

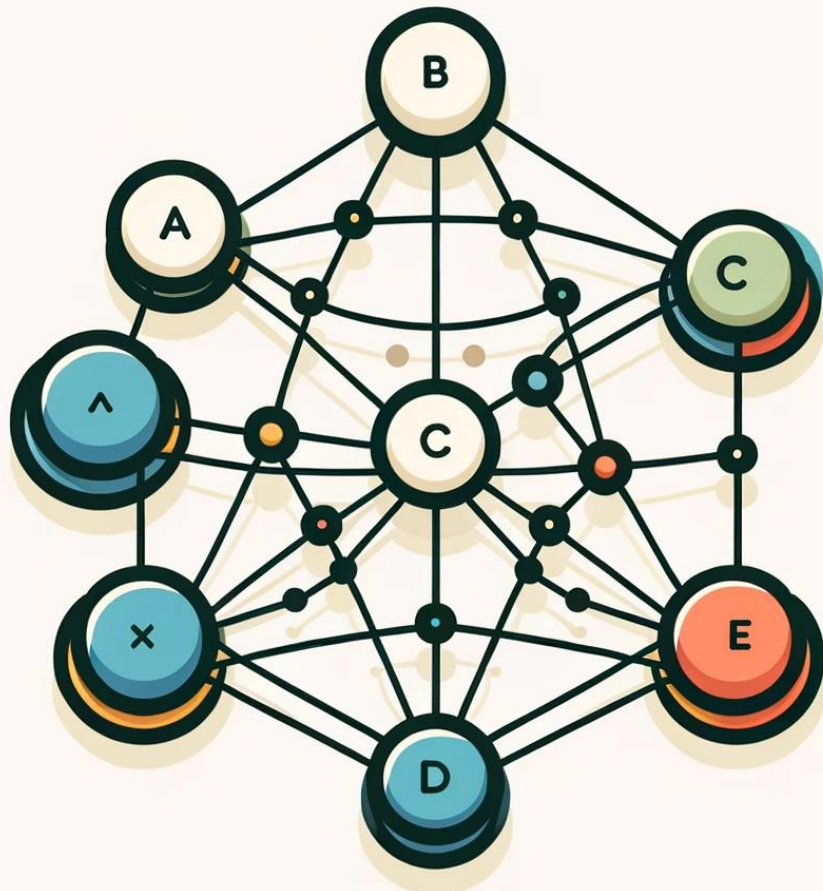
Temat 5 .

dr inż. Michał Malinowski

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

Bazy NoSQL i nie tylko

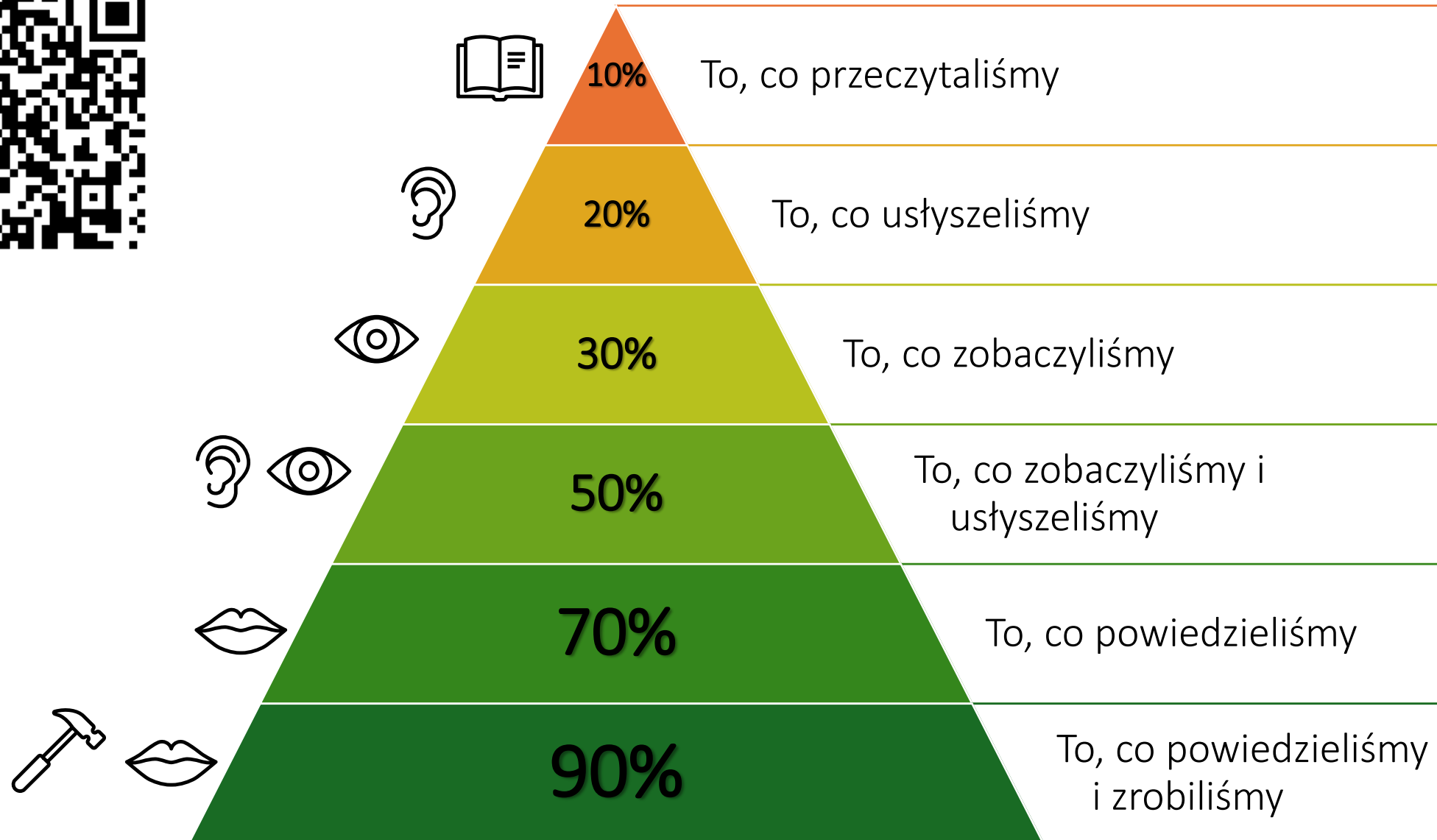
PODSTAWY BAZ DANYCH



Zagadnienia:

- Typy systemów baz danych
- Bazy NoSQL
- Bazy klucz-wartość
- Bazy wektorowe
- Bazy grafowe
- Cypher
- Algorytmy grafowe

Piramida zapamiętywania



Typy systemów baz danych

Typy

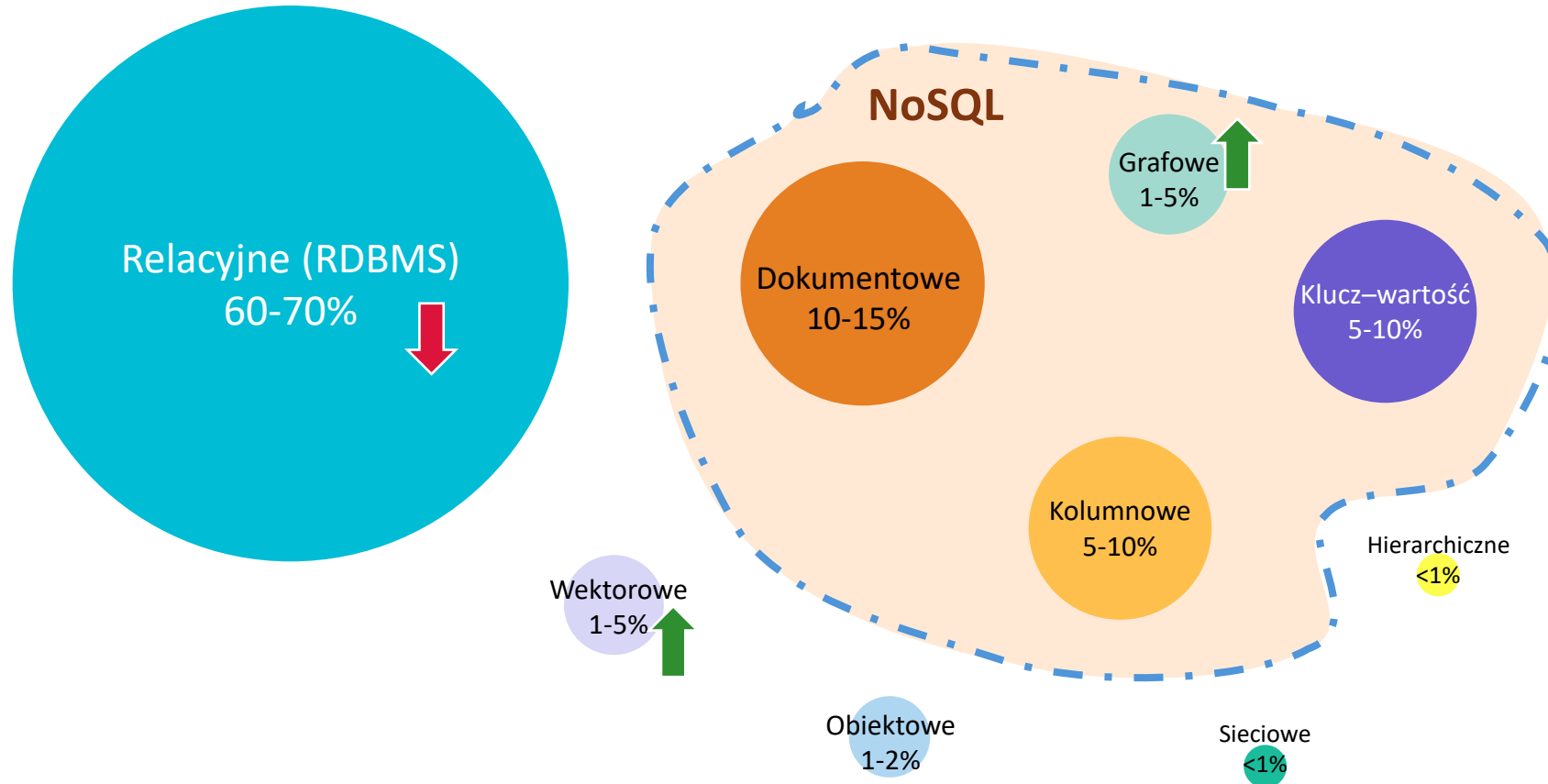
Typ : Podtyp	Opis i główne cechy	Najczęstsze zastosowania
Relacyjne (RDBMS)	- Dane przechowywane w tabelach (wiersze i kolumny). - Schematy z góry zdefiniowane (sztywne modele). - Wsparcie dla ACID (Atomicity, Consistency, Isolation, Durability). - Posługują się językiem SQL.	- Systemy transakcyjne (bankowość, księgowość). - Aplikacje biznesowe (ERP, CRM). - Analizy i raportowanie w oparciu o modele relacyjne (OLAP).
NoSQL: Dokumentowe	- Dane w formacie dokumentów (np. JSON, BSON, XML). - Brak sztywnego schematu, możliwość przechowywania obiektów o zróżnicowanej strukturze. - Dedykowane języki zapytań (np. MQL w MongoDB) lub zbliżone do SQL.	- Aplikacje webowe i mobilne (dynamiczne struktury danych). - Systemy CMS. - Logi i dane pół-strukturyzowane.
NoSQL: Klucz–wartość	- Dane w postaci par klucz → wartość. - Najprostszy model w rodzinie NoSQL. - Przeznaczone do szybkiego odczytu/zapisu w rozproszonych środowiskach.	- Przechowywanie sesji i cache. - Tymczasowe przechowywanie konfiguracji. - Proste dane klucz–definicja (np. słowniki, parametry).
NoSQL: Kolumnowe	- Dane przechowywane kolumnowo, co pozwala na wydajną kompresję i szybkie agregacje. - Przeznaczone do ogromnych zbiorów danych (Big Data). - Zazwyczaj działają w klastrach rozproszonych.	- Hurtownie danych. - Analiza wielkich zbiorów (Big Data). - Szybkie operacje agregujące (SUM, COUNT, AVG) na setkach milionów rekordów.
NoSQL: Grafowe	- Dane w postaci węzłów (nodes) i krawędzi (edges) z atrybutami. - Świetnie odwzorowują relacje i powiązania między obiektami. - Dedykowane zapytania grafowe (Cypher w Neo4j, Gremlin).	- Analiza sieci społecznościowych. - Systemy rekomendacyjne (powiązania, zależności). - Modelowanie i analiza złożonych relacji (np. w logistyce, sieciach transportu).

