Baze podataka vježbe

Pravilo 1:N - Ako je odnos između entiteta **E1** i **E2** tipa **1:N** (jedan-na-više), tada relacija za **E2** treba uključiti primarne atribute (primarni ključ) od **E1**.

Pravilo N:M - Ako je odnos između entiteta tipa **N:M** (više-na-više), uvijek se prikazuju posebnom relacijom koja uključuje primarne atribute oba entiteta, te još možda dodatne koje sama veza ima

Pravilo 1:1 - Ako je odnos između entiteta **E1** i **E2** tipa 1:1 (jedan-na-jedan), tada u jednu relaciju treba uključiti primarne atribute (primarni ključ) druge relacije (u relaciju u koju ima "više smisla").

Primjer 1

```
group: RDS-01
student = {
 id_student, ime, prezime, godiste
 1, 'Marko', 'Marić', 1997
 2, 'Toni', 'Milovan', 2000
 3, 'Ime', 'Prezime', 2003
 4, 'Ime2', 'Prezime', 2002
}
```

Rezultat 1

```
// 1. studente koji su rođeni nakon 2000. godine
σ godiste > 2000 student
// 2. studente koji se prezivaju "Marić"
 σ prezime= 'Marić' student
// 3. studente koji se prezivaju "Marić" ili su rođeni nakon i uključujući 2000. godinu
 σ prezime= 'Marić' ∨  godiste > 2000 student
// 4. samo atribut id studenata koji se prezivaju "Marić" ili su rođeni nakon i uključujući 2000. godinu
 π id_student ( σ prezime= 'Marić' ∨  godiste > 2000 (student))
```

Primjer 2

```
group: RDS-03
kolegij = {
 id_kolegij, naziv, semestar_izvodenja, sati_nastave
 1, 'Programiranje', 1, 30
 2, 'Baze podataka 1', 2, 30
 3, 'Baze podataka 2', 3, 30
 4, 'Napredne tehnike programiranja', 3, 30
}
student = {
 id_student, ime, prezime
 11, 'Marko', 'Marić'
 12, 'Toni', 'Milovan'
 13, 'Ime', 'Prezime'
}
student_na_kolegiju = {
 id_student_na_kolegiju, id_kolegij, id_student, ocjena
 21, 1, 11, 4
 22, 1, 12, 5
 23, 1, 13, 3
 24, 2, 12, 4
 25, 2, 13, 5
 26, 3, 12, 4
}
```

Rezultat 2

```
// 1. id-eve studenata koji su položili barem jedan kolegij sa ocjenom 5
π id_student (σ ocjena > 4 (student_na_kolegiju))
// 2. id-eve kolegija i id-eve studenata koji nisu uspješno položili (ocjena = 1)
 π id_student, id_kolegij (σ ocjena = 1 (student_na_kolegiju))
// 3. id-eve kolegija i id-eve studenata koji su kolegij položili sa ocjenom u rasponu [2-4]
 π id_student, id_kolegij (σ ocjena > 2 ∨ ocjena < 4 (student_na_kolegiju))
```

Primjer 3

```
group: RDS-04
korisnik = {
 id_korisnik, email, ime, prezime
 1, 'marko.maric@email.hr', 'Marko', 'Marić'
 2, 'toni.milovan@email.hr', 'Toni', 'Milovan'
 3, 'ime.prezime@email.hr', 'Ime', 'Prezime'
 4, 'ime2.prezime@email.hr', 'Ime2', 'Prezime'
}
video = {
 id_video, naslov, broj_pregleda, video_sadrzaj
 11, 'Formula 1 Australian Grand Prix', 500, 'video1'
 12, 'Learn Relational Algebra: Part II', 30, 'video2'
 13, '*** Music Video', 250, 'video3'
 14, 'Prezentacija projekta BP1', 300, 'video4'
}
komentar = {
 id_komentar, id_video, id_korisnik, datum, sadrzaj, id_nad_komentar
 21, 11, 1, '02.01.2020.', 'First!', NULL
 22, 11, 1, '04.01.2020.', 'I was first, just saying', NULL
 23, 11, 3, '04.01.2020.', 'What happened at 02:00?', NULL
 24, 12, 1, '07.01.2020.', 'What does "sigma" actually do?', NULL
 25, 12, 2, '07.01.2020.', 'This video was very helpful. Thanks!', NULL
 26, 12, 3, '07.01.2020.', 'It filter tuples based on the condition', 24
 27, 12, 3, '07.01.2020.', 'Basically, it is just a filter', 24
 28, 13, 1, '09.01.2020.', 'She sings amazing.', NULL
}
```

