Bazy danych NoSQL

Instytut Informatyki UPH
Multimedialne i Obiektowe Bazy Danych
Dr Artur Niewiadomski
artur.niewiadomski@uph.edu.pl

Agenda

- Wprowadzenie do baz NoSQL
- Bazy typu klucz-wartość
- Bazy dokumentowe
  - MongoDB
- Bazy grafowe
  - Neo4J

Bazy NoSQL

- **N**ot **o**nly **SQL**
  - SZBD, które nie bazują na modelu relacyjnym
  - Wiele baz z grupy NoSQL wspiera składnię SQL
- Nie jest to jedna technologia
  - Raczej zbiór rozwiązań
- Dane przechowywane w BD nie wymagają ściśle określonych schematów
- Zazwyczaj nie wymagają złączeń
  - Łatwe skalowanie w poziomie
  - Efektywna realizacja zapytań

Bazy NoSQL (c.d.)

- Wysokie wymagania stawiane przed współczesnymi aplikacjami, np.
  - Wysoka wydajność
  - Skalowalność
  - Jednoczesna obsługa dużej liczby użytkowników
- Coraz częściej bazy relacyjne nie są w stanie sprostać tym wymaganiom
- Alternatywne systemy (np. OBD) istniały od dawna, ale głównie w niszowych zastosowaniach
- Wiele rozwiązań NoSQL odchodzi od restrykcyjnych postulatów ACID na rzecz BASE

ACID

- Atomicity (atomowość transakcji)
  - Operacje zawarte w transakcji wykonane w całości albo wcale
- Consistency (spójność)
  - Stan bazy danych spójny po zatwierdzeniu transakcji
- Isolation (izolacja)
  - Transakcje nie „widzą” przejściowego stanu bazy danych spowodowanego przez niezakończone inne transakcję
- Durability (trwałość)
  - Po zatwierdzeniu transakcji dane będą utrwalone

BASE

- Basically Available,
  - Większość danych dostępna przez cały czas
- Soft state,
  - Dane wystarczająco świeże
  - Stan bazy może ulec zmianie w wyniku uzgadniania stanu między replikami
- Eventually consistent
  - Osiągnięcie spójności odsunięte w czasie, ale osiągalne
  - Niektóre repliki są aktualizowane szybciej niż inne

Wybrane rodzaje baz NoSQL

- Bazy klucz-wartość
- Bazy kolumnowe
- Bazy dokumentowe
- Bazy grafowe

Bazy klucz-wartość

- Prosta semantyka hashmapy
  - Dwie „kolumny”: klucz i wartość
- Łatwe do rozproszenia
- Bardzo szybkie
- Np.
  - Redis,
  - Dynamo,
  - Riak
  - BerkeleyDB
  - MemcachedDB

Bazy klucz-wartość – kiedy stosować

- Przykładowe scenariusze, w których sprawdzają się bazy typu klucz wartość:
  - Przechowywanie informacji o sesji
    - Unikalne sessionId jako klucz
    - Dane o sesji umieszczone w jedym obiekcie wymagają tylko jednej operacji GET do odczytania i jednej operacji PUT do zapisu
  - Profile i preferencje użytkownika
    - Id użytkownika jako klucz
  - Dane koszyka zakupów
    - Jeśli chcemy, aby koszyk był dostępny pomiędzy różnymi sesjami i różnymi przeglądarkami

Bazy klucz-wartość – kiedy **NIE** stosować

- Przykładowe scenariusze, w których zastosowanie baz typu klucz wartość może być utrudnione, lub nieefektywne obejmują m.in. sytuacje w których występują:
  - Relacje pomiędzy danymi
  - Transakcje dla wielu operacji
    - Jeśli zapisujesz wiele kluczy i przy zapisie któregoś wystąpi błąd, to nie ma możliwości automatycznego wycofania pozostałych operacji
  - Zapytania na danych
    - W zasadzie można pytać tylko o klucz
  - Operacje na zestawach danych

Bazy kolumnowe

- Dane składowane w sekcjach kolumn
  - W bazach relacyjnych dane w wierszach, w poziomie
  - Tutaj w pionie, w kolumnach
- Łatwa agregacja,
- Większa elastyczność niż w RDB
- Dobre wsparcie dla hurtowni danych (OLAP)
- Przykłady:
  - Cassandra (Facebook),
  - BigTable (Google),
  - SimpleDB (Amazon)

Bazy kolumnowe – kiedy stosować

- Przykładowe scenariusze, w których sprawdzają się bazy kolumnowe:
  - Logowanie zdarzeń
  - CMS'y i platformy blogerskie
  - Liczniki
  - Wygasające dane
    - Dane z określonym TTL, automatycznie usuwane z bazy po upłynięciu terminu

Bazy dokumentowe

- Modelem jest kolekcja dokumentów
  - Rozszerzenie struktury typu klucz-wartość
- Dokument zawiera dane w standardowych formatach
  - Np. XML, JSON, YAML, BSON.
- Dokumenty reprezentowane przez unikalny klucz,
- Oprócz wyszukiwania klucz–wartość, zazwyczaj dostępne wyszukiwanie na postawie zawartości
  - API lub język zapytań.

