_a_line(String nameI, String line_name, String start_time, String
end_time, String intro, String mileage, String color, String first_opening, String
url);
//删除一条地铁线
void delete_a_line(String name);

```

1.3 地铁站与地铁线的关系 *interface cConn.java*

```

//将一个地铁站在放入一个地铁线中指定的位置上
int putStationOnLine(String name, String l_name, int order);
//判断stations里有没有存在这个地铁站 返回true则有，false则无
boolean stationExist(String name);
//判断某条lines上有没有这个地铁站 返回true则有，false则无
boolean stationExistInLine(String name, String lineName);
//将一个地铁站从一个地铁线中移除
int deletestationFromLine(String l_name, String name);
//判断lines里有没有存在这个地铁站 返回true则有，false则无
boolean LineExist(String name);
//通过线路的名称获取线路编号 返回值是线路id
int getLineNoByName(String lineName);

```

1.4 查询某一个地铁站在一个地铁线路上向前数第n站与向后数第n站的地铁站名字 *interface Query.java*

```

//查询某一个地铁站在一个地铁线路上向前数第n站的车站名 返回值是n的值 答案在table里
String forwardQuery(String station_name, String line_name, int n,
DefaultTableModel tableModel);
//查询某一个地铁站在一个地铁线路上向后数第n站的车站名 返回值是n的值 答案在table里
String backwardQuery(String station_name, String line_name, int n,
DefaultTableModel tableModel);
//通过线路的名称获取线路编号 返回值是线路id
int getLineNoByName(String lineName);

```

1.5 实现乘客或公交卡上车功能 *interface TARide.java*

1.6 实现乘客或公交卡下车功能 *interface TARide.java*

1.7 查询当前时间已经上车但是还未出站的乘客与公交卡信息 *interface TARide.java*

```

//使用乘客信息上车
int UpSubwayPass(String pass_id, String station_n, int type);
//使用卡片信息上车
int UpSubwayCard(String card_id, String station_n, int type);
//返回值为1，车站不存在；返回值为2，乘客信息不存在；
//返回值为3，卡片信息不存在；返回值为4，车站状态不在“运营中”。

```

```

//使用乘客信息下车
//返回值为1，车站不存在；返回值为2，乘客信息不存在；返回值为3，乘客不在地铁上；返回值为4，车站状态不在“运营中”。
int DownSubwayPass(String pass_id, String station_n);

//使用卡片信息下车
//返回值为1，车站不存在；返回值为2，卡片信息不存在；返回值为3，卡片不在地铁上；返回值为4，车站状态不在“运营中”；返回值为9，卡上没钱了。
int DownSubwayCard(String card_id, String station_n);

```

```

//查询当前时间已经上车但是还未出站的乘客
void currentPass(DefaultTableModel tableModel);
//查询当前时间已经上车但是还未出站的卡片
void currentCard(DefaultTableModel tableModel);

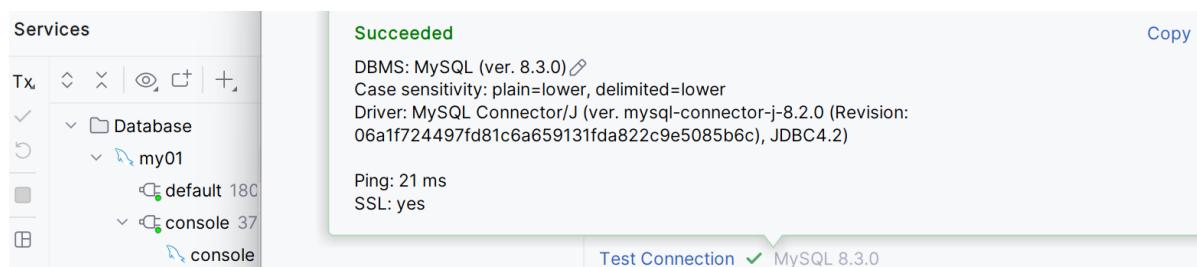
//判断这个乘客在不在车上 返回true为在，false为不在
boolean passExistInSub(String name);
////判断这个卡片在不在车上 返回true为在，false为不在
boolean cardExistInSub(String name);
//判断stations里有没有存在这个地铁站 返回true则有，false则无
boolean stationExist(String name);
//判断cards里有没有存在这个乘车卡 返回true则有，false则无
boolean cardExist(String name);
//判断passengers里有没有存在这个乘客 返回true则有，false则无
boolean passExist(String name);
//查看当前车站的状态：返回状态的名称。
String checkState(String name);

```

2 Advance Part

2.1 使用 MySQL数据库 完成项目。

在project2中，整个项目皆基于MySQL(ver.8.3.0)完成。



2.2 更多的API或其他功能设计

- 查询两个地铁站之间的路径 *interface Navigation.java*

