Krótki przewodnik po pracy z bazą danych neo4j

Opracował: Artur Gramacki

1. Uwagi wstępne

W niniejszym opracowaniu omawiamy tylko najprostsze polecenia służące do pracy z obiektami bazy neo4j. Pomijamy np. całkowicie polecenia administratora służące do zarządzania bazą danych, pracy z klastrami, monitorowaniem stanu bazy itd . Baza neo4j mimo pozornej prostoty, jest całkiem złożonym systemem informatycznym. Tradycyjnie odsyłamy do dokumentacji źródłowej, która jest najbardziej kompletnym źródłowe wiedzy na ten temat https://neo4j.com/docs/

Uwaga: w ramkach, w kolorze czarnym, podajemy wydawane polecenia natomiast na **niebiesko** pokazujemy otrzymywane wyniki.

2. Instalacja i uruchomienie bazy serwera neo4j

Pracujemy z wersją **Community**. Ma ona sporo ograniczeń w porównaniu do wersji płatnej **Enterprise**. Aby jednak poznać zasady pracy z tą bazą grafową neo4j wersja bezpłatna w zupełności nam wystarczy. Szczegóły patrz https://neo4j.com/subscriptions/#editions

Posiłkując się wiadomościami z wykładu zainstaluj serwer neo4j i go uruchom. W konsoli powinieneś zobaczyć m.in. taki tekst, który świadczy o tym, że serwer uruchomił się poprawnie.

```
neo4j.bat console

Bolt enabled on localhost:7687.
Remote interface available at http://localhost:7474/
Started.
```

W bazą neo4j pracujemy z poziomu **przeglądarki WWW** lub z poziomy **konsoli tekstowej** o nazwie **cypher-shell**. Jednak tylko w przeglądarce mamy możliwość wizualizacji grafów. Niemniej jednak konsola tekstowa jest bardzo przydatna, gdy musimy wykonać wiele poleceń w trybie wsadowym lub po prostu wynik w postaci tekstowej nam wystarcza. Trzeba też zauważyć, że możliwość wizualizacji grafów w konsoli WWW (trzeba przyznać, że bardzo efektowna) ma wartość w zasadzie tylko poglądową i zupełnie nie nadaje się do prezentacji większych zestawów danych.

Na początek warto zapoznać się z krótka stroną pomocy dla konsoli tekstowej.

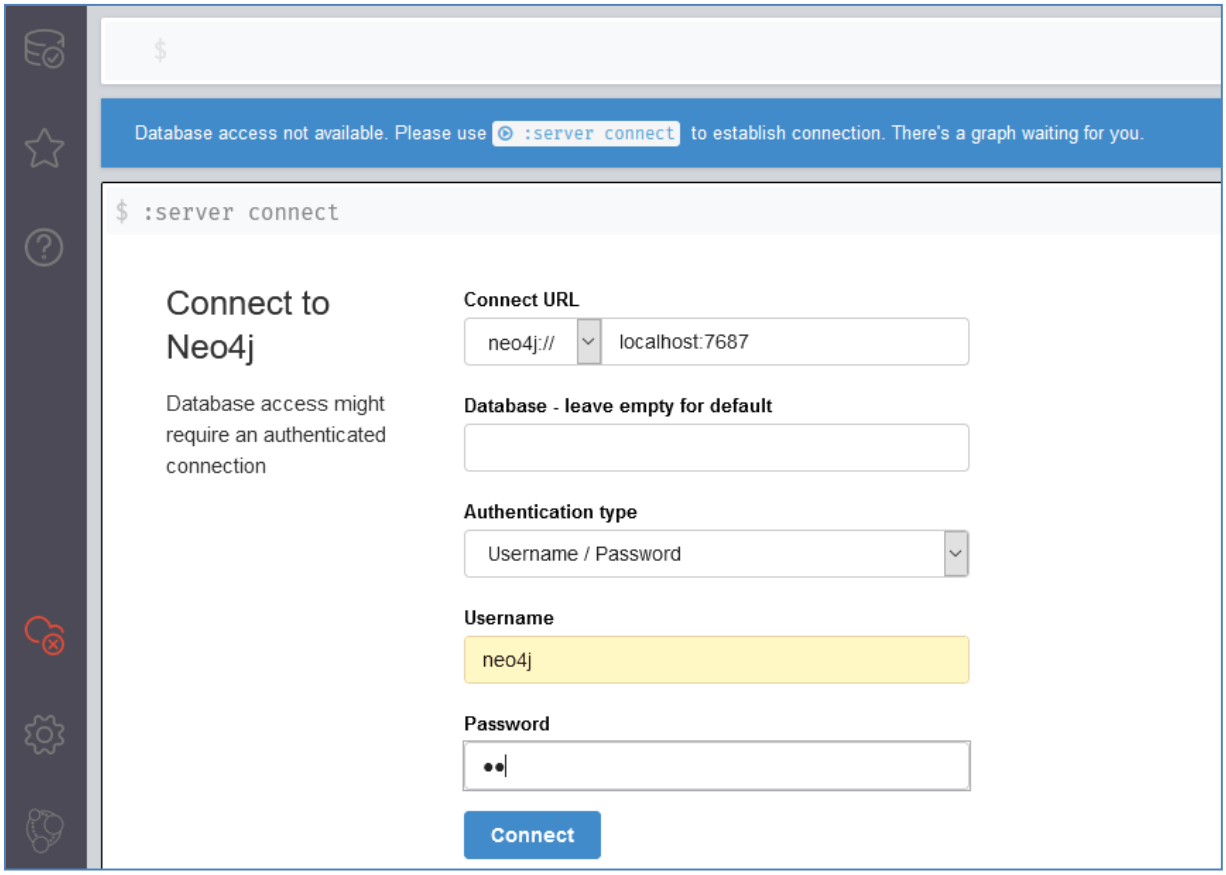
```
cypher-shell.bat -h
usage: cypher-shell [-h] [-a ADDRESS] [-u USERNAME] [-p PASSWORD]
```

