

非关系型数据存储技术及其应用实践项目报告

说明：这里的结果截图可能与演示时有所区别，因为在提交报告到演示的这段时间内可能会对某些前端界面加以美化，不过后端整体逻辑不会改变

技术栈

后端开发采用了spring生态，包括spring，spring boot，spring data JPA

前端开发采用了vue框架，包括html，css，JavaScript等

数据库采用的是neo4j图数据库

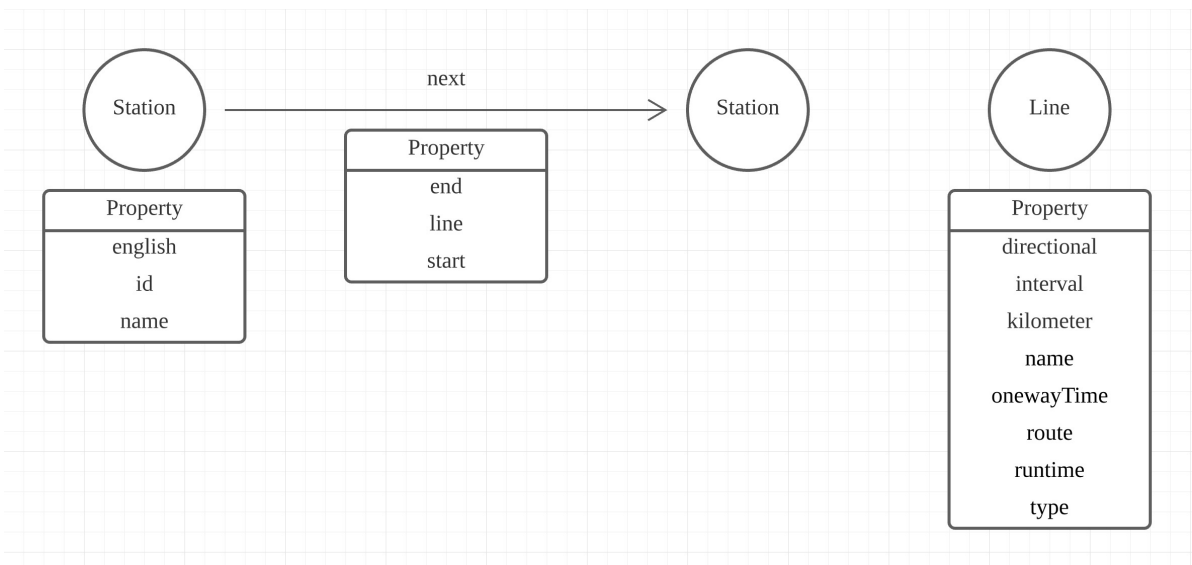
数据库选型

本项目的多个需求涉及到“线路站点查询”、“最短路径”等问题，需要考虑大量单位数据间的联系。如果采用传统的关系型数据库，我们必须针对每种关系存储大量的冗余数据，造成空间的浪费和查询效率的底下。

对于这类需要大量查询数据与数据间联系的问题，采用节点和节点间联系进行存储的图数据库是更好的选择。为了能够直观地查看数据间的联系、测试方法是否生效，我们选用了 Neo4j 图数据库作为本项目的数据库。

数据库设计

数据库设计如下图：



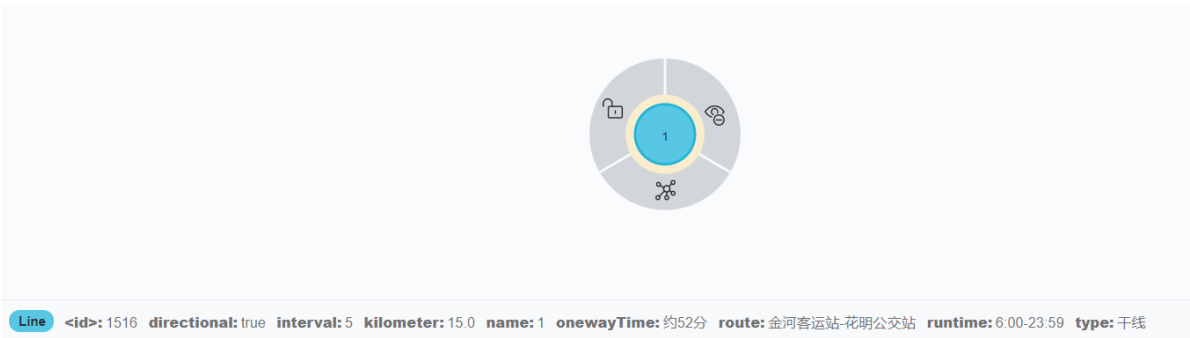
说明：

- line节点中对应存储的是lines.json文件中的数据，相当于MySQL中的一张lines表
- station节点中对应存储的是station.json文件中的数据，相当于MySQL中的一张station表

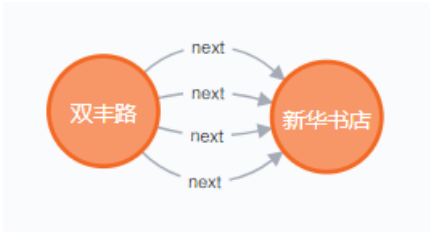
- 对于routes.json和timetable.json的处理，我们采取了在station和station之间建立next联系，具体为
 - 若站点s1有n条指向s2的路线，则在s1和s2间有n个next，其中每个next中的line属性对应路线名称
 - 每个next中的start数组表示某条线路到达站点s1的时间（如果s1是首发站，则表示从s1发车的时间），若该条线路有n趟班次，则start数组有n个元素；end数组中对应保存的是start数组中每趟班次到达s2的时间

