

Temat 5 .

dr inż. Michał Malinowski

<https://www.drmalinowski.edu.pl>

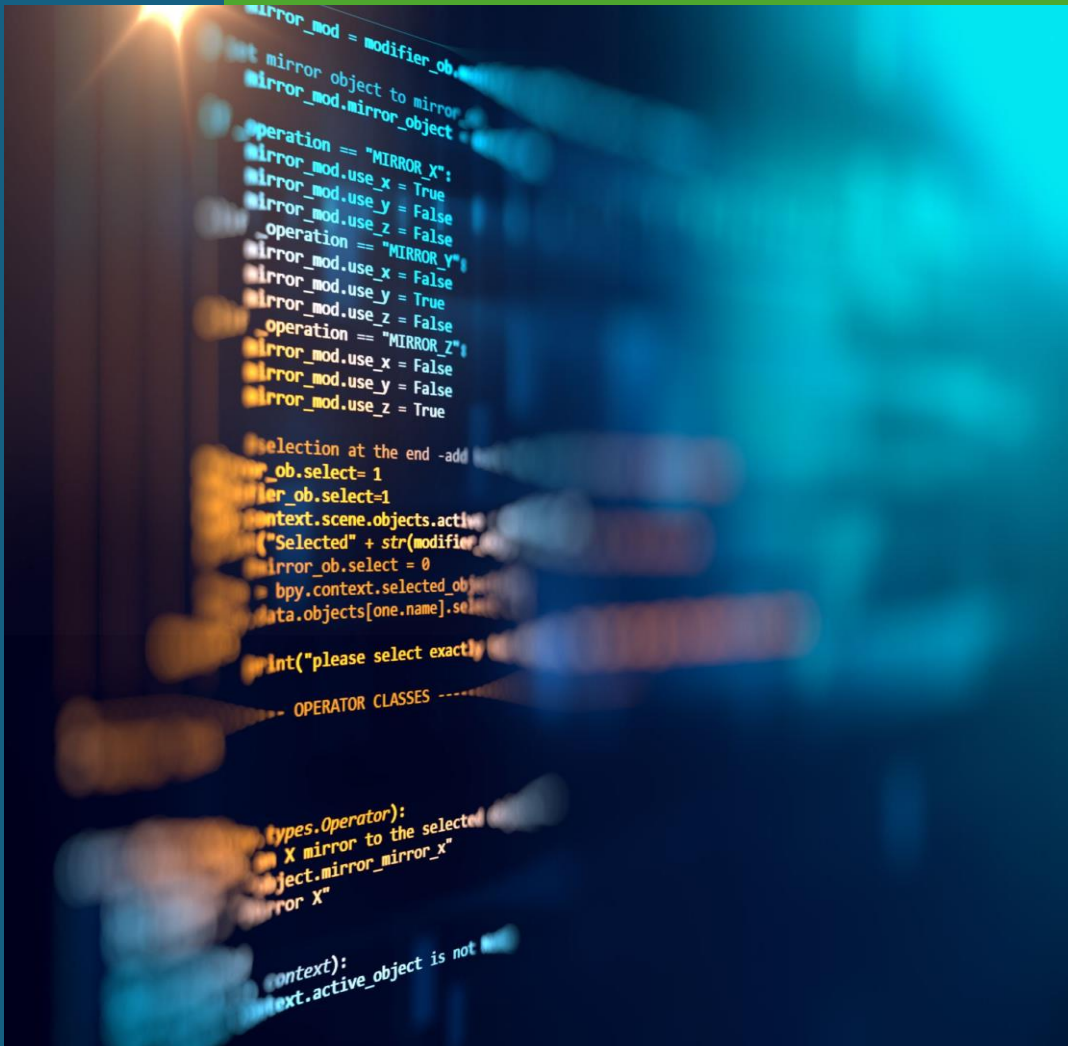
Techniki importowania danych CSV do/z Neo4j

Bazy Grafowe – część I
projektowanie Baz
Grafowych w systemie
zarządzania Neo4j



Zagadnienia (2h)

- Przygotowanie do importowania danych
- Korzystanie z importera danych Neo4j
- Przetwarzanie zaimportowanych danych
- Korzystanie z Cypher do importowania danych
- Eksport danych z Neo4j



Przygotowanie do importowania danych

Typowe źródła danych

- relacyjna baza danych (MySQL, MSSQL, Oracel, DB2)
- Web API
- publiczne katalogi danych
- narzędzia BI (Business Intelligence)
- Excel
- Plik tekstowy: CSV, JSON, XML
- Eksploracja danych (Data mining)

Przygotowanie do importowania danych

Zasady

Należy posiadać pliki CSV (XML, JSON), który reprezentują węzły i relacje grafu

W przypadku pliku CSV należy określić:

- znak separatora pomiędzy danymi (znak przecinka - domyślny)
- pierwszy wiersz zawierający informacje nagłówkowe
- w kolejnych wierszach podawane są dane odpowiadające nagłówkom i oddzielone separatorem

Wskazaniem jest, aby pliki danych nazywać zgodnie z etykietami węzłów z jakimi dane będą związane

- Węzły (:osoba) plik „osoba.csv”
- Węzły (:firma) plik „firma.csv”

Pliki przechowujące informacje o relacjach mogą się nazywać tak jak etykiety węzłów, które będą łączyć

- Relacje [:osoba -> :firma] plik „osoba_firma.csv”

Wiersze powinny zaczynać się od unikalnego klucza (np.: id)

Przygotowanie do importowania danych

Przykłady

osoba.csv

```
id,imie,nazwisko,dzien_urodzenia,student
1,Adam,Nowak,1990-04-03,true
2,Marek,Kowalski,1987-06-12,false
3,Krystyna,Zielinska,2001-11-11,true
4,Alicja,Habior,1996-03-24,false
```

firma.csv

```
id,nazwa,miestowosc
1,GoodHome,Warszawa
2,Precless,Łódź
3,Rokitka,Warszawa
```

osoba_firma.csv

```
idosoba,idfirma,stanowisko
1,1,Administrator
2,1,Ksiegowy
3,2,Specjalista
4,3,Programista
```

Korzystanie z importera danych Neo4j

The screenshot shows the Neo4j Importer interface. The top navigation bar has tabs for 'Explore', 'Query', and 'Import' (highlighted with a blue circle 1). The left sidebar shows a 'Files' list with three CSV files: 'firma.csv', 'osoba.csv', and 'osoba_firma.csv' (highlighted with a blue circle 2). The 'osoba_firma.csv' file is expanded, showing columns: 'idosoba', 'idfirma', and 'stanowisko'. The main workspace (highlighted with a blue circle 3) displays a graph visualization with two nodes, 'Firma' and 'Osoba', connected by a relationship labeled 'pracuje_w' (highlighted with a blue circle 4). The right sidebar contains configuration options: 'Relationship type' (highlighted with a blue circle 5) set to 'pracuje_w', 'File' (highlighted with a blue circle 6) set to 'osoba_firma.csv', 'Node ID mapping' table, and 'Properties' table.

