

Relacijska algebra II

BAZE PODATAKA I

doc. dr. sc. Goran Oreški
*Fakultet informatike,
Sveučilište Jurja Dobrile, Pula*

Sadržaj

- ponavljanje prethodnih predavanja:
 - jezici za rad s bazom podataka
 - relacijska algebra
 - osnovne operacije relacijske algebre
 - dodatne operacije relacijske algebre
- proširene operacije relacijske algebre
 - generalizirana projekcija
 - funkcije agregacije
 - dodatne join operacije
- promjene podataka u bazi podataka
 - insert
 - update
 - delete

Ponavljjanje

- jezici za rad s bazom podataka:
 - DML
 - proceduralni
 - deklarativni
 - DDL
 - specifikacija sheme
 - ograničenja na podacima
 - domene, referencijalni integritet, opća ograničenja, dozvole
- relacijska algebra
 - proceduralni upitni jezik
 - osnovne i dodatne operacije

Ponavljjanje

- relacijska algebra
 - šest osnovnih operacija
 - projekcija
 - selekcija
 - unija
 - razlika
 - Kartezijev produkt
 - preimenovanje
- svaka operacija uzima jednu ili dvije relacije
- stvara relaciju kao rezultat

Ponavljjanje

- dodatne operacije relacijske algebre
 - presjek
 - natural join
 - dijeljenje
 - dodjeljivanje
- operacije definirane pomoću osnovnih operacija
- česta primjena u složenim upitima
- omogućavaju pisanje kraćih upita

Proširene operacije

- osnovne operacije relacijske algebre su proširene s ciljem zadavanja upita koji nisu mogući korištenjem osnovnih operacija
- omogućavaju veću iskoristivost relacijske algebre
 - dodaju nove funkcionalnosti
- tri značajna proširenja su:
 - generalizirana projekcija
 - funkcije agregacije
 - dodatne join operacije
- sva proširenja su nalaze u SQL standardima

Operacija generalizirane projekcije

- želimo uključiti podatke koji su rezultat računanja
 - npr. Prikaži sve studente s iznosom neuplaćene školarine.
 - neuplaćena školarina = školarina – uplaćena školarina
- operacija projekcije je generalizirana na način da uključi izračunate rezultate
 - koristeći operaciju mogu se specificirati atributi (projekcija), kao i funkcije na atributima
 - mogu se dodijeliti imena izračunatim vrijednostima
 - preimenovanje atributa je također dozvoljeno

Operacija generalizirane projekcije

- izraz: $\Pi_{F_1, F_2, F_3, \dots, F_n}(E)$
 - F_i je aritmetički izraz
 - E je izraz koji rezultira relacijom
 - može se dodati ime vrijednosti: F_i **as** *name*
- koristi se za stvaranje izvedenih atributa:
 - vrijednosti se računaju na temelju vrijednosti drugih atributa spremljenih u bazi podataka (mogu se koristiti i konstantne vrijednosti)
 - aritmetičke operacije su $\times, \div, +, -$
 - dozvoljava operacije na znakovnim nizovima
 - spajanje nizova

Operacija generalizirane projekcije

- primjer: Prikaži sve studente s iznosom neuplaćene školarine!

$\Pi_{jmbag, (skolarina - uplacena_skolarina) \text{ AS } neuplacena_skolarina}(student)$

jmbag	prezime	smjer	skolarina	uplacena_skolarina
100234	Marić	Biologija	12000	4000
203345	Ivanović	Geofizika	10000	10000
121455	Horvat	Informatika	9000	5000
200032	Jurić	Kardiologija	13000	11000



jmbag	neuplacena_skolarina
100234	8000
203345	0
121455	4000
200032	2000

Funkcije agregacije

- često se javlja potreba za primjenom neke funkcije na skup vrijednosti da bi se generirala jedna vrijednost
- najčešće funkcije agregacije:
 - sum** zbrajaju se vrijednosti
 - avg** računa se prosjek vrijednosti
 - count** prebrojava se broj vrijednosti
 - min** minimalna vrijednost
 - max** maksimalna vrijednost
- funkcije agregacije rade na multiskupovima, ne skupovima
 - vrijednosti se mogu ponavljati

Funkcije agregacije

- Pronađite ukupan iznos školarina na sveučilištu!

$\mathcal{G}_{sum}(skolarina)(student)$

44000

jmbag	prezime	smjer	skolarina
100234	Marić	Biologija	12000
203345	Ivanović	Geofizika	10000
121455	Horvat	Informatika	9000
200032	Jurić	Kardiologija	13000

- Pronađite najveći dug prema sveučilištu po školarini! (slajd 9.)

