Neo4j
the graph database

Yuquan Lan
2021/10/28

Content

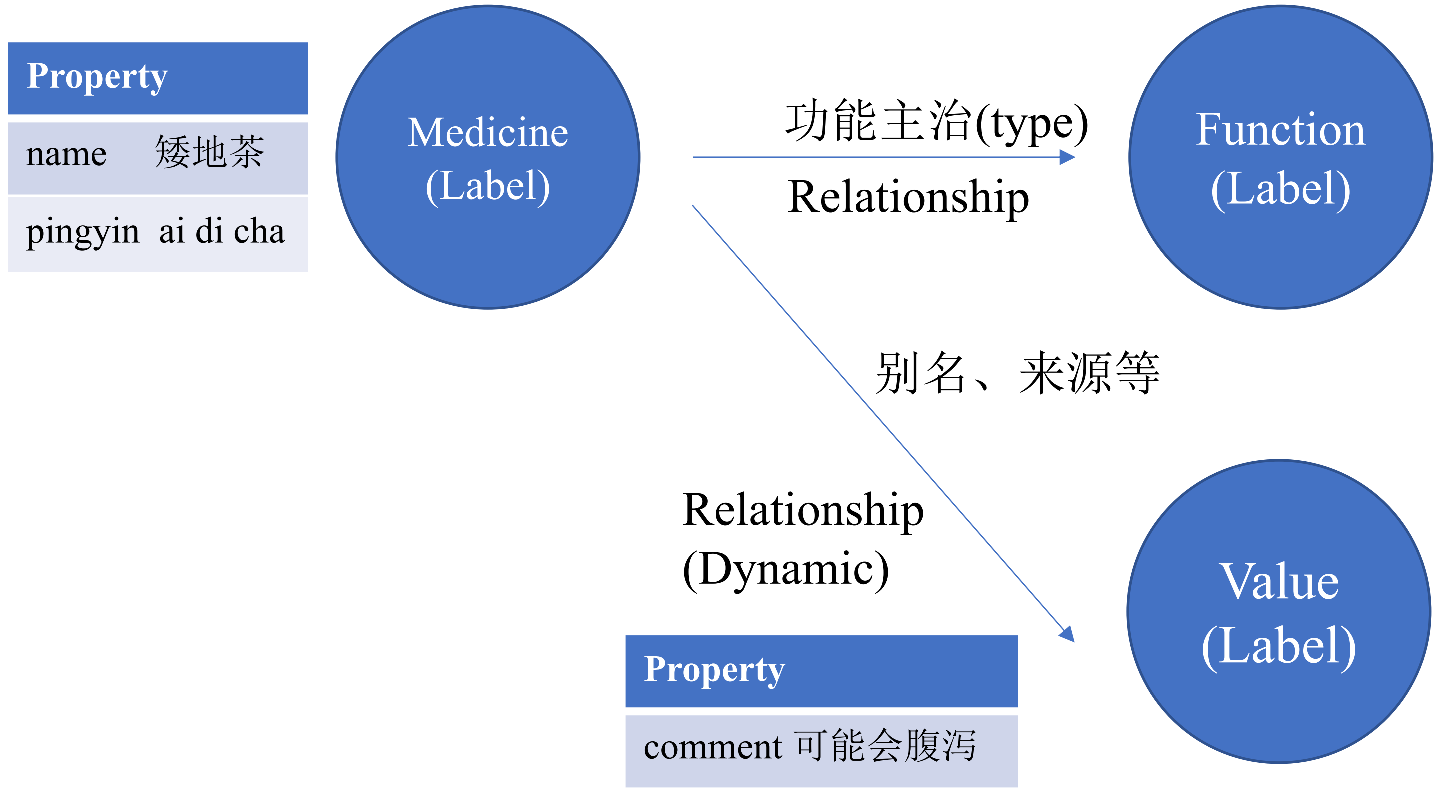
Installation

- Recommend Version: 3.5.28
- Download from https://neo4j.com/download-center/#community
- (Community is enough)
- bin/neo4j console
- http://localhost:7474/

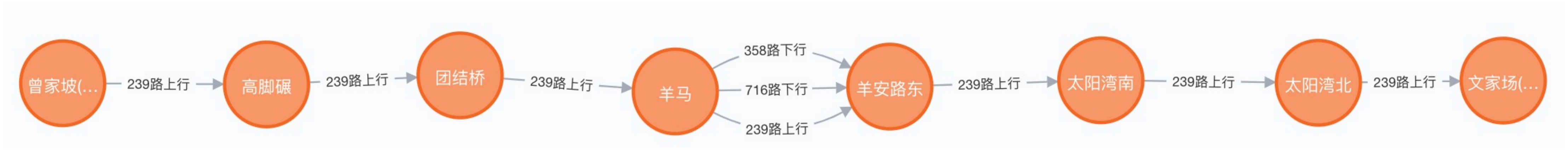
Core Concepts

- Node (Label)
- Relationship (Type)
- Properties
- Path

Core Concepts



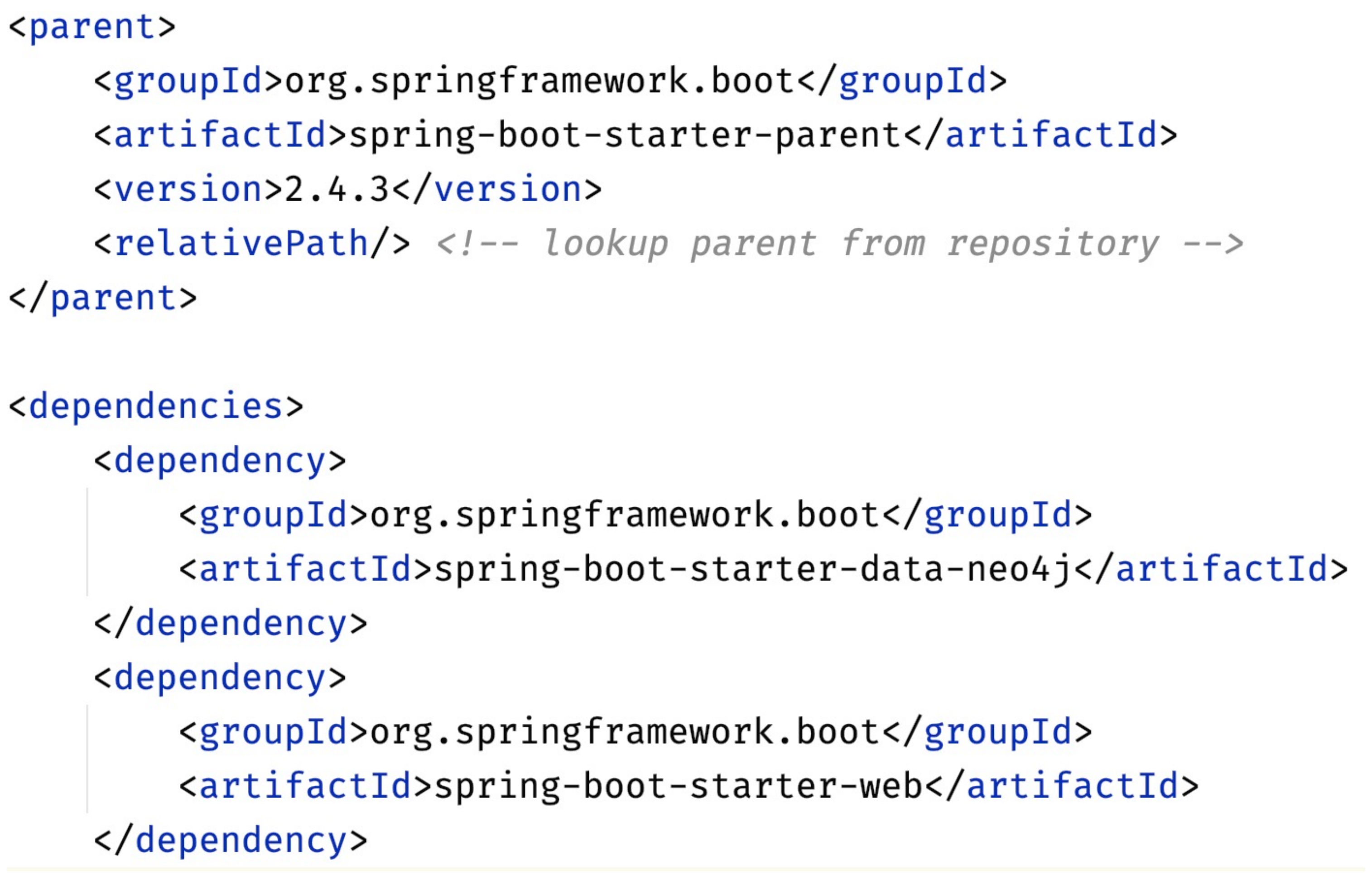
Core Concepts



Path

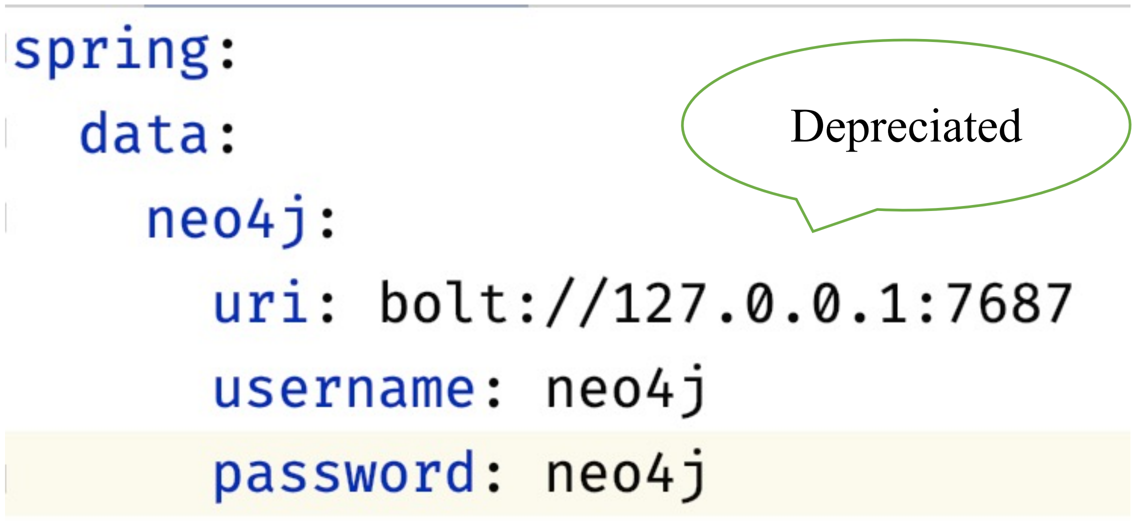
```
(Node)-[Relation]→(Node)-[Relation]→ ... (Node)
```