示例如下：

某个line节点：（这里的id是neo4j默认创建的，并不是我们设计的属性）：



某两个station：可以看到有四条线路经过双丰路到达了新华书店



某个next：可以看到四条线路中有一条是N11路下行，并且有很多趟班次（time表示的是从startStation到endStation所需要的时间，不过后来发现是冗余的，但考虑到占用的空间不多，因此没有删除）

"r"
{ <u>"start"</u> : ["23:48", "23:58", "00:08", "00:18", "00:28", "00:38", "00:48", "00:58", "01:08", "01:18", "01:28", "01:38", "01:48", "01:58", "02:08", "02:18", "02:28", "02:38", "02:48", "02:58", "03:08", "03:18", "03:28", "03:38", "03:48", "03:58", "04:08", "04:18", "04:28", "04:38", "04:48", "04:58", "05:08", "05:18", "05:28", "05:38", "05:48"], <u>"end"</u> : ["23:50", "00:00", "00:10", "00:20", "00:30", "00:40", "00:50", "01:00", "01:10", "01:20", "01:30", "01:40", "01:50", "02:00", "02:10", "02:20", "02:30", "02:40", "02:50", "03:00", "03:10", "03:20", "03:30", "03:40", "03:50", "04:00", "04:10", "04:20", "04:30", "04:40", "04:50", "05:00", "05:10", "05:20", "05:30", "05:40", "05:50"], "time": 2, <u>"line"</u> : "N11路下行"}

完成的需求

1. 完成的需求及分数：

需求	得分
1	1
2	2
3	2
4	3
5	6
6	2
7	2
8	2
9	2
10	2
11	4
12	2
13	2
14	4
16	2
17	2
20	2
总计	42

2. 需求的实现：

贴出的代码中，上半部分是DAO层的代码（即Cypher查询语句，如果有的话），下半部分是Service层的代码

（1）查询某条线路的基本信息：

```
Line findByName(String name);
```

通过继承Neo4jRepository<Line, Long>的类自动生成查询返回信息

查询30路的基本信息：

<input type="text" value="30"/>		<input type="button" value="查询"/>					
是否有向	班次间隔 (分钟)	单向里程 (公里)	线路名	运行时长	线路走向	运营时间	线路类型
是	6分钟	12公里	30	约48分钟	植原-北路湾公交站	6:30-22:30	干线

(2) 查询某条线路方向的全部站点信息：

```
@Query("MATCH p=(start:Station)-[r:next{line:$lineName}]->(end:Station)
RETURN p ORDER BY id(r)")
List<Station> findRouteStationsByLineName(@Param(value = "lineName") String
name);
```

这里的id(r)指的是neo4j对每个关系（节点）自动创建的id，因此可以通过order by id(r)保证顺序，因为插入数据时是有序的。

（要不要写一下数据入库为啥是有序的？）

查询2路上行的基本信息：

2路上行

查询

序号	Id	站点名称	英文名称
1	7542	兴义镇 (始发站)	XingYiZhen
2	7527	永盛 (始发站)	YongSheng
3	7504	南华大道	NanHuaDaDao
4	7485	南华大道南	NanHuaDaDao S
5	21460	金河客运站	JinHeKeYun
6	803	金河公园	JinHePark
7	98730	平桥	Peace Bridge
8	14214	食品城	Food City
9	761	生态公园	Ecologic Park
10	750	画展中心	Art Center
11	744	航天立交东	Hangtianlijiao E
12	5155	航天立交南	Hangtianlijiao S
13	5142	玩具城	Toyshop
14	28139	新山农贸市场	XinShan Market
15	28843	果丰	Guo Feng
16	28845	凤凰路口	Phoenix Kou
17	2128	杨柳河南	Willow River S
18	97825	杨柳河北	Willow River N
19	14417	双桥立交北	Shuangqiaozi N
20	14423	学府路	Xuefulu
21	97836	地铁东井巷	Dongjinxiang
22	16772	金飞路一段	JinFeilu One
23	2800	蓝天立交北	LanTianLiJiao N
24	6855	蓝天立交南	LanTianLiJiao S
25	6901	百花路东	Bai Hua Lu E
26	6854	百花路西	Bai Hua Lu W
27	6818	商贸城	Trace City
28	14495	火车西站公交站 (终点站)	Huo Che Zhan

(3) 查询站点（锦城广场站）停靠的所有线路：

```
@Query("match (s:Station{name:$stationName})-[r:next]-(:Station) return distinct s.id+collect(distinct r.line) as lines")
List<Value> newFindLinesByStationName(@Param(value = "stationName") String stationName);

@Override
public List<StationRoutes> newFindLinesByStationName(String stationName) {
```

```

List<Value> lineList =
lineRepository.newFindLinesByStationName(stationName);
List<Object> allLineList = new ArrayList<>();
List<StationRoutes> answer = new ArrayList<>();
for (Value line : lineList) {
    List<Object> list = line.asList();
    allLineList.add(list);
}
for(Object obj : allLineList) {
    Collection<?> collection = (Collection<?>) obj;
    StationRoutes temp = new StationRoutes();
    for(Object o : collection) {
        if(o instanceof Long) {
            temp.setId((Long) o);
        } else {
            temp.getLines().add((String) o);
        }
    }
    answer.add(temp);
}
return answer;
}

```

主要逻辑在cypher语句中完成：所有停靠该站点的线路，并定会经过该站点。并且以该站点为startStation：该站点是起始站；以该站点为endStation：该站点是终点站；以该站点位为startStation和endStation：该站点是经停站

Service中的代码只是对返回的结果进行封装。

查询锦城广场站停靠的所有线路

锦城广场	查询
同名的站点id	停靠的线路
58290	K5路上行
64355	N26路下行",G33路上行,17路下行
58289	K5路下行
64356	N26路上行,G33路下行,17路上行

