



dr inż. Michał Malinowski  
[michal.malinowski@uth.edu.pl](mailto:michal.malinowski@uth.edu.pl)

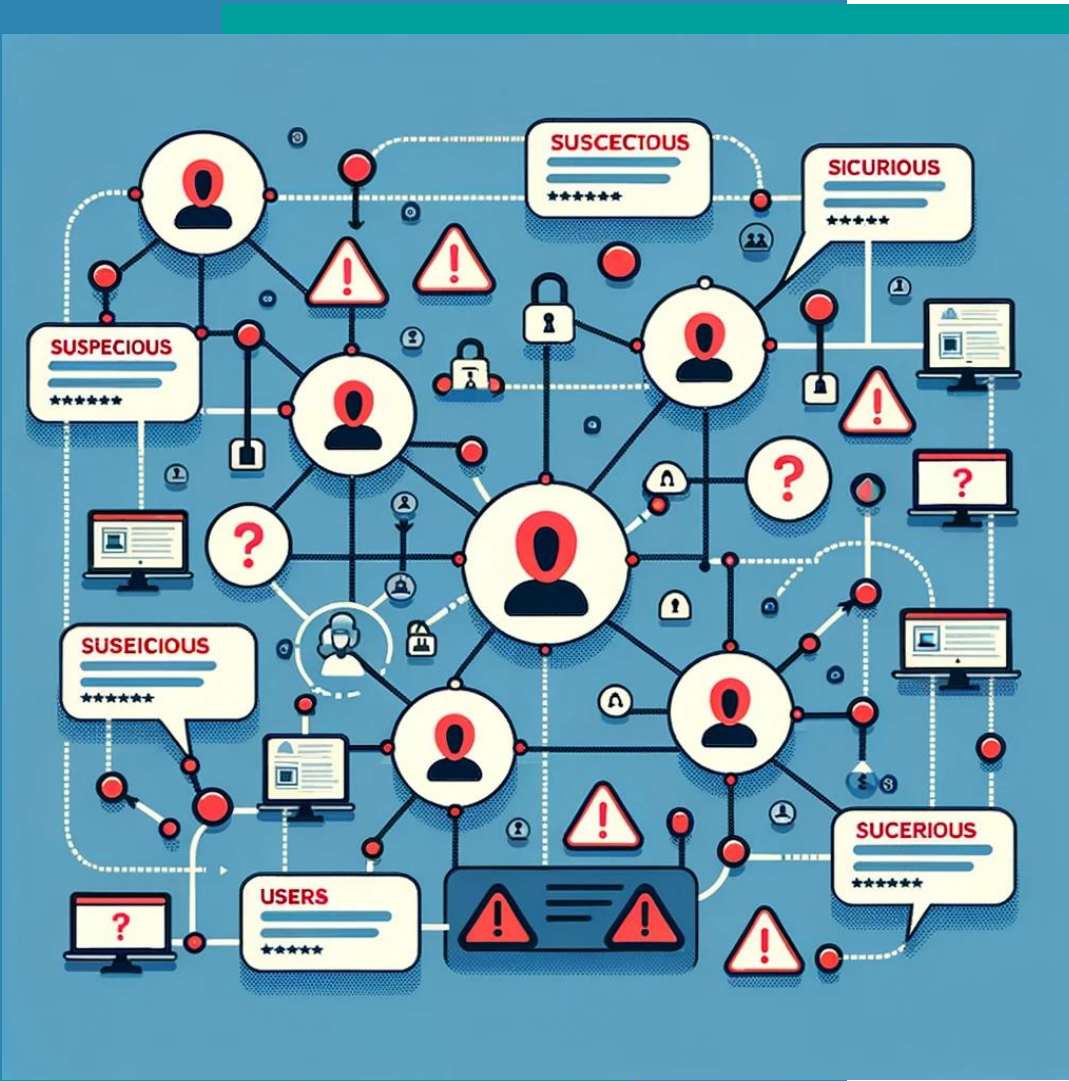


[https://www.linkedin.com/in/  
michał-malinowski-malkon/](https://www.linkedin.com/in/michał-malinowski-malkon/)

# Bazy grafowe

Ćwiczenie 09















ALGORYTMY  
I STRUKTURY DANYCH



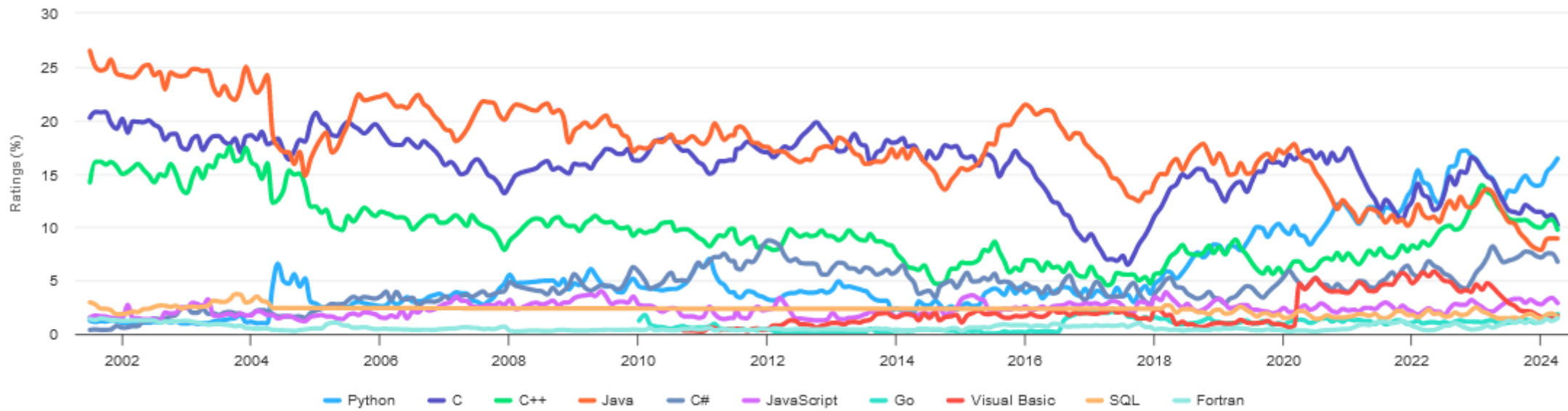
## Zagadnienia (2h)

