Napredne arhitekture informacionih sistema

Grafske baze podataka

Izvođači nastave:
dr Marko Vještica
Elena Akik
Sanja Radić

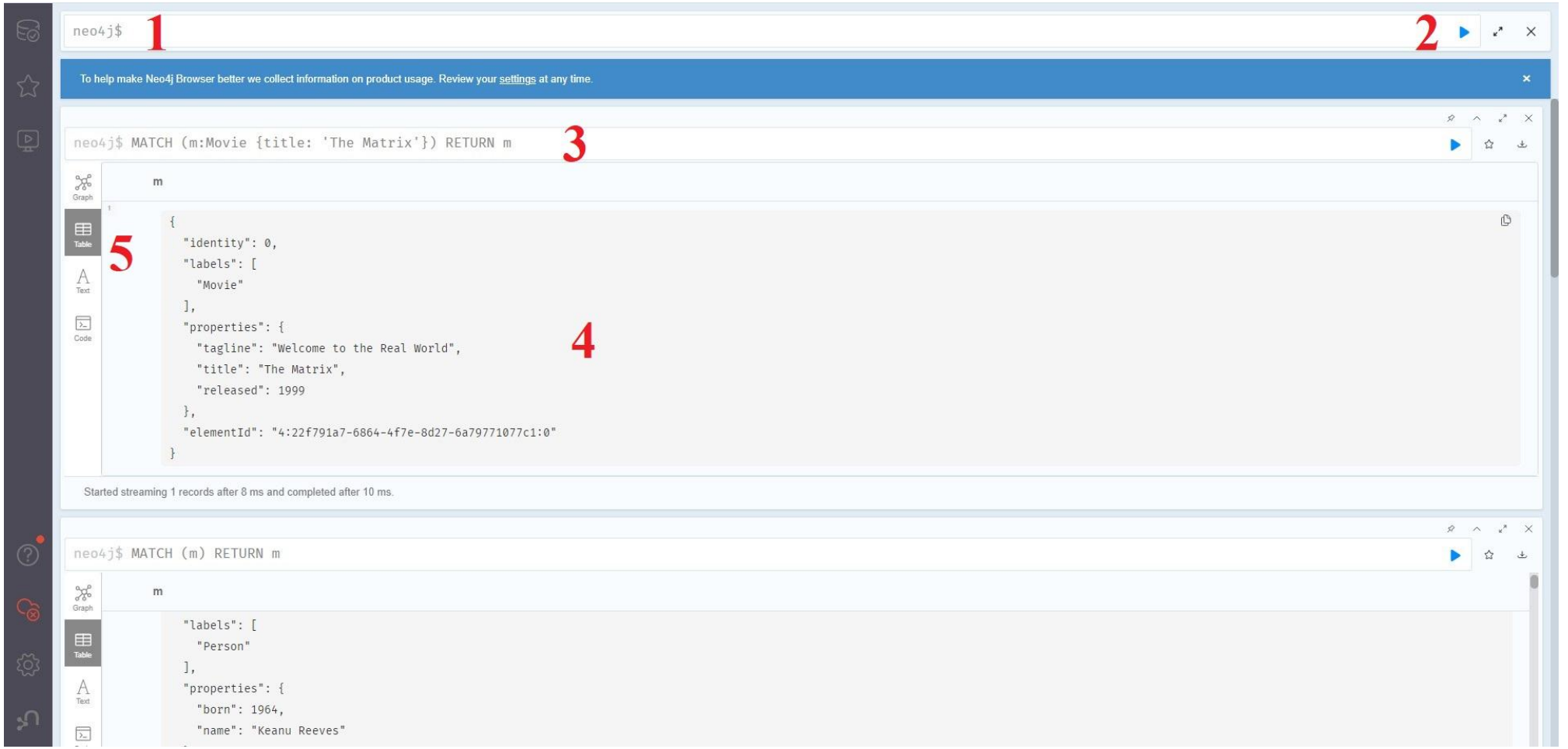
DAIS
Data Science and Information Systems Group

Sadržaj

Softverska podrška

- *Neo4j Database with Neo4j client (Neo4j desktop)*
  - *https://neo4j.com/download/*
- Drugi način korišćenja *Neo4j* je pomoću pokretanja *docker-compose* fajla koji se nalazi na repozitorijumu
  - Link do fajla: *http://www.acs.uns.ac.rs/sr/node/237/10768907*
  - Klijentskoj *Neo4j* aplikaciji moguće je pristupiti na adresi: *http://localhost:7474*
    - username: neo4j
    - password: password
  - Za potrebe vežbi uraditi *copy/paste* ili *drag-and-drop* skripte *movies_dump.cypher.txt* u okviru klijentske aplikacije i pokrenuti skriptu