```
//查询两个地铁站之间的路径, tableModel展示无需换乘的路径, tableModel2展示需要换乘的路径  
(每个地铁坐几站, 换乘站) (并按照总站数从短到长排序)  
//返回值为1, 车站不存在; 返回值为3, 展示无需换乘的tableModel; 返回值为4, 展示需要换乘的  
tableModel2  
int Nav(String start, String end, DefaultTableModel tableModel,  
DefaultTableModel tableModel2);  
//判断stations里有没有存在这个地铁站 返回true则有, false则无  
boolean stationExist(String name);
```

如下图为查询结果, 以世界之窗到大学城为例。

初始站	线路1	站数1	换乘站1	线路2	站数2	换乘站2	线路3	站数3	目的地	总站数
世界之窗	2号线	3	安托山	7号线	6	西丽	5号线	1	大学城	10
世界之窗	1号线	4	车公庙	7号线	8	西丽	5号线	1	大学城	13
世界之窗	1号线	4	桃园	12号线	5	灵芝	5号线	5	大学城	14
世界之窗	2号线	3	后海	11号线	2	前海湾	5号线	10	大学城	15
世界之窗	1号线	9	宝安中心	5号线	7			0	大学城	16
世界之窗	1号线	7	前海湾	5号线	10			0	大学城	17
世界之窗	1号线	7	会展中心	4号线	7	深圳北	5号线	3	大学城	17
世界之窗	1号线	4	车公庙	11号线	4	前海湾	5号线	10	大学城	18
世界之窗	2号线	10	市民中心	4号线	6	深圳北	5号线	3	大学城	19
世界之窗	1号线	10	科学馆	6号线	7	深圳北	5号线	3	大学城	20
世界之窗	1号线	8	岗厦	10号线	8	五和	5号线	5	大学城	21
世界之窗	2号线	11	岗厦北	10号线	7	五和	5号线	5	大学城	23
世界之窗	2号线	9	福田	11号线	5	前海湾	5号线	10	大学城	24
世界之窗	2号线	11	岗厦北	14号线	3	布吉	5号线	11	大学城	25
世界之窗	2号线	9	海上世界	12号线	11	灵芝	5号线	5	大学城	25

- 增加地铁站状态: 建设中、运营中、关闭中 (stations表)

受地铁站状态的限制:

- 乘客只能从正在运营中的地铁站出发, 在正在运营中的地铁站下车。
- 在基础模块的添加地铁站中, 手动添加的地铁站的状态为“建设中”, 如果要加入运营, 要先设定这个车站到所有车站的price/priceB, 再将状态更新至“运营中”。

合理使用: *interface cState.java*

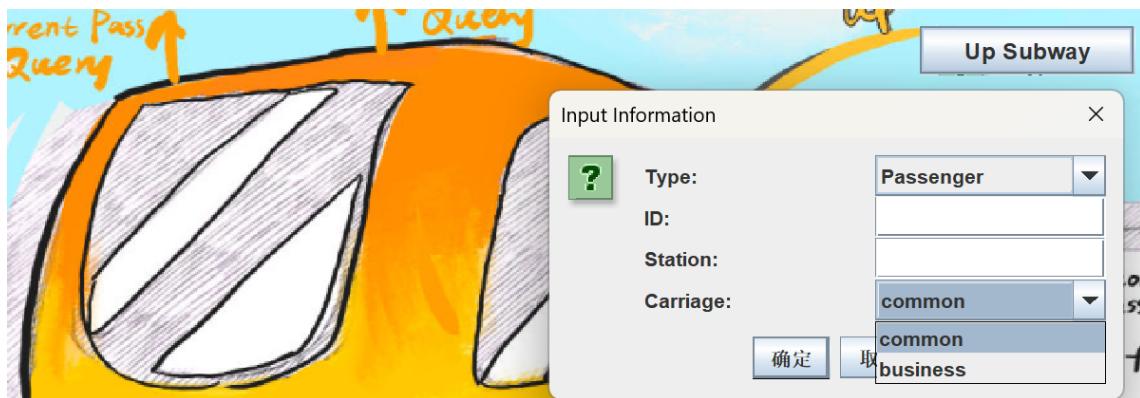
查看一个地铁站的状态, 开启一个地铁站 (? → 运营中), 关闭一个地铁站 (? → 关闭中)