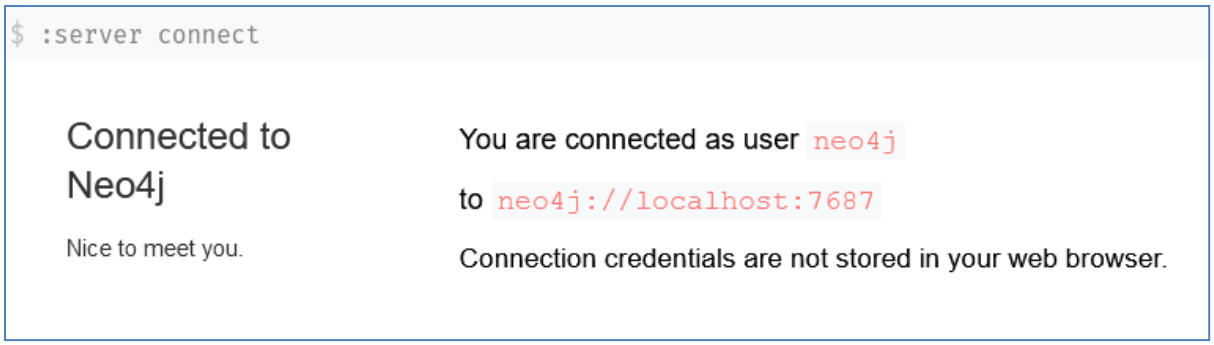
```
                    [--encryption {true,false,default}] [-d DATABASE]
                    [--format {auto,verbose,plain}] [-P PARAM] [--debug]
                    [--non-interactive] [--sample-rows SAMPLE-ROWS]
                    [--wrap {true,false}] [-v] [--driver-version] [-f FILE]
                    [--fail-fast | --fail-at-end] [cypher]

A command line shell where you can execute Cypher against an instance of
Neo4j. By default the shell is interactive but you can use it for
scripting by passing cypher directly on the command line or by piping a
file with cypher statements (requires Powershell on Windows).
```

Następnie możemy podłączyć się do serwera neo4j. Po przełączniku **-p** nie musimy wpisywać jawnym tekstem hasła. Możemy podać je później w czasie łączenia się do serwera.

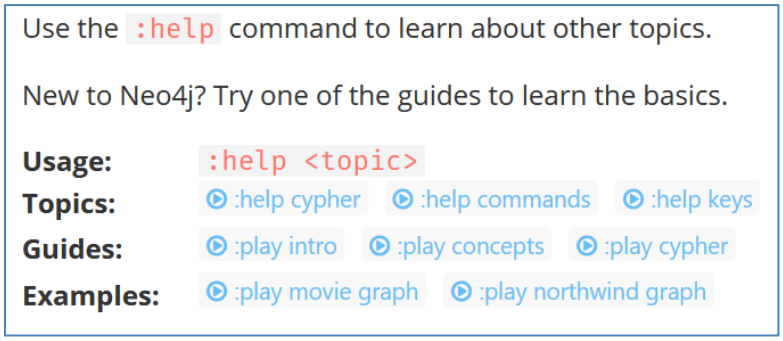
```
cypher-shell.bat -a //localhost:7687 -u neo4j -p ag -d neo4j

Connected to Neo4j using Bolt protocol version 4.2 at
neo4j://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j>
```

Po uruchomieniu klienta WWW (http://localhost:7474/) możemy połączyć się do serwera i rozpocząć pracę. Na początek mamy do dyspozycji tylko jedną bazę danych o nazwie neo4j oraz użytkownika też o nazwie neo4j. Przy pierwszym uruchomieniu zostaniemy poproszeni o zmianę pierwotnego hasła (też neo4j).