Softverska podrška



1) Linija za navođenje naredbi

2) Dugme za izvršavanje naredbi

3) Prethodno izvršena naredba

4) Rezultat izvršene naredbe

5) Format prikazivanja rezultata

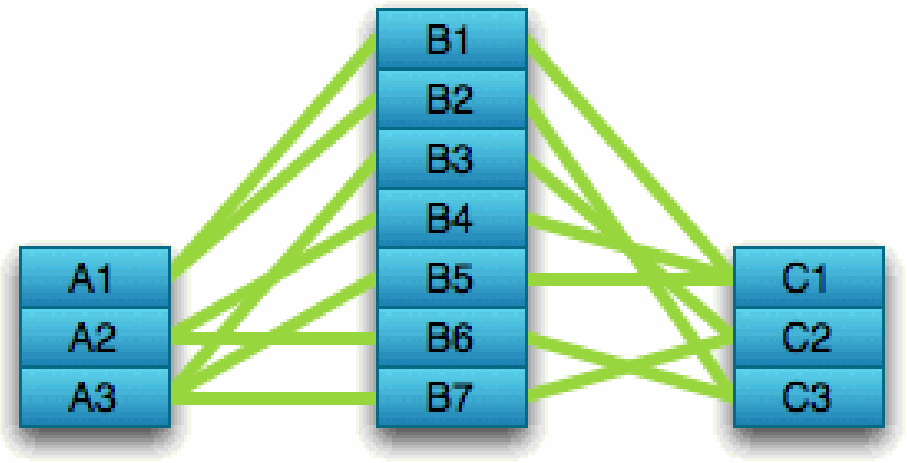
Sadržaj

Uvod u grafske baze podataka

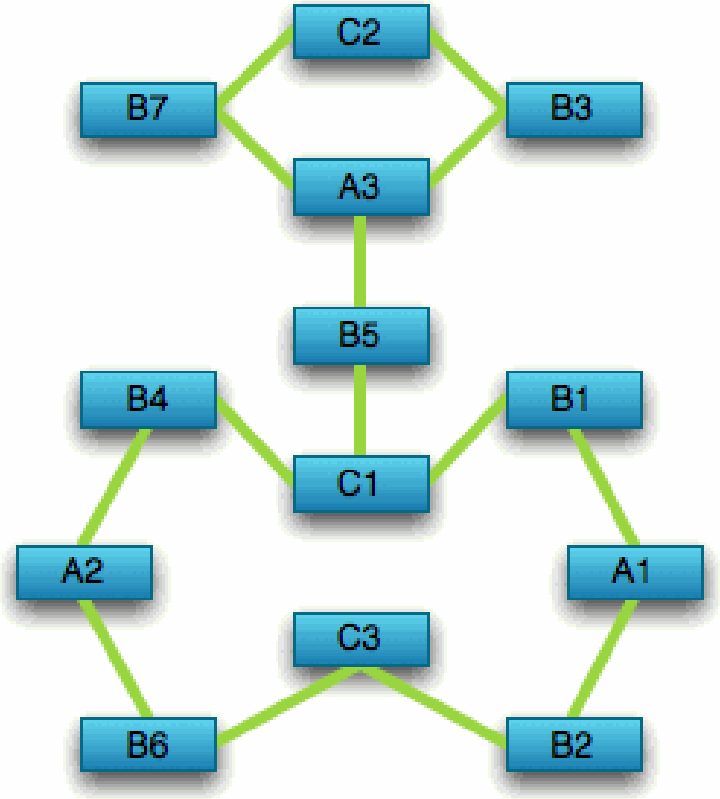
- **Grafske baze podataka** predstavljaju podatke kao čvorove, veze i osobine, omogućavajući efikasno upravljanje složenim odnosima
- Prva poznata grafska baza podataka nazvana je *Hierarchical Model*, razvijena od strane *IBM*-a, 1960tih
- Veliku popularnost stiču tokom 2000-ih godina, sa pojavom modernih grafskih baza podataka kao što su *Neo4j, Titan i OrientDB*

