实验 5　用 Cypher(SPARQL) 进行知识图谱查询

实验索引

5.1 实验目的

本次实验旨在：

- 熟悉 Cypher 的基本语法（如 MATCH、WHERE、RETURN 方法的使用）、模式匹配规则及图遍历操作，能够编写简单到复杂的图查询语句。
- 通过实际案例实现多跳查询、子图查询，理解 Cypher 在关联数据挖掘中的优势。
- 学习通过索引（如节点标签索引、关系类型索引）、查询重写（如避免全图扫描）及执行计划分析（EXPLAIN/PROFILE）提升查询性能。

5.2 实验平台

本次实验主要使用的工具和库包括：

- **Neo4j Browser**：用于可视化展示查询结果。
- **Cypher**：用于查询图数据库中实体和关系的代码语言。
- **DiseaseKG**：一个包含 8 类实体和 11 类关系的医疗知识图谱数据集。

5.3 实验内容和要求

请大家酌熟悉 Cypher 编程语句，并在 Neo4j Browser 环境下运行 Cypher 语句。实现特定节点和关系查询，子图查询，索引查询等任务。

5.3.1 实验步骤与代码示例

1. **启动 Neo4j Browser**

进入本地终端，输入`bin\neo4j console`启动图数据库，然后通过`localhost:7687`登录`Neo4j Browser`。

```
D:\neo4j\neo4j-community-5.25.1>bin\neo4j console
Directories in use:
home:         D:\neo4j\neo4j-community-5.25.1
config:       D:\neo4j\neo4j-community-5.25.1\conf
logs:         D:\neo4j\neo4j-community-5.25.1\logs
plugins:      D:\neo4j\neo4j-community-5.25.1\plugins
import:       D:\neo4j\neo4j-community-5.25.1\import
data:         D:\neo4j\neo4j-community-5.25.1\data
certificates: D:\neo4j\neo4j-community-5.25.1\certificates
licenses:     D:\neo4j\neo4j-community-5.25.1\licenses
run:          D:\neo4j\neo4j-community-5.25.1\run
Starting Neo4j.
2025-04-26 05:26:35.034+0000 INFO  Logging config in use: File 'D:\neo4j\neo4j-community-5.25.1\conf\user-logs.xml'
2025-04-26 05:26:35.041+0000 INFO  Starting...
2025-04-26 05:26:35.605+0000 INFO  This instance is ServerId{c6d3e801} (c6d3e801-45d7-401d-89fb-b98fcd2d5181)
2025-04-26 05:26:36.301+0000 INFO  ======== Neo4j 5.25.1 ========
2025-04-26 05:26:37.369+0000 INFO  Anonymous Usage Data is being sent to Neo4j, see https://neo4j.com/docs/usage-data/
2025-04-26 05:26:37.407+0000 INFO  Bolt enabled on localhost:7687.
2025-04-26 05:26:38.061+0000 INFO  HTTP enabled on localhost:7474.
2025-04-26 05:26:38.062+0000 INFO  Remote interface available at http://localhost:7474/
2025-04-26 05:26:38.067+0000 INFO  id: FE127B252BDD3702C88A48D86FD3E0B257898333FBF3F2308BF38DADC129D7D0
2025-04-26 05:26:38.067+0000 INFO  name: system
2025-04-26 05:26:38.068+0000 INFO  creationDate: 2025-04-26T05:03:28.911Z
2025-04-26 05:26:38.068+0000 INFO  Started.
```