Bazy dokumentowe

- Każdy dokument może mieć wiele pól,
- Nie musi definiować schematu
- Możliwe zagnieżdżanie i „linkowanie” dokumentów (referencje)
- Dokumenty dobrze odwzorowują obiekty
- Np.
  - MongoDB,
  - CouchDB
  - IBM Domino

Bazy dokumentowe – kiedy stosować

- Przykładowe scenariusze, w których sprawdzają się bazy dokumentowe:
  - Logowanie zdarzeń
    - Baza dokumentów jako centralne miejsce przechowywania logów z różnych aplikacji
  - CMS'y i platformy blogerskie
  - Aplikacje e-commerce
    - Elastyczny schemat przechowywania danych
    - Możliwość zmiany modelu danych bez konieczności zmiany schematu czy migracji danych

Bazy dokumentowe – kiedy **NIE** stosować

- Przykładowe scenariusze, w których zastosowanie baz dokumentowych może być utrudnione, lub nieefektywne obejmują m.in. sytuacje w których występują:
  - Złożone transakcje obejmujące różne operacje
  - Transakcje dla wielu operacji
    - Jeśli zapisujesz wiele kluczy i przy zapisie któregoś wystąpi błąd, to nie ma możliwości automatycznego wycofania pozostałych operacji
  - Zapytania ad hoc na zmiennych strukturach
    - Elastyczny schemat oznacza, że przechowywane dane mogą mieć różną formę

Bazy grafowe

- Model stanowią wierzchołki i krawędzie między nimi
- Wzbogacone o własności i etykiety
- Bezindeksowe sąsiedztwo
  - Bezpośrednie referencje do sąsiadujących elementów
- Np.
  - Neo4J,
  - OrientDB,
  - Titan
  - Allegro Graph

Bazy grafowe– kiedy stosować

- Przykładowe scenariusze, w których sprawdzają się bazy grafowe:
  - Dane połączone
    - Np. Sieci społecznościowe
  - Wytyczanie trasy, wysyłka, usługi oparte o położenie
    - Adresy jako węzły połączone krawędziami z przypisaną odległością lub czasem
    - Rekomendacja punktów znajdujących się w pobliżu
  - Silniki rekomendacji
    - Np. „inni kupili też…”

MongoDB

Mongo DB – wprowadzenie

- 2007 – początki rozwoju – firma 10gen
  - Pierwotnie jako komponent PaaS
- 2009 – open source
  - Plus komercyjne wsparcie, usługi i rozszerzenia
- 2013 – 10gen zmienia nazwę na MongoDB Inc.

- Dostępna na wiele platform baza dokumentowa
- Darmowa w wersji Community
- Dane jako dokumenty BSON (Binary JSON)
  - Możliwe użycie schematów i reguł do walidacji danych

Mongo DB – własności

- Wysoka wydajność i skalowalność
- Elastyczność
- Zapytania ad hoc
  - Wartości i zakresy pól
  - Wyrażenia regularne
  - Możliwe wykonywanie funkcji JavaScript na serwerze
- Wsparcie dla indeksów
- Rozproszenie, replikacja, równoważenie obciążeń
- Agregacja
- Wsparcie dla transakcji ACID

MongoDB a RDB

RDB	MongoDB
Tabela	Kolekcja
Wiersz / rekord	Dokument
Kolumna	pole
Klucz główny	_id
Złączenie	$lookup, zagnieżdżenie dokumentów
Relacja 1:N	Tablica, zagnieżdżenie dokumentów
Relacja M:N	Tablica referencji
Indeks	Indeks

MongoDB – model danych

- Dokumenty BSON
  - BSON – binarny JSON, więcej typów niż w JSON
  - Max 16MB
  - Pole _id może być dodane automatycznie

```
{
        _id: ObjectId("5099803df3f4948bd2f98391"),
        name: {
                        first: "Alan",
                        last: "Turing"
         },
        birth: new Date('Jun 23, 1912'),
        death: new Date('Jun 07, 1954'),
        contribs: [ "Turing machine", "Turing test", "Turingery" ],
        views : NumberLong(1250000)
}
```

MongoDB - shell

- use <database>
- db.myCollection.insertOne({ x: 1 });
- db.myCollection.insertMany(…)
- db.myCollection.find()
  - kryteria filtrowania
- db.myCollection.updateOne()
- db.myCollection.updateMany()
- db.myCollection.replaceOne()
- db.myCollection.deleteOne()
- db.myCollection.deleteMany()

MongoDB - przykłady

```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

collection

field: value

field: value

field: value

document

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

collection

query criteria

projection

cursor modifier

MongoDB – przykłady

```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

collection

update filter

update action

```
db.users.deleteMany(
   { status: "reject" }
)
```

collection

delete filter

MongoDB i Java

```
MongoClient mongoClient = new MongoClient( "localhost" , 27017 );
MongoDatabase database = mongoClient.getDatabase("mydb");
MongoCollection<Document> collection = database.getCollection("test");
Document doc = new Document("name", "MongoDB")
    .append("type", "database")
    .append("count", 1)
    .append("versions", Arrays.asList("v3.2", "v3.0", "v2.6"))
    .append("info", new Document("x", 203).append("y", 102));
```

```
{
 "name" : "MongoDB",
 "type" : "database",
 "count" : 1,
 "versions": [ "v3.2", "v3.0", "v2.6" ],
 "info" : { x : 203, y : 102 }
}
```

MongoDB i Java (c.d.)

```
collection.insertOne(doc);


List<Document> documents = new ArrayList<Document>();
for (int i = 0; i < 100; i++) {
    documents.add(new Document("i", i));
}
collection.insertMany(documents);
System.out.println(collection.count());
Document myDoc = collection.find().first();
System.out.println(myDoc.toJson());
```

```
{ "_id" : { "$oid" : "551582c558c7b4fbacf16735" },
  "name" : "MongoDB",
  "type" : "database",
  "count" : 1,
  "info" : { "x" : 203, "y" : 102 } }
```

MongoDB i Java (c.d.)

```
MongoCursor<Document> cursor = collection.find().iterator();
try {
    while (cursor.hasNext()) {
            System.out.println(cursor.next().toJson());
    }
} finally {
    cursor.close();
}

myDoc = collection.find(eq("i", 71)).first();
System.out.println(myDoc.toJson());
```

Grafowe bazy danych Neo4J

Co to jest graf?

