Napredne arhitekture informacionih sistema

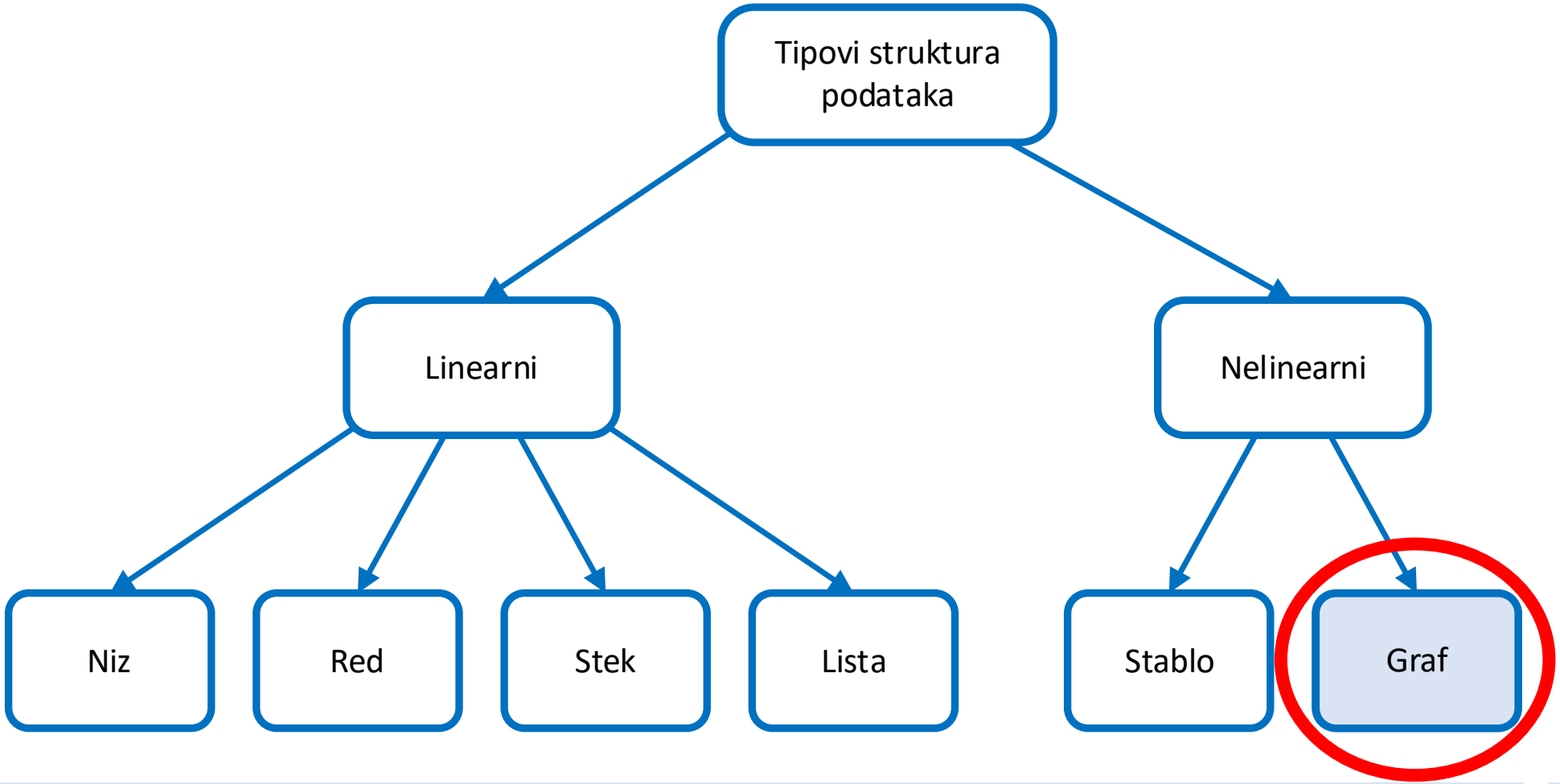
Grafske baze podataka

Predmetni nastavnik:
dr Marko Vještica

DAIS
Data Science and Information Systems Group

Sadržaj

Pregled osnovnih struktura podataka

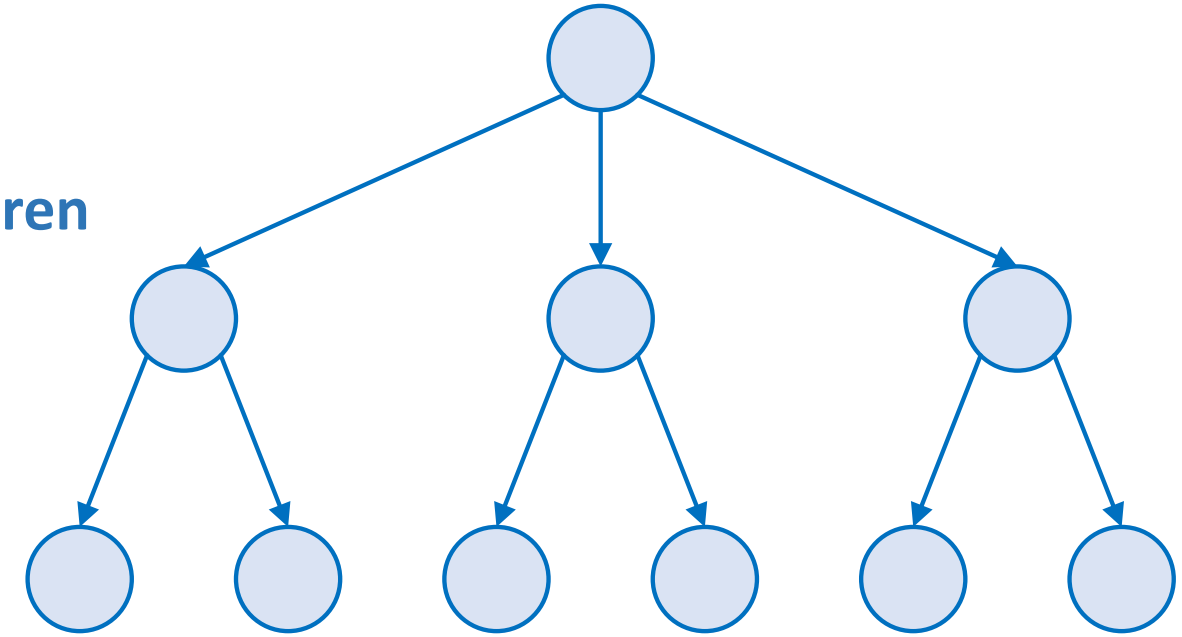


Pregled osnovnih struktura podataka

- **Linearne** strukture podataka:
  - Elementi strukture **organizovani sekvencijalno**, jedan za drugim
  - Moguć jednostavan pristup **prolaskom redom** kroz sve elemente
- **Nelinearne** strukture podataka:
  - Elementi strukture **nisu organizovani sekvencijalno**
  - Pojedinačni elementi mogu posedovati **više veza**
  - Moguće primeniti **pretrage po dubini i širini**

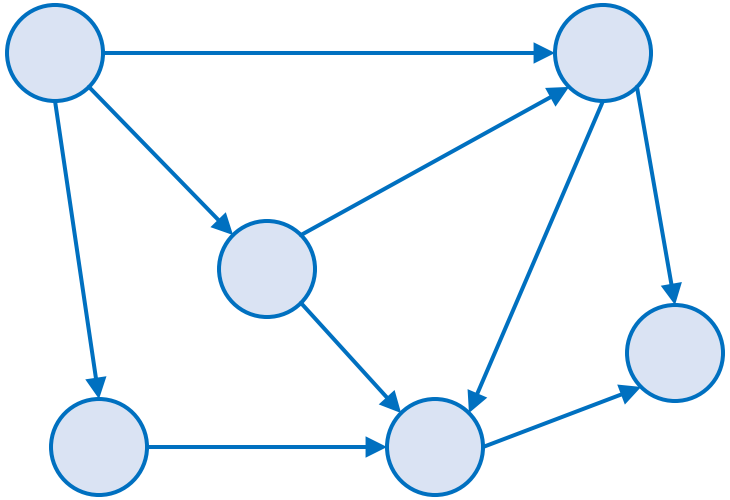
Struktura podataka tipa stablo

- **Stablo** predstavlja **hijerarhijsku strukturu podataka** koja sadrži skup čvorova i skup grana
  - **Čvorovi** sadrže relevantne podatke
  - **Grane** predstavljaju hijerarhijsku uređenost
- Stablo mora imati čvor koji predstavlja **koren**
  - Može da ima **više podstabala**
  - Čvorovi na poslednjem nivou hijerarhije nazivaju se **listovi**
- **Primeri primene:**
  - Indeksi u bazama podataka (npr. B-stabla)
  - Hijerarhijski organizovani podaci (npr. XML)
  - Parsiranje programskog koda (npr. stablo parsiranja)
  - Mašinsko učenje (npr. stablo odlučivanja)



Struktura podataka tipa graf

- **Graf** predstavlja **strukturu podataka** koja sadrži skup čvorova i skup grana
  - **Čvorovi** sadrže relevantne podatke
  - **Grane** predstavljaju odnose između čvorova
- Čvorovi grafa mogu biti **povezani** sa bilo kojim drugim čvorovima grafa
- **Primeri primene:**
  - NoSQL baze podataka (npr. grafske baze podataka)
  - Grafski organizovani podaci (npr. društvene mreže)
  - Računarske mreže (npr. WWW)
  - Sistemi preporuke (npr. Internet prodavnica)



Grafovi i stabla

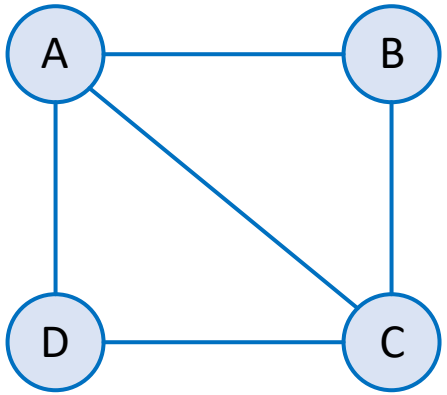
- Da li stablo predstavlja graf?
  - **Stabla predstavljaju podskup grafova**
  - Stabla imaju **dodatna ograničenja** u odnosu na grafove
  - **Svako stablo je graf, ali nije svaki graf stablo**

Stablo	Graf
Svi čvorovi moraju biti **povezani**	Ne moraju svi čvorovi biti **povezani**
Stablo od N čvorova ima **N-1 grana**	Ne postoji ograničenje **broja grana**
Ima tačno jedan **korenski čvor**	Ne mora imati **korenski čvor**
Svaki čvor stabla mora imati tačno jednu **putanju** od korena stabla	Ne postoji ograničenje u broju **putanja** do određenog čvora
Aciklična struktura	Može sadržati **petlje i cikluse**
Usmerene grane	Ne moraju biti **usmerene** grane
Najviše jedan **nadređeni** i nula ili više **podređenih** čvorova	Nema ograničenja **nadređenih i podređenih** čvorova

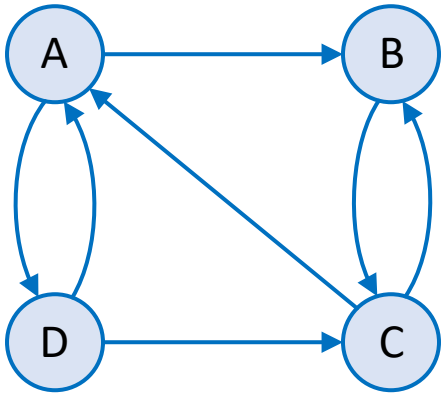
Sadržaj

Teorija grafova

- **Matematički graf** je uređeni par $G=(V, E)$ čija je prva komponenta **neprazan skup čvorova**, a druga komponenta **binarna relacija u skupu čvorova** koja odgovara granama grafa
- Postoje **neusmereni i usmereni** grafovi
- Primer **neusmerenog** grafa:
  V={A, B, C, D},
  E={{A,B}, {B,C}, {C,D}, {D,A}, {A,C}}

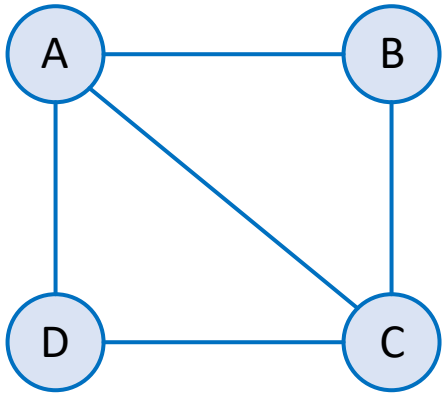


- Primer **usmerenog** grafa:
  V={A, B, C, D},
  E={(A,B), (A,D), (D,A), (B,C), (C,B), (C,A), (D,C)}

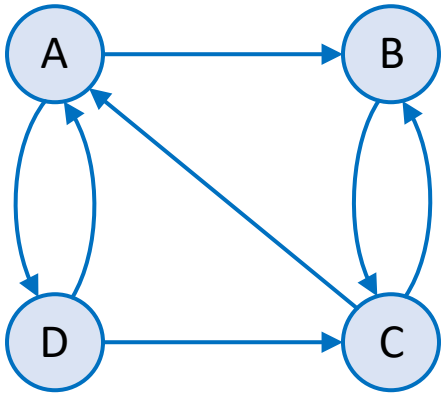


Primena neusmerenih i usmerenih grafova

- **Neusmereni** grafovi imaju **dvosmerne** grane
- **Usmereni** grafovi imaju **jednosmerne** grane
- **Primer primene grafa** u informacionim sistemima: društvene mreže
- Primer **neusmerenog** grafa:
  - Prijateljstvo na društvenim mrežama
  - Npr. ukoliko je osoba A prijatelj sa osobom B, onda je i osoba B prijatelj sa osobom A

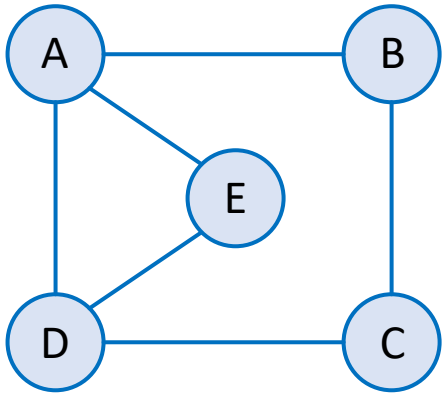


- Primer **usmerenog** grafa:
  - Praćenje na društvenim mrežama
  - Npr. ukoliko osoba A prati osobu B, to ne mora da znači da osoba B prati osobu A

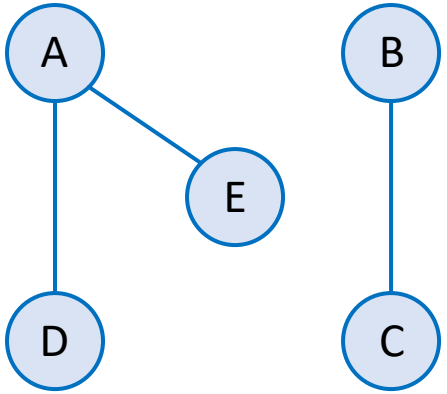


Povezani i nepovezani grafovi

- Grafovi se takođe dele na **povezane i nepovezane** grafove
- Svakom čvoru **povezanog grafa** moguće je pristupiti iz bilo kojeg drugog čvora u okviru grafa, dok unutar **nepovezanog grafa** postoji barem jedan čvor kojem nije moguće pristupiti iz nekog drugog čvora
- Primer **povezanog** grafa:
  - Svi gradovi unutar države povezani su direktno ili indirektno rutama

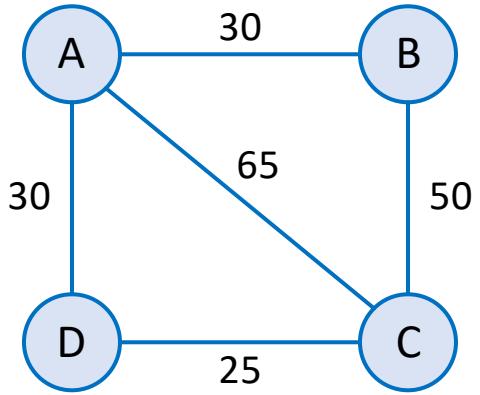


- Primer **nepovezanog** grafa:
  - Postoje grupe prijatelja na društvenoj mreži koji se međusobno ne poznaju



