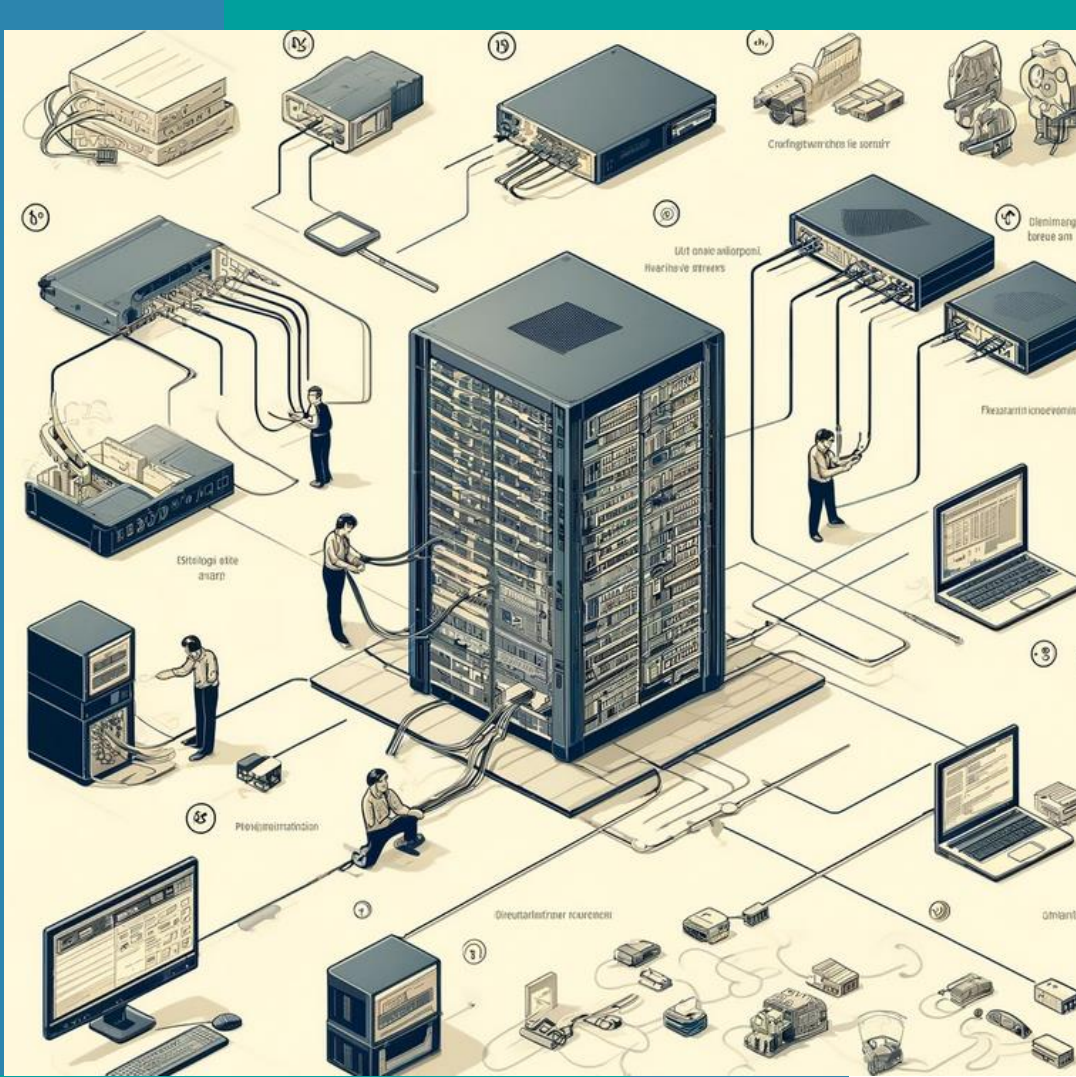




Przygotowanie środowiska
pracy systemu zarządzania
bazą danych Neo4j Desktop

ANALIZA DANYCH
Z WYKORZYSTANIEM BAZY
GRAFOWEJ NEO4J
I ALGORYTMÓW GRAFOWYCH



Zagadnienia

- [Instalacja Neo4j](#)
- [Konfiguracja środowiska pracy](#)
- [Zapytania Cypher](#)
- [Wyodrębnienie właściwości jako węzłów](#)

[Neo4j](#)[Products](#)[Use Cases](#)[Developers & Data Scientists](#)[Learn](#)[Pricing](#)[Contact Us](#)[Get Started Free](#)[Aura Login](#)[Partners](#)[Company](#)[Support](#)

FASTEST PATH TO GRAPH APPLICATION DEVELOPMENT

Neo4j Developer Tools

Neo4j provides advanced tools to make learning and development of graph applications easier. Free to use and run anywhere.

[NEW! Data Importer Tool](#) →



Neo4j Desktop

Experience Neo4j on Your Local Desktop. For Free.

[Download](#)[Learn More](#)



Neo4j Browser

Explore Your Database and Craft the Perfect Cypher Query.

[Learn More](#)

<https://neo4j.com/product/developer-tools/>

[Neo4j](#)[Products](#)[Use Cases](#)[Developers & Data Scientists](#)[Learn](#)[Pricing](#)[Contact Us](#)[Get Started Free](#)[Aura Login](#)[Partners](#)[Company](#)[Support](#)

Download Neo4j Desktop

Experience Neo4j on Your Desktop

Free. Get Started Today.

[Download](#) ↓

Includes Neo4j Enterprise 5.19.0 for Developers
[Learn more](#) | [System Requirements](#)

Are you a Startup?

Get started on AuraDB for Free today and avoid the overhead costs of self-hosted deployment. [Neo4j Startup Program](#)

Get Started Now

Please fill out this form to begin your download

By submitting this form, you confirm you are acting for your business/organisation.

☐ Yes! Please keep me updated on the latest about graph database products, services & events.

By downloading you agree to the [Neo4j License Agreement](#) for Neo4j Desktop Software.

Download Desktop

Formularz nie jest obowiązkowy

Wideo instrukcja

<https://youtu.be/hlvNexwVYNw?si=2cL90bReXTkuq18Z>

Podręcznik instalacji

<https://neo4j.com/download-thanks-desktop/?edition=desktop&flavour=winstall64&release=1.5.9&offline=true#installation-guide>

Klucz aktywacyjny

Thanks for downloading Neo4j Desktop

Your download should begin automatically in a few seconds. If it doesn't, Click one of the links : [Windows](#) · [OSX](#) · [Linux](#)

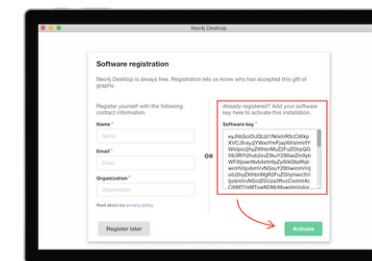
Recommended system requirements: MacOS 10.10 (Yosemite)+, Windows 8.1+ with Powershell 5.0+, Ubuntu 12.04+, Fedora 21, Debian 8.

Neo4j Desktop Activation Key

Use this key to activate your copy of Neo4j Desktop for use.

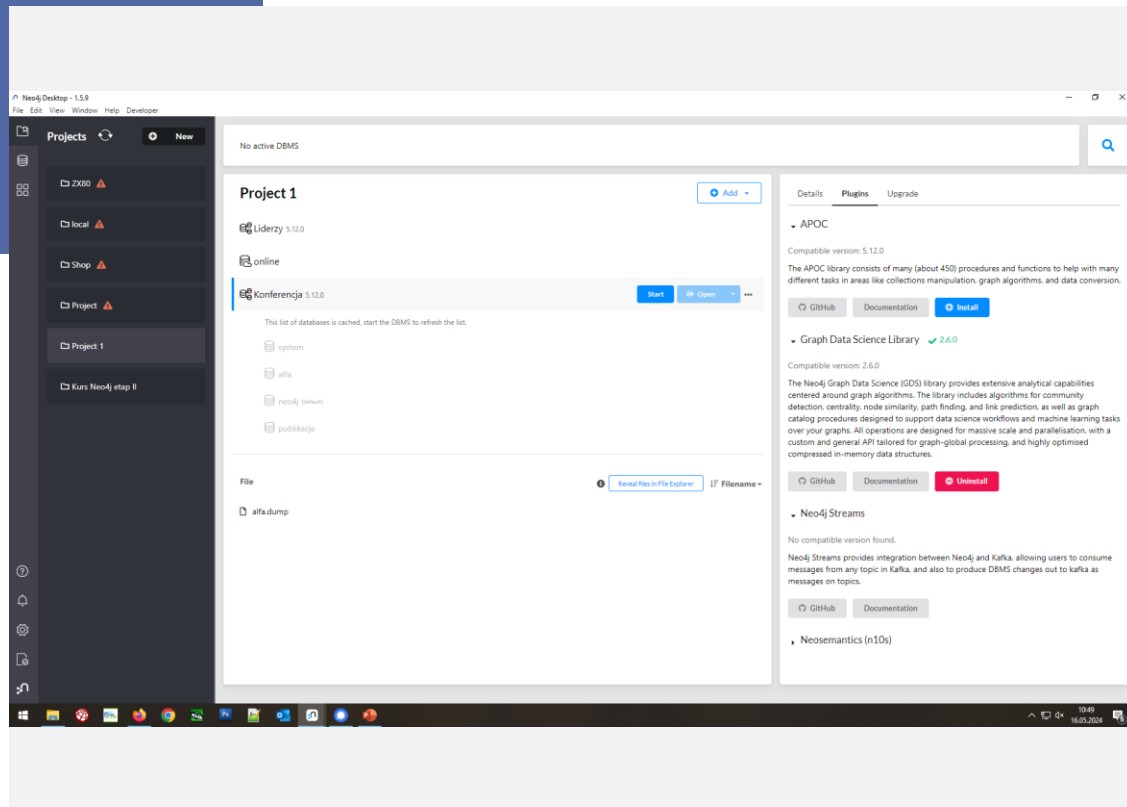
 Copy to clipboard

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFPbC16InB2d2sb2sYUbnbWFpbC5jb20iLCJtaXhwYW5lbElkljoiJGRldmJZToxODg5MDg4NmNhZjZlZiYyZGxMDFhM2I4YTgtZDU2NTQyOS0xZmE0MDAtMTg0OTA4ODZjYyY2ZTgiLCJtaXhwYW5lbFByb2p1Y3RjZC16IjRiZmlyNDE0YWI5NzNjNzQxYjZmMDY3YmYwNmQ1NTc1Iiwib3JnIjoic3MlLCJwdWl0eUZW80a15jb20iLCJyZWciOiJhZGFiIGFzbnRlIiwic3VlloibmVvNGotZGVza3RvcClsmV4cCI6MTc0NzE0NDY4NSwidmVljoilKilsmIzcyI6Im5lbzRqLmNvbSIsIm5lZi6MTcxNTg0ODY4NSwiaWF0IjojNzE0ODQ4Njg1LCJqdGkiOiJ0dEloNlg5RXl1fQ.J0f8Yyug8blY0BHQ3G0j9SnMfc5k01xzcksb5SxiXFXU5L5aXEJE5IHdYaswUuoEKivLFpNyFSyJFMQtbUSNfsxqpWHZ6aPcNi3RWKI-ujopU6ZGhxpkK2PbH09io-4zjofvJgB2xi6l9ROXPsWznxaqKz39W-T_8udGplAqpv2dyvO-yQHKG2iHQJ1RP4b5-DRhjz6m2wDdn6Fox5rqKNWtpil-cpQ69QEuZMShBddwk8qyrzGQ0WJGNubaxjZT9XRV4ggFcaZVr
```



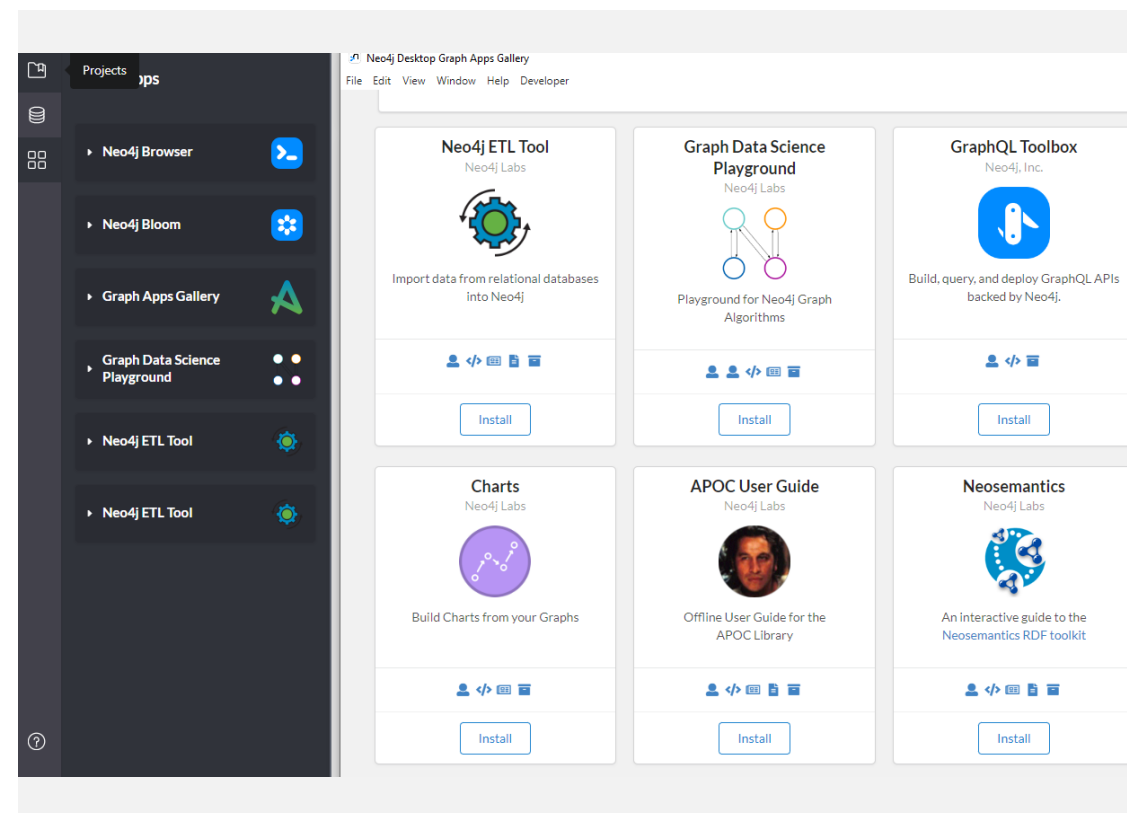
 Installation Video

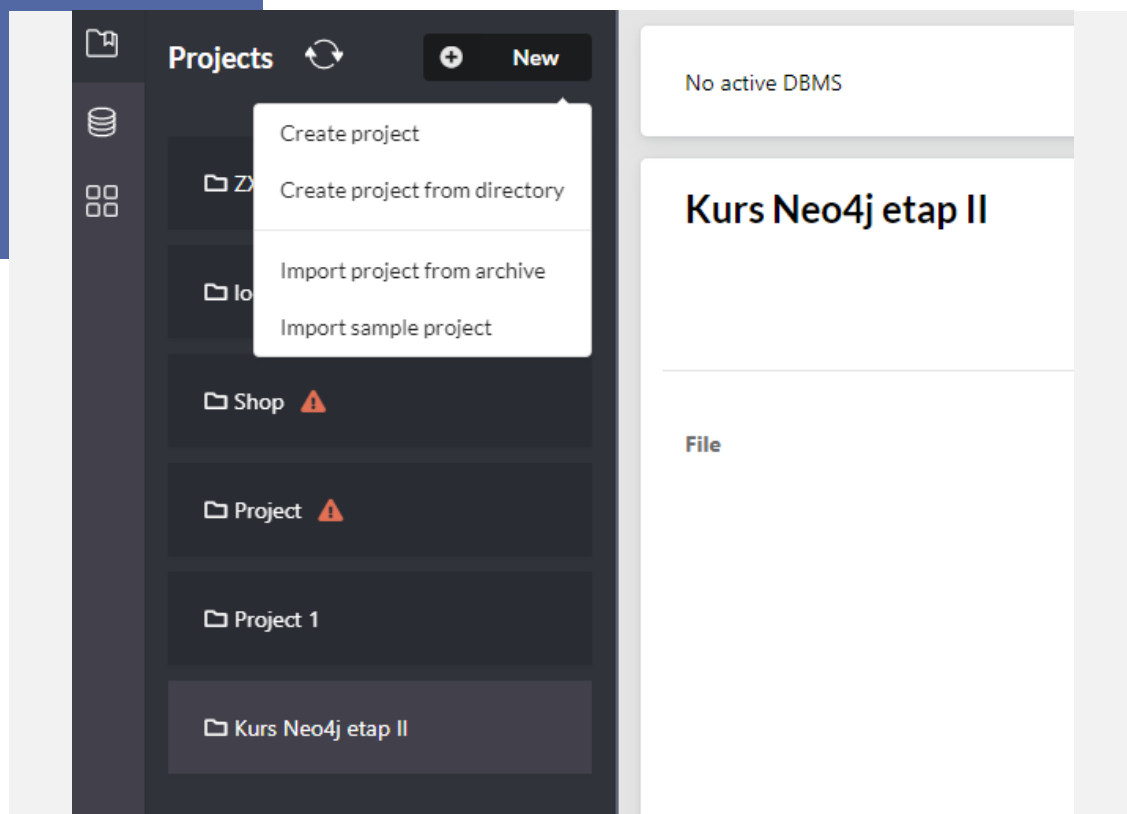
 Installation Guide



Zarządzanie projektami

Moduły / aplikacje / dodatki

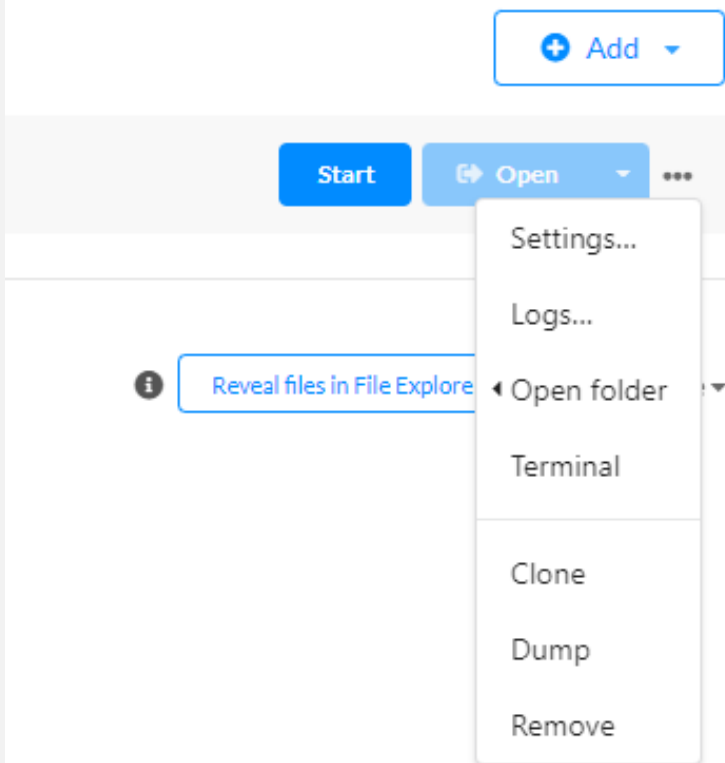




**Tworzymy projekt
„Kurs Neo4j etap II”**

Tworzymy DBMS (System Zarządzania Bazą Danych) „System 1”

A screenshot of the 'Kurs Neo4j etap II' DBMS creation form. The form has fields for 'Name' (containing 'System 1'), 'Password' (containing 'password'), and 'Version' (set to '5.12.0'). A dropdown menu is open next to the 'Add' button, showing options: 'Local DBMS', 'Remote connection', and 'File'. At the bottom, there are buttons for 'Cancel' and 'Create'.



Opcje związane z bazą

Instalacja Plugin'ów

Details

Plugins

Upgrade

▼ APOC ✓ 5.12.0

Compatible version: 5.12.0

The APOC library consists of many (about 450) procedures and functions to help with many different tasks in areas like collections manipulation, graph algorithms, and data conversion.

GitHub

Documentation

Uninstall

▼ Graph Data Science Library ✓ 2.6.6

Compatible version: 2.6.6

The Neo4j Graph Data Science (GDS) library provides extensive analytical capabilities centered around graph algorithms. The library includes algorithms for community detection, centrality, node similarity, path finding, and link prediction, as well as graph catalog procedures designed to support data science workflows and machine learning tasks over your graphs. All operations are designed for massive scale and parallelisation, with a custom and general API tailored for graph-global processing, and highly optimised compressed in-memory data structures.

GitHub

Documentation

Uninstall

▸ Neo4j Streams

▸ Neosemantics (n10s)

Neo4j Desktop is a control center to manage your local and remote Neo4j graphs in one place. You can [download and install it](#), and [learn more](#).

Graph Apps

The apps below are additions to Neo4j Desktop that provide new capabilities, like Monitoring, Import, Analysis, Running Graph Algorithms

Neo4j ETL Tool

Neo4j Labs



Import data from relational databases into Neo4j



Install

Graph Data Science Playground

Neo4j Labs



Playground for Neo4j Graph Algorithms



Install

GraphQL Toolbox

Neo4j, Inc.



Build, query, and deploy GraphQL APIs backed by Neo4j.



Install

Charts

Neo4j Labs



APOC User Guide

Neo4j Labs



Neosemantics

Neo4j Labs



Startujemy 😊

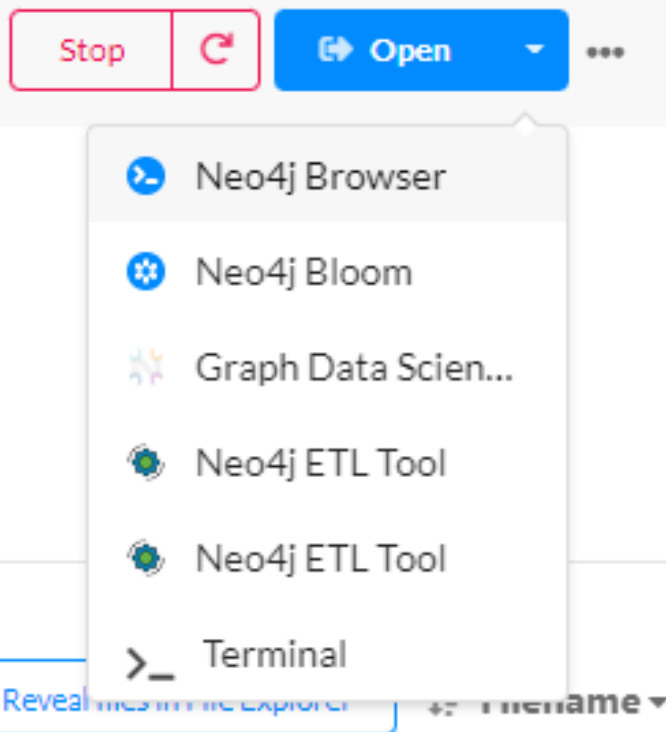
+ Add ▼

Start

➡ Open ▼



Plugin Graph Data Science Playground

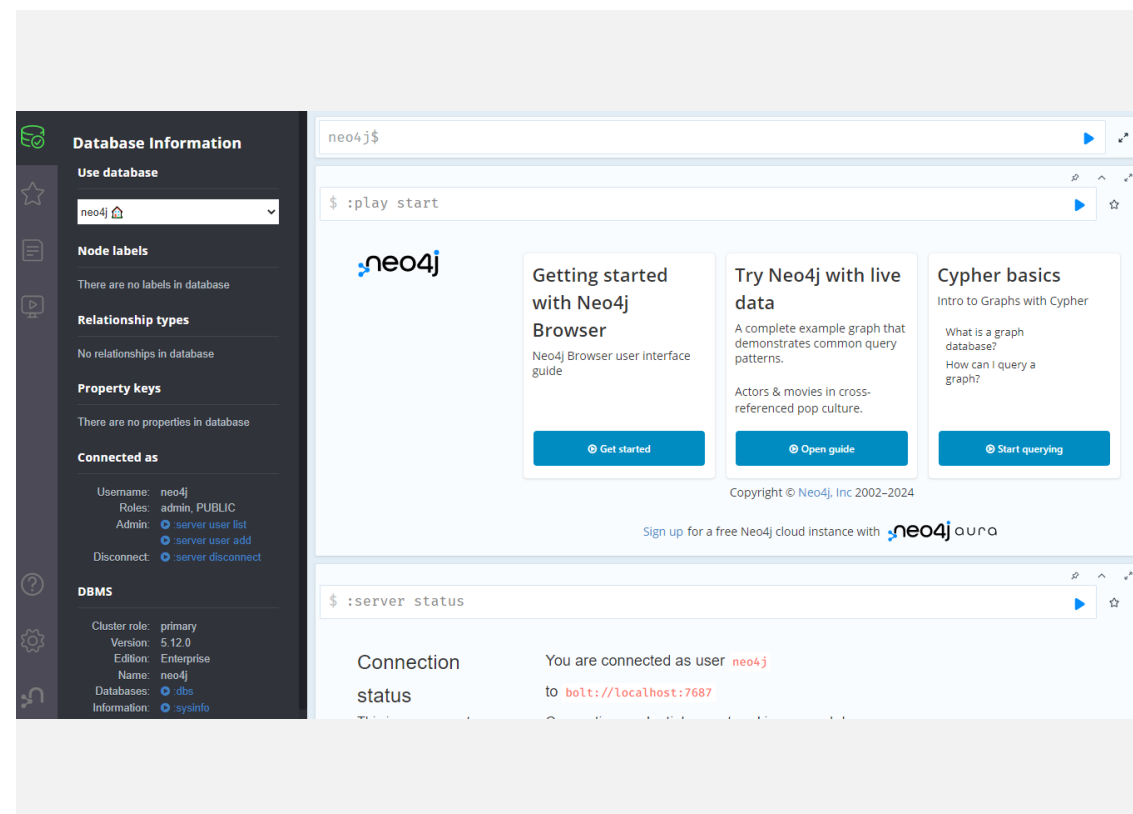


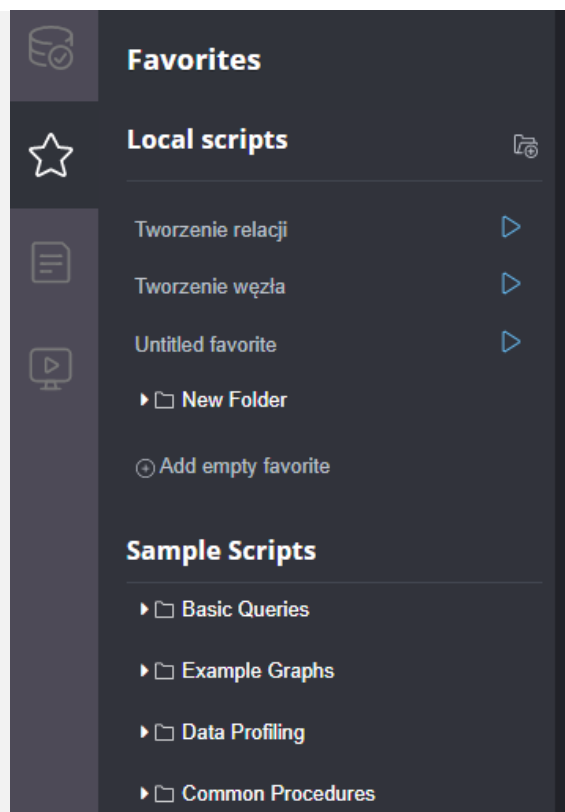
Różne konteksty uruchomienia DBMS

Desktop manual

<https://neo4j.com/docs/desktop-manual/current/visual-tour/>

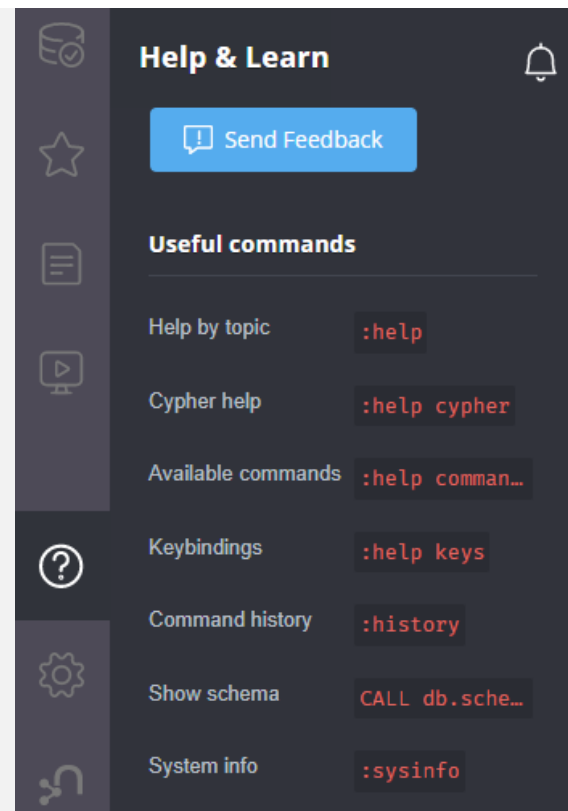
Neo4j Browser





Przegląd domyślnych zapytań

HELP



☆ Favorite: Tworzenie węzła*