- Graf to struktura matematyczna służąca do przedstawiania i badania relacji między obiektami.
- W uproszczeniu graf to zbiór wierzchołków, które mogą być połączone krawędziami w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków
- Formalnie:
  - G=(V, E), gdzie
  - V to zbiór wierzchołków
  - E to zbiór krawędzi, taki że $E \subseteq (V \times V)$

Grafowe bazy danych

- Bazy danych wykorzystujące struktury grafów z węzłami, krawędziami i własnościami do przedstawiania i przechowywania danych oraz do obsługi zapytań semantycznych.
- System pamięci masowej, który zapewnia bezindeksowe sąsiedztwo
  - każdy element bazy zawiera bezpośredni wskaźnik na sąsiadujące elementy i nie jest konieczne wyszukiwanie indeksowe.

Kiedy stosować grafowe bazy danych?

- Na pewno warto rozważyć stosowanie GDB pracując przy:
  - Sieciach społecznościowych
  - Systemach rekomendacji
  - Korelacji danych z różnych źródeł
  - Wykrywaniu nadużyć i oszustw
  - Zarządzaniu zasobami

Kiedy stosować grafowe bazy danych?

- Ogólnie, warto rozważyć stosowanie GDB gdy:
  - Mamy dużo relacji wiele-do-wielu
  - Powiązania między elementami są tak samo ważne lub nawet ważniejsze niż dane
  - Wydajność bazy relacyjnej jest niewystarczająca przy wykonywaniu skomplikowanych zapytań z dużą ilością złączeń

Neo4J

Neo4J

- Wysoko skalowalne oprogramowanie open source,
- obsługuje ACID,
- zapewnia klastry o wysokiej dostępności
- wyposażone w internetowe narzędzie administracyjne, które obejmuje pełne wsparcie transakcji i wizualny eksplorator wykresów łącza węzłów;
- dostępne z poziomu większości języków programowania za pomocą wbudowanego interfejsu API REST Web API oraz protokołu Bolt
- Najpopularniejsza grafowa baza danych stosowana w 2017 r.