Files

- firma.csv**
 - id: 1
 - nazwa: GoodHome
 - miestowosc: Warszawa
- osoba.csv**
 - id: 1
 - imie: Adam
 - nazwisko: Nowak
 - dzien_urodze...: 1990-04-03
 - student: true
- osoba_firma.csv**
 - idosoba: 1
 - idfirma: 1
 - stanowisko: *Administrat...

Relationship type

Name:

File

Name:

☐ Filter file

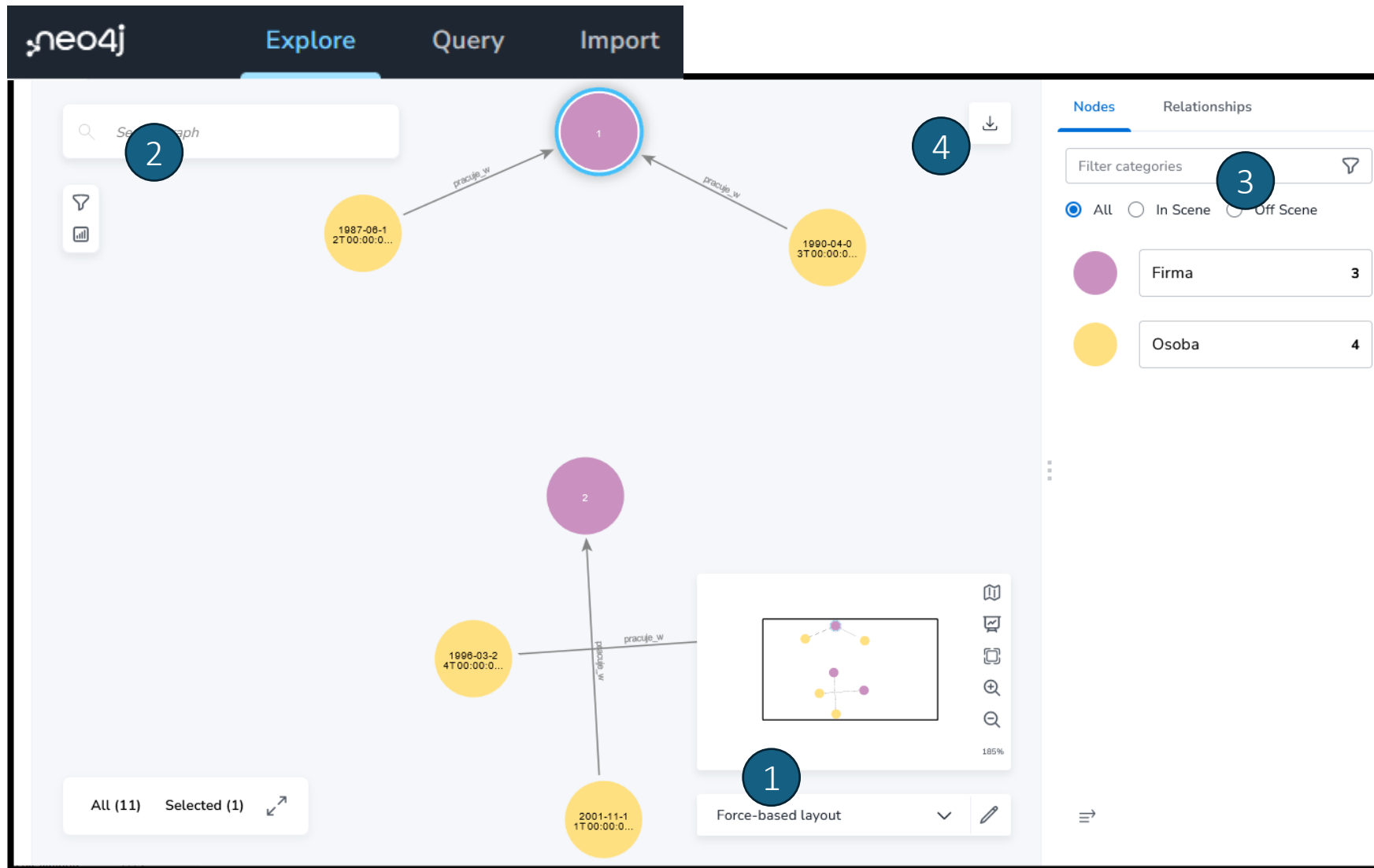
Node ID mapping

	Node	ID	ID column
From	Osoba	id	idosoba
To	Firma	id	idfirma

Properties

	Name	Type	Column
☐	idosoba	inte...	idosoba
☐	idfirma	inte...	idfirma

Korzystanie z importera danych Neo4j



Korzystanie z importera danych Neo4j

The screenshot displays the Neo4j web interface with the following components:

- Navigation Bar:** Includes the Neo4j logo and tabs for "Explore", "Query", and "Import".
- Database Information Panel (Left):**
 - Nodes (7):** Shows two categories: "Firma" (purple) and "Osoba" (blue). A red circle with the number "1" highlights the "Osoba" category.
 - Relationships (4):** Shows one category: "pracuje_w" (grey). A red circle with the number "2" highlights this category.
 - Property keys:** A list of property keys including "data", "dzien_urodzenia", "id", "idfirma", "idosoba", "imie", "miestowosc", "name", "nazwa", "nazwisko", "nodes", "relationships", "stanowisko", "student", and "style".
- Query Editor (Top Right):** Contains the Cypher query: `$ MATCH p=()-[:pracuje_w]→() RETURN p LIMIT 25;`. A red circle with the number "3" highlights the query text.
- Query Results (Bottom Right):** Displays a graph visualization of the query results. A red circle with the number "4" highlights the "Graph" tab. The graph shows four paths of relationships between nodes:
 - Top: "Krakow" (purple) ← "pracuje_w" → "Krystyn..." (blue)
 - Middle Left: "Alicja" (blue) → "wspina" → "Warsza..." (purple)
 - Middle Right: "Marek" (blue) → "pracuje_w" → "Warsza..." (purple)
 - Bottom: "Adam" (blue) → "pracuje_w" → "Warsza..." (purple)
- Status Bar (Bottom Right):** Indicates "Started streaming 4 records after 71ms and completed after 75ms."

