

Temat 1 .

Podstawy baz grafowych oraz systemu zarządzania bazą danych Neo4j

dr inż. Michał Malinowski

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

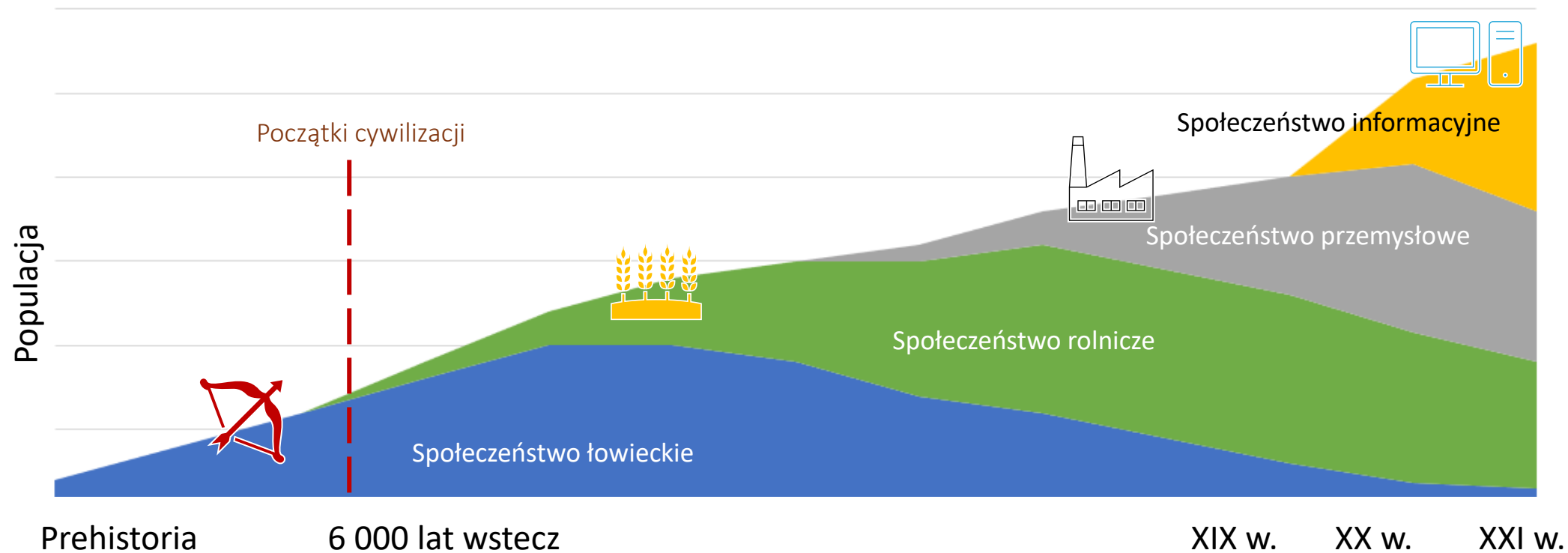
Bazy Grafowe – część I
projektowanie Baz
Grafowych w systemie
zarządzania Neo4j



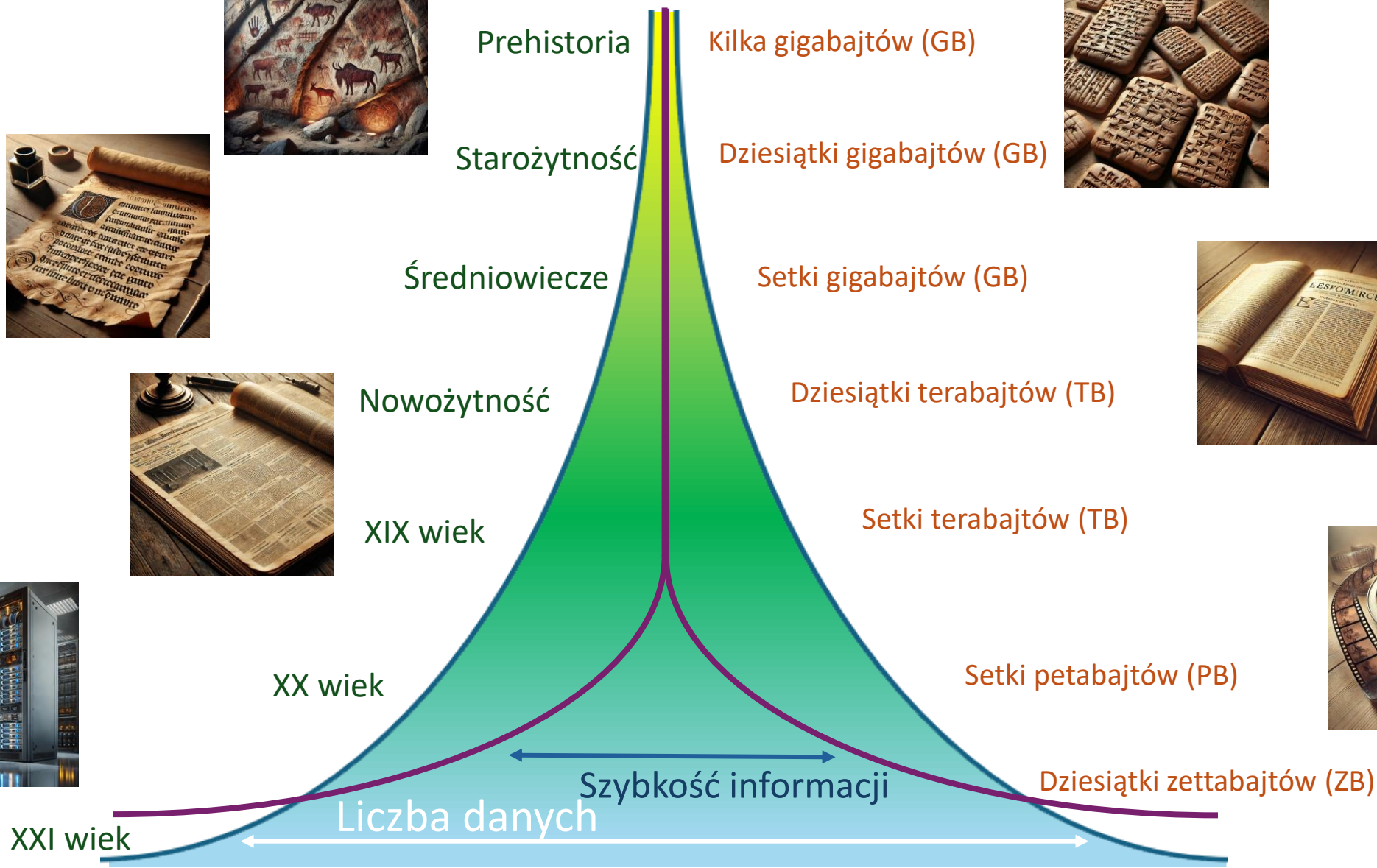
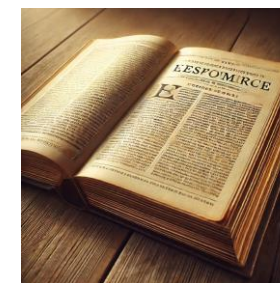
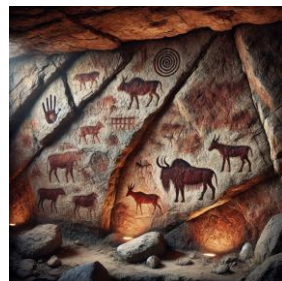
Zagadnienia (2h)

- Wprowadzenie do baz NoSQL
- Podstawy teorii grafów i elementy grafu
- Typowe przypadki użycia grafów
- Wybrane formy grafów
- Architektura systemu zarządzania bazą danych Neo4j

Gdzie teraz jesteśmy... ?



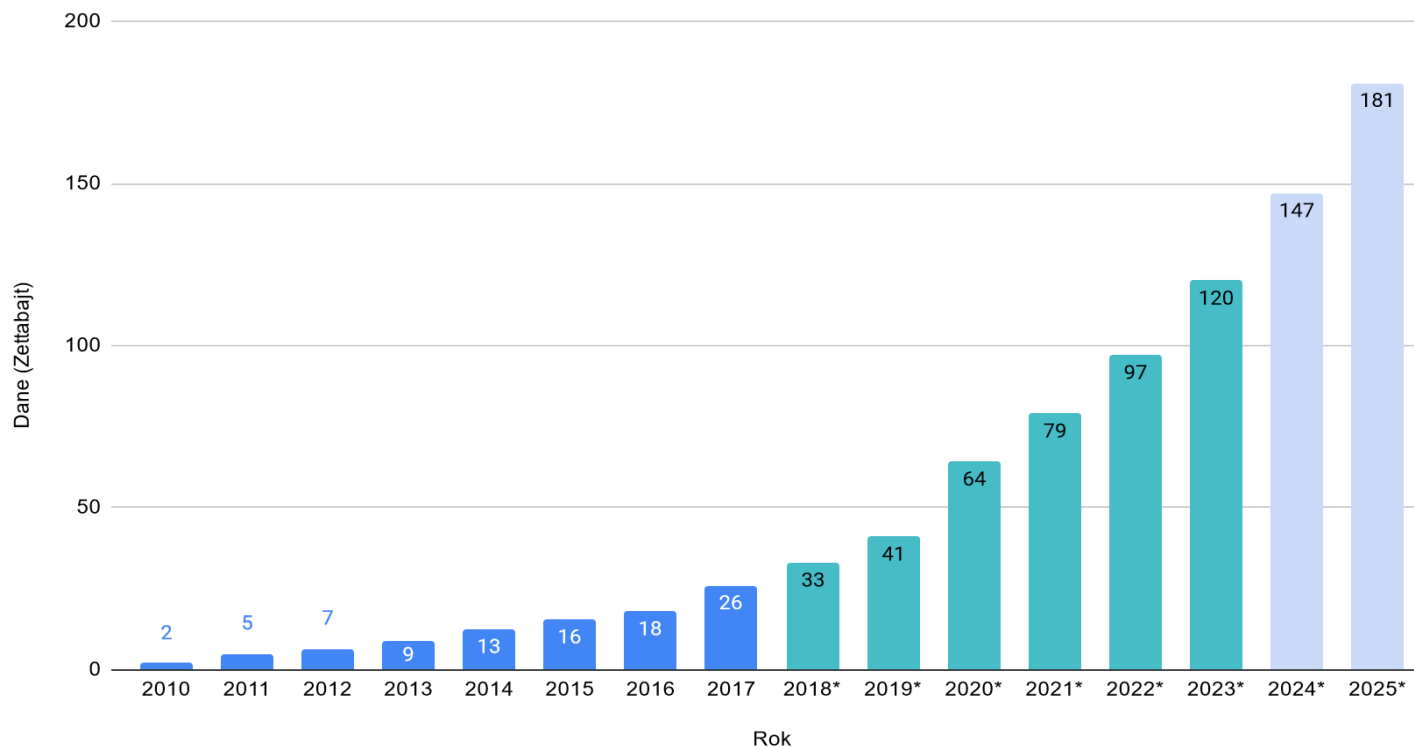
Ilość i nośniki danych na przestrzeni historii



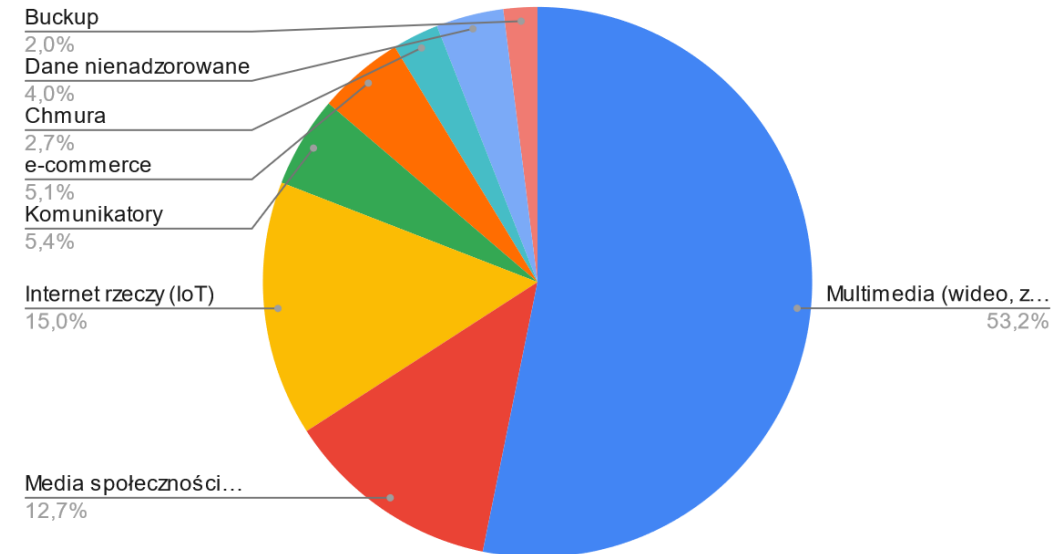
Wielkość danych na świecie



Wielkość danych w latach na Świecie



Typy danych



Na osobę	
2022*	12 TB
2023*	15 TB
2024*	18 TB
2025*	23 TB

Możliwości gromadzenia i propagowania informacji

Smartfon

2024r

124 GB

> 4 000 000 000
mieszkańców ziemi



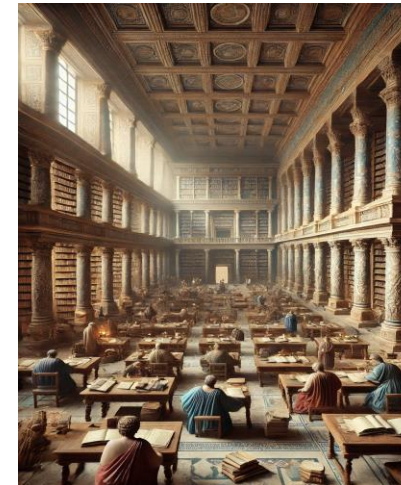
40 000 km
~ 200 ms

Biblioteka Aleksandryjska

II wieku p.n.e.

12,5 GB

Pojemność danych



centrum wiedzy
starożytnego świata

Czas propagacji
informacji



4 000 km
~ 11 dnia

Wprowadzenie do baz NoSQL

NoSQL (Not Only SQL)

kategoria systemów zarządzania bazami danych, która zapewnia mechanizmy do przechowywania i odzyskiwania danych, które są modelowane w sposób inny niż tabele stosowane w relacyjnych bazach danych.