- [Wprowadzenie do baz NoSQL](#)
- [Modelowanie węzłów i relacji](#)
- [Bazy grafowe vs. Bazy relacyjne](#)
- [Model grafu i dane grafu](#)
- [Baza grafowa Neo4j](#)
  - [Zadanie: Import danych do grafu](#)
- [Obszary zarządzania danymi](#)
- [Zapytania o dane CYPHER](#)
  - [Zadanie: System analizy danych](#)
- [Zadanie: Grafowy model danych](#)

# Ranking - Języki programowania

Apr 2024	Apr 2023	Change	Programming Language	Ratings	Change
1	1		 Python	16.41%	+1.90%
2	2		 C	10.21%	-4.20%
3	4	▲	 C++	9.76%	-3.20%
4	3	▼	 Java	8.94%	-4.29%
5	5		 C#	6.77%	-1.44%
6	7	▲	 JavaScript	2.89%	+0.79%
7	10	▲	 Go	1.85%	+0.57%
8	6	▼	 Visual Basic	1.70%	-2.70%
9	8	▼	 SQL	1.61%	-0.06%
10	20	▲	 Fortran	1.47%	+0.88%
11	11		 Delphi/Object Pascal	1.47%	+0.24%
12	12		 Assembly language	1.30%	+0.26%
13	18	▲	 Ruby	1.24%	+0.58%
14	17	▲	 Swift	1.23%	+0.51%
15	15		 Scratch	1.14%	+0.35%
16	14	▼	 MATLAB	1.11%	+0.25%
17	9	▼	 PHP	1.09%	-0.26%
18	38	▲	 Kotlin	1.05%	+0.80%
19	19		 Rust	1.03%	+0.41%
20	16	▼	 R	0.84%	+0.09%