Korzystanie z Cypher do importowania danych

Adres URL pliku źródłowego CSV jest określany za pomocą FROM, ponadto wymagane jest określenie właściwości dla danych CSV po klauzuli AS

Dostęp do pliku CSV może być online „https://”, „http://”, „ftp://” lub offline „file:///”

Istotna jest konwersja typów dla klucza i innych kluczowych właściwości

Po imporcie zalecane jest jawne nadanie unikalnego klucza na jedną z właściwości

```
LOAD CSV WITH HEADERS FROM 'https://serwer.pl/xx/osoba.csv' AS row  
CREATE (:Pracownik {id: toInteger(row.id), imie: row.imie,  
dzien_urodzenia: row.dzien_urodzenia})
```

```
LOAD CSV WITH HEADERS FROM 'file:///path/to/your/file.csv' AS ....
```

```
toInteger(row.id)
```

```
CREATE CONSTRAINT FOR (n:Pracownik) REQUIRE n.id IS UNIQUE
```

Korzystanie z Cypher do importowania danych

Konwersja typów właściwości

Funkcja	Opis
toInteger(expression)	Konwertuje wyrażenie na liczbę całkowitą
toFloat(expression)	Konwertuje wyrażenie na liczbę zmiennoprzecinkową
toString(expression)	Konwertuje wyrażenie na łańcuch znaków
toBoolean(expression)	Konwertuje wyrażenie na wartość logiczną
date(expression)	Konwertuje wyrażenie na datę
time(expression)	Konwertuje wyrażenie na czas
datetime(expression)	Konwertuje wyrażenie na datę i czas
localtime(expression)	Konwertuje wyrażenie na czas lokalny
localdatetime(expression)	Konwertuje wyrażenie na lokalną datę i czas

Eksport danych z Neo4j

Możliwości

- Neo4j ODBC Driver: Neo4j ODBC Driver to sterownik ODBC, który umożliwia eksportowanie danych z Neo4j do dowolnego programu zgodnego z ODBC.
- Neo4j JDBC Driver: Neo4j JDBC Driver to sterownik JDBC, który umożliwia eksportowanie danych z Neo4j do dowolnego programu zgodnego z JDBC.
- Neo4j cypher-shell: Neo4j cypher-shell to narzędzie wiersza poleceń, które umożliwia eksportowanie danych z Neo4j do formatu JSON lub CSV.
- Neo4j APOC: APOC to biblioteka Cypher, która umożliwia eksportowanie danych z Neo4j do różnych formatów, w tym JSON, CSV, XML i HTML.
- **Wtyczki w językach programowania**: Java, Python, C#, PHP, Ruby, JavaScript, itd...
- Neo4j Workspace: narzędzie web do zarządzania bazą danych.
- Neo4j Desktop: klient do zarządzania bazą danych.

Eksport danych z Neo4j

Zapytanie CYPHER

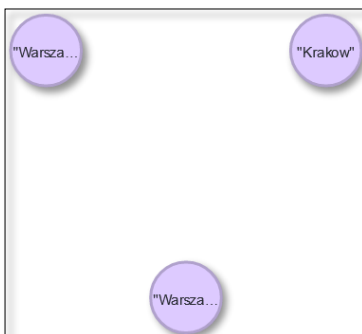
MATCH (n:Firma) RETURN n LIMIT 25;

The screenshot shows the Neo4j web interface. At the top, a text input field contains the Cypher query: `neo4j$ MATCH (n:Firma) RETURN n LIMIT 25;`. Below the query, there are three tabs: 'Graph', 'Table', and 'RAW'. The 'Graph' tab is selected, displaying a graph visualization with three purple circular nodes. Two nodes are labeled '"Warsza..."' and one is labeled '"Krakow"'. There are also three blue circular callouts with numbers 1, 2, and 3. Callout 1 points to the query input field. Callout 2 points to the 'Graph' tab. Callout 3 points to the graph visualization area.

Format CSV, JSON

```
n
(:Firma {miejscowosc: Warszawa, id: 1, nazwa: GoodHome})"
(:Firma {miejscowosc: Krakow, id: 2, nazwa: Precless})"
(:Firma {miejscowosc: Warszawa, id: 3, nazwa: Rokitka})"
```

Format PNG

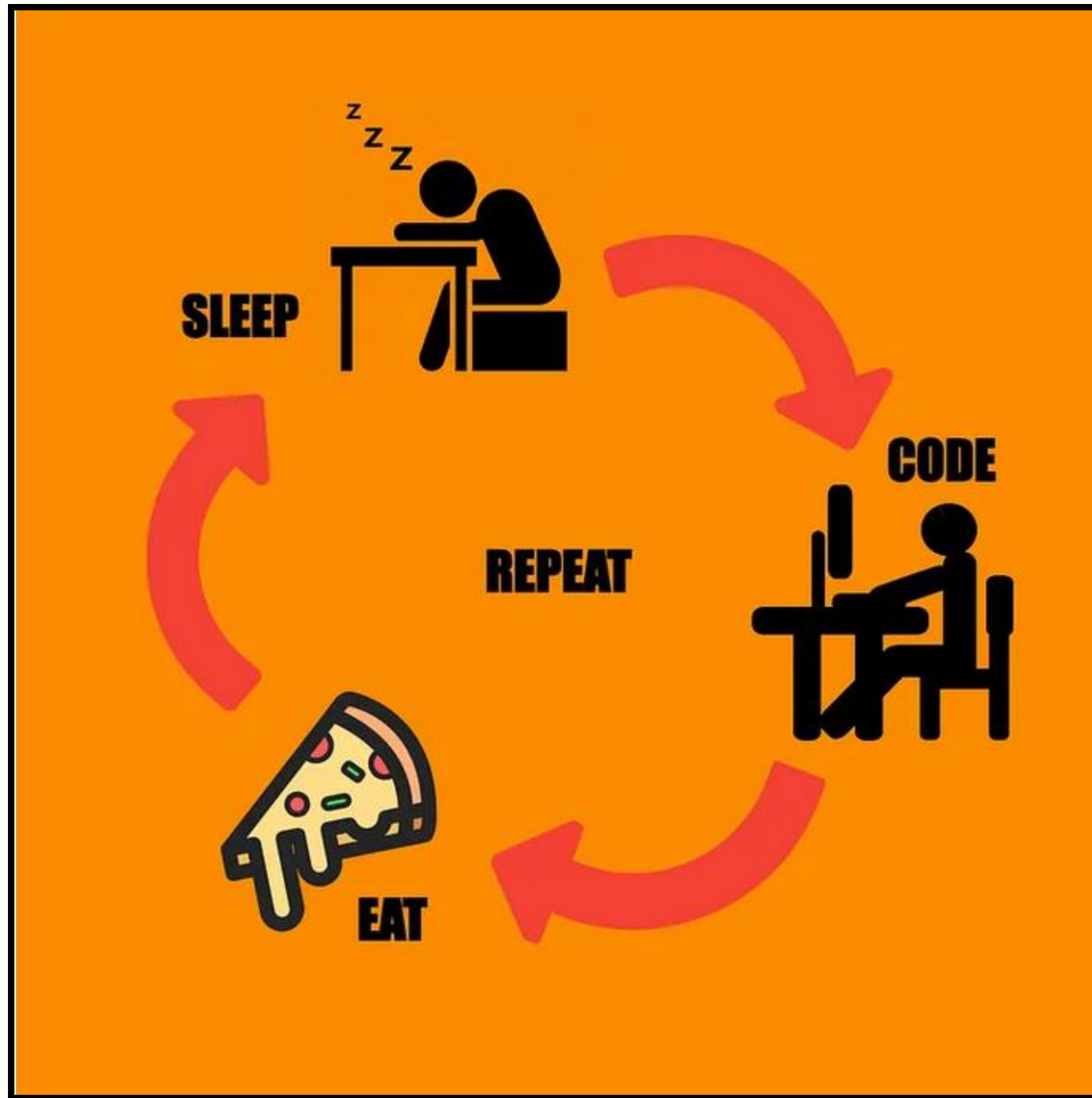


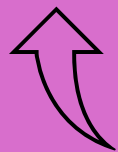
The screenshot shows the Neo4j web interface with the same Cypher query: `neo4j$ MATCH (n:Firma) RETURN n LIMIT 25;`. The 'Table' tab is selected, displaying a table with three rows of data. The first column is labeled 'n'. The rows contain JSON-like objects representing the nodes. There are two blue circular callouts with numbers 4 and 5. Callout 4 points to the table header. Callout 5 points to the table content area.