Systemy NoSQL są zaprojektowane w celu obsługi dużych wolumenów danych, wysokiej skalowalności, elastycznej schematy danych i łatwej dystrybucji danych.

Nie zawsze zapewniają pełne wsparcie dla standardów ACID*

***ACID** (akronim, który opisuje podstawowe właściwości transakcji w systemie zarządzania bazą danych)

Atomicity (Atomowość)

Wszystkie operacje w ramach transakcji są wykonane w całości lub żadna z nich. Jeśli jedna operacja się nie powiedzie, cała transakcja jest cofana.

Consistency (Spójność)

Transakcja przekształca bazę danych z jednego poprawnego stanu do innego, zachowując przy tym wszystkie reguły integralności.

Isolation (Izolacja)

Wyniki transakcji są niewidoczne dla innych transakcji, dopóki nie zostaną zatwierdzone, co zapobiega zakłóceniom między współbieżnymi transakcjami.

Durability (Trwałość)

Po zatwierdzeniu transakcji, jej wyniki są trwale zapisane w bazie danych, nawet w przypadku awarii systemu.

Wprowadzenie do baz NoSQL

Cechy baz NoSQL

Skalowalność

Bazy NoSQL łatwo i efektywnie skalują, rozpraszając dane na wiele serwerów.

Wydajność

Zaprojektowane do szybkiego przetwarzania dużej liczby operacji, dzięki nierelacyjnym modelom danych i równoległemu przetwarzaniu zapytań.

Elastyczność schematu

Dopuszczają dynamiczne dodawanie i modyfikowanie typów danych bez konieczności modyfikacji całego schematu.

Odporność na awarie

Zaprojektowane z myślą o redundancji, obsługują replikację danych dla lepszej odporności na awarie.

Zgodność zasady CAP

Zapewniają zrównoważone gwarancje co do spójności (Consistency), dostępności (Availability) i odporności na podział sieci (Partition Tolerance, P).

Wsparcie dla różnych typów danych

Obsługują różne typy danych, w tym dane strukturalne, półstrukturalne, nieskoordynowane i wielowartościowe.

Rodzaje baz NoSQL



Bazy klucz-wartość

(Key-Value): gdzie wszystko przechowywane jest jako para klucz-wartość.

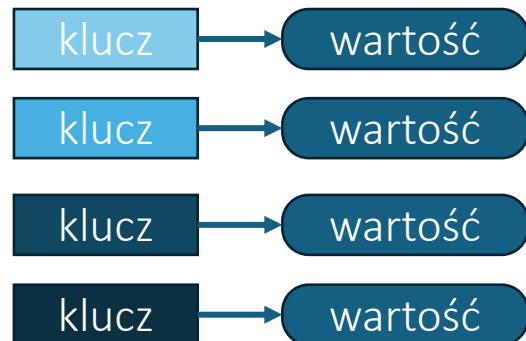
Przykłady: Redis, Amazon DynamoDB

Bazy dokumentowe

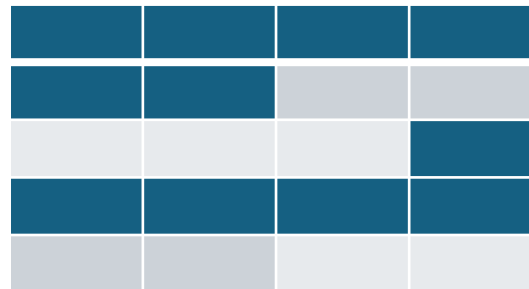
(Document): przechowują informacje jako dokumenty, często w formatach takich jak JSON.

Przykłady: MongoDB, Couchbase.

Bazy par klucz-wartość



Bazy rodziny kolumn

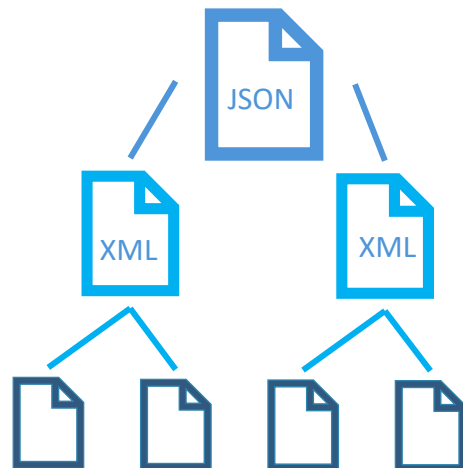


Bazy kolumnowe

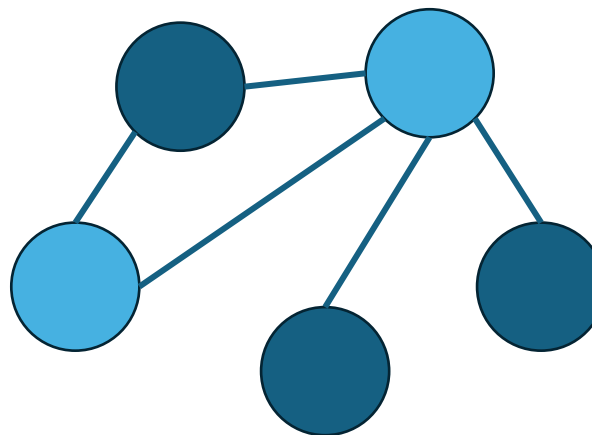
(Columnar): dane są przechowywane i wyszukiwane na podstawie kolumn, a nie wierszy.

Przykłady: Apache Cassandra, Google Bigtable.

Bazy dokumentów



Bazy grafowe



Bazy grafowe

(Graph): skoncentrowane na przechowywaniu informacji o węzłach i relacjach między nimi.

Przykład: Neo4j, JanusGraph

Bazy klucz-wartość

Bazy klucz-wartość

to jeden z najprostszych i najbardziej efektywnych modeli przechowywania danych, który polega na skojarzeniu unikalnego klucza z odpowiadającą mu wartością.

Chociaż ich struktura jest wyjątkowo prosta, bazy te stanowią fundament dla wielu zaawansowanych systemów, które obsługują ogromne ilości danych w czasie rzeczywistym.

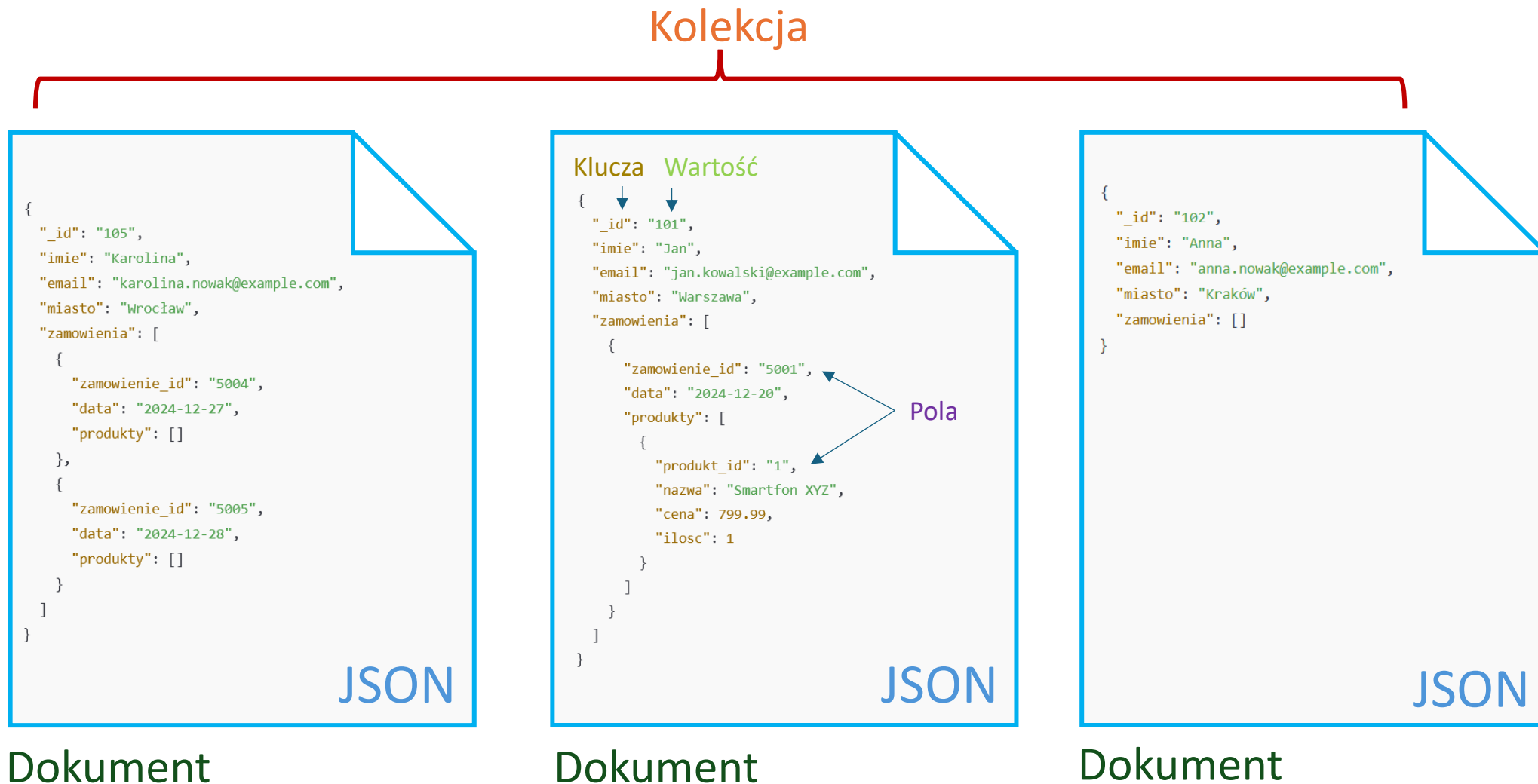
Przestrzeń: Klient	
osoba:1	Nowak
osoba:2	Kowalski
osoba:3	Malinowski

Przestrzeń: Piłkarz	
osoba:1	Smolarek
osoba:2	Boniek
osoba:3	Lato

Przestrzeń: Polityk	
osoba:1	Sikorski
osoba:2	Kuroń
osoba:3	Mazowiecki



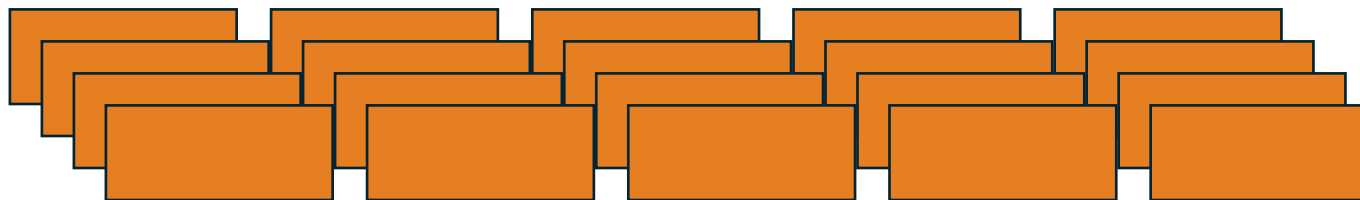
Baza dokumentów (NoSQL)



Bazy kolumnowe (NoSQL)



Kolumny



Bazy kolumnowe

Doskonale sprawdzają się w:

- zapytaniach dotyczących kilku kolumn
- zapytaniach agregujących duże ilości danych
- kompresji danych na poziomie kolumn

Bazy relacyjne

Doskonale sprawdzają się w:

- stopniowym ładowaniu danych
- Przetwarzaniu transakcyjnym
- zapytaniach dotyczących tylko kilku wierszy

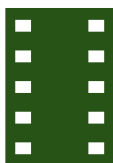
Wiersze



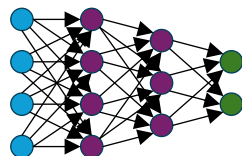
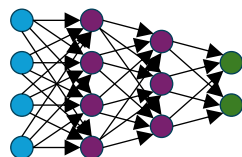
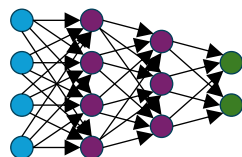
Bazy wektorowe



Dane



Embedding



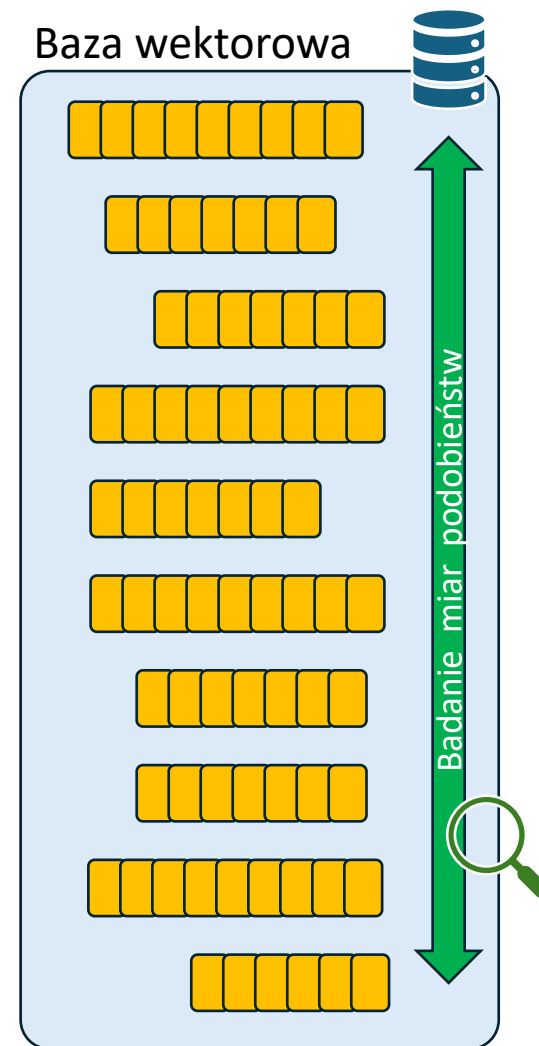
Wektory

$$\vec{v} = \begin{bmatrix} 5,5 \\ 5,5 \\ 3,4 \\ 2,5 \\ 4,6 \\ 8,4 \\ 6,8 \\ 9,1 \\ \dots \\ n \end{bmatrix}$$

