Neo4J

Warsztat z grafowej bazy danych Neo4J

Warsztaty: do zdobycia 20 pkt.

Projektowanie grafowych baz danych (1)

1. Identyfikujemy **węzły**, **etykiety** i **związki**

Identyfikujemy węzły jako obiekty o unikalnej tożsamości.

Przykład:

Ludzie **Tomek** i **Anna** są przyjaciółmi. Zarówno Tomek, jak i Anna przeczytali książkę **„Lubię Neo4J”**.

Projektowanie grafowych baz danych (2)

Identyfikujemy etykiety.

Etykiety mówią nam czym/kim są zidentyfikowane węzły.

Przykład etykiet:

Człowiek, Książka.

Projektowanie grafowych baz danych (3)

Przypisanie etykiet do węzłów.

Przykład:

Książka: „Lubię Neo4J”

Człowiek: Tomek, Anna

Projektowanie grafowych baz danych (4)

Powiązania pomiędzy obiektami.

Tomek **jest przyjacielem** Ani.

Ania **jest przyjaciółką** Tomka.

Tomek **przeczytał** „Lubię Neo4J”.

Ania **przeczytała** „Lubię Neo4J”.

Projektowanie grafowych baz danych (5)

- Kiedy Ania i Tomek zostali przyjaciółmi?
- Jaka jest ocena książki „Lubię Neo4J”?
- Kto jest autorem książki „Lubię Neo4J”?
- Ile lat ma Tomek?
- Ile lat ma Ania?
- Kto jest starszy Ania, czy Tomek?
- Kto pierwszy przeczytał książkę Tomek, czy Ania?

Projektowanie grafowych baz danych (6)

Identyfikacja własności.

jest przyjacielem: data poznania

przeczytał: data

Książka: ocena

Książka: autorzy

Człowiek: wiek

Projektowanie grafowych baz danych (7)

1. Opis dziedziny problemowej
2. Identyfikacja węzłów
3. Identyfikacja etykiet
4. Przypisanie etykiet do węzłów
5. Identyfikacja związków
6. Identyfikacja własności

Praca w grupach

Wybór tematów

Przykładowe tematy:

1. **Leczenie (symptomy jakich chorób ma dany pacjent, jeżeli ma alergię na dane produkty to co może zjeść, czy nowo przepisany lek jest w interakcji z aktualnie przyjmowanymi lekami itd.)**
2. **Podróże samolotowe (najbliższe lotnisko dla danego miasta, najkrótsza droga między dwoma miastami itd.)**
3. **Sieci społecznościowe**
4. **…**

Tworzenie grafowej bazy danych – CYPHER (1)

CREATE – tworzenie węzłów i związków

1. CREATE (n)
2. CREATE (n),(m)
3. CREATE (n:Person)
4. CREATE (n:Person:Swedish)
5. CREATE (n:Person { name : 'Andres', title : 'Developer' })
6. CREATE (a { name : 'Andres' }) RETURN a

Tworzenie grafowej bazy danych – CYPHER (2)

7. MATCH (a:Person), (b:Person)
   WHERE a.name = 'Node A' AND b.name = 'Node B‘
   CREATE (a)-[r:RELTYPE]->(b)
   RETURN r
8. MATCH (a:Person),(b:Person)
   WHERE a.name = 'Node A' AND b.name = 'Node B‘
   CREATE (a)-[r:RELTYPE { name : a.name + '<->' + b.name }]->(b)
   RETURN r

Tworzenie grafowej bazy danych – CYPHER (3)

9. CREATE

   p =(andres { name:'Andres' })-[:WORKS_AT]->(neo)<-[:WORKS_AT]-(michael { name:'Michael' })

   RETURN p

Tworzenie grafowej bazy danych – CYPHER (4)

MERGE – zapewnia, że dany wzorzec istnieje w grafie lub jest tworzony

1. MERGE (robert:Critic) RETURN robert, labels(robert)

Jeśli istnieje jakiś węzeł z etykietą Critic zwróci ten węzeł. Jeśli nie istnieje utworzy węzeł i przypisze mu etykietę Critic.

2. MERGE (charlie { name:'Charlie Sheen', age:10 }) RETURN charlie

Jeśli istnieje węzeł o właściwych wartościach **wszystkich** atrybutów zwróci ten węzeł, jeśli nie utworzy taki węzeł i nada mu odpowiednie wartości atrybutów.