Typy systemów baz danych

Typy

Typ : Podtyp	Opis i główne cechy	Najczęstsze zastosowania
Obiektowe	- Dane przechowywane jako obiekty (z atrybutami i metodami), zgodnie z paradygmatem OOP. - Eliminacja tzw. „impedance mismatch” między kodem obiektywowym a bazą. - Integracja z językami obiektowymi (Java, C++, C#).	- Systemy, w których bezpośrednio odwzorowuje się strukturę obiektową. - Aplikacje wymagające elastycznego przechowywania złożonych klas (zagnieżdżenia obiektów).
Wektorowe	- Specjalne bazy do przechowywania i wyszukiwania wektorów (embeddings). - Kluczowe w zastosowaniach AI, ML i wyszukiwaniu semantycznym. - Oferują m.in. nearest neighbor search w wysokowymiarowych przestrzeniach.	- Systemy rekomendacyjne. - Wyszukiwanie semantyczne (teksty, obrazy, dźwięki). - Analiza podobieństwa w przestrzeniach wielowymiarowych (klasyfikacja, klastrowanie).
Hierarchiczne	- Jeden z najstarszych modeli; dane w formie drzewa (hierarchii). - Bardziej sztywna struktura niż modele relacyjne czy NoSQL. - Głównie w starszych systemach mainframe.	- Starsze systemy bankowe, ubezpieczeniowe (mainframe). - Aplikacje korzystające z ustalonych, niezmiennych struktur danych.
Sieciowe	- Uogólnienie modelu hierarchicznego, gdzie każdy węzeł może łączyć się z wieloma innymi. - Prototyp baz grafowych, ale mniej elastyczny i oparty na starszych standardach (CODASYL). - Spotykany w starszych systemach mainframe.	- Dziedzictwo starszych aplikacji finansowych, kadrowych. - Historyczne systemy w dużych przedsiębiorstwach (często nadal utrzymywane z powodów biznesowych).

Typy systemów baz danych



Bazy 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.

Bazy 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, dostępności i odporności na podział sieci.

Wsparcie dla różnych typów danych

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

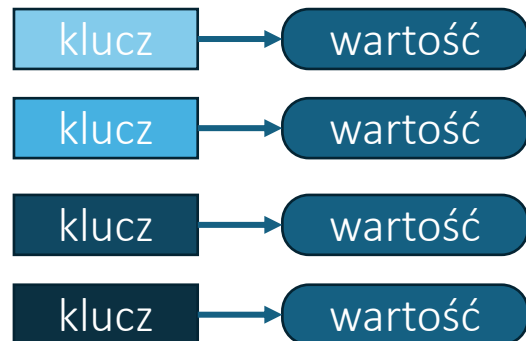
Bazy NoSQL

Bazy klucz-wartość

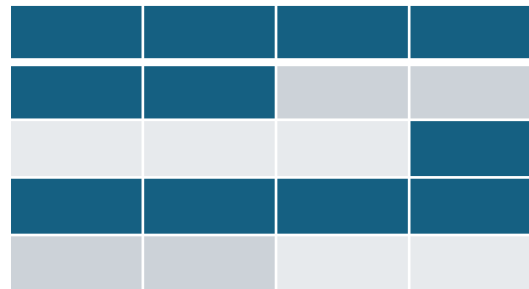
(Key-Value): gdzie wszystko przechowywane jest jako para klucz-wartość.

Przykłady: *Redis, Amazon DynamoDB*

Bazy klucz-wartość



Bazy kolumnowe



Bazy kolumnowe

(Columns): Organizują dane w kolumnach zamiast w wierszach, co umożliwia efektywne przechowywanie i przetwarzanie dużych zbiorów danych analitycznych.

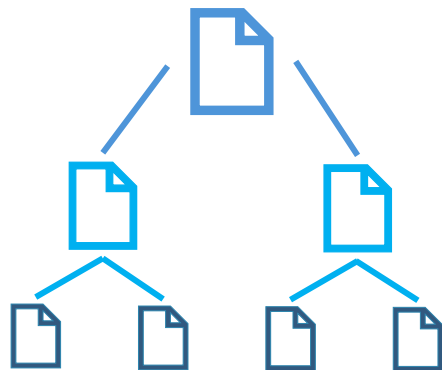
Przykłady: *Cassandra, HBase, Google Bigtable*.

Bazy dokumentowe

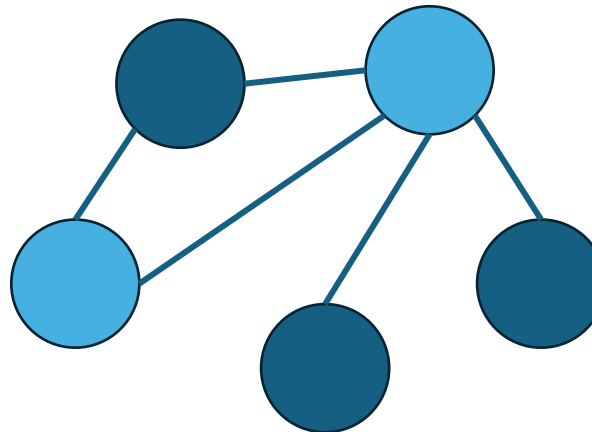
(Document): przechowują informacje jako dokumenty, często w formatach takich jak JSON.

Przykłady: *MongoDB, Couchbase*.

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



Bazy klucz-wartość

Zalety i wady

Szybkość działania:

- Bazy klucz-wartość oferują bardzo szybkie operacje odczytu i zapisu, ponieważ dane są bezpośrednio dostępne przez klucz.

Prostota i elastyczność:

- Brak schematu pozwala na łatwe przechowywanie danych o dowolnej strukturze, co jest szczególnie przydatne w dynamicznych aplikacjach.

Łatwe skalowanie poziome:

- Rozproszone bazy klucz-wartość (np. Redis, DynamoDB) można łatwo skalować, co pozwala obsłużyć ogromne ilości danych i ruchu.

Lekkość systemu:

- Ze względu na prostą strukturę, bazy te mają niskie wymagania zasobów i są łatwe w implementacji w różnych środowiskach.

Brak zaawansowanych operacji:

- Nie obsługują zapytań typu SQL ani relacji między danymi, co ogranicza ich zastosowanie w bardziej skomplikowanych scenariuszach.

Trudności w zarządzaniu złożonymi danymi:

- Wartości przechowywane w bazie mogą być trudne do analizy, jeśli są złożonymi strukturami, np. zagnieżdżonymi obiektami.

Ograniczone funkcje indeksowania:

- Brak możliwości tworzenia indeksów na danych innych niż klucze, co może spowolnić wyszukiwanie w dużych zbiorach.

Potencjalna redundancja danych:

- W przypadku przechowywania powiązanych informacji konieczne jest duplikowanie danych, ponieważ brakuje mechanizmu relacji, co może prowadzić do problemów z synchronizacją.

Bazy klucz-wartość

Tablice (asocjacyjne)

Typ danych, który przechowuje pary (*unikatowy klucz, wartość*) i umożliwia dostęp do wartości poprzez podanie klucza.

Klasyczne tablice:

- klucze stanowią liczby całkowite (id) z przedziału: (1,2,...,n)
- wartości w kolumnach tablicy są jednego typu

Tablice asocjacyjne:

- klucze stanowią dowolne typy: „styczeń”, „klient:1234:nazwisko”
- wartości w tablicy mogą być mieszane i różnego typu: liczby całkowite, zmiennoprzecinkowe, inne tablice, binarne, stringi, itd...

klucz



wartość

Definicja par (klucz, wartość):

```
kolor[„czerwony:hex”]=, #FF0000’;  
kolor[„czerwony:dec”]={RED=255, GREEN=0, BLUE=0};  
kolor[„niebieski:hex”]=, #0000FF’;  
kolor[„żółty:hex”]=, #FFFF00’;
```

Użycie tablicy asocjacyjnej:

```
<FONT COLOR=<?=tablica[„niebieski:hex”]?>> jestem blue </FONT>
```

Wynik: **jestem blue**

Bazy klucz-wartość

Klucz -> wartość

Klucz to odwołanie (adres) do wartości.

Kluczami mogą być:

ciągi znaków, liczby, lista, zbiory, zbiory uporządkowane, hasze, bitmapy

Wartość to obiekt, który jest skojarzony z kluczem

Każda wartość musi mieć unikatowy identyfikator w postaci klucza;

Wartościami mogą być:

liczby całkowite, liczby zmiennoprzecinkowe, ciągi znaków, duże obiekty binarne (BLOB),
konstrukcje mające strukturę np.: JSON, obrazy, pliki audio i video, itd...

Bazy klucz-wartość

Przestrzenie nazw (ang. namespaces)

to logiczny sposób organizowania kluczy w bazach klucz-wartość. Dzięki przestrzeniom nazw można grupować powiązane dane i uniknąć konfliktów nazw kluczy, zwłaszcza w dużych systemach współdzielonych przez różne aplikacje.