图 5.1: 启动 Neo4j Browser

2. **特定实体类型的所有节点查询**

```
MATCH (n:Symptom)
RETURN n
LIMIT 100
```

查询所有类型为 Symptom 的实体，并返回 100 个节点。n 是用户自定义的变量名，用于临时引用匹配到的节点。

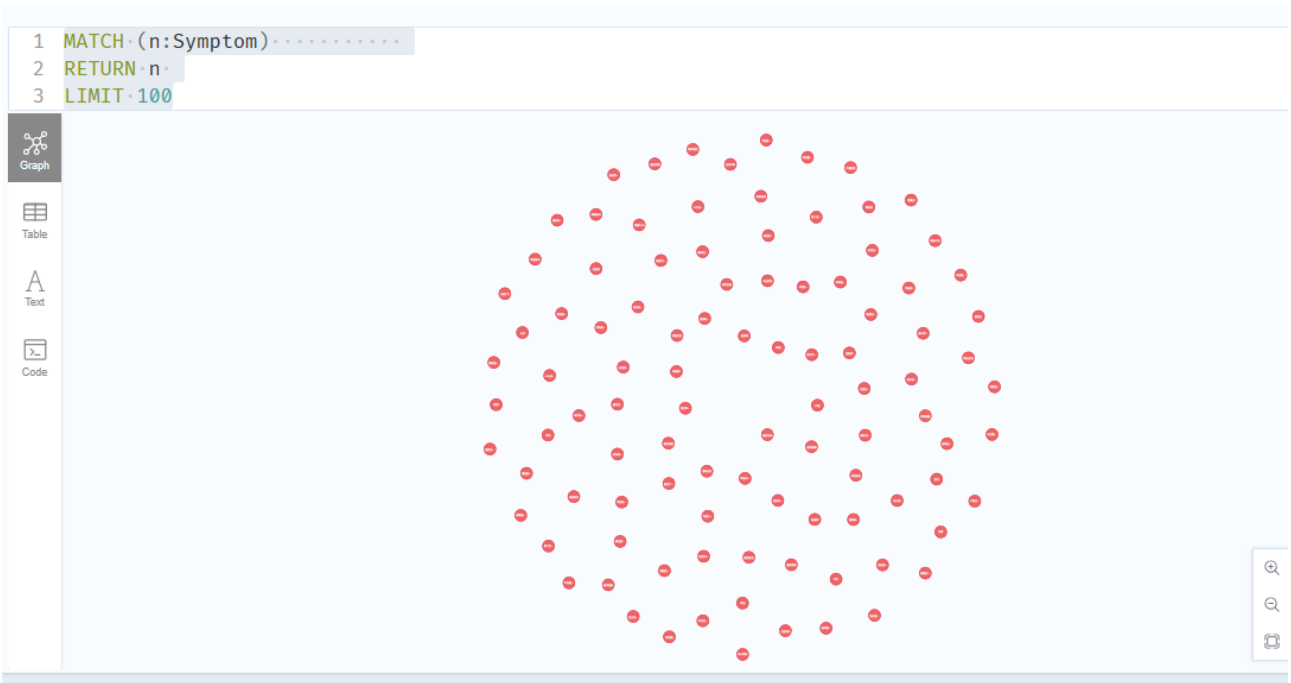

图 5.2: 特定实体类型的所有节点查询

3. **特定实体类型的某个节点查询**

```
MATCH (n:Symptom)
WHERE n.name = "发烧"
```

```
RETURN n
LIMIT 100
```

其中n.name = "发烧" 表示实体的一个属性为name。同时对应的值为“发烧”。

```
1 MATCH (n:Symptom)
2 WHERE n.name = "发烧"
3 RETURN n
4 LIMIT 100
```