SpringDataNeo4j



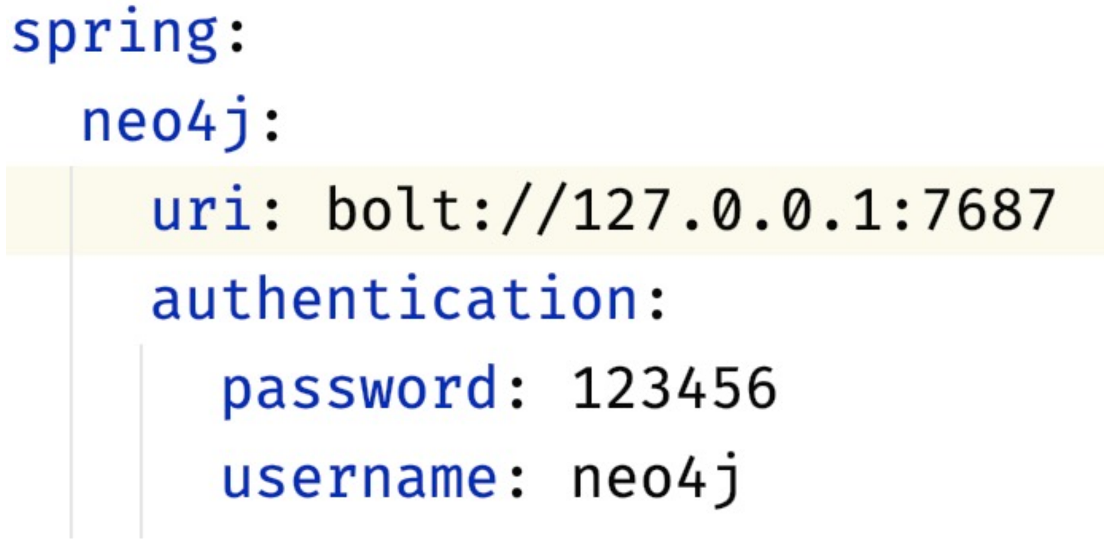
```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.3</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-neo4j</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

SpringDataNeo4j



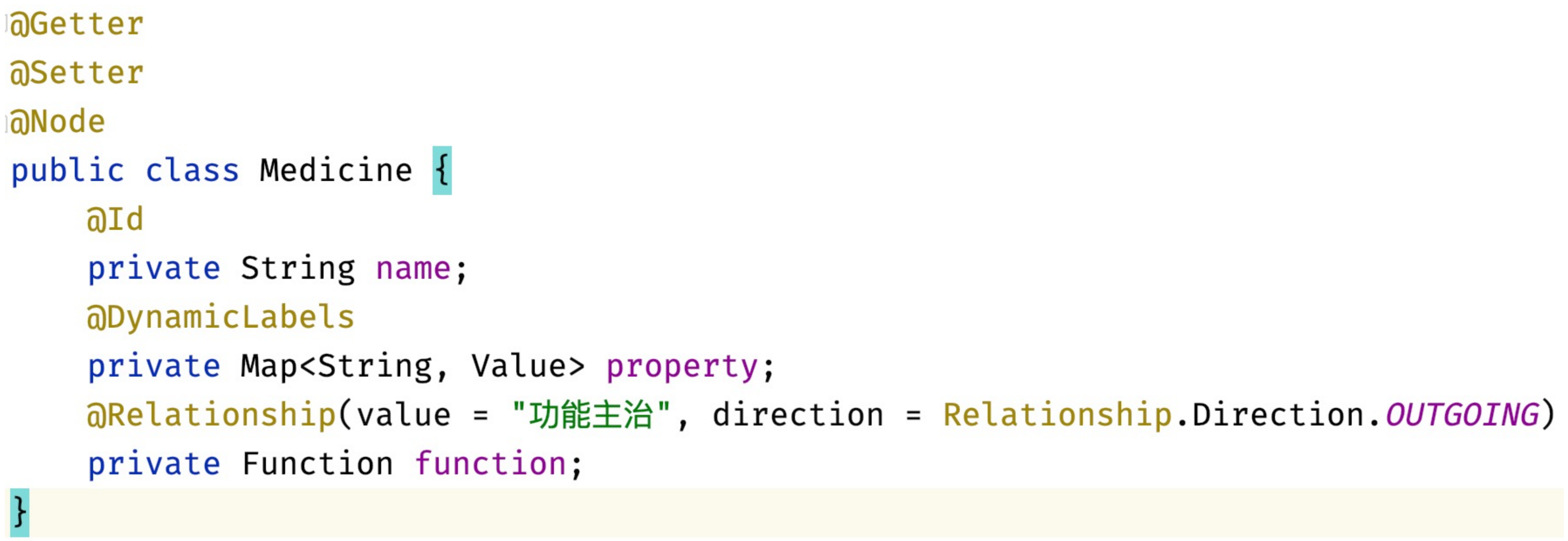
```yaml
spring:
  data:
    neo4j:
      uri: bolt://127.0.0.1:7687
      username: neo4j
      password: neo4j
```

Depreciated



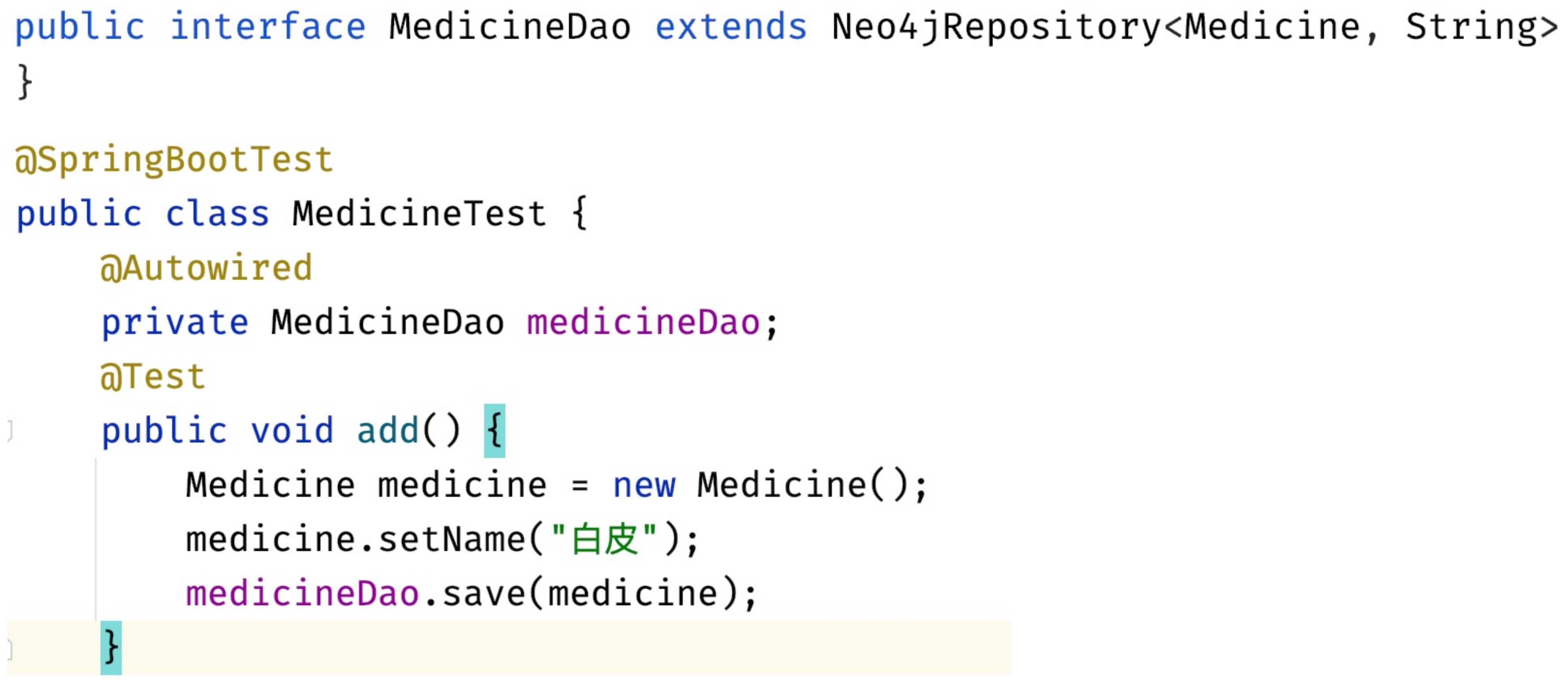
```yaml
spring:
  neo4j:
    uri: bolt://127.0.0.1:7687
    authentication:
      password: 123456
      username: neo4j
```

SpringDataNeo4j



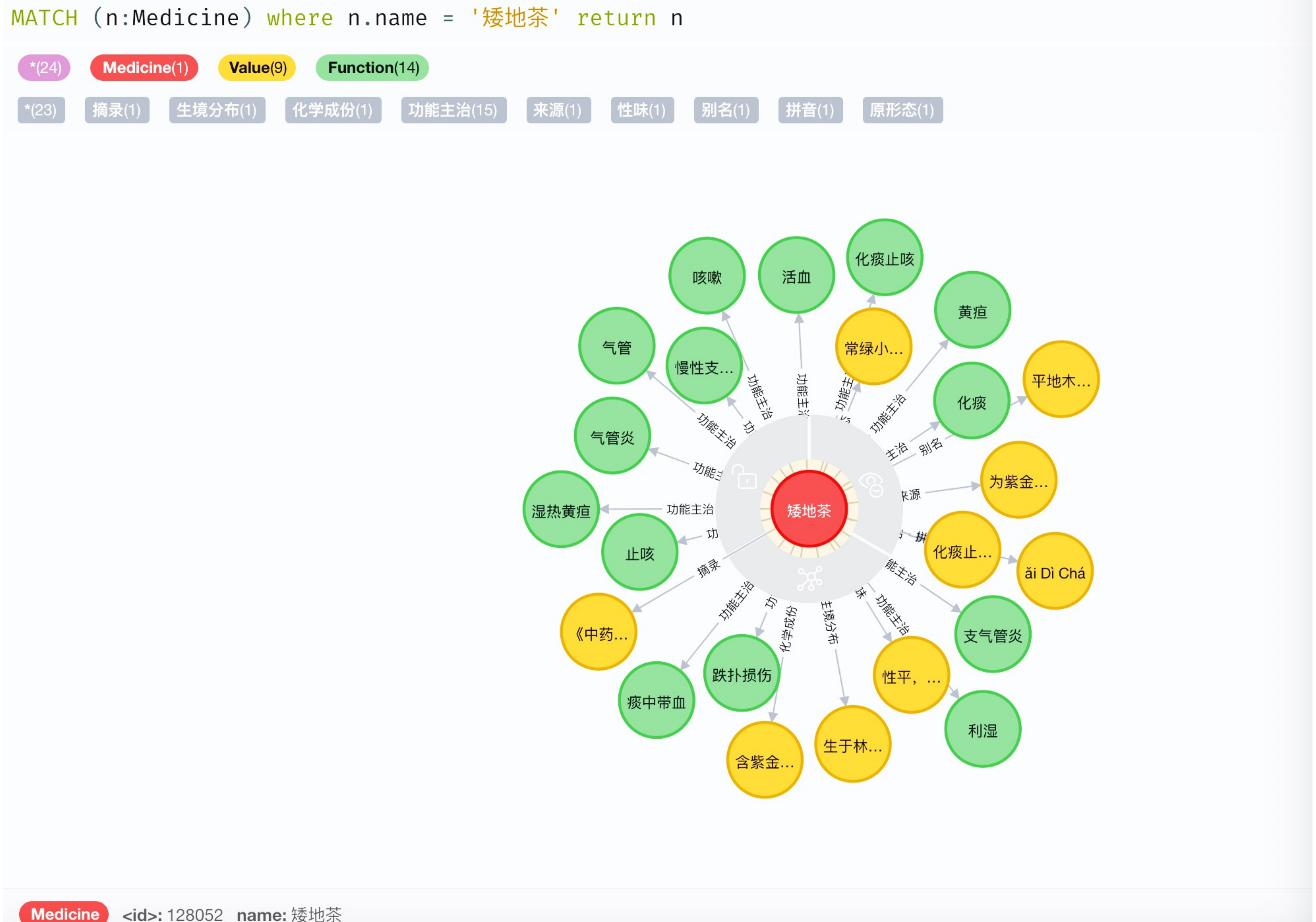
```java
@Getter
@Setter
@Node
public class Medicine {
    @Id
    private String name;
    @DynamicLabels
    private Map<String, Value> property;
    @Relationship(value = "功能主治", direction = Relationship.Direction.OUTGOING)
    private Function function;
}
```

SpringDataNeo4j



