Neo4j Movies Template

Updated to run on Windows with new Neo4j

Craig Miller

01/10/2017

Goals

Update project (neo4j-movies-template) to work on Windows with Neo4j Enterprise 3.1.0 (and fix numerous lint warnings).

Prerequisites

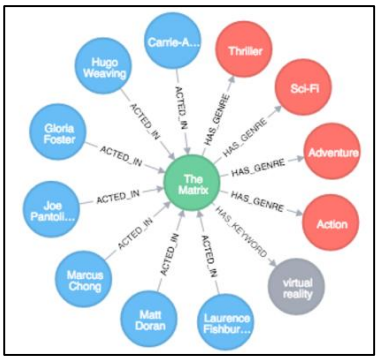
Familiarity with Python, Flask, and Node.js in addition to Neo4j will be required.

Overview

Default Project

This Neo4j-based node / react web app displays movie and person data in a manner similar to IMDB. It is designed to serve as a template for further development projects. Feel encouraged to fork and update this repo!

The Model



Nodes

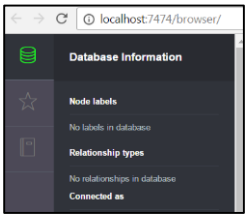
- `Movie`
- `Person`
- `Genre`
- `Keyword`

Relationships

- `(:Person)-[:ACTED_IN {role:"some role"}]->(:Movie)`
- `(:Person)-[:DIRECTED]->(:Movie)`
- `(:Person)-[:WRITER_OF]->(:Movie)`
- `(:Person)-[:PRODUCED]->(:Movie)`
- `(:MOVIE)-[:HAS_GENRE]->(:Genre)`

