

Projekt z przedmiotu Bazy danych

Implementacja systemu realizującego wybrane podstawowe operacje na bazie
Northwind

Bartosz Kordek
Marcin Włodarczyk
Grzegorz Zacharski

Stos technologiczny

- front-end: rezygnacja z tworzenia graficznego interfejsu użytkownika
- back-end: Node.js oraz Express.js (REST API)
- baza danych: grafowa baza danych Neo4j
- konteneryzacja: Docker
- system kontroli wersji: Git
- ciągła integracja (CI): GitHub Actions
- narzędzia: Postman

Dlaczego te technologie?

- Część technologii jest nam znana (JavaScript, Docker, GitHub Actions)
- Chęć poznania baz NoSQL oraz grafowej bazy danych Neo4j
- Łatwość połączenia z bazą danych Neo4j przy użyciu frameworka Express.js oraz Node.js
- Duża dostępność materiałów edukacyjnych w sieci związanych z wybranymi technologiami ze względu na ich popularność

Node.js oraz Express.js

Node.js

- otwartoźródłowe, wieloplatformowe środowisko uruchomieniowe do tworzenia aplikacji serwerowych w języku JavaScript
- odpowiednie np. do przeglądarkowych aplikacji czasu rzeczywistego (gry przeglądarkowe) oraz aplikacji używających wielu operacji wejścia/wyjścia
- powszechnie używane w aplikacjach komercyjnych (IBM, LinkedIn, Microsoft, Netflix, Yahoo!, Paypal)

Express.js

- otwartoźródłowy framework do tworzenia aplikacji serwerowych z wykorzystaniem Node.js
- najpopularniejszy framework dla back-endu wykorzystujący język JavaScript
- wykorzystywany do tworzenia aplikacji webowych (np. REST API) oraz mobilnych

Graflowe bazy danych

Na przykładzie Neo4J



Motywacja

- chęć poszerzenia swojej wiedzy
- wsparcie Neo4j dla JavaScript
- dobra dokumentacja oraz wiele dostępnych materiałów od Neo4j

Use Your Favorite Programming Language

Neo4j officially supports drivers for .Net, Java, JavaScript, Go and Python. Our community contributors provide many more, including PHP, Ruby, R, Erlang, Clojure and C/C++.

[Learn More →](#)

☒ JavaScript

☐ Python

☐ .NET

☐ Java

```
1 // npm install --save neo4j-driver
2 // node example.js
3 var neo4j = require('neo4j-driver');
4 var driver = neo4j.driver('bolt://<HOST>:<BOLTPORT>',
5     neo4j.auth.basic('<USERNAME>', '<PASSWORD>'),
6     {/* encrypted: 'ENCRYPTION_OFF' */});
7
8 var query =
9 `
10 MATCH (p:Product)-[:PART_OF]->(:Category)-[:PARENT*0..]->
11 (:Category {categoryName:$category})
12 RETURN p.productName as product
13 `;
14
```


Czym jest grafowa baza danych?

- Jest to baza danych, która przechowuje dane i relacje między nimi w postaci grafu (najczęściej skierowanego).
- Zaprojektowana w taki sposób, by relacje między danymi były tak samo ważne jak same dane.
- Ma na celu przechowywanie danych bez ograniczania ich do wcześniej ustalonego modelu.