```
public interface MedicineDao extends Neo4jRepository<Medicine, String>
}

@SpringBootTest
public class MedicineTest {
    @Autowired
    private MedicineDao medicineDao;
    @Test
    public void add() {
        Medicine medicine = new Medicine();
        medicine.setName("白皮");
        medicineDao.save(medicine);
    }
```

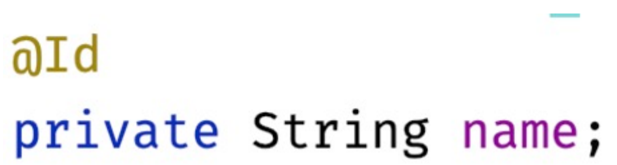
```
MATCH (n:Medicine) where n.name = '矮地茶' return n
```



Medicine <id>: 128052 name: 矮地茶

SpringDataNeo4j(@Id)

- @Id标识的字段
  - 可以使用findById()查询
  - 调用save()方法时，若Id字段已存在，则会更新；不存在，则插入。



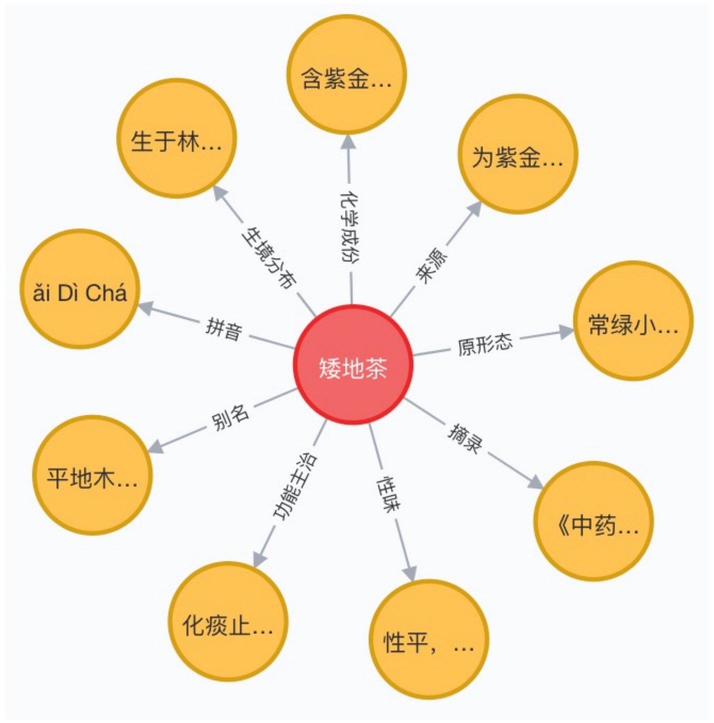
```
@Id
private String name;
```

- 注意：Cypher语句使用id(n)返回的是<id>的值。
  - match (n:Medicine) where n.name = ‘矮地茶’return id(n)
  - 返回：128052

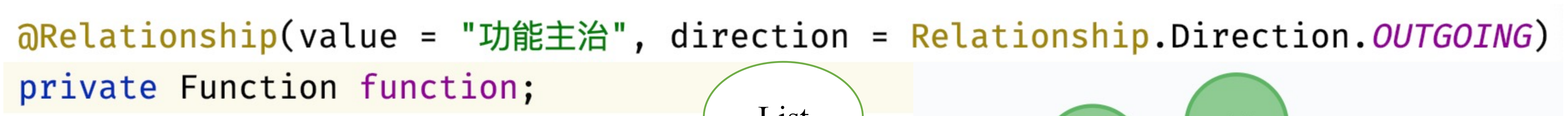
SpringDataNeo4j(@DynamicLabels)

```java
Medicine medicine = new Medicine();
medicine.setName("矮地茶");
Map<String, Value> propertyMap = new HashMap<>();
propertyMap.put("别名", new Value("平地木、老勿大、不出林、叶底珠"));
medicine.setProperty(propertyMap);
medicineDao.save(medicine);
```