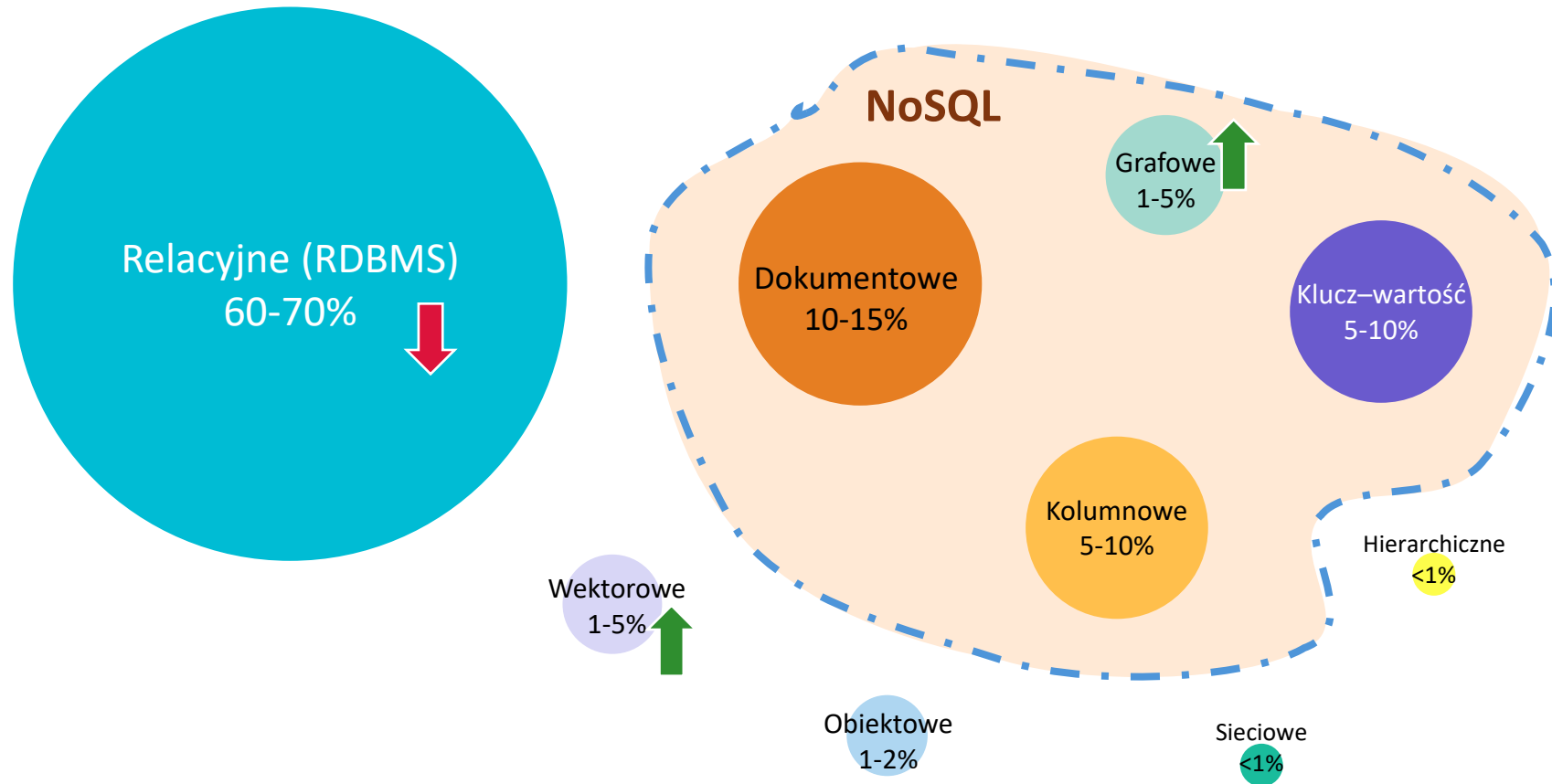
$$\vec{v} = \begin{bmatrix} 4,5 \\ 3,5 \\ 3,4 \\ 2,5 \\ 1,6 \\ 8,4 \\ 6,8 \\ 9,1 \\ \dots \\ n \end{bmatrix}$$

$$\vec{v} = \begin{bmatrix} 8,5 \\ 7,5 \\ 3,4 \\ 4,5 \\ 1,6 \\ 8,4 \\ 5,8 \\ 9,1 \\ \dots \\ n \end{bmatrix}$$

Baza wektorowa



Typy systemów baz danych



Wprowadzenie do baz NoSQL

Grafowa baza danych

to system zarządzania bazą danych zaprojektowany do przechowywania, zarządzania i wyszukiwania informacji w strukturach grafowych.



Elastyczność

Łatwość modelowania
złożonych relacji
i dynamicznych
schematów.



Wydajność

Szybka nawigacja
i zapytania w
złożonych sieciach
relacji.



Intuicyjność

Naturalne
odwzorowanie
złożonych struktur
i relacji świata
rzeczywistego.

Bazy grafowe należą do klasy baz **NoSQL**

Podstawy teorii grafów i elementy grafu

Definicja

W obszarze aparatu matematycznego Grafem jest uporządkowana para:

$$G = \langle N, E \rangle$$

gdzie:

N – zbiór wierzchołków

E – rodzina krawędzi (rodzina w odróżnieniu od zbioru może zawierać elementy powtarzające się)

Krawędzią grafu $e \in E$ jest uporządkowana para $e = \langle n_1, n_2 \rangle$ gdzie $n_1, n_2 \in N$.

ponadto:

$N \neq \emptyset$ - graf ma co najmniej jeden wierzchołek

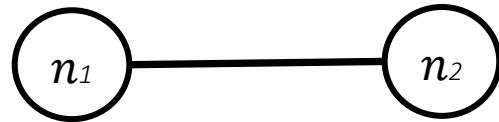
$N \cap E = \emptyset$ - zbiór wierzchołków i rodzina krawędzi nie mają części wspólnej.

Jeżeli rodzina E zawiera powtarzające się elementy, to takie krawędzie nazywamy równoległymi lub wielokrotnymi

Podstawy teorii grafów i elementy grafu

Klasyfikacja grafów

Graf niezorientowany (nieskierowany): jeżeli $\langle n_1, n_2 \rangle \in E$ i $\langle n_2, n_1 \rangle \in E$, to taka para jest nazywana krawędzią niezorientowaną (nieskierowaną) lub wprost krawędzią.

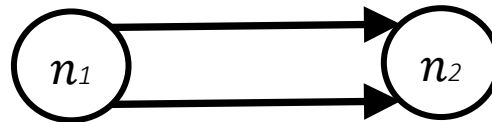


Graf zorientowany (skierowany): jeżeli $\langle n_1, n_2 \rangle \in E$ i $\langle n_2, n_1 \rangle \notin E$, to taka para jest nazywana krawędzią zorientowaną (skierowaną) lub wprost krawędzią..



Multigraf: graf z krawędziami równoległymi.

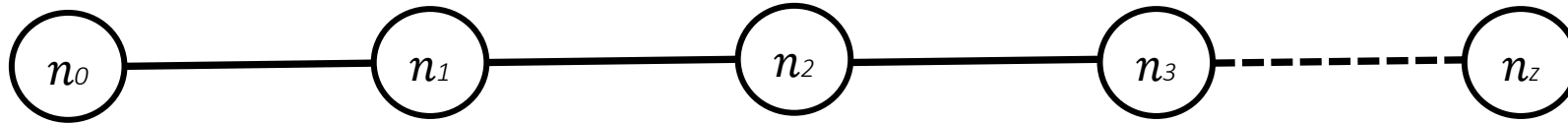
Unigraf: graf bez krawędzi równoległych.



Podstawy teorii grafów i elementy grafu

Ścieżka i droga w grafie

Ścieżką w grafie: to naprzemienny ciąg wierzchołków i krawędzi w postaci $\{n_0 < n_0, n_1 >, n_1 < n_1, n_2 > n_2 < n_2, n_3 >, \dots, n_{(z-1)} < n_{(z-1)}, n_z >, n_z\}$, gdzie *wszystkie krawędzie są różne*.



Droga w grafie: to ścieżka, w której *wszystkie wierzchołki* są różne ewentualnie z wyjątkiem pierwszego i ostatniego wierzchołka.

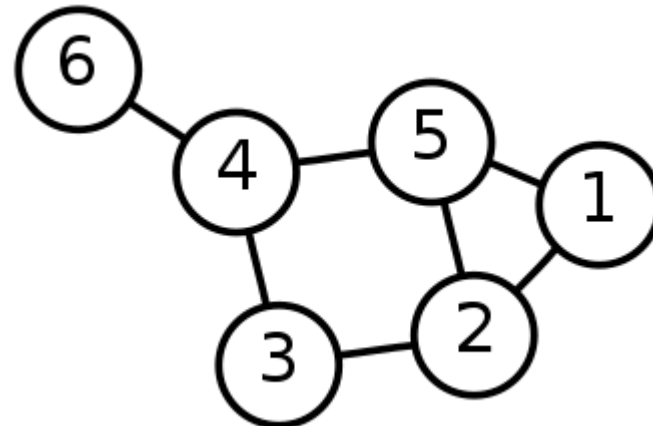
Podstawy teorii grafów i elementy grafu

Elementy Grafu

Wierzchołki (Węzły): Podstawowe jednostki, z których składa się graf. Mogą reprezentować dowolny typ obiektu.

Krawędzie (Łuki, Relacje): Reprezentują związki między węzłami. Mogą być skierowane (wskazujące kierunek relacji od jednego węzła do drugiego) lub nieskierowane.

Wagi (Wartości, Atrybuty, Etykiety): Wierzchołki i krawędzie mogą zawierać atrybuty, które przechowują dodatkowe informacje.



Obszary zastosowań baz grafowych

Wykrywanie i
analiza oszustw

Monitorowanie
infrastruktury
teleinformatycznej

Silnik rekomendacji
produktów

Analiza sieci
społecznościowych

Zarządzanie
łańcuchem dostaw

Sztuczna
inteligencja

Typowe przypadki użycia grafów

Przykładowe zastosowania grafów

Mapy Drogowe:

- Grafy są używane do reprezentowania sieci dróg, gdzie węzły reprezentują skrzyżowania, a krawędzie reprezentują drogi.

Sieci Społecznościowe:

- W sieciach społecznościowych, takich jak Facebook czy Instagram, użytkownicy są węzłami, a krawędzie reprezentują relacje między nimi (np. znajomi, followersi).

Sieci Komputerowe:

- Komputery są węzłami, a połączenia między nimi są krawędziami.

Struktura stron www:

- Strony internetowe są węzłami, a linki są krawędziami.

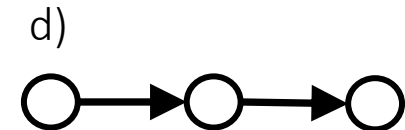
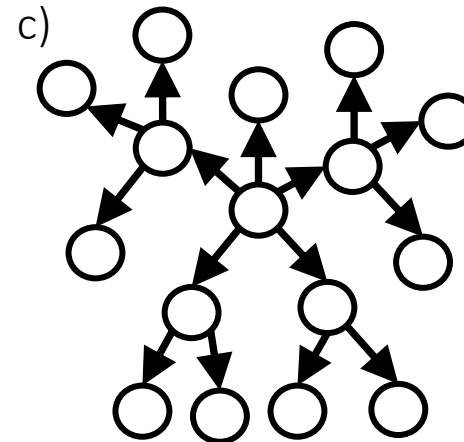
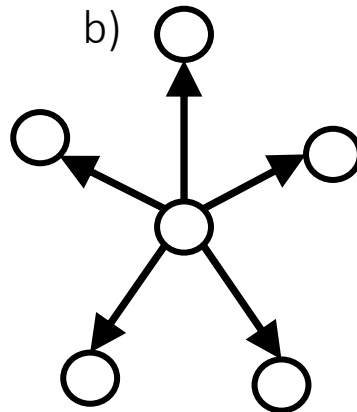
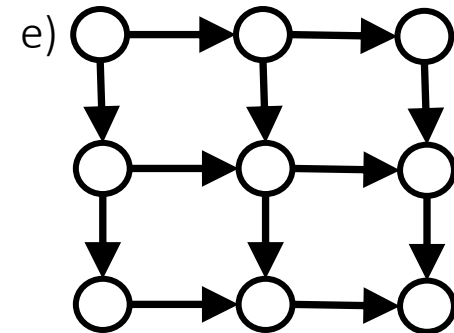
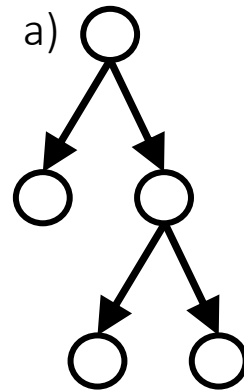
Rekomendacje Produktów:

- Produkty są węzłami, a krawędzie reprezentują relacje między produktami, takie jak "często kupowane razem".

Wybrane formy grafów

Przykładowe formy grafów

- a) Drzewo np.: struktura katalogów w systemie plików
- b) Gwiazda np.: komputerowa sieć lokalna
- c) Płatek śniegu np.: organizacja Państwa (województwa, powiaty, gminy, miasta...)
- d) Łańcuch np.: rzeka
- e) Siatka np.: sieć uliczna



Elementy baz grafowych

Węzły (Nodes)

Reprezentują obiekty takie jak osoby, miejsca, wydarzenia.

Krawędzie (Edges, Relationships)

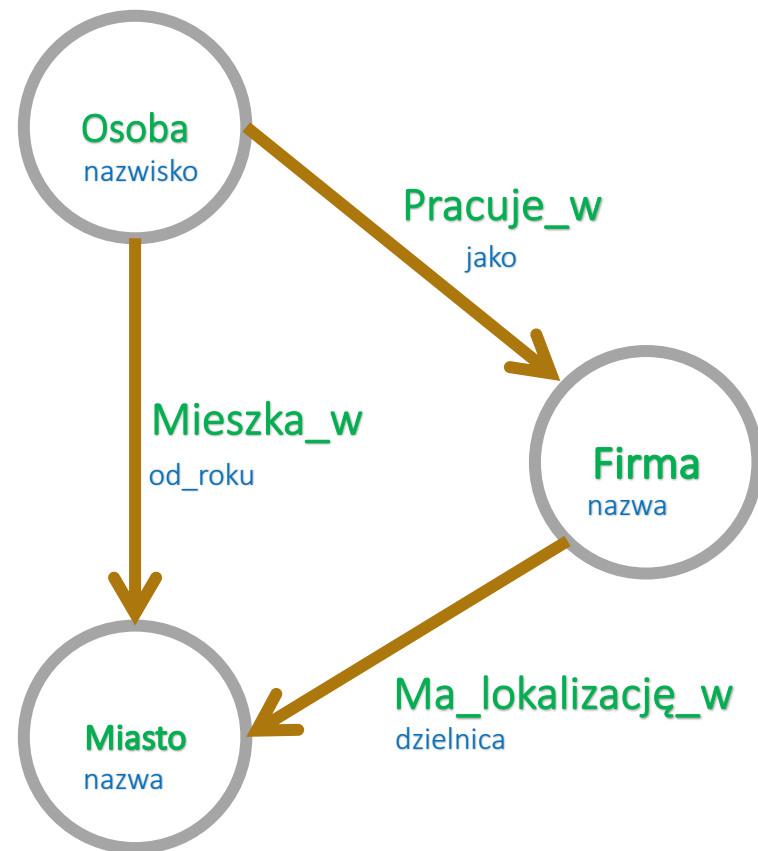
Reprezentują relacje między obiektami, np. znajomość, przepływ informacji.