Rezultat 4

```
// 1. id-eve videa koji imaju barem jedan komentar
 π id_video (video) - π id_video (komentar)
// 2. id videa koji imaju barem 300 pregleda
π id_video (σ broj_pregleda> 300 (video))
// 3.id videa koji imaju barem 300 pregleda, i niti jedan komentar
π id_video (σ broj_pregleda> 300 (video)) - π id_video (komentar)
// 4. id korisnika koji nisu objavili niti jedan komentar, a ime im je "Ime2"
π id_korisnik (σ ime= 'Ime2' korisnik) - π id_korisnik (komentar)
// 5. prikazi sve videa sa dodatnim stupcem uvecavn_broj_pregleda prikazivati broj pregleda uvecan za 10
π id_video, naslov, broj_pregleda, broj_pregleda+10 -> uvucen_broj_pregleda (video)
```

Primjer 4, 5

```
group: RDS-02
student = {
 id_student, ime, prezime, godiste, id_grad
 1, 'Marko', 'Marić', 1997, 22
 2, 'Toni', 'Milovan', 2000, 21
 3, 'Ime', 'Prezime', 2003, 21
 4, 'Ime2', 'Prezime', 2002, 22
 4, 'Tea', 'Bibić', 1999, NULL
}
nastavnik = {
 id_nastavnik, ime, prezime, titula, id_grad
 11, 'Mateo', 'Borić', 'dr.sc.', 21
 12, 'Pero', 'Perić', 'doc.', 21
 13, 'Ime', 'Prezime', 'mag.', 23
}
grad = {
 id_grad, naziv, postanski_broj
 21, 'Pula', 52100
 22, 'Rijeka', 51000
 23, 'Zagreb', 10000
}
```

Rezultat 4

```
// 1.Ispiši zajednička prezimena nastavnika i studenata
 π prezime (student) ∩  π prezime (nastavnik)
// 2.Ispiši sva prezimena nastavnika i studenata
 π prezime (student) ∪  π prezime (nastavnik)
// 3.Ispiši sva prezimena nastavnika i studenata, koji su rođeni nakon 2000
 π prezime (σ godiste>2000 (student)) ∪  π prezime (nastavnik)
// 4.sva prezimena nastavnika koji ne dijele prezime s nijednim studentom
 π prezime (nastavnik) -  π prezime (student)
```

Rezultat 5

```
// 1. Poveži relacije student i grad kako bi se povezale u svim mogućim kombinacijama
 σ student.id_grad = grad.id_grad (student × grad)
// 2. poveži relaciju student i grad kako bi se relacije povezale u svim mogućim kombinacijama
 student ⨝ student.id_grad = grad.id_grad grad
// Poveži relacije student student kako bi se relacije povezale u svim mogućim kombinacijama
student× ρ student1 student
// Prikaži sva prezimena (studenata ili nastavnika) koja se pojavljuju u svim gradovima
(π prezime, id_grad (student) ∪ π prezime, id_grad (nastavnik)) ÷ π id_grad  (grad)
```

```
 1. π naslov (σ korisnik.prezime = 'Marić' (korisnik) ⨝ video ⨝ ocjena)
 2. γ count(id_korisnik) -> broj_korisnika (korisnik)
 3. π id_video -> id_video, naslov -> naslov, broj_pregleda (video) ⨝ γ min(broj_pregleda) -> broj_pregleda (video)
 4. π id_video -> id_video, naslov -> naslov (video) ⨝ γ id_video; count(ocjena) -> broj_ocjena (ocjena)
 5. π id_korisnik, ime, prezime (σ prezime = 'Marić' ∨ prezime = 'Milovan' korisnik) ⨝
(γ id_korisnik; count(ocjena) -> broj_likeova (σ ocjena = 'L' (ocjena)))
```

```
-- γ max(stanje) -> najveci_iznos_stednja (stednja) ⋈ γ max(stanje) -> najveci_iznos_tekuci (tekuci_racun)

-- γ id_gradanin; avg(stanje) -> prosjecni_iznos (tekuci_racun)

-- prikaži sve štednje, te dodatno iznos koji će biti dobiven nakon obračunavanja kamatne stope
(stanje + stanje x kamatna_stopa/100) (rezultat: id_stednja, id_gradanin, broj_racuna, stanje, kamatna_stopa, iznos_nakon_obracuna)

-- π id_stednja, id_gradanin, broj_racuna, stanje, kamatna_stopa, stanje+stanje * kamatna_stopa /100 ->ukupno (stednja)

-- prikaži sve građanine, broj njihovih tekućih računa i broj štednji
(rezultat: id_gradanin, ime, prezime, broj_tekucih_racuna, broj_stednji)

-- π ime, prezime, id_gradanin (gradanin) ⋈ (γ id_gradanin; count(id_tekuci)->broj_tekucih_racuna (tekuci_racun)
⋈ γ id_gradanin; count(id_stednja)->broj_stednji (stednja))
```

```
// 1. prčune koje je izdao zaposlenik sa prezimenom 'Perić' (rezultat: id_racun, broj, datum_izdavanja)
  π id_racun,broj,datum_izdavanja (σ prezime = 'Perić' (racun ⋈ zaposlenik))
 // 2. prikaži ukupan broj zaposlenika (rezultat: broj_zaposlenika)
  γ count(*)-> ukupan_broj (zaposlenik)
 // 3. prikaži artikl sa najvećom cijenom (rezultat: id_artikl, naziv, cijena)
γ max(cijena)-> kurac (artikl) ⋈ (π id_artikl,naziv,cijena->kurac (artikl))
// 4. prikaži sve zaposlenike i broj računa koje su oni izdali (rezultat: id_zaposlenik, ime, prezime,broj_izdanih_racuna)
 γ id_zaposlenik , ime, prezime; count(*)->broj_racuna (racun ⋈ zaposlenik)
 // 5. prikaži sve artikle sa nazivima 'Puding' ili 'Kruh', zajedno sa ukupnom količinom u kojoj su izdani na računima (rezultat: id
γ id_artikl,naziv,cijena ;count(id_artikl)->ukupno (σ naziv = 'Puding' or naziv='Kruh' artikl ⋈ stavka_racun)
```

