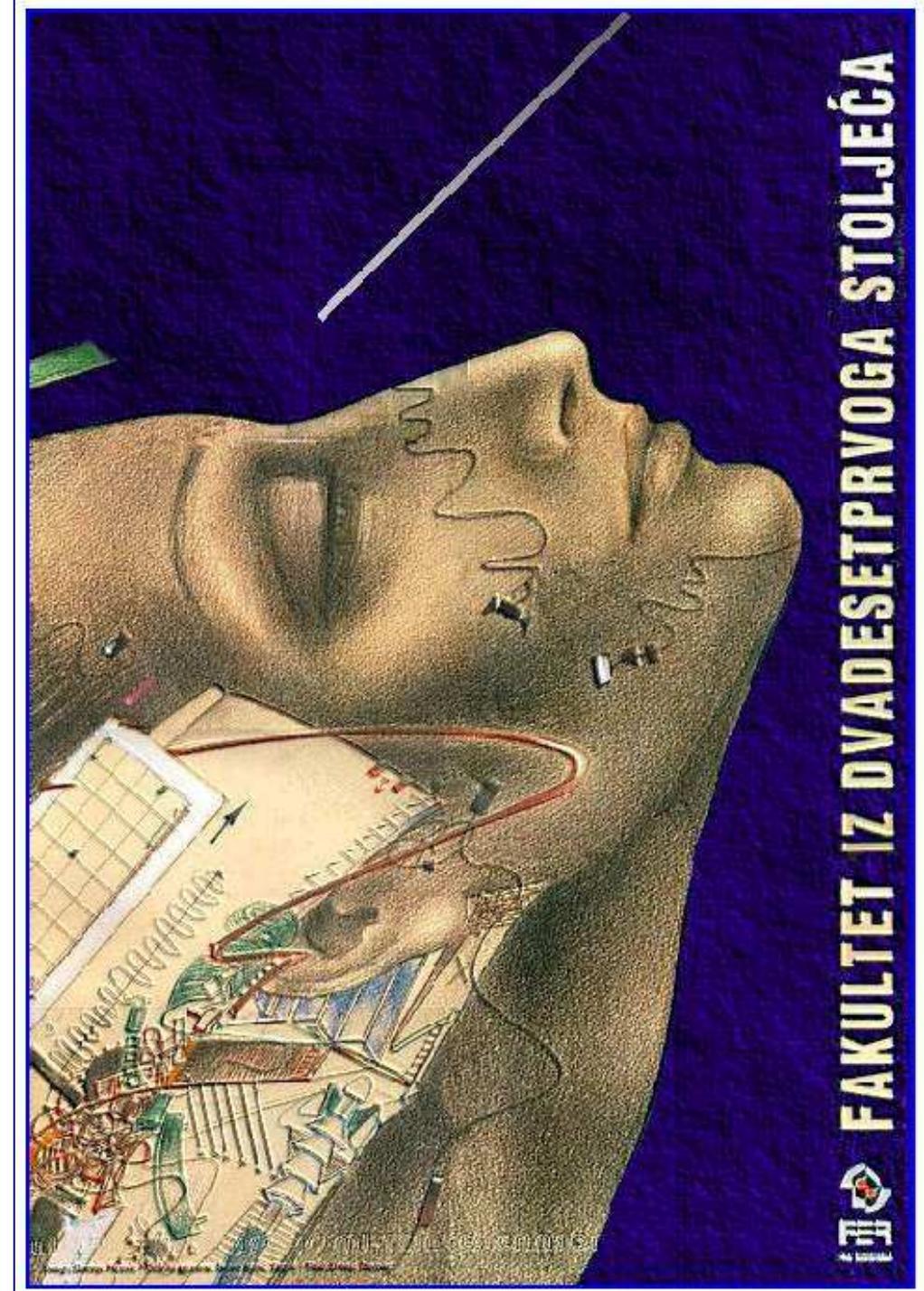


Baze podataka

Predavanja

1. Uvod

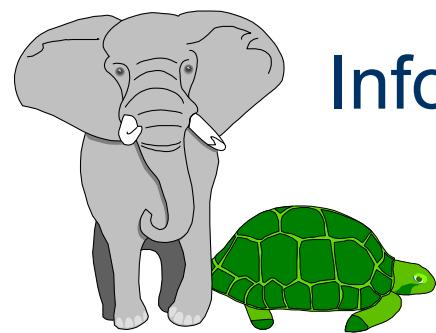
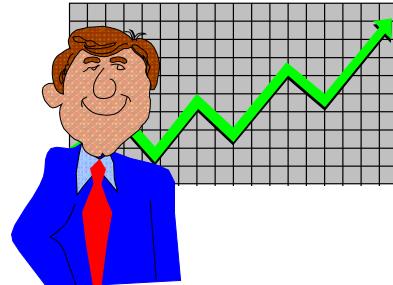
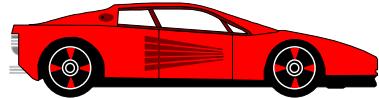
ožujak 2014.



"Način na koji prikupljate informacije,
upravljate njima i koristite ih, odredit će hoćete
li pobijediti ili izgubiti."

Bill Gates

Stvarni svijet



Informacije



Obrada podataka – sveučilište, knjižnica

- Sveučilište
 - podaci o studentima
 - podaci o nastavnicima
 - uspjeh na ispitu
 - ISVU - Informacijski sustav visokih učilišta RH
- Knjižnica
 - podaci o korisniku knjižnice
 - podaci o knjigama
 - traženje knjiga i časopisa
 - posuđivanje knjiga



Obrada podataka - bankarstvo



- otvaranje računa
- novčane transakcije
- praćenje stanja na računu
- praćenje kupnji ostvarenih putem kreditnih kartica

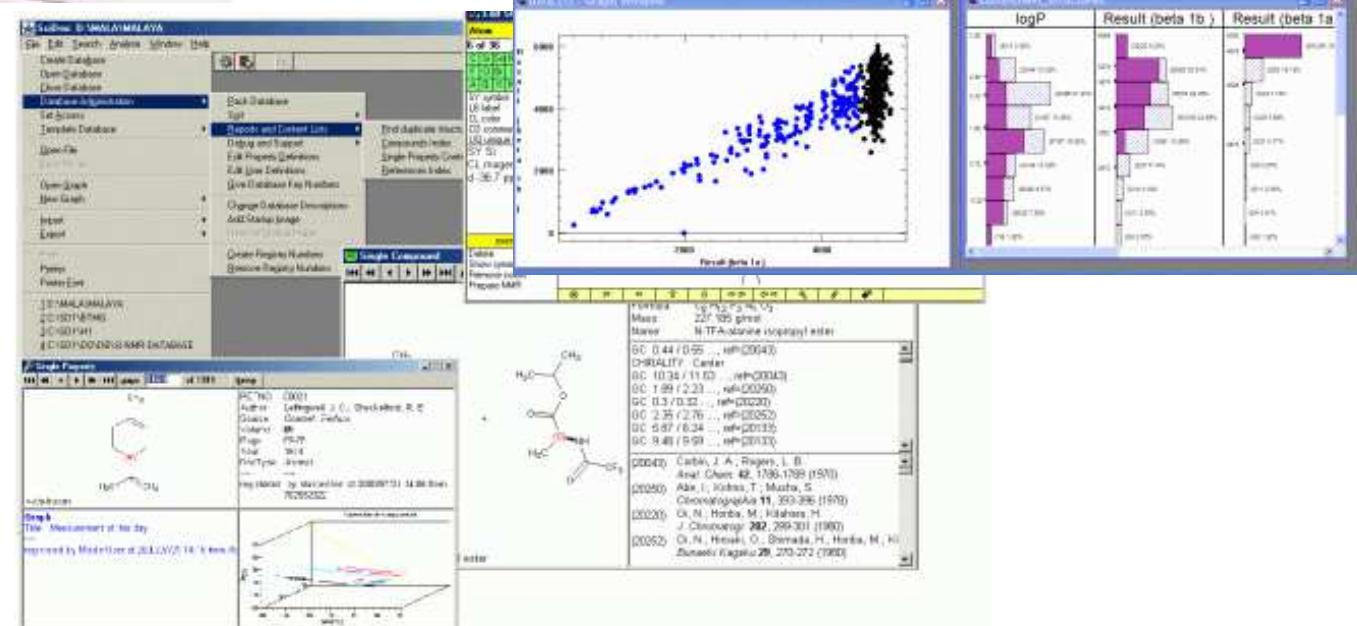
Obrada podataka - telekomunikacije



- podaci o pozivima
- telefonski računi
- podaci o mreži
- podaci o kvarovima



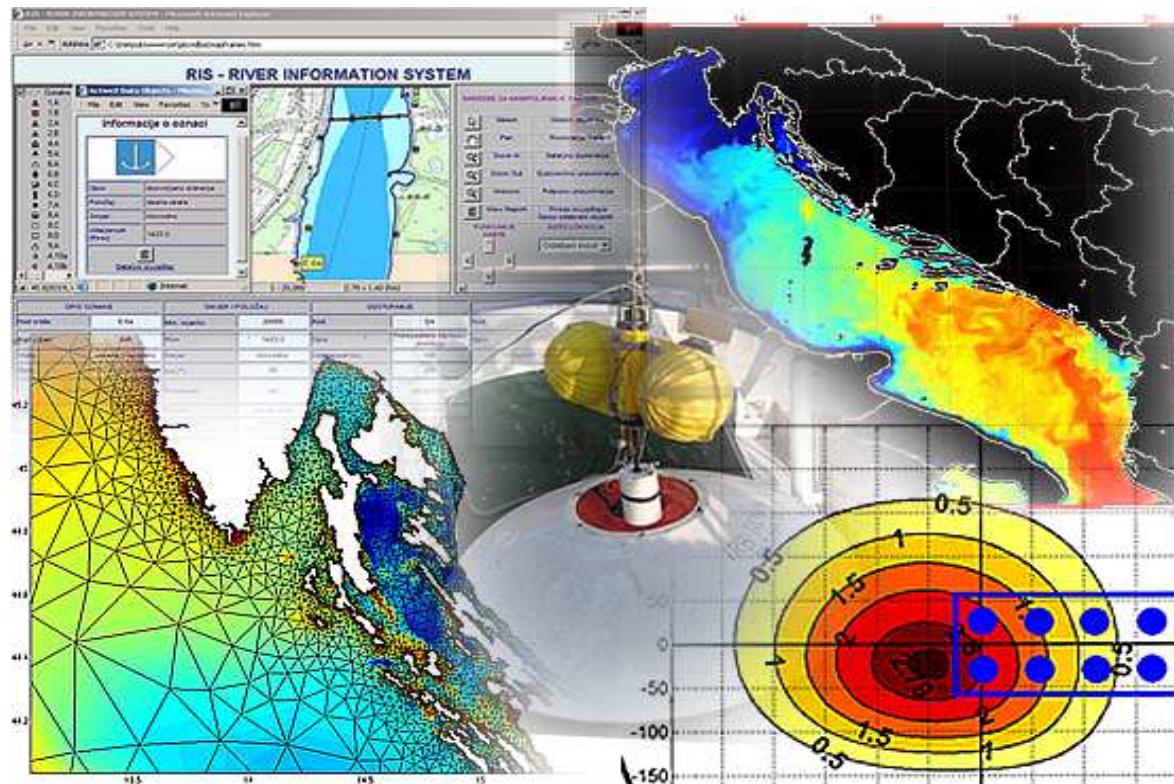
Obrada podataka - znanost



- podaci prikupljeni tijekom istraživanja u fizici, biologiji, kemiji...

Obrada podataka - istraživanje i zaštita okoliša

- informatika o okolišu
- satelitska oceanografija

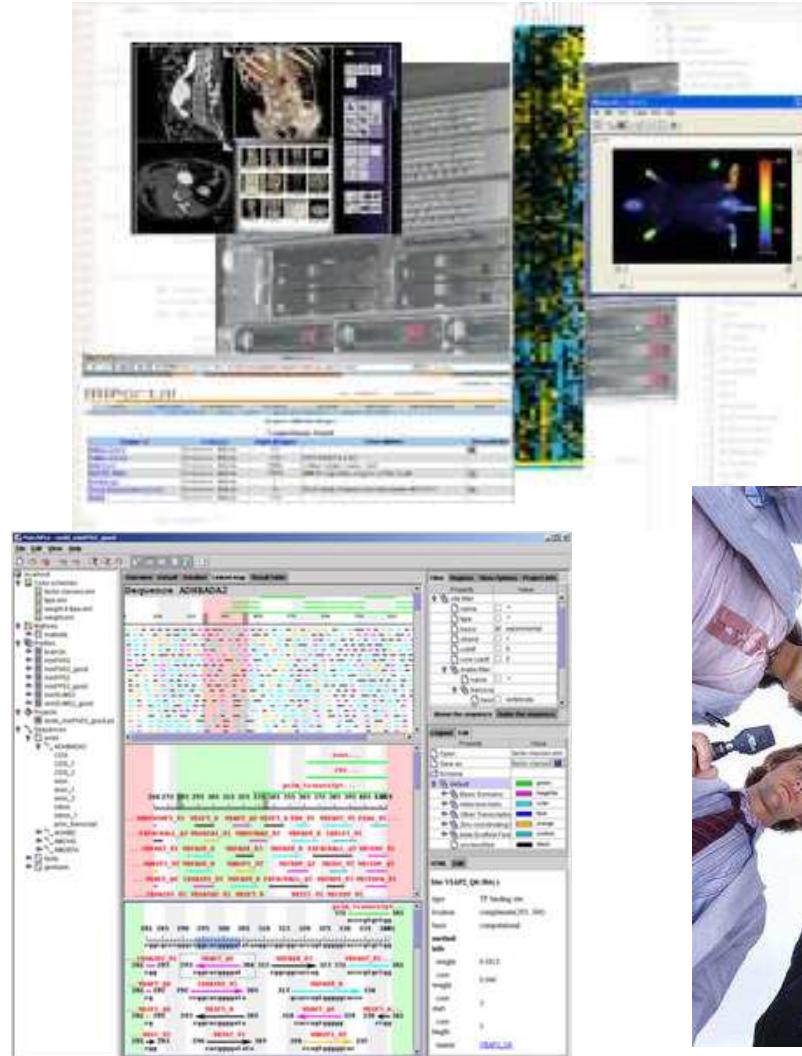


- ekološko modeliranje

- morski sustavi
(plimna dinamika,
temperatura, slanost,
širenje tvari u moru)

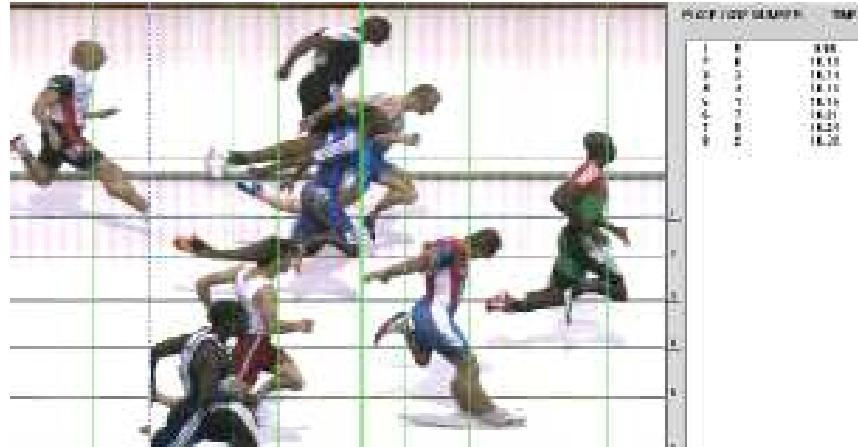
- rezultati mjerena
- satelitski podaci
- računalne simulacije

Obrada podataka - medicina



- podaci o pacijentu
- povijest bolesti
- rezultati testova
- podaci s bolničkih instrumenata i senzora

Obrada podataka – sport



Sarajevo		Zagreb													
10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01	10.01
10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02	10.02
10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03
10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04	10.04
10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05	10.05
10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06	10.06
10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07	10.07
10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08	10.08
10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09	10.09
10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10	10.10
10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11	10.11
10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12	10.12
10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13	10.13
10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14	10.14
10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15	10.15
10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16	10.16
10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17	10.17
10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18	10.18
10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19	10.19
10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20



- podaci o natjecateljima
- rezultati utakmica i utrka
- statistika

Obrada podataka – putovanja



- vozni red / red letenja
- rezervacija karata
- kupnja karata
- slobodna i zauzeta mjesta



Obrada podataka – (elektronička) trgovina

- Trgovina
 - računi
 - količina prodanih proizvoda
 - zarada
- Elektronička trgovina
 - narudžbe i prodaja putem Interneta



Analitički izvještaj		Analitički izvještaj	
Analitički izvještaj	Analitički izvještaj	Analitički izvještaj	Analitički izvještaj
Analitički izvještaj	Analitički izvještaj	Analitički izvještaj	Analitički izvještaj
Analitički izvještaj	Analitički izvještaj	Analitički izvještaj	Analitički izvještaj
Analitički izvještaj	Analitički izvještaj	Analitički izvještaj	Analitički izvještaj



Obrada podataka – portali

The image displays two web portal interfaces. The left portal is for the Faculty of Electrical Engineering and Computing (FER). It features a header with the FER logo and text 'Fakultet elektrotehnike i računarstva'. A large blue badge with 'ASIIN Accredited' is prominently displayed. The main content area shows a search result titled 'Pregled primjera' with several items listed. The right portal is for AHyCo (Faculty of Civil Engineering). It has a header with the AHyCo logo and a search bar. Below the search bar is a large yellow banner with the text 'ISVU •' and 'INTERDISCIPLINARY CENTER FOR STUDY OF MATERIALS'. Both portals have a sidebar with navigation links like 'Naslovica', 'O FER-u', 'Ustroj ustanove', etc.

- prikaz podataka iz različitih izvora

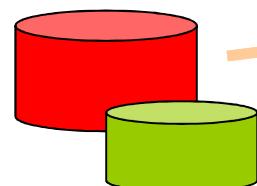
Obrada podataka – društvene mreže



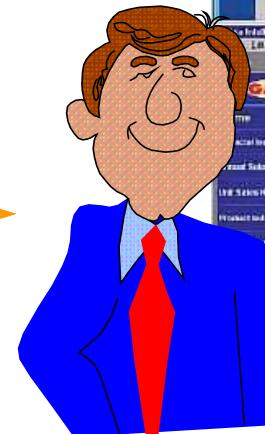
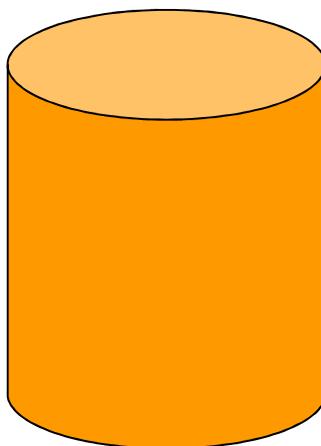
- Velik broj korisnika, konstantna aktivnost
- Facebook – 60 milijuna upita i 4 mil. promjena u sekundi

Obrada podataka – potpora odlučivanju

- integracija podataka iz različitih izvora



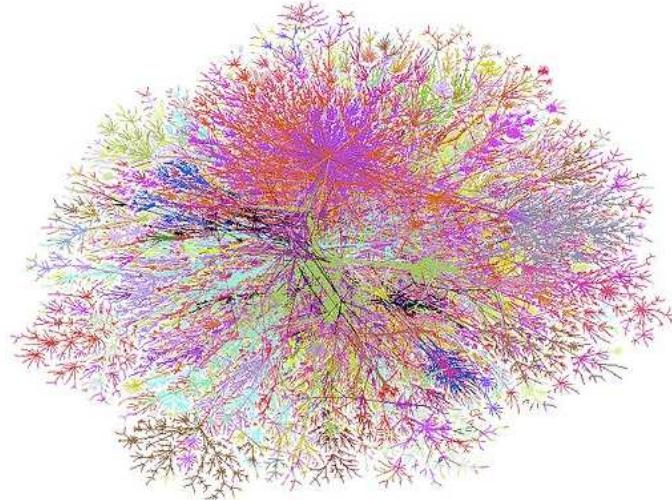
različiti izvori podataka



- izvještaji
- višedimenzionalni pogled na podatke
- dubinska analiza (data mining)
- vizualizacija podataka

Big Data / NoSQL sustavi

- Big Data 3Vs:
 - **Volume** = volumen, veličina
 - **Variety** = različitost, heterogenost
 - **Velocity** = brzina
- NoSQL – nerelacijski sustavi



Organizacija predmeta

Baze podataka

Ciljevi predmeta i očekivani ishod učenja

- Ovo je osnovni kolegij iz područja baza podataka kojemu je cilj upoznati studente sa sustavima za upravljanje bazama podataka, relacijskim modelom i relacijskim bazama podataka.
- Izučava se način oblikovanja relacijskih baza podataka i oblikovanje modela entiteti-veze, relacijska algebra, upitni jezik SQL i osnove zaštite baza podataka.
- Studenti će biti osposobljeni za modeliranje jednostavnijih baza podataka i postavljanje srednje složenih upita nad bazom podataka. Upoznat će se s osnovama zaštite baza podataka.

Ciljevi predmeta i očekivani ishod učenja

- Nakon uspješno savladanog predmeta, studenti će moći:
 - definirati osnovne koncepte baza podataka
 - opisati osnovne dijelove sustava za upravljanje bazama podataka
 - objasniti principe oblikovanja modela baza podataka
 - objasniti i razumjeti sintaksu i semantiku jezika SQL
 - objasniti osnovne principe zaštite baza podataka
 - primijeniti znanja o oblikovanju modela baza podataka na jednostavnije primjere iz prakse
 - upotrijebiti znanje relacijske algebre i SQL upita pri rješavanju novih zadaća

Predavači

- | | |
|-----|---|
| P01 | Prof.dr.sc. Zoran Skočir, ZOEM
Doc.dr.sc. Marko Banek, ZOEM |
| P02 | Prof.dr.sc. Boris Vrdoljak, ZPR |
| P03 | Prof.dr.sc. Mirta Baranović, ZPR |
| P04 | Doc.dr.sc. Ljiljana Brkić, ZPR |
| P05 | Doc.dr.sc. Damir Pintar, ZOEM
– predavanja na engleskom jeziku |

ZOEM – Zavod za osnove elektrotehnike i električna mjerena, zgrada C, VI kat

ZPR – Zavod za primjenjeno računarstvo, zgrada D, III kat

Asistenti

- ? Dr.sc. Krešimir Križanović, ZPR
- ? Dr.sc. Mihaela Vranić, ZOEM
- ? Mr.sc. Jasenka Anzil, ZPR

- ? Nikša Stanović, dipl. ing., ZPR
Tomislav Jagušt, dipl. ing., ZPR

ZOEM – Zavod za osnove elektrotehnike i električna mjerena, zgrada C, VI kat

ZPR – Zavod za primijenjeno računarstvo, zgrada D, III kat

Administracija

- ZPR – zgrada D, III kat - gđa. Sonja Majstorović
- ZOEM – zgrada C, VI kat - gđa. Jasenka Haladin

Literatura

- J. D. Ullman, J.Widom: **A First Course in Database Systems**, Prentice-Hall, 2001.
- A. Silberschatz, H.F. Korth, S. Sudarshan: **Database Systems Concepts**, 5th Edition, McGraw-Hill, 2005.
- C.J. Date: **An Introduction to Database Systems**, 8th Edition, Addison Wesley, 2003.
- T. M. Connolly, C. E. Begg: **Database Systems: A Practical Approach to Design, Implementation, and Management**, Addison Wesley, 2004.
- S. Tkac, **Relacijski model podataka**, DRIP, 1988.
- M. Varga: **Baze podataka**, DRIP, 1994.

Upute i obavijesti

- na URL stranici predmeta

<http://www.fer.hr/predmet/bazepod>

- forum na stranici predmeta služi isključivo za međusobnu komunikaciju studenata

Organizacija nastave

PREDAVANJA

- *Powerpoint* prezentacije
 - datoteke u PDF formatu nalazit će se u repozitoriju datoteka na URL stranici predmeta
 - Odabrani primjeri prikazivat će se pomoću prikladnih programskih alata

Organizacija nastave

SAMOSTALAN RAD

- Učenje (slajdovi s predavanja i ostala literatura)
- Vježbanje, rješavanje domaćih zadaća
 - rad na vlastitom računalu (kod kuće, u domu, ...)
 - instaliranje potrebnih programskih sustava i alata na vlastitom računalu (na raspolaganju će biti detaljne upute)

Instalacija računalne podrške

- Upute i programska podrška potrebna za obavljanje vježbi nalazi se na adresi:

<http://kent.zpr.fer.hr/bp/>

- Programska podrška uključuje:
 - virtualno računalo s instaliranim Informix SUBP i studAdmin bazom podataka
 - Vmware player
 - Server Studio
 - upute za instalaciju
- Podrška je prilagođena za Windows OS, no postoje i verzije za druge operacijske sustave

Interakcija među sudionicima nastave

KONZULTACIJE

- Asistenti
 - unaprijed određeni termini konzultacija
 - e-mail (*ime.prezime@fer.hr*)
- Predavači
 - unaprijed određeni termini konzultacija
 - e-mail (*ime.prezime@fer.hr*)
- Termini će biti objavljeni na stranici predmeta.

Kontinuirano praćenje - elementi ocjenjivanja

- Kratke provjere znanja na predavanjima ⇒ 5 bodova
 - rješavanje 4 domaćih zadaća ⇒ 4 boda
 - kontrolne zadaće 3 x 7 bodova ⇒ 21 bod
 - međuispit ⇒ 30 bodova
 - završni ispit ⇒ 40 bodova
-
- UKUPNO ⇒ 100 bodova
-
- Za prolaznu ocjenu potrebno je zadovoljiti **dva uvjeta**:
 - ostvariti ukupno ≥ 50 bodova
 - ostvariti ≥ 8 bodova na završnom ispitu

Međuispit i završni ispit

- Na međuispitu se provjerava znanje gradiva 1. nastavne cjeline
- Na završnom ispitu se provjerava znanje **čitavog** gradiva

Ispitni rokovi

- Za izlazak na ispitni rok je potrebno zadovoljiti **dva uvjeta**:
 - Min. 30% (**6,3 bodova**) na kontrolnim zadaćama
 - Min. 50% (**2 boda**) na domaćim zadaćama

Ankete

- Studenti sudjeluju u vrednovanju kvalitete nastave i nastavnika putem ankete. Anketiranje se provodi za svaki predmet dva puta tijekom semestra.

Članak 20. stavak 5. Pravilnika o sveučilišnom preddiplomskom i diplomskom studiju na Sveučilištu u Zagrebu, Fakultetu elektrotehnike i računarstva

Uvod u baze podataka

Organizacijski sustav

- ORGANIZACIJSKI SUSTAV je složeni sustav koji sadrži tehničke i humane podsustave
 - poduzeće, ustanova, djelatnost, društvena organizacija, tehnički sustav kao npr. telekomunikacijska mreža i sl.
 - često se organizacijski sustav naziva i POSLOVNIM SUSTAVOM, iako pojam organizacijski sustav ima nešto šire značenje.
- Primjeri organizacijskih sustava:
 - Knjižnica
 - Sveučilište
 - Zračna luka

Informacija, podatak (*Information, Data*)

- INFORMACIJA je sadržaj koji primatelju opisuje nove činjenice.
- Taj sadržaj se materijalizira u obliku PODATAKA koji služe za prikaz informacija u svrhu spremanja, prijenosa i obrade.
- Podatak je skup simbola (znakova).
- Informacija je i obrađeni podatak koji za primatelja ima karakter novosti, otklanja neizvjesnost i služi kao podloga za odlučivanje.
 - Podatak izvan konteksta nema značenja
 - **podatak:** **4.62**
 - Podatak koji interpretiramo i primjereno povežemo predstavlja informaciju
 - **informacija:** prosjek svih ocjena studenta Marka Horvata na studiju na FER-u u ovom trenutku je **4.62**

Informacijski sustav (*Information System*)

Definicija:

- Ukupna infrastruktura, organizacija, osoblje i komponente koje služe za prikupljanje, obradu, pohranu, prijenos, prikaz, širenje i raspolaganje informacijama
- *The entire infrastructure, organization, personnel, and components for the collection, processing, storage, transmission, display, dissemination, and disposition of information [INFOSEC-99].*

Informacijski sustav (*Information System*)

- Informacijski sustav je dio svakog organizacijskog sustava
- Svrha mu je prikupljanje, obrada, pohranjivanje i distribucija informacija, koje su potrebne za praćenje rada i upravljanje organizacijskim sustavom ili nekim njegovim podsustavom.
 - Informacijski sustav je aktivni sustav koji može (ali ne mora) koristiti suvremenu informacijsku tehnologiju
 - Središnji dio informacijskog sustava je BAZA PODATAKA

Informacijski sustav, 1868.



Baza podataka (*Database*)

- BAZA PODATAKA je skup podataka koji su pohranjeni i organizirani tako da mogu zadovoljiti zahtjeve korisnika.
(M. Vetter, 1981.)
- BAZA PODATAKA je skup međusobno povezanih podataka, pohranjenih zajedno, uz isključenje bespotrebne zalihosti (redundancije), koji mogu zadovoljiti različite primjene. Podaci su pohranjeni na način neovisan o programima koji ih koriste. Prilikom dodavanja novih podataka, mijenjanja i pretraživanja postojećih podataka primjenjuje se zajednički i kontrolirani pristup. Podaci su strukturirani tako da služe kao osnova za razvoj budućih primjena.
(J. Martin, 1979.).

Entitet (*Entity*)

- bilo što, što ima suštinu ili bit i posjeduje značajke s pomoću kojih se može razlučiti od svoje okoline
- osobe: studenti, radnici, građani, ...
 - student Horvat Ivan (0036123456)
- ostala bića
 - Bijeli bor, pas Lajka
- objekti: vozila, strojevi, uređaji, ulice, zgrade, mjesta, ...
 - Eiffelov toranj, cepelin Hindenburg
- apstraktni pojmovi: boje, predmeti nastavnog programa, ...
 - predmet Baze podataka u nastavnom programu FER-2
- događaji (nešto se desilo, dešava se ili se planira da će se desiti)
 - dana 24.11.2013. održava se proslava Dana Fakulteta na FER-u
- povezanost među objektima, osobama, događajima, ...
 - student Horvat Ivan (0036123456) **stanuje u Zagrebu**
- nešto o čemu želimo prikupljati i pohranjivati podatke

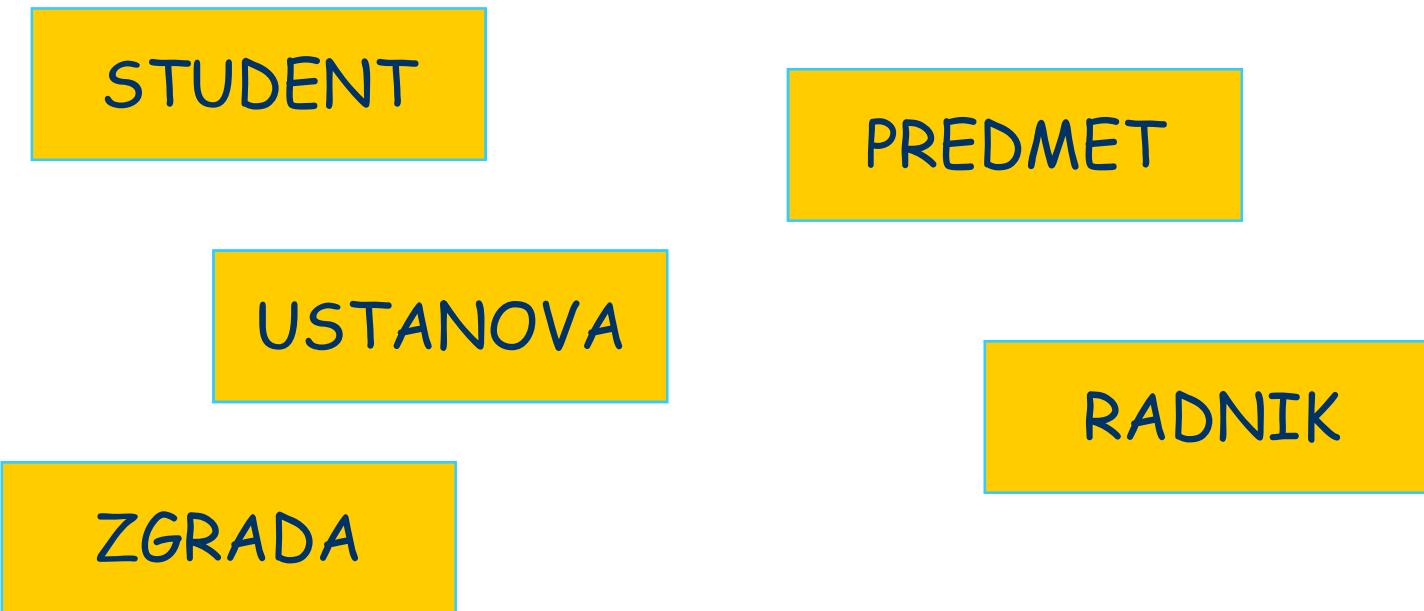
Atribut (*Attribute*)

- Entitet posjeduje neka SVOJSTVA ili ATRIBUTE koji ga karakteriziraju.
 - Za Informacijski sustav visokih učilišta (ISVU) važna svojstva studenta Horvat Ivana su:
 - Matični broj studenta (JMBAG)
 - Ime
 - Prezime
 - Datum rođenja, ...
- Izbor svojstava (atributa) koje ćemo pratiti ovisi o namjeni informacijskog sustava
 - Horvat Ivan u informacijskom sustavu MUP-a bit će karakteriziran i atributima:
 - Boja kose
 - Boja očiju
 - Otisak prsta, ...