n
(:Firma {miejscowosc: "Warszawa", id: 1, nazwa: "GoodHome"})
(:Firma {miejscowosc: "Krakow", id: 2, nazwa: "Precless"})
(:Firma {miejscowosc: "Warszawa", id: 3, nazwa: "Rokitka"})

Showing 1-3 of 3 results

Started streaming 3 records after 12ms and completed after 14ms.





Temat 6 .

dr inż. Michał Malinowski

<https://www.drmalinowski.edu.pl>

Złożone zapytania w języku Cypher

Bazy Grafowe – część I
projektowanie Baz
Grafowych w systemie
zarządzania Neo4j



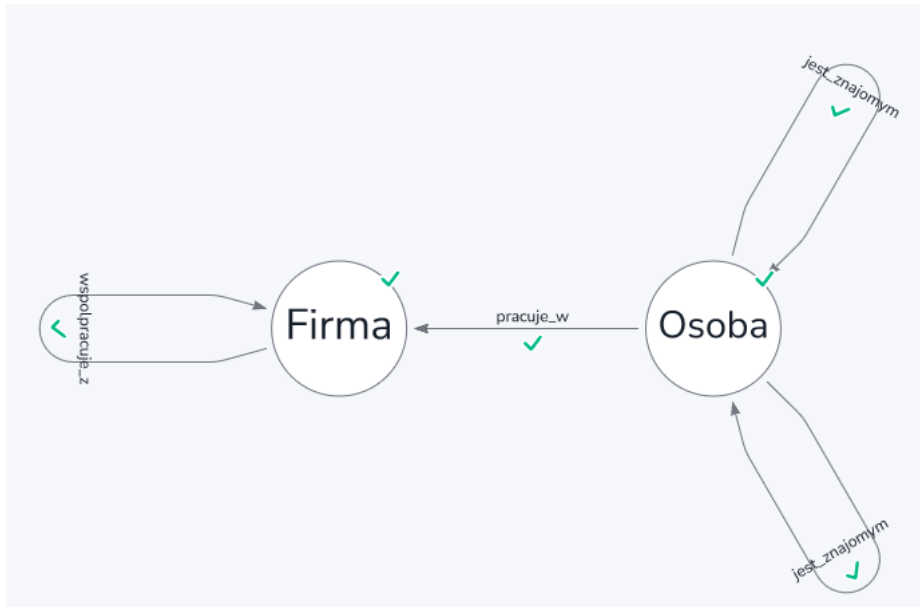
Zagadnienia (3h)

- Przegląd elementów modelu i danych grafu
- Filtrowanie zapytań
- Kontrolowanie zwracanych wyników
- Zmienne
- Podzapytania
- Parametry

Przegląd elementów modelu i danych grafu

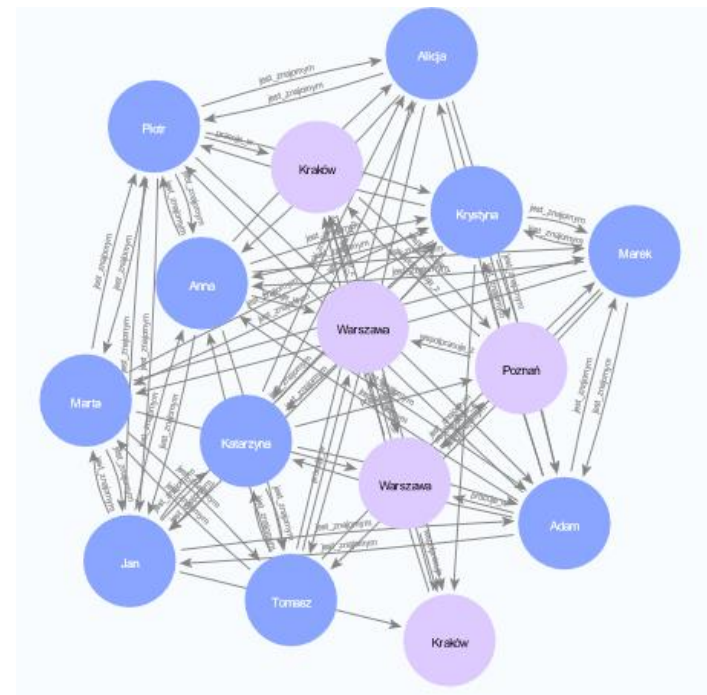
Model grafu

Definiuje schemat przechowywania danych w grafowej bazie danych



Dane grafu

Rzeczywiste informacje, które są przechowywane i reprezentowane w bazie na podstawie modelu grafu



Przegląd elementów modelu i danych grafu

Model grafu

`CALL db.schema.visualization()`

Wykaz właściwości węzłów

`CALL db.schema.nodeTypeProperties()`

Wykaz właściwości relacji

`CALL db.schema.relTypeProperties()`

Wykaz kluczy

`SHOW CONSTRAINTS`

Wykaz wszystkich węzłów

`MATCH (n) RETURN n`

Wykaz wszystkich relacji

`MATCH ()-[r]->() RETURN r`

Cały graf danych

`MATCH p=()-[]->() RETURN p`

Filtrowanie zapytań



Czym zajmowały się te osoby ?



Filtrowanie zapytań

Wykorzystanie klauzuli WHERE

Imiona osób zaczynające się na „M”

```
MATCH (p:Osoba)
WHERE p.imie STARTS WITH "M" RETURN p.imie
```

Imiona osób kończące się na „a”

```
MATCH (p:Osoba)
WHERE p.imie ENDS WITH "a" RETURN p.imie
```

Imiona osób zawierające „ry”

```
MATCH (p:Osoba)
WHERE p.imie CONTAINS "ry" RETURN p.imie
```

toLowerCase()

Osoby zaczynające się na „m”

```
MATCH (p:Osoba)
WHERE toLower(p.imie) STARTS WITH "m" RETURN p.imie
```

toUpperCase()

Osoby kończące się na „A”

```
MATCH (p:Osoba)
WHERE toUpper(p.imie) ENDS WITH "A" RETURN p.imie
```

Filtrowanie zapytań

Wykorzystanie klauzuli WHERE

Od kiedy Marta zna Piotra ?