```
//开启一个地铁站 (? -> 运营中) 返回值是1, 车站不存在; 返回值是0, 更新状态  
int restart(String name);  
//关闭一个地铁站 (? -> 关闭中) 返回值是1, 车站不存在; 返回值是0, 更新状态  
int stop(String name);  
//查看一个地铁站的状态 返回值是状态  
String checkState(String name);  
//判断stations里有没有存在这个地铁站 返回true则有, false则无  
boolean stationExist(String name);
```

- 商务车厢

解析原表Price.xlsx, 以及根据《深圳市发展和改革委员会关于公布深圳市轨道交通2023年新线开通票价表的通知》, 下载PriceB.xlsx (商务车厢), 通过遍历 (x_station, y_station), 得到对应的 priceOut.csv 和 priceOutB.csv, 转化脚本见 `PriceTrans.py`, `PriceTransB.py`, 并通过 `ImportPrice.py` 和 `ImportPriceB.py` 导入数据库表 price 和 priceB。

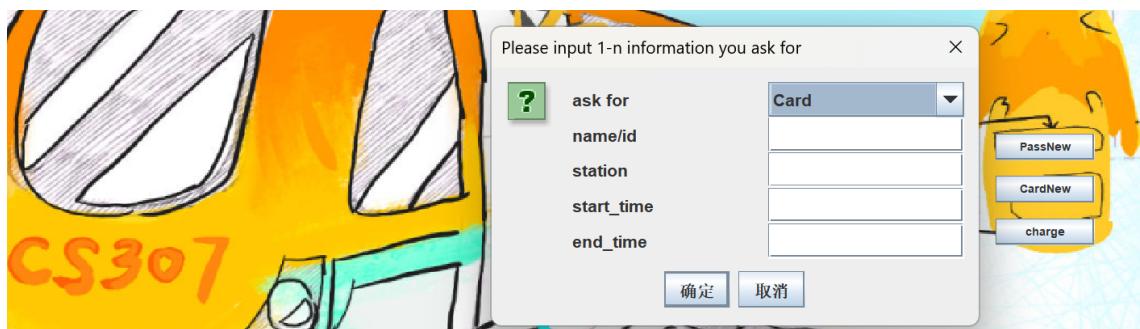
在乘车上下车时可以选择乘坐普通车厢还是商务车厢 (common/business), 将选择记录在 onSubPass 和 onSubCard 的 type 中, 根据 type 在下车时查询 price/priceB 对应的票价。



- 多参数搜索乘车记录功能 *interface askR.java*

通过地铁站、乘车人、时间段等实现1-n个参数搜索乘车记录功能，首先选择是根据card查询还是根据passenger查询，接下来的三个条件（姓名/卡号，上车/下车车站，时间区段）可填写1-n个，只根据已填写的信息进行乘车记录的查询。

```
//查询passenger的rides表 将结果存在tableModel里
//返回值为1，车站不存在；返回值为11，参数为0；返回值为2，没有查到记录；返回值为3，查到记录。
int askRide1(String name, String staN, String time1, String time2,
DefaultTableModel tableModel);
//查询card的rides2表 将结果存在tableModel里
//返回值为1，车站不存在；返回值为11，参数为0；返回值为2，没有查到记录；返回值为3，查到记录。
int askRide2(string id, string staN, string time1, string time2,
DefaultTableModel tableModel);
//判断stations里有没有存在这个地铁站 返回true则有，false则无
boolean stationExist(string name);
```



- 乘客注册，办卡，给卡充值 *interface PCnew.java*

