Neo4j and R: A network of possibilities

MAJ Nicholas Uhorchak

31 March, 2020

Introduction

Self described as the "World's leading graph database, with native graph storage and processing," neo4j is a graph database, with its own native query language, Cypher. Neo4j is extremely flexible, as both nodes and relationships contain attributes in the form of JSON documents.

Neo4j provides great tutorials here and here, as well as worst practices guide from their website. Although these tutorials are great references, users shouls familiarize themselves with both database and graph network basics prior to delveing deep into this demo.

```
library(neo4r)
library(magrittr)
```

For this project, a blank database was created using the neo4j desktop client (add graph function from the neo4j GUI). Once created, the 'Start' button must be clicked to move the database to an active state.

The database will model the hit comedy TV show 'The Office' and all references will revolve around the main cast of characters, found here. Relationships from the show are referenced both in the previous link as well as here.

- Please note, that Neo4j native connection protocol is *bolt* and the desktop client will automatically launch a web browser that way. The *current* neo4j driver package (neo4R) for R does not currently support bolt.
- Although provided for this demo, it is good practice not to store database names and passwords in public documents. The R Studio API provides a good mechanism to prompt for username and password.

First, a connection string is defined, using notation outlined in the reference documents linked above.

```
# Connect to Neo4J

con <- neo4j_api$new(url = "http://localhost:7474/", user = "neo4j", password = "markdown")
con$ping()
con$access()
```

Now that a connection is established to the database, we can begin to use neo4j from R. All future calls from R to the database will utilize 'call_neo4j(con = con)' to connect R to the DB using the connection string.

Manual creation of a database

This method entails building the database from *scratch*, using cypher queries to build both nodes and edges.

Everything in neo4j is stored as a JSON document, both nodes and edges. Creating a database from scratch entials defining nodes, edges and all the properties for each that will be contained in the respective JSON documents.

Create nodes

Creating nodes using cypher is straightforward, however care must be taken to ensure that consistency between labels is ensured (case sensitivity). Cypher commands can be found here.

We will begin by creating the main characters from the show. For each character, the node label, n, will reference the major section where they work, and supporting information will be entered as properties. For conformity, each node will be assigned the same set of properties.

Management

```
"create(:Management {Character_name: 'Michael Scott',
Played_by: 'Steve Carrell', Gender: 'Male',
Position: 'Regional Manager', Location: 'Scranton Branch',
Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Management {Character_name: 'Pam Beesly',
Played_by: 'Jenna Fischer', Gender: 'Female',
Position: 'Receptionist',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Management {Character_name: 'Bob Vance',
Played_by: 'Robert Shaefer', Gender: 'Male',
Position: 'Owner, Vance Refrigeration; Scranton Mob Boss(?)',
Location: 'Scranton ', Company: 'Vance Refrigeration'})" %>% call_neo4j(con = con)

"create(:Management {Character_name: 'Jan Levinson',
Played_by: 'Melora Hardin', Gender: 'Female',
Position: 'VP, Regional Sales',
Location: 'Northeast Region', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Management {Character_name: 'Erin Hannon',
Played_by: 'Ellie Kemper', Gender: 'Female',
Position: 'Receptionist',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)
```

Accounting

```
"create(:Accounting {Character_name: 'Angela Martin',
Played_by: 'Angela Kinsey', Gender: 'Female',
Position: 'Accountant',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)
```

```
"create(:Accounting {Character_name: 'Kevin Malone',
Played_by: 'Brian Bumgartner', Gender: 'Female',
Position: 'Accountant',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Accounting {Character_name: 'Oscar Matrinez',
Played_by: 'Oscar Nunez', Gender: 'Male',
Position: 'Accountant',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Accounting {Character_name: 'Stanley Hudson',
Played_by: 'Robert R. Shafer', Gender: 'Male',
Position: 'Accountant',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)
```

Customer Service

```
"create(:Customer_Service {Character_name: 'Kelly Kapoor',
Played_by: 'Mindy Kaling', Gender: 'Female',
Position: 'Customer service representative',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Customer_Service {Character_name: 'Meredith Palmer',
Played_by: 'Kate Flannery', Gender: 'Female',
Position: 'Supplier relations', Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4
```