Grafovi sa težinskim granama

- Grane unutar grafa mogu da imaju **težine**
  - Predstavljaju **vrednosti** koje imaju određeno **značenje**
- Na taj način moguće je dati **različit značaj grane** u odnosu na druge grane
- Primer upotrebe **grafa sa težinskim granama**:
  - Određivanje najefikasnijeg puta između dva mesta
  - Težine mogu da predstavljaju: npr. dužinu puta, vreme, utrošak energije ili gustinu saobraćaja



Primena grafova

- Primeri **primene** grafova:
  - **Informacioni sistemi:** sistem preporuke proizvoda kupcima
  - **Veštačka inteligencija:** neuronske mreže
  - **Transport:** pronalazak najkraćeg puta između dva mesta
  - **Industrija:** resursi unutar fabrike i njihove sposobnosti
  - **Telekomunikacije:** računarske mreže
  - **Internet:** povezanost web stranica
  - **Hemija:** predstavljanje molekula
  - **Sport:** analiza interakcije između igrača
  - **Elektronika:** elektronska kola
  - **Bankarstvo:** identifikacija prevara (transakcije, deljenje IP adrese)

Prednosti i nedostaci

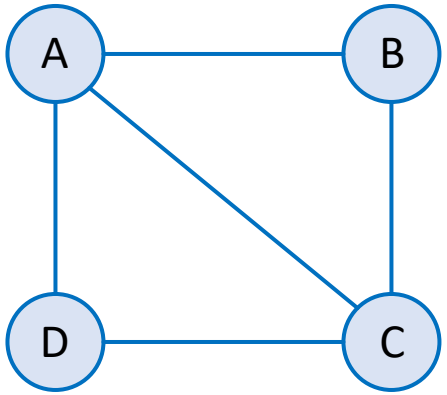
- **Prednosti** primene grafa:
  - **Vizualna reprezentacija** entiteta i njihovih odnosa
  - Različiti algoritmi za **analizu grafa**, odnosno analizu podataka
  - Primena u **različitim domenima**
- **Nedostaci** primene grafa:
  - Grafovi mogu postati **obimni i nepregledni**
  - **Algoritmi** za analizu grafa mogu zahtevati **veliku procesnu snagu**
    - Takođe mogu biti **izazovni za implementaciju**

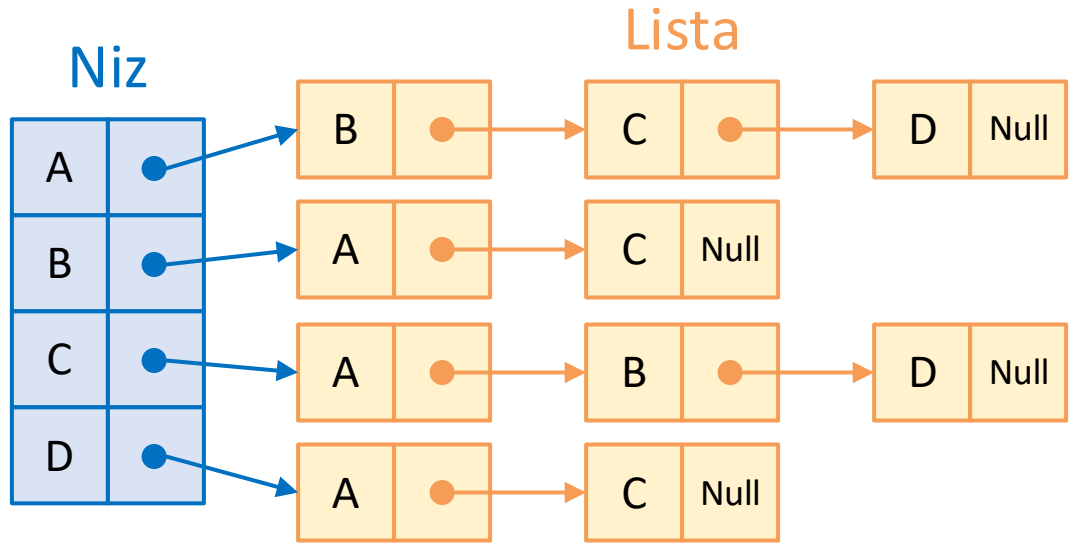
Sadržaj

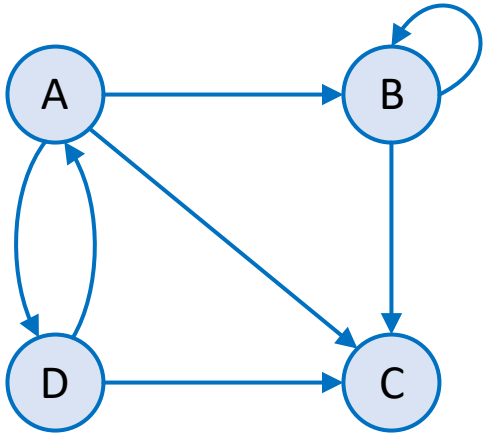
Grafska struktura podataka

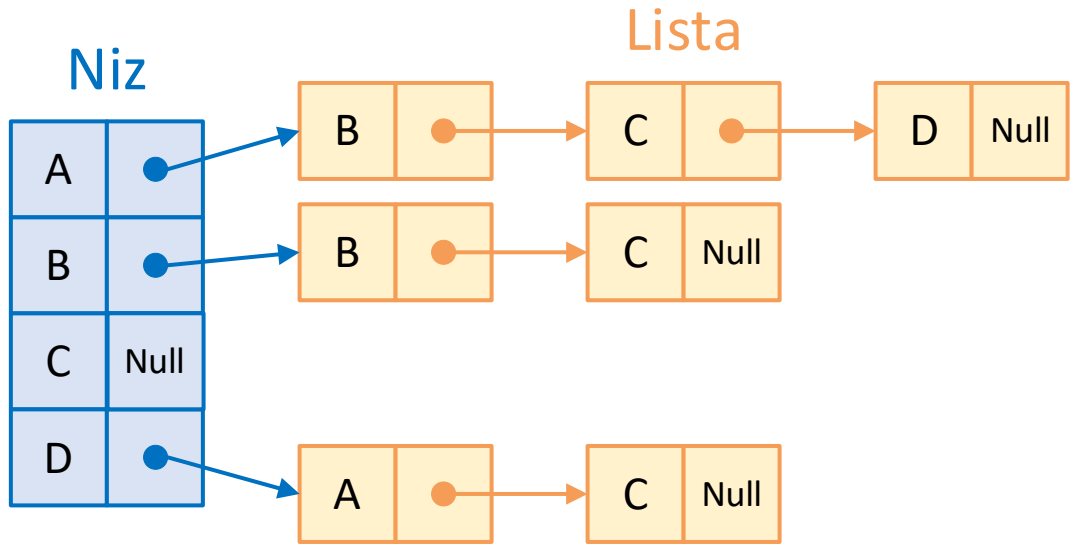
- Dva najčešća načina predstavljanja grafova kao **strukture podataka** su:
  - **Lista povezanosti** (engl. *Adjacency List*)
  - **Matrica povezanosti** (engl. *Adjacency Matrix*)
- Lista povezanosti i matrica povezanosti **organizuju čvorove i njihove susede**
- Omogućavaju primenu različitih **algoritama za pretragu grafa**

Lista povezanosti









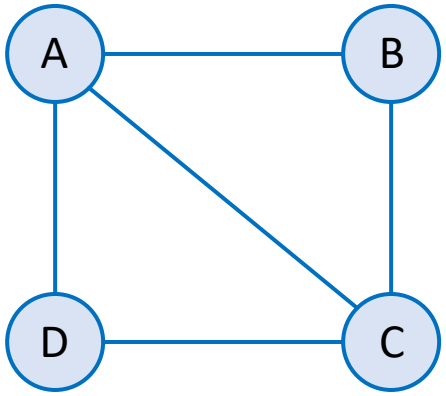
Lista povezanosti

- **Lista povezanosti** predstavlja strukturu podataka u kojoj su **čvorovi grafa smešteni u niz**, a svaki čvor u nizu sadrži **listu susednih čvorova**
- **Susedni čvorovi** predstavljaju čvorove koji su **direktno povezani**
  - Postoji grana koja **direktno povezuje dva susedna čvora**
- **Dužina niza** određena je **brojem čvorova** u grafu
- Moguće je na jednostavan način **izračunati broj čvorova i grana u grafu**
- Ukoliko se lista povezanosti posmatra kao dvodimenzionalni niz, onda ona predstavlja **niz nizova** koji mogu biti **različite dužine** (engl. *Jagged Array*)
- Pogodna je za upotrebu kada postoje **grafovi sa malim brojem grana**
- **Napomena:** Listu povezanosti moguće je implementirati i uz pomoć drugih struktura, poput vektora, skupova ili mapa

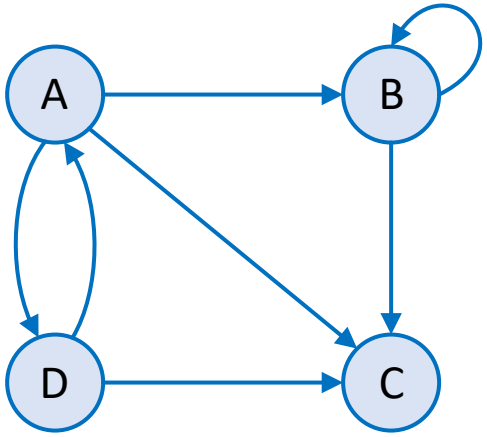
Lista povezanosti

- Listu povezanosti moguće je **kreirati** na sledeći način:
  - Kreirati niz dužine broja čvorova u grafu
  - Kreirati (praznu) listu susednih čvorova za svaki čvor u grafu
  - Za svaku granu (*p*, *q*), dodati čvor *q* u listu čvora *p*
    - Ukoliko su grane neusmerene, dodati takođe čvor *p* u listu čvora *q*
    - U slučaju težinskih grana, dodati i težinu

Matrica povezanosti



	A	B	C	D
A		1	1	1
B	1		1	
C	1	1		1
D	1		1	



	A	B	C	D
A		1	1	1
B		1	1	
C				
D	1		1	

Matrica povezanosti

- **Matrica povezanosti** predstavlja strukturu podataka matrice u okviru koje se čuvaju **susedstva između čvorova grafa**
- Matrica povezanosti je **kvadratna matrica veličine *N x N***, u kojoj *N* predstavlja broj čvorova grafa
- Vrednost ćelije matrice povezanosti je **nula**, ukoliko dva čvora nisu susedni ili je vrednost **različita od nule**, ukoliko dva čvora jesu susedni
- Moguće je na jednostavan način **izračunati broj grana u grafu**
- Ukoliko je **broj grana grafa mali**, matrica povezanosti ima značajno veći broj nula vrednosti, nego vrednosti različite od nula (engl. *Sparse Matrix*)
- Ukoliko je u pitanju **neusmereni graf**, matrica povezanosti biće **simetrična**
- Pogodne su za upotrebu kada postoje **grafovi sa velikim brojem grana**

Matrica povezanosti

- Matricu povezanosti moguće je **kreirati** na sledeći način:
  - Kreirati matricu veličine *N x N*, u kojoj *N* predstavlja broj čvorova u grafu
  - Inicijalizovati sve ćelije matrice na vrednost nula
  - Za svaku granu (*p*, *q*), postaviti vrednost ćelije (*p*, *q*) na vrednost različitu od nule
    - Ukoliko su grane neusmerene, učiniti isto i za ćeliju (*q*, *p*)
    - U slučaju težinskih grana, postaviti vrednost ćelije na vrednost težine grane

Poređenje liste povezanosti i matrice povezanosti

Lista povezanosti	Matrica povezanosti
Može da zauzima **veći memorijski prostor** u odnosu na matricu povezanosti u slučaju **velikog broja grana**	Može da zauzima nepotrebno **velik memorijski prostor** u slučaju **malog broja grana**, odnosno *sparse* matrica
Pogodnije za upotrebu kada postoji **mali broj grana**	Pogodnije za upotrebu kada postoji **velik broj grana**
Dodavanje čvora jednostavno usled proširenja niza	**Dodavanje čvora zahtevno** usled proširenja cele matrice
Jednostavno dodavanje grane proširenjem liste	**Jednostavno dodavanje grane** izmenom ćelije matrice
Brisanje čvora zahteva **pretragu niza čvorova i svih relevantnih grana**, a zatim i njihovo brisanje	Brisanje čvora zahteva **izmenu matrice**
Brisanje grane zahteva pronalazak i brisanje **grane u listi**	Brisanje grane jednostavnije usled **izmene ćelije** matrice
Pronalazak grane zahtevniji jer je potrebno **pretražiti listu** odgovarajućeg čvora	Pronalazak grane jednostavan usled **pristupanja ćeliji matrice**
Pronalazak svih suseda jednog čvora jednostavan prolaskom kroz **listu datog čvora**	Pronalazak svih suseda jednog čvora zahtevniji jer je potrebno pretražiti **ceo niz matrice** za dati čvor

Obilazak grafa

- **Obilazak grafa** (engl. *Graph Traversal*) predstavlja **proces pretrage grafa** u okviru kojeg se **pristupa čvorovima grafa posredstvom grana**
  - Omogućava pristup čvorovima **bez kreiranja petlje**
  - Odnosno, omogućava svakom čvoru da se pristupi **najviše jednom**
- Postoje **razni algoritmi obilaska grafa** (npr. *A* Search*, *Dijkstra's Algorithm*, *Bellman-Ford Algorithm*) koji se mogu koristiti u zavisnosti od različitih potreba pretrage i strukture grafa
- **Dva algoritma obilaska grafa** koji se često primenjuju:
  - **Pretraga u dubinu** (engl. *Depth First Search*) – algoritam počinje od određenog čvora grafa i pronalazi čvorove koji se nalazi na istoj grani, ali na različitim nivoima dubine; pronalaskom čvora koji nema više izlaznih grana, vraća se na prethodni čvor koji je imao dodatnih grana
  - **Pretraga u širinu** (engl. *Breadth First Search*) – algoritam počinje od određenog čvora grafa, za kojeg pronalazi sve susede na prvom nivou dubine, pretražujući zatim susede nivo po nivo

Sadržaj

Grafske baze podataka

- **Grafske baze podatka** koriste strukturu **grafa** za skladištenje podataka
  - Zasnovane na **teoriji grafa**
- Osnovni koncepti koje koristi grafska baza podataka su **čvorovi i grane**
  - Grafske baze podataka reprezentuju **realne entitete** kao **čvorove**
  - Dok **odnosne između entiteta** reprezentuju kao **grane**
- Grafske baze podataka uglavnom su napravljene za **transakcione, OLTP sisteme**
  - Optimizovane za **CRUD**
- **Dve osnovne komponente** sistema za upravljanje grafskom bazom podataka:
  - **Skladište podataka**
  - **Pogon za obradu podataka**

Grafske baze podataka

- **Skladište podataka:**
  - Grafske baze podataka koriste **izvorno grafsko skladište podataka** (engl. *Native Graph Storage*) koje čuva podatke u formi grafa
  - Napomena: pojedini sistemi za upravljanje grafskom bazom podataka implementiraju skladište u formi poput relacione baze podataka ili objektno-orijentisane baze podataka
- **Pogon za obradu podataka:**
  - Grafske baze podataka koriste **izvorno grafsku obradu podataka** (engl. *Native Graph Processing*)
  - **Susedni čvorovi fizički su povezani** u bazi podataka
  - **Nije potrebna dodatna struktura podataka**, poput indeksa, koja bi povezala čvorove i obezbedila odgovarajuću brzinu izvršavanja upita (engl. *Index-Free Adjacency*)
  - Koriste **algoritme obilaska grafa** za izvršavanje upita
    - **Prateći grane između čvorova**