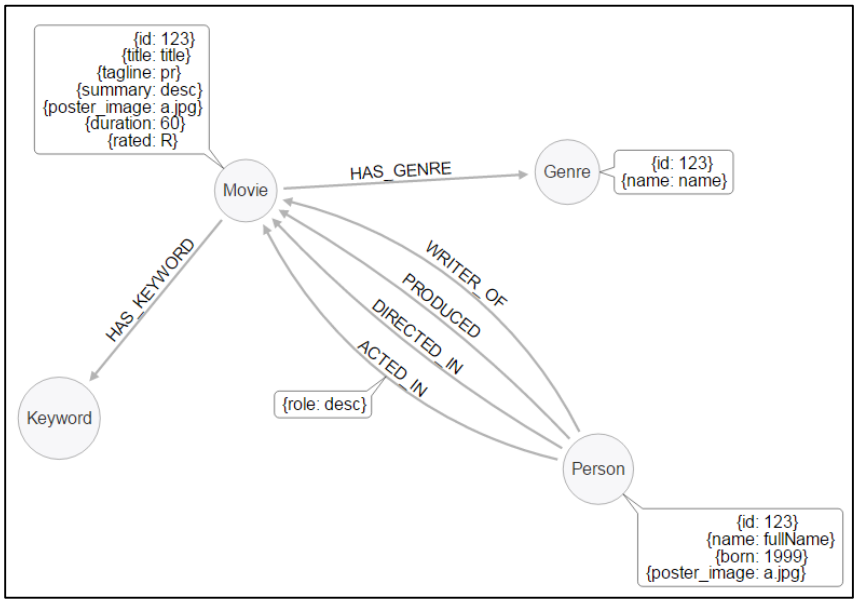
Database Setup

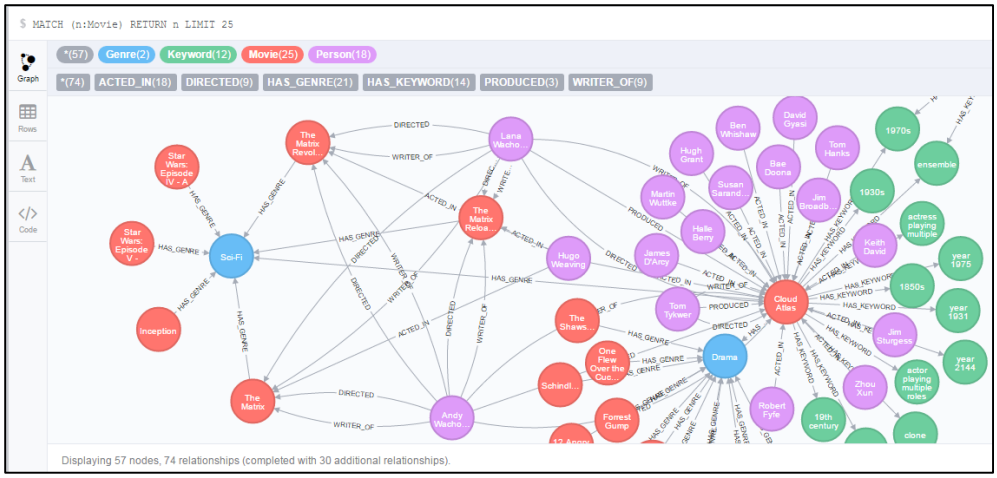
Import Data to Neo4j



[Download Neo4j 3.1.0 Enterprise Edition]

- Stop neo4j
- Delete **`%NEO4J_HOME%/data/database/graph.db`**
- Start neo4j again, you will have an empty database to start with
- Copy all the csv files into **`%NEO4J_HOME%/import`**
- Run the cypher script (see Appendix A)
- Stop the Neo4j server and zip up the graph.db directory so you have a backup, then restart the server and continue





If you see `Input error: Directory '…/data/databases/graph.db' already contains a database`, delete the `graph.db` directory and try again.

Start the Database!

- Start Neo4j if you haven't already!
- Set your username and password
- You should see a database populated with `Movie`, `Genre`, `Keyword`, and `Person` nodes.

Node API

From the root directory of this project:

- `cd api`
- `npm install`

```
$ npm install
npm WARN deprecated node-uuid@1.4.7: use uuid module instead
npm WARN deprecated jade@1.11.0: Jade has been renamed to pug, please install the latest version of pug instead of jade
npm WARN deprecated transformers@2.1.0: Deprecated, use jstransformer
[                  ..] | finalize:media-typer: sill finalize C:\code\Neo4j\code\neo4j-examples\neo4j-movies-template\neo4j-m
[                  ..] \ finalize:concat-map: sill finalize C:\code\Neo4j\code\neo4j-examples\neo4j-movies-template\neo4j-mov
[                  ..] - finalize:number-is-nan: sill finalize C:\code\Neo4j\code\neo4j-examples\neo4j-movies-template\neo4j-mov
neo4j-movies-api@0.0.20 C:\code\Neo4j\code\neo4j-examples\neo4j-movies-template\neo4j-movies-template\api
+-- body-parser@1.15.2
```

- in `config.js`, update the credentials for your database as needed
- `node app.js` starts the API

```
$ node app.js
You are using properties to be deprecated in v2.0.0
Please update to align with the swagger v2.0 spec.
[ 'definition' ]
You are using properties to be deprecated in v2.0.0
Please update to align with the swagger v2.0 spec.
[ 'definition' ]
You are using properties to be deprecated in v2.0.0
Please update to align with the swagger v2.0 spec.
[ 'definition' ]
You are using properties to be deprecated in v2.0.0
Please update to align with the swagger v2.0 spec.
[ 'definition' ]
Express server listening on port 3000 see docs at /docs
```

- Take a look at the docs at http://localhost:3000/docs

Alternative: Flask API

From the root directory of this project:

- `cd flask-api`
- `pip install -r requirements.txt` (you should be using a virtualenv)

```
$ python
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

```
$ pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
    100% |████████████████████████████████| 1.8MB 575kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.1.0
```

```
$ virtualenv pyenv360x64
Using base prefix 'c:\\bin\\python\\py360x64'
New python executable in C:\code\Neo4j\code\neo4j-examples\neo4j-movies-template\
neo4j-movies-template\flask-api\pyenv360x64\Scripts\python.exe
Installing setuptools, pip, wheel...done.
```

- `pyenv360x64\Scripts\activate`