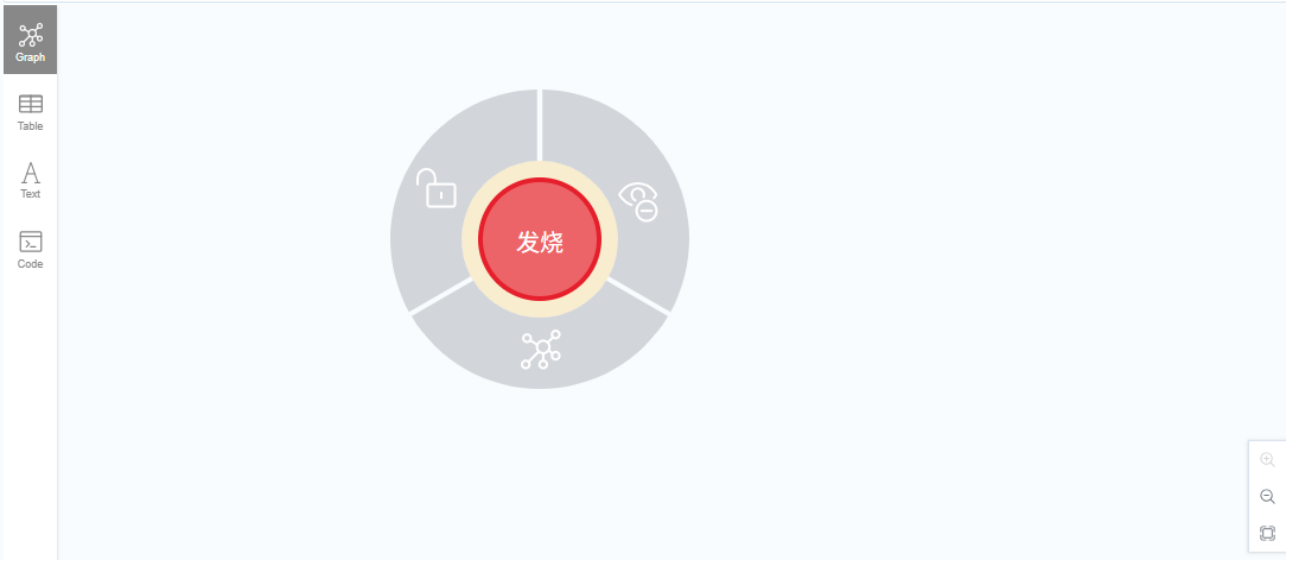

图 5.3: 特定类型的某个节点查询

4. **特定关系类型的所有三元组查询**

```
MATCH (a)-[r:has_symptom]->(b)
RETURN a, r, b
LIMIT 100
```

a表示起始节点（包含标签、属性等）；r表示关系（包含类型、属性等）。b表示目标节点（同 a）

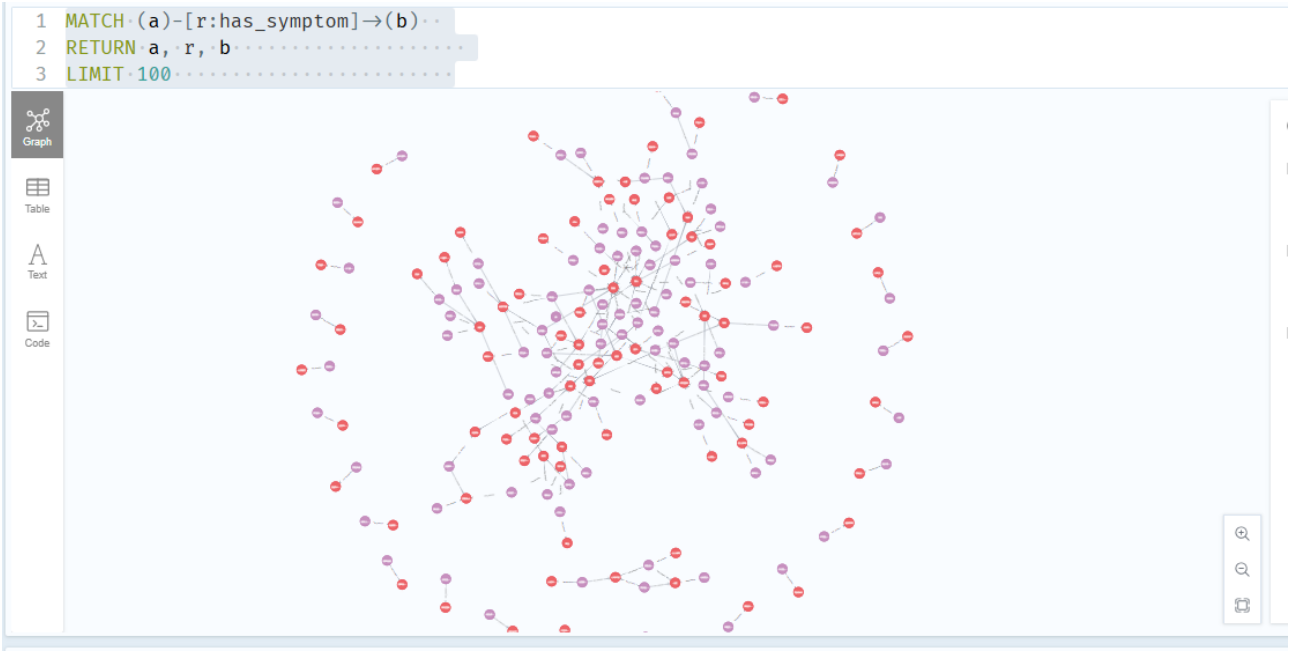

图 5.4: 特定关系类型的所有三元组查询

5. **特定关系类型的某个三元组查询**

```
MATCH (a)-[r:has_symptom]->(b)
WHERE a.name='大肠息肉' and b.name='大肠黑变'
RETURN a, r, b
LIMIT 100
```