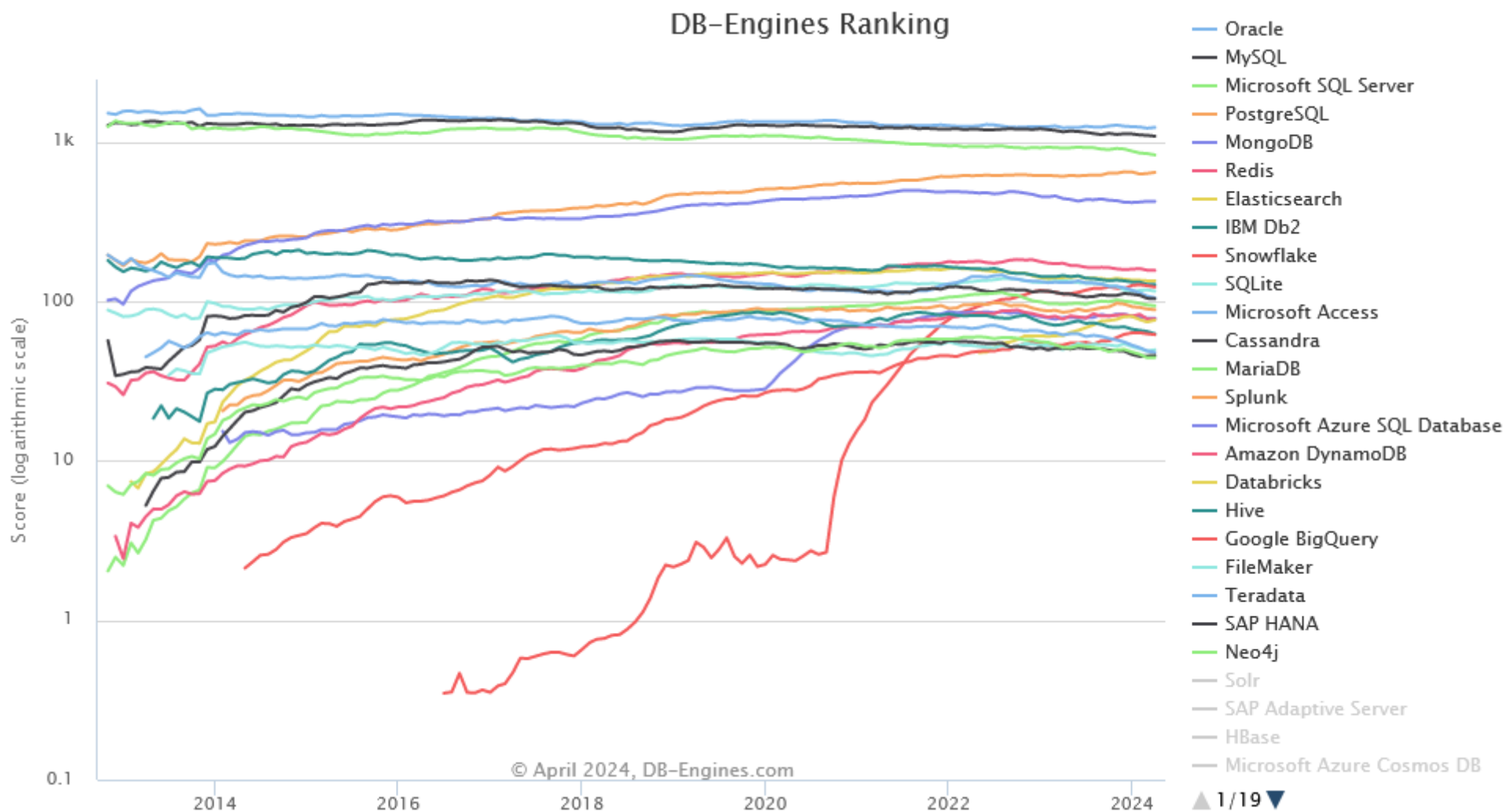
# Ranking - Języki programowania



# Ranking - Systemy baz danych

Rank			DBMS	Database Model	Score		
Apr 2024	Mar 2024	Apr 2023			Apr 2024	Mar 2024	Apr 2023
1.	1.	1.	Oracle <span>+</span>	Relational, Multi-model <span>i</span>	1234.27	+13.21	+5.99
2.	2.	2.	MySQL <span>+</span>	Relational, Multi-model <span>i</span>	1087.72	-13.77	-70.06
3.	3.	3.	Microsoft SQL Server <span>+</span>	Relational, Multi-model <span>i</span>	829.80	-16.01	-88.73
4.	4.	4.	PostgreSQL <span>+</span>	Relational, Multi-model <span>i</span>	645.05	+10.15	+36.64
5.	5.	5.	MongoDB <span>+</span>	Document, Multi-model <span>i</span>	423.96	-0.57	-17.93
6.	6.	6.	Redis <span>+</span>	Key-value, Multi-model <span>i</span>	156.44	-0.56	-17.11
7.	7.	<span>↑</span> 8.	Elasticsearch	Search engine, Multi-model <span>i</span>	134.78	-0.01	-6.29
8.	8.	<span>↓</span> 7.	IBM Db2	Relational, Multi-model <span>i</span>	127.49	-0.26	-18.00
9.	9.	<span>↑</span> 12.	Snowflake <span>+</span>	Relational	123.20	-2.18	+12.07
10.	10.	<span>↓</span> 9.	SQLite <span>+</span>	Relational	116.01	-2.15	-18.53
11.	11.	<span>↓</span> 10.	Microsoft Access	Relational	105.40	-2.52	-25.97
12.	12.	<span>↓</span> 11.	Cassandra <span>+</span>	Wide column, Multi-model <span>i</span>	103.86	-0.72	-7.94
13.	13.	13.	MariaDB <span>+</span>	Relational, Multi-model <span>i</span>	93.81	-1.22	-2.13
14.	14.	14.	Splunk	Search engine	88.71	-0.97	+3.27
15.	15.	15.	Microsoft Azure SQL Database	Relational, Multi-model <span>i</span>	78.40	-0.11	-0.66
16.	16.	16.	Amazon DynamoDB <span>+</span>	Multi-model <span>i</span>	77.57	-0.15	+0.12
17.	17.	<span>↑</span> 19.	Databricks <span>+</span>	Multi-model <span>i</span>	76.33	+1.99	+15.36
18.	18.	<span>↓</span> 17.	Hive	Relational	62.59	-2.24	-9.07
19.	19.	<span>↑</span> 20.	Google BigQuery <span>+</span>	Relational	61.90	-0.77	+8.59
20.	<span>↑</span> 21.	<span>↑</span> 23.	FileMaker	Relational	49.73	+0.92	-0.27
21.	<span>↓</span> 20.	<span>↓</span> 18.	Teradata	Relational, Multi-model <span>i</span>	47.84	-1.10	-13.75
22.	22.	22.	SAP HANA <span>+</span>	Relational, Multi-model <span>i</span>	45.84	+0.35	-5.24
23.	23.	<span>↓</span> 21.	Neo4j <span>+</span>	Graph	44.47	+0.11	-7.13
24.	24.	24.	Solr	Search engine, Multi-model <span>i</span>	44.28	+0.37	-3.94
25.	25.	25.	SAP Adaptive Server	Relational, Multi-model <span>i</span>	37.12	-0.37	-5.75

# Ranking - Systemy baz danych



# Wprowadzenie do baz NoSQL

## Definicja

- NoSQL (Not Only SQL) to termin odnoszący się do systemów zarządzania bazami danych (DBMS), które różnią się od klasycznych relacyjnych baz danych (RDBMS).
- Zaprojektowane z myślą o obsłudze dużej liczby danych, rozproszonych na wielu serwerach.
- Nie zawsze zapewniają pełne wsparcie dla standardów ACID\*

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

- Atomicity (Atomowość): Transakcje są niepodzielne.
- Consistency (Spójność): Każda transakcja przekształca bazę danych z jednego stanu spójnego do innego stanu spójnego.
- Isolation (Izolacja): Każda transakcja powinna być wykonywana w sposób izolowany od innych transakcji.
- Durability (Trwałość): Po zakończeniu transakcji, jej efekty są trwałe i niezawodne, nawet w przypadku awarii systemu.

# Wprowadzenie do baz NoSQL

## Rodzaje baz NoSQL

- Bazy klucz-wartość (Key-Value): najprostszy typ NoSQL, 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 kolumnowe (Columnar): dane są przechowywane i wyszukiwane na podstawie kolumn, a nie wierszy.  
Przykłady: Apache Cassandra, Google Bigtable.
- Bazy grafowe (Graph): skoncentrowane na przechowywaniu informacji o węzłach i relacjach między nimi.  
Przykład: Neo4j, JanusGraph

# Wprowadzenie do baz NoSQL

## Elementy bazy grafowej

### Węzły (Nodes)

Reprezentują obiekty takie jak osoby, miejsca, wydarzenia.