Etykiety (Labels)

Klasyfikują węzły i krawędzie, ułatwiając organizację i wyszukiwanie danych.

Właściwości (Properties)

Informacje dodatkowe przypisane do węzłów i krawędzi, np. nazwa, dzielnica, rozmiar itd.

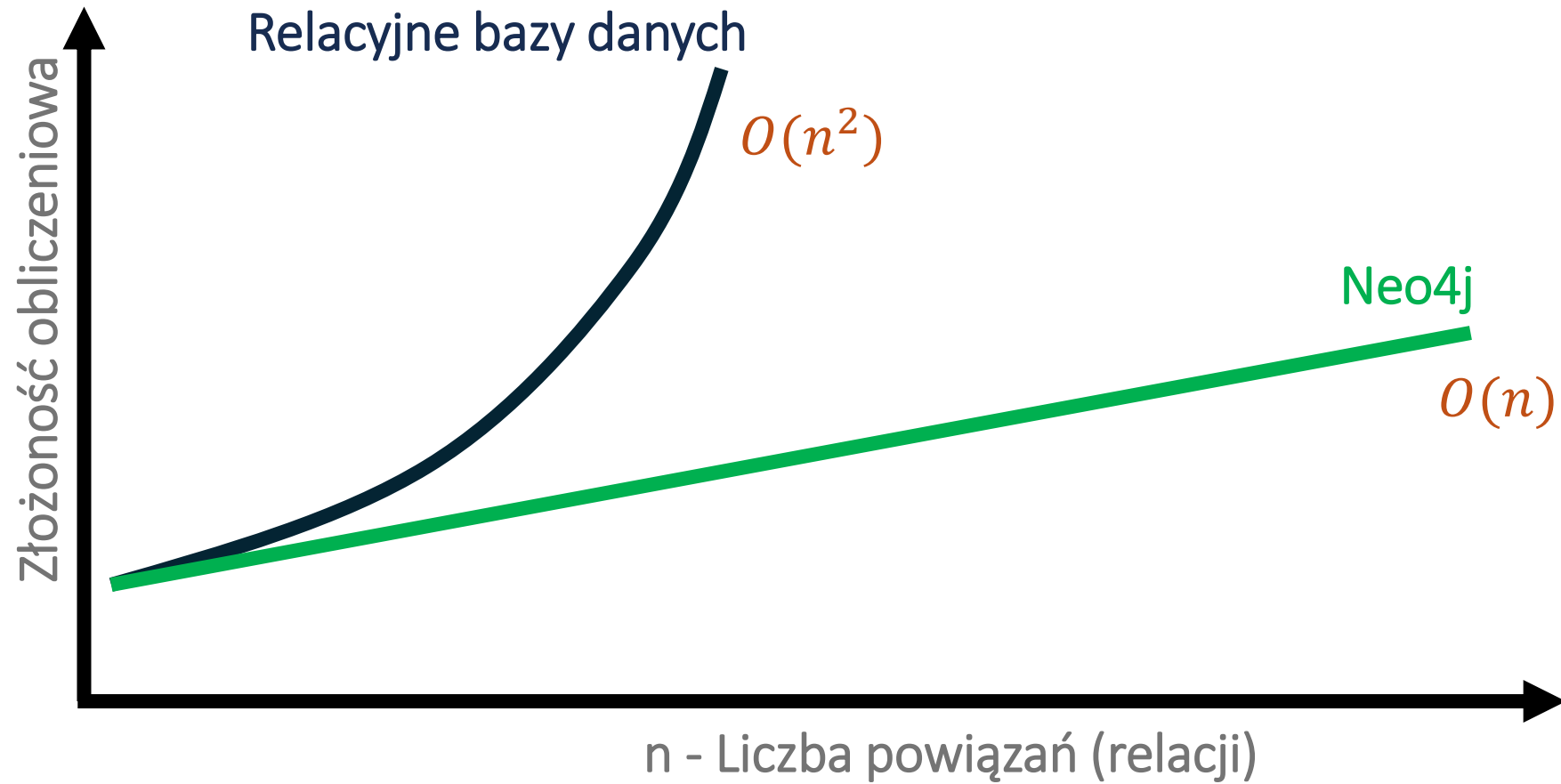


Bazy grafowe vs. Bazy relacyjne



Cecha	Baza relacyjna	Baza grafowa
Model danych	Model relacyjny	Model grafowy
Jednostki danych	Wiersze w tabeli	Węzły i krawędzie w grafie
Relacje między danymi	Klucze obce	Krawędzie
Zapytania	SQL	Cypher
Wydajność	Dobra dla prostych zapytań, zła dla złożonych zapytań z wieloma JOIN-ami	Dobra dla złożonych zapytań, szczególnie tych, które wymagają analizy relacji między danymi
Elastyczne modelowanie danych	Niezbyt elastyczne	Bardzo elastyczne
Typowe zastosowania	Systemy ERP, CRM, zarządzanie danymi transakcyjnymi	Sieci społecznościowe, analiza danych, wyszukiwarka internetowa, system rekomendacji

Bazy grafowe vs. Bazy relacyjne



Bazy grafowe vs. Bazy relacyjne



Relacyjna baza danych

Osoba

id	nazwisko
1	Nowak
2	Kowalski

Firma

id	Nazwa
1	Alfa
2	Beta

Miasto

id	nazwa
1	Warszawa
2	Kraków

Mieszka_w

idosoba	idmiasto	od_roku
1	2	1990
2	1	2011

Pracuje_w

idosoba	idfirma	jako
1	2	Mechanik
2	1	Księgowy

Ma_lokalizację_w

idfirma	idmiasto	dzielnica
1	1	Mokotów
1	2	Kazimierz
2	1	Ochota

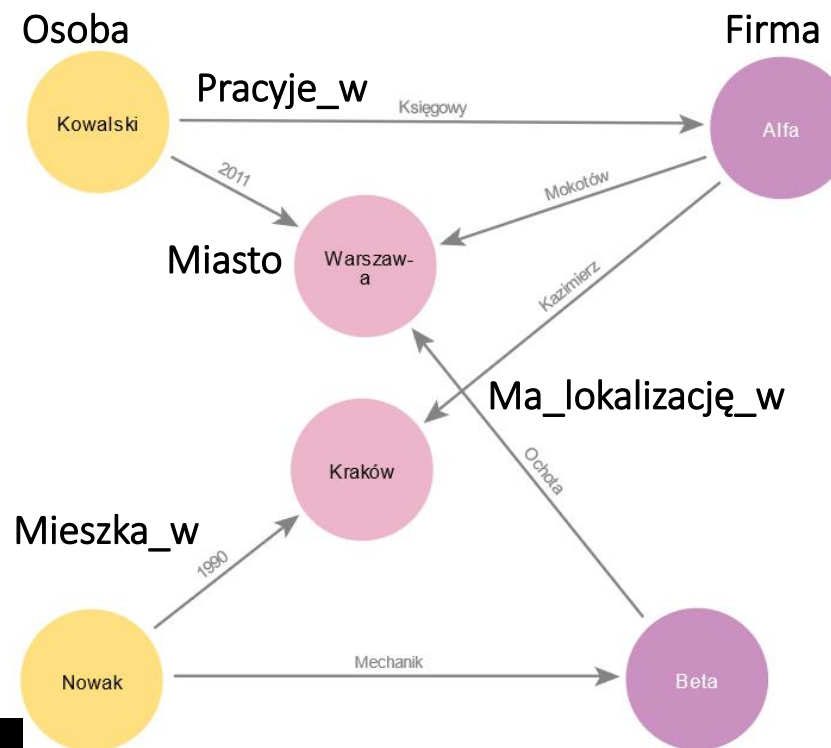
SQL

```
SELECT Firma.nazwa, Pracuje_w.jako FROM Firma  
JOIN Pracuje_w ON Pracuje_w.idfirma = Firma.id  
JOIN Osoba ON Osoba.id = Pracuje_w.idosoba  
WHERE Osoba.nazwisko = "Kowalski"
```

Jako kto i gdzie pracuje Kowalski?



Grafowa baza danych



Cypher

```
MATCH (o:Osoba {nazwisko:'Kowalski'})-[p:Pracuje_w]->(f:Firma)
RETURN f.nazwa, p.jako
```

nazwa	jako
Alfa	Księgowy

Cypher - Relacyjne Zapytania

Cypher (CQL)

to język zapytań deklaratywny, stworzony przez Neo4j dla baz danych grafowych. Umożliwia on efektywne tworzenie, modyfikowanie oraz przeszukiwanie grafów w bazie danych Neo4j.

Cypher skupia się na tym, co chcemy osiągnąć, a nie na tym, jak to osiągnąć

```
MATCH (o:Osoba)-[p:PRACUJE_DLA]->(f:Firma)
WHERE f.nazwa = 'Alfa' AND o.wiek > 30
RETURN o.imie, o.wiek, f.nazwa
ORDER BY o.wiek DESC LIMIT 10
```

CREATE, MERGE

- Tworzenie danych

MATCH

- Wyszukiwanie danych

SET, REMOVE

- Aktualizacja danych

DELETE

- Usuwanie danych

WITH, RETURN

- Kontrola przepływu

Porównanie Cypher z SQL

Powiązanie **zadań** z **procesem** i realizującymi je **osobami** pracującymi na **stanowiskach** w **komórkach** wewnętrznych organizacji

Cypher

```
MATCH d=(n:Proces {nazwa: 'Proces A'})-[:ZAWIERA_ZADANIE]->()-[:REALIZOWANE_PRZEZ]->()-[:PRACUJE_JAKO]->()-[:JEST_W]->(k)
RETURN d
```

SQL

```
SELECT * FROM Proces
INNER JOIN Zadanie ON Proces.id = Zadanie.idProces
INNER JOIN Osoba ON Zadanie.idOsoba = Osoba.id
INNER JOIN Stanowisko ON Osoba.idStanowisko = Stanowisko.id
INNER JOIN Komorka ON Stanowisko.idKomorka = Komorka.id
WHERE Proces.nazwa = 'Proces A';
```

Warunek, związki tylko 1:n

W przypadku związki n:n należało by dodać
- 4 tabele intersekcji
- 4 linie INNER JOIN



Metody wykorzystania grafów

Wizualizacja Danych

Bazy grafowe pozwalają na intuicyjną wizualizację złożonych struktur danych, co ułatwia zrozumienie relacji między nimi i identyfikację wzorców, co jest niezwykle przydatne w analizie danych organizacyjnych.

Relacyjne Zapytania

Dzięki językowi zapytań takiemu jak Cypher, bazy grafowe umożliwiają wykonywanie relacyjnych zapytań, które odzwierciedlają relacje między danymi, co pozwala na szybkie i efektywne pozyskiwanie informacji o organizacji.

Algorytmy Grafowe

Bazy grafowe oferują zestaw wbudowanych algorytmów grafowych, które umożliwiają zaawansowaną analizę danych, taką jak wykrywanie społeczności, ocena ważności węzłów czy analiza sieci, co pozwala na bardziej zaawansowane i precyzyjne podejmowanie decyzji w organizacji.

Algorytmy grafowe



Algorytmy wyszukiwania ścieżek

Wyszukują optymalne ścieżki pomiędzy węzłami dla określonych kryteriów



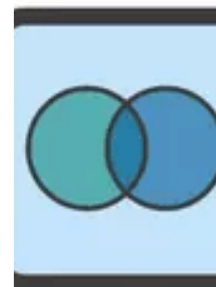
Algorytmy centralności

Pozwalają ocenić, jakie węzły w sieci są najważniejsze, czyli mogą mieć największy wpływ



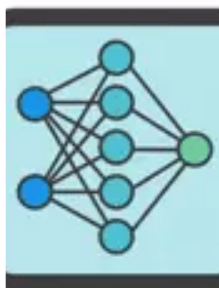
Algorytmy wspólnoty

Identyfikują skupiska węzłów, które są gęściej połączone między sobą niż z resztą sieci



Algorytmy podobieństwa

Obliczają stopień podobieństwa między elementami



Algorytmy detekcji anomalii

Wykorzystywane są do identyfikowania nieoczekiwanych wzorców lub obserwacji w danych



Algorytmy predykcji relacji

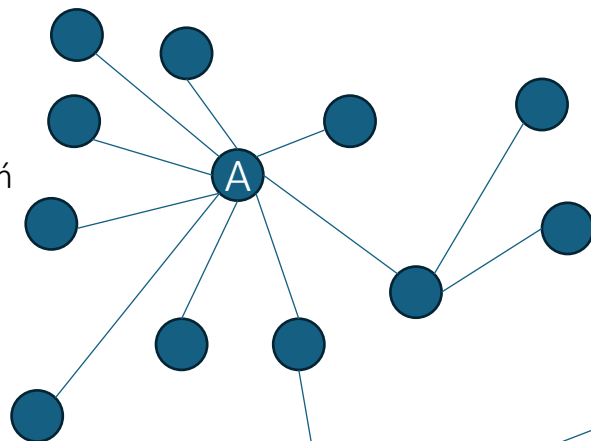
Prognozują czy dwa węzły w grafie mogą w przyszłości stworzyć połączenie

Algorytmy centralności

Degree Centrality
(Centralność stopnia)

Liczba połączeń?