Skup entiteta (*Entity Set*)

- Slični entiteti se svrstavaju u skupove entiteta
- Slični su oni entiteti kojima se promatraju ista svojstva
- Svi entiteti koji su članovi istog skupa entiteta imaju iste atributе
- "atributi entiteta" \Leftrightarrow "atributi skupa entiteta"
 - atributi skupa entiteta PREDMET
 - sifPred
 - nazPred
 - ectsBod
 - nastProg
 - atributi skupa entiteta STUDENT
 - jmbag
 - ime
 - prezime
 - datRod

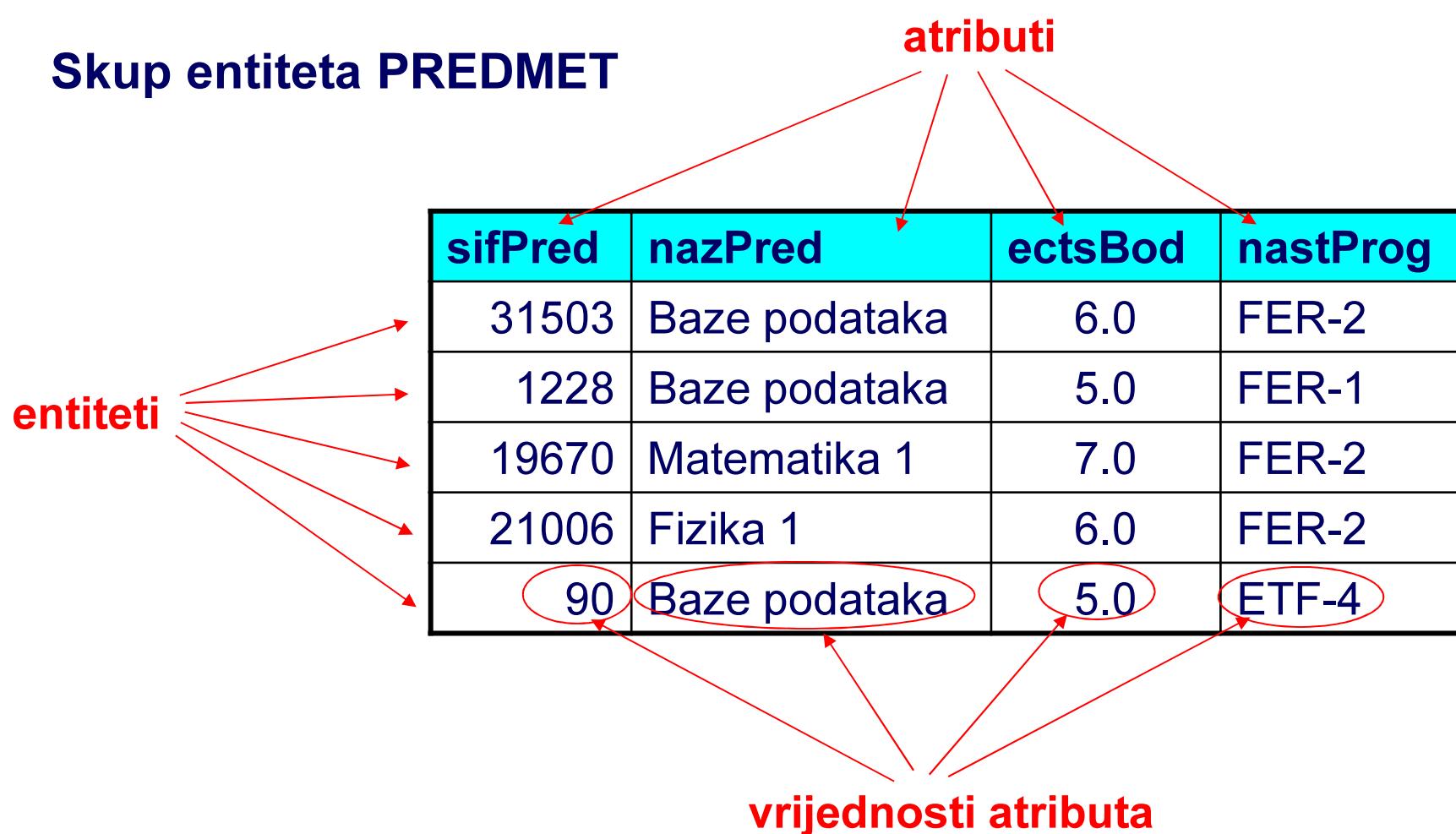
Skupovi entiteta - primjer



Domena i vrijednost atributa (*Domain, Attribute Value*)

- Za svaki entitet, atribut poprima vrijednosti iz određenog skupa vrijednosti koji predstavlja **domenu** tog atributa
 - domene atributa za skup entiteta **PREDMET**
 - sifPred: skup šifara predmeta - cijelih brojeva iz intervala [1,999999]
 - nazPred: skup naziva predmeta - nizova znakova duljine do 80 znakova
 - ectsBod: skup vrijednosti ECTS bodova - realnih brojeva iz intervala [0.5, 30.0] (s jednom znamenkom iza decimalne točke)
 - nastProg: skup oznaka nastavnih programa - nizova znakova duljine do 5 znakova
 - vrijednosti atributa za entitet **Baze podataka (31503)**
 - sifPred: 31503
 - nazPred: Baze podataka
 - ectsBod: 6.0
 - nastProg: FER-2

Skup entiteta - prikaz u obliku tablice



Identifikatori entiteta, ključevi

- Skupove atributa čije vrijednosti jednoznačno određuju entitet u promatranom skupu entiteta (dakle ne postoji dva entiteta s posve istim vrijednostima tih atributa) nazivamo IDENTIFIKATORIMA ili KLJUČEVIMA SKUPA ENTITETA.
- **Primjer:** u skupu entiteta PREDMET prikazanom na slici:

sifPred	nazPred	ectsBod	nastProg
31503	Baze podataka	6.0	FER-2
1228	Baze podataka	5.0	FER-1
19670	Matematika 1	7.0	FER-2
21006	Fizika 1	6.0	FER-2
90	Baze podataka	5.0	ETF-4

- skup atributa { sifPred } **jest** ključ skupa entiteta
- skup atributa { nazPred } **nije** ključ skupa entiteta
- skup atributa { nazPred, nastProg } **jest** ključ skupa entiteta

Modeliranje stvarnog svijeta

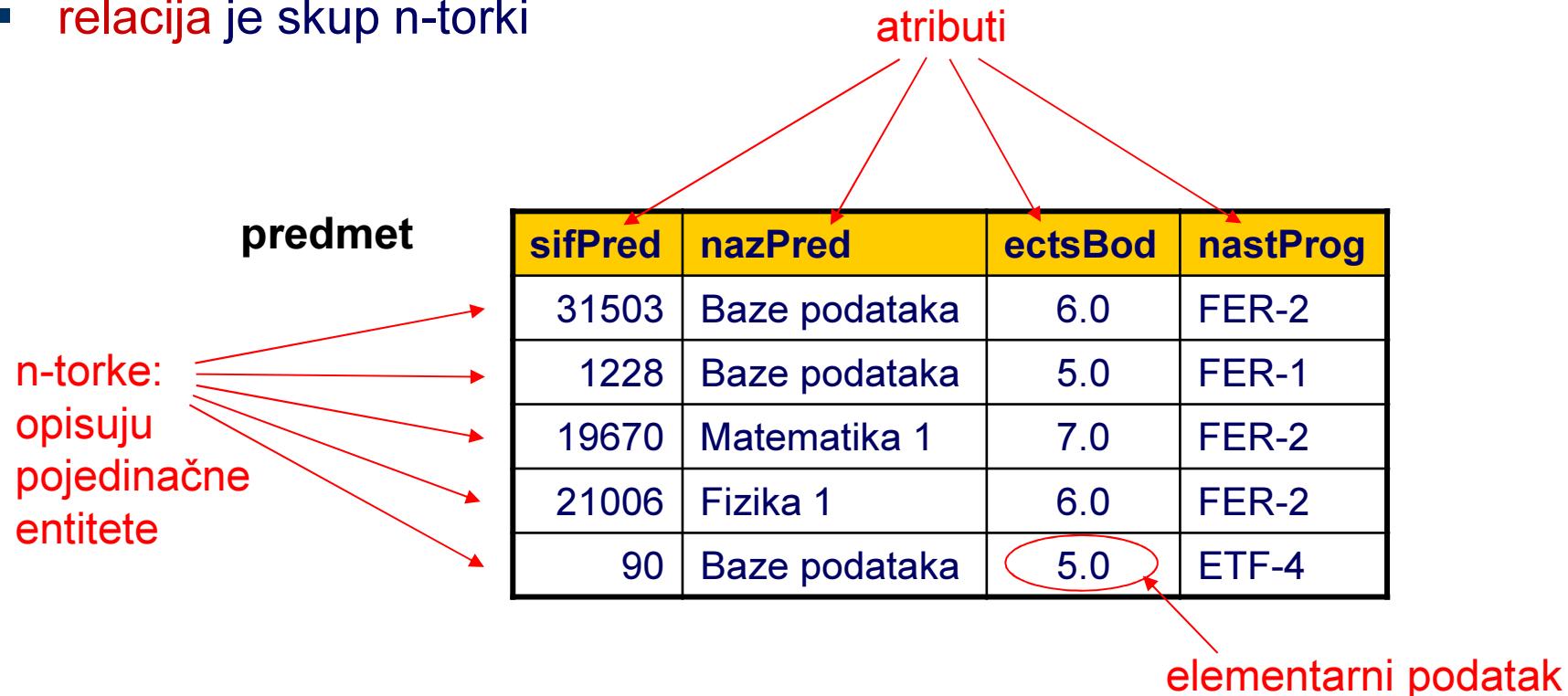
- Modeliranje stvarnog svijeta predstavlja preslikavanje stvarnog svijeta u oblik pogodan za računalnu obradu;
- Baza podataka nekog informacijskog sustava predstavlja sliku stvarnog organizacijskog sustava;
- Stvarni svijet, zbog njegove složenosti, ne možemo prikazati sa svim detaljima;
- Stvarni svijet predstavlja se pojednostavnjениm, nadomjesnim modelom;
- Model stvarnog svijeta predstavlja se uz pomoć nekog formalnog sustava;
- **Model podataka** je formalni sustav koji koristimo kod modeliranja baza podataka

Model podataka (*Data Model*)

- **Model podataka** je **formalni sustav** koji se sastoji od:
 - **skupa objekata** - osnovnih elemenata (koncepata) baze podataka
 - **skupa operacija** koje se provode nad objektima
 - **skupa integritetskih ograničenja** (*integrity constraints*)
 - implicitno ili eksplicitno definiraju skup konzistentnih stanja podataka, promjena stanja, ili oboje
- **Povijesni razvoj modela podataka:**
 - Hiperarhijski model
 - Mrežni model
 - Relacijski model
 - ER model
 - Objektni model
 - Objektno-relacijski model

Relacijski model podataka – objekti

- elementi skupa objekata u relacijskom modelu podataka su **relacije**
- relacija** je skup n-torki



- shema relacije** obuhvaća naziv relacijske sheme (PREDMET) i skup atributa: (sifPred, nazPred, ectsBod, nastProg)

Relacijski model podataka – operacije

- operacija "selekcija" u relacijskom modelu podataka:

predmet	sifPred	nazPred	ectsBod	nastProg
	31503	Baze podataka	6.0	FER-2
	1228	Baze podataka	5.0	FER-1
	19670	Matematika 1	7.0	FER-2
	21006	Fizika 1	6.0	FER-2
	90	Baze podataka	5.0	ETF-4

$\sigma_{ectsBod=6.0}$ (predmet)

sifPred	nazPred	ectsBod	nastProg
31503	Baze podataka	6.0	FER-2
21006	Fizika 1	6.0	FER-2

- ostale operacije u relacijskom modelu podataka
 - unija, razlika, presjek, projekcija, ...

Relacijski model podataka – integritetska ograničenja

- pravilo domenskog integriteta u relacijskom modelu podataka:

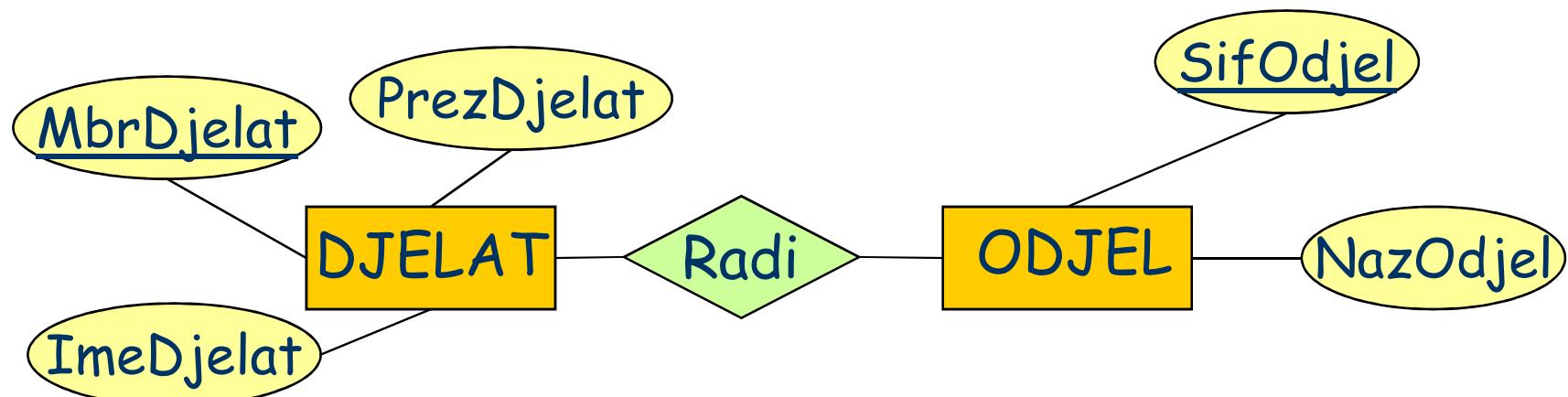
predmet	sifPred	nazPred	ectsBod	nastProg
	31503	Baze podataka	6.0	FER-2
	1228	Baze podataka	5.0	FER-1
	19670	Matematika 1	7.0	FER-2
	21006	Fizika 1	6.0	FER-2
	90	Baze podataka	5.0	ETF-4

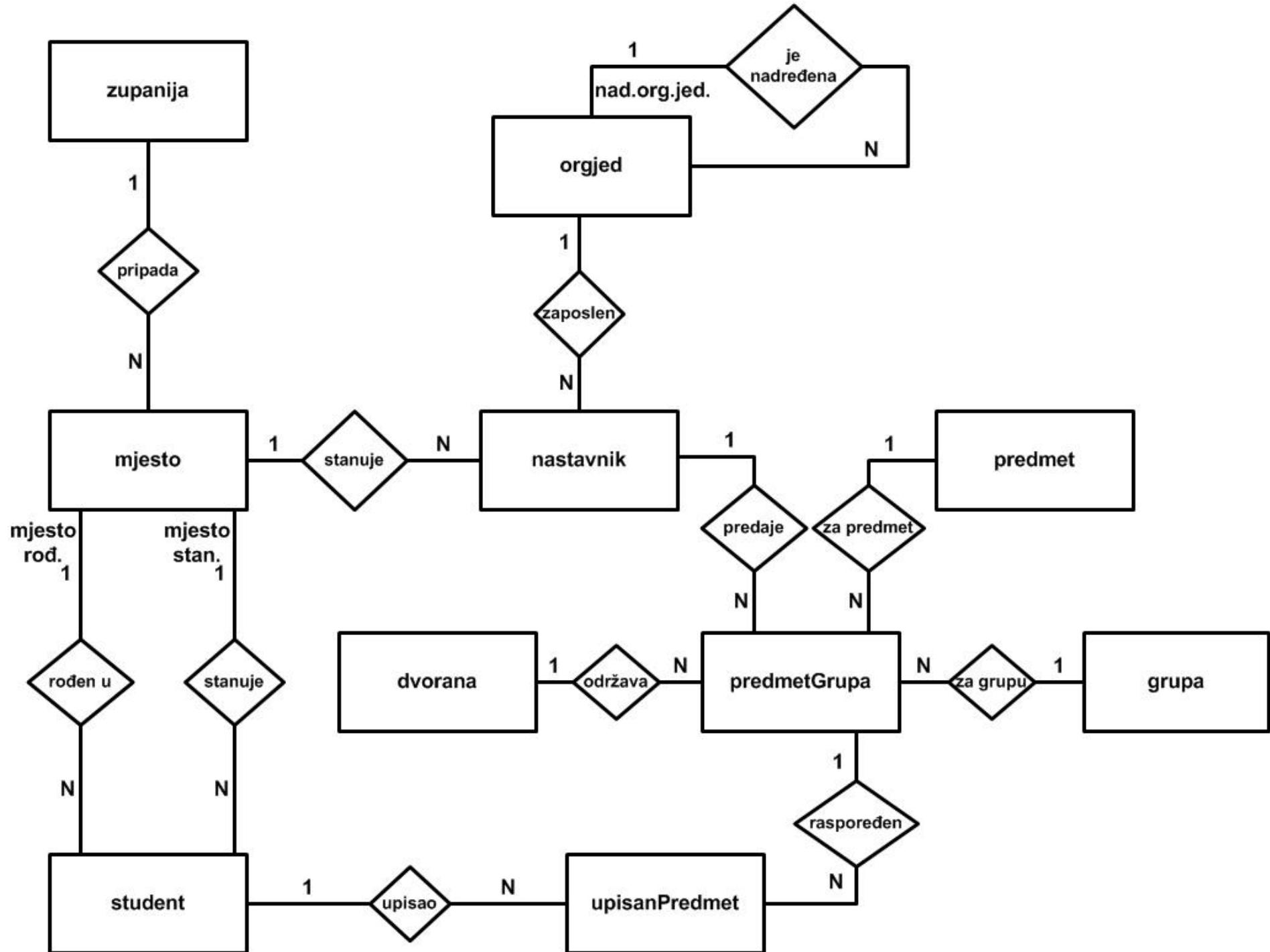
- domenski integritet:
 - vrijednost atributa sifPred mora biti iz intervala [1, 999999]
 - vrijednost atributa ectsBod mora biti iz intervala [0.5, 30.0]
 - ...
- ostala integritetska ograničenja u relacijskom modelu podataka:
 - entitetski, referencijski, ...

ER model podataka

Model entiteta-veze *Entity-Relationship Model*

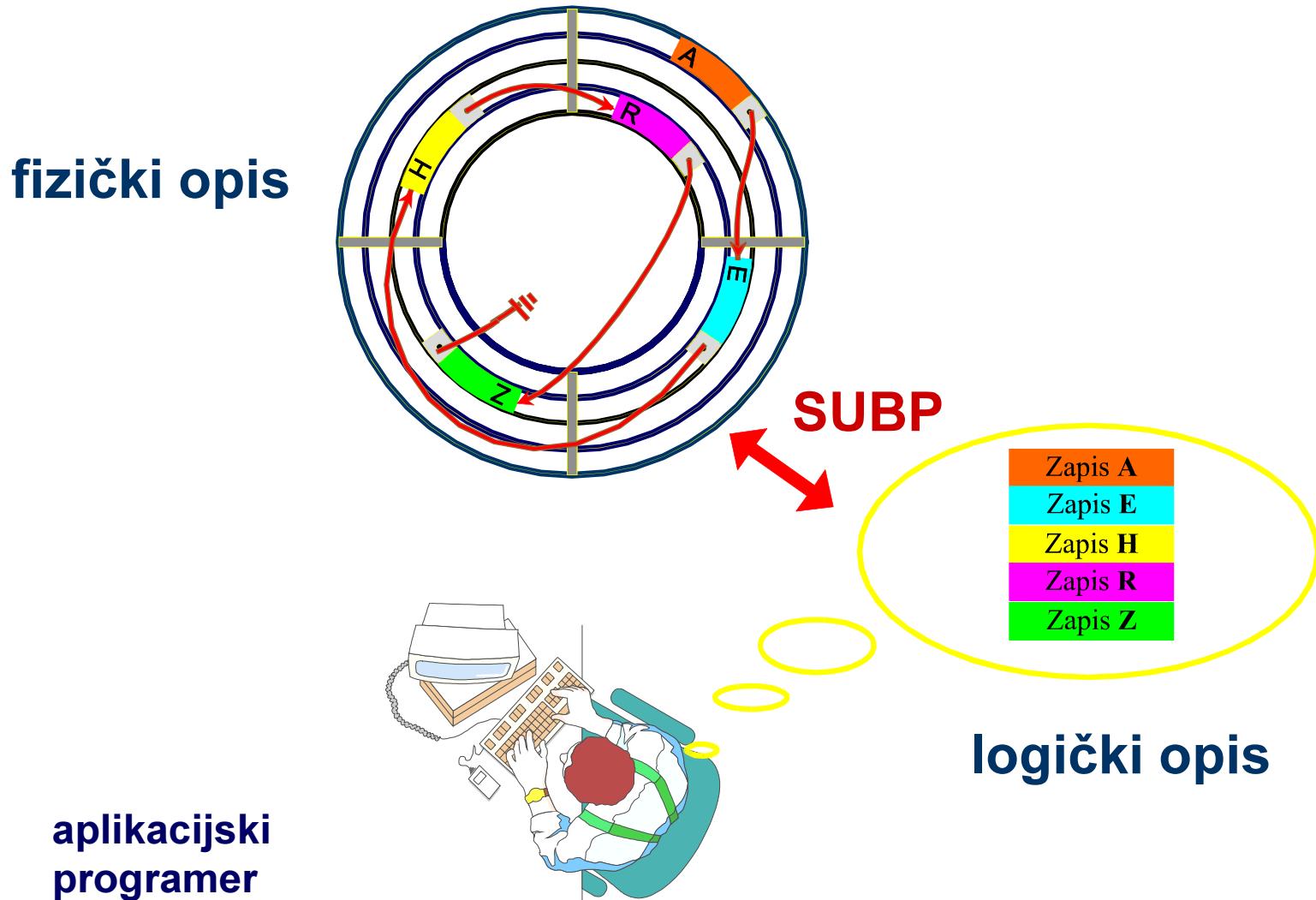
- Postrelacijski model
- Zadržava dobra svojstva relacijskog modela
- Objekti ER modela su entiteti i njihove međusobne veze



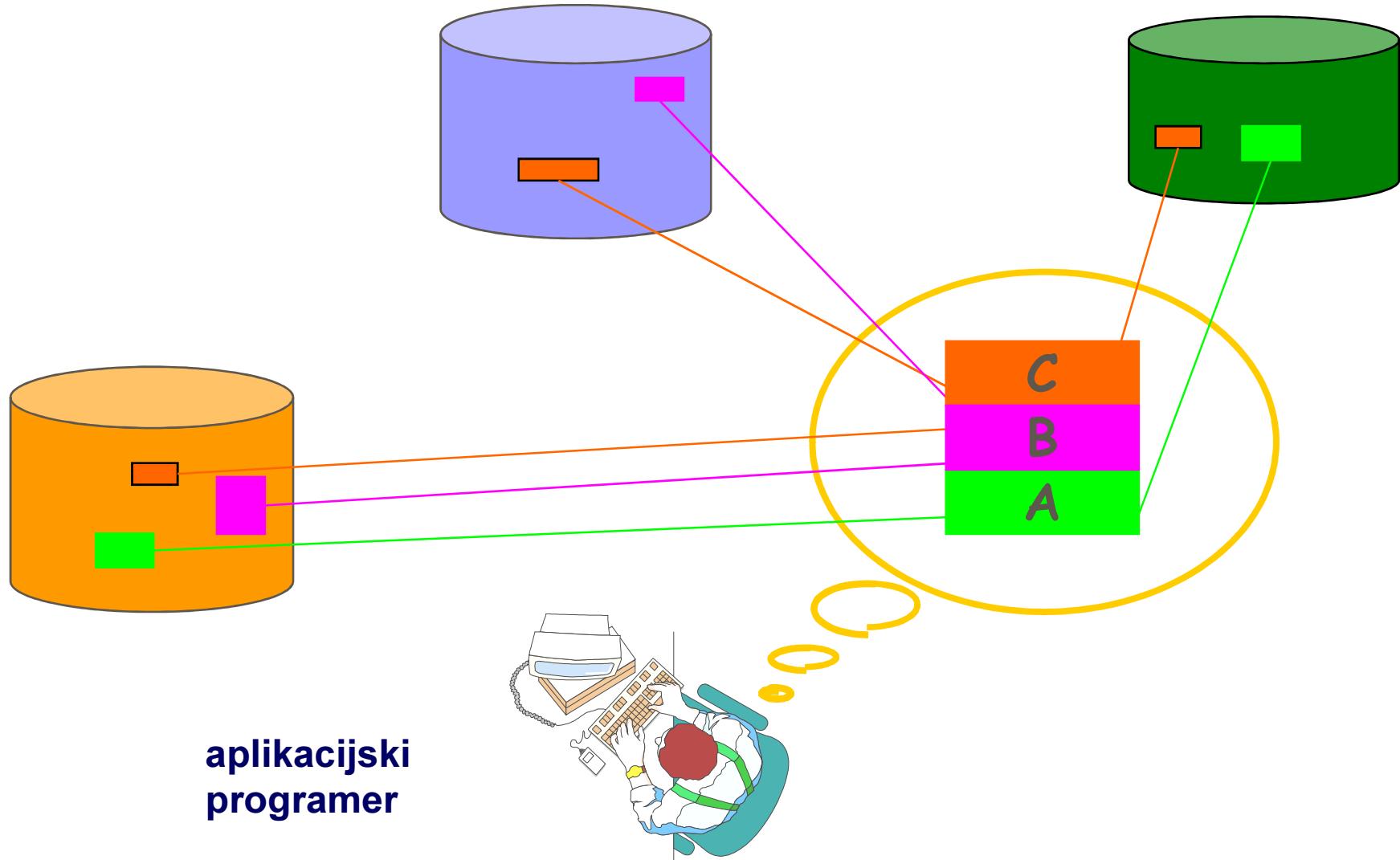


Arhitektura baze podataka

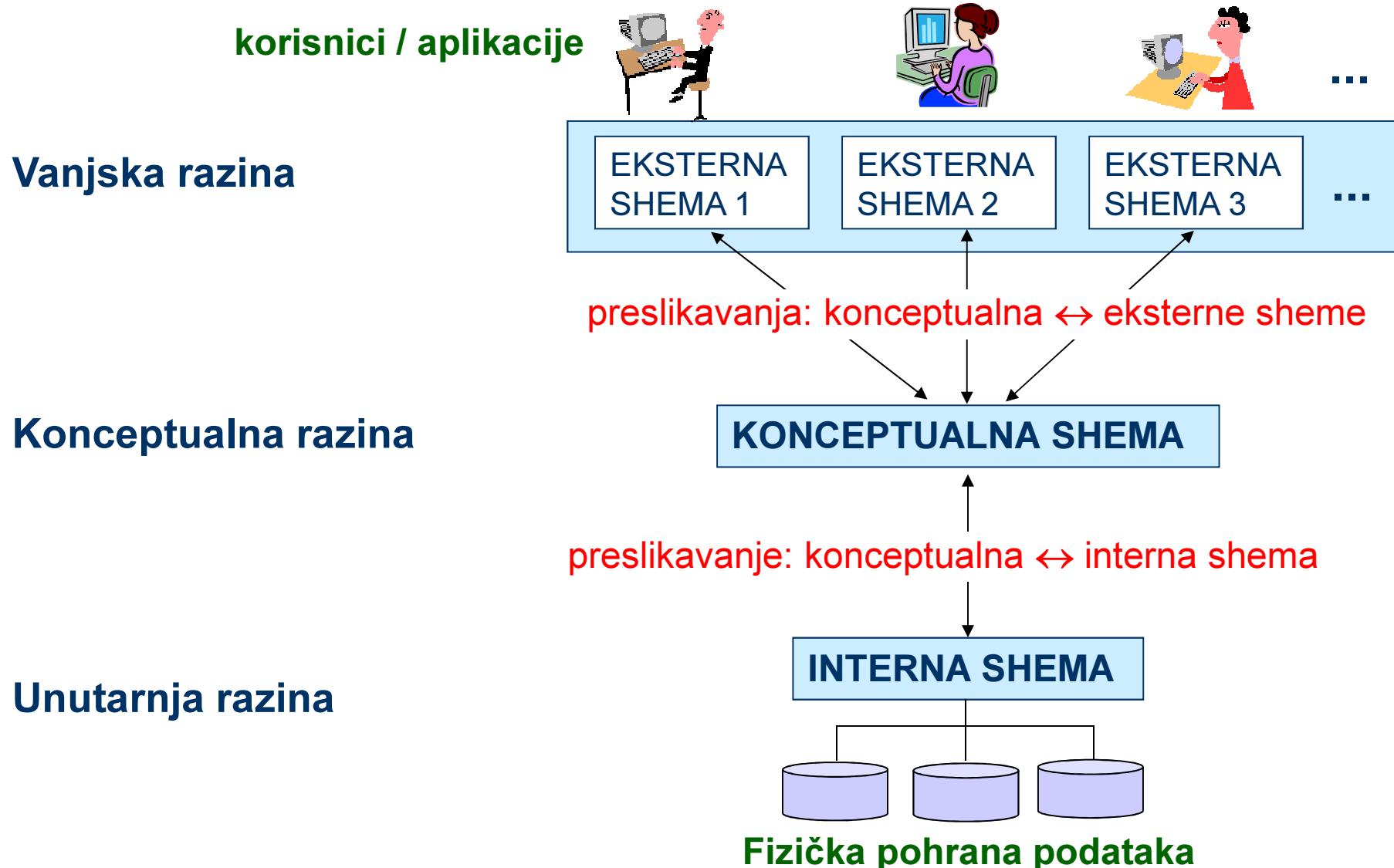
Fizička i logička organizacija podataka



Fizička i logička organizacija podataka



Arhitektura baze podataka

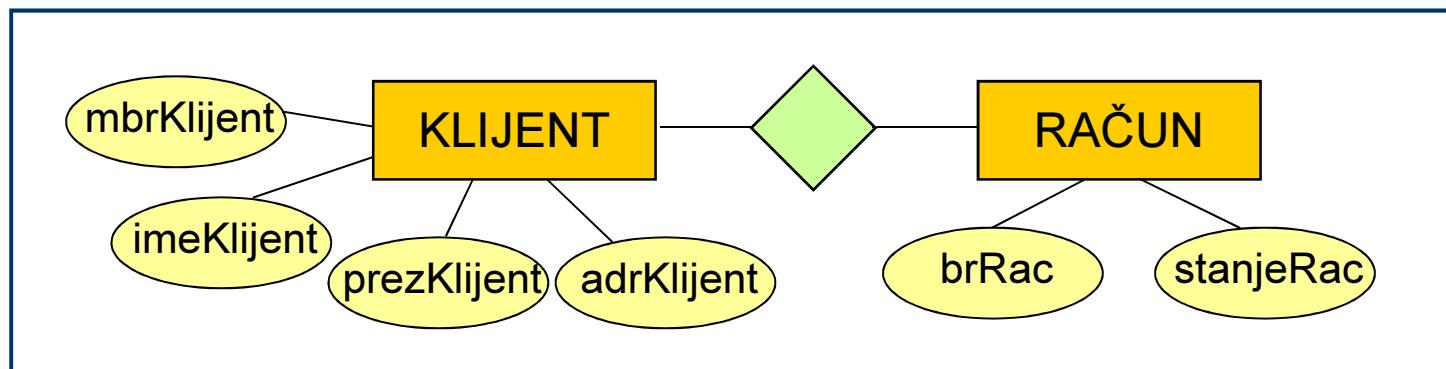


Arhitektura baze podataka

- Shema (struktura) baze podataka se opisuje na tri razine apstrakcije:
 - Na konceptualnoj razini opisuje se
 - **KONCEPTUALNA SHEMA**
 - Na unutarnjoj razini opisuje se
 - **INTERNA SHEMA**
 - Na vanjskoj razini opisuju se
 - **EKSTERNE SHEME**
- Jedna baza podataka ima jednu konceptualnu, jednu internu i (najčešće) više eksternih shema
- Shema baze podataka se *relativno rijetko mijenja*
- Sadržaj ili instanca baze podataka (skup svih podataka baze podataka u određenom trenutku) se **ČESTO mijenja**

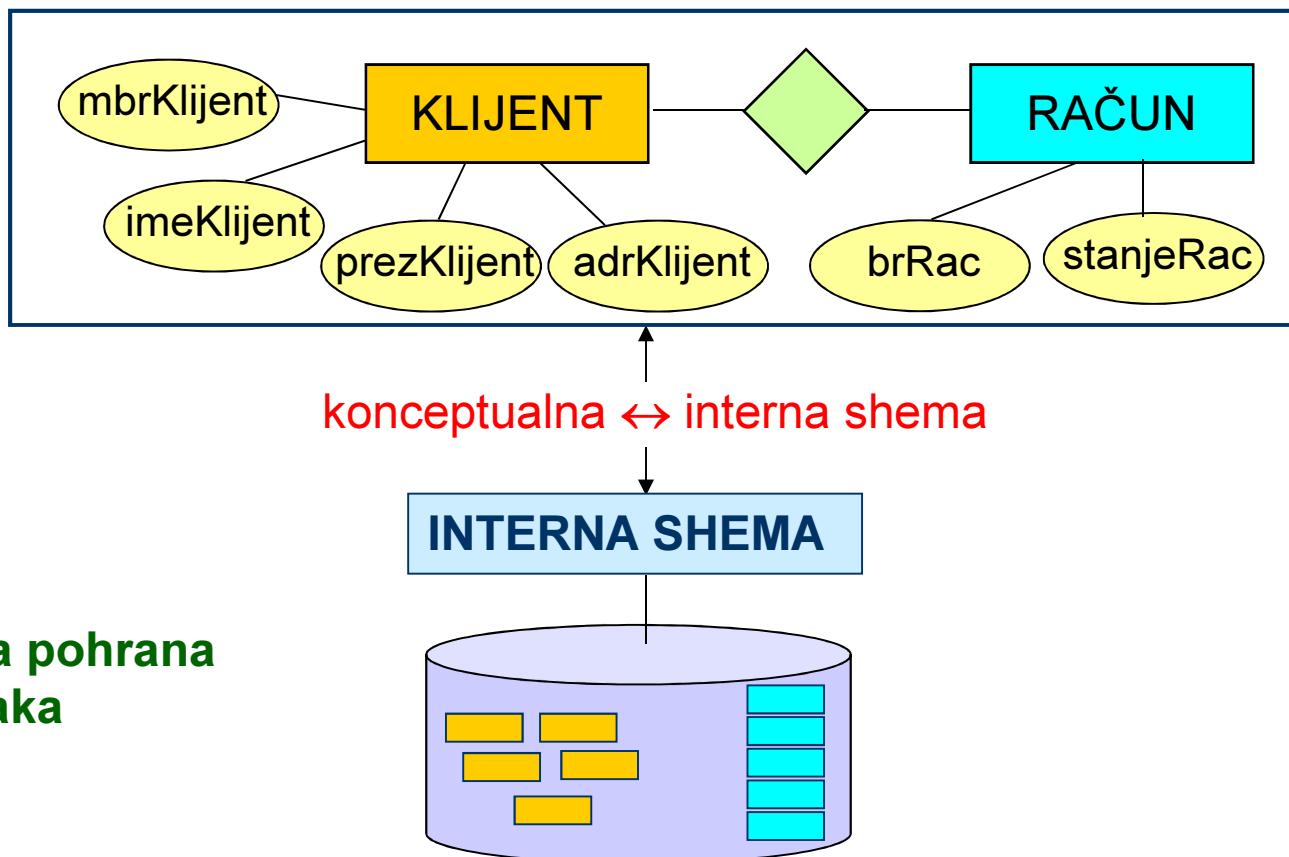
Konceptualna shema

- često se koristi i naziv LOGIČKA SHEMA
- sadrži opis svih entiteta i veza, atributa, domena i integritetska ograničenja
- konceptualna shema se može opisati korištenjem modela podataka, npr. relacijskog ili ER modela