$\mathcal{G}_{max}(neuplacena_skolarina)(\Pi_{(skolarina - uplacena_skolarina)} AS neuplacena_skolarina)(student)$

8000

Funkcije agregacije

- ponekad je potrebno računati agregacije prema specifičnoj vrijednosti nekog atributa
- vratimo se na bazu podataka položenih ispita
 - shema?
- pitanja
 - koliko je pojedini student položio ispita?
 - koliko studenata je položilo pojedini ispit?

prezime	kolegij_naziv
Horvat	Programiranje 1
Horvat	Baze podatka 1
Božić	Sustavi poslovne inteligencije
Blažević	Programiranje 1
Božić	Baze podataka 1
Novak	Baze podataka 1
Horvat	Sustavi poslovne inteligencije

položeno

kolegij_naziv
Programiranje 1
Baze podatka 1
Sustavi poslovne inteligencije

kolegij

Funkcije agregacije

- koliko je pojedini student položio ispita?

prezime **G**count(*kolegij_naziv*)(*polozeno*)

- prvo se relacija *polozeno* grupira po jedinstvenim vrijednostima atributa *prezime*
- potom se count(*naziv_kolegija*) primjenjuje na svaku grupu pojedinačno

prezime	kolegij_naziv
Horvat	Programiranje 1
Horvat	Baze podatka 1
Božić	Sustavi poslovne inteligencije
Blažević	Programiranje 1
Božić	Baze podataka 1
Novak	Baze podataka 1
Horvat	Sustavi poslovne inteligencije

polozeno

kolegij_naziv
Programiranje 1
Baze podatka 1
Sustavi poslovne inteligencije

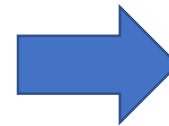
kolegij

Funkcije agregacije

prezime **G**count(*kolegij_naziv*)(*polozeno*)

Ulazna relacija je grupirana po atributu *prezime*.

<i>prezime</i>	<i>kolegij_naziv</i>
Horvat	Programiranje 1
Horvat	Baze podatka 1
Horvat	Sustavi poslovne inteligencije
Božić	Sustavi poslovne inteligencije
Božić	Baze podataka 1
Novak	Baze podataka 1
Blažević	Programiranje 1



Primjenjuje se *count* funkcija na sve grupe pojedinačno.

<i>prezime</i>	
Horvat	3
Božić	2
Novak	1
Blažević	1

Različite vrijednosti

- ponekad je potrebno izračunati agregacije na skupovima a ne multiskupovima
- izmjenimo db položenih ispita
 - dodajmo relaciju koja sadrži podatke o pristupanju ispitu i broju bodova
- na koliko kolegija je izašao pojedini student?
 - koristeći relaciju *pristupio*
 - student može izaći više puta na ispit istog kolegija

prezime	kolegij_naziv	bodovi
Horvat	Programiranje 1	12
Horvat	Baze podataka 1	54
Božić	Sustavi poslovne inteligencije	23
Blažević	Programiranje 1	76
Božić	Sustavi poslovne inteligencije	32
Novak	Baze podataka 1	100
Horvat	Programiranje 1	7

pristupio

Različite vrijednosti

- na koliko kolegija je izašao pojedini student?
- broj izlazaka iz pojedinog kolegija se može pojaviti više puta!

prezime	kolegij_naziv	bodovi
Horvat	Programiranje 1	12
Horvat	Baze podataka 1	54
Božić	Sustavi poslovne inteligencije	23
Blažević	Programiranje 1	76
Božić	Sustavi poslovne inteligencije	32
Novak	Baze podataka 1	100
Horvat	Programiranje 1	7

pristupio

- potrebno je prebrojati različita (*engl. distinct*) pojavljivanja svih kolegija

prezime **G**count-distinct(*kolegij_naziv*)(*pristupio*)

Eliminacija duplikata

- svakoj funkciji agregiranja se može dodati **–distinct** da bi se specificirala eliminacija duplikata
 - najčešće se koristi s funkcijom ***count***; ***count-distinct***
 - nema smisla korištenje s ***min*** i ***max***

Opći oblik funkcija agregacije

- izraz: $G_1, G_2, G_3, \dots, G_n \mathcal{G}_{F_1(A_1), F_2(A_2), F_3(A_3), \dots, F_m(A_m)}(E)$
 - E je izraz koji rezultira relacijom
 - vodeći G_i su atributi iz E na kojima se vrši grupiranje
 - svaka F_i je funkcija agregacije primijenjena na atributu A_i iz E
- prvi korak: ulazna relacija (podaci) se dijeli u grupe
 - ukoliko nije naveden niti jedan G_i tada je sve jedna grupa, tj. ne provodi se grupiranje
- drugi korak: izvodi se funkcija agregacije na svim grupama pojedinačno