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta' AND m.imie = 'Piotr' RETURN r.od_kiedy
```

Kogo zna Marta oprócz Piotra?

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta' AND m.imie <> 'Piotr' RETURN m.imie
```

Czy Marta zna Adama?

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta' AND m.imie = 'Adam' RETURN m.imie
```

Kogo zna Marta od 2018 roku?

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta' AND r.od_kiedy.year>=2018
RETURN m.imie, r.od_kiedy
```

Filtrowanie zapytań

Łączenie klauzuli MATCH

Kogo zna Marta od 2018 roku
i gdzie ten ktoś pracuje i na jakim
stanowisku?

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta' AND r.od_kiedy.year>=2018
MATCH (m)-[w:pracuje_w]->(f:Firma)
RETURN m.imie, r.od_kiedy, f.nazwa, w.stanowisko
```

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba),
(m)-[w:pracuje_w]->(f:Firma)
WHERE p.imie = 'Marta' AND r.od_kiedy.year>=2018
RETURN m.imie, r.od_kiedy, f.nazwa, w.stanowisko
```

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)-[w:pracuje_w]->(f:Firma)
WHERE p.imie = 'Marta' AND r.od_kiedy.year>=2018
RETURN m.imie, r.od_kiedy, f.nazwa, w.stanowisko
```

Filtrowanie zapytań

OPTIONAL MATCH

pozwała dopasować wzorzec w grafie, ale nie wymaga, aby ten wzorzec istniał

PROFILE MATCH

Plan wykonania zapytania

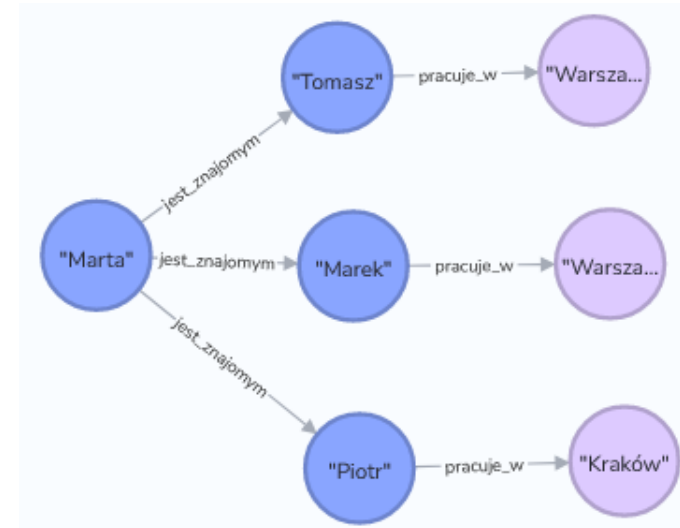
```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta'
OPTIONAL MATCH (m)-[w:pracuje_w]->(f:Firma)
WHERE f.nazwa='PixelVibe'
RETURN m.imie, r.od_kiedy, f.nazwa, w.stanowisko
```

```
PROFILE MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)-[w:pracuje_w]->(f:Firma)
WHERE p.imie = 'Marta' AND r.od_kiedy.year>=2018
RETURN m.imie, r.od_kiedy, f.nazwa, w.stanowisko
```

Kontrolowanie zwracanych wyników

W postaci grafu

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)-[w:pracuje_w]->(f:Firma)
WHERE p.imie = 'Marta' AND r.od_kiedy.year>=2018
RETURN p, r, m, w, f
```



W postaci zdania

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)-[w:pracuje_w]->(f:Firma)
WHERE p.imie = 'Marta' AND r.od_kiedy.year>=2018
RETURN p.imie,"jest znajomym",m.imie,"od",r.od_kiedy,"który pracuje
w",f.nazwa,"na stanowisku",w.stanowisko
```

Kontrolowanie zwracanych wyników

Sortowanie oraz liczba wyników

Kogo ostatnio poznała Marta?

ORDER BY

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta'
RETURN m.imie, r.od_kiedy
ORDER BY r.od_kiedy DESC
```

Kogo najdłużej zna Marta?

LIMIT

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta'
RETURN m.imie, r.od_kiedy
ORDER BY r.od_kiedy ASC
LIMIT 1
```

Paginacja / stronicowanie

SKIP

„nie pokazuj węzłów”

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta'
RETURN m.imie, r.od_kiedy
ORDER BY r.od_kiedy ASC
SKIP 3 LIMIT 2
```

Kontrolowanie zwracanych wyników

Osoby których imiona zawierają „Mar”?

Jako węzły

```
MATCH (p:Osoba) WHERE p.imie CONTAINS "Mar"  
RETURN p
```

Jako pełen JSON

```
MATCH (p:Osoba) WHERE p.imie CONTAINS "Mar"  
RETURN p { .* }
```

Jako wybrane właściwości JSON

```
MATCH (p:Osoba) WHERE p.imie CONTAINS "Mar"  
RETURN p { .imie, .nazwisko }
```

Jako wybrane właściwości JSON
+ stała

```
MATCH (p:Osoba) WHERE p.imie CONTAINS "Mar"  
RETURN p { .imie, .nazwisko, kraj:'Polska' }
```

Jako kolekcja/tablica

```
MATCH (p:Osoba)  
RETURN collect(p.imie)
```

Kontrolowanie zwracanych wyników

Osoby o nazwisku „Kowalski”?
ile mają lat?

```
MATCH (p:Osoba) WHERE p.nazwisko="Kowalski"  
RETURN p.imie, date().year - p.dzien_urodzenia.year + " lat" AS wiek
```

Przedziały wiekowe
CASE..WHEN..ELSE..END

```
MATCH (p:Osoba) RETURN p.imie, p.dzien_urodzenia.year,  
CASE  
  WHEN p.dzien_urodzenia.year < 1990 THEN "stary"  
  WHEN 1990 <= p.dzien_urodzenia.year < 2000 THEN "średni"  
  ELSE "młody"  
END  
AS wiek ORDER BY p.dzien_urodzenia.year DESC
```

Usuwanie duplikatów

DISTINCT

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)  
RETURN DISTINCT r.od_kiedy.year
```

Zmienne

Definiowanie zmiennych

Osoby i firmy z Krakowa

```
MATCH (p:Osoba)
MATCH (f:Firma)
WHERE p.miejscowosc = "Kraków" AND f.miejscowosc = "Kraków"
RETURN p, f
```

Definiowanie zmiennej

WITH

```
WITH "Kraków" AS szukana
MATCH (p:Osoba)
MATCH (f:Firma)
WHERE p.miejscowosc = szukana AND f.miejscowosc = szukana
RETURN p, f
```

Usuwa „null” z wyników.

UNWIND

```
MATCH (p:Osoba)
UNWIND p.miejscowosc AS miejscowosci
RETURN miejscowosci
```

Podzapytania

Gdzie pracują mieszkańcy
Warszawy?

Wywołanie podzapytania

CALL{}