```
$ pyenv360x64\Scripts\activate

(pyenv360x64) cmiller@CMILLER-SIM-LAP C:\code\Neo4j\code\neo4j-examples\neo4j-movies
-template\neo4j-movies-template\flask-api
```

- set your neo4j database username `set MOVIE_DATABASE_USERNAME=neo4j`
- set your neo4j database password `set MOVIE_DATABASE_PASSWORD=12345678`
- `set FLASK_APP=app.py`
- `flask run` starts the API

```
$ flask run
 * Serving Flask app "flask-api.app"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- Take a look at the docs at http://localhost:5000/docs

Frontend

Note: To get bower and gulp to work on Windows you may need to put npm in the path (in my case I also use nvm to manage node versions).

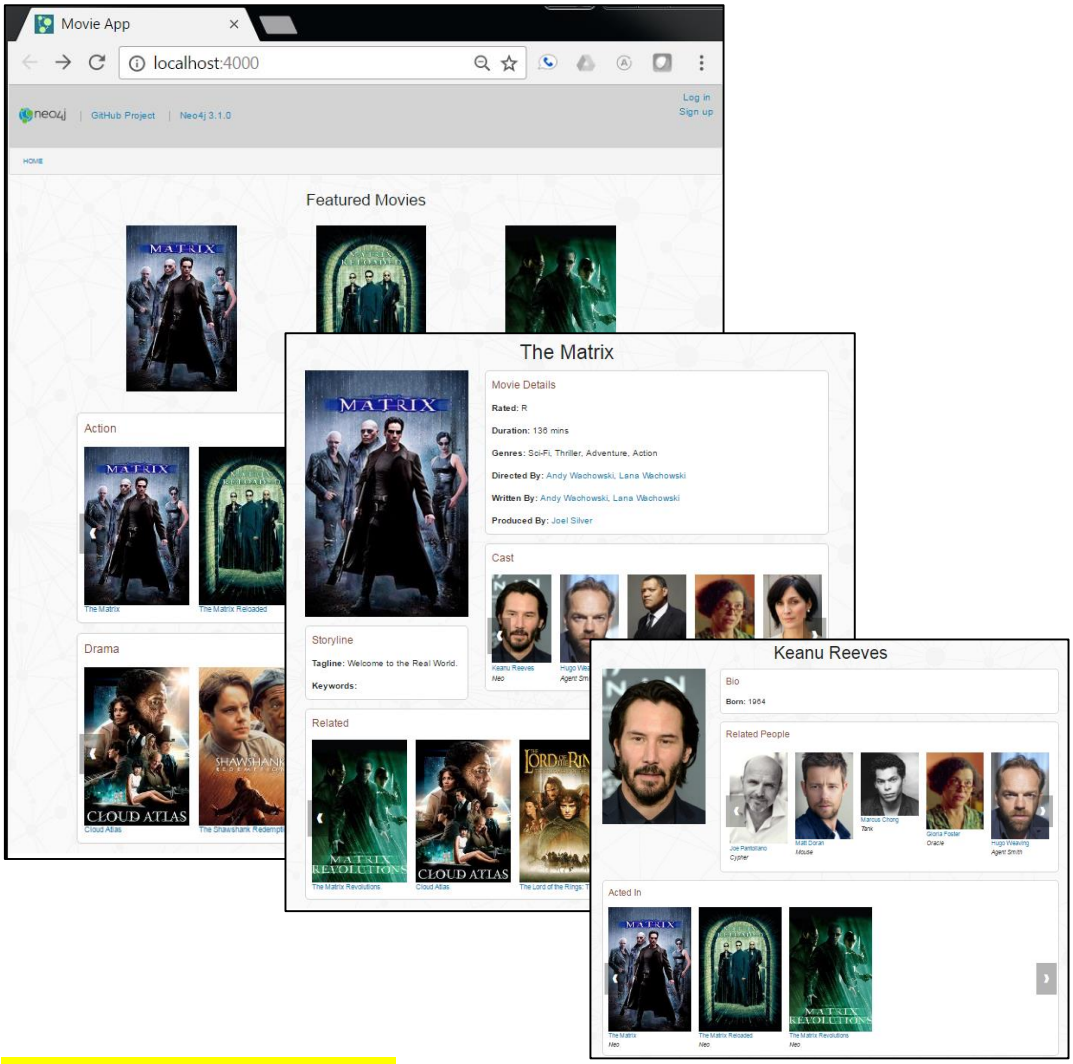
```
:: Node is managed by NVM-Windows via symlinks
set NVM_Home=%USERPROFILE%\AppData\Roaming\nvm
set NVM_SYMLINK=%BIN_DRIVE%\bin\NodeJS\NodeSymlink

:: NPM still needs to be added for Bower
set NPM_HOME=%USERPROFILE%\AppData\Roaming\npm

set "PATH=%NVM_HOME%;%NVM_SYMLINK%;%NPM_HOME%;%PATH%"
```