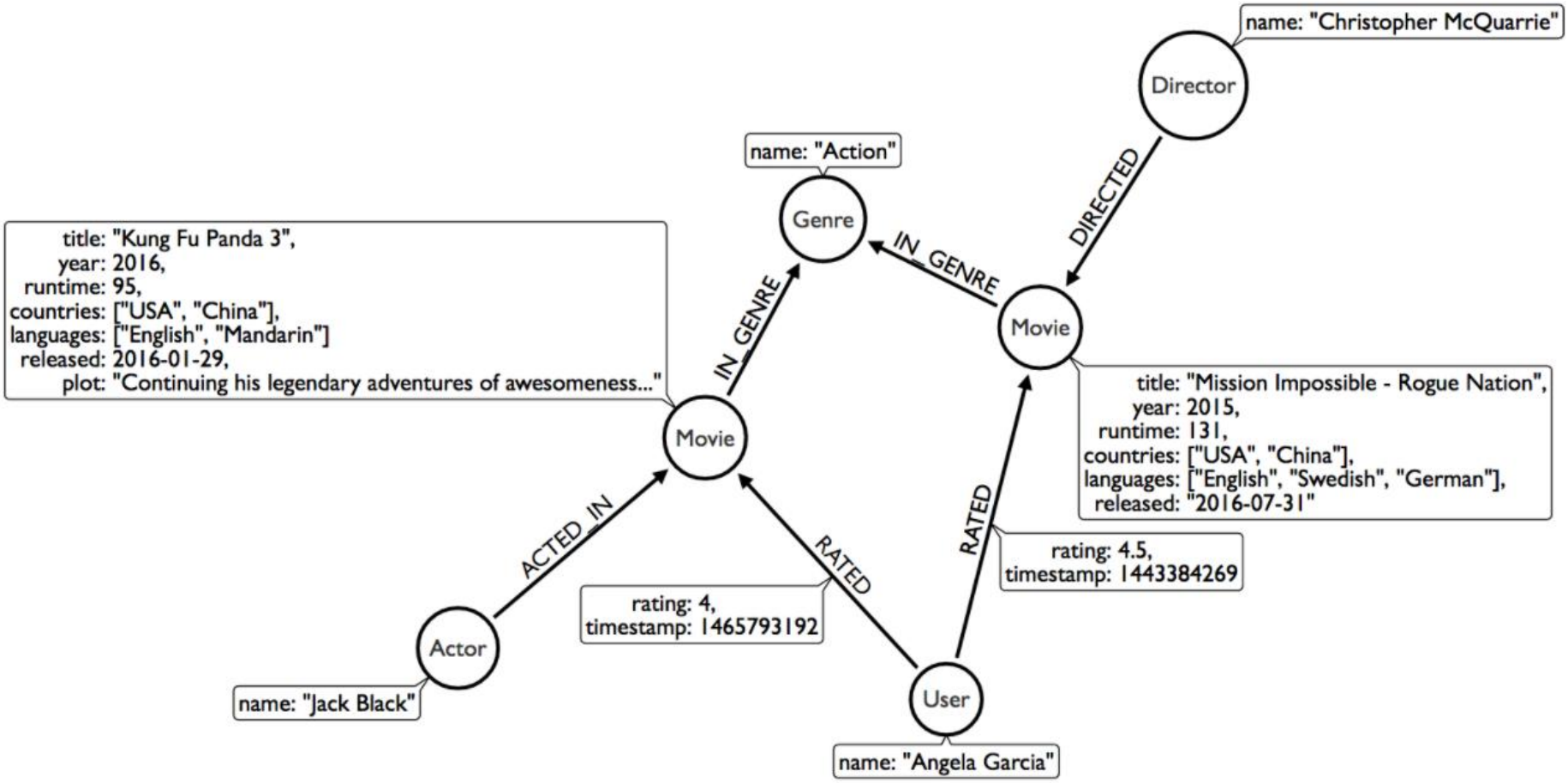
Język zapytań Cypher

- Deklaratywny
- Pozwala na wyszukiwanie oraz manipulację danymi
- Oparty na modelu grafowym rozszerzonym o właściwości i etykiety,
- Węzły mogą mieć zero lub więcej etykiet, a każda relacja (krawędź) ma dokładnie jeden typ.
- Węzły i relacje mają także zero lub więcej właściwości,
- Właściwość jest parą klucz-wartość; kluczowi o danej nazwie przypisana jest wartość z systemu typów Cypher,
  - node, relationship, path, map, list, integer, floating-point number, boolean, string

Język zapytań Cypher - składnia

- Składnia podobna trochę do SQL
- Najważniejsze klauzule to
  - MATCH – opisuje poszukiwany wzorzec
  - WHERE – dodatkowe warunki
  - RETURN – specyfikuje co ma być zwrócone
  - WITH – pozwala łączyć zapytania
  - CREATE – tworzy węzeł lub relację
  - DELETE – usuwa węzeł lub relację
  - SET – ustawia własności lub dodaje etykiety
  - REMOVE – usuwa własności lub etykiety
  - MERGE – zapewnia, że wzorzec istnieje w grafie
    - Uzupełnia brakujące elementy wzorca

Przykład – fragment bazy filmów



Odwołania do węzłów

Odwołania do krawędzi

- --
- --> lub <--
- -[:LABEL]->
- -[rel:LABEL]->
- -[{blocked: false}]->
- Typowe zapytania polegają na wskazaniu szukanego wzorca,
- Najczęściej są to wierzchołki połączone krawędziami, ew. dodatkowo scharakteryzowane własnościami

Obojętnie w którą stronę

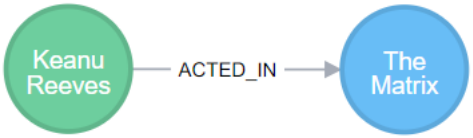
Przykład zapytania

```
MATCH (m:Movie)<-[:RATED]-(u:User)
 WHERE m.title CONTAINS "Matrix"
 WITH m.title AS movie, COUNT(*) AS reviews
 RETURN movie, reviews
 ORDER BY reviews DESC LIMIT 5;
```

5 tytułów filmów z serii „Matrix”
Z największą liczbą ocen

Przykład zapytania

```
MATCH (matrix:Movie {title:"The Matrix"} )
<-[role:ACTED_IN {roles:["Neo"]}]-
(keanu:Person {name:"Keanu Reeves"})
RETURN matrix, role, keanu
```



Zapytanie zwraca 2 węzły połączone krawędzią

matrix	role	keanu
{"tagline":"Welcome to the Real World","title":"The Matrix","released":1999}	{"roles":["Neo"]}	{"born":1964,"name":"Keanu Reeves"}

Scieżki (paths)

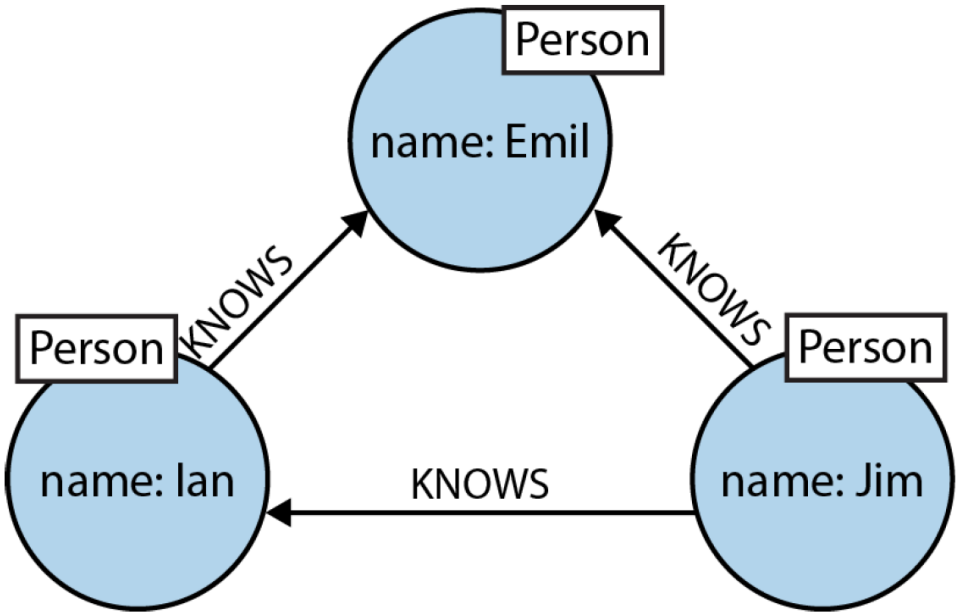
- Sekwencje naprzemiennych węzłów i krawędzi
- (a)-->(b)
- ()-[*]->()
- ()-[*2]->()
- ()-[*1..4]->()
- ()-[*2..]->()
- ()-[*..5]->()
- Wzorzec ścieżki musi się zaczynać i kończyć węzłem!