Poređenje relacione i grafske baze podataka

- Organizacija **relacionih** baza

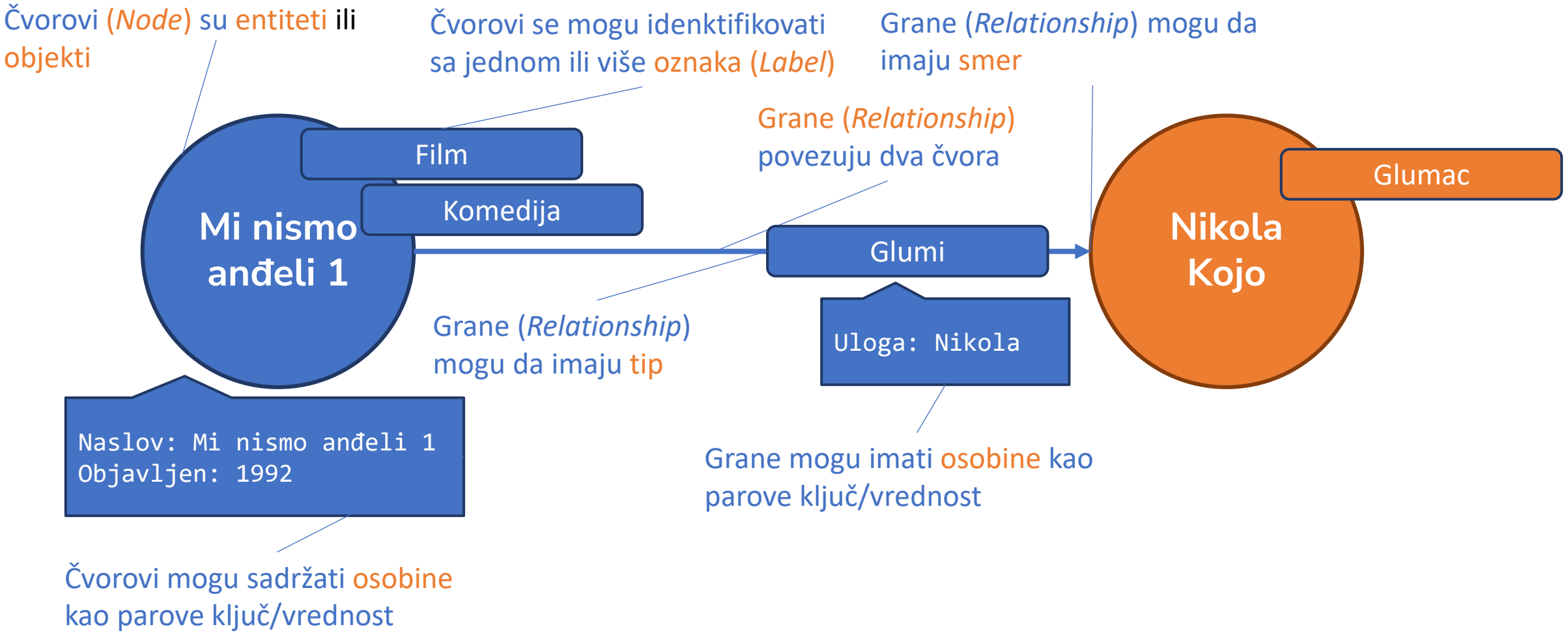


- Organizacija **grafskih** baza



Sadržaj

Struktura grafske baza



Sadržaj

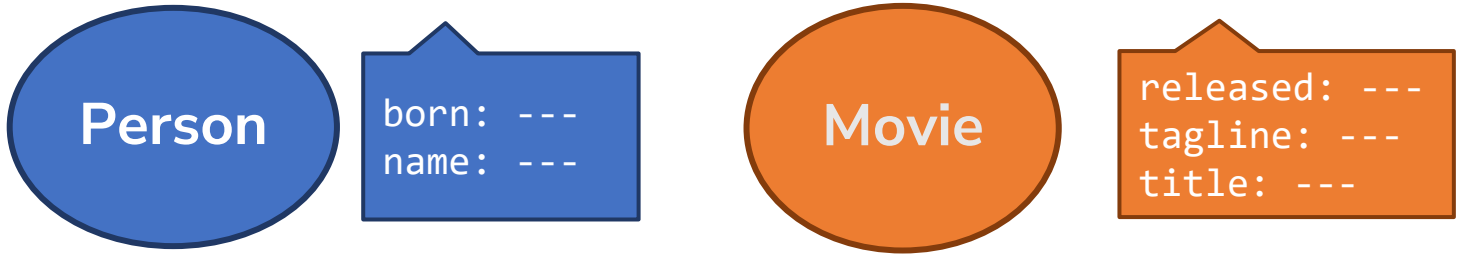
- Softverska podrška
- Uvod u grafske baze podataka
- Struktura grafske baze podataka
- *Neo4j* i *Cypher* jezik
- Upitni jezik *MATCH* naredba
- Jezik za manipulaciju podataka
- *Spring Data Neo4j*
- Algoritmi preporuke korišćenjem grafskih baza podataka
- Korisni linkovi