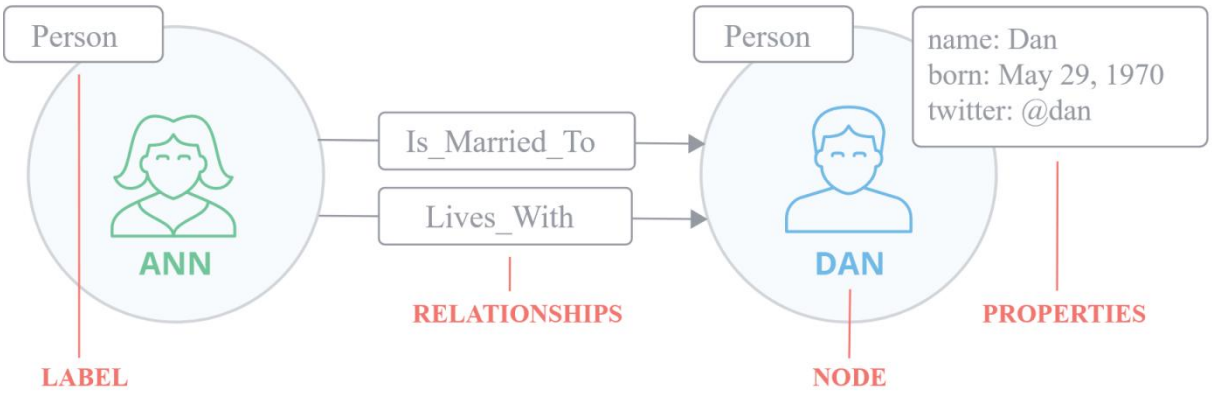


Na górze mamy pasek pleceń (ten ze znakiem `$`). Za jego pomocą możemy komunikować się z serwerem wydając polecenia w formacie „dwukropek + polecenie". Uruchom polecenie `:help` i zapoznaj się, na początku chociaż pobieżnie, z wszystkimi tematami (Topics, Guides, Examples). Przejście przez te wszystkie tematy daje naprawdę dobry obraz możliwości systemu Neo4j jako bazy grafowej oraz pozwala opanować technikalia pracy z nią.



Inne często wydawane polecenia to `:server connect` oraz `:clear`

3. Przypomnienie podstawowych pojęć



Polskie tłumaczenia:

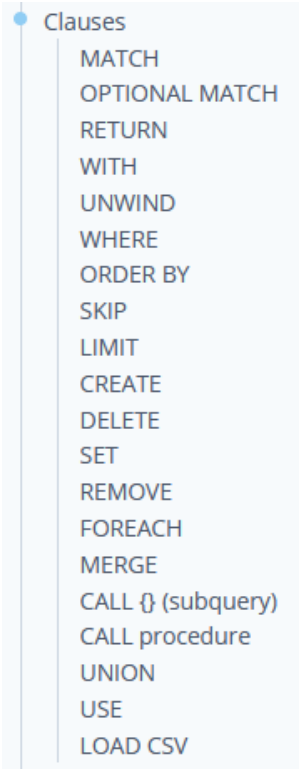
node - węzeł
label - etykieta
relationship -relacja
property - właściwość

4. Parę słów o języku Cypher

Komunikacja użytkownika z bazą neo4j odbywa się za pośrednictwem deklaratywnego języka o nazwie **Cypher**. Ponieważ baza neo4j przechowuje obiekty w strukturze grafu, więc język zapytań został tak zaprojektowany, aby takie struktury łatwo można było w nim odpytywać oraz manipulować nimi (tworzyć, modyfikować , kasować). Praca z językiem koncepcyjnie nieco przypomina pracę z językiem SQL, jednak szczegółowa składnia jest zasadniczo inna niż SQL-a.

Dokumentacja precyzyjnie wyjaśnia wszystkie szczegóły https://neo4j.com/docs/cypher-manual/current/, jest jednak dość obszerna. Natomiast najważniejsze informacje są przedstawione w bardzo przejrzysty sposób na tzw. Refcart https://neo4j.com/docs/cypher-refcard/current/.

W większości przypadków będziemy stosowali różne **klauzule**, z których najczęściej zapewne będzie używana klauzula **`MATCH`**.



5. Stworzenie najprostszego modelu (dwa węzły i dwie relacje)

Na początek warto poznać polecenie, które kasuje wszystkie obiekty (węzły i relacje) w bieżącej bazie danych. Wykonując różne ćwiczenia szybko dojdziemy do stanu „zabałaganionej" bazy. Wówczas wygodnie jest jednym poleceniem wykasować wszystko i zacząć tworzenie obiektów od nowa.

```
MATCH (n) DETACH DELETE n;

0 rows available after 20 ms, consumed after another 0 ms
Deleted 8 nodes, Deleted 9 relationships
```

Możemy też upewnić się, że rzeczywiście w naszej bazie nic już nie ma.

```
MATCH (n) RETURN n;
+---+
| n |
+---+
+---+

0 rows available after 0 ms, consumed after another 0 ms
```