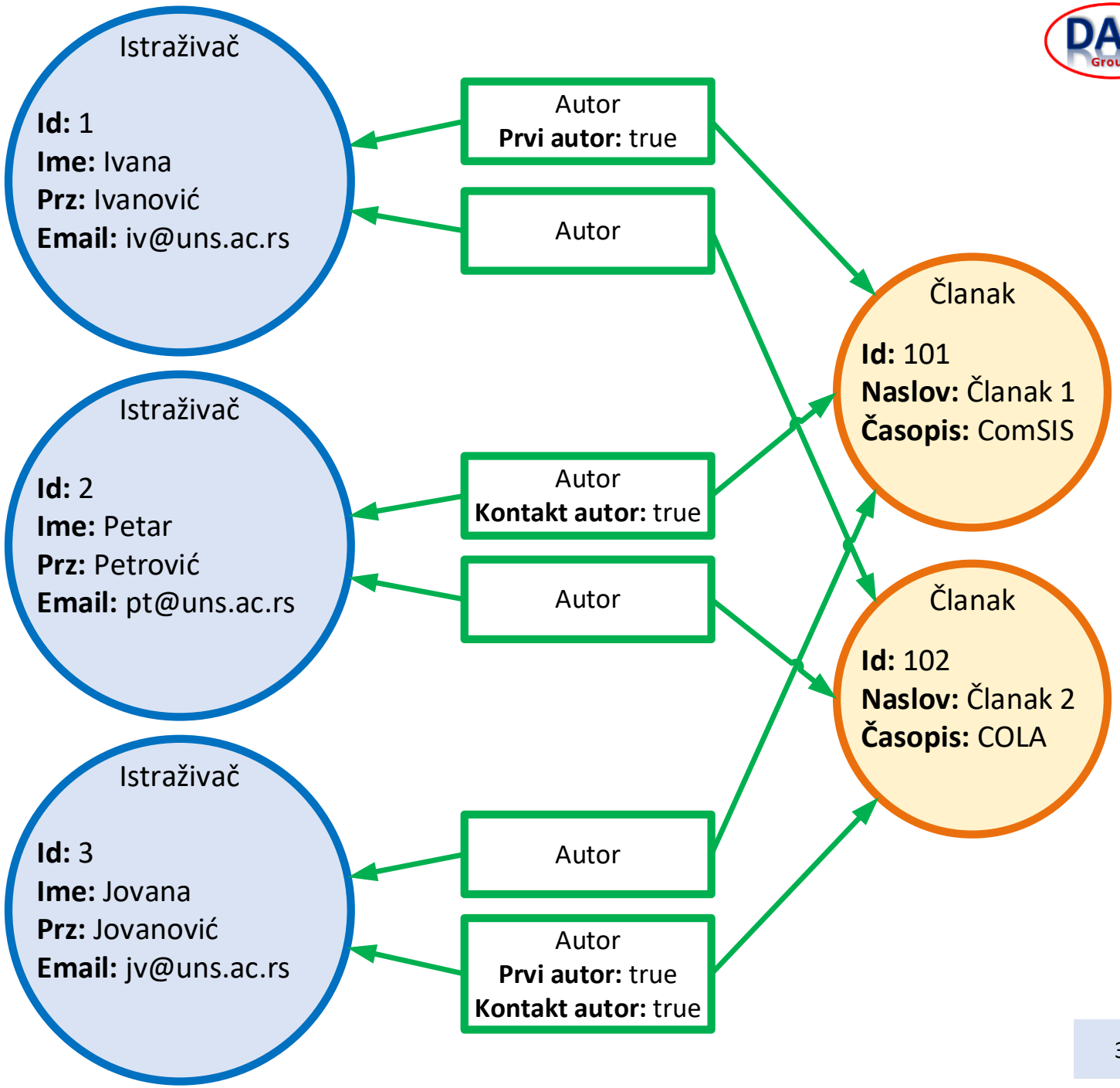
Grafski modeli podataka

- **Tri osnovna grafska modela podataka:**
  - **Model grafa sa svojstvima** (engl. *Property Graphs*)
  - **Model grafa trojki** (engl. *Triples*)
  - **Model hipergrafa** (engl. *Hypergraphs*)
- **Zajednička osobina** za sva tri modela je da poseduju čvorove i grane

Model grafa sa svojstvima

- **Model grafa sa svojstvima** sadrži **čvorove, grane i svojstva**
  - **Čvorovi** reprezentuju realne entitete
  - **Grane** reprezentuju odnose između realnih entiteta
    - Grane su **usmerene i uvek povezuju dva čvora** – početni i krajnji
  - Čvorovi i grane imaju **internu strukturu**
    - Sadrže **svojstva u formi ključ-vrednost**
- **Označeni grafovi sa svojstvima** (engl. *Labeled Property Graph*) predstavljaju proširenje grafova sa svojstvima
  - Sadrže sve što i grafovi sa svojstvima
  - Dodatno, čvorovi mogu da sadrže **jednu ili više oznaka** (labela)
  - Dodatno, grane mogu biti **imenovane**, odnosno biti tačno jednog odgovarajućeg **tipa**
- Primer SUBP koji podržava model označenog grafa sa svojstvima: *Neo4j*

Primer označenog grafa sa svojstvima



Model grafa trojki

- **Model grafa trojki** potiče od **semantičkog veba** (engl. *Semantic Web*), u kojem su dodavane semantičke oznake različitim resursima na vebu, kako bi računar mogao da ih procesira na automatizovan način
  - Nije bilo praktično izvodivo, stoga se prešlo na čuvanje podataka sa veba u **skladištima trojki**
- **Trojka** predstavlja strukturu podataka **subjekat-predikat-objekat**
  - Npr. Ivana Ivanović – napisala je – Zbirku zadataka iz matematike
  - Time su podaci iskazani na **strukturiran način** (u poređenju sa rečenicom u formi teksta), što omogućava računarima da bolje razumeju i procesiraju **značenje podataka**, a ne samo teksta
- Pojedinačne trojke ne donose veliku semantičku vrednost, ali **više spojenih trojki** mogu doneti **novo znanje i zaključke o novim vezama**
- Model grafa trojki koristi se za skladištenje podataka **formata *Resource Description Framework* (RDF)**

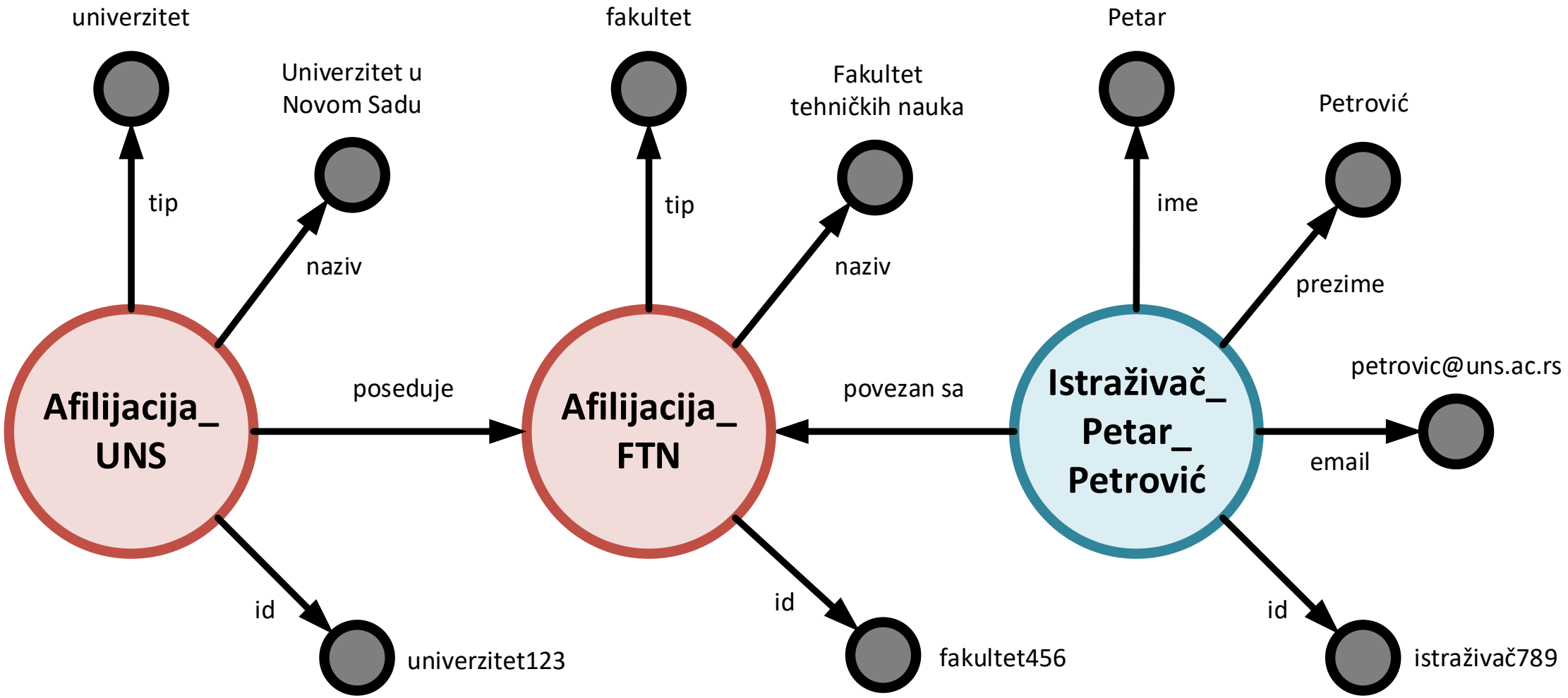
Model grafa trojki – RDF

- ***Resource Description Framework* (RDF)** je **grafski model podataka** razvijen kao standard od strane *World Wide Web Consortium* (W3C)
- Kreiran je za potrebe predstavljanja **meta-podataka resursa na vebu**
- Služi za **strukturiran opis resursa i razmenu podataka** u formi grafa
- Predstavlja **usmereni graf** sačinjen od iskaza **trojki**:
  - **Subjekat** – čvor koji predstavlja resurs
  - **Predikat** – grana između subjekta i objekta koja predstavlja svojstvo resursa (subjekta) ili odnos između dva resursa (subjekta i objekta)
  - **Objekat** – čvor koji predstavlja vrednost svojstva (literal) ili resurs
- Čvorovi i grane **nemaju internu strukturu**
- Resursi i svojstva mogu biti **identifikovani** uz pomoć **URI** (engl. *Uniform Resource Identifier*)
- Postoji **standardizovani jezik** (od strane W3C) za pristup grafovima tipa RDF – **SPARQL**

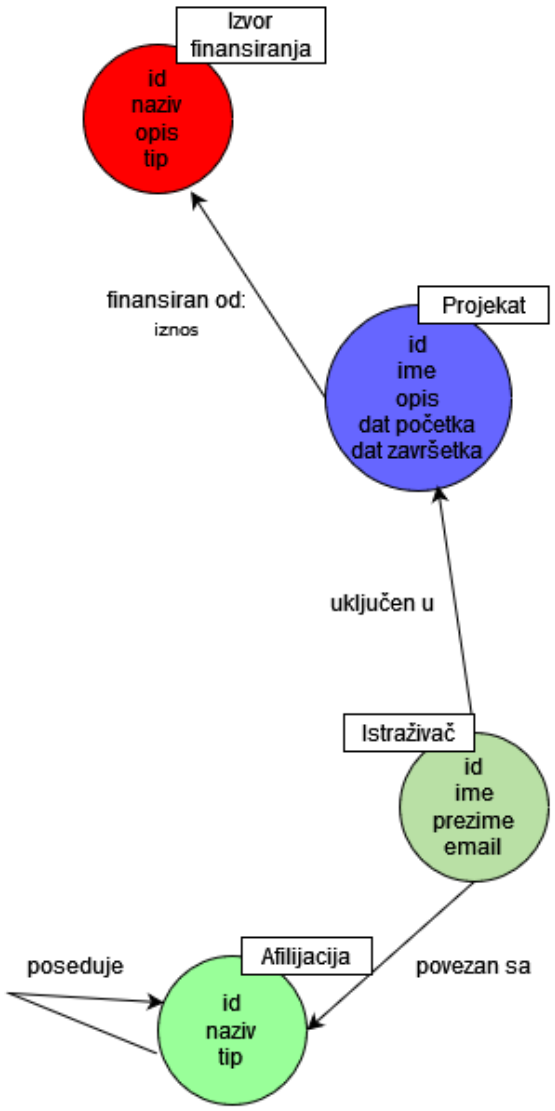
Model grafa trojki – RDF

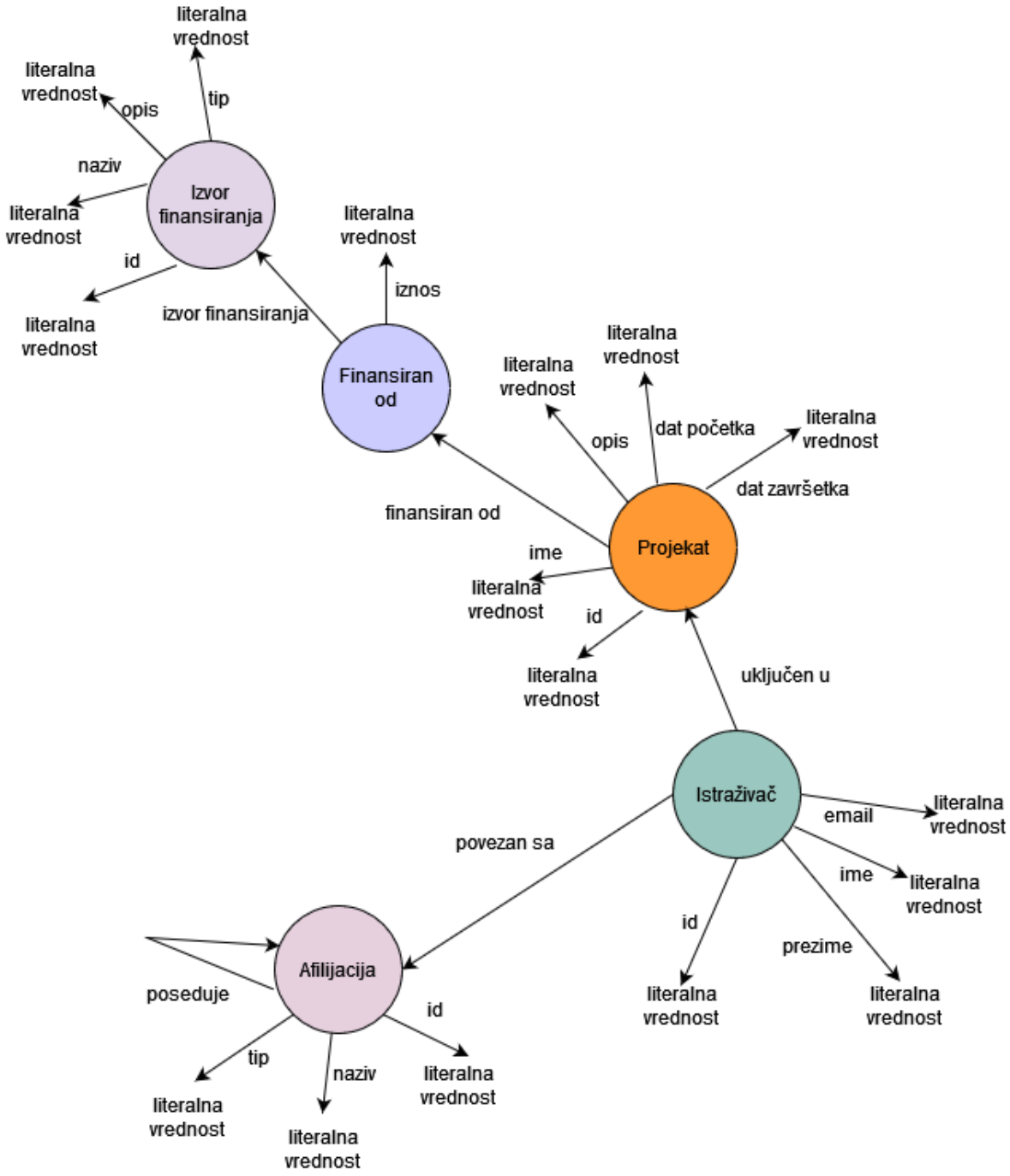
- Podaci formata RDF mogu biti **serijalizovani** na različite načine (npr. RDF/XML)
- Usled postojanja standarda serijalizacije podataka, moguće je **razmenjivati podatke** formata RDF **između različitih sistema**
  - Odnosno sistema koji ne koriste isti model podataka
- RDF trojke su često bile korišćene za predstavljanje **grafova znanja**
  - **Napomena:** ali RDF nije neophodan za kreiranje grafova znanja, uglavnom se koriste grafovi sa svojstvima u tu svrhu
  - U praksi, RDF se koristi kao format za **razmenu podataka između sistema**
- Za definisanje **tipova resursa i logike njihovih odnosa**, postoje jezici ontologije poput RDF Schema (RDFS) i Web Ontology Language (OWL)

Primer grafa trojki



Poređenje označenog grafa sa svojstvima i grafa trojki





Izvor: Draško Novaković, Poređenje modela i performansi izvršavanja upita grafskih baza podataka tipa LPG i RDF, Diplomski rad, 2024.