```
//乘客注册 首次乘车需要注册成为一个乘客，填写基本信息
void BeAPassenger(String name, String id, String phone, String gender, String district);
//办卡 想要使用乘车卡乘车，要办卡，并进行首次的充值
String TakeACard(double money);
//给卡充值 卡里钱不够的乘客无法完成乘车，要给卡进行充值
int recharge(String code, double money);
```



- 显示地图 showMap

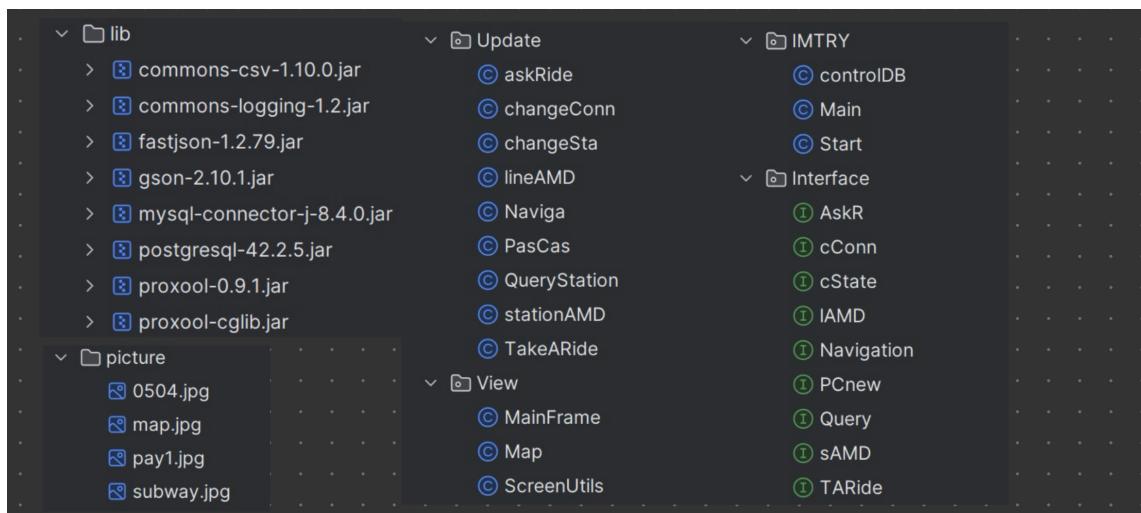
显示一个可以拖动的，可以放大缩小的可视化深圳地铁线路图，方便乘客进行更加清晰的路线规划。



2.3 后端设计

- 使用了代码包管理

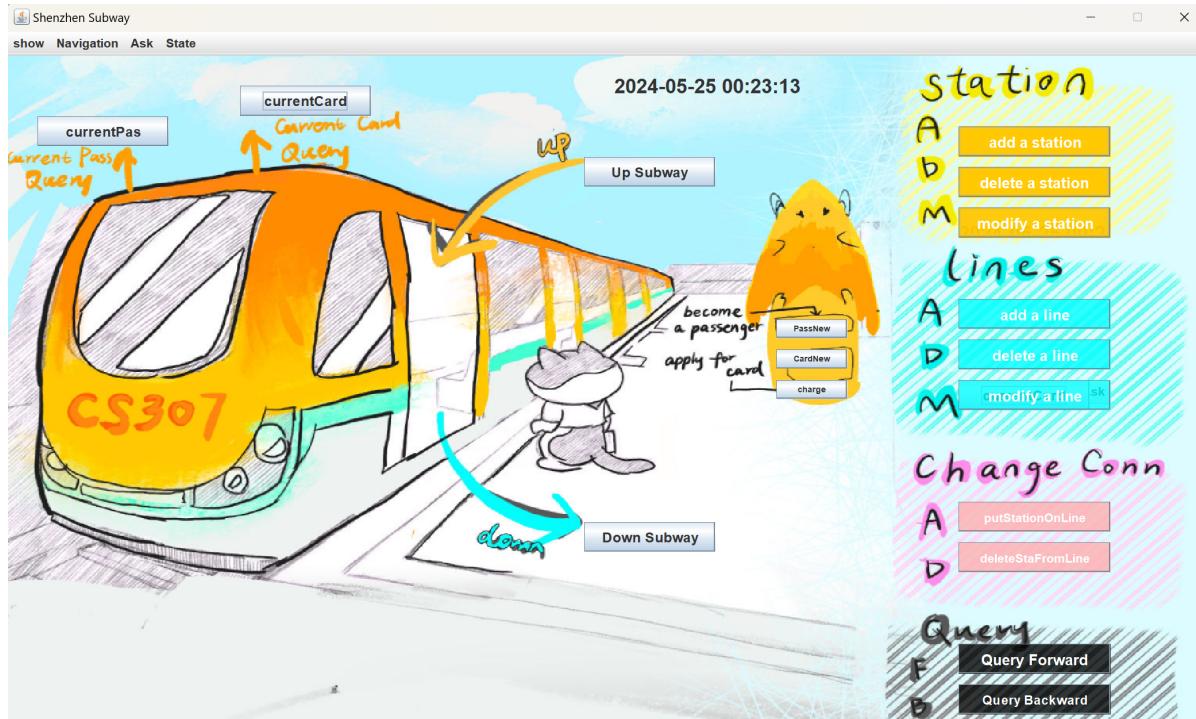
lib文件夹存放需要使用的库，picture文件夹存放前端需要使用的图片，Interface中是所有的API，Update里的文件实现了对应的接口，View文件夹存放了所有前端相关的代码，IMTRY文件夹中存放了程序起始代码和控制数据库开启关闭的代码。



- 符合API规范

代码符合API规范，以及整个程序启动只需要一次就可以加载所有功能。

2.4 一个好看的GUI设计



本次project2采取了 **Java Swing GUI** 进行前端设计，主要基于了Jframe, JButton, JMenu等进行了页面的基础设计，以实现对应功能的响应，右侧四个模块分别对应了车站的增删改，线路的增删改，车站在线路上的位置修改，车站在线路上的位置查询；up箭头的起点是上车的功能，down箭头的终点是下车的功能；地铁上方的两个箭头指向了当前地铁上的乘客/卡片；橙色小娃的身上的三个按钮分别对应了乘客注册，办卡，卡片充值；在JMenu里，show中的show map可以展示深圳地铁的线路图，Navigation的Navi way可以查询任意两个车站之间的路径，Ask中的ask Ride可以多参数搜索乘车记录功能，State里的check state, state open, emergency close可以检查、开启、关闭地铁站的状态，前端代码详见 `MainFrame.java`。

背景图线稿出自b站up主**幾加乘**，本人在五一期间使用**procreate**进行了颜色的填充和描改，并进行了功能的分块。【我不想坐地铁上班啊啊啊啊啊啊啊啊啊啊】(https://www.bilibili.com/video/BV1H34y1Z7mm/?spm_id_from=333.337.search-card.all.click&vd_source=b8eb575bc5111bfd0340f4c6fdee50af)

2.5 合理的使用数据库的其他功能

- 索引

对稳定的大数据集 rides, rides2建立索引，可以加快查询的速度。