```
1 //Tworzenie węzła
2 MERGE (n:Start)
```

```
neo4j$ //Tworzenie węzła MERGE (n:S
```



Table

(no changes, no records)

MERGE (n:Start)

CREATE (a:Publikacja)-[r:Podstawa_dla]->(b:Publikacja)

☆ Favorite: Tworzenie relacji*

```
1 //Tworzenie relacji
2 CREATE (a:Publikacja)-[r:Podstawa_dla]->(b:Publikacja)
3
```

```
neo4j$ //Tworzenie relacji CREATE (a:Publikacja)-[r:Podstawa_d
```



Table

Added 2 labels, created 2 nodes, created 1 relationship, completed after 1 ms.

Wyczyszczenie bazy

`MATCH (n) DETACH DELETE n`

Budowa sieci społecznościowej

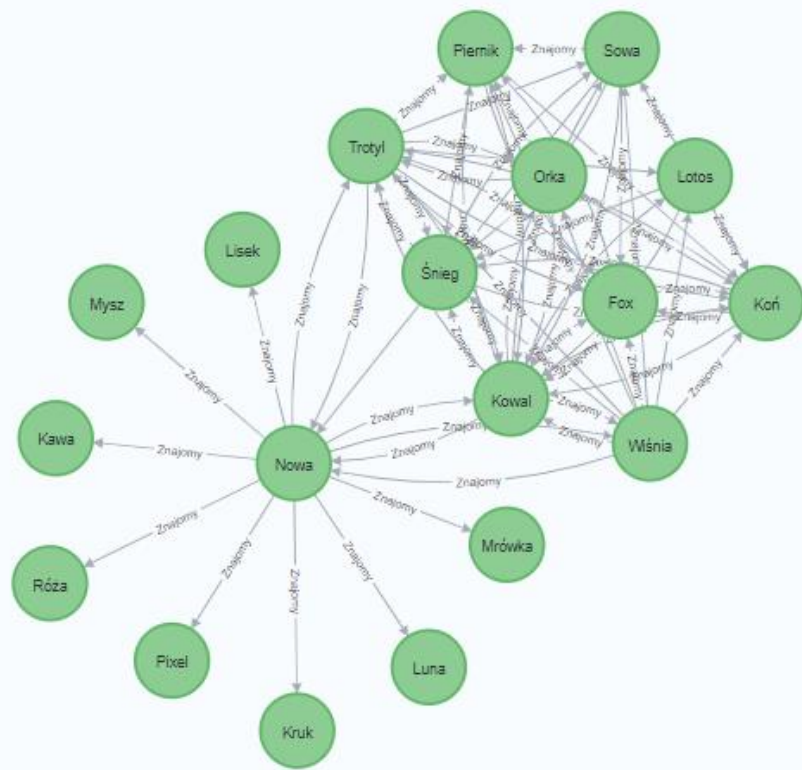
`LOAD CSV WITH HEADERS FROM 'file:///siecN.csv' AS row
MERGE (o:Osoba {id: row.id, nick: row.nick, email: row.email, wiek: row.wiek, plec: row.plec, miasto: row.miasto});`

`LOAD CSV WITH HEADERS FROM 'file:///siecR.csv' AS row
MATCH (u1:Osoba {id: row.osoba_id}),
 (u2:Osoba {id: row.znajomy_id})
CREATE (u1)-[:Znajomy]->(u2);`

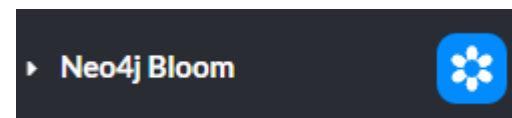
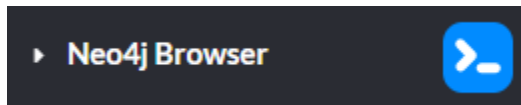
Neo4jDesktop > relate-data > dbmss > dbms-8ebf8cc2-e35b-4aad-bb96-f7668a255990 > import

Nazwa	Data modyfikacji
 siecN.csv	17.05.2024 13:05
 siecR.csv	17.05.2024 13:04

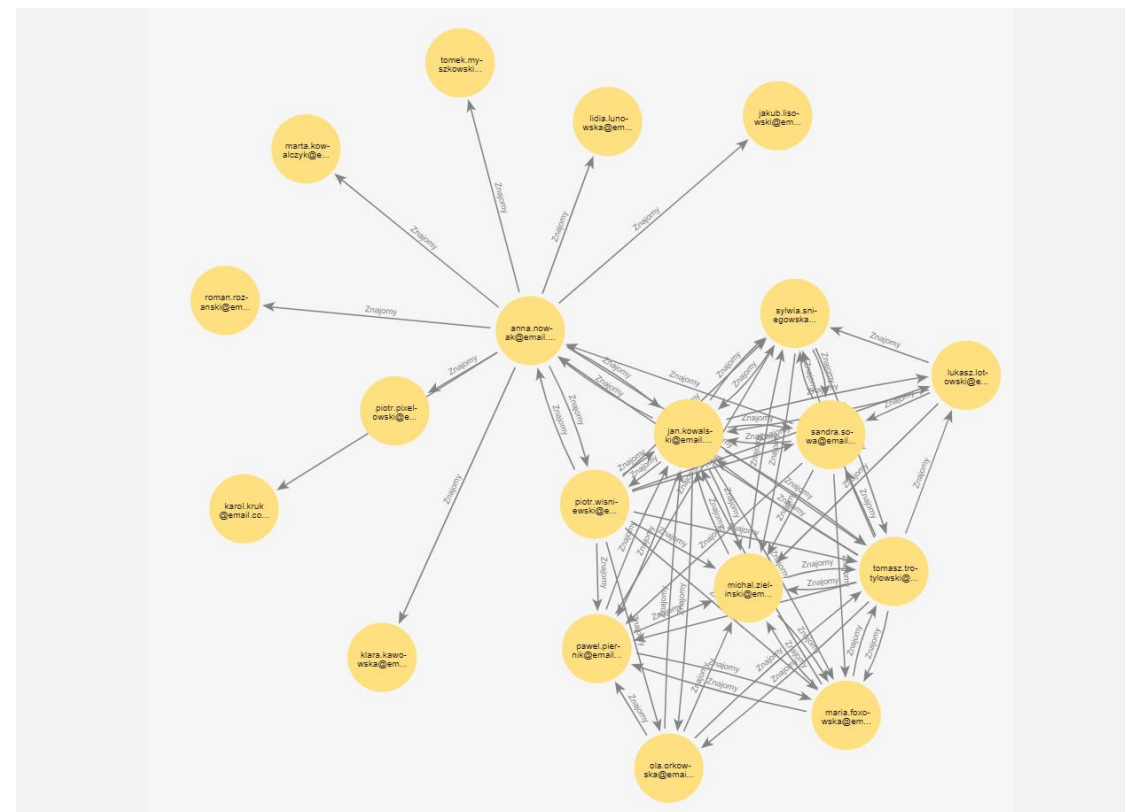
Katalog import



MATCH p=()-->() RETURN p



(any) — Znajomy — (any)



Zapytania Cypher

Znalezienie wszystkich
znajomych danej osoby

```
MATCH (p:Osoba {nick: 'Sowa'})-[:Znajomy]->(friends)
RETURN p.nick, friends.nick AS Znajomi;
```

Ścieżki między dwoma osobami

```
MATCH path = shortestPath((a:Osoba {nick: 'Sowa'})-[:Znajomy*]-
(b:Osoba {nick: 'Mysz'}))
RETURN path;
```

Osoby w danym mieście

```
MATCH (p:Osoba {miasto: 'Warszawa'})
RETURN p.nick;
```

Proponowanie nowych
znajomych na podstawie
wspólnych znajomych:

```
MATCH (p:Osoba {nick: 'Sowa'})-[:Znajomy*2]-(recommended)
WHERE NOT (p)-[:Znajomy]-(recommended) AND p <> recommended
WITH recommended, COUNT(*) AS wspolniZnajomi
RETURN recommended.nick AS RekomendowanyZnajomy, wspolniZnajomi
ORDER BY wspolniZnajomi DESC
LIMIT 5;
```

Zapytania Cypher

Znalezienie wszystkich
bezpośrednich znajomych

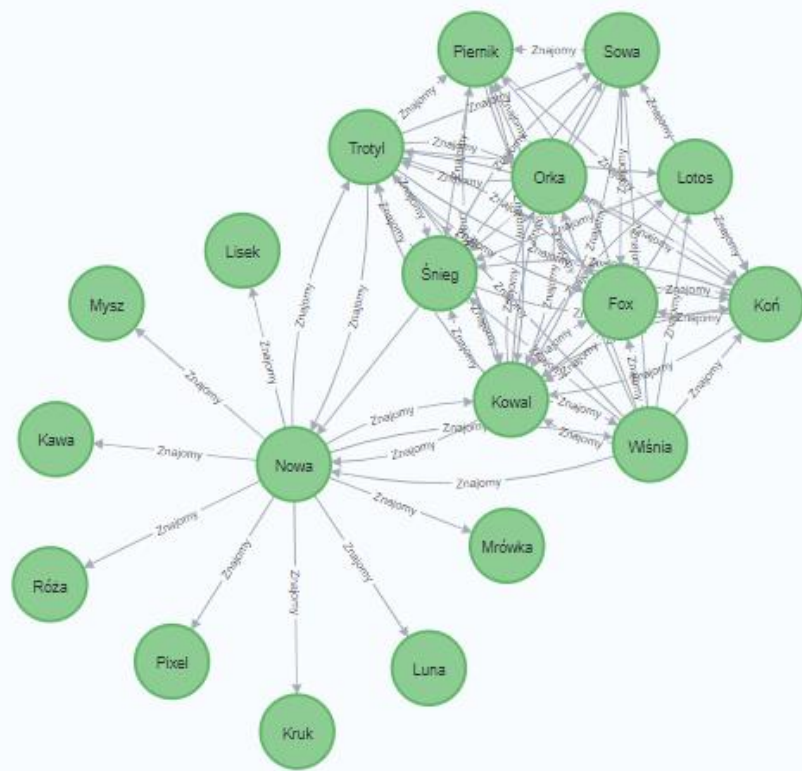
```
MATCH (s:Osoba {nick: 'Sowa'})-[:Znajomy]->(friend)
RETURN friend.nick AS Znajomi;
```

Znalezienie najbardziej
wpływowej osoby wśród
znajomych
(po liczbie sąsiadów)

```
MATCH (s:Osoba {nick: 'Sowa'})-[:Znajomy]->(friend)
WITH friend
MATCH (friend)-[:Znajomy]->(others)
RETURN friend.nick AS Nick, COUNT(DISTINCT others) AS Wpływ
ORDER BY Wpływ DESC
LIMIT 1;
```

Identyfikacja osób, które
nie mają wspólnych znajomych

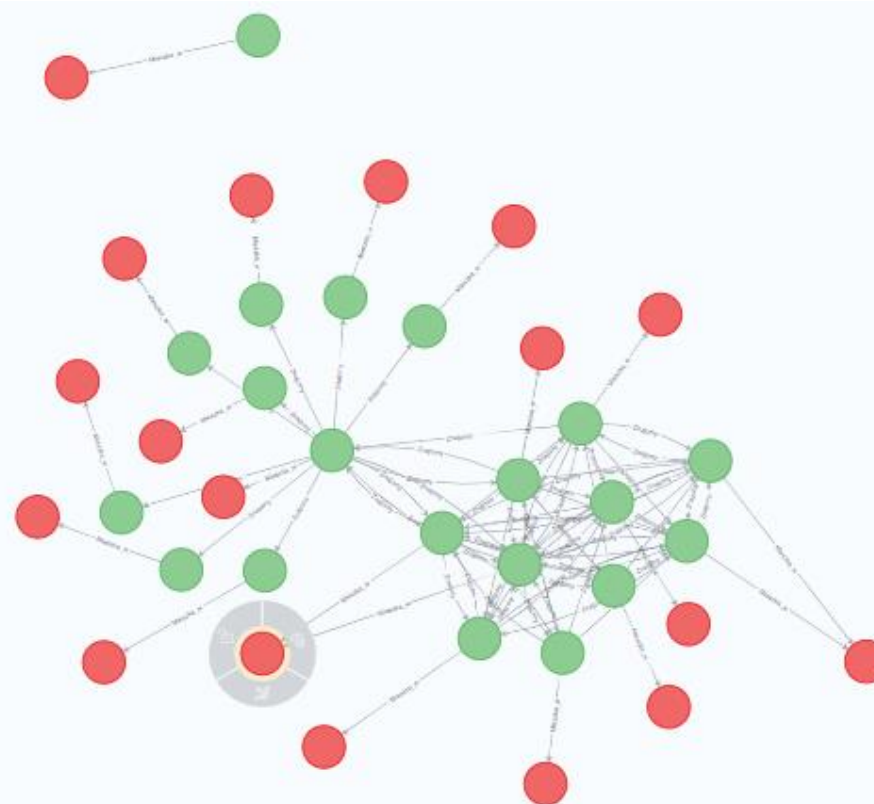
```
MATCH (s:Osoba {nick: 'Sowa'}), (o:Osoba)
WHERE NOT (s)-[:Znajomy*1..2]-(o)
RETURN o.nick AS Izolowani;
```



Osoba



Osoba - > Miasto



Wyodrębnienie właściwości jako węzłów

Budowa węzłów (:Miasto)

```
MATCH (p:Osoba)
WITH DISTINCT p.miasto AS cityName
MERGE (:Miasto {nazwa: cityName});
```

Budowa relacji [:Mieszka_w]

```
MATCH (p:Osoba), (m:Miasto)
WHERE p.miasto = m.nazwa
MERGE (p)-[:Mieszka_w]->(m);
```

Usunięcie właściwości „miasto”

```
MATCH (p:Osoba)
REMOVE p.miasto;
```

Dodanie właściwości z węzłów

Dodanie właściwości „miasto”
do węzłów (:Osoba)

```
MATCH (p:Osoba)-[:Mieszka_w]->(m:Miasto)  
SET p.miasto = m.nazwa
```

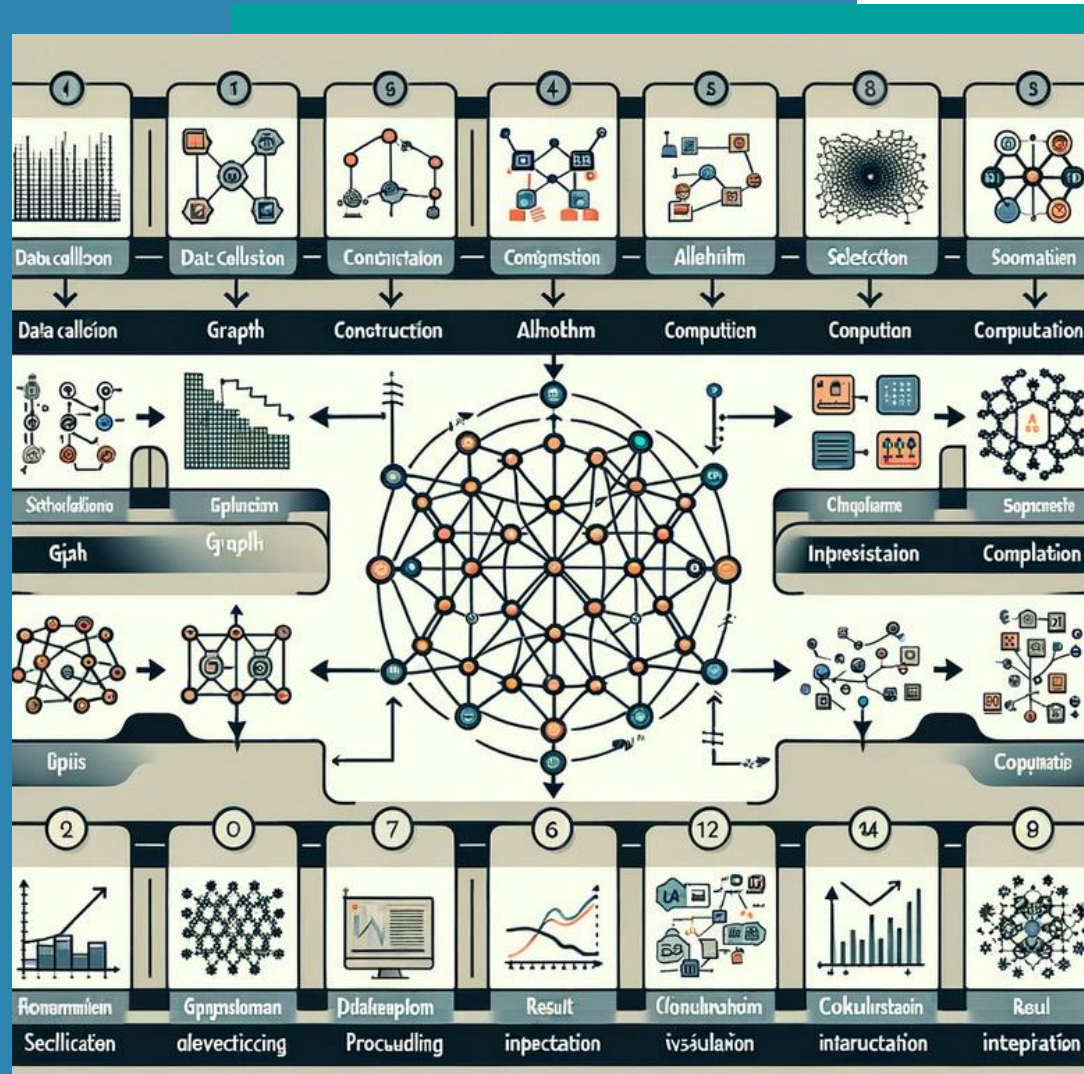
Usunięcie węzłów (:Miasto)

