Lab 4

Additya Dharangaonkar

001052304

Part 1 - Gremlin (4 points)

We'll use Cosmos DB (Gremlin API) for Part 1

1) Write Gremlin code to implement the attached employee data and work relationship graph in Cosmos DB (Gremlin API).

- g.addV('employee').property('EmployeeID', 2).property('LastName', 'Fuller').property('FirstName', 'Andrew').property('Department','NULL')

- g.addV('employee').property('EmployeeID', 3).property('LastName', 'Leverling').property('FirstName', 'Janet').property('Department','IT')

- g.addV('employee').property('EmployeeID', 5).property('LastName', 'Buchanan').property('FirstName', 'Steven').property('Department','Finance')

- g.addV('employee').property('EmployeeID', 7).property('LastName', 'King').property('FirstName', 'Robert').property('Department','Finance')

- g.addV('employee').property('EmployeeID', 12).property('LastName', 'Chang').property('FirstName', 'Leslie').property('Department','Finance')

- g.addV('employee').property('EmployeeID', 14).property('LastName', 'Ng').property('FirstName', 'Jordan').property('Department','Finance')

- g.addV('employee').property('EmployeeID', 15).property('LastName', 'Black').property('FirstName', 'Lela').property('Department','IT')

- g.addV('employee').property('EmployeeID', 21).property('LastName', 'Thompson').property('FirstName', 'Connie').property('Department','IT')

- g.V().hasLabel('employee').has('EmployeeID', 3).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 2))

- g.V().hasLabel('employee').has('EmployeeID', 7).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 2))

- g.V().hasLabel('employee').has('EmployeeID', 5).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 2))

- g.V().hasLabel('employee').has('EmployeeID', 15).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 2))

- g.V().hasLabel('employee').has('EmployeeID', 7).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 3))
- g.V().hasLabel('employee').has('EmployeeID', 7).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 5))

- g.V().hasLabel('employee').has('EmployeeID', 12).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 7))

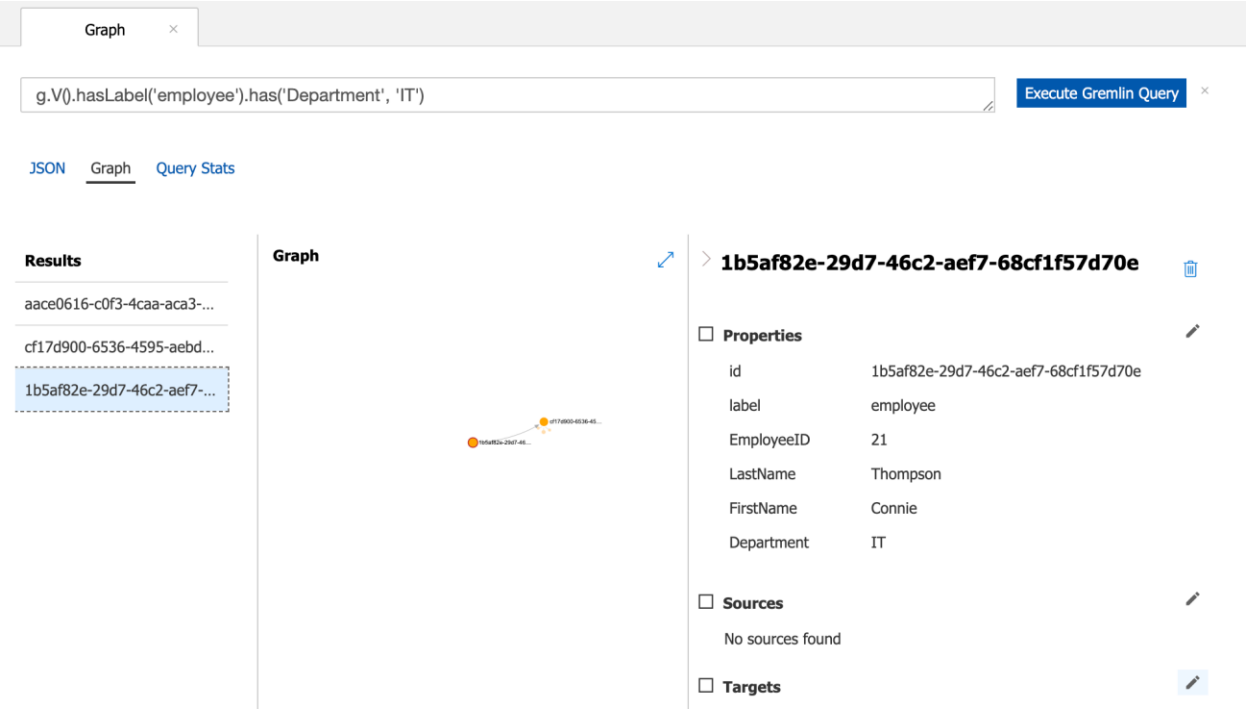
- g.V().hasLabel('employee').has('EmployeeID', 14).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 7))

