

Janusgraph 使用介绍

一、Janusgraph 部署

从 Linux 中部署 Janusgraph 单机版，需要 JDK，本文使用的 JDK 版本是 1.8.1_144。

从 Github 网站：<https://github.com/JanusGraph/janusgraph/releases/> 下载最新版的 Janusgraph，本文使用的版本是 0.3.1。

 janusgraph-0.3.1-hadoop2.zip	267,504 KB
--	------------

Janusgraph 单机版需要启动 ElasticSearch，该进程不能由 root 进行启动，会报出异常导致 Janusgraph 启动失败。创建用户新用户并将解压后的 janusgraph 文件夹的全部权限赋予该用户。

配置文件：

打开配置文件 ./janusgraph-0.3.1-hadoop2/conf/gremlin-server/gremlin-server.yaml

我的 gremlin-server.yaml 配置如下：


```
-Xms128m  
-Xmx128m
```

/jvm.options 中更改 即可（es 默认需要 1g 内存）。

如启动成功访问 <http://localhost:8182> 会出现

```
{  
  "message": "no gremlin script supplied"  
}
```

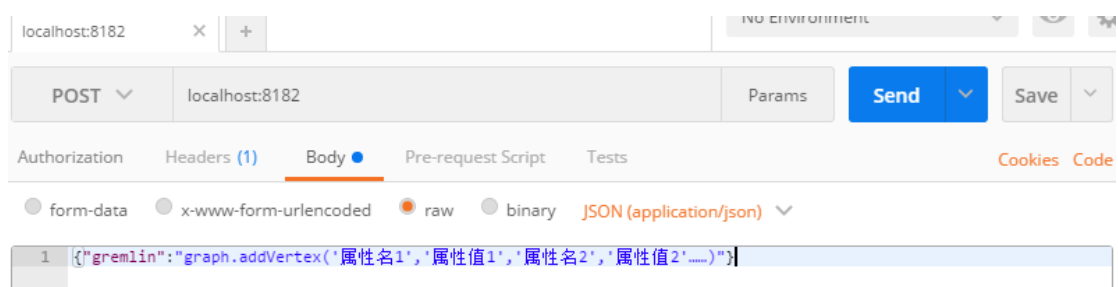
三、Janusgraph 的操作

操作在 https://blog.csdn.net/m0_37204491/article/details/80312278 这篇博客有详细介绍。使用的截图来源软件为 Postman。截图中上半部分 json 数据为 post 提交数据，下半部分为返回数据。

下面介绍的所有操作均为通过 post 提交 json 参数进行操作数据库的，header 为：

☒ Content-Type application/json

3.1 增加节点：



其中省略号代表可以无限多的属性名对属性值，此操作即可成功添加一个节点。

```

1  {
2    "requestId": "4b24b416-02f9-4d6a-aa75-8f10ef2b02cf",
3    "status": {
4      "message": "",
5      "code": 200,
6      "attributes": {}
7    },
8    "result": {
9      "data": {
10       "@type": "g:List",
11       "@value": [
12         {
13           "@type": "g:Vertex",
14           "@value": {
15             "id": {
16               "@type": "g:Int64",
17               "@value": 4160
18             },
19             "label": "vertex",
20             "properties": {
21               "属性名2": [
22                 {}
23               ],
24               "属性名1": [
25                 {}
26               ]
27             }
28           }
29         ]
30       }
31     },
32     "meta": {
33       "@type": "g:Map",
34       "@value": []
35     }
36   }
37 }

```

添加成功则返回如图所示 json；其中 4160 为该节点的 id。

3.2 查找节点：

查找可以直接根据 id 查找

```
1 [{"gremlin": "g.V(4160)"}]
```

Body Cookies Headers (3) Tests Status: 200 OK Time: 107 ms Size: 931 B

Pretty Raw Preview JSON

```
1 {
2   "requestId": "cafc5e8d-8563-4a1e-967e-b46449d0813f",
3   "status": {
4     "message": "",
5     "code": 200,
6     "attributes": {
7       "@type": "g:Map",
8       "@value": []
9     }
10  },
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": [
15        {
16          "@type": "g:Vertex",
17          "@value": {
18            "id": {
19              "@type": "g:Int64",
20              "@value": 4160
21            },
22            "label": "vertex",
23            "properties": {
```

也可以根据属性名查找节点：

```
1 [{"gremlin": "g.V().has('属性名2','属性值2')"}]
```

Body Cookies Headers (3) Tests Status: 200 OK Time: 156 ms Size: 931 B


Pretty Raw Preview JSON

```
1 {
2   "requestId": "1fd45d6d-b5c0-48e0-9ab9-eeb96be8cece",
3   "status": {
4     "message": "",
5     "code": 200,
6     "attributes": {
7       "@type": "g:Map",
8       "@value": []
9     }
10  },
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": [
15        {
16          "@type": "g:Vertex",
17          "@value": {
18            "id": {
19              "@type": "g:Int64",
20              "@value": 4160
21            },
22            "label": "vertex",
23            "properties": {
24              "aa": [
25                {
26                  "@type": "g:VertexProperty"
```