a表示起始节点（包含标签、属性等）；r表示关系（包含类型、属性等）。b表示目标节点（同 a）

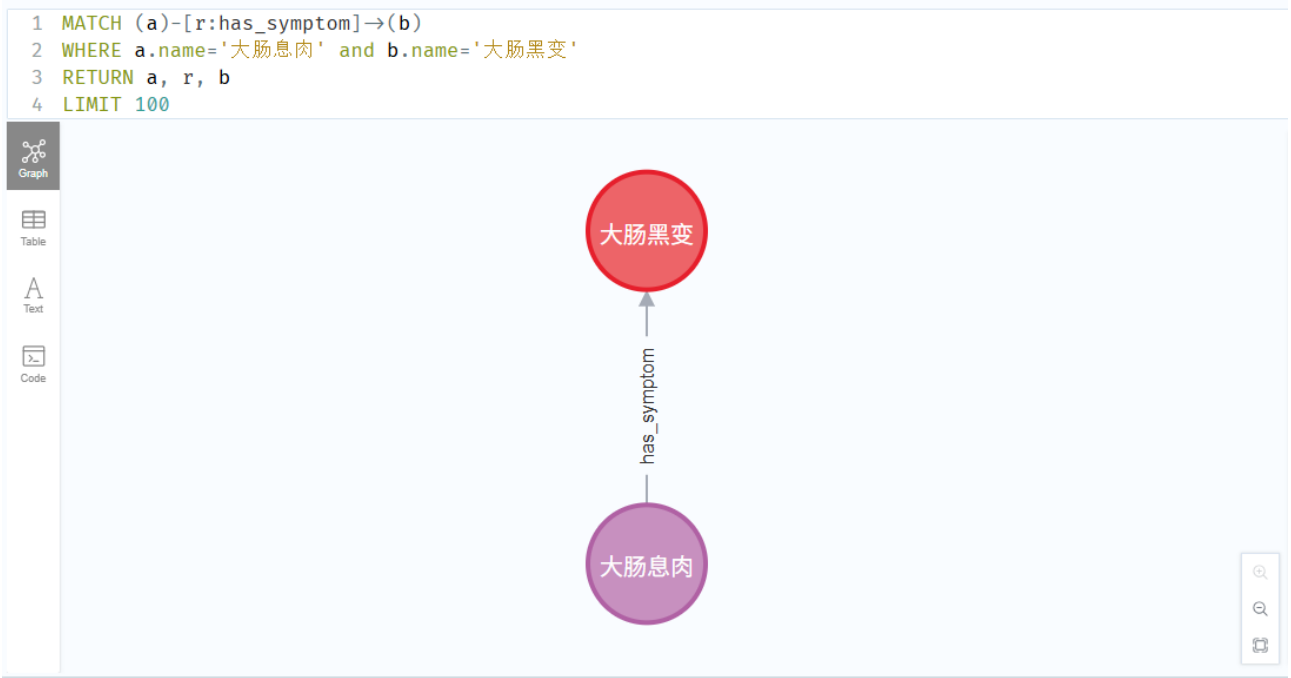

图 5.5: 特定关系类型的某个三元组查询

6. **以某个节点为中心的子图查询**

```
MATCH path = (center:Disease {name: '大肠息肉'})-[*1..1]-(neighbor)
RETURN nodes(path) , relationships(path)
```

明确节点标签（如 Disease、Drug），避免全图扫描（可省略，但会全图扫描，增加查询时间）；[*1..1] 表示深度为 1 的子图，[*1..2] 表示深度为 2 的子图，依次类推；nodes(path) 表示提取路径中的节点列表，relationships(path) 表示提取路径中的关系列表