Dowolna liczba krawędzi! Ostrożnie z tym

Dwie krawędzie

Od 1 do 4 krawędzi

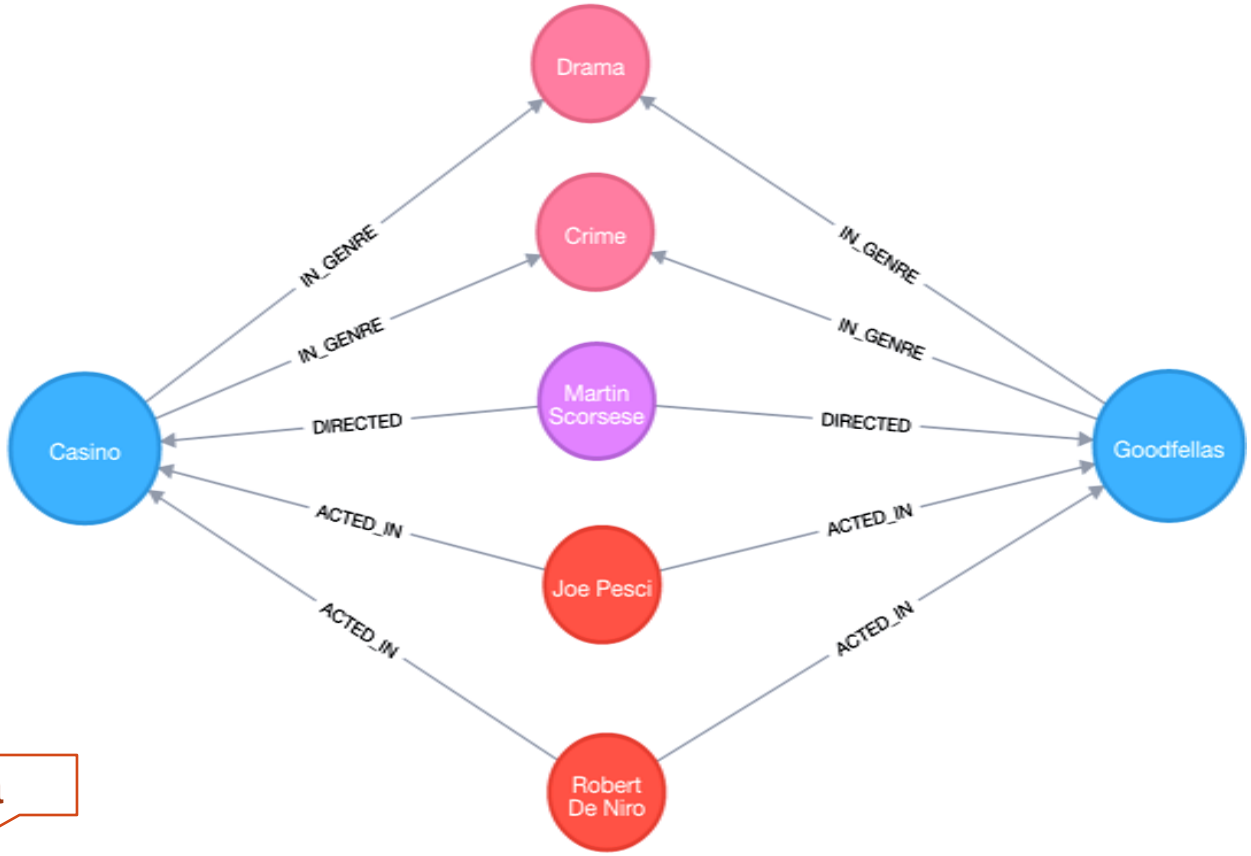
Przykład



(emil)<-[:KNOWS]-(jim)-[:KNOWS]->(ian)-[:KNOWS]->(emil)

(emil:Person {name:'Emil'})
<-[:KNOWS]-(jim:Person {name:'Jim'})
-[:KNOWS]->(ian:Person {name:'Ian'})
-[:KNOWS]->(emil)

Przykład



Zmienna

```
MATCH p=(m:Movie {title: "Net, The"})
-[:ACTED_IN|:IN_GENRE|:DIRECTED*2]-()
RETURN p LIMIT 25
```

Klauzula RETURN

- Definiuje co ma być zwrócone jako rezultat zapytania
- Może występować tylko raz w całym zapytaniu
- Pozwala na zwracanie węzłów, krawędzi, ścieżek, własności, lub dowolnych wyrażeń
- Wartości mogą być przemianowane za pomocą AS
  - Przykład na kolejnym slajdzie