- g.V().hasLabel('employee').has('EmployeeID', 15).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 7))

- g.V().hasLabel('employee').has('EmployeeID', 21).addE('knows').to(g.V().hasLabel('employee').has('EmployeeID', 15))

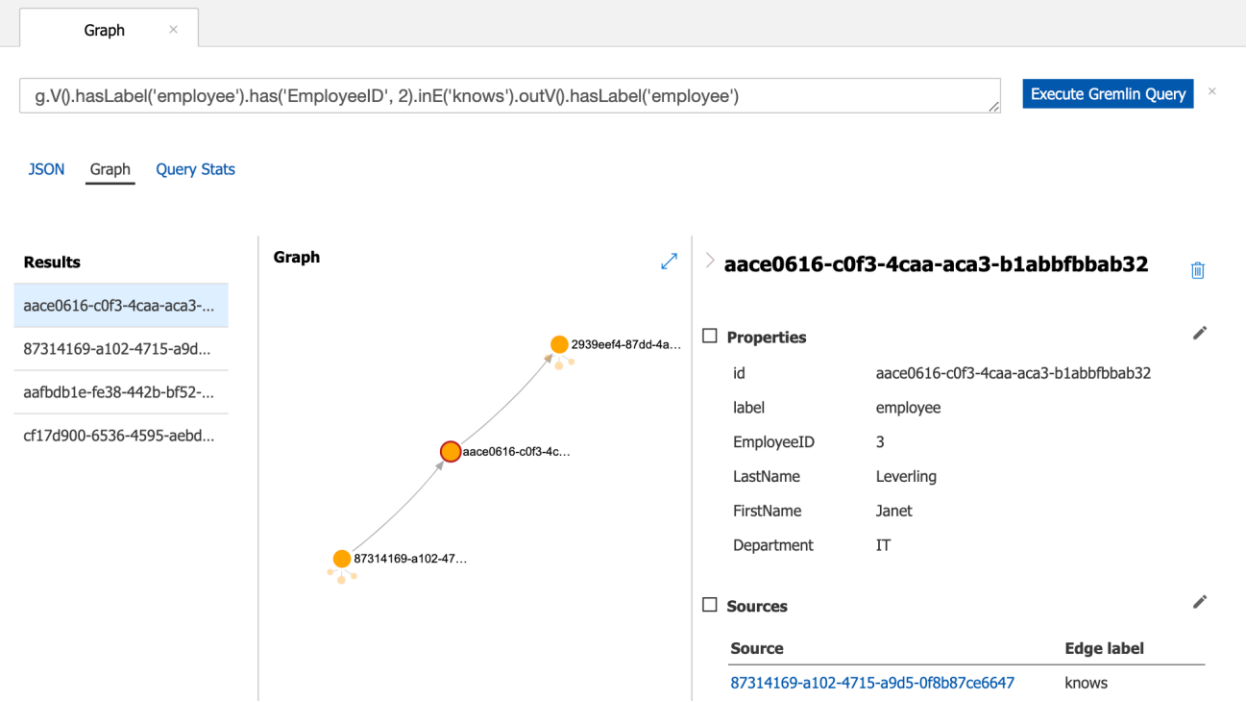
2) Write a Gremlin query to retrieve all employees in the IT department.