(4) 查询某条线路从某站到某站，线路的运行方向、沿路站点和运行时长：

```

@Override
public RoutedTO findRouteByLineAndStation(String lineName, String start,
String end) {
    RoutedTO answer = new RoutedTO();
    answer.setStations(new ArrayList<>());
    Line line = lineService.findLineByName(lineName);
    List<List<Station>> stations = stationService.findStationsByLine(line);
    if (stations.size() == 1) {

```

```

        answer.setName(lineName + "路");
        List<Station> route = stations.get(0);
        int startIndex = -1, endIndex = -1;
        buildRoute(start, end, answer, route, startIndex, endIndex);
    } else {
        List<Station> upRoute = stations.get(0);
        List<Station> downRoute = stations.get(1);
        int startIndex = -1, endIndex = -1;
        for (int i = 0; i < upRoute.size(); i++) {
            if (upRoute.get(i).getName().equals(start))
                startIndex = i;
            if (upRoute.get(i).getName().equals(end))
                endIndex = i;
        }
        if (startIndex <= endIndex) {
            answer.setName(lineName + "路上行");
            for (int i = startIndex; i <= endIndex; i++) {
                answer.getStations().add(upRoute.get(i));
            }
        } else {
            answer.setName(lineName + "路下行");
            buildRoute(start, end, answer, downRoute, startIndex, endIndex);
        }
    }
    List<Station> answerRoute = answer.getStations();
    int cnt = answerRoute.size();
    String startTimeString =
        stationRepository
            .findTimetableByLineAndStartStations(answer.getName(),
answerRoute.get(0).getName())
            .get(0)
            .get(0)
            .toString()
            .substring(1, 6);
    String endTimeString =
        stationRepository
            .findTimetableByLineAndEndStations(answer.getName(),
answerRoute.get(cnt - 1).getName())
            .get(0)
            .get(0)
            .toString()
            .substring(1, 6);
    int time = TimeUtil.calculateTime(startTimeString, endTimeString);
    answer.setTime(time);
    return answer;
}

```

这里代码有点长，并且调用了一些其他类的方法，因此就简单说明一下主要的思路：首先判断是否是区分上下行的线路，如果是，分别比对，否则直接比对确定起点和终点之间的距离

查询10路从大悦城到小吃街：

线路名称: 10路 起点站: 大悦城 终点站: 小吃街 查询

线路名称: 10路上行

序号	经过的站点id	站点名称	站点英文名称
1	62753	大悦城	DaYueCheng
2	62765	肖家沟	XiaoJiaGou
3	62737	华通商场	HuaTang SC
4	62729	东林南路	DongLinNanLu
5	6354	东林小区 (始发站)	DongLinXiaoQu
6	6363	东林路	DongLin Road
7	6377	小吃街	Snack Street

总运行时间: 13分钟

(5) 查询某两个站台之间的最短路径:

这里可能要孙哥补充一下细节, 然后如果到时候汇报演示的时候如果问到了具体细节可能也要麻烦帮忙回答一下

按照ID查询如下 (考虑了最短时间):

```
@Override
public PathDTO findShortestPathByStationIds(long startId, long endId) {
    List<Object> objects = lineRepository.findShortestPathByStationIds(startId,
endId);
    List<String> next = new ArrayList<>();
    List<Station> stations = new ArrayList<>();
    if (objects.size() == 0) {
        return new PathDTO(null, null, -1);
    }
    Object object = objects.get(0);
    PathValue pathValue = (PathValue) object;
    Iterator<Relationship> relationships =
pathValue.asPath().relationships().iterator();
    boolean isStart = true;
    int time = 0;
    while (relationships.hasNext()) {
        Relationship relationship = relationships.next();
        long startNodeId = relationship.startNodeId();
        long endNodeId = relationship.endNodeId();
        if (isStart) {
            stations.add(stationRepository.findStationByInnerId(startNodeId));
            stations.add(stationRepository.findStationByInnerId(endNodeId));
            isStart = false;
        } else {
            stations.add(stationRepository.findStationByInnerId(endNodeId));
        }
        Iterator<String> relKeys = relationship.keys().iterator();
        while (relKeys.hasNext()) {
```



```

        String relKey = relKeys.next();
        if (relKey.equals("line")) {
            String relValue = relationship.get(relKey).asObject().toString();
            next.add(relValue);
        }
        if (relKey.equals("time")) {
            time += relationship.get(relKey).asInt();
        }
    }
}
return new PathDTO(next, stations, time);
}

```

按照站点名字查询，则是综合各个按照id查询的路径取时间最短的：

```


@Override
public PathDTO findShortestPathByStationNames(String startName, String
endName) {
    PathDTO answer = new PathDTO();
    int minTime = Integer.MAX_VALUE;
    List<Station> startStations =
stationRepository.findStationsByName(startName);
    List<Station> endStations = stationRepository.findStationsByName(endName);
    for (Station start : startStations) {
        for (Station end : endStations) {
            PathDTO tmp = findShortestPathByStationIds(start.getId(), end.getId());
            if (tmp.getTime() < minTime) {
                answer = tmp;
                minTime = tmp.getTime();
            }
        }
    }
    return answer;
}

```

例如查询从红瓦寺到动物园的最短路径：（这里前端界面稍微优化了一下）

线路信息		
最短路径	站点1 <input type="text" value="红瓦寺"/> 站点2 <input type="text" value="动物园"/>	<input type="button" value="立即查询"/>
直达线路		