```
CALL {  
  MATCH (p:Osoba)  
  WHERE p.miejscowosc="Warszawa"  
  RETURN p  
}
```

```
MATCH (p)-[:pracuje_w]->(f:Firma)  
RETURN f.nazwa, f.miejscowosc
```

Łączenie wyników

UNION

```
MATCH (f:Firma)  
WHERE f.rok_powstania>=2000  
RETURN {grupa:"firma", nazwy:collect(f.nazwa)} AS data  
UNION  
MATCH (p:Osoba)  
WHERE p.dzien_urodzenia.year>=2000  
RETURN {grupa:"osoba", nazwy:collect(p.imie)} AS data
```

Parametry

Utworzenie `:param {miestowosc: "Kraków"}`

Lista parametrów `:param list`

Zmiana wartości `:param {miestowosc : "Warszawa"}`

Wyczyszczenie `:param clear`

Użycie
`MATCH (f:Firma)`
`WHERE f.miestowosc=$miestowosc`
`RETURN f.nazwa, f.miestowosc`

Menu

neo4j

ExploreQueryImport

Instance01 / neo4j

Send feedback

Saved Cypher

Select all

☐ Delete ALL

☒ Delete Parametry

☐ Load

☐ Osoba

☐ Parametry

☐ miejscowosc

☒ Osoba

☐ Firma

neo4j\$

neo4j\$ PROFILE MATCH p =allShortestPaths((start:Osoba

GraphPlanTableRAW

```
graph LR; Marta -- jest_znajomym --> Krystyna; Marta -- jest_znajomym --> Piotr; Marta -- jest_znajomym --> Anna; Marta -- jest_znajomym --> Marek; Marta -- jest_znajomym --> Jan; Marta -- jest_znajomym --> Tomasz; Krystyna -- jest_znajomym --> Piotr; Krystyna -- jest_znajomym --> Anna; Piotr -- jest_znajomym --> Anna; Piotr -- jest_znajomym --> Marek; Piotr -- jest_znajomym --> Jan; Anna -- jest_znajomym --> Marek; Anna -- jest_znajomym --> Tomasz; Jan -- jest_znajomym --> Marek; Jan -- jest_znajomym --> Tomasz; Tomasz -- jest_znajomym --> Marek;
```

Started streaming 20 records after 2ms and completed after 7ms.

The provided pattern is unbounded, consider adding an upper limit to the number of node hops.

Ścieżki

Jaki ma związek Marta z Anną?

... z ograniczeniem tylko do
znajomych

... o długości 2

... wszystkie najkrótsze ścieżki

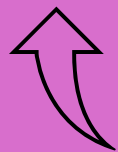
```
MATCH p = (start:Osoba{imie:'Marta'}) -[*]- (end:Osoba{imie:'Anna'})  
RETURN p
```

```
MATCH p = (start:Osoba{imie:'Marta'}) -[:jest_znajomym*]-  
(end:Osoba{imie:'Anna'})  
RETURN p
```

```
MATCH p = (start:Osoba{imie:'Marta'}) -[:jest_znajomym*2]-  
(end:Osoba{imie:'Anna'})  
RETURN p
```

```
PROFILE MATCH p =allShortestPaths((start:Osoba{imie:'Marta'}) -[*]-  
(end:Osoba{imie:'Anna'}))  
RETURN p
```





Temat 7 .

Funkcje agregujące i listy w języku Cypher

dr inż. Michał Malinowski

<https://www.drmalinowski.edu.pl>

Bazy Grafowe – część I
projektowanie Baz
Grafowych w systemie
zarządzania Neo4j



Zagadnienia (2h)

- Funkcje agregujące
- Praca z listami
- Wzorce
- Mapy danych

Funkcje agregujące

Ilu znajomych ma Marta?

count() – liczba węzłów

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
WHERE p.imie = 'Marta'
RETURN p.imie , count(m) AS liczba
```

Kto ma najwięcej znajomych?

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
RETURN p.imie , count(*) AS liczba
ORDER BY liczba DESC LIMIT 1
```

min() - minimum

max() - maksimum

avg() - średnia

sum() – suma

```
MATCH (p:Osoba)
RETURN
min(date().year - p.dzien_urodzenia.year) AS najmłodszy,
max(date().year - p.dzien_urodzenia.year) AS najstarszy,
avg(date().year - p.dzien_urodzenia.year) AS średni,
sum(date().year - p.dzien_urodzenia.year) AS suma
```

COALESCE - zwraca pierwszą napotkaną wartość nie będącą nullem

```
RETURN COALESCE(NULL, 'druga', 'trzecia');
```

Praca z listami

Kim są znajomi poszczególnych osób?

collect() - zbiera wartości dla danego węzła w jednej liście

Operacje na liście (kolekcji):

count() – liczba wierszy

size() – liczba elementów w liście

collect()[] – dostęp do pozycji w liście

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
RETURN p.imie , count(m) AS liczba,
collect(m.imie) AS znajomi
```

```
MATCH (p:Osoba)-[r:jest_znajomym]->(m:Osoba)
RETURN p.imie , count(m) AS liczba,
size(collect(m.imie)) AS rozmiar,
collect(m.imie)[0] AS znajomy, collect(m.imie)[1..] AS pozotali,
collect(m.imie) AS znajomi
```

Wzorce

Pracownicy firm i kontrahenci
[<wzór> | klucz(wartość)]

```
MATCH (f:Firma)
RETURN f.nazwa,
[(p:Osoba)-[:pracuje_w]->(f) | p.imie] AS pracownicy,
[(f)-[:wspolpracuje_z]->(g:Firma) | g.nazwa] AS kontrahenci
```

Czy Marta pracowała w firmie
która w nazwie ma „o”

```
MATCH (p:Osoba {imie: "Marta"})
RETURN
[(p)-->(f:Firma)
WHERE f.nazwa CONTAINS "o" | f.nazwa + " z " + f.miejscowosc]
```

Praca z listami

Listy / kolekcje

Jakich znajomych mają koledzy z pracy Marty?

Kolekcja

```
MATCH (p:Osoba)-[:pracuje_w]->(f:Firma)<-[:pracuje_w]-(d:Osoba)-[:jest_znajomym]->(k:Osoba)
WHERE p.imie="Marta"
RETURN f.nazwa, d.imie, collect(k.imie)
```

Graf

```
MATCH g=((p:Osoba)-[:pracuje_w]->(f:Firma)<-[:pracuje_w]-(d:Osoba)-[:jest_znajomym]->(k:Osoba))
WHERE p.imie="Marta"
RETURN g
```

Węzły w ścieżce

```
RETURN nodes(g)
```

Relacje

```
RETURN relationships(g)
```

Mapy danych

Mapa danych Cypher to lista par klucz/wartość, gdzie każdy element listy ma format "klucz: wartość".

Węzeł lub relacja może mieć właściwość, która jest mapą.