### Krawędzie (Edges, Relationships)

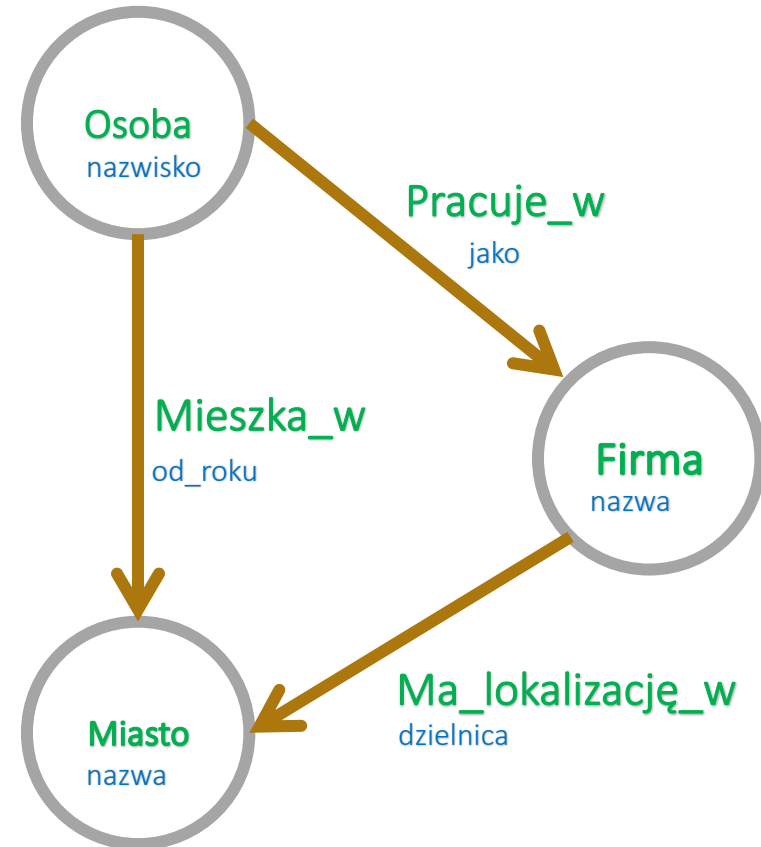
Reprezentują relacje między obiektami, np. znajomość, przepływ informacji.

### Etykiety (Labels)

Klasyfikują węzły i krawędzie, ułatwiając organizację i wyszukiwanie danych.

### Właściwości (Properties)

Informacje dodatkowe przypisane do węzłów i krawędzi, np. nazwa, data.



# Modelowanie węzłów i relacji

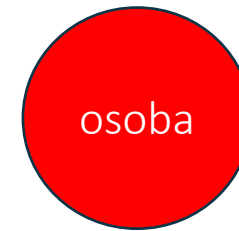
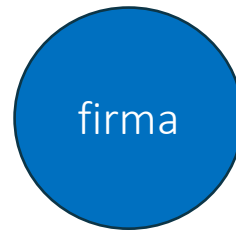
## Węzły

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 i 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**?