Opći oblik funkcija agregacije

- izraz: $G_1, G_2, G_3, \dots, G_n \mathcal{G}_{F_1(A_1), F_2(A_2), F_3(A_3), \dots, F_m(A_m)}(E)$
- n-torke u E su grupirane na način:
 - sve n-torke u istoj grupi imaju identičnu vrijednost za attribute $G_1, G_2, G_3, \dots, G_n$
 - n-torke u različitim grupama imaju različite vrijednosti za attribute $G_1, G_2, G_3, \dots, G_n$
- stoga vrijednosti $\{g_1, g_2, g_3, \dots, g_n\}$ u svakoj grupi jedinstveno identificiraju grupu
 - $\{G_1, G_2, G_3, \dots, G_n\}$ je super-ključ za rezultirajuću relaciju

Opći oblik funkcija agregacije

- izraz: $G_1, G_2, G_3, \dots, G_n \mathcal{G}_{F_1(A_1), F_2(A_2), F_3(A_3), \dots, F_m(A_m)}(E)$
- n-torke iz rezultata su oblika:
 $\{g_1, g_2, g_3, \dots, g_n, a_1, a_2, a_3, \dots, a_m\},$
 - g_i su vrijednosti za određenu grupu
 - a_i je rezultat primjene F_i na multiskup vrijednosti A_i iz te grupe
- $F_i(A_i)$ atributi su bez imena
 - ime se može specificirati kao i prije $F_i(A_i)$ **AS** *attr_name*

Primjer funkcije agregacije

- koliko studenata je položilo pojedini ispit?

kolegij_naziv **Gcount**(*prezime*)(*polozeno*)

- slučaj kada niti jedan student nije položio kolegij?
 - ako nitko nije položio neće se nalaziti u relaciji *polozeno*

prezime	kolegij_naziv
Horvat	Programiranje 1
Horvat	Baze podatka 1
Božić	Sustavi poslovne inteligencije
Blažević	Programiranje 1
Božić	Baze podataka 1
Novak	Baze podataka 1
Horvat	Sustavi poslovne inteligencije

polozeno

kolegij_naziv
Programiranje 1
Baze podatka 1
Sustavi poslovne inteligencije

kolegij

Primjer funkcije agregacije

- novi kolegij je dodan u relaciju
 - htjeli bi vidjeti {„Napredne tehnike programiranja, 0”} u rezultatima
 - „NTP” se neće pojaviti u rezultatima!
 - zašto?
- hoće li join pomoći?
 - ne
 - *natural join* neće proizvesti niti jedan red za „NTP”

prezime	kolegij_naziv
Horvat	Programiranje 1
Horvat	Baze podatka 1
Božić	Sustavi poslovne inteligencije
Blažević	Programiranje 1
Božić	Baze podataka 1
Novak	Baze podataka 1
Horvat	Sustavi poslovne inteligencije

položeno

kolegij_naziv
Programiranje 1
Baze podatka 1
Sustavi poslovne inteligencije
Napredne tehnike programiranja

kolegij

theta join

- theta join je varijanta natural join operacije koja omogućava da kombiniramo selekciju i Kartezijev produkt u jednoj operaciji
- uzmimo relacije $r(R)$ i $s(S)$ i recimo da je θ predikat na atributima sheme $R \cup S$
 - tada operaciju theta join $r \bowtie_{\theta} s$ definiramo kao
$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$
- upotreba:
 - natural join kada se atributi ne zovu jednako (*equi-join*)
 - skraćeni zapis selekcije i Kartezijevog produkta

Outer joins

- *hrv. vanjsko spajanje*
- *natural join* zahtjeva da lijeva i desna relacija sadrže istu n-torku za spajanje

$$r \bowtie s = \Pi_{R \cup S} (\sigma_{r.A_1=s.A_1 \wedge r.A_2=s.A_2 \wedge r.A_3=s.A_3 \wedge \dots \wedge r.A_n=s.A_n} (r \times s))$$

- *outer join* je dodatak join operaciji stvoren s ciljem upravljanja nedostajućim vrijednostima
- nedostajuće vrijednosti u rezultatu se predstavljaju pomoću ***null*** vrijednosti
 - *null* – nepostojeća ili nepoznata vrijednost

Outer joins

- left outer join: $r \bowtie_{\text{left}} s$
 - ako n-torka $t_r \in r$ nema pripadajuću n-torku u s , rezultat sadrži $\{t_r, \text{null}, \text{null}, \dots, \text{null}\}$
 - ako n-torka $t_s \in s$ nema pripadajuću n-torku u r , onda je isključena iz rezultata upita
- right outer join: $r \bowtie_{\text{right}} s$
 - ako n-torka $t_r \in r$ nema pripadajuću n-torku u s , onda je isključena iz rezultata upita
 - ako n-torka $t_s \in s$ nema pripadajuću n-torku u r , rezultat sadrži $\{\text{null}, \text{null}, \dots, \text{null}, t_s\}$