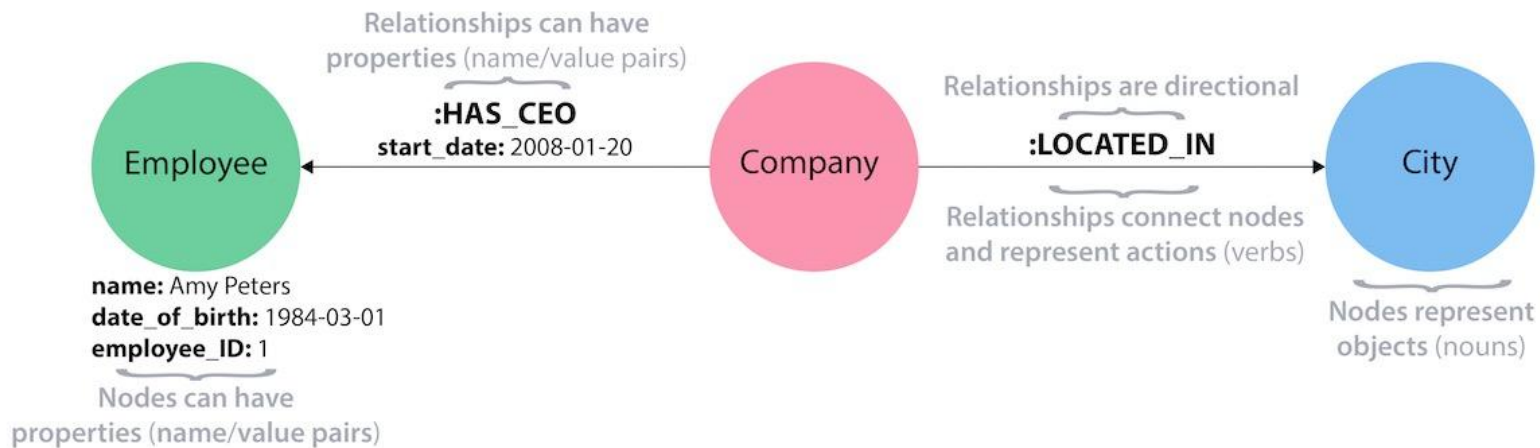
Z czego się składa?

Grafowa baza danych składa się z:

- **wierzchołka/węzłów** - mniej więcej odpowiednik **encji**
- **krawędzi/relacji** - opisuje **w jaki sposób** węzły są **połączone**. Zawsze posiada początek oraz koniec. Nazwa i kierunek pozwalają ustalić kontekst semantyczny jaki łączy dwa wierzchołki.

Każdy z wierzchołków oraz krawędzi może posiadać dowolną ilość:

- **etykiet** - jest to **nazwa** lub **identyfikator** wierzchołka lub krawędzi w grafie
- **właściwości** - są to elementy typu **klucz - wartość**. Zarówno wierzchołki oraz krawędzie mogą zawierać właściwości.



Cypher

- deklaratywny język zapytań
- używany w bazie Neo4j
- składnia pozwala w wizualny i logiczny sposób dopasować wzorce pomiędzy wierzchołkami i relacjami jakie są w grafie
- pozwala dodawać oraz wyciągać dane z grafu

Cypher using relationship 'likes'