```java
@DynamicLabels
private Map<String, Value> property;
```



SpringDataNeo4j(@Relationship)



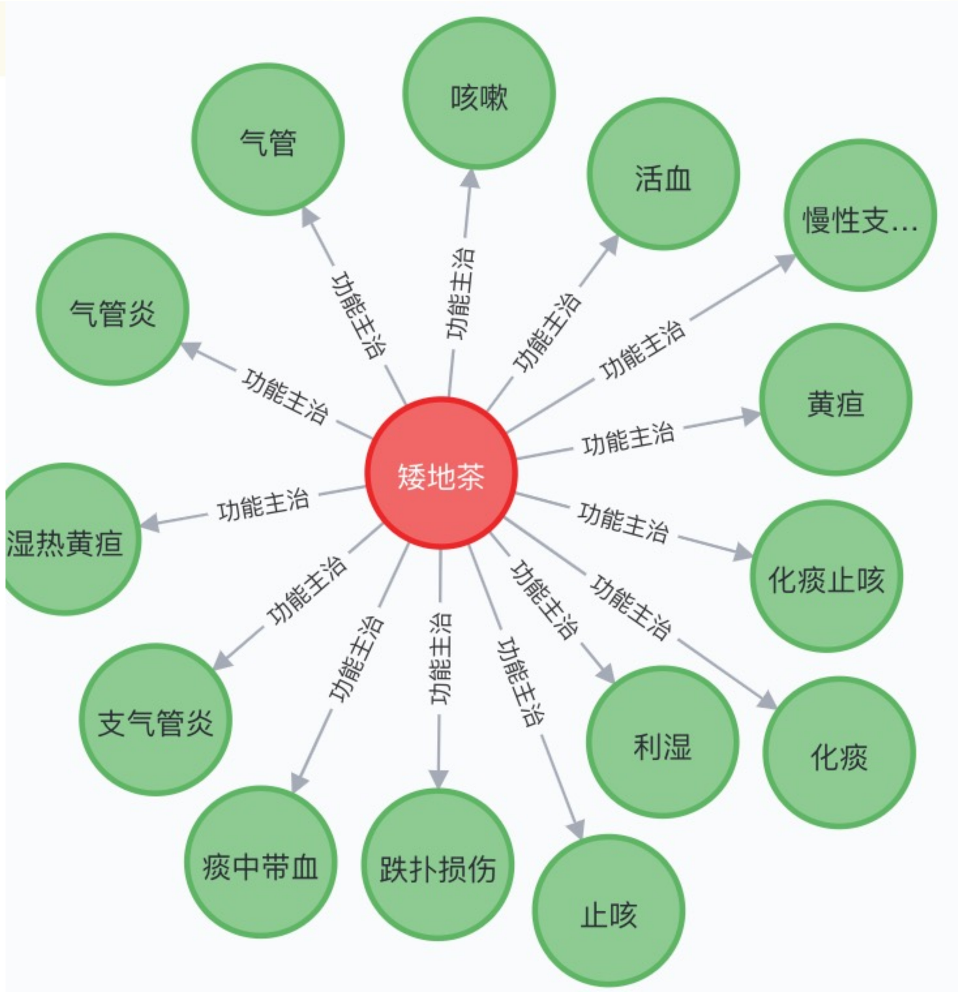
```java
@Relationship(value = "功能主治", direction = Relationship.Direction.OUTGOING)
private Function function;
```

List

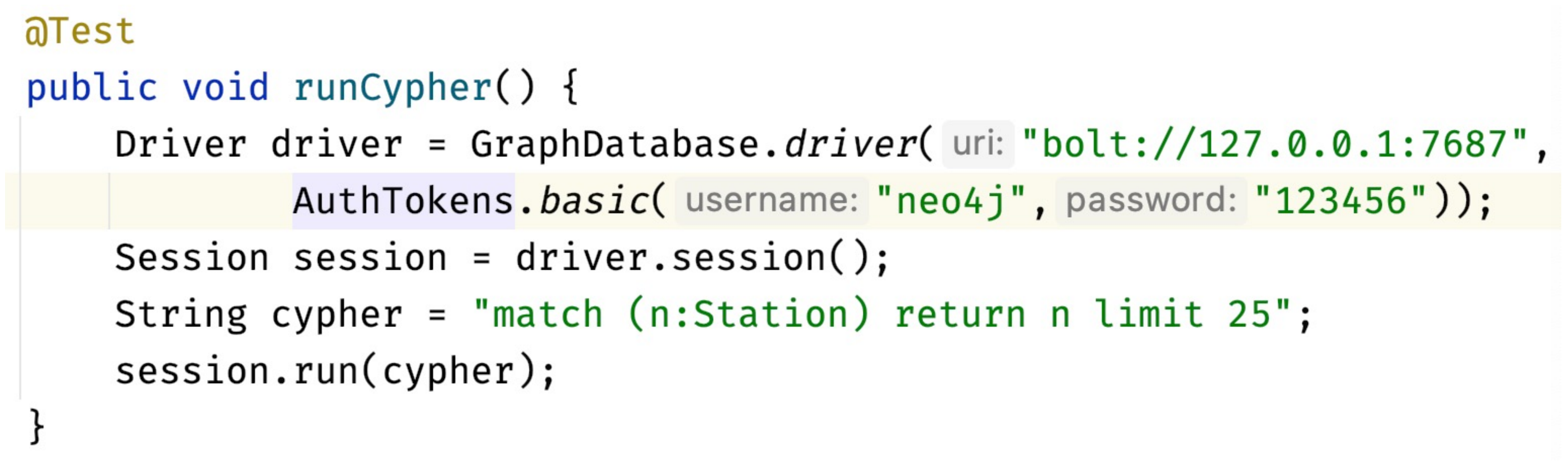
Medicine ⟶ Function (OUTGOING)
Medicine ⟵ Function (INCOMING)

One to One
@Relationship(value = “xxx”)
private Function function

One to Many
@Relationship(value = “xxx”)
private List<Function> function;

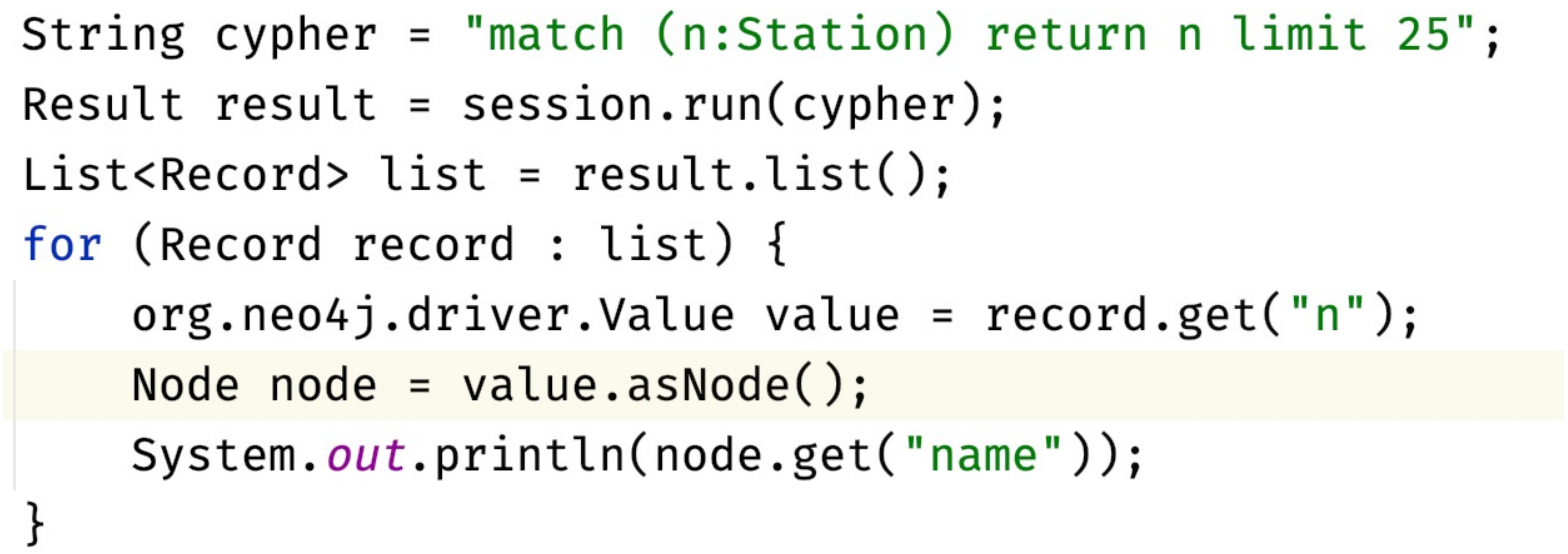


SpringDataNeo4j(Execute Cypher)



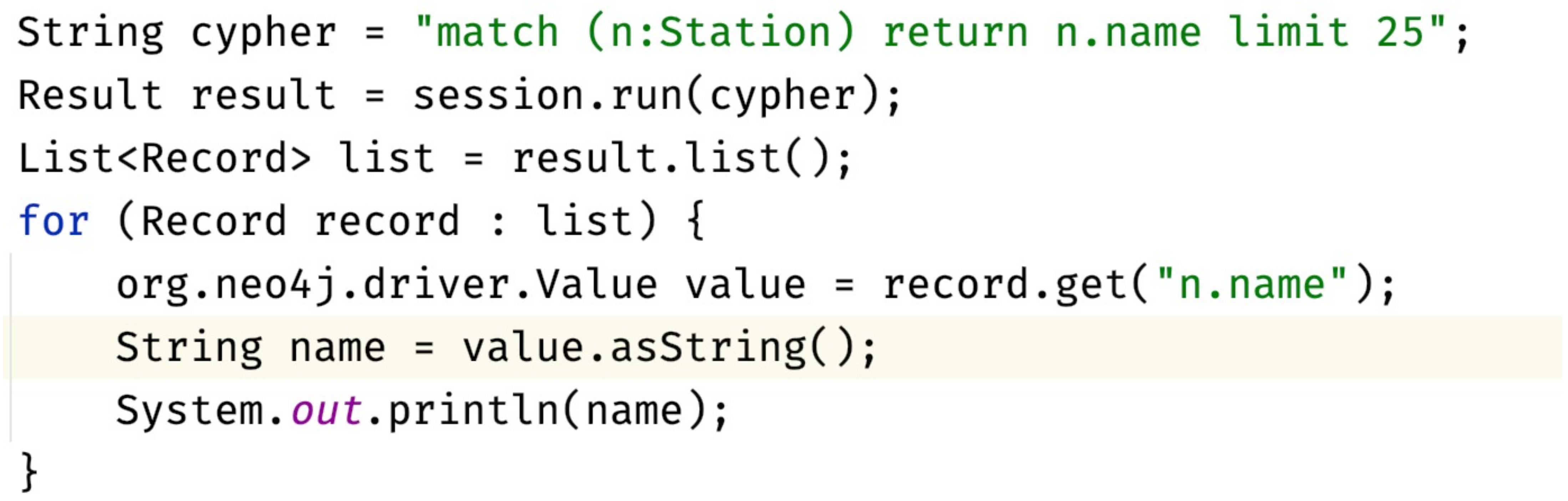
```
@Test
public void runCypher() {
    Driver driver = GraphDatabase.driver( uri: "bolt://127.0.0.1:7687",
            AuthTokens.basic( username: "neo4j", password: "123456"));
    Session session = driver.session();
    String cypher = "match (n:Station) return n limit 25";
    session.run(cypher);
}
```

SpringDataNeo4j(Parse Result of Node List)



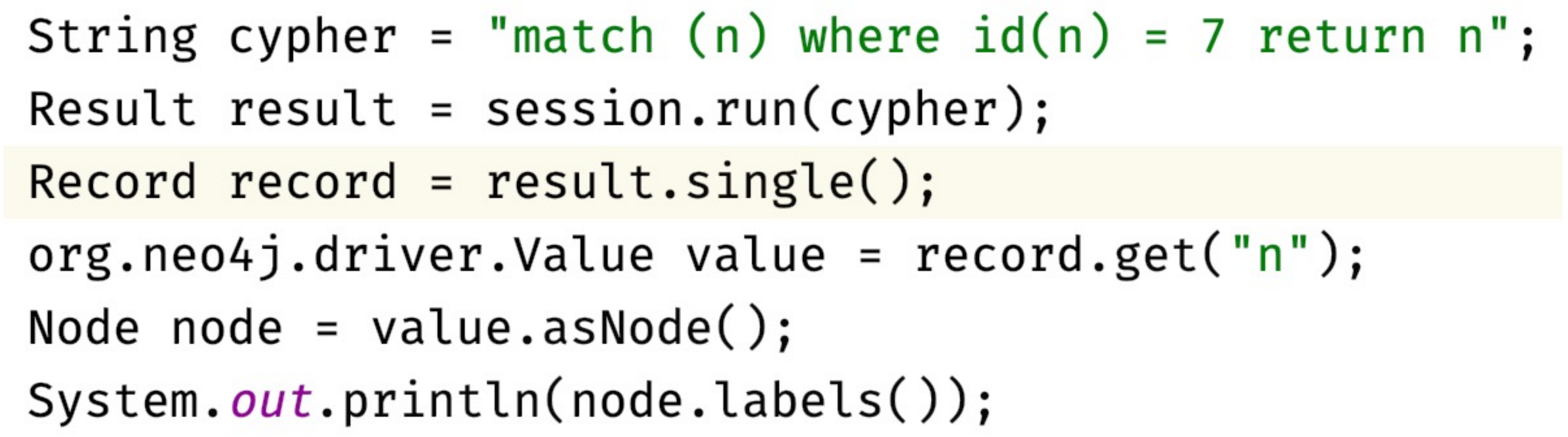
```java
String cypher = "match (n:Station) return n limit 25";
Result result = session.run(cypher);
List<Record> list = result.list();
for (Record record : list) {
    org.neo4j.driver.Value value = record.get("n");
    Node node = value.asNode();
    System.out.println(node.get("name"));
}
```

SpringDataNeo4j(Parse Result of Property)



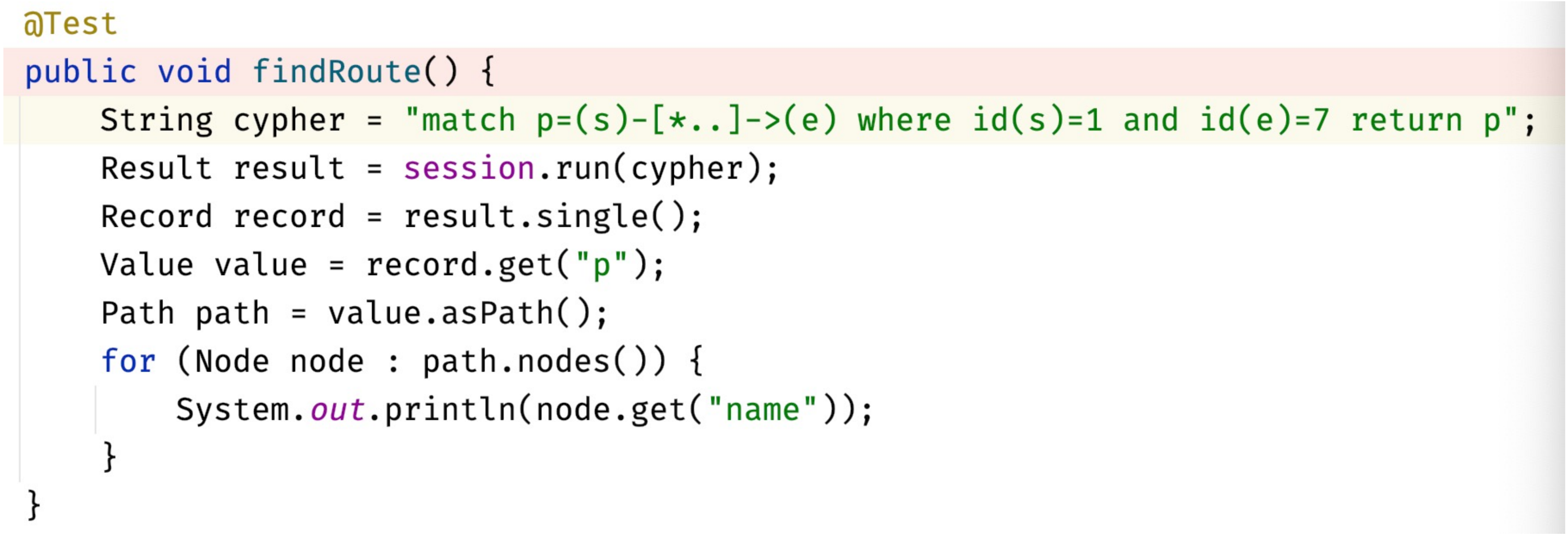
```java
String cypher = "match (n:Station) return n.name limit 25";
Result result = session.run(cypher);
List<Record> list = result.list();
for (Record record : list) {
    org.neo4j.driver.Value value = record.get("n.name");
    String name = value.asString();
    System.out.println(name);
}
```

SpringDataNeo4j(Parse Result of Node)



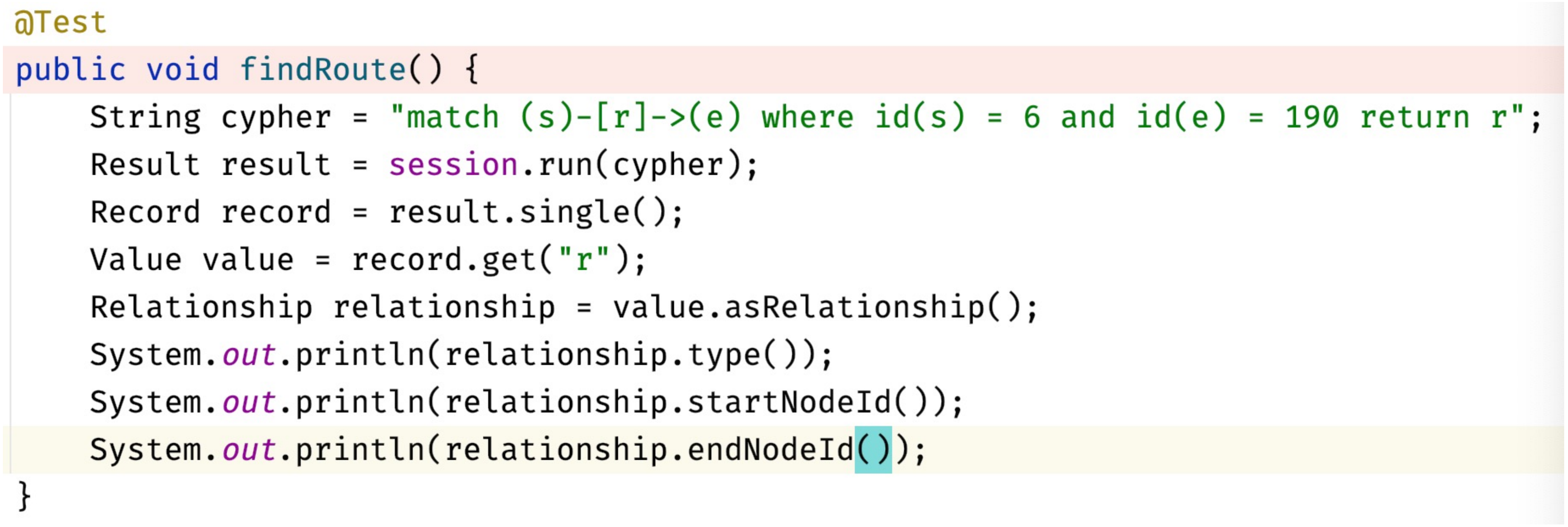
```java
String cypher = "match (n) where id(n) = 7 return n";
Result result = session.run(cypher);
Record record = result.single();
org.neo4j.driver.Value value = record.get("n");
Node node = value.asNode();
System.out.println(node.labels());
```

SpringDataNeo4j(Parse Result of Path)