 预计用时：9分钟



id: 16115

红瓦寺
Hongwasi



搭乘线路: G27路下行



id: 59548

天九街
TianJiuJie



搭乘线路: 70路上行



id: 5181

万安路东
Wan An Lu E



搭乘线路: N19路下行



id: 5197

万安路
Wan An Lu



搭乘线路: N19路下行



id: 5168

万安路西
Wan An Lu W



搭乘线路: 57路上行



id: 14768

动物园
Zoo

可以看到从红瓦寺到动物园的最短路径为：首先乘坐G27路下行到达天九街，然后乘坐70路上行到达万安路东，然后乘坐N19路下行到达万安路，然后继续乘坐N19路下行到达万安路西，最后乘坐57路上行到达动物园，预计用时9分钟

(6) 查询某两个站台间是否存在直达线路：

```
@Override
public List<String> isDirect(String start, String end) {
    List<String> answer = new ArrayList<>();
    Map<Station, List<String>> map =
lineService.findLinesByStationName(start); // 查询起点停靠的路线
    List<String> lineNames = new ArrayList<>();
    for (Map.Entry<Station, List<String>> entry : map.entrySet()) {
        lineNames.addAll(entry.getValue());
    }
    for (String lineName : lineNames) {
        List<Station> route =
stationRepository.findRouteStationsByLineName(lineName);
        int index = -1;
        for (int i = 0; i < route.size(); i++) {
            if (route.get(i).getName().equals(start)) {
                index = i;
                break;
            }
        }
        if (index == -1)
            continue;
        for (int i = index; i < route.size(); i++) {
            if (route.get(i).getName().equals(end)) {
                answer.add(lineName);
                break;
            }
        }
    }
    if (answer.isEmpty())
        answer.add("无直达线路");
    return answer;
}
```

这里考虑了方向，结合了需求3：“查询某个站点停靠的所有线路”和需求2：“查询某条线路方向的全部站点信息”。首先查找到起始站点停靠的所有线路，随后对于每条线路：查询该线路上的全部站点，如果结束站点在该线路上，并且出现在起始站点之后，则该条线路是直达线路。

查询从环球中心(始发站)到荷花池是否有直达线路：

起点站： 环球中心(始发站) 终点站： 荷花池 查询

N12路上行

(7) 查询某条线路某个方向的全部班次信息:

```
@Query(
    "MATCH p=(start:Station{name:$stationName})-[r:next{line:$lineName}]->
    (end:Station) RETURN r.start")
List<ListValue>
findTimetableByLineAndStartStations(
    @Param(value = "lineName") String lineName, @Param(value = "stationName")
String stationName);

@Data
public class ShiftDTO {
    List<Station> stations;
    List<List<String>> timetable;

    public ShiftDTO(List<Station> stations, List<List<String>> timetable) {
        this.stations = stations;
        this.timetable = timetable;
    }
}

@Override
public ShiftDTO findShiftInformation(String route) {
    List<Station> stations =
stationRepository.findRouteStationsByLineName(route);
    List<List<String>> timetable = new ArrayList<>();
    for (int i = 0; i < stations.size() - 1; i++) {
        List<String> time = new ArrayList<>();
        List<ListValue> tmp =
            stationRepository.findTimetableByLineAndStartStations(route,
stations.get(i).getName());
        for (ListValue it : tmp) {
            for (Object o : it.asList()) time.add(o.toString());
        }
        timetable.add(time);
    }
    List<ListValue> tmp = stationRepository.findTimetableByLineAndEndStations(
        route, stations.get(stations.size() - 1).getName());
    List<String> time = new ArrayList<>();
    for (ListValue it : tmp) {
        for (Object o : it.asList()) time.add(o.toString());
    }
    timetable.add(time);
    return new ShiftDTO(stations, timetable);
}
```

首先找到输入路线沿途所有站点，随后对于每一个站点（除了终点站）：找到该线路在这个站点的的时间表（即next中的start数组）；

对于终点站：找到该线路next的end数组。在前端页面中，第一行显示stations信息，第二行显示timetable的第一列，第三行显示timetable的第二列....直到遍历完所有班次。

查询239路上行的班次信息：结果太多了，截屏截不下来，因此交了一个录屏文件

(8) 查询某个时刻某个站台某个时段内即将停靠的线路:

```
@Query("match p=(s1:Station)-[r:next]->(s2:Station) where s2.id=$id return r.line+r.end")
List<List<Object>> findEndById(@Param("id") Integer id);

public void updateMap(Map<String,List<Integer>> map,String index,Integer
update)
{
    if(map.get(index)==null)
    {
        List<Integer> list=new ArrayList<>();
        list.add(update);
        map.put(index,list);
    }
    else
    {
        map.get(index).add(update);
    }
}

//需求8
@Override
public Map<String, List<Integer>> lineDocked(String now, Integer stationID,
int time) {
    Map<String, List<Integer>> map = new HashMap<>();
    Integer next_hour, next_minute;
    List<List<Object>> nexts = lineRepository.findEndById(stationID);
    String[] split = now.split(":");
    Integer now_hour = Integer.valueOf(split[0]);
    Integer now_minute = Integer.valueOf(split[1]);
    if(now_minute + time < 60)
    {
        next_hour = now_hour;
        next_minute = now_minute + time;
    }
    else
    {
        next_minute = now_minute + time - 60;
        next_hour = (now_hour + 1) % 24;
    }
    for(Object next : nexts)
    {
        String endList_tmp = next.toString();
        String regex = String.valueOf(endList_tmp.charAt(2));
        //      System.out.println(endList_tmp);
        String newList = endList_tmp.replace(regex,"").replace("
","").replace(",","").replace(")","").replace(" ", "");
        //      System.out.println(newList);
        String[] endList = newList.split(",");
        for (int i = 1; i < endList.length; i++) {
            String s = endList[i];
```

```

String[] split1 = s.split(":");
Integer hour = Integer.parseInt(split1[0]);
Integer minute = Integer.parseInt(split1[1]);
if(next_hour == 0 && now_hour == 23)
{
    if(hour == 23)
    {
        if(minute >= now_minute) updateMap(map,endList[0],minute -
now_minute);
    }
    if(hour == 0)
    {
        if(minute <= next_minute) updateMap(map,endList[0],minute -
now_minute);
    }
}
else
{
    if(hour == now_hour && minute >= now_minute)
    {
        if(hour == next_hour && minute <= next_minute)
        {
            updateMap(map,endList[0],minute -now_minute);
        }
        if(hour < next_hour)
        {
            updateMap(map,endList[0],minute -now_minute);
        }
    }
    if(hour > now_hour)
    {
        if(hour == next_hour && minute <= next_minute)
        {
            updateMap(map,endList[0],minute - now_minute + 60 * (hour -
now_hour));
        }
        if(hour < next_hour)
        {
            updateMap(map,endList[0],minute - now_minute + 60 * (hour -
now_hour));
        }
    }
}
}
return map;
}