Outer joins

- full outer join: $r \bowtie s$
 - uključuje n-torke iz r koje nemaju pripadajuću n-torku u s , i obrnuto one u s koje nemaju u r

- primjer:

$r =$

att1	att2
1	10
2	11
3	12

$s =$

att1	att3
2	100
3	101
4	102

$r \bowtie s$

att1	att2	att3
2	11	100
3	12	101

$r \bowtie s$

att1	att2	att3
1	10	<i>null</i>
2	11	100
3	12	101

$r \bowtie s$

att1	att2	att3
2	11	100
3	12	101
4	<i>null</i>	102

$r \bowtie s$

att1	att2	att3
1	10	<i>null</i>
2	11	100
3	12	101
4	<i>null</i>	102

null vrijednost

- uvođenje *null* vrijednosti ima veliki utjecaj na sve dosad obrađene dijelove relacijske algebre
 - *null* znači nepoznata ili nepostojeća vrijednost
- potrebno je definirati učinke na rezultat kada je *null* vrijednost prisutna
- aritmetičke operacije koje uključuju *null* uvijek daju rezultat *null*
 - npr. $5 + \text{null} = \text{null}$
- operatori usporedbe nad *null* daju rezultat nepoznato (*engl. unknown*)
 - čak $\text{null} = \text{null}$ daje vrijednost unknown

Logički operatori i unknown

- and

$\text{true} \wedge \text{unknown} = \text{unknown}$

$\text{false} \wedge \text{unknown} = \text{false}$

$\text{unknown} \wedge \text{unknown} = \text{unknown}$

- or

$\text{true} \vee \text{unknown} = \text{true}$

$\text{false} \vee \text{unknown} = \text{unknown}$

$\text{unknown} \vee \text{unknown} = \text{unknown}$

- not

$\neg \text{unknown} = \text{unknown}$

Relacijske operacije

- za svaku relacijsku operaciju potrebno je specificirati ponašanje u odnosu na *null* i *unknown* vrijednosti
- selekcija: $\sigma_P(r)$
 - ako P rezultira s *null* za neku n-torku, tada se ista n-torka isključuje iz rezultata (tj. definicija selekcije se ne mijenja)
- natural join: $r \bowtie s$
 - Kartezijev produkt potom selekcija
 - ukoliko zajednički atribut ima vrijednost *null*, tada se n-torka isključuje iz join rezultata
 - *null* je nepoznata vrijednost

Projekcija i operacije skupova

- projekcija: $\Pi(E)$
 - operacija projekcije mora ukloniti duplikate
 - *null* vrijednost se tretira kao bilo koja druga vrijednost
 - višestruke n-torke koje sadrže *null* vrijednost se također eliminiraju
- unija, presjek i razlika
 - *null* vrijednost se tretira kao bilo koja druga vrijednost
 - unija, presjek i razlika računaju se kao obično
- definirana pravila pomalo arbitrarna!?
 - u ovim slučajevima, dvije *null* vrijednosti se smatraju istima
 - ali dvije *null* vrijednosti nisu iste

Grupiranje i agregacije

- u fazi grupiranja:
 - *null* vrijednost se tretira kao bilo koja druga vrijednost
 - ako n-torke imaju istu vrijednost svih atributa za grupiranje (uključujući i vrijednost *null*), sve završe u istoj grupi
- u fazi agregacije
 - *null* vrijednosti se uklanjaju iz multiskupova prije nego što se primjeni funkcija agregacije
 - ako funkcija agregacije na ulazu dobije prazan multiskup, rezultat je *null*...
 - osim funkcije *count* koja vraća 0

Generalizirana projekcija, outer join

- operacija generalizirane projekcije
 - kombinacija projekcije i aritmetičke operacije
 - izvedite pravilo za zadaću koristeći prijašnja pravila
- outer joins:
 - ponaša se kao i natural join, razlika je u tome što se dodaje *null* umjesto nedostajućih vrijednosti

Povratak na položene ispite

- koliko studenata je položilo pojedini ispit?