Neo4j i Cypher

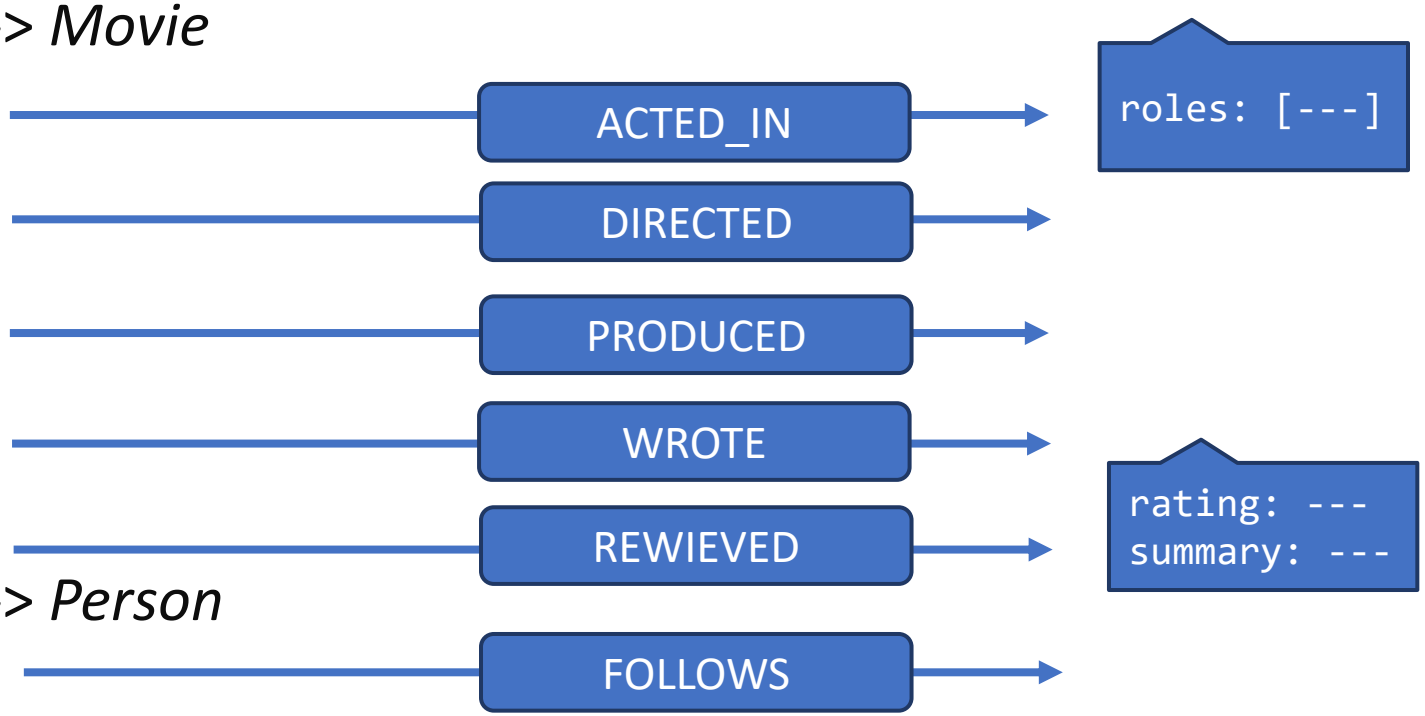
- ***Neo4j*** je pionir u domenu grafskih baza podataka, razvijen je od strane švedske kompanije *Neo Technology*
- Uporedo sa razvojem *Neo4j* razvijen je i *Cypher* jezik
- ***Cypher*** je deklarativni jezik koji omogućava identifikovanje obrazaca u podacima koristeći sintaksu u stilu ASCII umetnosti koja se sastoji od zagrada, crtica i strelica
- ***Cypher*** se može podeliti u dva podjezika:
  - Upitni jezik – MATCH naredba
  - Jezik za manipulaciju podacima

Primer – grafski model baze podataka

- Čvorovi:



- Veze: *Person -> Movie*



- Veze: *Person -> Person*

Sadržaj

MATCH klauzula

- ***MATCH*** klauzula – omogućava pronalaženje čvorova, veza i obrazaca unutar podataka

prosti uslov selekcije

```
1  MATCH (<promenljiva> [:<labela_čvora>] [{<obeležje: 'vrednost'>}])
   RETURN <promenljiva>
```

```
1  MATCH (<promenljiva> [:<labela_čvora>])
   WHERE <kompleksni_uslov_selekcije>
   RETURN <promenljiva>
```