```
MATCH (m:Miasto)  
DETACH DELETE m
```



Biblioteki analizy grafowej

ANALIZA DANYCH
Z WYKORZYSTANIEM BAZY
GRAFOWEJ NEO4J
I ALGORYTMÓW GRAFOWYCH



Zagadnienia

- [Awesome Procedures on Cypher \(APOC\)](#)
- [Graph Data Science \(GDS\)](#)
- [Projekcja grafu](#)

Awesome Procedures on Cypher (APOC)

Awesome Procedures on Cypher (APOC)

składa się z około 450 procedur i funkcji, które pomagają w wielu różnych zadaniach w obszarach takich jak: manipulacja kolekcjami, algorytmy grafowe i konwersja danych.

Manipulacja Danymi

- **Tworzenie i modyfikowanie węzłów i relacji:** Umożliwia bardziej zaawansowane operacje na węzłach i relacjach, takie jak masowe tworzenie, modyfikowanie i usuwanie.
- **Import i eksport danych:** Ułatwia importowanie danych z różnych źródeł (np. CSV, JSON) oraz eksportowanie danych do tych formatów.

Procedury Cypher

- **Rozszerzone funkcje Cypher:** Dostarcza dodatkowych funkcji i procedur, które można wywoływać z poziomu zapytań Cypher, takich jak procedury związane z przetwarzaniem tekstu, operacjami na danych, mapami, listami i innymi.
- **Algorytmy grafowe:** Implementuje różne algorytmy grafowe, takie jak PageRank, centralność, klasteryzacja itp.

Zarządzanie Bazą Danych

- **Backup i przywracanie:** Procedury do tworzenia kopii zapasowych bazy danych oraz jej przywracania.
- **Monitorowanie i zarządzanie:** Narzędzia do monitorowania stanu bazy danych, takie jak metryki wydajności i statystyki.

Integracja z innymi systemami

- **HTTP:** Umożliwia wykonywanie zapytań HTTP oraz parsowanie odpowiedzi, co ułatwia integrację z zewnętrznymi API.
- **JDBC:** Pozwala na wykonywanie zapytań SQL do baz danych za pomocą JDBC i przetwarzanie wyników w Neo4j.

Automatyzacja i Przetwarzanie

- **Procedury wsadowe:** Umożliwia wykonywanie operacji wsadowych, które są bardziej efektywne w przypadku dużych zbiorów danych.
- **Konwersje i przekształcenia danych:** Procedury do konwersji i przekształcania danych w różnych formatach.

Graph Data Science (GDS)

Graph Data Science (GDS)

Jest to biblioteka i umożliwia wykorzystanie zaawansowanych algorytmów nauki maszynowej i analizy grafów na dużą skalę.

Zaawansowane Algorytmy Grafowe:

Zawiera szeroki zakres algorytmów, które mogą być użyte do analizy połączeń w grafach, w tym algorytmy takie jak: wyszukiwanie najkrótszej ścieżki, wykrywanie społeczności, centralność i inne metryki grafowe.

Optymalizacja dla Dużych Grafów:

Jest zaprojektowane, by efektywnie obsługiwać bardzo duże grafy, co pozwala na skalowalne przetwarzanie nawet miliardów węzłów i krawędzi.

Integracja z Neo4j:

Działa bezpośrednio z bazą danych Neo4j, co pozwala na łatwe i szybkie przetwarzanie danych grafowych przechowywanych w Neo4j.

Elastyczność w Przetwarzaniu:

Użytkownicy mogą projektować swoje grafy w pamięci dla szybszego dostępu i efektywności, co jest szczególnie użyteczne w iteracyjnych i eksperymentalnych procesach analizy grafów.

Prostota Użycia:

Dzięki dostarczaniu algorytmów jako gotowych do użycia procedur, GDS znacząco upraszcza proces implementacji skomplikowanych operacji grafowych.

Graph Data Science (GDS)

Przykłady Zastosowań GDS

Analiza Sieci Społecznościowych:

Wykorzystanie algorytmów detekcji społeczności do identyfikacji grup i ich wzajemnych interakcji w sieciach społecznościowych.

Rekomendacje:

Zastosowanie technik, takich jak przejścia po grafach i algorytmy podobieństwa, do generowania rekomendacji produktów lub treści.

Detekcja Oszustw:

Analiza wzorców w grafach transakcji do identyfikacji potencjalnie podejrzanych zachowań.

Analiza Drogowa i Sieciowa:

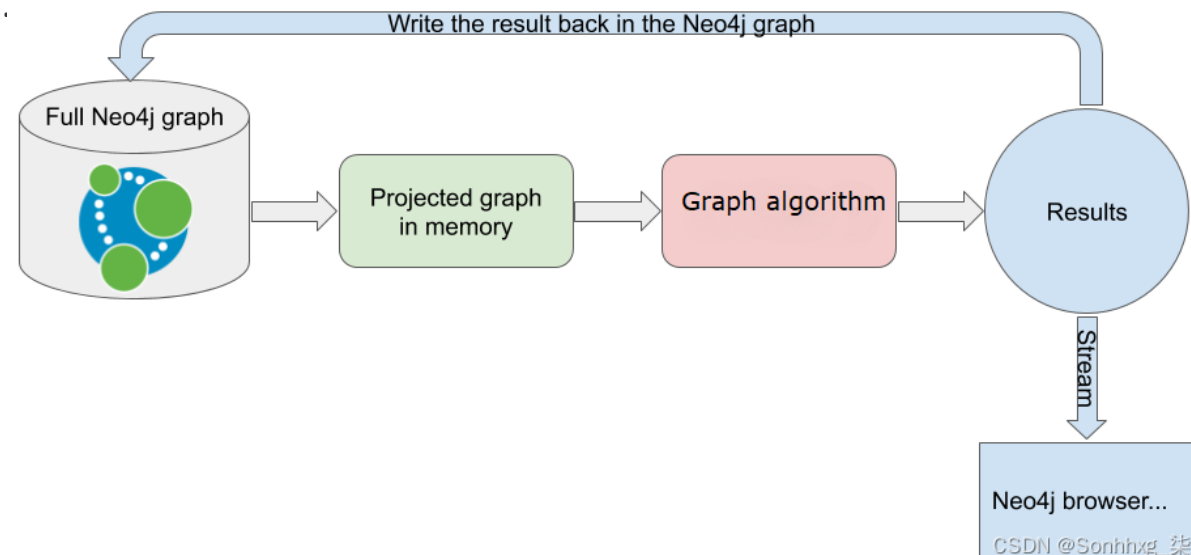
Użycie algorytmów do znajdowania najkrótszych ścieżek i innych metryk grafowych w celu optymalizacji tras i sieci.

Projekcja grafu

Funkcja `gds.graph.project`

w bibliotece Graph Data Science (GDS) w Neo4j służy do projekcji grafów w pamięci, co pozwala na szybsze wykonanie algorytmów analizy grafowej.

Ta funkcja jest kluczowa dla wydajności i elastyczności przy obsłudze dużych grafów oraz przy stosowaniu różnorodnych algorytmów z biblioteki GDS.



Projekcja grafu

Składnia gds.graph.project

```
CALL gds.graph.project(  
  graphName: String,  
  nodeProjection: String or List or Map,  
  relationshipProjection: String or List or Map,  
  configuration: Map
```

```
) YIELD
```

```
  graphName: String,  
  nodeProjection: Map,  
  nodeCount: Integer,  
  relationshipProjection: Map,  
  relationshipCount: Integer,  
  projectMillis: Integer
```

Nazwa	Typ	Opcjonalne	Opis
graphName	String	nie	Nazwa, pod którą graf jest przechowywany w katalogu.
nodeProjection	String, Lista lub Mapa	nie	Jedna lub więcej projekcji węzłów.
relationshipProjection	String, Lista lub Mapa	nie	Jedna lub więcej projekcji relacji.
configuration	Mapa	tak	Dodatkowe parametry do konfiguracji natywnej projekcji.

Nazwa	Typ	Opis
graphName	String	Nazwa, pod którą graf jest przechowywany w katalogu.
nodeProjection	Mapa	Projekcje węzłów używane do projektowania grafu.
nodeCount	Integer	Liczba węzłów przechowywanych w zaprojektowanym grafie.
relationshipProjection	Mapa	Projekcje relacji używane do projektowania grafu.
relationshipCount	Integer	Liczba relacji przechowywanych w zaprojektowanym grafie.
projectMillis	Integer	Milisekundy potrzebne na projektowanie grafu.

Projekcja grafu

Konfiguracja (configuration)

Nazwa	Typ	Domyślnie	Opis
readConcurrency	Integer	4	Liczba równocześnie używanych wątków do tworzenia grafu.
nodeProperties	String, Lista lub Mapa	{}	Właściwości węzłów do załadowania z węzłów, które pasują do dowolnej z etykiet określonych w nodeProjection.
relationshipProperties	String, Lista lub Mapa	{}	Właściwości relacji do załadowania z relacji, które pasują do dowolnego z typów określonych w relationshipProjection.
validateRelationships	Boolean	false	Czy zgłaszać błąd, jeśli relationshipProjection zawiera relacje między węzłami nieobecnymi w nodeProjection.
jobId	String	Wygenerowany wewnątrz	Identyfikator, który można dostarczyć, aby łatwiej śledzić postęp projekcji.

Wyjaśnienia

- **Map:** Struktura danych klucz-wartość, która zawiera szczegółowe informacje.
- **String:** Tekstowe reprezentacje danych.
- **Long, Integer:** Numeryczne typy danych używane do reprezentowania liczbowych wartości.
- **Double:** Typ danych dla wartości zmiennoprzecinkowych.
- **ZonedDateTime:** Typ danych używany do reprezentowania daty i czasu z uwzględnieniem strefy czasowej.

Składnia gds.graph.project

Nazwa grafu

Projekcja węzłów

Projekcja relacji

Wyniki

```
CALL gds.graph.project(  
  'socialGraph',  
  {  
    Osoba: {properties: 'wiek'},  
    Miasto: {}  
  },  
  {  
    Znajomy: {type: 'Znajomy', orientation: 'UNDIRECTED'}  
  }  
)  
YIELD  
  graphName, nodeProjection, nodeCount, relationshipProjection, relationshipC  
  ount, projectMillis  
RETURN *
```

Tylko liczby !



Przydatne działania

Sprawdzenie typów właściwości węzłów

```
CALL apoc.meta.nodeTypeProperties()  
YIELD nodeType, propertyName, propertyTypes  
RETURN nodeType, propertyName, propertyTypes
```

Zmiana typu właściwości

```
MATCH (p:Osoba)  
SET p.wiek = toInteger(p.wiek)
```

Lista grafów w pamięci

```
CALL gds.graph.list()
```

Usuwa graf z pamięci

```
CALL gds.graph.drop('socialGraph')
```

Graph Data Science (GDS)

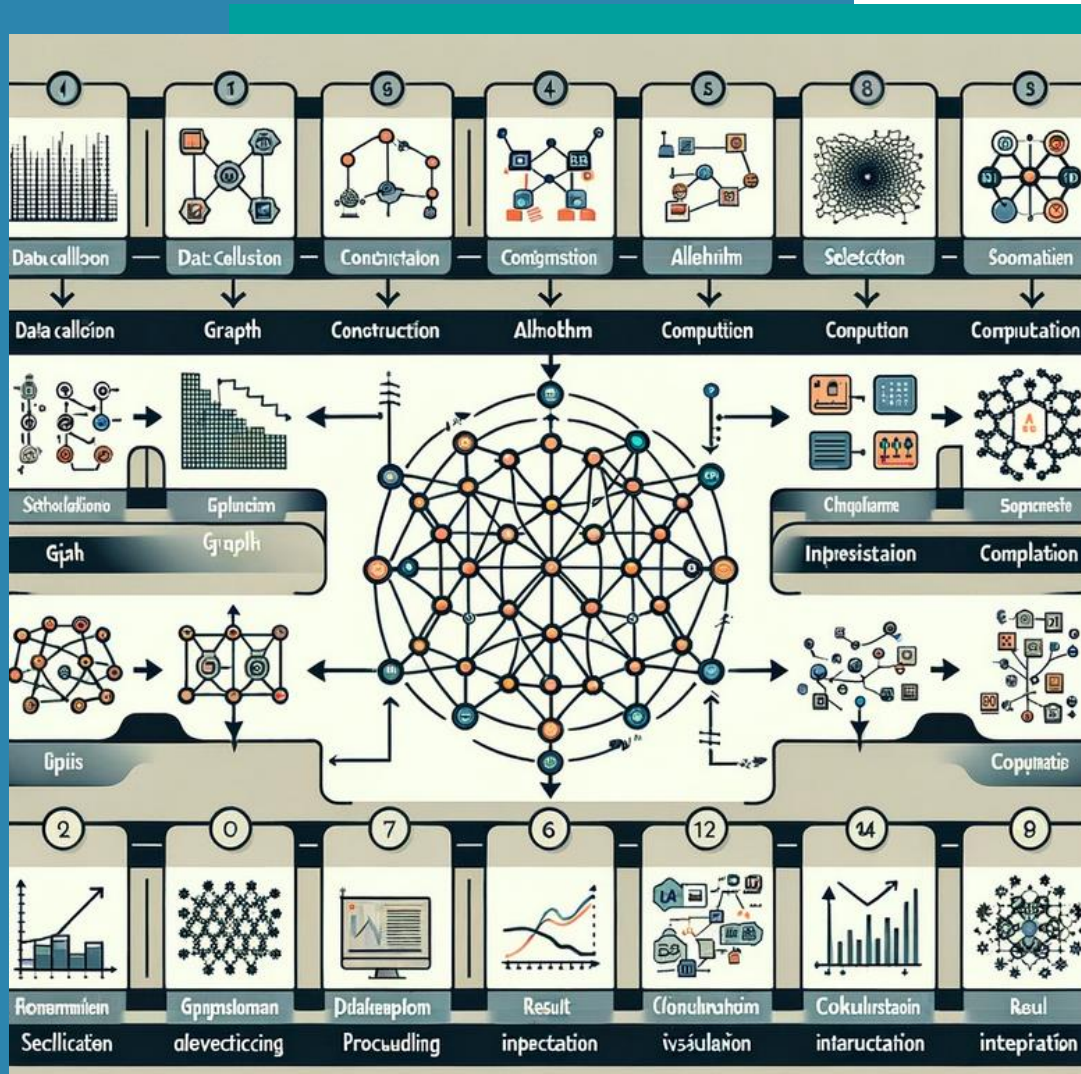
CALL gds.graph.list()

Nazwa	Typ	Opis
degreeDistribution	Map	Rozkład stopni węzłów w grafie pokazujący, jak wiele węzłów ma określoną liczbę krawędzi.
graphName	String	Nazwa grafu, pod którą jest przechowywany w katalogu GDS.
database	String	Nazwa bazy danych, z której graf został załadowany.
databaseLocation	String	Lokalizacja bazy danych, z której graf został załadowany.
memoryUsage	String	Wielkość pamięci RAM wykorzystywanej przez graf.
sizeInBytes	Long	Całkowity rozmiar grafu w bajtach.
nodeCount	Integer	Liczba węzłów zawartych w grafie.
relationshipCount	Integer	Liczba relacji (krawędzi) zawartych w grafie.
configuration	Map	Konfiguracja użyta do załadowania grafu, w tym projekcje węzłów i relacji oraz inne ustawienia.
density	Double	Gęstość grafu, wyrażona jako stosunek liczby istniejących relacji do liczby możliwych relacji.
creationTime	ZonedDateTime	Czas utworzenia grafu.
modificationTime	ZonedDateTime	Czas ostatniej modyfikacji grafu.
schema	Map	Schemat grafu, pokazujący, jakie etykiety węzłów i typy relacji są zawarte w grafie.
schemaWithOrientation	Map	Schemat grafu z dodatkową informacją o orientacji relacji (skierowane czy nieskierowane).



Przypadki użycia „Zadania”

ANALIZA DANYCH
Z WYKORZYSTANIEM BAZY
GRAFOWEJ NEO4J
I ALGORYTMÓW GRAFOWYCH



Zagadnienia

- [Analiza społeczności](#)
 - [Rozwiązanie w oparciu o Neo4j Browser](#)
 - [Rozwiązanie w oparciu o Neo4j Bloom](#)
 - [Rozwiązanie w oparciu o Neo4j Graph Data Science Playground](#)
- [Trasy w logistyce](#)
- [Budowa sieci komputerowej](#)
- [Incydenty bezpieczeństwa](#)
- [Egzamin: Obrona kraju](#)

Analiza społeczności

Obiekt analizy grafowej

Sieć społecznościowa znajomych



Cel analizy grafowej

Znalezienie osób, które w podgrupach przyjaciół najlepiej spełniają poniższe kryteria:

1. Mają duży wpływ na inne osoby.
2. Które łączą różne grupy w sieci.
3. Są dobrym przekąźnikiem informacji (plotkarze).

Zadanie: analiza społeczności

Kroki Realizacji Projektu

1. Stwórz graf: Użyj przykładowe dane do zbudowania grafu.
2. Wybierz algorytmy: Wybierz odpowiednie algorytmy do rozwiązania problemu.
3. Zaimplementuj rozwiązania: Napisz zapytania Cypher wykorzystujące wybrane algorytmy.
4. Rozszerz analizę: Przeprowadź normalizację i zbuduj ranking.
5. Wizualizuj wyniki: Prezentuj wyniki w czytelnej formie.

Rozwiązanie w oparciu o:

Neo4j Browser

Stop  Open  ...

 Neo4j Browser

 Neo4j Bloom

 Graph Data Scien...

JFrog Landing

 Neo4j ETL Tool

 Terminal



Reveal in Neo4j Explorer

File name

► Neo4j Browser



Zapytania Cypher

Database Information

Database Information

neo4j 

Node labels

(43) Book Miasto Osoba Person

Relationship types

(96) KNOWS Mieszka_w READ Znajomy

Property keys

age data email id isbn miasto name nazwa nick nodes numberOfPages pagerank plec price

neo4j\$

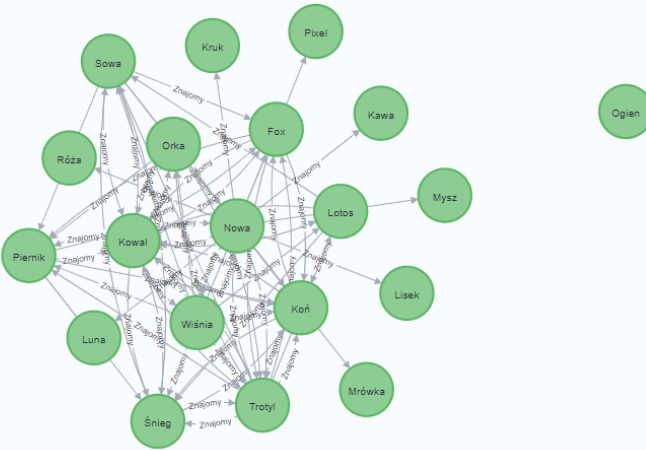
neo4j\$ MATCH (n:Osoba) RETURN n LIMIT 25

Graph

Table

Text

Code



Zadanie: analiza społeczności

Budowa grafu

Przekopiuj do katalogu /import pliki „siecN.csv” i „siecR.csv”

MATCH (n) DETACH DELETE n;

LOAD CSV WITH HEADERS FROM 'file:///siecN.csv' AS row

```

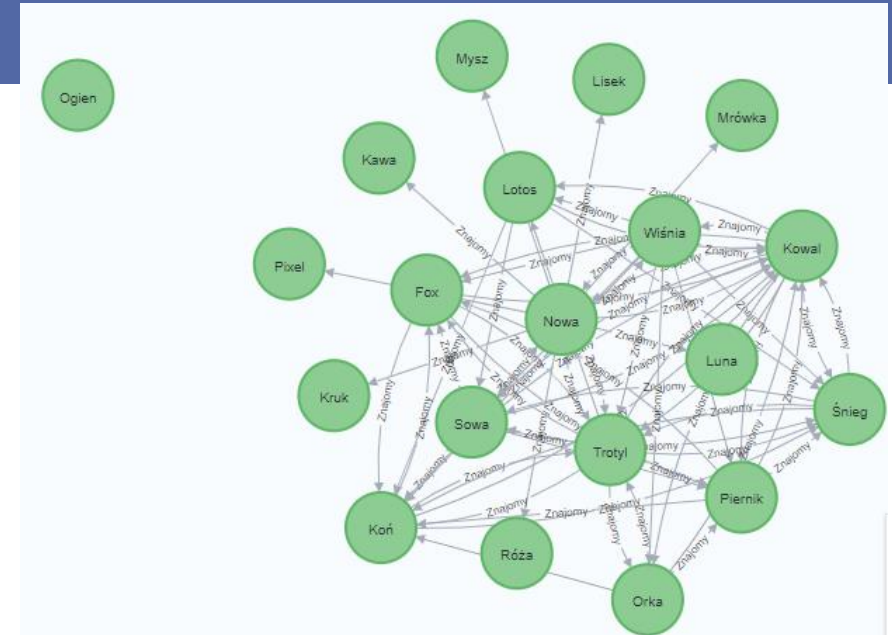
MERGE (o:Osoba {id: row.id, nick: row.nick, email: row.email, wiek: row.wiek, plec: row.plec, miasto: row.miasto});

```

LOAD CSV WITH HEADERS FROM 'file:///siecR.csv' AS row

```
MATCH (u1:Osoba {id: row.osoba_id}),
      (u2:Osoba {id: row.znajomy_id})
```

```
CREATE (u1)-[:Znajomy]->(u2);
```



Zadanie: analiza społeczności

Budowa grafu w pamięci

```
CALL gds.graph.project(  
  'znajomi',  
  ['Osoba'],  
  {  
    Znajomy: {type: 'Znajomy', orientation: 'UNDIRECTED'}  
  }  
);
```

Zadanie: analiza społeczności

Najbardziej wpływowe osoby - PageRank

Algorytm: PageRank

PageRank jest użyteczny do identyfikacji najbardziej wpływowych osób w sieci społecznej, mierząc "siłę" każdego węzła w oparciu o liczbę połączeń przychodzących i wychodzących.

```
CALL gds.pageRank.stream('znajomi')
```

```
YIELD nodeId, score
```

```
MATCH (n) WHERE id(n) = nodeId
```

```
RETURN id(n), n.nick, score
```

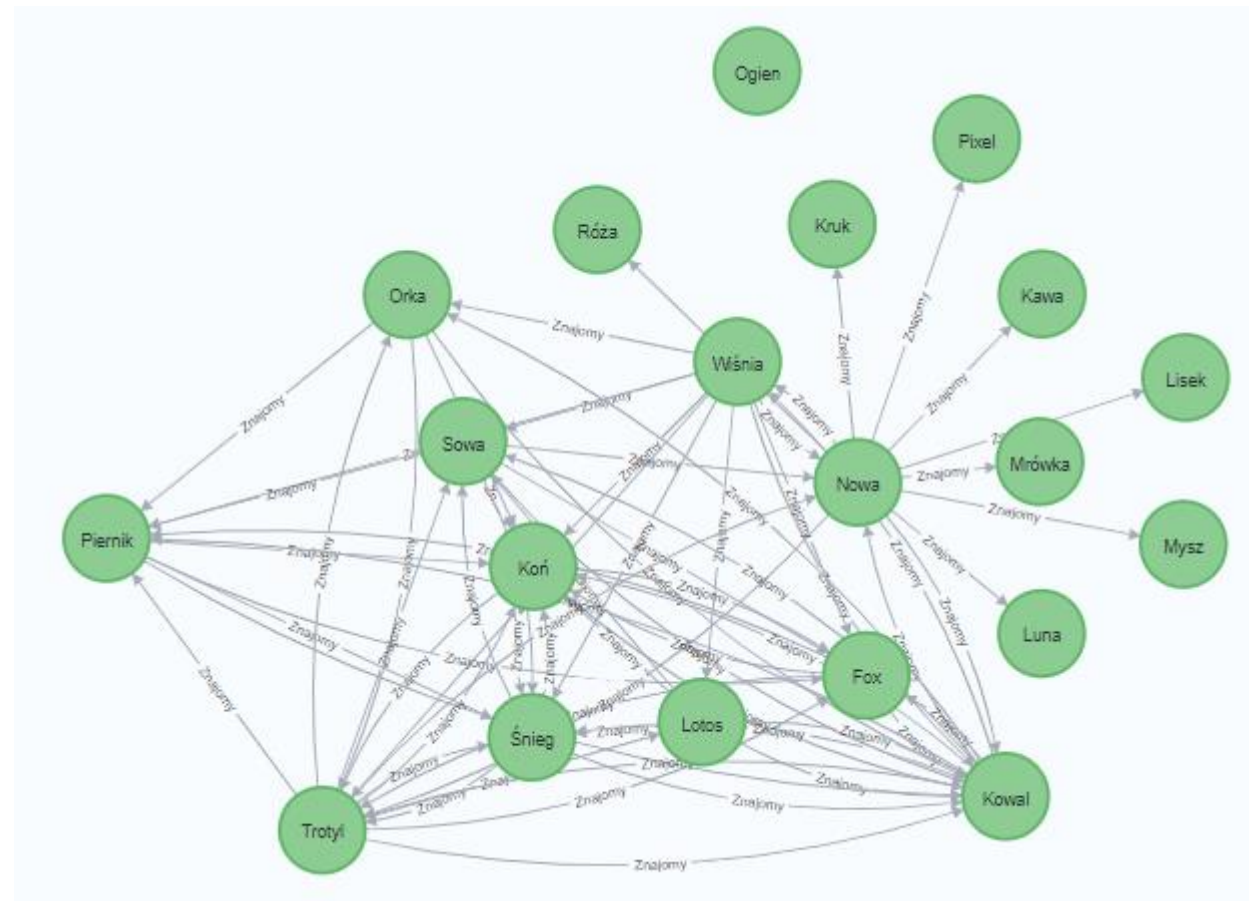
```
ORDER BY score DESC
```

Zadanie: analiza społeczności

Najbardziej wpływowe osoby - PageRank

Wynik

id(n)	n.nick	score
1	"Nowa"	2,955032067
0	"Kowal"	2,149014909
19	"Trotyl"	1,765948851
2	"Wiśnia"	1,398942469
4	"Koń"	1,348468171
12	"Sowa"	1,230033106
18	"Śnieg"	1,164271637
16	"Piernik"	1,068109284
6	"Fox"	1,064478951
8	"Lotos"	0,793244921
14	"Orka"	0,792837771
3	"Mrówka"	0,316648346
5	"Lisek"	0,316648346
9	"Mysz"	0,316648346
10	"Kawa"	0,316648346
11	"Pixel"	0,316648346
13	"Kruk"	0,316648346
15	"Róża"	0,316648346
17	"Luna"	0,316648346
7	"Ogien"	0,15



Zadanie: analiza społeczności

Składnia gds.pageRank.stream

CALL gds.pageRank.stream(
graphName: String,
configuration: Map
)

YIELD
nodeId: Integer,
score: Float

Konfiguracja

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
graphName	String	n/a	nie	Nazwa grafu przechowywanego w katalogu.
configuration	Mapa	{}	tak	Konfiguracja specyficzna dla algorytmu i/lub filtrowania grafu.

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
nodeLabels	Lista ciągów znaków	['*']	tak	Filtruje nazwany graf przy użyciu podanych etykiet węzłów. Węzły z każdą z podanych etykiet zostaną uwzględnione.
relationshipTypes	Lista ciągów znaków	['*']	tak	Filtruje nazwany graf przy użyciu podanych typów relacji. Relacje z każdym z podanych typów zostaną uwzględnione.
concurrency	Integer	4	tak	Liczba równoczesnych wątków używanych do uruchomienia algorytmu.
jobId	String	Generowane wewnętrznie	tak	ID, które można dostarczyć, aby łatwiej śledzić postęp algorytmu.
logProgress	Boolean	true	tak	Jeśli wyłączone, postęp w procentach nie będzie rejestrowany.
dampingFactor	Float	0.85	tak	Współczynnik tłumienia dla obliczeń PageRank. Musi być w zakresie [0,1].
maxIterations	Integer	20	tak	Maksymalna liczba iteracji PageRank do uruchomienia.
tolerance	Float	0.0000001	tak	Minimalna zmiana w wynikach między iteracjami. Jeśli wszystkie wyniki zmieniają się mniej niż wartość tolerancji, algorytm uznaje wynik za stabilny i kończy działanie.
relationshipWeightProperty	String	null	tak	Nazwa właściwości relacji używana jako wagi. Jeśli nieokreślona, algorytm wykonuje się bez wag.
sourceNodes	Lista węzłów lub numerów	[]	tak	Węzły lub ID węzłów używane do obliczenia spersonalizowanego PageRank.
scaler	String lub Mapa	None	tak	Nazwa skalera stosowanego do końcowych wyników. Obsługiwane wartości to None, MinMax, Max, Mean, Log i StdScore. Aby zastosować specyficzną konfigurację skalera, użyj składni Map: {scaler: 'nazwa'}.

Zadanie: analiza społeczności

Osoby, które łączą różne grupy w sieci - Betweenness Centrality

Algorytm: Centralność Pośrednictwa (Betweenness Centrality)

Miara ta pomaga zidentyfikować osoby, które pełnią rolę "mostów" łączących różne grupy w sieci.