Pytanie

- Co zwraca poniższe zapytanie?

```
MATCH (a:Person {name:'Jim'})-[:KNOWS]->(b)-[:KNOWS]->(c),
      (a)-[:KNOWS]->(c)
RETURN b, c
```

Przykład

```
1 MATCH (n)
2 RETURN n, "node " + id(n) +" is " +
3 CASE
4   WHEN n.title IS NOT NULL THEN "a Movie"
5   WHEN EXISTS(n.name) THEN "a Person"
6   ELSE "something unknown"
7 END AS about
```

"n"	"about"
{"name":"Anna"}	"node 2 is a Person"
{}	"node 20 is something unknown"
{"title":"The Matrix","tagline":"Welcome to the Real World","released":1999}	"node 21 is a Movie"
{"name":"Keanu Reeves","born":1964}	"node 22 is a Person"
{"name":"Carrie-Anne Moss","born":1967}	"node 23 is a Person"

Klauzula OPTIONAL MATCH

- Podobnie jak MATCH służy do specyfikowania szukanego wzorca
- Ale zwraca również niekompletne dopasowanie
  - Wstawia NULL zamiast brakujących elementów
- Podobnie jak OUTER JOIN w SQL

```
MATCH (a:Movie)
OPTIONAL MATCH (a)<-[:WROTE]-(x)
RETURN a.title, x.name
```

```
| (a:Movie) OPTIONAL MATCH (a)<-[:WROTE]-(x) RETURN a.
```

"a.title"	"x.name"
"The Matrix"	null
"The Matrix Reloaded"	null
"The Matrix Revolutions"	null
"The Devil's Advocate"	null
"A Few Good Men"	"Aaron Sorkin"
"Top Gun"	"Jim Cash"
"Jerry Maguire"	"Cameron Crowe"

Klauzula WHERE

- Użyta po (OPTIONAL) MATCH dodaje warunki do wzorca, staje się częścią wzorca
- Użyta po WITH tylko filtruje wyniki

- MATCH (n) WHERE n.name =~ 'A.*' RETURN n

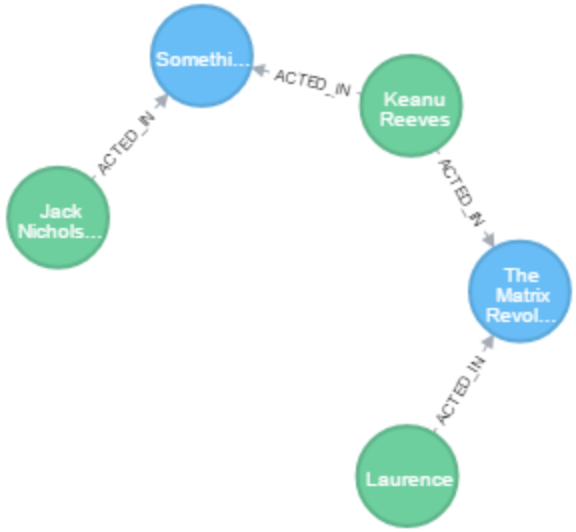
- Operator =~
  - Dopasowanie wyrażenia regularnego
  - Składnia Java regex

Najkrótsze ścieżki

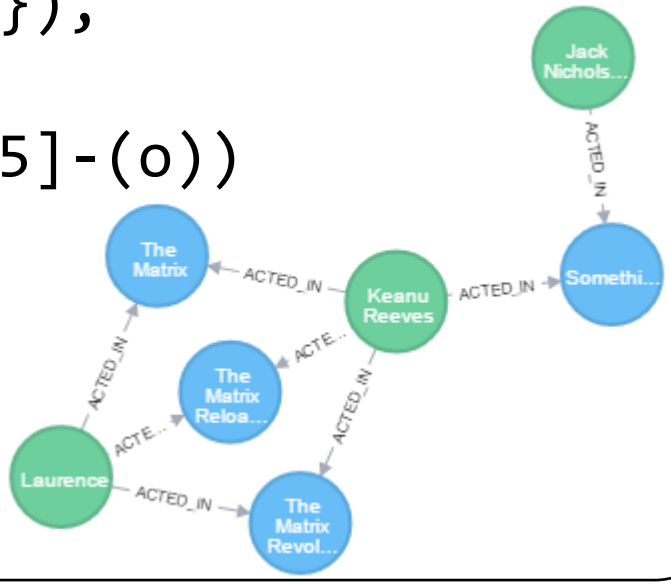
- Chodzi o ścieżki pomiędzy dwoma węzłami o minimalnej liczbie krawędzi
- Funkcja shortestPath lub allShortestPaths zaaplikowana do wzorca ścieżki
- Można podać dodatkowe predykaty w klauzuli WHERE
  - Uniwersalne (NONE lub ALL) są sprawdzane podczas szukania ścieżki
  - Inne predykaty są sprawdzane dopiero po odszukaniu ścieżki
- Dwie metody wyszukiwania
  - Dwukierunkowy BFS (predykaty uniwersalne lub brak predykatów)
  - DFS – wolniejszy – dla ścieżek z pozostałymi predykatami

Przykład

```
MATCH
  (m { name:"Laurence Fishburne"}),
  (o{name:"Jack Nicholson"}),
  p = shortestPath((m)-[*..15]-(o))
RETURN p
```