Klucze muszą być unikalne w ramach zdefiniowanej przestrzeni nazw

Przestrzeń nazw może stanowić cała baza lub inna zdefiniowana struktura w ramach jednej bazy (zależy od konkretnej implementacji bazy klucz-wartość)

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

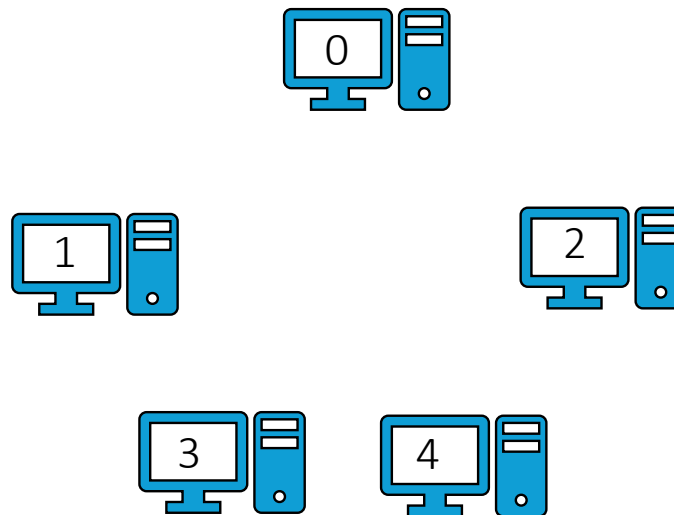
Bazy klucz-wartość

Partycjonowanie

technika dzielenia danych na mniejsze części (partycje), które są przechowywane na różnych serwerach lub w różnych lokalizacjach w celu poprawy wydajności i skalowalności.

Liczba partycji: 5

Klucze	Partycja
od A do D	0
od E do H	1
od I do L	2
od M do R	3
od S do Z	4

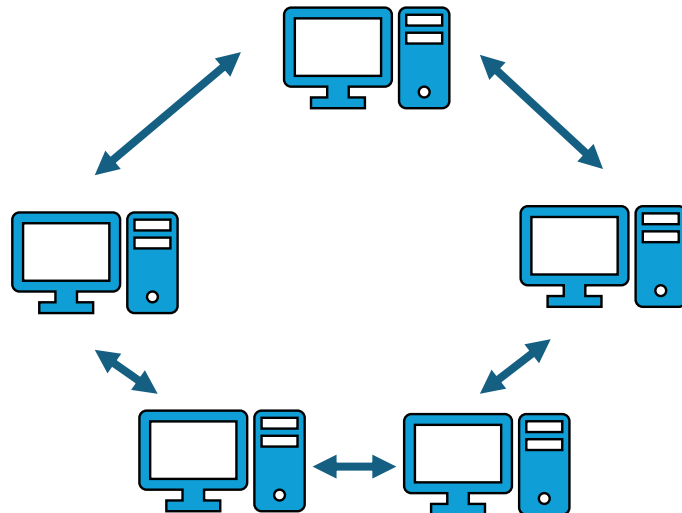


Bazy klucz-wartość

Pierścień

to logiczna struktura organizacji partycji.

Ma ona formę okręgu, w którym każdy serwer lub każda instancja bazy klucz-wartość działająca na serwerze mają połączenie z dwoma sąsiadującymi serwerami lub instancjami.



Bazy klucz-wartość

Ograniczone operacje

Wszystkie operacje na wartościach opierają się na kluczach

Dozwolone operacje:

- Pobranie wartości
`zmienna:=daneAplikacji[klient_23:nazwisko']`
- Ustawienie wartości
`daneAplikacji[klient_23:nazwisko']:=zmienna`
- Usunięcie wartości
`delete(daneAplikacji[klient_23:nazwisko'])`
- i to wszystko ! 

Konieczność implementacji przez programistów bardziej zaawansowanych operacji

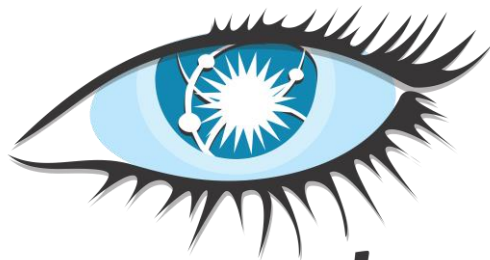
Wyszukiwanie określonych wartości lub ich zakresu wymaga przeszukania w pętli i porównania z wzorcem każdej wartości

Bazy klucz-wartość



Amazon DynamoDB

<https://aws.amazon.com/dynamodb>



cassandra

<https://cassandra.apache.org>

Redis

<https://redis.io>



<https://riak.com>

Bazy wektorowe

Wektorowa baza danych

to systemy przechowywania i organizacji danych, które wykorzystują matematyczne wektory do reprezentacji informacji.

Wektor to zbiór liczb opisujących pewne cechy obiektu, np. punktów w przestrzeni, słów w tekście lub właściwości obrazów. Dzięki temu można analizować i porównywać dane, szukając między nimi podobieństw. W praktyce bazy wektorowe stanowią kluczowe narzędzie w teleinformatyce oraz w obszarach związanych ze sztuczną inteligencją.

Zalety baz wektorowych

Szybkie wyszukiwanie podobieństw:

Idealne do analizy danych wielowymiarowych.

Skalowalność:

Możliwość pracy z ogromnymi zbiorami danych, np. miliony obrazów czy dokumentów.

Uniwersalność:

Obsługa różnych typów danych, takich jak tekst, obrazy, dźwięki czy dane przestrzenne.

Optymalizacja dla AI:

Umożliwiają integrację z modelami uczenia maszynowego i głębokiego.

Wady baz wektorowych

Wysokie zużycie pamięci

Wielowymiarowe wektory wymagają dużej przestrzeni dyskowej i pamięci operacyjnej.

Złożoność obliczeniowa

Wyszukiwanie i porównywanie wektorów w przestrzeni o dużej liczbie wymiarów jest czasochłonne.

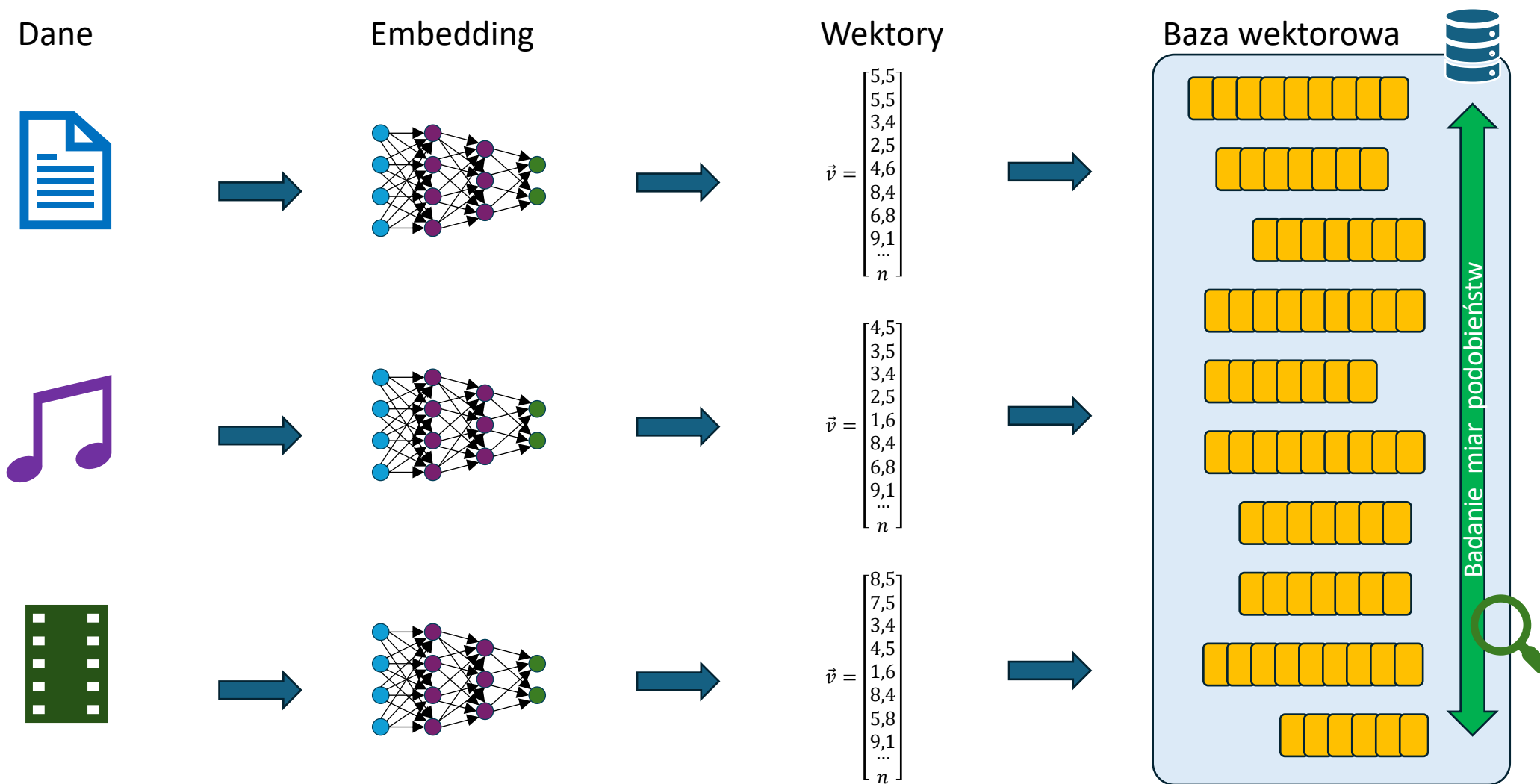
Przybliżone wyniki wyszukiwania

Algorytmy **Approximate Nearest Neighbor (ANN)** często zwracają wyniki przybliżone, a nie dokładne.

Trudność interpretacji

Wysokowymiarowe przestrzenie trudno jest wizualizować i zrozumieć intuicyjnie.