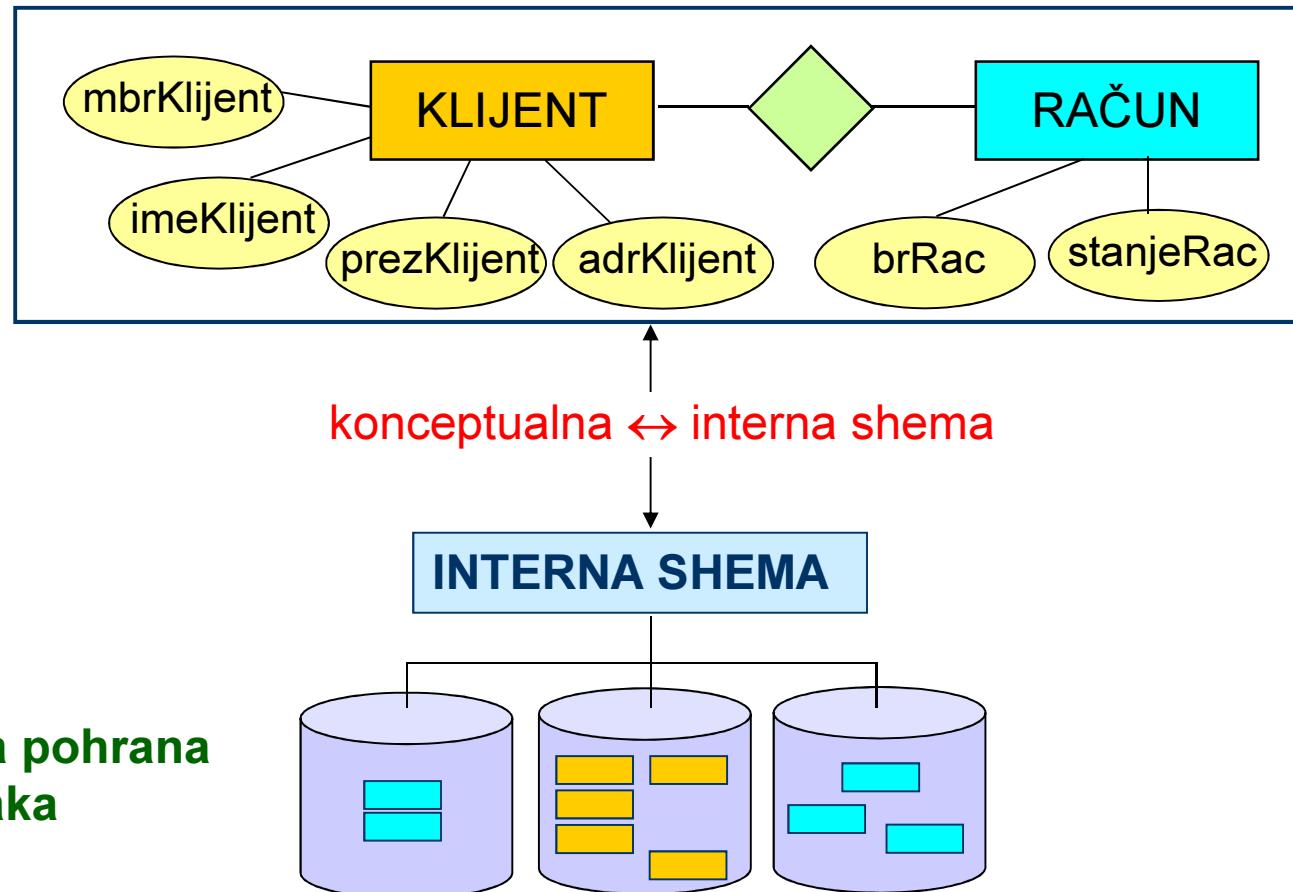
Interna shema

- opisuje detalje fizičke strukture pohrane i metode pristupa podacima: kako su podaci pohranjeni i koje se metode koriste za pristup podacima



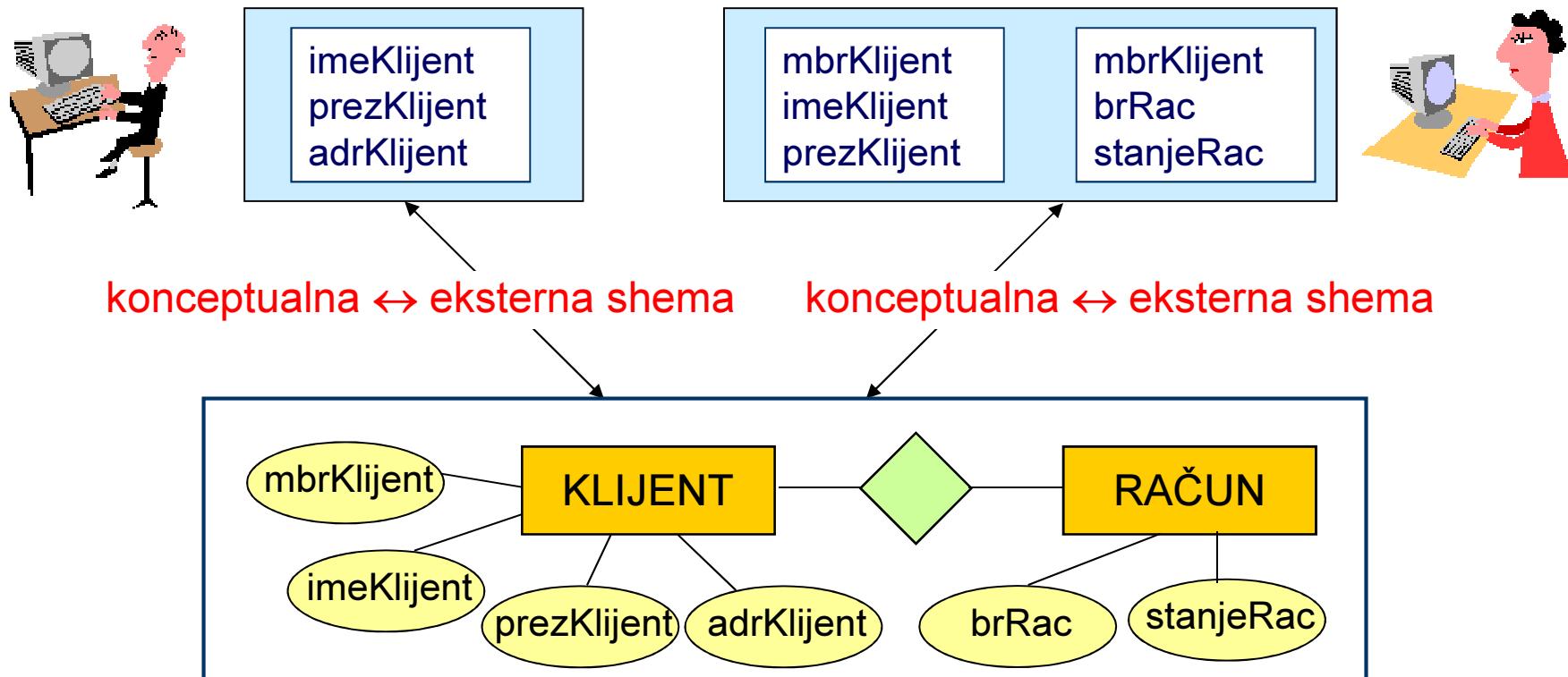
Fizička nezavisnost podataka

- izmjena interne sheme ne utječe na konceptualnu shemu



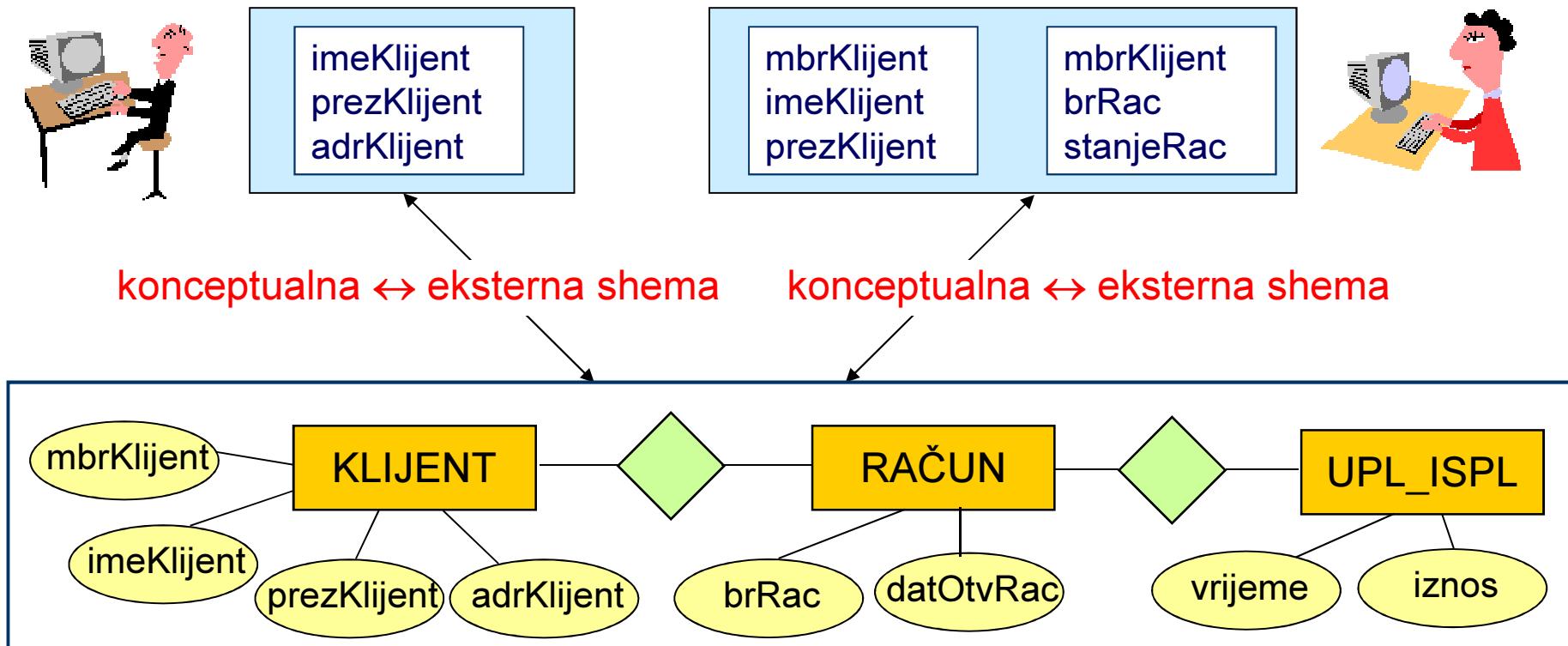
Eksterna shema

- eksterna shema opisuje "pogled" na dio baze podataka koji je namijenjen specifičnoj grupi korisnika
- osnova za opis eksternih shema je konceptualna shema



Logička nezavisnost podataka

- izmjena konceptualne sheme ne mora izazvati izmjenu eksternih shema → izmjena konceptualne sheme ne utječe na korisnike i aplikacijske programe koji ih koriste



Sustav za upravljanje bazom podataka - SUBP

Database Management System - DBMS

Sustav za upravljanje bazom podataka

Sustav za upravljanje bazom podataka - SUBP (*Database Management System - DBMS*) je **programski sustav** koji omogućava upravljanje podacima u bazi podataka. SUBP se temelji na odabranom modelu podataka.

Prema modelu podataka na kojem se temelje, **SUBP-ove** dijelimo na:

- hijerarhijske,
- mrežne,
- relacijske,
- objektno-relacijske,
- objektno-orientirane.

Sustav za upravljanje bazom podataka

- sakriva od korisnika detalje fizičke pohrane podataka
- osigurava logičku i fizičku nezavisnost podataka
- omogućuje definiciju i rukovanje podacima
 - DDL - Data Definition Language
 - DML - Data Manipulation Language
- obavlja funkciju zaštite podataka
 - integritet podataka
 - pristup podacima - autorizacija, sigurnost
 - kontrola paralelnog pristupa
 - obnova u slučaju razrušenja
- obavlja optimiranje upita

Jezici baze podataka

- DDL (*Data Definition Language*)
 - omogućava imenovanje i opis entiteta, atributa, veza i pripadnih ograničenja integriteta i pravila sigurnosti
 - koristi se za definiranje nove sheme baze podataka ili modificiranje postojeće
 - obavljanje DDL operacija rezultira izmjenom sadržaja rječnika podataka (metapodataka)
- DML (*Data Manipulation Language*)
 - omogućava korištenje skupa operacija za rukovanje podacima u bazi podataka
 - upitni jezik (*query language*)
 - ne sasvim korektno, pojam se koristi ne samo za operacije dohvata podataka, već i za operacije izmjene, brisanja i unosa podataka
 - proceduralni jezici, neproceduralni jezici

**Zašto koristimo sustave za upravljanje
bazama podataka?**

**Zašto ne bismo koristili samo
datotečne sustave?**

Zašto SUBP, a ne samo datoteke?

1. Jednostavan pristup podacima

Datotečni sustav - podaci su raspršeni po datotekama koje mogu biti u različitim formatima. Za dobivanje odgovora na netipična pitanja potrebno je ili dotjerivati stare ili pisati nove programe. Primjer:

Potrebno je naći sve korisnike banke koji žive u području s određenim poštanskim brojem. Pretpostavimo da ne postoji gotovi program koji može izdvojiti tražene korisnike, ali postoji program koji vraća sve korisnike. Dva su moguća rješenja:

- a) službenik će iz popisa svih korisnika ručno izdvojiti one s određenim poštanskim brojem;
- b) tražit će se od programera da napiše odgovarajući program.

Niti jedno od ova dva rješenja nije zadovoljavajuće. Svaki put kad se pojavi nova vrsta upita, opet će se morati ponoviti postupak

- SUBP omogućuje fleksibilniji dohvati i izmjenu podataka.

Zašto SUBP, a ne samo datoteke?

2. Upravljanje zalihostima i nekonzistentnošću

Isti podaci mogu biti pohranjeni u više datoteka, što može voditi ka nekonzistentnosti.

Prepostavimo da se adrese korisnika bankovnog računa pohranjuju u dvije datoteke (jedna za tekući, a druga za devizni račun). Nakon što je korisnik A promijenio adresu stanovanja, bankovni službenik promijenio je podatak o adresi u jednoj datoteci, ali je zaboravio promijeniti adresu u drugoj.

Ako se mjesecni izvještaji o stanju računa šalju na kućnu adresu korisnika na temelju adrese pohranjene u drugoj datoteci, korisnik A neće dobiti izvještaj na svoju novu adresu.

- SUBP omogućuje bolju kontrolu zalihosti od datotečnog sustava.

Zašto SUBP, a ne samo datoteke?

3. Transakcijska obrada

Treba prebaciti 1000 kuna s računa A na račun B (to je jedna transakcija).

Ako se dogodi hardverska ili softverska pogreška za vrijeme izvođenja programa, moguće je da se 1000 kuna skine s računa A, ali se ne uspije prebaciti na račun B, što dovodi do nekonzistentnosti u bazi podataka. Prebacivanje se mora obaviti u potpunosti.

- SUBP obično ima ugrađenu podršku za transakcijsku obradu, što je vrlo teško ostvariti u datotečnom sustavu.

4 . Složeni odnosi među podacima

- SUBP omogućuje predstavljanje različitih odnosa među podacima, definiranje novih odnosa kad se oni pojave te jednostavan dohvati i izmjenu međusobno povezanih podataka

Zašto SUBP, a ne samo datoteke?

5. Istovremeni pristup više korisnika

SUBP omogućuje većem broju korisnika pristup bazi podataka u isto vrijeme. Pri tome treba osigurati da u slučaju kad više korisnika koji nastoje promijeniti isti podatak, to se čini na kontrolirani način tako da rezultat izmjene bude točan i jednoznačan.

Pretpostavimo da službenici dve turističke agencije nastoje istovremeno rezervirati isto sjedalo u zrakoplovu. SUBP mora osigurati da rezervaciju određenog sjedala u određenom trenutku može obavljati najviše jedan službenik.

U datotečnom sustavu je kontrolu istovremenog pristupa daleko teže provesti.

Zašto SUBP, a ne samo datoteke?

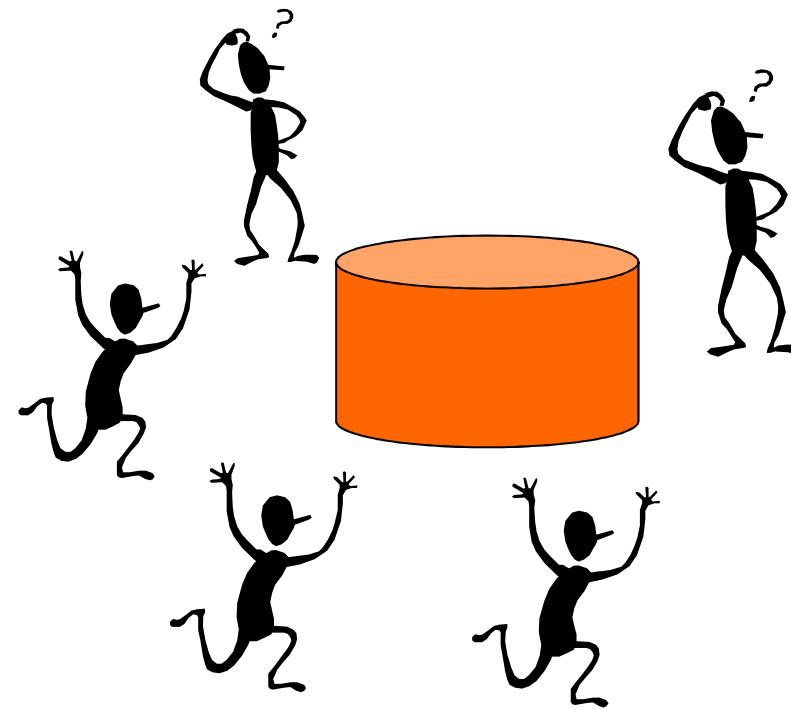
6. Autorizirani pristup

- Kad više korisnika koristi veću bazu podataka, obično neće svim korisnicima biti omogućen pristup svim podacima u bazi.

Na primjer, kako su podaci o financijama povjerljivi, samo će se autoriziranim osobama omogućiti pristup tim podacima.

- Neki korisnici će moći samo pregledavati podatke, dok će ih drugi moći i pregledavati i mijenjati.
- SUBP omogućava određivanje načina na koji će različiti korisnici pristupati podacima.
- U datotečnom sustavu je jednostavno odrediti je li neka datoteka namijenjena za čitanje, pisanje ili oboje, no nije jednostavno definirati različit način pristupa podacima za različite korisnike.

Uloge osoba u životnom ciklusu baze podataka



Baze podataka - uloge

1. Projektanti baze podataka

- razgovaraju s korisnicima da bi saznali njihove zahtjeve
- oblikuju bazu podataka prema zahtjevima korisnika definirajući strukturu za pohranu podataka (tj. model baze podataka)

2. Analitičari sustava i programeri aplikacija

- Analitičari sustava prikupljaju zahtjeve korisnika (npr. bankovnih službenika) i pišu specifikacije za razvoj aplikacija za pristup bazi podataka
- Programeri na temelju tih specifikacija izrađuju programe te ih testiraju, dokumentiraju i održavaju; moraju biti upoznati sa svojstvima i mogućnostima SUBP-a

Baze podataka - uloge

3. Administratori baze podataka

- instaliraju i nadograđuju SUBP
- odgovorni su za autorizaciju pristupa bazi podataka
- organiziraju, nadziru i optimiziraju korištenje baze

4. Korisnici

- Pristupaju bazi tako da postavljaju upite, mijenjaju podatke i izrađuju izvještaje

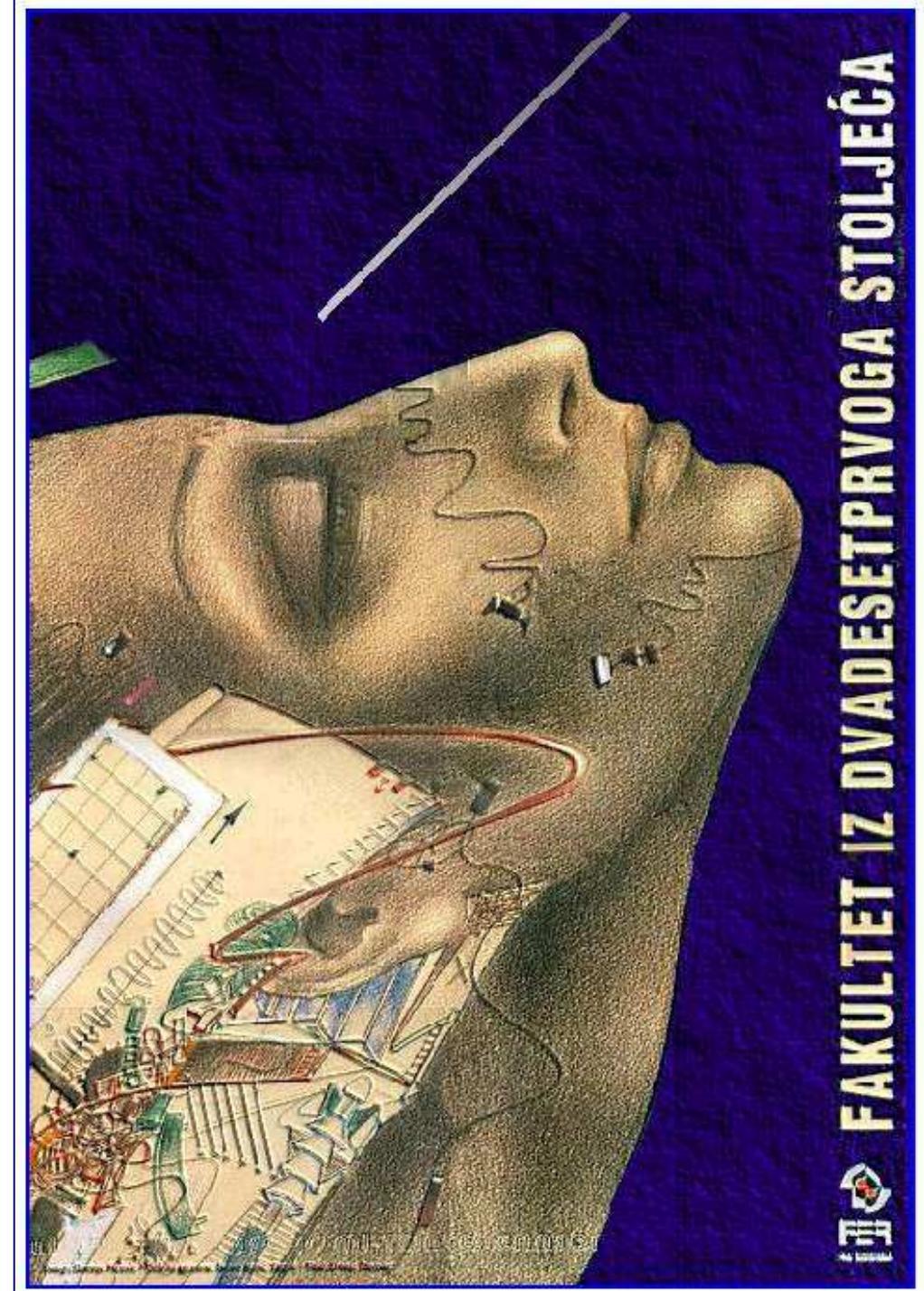
Razlikujemo nekoliko skupina korisnika:

- korisnici koji povremeno pristupaju bazi koristeći upitni jezik
- korisnici koji često koriste bazu postavljajući standardne upite i radeći standardne promjene koristeći programirana sučelja (npr. službenici u banci, turističkim agencijama...)
- sofisticirani korisnici koji su dobro upoznati s bazom podataka i koriste je na složeniji način (npr. inženjeri, znanstvenici, poslovni analitičari)

Baze podataka

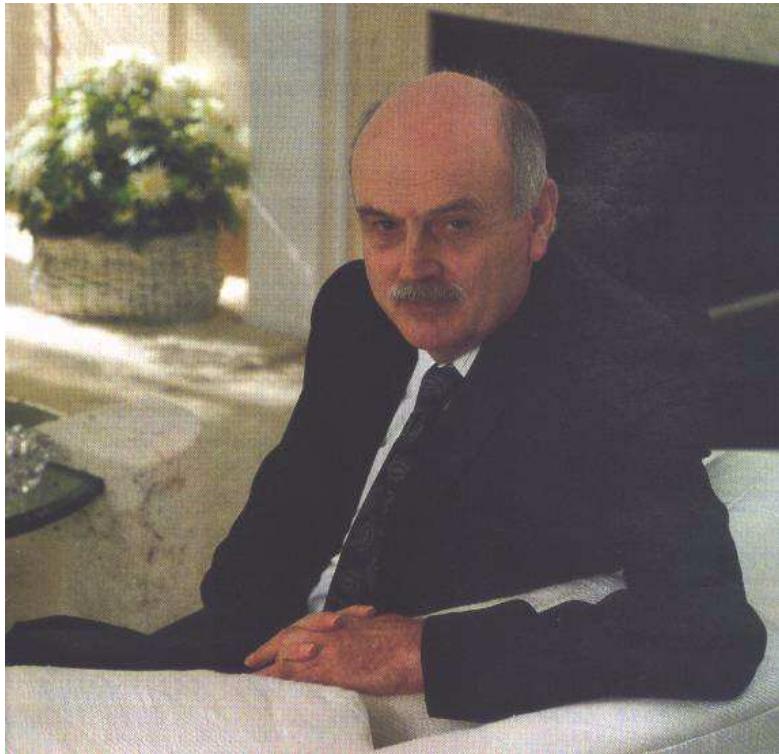
Predavanja
ožujak 2014.

2. Relacijski model podataka



Relacijski model podataka

- E. F. Codd: "A Relational Model of Data for Large Shared Data Banks", Comm. ACM 13, No. 6, June 1970.



Dr. Edgar Frank Codd (1923-2003)

Ciljevi relacijskog modela podataka:

- osigurati visoki stupanj nezavisnosti podataka
- postaviti temelje za rješavanje problema semantike, konzistentnosti i redundancije podataka (**normalizacija**)
- omogućiti razvoj DML jezika temeljenih na operacijama nad skupovima

Relacijski model podataka

- Važni projekti u ranim 70-tim: jezik ISBL temeljen na relacijskoj algebri, jezici SQUARE i SEQUEL (DBMS System R) temeljeni na relacijskoj algebri i predikatnom računu te Query-By-Example temeljen na predikatnom računu nad domenama
 - razvojem prototipova dokazuje se praktična upotrebljivost relacijskog modela
 - postavljaju se temelji za rješavanje problema implementacije u područjima upravljanja transakcijama, paralelnog pristupa, obnove, optimizacije upita, sigurnosti i konzistentnosti podataka
- Projekti su potaknuli:
 - razvoj strukturiranog upitnog jezika (SQL)
 - razvoj komercijalnih **relacijskih** sustava za upravljanje bazama podataka (RDBMS)
 - Ingres, Oracle, IBM DB2, Informix, ...
 - danas: u upotrebi je nekoliko stotina različitih RDBMS sustava

Relacijski model podataka

- objekti u relacijskom modelu podataka su **RELACIJE**

mjesto		
pbr	nazMjesto	sifZup
42000	Varaždin	7
51000	Rijeka	2
52100	Pula	4
51300	Delnice	2
42230	Ludbreg	7

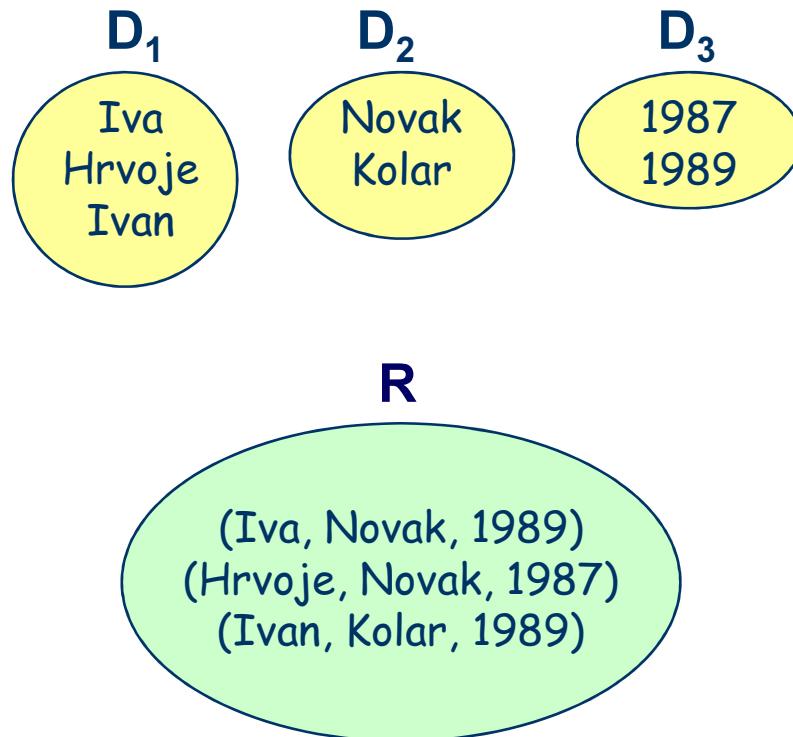
zupanija	
sifZup	nazZup
2	Primorsko-goranska
7	Varaždinska
4	Istarska

- neformalna definicija:** relacija je imenovana dvodimenzionalna tablica
 - atribut je imenovani stupac relacije
 - domena je skup dopuštenih vrijednosti atributa
 - nad istom domenom može biti definiran jedan ili više atributa
 - n-torka (*tuple*) je redak relacije

Matematička relacija

- Relacija R definirana nad skupovima D_1, D_2, \dots, D_n je podskup Kartezijevog produkta skupova D_1, D_2, \dots, D_n

$$R \subseteq D_1 \times D_2 \times \dots \times D_n$$



$$D_1 \times D_2 \times D_3$$

(Iva, Novak, 1987)
(Iva, Novak, 1989)
(Iva, Kolar, 1987)
(Iva, Kolar, 1989)
(Hrvoje, Novak, 1987)
(Hrvoje, Novak, 1989)
(Hrvoje, Kolar, 1987)
(Hrvoje, Kolar, 1989)
(Ivan, Novak, 1987)
(Ivan, Novak, 1989)
(Ivan, Kolar, 1987)
(Ivan, Kolar, 1989)

Relacijska shema (formalna definicija)

- Neka su zadani atributi A_1, A_2, \dots, A_n . Relacijska shema R (intenzija) je **imenovani skup atributa**

$$R = \{ A_1, A_2, \dots, A_n \}$$

- radi pojednostavljenja, koristit će se i sljedeća notacija:

$$R = A_1 A_2 \dots A_n$$

- uočite: poredak atributa u shemi relacije je nebitan

$$R = \{ A_1, A_2, A_3 \} \equiv \{ A_3, A_1, A_2 \}$$

- Primjer: relacijska shema MJESTO

$$MJESTO = \{ pbr, nazMjesto, sifZup \}$$

Relacijska shema (primjer)

- Zadani su atributi pbr, nazMjesto, sifZup
- Relacijska shema
 $MJESTO = \{ pbr, nazMjesto, sifZup \}$
identična je relacijskoj shemi
 $MJESTO = \{ sifZup, pbr, nazMjesto \}$

n-torka (formalna definicija)

- Neka je $R = \{ A_1, A_2, \dots, A_n \}$ relacijska shema; neka su D_1, D_2, \dots, D_n domene atributa A_1, A_2, \dots, A_n ; **n-torka t** definirana na relacijskoj shemi R je skup parova oblika *atribut:vrijednostAtributa*
$$t = \{ A_1:v_1, A_2:v_2, \dots, A_n:v_n \},$$
pri čemu je $v_1 \in D_1, v_2 \in D_2, \dots, v_n \in D_n$
- Uočite: poredak elemenata n-torke nije bitan
$$\{ A_1:v_1, A_2:v_2, A_3:v_3 \} \equiv \{ A_3:v_3, A_1:v_1, A_2:v_2 \}$$
- Ponekad će se koristiti pojednostavljena notacija: prepostavi li se da poredak vrijednosti atributa odgovara "poretku atributa" u relacijskoj shemi, n-torka se može prikazati na sljedeći način:
$$t = < v_1, v_2, \dots, v_n >$$

n-torka (primjer)

- Zadana je relacijska shema OSOBA = { matBr, ime, prez }, pri čemu su domene atributa:

$\text{dom}(\text{matBr}) = \{1234, 1235, 1236, 1237\}$

$\text{dom}(\text{ime}) = \{\text{Iva}, \text{Hrvoje}, \text{Ivan}\}$

$\text{dom}(\text{prez}) = \{\text{Novak}, \text{Kolar}\}$

$t_1 = \{\text{matBr:1234, ime:Iva, prez:Novak}\}$

$t_2 = \{\text{matBr:1236, ime:Hrvoje, prez:Novak}\}$

$t_3 = \{\text{matBr:1237, ime:Ivan, prez:Kolar}\}$

- n-torka t_1 se jednako ispravno može napisati na sljedeći način (poredak elemenata n-torke je nebitan)

$t_1 = \{\text{ime:Iva, prez:Novak, matBr:1234}\}$

- pojednostavljena notacija:

$t_1 = <1234, \text{Iva}, \text{Novak}>$

Relacija (formalna definicija)

- Neka je $R = \{ A_1, A_2, \dots, A_n \}$ relacijska shema; neka su D_1, D_2, \dots, D_n domene atributa A_1, A_2, \dots, A_n ; **relacija r** (instanca relacije) definirana na shemi relacije R je **skup n-torki** koje su definirane na relacijskoj shemi R
- kad se želi naglasiti da je relacija r definirana na shemi relacije R, kao oznaka za relaciju koristi se $r(R)$ ili $r(\{ A_1, A_2, \dots, A_n \})$ ili $r(A_1 A_2 \dots A_n)$
- relacijska shema R: mijenja se relativno rijetko
- instanca relacije r: predstavlja trenutnu vrijednost relacije i često se mijenja (pri unosu/brisanju/izmjeni podataka)

Relacija (primjer)

- Zadana je relacijska shema STUDENT = { matBr, prez, slika }, pri čemu su domene atributa:

- dom (matBr) = { 100, 102, 107, 111, 135 }
- dom (prez) = { Novak, Kolar, Horvat, Ban }
- dom (slika) = {  ,  ,  }

$\text{student(STUDENT)} = \{ \{ \text{matBr:102, prez:Novak, slika: } \text{  } \},$
 $\quad \quad \quad \{ \text{matBr:135, prez:Ban, slika: } \text{  } \} \}$

- IDENTIČNA RELACIJA (poredak n-torki i članova n-torki je nebitan):

$\text{student(STUDENT)} = \{ \{ \text{prez:Ban, matBr:135, slika: } \text{  } \},$
 $\quad \quad \quad \{ \text{slika: } \text{  }, \text{ matBr:102, prez:Novak } \} \}$

Svojstva relacija

- relacija posjeduje ime koje je jedinstveno unutar sheme baze podataka
- atributi unutar relacije imaju jedinstvena imena (**zašto?**)
- jedan atribut može poprimiti vrijednost iz samo jedne domene
- u jednoj relaciji ne postoje dvije jednake n-torce (**zašto?**)
- redoslijed atributa unutar relacije je nebitan (**zašto?**)
- redoslijed n-torki unutar relacije je nebitan (**zašto?**)



The diagram illustrates two tables representing the same data, separated by an equals sign (=). Both tables have a header row labeled "zupanija".

zupanija	
sifZup	nazZup
2	Primorsko-goranska
7	Varaždinska
4	Istarska

zupanija	
nazZup	sifZup
Varaždinska	7
Istarska	4
Primorsko-goranska	2

Relacija (primjer)

student(STUDENT) = { { prez:Ban, matBr:135, slika:  { slika: 

- pojednostavljenje prikaza relacije (vizualizacija relacije tablicom)

student (STUDENT)		
matBr	prez	slika
102	Novak	
135	Ban	

A-vrijednost n-torke, X-vrijednost n-torke

- Oznaka $t(A)$ predstavlja vrijednost koju atribut A poprima u n-torki t . $t(A)$ se naziva A-vrijednost n-torke t .
- Primjer:

$t = \{ \text{matBr:102, prez:Novak, slika: } \text{
 $t(\text{pres}) = \text{Novak}$$

- Neka je $X \subseteq R$. n-torka t reducirana na skup atributa X naziva se X-vrijednost n-torke t i označava s $t(X)$
- Primjer:

$t = \{ \text{matBr:102, prez:Novak, slika: } \text{
 $X = \{ \text{matBr, prez} \} \quad X \subseteq R$
 $t(X) = t(\{ \text{matBr, prez} \}) = \{ \text{matBr:102, prez:Novak} \}$$