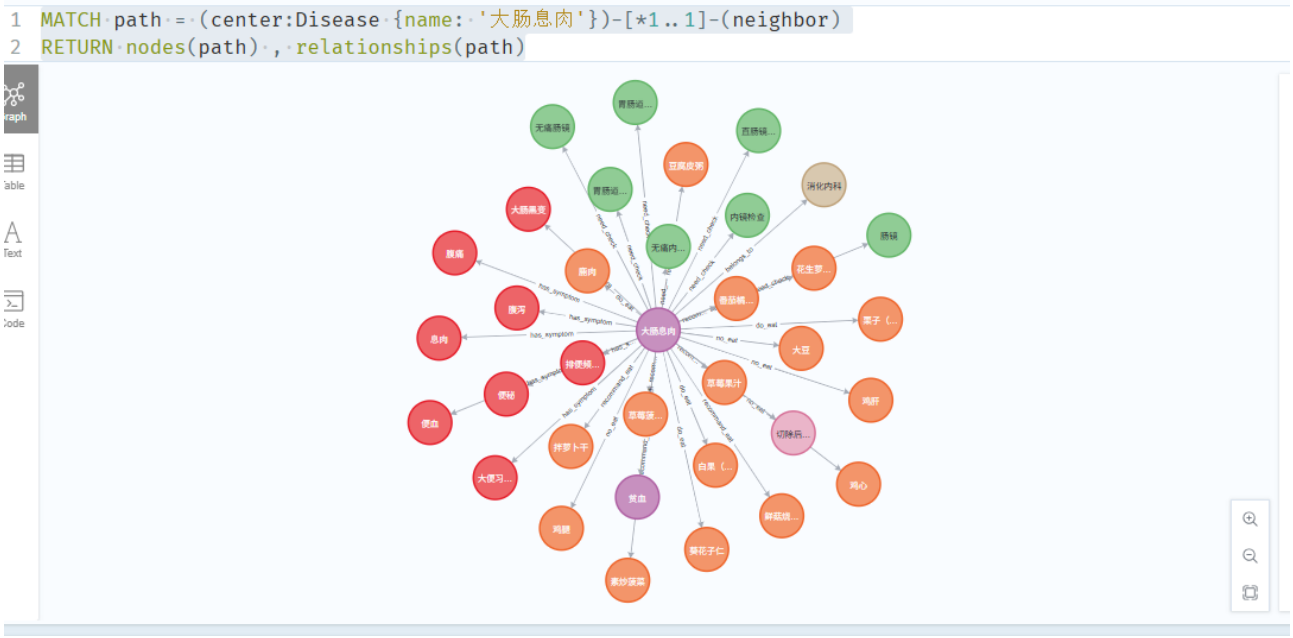

图 5.6: 以某个节点为中心的子图查询

7. **两个节点间的最短路径查询**

```
MATCH path = shortestPath((a:Food)-[*]-(b:Food))
WHERE a.name = "拌萝卜干" AND b.name = "素炒菠菜"
RETURN path
```

-[*]-表示无向图，返回结果path表示拌萝卜干到素炒菠菜之间的最短路径。-[*]->表示有向图

```
MATCH path = shortestPath((a:Food)-[*]-(b:Food))
WHERE a.name = "拌萝卜干" AND b.name = "素炒菠菜"
RETURN path
```