Węzeł, gdy jest zwracany w przeglądarce, jest mapą.

```
RETURN {Sty: 31, Lut: 28, Mar: 31, Kwi: 30, Maj: 31, Cze: 30, Lip: 31, Sie: 31,
Wrz: 30, Paź: 31, Lis: 30, Gru: 31}['Lut'] AS dniWLutym
```

```
RETURN {Sty: 31, Lut: 28, Mar: 31, Kwi: 30, Maj: 31, Cze: 30, Lip: 31, Sie: 31,
Wrz: 30, Paź: 31, Lis: 30, Gru: 31}.Lut AS dniWLutym
```

```
RETURN keys({Sty: 31, Lut: 28, Mar: 31, Kwi: 30, Maj: 31, Cze: 30, Lip: 31,
Sie: 31, Wrz: 30, Paź: 31, Lis: 30, Gru: 31}) AS miesiące
```

```
:param {kalendarz: {Sty: 31, Lut: 28, Mar: 31, Kwi: 30, Maj: 31, Cze: 30, Lip:
31, Sie: 31, Wrz: 30, Paź: 31, Lis: 30, Gru: 31}}
```

```
RETURN $kalendarz.Lut
```

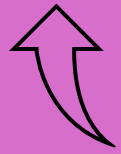
```
MATCH (m:Osoba)
RETURN m LIMIT 1
```

How users see the IT guy



How the IT guy see the users





Temat 8 .

dr inż. Michał Malinowski

<https://www.drmalinowski.edu.pl>

Indeksy i ograniczenia języka Cypher

Bazy Grafowe – część I
projektowanie Baz
Grafowych w systemie
zarządzania Neo4j



Zagadnienia (1h)

- Wykorzystanie indeksów i ograniczeń
- Tworzenie i używanie indeksów
- Tworzenie i używanie ograniczeń

Wykorzystanie indeksów i ograniczeń

Możliwości

Indeksy przyspieszają wykonywanie zapytań i pobieranie danych oraz ich walidację
ale spowalniają ich zapisywanie i zajmują dodatkową pamięć

Istnieją trzy rodzaje ograniczeń, które można zdefiniować:

- Unikalność dla wartości właściwości pojedynczego węzła.
- Istnienie dla właściwości węzła lub relacji.
- Istnienie oraz unikalność dla zestawu wartości właściwości węzła (zwanego kluczem węzła).

Najlepszą praktyką jest tworzenie ograniczeń przed załadowaniem danych !!!

Wykorzystanie indeksów i ograniczeń

Typy indeksów

Nazwa	Opis	Przykłady
RANGE	Używane do przechowywania i odzyskiwania wartości w określonym zakresie	Wyszukiwanie osób w określonym wieku; Produkty w przedziale cenowym
LOOKUP	Używane do wyszukiwania węzłów na podstawie etykiety lub relacji na podstawie typu	Wyszukiwanie wszystkich węzłów o etykiecie "Osoba"; Relacje typu "PRZYJACIELE"
TEXT	Specjalizowane do obsługi zapytań na łańcuchach znaków	Wyszukiwanie nazwisk na podstawie liter; Produkty o nazwach zawierających określone słowo
POINT	Używane do przechowywania i odzyskiwania danych przestrzennych, takich jak współrzędne	Wyszukiwanie restauracji w określonym promieniu; Lokalizacje bliskie danemu punktowi
Full-text	Specjalizowane do wsparcia zaawansowanego wyszukiwania pełnotekstowego	Wyszukiwanie artykułów zawierających frazy "Neo4j" i "indeks"

Tworzenie i używanie indeksów

Tworzenie indeksu dla węzła

```
CREATE [typ_indeksu] INDEX <nazwa_indeksu> IF NOT EXISTS  
FOR (x:<etykieta_węzła>)  
ON x.<klucz_właściwości>
```

Przykład

```
CREATE INDEX rok_powstania IF NOT EXISTS  
FOR (x:Firma)  
ON x.rok_powstania
```

Tworzenie indeksu dla relacji

```
CREATE [typ_indeksu] INDEX <nazwa_indeksu> IF NOT EXISTS  
FOR ()-[x:<typ_relacji>]-()  
ON (x.<klucz_właściwości>)
```

Przykład

```
CREATE TEXT INDEX stanowisko IF NOT EXISTS  
FOR ()-[x:pracuje_w]-()  
ON (x.stanowisko)
```

Tworzenie i używanie indeksów

Wymuszenie użycia indeksu
USING INDEX

```
MATCH (f:Firma)
  USING INDEX f:Firma(rok_powstania)
  WHERE f.rok_powstania >= 2000
  RETURN f.nazwa, f. rok_powstania
```

Wykaz indeksów w grafie

```
SHOW INDEXES
```

Usuwanie indeksu

```
DROP INDEX <index_name>
```

Przykład

```
DROP INDEX rok_powstania
```

Tworzenie i używanie ograniczeń

Tworzenie ograniczenia dla węzła

```
CREATE CONSTRAINT <nazwa_ograniczenia> IF NOT EXISTS  
FOR (x:<etykieta_węzła>)  
REQUIRE x.<klucz_właściwości> IS UNIQUE
```

Zły przykład

```
CREATE CONSTRAINT miejscowosc IF NOT EXISTS  
FOR (x:Osoba)  
REQUIRE x.miejscowosc IS UNIQUE
```

Dobry przykład

```
CREATE CONSTRAINT pesel IF NOT EXISTS  
FOR (x:Osoba)  
REQUIRE x.pesel IS UNIQUE
```

Tworzenie ograniczenia złożonego
dla węzła
„w każdej miejscowości jest
prowadzona oddzielna ewidencja
zezwoleń”

```
CREATE CONSTRAINT miejscowosc_zezwolenie IF NOT EXISTS  
FOR (x:Firma)  
REQUIRE (x.miejscowosc, x.zezwolenie) IS UNIQUE
```

Tworzenie i używanie ograniczeń

Tworzenie ograniczenia dla relacji

```
CREATE CONSTRAINT <nazwa_ograniczenia> IF NOT EXISTS  
FOR ()-[x:<typ_relacji>]-()  
REQUIRE x.<klucz_właściwości> IS UNIQUE
```

Przykład

„osoba może być tylko raz
zatrudniona w jednej firmie”

```
CREATE CONSTRAINT firma_osoba IF NOT EXISTS  
FOR ()-[x:pracuje_w]-()  
REQUIRE (x.idfirma, x.idosoba) IS UNIQUE
```