kolegij_naziv
Programiranje 1
Baze podataka 1
Sustavi poslovne inteligencije
Napredne tehnike programiranja

kolegij

- koristi se outer join da bi se uključili svi kolegiji, ne samo položeni

kolegij ⋈ *polozeno*

prezime	kolegij_naziv
Horvat	Programiranje 1
Horvat	Baze podataka 1
Božić	Sustavi poslovne inteligencije
Blažević	Programiranje 1
Božić	Baze podataka 1
Novak	Baze podataka 1
Horvat	Sustavi poslovne inteligencije

polozeno

kolegij_naziv	prezime
Programiranje 1	Horvat
Baze podataka 1	Horvat
Sustavi poslovne inteligencije	Božić
Programiranje 1	Blažević
Baze podataka 1	Božić
Baze podataka 1	Novak
Sustavi poslovne inteligencije	Horvat
Napredne tehnike programiranja	null

Zbrajanje položenih ispita

- korišćenje grupiranja i agregacije
 - grupiranje po kolegijima
 - zbrajanje studenata

kolegij_naziv **G**_{count}(*prezime*)(*kolegij* ⋈ *polozeno*)

kolegij_naziv	prezime
Programiranje 1	Horvat
Baze podatka 1	Horvat
Sustavi poslovne inteligencije	Božić
Programiranje 1	Blažević
Baze podataka 1	Božić
Baze podataka 1	Novak
Sustavi poslovne inteligencije	Horvat
Napredne tehnike programiranja	<i>null</i>



kolegij_naziv	prezime
Baze podatka 1	Horvat
Baze podatka 1	Novak
Baze podatka 1	Božić
Programiranje 1	Horvat
Programiranje 1	Blažević
Sustavi poslovne inteligencije	Horvat
Sustavi poslovne inteligencije	Božić
Napredne tehnike programiranja	<i>null</i>



kolegij_naziv	
Programiranje 1	3
Baze podatka 1	2
Sustavi poslovne inteligencije	2
Napredne tehnike programiranja	0

Promjene podataka u bazi podataka

- podaci se često mijenjaju u bazi podataka
- za promjenu podataka se koristi operator dodjeljivanja \leftarrow
- operacije:
 - $r \leftarrow r \cup E$ umetanje novih n-torki u relaciju
 - $r \leftarrow r - E$ brisanje n-torki iz relacije
 - $r \leftarrow \Pi(r)$ promjena n-torki koje se nalaze u relaciji
- na prethodnim predavanjima smo r definirali kao relacijsku varijablu, to znači:
 - operator dodjeljivanja postavlja novu relacijsku vrijednost u r
 - izraz često treba uključiti i postojeće n-torke, da se podaci ne bi izgubili

Dodavanje novih n-torki

- za umetanje novih n-torki koristi se unija

$$r \leftarrow r \cup E$$

- E mora biti ispravnog stupnja

- nove n-torke za umetanje se mogu i direktno specificirati

$$polozeno \leftarrow polozeno \cup$$

$\{ („Ivić”, „Programiranje I”), („Marić”, „Baze podataka I”) \}$

- dodaju se dvije n-torke *polozeno* relaciji

- specificirati se mogu konstante relacije kao skup vrijednosti

- svaka n-torka je zatvorena zagradama
- ukupan skup n-torki je zatvoren vitičastom zagradom



konstantna relacija

Dodavanje novih n-torki

- n-torke se mogu dodati i iz izraza
- primjer:
 - „Student Perić je došao s drugog fakulteta informatike i priznata su mu oba ispita koja drži profesor Marić a koja je već položio na starom fakultetu.”
 - potrebno je pronaći ispite koje drži profesor Marić te dodati dva zapisa kojima se označava da je student položio ispite
 - proširena baza podataka:
 - profesor(sifra profesora, prezime, fakultet, zvanje)*
 - student(JMBAG, prezime)*
 - kolegij(naziv kolegija, sifra profesora)*
 - polozeno(JMBAG, naziv kolegija, ocjena)*

Dodavanje novih n-torki

- potrebno je pronaći koje kolegije predaje profesor Marić te iste za studenta Perić dodati u relaciju *polozeno* (recimo i da prezime jedinstveno identifikira profesora i studenta, inače moramo raditi selekciju po šifri profesora)

$\sigma_{\text{prezime} = \text{"Marić"}}(\text{profesor})$

$(\sigma_{\text{prezime} = \text{"Marić"}}(\text{profesor})) \bowtie \text{kolegij}$

$\Pi_{\text{naziv_kolegija}}((\sigma_{\text{prezime} = \text{"Marić"}}(\text{profesor})) \bowtie \text{kolegij})$

$\{(\text{"Perić"})\} \times (\Pi_{\text{naziv_kolegija}}((\sigma_{\text{prezime} = \text{"Marić"}}(\text{profesor})) \bowtie \text{kolegij}))$

$\text{polozeno} \leftarrow \text{polozeno} \cup \{(\text{"Perić"})\} \times (\Pi_{\text{naziv_kolegija}}((\sigma_{\text{prezime} = \text{"Marić"}}(\text{profesor})) \bowtie \text{kolegij}))$

- oprez shema!