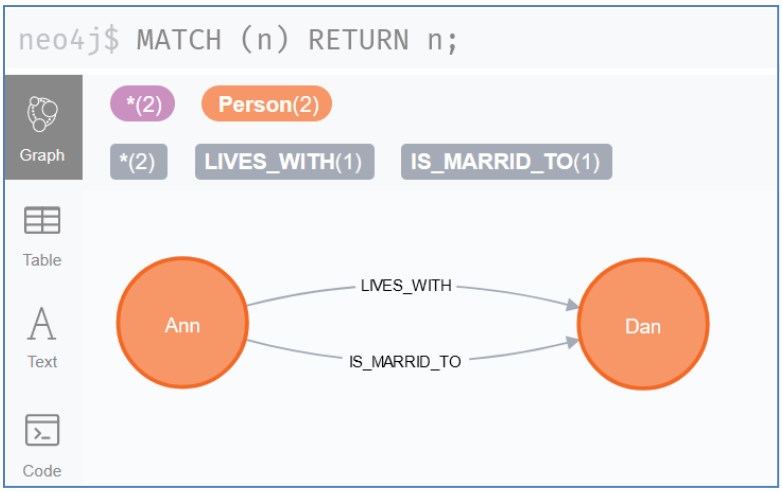
Stworzymy teraz dwa węzły i dwie relacje między nimi (jak na rysunku wyżej). Polecenia możemy wydawać albo w konsoli tekstowej albo (zapewne wygodniej) w konsoli WWW. W tej ostatniej mamy np. kolorowanie składni, co bardzo pomaga w niemyleniu się ☺

```
1 CREATE (ann:Person {name: 'Ann', born: '2000-09-30'})
2 CREATE (dan:Person {name: 'Dan', born: '1999-02-12'})
3 CREATE (ann)-[:IS_MARRID_TO {since: '2019-09-30'}]→(dan)
4 CREATE (ann)-[:LIVES_WITH {since: '2018-01-01'}]→(dan)
```

```
CREATE (ann:Person {name: 'Ann', born: '2000-09-30'})
CREATE (dan:Person {name: 'Dan', born: '1999-02-12'})
CREATE (ann)-[:IS_MARRID_TO {since: '2019-09-30'}]->(dan)
CREATE (ann)-[:LIVES_WITH {since: '2018-01-01'}]->(dan);

Added 2 nodes, Created 2 relationships, Set 6 properties, Added 2 labels
```

Możemy teraz zobaczyć co powstało. W konsoli zobaczymy tylko wynik w postaci tekstowej, natomiast w konsoli WWW będziemy mogli zobaczyć wizualizację. Poćwicz samodzielnie

```
MATCH (n) RETURN n;

+--------------------------------------------+
| n                                          |
+--------------------------------------------+
| (:Person {name: "Ann", born: "2000-09-30"}) |
| (:Person {name: "Dan", born: "1999-02-12"}) |
+--------------------------------------------+

2 rows available after 0 ms, consumed after another 0 ms
```



Kilka kolejnych poleceń pokazuje w jaki sposób można pracować z obiektami bazy neo4j. Dwa slash-e (`//`) to oczywiście sekwencja rozpoczynająca komentarz.

```
// Modyfikowanie właściwości węzła
MATCH (n {name: 'Dan'})
SET n.born = '1999-02-22'
RETURN n.name, n.born;

+----------------------+
| n.name | n.born      |
+----------------------+
| "Dan"  | "1999-02-22" |
+----------------------+

// Usuwanie właściwości węzła
MATCH (n {name: 'Dan'})
SET n.born = null
RETURN n.name, n.born;

+-----------------+
| n.name | n.born |
+-----------------+
| "Dan"  | NULL   |
+-----------------+

// Dodanie właściwości węzła
MATCH (p {name: 'Ann'})
SET p.email = 'ann@gmial.com';

MATCH (n) RETURN n;

+-------------------------------------------------------------------------+
| n                                                                       |
+-------------------------------------------------------------------------+
| (:Person {name: "Ann", email: "ann@gmial.com", born: "2000-09-30"}) |
| (:Person {name: "Dan"})                                                 |
+-------------------------------------------------------------------------+
```

Nie można wykasować węzła, gdy są zdefiniowane dla niego jakieś relacje. Próbujemy wykasować węzeł **Ann**, jednak system odpowiada błędem.

```
MATCH (n:Person {name: 'Ann'})
DELETE n;

Cannot delete node<16>, because it still has relationships. To delete this
node, you must first delete its relationships.
```

Trzeba więc najpierw usunąć relację (u nas dwie relacje) a potem dopiero usunąć węzeł.

```
MATCH (n {name: 'Ann'})-[r:LIVES_WITH]->()
DELETE r;
0 rows available after 16 ms, consumed after another 0 ms
Deleted 1 relationships

MATCH (n {name: 'Ann'})-[r:IS_MARRID_TO]->()
DELETE r;
0 rows available after 16 ms, consumed after another 0 ms
Deleted 1 relationships
```

```
MATCH (n:Person {name: 'Ann'})
DELETE n ;
0 rows available after 0 ms, consumed after another 0 ms
Deleted 1 nodes

MATCH (n) RETURN n;
+-------------------------+
| n                       |
+-------------------------+
| (:Person {name: "Dan"}) |
+-------------------------+
```