Stupanj i kardinalnost relacije

- stupanj relacije: broj atributa (stupaca) - *degree*
- kardinalnost relacije: broj n-torki (redaka) - *cardinality*

mjesto		
pbr	nazMjesto	sifZup
42000	Varaždin	7
51000	Rijeka	2
52100	Pula	4
51300	Delnice	2
42230	Ludbreg	7

kardinalnost = 5

stupanj = 3

Oznake:

$$\deg(mjesto) = 3$$

$$\text{card}(mjesto) = 5$$

Shema i instanca baze podataka

- Shema baze podataka je **skup relacijskih shema**

$$\mathcal{R} = \{ R_1, R_2, \dots, R_n \}$$

- očito, relacijske sheme u jednoj shemi baze podataka moraju imati različita imena

- Instanca baze podataka definirana na shemi baze podataka $\mathcal{R} = \{ R_1, R_2, \dots, R_n \}$ je **skup instanci relacija**

$$r = \{ r_1(R_1), r_2(R_2), \dots, r_n(R_n) \}$$

- shema baze podataka se relativno rijetko mijenja
- instanca baze podataka se često mijenja

Operacije u relacijskom modelu podataka

Relacijska algebra

- Operacije relacijske algebre su:

\cup	unija (<i>union</i>)
\cap	presjek (<i>intersection</i>)
\setminus	razlika (<i>set difference</i>)
\div	dijeljenje (<i>division</i>)
π	projekcija (<i>projection</i>)
σ	selekcija (<i>selection</i>)
\times	Kartezijev produkt (<i>Cartesian product</i>)
ρ	preimenovanje (<i>renaming</i>)
$\triangleright \triangleleft$	spajanje (<i>join</i>)
	agregacija, grupiranje

$$\text{Primjer: } r_4 = \sigma_{A=x \wedge B=y} (r_1 \cup (r_2 \cap r_3))$$

- Karakteristika relacijske algebre - proceduralnost - navodi se redoslijed operacija koje se provode nad relacijama

Predikatni račun

- Operacije se specificiraju navođenjem predikata

$$r = \{ t \mid F(t) \}$$

- t je varijabla koja predstavlja:

- n-torce - n-torski račun
 - rezultat r je skup n-torki t za koje je vrijednost predikata F istina
 - domene - domenski račun
 - rezultat je skup domena t za koje je vrijednost predikata F istina

- Primjer:

$$r_4 = \{ t \mid (r_1(t) \vee ((r_2(t) \wedge r_3(t))) \wedge t(A)=x \wedge t(B)=y \}$$

- Predikatni račun je neproceduralan

- ne navodi se redoslijed operacija
 - navode se predikati koje n-torce (domene) moraju zadovoljavati

SQL

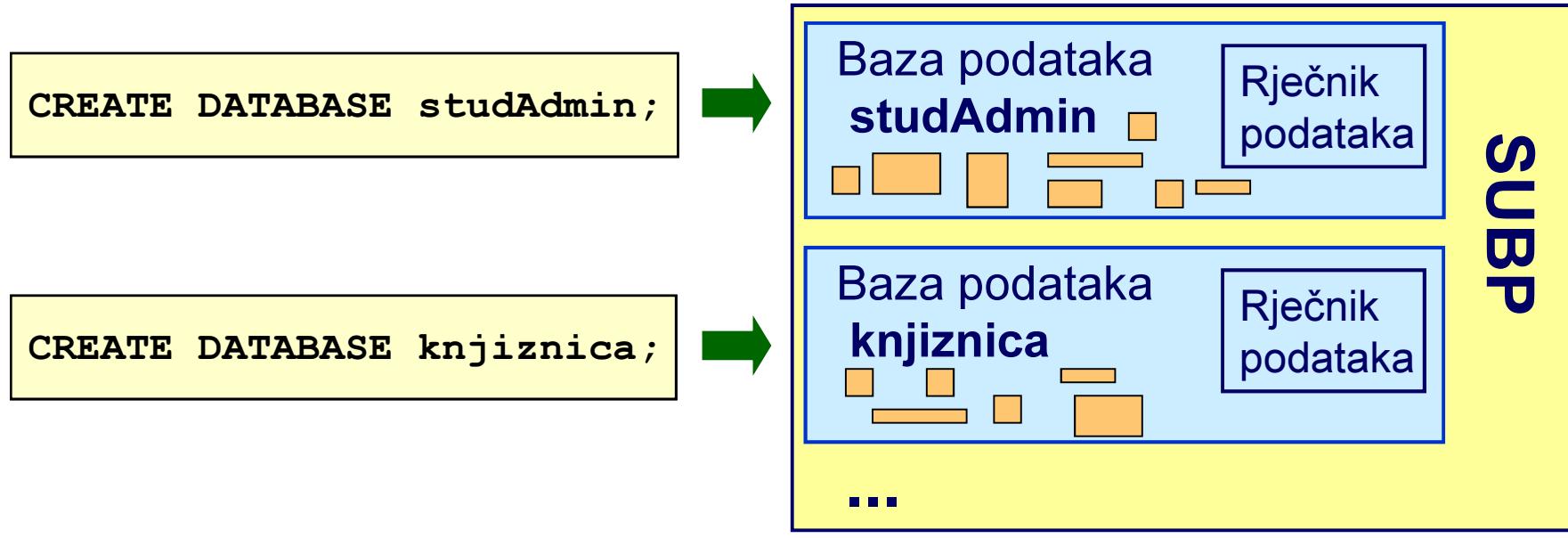
Kratki pogled

SQL - Kratki pogled

- SQL (*Structured Query Language*) je temeljen na relacijskom modelu podataka.
- nastao je na temelju jezika SEQUEL
- temelji se na predikatnom računu i relacijskoj algebri
- proglašen standardnim jezikom za relacijske sustave
- objekti u SQL-u su tablice, a ne (formalno definirane) relacije
 - poređak atributa (stupaca) u nekim je slučajevima značajan
 - u tablici ili rezultatu operacija nad tablicama moguća je pojava dvije ili više istih n-torki
 - ipak, postoje načini kako se to može spriječiti

SQL - Kratki pogled

- kreiranje nove instance baze podataka (kreiranje baze podataka)
 - jedan SUBP može istovremeno upravljati s više baza podataka



- Rječnik podataka sadrži opise relacijskih shema, integritetskih ograničenja, ...

`DROP DATABASE knjiznica;`



SQL - Kratki pogled

- opisivanje relacijske sheme (kreiranje relacije)
 - kreira praznu relaciju
 - ujedno je moguće definirati i integritetska ograničenja

```
CREATE TABLE mjesto (
    pbr          INTEGER
, nazMjesto   CHAR (30)
, sifZup      SMALLINT
);
```



mjesto		
pbr	nazMjesto	sifZup

```
DROP TABLE mjesto;
```



SQL - Kratki pogled

- upisivanje novih n-torki u relaciju

mjesto		
pbr	nazMjesto	sifZup

```
INSERT INTO mjesto  
VALUES (42000, 'Varaždin', 7);  
  
INSERT INTO mjesto  
VALUES (52100, 'Pula', 4);  
  
INSERT INTO mjesto  
VALUES (42230, 'Ludbreg', 7);
```



mjesto		
pbr	nazMjesto	sifZup
42230	Ludbreg	7
42000	Varaždin	7
52100	Pula	4

Treba li poredak n-torki u relaciji biti
u skladu s redoslijedom upisa?



SQL - Kratki pogled

- dohvati podataka iz relacije

mjesto

pbr	nazMjesto	sifZup
42230	Ludbreg	7
42000	Varaždin	7
52100	Pula	4

- dohvati podataka o mjestima čija šifra županije ima vrijednost 7

```
SELECT * FROM mjesto  
WHERE sifZup = 7;
```



pbr	nazMjesto	sifZup
42000	Varaždin	7
42230	Ludbreg	7

SQL - Kratki pogled

- izmjena vrijednosti atributa u relaciji

mjesto

pbr	nazMjesto	sifZup
42230	Ludbreg	7
42000	Varaždin	7
52100	Pula	4

- naziv mjesta s poštanskim brojem 42000 promijeniti u VARAŽDIN

```
UPDATE mjesto  
SET nazMjesto = 'VARAŽDIN'  
WHERE pbr = 42000;
```



mjesto

pbr	nazMjesto	sifZup
42230	Ludbreg	7
42000	VARAŽDIN	7
52100	Pula	4

SQL - Kratki pogled

- brisanje n-torki iz relacije

mjesto

pbr	nazMjesto	sifZup
42230	Ludbreg	7
42000	VARAŽDIN	7
52100	Pula	4

- obrisati mjesta za koje šifra županije ima vrijednost 7

```
DELETE FROM mjesto  
WHERE sifZup = 7;
```



mjesto

pbr	nazMjesto	sifZup
52100	Pula	4

Relacijska algebra

Relacijska algebra

- **Unarne operacije**
 - projekcija, selekcija, preimenovanje
 - agregacija, grupiranje
- **Binarne operacije**
 - skupovske operacije (*set operations*)
 - temelje se na relacijama kao skupovima n-torki
 - unija, presjek, razlika
 - ostale binarne operacije
 - Kartezijev produkt, dijeljenje, spajanje

Relacijska algebra

- obavljanje operacije ne utječe na operande, npr.

$$r_3 = r_1 \cup r_2$$

- obavljanjem prethodne operacije nastaje nova relacija r_3 , a relacije r_1 i r_2 se pri tome ne mijenjaju
- operandi su relacije, a rezultat obavljanja operacije je uvijek relacija. To znači:
 - skup relacija je **zatvoren** s obzirom na operacije relacijske algebre
 - ta činjenica omogućava da se rezultat jedne operacije upotrijebi kao operand u sljedećoj operaciji, što omogućava formiranje složenih izraza

$$r_5 = (r_1 \cup r_2) \times (r_3 \triangleright\triangleleft r_4)$$

Unijska kompatibilnost

- Dvije relacije su unijski kompatibilne ukoliko vrijedi:
 - relacije su istog stupnja
 - i
 - korespondentni atributi su definirani nad istim domenama

polozioMatem		
matBr	ime	prez
12345	Ivo	Kolar
13254	Ana	Horvat

polozioProgr		
mbr	prezSt	imeSt
92632	Ban	Jura
67234	Novak	Iva

- relacije su istog stupnja
- dom (matBr) = dom(mbr)
- dom (ime) = dom(imeSt)
- dom (prez) = dom(prezSt)

→ relacije su unijski kompatibilne

- kod ocjene jesu li relacije unijski kompatibilne
 - poredak atributa nije bitan
 - imena atributa nisu bitna

Unija kompatibilnosti

- dvije relacije koje imaju jednak broj atributa i jednaka imena atributa ne moraju ujedno biti uniji kompatibilne

zrakoplov		pecivo	
oznaka	naziv	oznaka	naziv
B-747	Boeing 747	ZE	Žemlja
A-360	Airbus 360	PR	Perec

- relacije su istog stupnja
- $\text{dom}(\text{zrakoplov.oznaka}) \neq \text{dom}(\text{pecivo.oznaka})$
- $\text{dom}(\text{zrakoplov.naziv}) \neq \text{dom}(\text{pecivo.naziv})$

→ relacije NISU uniji kompatibilne

- notacija *imeRelacije.imeAtributa* se često koristi kada je potrebno razlikovati istoimene attribute različitih relacija

Skupovske operacije: unija, presjek, razlika

- Skupovske operacije (unija, presjek, razlika) mogu se obavljati isključivo nad UNIJSKI KOMPATIBILNIM relacijama

Unija

- Rezultat operacije $r_1 \cup r_2$ je relacija čije su n-torce elementi relacije r_1 ili elementi relacije r_2 ili elementi obje relacije.
 - n-torce koje su elementi obje relacije u rezultatu se pojavljuju samo jednom (jer relacija je SKUP n-torki)

polozioMatem

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat

polozioProgr

mbr	ime	prez
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat

polozioBaremJedan =

polozioMatem \cup polozioProgr

polozioBaremJedan

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban
105	Rudi	Kolar
107	Jura	Horvat

studenti koji su položili ili
Matematiku ili
Programiranje ili
oba predmeta

$$r_1 \cup r_2 \equiv r_2 \cup r_1$$

Presjek

- Rezultat operacije $r_1 \cap r_2$ je relacija čije su n-torce elementi relacije r_1 i elementi relacije r_2

polozioMatem

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat

polozioProgr

mbr	ime	prez
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat

polozioOba =

polozioMatem \cap polozioProgr

polozioOba

mbr	ime	prez
102	Ana	Novak
107	Jura	Horvat

studenti koji su
položili i Matematiku
i Programiranje

$$r_1 \cap r_2 \equiv r_2 \cap r_1$$

Razlika

- Rezultat operacije $r_1 \setminus r_2$ je relacija čije su n-torce elementi relacije r_1 i nisu elementi relacije r_2

polozioMatem		
mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat

polozioProgr		
mbr	ime	prez
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat

**polozioSamoMatem =
polozioMatem \ polozioProgr**

polozioSamoMatem		
mbr	ime	prez
100	Ivan	Kolar
103	Tea	Ban

studenti koji su
položili Matematiku,
ali **nisu** položili
Programiranje

$$r_1 \setminus r_2 \neq r_2 \setminus r_1$$

polozioSamoProgr = polozioProgr \ polozioMatem

polozioSamoProgr		
mbr	ime	prez
105	Rudi	Kolar

ŠTO AKO SE IMENA KORESPONDENTNIH ATRIBUTA RAZLIKUJU

- Unija, presjek, razlika: u slučajevima kada su relacije unijski kompatibilne, ali se u relacijama koriste različita imena korespondentnih atributa, primjenjuje se sljedeći dogovor (konvencija): **kao imena atributa u rezultantnoj relaciji koriste se imena atributa prvog operanda**

polozioMatem		
mbr	imeSt	prezSt
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat

polozioProgr		
mbr	ime	prez
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat

$\text{polozioOba} = \text{polozioMatem} \cap \text{polozioProgr}$

polozioOba		
mbr	imeSt	prezSt
102	Ana	Novak
107	Jura	Horvat

Zadaci za vježbu

- zadane su unijski kompatibilne relacije
 - m (mbr ime prez) → studenti koji su položili Matematiku
 - d (mbr ime prez) → studenti koji su položili Dig. logiku
 - p (mbr ime prez) → studenti koji su položili Programiranje
- napisati izraze relacijske algebre koji određuju relacije koje sadrže studente (točnije rečeno n-torke):
 - a) koji su položili sva tri predmeta
 - b) koji su položili ili Matematiku ili Digitalnu logiku, ali ne oba predmeta (*ekskluzivni ili*)
 - c) koji su položili točno jedan (bilo koji) od ta tri predmeta
 - d) koji su položili bilo koja dva predmeta (ali nisu položili treći)

Dijeljenje (*division*)

- Zadane su relacije $r(R)$ i $s(S)$. Neka je $S \subseteq R$. Rezultat operacije $r \div s$ je relacija sa shemom $P = R \setminus S$. n -torka $t_r(P)$ se pojavljuje u rezultatu ako i samo ako za n -torku $t_r \in r$ vrijedi da se $t_r(P)$ u relaciji r pojavljuje u kombinaciji sa svakom n -torkom $t_s \in s$

polozen	
mbrSt	sifPred
100	1
100	2
101	1
101	2
101	3
102	2
102	3
103	1
103	2
103	3
104	3

predmet	
sifPred	
1	
2	
3	

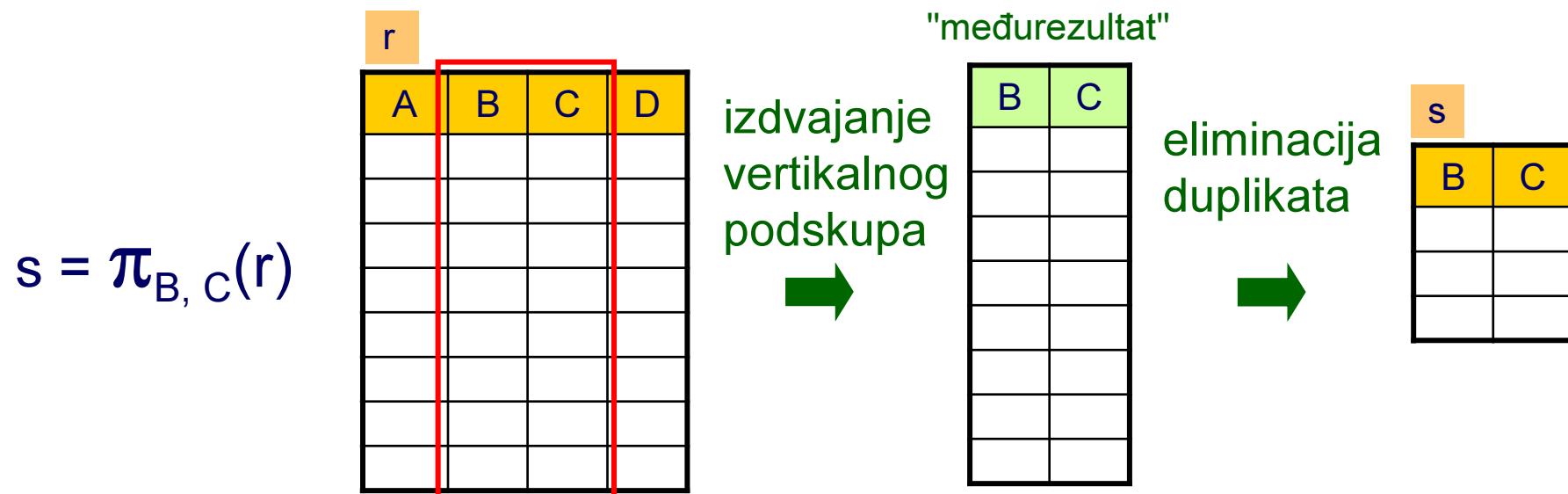
studenti koji su položili sve predmete
sa šiframa u relaciji predmet

poloziliSve = **polozen** \div **predmet**

poloziliSve	
mbrSt	
101	
103	

Projekcija

- Zadana je relacija $r(R)$. Neka je skup atributa $\{ A_1, A_2, \dots, A_k \} \subseteq R$
- Obavljanjem operacije $\pi_{A_1, A_2, \dots, A_k}(r)$ dobiva se relacija s sa shemom $\{ A_1, A_2, \dots, A_k \}$ koja sadrži vertikalni podskup relacije r
 - $\deg(s) = k$
 - $\text{card}(s) \leq \text{card}(r)$ (jer se eliminiraju duplikati)



Projekcija (primjer)

Relacija nastup: u kojim gradovima nastup
su nastupali koji tenori kojeg datuma

Traži se: u kojim gradovima su nastupali
koji tenori

$$\text{tenorGrad} = \pi_{\text{tenor}, \text{grad}}(\text{nastup})$$

tenor	grad	datum
P. Domingo	London	15.2.1976
P. Domingo	New York	27.3.1981
P. Domingo	London	11.4.1987
J. Carreras	New York	11.4.1987
L. Pavarotti	Sydney	22.6.1992
L. Pavarotti	London	15.2.1976
L. Pavarotti	Sydney	19.1.1993
L. Pavarotti	London	14.7.1993

→ "medrezultat"

tenor	grad
P. Domingo	London
P. Domingo	New York
P. Domingo	London
J. Carreras	New York
L. Pavarotti	Sydney
L. Pavarotti	London
L. Pavarotti	Sydney
L. Pavarotti	London

→ tenorGrad

tenor	grad
P. Domingo	London
P. Domingo	New York
J. Carreras	New York
L. Pavarotti	Sydney
L. Pavarotti	London

SQL - Lista za selekciju

mjesto	pbr	nazMjesto	sifZup
42000	Varaždin	7	
52100	Pula	4	

- **SELECT *SELECT List* FROM *table***
- ***SELECT List*** je lista za selekciju: dio SELECT naredbe koji određuje koji će se "stupci" pojaviti u rezultatu

`SELECT * FROM mjesto;`

\equiv

`SELECT mjesto.pbr
, mjesto.nazMjesto
, mjesto.sifZup
FROM mjesto;`

- uz ime atributa može se navesti ime relacije (radi izbjegavanja dvosmislenosti u slučajevima kada se podaci dohvaćaju istovremeno iz više relacija čija se imena atributa podudaraju)
imeRelacije.imeAtributa
- u slučajevima kada takva dvosmislenost ne postoji, ime relacije se može (ali ne mora) ispuštiti

SQL - Lista za selekciju

mjesto	pbr	nazMjesto	sifZup
	42000	Varaždin	7
	52100	Pula	4

zupanija	sifZupanija	nazZup
	7	Varaždinska
	4	Istarska

- u listi za selekciju se ne moraju navesti svi atributi relacije navedene u FROM dijelu naredbe:

```
SELECT nazMjesto  
      , pbr  
   FROM mjesto;
```



nazMjesto	pbr
Varaždin	42000
Pula	52100

- u listi za selekciju se mogu navesti samo oni atributi koji se nalaze u dosegu SELECT naredbe, tj. atributi relacije koja je navedena u FROM dijelu naredbe:

```
SELECT nazMjesto  
      , pbr  
      , nazZup  
   FROM mjesto;
```

Neispravna naredba

SQL - Projekcija

- za ispravno obavljanje projekcije nije dovoljno u listi za selekciju samo navesti imena atributa prema kojima se obavlja projekcija:

- primjer koji ujedno pokazuje kako rezultat SQL naredbe ne mora uvijek biti relacija



```
SELECT tenor  
      , grad  
  FROM nastup;
```

Neispravna projekcija

tenor	grad
P. Domingo	London
P. Domingo	New York
P. Domingo	London
J. Carreras	New York
L. Pavarotti	Sydney
L. Pavarotti	London
L. Pavarotti	Sydney
L. Pavarotti	London

$\pi_{\text{tenor}, \text{grad}}(\text{nastup})$

```
SELECT DISTINCT tenor  
      , grad  
  FROM nastup;
```

Ispravna projekcija

tenor	grad
P. Domingo	London
P. Domingo	New York
J. Carreras	New York
L. Pavarotti	Sydney
L. Pavarotti	London

Selekcija

- Zadana je relacija $r(R)$. Neka je F predikat (formula, uvjet, condition) koji se sastoji od operanada i operatora
 - operandi su:
 - imena atributa iz R
 - konstante
 - operatori su:
 - operatori usporedbe: $<$ \leq $=$ \neq $>$ \geq
 - logički operatori: \wedge \vee \neg
- Obavljanjem operacije $\sigma_F(r)$ dobiva se relacija sa shemom R koja sadrži one n-torce relacije r za koje je vrijednost predikata F istina (true)

Selekcija (primjer)

student	matBr	ime	prez	postBr
	100	Ivan	Kolar	52000
	102	Ana	Horvat	10000
	105	Jura	Novak	21000
	107	Ana	Ban	51000

rezultat = $\sigma_{ime = 'Ana' \vee postBr > 31000}$ (student)

- Za svaku pojedinu n-torku relacije:
 - vrijednosti atributa uvrštavaju se u predikat - uvrštavanjem vrijednosti u predikat dobiva se **sud**
 - onda i samo onda kada je vrijednost dobivenog suda istina (*true*), n-torka se pojavljuje u rezultatu selekcije

→ 'Ivan' = 'Ana' \vee 52000 > 31000 → *true*

→ 'Ana' = 'Ana' \vee 10000 > 31000 → *true*

'Jura' = 'Ana' \vee 21000 > 31000 → *false*

'Ana' = 'Ana' \vee 51000 > 31000 → *true*

rezultat	matBr	ime	prez	postBr
	100	Ivan	Kolar	52000
	102	Ana	Horvat	10000
	107	Ana	Ban	51000

SQL - Selekcija

- **SELECT *SELECT List* FROM *table***
[WHERE *Condition*]
- Uvjet (*Condition*) se sastoji od operanada i operatora
 - operandi su:
 - imena atributa iz relacije *table*
 - konstante
 - operatori su:
 - operatori usporedbe: < <= = <> > >=
 - logički operatori: AND OR NOT
- Vrijednosti svake n-torke iz relacije *table* se uvrštavaju u *Condition* (a to je u stvari predikat). Ako je dobiveni sud istinit (*true*), n-torka se pojavljuje u rezultatu.

SQL - Selekcija

student

	matBr	ime	prez	postBr
100	Ivan	Kolar	52000	
102	Ana	Horvat	10000	
105	Jura	Novak	21000	
107	Ana	Ban	51000	

$$\sigma_{ime = 'Ana' \vee postBr > 31000} (student)$$

```
SELECT * FROM student
WHERE ime = 'Ana'
OR postBr > 31000;
```



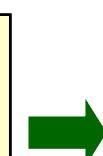
matBr	ime	prez	postBr
100	Ivan	Kolar	52000
102	Ana	Horvat	10000
107	Ana	Ban	51000

SQL - Projekcija i selekcija

student	matBr	ime	prez	postBr
	100	Ivan	Kolar	52000
	102	Ana	Horvat	10000
	105	Jura	Novak	21000
	107	Ana	Ban	51000

$$\pi_{\text{ime}}(\sigma_{\text{ime} = \text{'Ana'} \vee \text{postBr} > 31000}(\text{student}))$$

```
SELECT DISTINCT ime
  FROM student
 WHERE ime = 'Ana'
   OR postBr > 31000;
```



"međurezultat"

matBr	ime	prez	postBr
100	Ivan	Kolar	52000
102	Ana	Horvat	10000
107	Ana	Ban	51000



ime
Ivan
Ana

Kartezijev produkt

- Zadana je relacija $r(R)$ i relacija $s(S)$, pri čemu je $R \cap S = \emptyset$.
- Obavljanjem operacije $r \times s$ dobiva se relacija $p(P)$, $P = R \cup S$.
 n -torke relacije p se dobivaju spajanjem (ulančavanjem) svake n -torke iz relacije r sa svakom n -torkom iz relacije s
 - $\deg(p) = \deg(r) + \deg(s)$
 - $\text{card}(p) = \text{card}(r) \cdot \text{card}(s)$

Kartezijev produkt (primjer)

student		
mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban

predmet	
sifra	naziv
1	Programiranje
2	Matematika

upis = student \times predmet

upis				
mbr	ime	prez	sifra	naziv
100	Ivan	Kolar	1	Programiranje
100	Ivan	Kolar	2	Matematika
102	Ana	Novak	1	Programiranje
102	Ana	Novak	2	Matematika
103	Tea	Ban	1	Programiranje
103	Tea	Ban	2	Matematika

SQL - Kartezijev produkt

- **SELECT *SELECT List*
FROM *table* [, *table*]...
[WHERE *Condition*]**

- navede li se u FROM dijelu naredbe više od jedne relacije, obavlja se operacija Kartezijevog produkta navedenih relacija

student		
mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban

predmet	
sifra	naziv
1	Programiranje
2	Matematika

student × predmet

```
SELECT *
  FROM student, predmet;
```

```
SELECT student.* , predmet.*
  FROM student, predmet;
```

mbr	ime	prez	sifra	naziv
100	Ivan	Kolar	1	Programiranje
100	Ivan	Kolar	2	Matematika
102	Ana	Novak	1	Programiranje
102	Ana	Novak	2	Matematika
103	Tea	Ban	1	Programiranje
103	Tea	Ban	2	Matematika

SQL - Kartezijev produkt

- drugačija sintaksa:
- **SELECT *SELECT List***

FROM *table* [CROSS JOIN *table*] . . .
[WHERE *Condition*]

```
SELECT *
  FROM student CROSS JOIN predmet;
```

- Kartezijev produkt triju relacija:

```
SELECT *
  FROM r1 CROSS JOIN r2 CROSS JOIN r3;
```

Kartezijev produkt

- Što učiniti ukoliko je potrebno obaviti operaciju Kartezijevog produkta nad relacijama $r(R)$ i $s(S)$, u slučaju kada $R \cap S \neq \emptyset$

r	A	B
1	a	
2	b	
3	c	

s	B	C
c		α
d		β

	A	B	B	C
1	a		c	α
1	a		d	β
2	b		c	α
2	b		d	β
3	c		c	α
3	c		d	β



NIJE RELACIJA!

→ Potrebno je koristiti operaciju preimenovanja

Preimenovanje (relacije, atributa)

- Zadana je relacija $r(\{ A_1, A_2, \dots, A_n \})$
 - preimenovanje relacije:** operacijom preimenovanja $\rho_s(r)$ dobiva se relacija s koja ima jednaku shemu i sadržaj kao relacija r
 - preimenovanje relacije i atributa:** operacijom preimenovanja $\rho_{s(B_1, B_2, \dots, B_n)}(r)$ dobiva se relacija s čija shema umjesto atributa A_1, A_2, \dots, A_n sadrži attribute B_1, B_2, \dots, B_n , a sadržaj relacije s je jednak sadržaju relacije r

$$p = r \times \rho_{s(B_2, C)}(s)$$

r	A	B
1	a	
2	b	
3	c	

s	B	C
	c	α
	d	β

p	A	B	B2	C
1	a		c	α
1	a		d	β
2	b		c	α
2	b		d	β
3	c		c	α
3	c		d	β

SQL - Preimenovanje atributa

- ukoliko se drugačije ne navede, imena stupaca u rezultatu odgovaraju imenima atributa iz liste za selekciju
- implicitna imena stupaca rezultata se mogu promijeniti korištenjem operatora za preimenovanje **AS**

zupanija	sifZupanija	nazZup
7		Varaždinska
4		Istarska

```
SELECT sifZupanija AS sifraz  
      , nazZup AS nazZ  
FROM zupanija;
```

sifraZ	nazZ
7	Varaždinska
4	Istarska

- rezervirana riječ **AS** smije se ispuštiti

```
SELECT sifZupanija sifraz  
      , nazZup nazZ  
FROM zupanija;
```

SQL - Preimenovanje atributa

- Primjer u kojem je potrebno koristiti preimenovanje atributa
 - SQL naredba bi bila ispravna i bez preimenovanja, ali tada kao rezultat ne bismo dobili relaciju (jer bi u shemi rezultata postojala dva atributa istog imena)

r	A	B
1	a	
2	b	
3	c	

s	B	C
c		α
d		β

$r \times \rho_{s(B2, C)}(s)$

```
SELECT A, r.B, s.B AS B2, C  
FROM r, s;
```

A	B	B2	C
1	a	c	α
1	a	d	β
2	b	c	α
2	b	d	β
3	c	c	α
3	c	d	β

Spajanje uz uvjet ili θ - spajanje (θ - join)

- Zadane su relacije $r(R)$ i $s(S)$ pri čemu je $R \cap S = \emptyset$. Neka je F predikat oblika $r.A_i \theta s.B_j$, pri čemu je $A_i \in R$, $B_j \in S$, a θ je operator usporedbe iz skupa operatora $\{ <, \leq, =, \neq, >, \geq \}$
- Obavljanjem operacije $r \triangleright\!\!\! \triangleleft_F s$ dobiva se relacija koja sadrži n -torke iz $r \times s$ za koje je vrijednost predikata F istina (true), odnosno:

$$r \triangleright\!\!\! \triangleleft_F s = \Sigma_F(r \times s)$$

što možemo reći o stupnju i kardinalnosti rezultata?

- Umjesto jednostavnog predikata $r.A_i \theta s.B_j$, može se koristiti složeni predikat dobiven primjenom logičkih operatora nad jednostavnim predikatima oblika $r.A_i \theta s.B_j$
- Problem spajanja uz uvjet relacija $r(R)$ i $s(S)$ kod kojih je $R \cap S \neq \emptyset$, rješava se na jednak način kao kod Kartezijevog produkta (korištenjem operatora preimenovanja)

Spajanje uz uvjet (primjer)

linija	
let	udaljenost
CA-825	700
LH-412	4800
BA-722	15000
CA-311	13000

zrakoplov	
tip	dolet
B747	13000
A320	5400
DC-9	3100



mogućnost = linija \bowtie zrakoplov
dolet \geq udaljenost

mogućnost			
let	udaljenost	tip	dolet
CA-825	700	B747	13000
CA-825	700	A320	5400
CA-825	700	DC-9	3100
LH-412	4800	B747	13000
LH-412	4800	A320	5400
CA-311	13000	B747	13000



Linije i zrakoplovi koji na
tim linijama mogu letjeti

SQL - Spajanje uz uvjet

- Koristi se ekvivalencija

$$r \triangleright\triangleleft s = \sigma_F(r \times s)$$

linija $\triangleright\triangleleft$ zrakoplov
dolet \geq udaljenost

```
SELECT *
  FROM linija, zrakoplov
 WHERE dolet >= udaljenost;
```

Linije i zrakoplovi koji na
tim linijama mogu letjeti

linija

let	udaljenost
CA-825	700
LH-412	4800
BA-722	15000
CA-311	13000

zrakoplov

tip	dolet
B747	13000
A320	5400
DC-9	3100

Kartezijev produkt

Selekcija

let	udaljenost	tip	dolet
CA-825	700	B747	13000
CA-825	700	A320	5400
CA-825	700	DC-9	3100
LH-412	4800	B747	13000
LH-412	4800	A320	5400
CA-311	13000	B747	13000

SQL - Spajanje uz uvjet

- drugačija sintaksa:

- **SELECT *SELECT List***

```
FROM table [JOIN table ON joinCondition] . . .
[WHERE Condition]
```

```
SELECT *
FROM linija JOIN zrakoplov
    ON dolet >= udaljenost;
```

- Spajanje uz uvjet triju relacija:

```
SELECT *
FROM r1
    JOIN r2
        ON joinCondition
    JOIN r3
        ON joinCondition;
```

SQL - Spajanje uz uvjet i selekcija

- Kako pronaći linije i zrakoplove koji na tim linijama mogu letjeti, ali samo za one linije na kojima je udaljenost veća od 4000 km

$$\sigma_{\text{udaljenost} > 4000}(\text{linija} \bowtie \text{zrakoplov}) \\ \text{dolet} \geq \text{udaljenost}$$

```
SELECT *
FROM linija, zrakoplov
WHERE dolet >= udaljenost
AND udaljenost > 4000;
```

```
SELECT *
FROM linija
JOIN zrakoplov
ON dolet >= udaljenost
WHERE udaljenost > 4000;
```

let	udaljenost	tip	dolet
LH-412	4800	B747	13000
LH-412	4800	A320	5400
CA-311	13000	B747	13000

ili

SQL - Spajanje uz uvjet i projekcija

- Kako pronaći tipove zrakoplova koji se mogu iskoristiti za letove na postojećim linijama

$$\pi_{\text{tip}}(\text{linija} \bowtie \text{zrakoplov})$$

dolet \geq \text{udaljenost}

```
SELECT DISTINCT tip
  FROM linija, zrakoplov
 WHERE dolet >= udaljenost;
```

tip
B747
A320
DC-9

ili

```
SELECT DISTINCT tip
  FROM linija
    JOIN zrakoplov
      ON dolet >= udaljenost;
```

Spajanje s izjednačavanjem (*Equi-join*)

- Spajanje relacija s izjednačavanjem je poseban oblik spajanja uz uvjet u kojem se kao θ operator koristi isključivo operator jednakosti (=)

mjesto	pbr	nazMjesto	sifZup
42000	Varaždin	7	
52100	Pula	4	
42230	Ludbreg	7	

zupanija	sifZupanija	nazZup
	7	Varaždinska
	4	Istarska

$mjestouZupaniji = mjesto \bowtie zupanija$
 $sifZup = sifZupanija$

mjestouZupaniji	pbr	nazMjesto	sifZup	sifZupanija	nazZup
	42000	Varaždin	7	7	Varaždinska
	52100	Pula	4	4	Istarska
	42230	Ludbreg	7	7	Varaždinska

- Problem spajanja s izjednačavanjem relacija $r(R)$ i $s(S)$ kod kojih je $R \cap S \neq \emptyset$, rješava se na jednak način kao kod Kartezijskog produkta (korištenjem operatora preimenovanja)

SQL - Spajanje s izjednačavanjem

- Koristi se ekvivalencija

$$r \triangleright\!\!< s = \sigma_F(r \times s)$$

F

mjesto $\triangleright\!\!<$ zupanija
sifZup = sifZupanija

mjesto	pbr	nazMjesto	sifZup
	42000	Varaždin	7
	52100	Pula	4
	42230	Ludbreg	7

zupanija	sifZupanija	nazZup
	7	Varaždinska
	4	Istarska

```
SELECT *
  FROM mjesto, zupanija
 WHERE sifZup = sifZupanija;
```

ili

```
SELECT *
  FROM mjesto
 JOIN zupanija
    ON sifZup = sifZupanija;
```

SQL - Spajanje s izjednačavanjem

- U slučaju kada u relacijama postoje istoimeni atributi

mjesto	pbr	nazMjesto	sifZup
42000	Varaždin	7	
52100	Pula	4	
42230	Ludbreg	7	

zupanija	sifZup	nazZup
	7	Varaždinska
	4	Istarska

$$\text{mjesto} \underset{\text{sifZup} = \text{sifZup2}}{\triangleright\!\!\!<} \rho_{\text{zupanija}(\text{sifZup2}, \text{nazZup})} \text{zupanija}$$

Za razliku od relacijske algebre, u SQL-u nije nužno preimenovati atribut prije spajanja:

```
SELECT mjesto.*  
      , zupanija.sifZup AS sifZup2  
      , zupanija.nazZup  
FROM mjesto, zupanija  
WHERE mjesto.sifZup = zupanija.sifZup;
```

slično i u slučaju korištenja drugačije sintakse (ANSI join)

Prirodno spajanje (*Natural Join*)

- Prirodno spajanje obavlja se na temelju jednakih vrijednosti istoimenih atributa.
- Zadane su relacije $r(R)$ i $s(S)$. Neka je $R \cap S = \{ A_1, A_2, \dots, A_n \}$. Obavljanjem operacije $r \bowtie s$ dobiva se relacija sa shemom $R \cup S$ koja sadrži n-torce nastale spajanjem n-torki $t_r \in r$, $t_s \in s$, za koje vrijedi $t_r(A_1) = t_s(A_1) \wedge t_r(A_2) = t_s(A_2) \wedge \dots \wedge t_r(A_n) = t_s(A_n)$.

mjesto	pbr	nazMjesto	sifZup
	42000	Varaždin	7
	52100	Pula	4
	42230	Ludbreg	7

zupanija	sifZup	nazZup
	7	Varaždinska
	4	Istarska

$mjestouZupaniji = mjesto \bowtie zupanija$

mjestouZupaniji	pbr	nazMjesto	sifZup	nazZup
	42000	Varaždin	7	Varaždinska
	52100	Pula	4	Istarska
	42230	Ludbreg	7	Varaždinska

što možemo reći o stupnju rezultata?