$\text{polozeno} \leftarrow \text{polozeno} \cup ((\Pi_{\text{jmbag}}(\sigma_{\text{prezime} = \text{"Marić"}}(\text{student}))) \times$

$(\Pi_{\text{naziv_kolegija}}(\sigma_{\text{naziv_kolegija} = \text{"Programiranje"}}((\sigma_{\text{prezime} = \text{"Marić"}}(\text{profesor})) \bowtie \text{kolegij}))) \times \{(3)\})$

...

Brisanje n-torki

- za brisanje n-torki koristi se operacija –
 - $r \leftarrow r - E$
- *primjer:*
 - Potrebno je obrisati studenta Božić iz baze podataka sveučilišta, jer je ispisan iz fakulteta!
 - napomena: vrlo rijetko je potrebno brisati podatke iz baze podataka, gotovo uvijek moraju ostati radi praćenja povijesti
- problem
 - *polozeno* relacija referencira relaciju *student*
 - zbog referencijalnog integriteta potrebno je obrisati zapise iz *polozeno*

Brisanje n-torki

- *polozeno* referencira *student*
 - *JMBAG* atribut je ključ
 - *polozeno* relacija ne smije imati vrijednosti iz *student* koje ne postoje u relaciji *student*
 - prvo se briše n-torke iz *polozeno*
 - potom se brišu iz *student*
- rješenje

$$polozeno \leftarrow polozeno - \Pi_{JMBAG, naziv_kolegija, ocjena} (\sigma_{prezime = "Božić"}(student) \bowtie polozeno)$$

$$student \leftarrow student - \sigma_{prezime = "Božić"}(student)$$

Brisanje n-torki

- u relacijskoj algebri sami moramo paziti na ograničenja stranih ključeva
- relacijski sustav baze podataka na ta ograničenja pazi automatski, što ćemo vidjeti u nastavku

Promjena n-torki

- opći oblik koristi generaliziranu projekciju

$$r \leftarrow \Pi_{F_1, F_2, \dots, F_n}(r)$$

- mijenja sve n-torke u r

- Primjer:

- povećaj za jedan sve ECTS bodove kolegija

$$kolegij \leftarrow \Pi_{kolegij_naziv, ECTS + 1}(kolegij)$$

- moraju se uključiti i atributi koji se ne mijenjaju
 - zašto?
 - u suprotnom slučaju se mijenja shema *kolegija*

kolegij_naziv	ECTS
Programiranje 1	6
Baze podataka 1	7
Sustavi poslovne inteligencije	5
Napredne tehnike programiranja	5

kolegij

Promjena n-torki

- promjena samo nekih n-torki je nešto složenija
 - relacijska varijabla sadrži vrijednost koju vrati korišteni izraz
 - moraju se uključiti promijenjene n-torke i one koje nisu promijenjene
- primjer:
 - dodajte jedan ECTS svim kolegijima koji imaju manje od 7 ECTS-a

$$kolegij \leftarrow \Pi_{kolegij_naziv, ECTS + 1}(\sigma_{ECTS < 7}(kolegij)) \cup \sigma_{ECTS \geq 7}(kolegij)$$

- potrebno je stvoriti uniju promijenjenih i nepromijenjenih n-torki unutar relacije

Zadatak

- definirana je shema sveučilišne baze podataka (*shema preuzeta iz [DSC]*):

course(course_id, title, dept name, credits)

instructor(ID, name, dept name, salary)

section(course_id, sec_id, semester, year, building, room_number, time_slot_id)

teaches(ID, course_id, sec_id, semester, year)

student(ID, name, dept_name, tot_cred)

takes(ID, course_id, sec_id, semester, year, grade)

Zadatak

- Definirana je shema sveučilišne baze podataka (*shema preuzeta iz [DSC]*):

course(*course_id*, *title*, *dept name*, *credits*)

