

CMPE -272 Database Team based Assignment

- 1) Created an IBM Bluemix Graph Service named "IBM Graph-9b 272 assignment" with Graph "g". Create service credentials and use them to generate a GDS token using curl command on Cygwin.

```
Karan@LAPTOP-3B38JAAA ~  
$ curl -X GET -u db644938-c54f-4014-85cb-636eca3e919e 'https://ibmgraph-alpha.ng.bluemix.net/26c4954a-5502-4bc7-bda2-326dccf31df2/_session'  
Enter host password for user 'db644938-c54f-4014-85cb-636eca3e919e':  
{  
  "gds-token": "ZGI2NDQ5MzgtYzU0Zi00MDE0LTg1Y2ItNjM2ZW5MT110jE0ODgxNjk4NzY3NDE6MStqRzgwcmhhR25zcTRkQ3ZPYit2W1hIVnRrWkFmS1hpR2Zobm1xN2ErUT0="}  
Karan@LAPTOP-3B38JAAA ~
```

Attached schema and gremlin JSON file

i)



schema.json

ii)



gremlin.json

- 2) Then create graph database schema and input data

```
Karan@LAPTOP-3B38JAAA ~  
$ curl -X POST -H "Authorization: gds-token ZGI2NDQ5MzgtYzU0Zi00MDE0LTg1Y2ItNjM2ZW5MT110jE0ODgxNjk4NzY3NDE6MStqRzgwcmhhR25zcTRkQ3ZPYit2W1hIVnRrWkFmS1hpR2Zobm1xN2ErUT0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/26c4954a-5502-4bc7-bda2-326dccf31df2/g/schema -d @C:/Users/Karan/Downloads/272/schema.json  
{  
  "requestId": "d956d552-647d-4551-84fb-74ddfee8c6f5",  
  "status": {"message": "", "code": 200, "attributes": {}},  
  "result": {"data": [{"propertyKeys": [{"name": "name", "dataType": "String", "cardinality": "SINGLE"}, {"name": "gender", "dataType": "String", "cardinality": "SINGLE"}, {"name": "age", "dataType": "Integer", "cardinality": "SINGLE"}, {"name": "department", "dataType": "String", "cardinality": "SINGLE"}], "vertexLabels": [{"name": "student"}, {"name": "professor"}, {"name": "department"}], "edgeLabels": [{"name": "is_taught_by", "directed": true, "multiplicity": "MULTI"}, {"name": "specializes_in", "directed": true, "multiplicity": "MULTI"}, {"name": "provides_major_to", "directed": true, "multiplicity": "MULTI"}], "vertexIndexes": [{"name": "vByName", "composite": true, "unique": true, "propertyKeys": [{"name": "name"}, {"name": "vByDepartment", "composite": true, "unique": true, "propertyKeys": [{"department"}, {"name": "vByGender", "composite": true, "unique": false, "propertyKeys": [{"gender"}, {"name": "vByAge", "composite": true, "unique": false, "propertyKeys": [{"age"}], "requiresReindex": false, "type": "vertex"}], "edgeIndexes": []}], "meta": {}  
Karan@LAPTOP-3B38JAAA ~  
$ ^C  
  
Karan@LAPTOP-3B38JAAA ~  
$ curl -X POST -H "Authorization: gds-token ZGI2NDQ5MzgtYzU0Zi00MDE0LTg1Y2ItNjM2ZW5MT110jE0ODgxNjk4NzY3NDE6MStqRzgwcmhhR25zcTRkQ3ZPYit2W1hIVnRrWkFmS1hpR2Zobm1xN2ErUT0=" -H 'Content-Type: application/json' https://ibmgraph-alpha.ng.bluemix.net/26c4954a-5502-4bc7-bda2-326dccf31df2/g/gremlin -d @C:/Users/Karan/Downloads/272/gremlin.json  
{  
  "requestId": "3cea87ed-30cb-466d-a51f-96a0c2179850",  
  "status": {"message": "", "code": 200, "attributes": {}},  
  "result": {"data": [{"id": "4d5-6h4-u1h-9mg", "label": "provides_major_to", "type": "edge", "inVLabel": "student", "outVLabel": "department", "inV": 12472, "outV": 8392}], "meta": {}  
Karan@LAPTOP-3B38JAAA ~  
$ |
```

3) Run Basic queries to generate graphs

The screenshot shows the IBM Graph console interface. The browser address bar displays the URL: <https://console.ng.bluemix.net/data/graphdb/26c4954a-5502-4bc7-bda2-326dccc31df2/query>. The left sidebar contains icons for Upload, Samples, and Resources. The main area has a query input field with the text: `1 // Enter your Gremlin query here. (Shift + Enter) to execute.` Below the input field, the graph is visualized as a single green circle labeled "student". The query editor shows the following code:

```
def g = graph.traversal(); g.V().has("name", "David");
```

The result is displayed as a JSON object:

```
1 * [
2 * {
3 *   "id": 4136,
4 *   "label": "student",
5 *   "type": "vertex",
6 *   "properties": {
7 *     "gender": [
8 *       {
9 *         "id": "111-36w-nph",
10 *        "value": "male"
11 *      }
12 *     ],
13 *     "name": [
14 *       {
15 *         "id": "16t-36w-mx1",
16 *         "value": "David"
17 *       }
18 *     ]
19 *   }
20 * }
21 ]
```

At the bottom, there are filter buttons: Label, Type, Properties. The text "Vertices: 1" is shown at the bottom right.