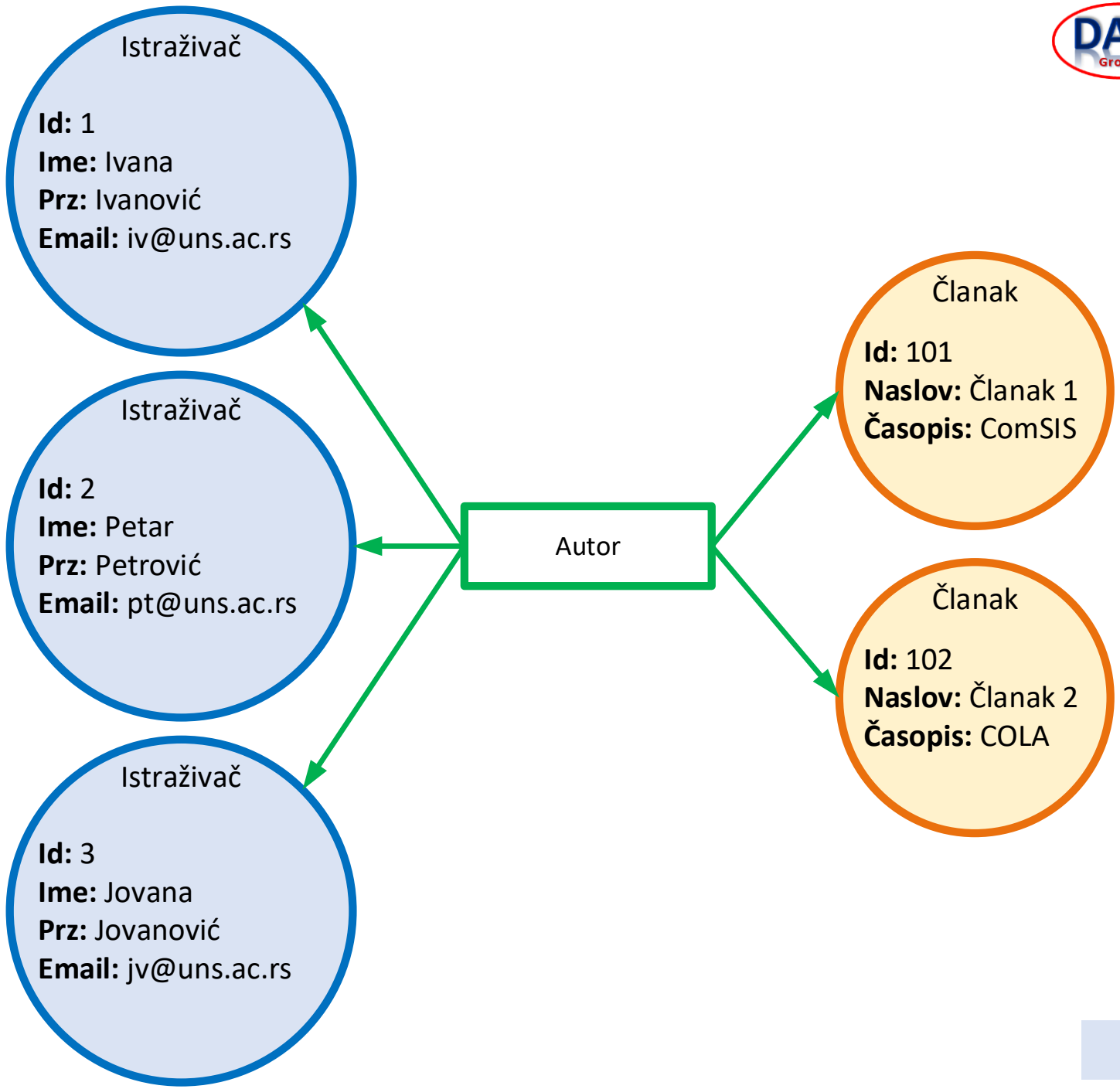
Model grafa trojki

- Grafske baze podataka koje podržavaju **model trojki**, sadrže podatke koji su logički povezani, ali **nisu fizički povezani**
- Na taj način **ne predstavljaju izvorno grafska skladišta podataka**
  - Ne podržavaju **susedstvo bez indeksa** (engl. *Index-Free Adjacency*)
- Ne skladište grafove, već **trojke kao nezavisne slogove**
  - Što omogućava **jednostavno horizontalno skaliranje** baze podataka
  - Ali time pogon za procesiranje podataka nije optimizovan za rad sa grafovima, odnosno **nije optimizovan za efikasan obilazak grafa**
  - Za potrebe izvršavanje upita, neophodno je najpre **povezati fizički razdvojene trojke**, što **usporava izvršavanje upita**
- Primer SUBP koji podržava model trojki: *GraphDB*

Model hipergrafa

- **Hipergraf** predstavlja generalizaciju grafa u kojoj je moguće da **jedna grana povezuje više od dva čvora**
  - Ovakve grane nazivaju se **hipergrane**
  - Moguće je imati **više od jednog čvora na oba kraja** hipergrane
- Hipergrafovi **pogodni su** u domenima koji su uglavnom sastavljeni od **više-prema-više (*N*:*M*) veza**
  - Jedna **hipergrana** može da **zameni više grana**
- Međutim, moguće je da prilikom modelovanja hipergrafa budu **izostavljeni pojedini važni detalji**

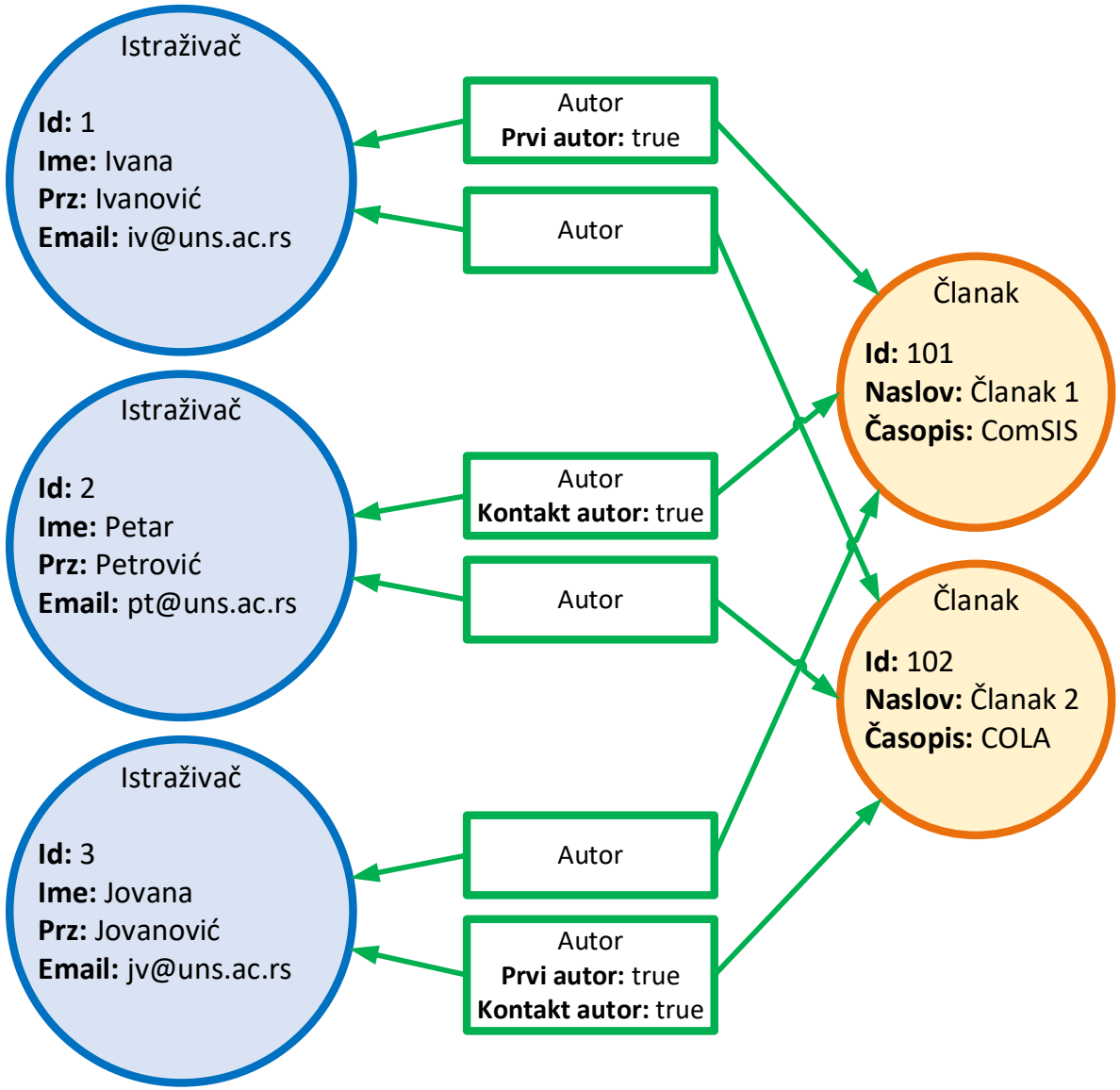
Primer hipergrafa

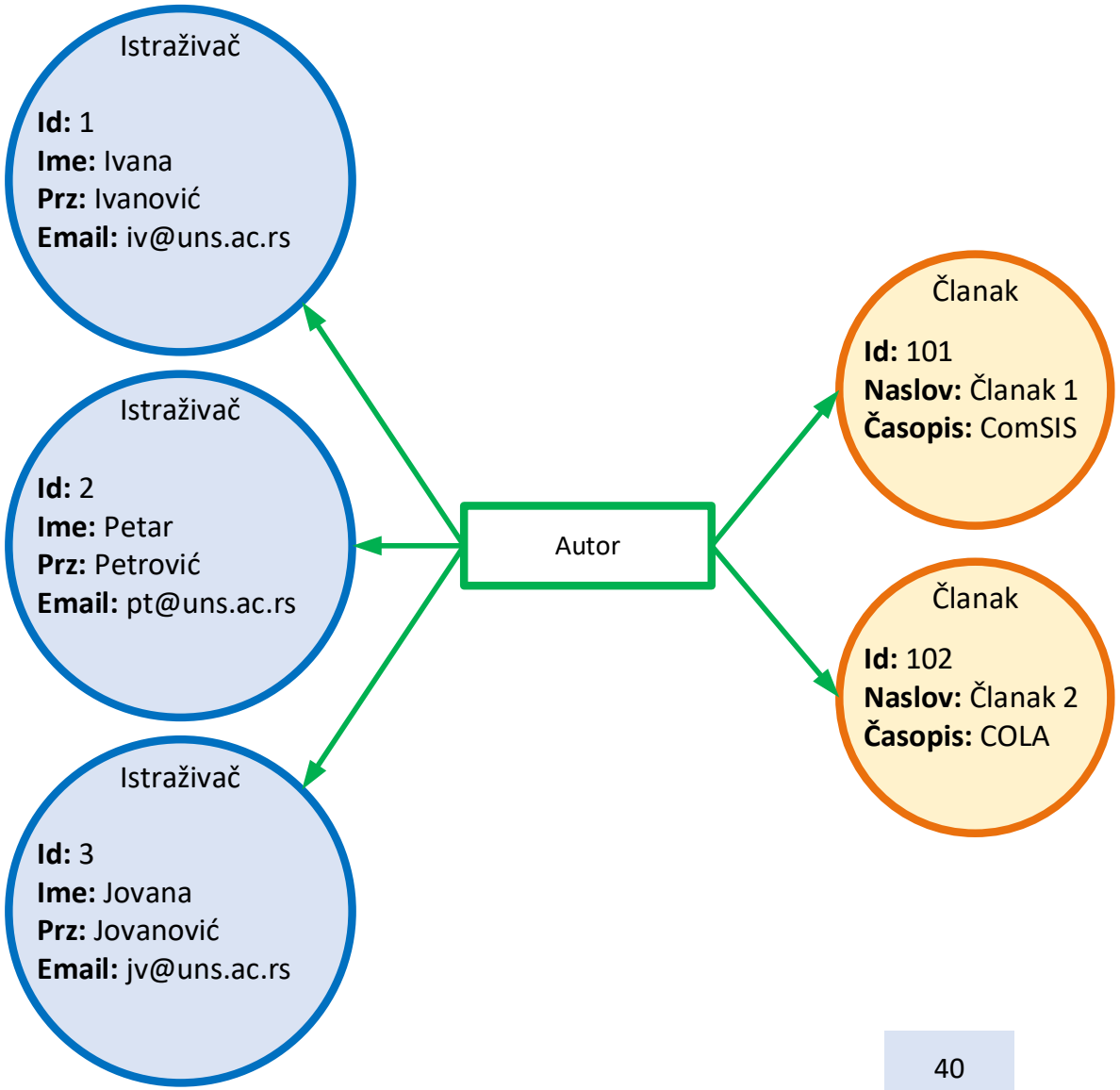


Model hipergrafa

- **U prethodnom primeru**, jedna hipergrana zamenila je šest grana predstavljenih u istom primeru sa označenim grafom sa svojstvima
  - Međutim, nedostaju informacije o prvim i kontakt autorima članaka
    - Predstavlja važne informacije koje su bile dodate u svojstvima grana
- **Rešenje:** Uvođenje tri različita tipa grana za autora, prvog autora i kontakt autora?
- Kako hipergraf predstavlja generalizaciju grafa, moguće je **transformisati hipergraf u označeni graf sa svojstvima** na automatizovan način
- **Označeni grafovi sa svojstvima** smatraju se **dobrim balansom u semantici koju pružaju i broja elemenata** neophodnih za modelovanje određenog sistema
- Primer SUBP koji podržava model hipergrafa: *TypeDB*
- **Napomena:** u nastavku prezentacije opisani su označeni grafovi sa svojstvima