```
CALL gds.betweenness.stream('znajomi')
```

```
YIELD nodeId, score
```

```
MATCH (n) WHERE id(n) = nodeId
```

```
RETURN id(n), n.nick, score
```

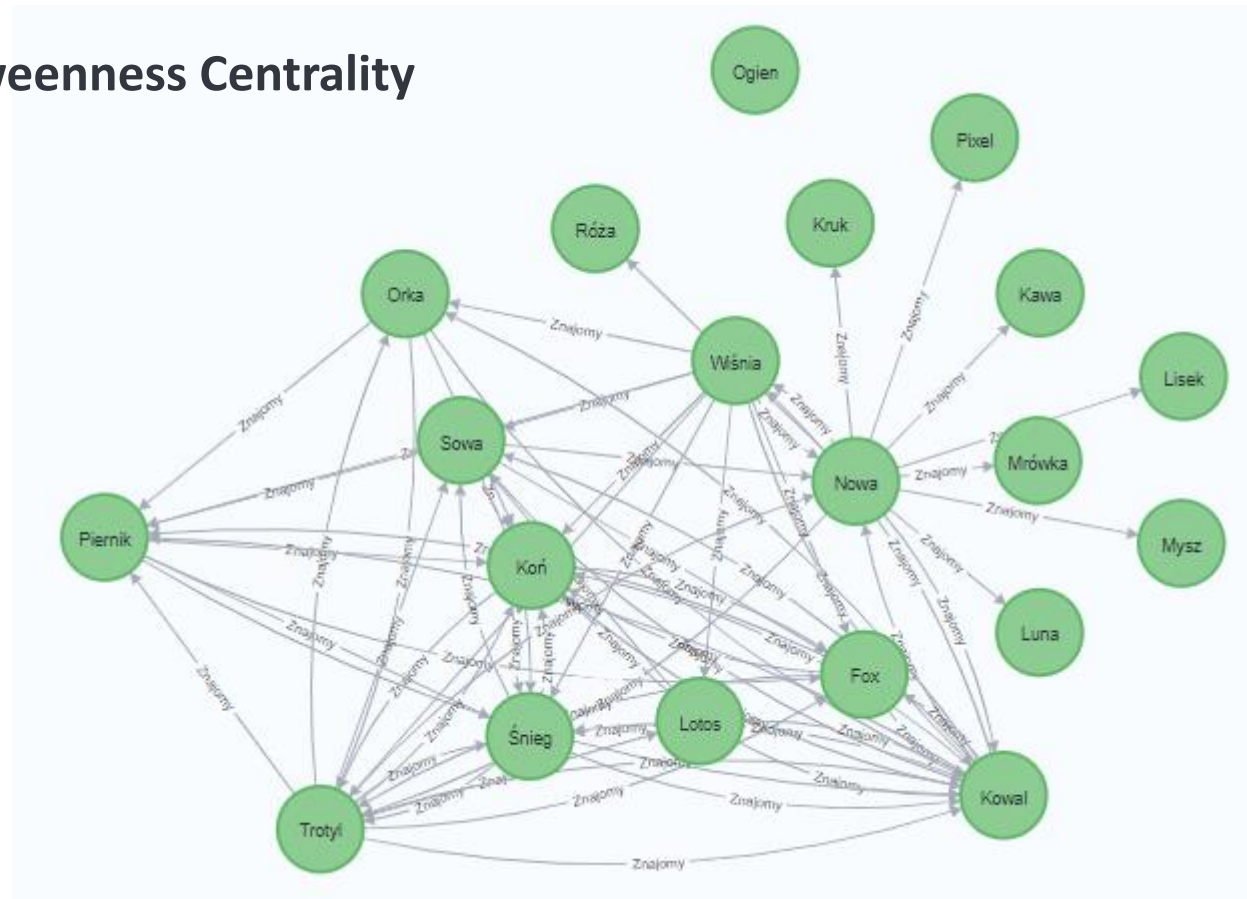
```
ORDER BY score DESC
```

Zadanie: analiza społeczności

Osoby, które łączą różne grupy w sieci - Betweenness Centrality

Wynik

id(n)	n.nick	score
1	"Nowa"	108
0	"Kowal"	23,81066319
19	"Trotyl"	18,80510764
2	"Wiśnia"	11,23902943
12	"Sowa"	5,465121806
4	"Koń"	1,10302916
16	"Piernik"	0,465937657
18	"Śnieg"	0,111111111
3	"Mrówka"	0
5	"Lisek"	0
6	"Fox"	0
7	"Ogień"	0
8	"Lotos"	0
9	"Mysz"	0
10	"Kawa"	0
11	"Pixel"	0
13	"Kruk"	0
14	"Orka"	0
15	"Róża"	0
17	"Luna"	0



Zadanie: analiza społeczności

Składnia gds.betweenness.stream

CALL gds.betweenness.stream(
graphName: String,
configuration: Map

)

YIELD

nodeId: Integer,

score: Float

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
graphName	String	n/a	nie	Nazwa grafu przechowywanego w katalogu.
configuration	Mapa	{}	tak	Konfiguracja specyficzna dla algorytmu i/lub filtrowania grafu.

Konfiguracja

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
nodeLabels	Lista ciągów znaków	[*]	tak	Filtruje nazwany graf przy użyciu podanych etykiet węzłów. Węzły z każdą z podanych etykiet zostaną uwzględnione.
relationshipTypes	Lista ciągów znaków	[*]	tak	Filtruje nazwany graf przy użyciu podanych typów relacji. Relacje z każdym z podanych typów zostaną uwzględnione.
concurrency	Integer	4	tak	Liczba równoczesnych wątków używanych do uruchomienia algorytmu.
jobId	String	Generowane wewnętrznie	tak	ID, które można dostarczyć, aby łatwiej śledzić postęp algorytmu.
logProgress	Boolean	true	tak	Jeśli wyłączone, postęp w procentach nie będzie rejestrowany.
samplingSize	Integer	liczba węzłów	tak	Liczba węzłów źródłowych do rozważenia podczas obliczania wyników centralności.
samplingSeed	Integer	null	tak	Wartość ziarna dla generatora liczb losowych, który wybiera węzły startowe.
relationshipWeightProperty	String	null	tak	Nazwa właściwości relacji używana jako wagi. Jeśli nieokreślona, algorytm wykonuje się bez wag.

Zadanie: analiza społeczności

Dobrych plotkarzy - Closeness Centrality

Algorytm: Bliskość Centralna (Closeness Centrality)

Pokazuje, jak blisko dana osoba jest do wszystkich innych osób w sieci, co może wskazywać na szybkość, z jaką informacje mogą być rozpowszechniane przez daną osobę.

```
CALL gds.closeness.stream('znajomi')
```

```
YIELD nodeId, score
```

```
MATCH (n) WHERE id(n) = nodeId
```

```
RETURN id(n), n.nick, score
```

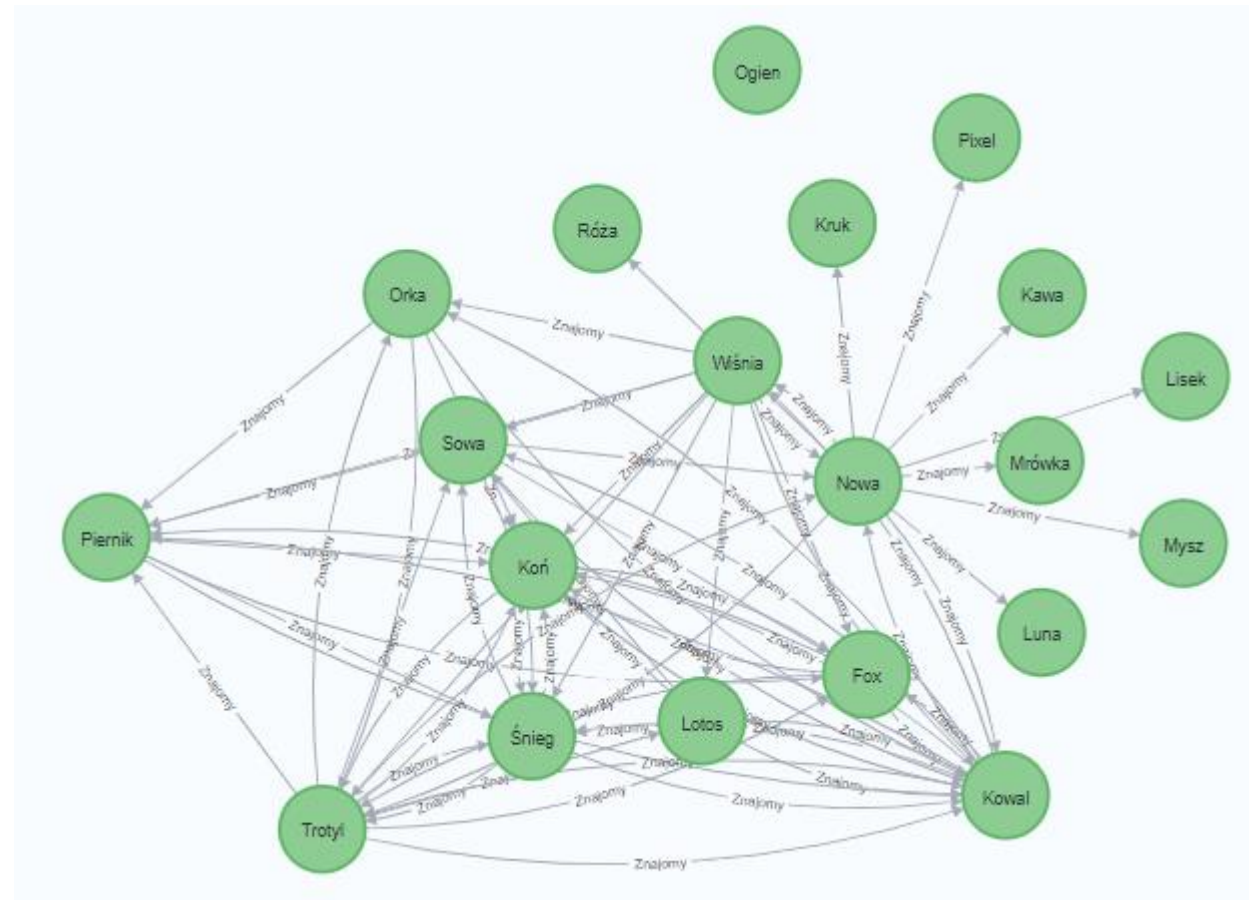
```
ORDER BY score DESC
```

Zadanie: analiza społeczności

Dobrych plotkarzy - Closeness Centrality

Wynik

id(n)	n.nick	score
1	"Nowa"	0,75
0	"Kowal"	0,692307692
2	"Wiśnia"	0,692307692
19	"Trotyl"	0,692307692
12	"Sowa"	0,666666667
4	"Koń"	0,514285714
16	"Piernik"	0,5
18	"Śnieg"	0,486486486
6	"Fox"	0,473684211
8	"Lotos"	0,473684211
14	"Orka"	0,461538462
3	"Mrówka"	0,43902439
5	"Lisek"	0,43902439
9	"Mysz"	0,43902439
10	"Kawa"	0,43902439
11	"Pixel"	0,43902439
13	"Kruk"	0,43902439
15	"Róża"	0,43902439
17	"Luna"	0,43902439
7	"Ogień"	0



Zadanie: analiza społeczności

Składnia gds.closeness.stream

```
CALL gds.closeness.stream(  
  graphName: String,  
  configuration: Map  
)
```

YIELD

```
  nodeId: Integer,  
  score: Float
```

Konfiguracja

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
graphName	String	n/a	nie	Nazwa grafu przechowywanego w katalogu.
configuration	Mapa	{}	tak	Konfiguracja specyficzna dla algorytmu i/lub filtrowania grafu.

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
nodeLabels	Lista ciągów znaków	['*']	tak	Filtruje nazwany graf przy użyciu podanych etykiet węzłów. Węzły z każdą z podanych etykiet zostaną uwzględnione.
relationshipTypes	Lista ciągów znaków	['*']	tak	Filtruje nazwany graf przy użyciu podanych typów relacji. Relacje z każdym z podanych typów zostaną uwzględnione.
concurrency	Integer	4	tak	Liczba równoczesnych wątków używanych do uruchomienia algorytmu.
jobId	String	Generowane wewnętrznie	tak	ID, które można dostarczyć, aby łatwiej śledzić postęp algorytmu.
logProgress	Boolean	true	tak	Jeśli wyłączone, postęp w procentach nie będzie rejestrowany.
useWassermanFaust	Boolean	false	tak	Używa ulepszonej formuły Wassermana-Fausta do obliczeń centralności bliskości.

Zadanie: analiza społeczności

Podgrupy przyjaźni – Louvain

Algorytm: Louvain Detekcja Wspólnot (Community Detection)

Pokazuje, jak blisko dana osoba jest do wszystkich innych osób w sieci, co może wskazywać na szybkość, z jaką informacje mogą być rozpowszechniane przez daną osobę.

```
CALL gds.louvain.stream('znajomi')
```

```
YIELD nodeId, communityId
```

```
MATCH (n) WHERE id(n) = nodeId
```

```
RETURN id(n), n.nick, communityId
```

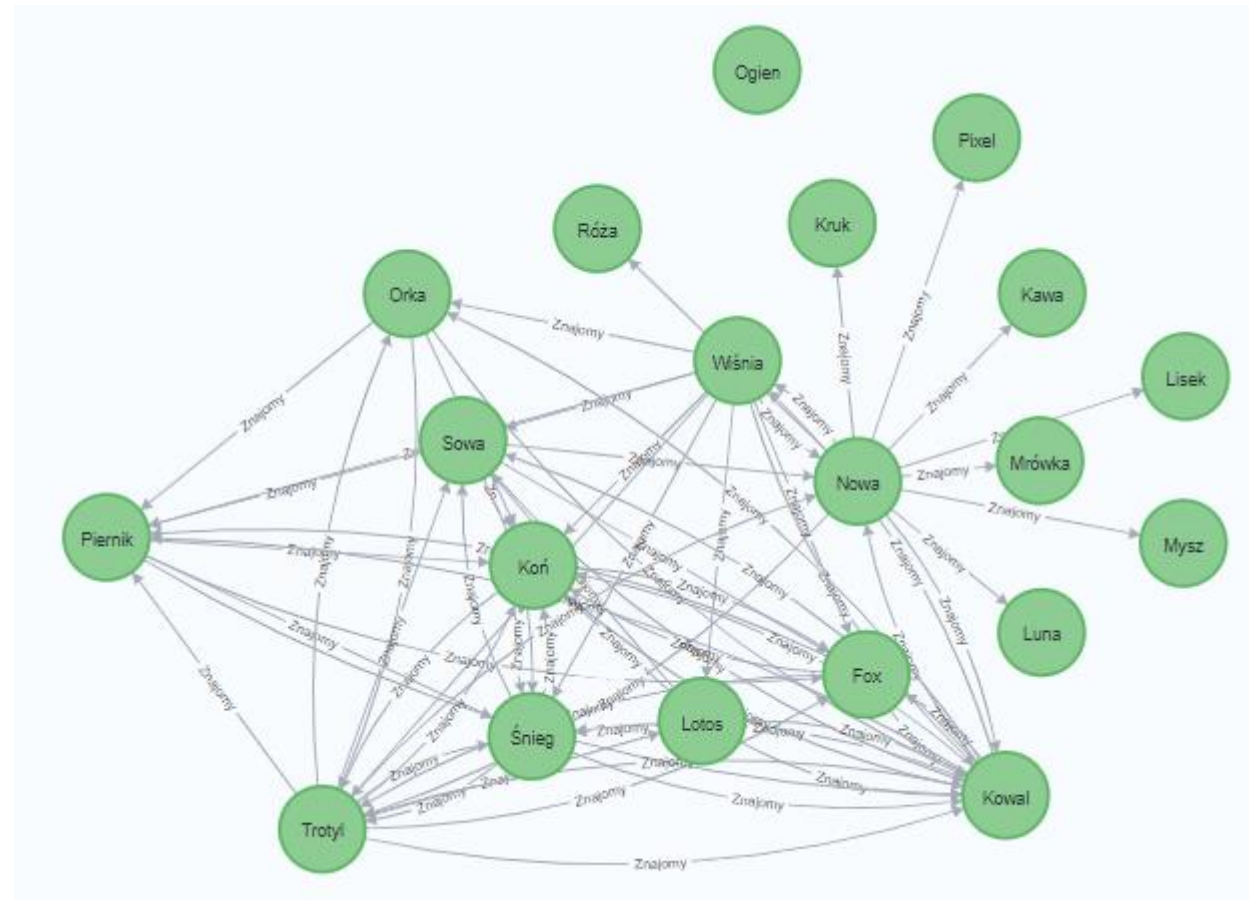
```
ORDER BY communityId DESC
```

Zadanie: analiza społeczności

Podgrupy przyjaźni – Louvain

Wynik

id(n)	n.nick	communityId
0	"Kowal"	16
2	"Wiśnia"	16
4	"Koń"	16
6	"Fox"	16
8	"Lotos"	16
12	"Sowa"	16
14	"Orka"	16
16	"Piernik"	16
18	"Śnieg"	16
19	"Trotyl"	16
7	"Ogień"	7
1	"Nowa"	1
3	"Mrówka"	1
5	"Lisek"	1
9	"Mysz"	1
10	"Kawa"	1
11	"Pixel"	1
13	"Kruk"	1
15	"Róża"	1
17	"Luna"	1



Zadanie: analiza społeczności

Składnia gds.louvain.stream

```
CALL gds.louvain.stream(  
  graphName: String,  
  configuration: Map  
)  
YIELD  
  nodeId: Integer,  
  communityId: Integer,  
  intermediateCommunityIds: List of Integer
```

Konfiguracja

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
nodeLabels	Lista ciągów znaków	['*']	tak	Filtruje nazwany graf przy użyciu podanych etykiet węzłów. Węzły z każdą z podanych etykiet zostaną uwzględnione.
relationshipTypes	Lista ciągów znaków	['*']	tak	Filtruje nazwany graf przy użyciu podanych typów relacji. Relacje z każdym z podanych typów zostaną uwzględnione.
concurrency	Integer	4	tak	Liczba równoczesnych wątków używanych do uruchomienia algorytmu.
jobId	String	Generowane wewnętrznie	tak	ID, które można dostarczyć, aby łatwiej śledzić postęp algorytmu.
logProgress	Boolean	true	tak	Jeśli wyłączone, postęp w procentach nie będzie rejestrowany.
relationshipWeightProperty	String	null	tak	Nazwa właściwości relacji używana jako wagi. Jeśli nieokreślona, algorytm wykonuje się bez wag.
seedProperty	String	n/a	tak	Używana do ustawienia początkowej społeczności dla węzła. Wartość właściwości musi być nieujemną liczbą.
maxLevels	Integer	10	tak	Maksymalna liczba poziomów, na których może działać algorytm.
maxIterations	Integer	10	tak	Maksymalna liczba iteracji, jakie algorytm może wykonać na każdym poziomie.
tolerance	Float	0.0001	tak	Minimalna zmiana w modularności między iteracjami. Jeśli modularność zmienia się mniej niż wartość tolerancji, wynik jest uznawany za stabilny i algorytm kończy działanie.
includeIntermediateCommunities	Boolean	false	tak	Wskazuje, czy zapisywać pośrednie wspólnoty. Jeśli ustawione na false, tylko końcowa wspólnota jest zapisywana.
consecutiveIds	Boolean	false	tak	Flagi, czy identyfikatory komponentów są mapowane na ciągłą przestrzeń identyfikatorów (wymaga dodatkowej pamięci). Nie można używać razem z flagą includeIntermediateCommunities.
minCommunitySize	Integer	0	tak	Tylko węzły wewnątrz społeczności równych lub większych od podanej wartości są zwracane.

Zadanie: analiza społeczności

Łączone zapytanie

```
// Wywołanie PageRank
CALL gds.pageRank.stream('znajomi')
YIELD nodeId AS pageRankNodeId, score AS pageRankScore

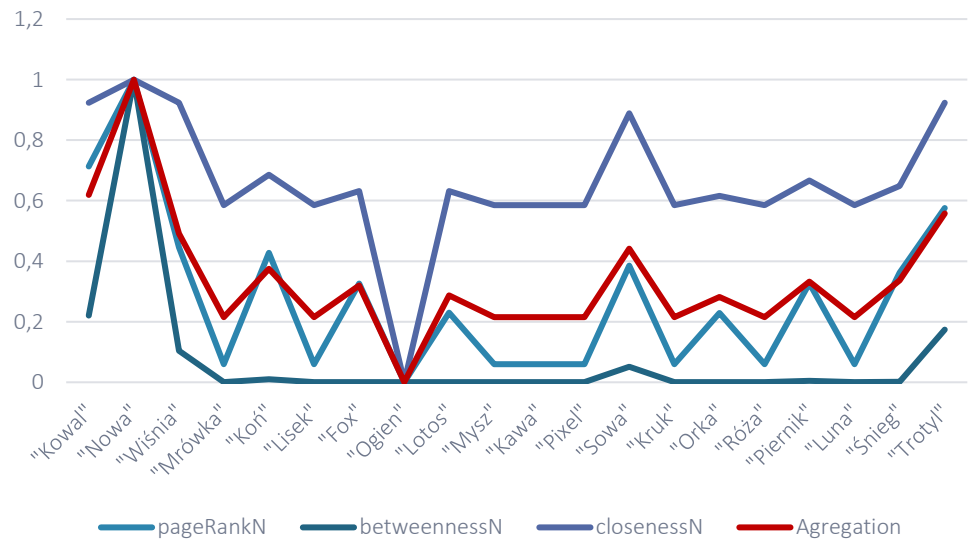
// Dołącz wyniki Betweenness Centrality
CALL gds.betweenness.stream('znajomi')
YIELD nodeId AS betweennessNodeId, score AS betweennessScore
WITH pageRankNodeId, pageRankScore, betweennessNodeId, betweennessScore
WHERE pageRankNodeId = betweennessNodeId