Prirodno spajanje

- Rezultat prirodnog spajanja relacija $r(R)$ i $s(S)$ za koje vrijedi da je $j e R \cap S = \emptyset$ identičan je rezultatu obavljanja operacije Kartezijskog produkta $r \times s$

mjesto	pbr	nazMjesto	sifZup
	42000	Varaždin	7
	52100	Pula	4
	42230	Ludbreg	7

zupanija	sifZupanija	nazZup
	7	Varaždinska
	4	Istarska

$mjestouZupaniji = mjesto \triangleright\triangleleft zupanija$

mjestouZupaniji	pbr	nazMjesto	sifZup	sifZupanija	nazZup
	42000	Varaždin	7	7	Varaždinska
	42000	Varaždin	7	4	Istarska
	52100	Pula	4	7	Varaždinska
	52100	Pula	4	4	Istarska
	42230	Ludbreg	7	7	Varaždinska
	42230	Ludbreg	7	4	Istarska

SQL - Prirodno spajanje

- prirodno spajanje se razlikuje od spajanja s izjednačavanjem po tome što se istoimeni atributi iz dviju relacija izbacuju (tako da od svakog ostane samo po jedan)

mjesto	pbr	nazMjesto	sifZup
	42000	Varaždin	7
	52100	Pula	4
	42230	Ludbreg	7

zupanija	sifZup	nazZup
	7	Varaždinska
	4	Istarska

```
SELECT mjesto.* , zupanija.nazZup
  FROM mjesto, zupanija
 WHERE mjesto.sifZup = zupanija.sifZup;
```

	pbr	nazMjesto	sifZup	nazZup
	42000	Varaždin	7	Varaždinska
	52100	Pula	4	Istarska
	42230	Ludbreg	7	Varaždinska

SQL - Prirodno spajanje

- drugačija sintaksa:

```
SELECT mjesto.* , zupanija.nazZup  
FROM mjesto JOIN zupanija  
ON mjesto.sifZup = zupanija.sifZup;
```

Agregacija (aggregation)

ispit	mbrStud	akGod	nazPred	ocjena
	100	2005	Matematika	3
	101	2005	Matematika	5
	102	2005	Matematika	2
	103	2006	Matematika	3
	100	2004	Fizika	5
	101	2006	Fizika	5
	102	2006	Fizika	2
	100	2005	Vjerojatnost	4

- Kako izračunati prosjek ocjena na svim ispitima?

prosjek	prosjOcj
	3.625

Agregacija

- Zadana je relacija $r(R)$. Neka je atribut $A \in R$. Neka je \mathcal{AF} agregatna funkcija. Rezultat operacije agregacije $G_{\mathcal{AF}(A)}(r)$ je relacija stupnja 1 i kardinalnosti 1, pri čemu je vrijednost atributa određena primjenom funkcije \mathcal{AF} nad vrijednostima atributa A u svim n-torkama relacije r . Funkcija \mathcal{AF} može biti jedna od:
 - COUNT određuje broj pojava (broji sve, eventualni duplikati se također broje)
 - SUM izračunava sumu vrijednosti
 - AVG izračunava aritmetičku sredinu vrijednosti
 - MIN izračunava najmanju vrijednost
 - MAX izračunava najveću vrijednost
- naziv rezultantne relacije i atributa nije definiran operacijom, stoga se najčešće koristi u kombinaciji s operacijom preimenovanja
- također se koriste agregatne funkcije
 - COUNT-DISTINCT, SUM-DISTINCT, AVG-DISTINCT

Agregacija

ispit	mbrStud	akGod	nazPred	ocjena
	100	2005	Matematika	3
	101	2005	Matematika	5
	102	2005	Matematika	2
	103	2006	Matematika	3
	100	2004	Fizika	5
	101	2006	Fizika	5
	102	2006	Fizika	2
	100	2005	Vjerojatnost	4

- Prosjek ocjena na svim ispitima (rješenje):

$$\rho_{\text{projek}(\text{projOcj})}(G_{\text{AVG}(\text{ocjena})}(\text{ispit}))$$

projek	projOcj
	3.625

```
SELECT AVG(ocjena) AS projOcj  
FROM ispit;
```

projOcj
3.625

Agregacija (primjeri ostalih agregatnih funkcija)

osoba		
sifra	tezina	visina
101	62	170
103	94	186
105	74	181
107	62	165

$$\rho_{\text{rez1(broj1)}}(G_{\text{COUNT(sifra)}}(\text{osoba}))$$

rez1	broj1
	4

$$\rho_{\text{rez2(broj2)}}(G_{\text{SUM(tezina)}}(\text{osoba}))$$

rez2	broj2
	292

$$\rho_{\text{rez3(broj3)}}(G_{\text{AVG(visina)}}(\text{osoba}))$$

rez3	broj3
	175.5

$$\rho_{\text{rez4(broj4)}}(G_{\text{MAX(visina)}}(\text{osoba}))$$

rez4	broj4
	186

$$\rho_{\text{rez5(broj5)}}(G_{\text{MIN(tezina)}}(\text{osoba}))$$

rez5	broj5
	62

Moguće je odjednom izračunati više agregatnih vrijednosti:

$$\rho_{\text{rez6(broj6, broj7, broj8)}}(G_{\text{MIN(tezina), AVG(visina), MAX(visina)}}(\text{osoba}))$$

rez6	broj6	broj7	broj8
	62	175.5	186

SQL - Agregatne funkcije

- naziv rezultantnog atributa nije definiran operacijom, stoga se koristi AS operator za preimenovanje

osoba	sifra	tezina	visina
	101	62	170
	103	94	186
	105	74	181
	107	62	165

```
SELECT COUNT(sifra) AS broj1 FROM osoba;
```

broj1
4

```
SELECT SUM(tezina) AS broj2 FROM osoba;
```

broj2
292

```
SELECT AVG(visina) AS broj3 FROM osoba;
```

broj3
175.5

```
SELECT MAX(visina) AS broj4,  
       MIN(tezina) AS broj5  
FROM osoba;
```

broj4
186

broj5
62

SQL - Agregatne funkcije

- agregatne funkcije s DISTINCT

osoba	sifra	tezina	visina
	101	62	170
	103	94	190
	105	74	170
	107	62	170

```
SELECT COUNT(DISTINCT visina) AS broj1  
FROM osoba;
```

broj1
2

```
SELECT SUM(DISTINCT tezina) AS broj2  
FROM osoba;
```

broj2
230

```
SELECT AVG(DISTINCT visina) AS broj3  
FROM osoba;
```

broj3
180

Agregacija i grupiranje

ispit	mbrStud	akGod	nazPred	ocjena
	100	2005	Matematika	3
	101	2005	Matematika	5
	102	2005	Matematika	2
	103	2006	Matematika	3
	100	2004	Fizika	5
	101	2006	Fizika	5
	102	2006	Fizika	2
	100	2005	Vjerojatnost	4

- Zadatak: izračunati prosječnu ocjenu za svaki pojedini predmet
 - prosjek za Matematiku
 - prosjek za Fiziku
 - ... i za sve ostale predmete čiji se naziv pojavljuje u relaciji

Agregacija i grupiranje

- Loše rješenje:

- Za svaki predmet napisati po jedan upit

$$\rho_{\text{projek}(\text{projOcjMat})}(G_{\text{AVG}(\text{ocjena})}(\sigma_{\text{nazPred} = \text{'Matematika'}}(\text{ispit})))$$

```
SELECT AVG(ocjena) AS projOcjMat
      FROM ispit
     WHERE nazPred = 'Matematika' ;
```

projOcjMat
3.25

$$\rho_{\text{projek}(\text{projOcjFiz})}(G_{\text{AVG}(\text{ocjena})}(\sigma_{\text{nazPred} = \text{'Fizika'}}(\text{ispit})))$$

```
SELECT AVG(ocjena) AS projOcjFiz
      FROM ispit
     WHERE nazPred = 'Fizika' ;
```

projOcjFiz
4

- itd. (za svaki naziv predmeta)
- postoji li bolje rješenje?

Grupiranje (*grouping*)

- Zadana je relacija $r(R)$. Neka su atributi $A_1, A_2, \dots, A_m, B_1, B_2, \dots, B_n$ atributi sheme R . Opći oblik operacije grupiranja je sljedeći:

$$A_1, A_2, \dots, A_m G_{\mathcal{AF}_1(B_1), \mathcal{AF}_2(B_2), \dots, \mathcal{AF}_n(B_n)} (r)$$

- određuju se grupe n-torki: u svakoj grupi se nalaze n-torke koje imaju jednake vrijednosti atributa A_1, A_2, \dots, A_m
- za svaku grupu n-torki izračunavaju se vrijednosti agregatnih funkcija $\mathcal{AF}_1(B_1), \mathcal{AF}_2(B_2), \dots, \mathcal{AF}_n(B_n)$
- za svaku grupu formira se n-torka s vrijednostima atributa A_1, A_2, \dots, A_m i izračunatim vrijednostima agregatnih funkcija

Agregacija i grupiranje

ispit	mbrStud	akGod	nazPred	ocjena
	100	2005	Matematika	3
	101	2005	Matematika	5
	102	2005	Matematika	2
	103	2006	Matematika	3
	100	2004	Fizika	5
	101	2006	Fizika	5
	102	2006	Fizika	2
	100	2005	Vjerojatnost	4



- Za svaki predmet ispisati prosječnu ocjenu (ispravno rješenje):

$$\rho_{\text{prosjek}(\text{nazPred}, \text{prosjOcj})}(\text{nazPred } G_{\text{AVG}(\text{ocjena})}(\text{ispit}))$$

- grupirati po nazPred
- za svaku grupu izračunati AVG(ocjena)
- za svaku grupu formirati po jednu n-torku s vrijednošću atributa nazPred i izračunatim prosjekom
- obaviti operaciju preimenovanja

prosjek	
nazPred	prosjOcj
Matematika	3.25
Fizika	4
Vjerojatnost	4

Agregacija i grupiranje

- Ispisati prosječnu i najveću ocjenu za svaki predmet i akademsku godinu:

ispit	mbrStud	akGod	nazPred	ocjena
	100	2005	Matematika	3
	101	2005	Matematika	5
	102	2005	Matematika	2
	103	2006	Matematika	3
	100	2004	Fizika	5
	101	2006	Fizika	5
	102	2006	Fizika	2
	100	2005	Vjerojatnost	4

- u istu grupu ulaze n-torce koje imaju jednake vrijednosti atributa nazPred i akGod

$$\rho_{\text{projek1}(\text{nazPred}, \text{akGod}, \text{projOcj}, \text{maxOcj})}(\text{nazPred}, \text{akGod}, G_{\text{AVG}(\text{ocjena}), \text{MAX}(\text{ocjena})}(\text{ispit}))$$

projek1	nazPred	akGod	projOcj	maxOcj
	Matematika	2005	3.333	5
	Matematika	2006	3	3
	Fizika	2004	5	5
	Fizika	2006	3.5	5
	Vjerojatnost	2005	4	4

SQL - Grupiranje

- **SELECT *SELECT List***
FROM ...
[WHERE *Condition*]
[GROUP BY *column* [, *column*] ...]

$\rho_{\text{projek1}(\text{nazPred}, \text{akGod}, \text{projOcj}, \text{maxOcj})}$ (nazPred, akGod $G_{\text{AVG}(\text{ocjena}), \text{MAX}(\text{ocjena})}$ (ispit))

```
SELECT nazPred  
      , akGod  
      , AVG(ocjena) AS projOcj  
      , MAX(ocjena) AS maxOcj  
  FROM ispit  
 GROUP BY nazPred, akGod;
```

nazPred	akGod	projOcj	maxOcj
Matematika	2005	3.333	5
Matematika	2006	3	3
Fizika	2004	5	5
Fizika	2006	3.5	5
Vjerojatnost	2005	4	4

SQL - Grupiranje

- svi atributi koji se nalaze u listi za selekciju, a koji nisu argumenti agregatnih funkcija, moraju biti navedeni u GROUP BY dijelu naredbe

```
SELECT nazPred  
      , akGod  
      , mbrStud NEISPRAVNO!  
      , AVG(ocjena) AS prosjOcj  
      , MAX(ocjena) AS maxOcj  
FROM ispit  
GROUP BY nazPred, akGod;
```

ispit

mbrStud	akGod	nazPred	ocjena
100	2005	Matematika	3
101	2005	Matematika	5
102	2005	Matematika	2
103	2006	Matematika	3
100	2004	Fizika	5
101	2006	Fizika	5
102	2006	Fizika	2
100	2005	Vjerojatnost	4

}

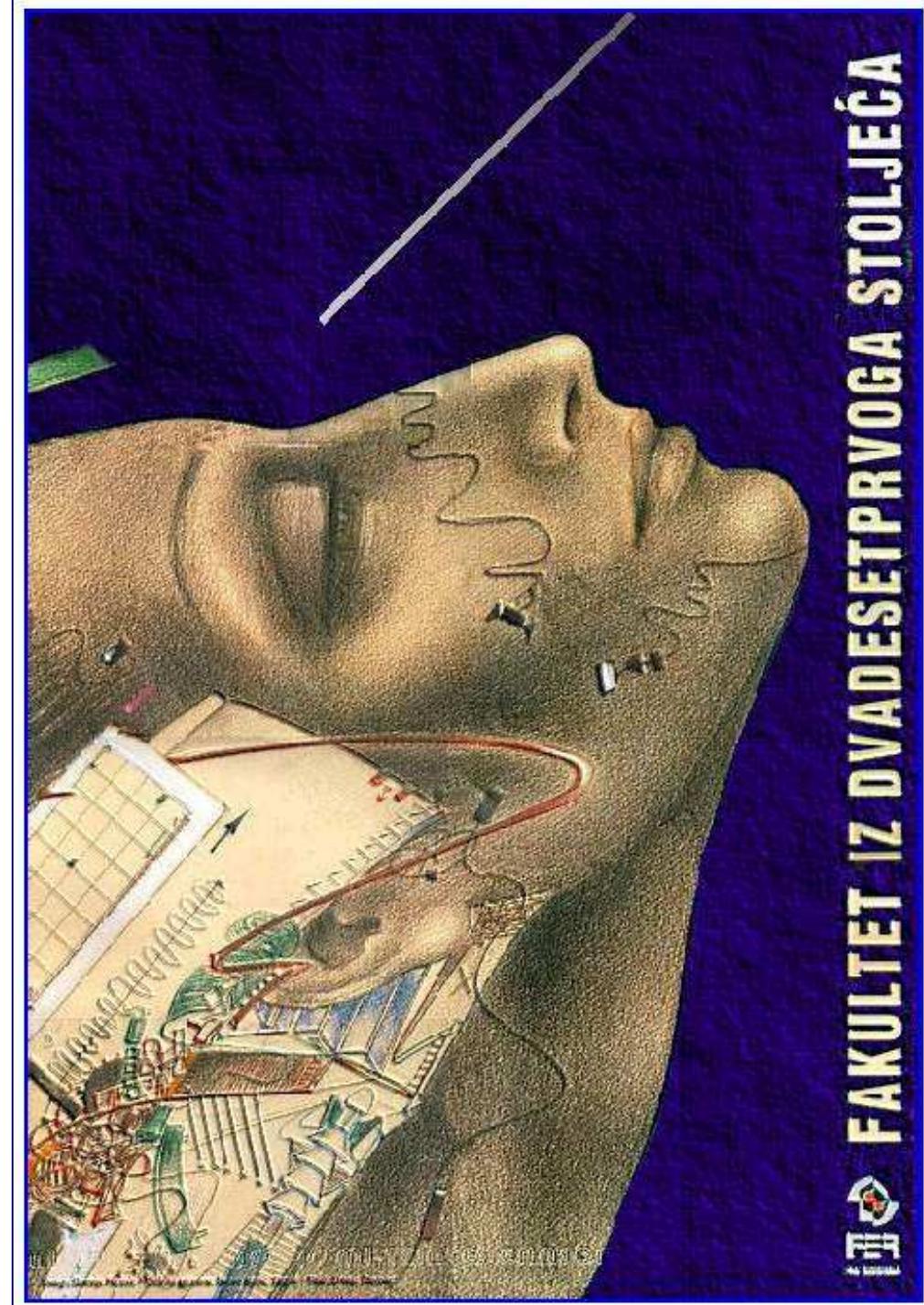
Zašto je to neispravno?
Za svaku grupu se formira samo po jedna n-torka: što s onim grupama u kojima postoji više vrijednosti atributa mbrStud?

nazPred	akGod	mbrStud	prosjOcj	maxOcj
Matematika	2005	100, 101, 102 ?	3.333	5
Matematika	2006	?	3	3
Fizika	2004	?	5	5
Fizika	2006	?	3.5	5
Vjerojatnost	2005	?	4	4

Baze podataka

Predavanja
ožujak 2014.

3. Nepotpune informacije i NULL vrijednosti



NULL vrijednosti

- Ponekad se dešava da informacije koje treba unijeti u bazu podataka nisu potpune
 - neke informacije trenutno nisu poznate
 - neke informacije uopće ne postoje (nisu primjenjive)
 - neke informacije postoje, ali do njih nije moguće doći
- Informacije koje nedostaju prikazuju se kao poseban oblik podatka: NULL vrijednost

nije primjenjivo
(vidi datum rođenja)

clanoviKnjiznice

mbr	ime	prez	pbr	datRodj	adresa	zanimanje
100	Maja	Novak	10000	01.5.2001	Ilica 1	NULL
105	Ivo	Kolar	21000	12.3.1973	NULL	odvjetnik
107	James	Bond	NULL	NULL	NULL	tajni agent

nedostupno

trenutno nepoznato

SQL - Interna pohrana NULL vrijednosti

- NULL vrijednost se interno pohranjuje drugačije od bilo koje druge dopuštene vrijednosti (nije 0, nije 0.0, nije prazan niz, ...)
- Način interne pohrane NULL vrijednosti je nebitan - NULL vrijednost je neovisna od tipa podatka kojeg predstavlja. U SQL naredbama, bez obzira na tip podatka, koristi se "konstanta" **NULL**

```
CREATE TABLE mjesto (
    pbr          INTEGER
, nazMjesto    CHAR(30)
, sifZupanija  SMALLINT
);
```

```
INSERT INTO mjesto VALUES (10000, 'Zagreb', NULL);
INSERT INTO mjesto VALUES (10001, NULL, 1);
UPDATE mjesto SET sifZupanija = NULL
WHERE pbr = 10001;
```

SQL - prikaz NULL vrijednosti

- Način na koji se NULL vrijednost prikazuje korisniku ovisi o programskom alatu koji se koristi:

AGS Server Studio SQL Client

The screenshot shows the AGS Server Studio SQL Client interface. The database is set to 'studadmin'. The SQL Editor window contains the query: 'SELECT * FROM clanoviKnjiznice'. The results grid displays three rows of data:

mbr	ime	prez	pbr	datrodj	adresa	zanimanje
1	100	Maja	Novak	10000	01.05.2001	Ilica 1
2	105	Ivo	Kolar	21000	12.03.1973	NULL
3	107	James	Bond	NULL	NULL	tajni agent

SQuirreL SQL Client

The screenshot shows the SQuirreL SQL Client interface. The catalog is set to 'studAdmin'. The SQL Editor window contains the query: 'SELECT * FROM clanoviKnjiznice'. The results grid displays three rows of data:

mbr	ime	prez	pbr	datrodj	adresa	zanimanje
100	Maja	Novak	10000	01.05.2001	Ilica 1	<null>
105	Ivo	Kolar	21000	12.03.1973	<null>	odvjetnik
107	James	Bond	<null>	<null>	<null>	tajni agent

SQL - Izrazi

- Izraz (*Expression*) se sastoji od
 - imena atributa
 - konstanti
 - operatora + - * / unarni + -
 - zagrada ()
- Izrazi se mogu koristiti
 - u listi za selekciju
 - u uvjetu u WHERE dijelu naredbe
 - i drugdje ...

SQL - Izrazi

- iznosi plaća za svaku osobu su navedeni u kolumnama
- ispisati matični broj, prezime i plaću izraženu u dolarima, za one osobe čija je plaća veća od 1000 eura
- $1 \text{ USD} = 5.6 \text{ KN}, 1 \text{ KN} = 0.136 \text{ EUR}$

dohodak		
mbr	prez	placa
100	Novak	7500
102	Horvat	5600
105	Kolar	9000
107	Ban	4200

```
SELECT mbr  
      , prez  
      , placac/5.6 ?  
  FROM dohodak  
 WHERE placac*0.136 > 1000;
```

mbr	prez	(expression)
100	Novak	1339.28571429
105	Kolar	1607.14285714

```
SELECT mbr  
      , prez  
      , placac/5.6 AS placacUSD  
  FROM dohodak  
 WHERE placac*0.136 > 1000;
```

mbr	prez	placaUSD
100	Novak	1339.28571429
105	Kolar	1607.14285714

NULL vrijednost u izrazima

- Neka je binarni operator $\alpha \in \{ +, -, *, / \}$, a X i Y su izrazi
 - ako jedan ili oba operanda X, Y poprimaju NULL vrijednost, tada je rezultat izraza X α Y također NULL vrijednost

5 + NULL → NULL

NULL - NULL → NULL

NULL * 0 → NULL

- Neka je unarni operator $\beta \in \{ +, - \}$, a X je izraz
 - ako operand X poprima NULL vrijednost, tada je rezultat izraza β X također NULL vrijednost

- NULL → NULL

SQL - NULL vrijednost u izrazima

bodovi

mbr	prez	bodLab	bodMI
101	Novak	12	NULL
103	Ban	NULL	NULL
107	Horvat	21	66.3
109	Kolar	NULL	54.3

```
SELECT mbr
      , prez
      , bodLab + bodMI AS ukupBodova
      , - bodLab AS negBodLab
FROM bodovi;
```

mbr	prez	ukupBodova	negBodLab
101	Novak	NULL	-12
103	Ban	NULL	NULL
107	Horvat	87.3	-21
109	Kolar	NULL	NULL

NULL vrijednost u uvjetima usporedbe

- Neka su X i Y izrazi, a γ je operator usporedbe
$$\gamma \in \{ <, \leq, =, \neq, >, \geq \}$$
- ako niti jedan od operanada X , Y nije NULL vrijednost, tada je rezultat izraza $X \gamma Y$ logička vrijednost istina (*true*) ili logička vrijednost laž (*false*)
- ako jedan ili oba operanda X , Y jesu NULL vrijednosti, tada je rezultat izraza $X \gamma Y$ logička vrijednost nepoznato (*unknown*)

$7 \geq 5$	\rightarrow true
'atlas' > 'zvuk'	\rightarrow false
-17.8 \leq NULL	\rightarrow unknown
NULL = NULL	\rightarrow unknown
NULL \neq NULL	\rightarrow unknown

Operacija selekcije - NULL vrijednosti

- Obavljanjem **operacije selekcije** $\sigma_F(r)$ dobiva se relacija koja sadrži samo one n-torke relacije r za koje je vrijednost predikata F istina (*true*). To znači da se n-torke za koje je vrijednost predikata F laž (*false*) ili nepoznato (*unknown*) ne pojavljuju u rezultatu

r	A	B
1	20	
2	NULL	
3	60	

$$s = \sigma_{B \leq 50}(r)$$

$20 \leq 50 \rightarrow \text{true}$
 $\text{NULL} \leq 50 \rightarrow \text{unknown}$
 $60 \leq 50 \rightarrow \text{false}$

s	A	B
	1	20

r	A	B
1	20	
2	NULL	
3	60	

$$s = \sigma_{B \neq 20}(r)$$

$20 \neq 20 \rightarrow \text{false}$
 $\text{NULL} \neq 20 \rightarrow \text{unknown}$
 $60 \neq 20 \rightarrow \text{true}$

s	A	B
	3	60

SQL - Selekcija i NULL vrijednosti

student	matBr	prez	postBr
	100	Kolar	52000
	102	Horvat	10000
	105	Novak	NULL
	107	Ban	10000

$\sigma_{\text{postBr} = 10000}$

```
SELECT * FROM student  
WHERE postBr = 10000;
```



matBr	prez	postBr
102	Horvat	10000
107	Ban	10000

$\sigma_{\text{postBr} \neq 10000}$

```
SELECT * FROM student  
WHERE postBr <> 10000;
```



matBr	prez	postBr
100	Kolar	52000

■ gdje je Novak ?

SQL - operatori usporedbe IS NULL, IS NOT NULL

student

	matBr	prez	postBr
100	Kolar	52000	
102	Horvat	10000	
105	Novak		NULL
107	Ban		10000

- U SQL-u nije dopušteno operatore usporedbe $<$, \leq , $=$, \neq , $>$, \geq koristiti u kombinaciji s "konstantom" NULL (npr. $=NULL$, $\neq NULL$, ...)

```
SELECT * FROM student  
WHERE postBr = NULL;
```

Neispravna naredba

```
SELECT * FROM student  
WHERE postBr IS NULL;
```



matBr	prez	postBr
105	Novak	NULL

```
SELECT * FROM student  
WHERE postBr IS NOT NULL;
```



matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
107	Ban	10000

- Rezultat logičkog izraza $X IS NULL$ ili logičkog izraza $X IS NOT NULL$ je uvijek ili *true* ili *false*

Trovalentna logika

- Osnovne logičke operacije - tablice istinitosti u prisustvu logičke vrijednosti *unknown*

AND	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	false
false	false	false	false

OR	true	unknown	false
true	true	true	true
unknown	true	unknown	unknown
false	true	unknown	false

NOT	
true	false
unknown	unknown
false	true

Selekcija, logički operatori i NULL vrijednosti

$$s = \sigma_{B \leq 50 \wedge C \neq 300}(r)$$

r	A	B	C
1	NULL	100	
2	20	200	
3	30	300	
4	40	NULL	
5	50	500	
6	60	NULL	

$\text{NULL} \leq 50 \wedge 100 \neq 300 \rightarrow \text{unknown}$ (*)
 $20 \leq 50 \wedge 200 \neq 300 \rightarrow \text{true}$
 $30 \leq 50 \wedge 300 \neq 300 \rightarrow \text{false}$
 $40 \leq 50 \wedge \text{NULL} \neq 300 \rightarrow \text{unknown}$
 $50 \leq 50 \wedge 500 \neq 300 \rightarrow \text{true}$
 $60 \leq 50 \wedge \text{NULL} \neq 300 \rightarrow \text{false}$ (**)

* $\text{NULL} \leq 50 \rightarrow \text{unknown}$; $100 \neq 300 \rightarrow \text{true}$; $\text{unknown} \wedge \text{true} \rightarrow \text{unknown}$

** $60 \leq 50 \rightarrow \text{false}$; $\text{NULL} \neq 300 \rightarrow \text{unknown}$; $\text{false} \wedge \text{unknown} \rightarrow \text{false}$

s	A	B	C
2	20	200	
5	50	500	

SQL - Logički operatori i NULL vrijednosti

bodovi

mbr	prez	bodLab	bodMI
101	Novak	6	NULL
103	Ban	NULL	NULL
105	Horvat	12	44.0
107	Kolar	NULL	85.0
109	Pevec	20	15.0

- Za prolaz je potrebno barem 10 bodova iz labosa i barem 50 bodova ukupno. Studentima koji nisu dolazili na labos ili izlazili na međuispite upisana je NULL vrijednost. Ispisati studente koji **nisu položili** ispit.

Ovaj upit ne daje zadovoljavajući rezultat

```
SELECT *
  FROM bodovi
 WHERE bodLab < 10
   OR bodMI + bodLab < 50;
```

mbr	prez	bodLab	bodMI
101	Novak	6	NULL
109	Pevec	20	15.0

```
SELECT *
  FROM bodovi
 WHERE bodLab IS NULL
   OR bodMI IS NULL
   OR bodLab < 10
   OR bodLab + bodMI < 50;
```

mbr	prez	bodLab	bodMI
101	Novak	6	NULL
103	Ban	NULL	NULL
107	Kolar	NULL	85.0
109	Pevec	20	15.0

NULL vrijednosti i skupovi

- Neka skup S sadrži vrijednosti: $S = \{1, 2, 3, \text{NULL}\}$
- NULL vrijednost je nepoznata, ali može poprimiti i neku od vrijednosti 1, 2 ili 3
 - kardinalnost skupa S je neodređena (može biti 3 ili 4)
 - narušena je definicija skupa (u skupu nije dozvoljena pojava dviju ili više jednakih vrijednosti)
 - što je logička vrijednost suda $\text{NULL} \in S \rightarrow \text{unknown}$
 - što je logička vrijednost suda $4 \in S \rightarrow \text{unknown}$

NULL vrijednosti i skupovi

- Sustavi za upravljanje bazama podataka nisu u stanju međusobno razlikovati NULL vrijednosti, stoga se kao konvencija koristi sljedeći model rukovanja s NULL vrijednostima u skupovima:
 - dopuštena je pojava jedne i samo jedne NULL vrijednosti u skupu
 - element **e** je **kopija** jednog od elemenata u skupu:
 - ako vrijednost elementa **e** nije NULL, a u skupu postoji element s jednakom vrijednošću
ili
 - ako vrijednost elementa **e** jest NULL, a u skupu **S** već postoji element s NULL vrijednošću

Kopija n-torke

- elementi relacije su n-torke
- Definicija kopije n-torke:
 - neka su t_1 i t_2 n-torke definirane na shemi $\{ A_1, A_2, \dots, A_n \}$
 - $t_1 = \langle d_1, d_2, \dots, d_n \rangle$, $t_2 = \langle e_1, e_2, \dots, e_n \rangle$
 - n-torka t_1 je kopija n-torke t_2 ako i samo ako $\forall i, 1 \leq i \leq n$, vrijedi:
 - $(d_i = e_i) \vee (d_i \text{ jest NULL} \wedge e_i \text{ jest NULL})$
- neformalno: ako su vrijednosti korespondentnih atributa n-torki ili jednake ili su obje NULL

Kopija n-torke

- Primjer:

osoba				student			
mbr	ime	prez	postBr	mbr	ime	prez	postBr
100	Ivan	Novak	10000	100	Ivan	Novak	NULL
102	Ana	Horvat	21000	102	Ana	Horvat	21000
103	Tea	Ban	52000	103	Tea	Ban	21000
105	NULL	Kolar	NULL	105	NULL	Kolar	NULL

Diagram illustrating the copying of rows from the 'osoba' table to the 'student' table. Red arrows labeled 'nije kopija' point from the 'osoba' table to the 'student' table for rows 100, 103, and 105. Green arrows labeled 'jest kopija' point from the 'osoba' table to the 'student' table for rows 102 and 100.