Poređenje označenog grafa sa svojstvima i hipergrafa

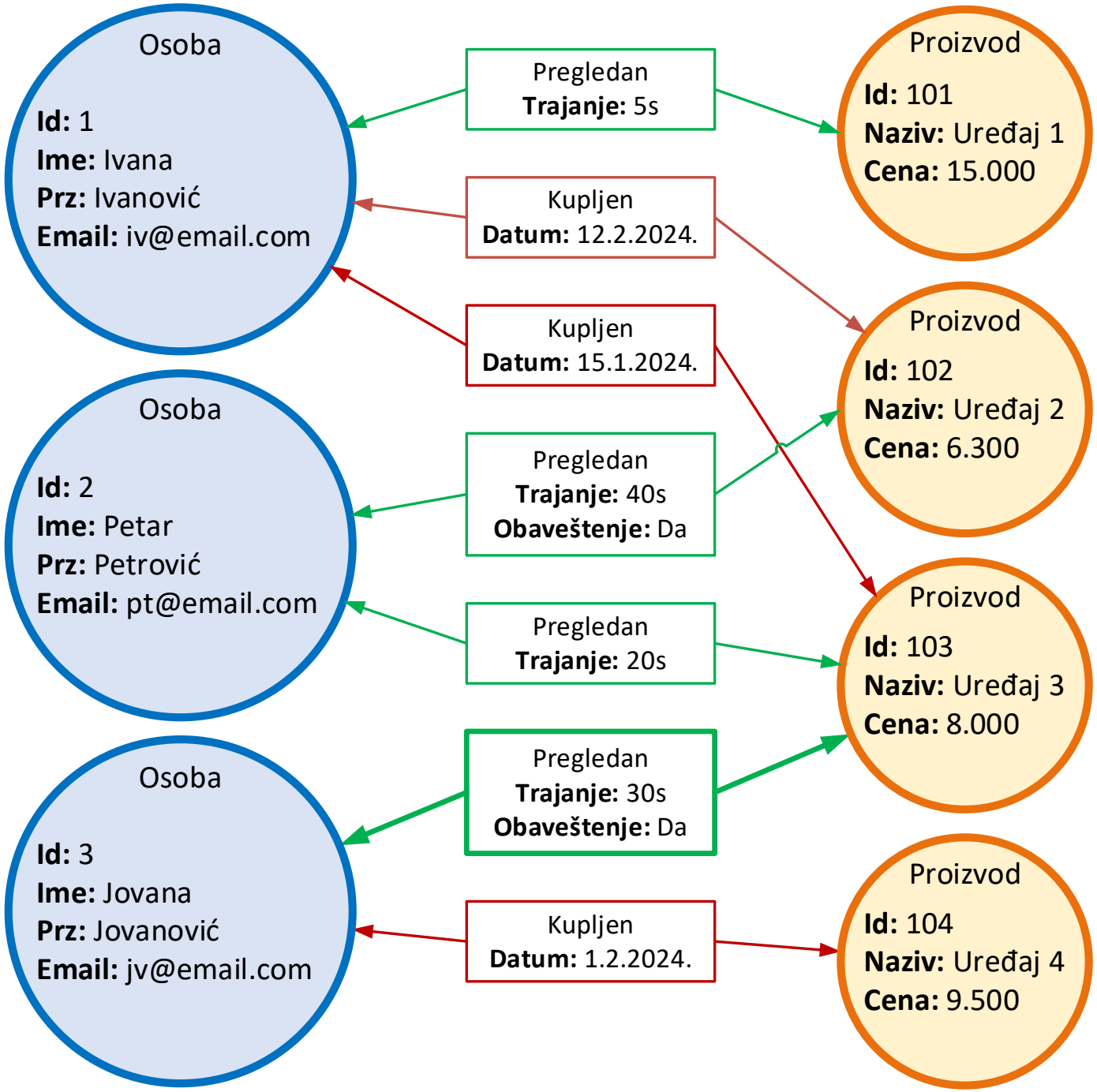




Sadržaj

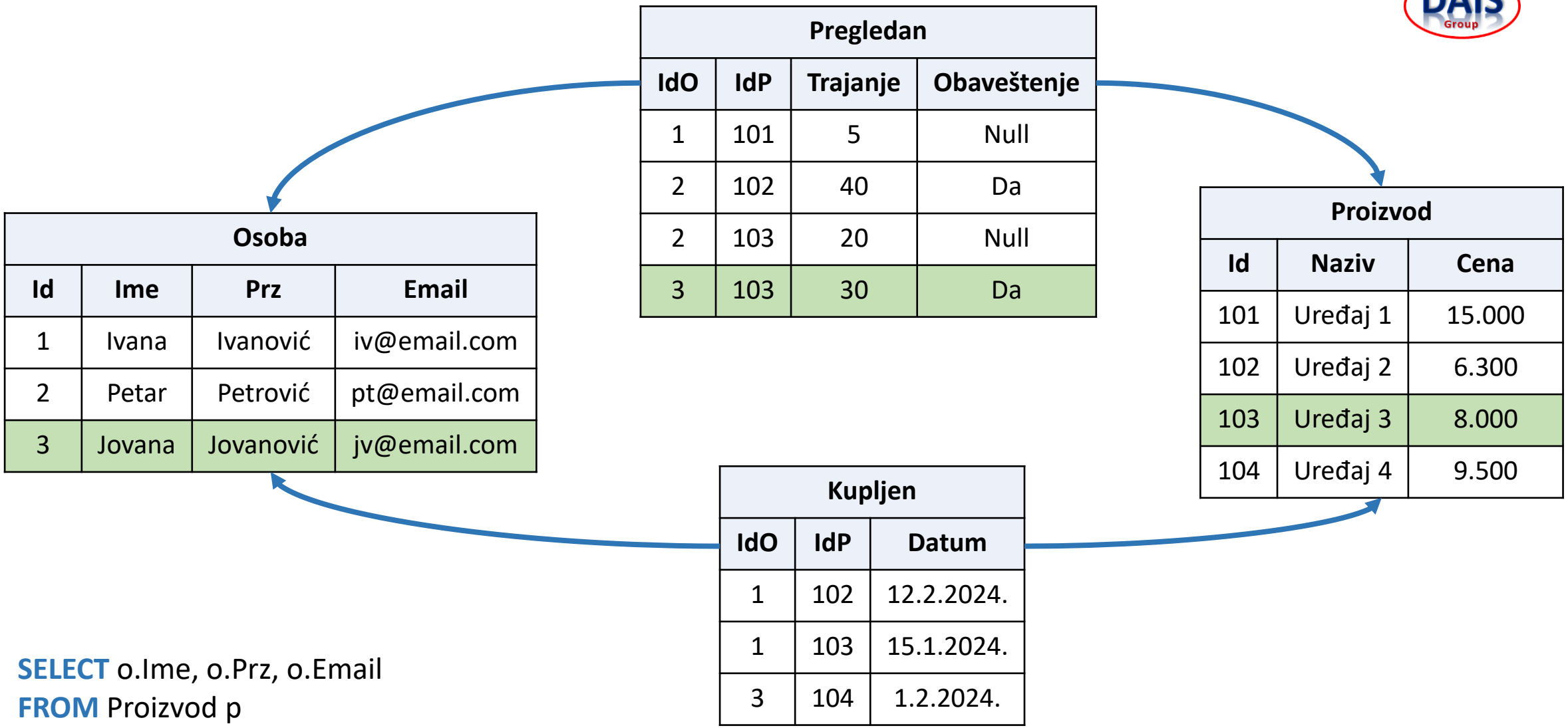
Baze podataka označenih grafova sa svojstvima

- **Baze podataka označenih grafova sa svojstvima** (engl. Labeled Property Graph) koriste model podataka koji sadrži čvorove, grane, svojstva, oznake i tipove
  - Čvorovi i grane imaju **internu strukturu**
- **Čvor** reprezentuje jedan realan entitet
  - Može da sadrži jednu ili više **oznaka**
    - Poput labela, tagova, tipa čvora
  - Može da sadrži **svojstva** u formi ključ-vrednost
- **Grana** reprezentuje odnos između dva realna entiteta (vezu između dva čvora)
  - Može biti **usmerena ili neusmerena** (dvosmerna)
  - Povezuju **tačno dva čvora**
    - Kod usmerenih grana u pitanju su **početni i krajnji čvor**
  - Može biti određenog **tipa** (odnosno, imenovana)
  - Može da sadrži **svojstva** u formi ključ-vrednost

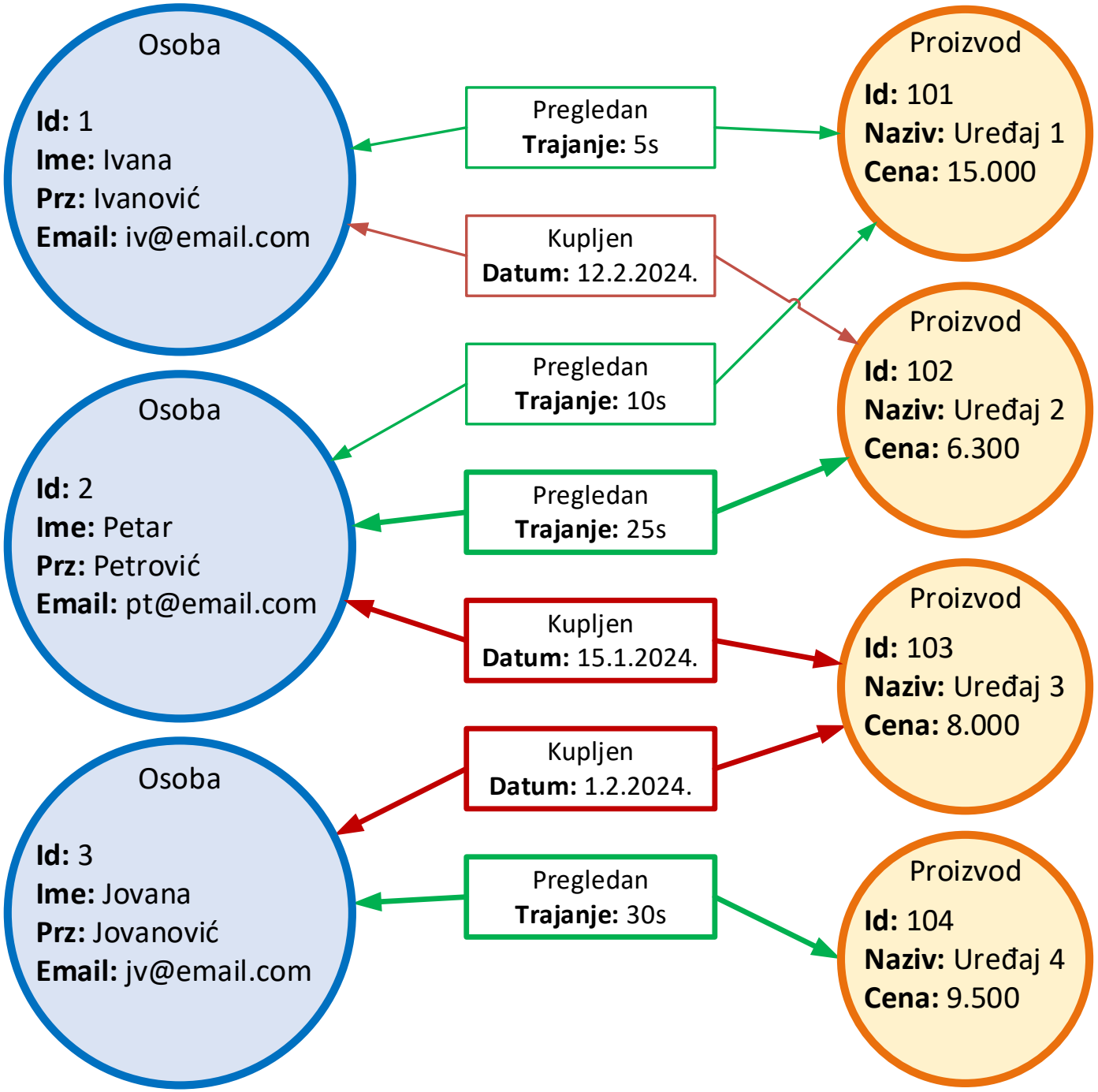


Primer 1: Uređaj 3 je na akciji. Potrebno je poslati obaveštenje svim osobama koje su zainteresovane za taj proizvod.

```
MATCH (p:Proizvod {naziv: 'Uređaj 3'})-
  [:Pregledan {Obaveštenje: 'Da'}]-(o:Osoba)
RETURN o.Ime, o.Prz, o.Email
```

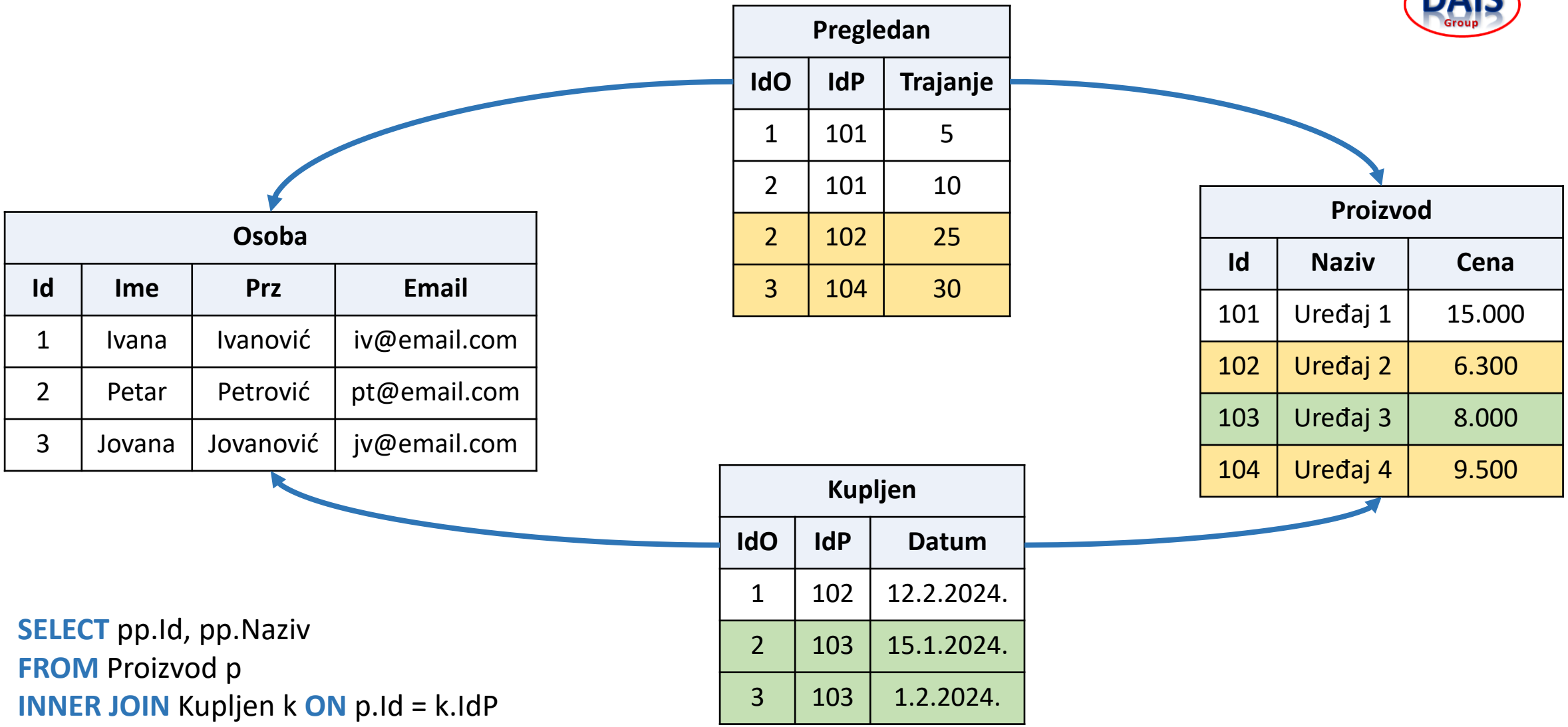



```
SELECT o.Ime, o.Prz, o.Email
FROM Proizvod p
INNER JOIN Pregledan pr ON p.Id = pr.IdP
INNER JOIN Osoba o ON pr.IdO = o.Id
WHERE p.Naziv = 'Uređaj 3' AND pr.Obaveštenje = 'Da';
```



Primer 2: Identifikovati sve proizvode za koje su zainteresovane osobe koje su kupile Uređaj 3. Pretpostavlja se da je osoba zainteresovana ukoliko je pregledala proizvod duže do 20 sekundi.

```
MATCH (p:Proizvod {naziv: 'Uređaj 3'})-
  [k:Kupljen]-(o:Osoba)-[pr:Pregledan]-
  (pp:Proizvod)
WHERE pr.trajanje > 20
RETURN pp.Id, pp.Naziv
```



Pregledan		
IdO	IdP	Trajanje
1	101	5
2	101	10
2	102	25
3	104	30

Osoba			
Id	Ime	Prz	Email
1	Ivana	Ivanović	iv@email.com
2	Petar	Petrović	pt@email.com
3	Jovana	Jovanović	jv@email.com

Proizvod		
Id	Naziv	Cena
101	Uređaj 1	15.000
102	Uređaj 2	6.300
103	Uređaj 3	8.000
104	Uređaj 4	9.500

Kupljen		
IdO	IdP	Datum
1	102	12.2.2024.
2	103	15.1.2024.
3	103	1.2.2024.

```
SELECT pp.Id, pp.Naziv
FROM Proizvod p
INNER JOIN Kupljen k ON p.Id = k.IdP
INNER JOIN Pregledan pr ON k.IdO = pr.IdO
INNER JOIN Proizvod pp ON pr.IdP = pp.Id
WHERE p.Naziv = 'Uređaj 3' AND pr.Trajanje > 20;
```

Grafske baze podataka

- Ne postoji definisana **šema baze podataka** (engl. *Schemaless Database*)
  - Obezbeđuje **fleksibilnost** prilikom unosa podataka
  - Ali i **potencijalnu nekonzistentnost** podataka
- Grafske baze podataka **optimizovane** su za rad kada postoji **velik broj veza** između realnih entiteta
  - Obezbeđujući **bolje performanse i fleksibilnost** u odnosu na relacione baze podataka prilikom višestrukih spajanja slogova
  - Mogu da koriste različite **algoritme obilaska grafa**