Bazy wektorowe



Embedding

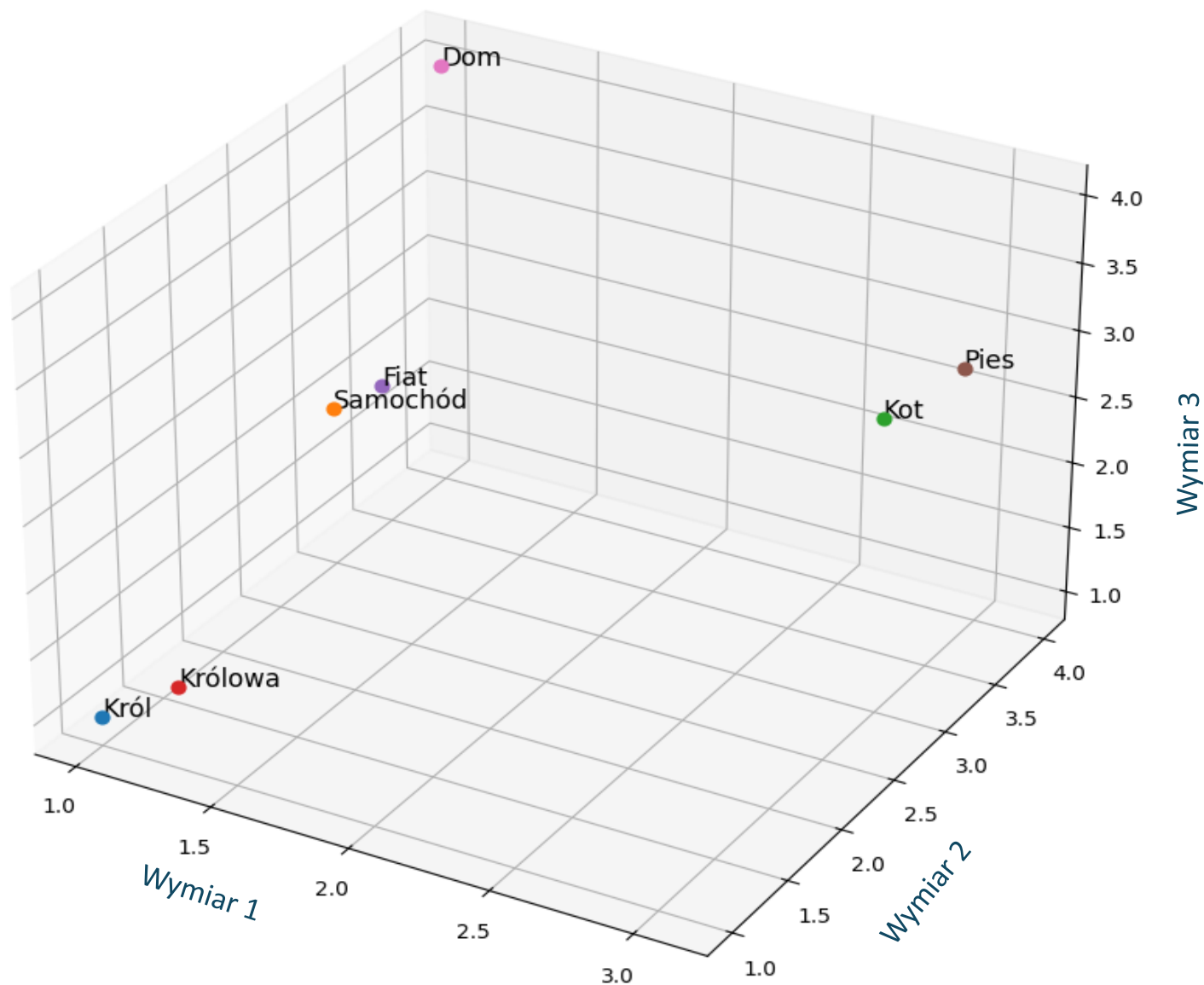


Wektory

'Król' $\rightarrow (1.0, 1.1, 1.0)$
 'Samochód' $\rightarrow (1.0, 3.1, 2.1)$
 'Kot' $\rightarrow (3.1, 3.2, 3.2)$
 'Królowa' $\rightarrow (1.2, 1.2, 1.2)$
 'Fiat' $\rightarrow (1.1, 3.2, 2.1)$
 'Pies' $\rightarrow (3.1, 3.5, 3.1)$
 'Dom' $\rightarrow (1.1, 4.1, 4.1)$

Macierz odległości (podobieństwa)

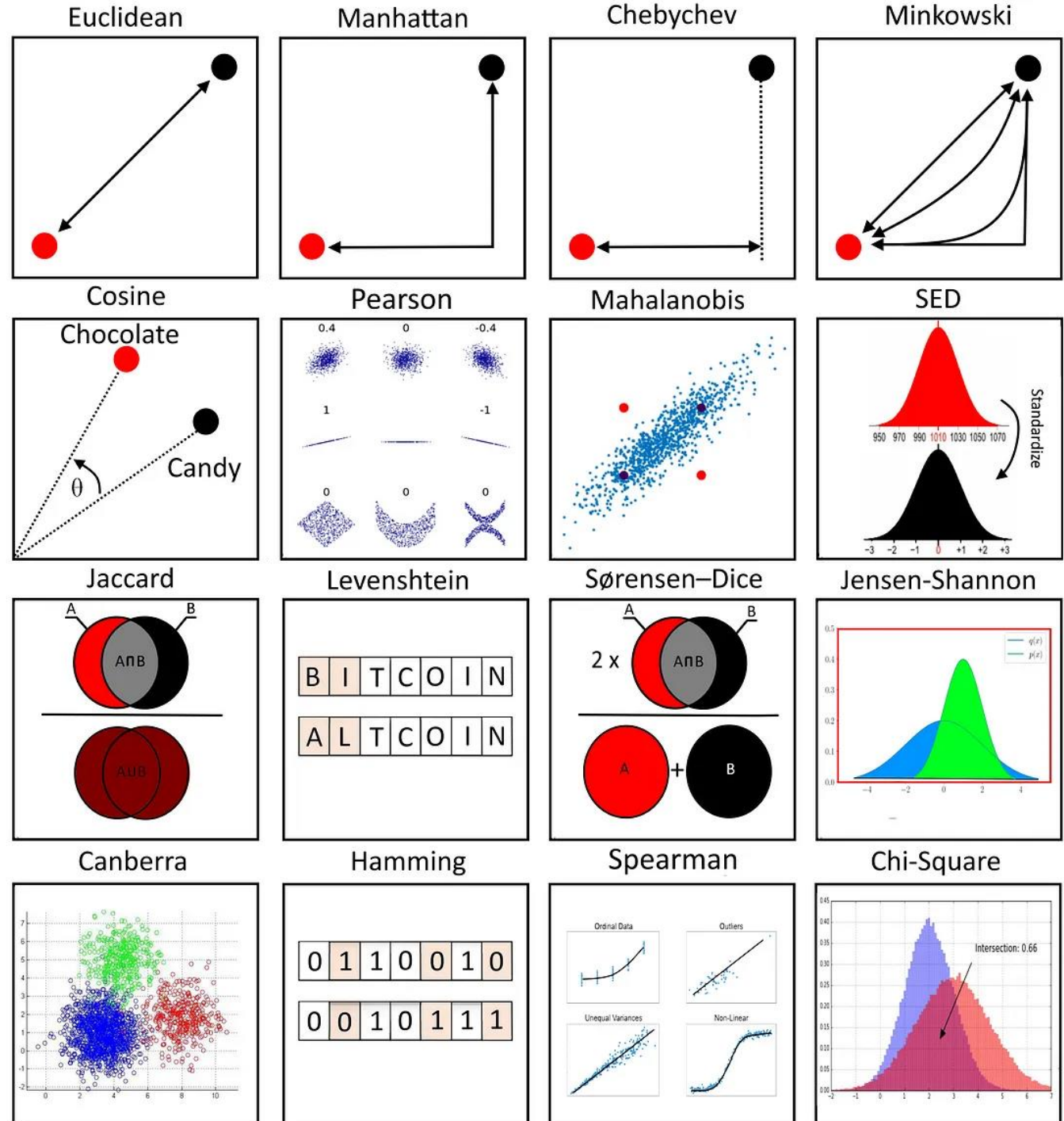
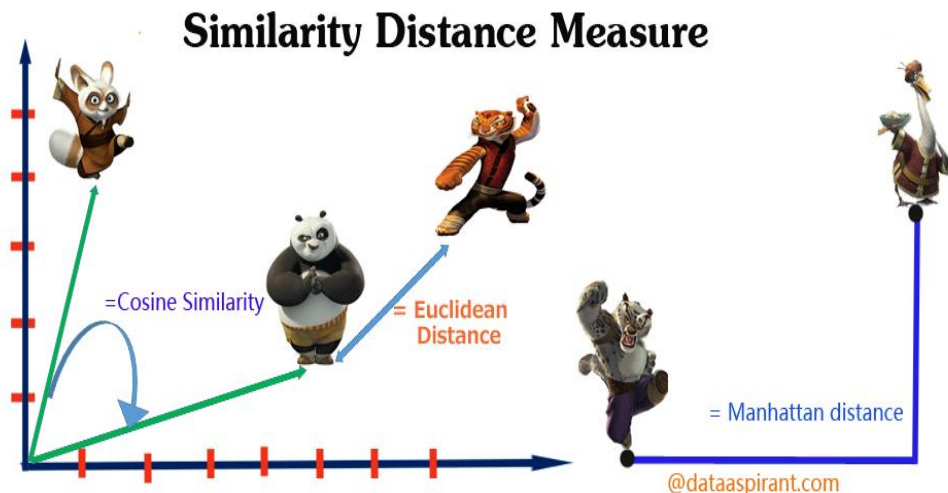
	Król	Samochód	Kot	Królowa	Fiat	Pies	Dom
Król	0,0000	2,2825	3,6959	0,3000	2,3728	3,8184	4,3151
Samochód	2,2825	0,0000	2,3728	2,1119	0,1414	2,3601	2,2383
Kot	3,6959	2,3728	0,0000	3,4073	2,2825	0,3162	2,3707
Królowa	0,3000	2,1119	3,4073	0,0000	2,1955	3,5369	4,1024
Fiat	2,3728	0,1414	2,2825	2,1955	0,0000	2,2561	2,1932
Pies	3,8184	2,3601	0,3162	3,5369	2,2561	0,0000	2,3152
Dom	4,3151	2,2383	2,3707	4,1024	2,1932	2,3152	0,0000