# 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

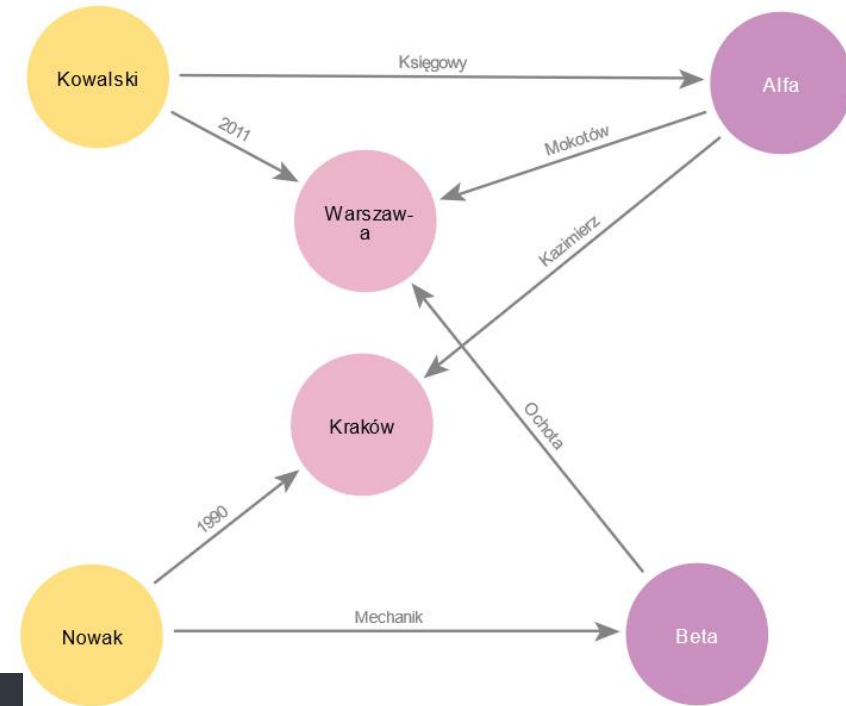
## Ma\_lokalizacje\_w

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"
```

Grafowa baza danych

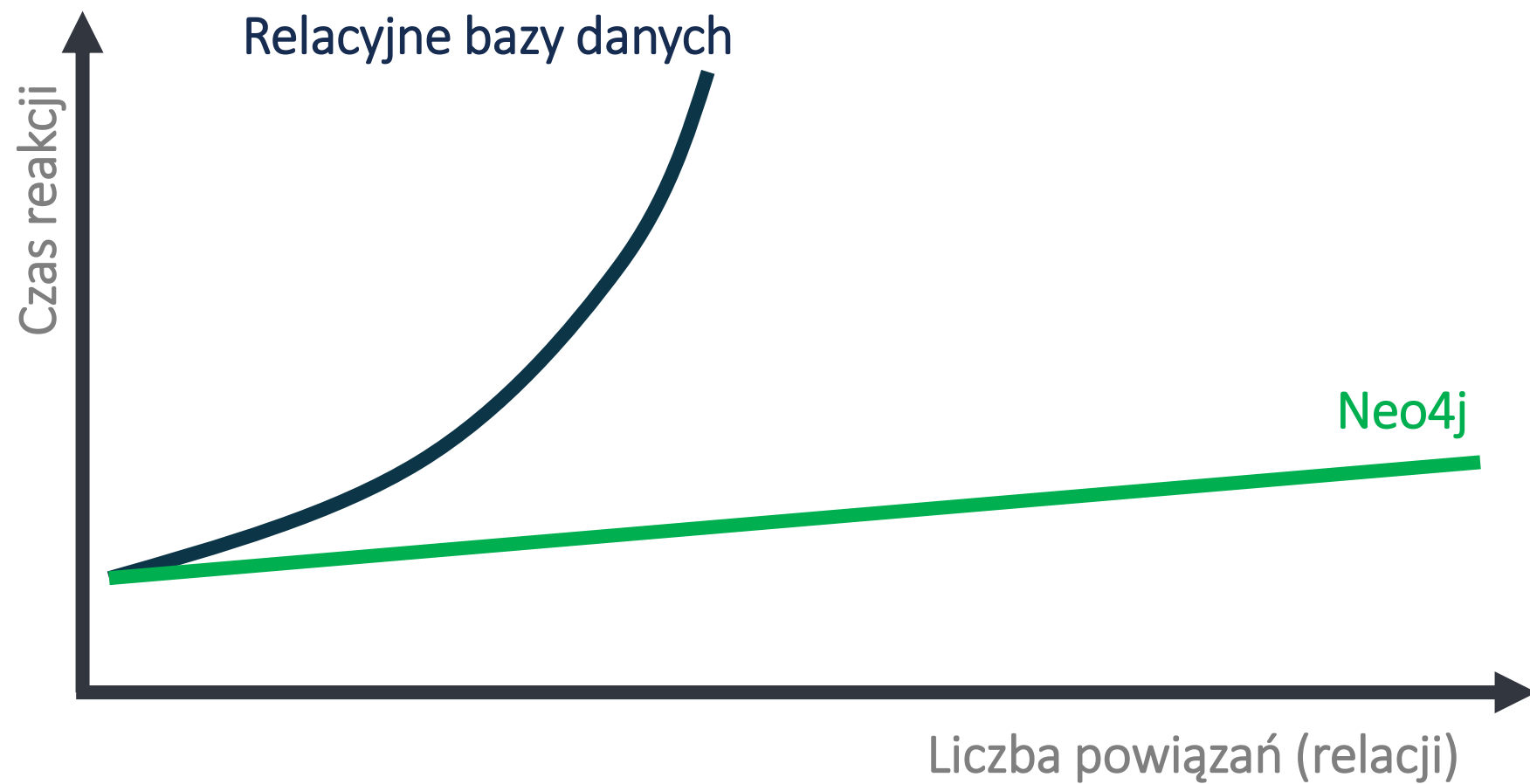


Cypher

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