Grafske baze podataka

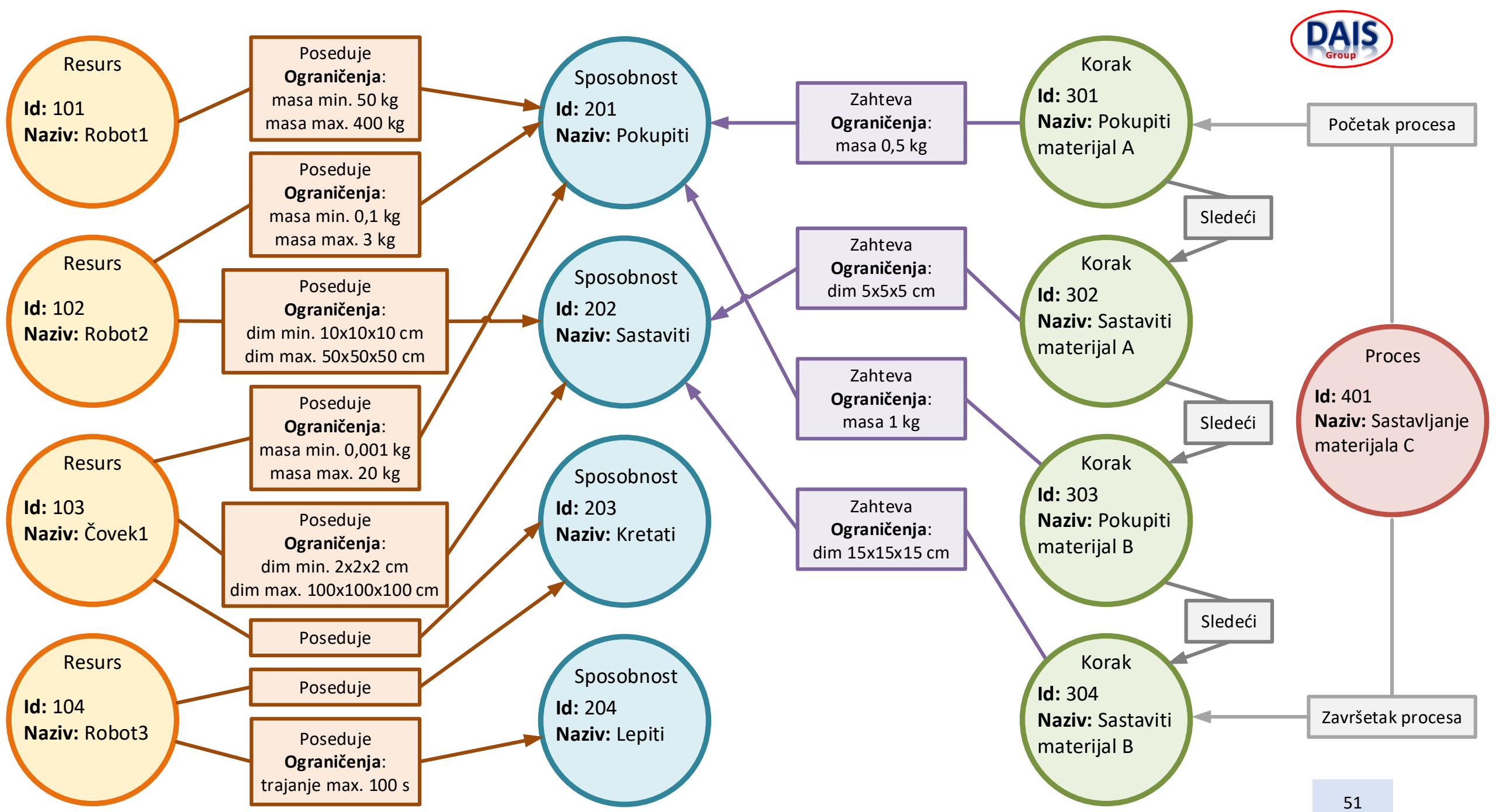
- Grafske baze podataka najčešće se koriste za potrebe **analize veza**
  - Kada je neophodno **pronaći veze** između različitih entiteta koje nisu očigledne („skrivene“ veze)
  - Ili napraviti relativno **dug lanac povezivanja** entiteta
  - Na primer, sa ciljem da se omoguće napredni sistemi preporuke
- Česte **primene** grafskih baza podataka:
  - **Pretrage** unutar grafa
  - Pronalazak najkraćeg, najdužeg ili najefikasnijeg **puta** sa stanovišta određenog kriterijuma
  - **Grupisanje** podataka na osnovu sličnosti određenih karakteristika
  - Identifikacija **particija** unutar grafa

Grafske baze podataka

- Grafske baze podataka **pogodno je koristiti**:
  - Kada postoji rad sa **veoma povezanim podacima**
  - Kada postoje **česte promene** strukture podataka ili samih podataka
    - Lako mogu da se menjaju i prilagođavaju, bez narušavanja postojećih funkcionalnosti
    - Performanse se ne manju mnogo povećanjem obima podataka
  - Kada je potrebno često **generisanje izveštaja** koji obuhvataju velik broj veza
    - Jer se **veze** između entiteta **skladište** u bazi podataka
      - Nije poput ograničenja referencijalnog integriteta u relacionoj bazi podataka o kojem brine SUBP
      - Veze nije potrebno procesirati prilikom izvršavanja upita poput spojeva u relacionoj bazi podataka
- Grafske baze podataka **nije pogodno koristiti**:
  - Ukoliko **ne postoji velik broj veza** između entiteta
  - Ukoliko **nije potrebno pronalaziti „skrivene“ veze** između entiteta
  - Ukoliko postoji potreba samo za skladištenjem **podataka tipa ključ-vrednost**
    - Postoji poseban tip baze podataka za skladištenje podataka tipa ključ-vrednost

Primer primene grafske baze podataka

- **Orkestracija dinamičke proizvodnje** u Industriji 4.0
  - Postoje **resursi** (roboti, mašine, ljudi) u fabrici koji **poseduju određene sposobnosti**
  - Postoje **procesni koraci** u okviru **proizvodnih procesa** koji **zahtevaju određene sposobnosti** da bi bili izvršeni
  - Neophodno je **povezati procesne korake i resurse** koji će ih izvršiti
  - Neophodno je poštovati zadata **ograničenja**


Resurs
Id: 101
Naziv: Robot1
Resurs
Id: 102
Naziv: Robot2
Resurs
Id: 103
Naziv: Čovek1
Resurs
Id: 104
Naziv: Robot3
Poseduje
Ograničenja:
masa min. 50 kg
masa max. 400 kg
Poseduje
Ograničenja:
masa min. 0,1 kg
masa max. 3 kg
Poseduje
Ograničenja:
dim min. 10x10x10 cm
dim max. 50x50x50 cm
Poseduje
Ograničenja:
masa min. 0,001 kg
masa max. 20 kg
Poseduje
Ograničenja:
dim min. 2x2x2 cm
dim max. 100x100x100 cm
Poseduje
Poseduje
Ograničenja:
trajanje max. 100 s
Sposobnost
Id: 201
Naziv: Pokupiti
Sposobnost
Id: 202
Naziv: Sastaviti
Sposobnost
Id: 203
Naziv: Kretati
Sposobnost
Id: 204
Naziv: Lepiti
Zahteva
Ograničenja:
masa 0,5 kg
Zahteva
Ograničenja:
dim 5x5x5 cm
Zahteva
Ograničenja:
masa 1 kg
Zahteva
Ograničenja:
dim 15x15x15 cm
Korak
Id: 301
Naziv: Pokupiti materijal A
Sledeći
Korak
Id: 302
Naziv: Sastaviti materijal A
Sledeći
Korak
Id: 303
Naziv: Pokupiti materijal B
Sledeći
Korak
Id: 304
Naziv: Sastaviti materijal B
Početak procesa
Proces
Id: 401
Naziv: Sastavljanje materijala C
Završetak procesa

Primer primene grafske baze podataka

- **Orkestracija dinamičke proizvodnje** u Industriji 4.0
  - Korak 301: Robot2 i Čovek1
  - Korak 302: Čovek1
  - Korak 303: Robot2 i Čovek1
  - Korak 304: Robot2 i Čovek1
  - Moguće je, na primer, da za izvršavanje procesa budu dodeljeni sledeći resursi:
    - Opcija 1: Čovek1 izvršava sve procesne korake
    - Opcija 2: Korake 301 i 302 izvršava Čovek1, a korake 303 i 304 izvršava Robot2

Sadržaj

Skladištenje podataka u grafskoj bazi podataka

- Struktura podataka u grafskim bazama podataka formirana je tako da bude **optimizovana za pretrage i obilaske grafa**
- Koncept **susedstva bez indeksa** (engl. *Index-Free Adjacency*) obezbeđuje da **svaki čvor direktno referencira svoje susedne čvorove**
  - **Grane** između čvorova **ugrađene su u strukturi podataka i skladište se** u memoriji
  - Na taj način **nisu neophodni globalni indeksi** na nivou grafa koji bi posredovali prilikom pretrage i pronalaženja povezanih čvorova
    - Svaki **čvor** moguće je posmatrati kao mali, lokalni indeks
  - Time **vreme izvršavanja upita** zavisi od **dela grafa koji se pretražuje**, a ne od celog grafa
    - Kao što bi bio slučaj sa upotrebom **globalnih indeksa ili relacionih baza podataka**
    - Odnosno, vreme izvršavanja upita **proporcionalno je obimu grafa koji se pretražuje**

Skladištenje podataka u grafskoj bazi podataka

- Susedstvo bez indeksa omogućava efikasno izvršavanje upita koji zahtevaju **povezivanje velikog broja različitih čvorova**
  - Omogućava efikasno sprovođenje algoritama **pretrage u dubinu i pretrage u širinu**
- Mnogi sistemi za upravljanje grafskom bazom podataka sprovode **keširanje podataka u radnu memoriju**, kako bi omogućili još bržu pretragu
  - SUBP brine da podaci kojima se **često pristupa** budu keširani

Primer skladištenja podataka baze podataka *Neo4j*

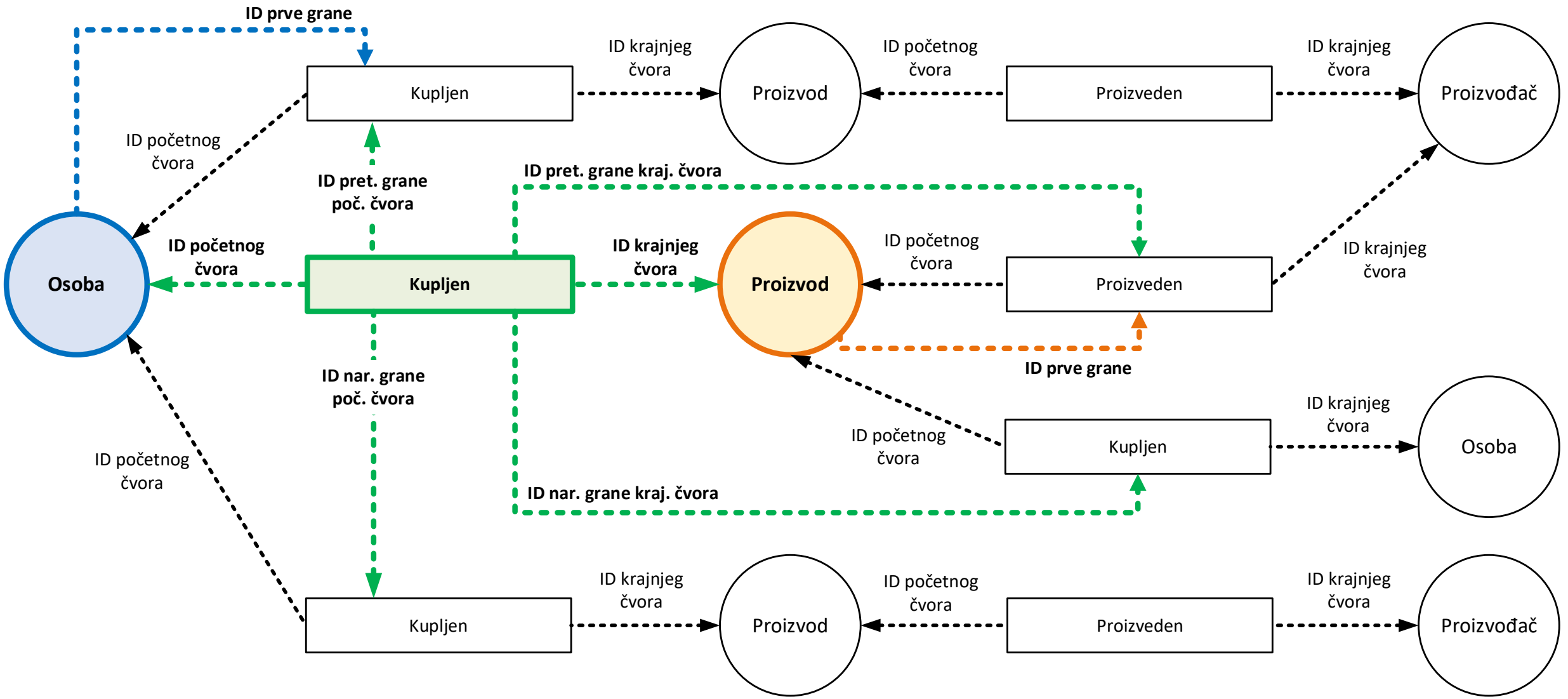
- Grafska baza podataka *Neo4j* čuva **elemente različitih tipova grafske strukture u različitim datotekama**
- Postoje **posebne datoteke** za čuvanje **čvorova, grana, svojstava, oznaka čvorova i tipova grana**, a svaki slog unutar datoteke je **fiksne dužine**
  - Što omogućava **jednostavan i direktan pristup zapisima datoteka** na osnovu **identifikatora zapisa pomnoženog sa dužinom zapisa**
  - Npr. svaki zapis u datoteci čvorova je fiksne dužine od 15 bajtova
    - Time bi čvor sa identifikatorom 10 počinjao na 150. bajtu

Primer skladištenja podataka baze podataka *Neo4j*

- (A) **Datoteka čvorova** skladišti zapise o pojedinačnim čvorovima
  - Svaki **zapis** u datoteci čvorova dužine je 15 bajtova i sadrži:
    - (1 B) **Indikator** da li je zapis u datoteci u **upotrebi** ili je slobodan za upis novog čvora
    - (4 B) **Identifikator prve grane** povezane sa čvorom susedom
      - Granama je moguće pristupiti u **datoteci grana**
    - (4 B) **Identifikator prvog svojstva** čvora
      - Svojstvima je moguće pristupiti u **datoteci svojstava**
    - (5 B) **Vrednosti oznaka** (kada je mali broj oznaka ili su oznake male dužine) **ili pokazivač ka zapisu u datoteci oznaka čvorova** (kada je velik broj oznaka)
      - Datoteka oznaka čvorova skladišti parove **identifikatore čvorova i njihovih oznaka**
    - (1 B) Različiti **indikatori** (engl. *flags*)
      - Npr. Indikator da li je u pitanju gusto povezani čvor (indikator za Neo4j koji algoritam pretrage da primeni)
  - **Identifikator čvora** određuje se **implicitno na osnovu pozicije** (engl. *offset*) zapisa unutar datoteke
    - Kako su zapisi fiksne dužine, uvek je poznato gde se čvor sa određenim **identifikatorom nalazi u datoteci**
  - **Identifikatore** je moguće posmatrati kao **pokazivače**

Primer skladištenja podataka baze podataka *Neo4j*

- (B) **Datoteka grana** skladišti zapise o pojedinačnim granama
  - Svaki **zapis** u datoteci grana dužine je 34 bajta i sadrži:
    - (1 B) **Indikator** da li je zapis u datoteci u **upotrebi** ili je slobodan za upis nove grane
    - (4 B) **Identifikator početnog čvora** i (4 B) **identifikator krajnjeg čvora grane**
    - (4 B) **Identifikator tipa grane**
      - Skladišten u **datoteci tipova grana**
    - (4 B) **Identifikator na prethodnu** i (4 B) **identifikator na narednu granu za početni čvor** tekuće grane
    - (4 B) **Identifikator na prethodnu** i (4 B) **identifikator na narednu granu za krajnji čvor** tekuće grane
    - (4 B) **Identifikator prvog svojstva** grane
    - (1 B) **Indikator** da li je **zapis grane prvi u lancu grana**
- Usled postojanja identifikatora **prethodnih i narednih grana** za početne i krajnje čvorove tekuće grane, moguće je **jednostavno kretati se kroz graf**
  - Moguće je brzo pronaći **sve grane određenog čvora**, nezavisno od **veličine celokupnog grafa**


ID prve grane
Kupljen
ID krajnjeg čvora
Proizvod
ID početnog čvora
Proizveden
ID krajnjeg čvora
Proizvođač
ID početnog čvora
ID pret. grane poč. čvora
ID pret. grane kraj. čvora
Osoba
ID početnog čvora
Kupljen
ID krajnjeg čvora
Proizvod
ID početnog čvora
Proizveden
ID krajnjeg čvora
ID prve grane
ID nar. grane poč. čvora
ID početnog čvora
Kupljen
ID krajnjeg čvora
Osoba
ID nar. grane kraj. čvora
ID početnog čvora
Kupljen
ID krajnjeg čvora
Proizvod
ID početnog čvora
Proizveden
ID krajnjeg čvora
Proizvođač

Primer skladištenja podataka baze podataka *Neo4j*

- **Datoteke čvorova i grana** sadrže podatke o **strukturi grafa** i nisu dodatno opterećene svojstvima
  - Omogućava da zapisi u datotekama budu **fiksne dužine**
  - Omogućava **efikasan obilazak grafa**
- Iz zapisa tekuće grane postoje **dvostruko spregnute liste grana** od početnog i krajnjeg čvora
  - Moguće je **kretati se kroz grane** u bilo kom smeru dok se ne pronađe grana od interesa
    - Moguće je pročitati njena svojstva prolaskom kroz **jednostruko spregnutu listu svojstava**
    - Moguće je pročitati **čvorove koje spaja**
  - **Upis i brisanje grana** je jednostavno proširenjem liste
  - **Jednostavan obilazak grafa** usled mogućnosti praćenja pokazivača