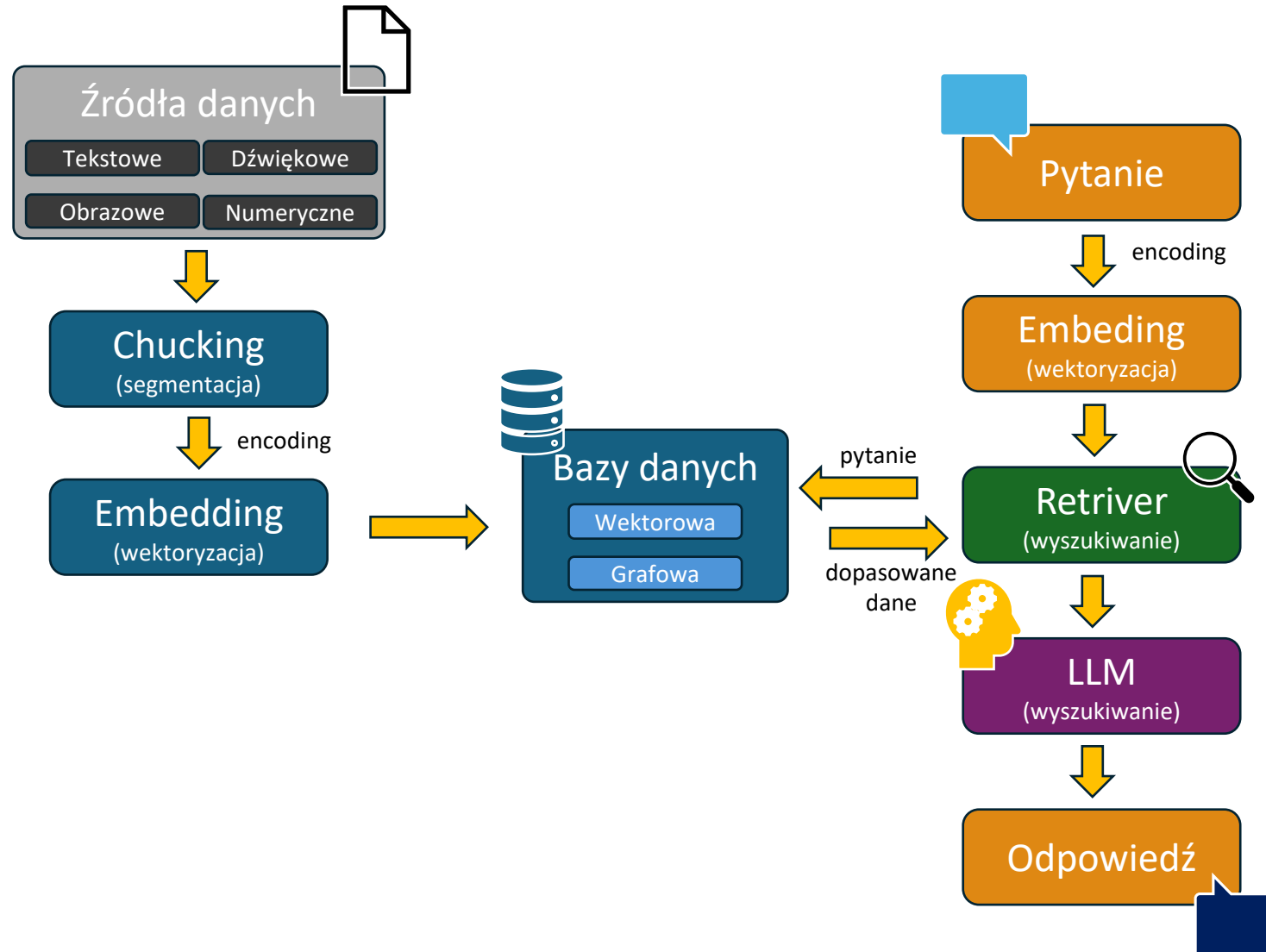
Miary podobieństw

Przez **miarę podobieństwa** (ang. *similarity measure*) rozumie się formalną funkcję lub procedurę służącą do ilościowej oceny stopnia podobieństwa pomiędzy dwiema jednostkami — mogą to być obiekty, zbiory danych, wektory cech, wzorce czy sygnały.

Innymi słowy, miara podobieństwa przypisuje parze obiektów wartość liczbową, która odzwierciedla, na ile oba obiekty są do siebie zbliżone pod względem zdefiniowanych cech lub kryteriów.



Retrieval-Augmented Generation (RAG) w AI



Bazy wektorowe



<https://www.pinecone.io>



<https://milvus.io>



<https://qdrant.tech>



<https://weaviate.io>

Bazy grafowe

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

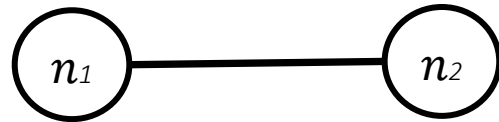
$N \cap E = \emptyset$ - zbiór wierzchołków i rodzina krawędzi nie mają części wspólnej.

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.

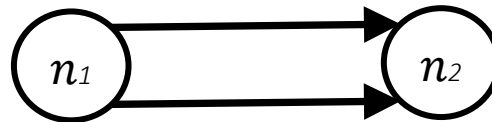


Graf zorientowany (skierowany): jeżeli $\langle n_1, n_2 \rangle \in E$ i $\langle n_2, n_1 \rangle \notin E$, to taka para jest nazywana krawędzią zorientowaną (skierowaną) lub wprost krawędzią..



Multigraf: graf z krawędziami równoległymi.

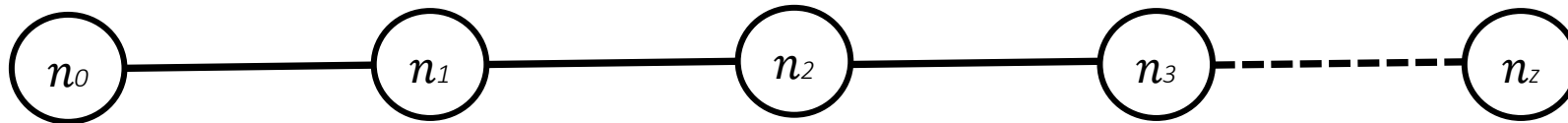
Unigraf: graf bez krawędzi równoległych.



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.

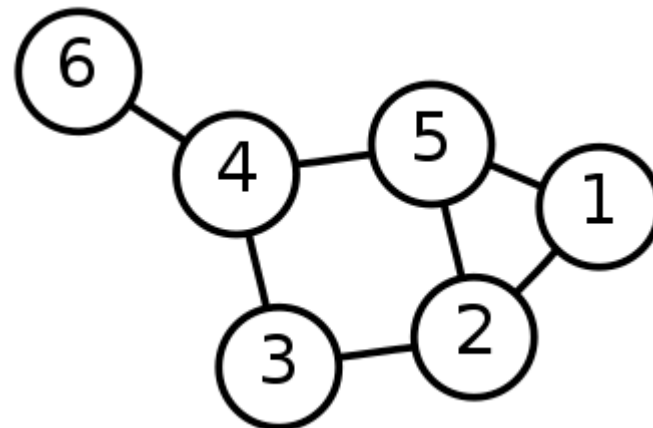
Podstawy teorii grafów i elementy grafu

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.



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.

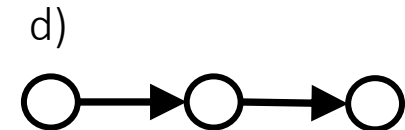
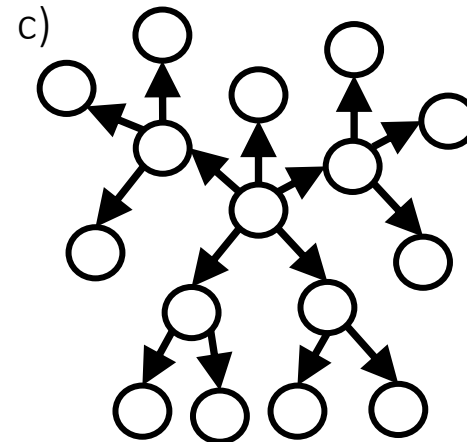
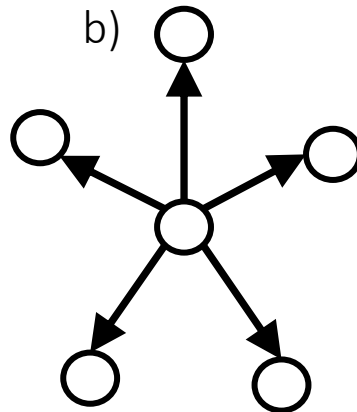
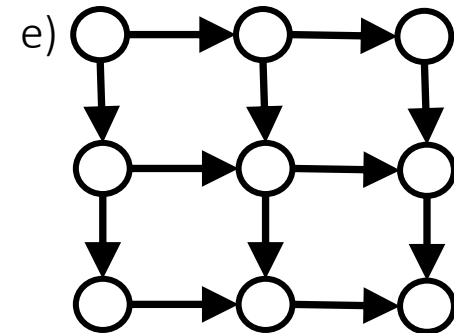
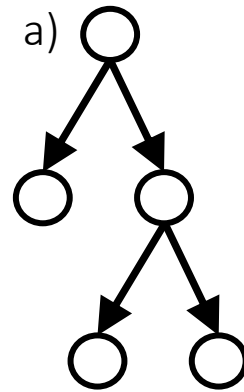
Rekomendacje Produktów:

- 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)

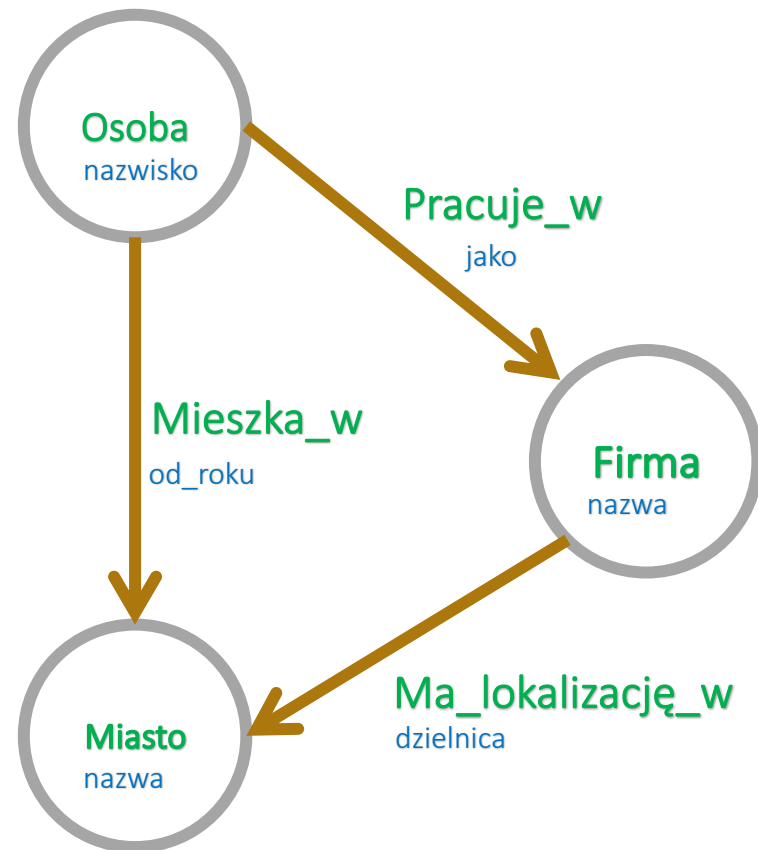
Reprezentują relacje między obiektami, np. znajomość, przepływ informacji.

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.