// Dołącz wyniki Closeness Centrality
CALL gds.closeness.stream('znajomi')
YIELD nodeId AS closenessNodeId, score AS closenessScore
WITH pageRankNodeId, pageRankScore, betweennessScore, closenessNodeId, closenessScore
WHERE pageRankNodeId = closenessNodeId

// Zwróć wyniki w jednej tabeli
MATCH (n) WHERE id(n) = pageRankNodeId
RETURN id(n) AS nodeId, n.nick AS nick, pageRankScore, betweennessScore, closenessScore
ORDER BY pageRankScore DESC, betweennessScore DESC, closenessScore DESC;
```

Zadanie: analiza społeczności

Normalizacja i agregacja



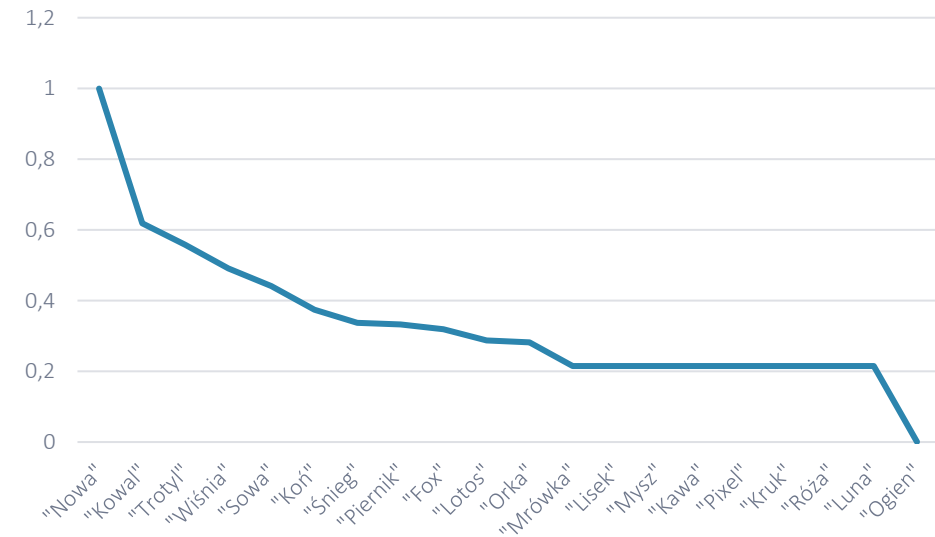
nick	Wyniki			Normalizacja			Agregation
	pageRank	betweenness	closeness	pageRankN	betweennessN	closenessN	
"Kowal"	2,149014909	23,81066319	0,692307692	0,712653139	0,220469104	0,923076923	0,618733055
"Nowa"	2,955032067	108	0,75	1	1	1	1
"Wiśnia"	1,398942469	11,23902943	0,692307692	0,445250692	0,104065087	0,923076923	0,490797567
"Mrówka"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Korń"	1,348468171	1,10302916	0,514285714	0,427256495	0,010213233	0,685714286	0,374394671
"Lisek"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Fox"	1,064478951	0	0,473684211	0,326013724	0	0,631578947	0,319197557
"Ogień"	0,15	0	0	0	0	0	0
"Lotos"	0,793244921	0	0,473684211	0,229318206	0	0,631578947	0,286965718
"Mysz"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Kawa"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Pixel"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Sowa"	1,230033106	5,465121806	0,666666667	0,385034139	0,05060298	0,888888889	0,441508669
"Kruk"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Orka"	0,792837771	0	0,461538462	0,229173056	0	0,615384615	0,281519224
"Róża"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Piernik"	1,068109284	0,465937657	0,5	0,327307946	0,004314238	0,666666667	0,33276295
"Luna"	0,316648346	0	0,43902439	0,059410496	0	0,585365854	0,21492545
"Śnieg"	1,164271637	0,111111111	0,486486486	0,361590033	0,001028807	0,648648649	0,337089163
"Trotyl"	1,765948851	18,80510764	0,692307692	0,576089261	0,174121367	0,923076923	0,557762517

MIN	0,150000000	0,000000000	0,000000000
MAX	2,955032067	108,000000000	0,750000000
waga	0,333333333	0,333333333	0,333333333

Zadanie: analiza społeczności

Ranking

nodeId	nick	Agregation	Pozycja	Louvain
1	"Nowa"	1	1	1
0	"Kowal"	0,618733055	2	16
19	"Trotyl"	0,557762517	3	16
2	"Wiśnia"	0,490797567	4	16
12	"Sowa"	0,441508669	5	16
4	"Koń"	0,374394671	6	16
18	"Śnieg"	0,337089163	7	16
16	"Piernik"	0,33276295	8	16
6	"Fox"	0,319197557	9	16
8	"Lotos"	0,286965718	10	16
14	"Orka"	0,281519224	11	16
3	"Mrówka"	0,21492545	12	1
5	"Lisek"	0,21492545	13	1
9	"Mysz"	0,21492545	14	1
10	"Kawa"	0,21492545	15	1
11	"Pixel"	0,21492545	16	1
13	"Kruk"	0,21492545	17	1
15	"Róża"	0,21492545	18	1
17	"Luna"	0,21492545	19	1
7	"Ogien"	0	20	7





Rozwiązanie w oparciu o:

Neo4j Bloom



Stop ↺ Open ▾ ⋮

Neo4j Browser

Neo4j Bloom

Graph Data Scien...

JFrog Landing

Neo4j ETL Tool

Terminal



Reveal in IDE Explorer

Filename ▾

Neo4j Bloom



Wyszukiwanie



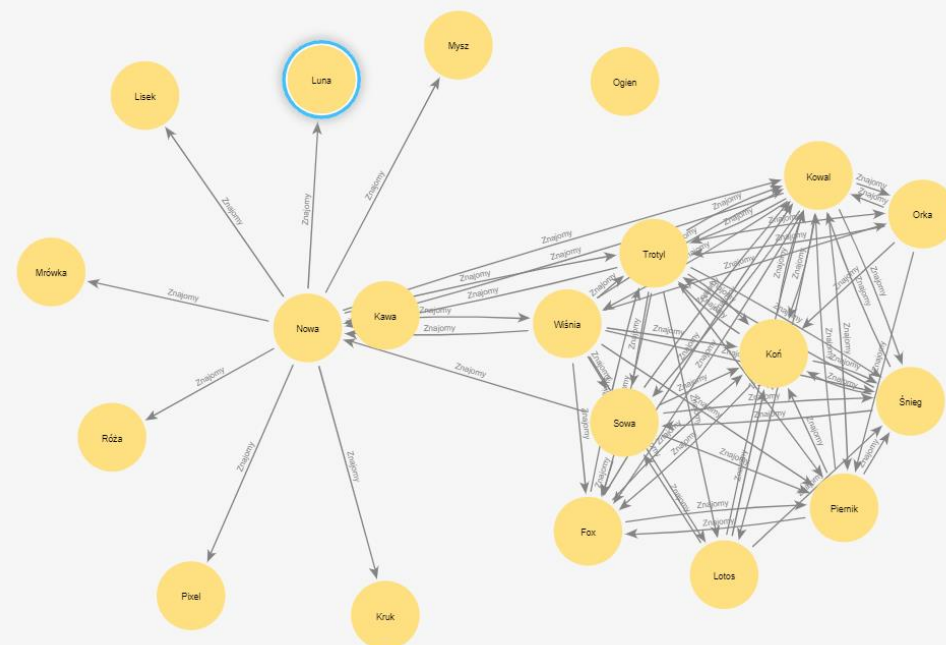
(any)

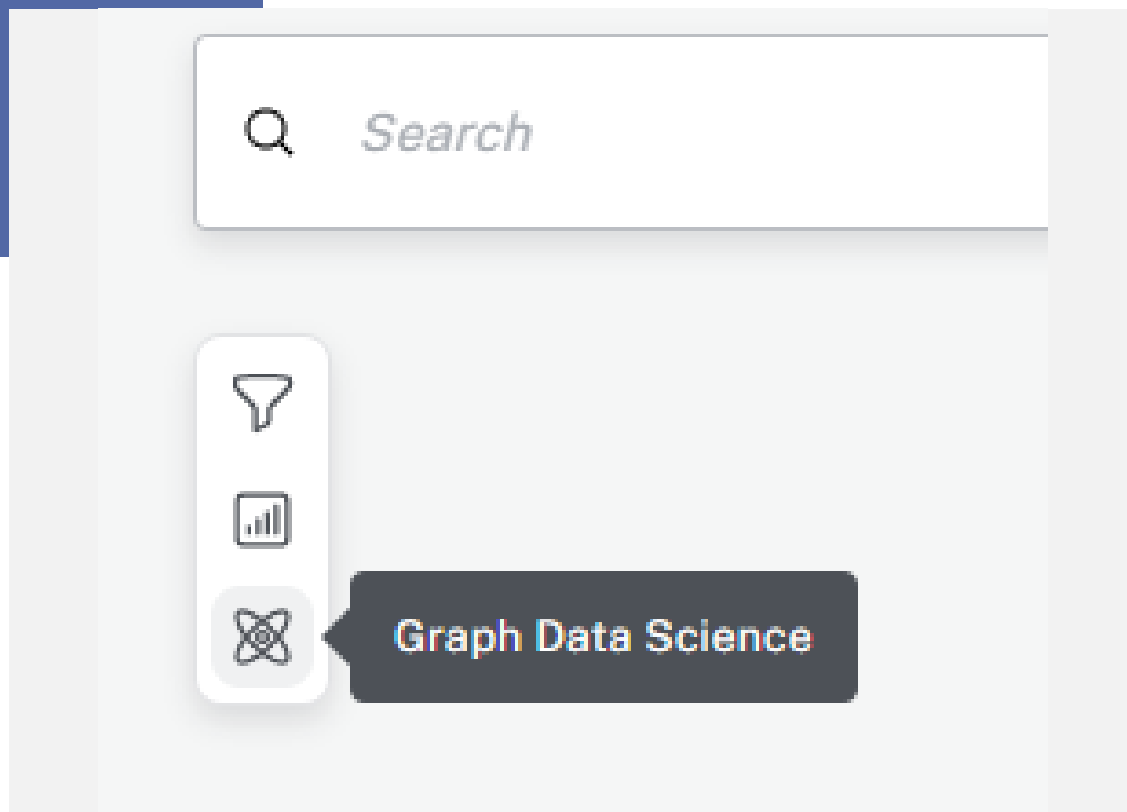
Znajomy

(any)



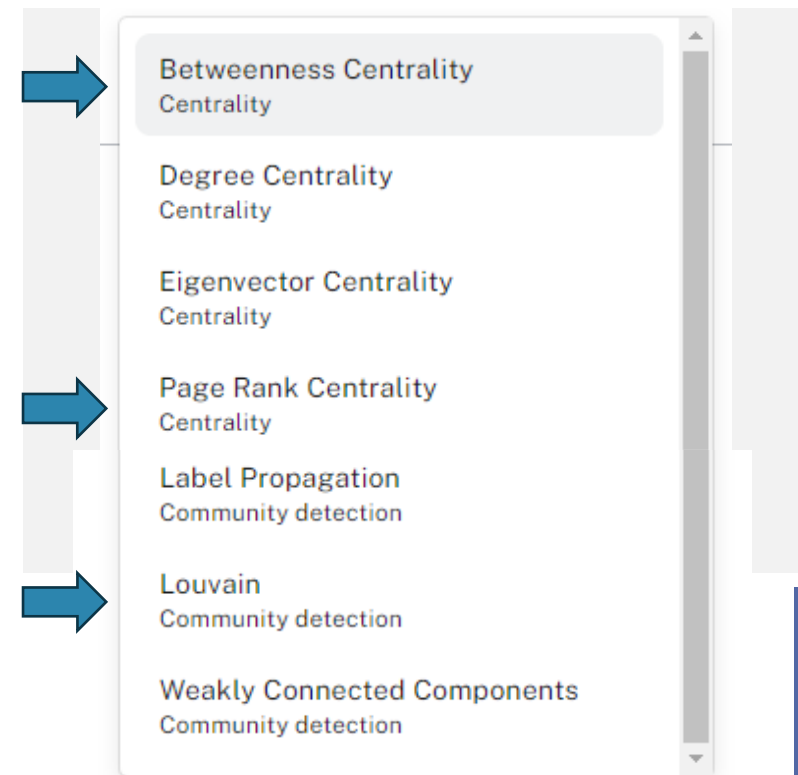
Osoba





Algorytmy grafowe

Wykaz algorytmów

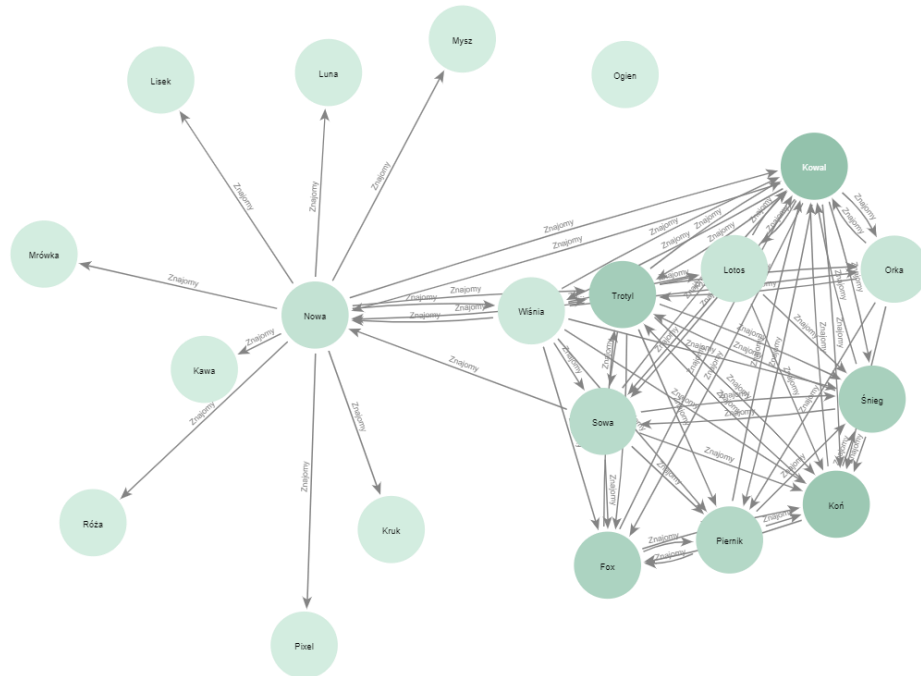


Brak: Bliskość Centralna (Closeness Centrality)

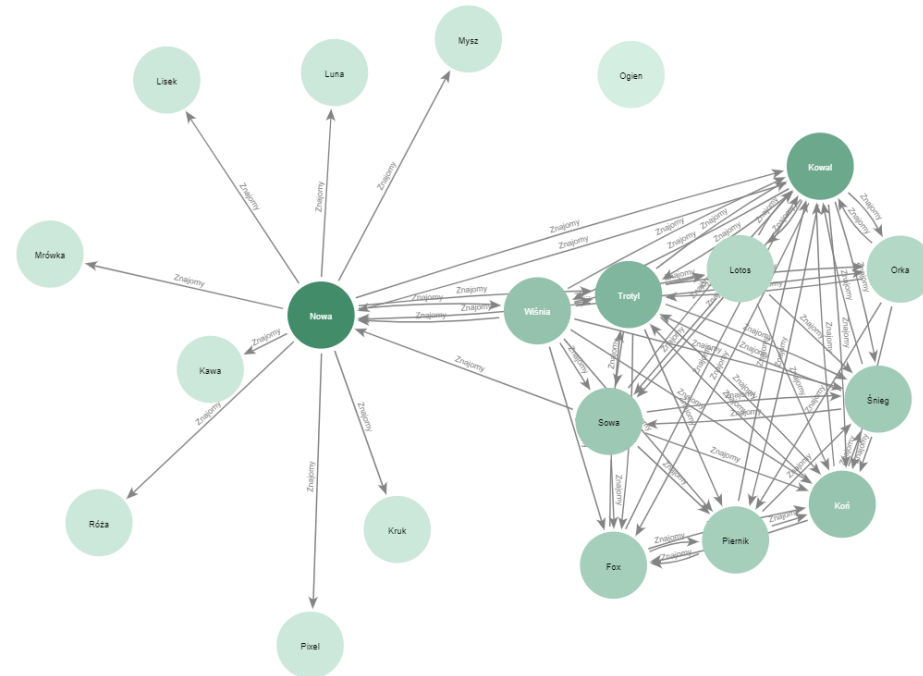
Zadanie: analiza społeczności

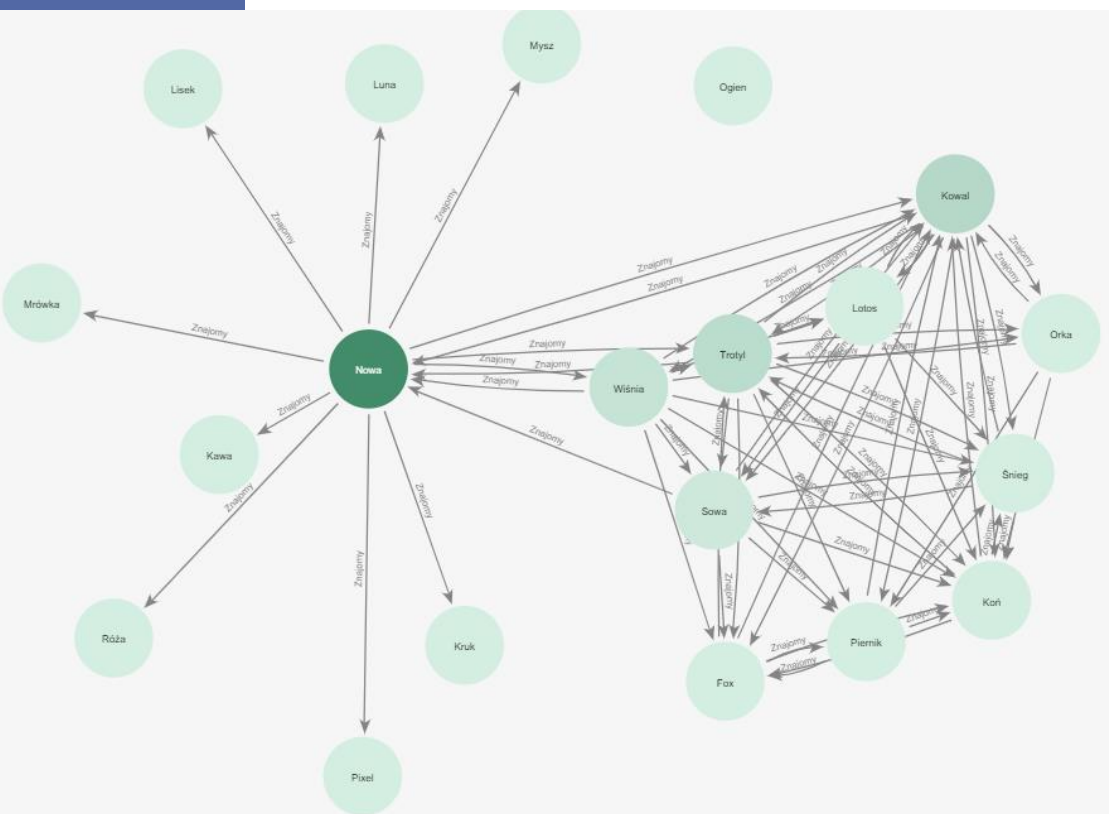
Najbardziej wpływowe osoby - PageRank

Relacje naturalne



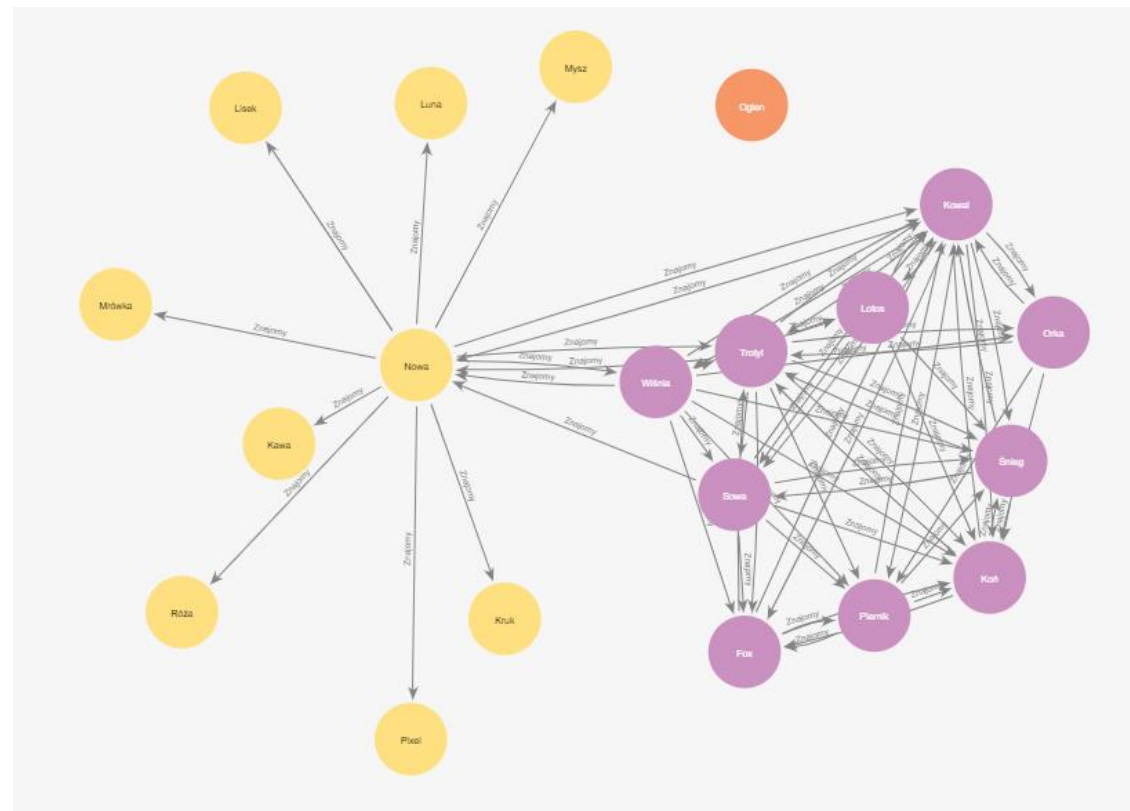
Relacje nieskierowane





**Osoby, które łączą różne grupy w sieci -
Betweenness Centrality**

Podgrupy przyjaźni – Louvain



Zadanie: analiza społeczności

Połączenie algorytmów PageRank i Louvain (rozmiar i kolor)

 Add algorithm

> Page Rank Centrality ● 

☒ Apply Styles

> Louvain ● 

☒ Apply Styles

Default **Rule-based** ●

 Add rule-based styling

>  Page Rank Centrality ● 

>  Louvain ● 

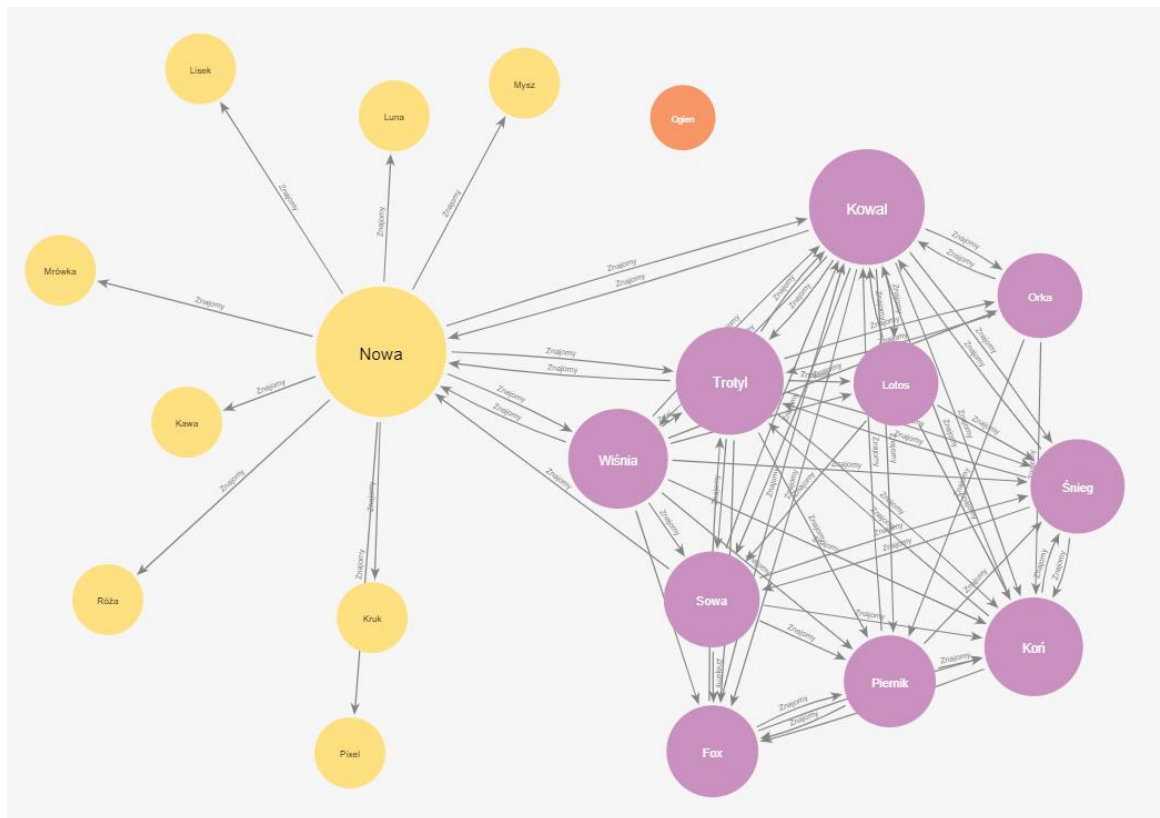
 Miasto

 Osoba

 Person

Zadanie: analiza społeczności

Połączenie algorytmów PageRank i Louvain (rozmiar i kolor)





Rozwiązanie w oparciu o:

**Neo4j Graph Data
Science Playground**



Stop ↺ Open ⌵ ...

- Neo4j Browser
- ⚙ Neo4j Bloom
- 👤 Graph Data Scien...
- JFrog Landing
- ⚙ Neo4j ETL Tool
- > Terminal



Reveal files in file explorer

Filename ▾

Graph Data Science
Playground



Wybór bazy danych

Launching NEuler - The Graph Data Science Playground

✓ Connecting to server ✓ Checking GDS plugin ✓ Checking APOC plugin  Select database

Connected to: neo4j@bolt://localhost:7687

Select database

neo4j (default) ▾

Database Contents

Refresh

Node Labels

Osoba Miasto Person Book

Relationship Types

KNOWS Mieszka_w READ Znajomy

Select database →

Update Osoba

Color

Caption

☒ nick

☐ id

☐ pagerank

☐ plec

☐ email

☐ wiek

Wybór właściwości i koloru węzła

Pojedynczy algorytm
lub zestaw algorytmów



Run single algorithm

Choose this option if you want to choose from any of the available algorithms in the Graph Data Science Library.

Select



Run Algorithm Recipe

Choose this option if you aren't sure which algorithm to run and would like to suggestions of commonly used combinations of algorithms.

Select

Algorithm

Page Rank

Measures the transitive influence or connectivity of nodes

Centralities

◀ Degree

Eigenvector

Page Rank

Article Rank

Betweenness

Approx Betweenness

Closeness

HITS ▶

Community Detection

◀ Louvain

Modularity Optimization

Label Propagation

K1-Coloring

Connected Components

Strongly Connected Components

Path Finding

◀ Shortest Path

A*

Single Source Shortest Path

All Pairs Shortest Path ▶

Similarity

◀ Node Similarity

K-Nearest Neighbors ▶

Graph Embeddings

◀ Node2Vec

FastRP ▶

Wybór algorytmu

Projected Graph

Label

Osoba

Relationship Type

Any

Relationship Orientation

Undirected

Weight Property

Weight Property

Parametry projekcji grafu

1. Configure 2. Results 3. Code

Table

Chart

Visualisation

Osoba

Node	Score
Nowa	2.9550320674726316
Kowal	2.1490149087177692
Trotyl	1.7659488507357377
Wiśnia	1.3989424690189816
Koń	1.3484681705126944
Sowa	1.230033106188341
Śnieg	1.164271637456127
Piernik	1.0681092838323
Fox	1.0644789513010575

Wyniki

Kod Cypher

1. Configure 2. Results 3. Code

Generate Neo4j Browser Guide

You can generate a Neo4j Browser guide that contains the code to reproduce the algorithm run:

▶ Send to Neo4j Browser

Run code fragments

Or you can reproduce the algorithm run by running the following code fragments:

Named Graph

A named graph is created in memory and remains there until it is explicitly deleted. Multiple algorithms can be run against a single named graph.

