数据存储格式

将图谱中的点和边信息存储在neo4j数据库中，多模态直接存储在文件系统中，在图谱中的点或边上添加`image`，`visual`，`audio`三个属性来记录相关的图片、gif和音频在文件系统中的相对路径和格式（由于neo4j不支持数组格式，因此将数组直接序列化为json格式的字符串存储）。

点的信息举例

```
{
  "identity": 0,
  "labels": [
    "Entity"
  ],
  "properties": {
    "name": "jersey girl",
    "image": '[{"@id": "/m/en/jersey_girl/n/image/0", "form": "jpg"}, {"@id":
"/m/en/jersey_girl/n/image/1", "form": "jpg"}, {"@id":
"/m/en/jersey_girl/n/image/2", "form": "jpg"}, {"@id":
"/m/en/jersey_girl/n/image/3", "form": "jpg"}, {"@id":
"/m/en/jersey_girl/n/image/4", "form": "jpg"}]',
    "visual": '[{"@id": "/m/en/jersey_girl/n/visual/0", "with-audio": false,
"form": "gif"}, {"@id": "/m/en/jersey_girl/n/visual/1", "with-audio": false,
"form": "gif"}, {"@id": "/m/en/jersey_girl/n/visual/2", "with-audio": false,
"form": "gif"}, {"@id": "/m/en/jersey_girl/n/visual/3", "with-audio": false,
"form": "gif"}, {"@id": "/m/en/jersey_girl/n/visual/4", "with-audio": false,
"form": "gif"}]',
    "audio": '[{"@id": "/m/en/jersey_girl/n/audio/0", "form": "mp3"}]'
  }
}
```

边的信息举例

```
{
  "identity": 513377,
  "start": 101316,
  "end": 1856,
  "type": "AtLocation",
  "properties": {
    "image": '[{"@id":
"/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/image/0", "form": "jpg"},
{"@id": "/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/image/1", "form":
"jpg"}, {"@id": "/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/image/2",
"form": "jpg"}, {"@id":
"/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/image/3", "form": "jpg"},
{"@id": "/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/image/4", "form":
"jpg"}]',
    "license": "cc:by/4.0",
    "visual": '[{"@id":
"/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/visual/0", "with-audio": false,
"form": "gif"}, {"@id":
"/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/visual/1", "with-audio": false,
"form": "gif"}, {"@id":
"/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/visual/2", "with-audio": false,
"form": "gif"}, {"@id":
"/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/visual/3", "with-audio": false,
"form": "gif"}, {"@id":
"/m/[/r/AtLocation/,/c/en/automaton/,/c/en/lab/]/visual/4", "with-audio": false,
"form": "gif"}]',
    "weight": 1.0,
    "surfaceText": "You are likely to find [[an automaton]] in [[lab]]"
  }
}
```

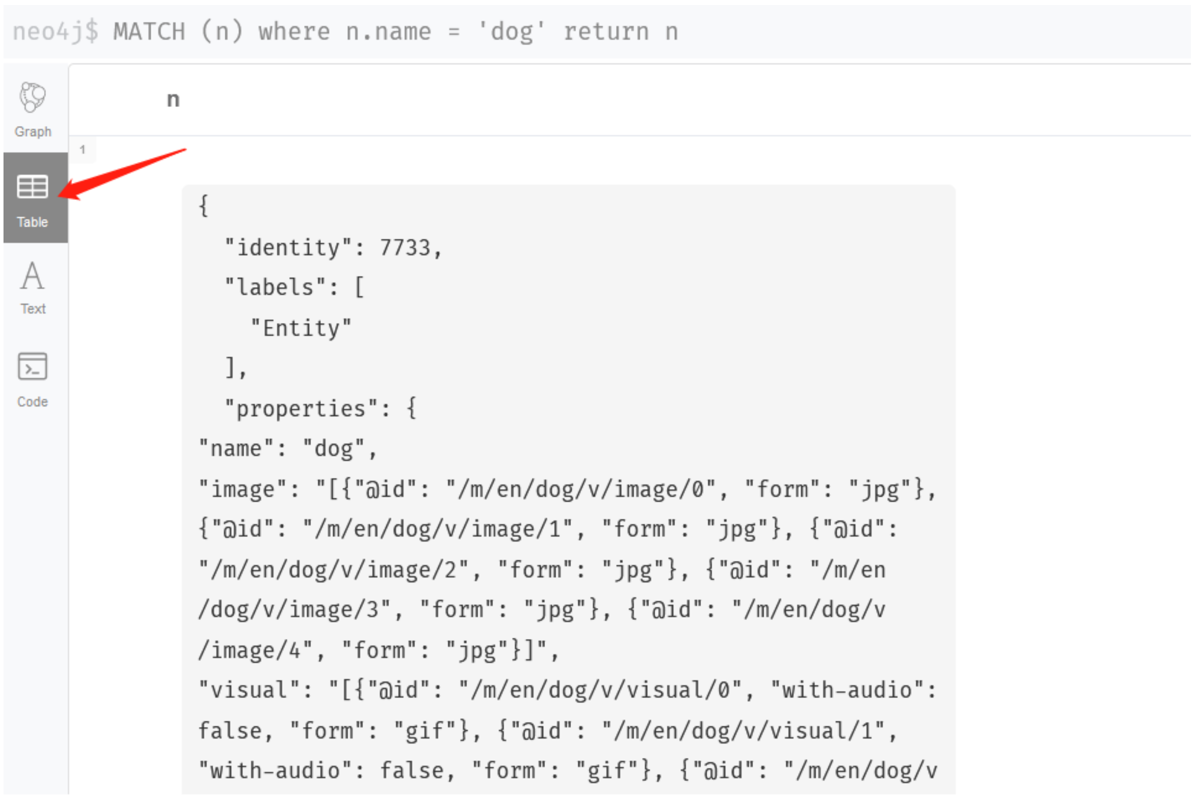
数据添加

当需要加入一个点时，以name作为关键字，先在数据库中寻找要添加的name是否出现过，如果没有出现则添加这个点，随后利用SET语句设置这个点的多模态信息。因此如果多次添加同名的某个点，只会保留一个，且多模态信息为最后添加的那一个。