Human Resources

```
"create(:Human_Resources {Character_name: 'Toby Flenderson',
Played_by: 'Paul Lieberstein', Gender: 'Male',
Position: 'Human Resources representative',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)
```

Quality Assurance

```
"create(:Quality_Assurance {Character_name: 'Creed Bratton',
Played_by: 'Creed Bratton', Gender: 'Male',
Position: 'Director, Quality Assurance', Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% c
```

Sales

```
"create(:Sales {Character_name: 'Dwight Schrute',
Played_by: 'Riann Wilson', Gender: 'Male',
Position: 'Salesman and Assistant Regional Manager',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)
```

```
"create(:Sales {Character_name: 'Jim Halpbert',
Played_by: 'John Krasinski', Gender: 'Male',
Position: 'Salesman; Assistant Regional Manager',
Location: 'Scranton Branch; Staford Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Sales {Character_name: 'Ryan Howard',
Played_by: 'B.J. Novak', Gender: 'Male',
Position: 'Salesman; VP Northeast Region',
Location: 'Scranton Branch; Northeast Region', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Sales {Character_name: 'Phyllis Vance',
Played_by: 'Phyllis Smith', Gender: 'Female',
Position: 'Saleswoman',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Sales {Character_name: 'Andy Bernard',
Played_by: 'Ed Helms', Gender: 'Female',
Position: 'Regional Director of Sales; Salesman',
Location: 'Stamford Branch; Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)
```

Supply

```
"create(:Supply {Character_name: 'Darryl Philbin',
Played_by: 'Craig Robinson', Gender: 'Male',
Position: 'Warehouse Foreman',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)

"create(:Supply {Character_name: 'Roy Anderson',
Played_by: 'David Denman', Gender: 'Male',
Position: 'Warehouse dock worker',
Location: 'Scranton Branch', Company: 'Dunder Mifflin'})" %>% call_neo4j(con = con)
```

Create Relationships

Please note, that although many relationships are captured, this is not an all inclusive list, given the breadth of possibilities that exist. The selection captures enough relationhips to demonstrate the use of neo4j and neo4r.

- Note, similar to the creation of nodes, eges are also JSON documents, and can therefore store properties as well as labels.
- Note - relationships, although defined by direction in the create statement, can inherentlybe travsersed in a bi-directional manner with cypher, so therefore 'if one implies the other' is true. See this for reference. This fact does not necessarily imply however, that some explicit definition (think call network) is not ever necessary.
- Note - the <> operator is the logical 'not' or '!' operator.

Work Relationships

```
"match (n:Management),(m)
where n.Character_name = 'Michael Scott' and m.Location =~ 'Scranton Branch.*' and m.Character_name <>
create (m)-[r:worked_for]->(n)
return n,m" %>% call_neo4j(con = con)
```

Additional relationship - late add to scranton branch

```
"match (n:Management {Character_name: 'Michael Scott'}), (m:Sales)
where m.Character_name = 'Andy Bernard'
create(m)-[r:worked_for]->(n)
return n,m" %>% call_neo4j(con = con)
```

Michael Scott - Jan Levinson

```
"match(n:Management),(m:Management)
where n.Character_name = 'Michael Scott' and m.Character_name = 'Jan Levinson'
create(n)-[r:worked_for {start:'Season 1',end:'Season 3'}]->(m)
create(n)-[s:dated {start:'Season 2',end:'Season 4'}]->(m)
return(n),(m)" %>% call_neo4j(con = con)
```

Roy Anderson - Pam Beesly

```
"match(n),(m)
where n.Character_name = 'Pam Beesly' and m.Character_name = 'Roy Anderson'
create(n)-[s:engaged_to {start:'Season 1',end:'Season 2'}]->(m)
return(n),(m)" %>% call_neo4j(con = con)
```

Angela Martin - Andy Bernard

```
"match(n),(m)
where n.Character_name = 'Angela Martin' and m.Character_name = 'Andy Bernard'
create(n)-[s:dated {start:'Season 4',end:'Season 5'}]->(m)
create(n)-[t:engaged_to {start:'Season 4',end:'Season 5'}]->(m)
return(n),(m)" %>% call_neo4j(con = con)
```