```
@Test
public void findRoute() {
    String cypher = "match p=(s)-[*..]->(e) where id(s)=1 and id(e)=7 return p";
    Result result = session.run(cypher);
    Record record = result.single();
    Value value = record.get("p");
    Path path = value.asPath();
    for (Node node : path.nodes()) {
        System.out.println(node.get("name"));
    }
}
```

SpringDataNeo4j(Parse Result of Relation)



```
@Test
public void findRoute() {
    String cypher = "match (s)-[r]->(e) where id(s) = 6 and id(e) = 190 return r";
    Result result = session.run(cypher);
    Record record = result.single();
    Value value = record.get("r");
    Relationship relationship = value.asRelationship();
    System.out.println(relationship.type());
    System.out.println(relationship.startNodeId());
    System.out.println(relationship.endNodeId());
}
```

Cypher(Match Node)

- match (n) return n
- match (n:Station) return n
- match (n:Station) return n.name
- match (n:Station {name:'Global Center'}) return n
- match (n:Station) where n.name = 'Global Center' return n
- match (n:Station) where id(n) = 72135
- match (n:Person) where n.id in ['226','227'] return n
- match (n:Station),(m:Station) return n,m

Cypher(Match Relation)

- match (n:Station)-[]->(m:Station) where id(n)=1 return m
- match (n:Station)-[]-(m:Station) where id(n)=1 return m
- match (n:Station)-[r]-() where id(n)=1 return type(r)

return r?

Out of order!!!

- match (n)-[r:`1路上行`]-() return n
- match (n)-[r:`1路上行`|`2路上行`]-() return n
- match (n:Station) where not exists ((n)-[]->()) return n

Cypher(Match Path)

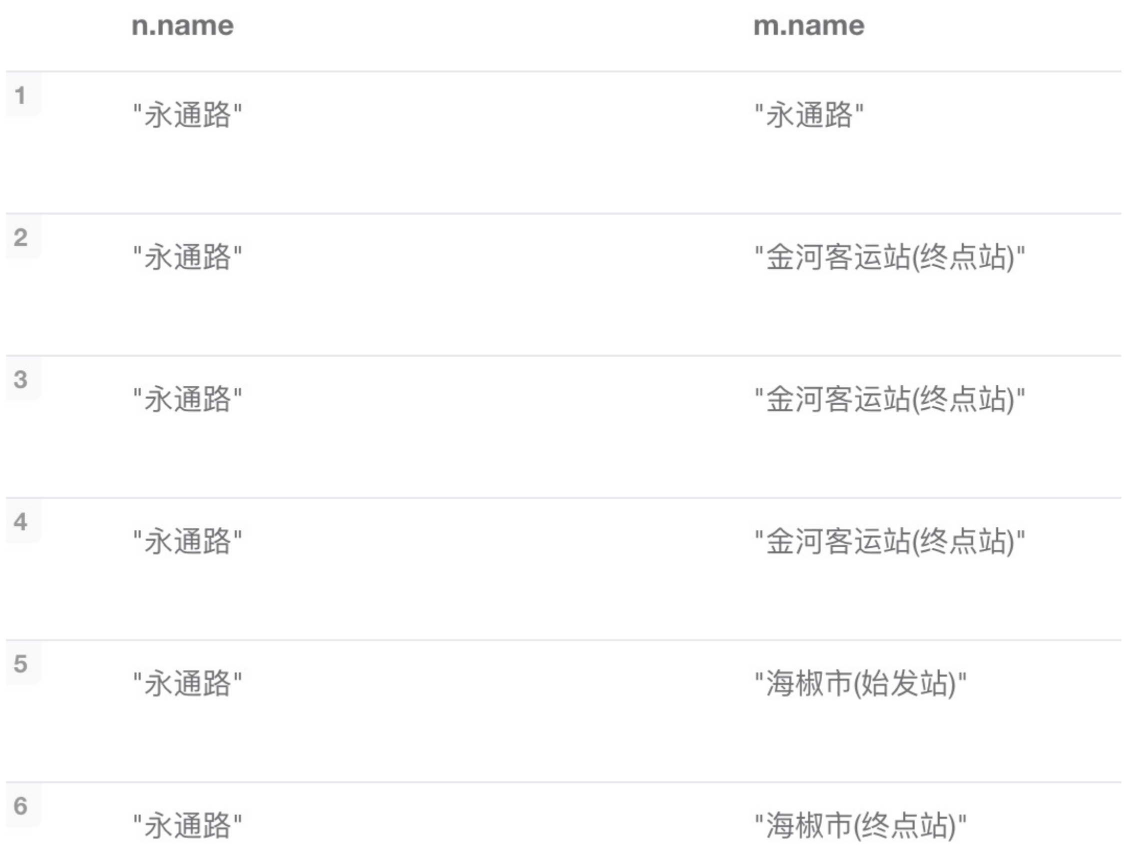
- match p=()-[r:`1路上行`]->() return p
- match p=(n)-[r:`101路`*..]->(m) where id(n)=149 and id(m)=413 return p
- match p=(n:Medicine)-[r]-()-[]-(m:Medicine) where n.name = '矮地茶' and type(r) = '功能主治' return p
- match p = shortestPath((n:Person {id:'mom'})-[*..3]-(b:Person {id:'grandmom'})) return p
- match p = allshortestPath((n:Person {id:'mom'})-[*..3]-(b:Person {id:'grandmom'})) return p

Cypher(Return)

- Return nodes, relationships, property
  - match (n:Station) return n
  - match (:Station)-[r]-(:Station) return r
  - match (n:Station) return n.name
- Alias
  - match (n:Station) return n.name as name
- Unique
  - match (n:Station) return distinct n.name
- Cartesian Product
  - match (n:Station), (m:Station) return n, m

Cypher(Return(Cartesian Product))

- match (n:Station), (m:Station) return n.name, m.name limit 25

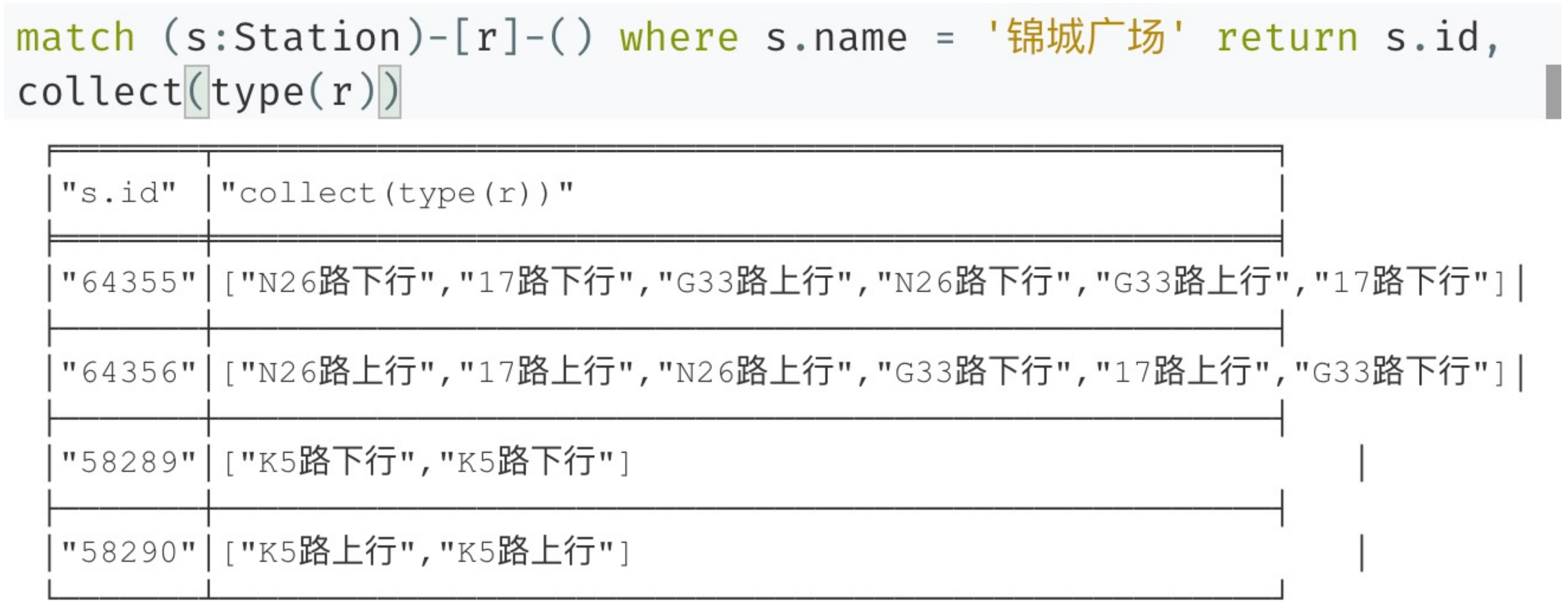


	n.name	m.name
1	"永通路"	"永通路"
2	"永通路"	"金河客运站(终点站)"
3	"永通路"	"金河客运站(终点站)"
4	"永通路"	"金河客运站(终点站)"
5	"永通路"	"海椒市(始发站)"
6	"永通路"	"海椒市(终点站)"

Cypher(Return(Group By))

- Neo4j doesn’t provide ' group by ' operation, but ' return ' is enough!

- match (s:Station)-[r]-() where s.name = ' xxx' return s.id, collect(type(r))
  - one (s) maps multiple [r]