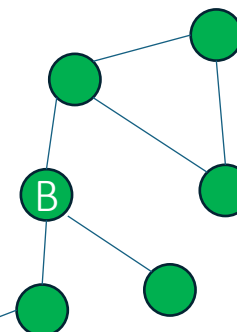
"A" ma najwyższy stopień



Closeness Centrality
(Centralność pośrednictwa)

Który węzeł może najłatwiej dotrzeć do wszystkich innych węzłów w grafie?

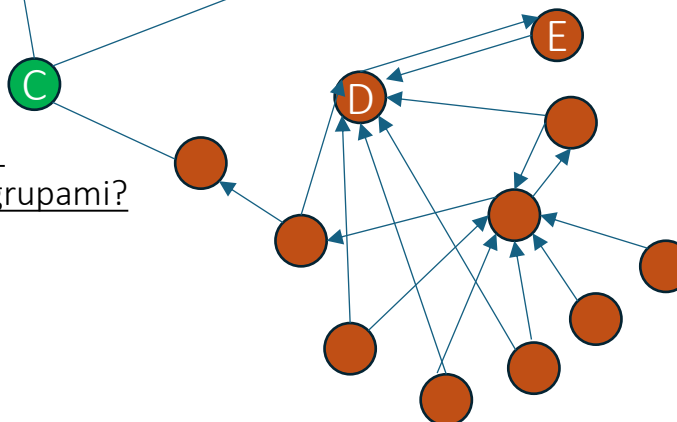
"B" jest najbliższy z najmniejszą liczbą połączeń



Betweenness Centrality
(Centralność bliskości)

Który węzeł ma największą kontrolę nad przepływem między węzłami i grupami?

"C" jest mostem



Louvain Modularity
(Modułowość Louvain)

Jak efektywnie podzielić graf na społeczności?



PageRanking

Który węzeł jest najważniejszy?

"D" jest na czele na podstawie liczby i wagi linków przychodzących

"E" jest następny, ze względu na wpływ linku "D"

Bazy grafowe



<https://neo4j.com/>



<https://www.ontotext.com/products/graphdb/>



<https://orientdb.org/>



<https://memgraph.com/memgraphdb>



<https://arangodb.com/>

Narzędzia do pracy ze strukturami grafowymi



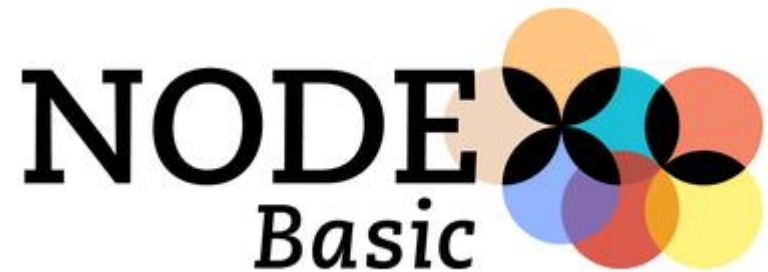
<https://neo4j.com/>



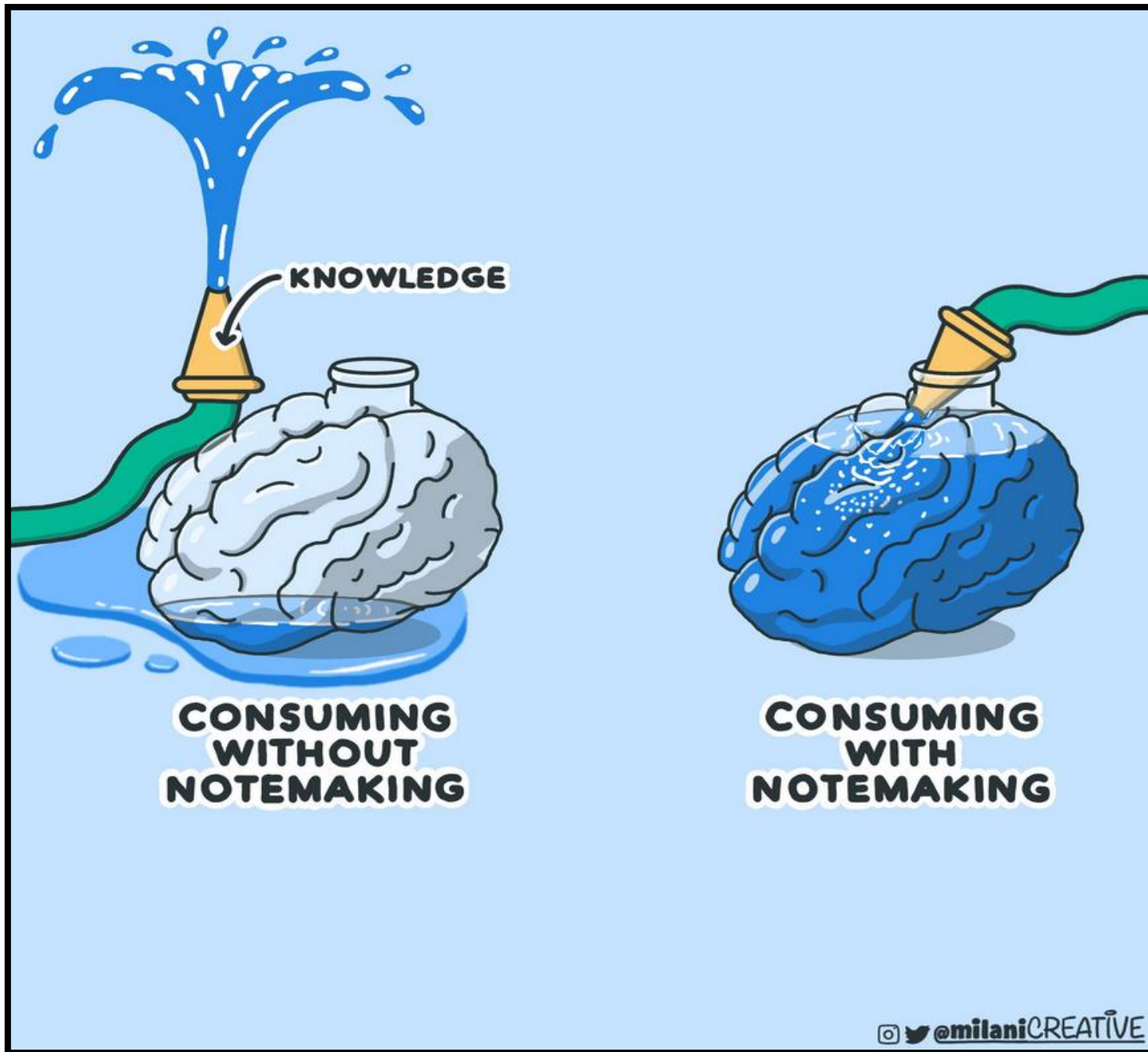
<https://gephi.org/>

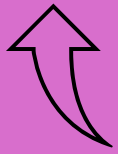


<https://cytoscape.org/>



<https://nodexl.com/>





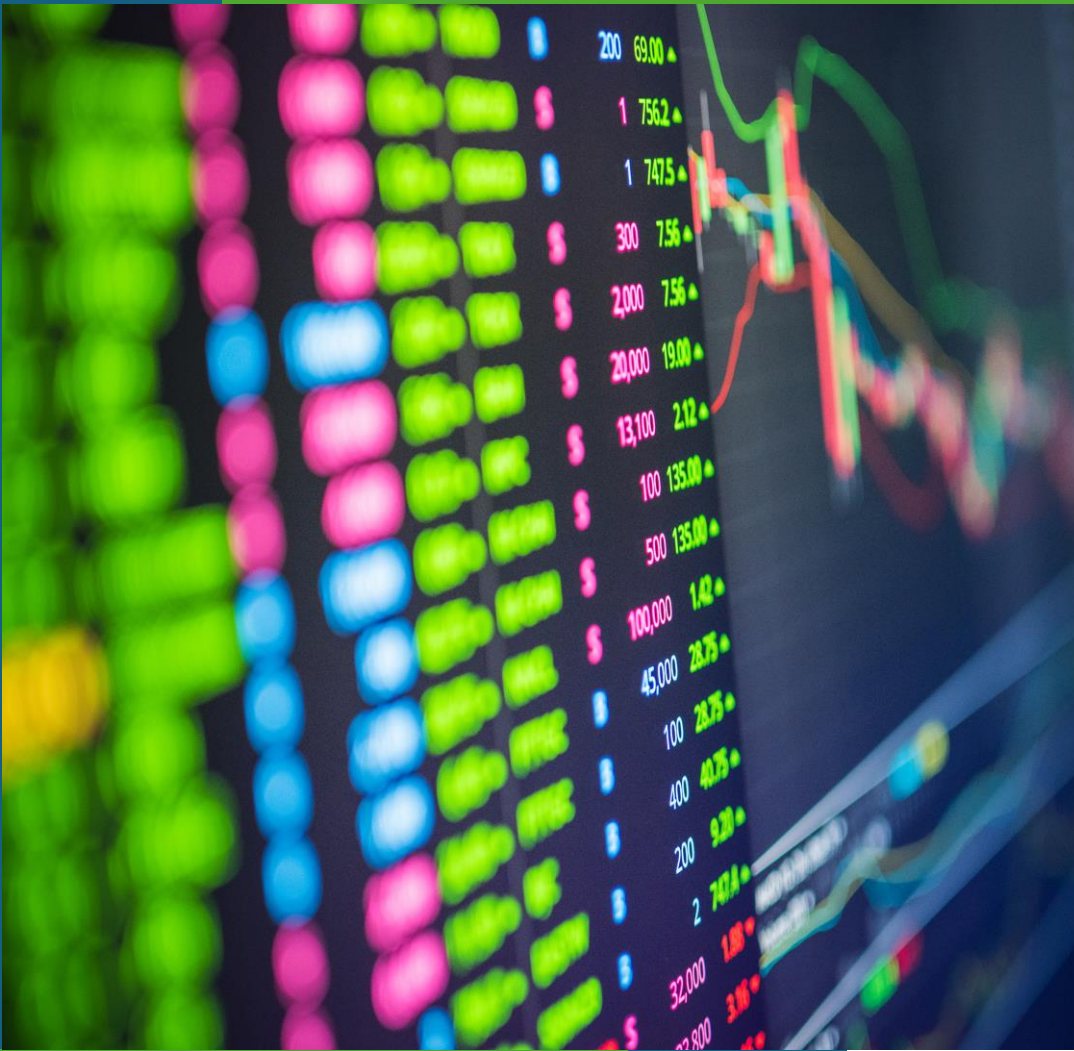
Temat 2 .

dr inż. Michał Malinowski

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

Przygotowanie środowiska
pracy systemu zarządzania
bazą danych Neo4j

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



Zagadnienia (2h)

- Przygotowanie środowiska
- Założenie kont dostępowych
- Inicjacja dedykowanych instancji baz danych
- Poznanie podstawowych funkcji środowiska

Architektura systemu zarządzania bazą danych Neo4j

Wprowadzenie

Neo4j to open-source'owy system zarządzania bazą danych typu NoSQL, zorientowany na grafy.

Dostępny w dwóch wersjach:

- Community Edition (open-source)
- Enterprise Edition (komercyjna).

Graf składa się z:

- węzłów (Nodes) reprezentujących obiekty (np. osoby, produkty)
- krawędzi (Relationships) reprezentujących relacje między tymi obiektami.

Węzły i krawędzie mogą przechowywać dodatkowe informacje jako właściwości (Properties).

Architektura systemu zarządzania bazą danych Neo4j

Model Danych i Język Zapytań Cypher

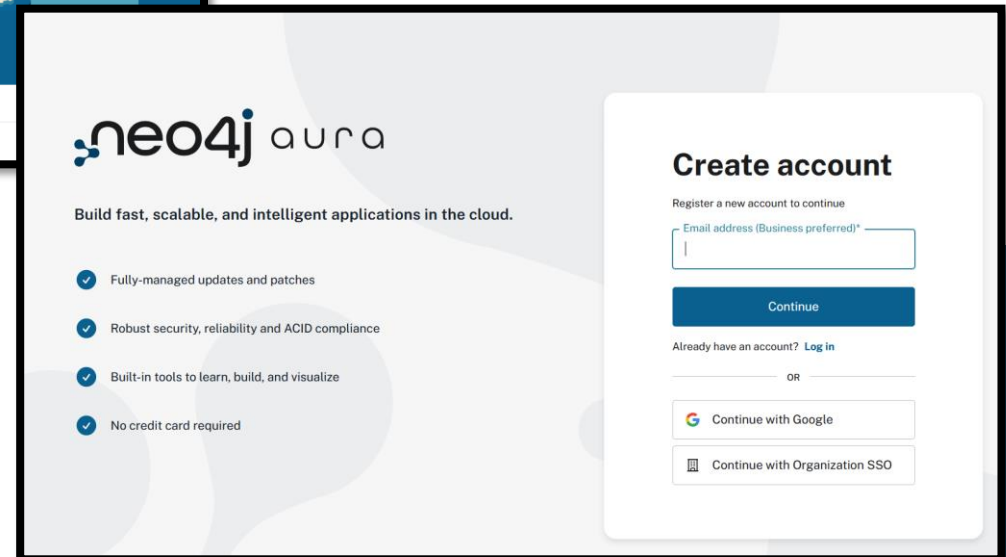
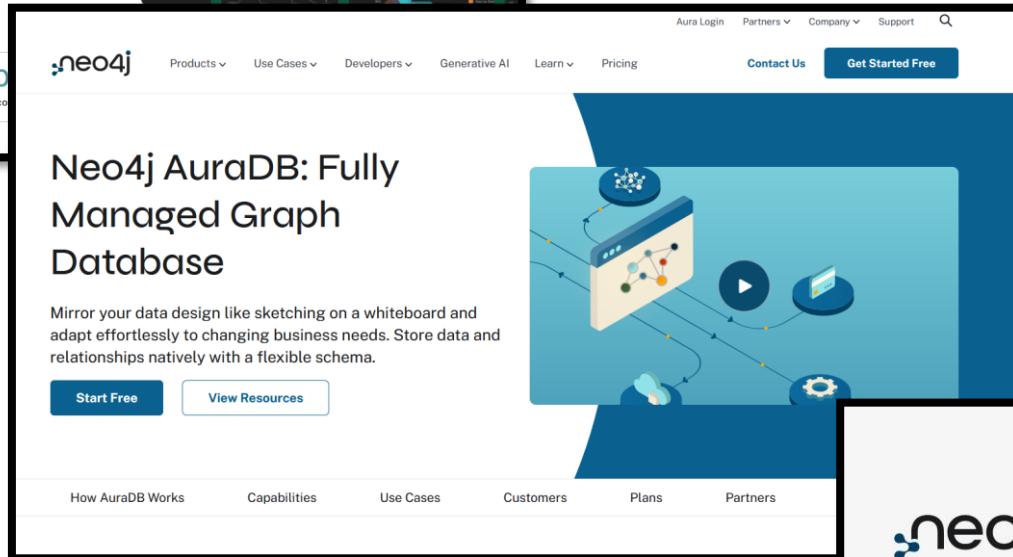
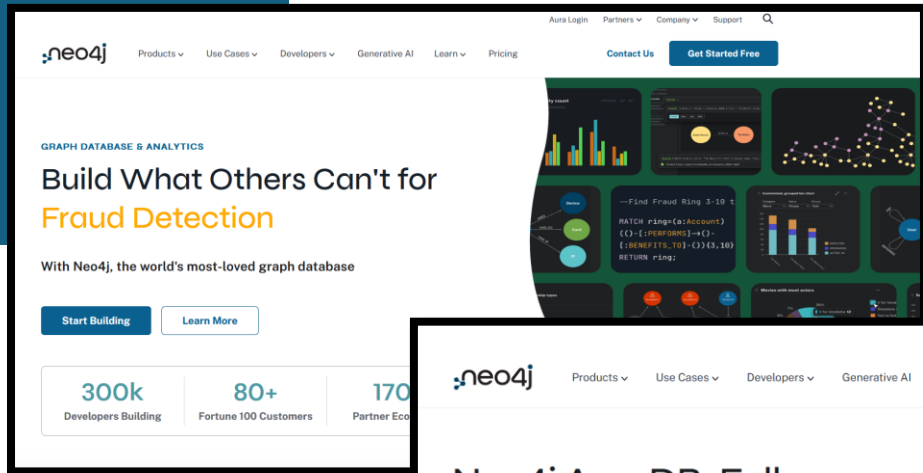
Neo4j stosuje model grafu skierowanego, etykietowanego (ważonego)

Etykiety (Labels) są używane do definiowania typów węzłów i relacji oraz do grupowania i szybkiego wyszukiwania węzłów o określonym typie.