```

代码也很长，不过主要的逻辑并不复杂：对于某个站点：返回所有到达该站点的线路，然后在这些线路里面，判断next.end是否在给定的时间范围内，如果在给定的时间范围内，则将该线路加入到结果集中，并且计算即将到达的时间，如果有多个，则返回该线路在接下来的时间里所有即将到达的时间。考虑到实现情况，这里只允许输入在接下来的60分钟内可能到达的线路

查询08:37分、ID=16147 (新华书店)、10分钟内即将停靠的线路：

现在的时间: 08:37 站点ID: 16147 X分钟内即将停靠的站点: 10 查询

序号	线路名称	即将停靠的时间
1	53路下行	即将在1分钟, 7分钟后到达
2	70路上行	即将在5分钟后到达
3	174路下行	即将在3分钟后到达
4	101路	该时刻已经到站, 且即将在5分钟, 10分钟后到达

(9) 查询某个时刻某个站台线路最近的3趟班次信息:

```
@Override
public Map<String, Integer> shiftsDocked(String now, Integer stationID) {
    Map<String, Integer> map = new HashMap<>();
    Integer next_hour, next_minute;
    List<List<Object>> nexts = lineRepository.findEndById(stationID);
    String[] split = now.split(":");
    Integer now_hour = Integer.valueOf(split[0]);
    Integer now_minute = Integer.valueOf(split[1]);
    for(Object next : nexts)
    {
        String endList_tmp = next.toString();
        String regex = String.valueOf(endList_tmp.charAt(2));
        String newList=endList_tmp.replace(regex,"").replace("
","").replace(",","").replace(")","").replace(" ", "");
        String[] endList = newList.split(",");
        int j = 1;

        for (int i = 1; i < endList.length; i++) {
            String s = endList[i];
            String[] split1 = s.split(":");
            Integer hour = Integer.parseInt(split1[0]);
            Integer minute = Integer.parseInt(split1[1]);

            if ((hour > now_hour) && j < 4) {
                map.put(endList[0] + "班次" + String.valueOf(j), minute - now_minute +
60 * (hour - now_hour));
                j++;
            }
            else if ((hour == now_hour && minute >= now_minute) && j < 4) {
                map.put(endList[0] + "班次" + String.valueOf(j), minute - now_minute);
                j++;
            }
            else if ((hour < now_hour) && j != 1 && j < 4) {
                map.put(endList[0] + "班次" + String.valueOf(j), (24 - now_hour + hour)
* 60 + minute - now_minute);
                j++;
            }
        }
    }
    return map;
}
```

和需求8差不多：返回所有到达该站点的线路，然后对于每条路线：根据next.end计算即将到达的时间，然后根据即将到达的时间取前三趟最早到达的班次（如果不足三趟则全部返回）

查询10:38分、ID=59760(地铁万盛)所有线路最近的三趟班次

现在的时间: 站点ID:

序号	线路名称	即将到达的时间
1	106路上行班次1	1分钟后到站
2	106路上行班次2	9分钟后到站
3	106路上行班次3	17分钟后到站
4	82路上行班次1	1分钟后到站
5	82路上行班次2	7分钟后到站
6	82路上行班次3	13分钟后到站
7	99路上行班次1	4分钟后到站
8	99路上行班次2	10分钟后到站
9	99路上行班次3	16分钟后到站

(10) 统计停靠路线最多的站点并排序：

```
@Query("match (s:Station)-[r:next]-(:Station) return s.name+collect(distinct r.line)+s.id as lines order by size(lines) DESC LIMIT 15")
List<Value> findStationsWithMostLines();
```

```
@Override
```

```
public List<StationRoutes> findStationsWithMostLines() {
    List<Value> lineList = stationRepository.findStationsWithMostLines();
    List<Object> allLineList = new ArrayList<>();
    List<StationRoutes> answer = new ArrayList<>();
    for (Value line : lineList) {
        List<Object> list = line.asList();
        allLineList.add(list);
    }
    for (Object obj : allLineList) {
        Collection<?> collection = (Collection<?>) obj;
        StationRoutes temp = new StationRoutes();
        temp.setRouteNum(collection.size()-2);
        for (Object o : collection) {
            if (o instanceof Long) {
                temp.setId((Long) o);
            } else {
                String judge = (String) o;
                if (judge.endsWith("路") || judge.endsWith("路上行") ||
judge.endsWith("路下行")) {
                    temp.getLines().add(judge);
                } else {
```



```
        temp.setName(judge);
    }
}
}
answer.add(temp);
}
return answer;
}
```

主要逻辑在cypher语句中完成：对于某个站的停靠路线：只要某条路线经过这个站点，则这个站点的停靠路线加1。同时为了避免重复计算一个线路两次，需要采用distinct