当需要加入一条边时，先检测其两端的点是否存在，如果不存在则添加一个只有name属性而没有多模态信息的点。随后在这两个点上连边，以label、weight和surfaceText作为判断重复的关键字，多次添加重复的边也只会保留最后一次添加的多模态信息。

数据查询

由于多模态信息比较长，在Graph模式下无法显示，可以切换到Table或Text模式


```
neo4j$ MATCH (n) where n.name = 'dog' return n
```

```
{
  "identity": 7733,
  "labels": [
    "Entity"
  ],
  "properties": {
"name": "dog",
"image": "[{"@id": "/m/en/dog/v/image/0", "form": "jpg"},
{"@id": "/m/en/dog/v/image/1", "form": "jpg"}, {"@id":
"/m/en/dog/v/image/2", "form": "jpg"}, {"@id": "/m/en
/dog/v/image/3", "form": "jpg"}, {"@id": "/m/en/dog/v
/image/4", "form": "jpg"}]",
"visual": "[{"@id": "/m/en/dog/v/visual/0", "with-audio":
false, "form": "gif"}, {"@id": "/m/en/dog/v/visual/1",
"with-audio": false, "form": "gif"}, {"@id": "/m/en/dog/v
```

依名字查看某个点:

```
MATCH (n) where n.name = 'dog' return n
```

依id查看某个点:

```
MATCH (n) where id(n) = 10 return n
```

查看某个点指向其他点的所有边

```
MATCH (n)-[r]->(m) where n.name = "dog" return n, r, m
```

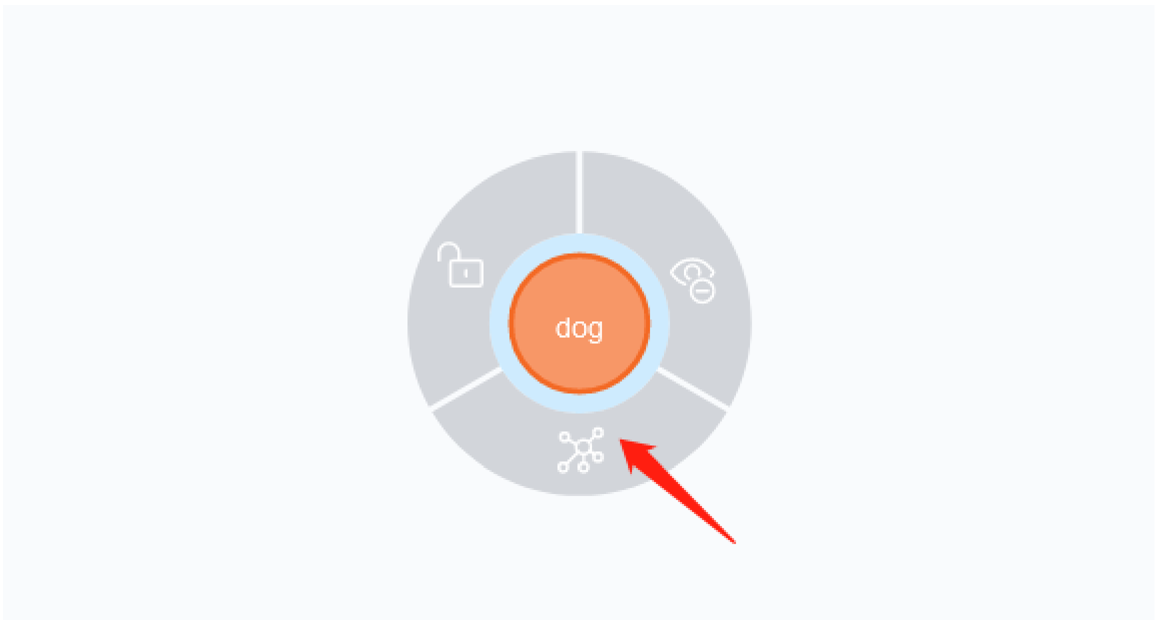
查看其他点指向某个点的特定类型的边

```
MATCH (n)<-[r:DistinctFrom]-(m) where n.name = "dog" return n, r, m
```

查看某个点连的所有边

```
MATCH (n)-[r]-(m) where n.name = "dog" return n, r, m
```

也可以选中某个点后点击相关按键



查看某两点之间连的所有边

```
MATCH (n)-[r]-(m) where n.name = "dog" and m.name = "cat" return n, r, m
```

查询图片信息不为空的点的个数

```
MATCH (n) where n.image <> "" return count(n)
```

查询两点之间长度不超过3的所有路径，按长度排序

```
MATCH (n {name: "dog"})
MATCH (m {name: "cat"})
MATCH p= (n)-[*..3]-(m)
RETURN p
ORDER BY LENGTH(p) DESC
```

查询两点之间长度不超过10且边的类型不为某些特定值的最短路径

```
MATCH (n {name: "dog"})
MATCH (m {name: "cat"})
MATCH p=shortestpath((n)-[r*..10]-(m))
where all(x in r where not(type(x) in ["HasContext", "SimilarTo",
"DistinctFrom", "IsA"]))
RETURN p
```

在python中只需要安装py2neo库就可以运行Cypher代码，例如

```
graph = Graph("http://45.76.225.31:7474", auth=("neo4j", "root"))
data = graph.run(".......(Your code)").data()
```