Primer skladištenja podataka baze podataka *Neo4j*

- (C) **Datoteka oznaka čvorova** skladišti zapise o parovima identifikatora čvorova i oznaka
  - Koristi se za skladištenje identifikatora oznaka za određeni čvor u slučaju kada oznake ne mogu da stanu u 5 bajtova unutar zapisa čvora dodeljenih za oznake
- (D) **Datoteka naziva oznaka** skladišti zapise o nazivima svih oznaka čvorova, a (E) **Datoteka naziva tipova** skladišti zapise o nazivima svih tipova grana
  - Predstavljaju registre svih oznaka, odnosno svih tipova
  - Koriste se kako **isti nazivi oznaka ili tipova ne bi bili skladišteni više puta** (posebno za svaki čvor, odnosno posebno za svaku granu)
    - Nazivi oznaka i tipovu mogu biti **duži stringovi** koji zauzimaju relativno veliki memorijski prostor
    - Na taj način, **svaki naziv se skladišti samo jednom**, dok se **jedinstveni identifikator** oznake ili tipa (od 4 B, koji je manji od stringa) koristi u datotekama oznaka čvorova, odnosno grana

Primer skladištenja podataka baze podataka *Neo4j*

- (F) **Datoteka svojstava** skladišti zapise o pojedinačnim svojstvima čvorova i grana u formi **parova ključ-vrednost** (1/3)
  - Svaki **zapis** u datoteci svojstava dužine je 41 bajt i sadrži:
    - (1 B) **Indikator** da li je zapis u datoteci u **upotrebi** ili je slobodan za upis novog svojstva
    - (32 B) **Četiri bloka svojstava**, po 8 bajtova svaki
    - (8 B) **Identifikator narednog zapisa u lancu svojstava**
      - Koristi se 8 B umesto 4 B (kako je bilo u datotekama čvorova i grana), jer su mogući **dugački lanci zapisa svojstava**, stoga se ostavlja prostor za duže adrese
      - Odnosno, 32-bitne adrese za prva svojstva u lancima i 64-bitne adrese za ostala svojstava u lancima
  - Unutar jednog zapisa, svako svojstvo može da zauzme **između jednog i četiri bloka svojstava**
    - To znači da u **jednom zapisu** može biti **najviše četiri svojstva skladišteno**
    - Ukoliko čvor ima **više svojstava**, koji ne mogu da stanu u jedan zapis, moraju biti kreirani dodatni zapisi u datoteci svojstava **povezani u jednostruko spregnutu listu**

Primer skladištenja podataka baze podataka *Neo4j*

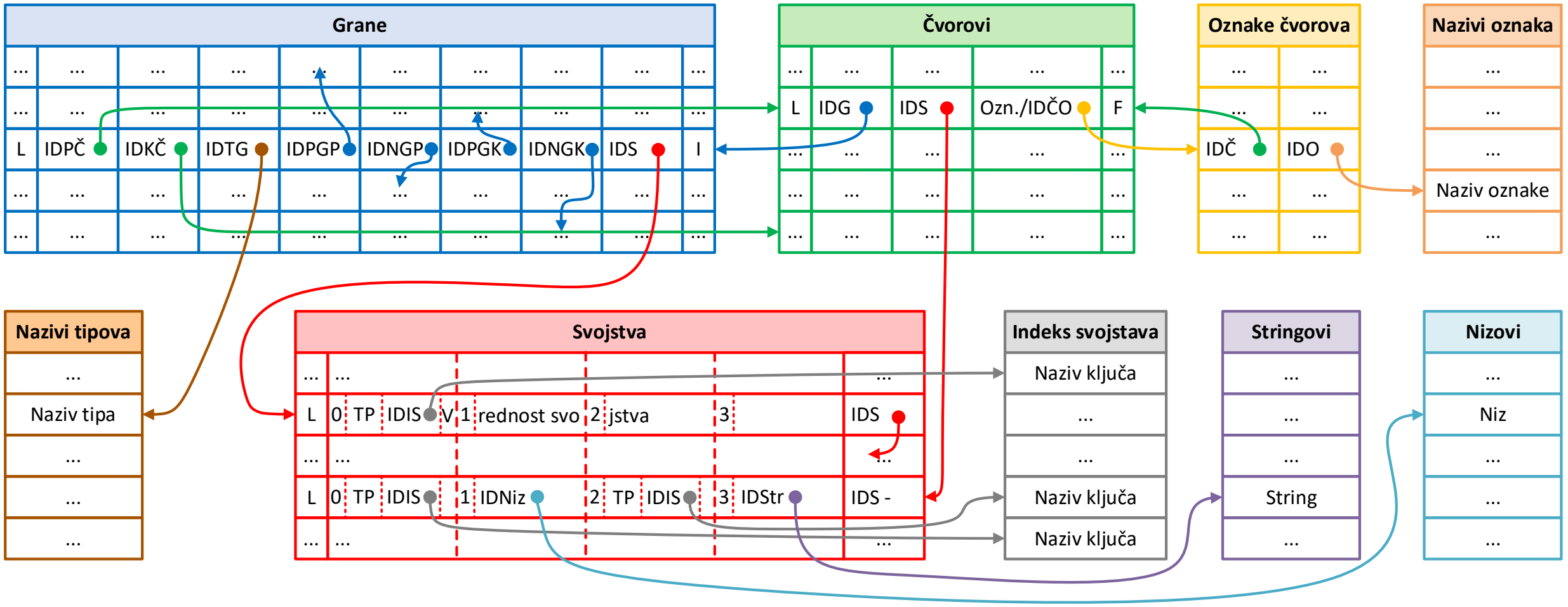
- (F) **Datoteka svojstava** skladišti zapise o pojedinačnim svojstvima čvorova i grana u formi **parova ključ-vrednost** (2/3)
  - Svako **svojstvo** sadrži **tip podatka, naziv (ključ) i vrednost**
    - **Tip podatka svojstva** može predstavljati jednostavne **primitivne tipove, ali i stringove i nizove**
    - **Ključ svojstva** skladišti se kao **identifikator zapisa u datoteci indeksa svojstava**
      - U kojoj su **skladišteni nazivi ključeva svih svojstava**, jer oni mogu biti **deljeni** među čvorovima i granama
      - Stoga se **poništava ponavljanje skladištenja naziva** istih svojstava ali u različitim čvorovima i granama
      - Umesto naziva ključa, u datoteci svojstava se skladišti **identifikator ključa svojstva**
      - Npr. u grafu društvene mreže, milioni čvorova korisnika će imati svojstva ime, prezime, email
    - **Vrednost svojstva** može biti skladištena **direktno u blokovima**, ili u slučaju stringova i nizova skladišti se **identifikator** zapisa **datoteke stringova**, odnosno **datoteke nizova**

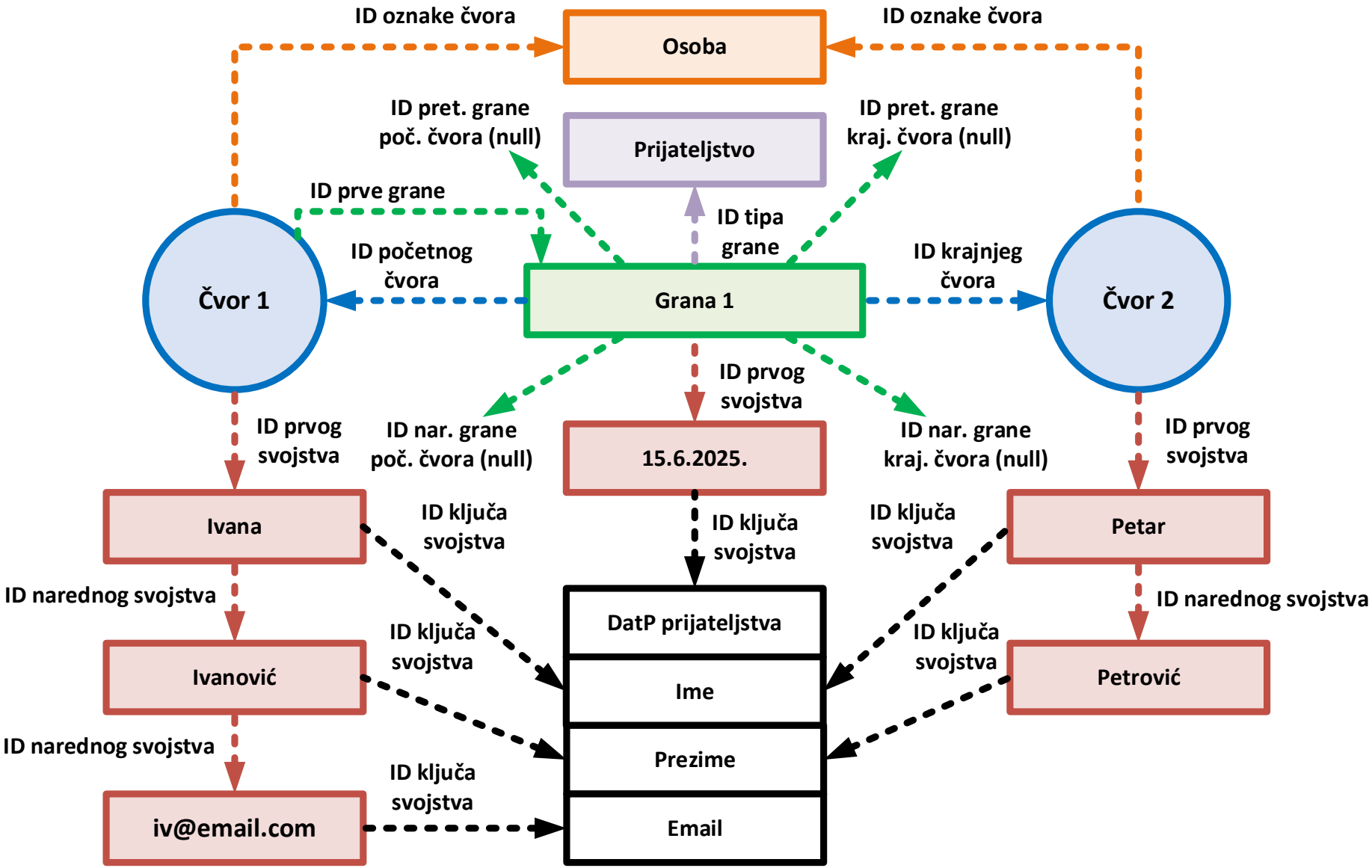
Primer skladištenja podataka baze podataka *Neo4j*

- (F) **Datoteka svojstava** skladišti zapise o pojedinačnim svojstvima čvorova i grana u formi **parova ključ-vrednost** (3/3)
  - **Blokovi svojstava** mogu sadržati tipove, ključeve i vrednosti jednog ili više svojstava (32 B):
    - (3,5 B) **Oznaku pozicije bloka** unutar zapisa (4 b) i tip podatka (24 b)
    - (4 B) **Identifikator ključa svojstva**
    - **Ostali bajtovi unutar blokova** rezervisani su za **vrednost svojstva** (preostala 4 bita mogu biti iskorišćena za vrednost tipa podatka *boolean*), a ukoliko je **vrednost velika** (ostali tipovi podataka), **dodatno se zauzima** jedan ili više narednih blokova
      - U zavisnosti od **tipa podatka**, vrednost će zauzimati određen prostor u blokovima
      - U slučaju stringova i nizova postojaće **identifikator zapisa** u datotekama stringova ili nizova
      - U slučaju **kratkih stringova ili nizova**, njihove vrednosti čuvaju se u blokovima svojstava

Primer skladištenja podataka baze podataka *Neo4j*

- (G) **Datoteka indeksa svojstava** skladišti nazive ključeva svojstava
  - **Tip podatka se ne skladišti sa nazivom ključa svojstva** jer za isti ključ mogu biti **različiti tipovi podataka** u različitim čvorovima ili granama, usled **nepostojanja šeme baze podataka**
    - Npr. svojstvo sa ključem „godine“ može imati vrednost „dvadeset“ ili 20 u različitim čvorovima
  - Datoteka indeksa svojstava služi da **dugačak naziv ključa preslika u kratak identifikator** koji će biti korišćen u datoteci svojstava
- (H) **Datoteka stringova** skladišti vrednosti stringova, a (I) **Datoteka nizova** skladišti vrednosti nizova
  - **Zapisi dinamičkih stringova i nizova** uvezani su u **listu zapisa fiksne dužine od 128 B**
    - U slučaju da je potrebno skladištiti **veoma dugačke stringove ili nizove**, moguće je da pojedinačni string ili niz zauzme **više od jednog zapisa**
  - Usled **velikog obima i varijabilne dužine**, stringovi i nizovi skladište se u posebno datoteke stringova i nizova, gde dobijaju **identifikator korišćen kao referenca u datoteci svojstava**


Grane
L
IDPČ
IDKČ
IDTG
IDPGP
IDNGP
IDPGK
IDNGK
IDS
I
Čvorovi
L
IDG
IDS
Ozn./IDČO
F
Oznake čvorova
IDČ
IDO
Nazivi oznaka
Naziv oznake
Nazivi tipova
Naziv tipa
Svojstva
L 0 TP IDIS V 1 rednost svo 2 jstva 3
IDS
L 0 TP IDIS 1 IDNiz 2 TP IDIS 3 IDStr
IDS -
Indeks svojstava
Naziv ključa
Naziv ključa
Naziv ključa
Stringovi
String
Nizovi
Niz


ID oznake čvora
Osoba
ID oznake čvora
ID pret. grane poč. čvora (null)
Prijateljstvo
ID pret. grane kraj. čvora (null)
ID prve grane
ID tipa grane
ID početnog čvora
Čvor 1
Grana 1
ID krajnjeg čvora
Čvor 2
ID prvog svojstva
ID nar. grane poč. čvora (null)
15.6.2025.
ID nar. grane kraj. čvora (null)
ID prvog svojstva
ID prvog svojstva
Ivana
ID ključa svojstva
ID ključa svojstva
ID ključa svojstva
Petar
ID narednog svojstva
DatP prijateljstva
ID narednog svojstva
ID ključa svojstva
ID ključa svojstva
Ivanović
Ime
Petrović
ID narednog svojstva
Prezime
ID ključa svojstva
iv@email.com
Email

Primer skladištenja podataka baze podataka *Neo4j*

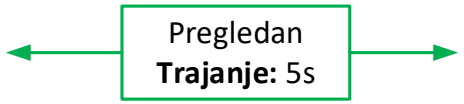
- Moguća **primena različitih indeksa** u bazi podataka *Neo4j*
- **Indeksi za performantnu pretragu** (engl. *Search-Performance Index*) služe da ubrzaju pretragu grafa na osnovu **tačno zadatih vrednosti**
  - **Indeks opsega** (engl. *Range Index*) – za pretragu vrednosti koje su u određenom **opsegu**
    - Pogodan kada je u pitanju svojstvo čije su vrednosti **uređene, odnosno uporedive**
  - **Indeks teksta** (engl. *Text Index*) – za pretragu sadržaja svojstava **tipa string**
    - Npr. upit da li string sadrži neki deo teksta ili se završava sa nekim tekstom
  - **Indeks tačaka u prostoru** (engl. *Point Index*) – za pretragu **tačaka u prostoru**
    - Za filtriranje tačaka na osnovu **rastojanja** ili kada se tačke nalaze u okviru **zadatog prostora**
  - **Indeks pretrage tokena** (engl. *Token Lookup Index*) – za pretragu **po oznakama čvorova ili po tipovima grana**
    - Na nivou baze podataka **automatski se kreiraju dva ovakva indeksa** – jedan za sve oznake čvorova grafa, a drugi za sve tipove grana grafa
- ...