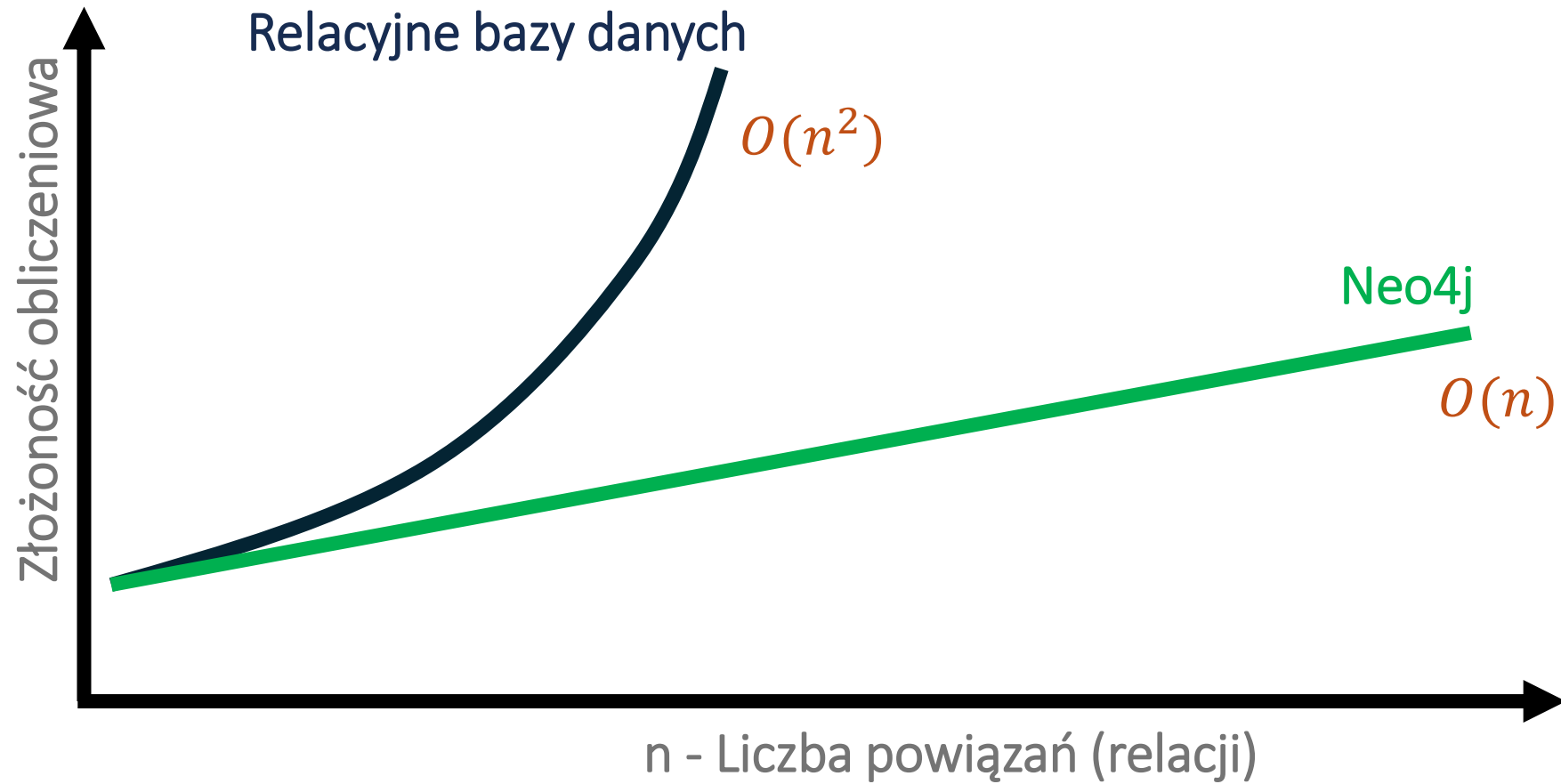


Bazy grafowe vs. Bazy relacyjne



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

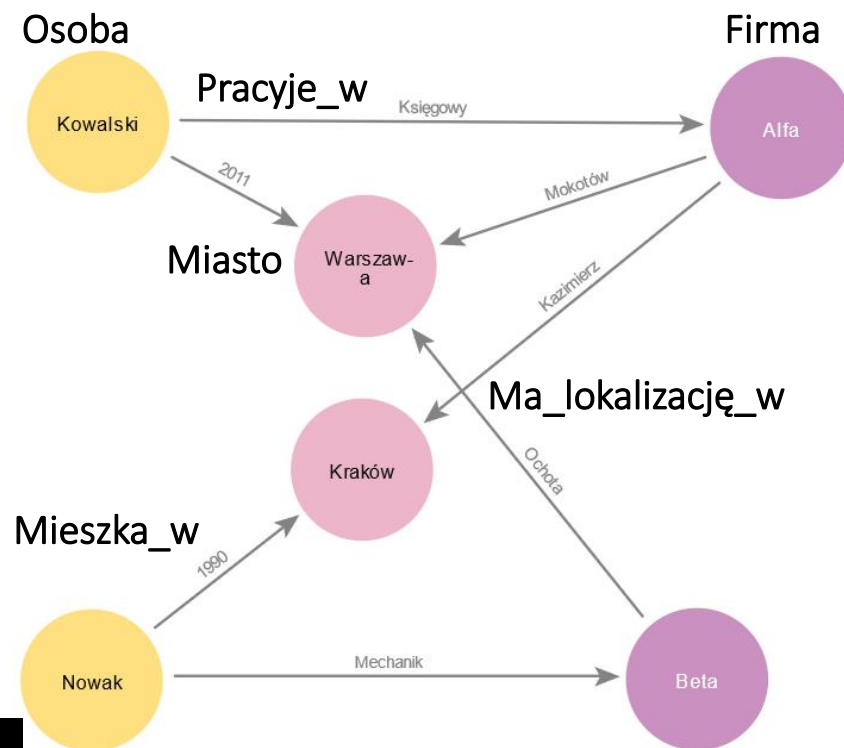
SQL

```
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



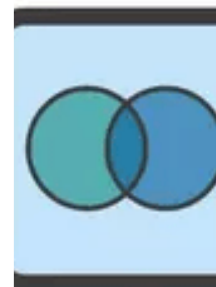
Algorytmy centralności

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



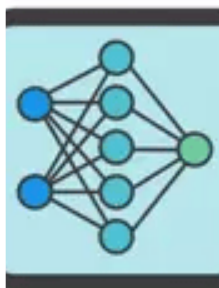
Algorytmy wspólnoty

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

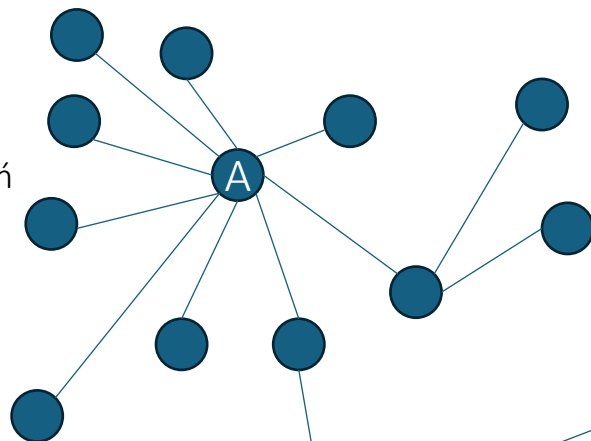
Prognozują czy dwa węzły w grafie mogą w przyszłości stworzyć połączenie

Algorytmy centralności

Degree Centrality
(Centralność stopnia)

Liczba połączeń?

"A" ma najwyższy stopień

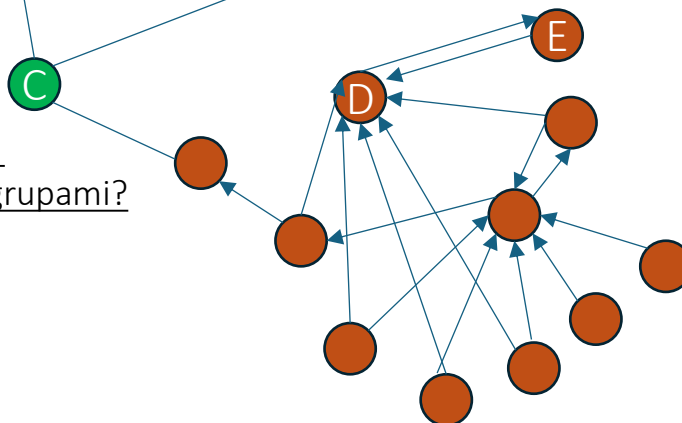


Betweenness Centrality
(Centralność bliskości)

Który węzeł ma największą kontrolę

nad przepływem między węzłami i grupami?

"C" jest mostem



Louvain Modularity
(Modułowość Louvain)

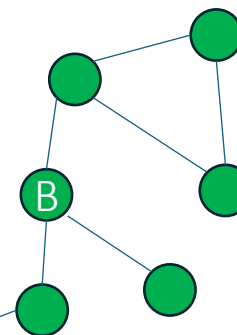
Jak efektywnie podzielić graf na społeczności?



Closeness Centrality
(Centralność pośrednictwa)

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

"B" jest najbliższy z najmniejszą
liczbą połączeń

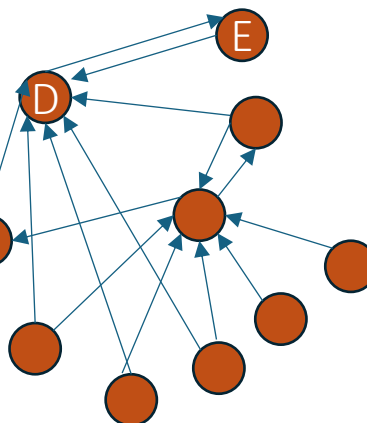


PageRanking

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



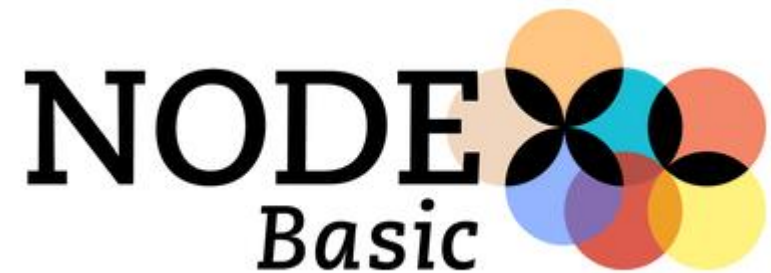
<https://neo4j.com/>



<https://gephi.org/>



<https://cytoscape.org/>



<https://nodexl.com/>

Neo4j



<https://neo4j.com/>



Build fast, scalable, and intelligent applications in the cloud.

- ✓ Fully-managed updates and patches
- ✓ Robust security, reliability and ACID compliance
- ✓ Built-in tools to learn, build, and visualize
- ✓ No credit card required

Log in

Log in to Neo4j to continue

Email address*

Log in

Don't have an account? [Sign up](#)

OR

Continue with Google

Continue with Organization SSO

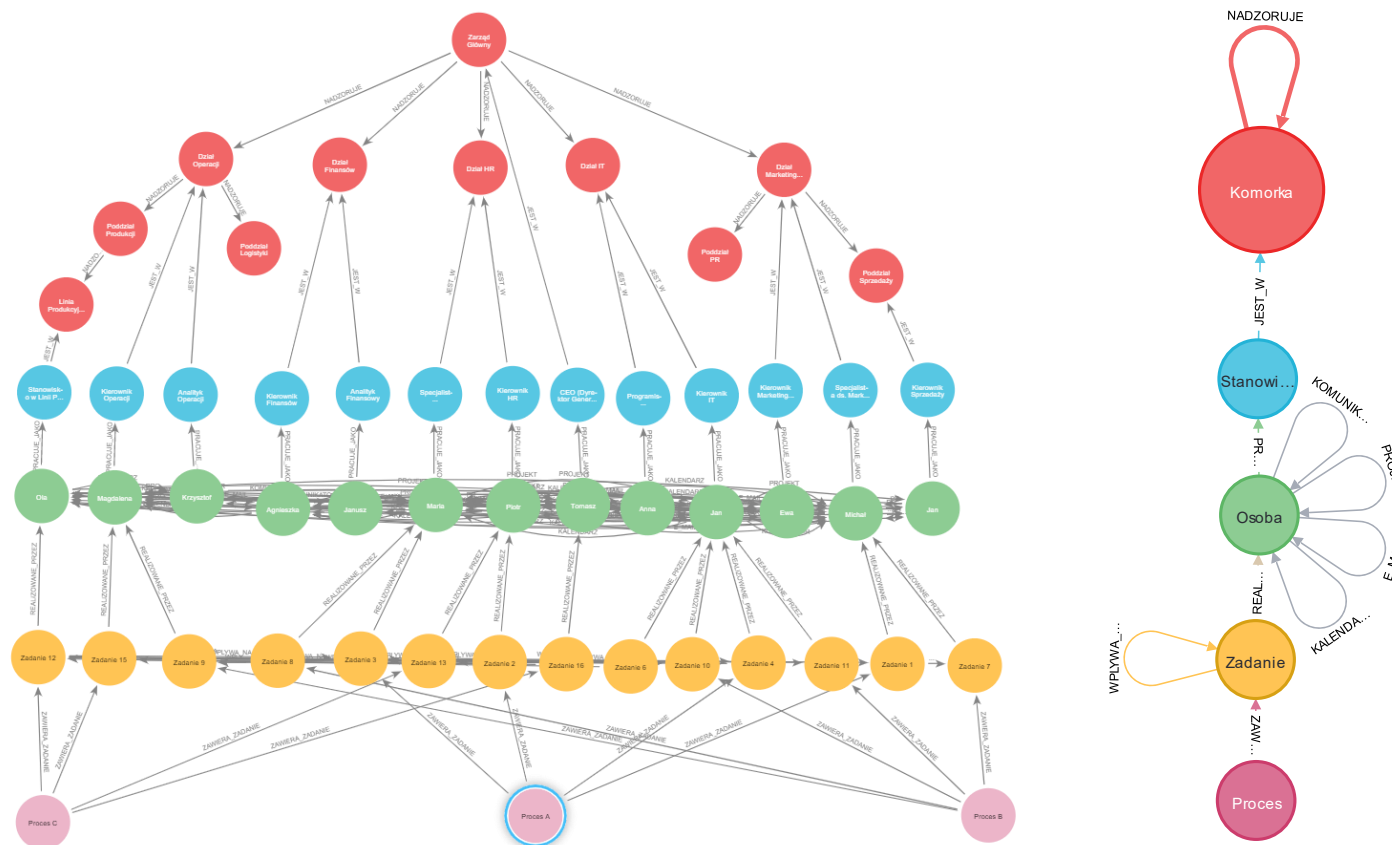
The screenshot shows the Neo4j Aura web interface. On the left is a sidebar with navigation options: Data services (Instances, Import, Tools, Explore, Query), Operations (Metrics, Logs), and Project (Users, Billing, Settings). The main area is titled 'Database information' for 'Instance: Instance01', 'Database: neo4j', and 'User: neo4j'. It displays 'Nodes (220)' with a visual representation of 'Pojecie' and 'Publikacja', and 'Relationships (224)' with 'Cytowana_przez', 'Podstawa_dla', and 'Powiazane'. A 'Property keys' section lists attributes like ~end_node_id, ~start_node_id, abstrakt, autor, data, description, dzielnica, end_node_id, id, ID, idfirma, idmiasto, idosoba, jako, labels, link, name, nazwa, and nazwisko. A query editor shows the Cypher query: `neo4j$ MATCH p=()-[:Cytowana_przez]->() RETURN p LIMIT 25;`. The results are shown in a 'Graph' view, displaying a network of nodes labeled 'wydana' connected by 'Cytowana_przez' relationships. A 'Results overview' panel on the right shows 'Nodes (11)' and 'Relationships (13)'. The bottom right corner indicates 'Started streaming 13 records after 51 ms and...'.