```
match (s:Station)-[r]-() where s.name = '锦城广场' return s.id,
collect(type(r))
```

```
╒═══════╤════════════════════════════════════════════════════════════╕
│"s.id" │"collect(type(r))"                                          │
╞═══════╪════════════════════════════════════════════════════════════╡
│"64355"│["N26路下行","17路下行","G33路上行","N26路下行","G33路上行","17路下行"]│
├───────┼────────────────────────────────────────────────────────────┤
│"64356"│["N26路上行","17路上行","N26路上行","G33路下行","17路上行","G33路下行"]│
├───────┼────────────────────────────────────────────────────────────┤
│"58289"│["K5路下行","K5路下行"]                                     │
├───────┼────────────────────────────────────────────────────────────┤
│"58290"│["K5路上行","K5路上行"]                                     │
└───────┴────────────────────────────────────────────────────────────┘
```

Cypher(Create & Merge)

- create (p1:Person{name:'1'})
- match (s:Station),(s1:Station) where id(s)=1 and id(s1)=2 create (s)-[:Next{line:'1路上行'}]->(s1)
- match (n:Person {id:‘226’}),(m:Person {id:‘227‘}) merge (n)-[:boss]->(m) return n, m
- Using SpringDataNeo4j to save is recommended.

Cypher(Set)

- match (n:Person {id:'226'}) set n.name = 'onion’

- match(n:old) remove n:old set n:new

- match(n)-[r:old]->(m) create(n)-[r2:new]->(m) set r2=r with r delete r

- match (u:user),(n:note)
where u.email = " xxx@qq.com" and n.noteId = "1"
merge (u)-[v:view]-(n)
on create set v.viewTimes = 0
on match set v.viewTimes = v.viewTimes + 1

Cypher(Delete)

- match (t:Teacher)-[r:teach]->(s:Student) delete t, r, s
- match (r) detach delete r
- match (n) optional match (n)-[r]-() delete n, r
- match p=()-[:friend]-() delete p
- match (n:user)-[s:star]->(m:user) create (n)-[s2:unstar]->(m) set s2 = s with s delete s

Cypher(Remove)

- match(n:old) remove n:old set n:new

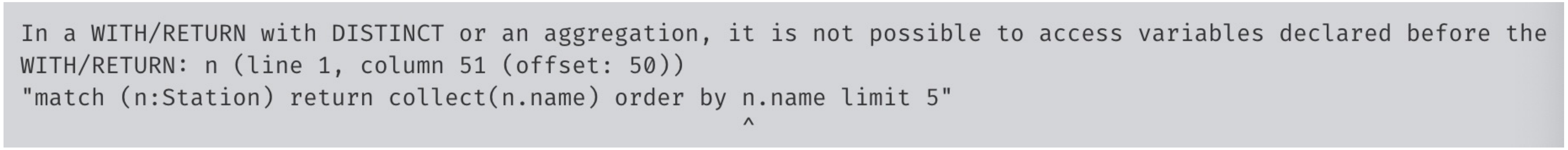
- match (n:Person {id:'226'}) remove n.age

Cypher(With)

- Using WITH to manipulate the output before it is passed on to the following query parts

- What’s the difference of these two cypher?
  - match (n:Station) with n order by n.name desc limit 5 return n.name
  - match (n:Station) return n.name order by(n.name) desc limit 5

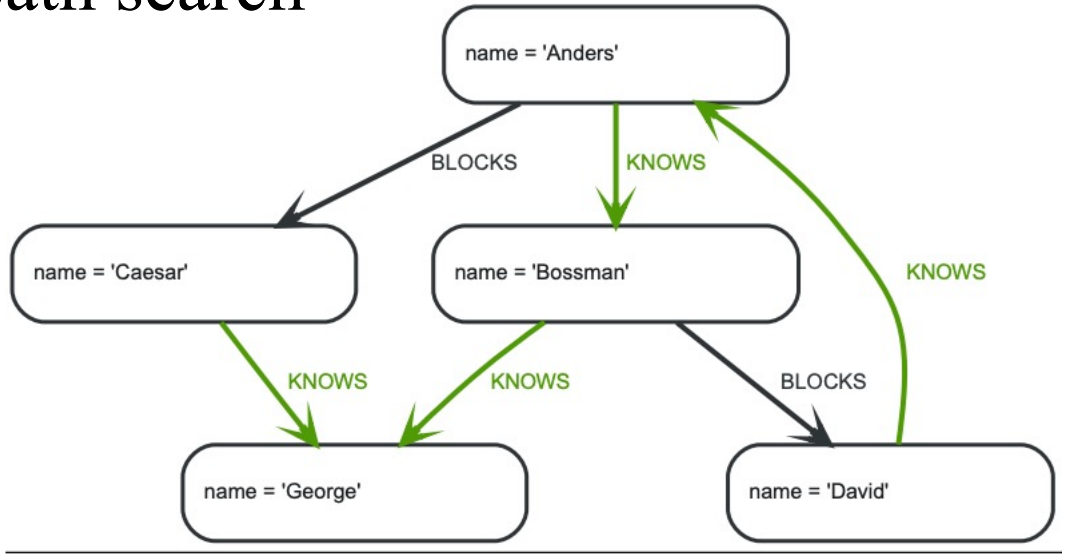
  - match (n:Station) with n order by n.name desc limit 5 return collect(n.name)



```
In a WITH/RETURN with DISTINCT or an aggregation, it is not possible to access variables declared before the
WITH/RETURN: n (line 1, column 51 (offset: 50))
"match (n:Station) return collect(n.name) order by n.name limit 5"
                                                  ^
```

Cypher(With)

- Using WITH to limit branching of a path search



- match (n {name: 'Anders'})--(m)
with m order by m.name desc limit 1
match (m)--(o) return o.name

- "Anders" "Bossman"

Cypher(Unwind)

- UNWIND expands a list into a sequence of rows

- unwind [1, 2, 3, null] as x return x, 'val' as y

x	y
1	'val'
2	'val'
3	'val'
<null>	'val'

Cypher(Aggregation Function)

- match (n:Person) return n order by n.age desc skip 2 limit 25

- match (n:Person) where n.age > 20 return n.id,n.age

union all match (n:Person) where n.age < 18 and n.name is not null

return n.id, n.age

- match (n:Person) return count(*)

- match (n:Person) return avg(n.age)

Cypher(Reference)

- https://neo4j.com/docs/pdf/neo4j-cypher-manual-4.3.pdf

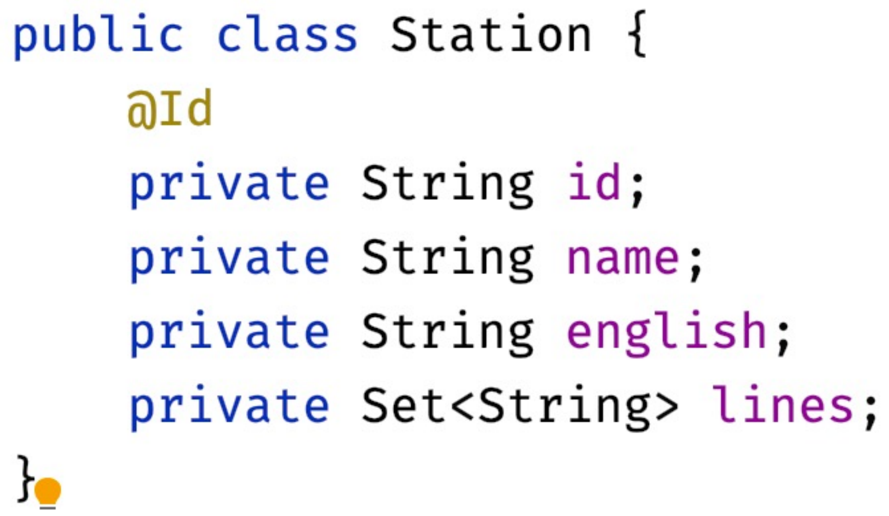
Schema Design

- How to construct the route of a specific line?

  - Design1:
    - (:Station)-[:Next{line: 'xxx'}]->(:Station)
    - line number is the property of relation ‘Next’

  - Design2:
    - (:Station)-[:`xxx`]->(:Station)
    - line number is the relation type itself
    - set runtime between two station as the relation property

Schema Design