- g.V().hasLabel('employee').has('Department', 'IT')



3) Write a gremlin query to retrieve all employees who have EmpID 2 as either a direct manager or an indirect manager.

- g.V().hasLabel('employee').has('EmployeeID', 2).inE('knows').outV().hasLabel('employee')



Part 2 – MongoDB (4 points)

We'll use MongoDB on MongoDB Atlas for Part 2

(1) Implement the "Employee Data" and "Work Relationship Graph" in Mongo Shell.

```
db.Lab4.insertMany([

{"_id": 1, "EmployeeID": 2, "LastName": "Fuller", "FirstName": "Andrew", "Department": null,
"TeamLead": null},

{"_id": 2, "EmployeeID": 3, "LastName": "Leverling", "FirstName": "Janet", "Department": "IT",
"TeamLead": 2},

{"_id": 3, "EmployeeID": 5, "LastName": "Buchanan", "FirstName": "Steven", "Department":
"Finance", "TeamLead": 2},

{"_id": 4, "EmployeeID": 7, "LastName": "King", "FirstName": "Robert", "Department": "Finance",
"TeamLead": [2,3,5]},

{"_id": 5, "EmployeeID": 12, "LastName": "Chang", "FirstName": "Leslie", "Department": "Finance",
"TeamLead": 7},

{"_id": 6, "EmployeeID": 14, "LastName": "Ng", "FirstName": "Jordan", "Department": "Finance",
"TeamLead": 7},

{"_id": 7, "EmployeeID": 15, "LastName": "Black", "FirstName": "Lela", "Department": "IT",
"TeamLead": 7},

{"_id": 8, "EmployeeID": 21, "LastName": "Thompson", "FirstName": "Connie", "Department": "IT",
"TeamLead": 15}

])
```

(2) Write JavaScript to retrieve all employees who have EmpID 2 as either a direct manager or an indirect manager.

```
const MongoClient= require('mongodb').MongoClient;
const assert = require('assert');

var url=
'mongodb+srv://usergraph:employeedb@cluster0.blio04.mongodb.net/employeegraph?retryW
rites=true&w=majority'

const dbName= 'employeegraph';

MongoClient.connect(url, { useNewUrlParser: true }, function(err, client) {
assert.equal(null, err);
```

```
console.log("Connected successfully to server");

const db= client.db(dbName);

db.collection("empgraph").aggregate([
{ $match: { "EmployeeID": 2 } },
{
$graphLookup: {
   from: 'empgraph',
   startWith: '$EmployeeID',
   connectFromField: 'EmployeeID',
   connectToField: 'TeamLead',
   as: TL,
   maxDepth: 1,
   depthField: 'Level'
}},
{
$project: {
   "Direct and Indirect Reportees of EmpID 2": "$TL.EmployeeID" , _id: 0
}}
]).toArray(function(err, result) {
if (err) throw err;
console.log(result);

console.log("Disconnecting .....");
client.close();
});
});
```

(3) Write a JavaScript query to retrieve Employee 12's skip supervisor(s). Employee 12 reports to his skip supervisor(s) via a middle supervisor.

```
const MongoClient= require('mongodb').MongoClient;

const assert = require('assert');

var url=
'mongodb+srv://usergraph:employeedb@cluster0.blio04.mongodb.net/employeegraph?retryWrites=true&w=majority'

const dbName= 'employeegraph';

MongoClient.connect(url, { useNewUrlParser: true }, function(err, client) {

assert.equal(null, err);
```

```
console.log("Connected successfully to server");
const db= client.db(dbName);
db.collection("empgraph").aggregate([
{ $match: { "EmployeeID": 12 } },
{
$graphLookup: {
    from: 'empgraph',
    startWith: '$TeamLead',
    connectFromField: 'TeamLead',
    connectToField: 'EmployeeID',
    as: TL,
    maxDepth: 1,
    depthField: 'Level'
}},
{
$project: {
    "Skip & Middle Supervisors": "$TL.EmployeeID" , _id: 0
}}
]).toArray(function(err, result) {
if (err) throw err;
console.log(result);
console.log("Disconnecting .....");
client.close();
});
});
```