Ryan Howard - Kelly Kapoor

```
"match(n),(m)
where n.Character_name = 'Ryan Howard' and m.Character_name = 'Kelly Kapoor'
create(n)-[s:dated {start:'Season 1',end:'Season 9', frequency:'intermittent'}]->(m)
return(n),(m)" %>% call_neo4j(con = con)
```

Andy Bernard - Erin Hannon

```
"match(n),(m)
where n.Character_name = 'Andy Bernard' and m.Character_name = 'Erin Hannon'
create(n)-[s:dated {start:'Season 6',end:'Season 9'}]->(m)
return(n),(m)" %>% call_neo4j(con = con)
```

Kelly Kapoor - Darryl Philbin

```
"match (n),(m)
where n.Character_name = 'Kelly Kapoor' and m.Character_name = 'Darryl Philbin'
create (n)-[r:dated {start:'Season 4', end: 'Season 5'}]->(m)
return n,m" %>% call_neo4j(con = con)
```

Dwight Schrute - Angela Martin

- Note, when creating properties of a node or relationship, property names cannot be duplicitive, as doing so will overwrite the previous instance with the last one.

```
"match(n),(m)
where n.Character_name = 'Dwight Schrute' and m.Character_name = 'Angela Martin'
create(n)-[r:dated {start1:'Season 2', finish1:'Season 4', start2: 'Season 5', finish2: 'Season 5'}]->(
create(n)-[s:engaged_to {start:'Season 9', finish:''}]->(m)
return (n),(m)" %>% call_neo4j(con = con)
```

Phyllis Vance - Bob Vance

```
"match(n),(m)
where n.Character_name = 'Phyllis Vance' and m.Character_name = 'Bob Vance'
create(n)-[r:dated {start1:'Season 2', finish1:'Season 3'}]->(m)
create(n)-[s:engaged_to {start:'Season 3', finish:'Season 3'}]->(m)
create(n)-[t:married_to {start:'Season 3', finish:''}]->(m)
return (n),(m)" %>% call_neo4j(con = con)
```

Jim Halpbert - Pam Beesly

```
"match(n),(m)
where n.Character_name = 'Jim Halpbert' and m.Character_name = 'Pam Beesly'
create(n)-[r:dated {start1:'Season 4', finish1:'Season 5'}]->(m)
create(n)-[s:engaged_to {start:'Season 5', finish:'Season 6'}]->(m)
create(n)-[t:married_to {start:'Season 6', finish:''}]->(m)
return (n),(m)" %>% call_neo4j(con = con)
```

Importing graphs

In the neo4j.conf file, the following lines must be added to ensure that you can both import and export files using the apoc calls and specify the export directory:

```
apoc.export.file.enabled=true
apoc.import.file.enabled=true
dbms.directories.import=C:/Users/NickUhorchak/OneDrive - SOFDS/Desktop
dbms.security.allow_csv_import_from_file_urls=true
```

Keep in mind (a late find) that each database that you create with the neo4j desktop application, has its own config file and additional packages (APOC being one) and you must therefore change it for the DB instance. (I have not seen a global config file, as of writing this guide). Changes to the config file can be made through the Neo4j GUI, as well as adding plugins.

Import from csv

This method entails building the database from a pre filled *csv* document, stiching together nodes, edges and information using csv document and cypher commands.

Import from JSON

This method entails building the database from a *JSON* document, where nodes, eges and relationships are predefined and stored in a format native to neo4j.

Exporting graphs

Exporting a graph to a csv file can be completed multiple ways, as documented here.

CALL apoc.export.csv.all("markdown.csv", {}) will export a neo4j graph database, into a .csv file. The arguments are apoc.export.csv.all(file,config), where "markdown.csv" is the filename, and config is the "{}" empty bracket, instructing the selection of all elements in the database.

```
"CALL apoc.export.csv.all('markdown.csv', {})" %>% call_neo4j(con = con)
```

CALL apoc.export.json.all("markdown.JSON", {})will export a neo4j graph database into a JSON document. The arguments are apoc.export.csv.all(file,config), where "markdown.JSON" is the filename, and config is the "{}" empty bracket, instructing the selection of all elements in the database.

```
"CALL apoc.export.csv.all('markdown.JSON', {})" %>% call_neo4j(con = con)
```

Note: If you have issues exporting a database to csv, some have found that you must create a new database, amend the config file and then re-run the procedure call. Reference to this issue is here