Język zapytań Cypher jest deklaratywny, zaprojektowany specjalnie do pracy z grafami, z intuicyjną, czytelną składnią.



<https://neo4j.com>



Przygotowania środowiska

Modele architektury Neo4j

Neo4j AuraDB to w pełni zarządzana baza danych grafowych w chmurze, która jest hostowana i zarządzana przez Neo4j. Użytkownicy mogą uzyskiwać dostęp do swoich danych za pomocą interfejsu API REST, a także za pomocą różnych narzędzi i klientów grafowych.

Jest dobrym wyborem dla użytkowników, którzy cenią prostotę.

Neo4j AuraDS to baza danych grafowych w chmurze, którą użytkownicy mogą hostować i zarządzać samodzielnie. Użytkownicy mają większą kontrolę nad swoimi danymi i infrastrukturą, ale również ponoszą większą odpowiedzialność za ich bezpieczeństwo i konserwację.

Jest dobrym wyborem dla użytkowników, którzy chcą mieć większą kontrolę nad swoimi danymi i infrastrukturą, lub potrzebują grafowej bazy danych, która jest zgodna z określonymi wymaganiami bezpieczeństwa

Przygotowania środowiska

Kluczowe różnice między architekturami AuraDB a AuraDS

Cecha	Neo4j AuraDB	Neo4j AuraDS
Hostowane	W pełni zarządzane przez Neo4j	Samoobsługowe
Kontrola	Mniejsza kontrola	Większa kontrola
Odpowiedzialność	Mniejsza odpowiedzialność	Większa odpowiedzialność
Cena	Zazwyczaj droższe	Zazwyczaj tańsze
Skalowalność	Automatyczna skalowalność w górę i w dół	Skalowalność wymaga interwencji użytkownika
Dostępność	99,9% SLA	99,99% SLA
Bezpieczeństwo	Zarządzane przez Neo4j	Zarządzane przez użytkownika
Obsługa klienta	24/7/0365	24/7/0365
Funkcje	Wszystkie funkcje Neo4j	Wybrane funkcje Neo4j
Integracja z innymi produktami Neo4j	Łatwa integracja	Wymaga dodatkowego wysiłku
Łatwość użycia	Łatwe w użyciu	Wymaga więcej doświadczenia
Możliwość dostosowywania	Ograniczona możliwość dostosowywania	Duża możliwość dostosowywania

Data services

Instances

Import

Data API

Tools

Query

Explore

Operations

Metrics

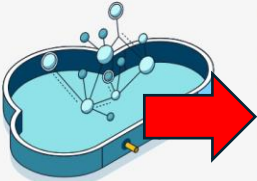
Logs

Project

Users

Billing

Settings



Create your first instance

Configure your instance and start realizing the endless possibilities with graphs.

[Have data? Import existing data into Neo4j.](#)

Create instance

Credentials for Instance01

Username
neo4j

Password

Note that the password will not be available after this point.

Download and continue

Data services

Instances

Import

Data API

Tools

Query

Explore

Operations

Metrics

Logs

Project

Users

Billing

Settings

Search

Create instance

Instance01

● RUNNING

ID: 9a23656e

Type: AuraDB Free Nodes: 0 (0%) Relationships: 0 (0%) Upgrade

> Metrics

Data services

Instances

Import

Data API

Tools

Query

Explore

Operations

Metrics

Logs

Project

Users

Billing

Settings

Instance: Instance01 Database: neo4j User: Aura (gotyx@gra24h.pl)

Database information

Nodes (0)

Relationships (0)

Property keys

neo4j\$

\$:welcome

GUIDE

Query fundamentals

You'll learn to:

- write basic queries;
- view graph and tabular results;
- perform queries to answer questions;
- perform advanced queries.

DATASET

Try Neo4j with the Movie Graph

Dive into a fully-built graph example to kickstart your Neo4j journey. Discover key query patterns through a fun, familiar world of actors and films.

Inicjacja dedykowanych instancji baz danych

Protokoły używane do łączenia się z bazą danych Neo4j

Neo4j+s to protokół tekstowy, który jest używany do wykonywania zapytań Cypher i uzyskiwania dostępu do danych w bazie danych.

Bolt+s to protokół binarny, który jest używany do wykonywania bardziej złożonych operacji, takich jak tworzenie i usuwanie węzłów i relacji, a także zarządzania bazą danych.

Cecha	Neo4j+s	Bolt+s
Typ danych	Tekst	Binarne
Zapytania	Cypher	Cypher, dowolne inne
Wydajność	Mniejsza	Większa
Skomplikowanie	Prostsze	Bardziej skomplikowane
Trudność w użyciu	Łatwe do użycia	Trudne w użyciu

Data services

Instances

Import

Data API

Tools

Query

Explore

Operations

Metrics

Logs

Project

Users

Billing

Settings

Instance: Instance01 Database: neo4j User: Aura (gotyx@gra24h.pl)

Database information

Nodes (0)

*

Relationships (0)

*

Property keys

neo4j\$

neo4j\$ MATCH (n) RETURN n LIMIT 25;

No changes, no records

Completed after 44 ms

\$:welcome

GUIDE

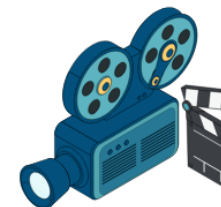


Query fundamentals

You'll learn to:

- write basic queries;
- view graph and tabular results;
- perform queries to answer questions;
- perform advanced queries.

DATASET



Try Neo4j with the Movie Graph

Dive into a fully-built graph example to kickstart your Neo4j journey. Discover key query patterns through a fun, familiar world of actors and films.

Go to Learn Center

See the full catalogue of guides to help you get the most out of Neo4j.

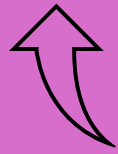
[To Learn Center](#)

\$:connect

Connected to Instance01

NIE MOŻESZ
PRZESUNĄĆ OBRAZU PALCEM,
MUSISZ PODEJŚĆ DO
NASTĘPNEGO.





Temat 3 .

Podstawy korzystania z języka zapytań Cypher

dr inż. Michał Malinowski

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

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



Zagadnienia (2h)

- Składnia języka Cypher
- Dopasowywanie wzorców
- Tworzenie węzłów i relacji
- Nadawanie etykiet i właściwości
- Komenda MARGE
- Operacje usuwania
- Typy właściwości

Składnia języka Cypher

Wprowadzenie

- Cypher to deklaratywny język zapytań zaprojektowany dla grafów.
- Węzły (*node*) są reprezentowane przez nawiasy ()
- Używamy dwukropka do oznaczenia etykiety (*label*), na przykład (:Osoba)
- Relacje (*relationship*) między węzłami są zapisywane za pomocą dwóch myślników, na przykład (:Osoba)--(:Firma)
- Kierunek (*direction*) relacji jest wskazywany za pomocą symbolu < lub >, na przykład (:Osoba)->(:Firma)
- Typ (*type*) relacji jest zapisywany za pomocą nawiasów kwadratowych po dwukropku, na przykład [:PRACUJE_W]
- Właściwości (*property*) to pary klucz/wartość (*key/value*) zapisywane w nawiasach sześciennych, z dwukropkiem i w apostrofach, na przykład {nazwisko: 'Kowalski'}

```
(p:Firma {nazwa: 'GoodDom'})<-[:PRACUJE_W]-(m:Osoba {nazwisko: 'Kowalski'})
```

Dopasowywanie wzorców

Dopasowanie

Cypher działa poprzez dopasowywanie wzorców za pomocą słowa kluczowego MATCH. Klauzula MATCH jest podobna do klauzuli FROM w składni SQL.

Rozróżnia duże i małe litery !!!

Cypher

```
MATCH (p:Osoba {nazwisko: 'Kowalski'}) RETURN p
```

Cypher

```
MATCH (p:Osoba) WHERE p.nazwisko = 'Kowalski' RETURN p
```

SQL

```
SELECT * FROM osoba WHERE nazwisko='Kowalski'
```

Tworzenie węzłów i relacji

Węzły (node)

CREATE (n), (m)

MATCH (n)
RETURN n

CREATE (n), (m), (k)

The screenshot displays the Neo4j Cypher console interface. The query entered is `neo4j$ MATCH (n)*RETURN n`. The interface has three tabs: 'Graph', 'Table', and 'RAW'. The 'Graph' tab is active, showing a visual representation of the query results as five purple circular nodes. Each node contains the text `4:49a54 896-a20...`. To the right of the graph is a 'Results overview' panel showing 'Nodes (5)' and a summary `* (5)`. At the bottom right, a status message reads: 'Started streaming 5 records after 17 ms and completed after 18 ms.'

Tworzenie węzłów i relacji

Relacje (relationship)

MATCH (a), (b)

CREATE (a)-[r:relacja]->(b)

RETURN type(r)

MATCH p=()-[:relacja]->()

RETURN p

neo4j\$

```
$ MATCH p=()-[:relacja]->() RETURN p;
```

Graph

Table

RAW



Created

Nadawanie etykiet i właściwości

Etykiety (label)

```
CREATE (n:Osoba)
```

```
CREATE (n:Osoba:Polska)
```

```
CALL db.labels()
```

Właściwości (property) klucz/wartość (key/value)

```
CREATE (n:Osoba {imie: 'Jan',  
nazwisko: 'Nowak', rola: 'Klient'})
```

Usuwanie właściwości

```
MATCH (n:Osoba {nazwisko:  
'Nowak'})
```

```
REMOVE n.rola
```

```
RETURN n
```

Graph	Table	RAW
n		
(:Osoba)		
(:Osoba)		
(:Osoba:Polska)		
(:Osoba {imie: "Jan", nazwisko: "Nowak", rola: "Admin"})		
Showing 1-4 of 4 results		

Komenda MERGE

Kluczowe różnice między komendami MERGE a CREATE

Główna różnica między nimi polega na tym, że MERGE próbuje najpierw znaleźć węzeł lub relację o określonym identyfikatorze, a jeśli go nie znajdzie, tworzy nowy. Z kolei CREATE zawsze tworzy nowy węzeł lub relację.

Cecha	MERGE	CREATE
Próbuje znaleźć istniejący węzeł lub relację	Tak	Nie
Jeśli nie znajdzie istniejącego węzła lub relacji, tworzy nowy	Tak	Tak
Jest bardziej wydajny, jeśli węzeł lub relacja już istnieje	Tak	Nie
Jest bardziej uniwersalny	Nie	Tak

Komenda MERGE

Tworzy węzły i relacji podobnie jak CREATE

Węzły (node)

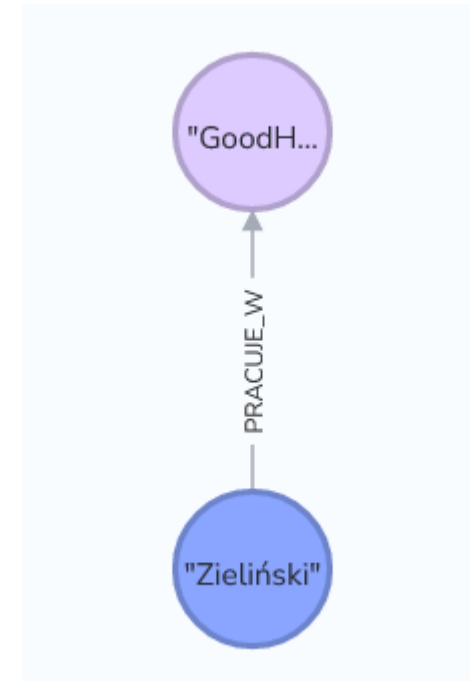
MERGE (n:Osoba {imie: 'Adam',
nazwisko: 'Kowalski', rola: 'Klient'})

Relacje (relationship)

MERGE (p:Osoba {nazwisko:
'Zieliński'})

MERGE (m:Firma {nazwa:
'GoodHome'})

MERGE (p)-[:PRACUJE_W {rola:
'Admin'}]->(m) RETURN p,m



Typy właściwości

Typ	Przykład
Liczba całkowita (INTEGER)	wiek: 30
Liczba zmiennoprzecinkowa (FLOAT)	waga: 70.5
Łańcuchu znaków (STRING)	imie: 'Alice'
Wartość logiczna (BOOLEAN)	jest_studentem: false
Data i czas (DATE & TIME)	dzien_urodzenia: date('1992-06-15')
Tablica (ARRAY)	procenty: [85, 90, 95]
Typ geoprzestrzenny (GEOSPATIAL)	polozenie: point({ latitude: 40.73, longitude: -73.93 })

```
CREATE (:Osoba {imie: 'Alice', wiek: 30, jest_studentem: false, procenty: [85, 90, 95],  
dzien_urodzenia: date('1992-06-15'), polozenie: point({ latitude: 40.73, longitude: -73.93 })})
```

Operacje usuwania

Węzły (node)

```
MATCH (o:Osoba)
WHERE o.nazwisko = 'Nowak'
DELETE o
```

Węzły i relacje

```
MATCH (o:Osoba)
WHERE o.nazwisko = 'Nowak'
DETACH DELETE o
```

Relacje (relationship)

```
MATCH (o:Osoba {nazwisko: "Zieliński"})
-[r:PRACUJE_W ]
->(f:Firma {title: "GoodHome"})
DELETE r
```

Etykiety (label)

```
MATCH (p:Osoba {nazwisko: "Kowalski"})
REMOVE p:Osoba RETURN p
```

```
MATCH (p {nazwisko: 'Kowalski'})
DETACH DELETE p
```

Właściwości (property)

```
MATCH (p:Osoba {nazwisko: "Kowalski"})
REMOVE p.rola
RETURN p
```

Wszystko (all)

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

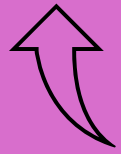
Operacje usuwania

Kryterium	DELETE	REMOVE
Co jest usuwane	Węzły, relacje, ścieżki	Właściwości węzłów, etykiety węzłów
Typ operacji	Destrukcyjna (usuwa elementy grafu z bazy)	Modyfikująca (zmienia istniejące elementy bez usuwania)
Efekt	Trwałe usunięcie danych z bazy	Usunięcie określonych informacji z węzła, węzeł pozostaje w bazie



"U mnie działa!"