Service中的代码只是对返回的结果进行封装。

显示前15个：

查询前15个停靠线路最多的站点：

查询

序号	ID	站点名	线路条数	停靠线路
1	818	金河客运站(下客点)	13	"N12路下行", "N11路下行", "G38路上行", "G37路下行", "G28路下行", "G22路下行", "759路上行", "716路下行", "358路下行", "218路上行", "72路上行", "8路下行", "1路下行"
2	24646	凤溪大道和桥	11	"N31路下行", "G90路下行", "G41路", "736路上行", "735路上行", "727A路上行", "727路上行", "322路上行", "261路上行", "74路下行", "70路下行"
3	24645	凤溪大道和桥	11	"N31路上行", "G90路上行", "G41路", "736路下行", "735路下行", "727A路下行", "727路下行", "322路下行", "261路下行", "74路上行", "70路上行"
4	16433	科北路口	10	"N31路下行", "N11路下行", "G90路下行", "727A路上行", "727路上行", "243路下行", "101路", "74路下行", "53路下行", "15路下行"
5	24672	凤溪大道中	9	"N31路下行", "G41路", "736路上行", "735路上行", "727路上行", "322路上行", "261路上行", "74路下行", "70路下行"
6	17823	孵化园	9	"G62路", "G41路", "G37路上行", "735路下行", "102路", "101路", "83路上行", "30路上行", "15路下行"
7	24674	凤溪大道中	9	"N31路上行", "G41路", "736路下行", "735路下行", "727路下行", "322路下行", "261路下行", "74路上行", "70路上行"
8	59162	金河南站公交站(下客点)	9	"K6路下行", "K5路下行", "K2路上行", "N26路下行", "N4路上行", "G91路上行", "106路上行", "17路下行", "4路上行"
9	16432	科北路口	9	"N31路上行", "N11路上行", "G90路上行", "727A路下行", "243路上行", "102路", "74路上行", "53路上行", "15路上行"
10	803	金河公园	9	"N12路上行", "N11路上行", "G38路下行", "759路下行", "218路下行", "72路下行", "43路下行", "2路上行", "1路上行"
11	24563	凤溪大道南	8	"N31路下行", "G41路", "736路上行", "735路上行", "727路上行", "322路上行", "74路下行", "70路下行"
12	17848	金河市政府(始发站)	8	"G62路", "G37路下行", "735路上行", "102路", "101路", "83路下行", "30路下行", "15路上行"
13	98730	平桥	8	"N12路上行", "N11路上行", "218路下行", "72路下行", "57路上行", "43路下行", "2路上行", "1路上行"
14	24564	凤溪大道南	8	"N31路上行", "G41路", "736路下行", "735路下行", "727路下行", "322路下行", "74路上行", "70路上行"
15	749	画展中心	8	"N19路下行", "N12路下行", "G91路下行", "141路下行", "118路下行", "102路", "2路下行", "1路下行"

(11) 统计站点数量

A.统计特殊站台

```
@Query("MATCH (s:Station) where s.name =~ '地铁.*' return DISTINCT s.name")
List<String> findSubwayStations();

@Query("MATCH (s:Station) where s.name =~ '.*始发站.*' return DISTINCT s.name")
List<String> findDepartureStations();

@Query("MATCH (s:Station) where s.name =~ '.*终点站.*' return DISTINCT s.name")
List<String> findTerminalStations();
```

查询地铁站：

查询地铁站：

序号	名称	序号	名称
1	地铁惠南	11	地铁七河路口(终点站)
2	地铁摩尔百货	12	地铁钟楼(终点站)
3	地铁艺术馆	13	地铁钟楼(始发站)
4	地铁金河大道东(始发站)	14	地铁康河
5	地铁东井巷(始发站)	15	地铁蔡桥
6	地铁东井巷(终点站)	16	地铁金花
7	地铁七河路口	17	地铁四河(终点站)
8	地铁钟楼	18	地铁四河(始发站)
9	地铁金河大道东	19	地铁万盛
10	地铁七河路口(始发站)	20	地铁东井巷

查询起点站：

查询起点站：

序号	名称	序号	名称	序号	名称
1	海椒市(始发站)	21	地铁金河大道东(始发站)	41	文家场(始发站)
2	新南天地(始发站)	22	金河南站公交站(始发站)	42	天河公交站(始发站)
3	SM社区(始发站)	23	东林小区(始发站)	43	八里小区(始发站)
4	和盛苑(始发站)	24	金河市政府(始发站)	44	地铁七河路口(始发站)
5	植物园(始发站)	25	地铁东井巷(始发站)	45	桦林园(始发站)
6	金河旅游区(始发站)	26	钟楼(始发站)	46	地铁钟楼(始发站)
7	银杏园(始发站)	27	动物园公交站(始发站)	47	施家沟(始发站)
8	火车西站公交站(始发站)	28	永盛(始发站)	48	机投公交站(始发站)
9	高新软件园(始发站)	29	燎原(始发站)	49	驿都路(始发站)
10	合江客运(始发站)	30	清河镇(始发站)	50	普光公交站(始发站)
11	环球中心(始发站)	31	科北路(始发站)	51	兴义镇(始发站)
12	永丰公交站(始发站)	32	万寿乡(始发站)	52	币家店(始发站)
13	北路湾公交站(始发站)	33	高朋路(始发站)	53	三里坪公交站(始发站)
14	大同镇(始发站)	34	金河客运站(始发站)	54	东源路(始发站)
15	天华一路(始发站)	35	鸿石社区(始发站)	55	地铁四河(始发站)
16	两河路(始发站)	36	新兴工业园(始发站)	56	四医院(始发站)
17	庆安小区(始发站)	37	金融城(始发站)	57	高新新区公交站(始发站)
18	北客站(始发站)	38	花明公交站(始发站)	58	中海国际(始发站)
19	瘠后门(始发站)	39	庆安公交站(始发站)	59	曾家坡(始发站)
20	锦城西路(始发站)	40	鱼凫西路南(始发站)		