4) Run transform queries

i) In() query

The screenshot shows the IBM Graph console interface. The browser address bar displays the URL: <https://console.ng.bluemix.net/data/graphdb/26c4954a-5502-4bc7-bda2-326dccc31df2/query>. The left sidebar contains icons for Upload, Samples, and Resources. The main area has a query input field with the text: `1 |`. Below the input field, the graph is visualized as a single green circle labeled "depa:". The query editor shows the following code:

```
def gt = graph.traversal();gt.V().has('name', 'David').in()
```

The result is displayed as a JSON object:

```
1 * [
2 * {
3 *   "id": 20664,
4 *   "label": "department",
5 *   "type": "vertex",
6 *   "properties": {
7 *     "department": [
8 *       {
9 *         "id": "93r-fy0-pad",
10 *        "value": "SE"
11 *      }
12 *     ]
13 *   }
14 * }
15 ]
```

At the bottom, there are filter buttons: Label, Type, Properties. The text "Vertices: 1" is shown at the bottom right.

II) Out() query

The screenshot shows the GraphDB Query Editor interface. The top bar includes 'GraphDB > g' and 'Tutorials Query'. On the left, there are icons for 'Upload', 'Samples', and 'Resources'. The main query area contains the following code:

```
def gt = graph.traversal();gt.V().has('name', 'Jason').out().values('department')
```

The query results are displayed in a table with 3 rows:

1	[
2	"CE"
3]

At the bottom, there is a 'Filter:' section and a 'Vertices: 0' indicator.

III) inE and outV query

The screenshot shows the GraphDB Query Editor interface. The top bar includes 'GraphDB > g' and 'Tutorials Query'. On the left, there are icons for 'Upload', 'Samples', and 'Resources'. The main query area contains the following code:

```
def gt = graph.traversal();gt.V().has('department', 'SE').inE('specializes_in').outV().path()
```

The query results are displayed in a table with 16 rows:

1	[
2	{
3	"labels": [
4	[],
5	[],
6	[]
7],
8	"objects": [
9	{
10	"id": 20664,
11	"label": "department",
12	"type": "vertex",
13	"properties": {
14	"department": [
15	{
16	"id": "93r-fy0-pad",

At the bottom, there is a 'Filter:' section with buttons for 'Label', 'Type', and 'Properties', and a 'Vertices: 2' indicator.

IV) outE and inV query

GraphDB > g

Upload

Samples

Resources

```
1 |
```

```
def gt = graph.traversal();gt.V().has('gender', 'male').outE('is_taught_by').inV()
```

```
1 * [
2 * {
3 *   "id": 4272,
4 *   "label": "professor",
5 *   "type": "vertex",
6 *   "properties": {
7 *     "gender": [
8 *       {
9 *         "id": "11i-3ao-nph",
10 *        "value": "male"
11 *      }
12 *     ],
13 *     "name": [
14 *       {
15 *         "id": "17a-3ao-mx1",
16 *         "value": "Robert"
17 *       }
18 *     ]
19 *   }
20 * }
21 ]
```

Filter: Label Type Properties

Vertices: 4

V) path query

IBM Graph

GraphDB > g

Documentation

Bluemix Dashboard

Tutorials

Query

Upload

Samples

Resources

```
1 |
```

Graph: g

```
def gt = graph.traversal();gt.V().has('department', 'SE').outE('provides_major_to').inV().path()
```

```
1 * [
2 * {
3 *   "labels": [
4 *     [],
5 *     [],
6 *     []
7 *   ],
8 *   "objects": [
9 *     {
10 *      "id": 20664,
11 *      "label": "department",
12 *      "type": "vertex",
13 *      "properties": {
14 *        "department": [
15 *          {
16 *            "id": "93n-fy8-pad",
17 *            "value": "SE"
18 *          }
19 *        ]
20 *      }
21 *     }
22 *   ]
23 * }
24 ]
```

Filter: Label Type Properties

Vertices: 5

VI) both query

IBM Graph Documentation Bluemix Dashboard

GraphDB > g

Upload

Samples

Resources

1 |

Graph: g

```
def gt = graph.traversal();gt.V().has('department', 'SE').both()
```

Filter: Label Type Properties

Vertices: 5

VII) bothE and bothV query

IBM Graph Documentation Bluemix Dashboard

GraphDB > g

Upload

Samples

Resources

1 |

Graph: g

```
def gt = graph.traversal();gt.V().has('department', 'SE').bothE('specializes_in', 'provides_major_to').bothV().path
```

Filter: Label Type Properties

Vertices: 6