Wykaz ograniczeń w grafie

```
SHOW CONSTRAINTS
```

Usuwanie indeksu

```
DROP CONSTRAINT <index_name>
```

Przykład

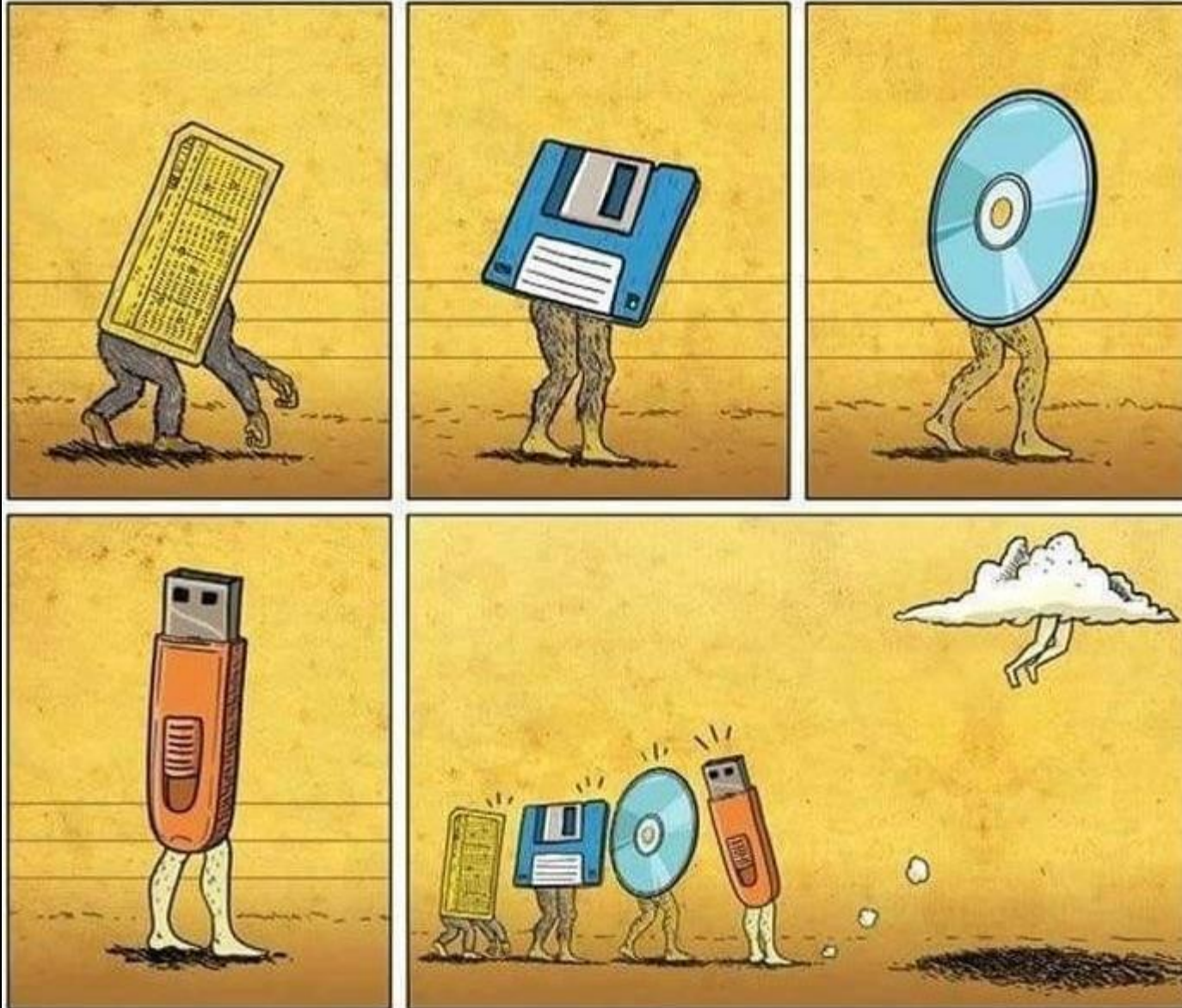
```
DROP CONSTRAINT firma_osoba
```

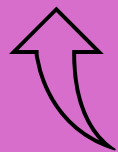
Wykorzystanie indeksów i ograniczeń

Korzyści

- Optymalizacja Zapytań: Indeksy przyspieszają wyszukiwanie węzłów i relacji w bazie danych, co znacznie zwiększa wydajność zapytań. (Widać od około ponad 1000 pozycji)
- Unikalność: Ograniczenia unikalności zapobiegają zduplikowanym danym, zapewniając spójność w całej bazie danych.
- Integralność Danych: Ograniczenia zapewniają, że dane wprowadzane do bazy danych są zgodne z określonymi regułami i normami.
- Skalowalność i Wydajność: Indeksy i ograniczenia pomagają Neo4j działać efektywnie i skutecznie na dużą skalę (>1000).
- Jakość Danych: Pomagają w utrzymaniu jakości danych przez egzekwowanie reguł i unikanie duplikatów.

Evolution of Memory Storage





Temat 9 .

Funkcje czasu w języku Cypher

dr inż. Michał Malinowski

<https://www.drmalinowski.edu.pl>

Bazy Grafowe – część I
projektowanie Baz
Grafowych w systemie
zarządzania Neo4j



Zagadnienia (1h)

- Organizacja funkcji czasu
- Wykorzystanie funkcji czasu
- Działanie funkcji czasu

Organizacja funkcji czas

Struktura Date, Time, DateTime

date() – dzień

time() – godzina (serwera)

datetime() – dzień i godzina

Formatowanie: yyyy-mm-dd

Zdefiniowanie czasu

Dodawanie dni

- **P**: Oznacza "Period" (Okres) i jest to obowiązkowy znak rozpoczynający zapis okresu czasu.
- **3**: Liczba jednostek okresu.
- **D**: Oznacza "Days" (Dni).

RETURN date(), time(), datetime()

RETURN datetime({year: 2023, month: 8, day: 23}) AS datetime

RETURN date("2023-08-23") AS date

RETURN time(), time() + duration({hours: 1}) AS za_minute

RETURN date() + duration("P3D") AS przyszly_dzien

Wykorzystanie funkcji czasu

Wydobywanie:
roku, miesiąca i dnia

```
RETURN date().year AS rok  
RETURN date().month AS miesiac  
RETURN date().day AS dzien
```

Porównanie dat

```
RETURN date("2023-08-23") > date("2022-08-23") AS porownanie
```

Utworzenie węzła z jednostką
czasu

```
MERGE (x:Czas {id: 1})  
SET x.date = date(),  
    x.datetime = datetime(),  
    x.time = time()  
RETURN x
```

Działanie funkcji czasu

Odczyt składowych czasu

```
MATCH (x:Czas {id: 1})  
RETURN x.date.day,  
x.date.year,  
x.datetime.year,  
x.datetime.hour,  
x.datetime.minute
```

Ustawienie dat

```
MATCH (x:Czas {id: 1})  
SET x.date1 = date('2022-01-01'), x.date2 = date('2022-01-15')  
RETURN x
```

Obliczanie różnicy dat

```
MATCH (x:Czas {id: 1})  
RETURN x.date1,x.date2,duration.between(x.date1,x.date2).days
```

Dodanie odcinku czasu

```
MATCH (x:Czas {id: 1})  
RETURN x.date1,x.date1 + duration({months: 6})
```

Składnia CQL (Cypher)