Jest też polecenie, które kasuje węzeł wraz ze wszystkimi relacjami. Domyślamy się jednak, że jest to polecenie dość niebezpieczne i trzeba je używać z rozwagą i zawsze świadomie (dlaczego?).

```
MATCH (n) RETURN n;
+-------------------------------------------+
| n                                         |
+-------------------------------------------+
| (:Person {name: "Ann", born: "2000-09-30"}) |
| (:Person {name: "Dan", born: "1999-02-12"}) |
+-------------------------------------------+

MATCH (n {name: 'Ann'})
DETACH DELETE n;
0 rows available after 16 ms, consumed after another 0 ms
Deleted 1 nodes, Deleted 2 relationships

MATCH (n) RETURN n;
+-------------------------------------------+
| n                                         |
+-------------------------------------------+
| (:Person {name: "Dan", born: "1999-02-12"}) |
+-------------------------------------------+
```

6. Modele demonstracyjne

Baza neo4j dostarczana jest z dwoma demonstracyjnymi modelami:

a) Movie Graph,
b) Northwind Graph.

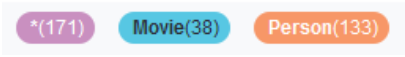
Pierwszy to baza danych **filmów**, **aktorów** oraz **reżyserów** oraz relacji między nimi. Rejestrowane są takie relacje jak: w jakim filmie grał jaki aktor (`ACTED_IN`) , kto reżyserował dany film (`DIRECTED`), kto był producentem (`PRODUCED`), autor scenariusz (`WROTE`), autor recenzji (`REVIEWED`) oraz polubienia (`FOLLOWS`).

Drugi model to przykład konwersji klasycznej bazy danych **Northwind** do formatu bazy grafowej. Baza Northwind została „całe wieki temu” stworzona przez Microsoft i na początku była udostępniona w bazie danych Access. Z czasem stała się bardzo popularna i wykorzystywana była w wielu różnych tutorialach, demonstracjach, szkoleniach itd. Została przeniesiona do wielu innych systemów baz relacyjnych. Z poziomy interfejsu WWW mamy gotowe skrypty tworzące obie demonstracyjne

bazy. Dane dla bazy Northwind Graph pobierane są najpierw z plików csv i następnie tworzone są odpowiednimi poleceniami odpowiednie węzły i relacje. Tym modelem nie będziemy zajmować się w tym tutorialu. Osoby zainteresowane mogą wydać w konsoli WWW polecenie `:play northwind-graph` aby przywołać ten przykład.

W bazie Movie Graph zdefiniowano:

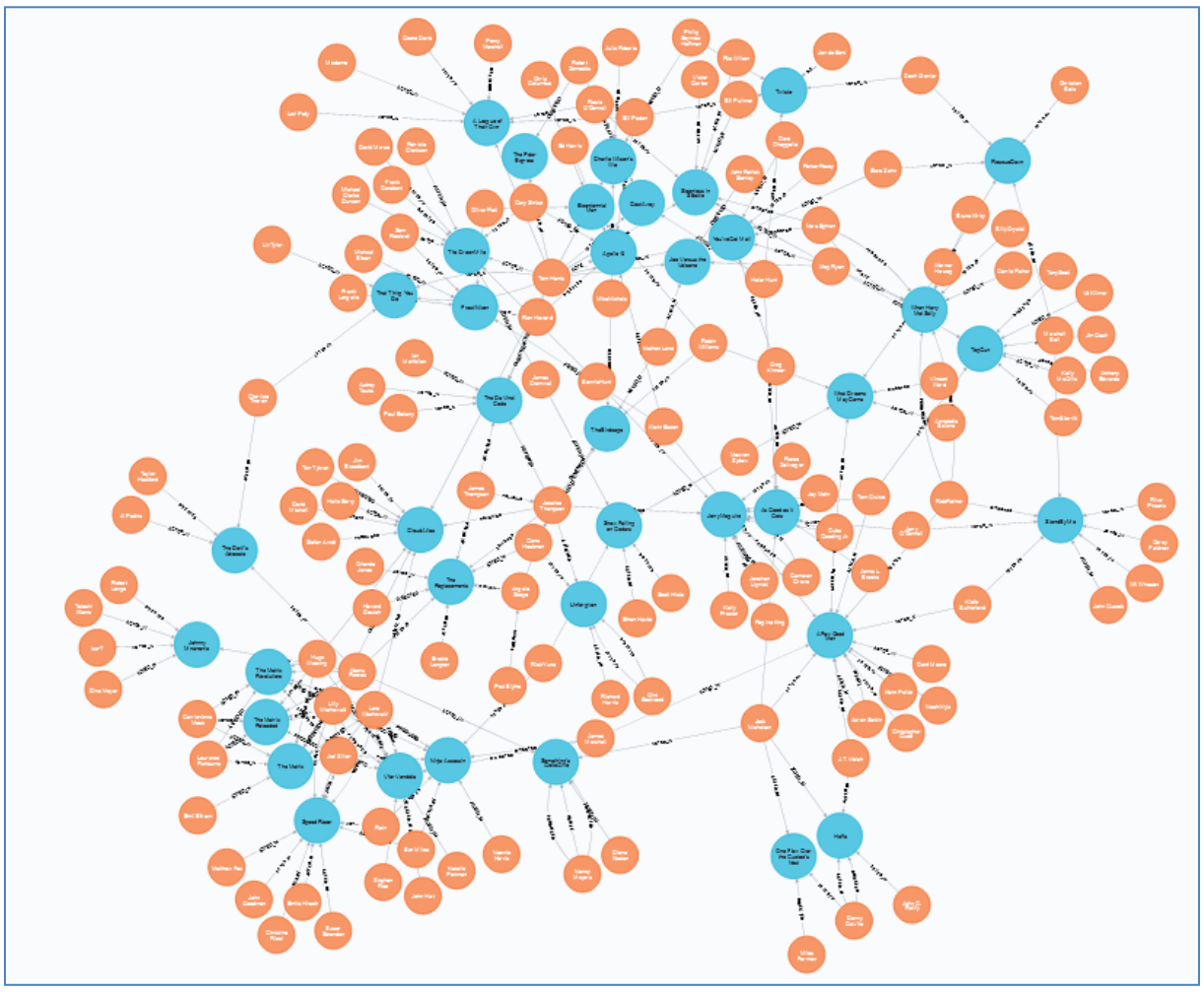
a) 171 węzłów (38 filmów oraz 133 osoby),