查询终点站：

查询终点站:

序号	名称	序号	名称	序号	名称
1	金河客运站(终点站)	21	北客站(终点站)	41	金河大道东(终点站)
2	海椒市(终点站)	22	动物园公交站(终点站)	42	地铁七河路口(终点站)
3	植物园(终点站)	23	庾后门(终点站)	43	桦林园(终点站)
4	金河旅游区(终点站)	24	燎原(终点站)	44	驿都路(终点站)
5	和盛苑(终点站)	25	金河市政府(终点站)	45	地铁钟楼(终点站)
6	新南天地(终点站)	26	地铁东井巷(终点站)	46	施家沟(终点站)
7	八里小区(终点站)	27	金融城(终点站)	47	机投公交站(终点站)
8	火车西站公交站(终点站)	28	清河镇(终点站)	48	环球中心(终点站)
9	银杏园(终点站)	29	北路湾公交站(终点站)	49	东林小区(终点站)
10	锦城西路(终点站)	30	科北路(终点站)	50	普光公交站(终点站)
11	高新软件园(终点站)	31	金河南站公交站(终点站)	51	币家店(终点站)
12	大同镇(终点站)	32	鸿石社区(终点站)	52	兴义镇(终点站)
13	永丰公交站(终点站)	33	新兴工业园(终点站)	53	三里坪公交站(终点站)
14	天河公交站(终点站)	34	高朋路(终点站)	54	东源路(终点站)
15	天华一路(终点站)	35	钟楼(终点站)	55	地铁四河(终点站)
16	永盛(终点站)	36	花明公交站(终点站)	56	高新新区公交站(终点站)
17	两河路(终点站)	37	鱼凫西南(终点站)	57	曾家坡公交站(终点站)
18	庆安小区(终点站)	38	庆安公交站(终点站)	58	中海国际(终点站)
19	万寿乡(终点站)	39	文家场(终点站)	59	合江客运(终点站)
20	四医院(终点站)	40	SM社区(终点站)		

B.统计某条线路单行站

```
@Override
public List<String> findSingleStations(String lineName) {
    Line line = lineRepository.findByName(lineName);
    Set<String> singleStation = new HashSet<>();
    List<List<Station>> routes = findStationsByLine(line);
    if (routes.size() == 2) {
        Set<String> upRouteSet = new HashSet<>();
        Set<String> downRouteSet = new HashSet<>();
        List<Station> upRoute = routes.get(0);
        List<Station> downRoute = routes.get(1);
        for (Station station : upRoute) {
            upRouteSet.add(station.getName());
        }
        for (Station station : downRoute) {
            downRouteSet.add(station.getName());
        }
        // System.out.println(upRouteSet);
        // System.out.println(downRouteSet);
        Set<String> tmp = new HashSet<>(upRouteSet);
        tmp.removeAll(downRouteSet);
        singleStation.addAll(tmp);
        tmp = new HashSet<>(downRouteSet);
        tmp.removeAll(upRouteSet);
        singleStation.addAll(tmp);
    }
}
```

```

        tmp.clear();
    }
    return new ArrayList<>(singleStation);
}

```

首先判断是否为环线，如果不是环线，就查询上行和下行的所有站点，之后取差集，就是单行站点。

查询208路的单行站：

序号	站点名称
1	金河旅游区(终点站)
2	金河旅游区
3	莲花一区
4	东源路(终点站)
5	东源路(始发站)
6	金河旅游区(始发站)

(12) 统计路线类型

```

    @Query("MATCH (l:Line) WHERE l.type = \"干线\" OR l.type = \"支线\" OR l.type = \"城乡线\" OR l.type = \"驳接线\" OR l.type = \"社区线\" RETURN COUNT (DISTINCT l)")
    int findNormalLineCount();
    @Query("MATCH (l:Line) WHERE l.type = \"快速公交\" RETURN COUNT (DISTINCT l)")
    int findKLineCount();
    @Query("MATCH (l:Line) WHERE l.type = \"高峰线\" RETURN COUNT (DISTINCT l)") int findGLineCount();
    @Query("MATCH (l:Line) WHERE l.type = \"夜班线\" RETURN COUNT (DISTINCT l)") int findNLineCount();

    @Override
    public List<Integer> findDifferentLinesCount() {
        List<Integer> answer = new ArrayList<>();
        answer.add(lineRepository.findNormalLineCount());
        answer.add(lineRepository.findKLineCount());
        answer.add(lineRepository.findGLineCount());
        answer.add(lineRepository.findNLineCount());
        return answer;
    }

```

主要逻辑在cypher语句中完成

结果为：

统计线路类型:

序号	线路类型	数量
1	常规公交	59
2	快速公交	7
3	高峰公交	16
4	夜班公交	11

(13) 查询两条线路重复的站点名:

```
@Override
public Set<Station> findSameStationsByRouteNames(String routeName1, String
routeName2) {
    List<Station> routeStations1 =
stationRepository.findRouteStationsByLineName(routeName1);
    List<Station> routeStations2 =
stationRepository.findRouteStationsByLineName(routeName2);
    Set<Station> answer = new HashSet<>();
    for (Station s1 : routeStations1) {
        for (Station s2 : routeStations2) {
            if (s1.getName().equals(s2.getName())) {
                answer.add(s1);
            }
        }
    }
    return answer;
}
```