```
:use neo4j;
```

Copy

```
:param limit => ( 42);

:param config => ({
  relationshipWeightProperty: null,
  dampingFactor: 0.85,
  maxIterations: 20
});
```

Copy

```
:param graphConfig => ({
  nodeProtection: 'Osoba'
```

Zadanie: analiza społeczności

Porównać wyniki generowane przez Neo4j Browser z Graph Data Science Playground

Zadanie do realizacji własnej.....

Trasy w logistyce

Obiekt analizy grafowej

Mapa tras



Cel analizy grafowej

Wyszukiwanie i dopasowanie tras
drogowych na potrzeby logistyki

Zadanie: trasy w logistyce

Budowa grafu

Przekopiuj do katalogu /import pliki „trasaN.csv” i „trasaR.csv”

```
MATCH (n) DETACH DELETE n;
```

```
LOAD CSV WITH HEADERS FROM 'file:///trasaN.csv' AS row
```

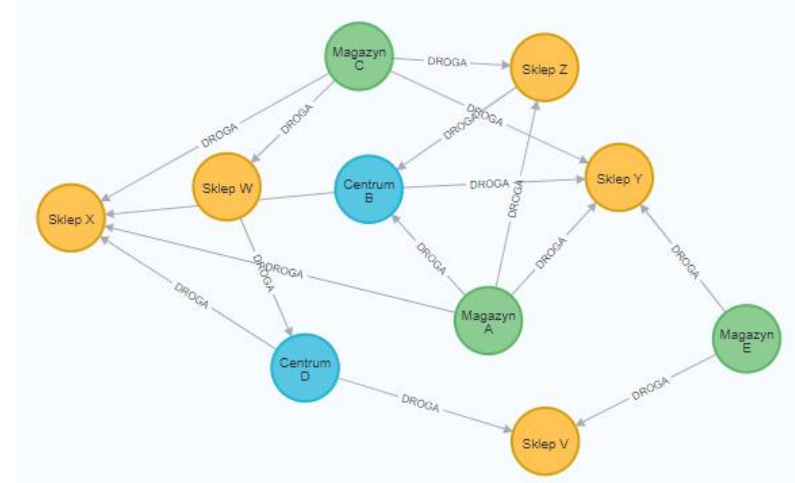
```
CALL apoc.create.node([row.typ, 'Lokalizacja'], {id: toInteger(row.id), nazwa: row.nazwa, lokalizacja: point({latitude: toFloat(row.latitude), longitude: toFloat(row.longitude)}})) YIELD node
```

```
RETURN node;
```

```
LOAD CSV WITH HEADERS FROM 'file:///trasaR.csv' AS row
```

```
MATCH (start:Lokalizacja {id: toInteger(row.start_id)}), (end:Lokalizacja {id: toInteger(row.end_id)})
```

```
CREATE (start)-[:DROGA {odległość: toInteger(row.odległość), czas_przejazdu: toFloat(row.czas_przejazdu)}]->(end);
```



Zadanie: trasy w logistyce

Proponowane algorytmy

Znajdź najkrótsze trasy między magazynami a punktami dostawy na podstawie odległości.

Algorytm Najkrótszej Ścieżki (Shortest Path)

Znajdź najlepiej usytuowane magazyny, które minimalizują czas dostawy do wszystkich innych lokalizacji.

Centralność Bliskości (Closeness Centrality)

Znajdź Centrum Dostaw, którego awaria będzie najbardziej dotkliwa.

Centralność Pośrednictwa (Betweenness Centrality)

Zadanie: trasy w logistyce

Budowa grafu w pamięci

```
CALL gds.graph.project(
  'mapa',
  {
    PunktDostawy: {
      label: 'PunktDostawy'
    },
    Magazyn: {
      label: 'Magazyn'
    },
    CentrumDystrybucji: {
      label: 'CentrumDystrybucji'
    }
  },
  {
    DROGA: {
      type: 'DROGA',
      properties: ['odległość'],
      orientation: 'UNDIRECTED'
    }
  }
);
```

Zadanie: trasy w logistyce

Najkrótsze trasy między magazynami a punktami dostawy na podstawie odległości

Algorytm: Dijkstry Shortest Path

Algorytm Dijkstry Shortest Path oblicza najkrótszą ścieżkę między węzłami w kontekście wagi. Algorytm obsługuje grafy ważone z dodatnimi wagami relacji.

Odległość to waga relacji DROGA

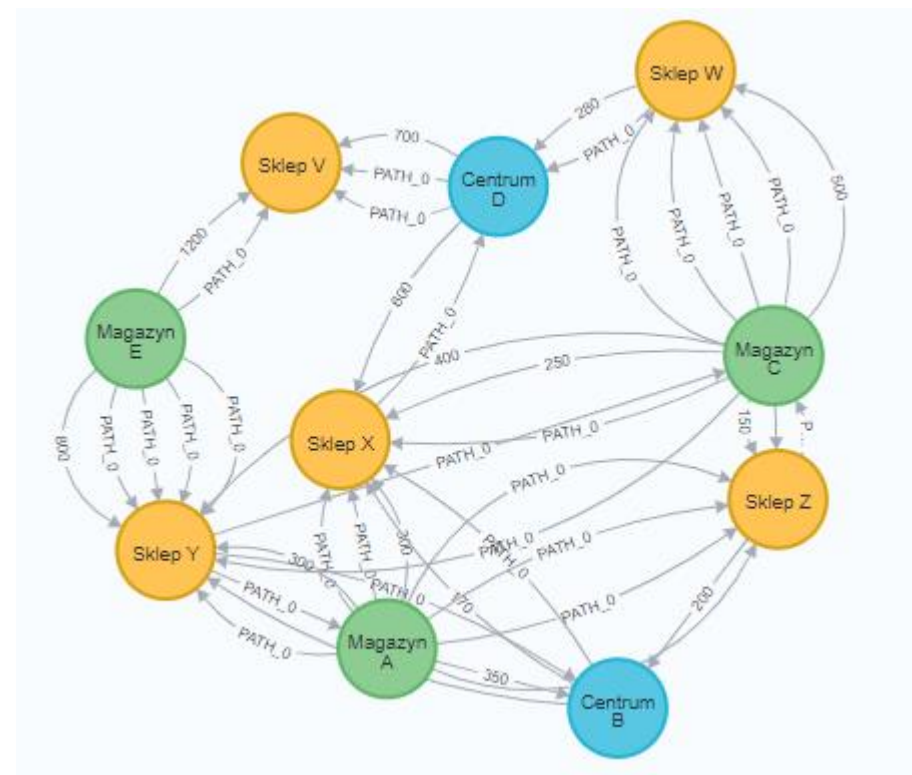
```
MATCH (start:Magazyn), (end:PunktDostawy)
CALL gds.shortestPath.dijkstra.stream('mapa', {
  sourceNode: id(start),
  targetNode: id(end),
  relationshipWeightProperty: 'odległość'
})
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs
MATCH (m) WHERE id(m) = sourceNode
MATCH (p) WHERE id(p) = targetNode
RETURN m.nazwa AS Magazyn, p.nazwa AS PunktDostawy, total
Cost+' km' AS Odległość, nodeIds, costs
ORDER BY m.nazwa, totalCost;
```

Zadanie: trasy w logistyce

Najkrótsza trasa między magazynem a punktem dostawy na podstawie odległości

Wynik

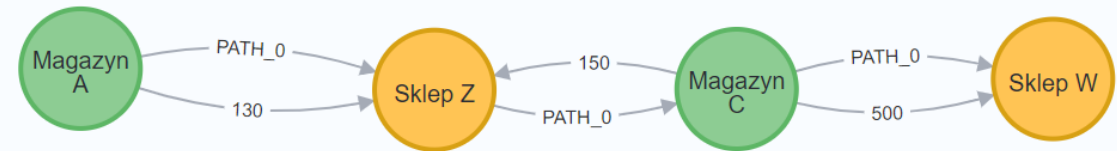
Magazyn	PunktDostawy	Odległość	nodeIds	costs
"Magazyn A"	"Sklep Z"	"130.0 km"	[5, 9]	[0.0, 130.0]
"Magazyn A"	"Sklep Y"	"300.0 km"	[5, 7]	[0.0, 300.0]
"Magazyn A"	"Sklep X"	"300.0 km"	[5, 6]	[0.0, 300.0]
"Magazyn A"	"Sklep W"	"780.0 km"	[5, 9, 10, 11]	[0.0, 130.0, 280.0, 780.0]
"Magazyn A"	"Sklep V"	"1600.0 km"	[5, 6, 12, 13]	[0.0, 300.0, 900.0, 1600.0]
"Magazyn C"	"Sklep Z"	"150.0 km"	[10, 9]	[0.0, 150.0]
"Magazyn C"	"Sklep X"	"250.0 km"	[10, 6]	[0.0, 250.0]
"Magazyn C"	"Sklep Y"	"400.0 km"	[10, 7]	[0.0, 400.0]
"Magazyn C"	"Sklep W"	"500.0 km"	[10, 11]	[0.0, 500.0]
"Magazyn C"	"Sklep V"	"1480.0 km"	[10, 11, 12, 13]	[0.0, 500.0, 780.0, 1480.0]
"Magazyn E"	"Sklep Y"	"800.0 km"	[14, 7]	[0.0, 800.0]
"Magazyn E"	"Sklep V"	"1200.0 km"	[14, 13]	[0.0, 1200.0]
"Magazyn E"	"Sklep Z"	"1230.0 km"	[14, 7, 5, 9]	[0.0, 800.0, 1100.0, 1230.0]
"Magazyn E"	"Sklep X"	"1240.0 km"	[14, 7, 8, 6]	[0.0, 800.0, 1070.0, 1240.0]
"Magazyn E"	"Sklep W"	"1700.0 km"	[14, 7, 10, 11]	[0.0, 800.0, 1200.0, 1700.0]



Zadanie: trasy w logistyce

Najkrótsza trasa między magazynem a punktem dostawy na podstawie odległości

```
MATCH (start:Magazyn {id: 1}), (end:PunktDostawy {id: 7})
CALL gds.shortestPath.dijkstra.stream('mapa', {
  sourceNode: id(start),
  targetNode: id(end),
  relationshipWeightProperty: 'odległość'
})
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
RETURN sourceNode, targetNode, totalCost, nodeIds, costs, path;
```



Zadanie: trasy w logistyce

Składnia gds.shortestPath.stream

```
CALL gds.shortestPath.dijkstra.stream(  
  graphName: String,  
  configuration: Map  
)
```

YIELD

index: Integer,
sourceNode: Integer,
targetNode: Integer,
totalCost: Float,
nodeIds: List of Integer,
costs: List of Float,
path: Path

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
graphName	String	n/a	nie	Nazwa grafu przechowywanego w katalogu.
configuration	Mapa	{}	tak	Konfiguracja specyficzna dla algorytmu i/lub filtrowania grafu.

Konfiguracja

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
nodeLabels	Lista Stringów	[*]	tak	Filtruje nazwany graf używając podanych etykiet węzłów. Węzły z dowolną z podanych etykiet będą uwzględnione.
relationshipTypes	Lista Stringów	[*]	tak	Filtruje nazwany graf używając podanych typów relacji. Relacje z dowolnym z podanych typów będą uwzględnione.
concurrency	Liczba całkowita	1	tak	Algorytm jest jednowątkowy, zmiana parametru współbieżności nie ma wpływu na czas wykonania.
jobId	String	Generowane wewnętrznie	tak	ID, które można podać, aby łatwiej śledzić postęp algorytmu.
logProgress	Boolean	true	tak	Jeśli wyłączone, procentowy postęp nie będzie logowany.
sourceNode	Liczba całkowita	n/d	nie	Węzeł źródłowy lub ID węzła w Neo4j.
targetNode	Liczba całkowita	n/d	nie	Węzeł docelowy lub ID węzła w Neo4j.
relationshipWeightProperty	String	null	tak	Nazwa właściwości relacji używana jako wagi. Jeśli nie podana, algorytm działa bez wag.

Zadanie: trasy w logistyce

Składnia gds.**shortestPath**.stream

Wyniki

Nazwa	Typ	Opis
index	Liczba całkowita	Indeks znalezionej ścieżki, zaczynający się od 0.
sourceNode	Liczba całkowita	Węzeł źródłowy ścieżki.
targetNode	Liczba całkowita	Węzeł docelowy ścieżki.
totalCost	Liczba zmiennoprzecinkowa	Całkowity koszt od źródła do celu.
nodeIds	Lista Liczb całkowitych	Identyfikatory węzłów na ścieżce w kolejności przeszukiwania.
costs	Lista Liczb zmiennoprzecinkowych	Suma kosztów dla każdego węzła na ścieżce.
path	Ścieżka	Ścieżka reprezentowana jako encja Cypher.

Zadanie: trasy w logistyce

Najlepiej usytuowane magazyny

Algorytm: Bliskość Centralna (Closeness Centrality)

Pokazuje, jak blisko dany magazyn jest do wszystkich innych lokacji w sieci, co może wskazywać na szybkość, z jaką towary mogą być dostarczane z tego magazynu.

```
CALL gds.closeness.stream('mapa')
```

```
YIELD nodeId, score
```

```
MATCH (n:Magazyn) WHERE id(n) = nodeId
```

```
RETURN id(n), n.nazwa, score
```

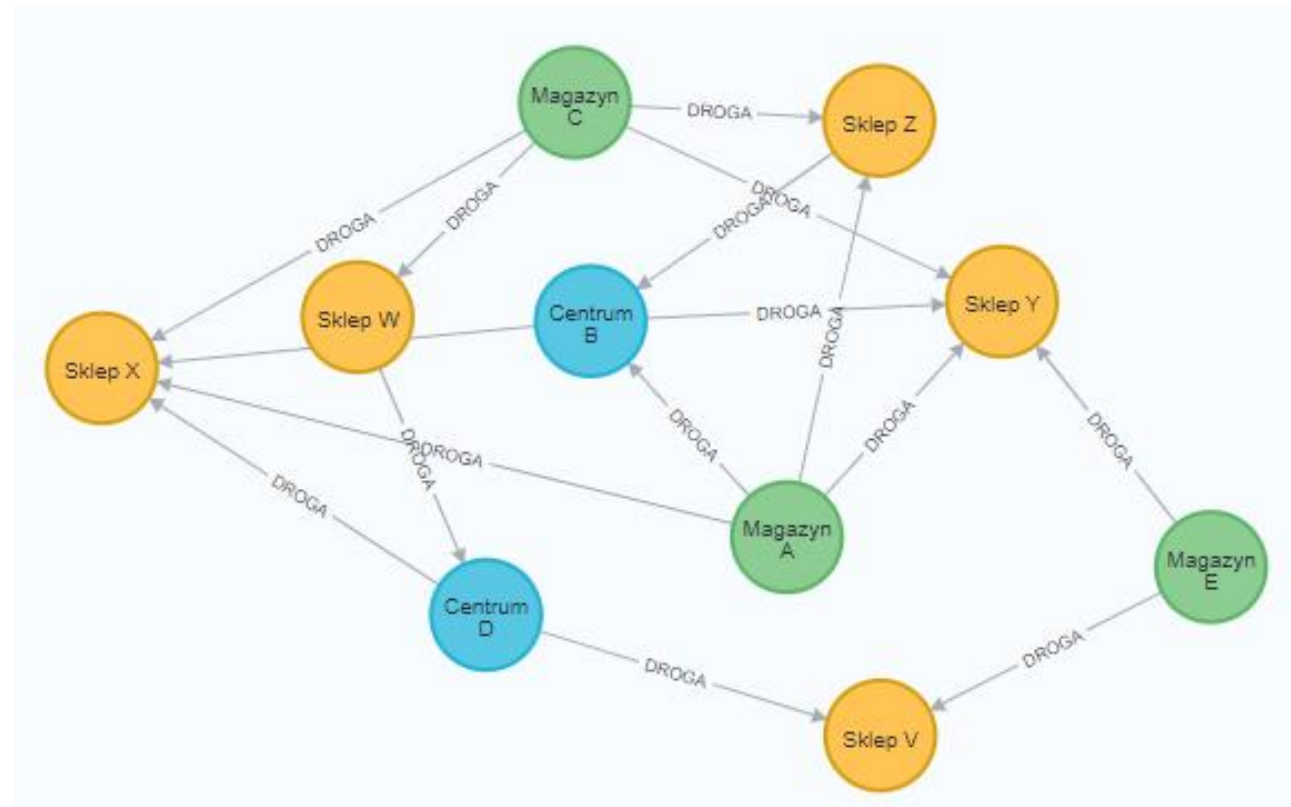
```
ORDER BY score DESC
```

Zadanie: trasy w logistyce

Najlepiej usytuowane magazyny

Wynik

id(n)	n.nazwa	score
10	"Magazyn C"	0,6
5	"Magazyn A"	0,5625
14	"Magazyn E"	0,473684211



Zadanie: trasy w logistyce

Którego Centrum Dystrybucji awaria będzie najbardziej dotkliwa?

Zadanie do realizacji własnej.....

Budowa sieci komputerowej

Obiekt analizy grafowej

Infrastruktura kompleksu



Cel analizy grafowej

Projekt optymalnej sieci komputerowej

Zadanie: Budowa sieci komputerowej

Treść zadania

Masz kompleks złożony z N budynków. Każdy budynek ma swój numer od 1 do N . Każdy budynek ma także swoją lokalizację na mapie określoną współrzędnymi geograficznymi. Odległość pomiędzy każdymi dwoma budynkami jest określona przez $d(i,j)$, gdzie i oraz j to numery budynków.

Twoje zadanie polega na:

1. Światłowód: Znalezieniu najkrótszej drogi (minimalna suma odległości d), która połączy wszystkie budynki.
2. WiFi: Wyborze $N/2$ budynków, w których zostaną rozlokowane punkty dostępu Wi-Fi, tak aby pokryć jak największą liczbę budynków.

Zadanie: Budowa sieci komputerowej

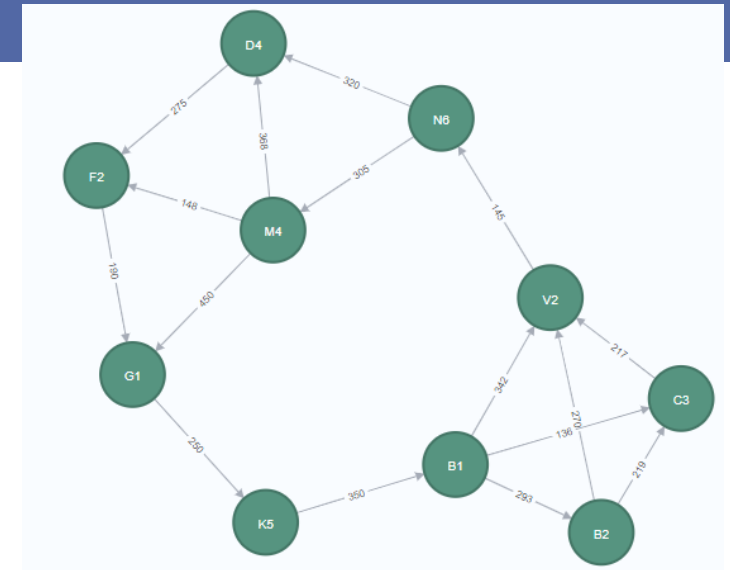
Budowa grafu

Przekopiuj do katalogu /import pliki „kompleksN.csv” i „kompleksR.csv”

```
MATCH (n) DETACH DELETE n;
```

```
LOAD CSV WITH HEADERS FROM 'file:///kompleksN.csv' AS row  
CREATE (b:Budynek {id: toInteger(row.id), nazwa: row.nazwa,  
users: toInteger(row.users), latitude: toFloat(row.latitude), longitude: toFloat(row.longitude)});
```

```
LOAD CSV WITH HEADERS FROM 'file:///kompleksR.csv' AS row MATCH (b1:Budynek {id: toInteger(row.start_id)}), (b2  
:Budynek {id: toInteger(row.end_id)}) CREATE (b1)-[:Odleglosc {metry: toInteger(row.distance)}]->(b2);
```



Zadanie: Budowa sieci komputerowej

Proponowane algorytmy

Światłowód: Znalezieniu najkrótszej drogi (minimalna suma odległości d), która połączy wszystkie budynki.

Algorytm Minimalnego Drzewa Rozpinającego (MST)

WiFi: Wyborze $N/2$ budynków, w których zostaną rozlokowane punkty dostępu Wi-Fi, tak aby pokryć jak największą liczbę budynków.

Centralność Bliskości (Closeness Centrality)

Zadanie: Budowa sieci komputerowej

Budowa grafu w pamięci

```
CALL gds.graph.project(  
  'kompleks',  
  ['Budynek'],  
  {  
    DROGA: {  
      type: 'Odleglosc',  
      properties: ['metry'],  
      orientation: 'UNDIRECTED'  
    }  
  }  
);
```

Zadanie: Budowa sieci komputerowej

Najkrótsza droga (minimalna suma odległości d), która połączy wszystkie budynki

Algorytm: Algorytm Minimalnego Drzewa Rozpinającego (MST)

Drzewo rozpinające o minimalnej wadze (MST) rozpoczyna się od danego węzła, znajduje wszystkie osiągalne węzły i zwraca zbiór relacji, które łączą te węzły ze sobą, mając minimalną możliwą wagę.

Funkcja `gds.util.asNode` jest używana do konwersji identyfikatorów węzłów na same węzły w wynikach zapytań, co umożliwia łatwiejsze odczytanie właściwości węzłów.