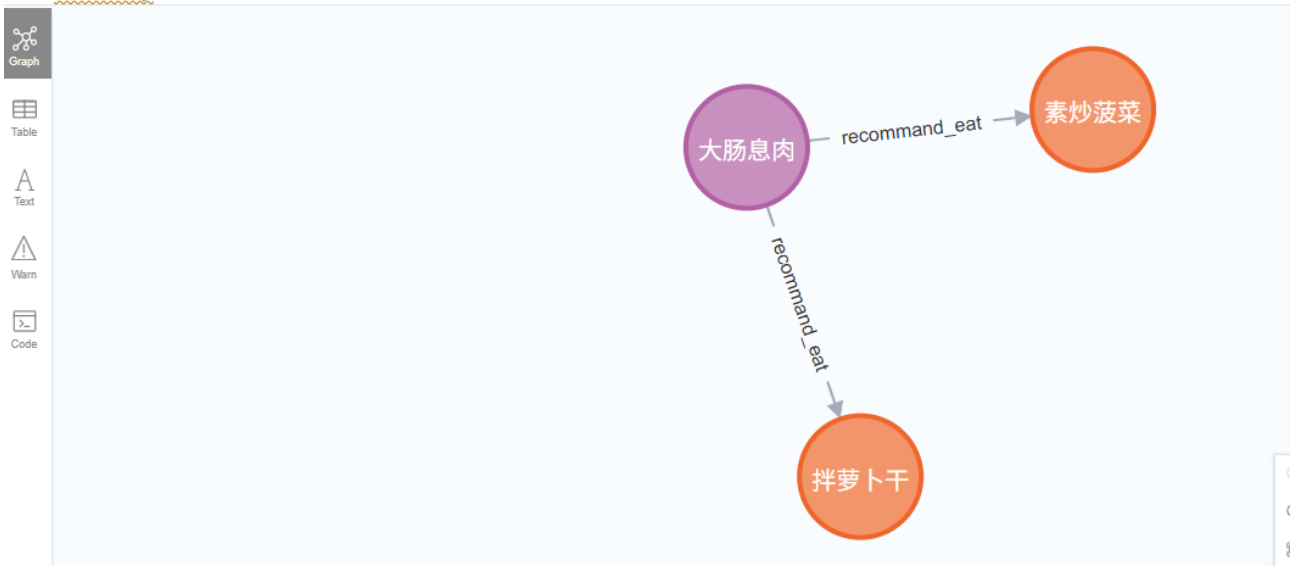

图 5.7: 最短路径查询

8. **分组统计**

```
MATCH (d:Disease)-[:has_symptom]->(s)
RETURN d.name AS 疾病, COUNT(s) AS 症状数量
ORDER BY 症状数量 DESC
对所有疾病的症状数量进行统计
```

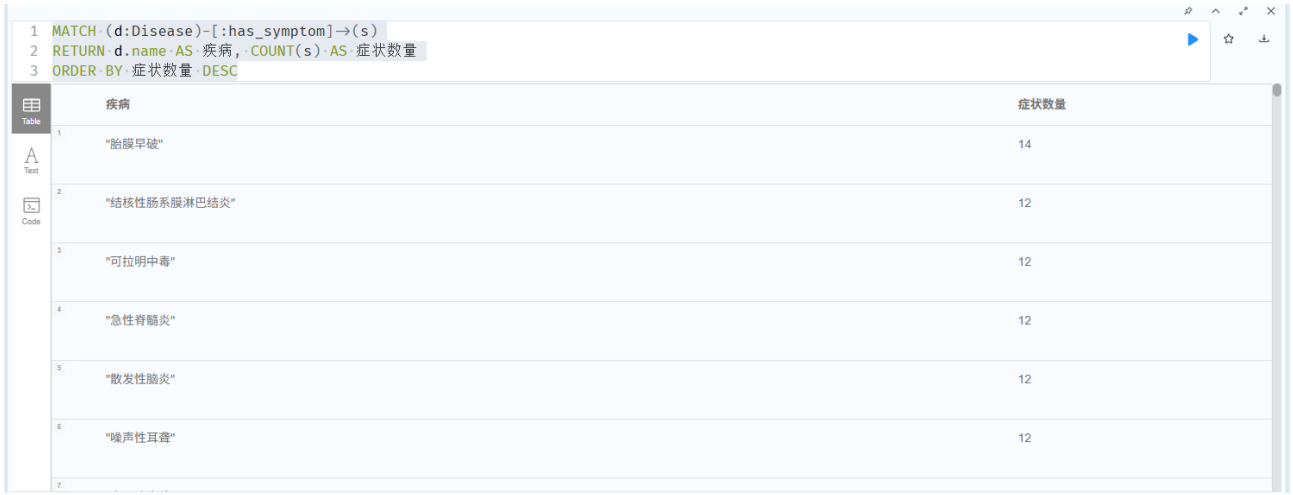

图 5.8: 分组统计

9. **多跳查询**

```
MATCH path = (d:Disease {name: "糖尿病"})
      -[:acompany_with]->(c:Disease)
      -[:has_symptom]->(s:Symptom)
      WHERE NOT (d)-[:has_symptom]->(s)
RETURN nodes(path),relationships(path)
```