Unija, razlika i presjek - NULL vrijednosti

- unija, razlika i presjek su skupovske operacije: pri usporedbi elemenata (n-torki) treba voditi računa o definiciji kopije n-torke

r	A	B	C
1	a	α	
2	b	NULL	
3	NULL	γ	
4	NULL	NULL	

s	A	B	C
1	a	α	
2	b	NULL	
3	c	γ	
4	NULL	NULL	

$r \cup s$	A	B	C
1	a	α	
2	b	NULL	
3	NULL	γ	
3	c	γ	
4	NULL	NULL	

$r \cap s$	A	B	C
1	a	α	
2	b	NULL	
4	NULL	NULL	

$r \setminus s$	A	B	C
3	NULL	γ	

$s \setminus r$	A	B	C
3	c	γ	

Projekcija - NULL vrijednosti

- pri obavljanju operacije **projekcije** potrebno je u fazi eliminacije duplikata voditi računa o definiciji kopije n-torce

$$s = \pi_{B, C}(r)$$

r	A	B	C
1	a		α
2	b		NULL
3	NULL		γ
4	a		α
5	NULL		NULL
6	NULL		γ
7	NULL		NULL

izdvajanje
vertikalnog
podskupa



"međurezultat"

	B	C
a		α
b		NULL
NULL		γ
a		α
NULL		NULL
NULL		γ
NULL		NULL

eliminacija
duplikata



s	B	C
a		α
b		NULL
NULL		γ
NULL		NULL

Kartezijev produkt - NULL vrijednosti

- pri obavljanju operacije **Kartezijevog produkta** NULL vrijednosti nemaju utjecaja

r	A	B	C
	1	a	α
	2	b	NULL
	3	NULL	NULL

s	E	F
	1	NULL
	NULL	f

r × s	A	B	C	E	F
	1	a	α	1	NULL
	2	b	NULL	1	NULL
	3	NULL	NULL	1	NULL
	1	a	α	NULL	f
	2	b	NULL	NULL	f
	3	NULL	NULL	NULL	f

Spajanje uz uvjet i spajanje s izjednačavanjem - NULL vrijednosti

- pri obavljanju operacija **spajanja uz uvjet** i **spajanja s izjednačavanjem** potrebno je voditi računa o tome da se spajaju samo one n-torce za koje uvjet spajanja ima logičku vrijednost istina (*true*)

linija

let	udaljenost
CA-825	700
LH-412	NULL
BA-722	4100
CA-311	13000

zrakoplov

tip	dolet
B747	NULL
A320	5400
DC-9	3100

mogućnost = linija \bowtie zrakoplov
 $dolet \geq udaljenost$

mogućnost

let	udaljenost	tip	dolet
CA-825	700	A320	5400
CA-825	700	DC-9	3100
BA-722	4100	A320	5400

Prirodno spajanje - NULL vrijednosti

- slično, pri obavljanju operacije **prirodnog spajanja** potrebno je voditi računa o tome da se spajaju samo one n-torce za koje uvjet spajanja ima logičku vrijednost istina (*true*)

mjesto	pbr	nazMjesto	sifZup
	42000	Varaždin	7
	42230	Ludbreg	NULL
	42220	Novi Marof	7

zupanija	sifZup	nazZup
	7	Varaždinska
	NULL	Istarska

$\text{mjestouZupaniji} = \text{mjesto} \triangleright\triangleleft \text{zupanija}$

mjestouZupaniji	pbr	nazMjesto	sifZup	nazZup
	42000	Varaždin	7	Varaždinska
	42220	Novi Marof	7	Varaždinska

- n-torka $\langle 42230, \text{Ludbreg}, \text{NULL} \rangle$ neće se spojiti s n-torkom $\langle \text{NULL}, \text{Istarska} \rangle$ jer je rezultat usporedbe $\text{NULL}=\text{NULL} \rightarrow \text{unknown}$

Agregacija - NULL vrijednosti

- ako su sve vrijednosti za koje se izračunava agregatna funkcija NULL vrijednosti, ili ako se aggregatna funkcija izračunava za prazan skup vrijednosti
 - rezultat aggregatne funkcije COUNT je nula
 - rezultat ostalih aggregatnih funkcija je NULL
- ako među vrijednostima za koje se izračunava aggregatna funkcija postoje vrijednosti koje nisu NULL vrijednosti
 - aggregatna funkcija se izračunava tako da se NULL vrijednosti zanemaruju (ne uzimaju se u obzir pri izračunavanju)

Agregacija - NULL vrijednosti

ispit

	mbrStud	nazPred	ocjena
100	Matematika	NULL	
101	Matematika	4	
102	Matematika	3	
103	Matematika	3	
100	Fizika	NULL	
101	Fizika	3	

```
SELECT COUNT(mbrStud) AS broj1  
FROM ispit;
```

broj1
6

```
SELECT COUNT(ocjena) AS broj2  
FROM ispit;
```

broj2
4

```
SELECT COUNT(ocjena) AS broj3 FROM ispit  
WHERE mbrStud = 100;
```

broj3
0

```
SELECT COUNT(ocjena) AS broj4 FROM ispit  
WHERE mbrStud = 200;
```

broj4
0

```
SELECT AVG(ocjena) AS broj5 FROM ispit;
```

broj5
3.25

```
SELECT AVG(ocjena) AS broj6 FROM ispit  
WHERE mbrStud = 100;
```

broj6
NULL

```
SELECT AVG(ocjena) AS broj7 FROM ispit  
WHERE mbrStud = 200;
```

broj7
NULL

Agregatna funkcija COUNT(*)

- Agregatna funkcija COUNT(*imeAtributa*)
 - broji n-torke u kojima vrijednost atributa *imeAtributa* nije NULL vrijednost
- Agregatna funkcija COUNT(*)
 - broji n-torke zanemarujući njihov sadržaj

ispit	mbrStud	nazPred	ocjena
	100	Matematika	NULL
	101	Matematika	4
	102	Matematika	3
	103	Matematika	3
	100	Fizika	NULL
	101	Fizika	3

```
SELECT COUNT(ocjena) AS brojOcj,  
       COUNT(*) AS brojRedaka  
FROM ispit;
```

brojOcj	brojRedaka
4	6

- Ne postoji agregatna funkcija COUNT(DISTINCT *)

Grupiranje - NULL vrijednosti

- pri obavljanju operacije grupiranja, grupiranje n-torki se obavlja tako da se vodi računa o definiciji kopije n-torki
 - ako se grupiranje obavlja prema atributima iz skupa X, tada u istu grupu ulaze one n-torke čije su X-vrijednosti međusobne kopije

```
SELECT akGod, nazPred, AVG(ocjena) AS prosj
FROM ispit
GROUP BY akGod, nazPred;
```

ispit

	mbrStud	akGod	nazPred	ocjena
t ₁	100	2005	NULL	3
t ₂	101	NULL	NULL	5
t ₃	102	2005	NULL	2
t ₄	103	2006	Fizika	3
t ₅	100	NULL	NULL	5
t ₆	101	2006	Fizika	5
t ₇	102	2005	NULL	2

$$X = \{ \text{akGod}, \text{nazPred} \}$$

$t_1(X), t_3(X)$ i $t_7(X)$ su međusobne kopije

$t_2(X)$ i $t_5(X)$ su međusobne kopije

$t_4(X)$ i $t_6(X)$ su međusobne kopije

akGod	nazPred	prosj
2005	NULL	2.3333
NULL	NULL	5
2006	Fizika	4

Vanjsko spajanje - uvod

student	matBr	prez
101	Kolar	
102	Horvat	
103	Novak	

upisanPred	matBr	nazPred
101	Matematika	
101	Fizika	
101	Programiranje	
102	Fizika	

upisani = student \bowtie upisanPred

upisani	matBr	prez	nazPred
101	Kolar	Matematika	
101	Kolar	Fizika	
101	Kolar	Programiranje	
102	Horvat	Fizika	

- n-torka **<103, Novak>** neće se pojaviti u rezultatu jer u relaciji **upisanPred** ne postoji niti jedna n-torka koja zadovoljava uvjet spajanja s tom n-torkom

Vanjsko spajanje - uvod

mjesto	pbr	nazMjesto	sifZup
	42000	Varaždin	7
	42230	Ludbreg	NULL
	42220	Novi Marof	7

zupanija	sifZup	nazZup
	7	Varaždinska
	NULL	Istarska

mjestouZupaniji = mjesto $\triangleright\triangleleft$ zupanija

mjestouZupaniji	pbr	nazMjesto	sifZup	nazZup
	42000	Varaždin	7	Varaždinska
	42220	Novi Marof	7	Varaždinska

- n-torka <42230, Ludbreg, NULL> neće se pojaviti u rezultatu jer u relaciji **zupanija** ne postoji niti jedna n-torka koja zadovoljava uvjet spajanja (a ne može je niti biti, jer sifZup ima NULL vrijednost)

Lijevo vanjsko spajanje (*Left outer join*)

- sve n-torke relacije **student** će se pojaviti u rezultatu spajanja ako se primijeni operacija **lijevog vanjskog spajanja**

student	matBr	prez
101	Kolar	
102	Horvat	
103	Novak	

upisanPred	matBrSt	nazPred
101	Matematika	
101	Fizika	
101	Programiranje	
102	Fizika	

upisano = student * $\triangleright\triangleleft$ upisanPred

matBr = matBrSt

upisano	matBr	prez	matBrSt	nazPred
	101	Kolar	101	Matematika
	101	Kolar	101	Fizika
	101	Kolar	101	Programiranje
	102	Horvat	102	Fizika
	103	Novak	NULL	NULL

- n-torkama "lijeve" relacije za koje ne postoji odgovarajuće n-torke u "desnoj" relaciji se kao vrijednosti atributa iz "desne" relacije postavljaju NULL vrijednosti

Lijevo vanjsko spajanje (*Left outer join*)

mjesto	pbr	nazMjesto	sifZupMj
	42000	Varaždin	7
	42230	Ludbreg	NULL
	42220	Novi Marof	7

zupanija	sifZup	nazZup
	7	Varaždinska
	NULL	Istarska

$mjestouZupaniji = mjesto * \triangleright \triangleleft zupanija$
 $sifZupMj = sifZup$

mjestouZupaniji	pbr	nazMjesto	sifZupMj	sifZup	nazZup
	42000	Varaždin	7	7	Varaždinska
	42230	Ludbreg	NULL	NULL	NULL
	42220	Novi Marof	7	7	Varaždinska

SQL - Lijevo vanjsko spajanje (*Left outer join*)

mjesto	pbr	nazMjesto	sifZupMj
	42000	Varaždin	7
	42230	Ludbreg	NULL
	42220	Novi Marof	7

zupanija	sifZup	nazZup
	7	Varaždinska
	NULL	Istarska

mjesto * $\triangleright \triangleleft$ zupanija
sifZupMj = sifZup

```
SELECT mjesto.* , zupanija.* ←  
FROM mjesto LEFT OUTER JOIN zupanija  
ON sifZupMj = sifZup;
```

ili SELECT *

pbr	nazMjesto	sifZupMj	sifZup	nazZup
42000	Varaždin	7	7	Varaždinska
42230	Ludbreg	NULL	NULL	NULL
42220	Novi Marof	7	7	Varaždinska

Desno vanjsko spajanje (*Right outer join*)

- sve n-torce relacije **nastavnik** će se pojaviti u rezultatu spajanja ako se primjeni operacija **desnog vanjskog spajanja**

student	mbrSt	prezSt	temaSt
	101	Horvat	Tranzistori
	103	Novak	Teslini izumi
	105	Kolar	Teorija kaosa

nastavnik	sifNast	prezNast	temaNast
	202	Ban	Teslini izumi
	204	Toplek	Elektrane
	206	Oreb	Teslini izumi
	209	Pernar	Teorija kaosa

moguciMent = student $\triangleright\triangleleft^*$ nastavnik
temaSt = temaNast

- n-torkama "desne" relacije za koje ne postoji odgovarajuće n-torce u "lijevoj" relaciji se kao vrijednosti atributa iz "lijeve" relacije postavljaju NULL vrijednosti

moguciMent	mbrSt	prezSt	temaSt	sifNast	prezNast	temaNast
	103	Novak	Teslini izumi	202	Ban	Teslini izumi
	NULL	NULL	NULL	204	Toplek	Elektrane
	103	Novak	Teslini izumi	206	Oreb	Teslini izumi
	105	Kolar	Teorija kaosa	209	Pernar	Teorija kaosa

SQL - Desno vanjsko spajanje (*Right outer join*)

student

mbrSt	prezSt	temaSt
101	Horvat	Tranzistori
103	Novak	Teslini izumi
105	Kolar	Teorija kaosa

nastavnik

sifNast	prezNast	temaNast
202	Ban	Teslini izumi
204	Toplek	Elektrane
206	Oreb	Teslini izumi
209	Pernar	Teorija kaosa

student $\triangleright\triangleleft^*$ nastavnik
temaSt = temaNast

```
SELECT student.* , nastavnik.* ←  
FROM student RIGHT OUTER JOIN nastavnik  
ON temaSt = temaNast;
```

ili SELECT *

mbrSt	prezSt	temaSt	sifNast	prezNast	temaNast
103	Novak	Teslini izumi	202	Ban	Teslini izumi
NULL	NULL	NULL	204	Toplek	Elektrane
103	Novak	Teslini izumi	206	Oreb	Teslini izumi
105	Kolar	Teorija kaosa	209	Pernar	Teorija kaosa

Puno vanjsko spajanje (*Full outer join*)

- sve n-torce iz obje relacije će se pojaviti u rezultatu spajanja ako se primjeni operacija **punog vanjskog spajanja**

student		
mbrSt	prezSt	temaSt
101	Horvat	Tranzistori
103	Novak	Teslini izumi
105	Kolar	Teorija kaosa

nastavnik		
sifNast	prezNast	temaNast
202	Ban	Teslini izumi
204	Toplek	Elektrane
206	Oreb	Teslini izumi
209	Pernar	Teorija kaosa

student*▷◁***nastavnik**
temaSt = temaNast

moguciMent	mbrSt	prezSt	temaSt	sifNast	prezNast	temaNast
	101	Horvat	Tranzistori	NULL	NULL	NULL
	103	Novak	Teslini izumi	202	Ban	Teslini izumi
	NULL	NULL	NULL	204	Toplek	Elektrane
	103	Novak	Teslini izumi	206	Oreb	Teslini izumi
	105	Kolar	Teorija kaosa	209	Pernar	Teorija kaosa

SQL - Puno vanjsko spajanje (*Full outer join*)

student

mbrSt	prezSt	temaSt
101	Horvat	Tranzistori
103	Novak	Teslini izumi
105	Kolar	Teorija kaosa

nastavnik

sifNast	prezNast	temaNast
202	Ban	Teslini izumi
204	Toplek	Elektrane
206	Oreb	Teslini izumi
209	Pernar	Teorija kaosa

student*▷◁*nastavnik

temaSt = temaNast

```
SELECT student.* , nastavnik.*  
FROM student FULL OUTER JOIN nastavnik  
ON temaSt = temaNast;
```

ili SELECT *

Prirodno vanjsko spajanje

- kod vanjskog spajanja uz uvjet i vanjskog spajanja s izjednačavanjem u shemi rezultata se pojavljuju **svi atributi obje relacije**
- kod prirodnog lijevog vanjskog spajanja iz sheme rezultata se **izbacuju istoimeni atributi desnog operanda** (jer ionako mogu poprimiti ili vrijednosti jednake vrijednostima korespondentnih atributa lijevog operanda ili NULL vrijednosti)
- kod prirodnog desnog vanjskog spajanja iz rezultata se **izbacuju istoimeni atributi lijevog operanda** (jer ionako mogu poprimiti ili vrijednosti jednake vrijednostima korespondentnih atributa desnog operanda ili NULL vrijednosti)
- kod prirodnog punog vanjskog spajanja potrebno je **u shemi rezultata zadržati sve attribute obje relacije, te primijeniti operator preimenovanja atributa**

Prirodno vanjsko spajanje

student

mbrSt	prezSt	tema
101	Horvat	Tranzistori
103	Novak	Teslini izumi
105	Kolar	Teorija kaosa

nastavnik

sifNast	prezNast	tema
202	Ban	Teslini izumi
204	Toplek	Elektrane
206	Oreb	Teslini izumi
209	Pernar	Teorija kaosa

student*▷◁nastavnik

mbrSt	prezSt	tema	sifNast	prezNast
101	Horvat	Tranzistori	NULL	NULL
103	Novak	Teslini izumi	202	Ban
103	Novak	Teslini izumi	206	Oreb
105	Kolar	Teorija kaosa	209	Pernar

student▷◁*nastavnik

mbrSt	prezSt	sifNast	prezNast	tema
103	Novak	202	Ban	Teslini izumi
NULL	NULL	204	Toplek	Elektrane
103	Novak	206	Oreb	Teslini izumi
105	Kolar	209	Pernar	Teorija kaosa

Puno prirodno vanjsko spajanje

- to je u stvari **puno vanjsko spajanje s izjednačavanjem** (s preimenovanjem atributa)

student		
mbrSt	prezSt	tema
101	Horvat	Tranzistori
103	Novak	Teslini izumi
105	Kolar	Teorija kaosa

nastavnik		
sifNast	prezNast	tema
202	Ban	Teslini izumi
204	Toplek	Elektrane
206	Oreb	Teslini izumi
209	Pernar	Teorija kaosa

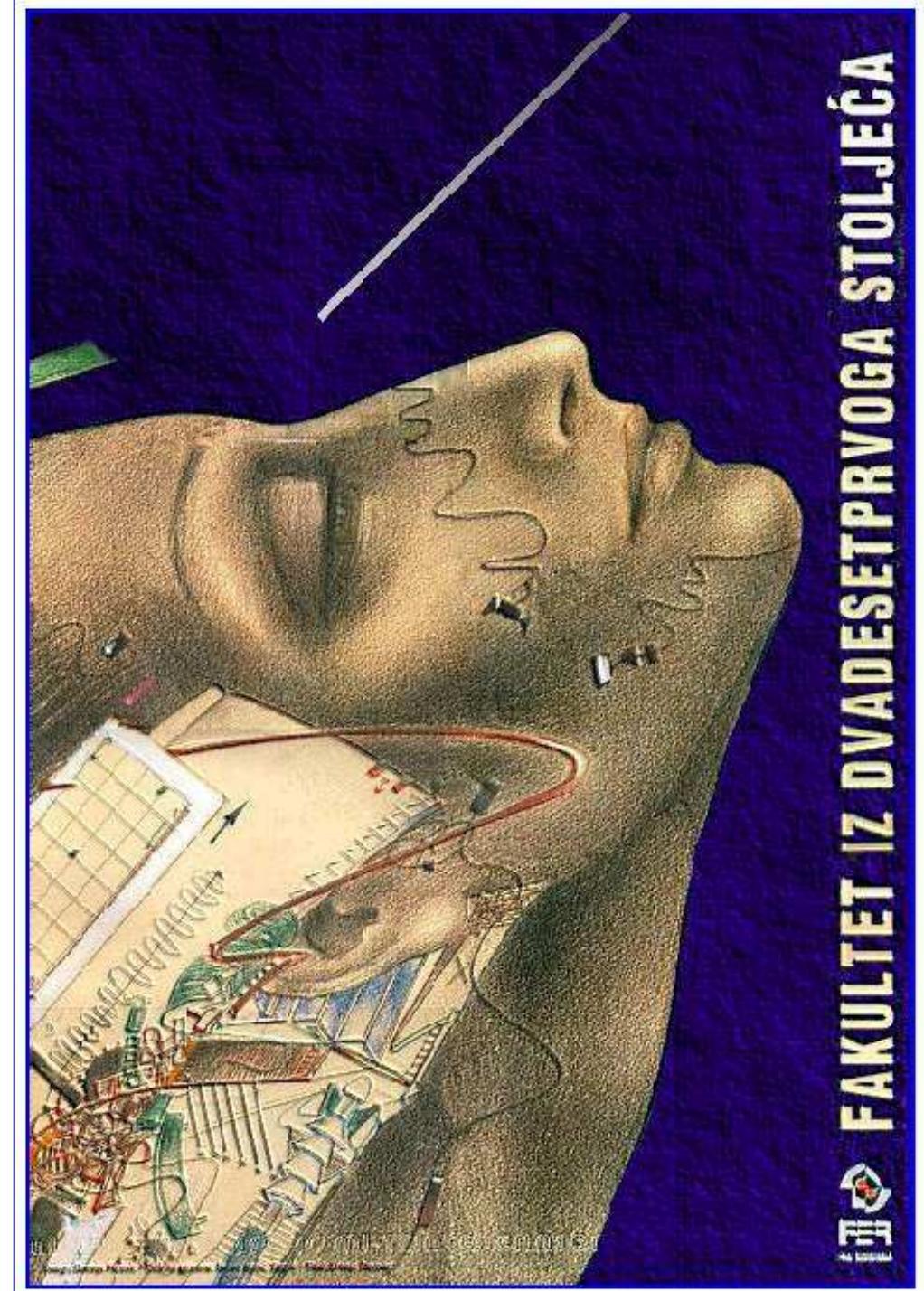
$\text{student}^* \triangleright \triangleleft^* \rho_{\text{nastavnik}(\text{sifNast}, \text{prezNast}, \text{temaNast})} (\text{nastavnik})$
tema = temaNast

mbrSt	prezSt	tema	sifNast	prezNast	temaNast
101	Horvat	Tranzistori	NULL	NULL	NULL
103	Novak	Teslini izumi	202	Ban	Teslini izumi
NULL	NULL	NULL	204	Toplek	Elektrane
103	Novak	Teslini izumi	206	Oreb	Teslini izumi
105	Kolar	Teorija kaosa	209	Pernar	Teorija kaosa

Baze podataka

Predavanja
ožujak 2014.

4. SQL (1. dio)



SQL - Uvod

- objedinjuje funkcije jezika za definiciju podataka (DDL) i jezika za rukovanje podacima (DML)
- razvoj započeo 70-tih godina
 - IBM San José Research Laboratory (California, USA)
- *Structured Query Language* je standardni jezik relacijskih baza podataka (*database language*)
 - 1986. godine - SQL-86 ili SQL1 (prva verzija standarda)
 - 1992. godine - SQL-92 ili SQL2
 - 1999. godine - SQL:1999
 - 2003. godine - SQL:2003
- proizvođači komercijalnih sustava često ugrađuju i svoje nestandardne DDL i DML naredbe
 - programski kod postaje neprenosiv između različitih SQL sustava
 - otežava se usaglašavanje oko budućih standarda.

SQL - Uvod

- neproceduralnost - naredbom je dovoljno opisati što se želi dobiti kao rezultat - nije potrebno definirati kako do tog rezultata doći
- u SUBP ugrađeni optimizator upita pronađi najefikasniji način obavljanja upita

zupanija		mjesto		
sifZup	nazZup	pbr	nazMjesto	sifZup
2	Primorsko-goranska	42000	Varaždin	7
7	Varaždinska	51000	Rijeka	2
4	Istarska	52100	Pula	4
		42230	Ludbreg	7

- ispisati podatke o mjestima u Varaždinskoj županiji. Rezultate poredati prema nazivu mjesta

```
SELECT mjesto.* FROM mjesto, zupanija  
WHERE mjesto.sifZup = zupanija.sifZup  
AND nazZup = 'Varaždinska'  
ORDER BY nazMjesto;
```

SQL - Vrste objekata

- Baza podataka *Database*
- Relacija (tablica) *Table*
- Atribut (stupac, kolona) *Column*
- Virtualna tablica (pogled) *View*
- Sinonim *Synonym*
- Integritetsko ograničenje *Constraint*
- Indeks *Index*
- Pohranjena procedura *Stored Procedure*
- Varijabla u pohranjenoj proceduri *SPL variable*
- Okidač *Trigger*

SQL - Identifikatori

- Identifikatori (imena objekata) se formiraju iz slova, znaka '_' i znamenki. Prvi znak od ukupno 128 značajnih (signifikantnih) znakova mora biti slovo ili znak '_'
- ispravno formirani identifikatori

`stud`

`ispiti2000godine`

`stud_ispit`

`_1mjesece`

- neispravno formirani identifikatori

`_11.mjesece`

`11mjesece`

`stud-ispit`

SQL - Rezervirane riječi

- SQL je "neosjetljiv" (case *insensitive*) na razliku između velikih i malih slova kada su u pitanju rezervirane riječi (SELECT, UPDATE, DELETE, FROM, WHERE, ...) i identifikatori

```
SELECT * FROM mjesto  
WHERE sifZupanija = 7
```

=

```
select * FrOm MJesto  
wHERE SIFZupanIJA = 7
```

- Međutim, razlika između velikih i malih slova **postoji** kad su u pitanju nizovi znakova

' Ivan ' ≠ ' IVAN '

SQL - Format naredbi

- SQL je jezik slobodnog formata naredbi (jednako kao C)

```
SELECT * FROM mjesto  
WHERE sifZupanija = 7
```

≡

```
SELECT  
*  
FROM  
mjesto  
WHERE  
sifZupanija = 7
```

SQL - Korištenje komentara

- "blok komentari" (jednako kao u programskom jeziku C)
 - dio teksta omeđen oznakama /* i */

```
/* ovo je komentar koji se  
proteže kroz više redaka teksta */
```

- "linijski komentari"
 - mjesto u retku na kojem se nalaze znakovi -- predstavlja početak komentara koji se proteže do kraja retka

```
-- ovo je komentar  
SELECT * FROM mjesto    -- ovo je komentar  
      WHERE pbr = 10000    -- ovo je komentar
```

SQL - Tipovi podataka

■ INTEGER

- cijeli broj pohranjen u 4 bajta u aritmetici dvojnog komplementa. Dopušteni raspon brojeva određen je intervalom

$$[-2^{n-1}, 2^{n-1} - 1] \quad n=32$$

- dakle, raspon brojeva bi trebao biti:

$$[-2147483648, 2147483647]$$

- u stvarnosti je manji, jer se vrijednost -2147483648 koristi za pohranu *NULL* vrijednosti. Raspon brojeva koji se mogu prikazati je:

$$[-2147483647, 2147483647]$$

Konstante:

5 -30000 0 1765723712 NULL

SQL - Tipovi podataka

▪ **SMALLINT**

- cijeli broj pohranjen u 2 bajta. Raspon brojeva koji se mogu prikazati je [-32767, 32767]

Konstante:

5 -30000 0 NULL

▪ **CHAR(m)**

- znakovni niz (*string*) s unaprijed definiranom maksimalnom duljinom $m \leq 32767$. Npr: CHAR(24).

Konstante:

'Ana' '12345' NULL
'Dvostruki navodnik " unutar niza'
'Jednostruki navodnik '' unutar niza'

- **uočite:** koriste se **jednostruki** navodnici (drugačije nego u jeziku C)

SQL - Tipovi podataka

- **NCHAR(m)**

- jednakо као и CHAR tip podatka, ali omogućava ispravno leksikografsko uređenje nizova znakova koji sadrže znakove iz nacionalnih kodnih stranica (*character set*). Koristi se onda kada se predviđa potreba za leksikografskim poretkom nizova znakova u kojima se pojavljuju specifični nacionalni znakovi (Č, Ć, Š, Đ, Ž, ...), npr. za atribut prezime

SQL - Tipovi podataka

▪ **REAL**

- odgovara tipu podatka **float** u jeziku C (IEEE-754 format prikaza - jednostruka preciznost)

Konstante:

23 -343.23 232.233E3 23.0e-24 NULL

▪ **DOUBLE PRECISION**

- odgovara tipu podatka **double** u jeziku C (IEEE-754 format prikaza - dvostruka preciznost)

Konstante:

23 -343.23 232.233E3 23.0e-302 NULL

SQL - Tipovi podataka

- **DECIMAL(m, n)**
 - ukupni broj znamenki (*precision*, $m \leq 32$)
 - broj znamenki iza decimalne točke (*scale*, $n \leq m$)
 - npr, DECIMAL (15, 3) predstavlja decimalni broj sa ukupno najviše 15 znamenki, od toga se najviše 3 znamenke nalaze iza decimalne točke
 - razlikuje se od **float** ili **double** tipa podatka u jeziku C
 - ako se za pohranu broja 1.3 koristi tip podatka DECIMAL(2,1), broj će biti pohranjen **bez numeričke pogreške**
 - ako se za pohranu broja 1.3 koristi tip podatka **float** u jeziku C, u memoriji će se zapravo pohraniti broj 1.2999999523162842 (num. pogreška zbog karakteristika IEEE-754 formata pohrane)

Konstante - primjer za za DECIMAL(7, 2):

5	8.1	-12345.67	0	NULL
---	-----	-----------	---	------

SQL - Tipovi podataka

■ DATE

- podaci ovog tipa se uvijek prikazuju u obliku datuma (npr. 18.11.2006). Interno je podatak predstavljen brojem dana proteklih od 31.12.1899. Ovaj tip podatka omogućava korištenje sljedećih operacija zbrajanja i oduzimanja:

- **dat1 – dat2**

rezultat je podatak tipa INTEGER - broj dana proteklih između *dat2* i *dat1*

- **dat + cijeliBroj**

rezultat je podatak tipa DATE - izračunava koji datum je *cijeliBroj* dana nakon dana *dat*

- **dat – cijeliBroj**

rezultat je podatak tipa DATE - izračunava koji datum je *cijeliBroj* dana prije dana *dat*

Konstante:

'17.2.2007'

'16.07.1969'

NULL

NULL vrijednost

clanoviKnjiznice

mbr	ime	prez	pbr	datRodj	adresa	zanimanje
100	Maja	Novak	10000	01.5.2001	Ilica 1	NULL
105	Ivo	Kolar	21000	12.3.1973	NULL	odvjetnik
107	James	Bond	NULL	NULL	NULL	tajni agent

nije primjenjivo
(vidi datum rođenja)

nedostupno

trenutno nepoznato

- Način na koji se NULL vrijednost prikazuje korisniku ovisi o programskom alatu koji se koristi. Npr:

The screenshot shows a SQL editor interface with the following details:

- Toolbar:** Various icons for database management, including properties, new query, save, and execute.
- Properties Tab:** Shows the current connection details: Database: studadmin, Server: ifx01, Host: 192.168.40.128:9088, User: bpadmin.
- Query Window:** Displays the SQL command: `SELECT * FROM clanoviKnjiznice`.
- Result Grid:** Shows the data from the 'clanoviKnjiznice' table with the following rows:

mbr	ime	prez	pbr	datrodj	adresa	zanimanje
1	100	Maja	Novak	10000	01.05.2001	Ilica 1
2	105	Ivo	Kolar	21000	12.03.1973	NULL
3	107	James	Bond	NULL	NULL	tajni agent

Fizička pohrana NULL vrijednosti

- NULL vrijednost se interno pohranjuje drugačije od bilo koje druge dopuštene vrijednosti (nije 0, nije 0.0, nije prazan niz, ...)
- Primjer:
 - -32768 se interno koristi za prikaz NULL vrijednost kada se radi o vrijednosti atributa ili variable tipa SMALLINT. Zato je dopušteni raspon vrijednosti za taj tip samo [-32767, 32767]
 - -2147483648 se interno koristi za prikaz NULL vrijednosti kada se radi o vrijednosti atributa ili variable tipa INTEGER
- Interni prikaz NULL vrijednosti je za korisnika nevažan - NULL vrijednost je neovisna od tipa podatka kojeg predstavlja. Bez obzira na tip podatka, uvijek se koristi "konstanta" **NULL**

```
INSERT INTO mjesto VALUES (10000, 'Zagreb', -32768);
```

```
INSERT INTO mjesto VALUES (10000, 'Zagreb', NULL);
```

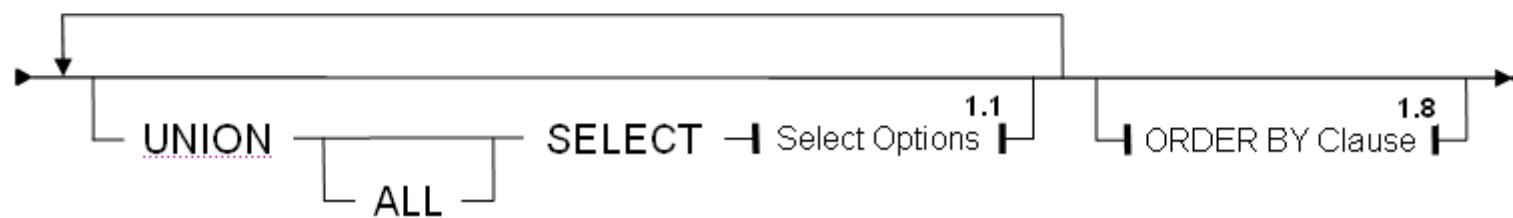


SELECT Statement

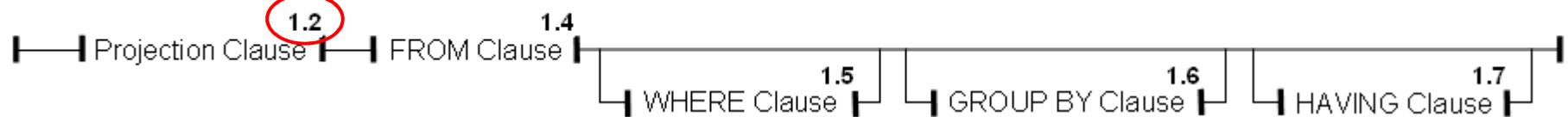
1. SELECT Statement

► SELECT → Select Options 1.1

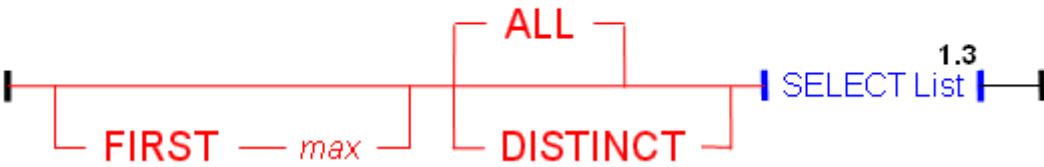
■ Sintaksni dijagrami



1.1. SELECT Options



1.2. Projection Clause



Projection Clause

- Primjeri:

student

matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
105	Kolar	52000
107	Ban	10000

```
SELECT ALL prez  
      , postbr  
  FROM student;
```

=

```
SELECT prez  
      , postbr  
  FROM student;
```

prez	postBr
Kolar	52000
Horvat	10000
Kolar	52000
Ban	10000

```
SELECT DISTINCT prez  
      , postbr  
  FROM student;
```

prez	postBr
Kolar	52000
Horvat	10000
Ban	10000

```
SELECT FIRST 2 *  
  FROM student;
```

matBr	prez	postBr
102	Horvat	10000
105	Kolar	52000

Ne zna se koje dvije n-torce
će se dobiti kao "prve dvije" -
poredak n-torki u relaciji (niti u
SQL tablici) nije definiran

Projection Clause

- Primjeri:

student

matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
105	Kolar	52000
107	Ban	10000

```
SELECT FIRST 2 DISTINCT prez  
FROM student;
```

prez
Horvat
Ban

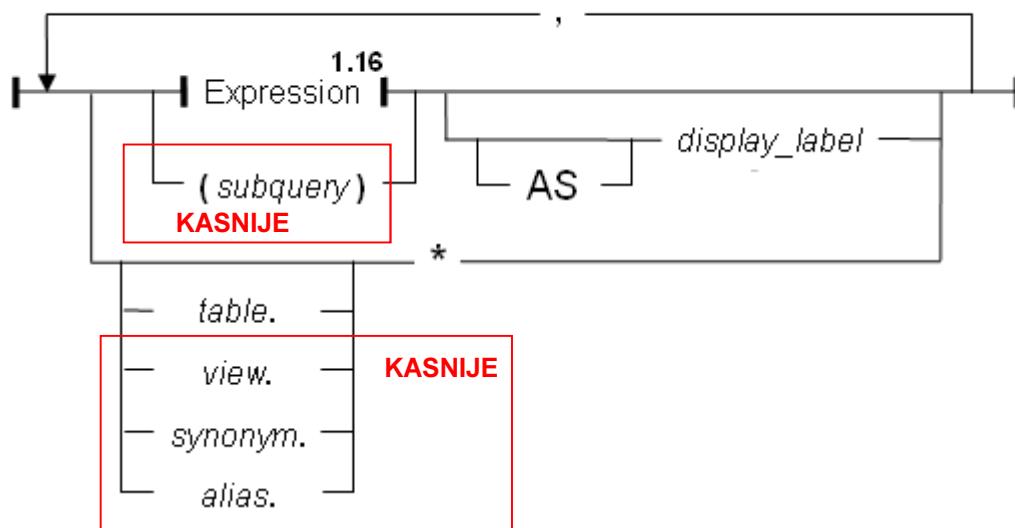
Još jednom: ne zna se koje su to "prve dvije" n-torke

```
SELECT FIRST 100 *  
FROM student;
```

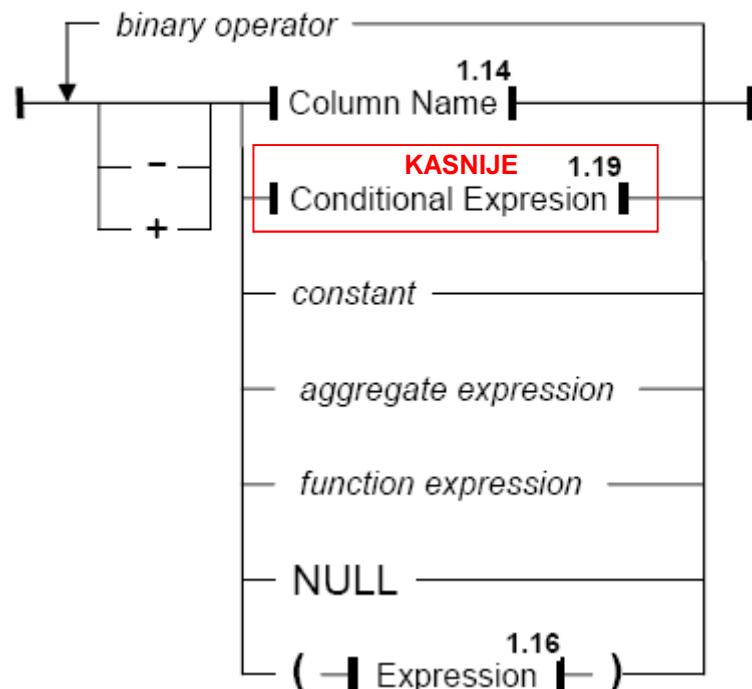
matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
105	Kolar	52000
107	Ban	10000

SELECT List

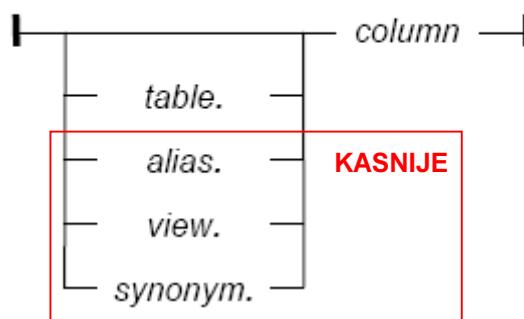
1.3 SELECT List



1.16. Expression



1.14. Column Name



SELECT List

- Primjer:

mjesto

pbr	nazMjesto	sifZup
42000	Varaždin	7
52100	Pula	4

```
SELECT mjesto.pbr, *, pbr, mjesto.*  
FROM mjesto
```

pbr	pbr	nazMjesto	sifZup	pbr	pbr	nazMjesto	sifZup
42000	42000	Varaždin	7	42000	42000	Varaždin	7
52100	52100	Pula	4	52100	52100	Pula	4

U ovom primjeru rezultat nije relacija!