```
MATCH (n:Budynek {id: 1})
CALL gds.spanningTree.stream('kompleks', {
  sourceNode: n,
  relationshipWeightProperty: 'metry'
})
```

```
[ YIELD nodeId, parentId, weight
  WITH gds.util.asNode(nodeId) AS node, gds.util.asNode(parentId) AS
  parent, weight
  MERGE (node)-[m:MST {weight: weight}]->(parent)
  RETURN node, m, parent
```

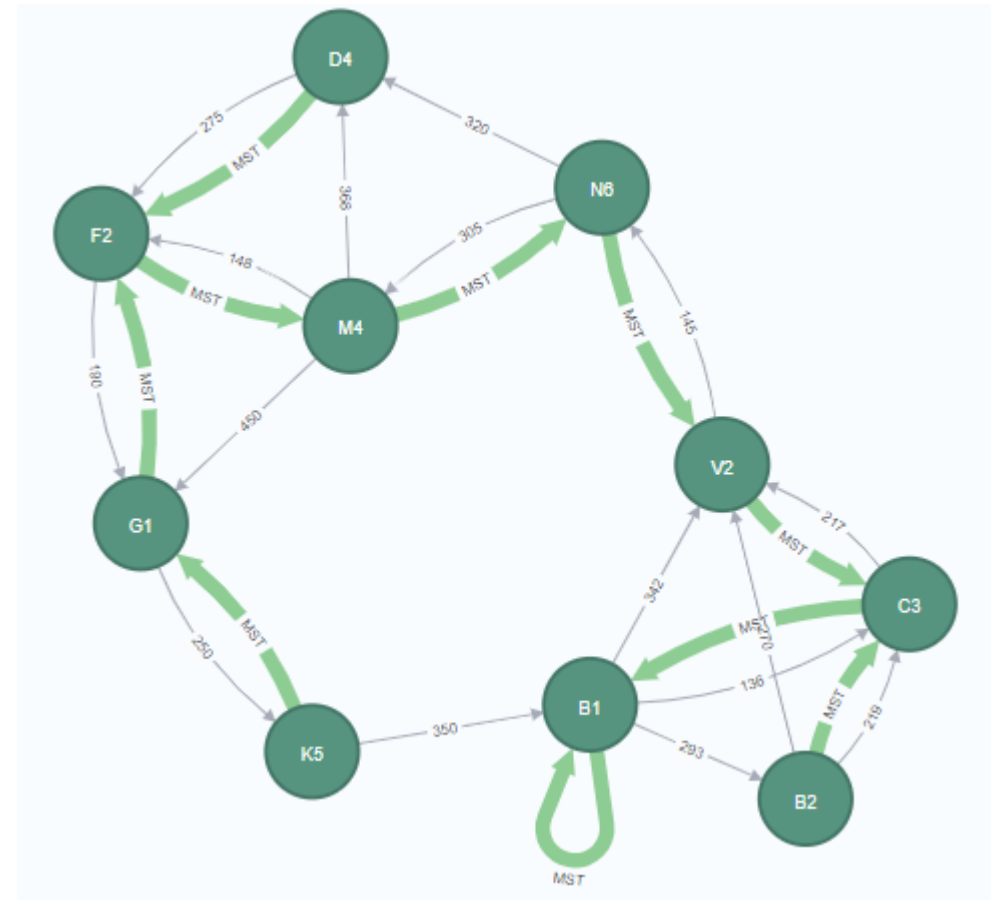
```
[ YIELD nodeId, parentId, weight
  WITH gds.util.asNode(nodeId) AS node, gds.util.asNode(parentId) AS
  parent, weight
  RETURN node.nazwa AS BudynekOd, parent.nazwa AS BudynekDo, we
  ight AS Metry
```

Zadanie: Budowa sieci komputerowej

Najkrótsza droga (minimalna suma odległości d),
która połączy wszystkie budynki

Wynik

BudynekOd	BudynekDo	Metry
"B1"	"B1"	0.0
"B2"	"C3"	219.0
"C3"	"B1"	136.0
"V2"	"C3"	217.0
"N6"	"V2"	145.0
"M4"	"N6"	305.0
"D4"	"F2"	275.0
"F2"	"M4"	148.0
"G1"	"F2"	190.0
"K5"	"G1"	250.0



Zadanie: Budowa sieci komputerowej

Składnia gds.spanningTree.stream

CALL gds.spanningTree.stream(
graphName: String,
configuration: Map
)

YIELD

nodeId: Integer,
parentId: Integer,
weight: Float

Konfiguracja

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
graphName	String	n/a	nie	Nazwa grafu przechowywanego w katalogu.
configuration	Mapa	{}	tak	Konfiguracja specyficzna dla algorytmu i/lub filtrowania grafu.

Nazwa	Typ	Domyślnie	Opcjonalne	Opis
nodeLabels	Lista Stringów	[*]	tak	Filtruje nazwany graf używając podanych etykiet węzłów. Węzły z dowolną z podanych etykiet będą uwzględnione.
relationshipTypes	Lista Stringów	[*]	tak	Filtruje nazwany graf używając podanych typów relacji. Relacje z dowolnym z podanych typów będą uwzględnione.
concurrency	Liczba całkowita	1	tak	Algorytm jest jednowątkowy i zmiana parametru współbieżności nie ma wpływu na czas wykonania.
jobId	String	Generowane wewnętrznie	tak	ID, które można podać, aby łatwiej śledzić postęp algorytmu.
logProgress	Boolean	true	tak	Jeśli wyłączone, procentowy postęp nie będzie logowany.
sourceNode	Liczba całkowita	null	n/d	ID węzła początkowego.
relationshipWeightProperty	String	null	tak	Nazwa właściwości relacji używanej jako wagi. Jeśli nie podana, algorytm działa bez wag.
objective	String	'minimum'	tak	Jeśli podany, parametr określa, czy znaleźć minimalne, czy maksymalne drzewo rozpinające. Domyślnie zwracane jest minimalne drzewo rozpinające. Dopuszczalne wartości to 'minimum' i 'maximum'.

Zadanie: Budowa sieci komputerowej

Składnia gds.spanningTree.stream

Wyniki

Nazwa	Typ	Opis
nodeId	Integer	Węzeł w odkrytym drzewie rozpinającym.
parentId	Integer	Rodzic nodeId w drzewie rozpinającym lub nodeId, jeśli jest równe węzłowi początkowemu.
weight	Float	Waga relacji od parentId do nodeId.

Zadanie: Budowa sieci komputerowej

Lokalizacja punktów dostępowych Wi-Fi, tak aby pokryć jak największą liczbę budynków

Algorytm: Centralność Bliskości (Closeness Centrality)

Oszacowanie $N/2$ ruterów

`MATCH (b:Budynek) RETURN (count(b) / 2) AS limit;`

Wyniki jako LIMIT dla ORDER BY

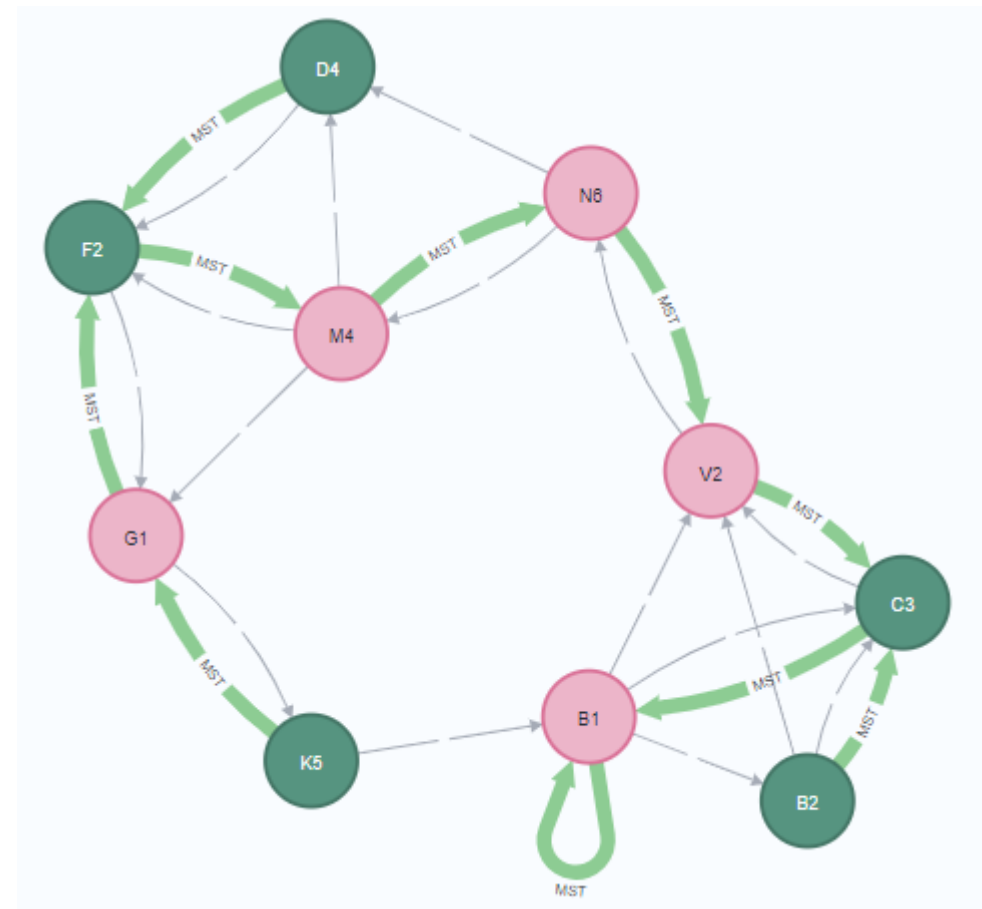
```
CALL gds.closeness.stream('kompleks')
YIELD nodeId, score
MATCH (b:Budynek) WHERE id(b) = nodeId
WITH b, score
ORDER BY score DESC
WITH b, score
LIMIT 5
SET b:WiFi
RETURN b.nazwa, score;
```

Zadanie: Budowa sieci komputerowej

Lokalizacja punktów dostępowych Wi-Fi, tak aby pokryć jak największą liczbę budynków

Wynik

Budynek	score
„N6”	0.5625
„V2”	0.5625
„M4”	0.5294117647058824
„B1”	0.5294117647058824
„G1”	0.5



Zadanie: Budowa sieci komputerowej

Który z budynków w kompleksie najlepiej nadaje się na sztab?

Zadanie do realizacji własnej.....

- **Betweenness Centrality:** Dobrze do identyfikacji węzłów, które kontrolują przepływ informacji.
- **Closeness Centrality:** Użyteczne do znalezienia węzłów, które mogą szybko komunikować się z innymi.
- **Degree Centrality:** Proste i szybkie, wskazuje na węzły z dużą liczbą bezpośrednich połączeń.
- **PageRank:** Pomaga znaleźć węzły z dużym wpływem w sieci.

A może ranking ?

Incydenty bezpieczeństwa

Obiekt analizy grafowej

System Teleinformatyczny



Cel analizy grafowej

Zwiększenie poziomu bezpieczeństwa systemu teleinformatycznego

Zadanie: Incydenty bezpieczeństwa

Treść zadania

Od tygodnia odpowiadasz za bezpieczeństwo systemu teleinformatycznego w organizacji. Organizacja posiada sieć komputerową złożoną z takich elementów jak stacje robocze (komputery), serwery i urządzenia sieciowe (rutery). W sieci dochodzi często do incydentów bezpieczeństwa w postaci ataków DDoS, malware oraz phishing. Widzisz, że dotychczasowe działania obronne ograniczały się do postawienia Firewall, objęcia komputerów ochroną AV i skrupulatną ewidencją incydentów bezpieczeństwa.

Masz plan na zwiększenie poziomu bezpieczeństwa poprzez:

1. Mitygację najbardziej podobnych incydentów.
2. Zwiększenie ochrony najbardziej zagrożonych incydentami elementów.

Zadanie: Incydy bezpieczeństwa

Budowa grafu

Przekopiuj do katalogu /import pliki „incydentN.csv”, „incydentRincydent.csv” i „incydentNelement.csv”

MATCH (n) **DETACH DELETE** n;

LOAD CSV WITH HEADERS FROM 'file:///incydentN.csv' **AS** row

CREATE (:Incydent {id: toInteger(row.id_incydentu), typ: row.typ_incydentu, czas: datetime(row.czas_wystapienia), kod: row.kod});

LOAD CSV WITH HEADERS FROM 'file:///incydentNelement.csv' **AS** row

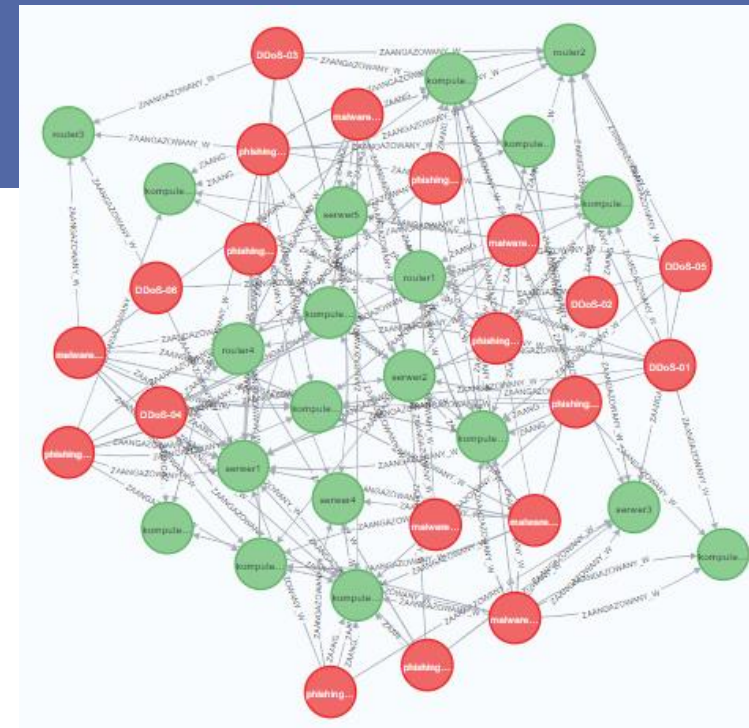
CREATE (:Element {id: toInteger(row.id_elementu), nazwa: row.nazwa_elementu, typ: row.typ_elementu});

LOAD CSV WITH HEADERS FROM 'file:///incydentRincydent.csv' **AS** row

MATCH (i:Incydent {id: toInteger(row.id_incydentu)})

MATCH (e:Element {id: toInteger(row.id_elementu)})

CREATE (i)-[:ZAANGAZOWANY_W]->(e);



Zadanie: Incydenty bezpieczeństwa

Proponowane algorytmy

Podobieństwo incydentów. Mitygację najbardziej podobnych incydentów.

Algorytm: Node Similarity

Przewidywanie incydentów. Zwiększenie ochrony najbardziej zagrożonych incydentami elementów.

Algorytm: Adamic Adar

Zadanie: Incydenty bezpieczeństwa

Budowa grafu w pamięci

```
CALL gds.graph.project(  
  'monitoring',  
  '*',  
  {  
    relType: {  
      type: 'ZAANGAZOWANY_W',  
      orientation: 'NATURAL'  
    }  
  }  
);
```

Tylko krawędzie naturalne czyli skierowane !



Zadanie: Incydenty bezpieczeństwa

Podobieństwo incydentów

Algorytm: Node Similarity

Porównuje zestaw węzłów na podstawie węzłów, z którymi są połączone. Dwa węzły są uważane za podobne, jeśli mają wielu takich samych sąsiadów. Node Similarity oblicza podobieństwa parami w oparciu o metrykę Jaccarda.

```
CALL gds.nodeSimilarity.stream('monitoring')
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).kod AS Incydent1,
gds.util.asNode(node2).kod AS Incydent2, similarity
ORDER BY similarity DESCENDING, Incydent1, Incydent2
```

Zadanie: Incydenty bezpieczeństwa

Podobieństwo incydentów

Wynik

Incydent1	Incydent2	similarity
"DDoS-04"	"phishing-01"	0.5
"malware-02"	"malware-05"	0.5
"malware-05"	"malware-02"	0.5
"malware-05"	"phishing-03"	0.5
"phishing-01"	"DDoS-04"	0.5
"phishing-01"	"phishing-05"	0.5
"phishing-03"	"malware-05"	0.5
"phishing-05"	"phishing-01"	0.5
"DDoS-05"	"malware-04"	0.4444444444444444
"malware-04"	"DDoS-05"	0.4444444444444444
"malware-05"	"phishing-08"	0.4444444444444444
"phishing-08"	"malware-05"	0.4444444444444444
"DDoS-01"	"malware-06"	0.42857142857142855
"malware-06"	"DDoS-01"	0.42857142857142855
"DDoS-04"	"malware-03"	0.4166666666666667
"malware-03"	"DDoS-04"	0.4166666666666667
"malware-01"	"phishing-03"	0.4
"malware-02"	"phishing-03"	0.4
"phishing-02"	"phishing-04"	0.4
"phishing-03"	"malware-01"	0.4
"phishing-03"	"malware-02"	0.4
"phishing-04"	"phishing-02"	0.4
"malware-03"	"malware-05"	0.38461538461538464
"malware-05"	"malware-03"	0.38461538461538464
"malware-02"	"phishing-05"	0.375
"phishing-02"	"phishing-06"	0.375
"phishing-05"	"malware-02"	0.375
"phishing-06"	"phishing-02"	0.375
"malware-01"	"malware-05"	0.36363636363636365

Wnioski:

- Podobne przyczyny:** Incydenty o wysokim współczynniku podobieństwa mogą mieć wspólne przyczyny lub być wynikiem podobnych luk w zabezpieczeniach.
- Wspólne cele:** Podobne incydenty mogą wskazywać na wspólne cele ataków, na przykład ataki skierowane na konkretne komponenty systemu lub dane.
- Źródła ataków:** Podobne incydenty mogą mieć wspólne źródła, takie jak określone grupy hakerskie lub narzędzia ataku.
- Skuteczniejsze reakcje:** Zrozumienie, które incydenty są do siebie podobne, może pomóc w opracowaniu bardziej skutecznych strategii reagowania na incydenty, ponieważ podobne incydenty mogą być rozwiązywane w podobny sposób.
- Identyfikacja trendów:** Monitorowanie podobieństwa incydentów w czasie może pomóc w identyfikacji trendów i przewidywaniu przyszłych zagrożeń.
- Grupowanie incydentów:** Wysoki stopień podobieństwa może być użyty do grupowania incydentów w klastry, co może pomóc w zarządzaniu i raportowaniu incydentów.
- Priorytetyzacja:** Klasyfikacja incydentów na podstawie podobieństwa może pomóc w ustaleniu priorytetów dla działań naprawczych.
- Powiązane elementy:** Możliwość powiązania incydentów z elementami, których dotyczą, pozwala na głębszą analizę i lepsze zrozumienie, które elementy systemu są najbardziej narażone na ataki.

Zadanie: Incydenty bezpieczeństwa

Powiązane elementy

```
CALL gds.nodeSimilarity.stream('monitoring')
YIELD node1, node2, similarity
WITH gds.util.asNode(node1) AS incydent1, gds.util.asNode(node2) AS incydent2, similarity
MATCH (incydent1)-[:ZAANGAZOWANY_W]->(element1:Element), (incydent2)-[:ZAANGAZOWANY_W]->(element2:Element)
RETURN incydent1.kod AS Incydent1, collect(DISTINCT element1.nazwa) AS Elementy1,
       incydent2.kod AS Incydent2, collect(DISTINCT element2.nazwa) AS Elementy2,
       similarity
ORDER BY similarity DESCENDING, Incydent1, Incydent2
```

Wynik:

Incydent1	Elementy1	Incydent2	Elementy2	similarity
"DDoS-04"	["serwer1", "komputer10", "komputer3", "router4", "komputer5", "router1", "serwer2"]	"phishing-01"	["serwer3", "router4", "serwer2", "komputer5", "serwer1"]	0.5
"malware-02"	["komputer6", "komputer9", "komputer5", "serwer2", "serwer4", "komputer1", "serwer1"]	"malware-05"	["serwer1", "komputer2", "komputer1", "komputer7", "komputer6", "komputer9", "router3", "komputer5"]	0.5
"malware-05"	["komputer5"]	"malware-02"	["komputer6", "komputer9", "komputer5", "serwer2", "serwer4", "komputer1", "serwer1"]	0.5
"malware-05"	["serwer1", "komputer2", "komputer1", "komputer7", "komputer6", "komputer9", "router3", "komputer5"]	"phishing-03"	["komputer1", "komputer5", "komputer4", "komputer2", "serwer1", "komputer11", "komputer6"]	0.5
"phishing-01"	["serwer3", "router4", "serwer2", "komputer5", "serwer1"]	"DDoS-04"	["serwer1", "komputer10", "komputer3", "router4", "komputer5", "router1", "serwer2"]	0.5
"phishing-01"	["serwer3", "router4", "serwer2", "komputer5", "serwer1"]	"phishing-05"	["serwer3", "serwer4", "komputer5", "serwer1"]	0.5
"phishing-03"	["komputer1", "komputer5", "komputer4", "komputer2", "serwer1", "komputer11", "komputer6"]	"malware-05"	["serwer1", "komputer2", "komputer1", "komputer7", "komputer6", "komputer9", "router3", "komputer5"]	0.5
"phishing-05"	["serwer3", "serwer4", "komputer5", "serwer1"]	"phishing-01"	["serwer3", "router4", "serwer2", "komputer5", "serwer1"]	0.5
"DDoS-05"	["serwer3", "router2", "router1", "komputer2"]	"malware-04"	["komputer2", "router2", "komputer8", "router1", "komputer1", "serwer5", "komputer6", "komputer9", "serwer3"]	0.4444444444444444
"malware-04"	["komputer2", "router2", "komputer8", "router1", "komputer1", "serwer5", "komputer6", "komputer9", "serwer3"]	"DDoS-05"	["serwer3", "router2", "router1", "komputer2"]	0.4444444444444444
"malware-05"	["serwer1", "komputer2", "komputer1", "komputer7", "komputer6", "komputer9", "router3", "komputer5"]	"phishing-08"	["komputer7", "serwer1", "komputer1", "komputer8", "komputer9"]	0.4444444444444444
"phishing-08"	["komputer7", "serwer1", "komputer1", "komputer8", "komputer9"]	"malware-05"	["serwer1", "komputer2", "komputer1", "komputer7", "komputer6", "komputer9", "router3", "komputer5"]	0.4444444444444444

Zadanie: Incydenty bezpieczeństwa

Przewidywanie incydentów

Algorytm: Adamic Adar

Algorytm przewidywania powiązań pomaga określić bliskość pary węzłów przy użyciu topologii grafu. Obliczone wyniki można wykorzystać do przewidywania nowych relacji między nimi.