```
public class Station {
    @Id
    private String id;
    private String name;
    private String english;
    private Set<String> lines;
}
```

- 需求：查询锦城广场站停靠的所有线路。
- Design1:
  - match (s:Station)-[r]-() where s.name = '锦城广场' return s.id, collect(type(r))
- Design2:
  - create a field to store all the lines that stop at the station
  - match (s:Station) where s.name = '锦城广场' return s.id, s.lines

Project(Grading scale)

选需求分实现

需求分不是最终得分

根据需求分确定得分上限比例

	60%	70%	80%	90%	100%
2-3人组	20分	22分	24分	28分	32分
4人组	28分	30分	32分	36分	40分

Project(Requirements)

Total scores: 54

ID	Score	ID	Score
1 线路基本信息	1	11 统计站点数量	4(2+2)
2 线路站点信息	2	12 统计路线类型	2
3 站点停靠线路	2	13 查询重复站点	2
4 起止沿线站点	3(2+1)	14 查询线路换乘	4(2/3/4)
5 最短路径	6(2/4/6)	15 统计站台连接	2
6 直达路线判断	2	16 统计路线站点	2
7 线路班次信息	2	17 统计运行时间	2
8 站点某时线路	2	18 计算重复系数	4
9 站点某时某线	2	19 线路创建	4(2/4)
10 统计停靠路线	2	20 线路删除更新	4(2+2)

Project(Test Cases)

1.查询30路公交的基本信息。 (1分)

变更：2分变更为1分

```
{
        "route": "燎原-北路湾公交站",
        "onewayTime": "约49分",
        "directional": true,
        "kilometer": "12.0",
        "name": "30",
        "runtime": "6:30-22:30",
        "interval": 8,
        "type": "干线”
 }
```

Project(Test Cases)

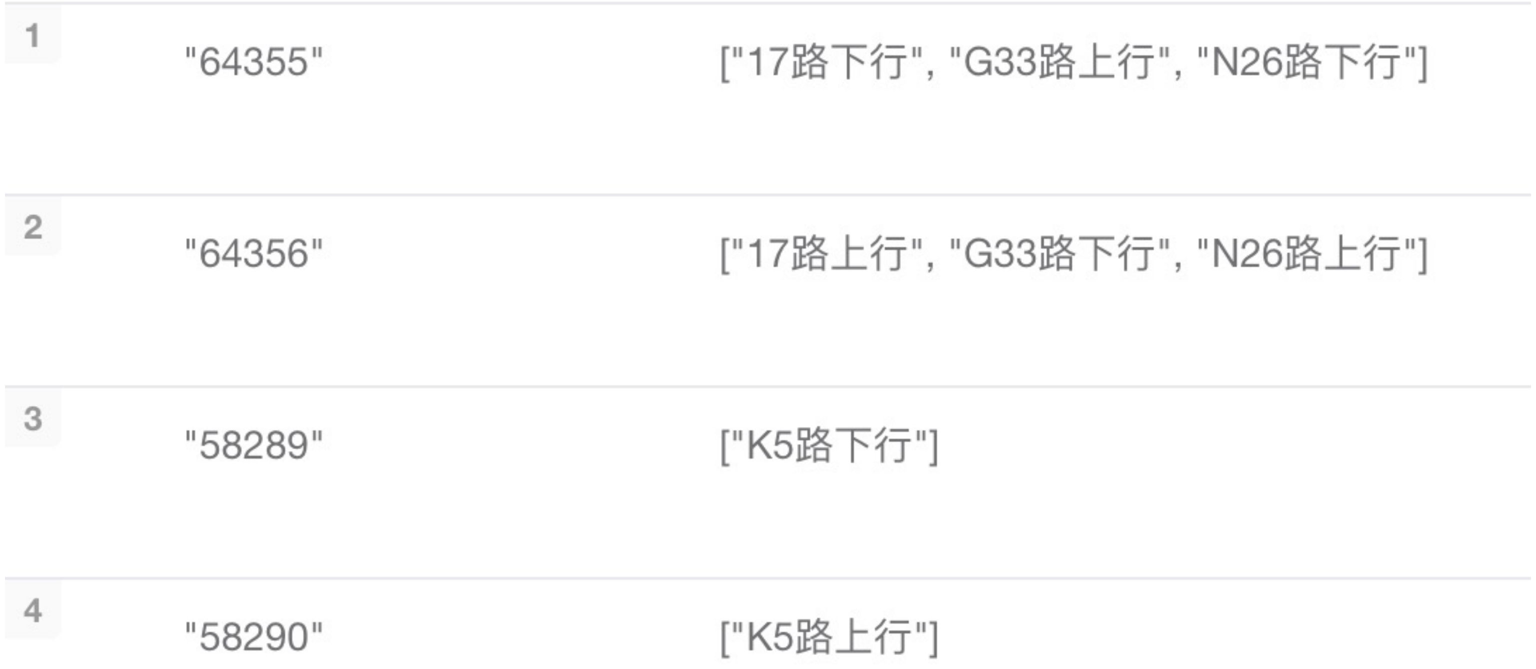
2.查询2路上行的全部站点信息。(方向性、区分上下行、顺序性，2分)