```
create index rides_index on rides(id);
create index rides2_index on rides2(id);
create index price_index on price(id);
create index priceb_index on priceb(id);
```

如图以 rides 表为例，在id上添加index，可以看到一次查询的时间从240ms减小到119ms。

```
m1> SELECT * FROM rides
      WHERE (start_time >= '2023-02-07 19:53:44' and end_time <= '2023-02-09 20:00:44')
[2024-05-29 15:44:08] 276 rows retrieved starting from 1 in 240 ms (execution: 48 ms, fetching: 192 ms)
m1> create index rides_index on rides(id)
[2024-05-29 15:46:24] completed in 265 ms
m1> SELECT * FROM rides
      WHERE (start_time >= '2023-02-07 19:53:44' and end_time <= '2023-02-09 20:00:44')
[2024-05-29 15:46:31] 276 rows retrieved starting from 1 in 119 ms (execution: 7 ms, fetching: 112 ms)
```

- 触发器

一个trigger，当往onsubpass/onpasscard表里插入上车的乘客/卡片的信息的时候触发它，触发后检查插入的start_time是不是早于05:49:00以及是不是晚于23:30:00（地铁线路起始运营最早时间和结束运营最晚时间），如果时间不在这个范围内，则中断插入，并返回65，提示报错。

```
CREATE TRIGGER trg_check_time_onsubpass
BEFORE INSERT ON onsubpass
FOR EACH ROW
BEGIN
    IF HOUR(NEW.start_time) < 5 OR
        (HOUR(NEW.start_time) = 5 AND MINUTE(NEW.start_time) < 49) OR
        HOUR(NEW.start_time) > 23 OR
        (HOUR(NEW.start_time) = 23 AND MINUTE(NEW.start_time) > 30) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid start_time. It
should be between 05:49:00 and 23:30:00';
    END IF;
END;
```

2.6 大数据管理

对于rides和rides2，在开始proj2之前，去掉了大数据集 rides 和 rides2 的外键，增加两个表 onSubCard 和 onSubPass（相当于两个过渡表，存放临时的乘车记录），存储正在地铁上的乘客和乘车卡，在乘客下车后，再把乘客/乘车卡的乘车记录存进 rides/rides2。这样在频繁地上下车中，也加快了整个数据的管理效率。

项目2报告到此结束。