```
nazwa      jako  
Alfa       Księgowy
```

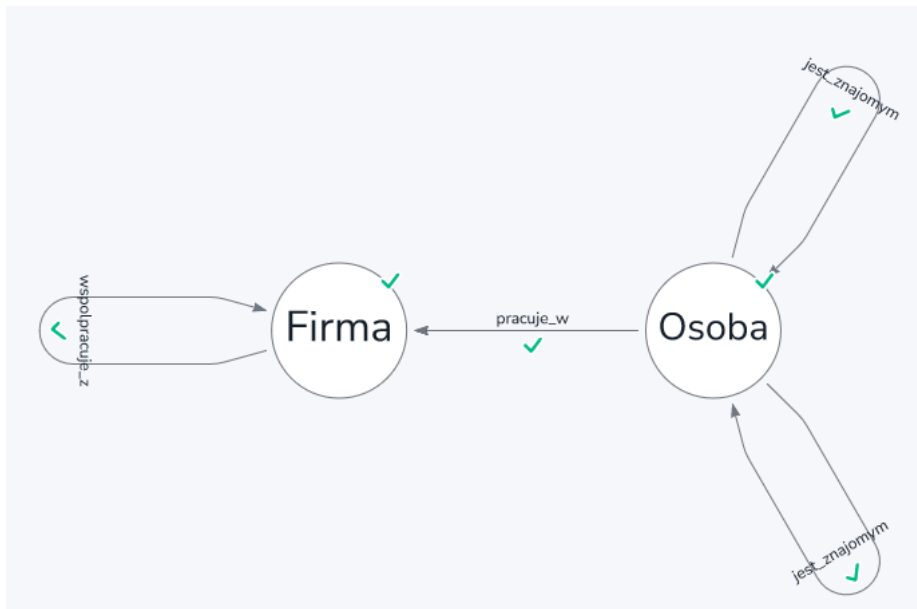
# Bazy grafowe vs. Bazy relacyjne



# Model grafu i dane grafu

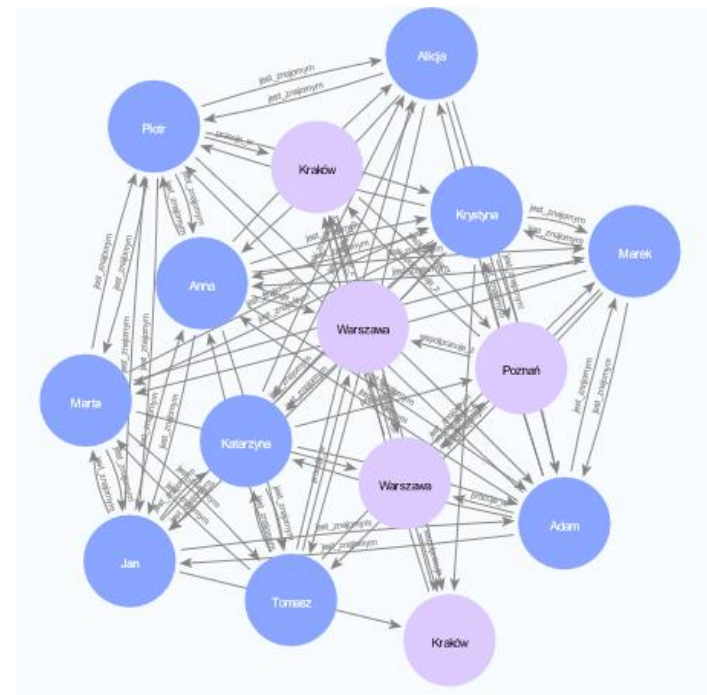
## Model grafu

Definiuje schemat przechowywania danych w grafowej bazie danych



## Dane grafu

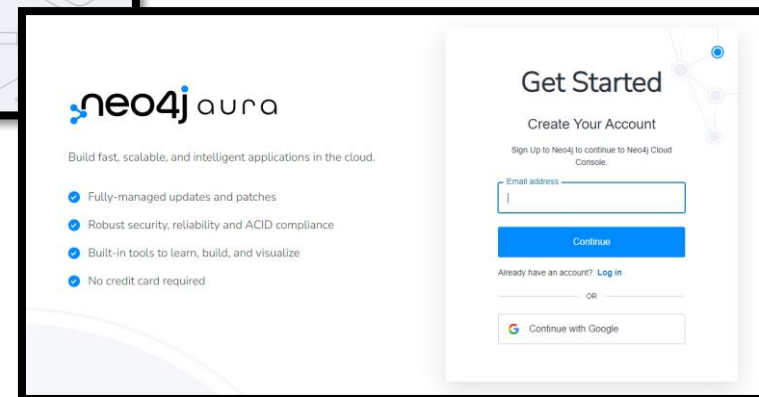
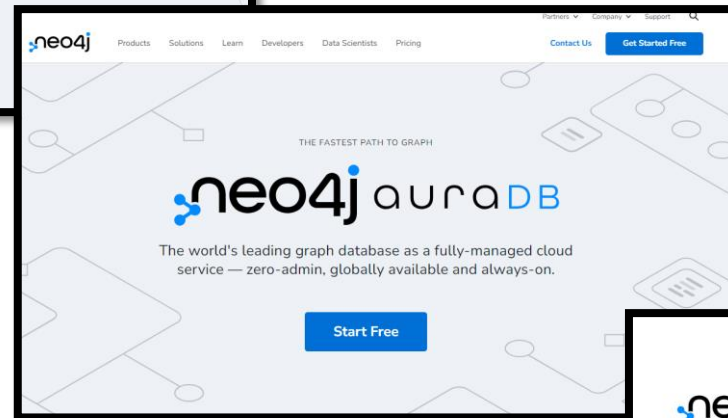
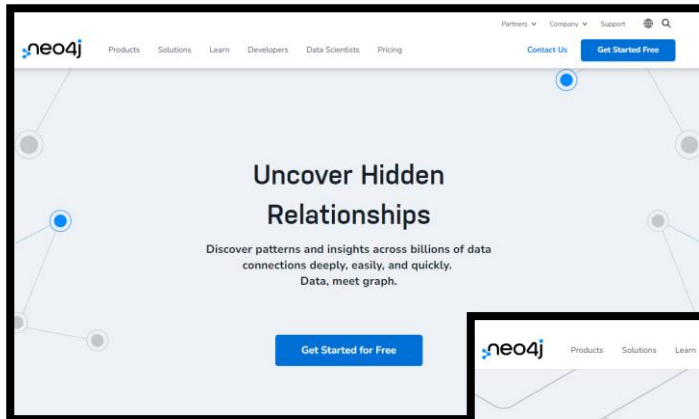
Rzeczywiste informacje, które są przechowywane i reprezentowane w bazie na podstawie modelu grafu



# Baza grafowa Neo4j

 **neo4j** auraDB

<https://neo4j.com>



# Baza grafowa Neo4j

The screenshot shows the 'Create an AuraDB instance' page. On the left, there is a navigation menu with 'Instances' selected. The main content area is titled 'Create an AuraDB instance' and 'Instance type:'. It offers two options: 'Free' and 'Professional'. The 'Free' option is selected and highlighted with a blue border. Below the options, there is a table comparing the two instance types across various metrics.

Metric	Free	Professional
Memory	1 GB / instance	up to 64 GB / instance
Graph size	200K nodes, 400K relationships	Unlimited
Graph tools	●	●
Snapshots	One on-demand only	Daily scheduled and on-demand
Pause	Automatic after 3 days of inactivity	On-demand
Resume	On-demand	On-demand
Clone	—	●
API	—	●
Cloud provider options	GCP	GCP, AWS*, Azure*

The screenshot shows the 'Credentials for Instance01' page. It displays the username 'neo4j' and a generated password 'xJKoW6mSdK2gnttYPCgMGc0s8fM1NPU'. There is a 'Download' button next to the password. A message states: 'We strongly advise changing this initial password.' Below this, there is a checkbox that is checked, with the text: 'I confirm I have copied or downloaded the above credentials, as this password will not be available after this point.' A 'Continue' button is at the bottom right.

The screenshot shows the 'Instances' page. On the left, there is a navigation menu with 'Instances' selected. The main content area is titled 'Instances' and has a 'New Instance' button. Below this, there is a card for 'Instance01' which is 'Free' and 'Running'. The card displays the following information: Neo4j version 5, Nodes 0 / 200000 (0%), Relationships 0 / 400000 (0%), Region Belgium (europe-west1), and Connection URI neo4j+s://c99c897b.databases.neo4j.io. There are icons for deleting and opening the instance.

The screenshot shows the 'Connect to instance' page. It displays the 'neo4j' logo and the title 'Connect to instance'. Below this, there is a form with the following fields: Scheme (neo4j+s), Connection URL (c99c897b.databases.neo4j.io:7687), Connect with SSO (disabled), Database user (neo4j), and Password (masked with dots). A message states: 'Username and password are stored in the file that you downloaded when you created this instance.' There are 'Connect' and 'Cancel' buttons at the bottom.

# Zadanie: Import danych do grafu

## Zasady

- Należy przygotować pliki CSV, które reprezentują węzły i relacje grafu, gdzie 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
- 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 nazywać od etykiet węzłów, które będą łączyć
  - Relacje [:osoba -> :firma] plik „osoba\_firma.csv”
- Wiersze mają zaczynać się od unikalnego klucza (id)

# Zadanie: Import danych do grafu

## Dane

### 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
```