Pierwsze prawo informatyka



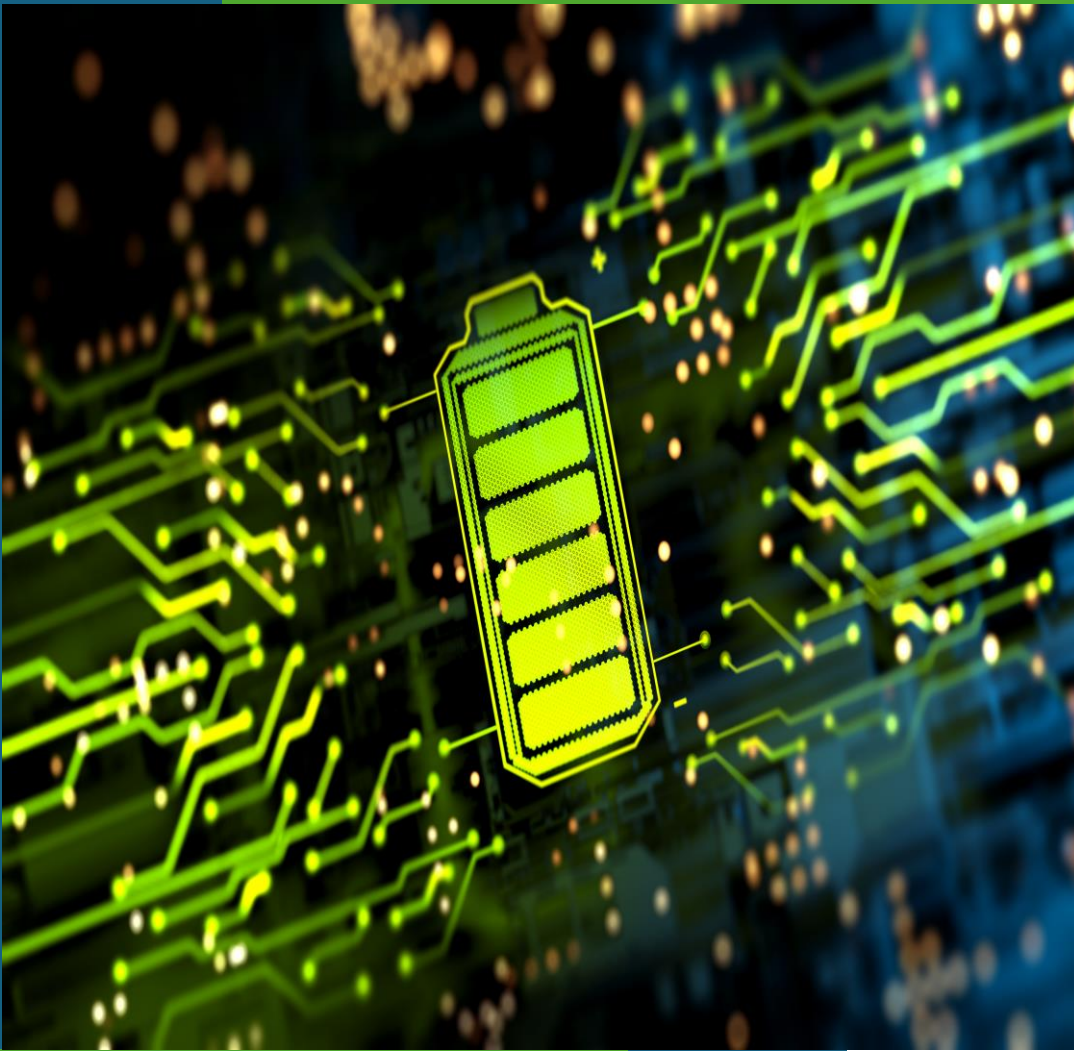
Temat 4 .

Podstawy modelowania danych grafowych

dr inż. Michał Malinowski

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

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



Zagadnienia (2h)

- Model grafu
- Modelowanie węzłów
- Modelowanie relacji
- Testowanie modelu danych grafu
- Refaktoryzacja (ulepszenie)
- Problem duplikatów danych
- Węzły pośrednie

Model grafu

Wprowadzenie

- Model grafu to sposób przechowywania danych w postaci węzłów i relacji. Węzły reprezentują poszczególne obiekty danych, takie jak osoby, produkty lub miejsca. Relacje reprezentują powiązania między obiektami, takie jak znajomości, transakcje lub relacje przestrzenne.
- Model grafu jest bardzo dobry do przedstawiania informacji o relacjach między obiektami.
- Model grafu jest bardzo wydajny do wykonywania zapytań, które wymagają analizy relacji między obiektami.
- Cele modelu:
 - integracja z docelową aplikacją (Java, Python, C#, PHP, Ruby, JavaScript, itd...)
 - wsparcie kluczowych przypadki użycia aplikacji (funkcjonalności)
 - zapewnienie wydajnych instrukcji Cypher dla kluczowych przypadków użycia.
- Komponenty Neo4j używane do definiowania modelu grafu to: węzły, relacje, etykiety, właściwości

Model grafu

Kroki tworzenia grafu

1. Zrozumienie wspieranego zagadnienia i zdefiniowanie konkretnych przypadków użycia (pytań) dla aplikacji.
2. Opracowanie wstępnego modelu grafu:
 - modelowanie węzłów
 - modelowanie relacji między węzłami
3. Przetestowanie przypadków użycia względem danych testowych i modelu grafu.
4. Utworzenie grafu na podstawie danych (zewnętrznych) przy użyciu Cypher.
5. Przetestowanie wydajności utworzonego grafu.
6. Refaktoryzacja (ulepszenie) modelu grafu ze względu na:
 - indeksowanie i ograniczenia
 - zmianę kluczowych przypadków użycia
 - lepszą wydajność

Modelowanie węzłów

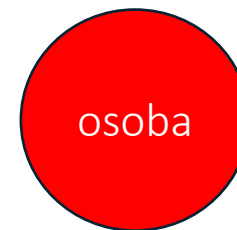
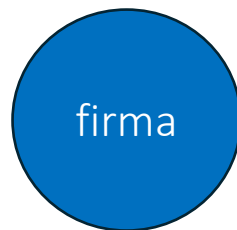
Węzły

Węzły przeważnie będą odpowiadały rzeczownikom z opisu rzeczywistości a w szczególności z przypadków użycia.

Węzły będą otrzymywały etykiety odpowiadające zidentyfikowanym rzeczownikom lub ich synonimom (mianownik w liczbie pojedynczej).

Przypadki użycia

1. W jakiej **firmie** pracuje ta **osoba**?



2. Jaką **rolę** pełni ta **osoba**?
3. Jakie **składniki** są używane w **przepisie**?
4. Kto jest **mężem** tej **osoby**?
5. Jakie **osoby** zagrały w **filmie**?
6. Kim są **dziadkowie** tego **dziecka**?

Modelowanie węzłów

Właściwości węzłów

Właściwości węzła są używane do:

- unikalnej identyfikacji węzła (klucz)
- odpowiadania na przypadki użycia
- zwracania danych

Klucz musi być unikalny w zbiorze danej etykiety !!!

```
MATCH (o:Osoba {id: 23456})  
RETURN o
```

```
MATCH (o:Osoba {nazwisko: "Kowalski"})  
RETURN o
```

```
MATCH (o:Osoba {nazwisko: "Zieliński"})  
-[r:PRACUJE_W ] ->(f:Firma)  
RETURN f
```

```
MATCH (o:Osoba {nazwisko: "Zieliński"})  
-[r:PRACUJE_W ] ->(f:Firma)  
RETURN f.nazwa, f.adres
```

~~(o:Osoba {id: 23456, nazwisko: "Kowalski"})~~

~~(o:Osoba {id: 23456, nazwisko: "Nowak"})~~

Modelowanie węzłów

Oprócz klucza {id}, który jest używany do jednoznacznej identyfikacji węzła, należy przeanalizować przypadki użycia, aby określić właściwości, które węzeł powinien przechowywać.

Podstawowe typy właściwości:
STRING, INTEGER, BOOLEAN,
FLOAT, DATE & TIME

Przypadki użycia:

1. W jakiej **firmie** pracuje ta **osoba**?
firma.nazwa, firma.adres
osoba.nazwisko, osoba.imie, osoba.wiek
2. Jaką **rolę** pełni ta **osoba**?
rola.nazwa, rola.stawka_godzina
osoba.nazwisko, osoba.imie
3. Jakie **składniki** są używane w **przepisie**?
skladnik.nazwa, skladnik.waga
przepis.nazwa, przepis.kraj

{nazwisko:'Nowak', imie:'Jan', wiek:39,
stawka_godzina:12.9, ma_dzieci:true}

Modelowanie relacji

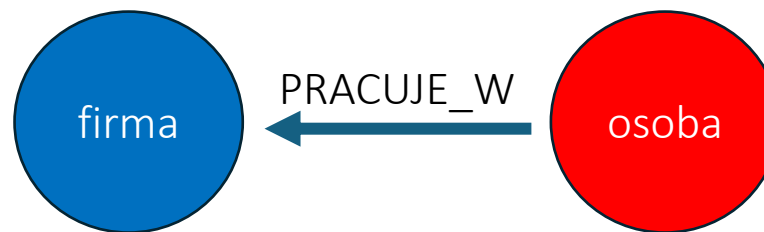
Relacje

Relacje przeważnie będą odpowiadały czasownikom z opisu rzeczywistości, a w szczególności z przypadków użycia.

Relacje będą otrzymywały etykiety odpowiadające zidentyfikowanym czasownikom lub ich synonimom (forma osobowa liczba pojedyncza).

Przypadki użycia

1. W jakiej **firmie** pracuje ta **osoba**?



2. Jaką **rolę** pełni ta **osoba**?

3. Jakie **składniki** są używane w **przepisie**?

4. Kto jest w związku małżeńskim z tą **osobą**?

5. Jakie **osoby** zagrały w **filmie**?

6. Kim są **dziadkowie** tego **dziecka**?

Modelowanie relacji

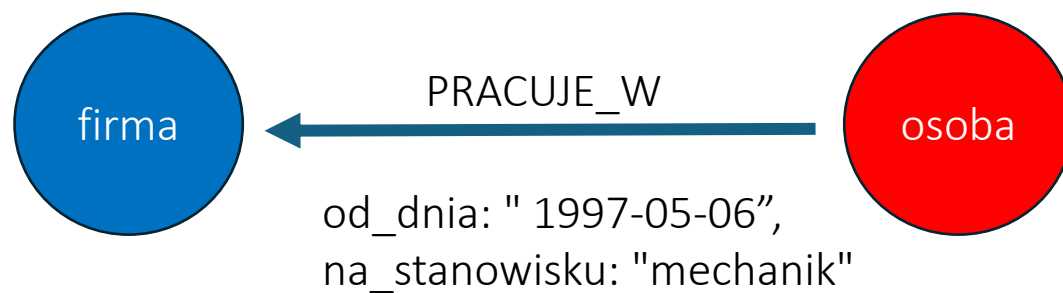
Właściwości relacji

Właściwości dla relacji służą do wzbogacenia sposobu powiązania dwóch węzłów i bardziej szczegółowego odpowiadania na pytania przypadków użycia .

Interpretujemy zgodnie z kierunkiem strzałki 😊

Przypadki użycia

1. W jakiej **firmie** pracuje ta **osoba**, od kiedy i na jakim stanowisku?

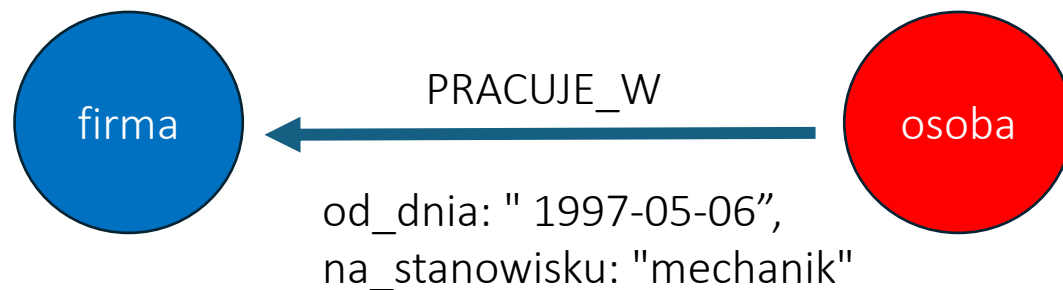


Testowanie modelu grafu

Testowanie polega na wykonaniu kodu Cypher względem modelu w celu sprawdzenia, czy graf i budowane zapytania obsługują zdefiniowane przypadki użycia.