首先查询两条线路的所有站点，随后对这两个集合取交集

查询15路上行、30路下行的重复的站点名:

第一条线路名称:

第二条线路名称:

序号	id	站点名称	英文名称
1	17824	孵化园	FuHuaYuan
2	17848	金河市政府(始发站)	Government
3	17809	北门立交东	BeiMenLiJiao E

(14) 查询线路换乘:

```
@Override
public Map<String, List<String>> findTransferRoutes(String routeName) {
    Map<String, List<String>> map = new HashMap<>();
    List<Station> route =
stationRepository.findRouteStationsByLineName(routeName);
    for (Station station : route) {
        Set<String> tmp = new HashSet<>
(lineRepository.findLineNameByStationId(station.getId()));
        tmp.remove(routeName);
        if (tmp.size() != 0)
            map.put(station.getName(), new ArrayList<>(tmp));
    }
    return map;
}
```

查询该线路的所有站点，之后对于每一个站点查询经过该站点的路线（除去该线路本身），即为该站点的可换乘路线

查询261路上行的可换乘线路：

261路上行

查询

序号	站点名称	线路
1	凤溪大道中	322路上行, 736路上行, 727路上行, 735路上行, N31路下行, G41路, 70路下行, 74路下行
2	培风路北	0路下行, 208路上行
3	双水村	16路下行, 15路下行, N20路上行
4	医学院	30路下行, 208路上行
5	花博路	736路上行, G41路, 70路下行, 74路下行
6	凤溪大道和桥	727A路上行, 322路上行, 736路上行, 727路上行, 735路上行, N31路下行, G41路, 70路下行, G90路下行, 74路下行
7	星空大道武科路口	16路下行, 15路下行, N20路上行, G41路
8	天府家园	16路下行, 15路下行, N20路上行
9	花博路口	736路上行, G41路, 70路下行, 74路下行
10	星空大道南	30路下行, 16路下行, 735路上行, N20路上行

(16) 根据站点数量对线路进行排序:

```
@Override
public List<Map.Entry<String, Integer>> findMostStationsRoutes() {
    List<Map.Entry<String, Integer>> answer = new ArrayList<>();
    List<String> allRoute = findAllRoutes();
    for (String route : allRoute) {
        int cnt = stationRepository.findRouteStationsByLineName(route).size();
        answer.add(Map.entry(route, cnt));
    }
}
```

```

    }
    answer.sort((a, b) -> b.getValue().compareTo(a.getValue()));
    List<Map.Entry<String, Integer>> limitedAnswer = new ArrayList<>();
    for (int i = 0; i < 15; i++) {
        limitedAnswer.add(answer.get(i));
    }
    return limitedAnswer;
}

```

查询所有线路，对于每条线路，分别查询所有站点的数量，然后进行排序

结果如下：

查询前15个站点数量最多的路线：

序号	线路名称	站点数量
1	736路上行	47
2	736路下行	46
3	735路上行	44
4	735路下行	44
5	727路下行	39
6	727路上行	37
7	82路上行	35
8	322路下行	35
9	17路上行	34
10	17路下行	34
11	322路上行	34
12	334路下行	34
13	47路上行	33
14	82路下行	33
15	334路上行	33

(17) 根据运行时间对线路进行排序

```

@Override
public List<Map.Entry<String, Integer>> findLongestRunTimeRoutes() {
    List<Map.Entry<String, Integer>> answer = new ArrayList<>();
    List<Map.Entry<String, Integer>> limitedAnswer = new ArrayList<>();
    List<String> routes = findAllRoutes();
    for (String route : routes) {
        List<Station> stations =
stationRepository.findRouteStationsByLineName(route);
        String startTime =
            stationRepository.findTimetableByLineAndStartStations(route,
stations.get(0).getName())

```

```

        .get(0)
        .get(0)
        .toString()
        .substring(1, 6);
String endTime =
    stationRepository
        .findTimetableByLineAndEndStations(route,
stations.get(stations.size() - 1).getName())
        .get(0)
        .get(0)
        .toString()
        .substring(1, 6);
int time = TimeUtil.calculateTime(startTime, endTime);
answer.add(Map.entry(route, time));
}
answer.sort((a, b) -> b.getValue().compareTo(a.getValue()));
for (int i = 0; i < 15; i++) {
    limitedAnswer.add(answer.get(i));
}
return limitedAnswer;
}

```

查询路线，该路线起始站的start和终点站的对应end相减即为该路线的运行时间，随后排序。

结果为：

查询前15个运行时间最长的路线:

序号	线路名称	运行时间 (分钟)
1	736路上行	90
2	736路下行	90
3	735路上行	86
4	735路下行	86
5	727路下行	75
6	727路上行	72
7	17路上行	65
8	17路下行	65
9	334路下行	65
10	82路上行	64
11	334路上行	64
12	716路上行	64
13	716路下行	63
14	47路上行	62
15	82路下行	62

(20) 线路删除

```
@Query("MATCH p=()-[r:next{line:$route}]->() DETACH DELETE r")
void deleteRoute(@Param(value = "route") String route);

@Query("MATCH (n:Station) WHERE NOT (n)--() DELETE n")
List<String> deleteAllIsolatedStations();

@Override
public void deleteLine(Line line) {
    if (line.isDirectional()) {
        String upRoute = line.getName() + "路上行";
        lineRepository.deleteRoute(upRoute);
        String downRoute = line.getName() + "路下行";
        lineRepository.deleteRoute(downRoute);
    } else {
        String route = line.getName() + "路";
        lineRepository.deleteRoute(route);
    }
    stationRepository.deleteAllIsolatedStations();
}
```

首先，删除这条路线（next.line=target），随后删除所有孤立的节点
这里就不演示了