上述表示糖尿病的并发症有哪些症状

```
MATCH path = (d:Disease {name: "糖尿病"})
      -[:acompany_with]→(c:Disease)
      -[:has_symptom]→(s:Symptom)
      WHERE NOT (d)-[:has_symptom]→(s)
RETURN nodes(path),relationships(path)
```

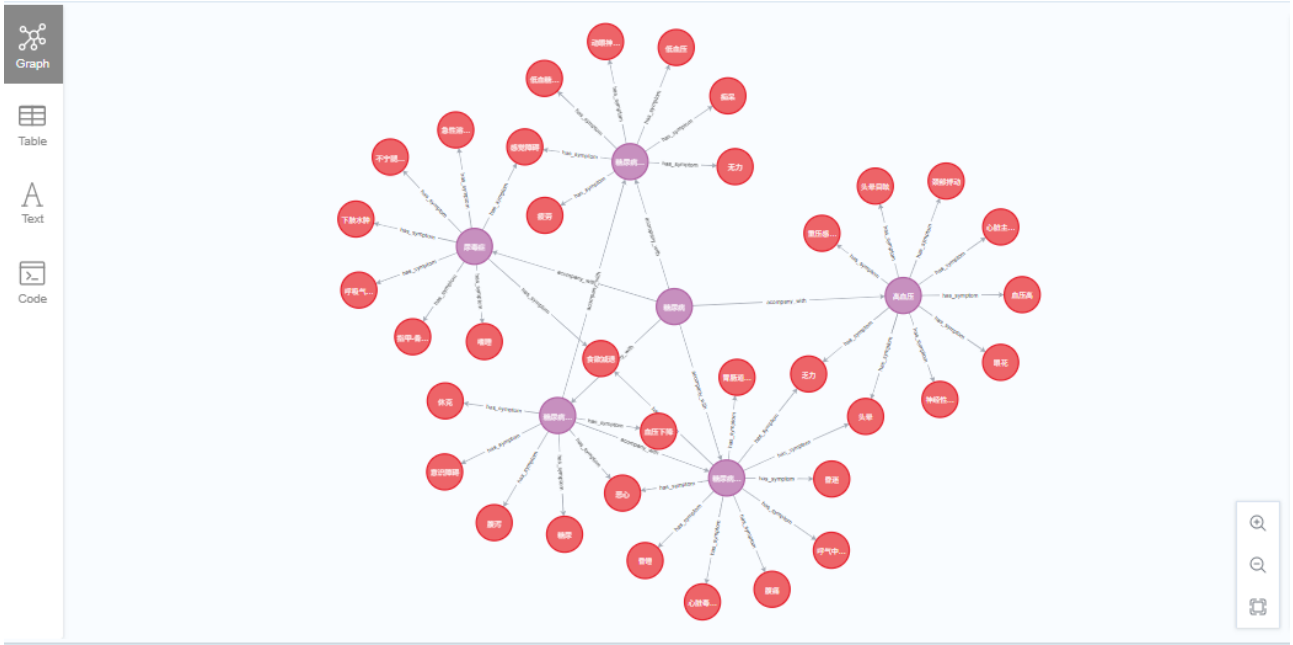

图 5.9: 多跳查询

5.4 实验报告

5.4.1 实验过程

- **数据准备**：已经下载 DiseaseKG 数据集，并将数据导入到本地 neo4j 图数据库。
- **编写 Cypher 语句**：根据特定的查询任务，编写正确的 Cypher 语句。
- **结果评估**：运行编写的 Cypher 语句，检验是否实现任务要求。

5.4.2 数据集详细说明

- **DiseaseKG**：
  - 知识图谱主要包括 8 类实体共 44656 条数据，分别为 Disease、Drug、Food、Check、Department、Producer、Symptom 和 Cure；11 类关系共 312159 条数据，分别为 belongs_to、common_drug、do_eat、drugs_of、need_check、no_eat、recommand_drug、recommand_eat、has_symptom、acompany_with 和 cure_way；其中疾病（*Disease*）实体相较其他实体有额外的属性，分别为 name、desc、prevent、cure_lasttime、cured_prob 和 easy_get。
  - 常用于医疗问答。
  - 获取方式：GitHub 仓库