```
MATCH (e1:Element)<-[:ZAANGAZOWANY_W]-(i:Incydent)-[:ZAANGAZOWANY_W]->(e2:Element)
WHERE e1 <> e2
WITH DISTINCT e1, e2, i
RETURN e1.nazwa AS Element1, e2.nazwa AS Element2, collect(i.kod) AS Incydenty, gds.alpha.linkprediction.adamicAdar(e1, e2) AS score
ORDER BY score DESCENDING, Element1, Element2;
```

Zadanie: Incydenty bezpieczeństwa

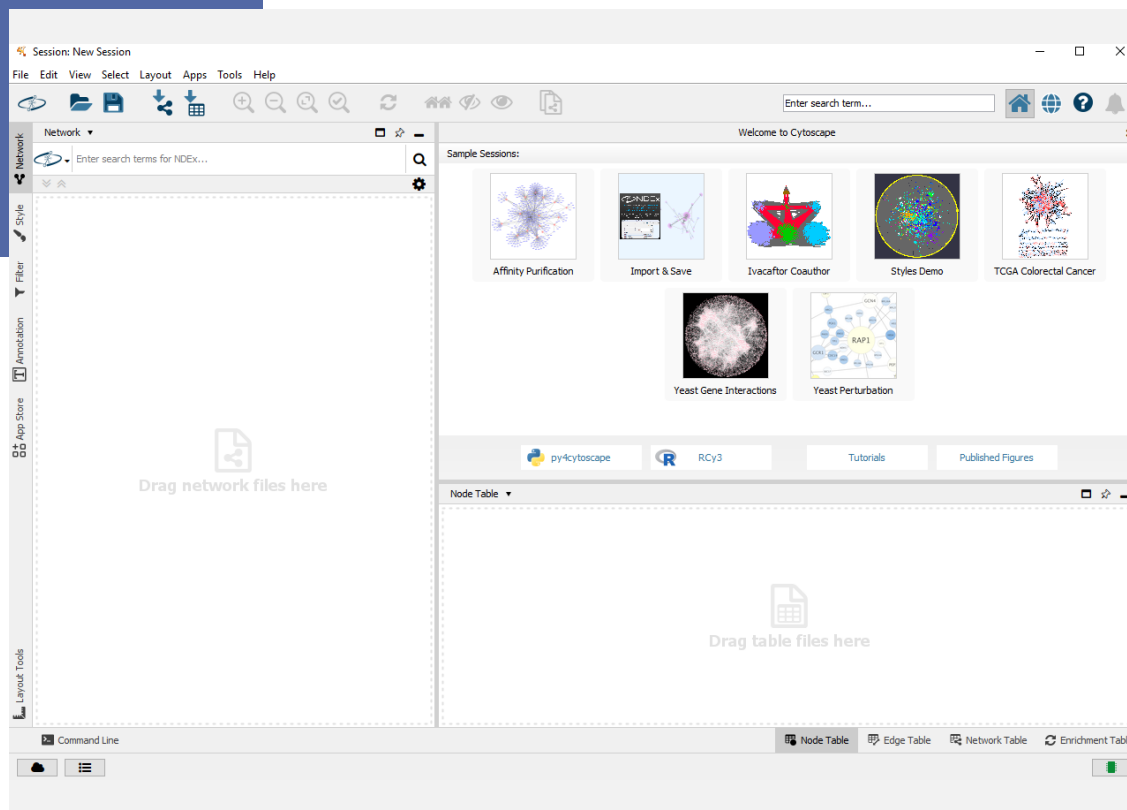
Przewidywanie incydentów

Wynik:

Element1	Element2	Incydenty	score
"komputer5"	"serwer1"	["phishing-01", "malware-02", "phishing-03", "malware-03", "phishing-05", "DDoS-04", "malware-05"]	3.375306142968446
"serwer1"	"komputer5"	["phishing-01", "malware-02", "phishing-03", "malware-03", "phishing-05", "DDoS-04", "malware-05"]	3.375306142968446
"komputer2"	"komputer6"	["malware-01", "phishing-03", "malware-03", "malware-04", "malware-05", "phishing-07"]	3.068012586595252
"komputer6"	"komputer2"	["malware-01", "phishing-03", "malware-03", "malware-04", "malware-05", "phishing-07"]	3.068012586595252
"serwer1"	"serwer2"	["phishing-01", "malware-02", "malware-03", "phishing-04", "DDoS-04", "phishing-06"]	2.9029244391760223
"serwer2"	"serwer1"	["phishing-01", "malware-02", "malware-03", "phishing-04", "DDoS-04", "phishing-06"]	2.9029244391760223
"router3"	"serwer1"	["DDoS-03", "phishing-04", "malware-05", "DDoS-06"]	2.3196767265055827
"serwer1"	"router3"	["DDoS-03", "phishing-04", "malware-05", "DDoS-06"]	2.3196767265055827
"komputer6"	"serwer1"	["malware-02", "phishing-03", "malware-03", "DDoS-03", "malware-05"]	2.2988515950954156
"serwer1"	"komputer6"	["malware-02", "phishing-03", "malware-03", "DDoS-03", "malware-05"]	2.2988515950954156
"router1"	"serwer2"	["DDoS-01", "malware-03", "DDoS-04", "malware-06"]	2.237510844677921
"serwer2"	"router1"	["DDoS-01", "malware-03", "DDoS-04", "malware-06"]	2.237510844677921

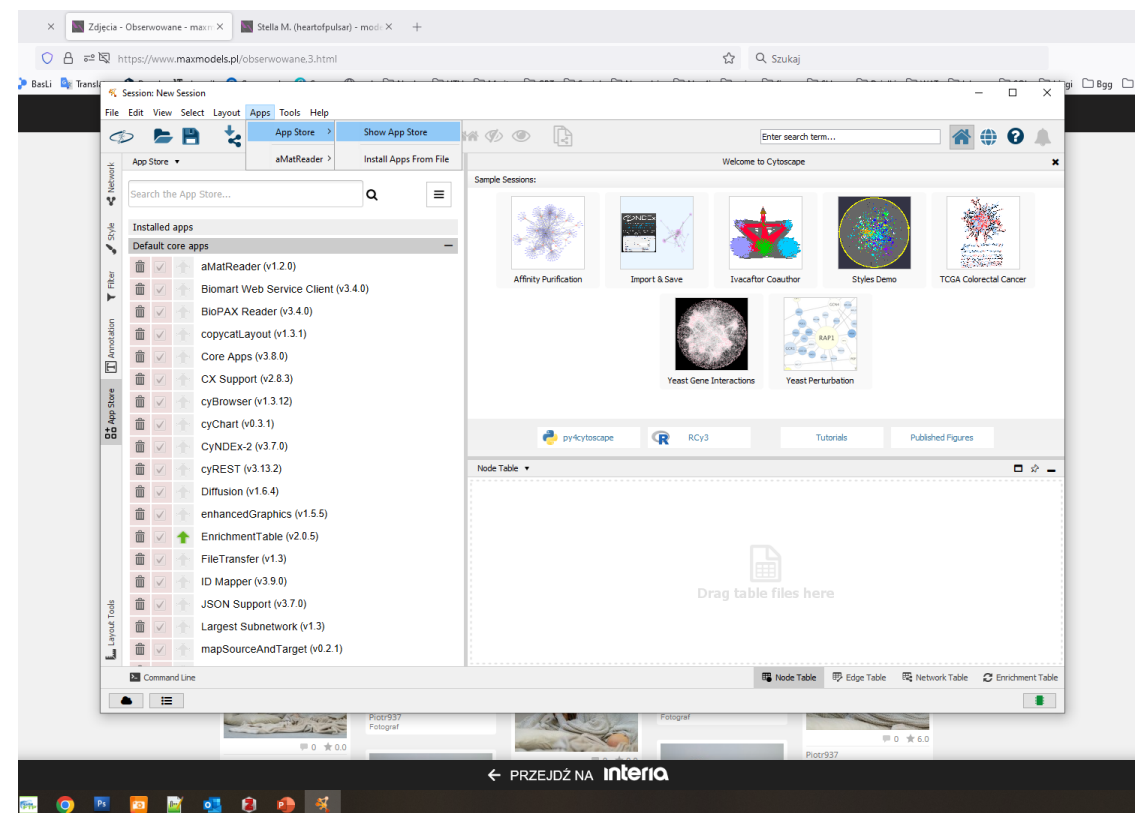
Wnioski:

- Przewidywanie przyszłych incydentów:** Wysoki wynik Adamic-Adar dla par elementów sugeruje, że te elementy mogą być zaangażowane w przyszłe incydenty. Proaktywne monitorowanie i wzmocnienie zabezpieczeń tych elementów może zapobiec przyszłym atakom.
- Kluczowe węzły:** Elementy o wysokim wyniku Adamic-Adar są często powiązane z wieloma incydentami, co wskazuje na ich krytyczną rolę w systemie. Wzmocnienie zabezpieczeń tych elementów może znacznie poprawić ogólny poziom bezpieczeństwa systemu.
- Wzorce ataków:** Analiza powiązań między elementami poprzez wspólne incydenty może ujawnić wzorce ataków, takie jak typy elementów najczęściej atakowane w określony sposób. Na przykład, jeśli określone typy urządzeń są często zaangażowane w różne incydenty, można podjąć kroki w celu zwiększenia ich zabezpieczeń.
- Wczesne ostrzeganie:** Elementy o wysokim wyniku mogą służyć jako wskaźniki wczesnego ostrzegania, pozwalając zespołom ds. bezpieczeństwa na szybszą reakcję na nowe zagrożenia.
- Zwiększenie efektywności SIEM:** Integracja wyników predykcji Adamic-Adar z systemami SIEM (Security Information and Event Management) pozwala na bardziej zaawansowane i ukierunkowane monitorowanie incydentów bezpieczeństwa.



Pulpit startowy

App Store



Apps



cyNeo4j

3.0+

A Cytoscape app to connect to a Neo4j database and execute



Cytoscape Neo4j Plugin

3.0+



Cytoscape RedisGraph
Plugin

3.0+

<https://apps.cytoscape.org/search?q=neo4j>

Cytoscape Neo4j Plugin



Cytoscape Neo4j Plugin

★★★★★ (5)

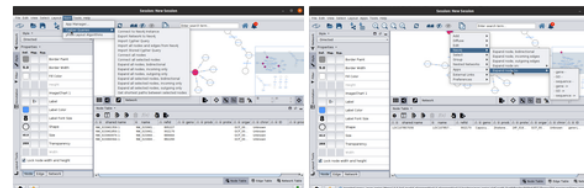
9219 downloads | discussions

3.0+

Details

Release History

Categories: [database](#), [graph database](#), [Neo4j](#)



Cytoscape plugin for Neo4j

Neo4j Graphs are often too large for Cytoscape: this plugin allows you to write Cypher queries and import the result as a network. Queries can be parameterized and stored for reuse.

Citation: [Mining functional annotations across species](#); Sven Warris, Steven Dijkshoorn, Teije van Sloten, T.L.H. van de Vossenbergh, [bioRxiv](#), 2018

CYTOSCAPE 3

✓ Installed

Or, for JAR file, click [here](#) to download

Version 0.5.1

Released 1 Feb 2022

Works with Cytoscape 3.9

Download Stats [Click here](#)

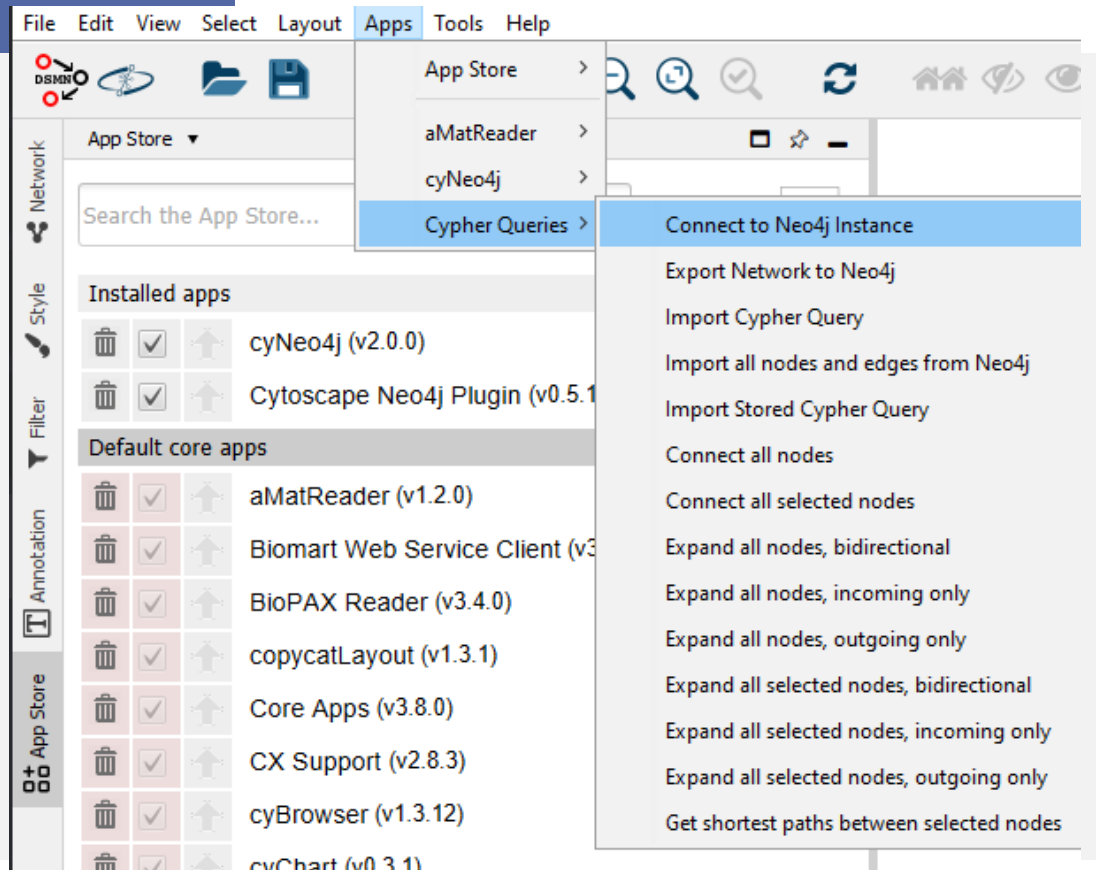
cytoscape v3.10.2

RESOURCES

🗨 Ask a question

🔍 Search BioStars

📄 Code Repository



Connect

Uwierzytelnienie

 Connect to Neo4j 

Protocol

BOLT

Hostname

localhost

Port

7687

Database

neo4j

Username

neo4j

Password

Cancel

Connect

Execute Cypher Query

network

Network

Visual Style

Directed

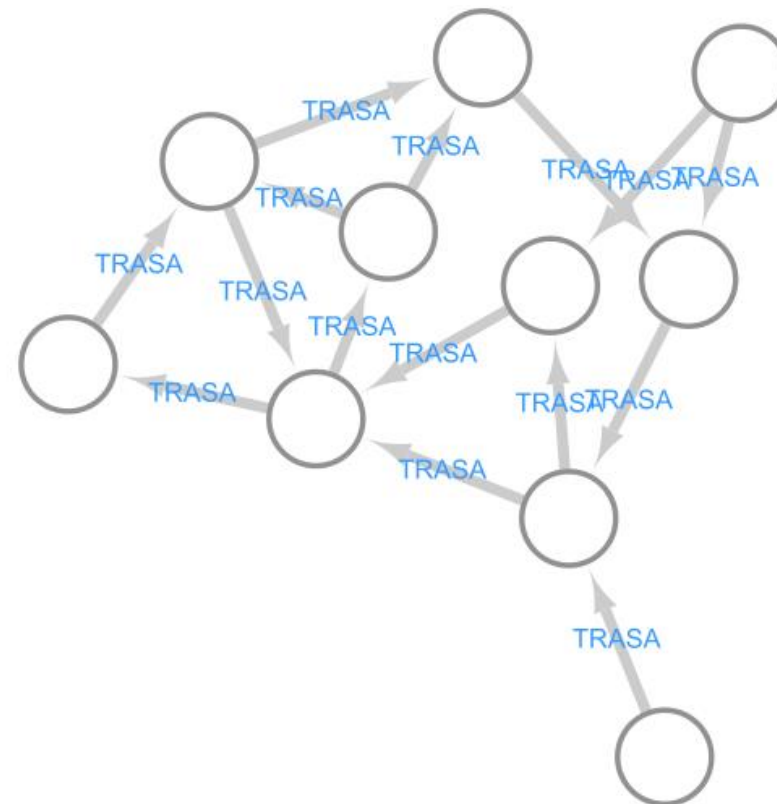
match (n)-[r]->(m) return n,r,m LIMIT 25

Cancel

Execute Query

Import Cypher Query

Po imporcie



All Apps

Categories

collections
data visualization
network generation
network analysis
graph analysis
online data import
automation
clustering
integrated analysis
utility
systems biology
enrichment analysis
visualization
data integration
core app
layout
annotation
pathway database
ontology analysis
import

more »

Graph analysis

🔍 Search for posts about graph analysis →

💬 Ask a question about graph analysis →



BNMatch2

3.0h

BNMatch provides a reliable and optimized mapping between



CentiScaPe

3.0h

Find the most important nodes in a network, calculating centrality



CONAN

3.0h

Analysis of atom interaction networks for structure ensembles



CoordinatesLayout

3.0h

Load a graph with a layout that locates its nodes according to



CountTriplets

3.0h

Counts graph triplets and plots significance profiles



CycleDetector

3.0h

Detects Cycles in a Cytoscape network



CyLinkPredictor

3.0h

Link Predictor Based on Common Neighbors



CytoCluster

3.0h

a Cytoscape app for analysis and visualization of clusters from



CytoCopteR

3.0h

A Cytoscape plug-in for training logic models



cytoHubba

3.0h

Predicts and explores important nodes and subnetworks in a given



CytoMOBAS

3.0h

Identifies and analyzes disease associated and highly connected



CytoNCA

3.0h

Providing calculation, evaluation and visualization analysis for

https://apps.cytoscape.org/apps/with_tag/graphanalysis

Egzamin: Obrona kraju

Obiekt analizy grafowej

Obrona kraju



Cel analizy grafowej

Optymalizacja przydziału
środków obrony przeciwlotnicze

Egzamin: Obrona kraju

Treść zadania

Celem badania jest opracowanie strategii rozmieszczenia wyrzutni rakiet przeciwlotniczych oraz magazynów z zapasowymi rakietami w sposób maksymalizujący efektywność obrony powietrznej określonego obszaru. Obszar ten charakteryzuje się rozwiniętą siecią drogową, kluczową dla transportu strategicznych materiałów. Zadanie obejmuje identyfikację kluczowych węzłów w sieci drogowej do rozmieszczenia wyrzutni oraz optymalizację lokalizacji magazynu zaopatrzenia w rakiety, aby zapewnić minimalny czas dostarczenia rakiet do wyrzutni.

Liczba wyrzutni: 5

Liczba magazynów: 2

Twoim celem jest najefektywniejsza obrona twojego obszaru poprzez :

1. Lokację wyrzutni rakiet przeciwlotniczych w kluczowych węzłach sieci drogowej.
2. Lokalizację magazynów z zapasowymi rakietami, aby najszybciej dostarczyć rakiety do wyrzutni.

Egzamin: Obrona kraju

Budowa grafu

Przekopiuj do katalogu /import pliki „wezelN.csv” i „wezelR.csv”

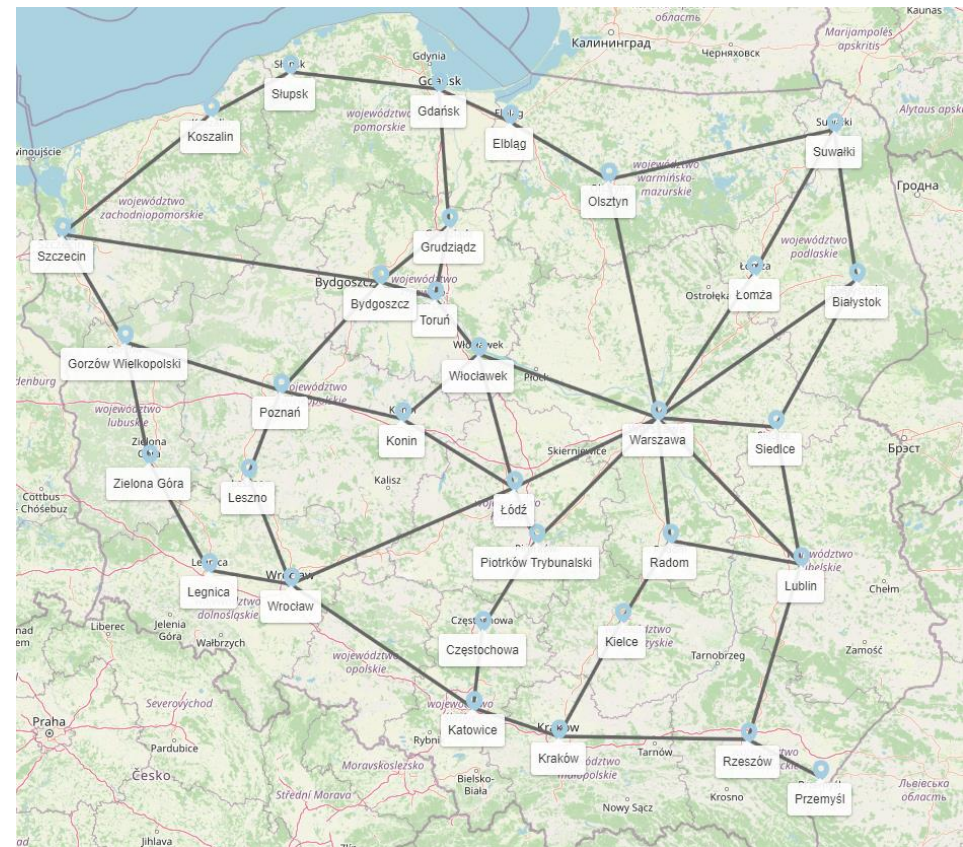
```
MATCH (n) DETACH DELETE n;
```

```
LOAD CSV WITH HEADERS FROM 'file:///wezelN.csv' AS row
```

```
CREATE (n:Wezel {  
  id: toInteger(row.id),  
  nazwa: row.nazwa,  
  latitude: toFloat(row.latitude),  
  longitude: toFloat(row.longitude)  
});
```

```
LOAD CSV WITH HEADERS FROM 'file:///wezelR.csv' AS row
```

```
MATCH (start:Wezel {id: toInteger(row.od)})  
MATCH (end:Wezel {id: toInteger(row.do)})  
CREATE (start)-[:TRASA {  
  km: toInteger(row.km),  
  nazwa: row.road  
}]->(end);
```



Egzamin: Obrona kraju

Proponowane algorytmy

Lokację wyrzutni rakiet przeciwlotniczych w kluczowych węzłach sieci drogowej na twoim obszarze.

???

Lokalizację magazynu z zapasowymi rakietami, aby najszybciej dostarczyć rakiety do wyrzutni.

???

Egzamin: Obrona kraju

Implementacja algorytmów grafowych

???

Wyniki algorytmów grafowych

???

Wykorzystane algorytmy

Algorytm	Grupa	Problem
PageRank	Centralności	Najbardziej wpływowe osoby
Centralność Pośrednictwa (Betweenness Centrality)	Centralności	Osoby, które łączą różne grupy w sieci
Bliskość Centralna (Closeness Centrality)	Centralności	Dobrych plotkarze
Louvain Detekcja Wspólnot (Community Detection)	Wspólnoty	Podgrupy przyjaźni
Dijkstry Shortest Path	Wyszukiwania ścieżek	Najkrótsze trasy między magazynami a punktami dostawy na podstawie odległości
Bliskość Centralna (Closeness Centrality)	Centralności	Najlepiej usytuowane magazyny
Centralność Pośrednictwa (Betweenness Centrality)	Centralności	Którego Centrum Dystrybucji awaria będzie najbardziej dotkliwa?
Algorytm Minimalnego Drzewa Rozpinającego (MST)	Wyszukiwania ścieżek	Najkrótsza droga (minimalna suma odległości d), która połączy wszystkie budynki
Centralność Bliskości (Closeness Centrality)	Centralności	Lokalizacja punktów dostępowych Wi-Fi, tak aby pokryć jak największą liczbę budynków
Bliskość Centralna (Closeness Centrality)	Centralności	Który z budynków w kompleksie najlepiej nadaje się na sztab?
Node Similarity	Podobieństwa	Podobieństwo incydentów
Adamic Adar	Predykcji relacji	Przewidywanie incydentów
???		Lokację wyrzutni rakiet przeciwlotniczych w kluczowych węzłach
???		Lokalizację magazynu z zapasowymi rakietami