- `<promenljiva>` - unutar promenljive smeštamo rezultate nakon selekcije; a ukoliko ne koristimo promenljivu moguće je izostaviti
- `[:<labela_čvora>] [{<obeležje: 'vrednost'>}]` – prosti uslov selekcije u kome je moguće porediti da li neko obeležje ima tačno navedenu vrednost
- **`WHERE`** `<kompleksni_uslov_selekcije>` - kompleksni uslov selekcije omogućuje da se vrše poređenja koje ne zahtevaju samo operator '=', nego je moguće koristiti i ostale operatore
- **`RETURN`** – naredba koja ispisuje rezultat korisniku, ujedno mora biti i poslednja naredba u *Cypher* upitu

MATCH klauzula – zadatak za vežbu

- Prikazati sve podatke

```
1  MATCH (n)
   RETURN n
```

- Prikazati sve čvorove koji imaju labelu *Movie*

```
1  MATCH (m: Movie)
   RETURN m
```

- Prikazati čvor koji ima labelu *Movie* i naziv (*title*) *The Matrix*

```
1  MATCH (m: Movie {title: 'The Matrix'})
   RETURN m
```

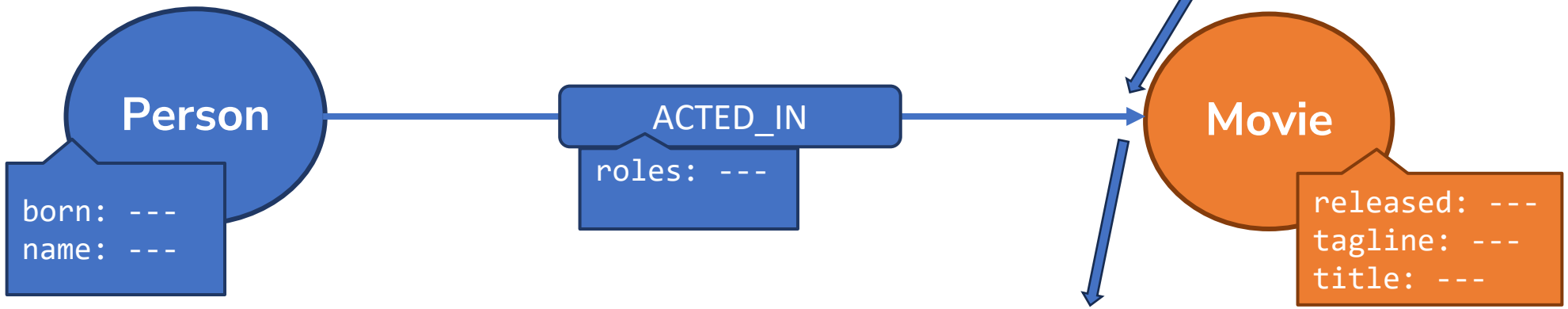
MATCH klauzula - povezivanje čvorova

```
1    MATCH ([<promenljiva>]: <labela_čvora>) - [[<promenljiva1>]: <labela_grane>]
     -> ([<promenljiva2>]: <labela_čvora>])
     RETURN <lista_promenljivih>
```

- Prikazati sve glumce koji su glumili u nekom filmu

Napomena:
Usmerenje iz podataka se prenosi i na upit

Primer:




```
1    MATCH (p: Person) – [a: ACTED_IN] -> (m: Movie)
     RETURN p, a, m
```

MATCH klauzula – zadatak za vežbu

- Prikazati ime (*name*), godinu rođenja (*born*) glumaca kao i naziv (*title*) filma u kom glume

```
1  MATCH (p: Person) - [a: ACTED_IN] -> (m: Movie)
   RETURN p.name, p.born, m.title
```

MATCH klauzula – zadatak za vežbu

- Prikazati sve filmove u kojima je glumio glumac *Keanu Reeves*

```
1  MATCH (:Person {name: 'Keanu Reeves'}) - [a: ACTED_IN] -> (m: Movie)
   RETURN m
```

Napomena: Ukoliko ne koristimo promenljive moguće je ostaviti i prazno mesto

- Prikazati glumca koji je imao ulogu *John Milton*

(**Napomena**: Da bismo poredili nizove koristiti *IN* I *NOT IN* operatore).

```
1  MATCH (p: Person) - [a: ACTED_IN] -> (m: Movie)
   WHERE 'John Milton' IN a.roles
   RETURN p
```