# Zadanie: Import danych do grafu

## Import danych do grafu

Zaimportuj z wykorzystaniem „importera” przygotowane pliki CSV do bazy grafowej Neo4j.

# Obszary zarządzania danymi

The screenshot displays the Neo4j Import interface with several key components:

- Files Panel (4):** Lists three CSV files: `firma.csv`, `osoba.csv`, and `osoba_firma.csv`. Each file has a list of columns and their corresponding values.
- Relationship Mapping (3):** A diagram showing a relationship named `pracuje_w` between `Firma` and `Osoba` nodes. The relationship is bidirectional, indicated by arrows pointing in both directions.
- Node ID Mapping (6):** A table defining how node IDs are mapped from the source files to the target nodes.

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

**Relationship type (5):** Name: `pracuje_w`. Filter file: `osoba_firma.csv`.

**Properties (6):** Map from file (+)

Name	Type	Column
<code>idosoba</code>	inte...	idosoba
<code>idfirma</code>	inte...	idfirma

# Obszary zarządzania danymi

The screenshot displays the Neo4j Explore interface. At the top, there are tabs for 'neo4j', 'Explore', 'Query', and 'Import'. The main area shows a graph with nodes and relationships. A search bar is located at the top left, and a toolbar with various icons is on the left. The graph consists of several nodes: a central purple node (labeled '1'), two yellow nodes (labeled '4'), and two more yellow nodes (labeled '2'). Relationships are labeled 'pracuje\_w'. A right-hand sidebar shows a 'Nodes' tab with a 'Filter categories' dropdown (labeled '3') and radio buttons for 'All', 'In Scene', and 'Off Scene'. Below this, there are two node categories: 'Firma' (3) and 'Osoba' (4). At the bottom left, there is a status bar showing 'All (11)' and 'Selected (1)'. At the bottom center, there is a 'Force-based layout' button (labeled '1') and a zoom level of '185%'. A download icon (labeled '4') is visible in the top right corner of the graph area.

# Obszary zarządzania danymi

The screenshot displays the Neo4j Query interface. On the left, the 'Database Information' panel shows 7 nodes (Firma, Osoba) and 4 relationships (pracuje\_w). The main query editor contains the Cypher query: `$ MATCH p=()-[:pracuje_w]->() RETURN p LIMIT 25;`. The results are shown as a graph with nodes like 'Krakow', 'Krystyn...', 'Alicja', 'Marek', 'Warsza...', and 'Adam' connected by relationships. The interface includes a navigation bar with 'Explore', 'Query', and 'Import' tabs, and a status bar at the bottom indicating 'Started streaming 4 records after 71ms and completed after 75ms.'

Database Information

Nodes (7)

- Firma
- Osoba

Relationships (4)

- pracuje\_w

Property keys

- data
- dzien\_urodzenia
- id
- idfirma
- idosoba
- imie
- miescowosc
- name
- nazwa
- nazwisko
- nodes
- relationships
- stanowisko
- student
- style

neo4j\$

```
$ MATCH p=()-[:pracuje_w]->() RETURN p LIMIT 25;
```

Graph Table RAW

Started streaming 4 records after 71ms and completed after 75ms.

# Zapytania o dane CYPHER

Zapytanie o wszystkich studentów:

```
MATCH (o:Osoba)
WHERE o.student = true
RETURN o.imie, o.nazwisko, o.dzien_urodzenia
```

Zapytanie o wszystkie firmy z  
Warszawy:

```
MATCH (f:Firma)
WHERE f.miejscowosc = 'Warszawa'
RETURN f.nazwa
```

Zapytanie o osoby pracujące w  
firmie "GoodHome":

```
MATCH (o:Osoba)-[r:pracuje_w]->(f:Firma)
WHERE f.nazwa = 'GoodHome'
RETURN o.imie, o.nazwisko, r.stanowisko
```

# Zapytania o dane CYPHER

Zapytanie o osoby, które nie są studentami i pracują w Warszawie:

```
MATCH (o:Osoba)-[r:pracuje_w]->(f:Firma)
WHERE o.student = false AND f.miejscowosc = 'Warszawa'
RETURN o.imie, o.nazwisko, f.nazwa, r.stanowisko
```

Zapytanie o firmy, w których pracują osoby urodzone po 1995 roku:

```
MATCH (o:Osoba)-[r:pracuje_w]->(f:Firma)
WHERE o.dzien_urodzenia >= '1995-01-01'
RETURN f.nazwa, COLLECT(o.imie + ' ' + o.nazwisko) AS pracownicy
```

Zapytanie o liczbę osób pracujących w każdej firmie:

```
MATCH (o:Osoba)-[r:pracuje_w]->(f:Firma)
RETURN f.nazwa, COUNT(o) AS liczba_pracownikow
```