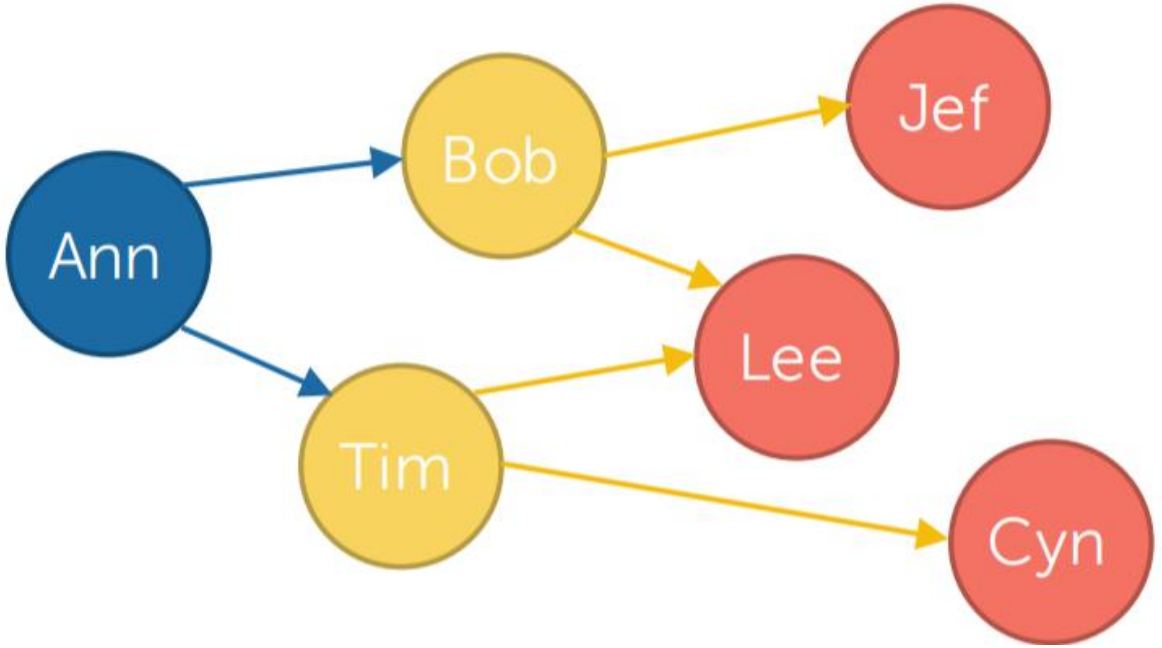

```
MATCH
  (m { name:"Laurence Fishburne"}),
  (o{name:"Jack Nicholson"}),
  p = allShortestPaths((m)-[*..15]-(o))
RETURN p
```



Agregacja

- Analogicznie do SQL-owego GROUP BY
  - W języku Cypher mamy niejawne grupowanie
- W klauzuli RETURN lub WITH
- DISTINCT
- COUNT, SUM, AVG, MIN, MAX,
- STDEV, STDEVP, PERCENTILEDISC, PERCENTILECONT
- COLLECT – składa wartości w listę

Przykład




```
MATCH (me:Person {name:'Ann'})-->(friend:Person)-->(friend_of_friend:Person)
RETURN me.name, count(DISTINCT friend_of_friend), count(friend_of_friend)
```

me	COUNT DISTINCT	COUNT
Ann	3	4

Składanie zapytań za pomocą WITH

- Działa podobnie jak operator potoku
- Pozwala łączyć ze sobą zapytania
- Przekazuje wynik jednego zapytania jako wejście do następnego
- WITH oblicza wynik (włącznie z agregacją) zanim zostanie on przekazany dalej

- Filtrowanie agregatów

```
MATCH (p)-[:PLAYS]->(t)
WITH t, AVG(p.age) AS a
WHERE a < 25
RETURN t
```

Drużyna piłkarska o średniej wieku mniejszej niż 25

Agregacja, WITH c.d.

- Agregacja agregatów

```
MATCH (p)-[:PLAYS]->(t)
WITH t, MIN(p.age) AS a
RETURN AVG(a)
```