MATCH klauzula – zadatak za vežbu

- Prikazati glumce koji su glumili u istim filmovima kao i glumac *Al Pacina*

```
1  MATCH (p: Person) – [:ACTED_IN] -> (m: Movie) <- [:ACTED_IN] – (p1:Person)
   WHERE p.name = 'Al Pacino'
   RETURN p1
```

Napomena: Zbog toga što radimo poređenje sa operatorom jednako, mogli smo upotrebiti i prosti uslov selekcije

```
1  MATCH (p: Person) - [:ACTED_IN] -> (m: Movie)
   MATCH (m) <- [:ACTED_IN] - (p1: Person)
   WHERE p.name = 'Al Pacino' AND p <> p1
   RETURN p1
```

Napomena: U prvom upitu radimo samo jedan obilazak grafa (jednom pristupamo podacima), dok u drugom sa svakom MATCH naredbom ponovo pristupamo podacima, te je zbog toga prvi pristup bolji što se tiče performansi. Takođe zbog ponovnog pristupanja podacima moramo da isključimo čvorove iz prvog poređenja (p <> p1)

MATCH klauzula – *GROUP BY* – Primer

- ***Group by*** operacija unutar *Cypher* jezika se radi impicitno pomoću funkcija za agregaciju unutar *RETURN/WITH* klauzule. Obeležja nad kojim nije primenjena ni jedna funkcija za agregaciju će postati ključ za *GROUP BY* operaciju.
- Moguće je uraditi grupisanje i po obeležju i po celoj promenljivoj
- Prikazati broj filmova po godini – grupisanje po obeležju

```
1  MATCH (m: Movie)
   RETURN m.released as YEAR, count(m) AS NumberOfMovies
```

- Za svakog glumca prikazati ukupan broj filmova u kojima je igrao – grupisanje po promenljivoj

```
1  MATCH (p: Person) - [:ACTED_IN] -> (m: Movie)
   RETURN p as Actor, COUNT(m) AS NumberOfMovies
```

MATCH klauzula – *GROUP BY* – zadatak za vežbu

- Prikazati ime glumca, naziv filma i broj drugih glumaca na tom filmu

```
1  MATCH (p: Person) - [:ACTED_IN] -> (m: Movie) <- [:ACTED_IN] - (p1: Person)
   RETURN p.name, m.title, count(p1) as NumberOfActors
```

Napomena: Nije nam potrebno dodavati -1 jer osoba ***p*** svakako nije obuhvaćena

```
1  MATCH (p: Person) - [:ACTED_IN] -> (m: Movie)
   MATCH (m) <- [:ACTED_IN] - (p1: Person)
   RETURN p.name, m.title, count(p1) - 1 as NumberOfActors
```

Napomena: Moramo dodati -1 kako ne bismo brojali i osobu za koju prikazujemo broj glumaca

MATCH klauzula – *WITH* – Primer

- ***WITH*** klauzula omogućava sledeće:
  - **Prenos rezultata** – prenos rezultata iz prethodnog koraka upita u naredni korak
  - **Filtritranje** – može se koristiti da bi se smanjio skup podataka koji se kasnije obrađuju
  - **Trasformacije** – omogućava oblikovanje rezultata na određeni način, dodajući ili izbacujući određene podatke
  - **Grupisanje** – može se koristiti zajedno sa implicitnom *GROUP BY* klauzulom kako bi se grupisali rezultati po određenom kriterijumu, ili filtrirale grupe (slično *HAVING* klauzuli u jeziku *SQL*)
- Primer: Prikazati filmove koji su imali preko 5 glumaca

```
1  MATCH (m: Movie) <- [:ACTED_IN] - (p: Person)
   WITH m, count(p) AS NumOfActors
   WHERE NumOfActors > 5
   RETURN m.title AS Movie, NumOfActors
```

MATCH klauzula – *WITH* – zadatak za vežbu

- Prikazati naziv i prosečnu ocenu za svaki film čija prosečnu ocena je veća od 80 (**Napomena**: koristiti polje *rating* iz veze *REVIEWED*).

```
1  MATCH (m: Movie) <- [r: REVIEWED] - ()
   WITH m, avg(r.rating) AS averageRating
   WHERE averageRating > 81
   RETURN m.title AS Movie, averageRating AS AverageRating
```

Napomena: Ukoliko nije bitan čvor nego samo grana, moguće je ostaviti i prazne zagrade

Sadržaj

CREATE klauzula

- ***CREATE*** – dodavanje novog čvora

```
1 CREATE ([<promenljiva>]: <labela_čvora> [{<obeležje: 'vrednost'>}]
```