Izraz (*Expression*)

- Unarni operatori

+ -

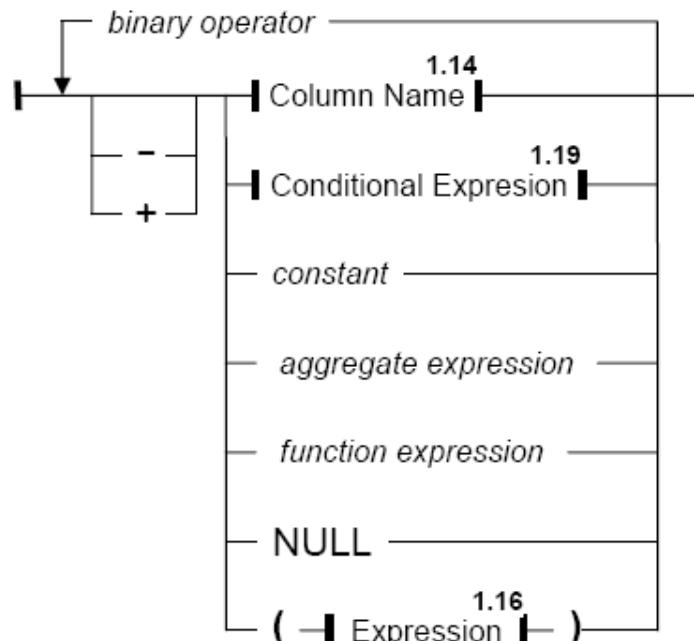
- Binarni operatori

+ - * /

| | ulančavanje nizova znakova

(nadovezivanje, konkatenacija)

1.16. Expression



- Redoslijed obavljanja operacija u složenim izrazima određuje se prema istim pravilima kao u programskom jeziku C (implicitni redoslijed obavljanja se može promijeniti upotrebom okruglih zagrada)
- Konverzija tipova podataka tijekom evaluacije izraza obavlja se prema sličnim pravilima kao u programskom jeziku C

Izraz (primjeri)

- unarni, binarni operatori i konstante

```
CREATE TABLE bodovi (
    mbr      INTEGER
,  ime      CHAR(10)
,  prez     CHAR(10)
,  bodLab   INTEGER
,  bodMI    DECIMAL(4,1)) ;
```

```
SELECT mbr,
       bodLab + bodMI,
       (bodLab + bodMI) / 100
  FROM bodovi;
```

```
SELECT mbr, - bodLab
  FROM bodovi;
```

```
SELECT mbr, ime || prez
  FROM bodovi;
```

```
SELECT mbr || '-' || ime
  FROM bodovi;
```

bodovi

mbr	ime	prez	bodLab	bodMI
100	Ana	Novak	12	67.2
107	Ivo	Ban	17	54.3

mbr	(expression)	(expression)
100	79.2	0.792
107	71.3	0.713

mbr	(expression)
100	-12
107	-17

mbr	(expression)
100	Ana Novak
107	Ivo Ban

(expression)
100-Ana
107-Ivo

Funkcije (*function expression*)

- ABS
- MOD
- ROUND
- SUBSTRING
- UPPER
- LOWER
- TRIM
- CHAR_LENGTH
- OCTET_LENGTH
- MDY
- DAY
- MONTH
- YEAR
- WEEKDAY
- TODAY
- USER

Funkcije (*function expression*)

- **ABS (*num_expression*)**

- računa absolutnu vrijednost izraza

num_expression – mora biti numerički tip podatka (INTEGER, DECIMAL, FLOAT, ...)

rezultat funkcije – tip podatka ovisi o tipu podatka ulaznog argumenta

- **MOD (*dividend, divisor*)**

- računa ostatak cijelobrojnog dijeljenja djeljenika i djelitelja
(djelitelj ne smije biti 0)
 - pri računanju uzima se samo cijelobrojni dio argumenata

dividend (djeljenik) – numerički tip podatka (INTEGER, DECIMAL, FLOAT, ...)

divisor (djelitelj) – numerički tip podatka (INTEGER, DECIMAL, FLOAT, ...)

rezultat funkcije – cijeli broj

Funkcije (*function expression*)

- **ROUND (*expression[, rounding_factor*)**
 - zaokružuje vrijednost izraza (*expression*)
 - ako se ne navede *rounding_factor*, uzima se da je njegova vrijednost 0

expression (*izraz koji se zaokružuje*) –
numerički tip podatka (INTEGER, DECIMAL, FLOAT, ...)

rounding_factor (*preciznost na koju se vrši zaokruživanje*) –
cjelobrojni tip podatka

rezultat funkcije – tip podatka ovisi o tipu podatka ulaznog argumenta (*expression*)

Funkcije (*function expression*)

- **SUBSTRING (*source_string FROM start_position [FOR length]*)**
 - vraća podniz zadanog niza
 - ako se *length* ne navede vraća se podniz koji počinje na *start_position*, a završava gdje i niz *source_string*

source_string – zadani niz čiji se podniz traži funkcijom

mora biti izraz tipa niza znakova

start_position – broj koji predstavlja poziciju prvog znaka podniza u zadanim nizu

source_string;

mora biti izraz cijelobrojnog tipa

length(duljina) – broj znakova koje funkcija treba vratiti počevši od *start_position*;

mora biti izraz cijelobrojnog tipa

Funkcije (*function expression*)

- **UPPER (expression)**

- sva mala slova (a-z) koja se pojavljuju u zadanom nizu *expression* zamjenjuje odgovarajućim velikim slovima (A-Z)

- **LOWER (expression)**

- sva velika slova (A-Z) koja se pojavljuju u zadanom nizu *expression* zamjenjuje odgovarajućim malim slovima (a-z)

expression – zadani niz nad kojim se vrši pretvorba slova
mora biti izraz tipa niza znakova

Funkcije (*function expression*)

- **TRIM(*source_expression*)**
 - funkcija vraća niz znakova koji nastaje tako da se s početka i kraja niza *source_expression* izbace sve praznine

expression – zadani niz iz kojeg funkcija izbacuje praznine
mora biti izraz tipa niza znakova

Funkcije (*function expression*)

- **CHAR_LENGTH(*expression*)**
 - funkcija vraća broj znakova u zadanom nizu *expression* uključujući i prateće praznine
- **OCTET_LENGTH(*expression*)**
 - funkcija vraća broj byte-ova zadatog niza *expression* uključujući i prateće praznine

expression – mora biti izraz tipa niza znakova

Funkcije (*function expression*)

- **USER**
 - funkcija vraća *login* korisnika koji je trenutno prijavljen za rad sa bazom podataka

- **TODAY**
 - funkcija vraća današnji datum (dobiven iz operacijskog sustava)

Funkcije (*function expression*)

- **MDY(*month*, *day*, *year*)**
 - funkcija vraća varijablu tipa DATE, odnosno izračunava datum iz tri INTEGER variable koje predstavljaju dan, mjesec i godinu

<i>month</i>	– broj koji predstavlja broj mjeseca mora biti cijeli broj iz intervala [1,12]
<i>day</i>	– broj koji predstavlja redni broj dana u mjesecu mora biti cijeli broj veći od 0 i manji od broja dana u određenom mjesecu
<i>year</i>	– broj koji predstavlja godinu mora biti četveroznamenkasti broj cjelobrojnog tipa (ne može se koristiti dvoznamenkasta skraćenica)

Funkcije (*function expression*)

- **DAY(*date_expression*)**
 - funkcija vraća redni broj dana u mjesecu za zadani datum
- **MONTH(*date_expression*)**
 - funkcija vraća redni broj mjeseca za zadani datum
- **YEAR(*date_expression*)**
 - funkcija vraća redni broj godine za zadani datum
- **WEEKDAY(*date_expression*)**
 - funkcija vraća redni broj dana u tjednu za zadani datum
(0 – nedjelja, 1 – ponedjeljak, 2 – utorak, itd...)

date_expression – izraz tipa DATE

Funkcije (primjeri) – matematičke funkcije

```
CREATE TABLE upl_ispl (
    rbr      INTEGER
,   racun    INTEGER
,   datum    DATE
,   iznos    DECIMAL(9,2));
```

```
SELECT rbr, ABS(iznos)
  FROM upl_ispl;
```

```
SELECT rbr, ROUND(iznos, 1)
  FROM upl_ispl;
```

```
SELECT rbr, MOD(iznos, 10)
  FROM upl_ispl;
```

upl_ispl

rbr	racun	datum	iznos
1	123456	22.02.2007	-120.00
2	878341	23.02.2007	173.47

rbr	(expression)
1	120.00
2	173.47

Ispisuje apsolutne vrijednosti iznosa

rbr	(expression)
1	-120.0
2	173.5

Ispisuje iznose zaokružene na jednu decimalu

rbr	(expression)
1	0
2	3

Ispisuje ostatak dijeljenja iznosa sa 10

Funkcije (primjeri) – funkcije s nizovima

```
CREATE TABLE student (
    jmbag CHAR(10)
, ime NCHAR(25)
, prezime NCHAR(25));
```

student

jmbag	ime	prezime
0036368145	Tomislav	Babić
0036369296	Linda	Jurić

Ispisuje jmbag i inicijale studenata

```
SELECT jmbag
, SUBSTRING(ime FROM 1 FOR 1) || '.' ||
SUBSTRING(prezime FROM 1 FOR 1) || '.'
FROM student;
```

jmbag	(expression)
0036368145	T.B.
0036369296	L.J.

Ispisuje imena velikim slovima, a prezimena malim slovima

```
SELECT UPPER(ime)
, LOWER(prezime)
FROM student;
```

(expression)	(expression)
TOMISLAV	babić
LINDA	jurić

Funkcije (primjeri) – funkcije s nizovima

```
CREATE TABLE student (
    jmbag CHAR(10)
, ime NCHAR(25)
, prezime NCHAR(25));
```

student

jmbag	ime	prezime
0036368145	Tomislav	Ban
0036369296	Linda	Kekez

Ispisuje imena studenata iz kojih su izbačene praznine

```
SELECT ime
, TRIM(ime)
FROM student;
```

ime	(expression)
Tomislav	Tomislav
Linda	Linda

```
SELECT ime || prezime
, TRIM(ime || prezime)
FROM student;
```

(expression)	(expression)
Tomislav	Ban
Linda	Kekez

Funkcije (primjeri) – funkcije s nizovima

```
CREATE TABLE student (
    jmbag CHAR(10)
, ime NCHAR(25)
, prezime NCHAR(25));
```

student

jmbag	ime	prezime
0036368145	Tomislav	Božanić
0036369296	Linda	Kekez

Ispisuje korisničko ime i broj znakova koji ga čine

```
SELECT USER
, CHAR_LENGTH(USER)
FROM student;
```

(expression)	(expression)
badmin	7
badmin	7

Ispisuje broj znakova u imenu i broj znakova u imenu iz kojeg su izbačene praznine

```
SELECT CHAR_LENGTH(ime)
, CHAR_LENGTH(TRIM(ime))
FROM student;
```

(expression)	(expression)
25	8
25	5

Ispisuje broj znakova i broj bajtova koji čine prezime iz kojeg su izbačene praznine

```
SELECT CHAR_LENGTH(TRIM(prezime))
, OCTET_LENGTH(TRIM(prezime))
FROM student;
```

(expression)	(expression)
7	9
5	5

Zbog utf8:
ž-2 okteta
ć-2 okteta

Funkcije (primjeri) – funkcije s datumom

```
CREATE TABLE nastavnik (
    sifNastavnik      INTEGER
,   datumZaposlenOd   DATE
,   datumZaposlenDo    DATE) ;
```

nastavnik

sifNastavnik	datumZaposlenOd	datumZaposlenDo
1	22.01.1995	20.02.2007
2	01.06.2004	01.03.2007

Napomena: pretpostavka je da se sljedeći upit izveo dana 27.02.2007.

Broj dana koji je protekao nakon prestanka zaposlenja nastavnika

```
SELECT sifNastavnik
      , TODAY - datumZaposlenDo
  FROM nastavnik;
```

sifNastavnik	(expression)
1	7
2	-2

Ispisuje dan, mjesec i godinu datuma zaposlenja nastavnika

```
SELECT DAY(datumZaposlenOd)
      , MONTH(datumZaposlenOd)
      , YEAR(datumZaposlenOd)
  FROM nastavnik;
```

(expression)	(expression)	(expression)
22	1	1995
1	6	2004

Funkcije (primjeri) – funkcije s datumom

```
CREATE TABLE nastavnik (
    sifNastavnik      INTEGER
,   datumZaposlenOd   DATE
,   datumZaposlenDo    DATE) ;
```

nastavnik

sifNastavnik	datumZaposlenOd	datumZaposlenDo
1	22.01.1995	20.02.2007
2	01.06.2004	01.03.2007

Ispisuje redni broj dana u tjednu datuma prestanka zaposlenja nastavnika

```
SELECT sifNastavnik
      , WEEKDAY(datumZaposlenDo)
    FROM nastavnik;
```

sifNastavnik	(expression)
1	2
2	4

Ispisuje datum koji odgovara sljedećem danu nakon prestanka zaposlenja nastavnika

~~```
SELECT MDY(MONTH(datumZaposlenDo)
 , DAY(datumZaposlenDo) +1
 , YEAR(datumZaposlenDo))
 FROM nastavnik;
```~~

| (expression) |
|--------------|
| 21.02.2007   |
| 02.03.2007   |

```
SELECT datumZaposlenDo+1
 FROM nastavnik;
```

# Funkcije i NULL vrijednosti

---

- Neka je binarni operator  $\alpha \in \{ +, -, *, /, || \}$ , a X i Y su izrazi
  - ako jedan ili oba operanda X, Y poprimaju NULL vrijednost, tada je rezultat izraza X  $\alpha$  Y također NULL vrijednost
- Neka je unarni operator  $\beta \in \{ +, - \}$ , a X je izraz
  - ako operand X poprima NULL vrijednost, tada je rezultat izraza  $\beta$  X također NULL vrijednost
- Slično vrijedi i za funkcije
  - ako se kao jedan ili više argumenata funkcije zada NULL vrijednost, rezultat funkcije će također biti NULL vrijednost

# Funkcije i NULL vrijednosti (primjer)

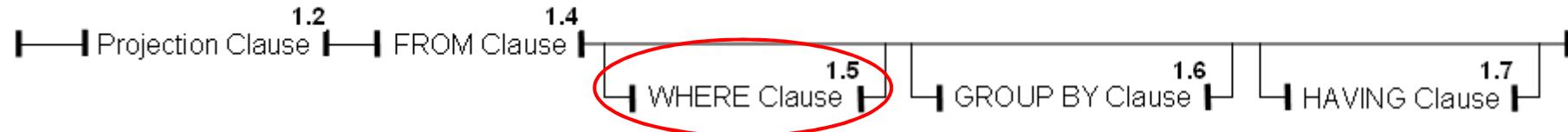
| bodovi | mbr | prez  | bodLab |
|--------|-----|-------|--------|
|        | 101 | Novak | 12     |
|        | 103 | Ban   | NULL   |
|        | 107 | NULL  | 21     |
|        | 109 | Kolar | NULL   |

```
SELECT mbr
 , MOD(bodLab, 10) AS ostatak
 , SUBSTRING(prez FROM 1 FOR 2) AS podniz
 FROM bodovi;
```

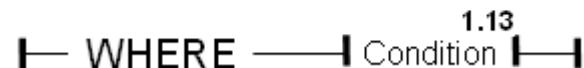
| mbr | ostatak | podniz |
|-----|---------|--------|
| 101 | 2       | No     |
| 103 | NULL    | Ba     |
| 107 | 1       | NULL   |
| 109 | NULL    | Ko     |

# WHERE Clause

## 1.1. SELECT Options



## 1.5. WHERE Clause

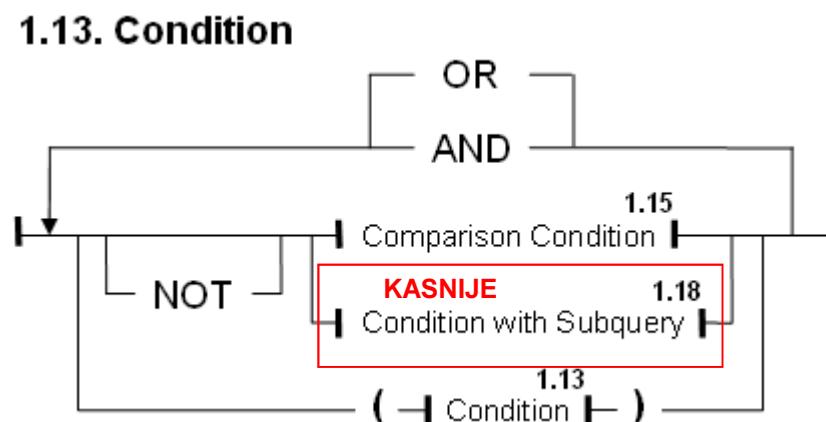


- Vrijednosti svake n-torke iz relacije *table* se uvrštavaju u *Condition* (a to je u stvari predikat). Ako je dobiveni sud istinit (*true*), n-torka se pojavljuje u rezultatu
- Mogući rezultati izračunavanja uvjeta: *true*, *false*, *unknown*

# Condition (ponavljanje)

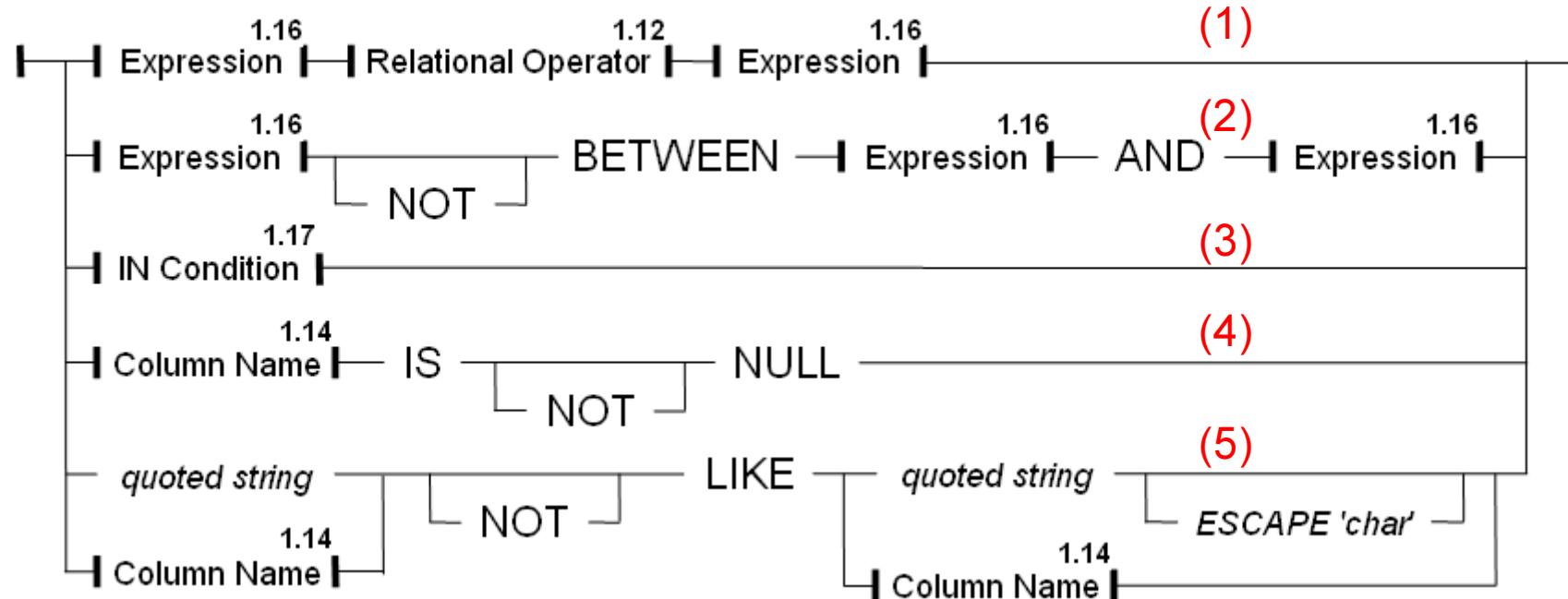
- `SELECT SELECT List FROM table [WHERE Condition]`
- Uvjet (*Condition*) se sastoji od operanada i operatora
  - operandi su:
    - imena atributa iz relacije *table*
    - konstante
  - operatori su:
    - operatori usporedbe: `<` `<=` `=` `<>` `>` `>=`
    - logički operatori: `AND` `OR` `NOT`

- SQL omogućava dodatne oblike za opisivanje uvjeta



# Uvjet usporedbe (Comparison Condition)

## 1.15. Comparison Condition

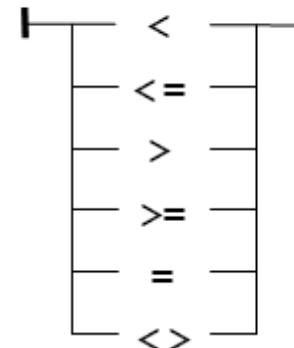


# Uvjet usporedbe (*Comparison Condition*) (1)

1.16 Expression      1.12 Relational Operator      1.16 Expression

| student | matBr | ime  | prez   | postBr |
|---------|-------|------|--------|--------|
|         | 100   | Ivan | Kolar  | 52000  |
|         | 102   | Ana  | Horvat | 10000  |
|         | 105   | Jura | NULL   | 21000  |

## 1.12. Relational Operator



```
SELECT * FROM student
WHERE prez <> 'Kolar' ;
```

| matBr | ime | prez   | postBr |
|-------|-----|--------|--------|
| 102   | Ana | Horvat | 10000  |

# Uvjet usporedbe (*Comparison Condition*) (2)



| stanjeSklad | sifArt | minS | maxS | stanje |
|-------------|--------|------|------|--------|
| 1           | 1      | 10   | 50   | 50     |
| 2           | 2      | 20   | 60   | 30     |
| 3           | 3      | 10   | 80   | 5      |
| 4           | 4      | NULL | 10   | 15     |
| 5           | 5      | 10   | 20   | NULL   |

```
SELECT * FROM stanjeSklad
WHERE stanje BETWEEN minS AND maxS;
```

| sifArt | minS | maxS | stanje |
|--------|------|------|--------|
| 1      | 10   | 50   | 50     |
| 2      | 20   | 60   | 30     |

```
SELECT * FROM stanjeSklad
WHERE stanje NOT BETWEEN minS AND maxS;
```

| sifArt | minS | maxS | stanje |
|--------|------|------|--------|
| 3      | 10   | 80   | 5      |
| 4      | NULL | 10   | 15     |

# Uvjet usporedbe (*Comparison Condition*) (3)

## 1.17. IN Condition



```
SELECT * FROM student
WHERE prez IN ('Kolar', 'Horvat');
```

| student |        |
|---------|--------|
| matBr   | prez   |
| 100     | Kolar  |
| 102     | Horvat |
| 103     | Novak  |
| 105     | Horvat |
| 107     | NULL   |
| 109     | Ban    |

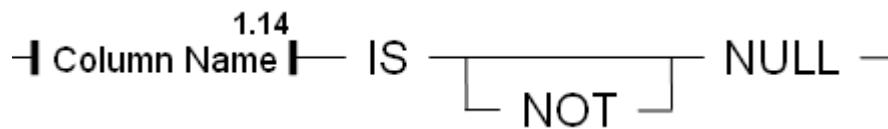
| matBr | prez   |
|-------|--------|
| 100   | Kolar  |
| 102   | Horvat |
| 105   | Horvat |

```
SELECT * FROM student
WHERE prez NOT IN ('Kolar', 'Horvat');
```

| matBr | prez  |
|-------|-------|
| 103   | Novak |
| 109   | Ban   |

- ako *Expression* ima vrijednost NULL, tada je rezultat logička vrijednost *unknown*, bez obzira na vrijednosti navedene u skupu

# Uvjet usporedbe (*Comparison Condition*) (4)



| student | matBr | prez   | postBr |
|---------|-------|--------|--------|
|         | 100   | Kolar  | 52000  |
|         | 102   | Horvat | 10000  |
|         | 105   | Novak  | NULL   |
|         | 107   | Ban    | 10000  |

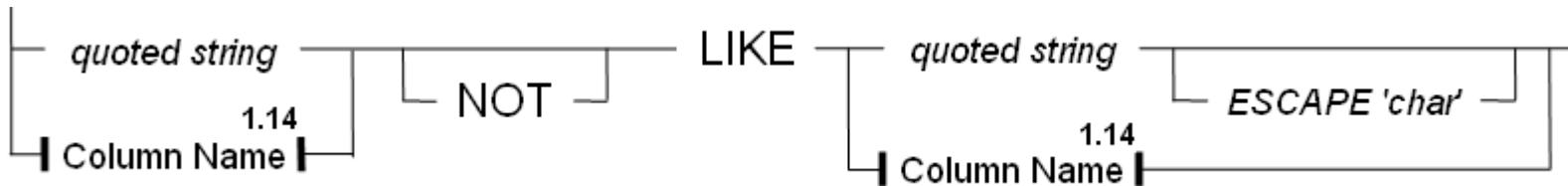
```
SELECT * FROM student
WHERE postBr IS NULL;
```

```
SELECT * FROM student
WHERE postBr IS NOT NULL;
```

| matBr | prez  | postBr |
|-------|-------|--------|
| 105   | Novak | NULL   |

| matBr | prez   | postBr |
|-------|--------|--------|
| 100   | Kolar  | 52000  |
| 102   | Horvat | 10000  |
| 107   | Ban    | 10000  |

## Uvjet usporedbe (*Comparison Condition*) (5)



- služi za ispitivanje zadovoljava li (ili ne zadovoljava) vrijednost atributa ili znakovna konstanta zadani uzorak (*pattern*)
- mogu se koristiti sljedeći *wildcard* znakovi:
  - znak **%** zamjenjuje bilo koju kombinaciju znakova (0 ili više znakova)
  - znak **\_** zamjenjuje točno jedan znak

# Uvjet usporedbe (Comparison Condition) (5)

| osoba |         |
|-------|---------|
| matBr | ime     |
| 1     | Matija  |
| 2     | Metka   |
| 3     | Matilda |
| 4     | Ratkec  |
| 5     | Marko   |
| 6     | Ivan    |

```
SELECT * FROM osoba
WHERE ime LIKE 'M%';
```

| matBr | ime     |
|-------|---------|
| 1     | Matija  |
| 2     | Metka   |
| 3     | Matilda |
| 5     | Marko   |

```
SELECT * FROM osoba
WHERE ime LIKE 'Mat%';
```

| matBr | ime     |
|-------|---------|
| 1     | Matija  |
| 3     | Matilda |

```
SELECT * FROM osoba
WHERE ime LIKE 'Ma_k%';
```

| matBr | ime   |
|-------|-------|
| 5     | Marko |

```
SELECT * FROM osoba
WHERE ime LIKE '%tk_';
```

| matBr | ime   |
|-------|-------|
| 2     | Metka |

```
SELECT * FROM osoba
WHERE ime LIKE '%tk%';
```

| matBr | ime    |
|-------|--------|
| 2     | Metka  |
| 4     | Ratkec |

# Uvjet usporedbe (Comparison Condition) (5)

tekstovi

| rbr | tekst        |
|-----|--------------|
| 1   | deset %      |
| 2   | pet % kisika |
| 3   | nije pet     |
| 4   | nije_pet     |
| 5   | % i _        |

- znak *char* naveden iza ESCAPE služi za poništavanje specijalnog značenja znakova % ili \_ koji su navedeni neposredno iza znaka *char*

```
SELECT * FROM tekstovi
WHERE tekst LIKE '#%%'
 ESCAPE '#';
```

| rbr | tekst |
|-----|-------|
| 5   | % i _ |

```
SELECT * FROM tekstovi
WHERE tekst LIKE '%$%'
 ESCAPE '$';
```

| rbr | tekst   |
|-----|---------|
| 1   | deset % |

```
SELECT * FROM tekstovi
WHERE tekst LIKE '%$_%'
 ESCAPE '$';
```

| rbr | tekst        |
|-----|--------------|
| 1   | deset %      |
| 2   | pet % kisika |
| 5   | % i _        |

```
SELECT * FROM tekstovi
WHERE tekst LIKE '%!_pet'
 ESCAPE '!' ;
```

| rbr | tekst    |
|-----|----------|
| 4   | nije_pet |

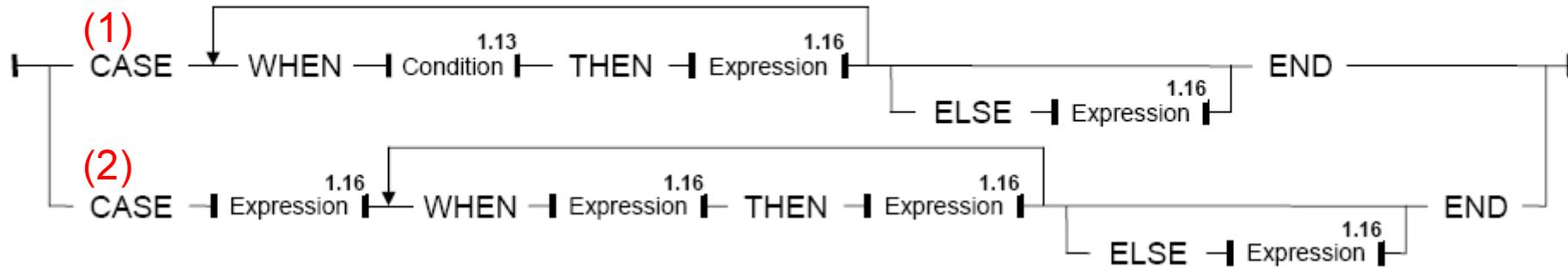
## Uvjet usporedbe (*Comparison Condition*) (5)

---

- $x \text{ LIKE } 'AB\%'$       *true za svaki x koji započinje s AB*
- $x \text{ LIKE } '%AB'$       *true za svaki x koji završava s AB*
- $x \text{ LIKE } '%%AB'$       *true za svaki x koji završava s AB*
  
- $x \text{ LIKE } 'AB\%CD'$       *true za svaki x koji započinje s AB i završava s CD*
- $x \text{ LIKE } '%AB\%'$       *true za svaki x koji sadrži AB*
- $x \text{ LIKE } '_AB'$       *true za svaki x duljine 3 znaka koji završava s AB*
- $x \text{ LIKE } '__AB'$       *true za svaki x duljine 4 znaka koji završava s AB*
- $x \text{ LIKE } 'AB__'$       *true za svaki x duljine 4 znaka koji započinje s AB*
- $x \text{ LIKE } '_AB\%'$       *true za svaki x koji započinje bilo kojim znakom, nastavlja se sa znakovima AB, te završava s bilo kojim znakovima*

# Uvjetni izraz (*Conditional Expression*)

## Conditional Expression



- 1. oblik izraza je sličan **if • else if • else** naredbi za višestranu selekciju u programskom jeziku C
- 2. oblik izraza je sličan **switch • case • default** naredbi za selekciju u programskom jeziku C
- pri čemu postoji bitna razlika:
  - C naredbama "odlučuje se" koje će se naredbe obaviti
  - SQL uvjetnim izrazom "odlučuje se" koja **vrijednost** predstavlja **rezultat** uvjetnog izraza

# Uvjetni izraz (*Conditional Expression*) (1)

```
SELECT *
 , CASE
 WHEN ocjena = 5 THEN 'izvrstan'
 WHEN ocjena = 4 THEN 'vrlo dobar'
 WHEN ocjena = 3 THEN 'dobar'
 WHEN ocjena = 2 THEN 'dovoljan'
 WHEN ocjena = 1 THEN 'nedovoljan'
 WHEN ocjena IS NULL THEN 'nepoznato'
 ELSE 'neispravno'
 END AS opis
 FROM ispit;
```

| ispit | matBr | ocjena |
|-------|-------|--------|
|       | 100   | 5      |
|       | 102   | 3      |
|       | 103   | 1      |
|       | 107   | NULL   |
|       | 109   | 6      |

| matBr | ocjena | opis       |
|-------|--------|------------|
| 100   | 5      | izvrstan   |
| 102   | 3      | dobar      |
| 103   | 1      | nedovoljan |
| 107   | NULL   | nepoznato  |
| 109   | 6      | neispravno |

- ako se više izraza uz WHEN izračuna kao *true*, rezultat izraza je *Expression* naveden uz prvi WHEN čiji se uvjet izračuna kao *true*
- ako se ELSE dio izraza ne navede, a niti jedan uvjet uz WHEN se ne izračuna kao *true*, tada je rezultat izraza NULL vrijednost

## Uvjetni izraz (*Conditional Expression*) (2)