Średni wiek najmłodszych graczy z wszystkich drużyn

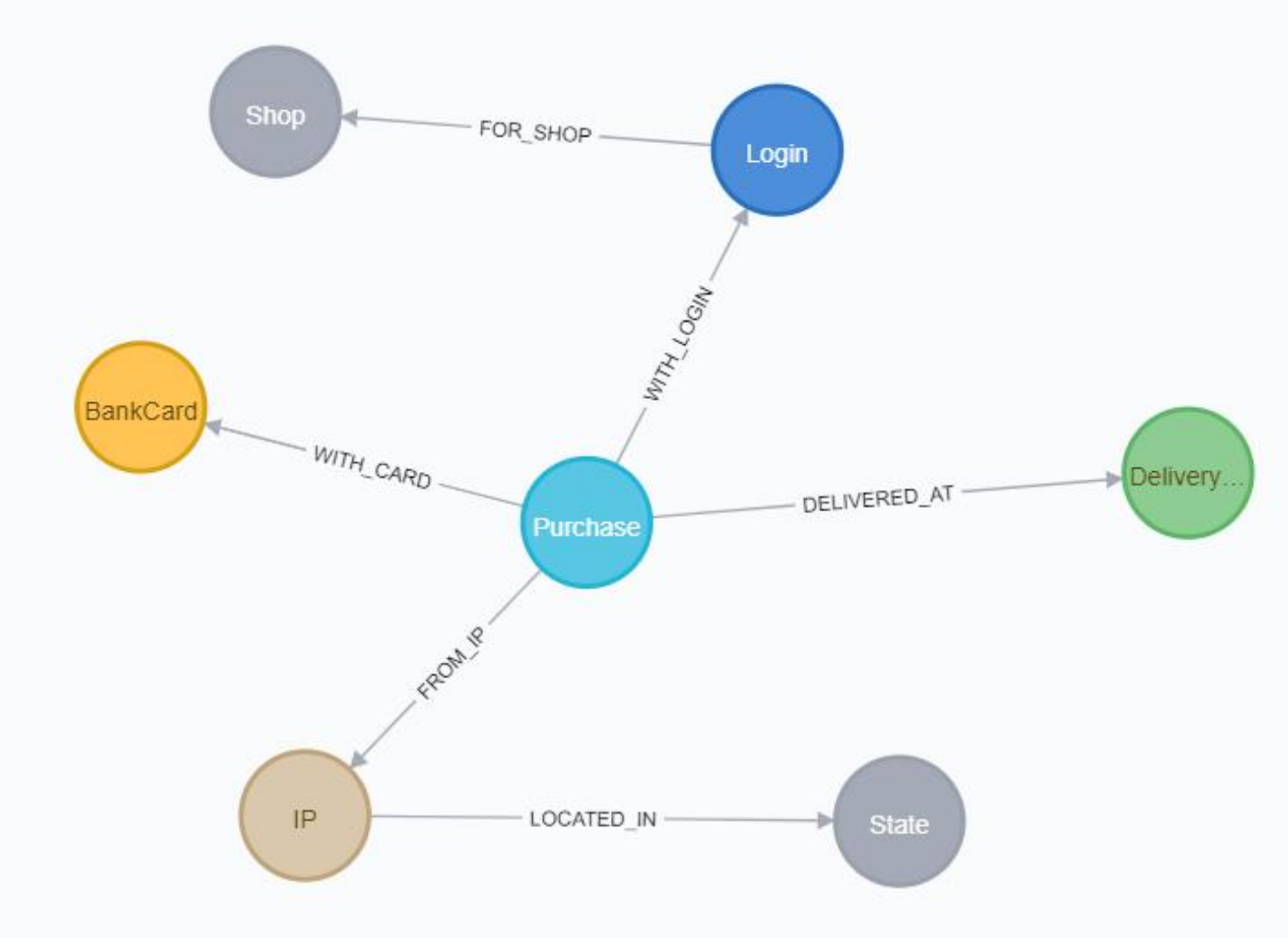
- Ograniczenie przestrzeni przeszukiwań na podstawie kolejności obliczania wartości lub agregatów

```
MATCH (p)-[f:FRIENDS]->(p2)
WITH f,p2 ORDER BY f.rating DESC LIMIT 5
MATCH (p2)-[f:FRIENDS]->(p3)
RETURN DISTINCT p3
```

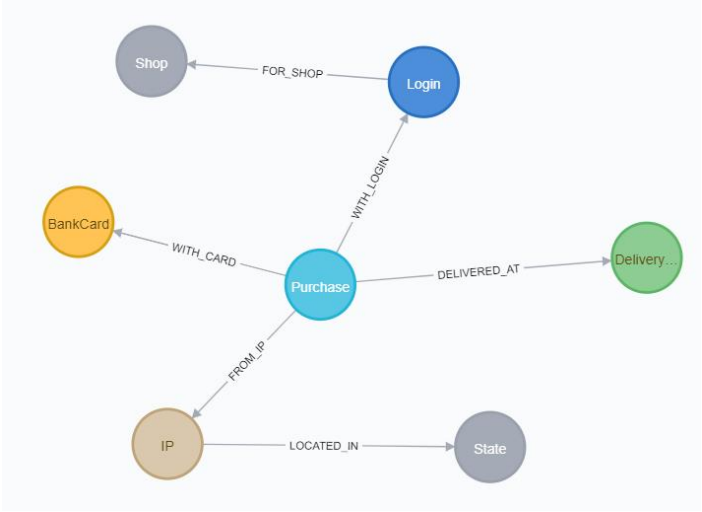
Przyjaciele pięciu najlepszych przyjaciół

Neo4J – przykłady zapytań Cypher

Struktura przykładowej bazy



Struktura przykładowej bazy



Wierzchołki i ich właściwości:

IP {ip}
State {name}
Shop {business, name, shopID}
Login {firstName, lastName, mail, startDate, userid}
BankCard {accountNumber, balance, expirationDate, latePayments, limit}
Purchase {amount, date, purchaseCode, time}
DeliveryAddress {city, state, streetAddress, zip}

Zadania

- Napisz zapytania w języku Cypher, które zwrócą:
  - pięć najdroższych transakcji: ich kwoty oraz nazwiska osób, które dokonały zakupu
  - nazwiska pięciu osób, które wykonały najwięcej transakcji, liczbę tych transakcji i ich średnie kwoty
  - nazwiska pięciu osób, które wykonały zakupy na (w sumie) największą kwotę i sumy tych transakcji
  - nazwiska pięciu osób, które użyły największej liczby kart oraz liczbę użytych kart
  - nazwa najpopularniejszego sklepu w każdym stanie (z największą liczbą transakcji)