From the root directory of this project, set up and start the frontend with:

- `cd web`
- `npm install` (if `package.json` changed)
- `npm install -g bower`
- `bower install` to install the styles
- update config.settings.js file
  - if you are using the Node API: `cp config/settings.example.js config/settings.js`
  - if you are using the flask api then edit `config/settings.js` and change the `apiBaseURL` to `http://localhost:5000/api/v0`
- `npm install -g gulp`
- `gulp` starts the app on http://localhost:4000/



Netflix, eat your heart out ;-)

Ratings and Recommendations

Load some fake users and ratings

If you're running the app locally, you might want to tweak or explore ratings without having a robust community of users. In the `/csv` directory, note that there is a file called `ratings.csv`. This file contains some pseudo-randomly generated users and ratings. Load the users and ratings:
Move `ratings.csv` into the `import` directory of your database either by dragging and dropping or using

```
copy csv/ratings.csv $NEO4J_HOME/import/ratings.csv
```

put ratings.csv into the import directory

Assuming your database is running, paste the following query into the Neo4j browser:

```
LOAD CSV WITH HEADERS FROM 'file:///ratings.csv' AS line
MATCH (m:Movie {id:toInteger(line.movie_id)})
MERGE (u:User {id:line.user_id, username:line.user_username}) // user ids are strings
MERGE (u)-[r:RATED]->(m)
SET r.rating = toInt(line.rating)
RETURN m.title, r.rating, u.username
```

m.title	r.rating	u.username
Elysium	1	William
12 Angry Men	3	James
Pulp Fiction	3	Joseph
The Dark Knight	3	Joseph
The Lord of the Rings: The Fellowship of the Ring	3	Thomas
12 Years a Slave	3	Robert
12 Angry Men	1	Edward
The Matrix	3	David
The Matrix Reloaded	3	David
Dallas Buyers Club	5	David
Mr. Peabody & Sherman	5	David
The Avengers	5	David
The Silence of the Lambs	5	David
Capote	5	David
Lost in Translation	5	David
3 Days to Kill	3	Louis

If you don't want to use the browser, you can uncomment out the above query in `setup.cql` and run it again using `$NEO4J_HOME/bin/neo4j-shell < setup.cql`

User-Centric, User-Based Recommendations

Based on my similarity to other users, user `Sherman` might be interested in movies rated highly by users with similar ratings as himself.

```
MATCH (me:User {username:'Sherman'})-[my:RATED]->(m:Movie)
MATCH (other:User)-[their:RATED]->(m)
WHERE me <> other
AND abs(my.rating - their.rating) < 2
WITH other,m
MATCH (other)-[otherRating:RATED]->(movie:Movie)
WHERE movie <> m
WITH avg(otherRating.rating) AS avgRating, movie
RETURN movie
ORDER BY avgRating desc
LIMIT 25
```

Movie-Centric, Keyword-Based Recommendations

Site visitors interested in movies like `Elysium` will likely be interested in movies with similar keywords.

```
MATCH (m:Movie {title:'Elysium'})
MATCH (m)-[:HAS_KEYWORD]->(k:Keyword)
MATCH (movie:Movie)-[r:HAS_KEYWORD]->(k)
WHERE m <> movie
WITH movie, count(DISTINCT r) AS commonKeywords
RETURN movie
ORDER BY commonKeywords DESC
LIMIT 25
```

User-Centric, Keyword-Based Recommendations

`Sherman` has seen many movies, and is looking for movies similar to the ones he has already watched.

```
MATCH (u:User {username:'Sherman'})-[:RATED]->(m:Movie)
MATCH (m)-[:HAS_KEYWORD]->(k:Keyword)
MATCH (movie:Movie)-[r:HAS_KEYWORD]->(k)
WHERE m <> movie
WITH movie, count(DISTINCT r) AS commonKeywords
RETURN movie
ORDER BY commonKeywords DESC
LIMIT 25
```

Contributing

Node.js/Express API

The Express API is located in the `/api` folder.