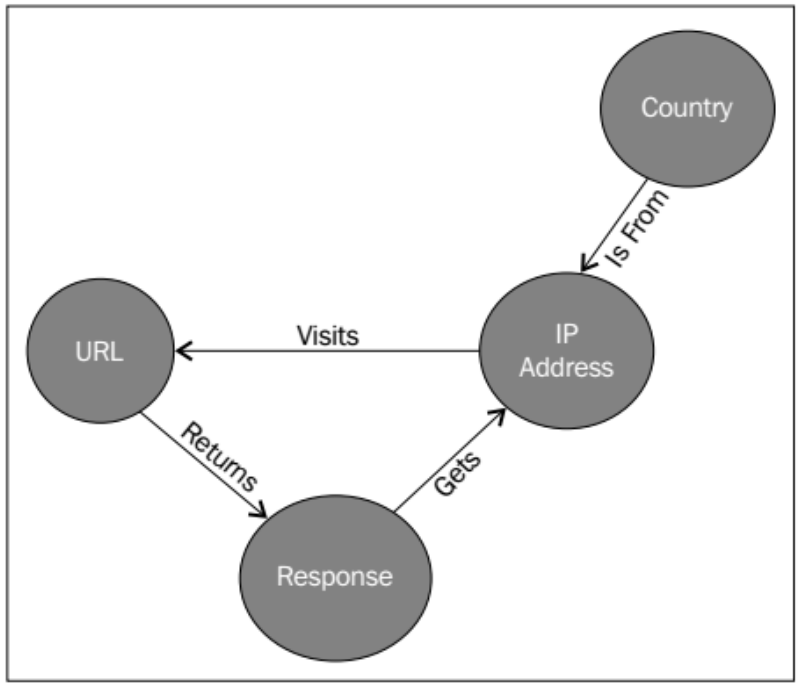
Tworzenie grafowej bazy danych – CYPHER (5)

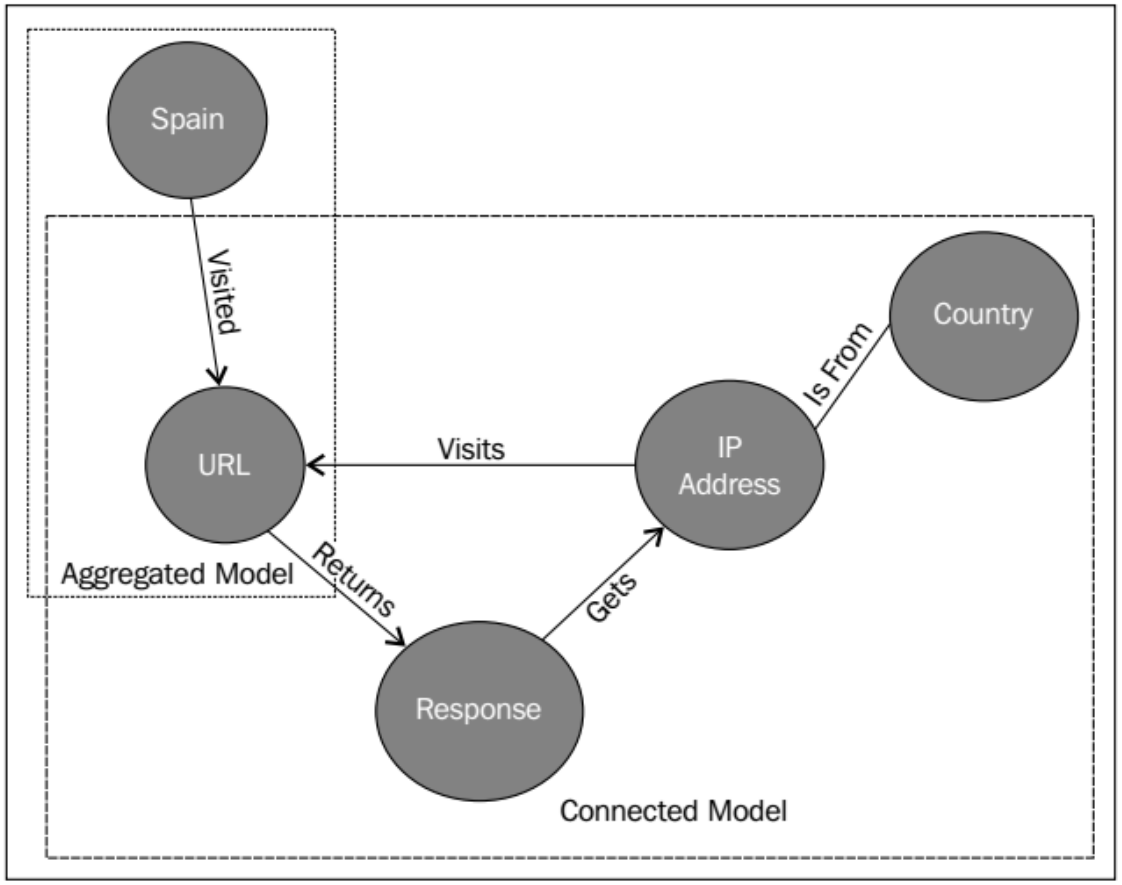
3. MATCH (person:Person)
   MERGE (city:City { name: person.bornIn })
   RETURN person.name, person.bornIn, city
4. MERGE (keanu:Person { name:'Keanu Reeves' })
   ON CREATE SET keanu.age = 45
   RETURN keanu.name, keanu.age
5. MERGE (keanu:Person { name:'Keanu Reeves' })
   ON MATCH SET keanu:Actor
   RETURN labels(keanu)