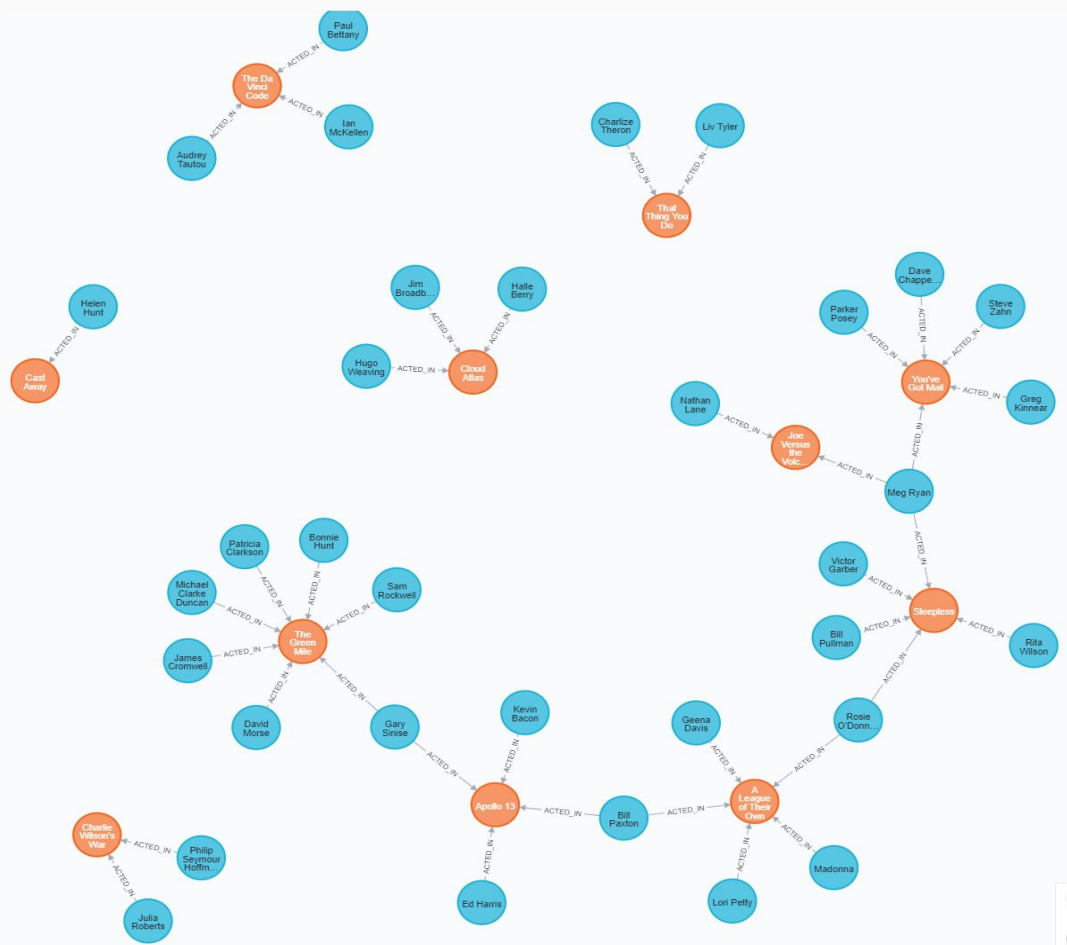
Cypher

(a) -[:LIKES]-> (b)