Publikacja



Wykorzystanie nowoczesnych metod z obszaru inżynierii w postaci baz grafowych w zarządzaniu organizacją

<https://www.drmalinowski.edu.pl/publications/39843-wykorzystanie-nowoczesnych-metod-z-obszaru-inzynierii-w-postaci-baz-grafowych-w-zarzadzaniu-organizacja>



Baza dokumentów (NoSQL)

Baza dokumentów

to jeden z typów baz danych NoSQL, które oferują elastyczność w przechowywaniu danych w formacie dokumentów, zwykle w strukturze JSON lub BSON.

W przeciwieństwie do relacyjnych baz danych, bazy dokumentów nie wymagają sztywnego schematu, co czyni je idealnym rozwiązaniem dla dynamicznych aplikacji

Cechy

Brak sztywnego schematu

- pozwala na różne struktury danych w tej samej kolekcji.

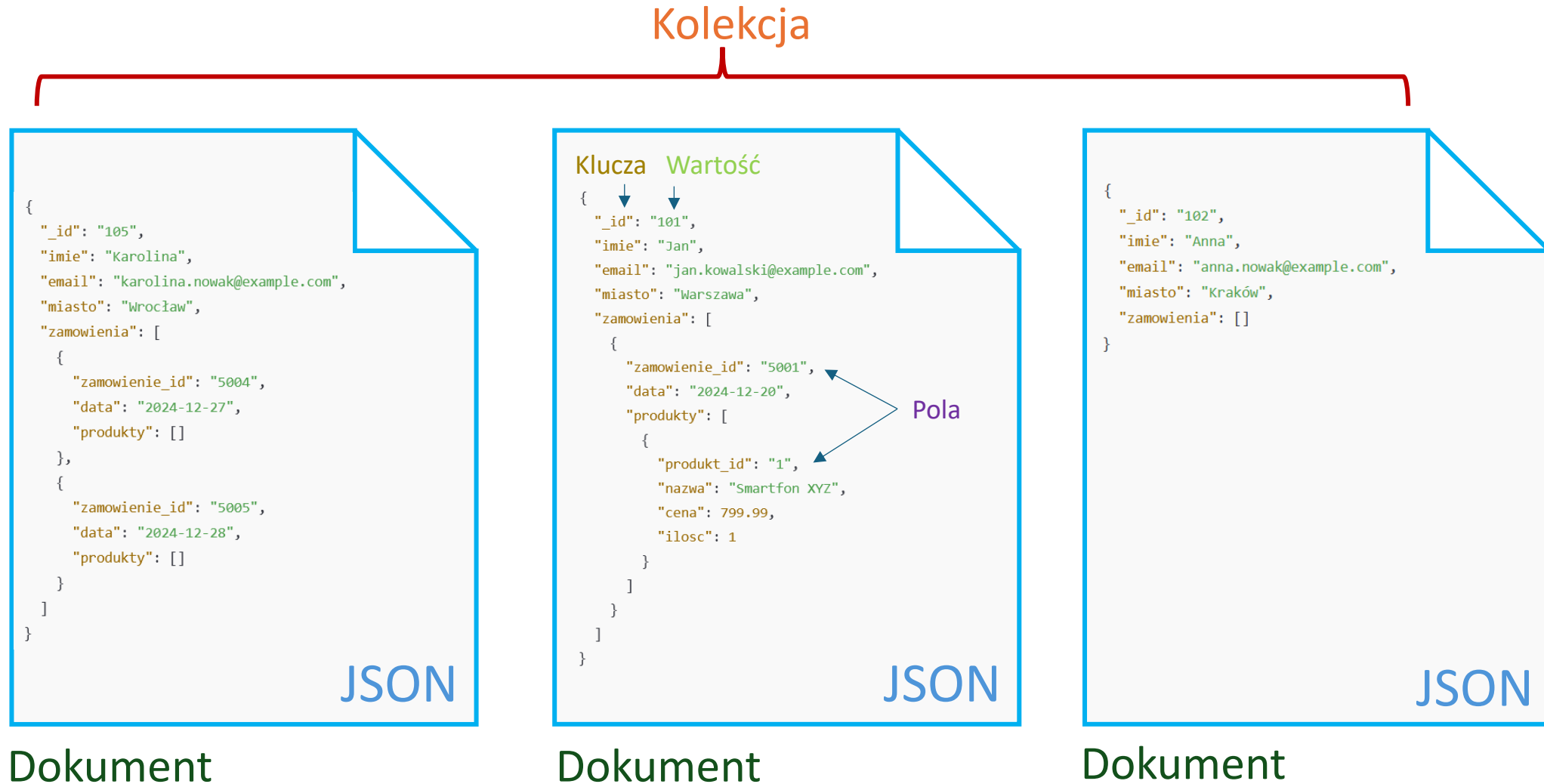
Łatwość przechowywania danych złożonych

- zagnieżdżone pola umożliwiają przechowywanie powiązanych informacji w jednym dokumencie.

Optymalizacja pod kątem odczytu

- dane są często pobierane w jednym zapytaniu, eliminując potrzebę dołączania (joinów).

Baza dokumentów (NoSQL)



Baza dokumentów (NoSQL)

Typy plików mogące być dokumentami

JSON

- standardowy format dla baz dokumentów, idealny do przechowywania złożonych struktur danych.

XML

- używany w niektórych systemach, szczególnie w starszych aplikacjach.

YAML

- bardziej czytelny dla człowieka format podobny do JSON.

Pliki binarne

- np. obrazy, dźwięki czy filmy, które mogą być przechowywane jako załączniki lub kodowane w formacie BSON (np. w MongoDB).

Dokumenty tekstowe

- takie jak pliki TXT czy Markdown, przechowywane jako pola w dokumentach.

Baza dokumentów (NoSQL)

Przykłady zastosowań

Aplikacje mobilne i webowe

- Przechowywanie profili użytkowników, preferencji czy historii zakupów.

Systemy e-commerce

- Zarządzanie produktami, zamówieniami i koszykami zakupowymi.

Analiza danych

- Przechowywanie niestukturalnych danych, takich jak logi systemowe czy dane IoT.

Baza dokumentów (NoSQL)



<https://www.mongodb.com/>



relax

<https://couchdb.apache.org/>



Amazon
DocumentDB

<https://aws.amazon.com/documentdb/>



ArangoDB

<https://arangodb.com/>

Bazy kolumnowe (NoSQL)

Baza dokumentów

to jeden z rodzajów baz danych NoSQL, który charakteryzuje się przechowywaniem danych w kolumnach zamiast tradycyjnych wierszy, jak w relacyjnych bazach danych.

Ten model przechowywania jest optymalny dla aplikacji wymagających szybkiego przetwarzania dużych ilości danych i analizowania ich w czasie rzeczywistym.

Cechy

Wysoka wydajność odczytu danych

- Idealne do analizy danych, ponieważ łatwo można odczytywać tylko te kolumny, które są potrzebne, bez przeszukiwania całego wiersza.
- Redukcja ilości operacji odczytu dzięki przechowywaniu kolumn razem.

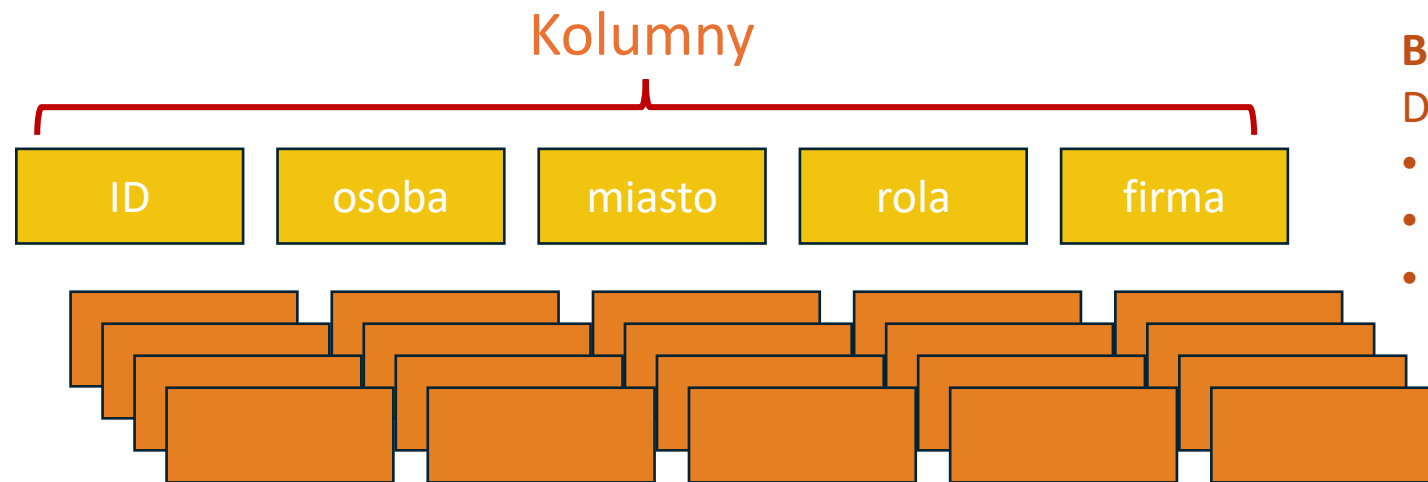
Elastyczne schematy

- Umożliwiają dodawanie nowych kolumn bez konieczności modyfikowania istniejącego schematu.