b) 253 relacje,

c) 564 właściwości

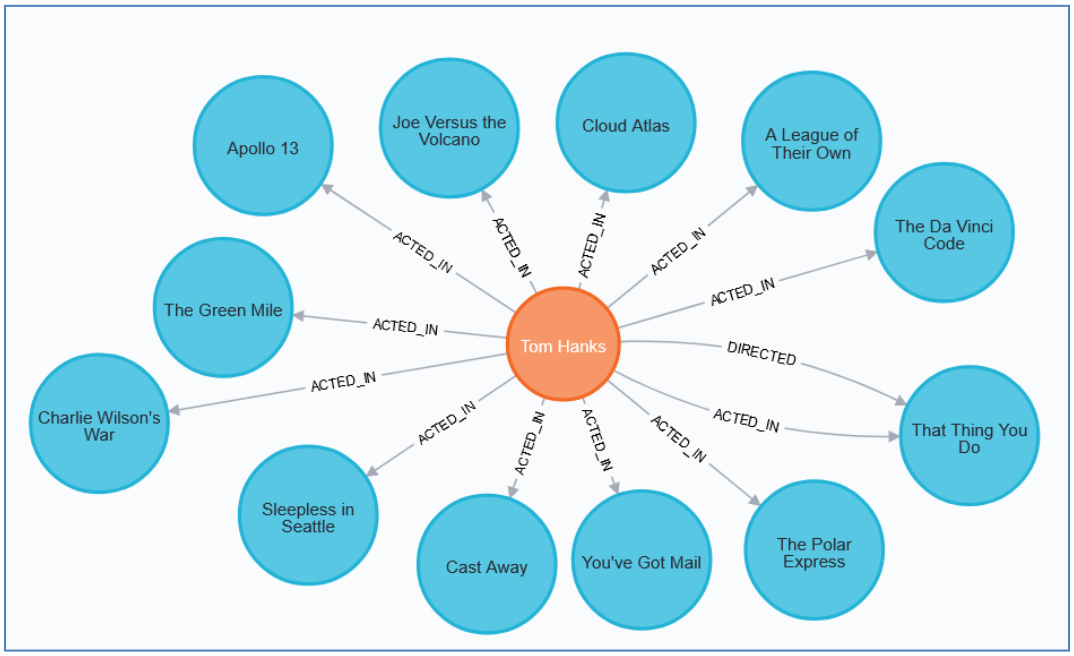
Jest to więc dość duży model, jak na potrzeby początkującego użytkownika bazy neo4j. Cały model można wyświetlić w interfejsie WWW, ale w zasadzie tylko do celów poglądowych jako swego rodzaju „bajer”. Aby odczytać z niego jakieś sensowne dane należy pracować z większymi powiększeniami i raczej nie wyświetlać wszystkich obiektów. Poniżej zielone kółka oznaczają filmy a pomarańczowe osoby.