- ***CREATE*** – dodavanje nove grane

```
1 MATCH (<promenljiva1>: <labela_čvora1>), (<promenljiva2>: <labela_čvora2>)
  WHERE <uslovi_za_pronalazak_prvog_i_drugog_čvora>
  CREATE (promenljiva1) - [:TIP_RELACIJE [{<obeležje>: 'vrednost'>}]] ->
  (promenljiva2)
```

CREATE klauzula – primer

- ***CREATE*** – dodavanje novog čvora

```
1  CREATE (Kojo: Person {name: 'Nikola Kojo', born: 1967})
2  CREATE (Mare: Person {name: 'Milan Maric', born: 1990})
3  CREATE (Toma: Movie {released: 2020, title: 'Toma'})
```

Napomena:
Kojo, Mare, Toma su proizvoljni nazivi promenljivih i nisu neophodni

- ***CREATE*** – dodavanje nove grane

```
1  MATCH (p: Person), (m: Movie)
   WHERE p.name = 'Milan Maric' AND m.title = 'Toma'
   CREATE (p) - [:ACTED_IN {roles: ['Toma Zdravkovic']}] -> (m)
```

DELETE klauzula

- ***DELETE*** – brisanje podataka

```
1  MATCH (<promenljiva> [: <labela_čvora> [{<obeležje>: 'vrednost'}] ])
   DELETE <promenljiva>
```

prosti uslov selekcije

DELETE klauzula – primer

- ***DELETE*** – brisanje čvora

```
1  MATCH (m: Movie {title: 'The Matrix'})
   DELETE m
```

```
1  MATCH (m: Movie {title: 'A Few Good Men'})
   DELETE m
```

- ***DELETE*** – brisanje svih čvorova i grana

```
1  MATCH (m)
   DETACH DELETE m
```

Napomena: Kako bismo obrisali sve čvorove i grane, moramo da prvo raskinemo sve postojeće veze i to radimo pomoću *DETACH* klauzule

SET klauzula – ažuriranje podataka

- ***SET*** – ažuriranje podataka

```
1 MATCH (<promenljiva>: <labela čvora> [{<obeležje>: 'vrednost'}])
  SET <promenljiva>.<obeležje> = 'nova_vrednost'
```

SET klauzula – primer

- ***SET*** – ažuriranje podataka

```
1  MATCH (m: Movie {title: 'The Matrix'})
   SET m.released = 1999
```

```
1  MATCH (p: Person {name: 'Keanu Reeves'})
   SET p.born = 1964
```

Zadatak za vežbu

- Kreirati novi čvor čiji je naziv labele *Genre,* i koja ima obeležje *name*, a čija je vrednost *Thriller.* Nakon kreiranja navedenog čvora, potrebno je film *The Matrix* povezati sa ovim čvorom putem veze čija je labela *IN_GENRE*

```
1  CREATE (g: Genre {name : 'Thriller'})
2  MATCH (g: Genre), (m: Movie)
   WHERE g.name = 'Thriller' and m.title = 'The Matrix'
   CREATE (m) - [:IN_GENRE] -> (g)
```

```
1  CREATE (g: Genre {name: 'Thriller'})
   WITH g
   MATCH (m: Movie {title: 'The Matrix'})
   CREATE (m) - [:IN_GENRE] -> (g)
```

Sadržaj

Maven zavisnost

- Za potrebe povezivanja aplikacije *Spring Boot* i baze podataka *Neo4j*, upotrebljena je **zavisnost** (engl. *Dependency*) ***Spring Data Neo4j***
- Prednosti:
  - Jednostavno dodavanje podrške za *Neo4j* u aplikaciju *Spring Boot*
  - Automatska konfiguracija i upravljanje *bean*-ovima i vezama sa bazom podataka *Neo4j*
  - Omogućava brži razvoj aplikacija koje koriste bazu podataka *Neo4j*

```
1 <dependency>
2     <groupId>org.springframework.boot</groupId>
3     <artifactId>spring-boot-starter-data-neo4j</artifactId>
4 </dependency>
```

Mapiranje čvora na klase pomoću *Spring Data Neo4j*

- Anotacija @*Node* iznad deklaracije *Java* klase omogućuje označavanja da je ta klasa entitet odnosno čvor unutar baze podataka *Neo4j*

```
1 @Node
2 public class Person {
3     @Id @GeneratedValue
4     private Long id;
5     private String name;
6     private Integer born;
7 }
```

- @*Id* omogućava označavanje da je neko polje ključ tipa entiteta;
  Ukoliko se rukuje sa nativnim ključem u *Neo4j*, tip podatka mora biti *Long*