Cypher - podstawy

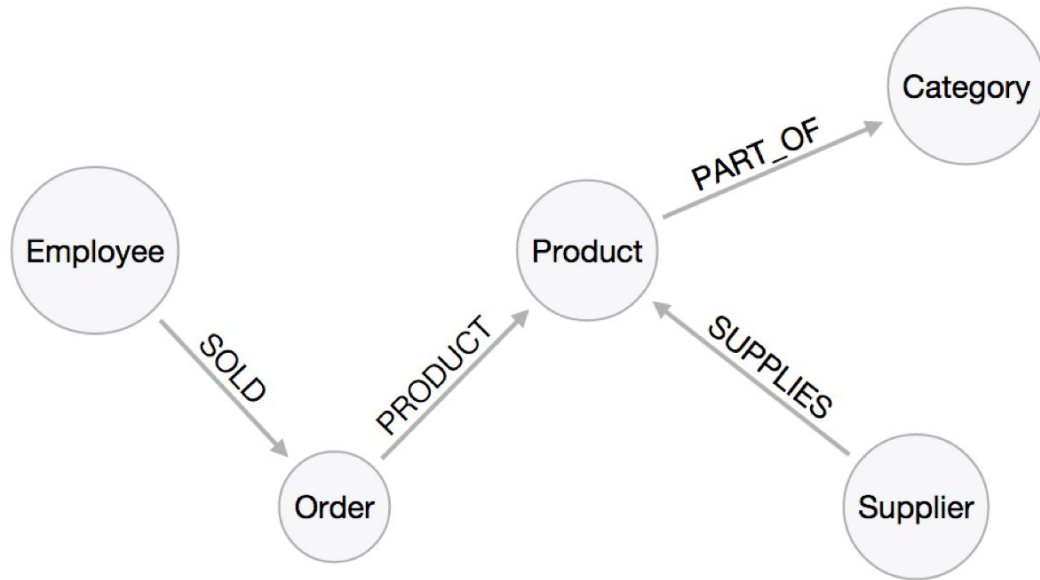
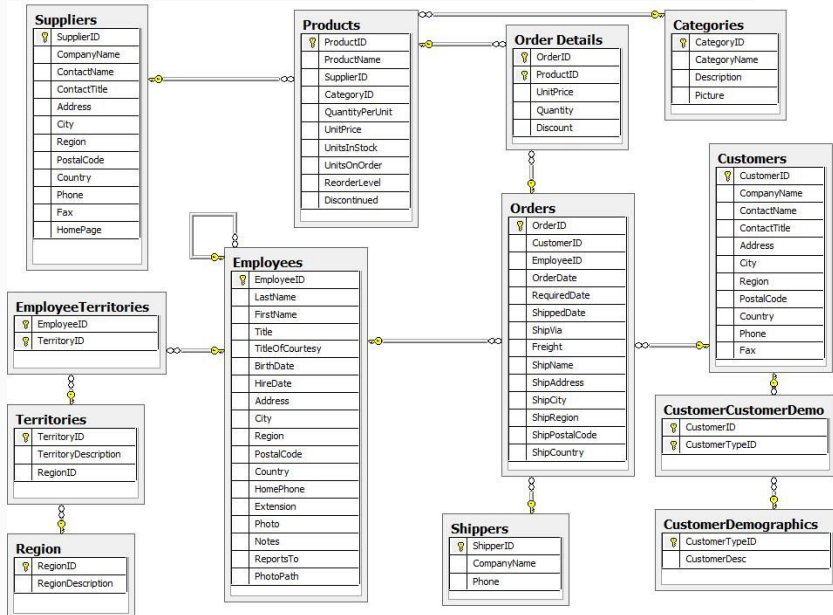
- **(m:Movie)** - nawiasy okrągłe oznaczają wierzchołek
- **[d:Directed]** - nawiasy kwadratowe oznaczają relację
- **->** oznacza kierunek relacji
- **CREATE** - tworzenie danych
- **MATCH** - określa wzór który będzie wyszukiwany w bazie danych
- **RETURN** - definiuje co powinno być zwrócone z bazy danych, np. wierzchołki na podstawie ustalonego uprzednio wzorca

Dopasuj osoby które grały razem w filmie z Tom Hanks. Zwróć film i aktora.



`MATCH (tom:Person {name:'Tom Hanks'})-[:ACTED_IN]->(m:Movie)-[:ACTED_IN]-(p:Person) RETURN p,m`

Baza grafowa na przykładzie Northwind



Zalety i wady neo4j

Zalety:

- bardzo szybki jeśli chodzi o wyszukiwanie zależności pomiędzy encjami
- elastyczne w rozbudowie
- wysoka wydajność odczytu i zapisu
- wsparcie wielu technologii/języków programowania, np. Java, .NET, JavaScript, Python, C/C++, Perl, PHP

Wady:

- brak jednego języka zapytań (ogólna wada podejścia grafowego)
- słaby mechanizm indeksowania
- mechanizm master-slave, master koordynuje wszystkie zapisy

Porównanie z relacyjnym modelem BD

RDBMS	Graph DB
Właściwości: Sztywny model - w przypadku zmiany konieczna przebudowa systemu, dodanie większej liczby encji i relacji wymaga ciągłego dbania o migracje i spójność	Właściwości: Brak sztywnego modelu - w każdej chwili można stworzyć dowolny wierzchołek (wraz z właściwościami) lub krawędź; brak konieczności przebudowy całego modelu
Zastosowanie: systemy niewymagające wyszukiwania wzajemnego powiązania między elementami, np. systemy magazynowe	Zastosowanie: systemy wymagające wyszukiwania wzajemnego powiązania między elementami, np. social networks

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

Porównanie wydajności rozwiązania relacyjnego oraz grafowego w znajdowaniu wzajemnych powiązań pomiędzy danymi.

Źródło: Graph Databases by Ian Robinson, Jim Webber and Emil aEifrem

Dziękujemy za uwagę!

Prezentację przygotowali:

Bartosz Kordek

Marcin Włodarczyk

Grzegorz Zacharski