Chcąc na przykład wyświetlić wszystkie filmy, w których grał Tom Hanks wydajemy poniższe polecenie i otrzymujemy wynik w postaci tekstowej oraz jego wizualizację graficzną.

```
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(tomHanksMovies)
RETURN tom, tomHanksMovies
```

```
+-------------------------------------------------------------------------+
| tom                                          | tomHanksMovies.title     |
+-------------------------------------------------------------------------+
| (:Person {name: "Tom Hanks", born: 1956}) | "You've Got Mail"          |
| (:Person {name: "Tom Hanks", born: 1956}) | "The Polar Express"        |
| (:Person {name: "Tom Hanks", born: 1956}) | "A League of Their Own"    |
| (:Person {name: "Tom Hanks", born: 1956}) | "The Da Vinci Code"        |
| (:Person {name: "Tom Hanks", born: 1956}) | "Apollo 13"                |
| (:Person {name: "Tom Hanks", born: 1956}) | "Joe Versus the Volcano"   |
| (:Person {name: "Tom Hanks", born: 1956}) | "The Green Mile"           |
| (:Person {name: "Tom Hanks", born: 1956}) | "Charlie Wilson's War"     |
| (:Person {name: "Tom Hanks", born: 1956}) | "Cloud Atlas"              |
| (:Person {name: "Tom Hanks", born: 1956}) | "Cast Away"                |
| (:Person {name: "Tom Hanks", born: 1956}) | "That Thing You Do"        |
| (:Person {name: "Tom Hanks", born: 1956}) | "Sleepless in Seattle"     |
+-------------------------------------------------------------------------+
```



7. Movie Graph, wybrany fragment

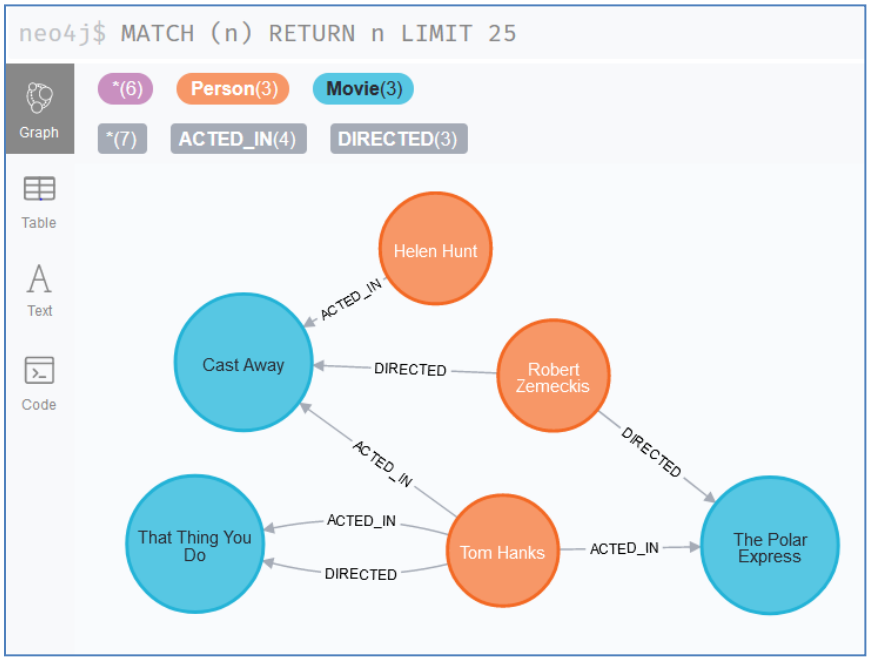
Z modelu demonstracyjnego do dokładniejsze analizy wybieramy jego mały fragment skryptu. Kasujemy wszystkie obiekty, jakie mamy w bazie danych i wykonujemy następujące polecenia. Powstaje 6 węzłów (3 osoby + 3 filmy) oraz 7 relacji

```
CREATE (HelenH:Person {name:'Helen Hunt', born:1963})
CREATE (TomH:Person {name:'Tom Hanks', born:1956})
CREATE (RobertZ:Person {name:'Robert Zemeckis', born:1951})
```

```
CREATE (CastAway:Movie {title:'Cast Away', released:2000,
        tagline:'At the edge of the world, his journey begins.'})
CREATE (ThatThingYouDo:Movie {title:'That Thing You Do', released:1996,
        tagline:'In every life ...'})
CREATE (ThePolarExpress:Movie {title:'The Polar Express', released:2004,
        tagline:'This Holiday Season... Believe'})

CREATE
       (TomH)-[:ACTED_IN {roles:['Chuck Noland']}]->(CastAway),
       (HelenH)-[:ACTED_IN {roles:['Kelly Frears']}]->(CastAway),
       (RobertZ)-[:DIRECTED]->(CastAway),
       (TomH)-[:ACTED_IN {roles:['Mr. White']}]->(ThatThingYouDo),
       (TomH)-[:DIRECTED]->(ThatThingYouDo),
       (TomH)-[:ACTED_IN {roles:['Hero Boy', 'Father', 'Conductor', 'Hobo',
              'Scrooge', 'Santa Claus']}]->(ThePolarExpress),
       (RobertZ)-[:DIRECTED]->(ThePolarExpress);

Added 6 nodes, Created 7 relationships, Set 19 properties, Added 6 labels
```