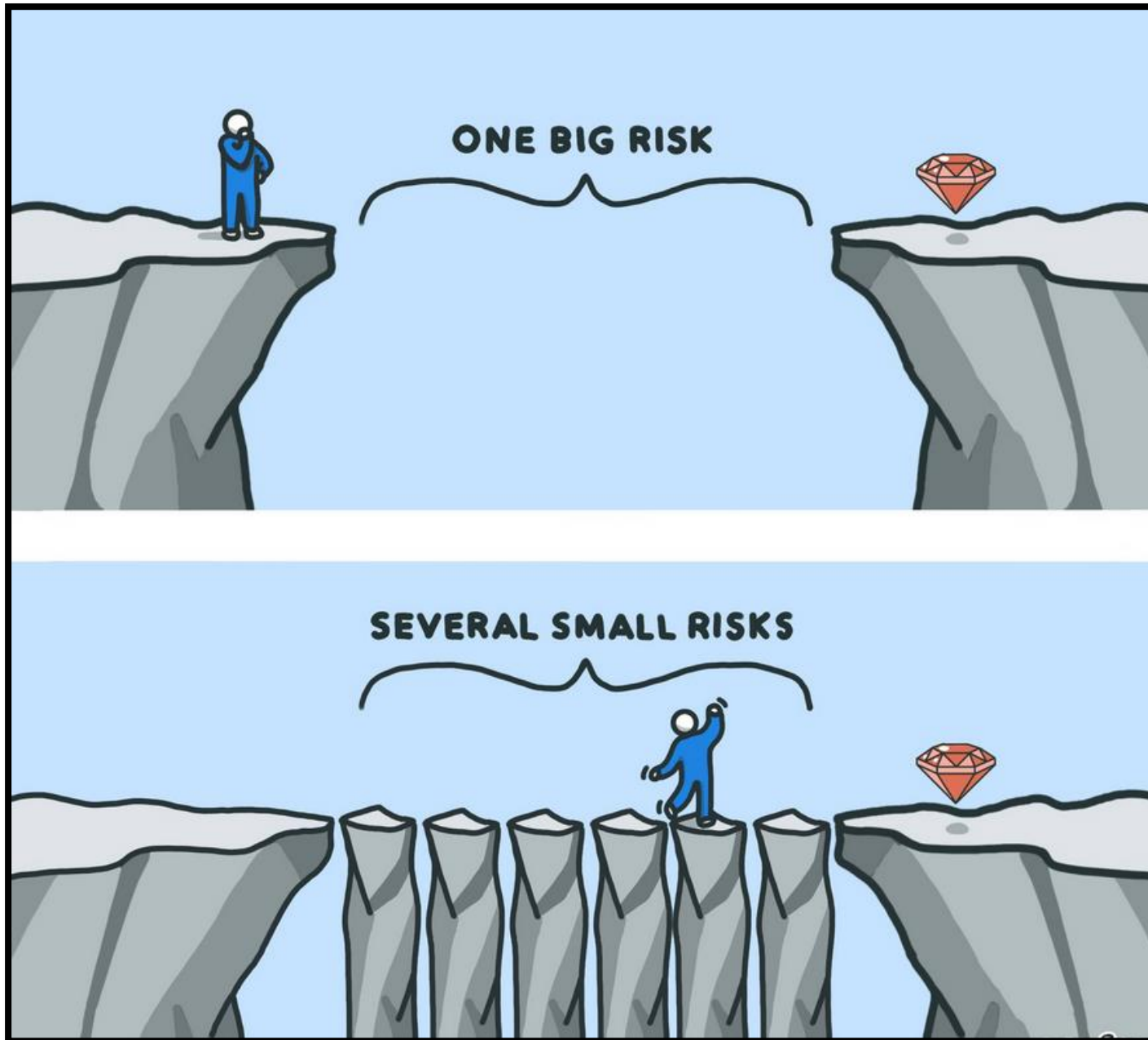
# Zapytania o dane CYPHER

The screenshot displays the Neo4j web interface. On the left, the 'Saved Cypher' sidebar is visible, featuring a list of saved queries with checkboxes for actions like 'Delete ALL', 'Delete Parametry', 'Load', and 'Osoba'. A blue circle with the number '1' highlights the bookmark icon, and a blue circle with the number '2' highlights the 'Osoba' query. A blue circle with the number '3' highlights the 'Import' tab in the top navigation bar. The main area shows a Cypher query: `neo4j$ PROFILE MATCH p =allShortestPaths((start:Osoba`. Below the query, the 'Graph' view is selected, showing a network of nodes (Marta, Piotr, Krystyna, Anna, Jan, Tomasz, Marek) connected by edges labeled 'jest\_znajomym'. A blue circle highlights the 'Marek' node. At the bottom, a status message reads: 'Started streaming 20 records after 2ms and completed after 7ms.' and a warning icon indicates: 'The provided pattern is unbounded, consider adding an upper limit to the number of node hops.'

# Zadanie: System analizy danych

## System analizy danych

Zbuduj menu w bazie Neo4j bazując na przykładach zapytań CYPHER dla Systemu analizy danych





# Zadanie: Grafowy model danych

## Zadanie (2 pkt)

Zamodeluj i wykonaj graf w bazie grafowej Neo4j reprezentujący .

Węzły (Nodes):

Produkty: 10 elementów

nazwa: Nazwa produktu, np. "Smartfon"

sku: Unikalny kod produktu (SKU - Stock Keeping Unit)

Magazyny: 3 elementy

lokalizacja: Lokalizacja magazynu, np. "Miasto A"

pojemnosc: Maksymalna pojemność magazynu

Krawędzie (Edges):

Dostarczanie (DOSTARCZONO\_DO): 20 elementów

data\_dostarczenia: Data dostarczenia produktów do magazynu

ilosc\_dostarczona: Ilość dostarczonych produktów



## Przedstawienie wyniku

- Obrazek grafu w PDF
- Umieszczenie wyniku w Moodle

## Termin

- Podany przy zadaniu w Moodle (około 2 tygodnie)