- Każdy wiersz może mieć różną liczbę kolumn, co zwiększa elastyczność przechowywania danych.

Optymalizacja dla dużej skali danych

- Bazy kolumnowe są zaprojektowane do przechowywania petabajtów danych i obsługi wielu równoległych zapytań.
- Rozproszona architektura zapewnia skalowalność poziomą.

Bazy kolumnowe (NoSQL)



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



Bazy kolumnowe (NoSQL)

Przykłady zastosowań

Analiza danych i Big Data

- Przetwarzanie dużych zbiorów danych, np. logów systemowych czy danych z urządzeń IoT.

Systemy rekomendacyjne

- Budowanie rekomendacji na podstawie analizy zachowań użytkowników.

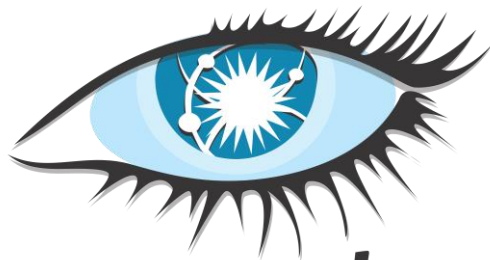
Obsługa dynamicznych raportów

- Generowanie złożonych raportów na podstawie danych w czasie rzeczywistym.

Bazy kolumnowe (NoSQL)



<https://hbase.apache.org>



cassandra

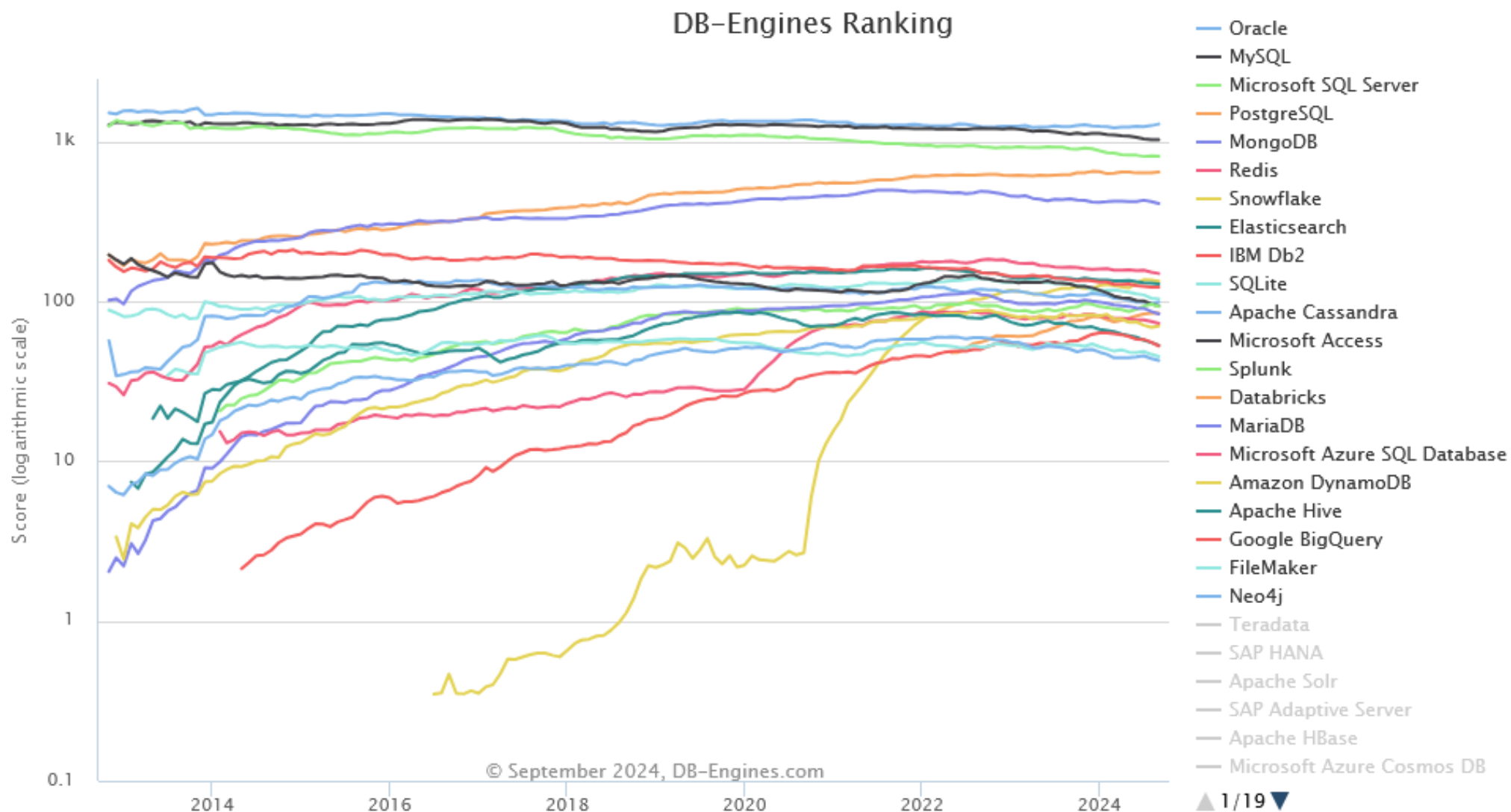
<https://cassandra.apache.org>



ScyllaDB

<https://www.scylladb.com>

Ranking - Systemy baz danych



DB-Engines Ranking

<https://db-engines.com/en/ranking>

Ranking - Systemy baz danych



423 systems in ranking, September 2024

Rank			DBMS	Database Model	Score		
Sep 2024	Aug 2024	Sep 2023			Sep 2024	Aug 2024	Sep 2023
1.	1.	1.	Oracle +	Relational, Multi-model i	1286.59	+28.11	+45.72
2.	2.	2.	MySQL +	Relational, Multi-model i	1029.49	+2.63	-82.00
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model i	807.76	-7.41	-94.45
4.	4.	4.	PostgreSQL +	Relational, Multi-model i	644.36	+6.97	+23.61
5.	5.	5.	MongoDB +	Document, Multi-model i	410.24	-10.74	-29.18
6.	6.	6.	Redis +	Key-value, Multi-model i	149.43	-3.28	-14.26
7.	7.	↑ 11.	Snowflake +	Relational	133.72	-2.25	+12.83
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model i	128.79	-1.04	-10.20
9.	9.	↓ 8.	IBM Db2	Relational, Multi-model i	123.05	+0.04	-13.67
10.	10.	↓ 9.	SQLite +	Relational	103.35	-1.44	-25.85
11.	11.	↑ 12.	Apache Cassandra +	Wide column, Multi-model i	98.94	+1.94	-11.11
12.	12.	↓ 10.	Microsoft Access	Relational	93.76	-2.61	-34.81
13.	13.	↑ 14.	Splunk	Search engine	93.02	-3.08	+1.63
14.	↑ 15.	↑ 17.	Databricks +	Multi-model i	84.24	-0.22	+9.06
15.	↓ 14.	↓ 13.	MariaDB +	Relational, Multi-model i	83.44	-3.09	-17.01
16.	16.	↓ 15.	Microsoft Azure SQL Database	Relational, Multi-model i	72.95	-2.08	-9.78
17.	17.	↓ 16.	Amazon DynamoDB +	Multi-model i	70.06	+1.15	-10.85
18.	↑ 19.	18.	Apache Hive	Relational	53.07	-2.17	-18.76
19.	↓ 18.	↑ 20.	Google BigQuery +	Relational	52.67	-2.86	-3.80
20.	20.	↑ 21.	FileMaker	Relational	45.20	-1.47	-8.40
21.	21.	↑ 23.	Neo4j +	Graph	42.68	-1.22	-7.71
22.	↑ 23.	↓ 19.	Teradata	Relational, Multi-model i	41.47	-0.78	-18.86
23.	↓ 22.	↓ 22.	SAP HANA +	Relational, Multi-model i	41.32	-1.04	-9.29
24.	24.	24.	Apache Solr	Search engine, Multi-model i	34.15	-2.14	-12.94
25.	25.	25.	SAP Adaptive Server	Relational, Multi-model i	32.67	-0.61	-10.64

DB-Engines Ranking

<https://db-engines.com/en/ranking>



ROBERTOFERRARO.ART