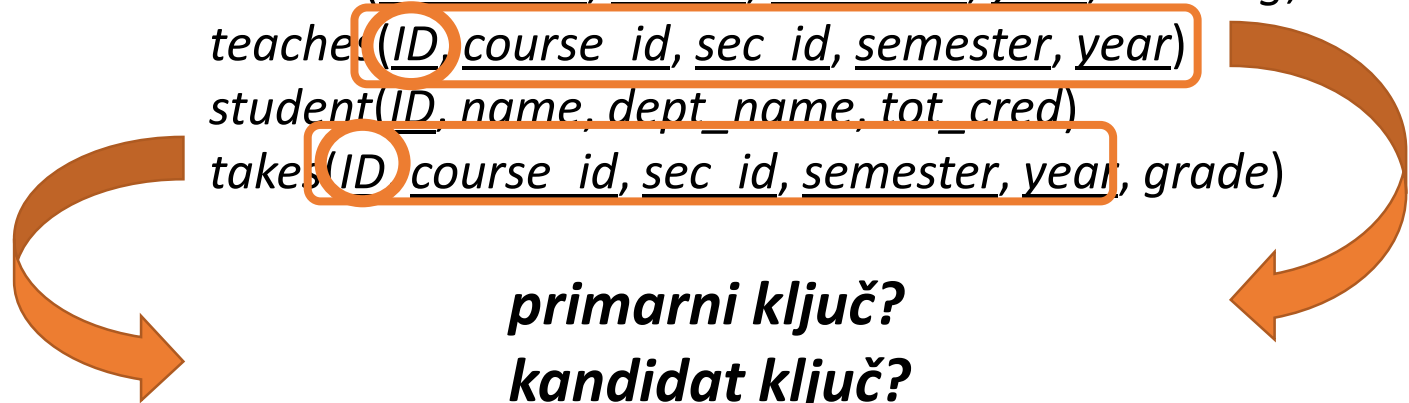
instructor(*ID*, *name*, *dept name*, *salary*)

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

teacher(*ID*, *course_id*, *sec_id*, *semester*, *year*)

student(*ID*, *name*, *dept_name*, *tot_cred*)

takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)



primarni ključ?
kandidat ključ?
super ključ?

Zadatak

- pronadite sve studente koji su imali predavanja kod profesora Radmana; u rezultatu ne smije biti duplikata

$\Pi_{ID}(\sigma_{IID='Radman'}(\text{takes} \bowtie \rho_{t1(IID, \text{course id, section id, semester, year})} \text{teaches}))$

- duplikati?
 - ako se koristi relacijska algebra koja koristi skupove tada je navedeni izraz dovoljan
 - ako se koristi relacijska algebra koja koristi multi-skupove tada se može koristiti operator grupiranja
 - za vježbu napišite izraz koji podrazumijeva korištenje multi-skupova

Zadatak

- pronađite najvišu plaću profesora (*instructor*)

$\mathcal{G}_{\max(\text{salary})}(\text{instructor})$

- pronađite sve profesore s najvišom plaćom (može biti više profesora s najvišom plaćom)

$\text{instructor} \bowtie (\mathcal{G}_{\max(\text{salary})} \text{ as salary}(\text{instructor}))$

- pronađite broj upisanih studenata na svim ponuđenim kolegijima i grupama (*section*) za zimski semestar 2019

$\text{course_id, section_id } \mathcal{G}_{\text{count}(*)} \text{ as enrollment } (\sigma_{\text{year}=2019 \wedge \text{semester}=\text{Winter}}(\text{takes}))$

Zadatak

- pronađite najveći broj upisanih studenata na svim ponuđenim kolegijima i grupama (*section*) za zimski semestar 2019

$t1 \leftarrow \text{course_id, section_id } \mathcal{G}_{\text{count}(*)} \text{ as enrollment } (\sigma_{\text{year}=2019 \wedge \text{semester}=\text{Winter}}(\text{takes}))$
 $\mathcal{G}_{\text{max}(\text{enrollment})}(t1)$

- pronađite ponuđene kolegije i grupe (*sections*) koje su imale najveći broj upisanih studenata

$t2 \leftarrow \mathcal{G}_{\text{max}(\text{enrollment})} \text{ as enrollment } (t1)$
 $t1 \bowtie t2$

Zadatak

- natural outer-join operacije proširuju natural join tako da se n-torke iz pripadajućih relacija ne izgube u rezultatu join-a ukoliko nisu uparene
- opišite kako proširiti operacije theta-joina tako da se relacije iz:
 - lijeve relacije
 - desne relacije
 - obje relacijene gube u rezultatima theta-joina
- postupak prikažite pomoću izraza relacijske algebre

Zadatak

- left outer theta join na relacijama $r(R)$ i $s(S)$ ili $r \bowtie_{\theta} s$
- se može definirati:

$$(r \bowtie_{\theta} s) \cup ((r - \Pi_R(r \bowtie_{\theta} s)) \times (\text{null}, \text{null}, \dots, \text{null}))$$

- podijelimo izraz:
 - $(r \bowtie_{\theta} s)$
 - $(r - \Pi_R(r \bowtie_{\theta} s))$
 - $(\text{null}, \text{null}, \dots, \text{null})$
 - veličina n-torke koja sadrži samo *null* je jednaka broju atributa iz S