也可以直接获取相应节点的 id

```
1 {"gremlin":"g.V().has('属性名2','属性值2').id()"}

Body Cookies Headers (3) Tests

Pretty Raw Preview JSON 

1 {
2   "requestId": "928525b6-f3c5-4119-9945-5cbb63cd651a",
3   "status": {
4     "message": "",
5     "code": 200,
6     "attributes": {
7       "@type": "g:Map",
8       "@value": []
9     }
10  },
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": [
15        {
16          "@type": "g:Int64",
17          "@value": 4160
18        }
19      ]
20    },
21    "meta": {
22      "@type": "g:Map",
23      "@value": []
24    }
25  }
26 }
```

可以直接查找相应节点属性值：

```
1 {"gremlin":"g.V(4160).values('属性名2')"}

Body Cookies Headers (3) Tests

Pretty Raw Preview JSON

1 {
2   "requestId": "e50cac69-6866-4619-b75c-f11c2aeb678e",
3   "status": {
4     "message": "",
5     "code": 200,
6     "attributes": {
7       "@type": "g:Map",
8       "@value": []
9     }
10  },
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": [
15        "属性值2"
16      ]
17    },
18    "meta": {
19      "@type": "g:Map",
20      "@value": []
21    }
22  }
23 }
```

3.3 修改节点：

修改节点属性（先查询到相应节点）：

```
1 {"gremlin":"g.V(4160).property('属性名1','666')"}

```

Body Cookies Headers (3) Tests

Pretty

Raw

Preview

JSON ▾



```
1 {
2   "requestId": "6fd8df74-b3a6-4ac4-80de-401ed3b480df",
3   "status": {},
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": [
15        {
16          "@type": "g:Vertex",
17          "@value": {
18            "id": {},
22            "label": "vertex",
23            "properties": {
24              "aa": [],
39              "属性名2": [
40                {}
53            ],
54            "属性名1": [
55              {
56                "@type": "g:VertexProperty",
57                "@value": {
58                  "id": {},
64                  "value": "666",
65                  "label": "属性名1"
66                }
67              }
68            ]
69          }

```


3.4 添加关系：

```
1 {"gremlin":"g.V(4160).addE('myself').from(g.V(4160))"}

Body Cookies Headers (3) Tests

Pretty Raw Preview JSON ↗

1 {
2   "requestId": "44cbe88d-36d4-4729-8cc1-ab4074820ace",
3   "status": {
4     "message": "",
5     "code": 200,
6     "attributes": {
7       "@type": "g:Map",
8       "@value": []
9     }
10  },
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": [
15        {
16          "@type": "g:Edge",
17          "@value": {
18            "id": {
19              "@type": "janusgraph:RelationIdentifier",
20              "@value": {
21                "relationId": "2rs-37k-8v85-37k"
22              }
23            },
24            "label": "myself",
25            "inVLabel": "vertex",
26            "outVLabel": "vertex",
27            "inV": {
28              "@type": "g:Int64",
29              "@value": 4160
30            }
31          }
32        }
33      ]
34    }
35  }
36}
```

添加关系中，“myself”是该条关系的名称，from 是边的朝向，与之相对的是“to”，from 后边的括号中接的是另一个节点的查询。

3.5 根据关系查找节点：

```
1 {"gremlin":"g.V(4160).in('myself')"}

Body Cookies Headers (3) Tests Status: 200 OK Time: 105 ms Size: 932 B

Pretty Raw Preview JSON

1 {
2   "requestId": "d672a071-a1d6-4cef-b842-5b18d3eb3ac1",
3   "status": {
4     "message": "",
5     "code": 200,
6     "attributes": {
7       "@type": "g:Map",
8       "@value": []
9     }
10  },
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": [
15        {
16          "@type": "g:Vertex",
17          "@value": {
18            "id": {
19              "@type": "g:Int64",
20              "@value": 4160
21            },
22            "label": "vertex",
23            "properties": {
24              "aa": [
25                {
26                  "@type": "g:VertexProperty",
27                  "@value": {
28                    "id": {
29                      "@type": "janusgraph:RelationIdentifier",
30                      "@value": {
```

该条语句的含义是 id 为 4160 的节点对应关系为 myself 指向的节点，与 in 相对的是“out”。

3.6 删除：

所有查询后接.drop()即为删除该查询的内容，查询的是节点就删除的是节点，是属性就删除的是属性。

```
1 {"gremlin":"g.V(4160).properties('aa').drop()"}

Body Cookies Headers (3) Tests

Pretty Raw Preview JSON

1 {
2   "requestId": "5e8050a2-9ab5-4a66-b389-0ce43cc0ab79",
3   "status": {
4     "message": "",
5     "code": 200,
6     "attributes": {
7       "@type": "g:Map",
8       "@value": []
9     }
10  },
11  "result": {
12    "data": {
13      "@type": "g:List",
14      "@value": []
15    },
16    "meta": {
17      "@type": "g:Map",
18      "@value": []
19    }
20  }
21 }
```

上图为删除一个属性。

3.7 统计:

`{"gremlin":"g.V().count()"}` 统计节点

`{"gremlin":"g.E().count()"}` 统计边

`{"gremlin":"g.V()"}` 所有节点

`{"gremlin":"g.E()"}` 所有边