vjezba za kolokvij

```
//(1) prikaži sve aerodrome koji imaju kapacitet aviona veći ili jednak 10 (rezultat: id_aerodrom,naziv, kapacitet_aviona)
 σ kapacitet_aviona >= 10 (aerodrom)

// (1) prikaži sve aerodrome sa dodatnim stupcem (sa nazivom 'uvecani_kapacitet') koji će prikazati kapacitet aerodroma uvećan za 20
 π id_aerodrom,naziv,kapacitet_aviona,kapacitet_aviona+0.2*kapacitet_aviona->kurac (aerodrom)

// (1) prikaži sve avione koji su u vlasništvu aviokompanije sa nazivom 'Croatia airline' (rezultat:id_avion, reg_oznaka, datum_izra
 π id_avion,reg_oznaka,datum_izrade σ naziv = 'Croatia airline' (aviokompanija ⋈ avion)

// (2) prikaži sve aviokompanije sa dodatnim stupcem koji će prikazati broj aviona u njihovom vlasništvu (rezultat: id_aviokomapnija
  γ id_aviokomapnija,naziv,oib;count(id_avion)->broj_aviona (avion⋈aviokompanija)

// prikaži sve aviokompanije koje nemaju niti jedan avion u svojem vlasništvu (rezultat:id_aviokomapnija, naziv, oib)
  kurac = γ id_aviokomapnija,naziv,oib;count(id_avion)-> vlasnistvo (avion⋈aviokompanija)
  π id_aviokomapnija, naziv,oib σ vlasnistvo=0 (kurac)

// prikaži sve avione koji su odletjeli barem dva leta (rezultat: id_avion, reg_oznaka)
 test = γ id_avion; count(id_let)->letovi  (let ⋈ avion)
 π id_avion, reg_oznaka σ letovi>1 (test⋈avion)

//  prikaži sve avione koji su letjeli iz aerodroma sa nazivom 'Aerodrom Pula' u aerodrom sa nazivom 'Aerodrom Zagreb' (rezultat: id

 π id_avion,reg_oznaka σ (naziv='Aerodrom Pula' and id_aerodrom = id_polaziste) ∨ (naziv='Aerodrom Zagreb' and id_aerodrom = id_odre
 // 8. (3) prikaži aviokompaniju koja ima najmanje aviona-- nije dobro
 -- uvlasništvu(rezultat:id_aviokomapnija,naziv, oib)
-- γ id_aviokomapnija,naziv, oib; min(brojav)→minin (aviokompanija) (γ id_aviokomapnija; count(id_avion)→ brojav (avion))
```