Create Endpoint

The API itself is created using the Express web framework for Node.js. The API endpoints are documented using swagger and swagger-jsdoc module.

To add a new API endpoint there are 3 steps:

1. Create a new route method in `/api/routes` directory
2. Describe the method with swagger specification inside a JSDoc comment to make it visible in swagger
3. Add the new route method to the list of route methods in `/api/app.js`.

Flask API

The flask API is located in the flask-api folder. The application code is in the `app.py` file.

Create Endpoint

The API itself is created using the Flask-RESTful library. The API endpoints are documented using swagger with the flask-restful-swagger-2 library.

To add a new API endpoint there are 3 steps:

1. Create a new Flask-RESTful resource class
2. Create an endpoint method including the swagger docs decorator.
3. Add the new resource to the API at the bottom of the file.

Appendix A: UploadAllCsvFiles Cypher Script

```
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Craig Miller
// 01/10/2017
// File   : UploadAllCsvFiles.cql
// Project: neo4j-movies-template
// Model  : 3 nodes: Person, Movie, Genre
//          6 relations
// Note   : 1. use CONSTRAINT to handle missing (also creates index)
//          2. small files so no need to chunk (USING PERIODIC COMMIT 1000)
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Neo4j 3.1.0 Enterprise server is restricted, by default, to import
// from the %NEO4J_HOME%/import directory, so copy all your csv file
// to that directory before running this script

// ================================================================
// clear the existing database (a better approach would be to delete
// the %NEO4J_HOME%/data/database/graph.db file)
MATCH (n)
WITH n LIMIT 10000
OPTIONAL MATCH (n)-[r]->()
DELETE n,r;

// ================================================================
// test
LOAD CSV WITH HEADERS FROM "file:///person_node.csv" AS r FIELDTERMINATOR ';'
WITH r LIMIT 10 WHERE r.`id:ID(Person)` IS NOT NULL
RETURN r.`id:ID(Person)`, r.name, r.`born:int`, r.poster_image

// test property array
LOAD CSV WITH HEADERS FROM "file:///acted_in_rels.csv" AS r FIELDTERMINATOR ';'
WITH r LIMIT 10
RETURN r.`:START_ID(Person)`, r.`:END_ID(Movie)`, SPLIT(r.role, '/')

// ================================================================
// Upload nodes

// ------------------------------------------------------

CREATE CONSTRAINT ON (g:Genre) ASSERT g.id IS UNIQUE;

// Added 1 constraint, statement completed in 127 ms

LOAD CSV WITH HEADERS FROM "file:///genre_node.csv" AS r FIELDTERMINATOR ';'
CREATE (g:Genre {
  id: toInteger(r.`id:ID(Genre)`),
  name: r.name
});

// Added 19 labels, created 19 nodes, set 38 properties, statement completed in 202 ms.

// ------------------------------------------------------
CREATE CONSTRAINT ON (k:Keyword) ASSERT k.id IS UNIQUE;

// Added 1 constraint, statement completed in 35 ms.

LOAD CSV WITH HEADERS FROM "file:///keyword_node.csv" AS r FIELDTERMINATOR ';'
CREATE (k:Keyword {
  id: toInteger(r.`id:ID(Keyword)`),
  name: r.name
});

// Added 3253 labels, created 3253 nodes, set 6506 properties, statement completed in 356 ms.
```

```
// ------------------------------------------------------