Primer skladištenja podataka baze podataka *Neo4j*

- …
- **Semantički indeksi** (engl. *Semantic Index*) služe da omoguće pretragu grafa na osnovu **semantike upita**
  - **Indeks za punu pretragu teksta** (engl. *Full-Text Index*) – za semantičku pretragu **sadržaja svojstava tipa string**
    - Vraća **ocenu** sličnosti teksta upita i sadržaja svojstva tipa string
    - Vrši **tokenizaciju teksta**
  - **Indeks vektora** (engl. *Vector Index*) – za semantičku pretragu **sličnosti upita i elemenata grafa**
    - Elementi grafa (npr. čvorovi i svojstva) predstavljeni su kao **vektori u visokodimenzionalnom prostoru**
    - Primenom **metrika sličnosti**, moguće je pronaći vektore elemenata grafa koji su najbliži vektoru upita
  - Oba tipa indeksa koriste biblioteku ***Apache Lucene*** i funkcionišu na osnovu **procene sličnosti** između korisničkog upita i sadržaja grafske baze podataka

Sadržaj

Koncepti grafske i relacione baze podataka

Grafska baza podataka	Relaciona baza podataka
Oznaka 	Naziv šeme relacije (naziv tabele)
Svojstvo 	Obeležje (kolona)
Čvor 	N-torka (slog, red)
Grana 	Ograničenje referencijalnog integriteta (strani ključ)

Proizvod		
Id	**Naziv**	**Cena**

Id	**Naziv**	**Cena**
101	Uređaj 1	15.000

Id	**Naziv**	**Cena**
101	Uređaj 1	15.000

IdO	**IdP**	**Trajanje**
1	101	5

Relacione i grafske baze podataka

- **Relacione baze podataka** mogu da čuvaju veze između entiteta uz pomoć **ograničenja referencijalnog integriteta**
  - Međutim, **spajanje** relacija je **skupa operacija**, posebno ukoliko je potrebno napraviti veliki broj spojeva
- **Grafske baze podataka optimizovane** su za rad sa **spojevima**
  - Moguće je efikasno **izvršavanje** upita koji obuhvataju spojeve
  - Grafske baze podataka **nemaju ograničenje referencijalnog integriteta i operacije spajanja** kao u relacionoj bazi podataka
  - Spojevi, odnosno **grane**, predstavljaju **zapise** koji se čuvaju u bazi podataka
    - **Ne predstavljaju ograničenja** o čijem zadovoljenju brine SUBP

Primena relacione i grafske baze podataka

- Primena **relacione baze podataka**:
  - **Struktura** podataka, odnosno šema baze podataka, **jasno definisana i ne menja se** (često)
  - Relativno **mali broj veza** između entiteta
    - **Ne postoji** potreba za značajnom **analizom veza** između entiteta
  - **Analiza** velikog broja slogova **u jednoj relaciji (tabeli)**
- Primena **grafske baze podataka**:
  - **Struktura** podataka **ne mora biti jasno definisana**
    - **Česte promene poslovanja** i potencijalne izmene strukture podataka
  - Relativno **velik broj veza** između entiteta
    - **Postoji** potreba za **analizom veza** između entiteta
    - **Veze** između entiteta poseduju **dodatna svojstva i semantiku**
  - Potreba za **horizontalnim skaliranjem**

Primer performansi relacione i grafske baze podataka

- **Primer poređenja performansi** izvršavanja upita u relacionoj i grafskoj bazi podataka
  - Korišćeni relacioni ***MySQL*** i grafski ***Neo4j*** sistemi za upravljanje bazama podataka
  - Primer: pronaći ukupan **broj prijatelja do 5 nivoa** dubine pretrage
  - Prebrojavali su prijatelje kako ne bi **opteretili procesor ili memoriju** učitavanjem podataka
  - Baze podataka sadržale su **1.000 ili 1.000.000 korisnika**, gde svaki ima u proseku po **50 prijatelja**
  - Korišćen računar sa Intel i7 procesorom i 8 GB RAM memorije

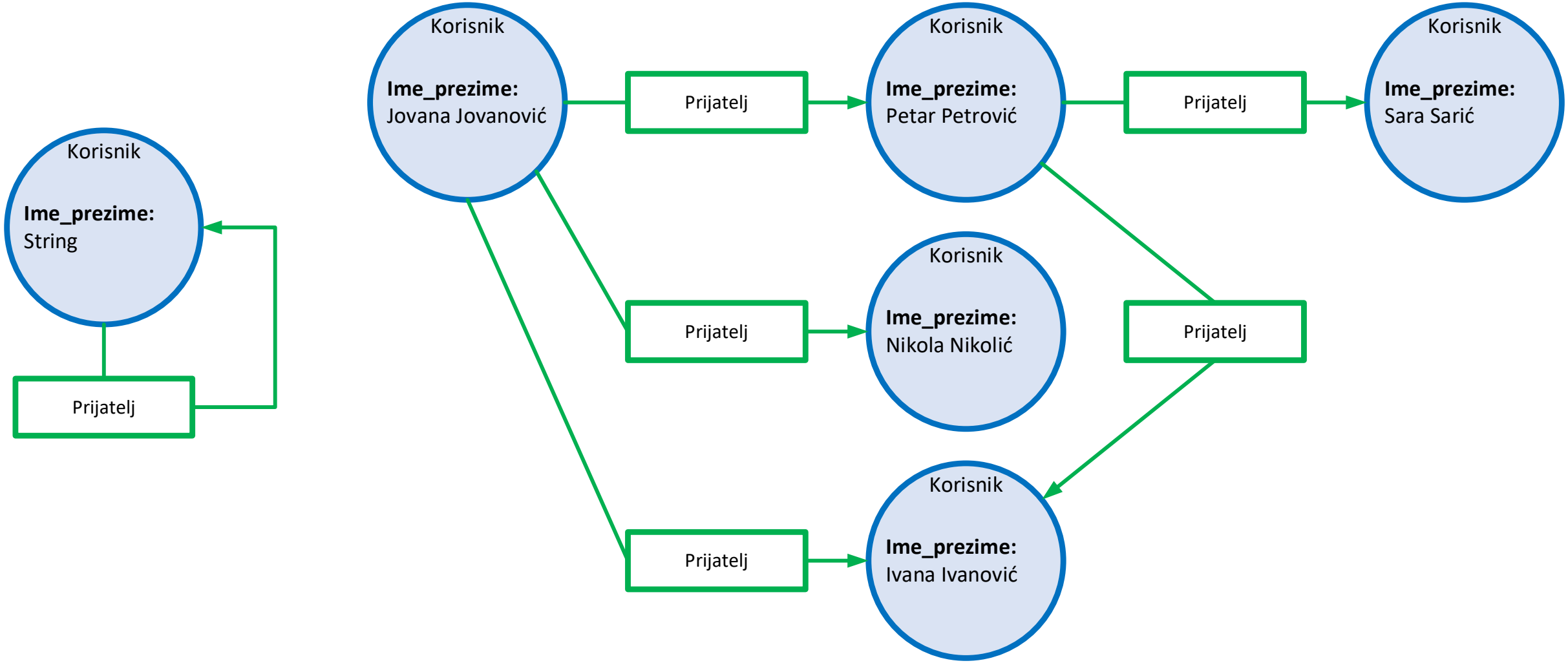
Izvor: Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, Jonas Partner, Neo4j in Action, 1st Edition, Manning Publications Co., 2015.

Primer performansi relacione i grafske baze podataka

Korisnik	
id	**ime_prezime**
1	Jovana Jovanović
2	Petar Petrović
3	Ivana Ivanović
4	Nikola Nikolić
5	Sara Sarić
…	…

Prijateljstvo		
id	**korisnik_1**	**korisnik_2**
101	1	2
102	1	3
103	1	4
104	2	3
105	2	5
…	…	…

Primer performansi relacione i grafske baze podataka



Primer performansi relacione i grafske baze podataka

- Performanse izvršavanja upita prebrojavanja prijatelja na **1.000 korisnika i 50 prijatelja po korisniku**

	1.000 korisnika 50 prijatelja po korisniku			
Dubina	***MySQL* vreme (s)**	***Neo4j* vreme (s)**	**Odnos Relaciona/Grafska**	**Broj prijatelja**
2	**0,028**	0,04	0,7	~900
3	0,213	**0,06**	3,55	~999
4	10,273	**0,07**	146,76	~999
5	92,613	**0,07**	1.323,04	~999

Izvor: Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, Jonas Partner, Neo4j in Action, 1st Edition, Manning Publications Co., 2015.

Primer performansi relacione i grafske baze podataka

- U slučaju 1.000 korisnika i 50 prijatelja po korisniku, tabela prijateljstva sadrži **50.000 slogova**
- **Relaciona baza podataka** *MySQL*
  - Optimizovana je za upite sa **spojem dve tabele** (potencijalna upotreba indeksa nastalih usled ograničenja primarnog ključa)
  - Upotreba **spojeva više tabela značajno smanjuje performanse** izvršavanja upita
  - Svaki spoj između tabela pravi **dekartov proizvod** između slogova tabela koje se spajaju
  - Kako bi bili pronađeni svi prijatelji **do 5. nivoa**, potrebno je **50.000^5 spojeva**
    - Potrebno mnogo procesorske snage prilikom spajanja, a dobija se **velik broj redundantnih spojeva**
    - Rezultat izvršavanja upita iznosi skoro svih 1.000 korisnika, a **preko 99% spojeva je nepotrebno** jer će se korisnici kao prijatelji ponavljati
- **Grafska baza podataka** *Neo4j*
  - Za razliku od *MySQL*, krenuće sa pretragom čvorova i **zaustaviti se kada dođe do ponavljanja korisnika**
  - Koristi **algoritme obilaska grafa**
    - Proces obilaska grafa je **lokalizovan**
    - Zahteva pristup samo **neophodnim entitetima**, bez potrebe za skupim operacijama spajanja nad celim skupovima entiteta poput operacije spajanja u relacionoj bazi podataka

Primer performansi relacione i grafske baze podataka

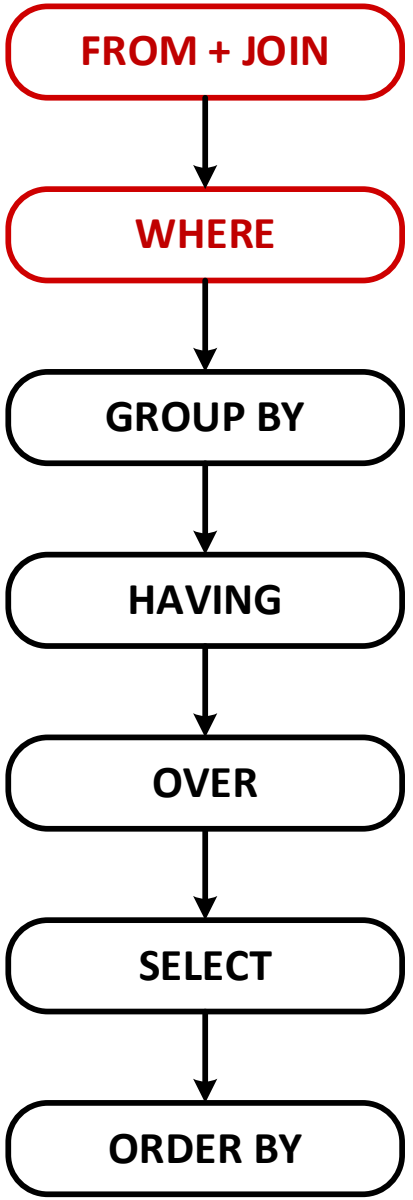
- Performanse izvršavanja upita prebrojavanja prijatelja na **1.000.000 korisnika i 50 prijatelja po korisniku**

	1.000.000 korisnika 50 prijatelja po korisniku			
Dubina	***MySQL* vreme (s)**	***Neo4j* vreme (s)**	**Odnos Relaciona/Grafska**	**Broj prijatelja**
2	0,016	**0,01**	1,6	~2.500
3	30,267	**0,168**	180,16	(relaciona) ~125.000 (graf) ~110.000
4	1.543,505	**1,359**	1.135,77	~600.000
5	Prekinuto izvršavanje nakon jednog sata	**2,132**	>> 1.688	~800.000

Izvor: Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, Jonas Partner, Neo4j in Action, 1st Edition, Manning Publications Co., 2015.

Izvršavanje upita

- **Izvršavanje upita** relacione baze podataka:
  - Operacija **spajanja** (***JOIN***) tabela stvara **dekartov proizvod** slogova u spoju
  - Nakon dekartovog proizvoda sledi **filtriranje** slogova (***WHERE***)
  - **Spoj je skupa operacija, a takođe i filtriranje velikog broja slogova**



Poređenje karakteristika grafske i relacione baze podataka

Grafska baza podataka	Relaciona baza podataka
Ne mora postojati šema baze podataka	**Postoji jasno definisana šema baze podataka**
Porastom broja entiteta i njihovih veza, **performanse** izvršavanja upita često **nisu mnogo promenjene** – mogućnost **lokalizacije upita**	**Porastom** broja entiteta i njihovih veza, **performanse** izvršavanja upita **mogu da opadnu** – često je neophodno **pretražiti celu tabelu** bez obzira na broj entiteta i obuhvatiti **veći broj tabela u spoju**
Efikasne prilikom **generisanja izveštaja** – moguće je relativno **brzo procesirati entitete i njihove veze** jer se čuvaju kao zapisi u datotekama, ali postoji mogućnost **nedostatka** određenih podataka zbog nepostojanja šeme baze podataka	Efikasne prilikom **generisanja izveštaja** – usled postojanja **jasno definisane** šeme baze podataka, ali postoji mogućnost **lošijih performansi** prilikom velikog broja spojeva između entiteta
Mogu zauzeti **više memorijskog prostora** (npr. veze se čuvaju kao zapisi u datoteci grana)	Mogu zauzeti **manje memorijskog prostora** (npr. veze su predstavljene ograničenjima referencijalnog integriteta)

Poređenje karakteristika grafske i relacione baze podataka

Grafska baza podataka	Relaciona baza podataka
Efikasne za **pretragu u dubinu ili u širinu** i **otkrivanje** povezanih entiteta i **„skrivenih“ veza**	Efikasne za rukovanje **strukturiranim podacima**, posebno prilikom **filtriranja** podataka
Primena kada postoje česte, **dinamičke promene**, poput sistema za preporuku	**Primena** kada je važno **očuvanje integriteta podataka**, poput transakcija i osetljivih podataka
Jednostavniji upiti prilikom višestrukog spajanja; Primenom jezika grafske baze podataka moguće je postaviti upit **direktno nad čvorovima i vezama**	**Kompleksniji upiti** prilikom višestrukog spajanja; Primenom jezika relacione baze podataka potrebno je izvršiti spajanje tabela pomoću **stranih ključeva**, koristiti **ugnježdene upite** i/ili **klauzulu *WITH***
Za potrebe **skalabilnosti**, moguće relativno jednostavno **horizontalno skaliranje** podelom grafa na različite delove koji bi bili čuvani na različitim serverima i potencijalno izvršiti delove upita u paraleli	Za potrebe **skalabilnosti**, često je jednostavnije **vertikalno skaliranje** proširenjem hardvera, umesto horizontalnog skaliranja podelom tabela na delove

Sadržaj

Jezici za rad sa grafskom bazom podataka

- Postoje **razni jezici** za rad sa grafskim bazama podataka koji su **specifični za određeni SUBP**
  - Koriste **različitu sintaksu** jezika
  - Izazovno prilikom **prelaska** sa jednog SUBP-a na drugi
- **Popularni jezici** grafskih baza podataka:
  - ***Cypher*** – jezik razvijen za potrebe *Neo4j*
  - ***Gremlin*** – jezik razvijen u okviru radnog okvira *Apache TinkerPop*
  - ***SPARQL*** – jezik razvijen za potrebe grafova znanja u formatu *RDF*
- ***Graph Query Language (GQL)***
  - **Standardizovani upitni jezik** za baze podataka **modela grafa sa svojstvima**
  - **Standard:** *ISO/IEC FDIS 39075 - Information technology - Database languages*
  - Započeta **standardizacija grafskog upitnog jezika** 2019. godine, a završena 2024. godine
    - **Prethodni jezik iz baza podataka** koji je *ISO* standardizovao bio je *SQL* 1987. godine
  - **Deklarativan upitni jezik** sličan jeziku ***SQL***, a takođe inspirisan jezikom ***Cypher***

Sadržaj

Literatura

- Ian Robinson, Jim Webber, Emil Eifrem, Graph Databases: New Opportunities for Connected Data, 2nd Edition, O'Reilly Media, Inc., 2015.
- Jesús Barrasa, Jim Webber, Building Knowledge Graphs: A Practitioner's Guide, 1st Edition, O'Reilly Media, Inc., 2023.
- Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, Jonas Partner, Neo4j in Action, 1st Edition, Manning Publications Co., 2015.
- Dave Bechberger, Josh Perryman, Graph Databases in Action: Examples in Gremlin, 1st Edition, Manning Publications Co., 2020.

Napredne arhitekture informacionih sistema

Grafske baze podataka

Pitanja?

Predmetni nastavnik:
dr Marko Vještica

DAIS
Data Science and Information Systems Group