Powstały model możemy pokazać w postaci tekstowej oraz graficznej

```
MATCH (n) RETURN n;

+-------------------------------------------------------------------------------------------------------------------+
| n                                                                                                                 |
+-------------------------------------------------------------------------------------------------------------------+
| (:Person {name: "Helen Hunt", born: 1963})                                                                        |
| (:Person {name: "Tom Hanks", born: 1956})                                                                         |
| (:Person {name: "Robert Zemeckis", born: 1951})                                                                   |
| (:Movie {tagline: "At the edge of the world, his journey begins.", title: "Cast Away", released: 2000})           |
| (:Movie {tagline: "In every life ...", title: "That Thing You Do", released: 1996})                               |
| (:Movie {tagline: "This Holiday Season... Believe", title: "The Polar Express", released: 2004})                  |
+-------------------------------------------------------------------------------------------------------------------+
```



Na koniec pokazujemy kilka przykładowych zapytań do utworzonego modelu.

```
// Zwraca tytuły wszystkich filmów
MATCH (movie:Movie) RETURN movie.title;
```

```
+---------------------+
| movie.title         |
+---------------------+
| "Cast Away"         |
| "That Thing You Do" |
| "The Polar Express" |
+---------------------+

// Zwraca wszystkie filmy reżyserowane przez Roberta Zemeckisa
// The symbol -- means related to, without regard to type or
// direction of the relationship.
MATCH (director {name: 'Robert Zemeckis'})--(movie) RETURN movie.title;

+---------------------+
| movie.title         |
+---------------------+
| "The Polar Express" |
| "Cast Away"         |
+---------------------+

// Zwraca wszystkie węzły połączone z :Person równą 'Tom Hanks',
// które są filmami (:Movie)
// Innymi słowy: zwraca wszystkie filmy, gdzie brał udział Tom Hanks
// (bez znaczenia, jaką pełnił tam rolę)
MATCH (:Person {name: 'Tom Hanks'})--(movie:Movie) RETURN movie.title;

+---------------------+
| movie.title         |
+---------------------+
| "The Polar Express" |
| "That Thing You Do" |
| "That Thing You Do" |
| "Cast Away"         |
+---------------------+

Jak wyżej ale tym razem jawnie wskazujemy kierunek relacji
MATCH (:Person {name: 'Tom Hanks'})-->(movie) RETURN movie.title;

+---------------------+
| movie.title         |
+---------------------+
| "The Polar Express" |
| "That Thing You Do" |
| "That Thing You Do" |
| "Cast Away"         |
+---------------------+

// Tym razem zwracamy rodzaj relacji (ACTED_IN lub DIRECTED)
(:Person {name: 'Tom Hanks'})-[r]->(movie) RETURN type(r);

+------------+
| type(r)    |
+------------+
| "ACTED_IN" |
| "DIRECTED" |
| "ACTED_IN" |
| "ACTED_IN" |
+------------+

// Zwraca wszystkich aktorów, którzy grali w filmie 'Cast Away'
```

```
MATCH (CastAway:Movie {title: 'Cast Away'})<-[:ACTED_IN]-(actor)
RETURN actor.name;

+--------------+
| actor.name   |
+--------------+
| "Helen Hunt" |
| "Tom Hanks"  |
+--------------+
```

8. Polecenia do samodzielnego wykonania

Przygotuj samodzielnie skrypt, który utworzy:

a) około 12-15 węzłów,
b) około 20-25 relacji,
c) sensowne 2-3 etykiety dla każdego węzła,
d) sensowne 2-3 właściwości dla każdego węzła oraz relacji,
e) sensowne 2-3 właściwości dla każdej relacji.

Dane mogą być oczywiście **fikcyjne**, jednak muszą być **realistyczne**. Zanim rozpoczniesz realizację zadań **skonsultuj swój wybór z prowadzącym, który zatwierdzi go** (lub poprosi o wprowadzenie poprawek).

Wykonaj przynajmniej 10 sensownych zapytań do utworzonej struktury grafowej. Użyj też **funkcji agregujących**. Nie było to omawiane w tutorialu, samodzielnie odszukaj odpowiedni fragment w dokumentacji i przyswój sobie to zagadnienie.

Wykonaj przynajmniej 10 sensownych poleceń **zmieniających dane** (modyfikacja, dodawanie, kasowanie).

Przygotuj również dane w postaci **plików csv** i zademonstruj, jak można do neo4j załadować ich zawartość (klauzula `LOAD CSV FROM`). Możesz wzorować się na demonstracyjnej bazie Northwind Graph, gdzie zastosowano takie właśnie podejście.