CREATE CONSTRAINT ON (m:Movie) ASSERT m.id IS UNIQUE;

// Added 1 constraint, statement completed in 44 ms.

LOAD CSV WITH HEADERS FROM "file:///movie_node.csv" AS r FIELDTERMINATOR ';'
CREATE (m:Movie {
  id: toInteger(r.`id:ID(Movie)`),
  title: r.title,
  tagline: r.tagline,
  summary: r.summary,
  poster_image: r.poster_image,
  duration: toInteger(r.`duration:int`),
  rated: r.rated
});

// Added 54 labels, created 54 nodes, set 378 properties, statement completed in 55 ms.

// ------------------------------------------------------
CREATE CONSTRAINT ON (p:Person) ASSERT p.id IS UNIQUE;

// Added 1 constraint, statement completed in 38 ms.

LOAD CSV WITH HEADERS FROM "file:///person_node.csv" AS r FIELDTERMINATOR ';'
CREATE (p:Person {
  id: toInteger(r.`id:ID(Person)`),
  name: r.name,
  born: toInteger(r.`born:int`),
  poster_image: r.poster_image
});

// Added 665 labels, created 665 nodes, set 2660 properties, statement completed in 54 ms.

// ------------------------------------------------------
// Movie-[:HAS_GENRE]->Genre

LOAD CSV WITH HEADERS FROM "file:///has_genre_rels.csv" AS r FIELDTERMINATOR ';'
MATCH (m:Movie {id: toInteger(r.`:START_ID(Movie)`)}), (g:Genre {id: toInteger(r.`:END_ID(Genre)`)})
CREATE (m)-[:HAS_GENRE]->(g);

// Created 152 relationships, statement completed in 198 ms.

// ------------------------------------------------------
// Movie-[HAS_KEYWORD]->Keyword

LOAD CSV WITH HEADERS FROM "file:///has_keyword_rels.csv" AS r FIELDTERMINATOR ';'
MATCH (m:Movie {id: toInteger(r.`:START_ID(Movie)`)}), (k:Keyword {id: toInteger(r.`:END_ID(Keyword)`)})
CREATE (m)-[:HAS_KEYWORD]->(k);

// Created 5118 relationships, statement completed in 607 ms.

// ------------------------------------------------------
// Person-[WRITER_OF]->Movie

LOAD CSV WITH HEADERS FROM "file:///writer_of_rels.csv" AS r FIELDTERMINATOR ';'
MATCH (p:Person {id: toInteger(r.`:START_ID(Person)`)}), (m:Movie {id: toInteger(r.`:END_ID(Movie)`)})
CREATE (p)-[:WRITER_OF]->(m);

// Created 107 relationships, statement completed in 18 ms.

// ------------------------------------------------------
// Person-[PRODUCED]->Movie

LOAD CSV WITH HEADERS FROM "file:///produced_rels.csv" AS r FIELDTERMINATOR ';'
MATCH (p:Person {id: toInteger(r.`:START_ID(Person)`)}), (m:Movie {id: toInteger(r.`:END_ID(Movie)`)})
CREATE (p)-[:PRODUCED]->(m);

// Created 60 relationships, statement completed in 13 ms.
```

```
// ------------------------------------------------------
// Person-[DIRECTED]->Movie

LOAD CSV WITH HEADERS FROM "file:///directed_rels.csv" AS r FIELDTERMINATOR ';'
MATCH (p:Person {id: toInteger(r.`:START_ID(Person)`)}), (m:Movie {id: toInteger(r.`:END_ID(Movie)`)})
CREATE (p)-[:DIRECTED]->(m);

// Created 65 relationships, statement completed in 111 ms.

// ------------------------------------------------------
// Person-[ACTED_IN]->Movie
// Note: relationship property 'role' is an array

LOAD CSV WITH HEADERS FROM "file:///acted_in_rels.csv" AS r FIELDTERMINATOR ';'
MATCH (p:Person {id: toInteger(r.`:START_ID(Person)`)}), (m:Movie {id: toInteger(r.`:END_ID(Movie)`)})
CREATE (p)-[:ACTED_IN{role:SPLIT(r.role, '/')}]->(m);

// Set 665 properties, created 665 relationships, statement completed in 187 ms.

// ------------------------------------------------------
```