Mapiranje grana na klase pomoću *Spring Data Neo4j*

- Anotacija @*Relationship* omogućava mapiranje grana između pojedinih čvora
- Ograničenje ovakvog pristupa jeste što je nemoguće dodati obeležje na vezu

```
 1 @Node
 2 public class Movie {
 3   @Id
 4   private String title;
 5
 6   @Property("tagline")
 7   private String description;
 8
 9  @Relationship(value="DIRECTED", direction=Relationship.Direction.INCOMING)
10   private List<Person> directors;
11
12 }
```

Mapiranje grana na klase pomoću *Spring Data Neo4j*

- Anotacija @*RelationshipProperties* omogućava mapiranje grana između pojedinih čvorova, unutar koje se nalaze dodatna obeležja

```
 1 @RelationshipProperties
 2 public final class Actor {
 3   @RelationshipId
 4   private Long id;
 5
 6   @TargetNode
 7   private Person person;
 8
 9   private List<String> roles;
10 }
```

Kreiranje upita pomoću *Spring Data Neo4j*

- Proširenjem interfejsa repozitorijuma klasom ***Neo4jRepository***, omogućuje se kreiranje jednostavnih i složenih upita
- Za složene upite koristi se anotacija @*Query*

```
1  @Repository
2  public interface PersonRepository extends Neo4jRepository<Person, Long> {
3      Person findByName(String name);
4
5      @Query("MATCH (p: Person) - [:ACTED_IN] -> (m: Movie)
                WHERE m.released > $year AND NOT EXISTS((p) - [:DIRECTED] -> (m))
                RETURN DISTINCT p")
6      List<Person> findActorsInMoviesByYear(@Param("year") int year);
7  }
```

Sadržaj

Algoritmi preporuke

- **Algoritmi preporuke** koriste se za predlaganje stavki korisnicima na osnovu njihovih preferencija ili ponašanja
- Primenjuju se u različitim aplikacijama kao što su e-trgovina, *streaming* servisi i društvene mreže
- Koristeći algoritme preporuke, platforme mogu povećati angažman korisnika i poboljšati prodaju preporučujući relevantne proizvode ili sadržaj

Primena baze podataka *Neo4j*

- ***Neo4j*** je baza podataka zasnovana na grafu, što je čini pogodnom za kreiranje sistema preporuka
- Grafska struktura omogućava efikasno modelovanje veza između entiteta kao što su korisnici, proizvodi i interakcije
- Neo4j omogućava brzu obradu veza među entitetima, što je ključno za algoritme preporuka koji analiziraju složene veze
- Grafska struktura olakšava navigaciju kroz podatke, omogućavajući efikasno pronalaženje povezanih entiteta i otkrivanje skrivenih uzroka ponašanja korisnika ili relacija među proizvodima

Primeri algoritama preporuka

- **Saradničko filtriranje (engl. *Collaborative Filtering*):**
  - Analizira istorijske interakcije korisnika sa stavkama kako bi pronašao sličnosti među korisnicima ili stavkama
  - Na osnovu pronađenih sličnosti, algoritam može da predloži stavke koje bi mogle da budu zanimljive korisnicima
  - U bazi podataka *Neo4j*, saradničko filtriranje se može implementirati kroz analizu grafa korisnika i njihovih interakcija sa stavkama
- **Funkcije sličnosti (engl. *Similarity Functions*):**
  - Koriste metrike sličnosti kako bi odredili koliko su stavke ili korisnici slični jedni drugima
  - Na osnovu metrika sličnosti, funkcije sličnosti mogu predložiti stavke koje su najviše slične preferencijama korisnika
  - U bazi podataka *Neo4j*, funkcije sličnosti se mogu primeniti kroz analizu grafa i preračunavanje sličnosti između čvorova
- **Matrično faktorizovanje (engl. *Matrix Factorization*):**
  - Razlaže matricu ocena korisnika i stavki na manje dimenzije kako bi otkrio skrivene obrasce i preferencije korisnika
  - Na osnovu obrazaca, algoritam može da predloži nove stavke korisnicima
  - U bazi podataka *Neo4j*, matrično faktorizovanje se može primeniti kroz analizu grafa korisnika i stavki i prilagođavanje ocena na osnovu pronađenih obrazaca

Sadržaj

Korisni linkovi

- **Neo4j – zvanična dokumentacija**
  - *https://neo4j.com/docs/*
- **Spring Data Neo4j – dokumentacija**
  - *https://docs.spring.io/spring-data/neo4j/reference/getting-started.html*

Napredne arhitekture informacionih sistema

Grafske baze podataka

Pitanja?

Predmetni nastavnik:
dr Marko Vještica

DAIS
Data Science and Information Systems Group