Zadatak

- right outer theta join na relacijama $r(R)$ i $s(S)$ ili $r \bowtie_{\theta} s$
- se može definirati:

$$(r \bowtie_{\theta} s) \cup ((\text{null}, \text{null}, \dots, \text{null}) \times (s - \Pi_S(r \bowtie_{\theta} s)))$$

- podijelimo izraz:
 - $(r \bowtie_{\theta} s)$
 - $(\text{null}, \text{null}, \dots, \text{null})$
 - veličina n -torke koja sadrži samo *null* je jednaka broju atributa iz R
 - $(s - \Pi_S(r \bowtie_{\theta} s))$

Zadatak

- full outer theta join na relacijama $r(R)$ i $s(S)$ ili $r \bowtie_{\theta} s$
- se može definirati:

$$(r \bowtie_{\theta} s) \cup ((\text{null}, \text{null}, \dots, \text{null}) \times (s - \Pi_S(r \bowtie_{\theta} s))) \cup ((r - \Pi_R(r \bowtie_{\theta} s)) \times (\text{null}, \text{null}, \dots, \text{null}))$$

- podijelimo izraz:
 - $(r \bowtie_{\theta} s)$
 - $((\text{null}, \text{null}, \dots, \text{null}) \times (s - \Pi_S(r \bowtie_{\theta} s)))$
 - $((r - \Pi_R(r \bowtie_{\theta} s)) \times (\text{null}, \text{null}, \dots, \text{null}))$

Zadatak

- zadana je slijedeća shema baze podataka (*shema preuzeta iz [DSC]*)

employee (*person name*, *street*, *city*)

works (*person name*, *company_name*, *salary*)

company (*company name*, *city*)

manages (*person name*, *manager_name*)

- zadaci:
 - pronađite imena svih zaposlenika koji žive u istoj ulici i gradu kao njihovi menadžeri
 - pronađite imena svih zaposlenika koji ne rade u Karlovačkoj banci
 - pronađite imena svih zaposlenika koji zarađuju više od svih zaposlenika u Zagrebačkoj banci

Zadatak

- pronađite imena svih zaposlenika koji žive u istoj ulici i gradu kao njihovi menadžeri

$\Pi_{\text{person_name}}((\text{employee} \bowtie \text{manages}) \bowtie_{(\text{manager_name} = \text{employee2.person_name} \wedge \text{employee.street} = \text{employee2.street} \wedge \text{employee.city} = \text{employee2.city})}(\rho_{\text{employee2}}(\text{employee})))$

- pronađite imena svih zaposlenika koji ne rade u Karlovačkoj banci
 - ako podrazumijevamo da svi zaposlenici rade u točno jednoj tvrtki

$\Pi_{\text{person_name}}(\sigma_{\text{company_name} \neq \text{"Karlovačka banka"}}(\text{works}))$

- ako podrazumijevamo da ljudi ne moraju biti zaposleni

$\Pi_{\text{person_name}}(\text{employee}) - \Pi_{\text{person_name}}(\sigma_{\text{company_name} = \text{"Karlovačka banka"}}(\text{works}))$

Zadatak

- pronađite imena svih zaposlenika koji zarađuju više od svih zaposlenika u Zagrebačkoj banci

$$\Pi_{\text{person_name}}(\text{works}) - (\Pi_{\text{works.person_name}}(\text{works} \bowtie_{(\text{works.salary} \leq \text{works2.salary} \wedge \text{works2.company_name} = \text{"Zagrebačka banka"})} \rho_{\text{works2}}(\text{works})))$$

Relacijska algebra sažetak

- vrlo izražajan upitni jezik za dohvat podataka iz relacijske baze podataka
 - jednostavna selekcija, projekcija
 - računanje korelacije između relacija koristeći join operacije
 - operacije grupiranja i agregacije
- jezik omogućava i promjenu vrijednosti podataka spremljenih u bazu podataka
 - insert, update, delete
- relacijska algebra je proceduralni upitni jezik
 - mora se navesti sekvenca operacija da bi se dobio rezultat

Relacijska algebra sažetak

- korist relacijske algebre jest što može biti formalno precizirana i obrazložena
- izrazi znaju biti vrlo dugački
- sustavi baze podataka pružaju jednostavnije upitne jezike za manipulaciju podataka
 - najpopularniji je SQL
- mnoge baze podataka koriste jezik nalik na relacijsku algebru interno, skriveno
 - odlični za reprezentaciju upitnih planova, zbog proceduralnog oblika

Nastavak

- prelazak s relacijske algebra na SQL
- počinjemo raditi s pravim bazama podataka

Literatura

- Pročitati
 - [DSC] poglavlje 6.1. (ostatak)
 - Caltech CS121
- Slijedeće predavanje
 - [DSC] poglavlje 3.1 – 3.5.
 - Caltech CS121 – P4