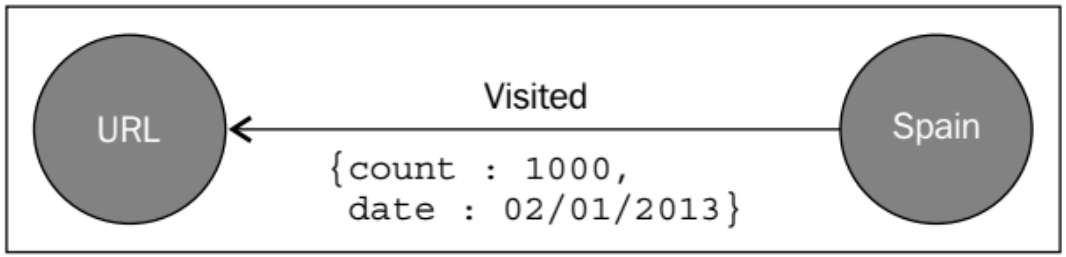
Tworzenie grafowej bazy danych – CYPHER (6)

6. MATCH
   (charlie:Person { name:'Charlie Sheen' }),(wallStreet:Movie
   { title:'Wall Street' })
   MERGE (charlie)-[r:ACTED_IN]->(wallStreet)
   RETURN charlie.name, type(r), wallStreet.title

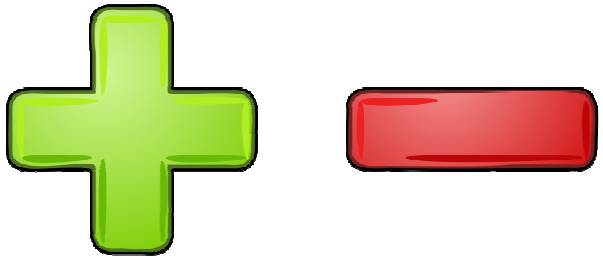
Jaki model?


Country
Is From
URL
Visits
IP
Address
Returns
Gets
Response


Spain
Visited
Country
Is From
URL
Visits
IP
Address
Aggregated Model
Returns
Gets
Response
Connected Model

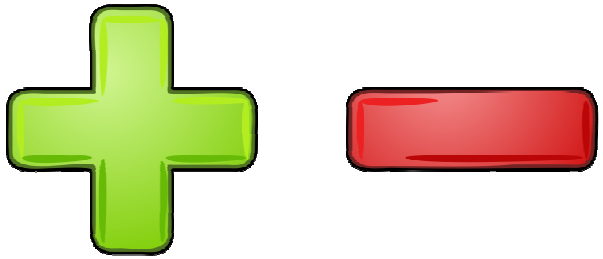

URL
Visited
Spain
{count : 1000,
date : 02/01/2013}

Model powiązań (ang. *connected model*)



1. Pozwala łatwo definiować nowe przypadki. 2. Stanowi źródło surowych danych dla innych źródeł	1. Wielkość bazy łatwo wzrasta. 2. Agregacje i inne zapytania mogą działać wolno na olbrzymiej liczbie węzłów i krawędzi.

Model zagregowany (ang. *agregated model*)



1. Szybko wykonywane są zapytania o agregaty. 2. Wielkość bazy szybko nie wzrasta.	1. Nowe agregacje nie mogą być wywiedzione z danych. 2. Potrzebne jest inne źródło danych, aby przechowywać surowe dane.

Model hybrydowy (ang. *hybrid model*)

Ma zalety obu modeli.

Zadanie dla studentów:

1. **Przygotuj listę przykładowych zapytań o agregaty.**
2. **Zaprojektuj hybrydowy model dla wybranej bazy.**

Definiowanie schematu (1)

Część możliwości daje tylko wersja ENTERPRISE!

Indeksy – możliwość tworzenia indeksu na zadanej własności, dla wszystkich węzłów o podanej etykiecie.

Indeksy są automatycznie uaktualniane.

Wolniejszy zapis i dane zajmują więcej miejsca.

```
CREATE INDEX ON :Person(name)
DROP INDEX ON :Person(name)
```

Definiowanie schematu (2)

Ograniczenia (ang. *constraints*) – nakładane na węzły lub związki

CREATE CONSTRAINT ON (book:Book) ASSERT book.isbn IS UNIQUE

DROP CONSTRAINT ON (book:Book) ASSERT book.isbn IS UNIQUE

Nie uda się utworzyć ograniczenia unikalności w sytuacji, gdy w bazie znajdują się dwie książki o tym samym ISBN.

Definiowanie schematu (3)

Każda książka musi mieć ustaloną własność isbn.

```
CREATE CONSTRAINT ON (book:Book) ASSERT exists(book.isbn)
DROP CONSTRAINT ON (book:Book) ASSERT exists(book.isbn)
```

Nie można usunąć własności!

```
MATCH (book:Book { title: 'Graph Databases' })
REMOVE book.isbn
```

Definiowanie schematu (4)

CREATE CONSTRAINT ON ()-[like:LIKED]-() ASSERT exists(like.day)

DROP CONSTRAINT ON ()-[like:LIKED]-() ASSERT exists(like.day)

Praca w grupach

Zaprojektuj ograniczenia i indeksy dla opracowywanej bazy.

Umieść plik na moodlu.