Przypadki użycia:

1. W jakiej **firmie** pracuje ta **osoba**, od kiedy i na jakim stanowisku?



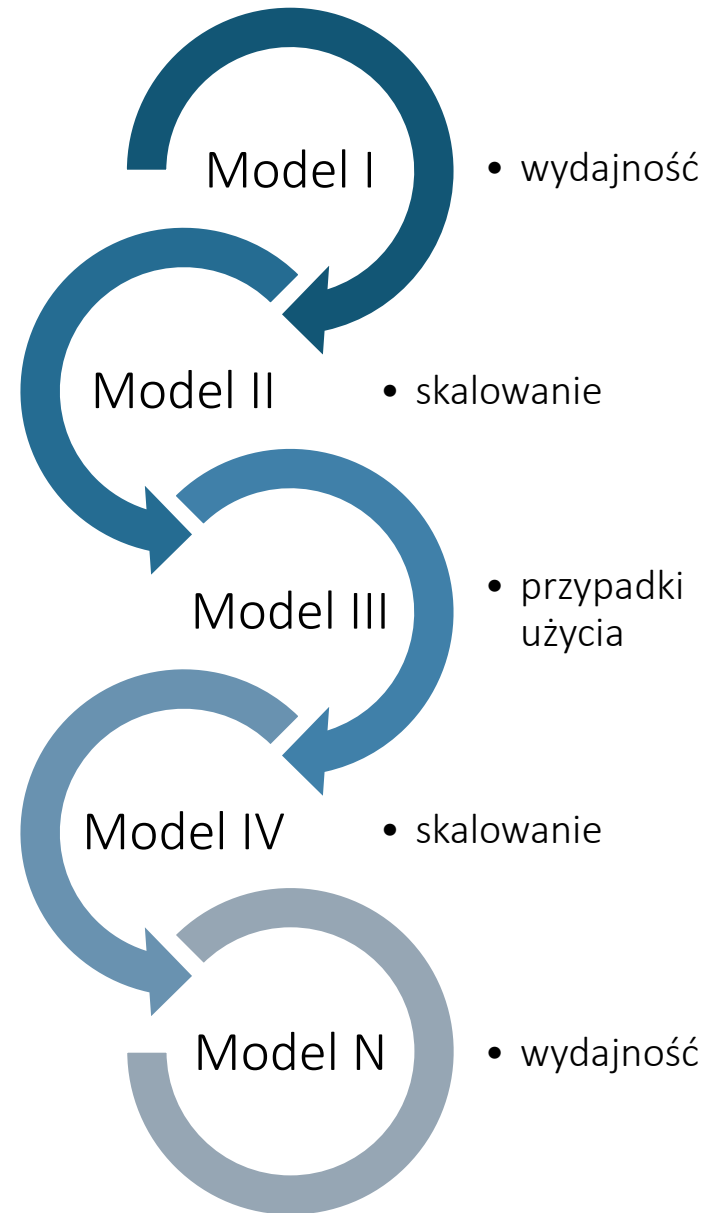
```
MATCH (o:Osoba {id:3})-[r:PRACUJE_W]->(f:Firma)
RETURN o.nazwisko, f.nazwa , r.od_dnia,
r.na_stanowisku
```

o.nazwisko	f.nazwa	r.od_dnia	r.na_stanowisku
"Grudniewski"	"Polexpo"	"1997-05-06"	"Mechanik"

Refaktoryzacja (ulepszenie)

Modelowanie danych grafu jest procesem iteracyjnym.

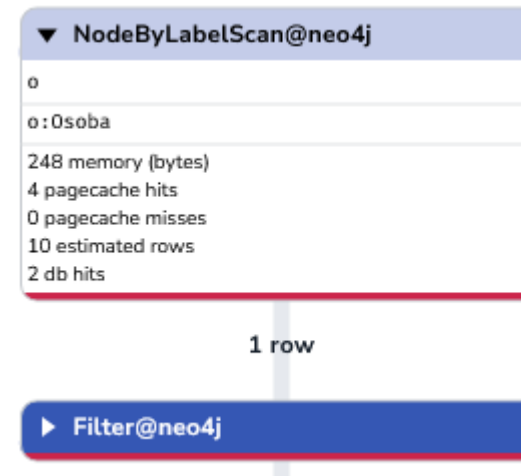
Początkowy model danych grafu jest punktem wyjścia, który jest wielokrotnie modyfikowany z powodu zmian przypadków użycia lub ich dookreślenia lub skalowania modelu lub w celu uzyskania lepszej wydajności.



Refaktoryzacja (ulepszenie)

PROFILE MATCH służy do śledzenia wykonywania zapytania w celu zidentyfikowania potencjalnych obszarów, w których zapytanie może zostać zoptymalizowane.

PROFILE MATCH (o:Osoba {id:3})-[r:PRACUJE_W]->(f:Firma) RETURN o.nazwisko, f.nazwa , r.od_dnia, r.na_stanowisku



Problem duplikatów danych

Duplikaty węzłów:

dwa węzły z tą samą etykietą i identycznymi właściwościami lub tym samym kluczem

```
CREATE (:Osoba {imie: 'Alicja', wiek: 30});  
CREATE (:Osoba {imie: 'Alicja', wiek: 30});
```

```
CREATE (:Osoba {imie: 'Alicja', id: 30});  
CREATE (:Osoba {imie: 'Adam', id: 30});
```

Duplikaty relacji:

dwie relacje z tą samą etykietą i właściwościami między tymi samymi węzłami.

```
MATCH (a:Osoba {id: 123}), (a:Osoba {id: 345})  
CREATE (a)-[:ZNA {od_roku: '2020'}]->(b);  
CREATE (a)-[:ZNA {od_roku: '2020'}]->(b);
```

Duplikaty właściwości:

dwie różne pary klucz-wartość mają tę samą wartość, ale różne klucze

```
CREATE (:Osoba {imie: 'Alicja', nazwa: 'Alicja', id: 30});
```

Problem duplikatów danych

Powody występowania duplikatów

- Błędy w danych
- Złe zaprojektowanie modelu danych
- Złe praktyki programistyczne
- Złe operacje dokonywane na danych

Na co wpływają duplikaty

- Błędy w wynikach zapytań
- Problemy z wydajnością
- Większe zużycie miejsca na dysku
- Zwiększone ryzyko bezpieczeństwa

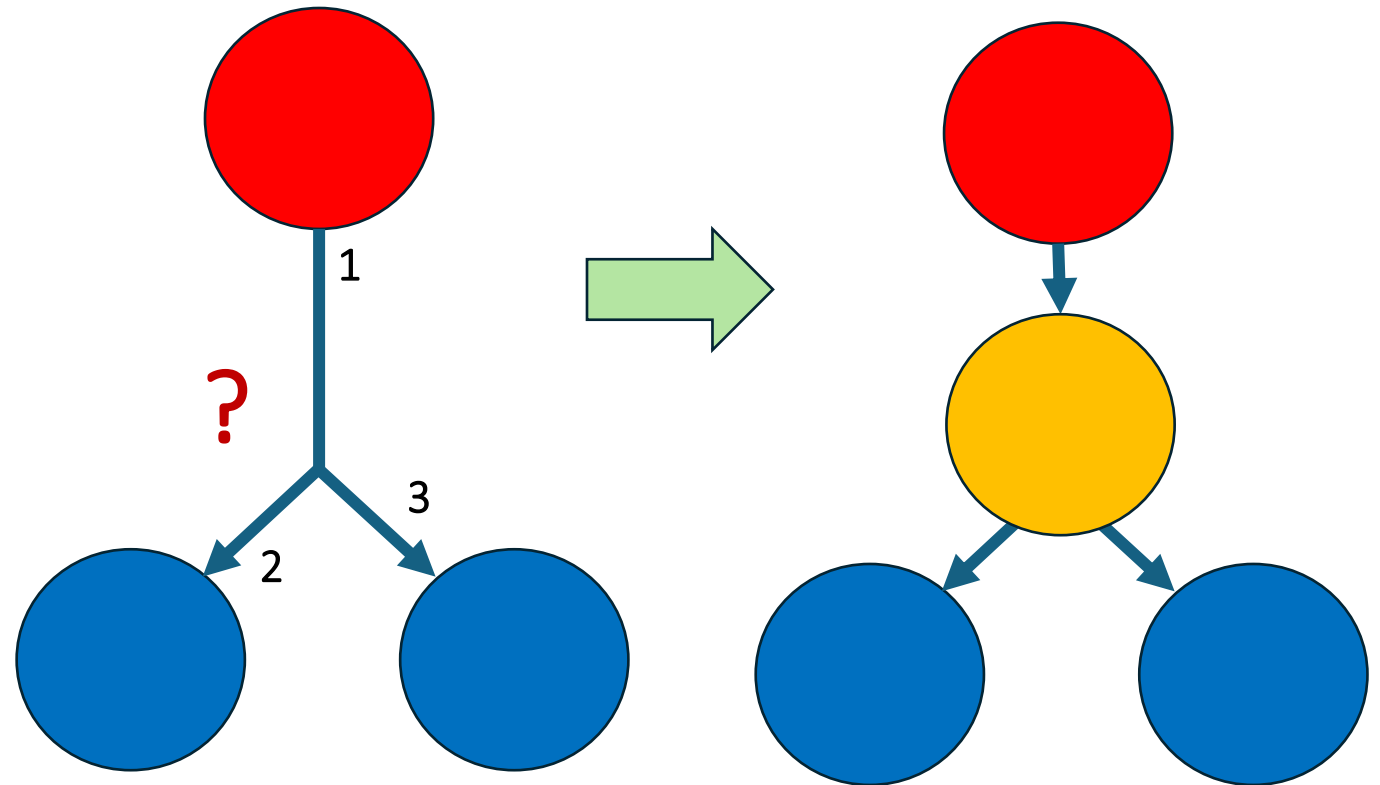
Uniknięcie duplikatów

- Dobrze zaprojektowany model danych
- Dokładna walidacja danych przed ich wprowadzeniem do bazy danych
- Dobre praktyki programistyczne
- Wykorzystanie MERGE zamiast CREATE (węzeł lub relacja zostanie utworzona tylko wtedy, gdy nie istnieje)

Węzły pośrednie

W przypadku, w którym konieczne jest podłączenie większej liczby niż 2 węzłów do relacji, stosuje się rozwiązanie w postaci węzłów pośrednich.

W matematyce odpowiada to Hipergrafom



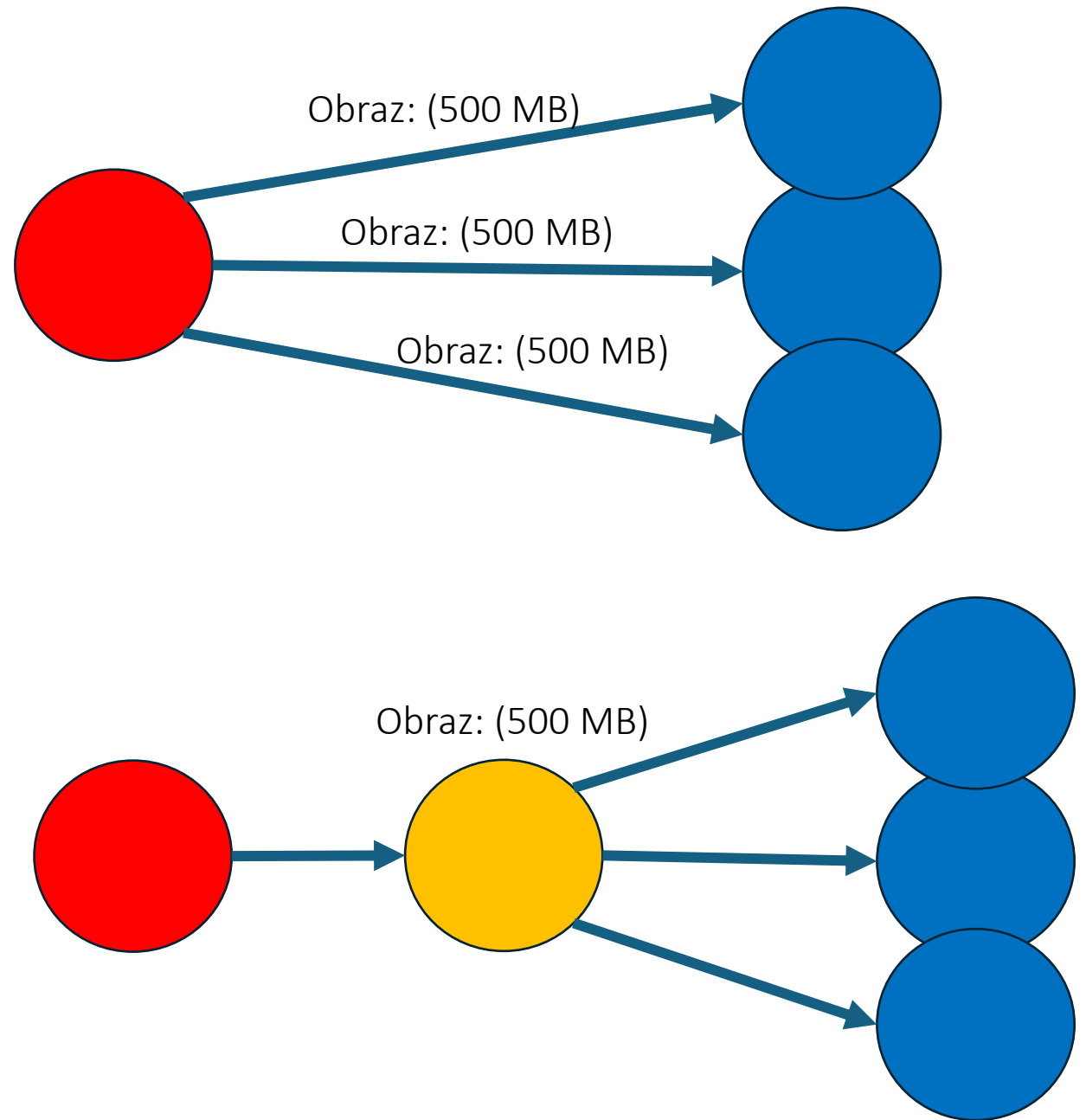
Węzły pośrednie

W przypadku, kiedy relacja współdzieli duże zbiory danych, dzięki czemu model może zawierać mniej zduplikowanych danych o dużej wielkości.

Na przykład relacja ma właściwość „obraz”, która przechowuje tą samą dużą grafikę

$3 \times 500\text{MB} > 500\text{MB}$

Dodanie węzła Obraz:



Homo PC