```
[
        {
                "name" : "兴义镇(始发站) ",
                "english":"XingYiZhen",
               "id":"7542"
        },
        {
               "english":"YongSheng",
               "name":"永盛(始发站)",
               "id":"7527 "
        },
        …
] 共28个站点
```

Project(Test Cases)

3.查询锦城广场站停靠的所有线路。 (同名站点按ID分组，2分)

变更：原需求3（查询全部公交的全部站点信息，按照字典序排序）已废除。



1	"64355"	["17路下行", "G33路上行", "N26路下行"]
2	"64356"	["17路上行", "G33路下行", "N26路上行"]
3	"58289"	["K5路下行"]
4	"58290"	["K5路上行"]

Project(Test Cases)

4.查询乘坐10路从大悦城到小吃街，线路的运行方向(上行或下行)、沿路站点、运行时

变更：2分变更为3分

运行方向：上行(1分)

沿途站点：大悦城、肖家沟、华通商场、东林南路、东林小区、东林路、小吃街(1分

运行时长：(1分)

Project(Test Cases)

5.查询从id为16115的站台(红瓦寺)到id为14768的站台（动物园）的最短路径。 (6分)

变更：4分变更为6分

评分标准：

- 使用id进行最短路径查询。 (2分)
- 使用name进行最短路径查询。(4分)
- 考虑更多场景的路径查询，例如最少换乘、最短运行时间等。 (6分)

Project(Test Cases)

6.查询从荷花池到环球中心(始发站)是否存在直达线路。

若存在，返回线路方向；不存在则提示用户。　(2分)

返回：N12路上行、N12路下行

Project(Test Cases)

7.查询N8路（环线）的全部班次信息。(2分)

Project(Test Cases)

8.查询上午08:37分新华书店站10分钟内即将停靠的线路，

并显示几分钟后某线路即将到站。

若查询到多个同名站点，需要按照id进行分组。 (2分)

Project(Test Cases)

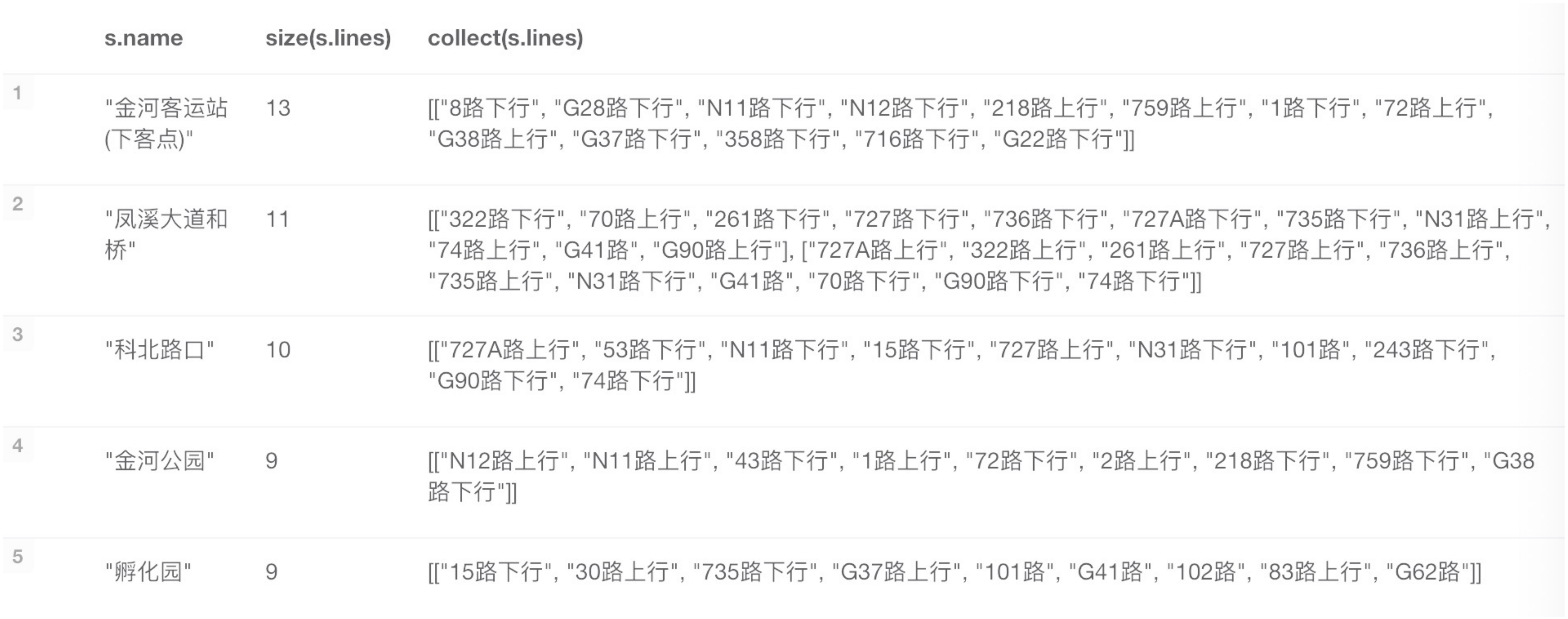
9.查询上午10:32分地铁万盛站82路的最近的3趟班次信息。

可能当前时间下暂无班次。(2分)

Project(Test Cases)

10.统计停靠路线最多的站点(按照id统计)并排序，显示前15个。(2分)

变更：显示前15个即可。



	s.name	size(s.lines)	collect(s.lines)
1	"金河客运站(下客点)"	13	[["8路下行", "G28路下行", "N11路下行", "N12路下行", "218路上行", "759路上行", "1路下行", "72路上行", "G38路上行", "G37路下行", "358路下行", "716路下行", "G22路下行"]]
2	"凤溪大道和桥"	11	[["322路下行", "70路上行", "261路下行", "727路下行", "736路下行", "727A路下行", "735路下行", "N31路上行", "74路上行", "G41路", "G90路上行"], ["727A路上行", "322路上行", "261路上行", "727路上行", "736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"]]
3	"科北路口"	10	[["727A路上行", "53路下行", "N11路下行", "15路下行", "727路上行", "N31路下行", "101路", "243路下行", "G90路下行", "74路下行"]]
4	"金河公园"	9	[["N12路上行", "N11路上行", "43路下行", "1路上行", "72路下行", "2路上行", "218路下行", "759路下行", "G38路下行"]]
5	"孵化园"	9	[["15路下行", "30路上行", "735路下行", "G37路上行", "101路", "G41路", "102路", "83路上行", "G62路"]]

Project(Test Cases)

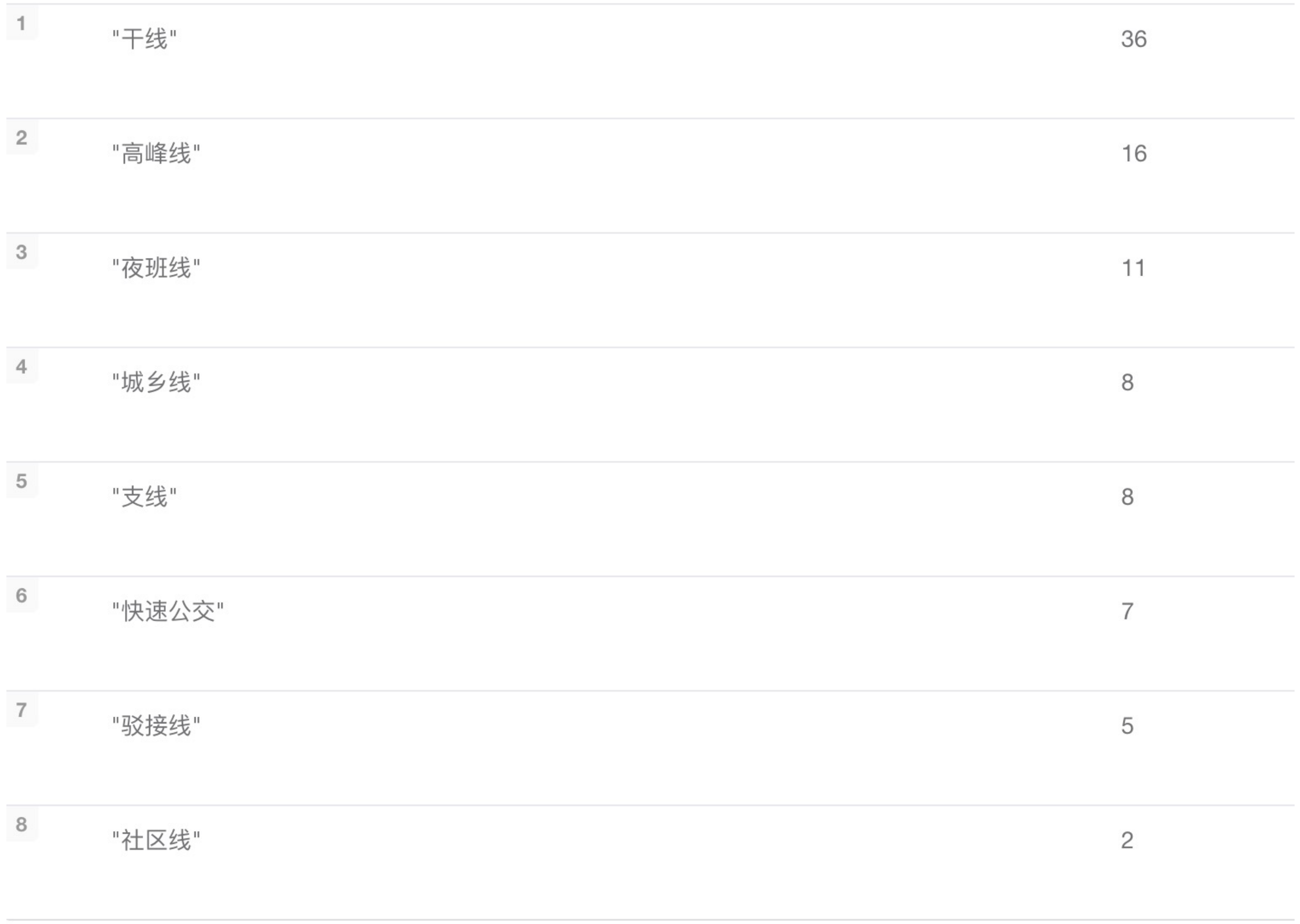
11.统计地铁站数量(以地铁开头)、起点站(末尾标识始发站)数量、终点站(末尾标识终点站)数量、单行站(比较上下行确定单行站)数量。并返回站点名，注意去重。 (4分)
变更：2分变更为4分。

评分标准：
2分：统计地铁站、起点站、终点站数量，并返回站点名
4分：额外统计单行站的数量

地铁站数量：20
起点站数量：59
终点站数量：59

Project(Test Cases)

12.分组统计常规公交(包括干线、支线、城乡线、驳接线、社区线)、快速公交(K字开头)、高峰公交(G字开头)、夜班公交(N字开头)的数量。(2分)



1	"干线"	36
2	"高峰线"	16
3	"夜班线"	11
4	"城乡线"	8
5	"支线"	8
6	"快速公交"	7
7	"驳接线"	5
8	"社区线"	2

Project(Test Cases)

13.查询15路上行和30路下行重复的站点名，并统计站点数。(2分)

变更：区分方向

重复站点名：金河市政府(始发站)、孵化园、北门立交东

Project(Test Cases)

14.查询261路上行一共有多少条可以换乘的线路，注意去重。
换乘线路数即261路上行停靠的所有站台停靠其他线路的数量的总和。(4分)
变更：2分变更为4分
提示：可以先求出沿线所有站点。
评分标准：
2分：统计可以换乘的线路数量　16条

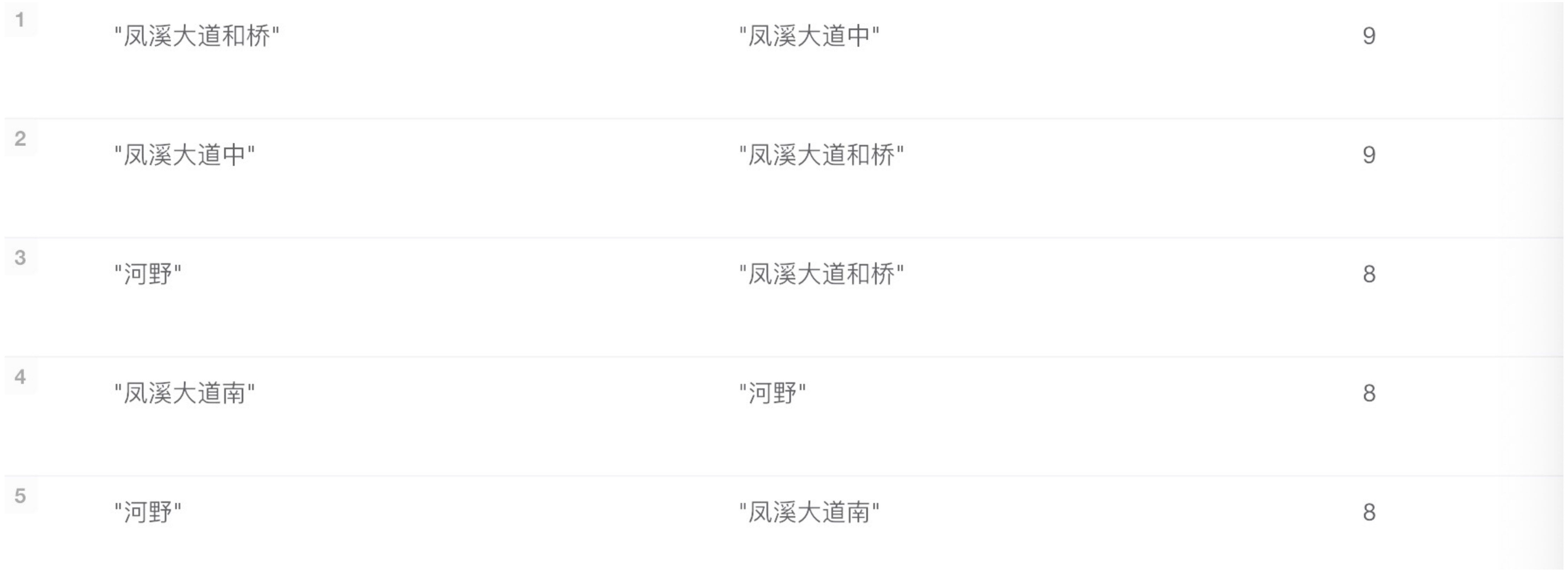
3分：统计可以换乘的线路名称
["261路上行", "208路上行", "30路下行", "N20路上行", "735路上行", "16路下行", "15路下行", "G41路", "G90路下行", "727A路上行", "74路下行", "70路下行", "N31路下行", "736路上行", "727路上行", "322路上行"]

4分：统计沿线每个站点可以换乘的线路
{
“培风路北”：[“30路下行”, “208路上行”],
“星空大道南”：[“30路下行”, “16路下行”, “735路上行”, “N20路上行”],
…
}

Project(Test Cases)

15.查询连接两个站台之间线路最多的两个站台并且按照降序排列，显示前15个。(2分)

变更：需求(根据线路可换乘数量降序排序，显示前15条)已经被废除。



1	"凤溪大道和桥"	"凤溪大道中"	9
2	"凤溪大道中"	"凤溪大道和桥"	9
3	"河野"	"凤溪大道和桥"	8
4	"凤溪大道南"	"河野"	8
5	"河野"	"凤溪大道南"	8

Project(Test Cases)

16.根据站点数量对线路进行排序，显示前15条。(2分)

变更： 显示前15条即可。

Project(Test Cases)

17.根据运行时间对线路进行排序(运行时间由班次数据计算而得)，显示前15条。(2分

变更：显示前15条即可。

Project(Test Cases)

18.计算某条线路之间的重复系数。 (4分)
对于一条线路，有若干个站台A、B、C...。
假如A与B之间有4条线路，则站台A与B 的非重复系数为 $\frac{1}{4}$。
一条线路的非重复系数则是线路沿线所有站台间的非重复系数的平均。

提示：使用Reduce函数(自学)

N12路上行：0.47
G95路上行：1.0 (G95路全程两个站，且只有一条线路直接相连，所以非重复系数为1
208路上行：0.88
30路上行：0.63

Project(Test Cases)

19.添加一条站点数不少于10的线路。 (4分)

变更：2分变更为4分

评分标准：

2分：添加线路，需要设置站点和路线信息。

4分：添加线路，并且加入班次信息。

Project(Test Cases)

20.线路的删除与更新。 (4分)

变更：减少了繁琐的操作。

2分：删除某条线路，若沿途站点为该线路独有，也需要删除该站点。

2分：将某条线路沿线的某个站点替换为另一个站点，不需要修改班次信息。返回新的路线沿途站点。

QA