```
SELECT *
 , CASE ocjena
 WHEN 5 THEN 'izvrstan'
 WHEN 4 THEN 'vrlo dobar'
 WHEN 3 THEN 'dobar'
 WHEN 2 THEN 'dovoljan'
 WHEN 1 THEN 'nedovoljan'
 ELSE 'neispravno'
 END AS opis
FROM ispit;
```

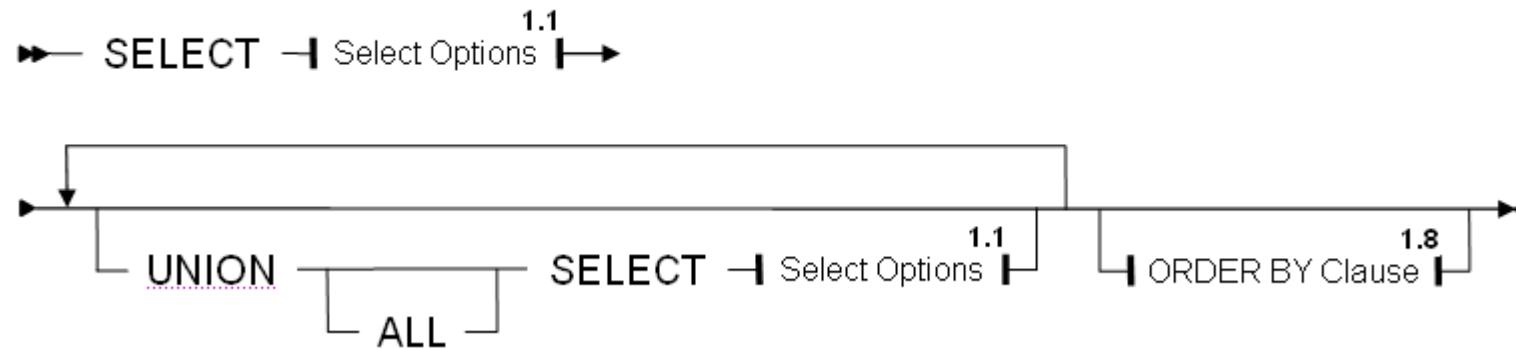
| ispit | matBr | ocjena |
|-------|-------|--------|
|       | 100   | 5      |
|       | 102   | 3      |
|       | 103   | 1      |
|       | 107   | NULL   |
|       | 109   | 6      |

| matBr | ocjena | opis       |
|-------|--------|------------|
| 100   | 5      | izvrstan   |
| 102   | 3      | dobar      |
| 103   | 1      | nedovoljan |
| 107   | NULL   | neispravno |
| 109   | 6      | neispravno |

- ako više izraza uz WHEN zadovoljava uvjet jednakosti, rezultat izraza je *Expression* naveden uz prvi WHEN koji zadovoljava uvjet
- ako se ELSE dio izraza ne navede, a niti jedan izraz ne zadovoljava uvjet jednakosti, tada je rezultat izraza NULL vrijednost

# Unija (*UNION*)

## 1. SELECT Statement



- *SELECT Statement* može se graditi od jednog ili više *SELECT dijelova*
- **UNION** - uz izbacivanje duplikata (kopija n-torki)
- **UNION ALL** - bez izbacivanja duplikata (kopija n-torki)
- imena stupaca (atributa rezultantne relacije) određuju se na temelju imena stupaca iz prvog navedenog *SELECT dijela*

# Unija (*UNION*)

- polozioMat  $\cup$  polozioProg  $\cup$  polozioDiglog

polozioMat

| mbr | imeSt | prezSt |
|-----|-------|--------|
| 100 | Ivan  | NULL   |
| 102 | Ana   | Novak  |
| 105 | Rudi  | Kolar  |
| 111 | Jura  | Horvat |

polozioProg

| mbr | ime  | prez  |
|-----|------|-------|
| 100 | Ivan | NULL  |
| 103 | NULL | Ban   |
| 105 | Rudi | Kolar |

polozioDiglog

| mbr | ime  | prez   |
|-----|------|--------|
| 102 | Ana  | Novak  |
| 103 | NULL | Ban    |
| 105 | Rudi | Kolar  |
| 111 | Jura | Horvat |

```
SELECT * FROM polozioMat
UNION
SELECT * FROM polozioProg
UNION
SELECT * FROM polozioDiglog;
```

| mbr | imeSt | prezSt |
|-----|-------|--------|
| 100 | Ivan  | NULL   |
| 102 | Ana   | Novak  |
| 103 | NULL  | Ban    |
| 105 | Rudi  | Kolar  |
| 111 | Jura  | Horvat |

# Unija (*UNION*)

- rezultat sljedeće naredbe nije relacija!

polozioMat

| mbr | ime  | prez   |
|-----|------|--------|
| 100 | Ivan | NULL   |
| 102 | Ana  | Novak  |
| 105 | Rudi | Kolar  |
| 111 | Jura | Horvat |

polozioProg

| mbr | imeSt | prezSt |
|-----|-------|--------|
| 100 | Ivan  | NULL   |
| 103 | NULL  | Ban    |
| 105 | Rudi  | Kolar  |

polozioDiglog

| mbr | imeSt | prezSt |
|-----|-------|--------|
| 102 | Ana   | Novak  |
| 103 | NULL  | Ban    |
| 105 | Rudi  | Kolar  |
| 111 | Jura  | Horvat |

```
SELECT * FROM polozioMat
UNION ALL
SELECT * FROM polozioProg
UNION ALL
SELECT * FROM polozioDiglog;
```

| mbr | ime  | prez   |
|-----|------|--------|
| 100 | Ivan | NULL   |
| 102 | Ana  | Novak  |
| 105 | Rudi | Kolar  |
| 111 | Jura | Horvat |
| 100 | Ivan | NULL   |
| 103 | NULL | Ban    |
| 105 | Rudi | Kolar  |
| 102 | Ana  | Novak  |
| 103 | NULL | Ban    |
| 105 | Rudi | Kolar  |
| 111 | Jura | Horvat |

# Unija (**UNION**)

- naredba je ispravna ako su korespondentni atributi istih tipova podataka (INTEGER-INTEGER, CHAR-CHAR, ...), ali odgovornost je korisnika (programera) voditi računa o unijskoj kompatibilnosti
- npr. sljedeća naredba će se obaviti, ali rezultat je besmislen

pecivo

| oznaka | naziv            |
|--------|------------------|
| ZE-33  | Žemlja s makom   |
| PR-3   | Perec sa sezamom |

zrakoplov

| oznaka | naziv         |
|--------|---------------|
| PR-3   | Piper J-3 Cub |
| B-747  | Boeing 747    |
| A-360  | Airbus 360    |

```
SELECT * FROM pecivo
UNION
SELECT * FROM zrakoplov;
```

| oznaka | naziv            |
|--------|------------------|
| ZE-33  | Žemlja s makom   |
| PR-3   | Perec sa sezamom |
| PR-3   | Piper J-3 Cub    |
| B-747  | Boeing 747       |
| A-360  | Airbus 360       |

Piper J-3 Cub

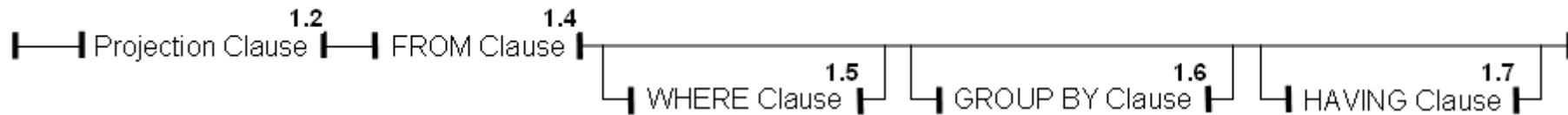


Perec sa sezamom

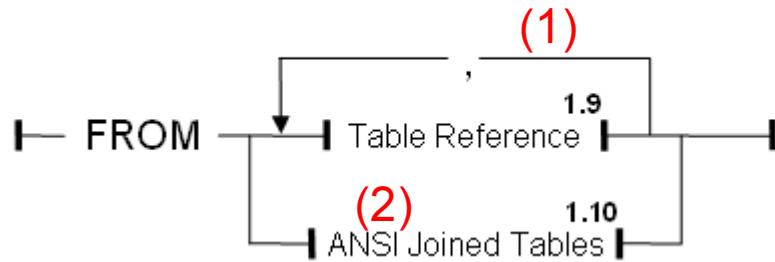


# FROM Clause

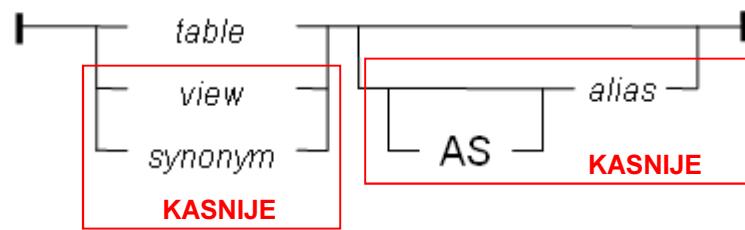
## 1.1. SELECT Options



## 1.4. FROM Clause



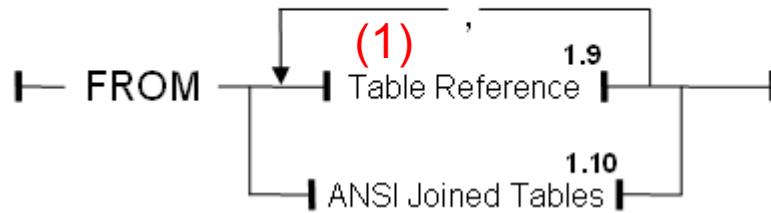
## 1.9. Table Reference



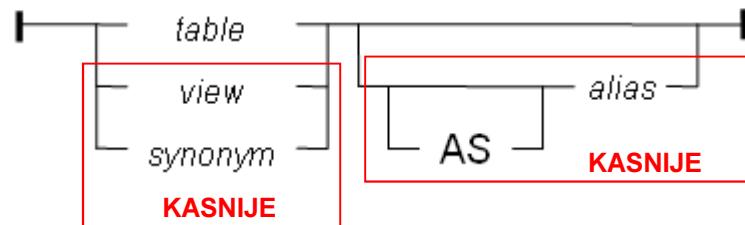
- (1) klasična sintaksa (*classical, comma-delimited*) za spajanje relacija
- (2) ANSI sintaksa za spajanje relacija

# FROM Clause (1)

## 1.4. FROM Clause



## 1.9. Table Reference



- klasična sintaksa (*classical, comma-delimited*) može se koristiti za obavljanje operacija:
  - Kartezijev produkt
  - spajanje uz uvjet i spajanje s izjednačavanjem
  - prirodno spajanje
- uvjeti spajanja se navode u WHERE dijelu SELECT naredbe, zajedno s eventualnim uvjetima selekcije (uvjeti spajanja i uvjeti selekcije se u tom slučaju povezuju logičkim operatorom AND)

# **FROM Clause (1)**

---

- Zadane su relacije:  $r (\{ A, B \})$     $s (\{ C, D \})$     $t (\{ D, E \})$

$r \times s$

```
SELECT *
 FROM r, s;
```

$r \triangleright\triangleleft s$   
 $A = C \wedge B \geq D$

```
SELECT *
 FROM r, s
 WHERE A = C
 AND B >= D;
```

$r \triangleright\triangleleft s$   
 $B = C$

```
SELECT *
 FROM r, s
 WHERE B = C;
```

$\sigma_{D>5}(r \triangleright\triangleleft s)$

```
SELECT *
 FROM r, s
 WHERE B = C
 AND D > 5;
```

## **FROM Clause (1)**

---

- $r(\{A, B\}) \quad s(\{C, D\}) \quad t(\{D, E\})$

$(r \times s) \bowtie t$

```
SELECT r.* , s.* , t.E
FROM r , s , t
WHERE s.D = t.D;
```

$(r \bowtie s) \bowtie t$

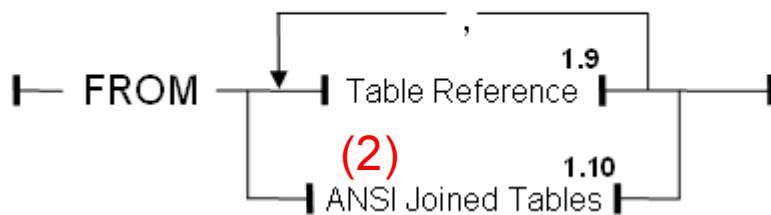
```
SELECT r.* , s.* , t.E
FROM r , s , t
WHERE s.D = t.D;
```

$\sigma_{C=100}(s \bowtie t)$

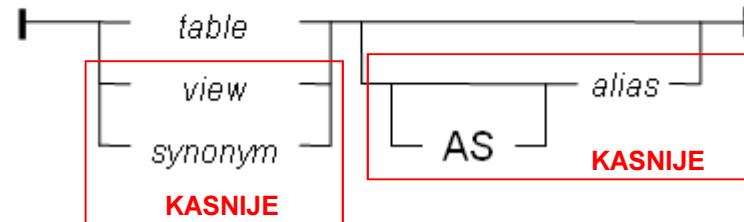
```
SELECT s.* , t.E
FROM s , t
WHERE s.D = t.D
AND C = 100;
```

# **FROM Clause (2)**

## 1.4. FROM Clause



## 1.9. Table Reference

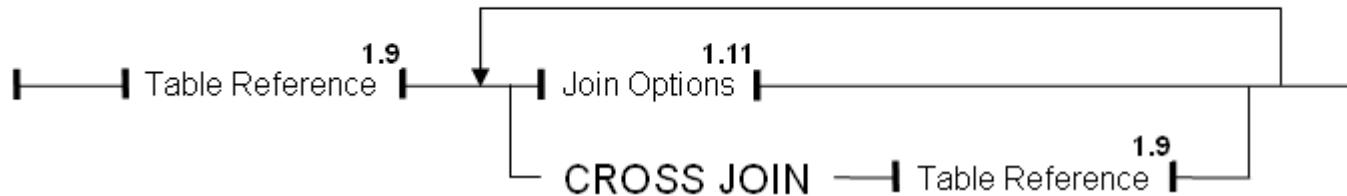


- ANSI sintaksa za spajanje relacija. Može se koristiti za obavljanje operacija:
  - Kartezijev produkt
  - spajanje uz uvjet i spajanje s izjednačavanjem
  - prirodno spajanje
  - vanjsko spajanje
    - vanjsko spajanje uz uvjet
    - vanjsko spajanje s izjednačavanjem
    - prirodno vanjsko spajanje

# **FROM Clause (2)**

---

## 1.10. ANSI Joined Tables



## 1.11. Join Options



- r CROSS JOIN s
- r INNER JOIN s ON uvjetSpajanja
- r LEFT OUTER JOIN s ON uvjetSpajanja
- r RIGHT OUTER JOIN s ON uvjetSpajanja
- r FULL OUTER JOIN s ON uvjetSpajanja

## ***FROM Clause (2)***

---

- Rezervirane riječi OUTER i INNER se smiju izostaviti:

**r INNER JOIN s**      ≡    **r JOIN s**

**r LEFT OUTER JOIN s**      ≡    **r LEFT JOIN s**

**r RIGHT OUTER JOIN s**      ≡    **r RIGHT JOIN s**

**r FULL OUTER JOIN s**      ≡    **r FULL JOIN s**

## *FROM Clause (2)*

- Zadane su relacije:  $r (\{ A, B \})$     $s (\{ C, D \})$     $t (\{ D, E \})$

$r \times s$

```
SELECT *
 FROM r CROSS JOIN s;
```

$r \triangleright\triangleleft s$   
 $A = C \wedge B \geq D$

```
SELECT *
 FROM r
 INNER JOIN s
 ON A = C AND B >= D;
```

$r \triangleright\triangleleft s$   
 $B = C$

```
SELECT *
 FROM r
 INNER JOIN s
 ON B = C;
```

$\sigma_{D>5}(r \triangleright\triangleleft s)$   
 $B = C$

```
SELECT *
 FROM r
 INNER JOIN s
 ON B = C
 WHERE D > 5;
```

## **FROM Clause (2)**

---

- $r(\{A, B\}) \quad s(\{C, D\}) \quad t(\{D, E\}) \quad p(\{E, F\})$

$(r \times s) \bowtie t$

```
SELECT r.* , s.* , t.E
FROM r
CROSS JOIN s
INNER JOIN t
ON s.D = t.D;
```

$(s \bowtie t) \bowtie p$

```
SELECT s.* , t.E , p.F
FROM s
INNER JOIN t
ON s.D = t.D
INNER JOIN p
ON t.E = p.E;
```

$\sigma_{C=100}(s \bowtie t)$

```
SELECT s.* , t.E
FROM s
INNER JOIN t
ON s.D = t.D
WHERE C = 100;
```

## **FROM Clause (2)**

---

- $r(\{A, B\}) \quad s(\{C, D\}) \quad t(\{D, E\}) \quad p(\{E, F\})$

$(r \times s)^* \triangleright \triangleleft t$

```
SELECT r.* , s.* , t.E
FROM r
CROSS JOIN s
LEFT OUTER JOIN t
ON s.D = t.D;
```

$(s^* \triangleright \triangleleft^* t) \triangleright \triangleleft^* p$

```
SELECT s.* , t.D AS D1 , p.*
FROM s
FULL OUTER JOIN t
ON s.D = t.D
RIGHT OUTER JOIN p
ON t.E = p.E;
```

$\sigma_{C=100}(s^* \triangleright \triangleleft t)$

```
SELECT s.* , t.E
FROM s
LEFT OUTER JOIN t
ON s.D = t.D
WHERE C = 100;
```

## **FROM Clause (2)**

- Ako se obavlja operacija spajanja i selekcija, uvjete spajanja treba navesti u ON dijelu, a uvjete selekcije treba navesti u WHERE dijelu SELECT naredbe
  - iako, u slučaju kada se ne koristi vanjsko spajanje, rezultat upita ne ovisi o tome je li uvjet selekcije naveden u ON ili WHERE dijelu naredbe

| student | matBr  | prez  | pbrSt |
|---------|--------|-------|-------|
| 101     | Kolar  | 10000 |       |
| 102     | Horvat | 21000 |       |

| mjesto | pbr    | nazMjesto |
|--------|--------|-----------|
| 10000  | Zagreb |           |
| 21000  | Split  |           |

```
SELECT *
 FROM student
 INNER JOIN mjesto
 ON pbrSt = pbr
 WHERE prez = 'Kolar';
```

```
SELECT *
 FROM student
 INNER JOIN mjesto
 ON pbrSt = pbr
 AND prez = 'Kolar';
```

- u ovom slučaju, oba upita daju isti rezultat

| matBr | prez  | pbrSt | pbr   | nazMjesto |
|-------|-------|-------|-------|-----------|
| 101   | Kolar | 10000 | 10000 | Zagreb    |

## FROM Clause (2)

- Ako se koristi vanjsko spajanje, navođenje uvjeta selekcije u ON dijelu umjesto WHERE dijelu može **bitno** utjecati na rezultat

| student | matBr | prez   | pbrSt |
|---------|-------|--------|-------|
|         | 101   | Kolar  | 10000 |
|         | 102   | Horvat | 21000 |

| mjesto | pbr   | nazMjesto |
|--------|-------|-----------|
|        | 10000 | Zagreb    |
|        | 21000 | Split     |

```
SELECT *
 FROM student
 LEFT OUTER JOIN mjesto
 ON pbrSt = pbr
 WHERE prez = 'Kolar';
```

- Tek nakon obavljenog spajanja prema uvjetu navedenom u ON dijelu naredbe, obavlja se selekcija n-torki prema uvjetu navedenom u WHERE dijelu naredbe

| matBr | prez  | pbrSt | pbr   | nazMjesto |
|-------|-------|-------|-------|-----------|
| 101   | Kolar | 10000 | 10000 | Zagreb    |

## FROM Clause (2)

- Ovdje je prikazan upit sličan prethodnom, ali u kojem je uvjet selekcije napisan na "pogrešnom" mjestu

| student | matBr  | prez  | pbrSt |
|---------|--------|-------|-------|
| 101     | Kolar  | 10000 |       |
| 102     | Horvat | 21000 |       |

| mjesto | pbr    | nazMjesto |
|--------|--------|-----------|
| 10000  | Zagreb |           |
| 21000  | Split  |           |

```
SELECT *
 FROM student
 LEFT OUTER JOIN mjesto
 ON pbrSt = pbr
 AND prez = 'Kolar' ;
```

- Ovdje će se pojaviti sve n-torke iz relacije student - uz one n-torke relacije student koje ne zadovoljavaju uvjet spajanja (uočite koji je uvjet spajanja ovdje naveden) dodat će se NULL vrijednosti

| matBr | prez   | pbrSt | pbr   | nazMjesto |
|-------|--------|-------|-------|-----------|
| 101   | Kolar  | 10000 | 10000 | Zagreb    |
| 102   | Horvat | 21000 | NULL  | NULL      |

## ***FROM Clause (2)***

---

- **Logički promatrano\***, kada se u upitu spajaju više od dvije relacije, redoslijed spajanja je s lijeva na desno: spajaju se prve dvije relacije, zatim se dobiveni rezultat spaja s trećom navedenom relacijom, zatim se dobiveni rezultat spaja s četvrtom navedenom relacijom, itd.
- (\*) konačni rezultat će sigurno odgovarati rezultatu koji bi se dobio kada bi se relacije spajale s lijeva na desno. Fizički promatrano, upit će se možda izvesti drugačijim redoslijedom, ali o tome brine dio SUBP-a koji se naziva optimizator upita

## ***FROM Clause (2)***

---

- ako se ne koristi vanjsko spajanje, redoslijed spajanja je ionako irelevantan, jer vrijedi:

$$( r_1 \bowtie r_2 ) \bowtie r_3 \quad \equiv \quad r_1 \bowtie ( r_2 \bowtie r_3 )$$

- ako se koristi vanjsko spajanje, redoslijed spajanja jest važan jer:

$$( r_1 * \bowtie r_2 ) \bowtie r_3 \quad \neq \quad r_1 * \bowtie ( r_2 \bowtie r_3 )$$

## FROM Clause (2)

| stud |        |       |
|------|--------|-------|
| mbr  | prez   | pbrSt |
| 101  | Horvat | 42000 |
| 102  | Novak  | 21000 |

| mjesto |           |          |
|--------|-----------|----------|
| pbr    | nazMjesto | sifZupMj |
| 42000  | Varaždin  | 7        |
| 21000  | Split     | NULL     |

| zupanija |             |
|----------|-------------|
| sifZup   | nazZup      |
| 7        | Varaždinska |
| 4        | Istarska    |

( stud \*  $\triangleright\triangleleft$  mjesto )  $\triangleright\triangleleft$  zupanija  
pbrSt=pbr                    sifZupMj=sifZup

```
SELECT stud.* , mjesto.* , zupanija.*
FROM stud
LEFT OUTER JOIN mjesto
ON pbrSt = pbr
INNER JOIN zupanija
ON sifZupMj = sifZup;
```

- prvo se spajaju relacije stud i mjesto, a zatim se dobiveni rezultat spaja s relacijom zupanija

| mbr | prez   | pbrSt | pbr   | nazMjesto | sifZupMj | sifZup | nazZup      |
|-----|--------|-------|-------|-----------|----------|--------|-------------|
| 101 | Horvat | 42000 | 42000 | Varaždin  | 7        | 7      | Varaždinska |

## FROM Clause (2)

stud \*▷◁ ( mjesto ▷◁ zupanija )  
pbrSt=pbr sifZupMj=sifZup

≡ ( mjesto ▷◁ zupanija ) ▷◁\* stud  
sifZupMj=sifZup pbrSt=pbr

da bismo izraz  
relacijske algebre mogli  
napisati u obliku SQL  
naredbe, napisat ćemo  
ga u drugaćijem obliku

```
SELECT stud.* , mjesto.* , zupanija.*
 FROM mjesto
 INNER JOIN zupanija
 ON sifZupMj = sifZup
 RIGHT OUTER JOIN stud
 ON pbrSt = pbr;
```

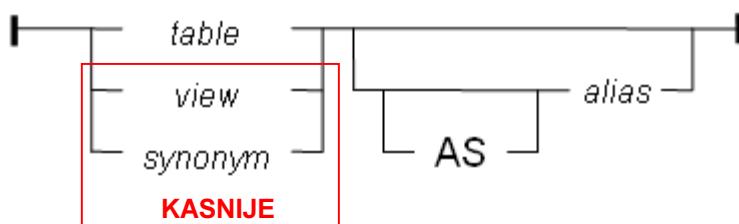
- prvo se spajaju relacije mjesto i zupanija, a zatim se  
dobiveni rezultat spaja s relacijom stud

| mbr | prez   | pbrSt | pbr   | nazMjesto | sifZupMj | sifZup | nazZup      |
|-----|--------|-------|-------|-----------|----------|--------|-------------|
| 101 | Horvat | 42000 | 42000 | Varaždin  | 7        | 7      | Varaždinska |
| 102 | Novak  | 21000 | NULL  | NULL      | NULL     | NULL   | NULL        |

# Preimenovanje relacija unutar upita

- relacija se unutar upita može preimenovati u *alias* ime
  - alias* ime je vidljivo samo unutar upita (ne utječe na stvarno ime relacije u bazi podataka)

## 1.9. Table Reference



- rezervirana riječ AS se smije ispuštiti
- na relaciju koja je u upitu dobila alias ime, moguće je referencirati se isključivo preko tog istog alias imena

```
SELECT nazMjesto, nazZupanija
 FROM mjesto AS town
 , zupanija AS county
 WHERE town.sifZupanija = county.sifZupanija;
```

```
SELECT nazMjesto, nazZupanija
 FROM mjesto AS town
 JOIN zupanija AS county
 ON town.sifZupanija = county.sifZupanija;
```

# Preimenovanje relacija unutar upita

- iako se preimenovanjem relacija može skratiti duljina teksta upita, u praksi se to **ne preporuča** jer upiti postaju manje razumljivi

```
SELECT o.jmbg, prezime, m.pbr, nazMjesto
 FROM osoba AS o
 , mjesto AS m
 , zaposlenje AS z1
 , zupanija AS z2
 WHERE o.jmbg = z1.jmbg
 AND o.pbr = m.pbr
 AND m.sifZup = z2.sifZup
 AND z2.nazZup = 'Varaždinska'
 AND z1.radnoMjesto = 'Dimnjačar'
```

- preimenovanje relacija unutar upita treba se koristiti onda kada se ista relacija pojavljuje u više uloga unutar istog upita

# Paralelno spajanje

| student | mbr | prez  | pbrRod | pbrStan |
|---------|-----|-------|--------|---------|
|         | 100 | Kolar | 10000  | 21000   |
|         | 102 | Novak | 21000  | 10000   |
|         | 103 | Ban   | 10000  | 10000   |

| mjesto | pbr   | nazMjesto |
|--------|-------|-----------|
|        | 10000 | Zagreb    |
|        | 21000 | Split     |

- Kako dobiti sljedeći rezultat:

| mbr | prez  | pbrRod | pbrStan | nazMjestoR |
|-----|-------|--------|---------|------------|
| 100 | Kolar | 10000  | 21000   | Zagreb     |
| 102 | Novak | 21000  | 10000   | Split      |
| 103 | Ban   | 10000  | 10000   | Zagreb     |

- To je lako:

```
SELECT student.* , mjesto.nazMjesto AS nazMjestoR
 FROM student
 , mjesto
 WHERE student.pbrRod = mjesto.pbr;
```

# Paralelno spajanje

---

student

| mbr | prez  | pbrRod | pbrStan |
|-----|-------|--------|---------|
| 100 | Kolar | 10000  | 21000   |
| 102 | Novak | 21000  | 10000   |
| 103 | Ban   | 10000  | 10000   |

mjesto

| pbr   | nazMjesto |
|-------|-----------|
| 10000 | Zagreb    |
| 21000 | Split     |

- Kako dobiti sljedeći rezultat:

| mbr | prez  | pbrRod | nazMjestoR | pbrStan | nazMjestoS |
|-----|-------|--------|------------|---------|------------|
| 100 | Kolar | 10000  | Zagreb     | 21000   | Split      |
| 102 | Novak | 21000  | Split      | 10000   | Zagreb     |
| 103 | Ban   | 10000  | Zagreb     | 10000   | Zagreb     |

# Paralelno spajanje

| student | mbr | prez  | pbrRod | pbrStan |
|---------|-----|-------|--------|---------|
|         | 100 | Kolar | 10000  | 21000   |
|         | 102 | Novak | 21000  | 10000   |
|         | 103 | Ban   | 10000  | 10000   |

| mjesto | pbr   | nazMjesto |
|--------|-------|-----------|
|        | 10000 | Zagreb    |
|        | 21000 | Split     |

```
SELECT mbr, prez
 , pbrRod, mjesto.nazMjesto AS nazMjestoR
 , pbrStan, mjesto.nazMjesto AS nazMjestoS
 FROM student
 , mjesto
 WHERE student.pbrRod = mjesto.pbr
 AND student.pbrStan = mjesto.pbr;
```

**NEISPRAVNO RJEŠENJE**

| mbr | prez | pbrRod | nazMjestoR | pbrStan | nazMjestoS |
|-----|------|--------|------------|---------|------------|
| 103 | Ban  | 10000  | Zagreb     | 10000   | Zagreb     |

- Upit **nije dobar** jer jednu n-torku iz relacije student pokušavamo spojiti s jednom n-torkom iz relacije mjesto uz sljedeći uvjet spajanja: vrijednost atributa pbrRod, te **istovremeno** i vrijednost atributa pbrStan iz relacije student su jednake vrijednosti atributa pbr iz relacije mjesto

# Paralelno spajanje

- Kad bismo načinili dvije kopije relacije mjesto: mjestoR i mjestoS, sa shemama i sadržajem jednakim relaciji mjesto:



```
SELECT mbr, prez
 , pbrRod, mjestoR.nazMjesto AS nazMjestoR
 , pbrStan, mjestoS.nazMjesto AS nazMjestoS
 FROM student, mjestoR, mjestoS
 WHERE student.pbrRod = mjestoR.pbr
 AND student.pbrStan = mjestoS.pbr;
```

- konačni rezultat je ispravan, ali radi se o vrlo lošem rješenju!

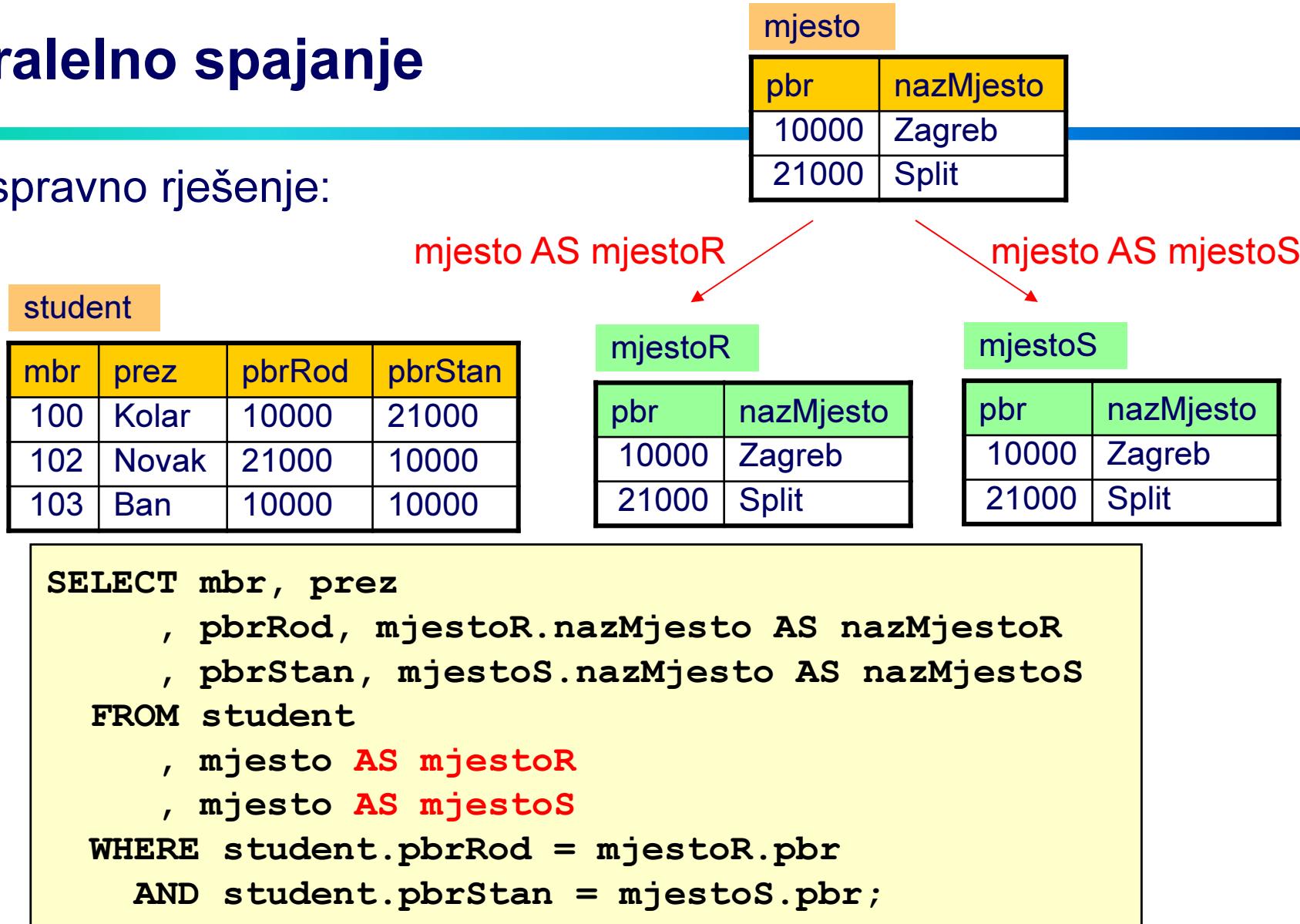
# Paralelno spajanje

---

- rješenje pomoću kopiranja relacija ima velike nedostatke
- postoji bolje (ispravno) rješenje

# Paralelno spajanje

- Ispravno rješenje:



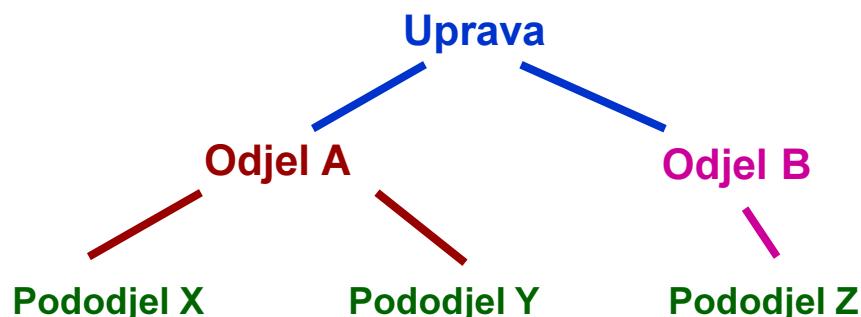
- u upitu se ista relacija pojavljuje u dvije različite uloge

# Refleksivno spajanje

- Pojedine n-torce iz relacije povezane su s drugim n-torkama iz iste relacije

| orgjed | sifOrgjed | nazOrgjed  | sifNadorgjed |
|--------|-----------|------------|--------------|
|        | 1         | Uprava     | NULL         |
|        | 2         | Odjel A    | 1            |
|        | 3         | Odjel B    | 1            |
|        | 4         | Pododjel X | 2            |
|        | 5         | Pododjel Y | 2            |
|        | 6         | Pododjel Z | 3            |

- Uprava nema nadređenu org. jedinicu
- Odjelu A neposredno nadređena jedinica je Uprava
- Odjelu B neposredno nadređena jedinica je Uprava
- Pododjelu X neposredno nadređena jedinica je Odjel A
- itd.



# Refleksivno spajanje

---

- Kako dobiti sljedeći rezultat

| sifOrgjed | nazOrgjed  | sifNadorgjed | nazNadorgjed |
|-----------|------------|--------------|--------------|
| 1         | Uprava     | NULL         | NULL         |
| 2         | Odjel A    | 1            | Uprava       |
| 3         | Odjel B    | 1            | Uprava       |
| 4         | Pododjel X | 2            | Odjel A      |
| 5         | Pododjel Y | 2            | Odjel A      |
| 6         | Pododjel Z | 3            | Odjel B      |

- radi se o spajanju relacije same sa sobom
- problem je sličan i slično se rješava kao u slučaju paralelnog spajanja
- relacija orgjed treba se u upitu pojaviti dva puta, jednom u ulozi organizacijske jedinice, a jednom u ulozi njezine nadređene organizacijske jedinice

# Refleksivno spajanje

orgjed

| sifOrgjed | nazOrgjed  | sifNadorgjed |
|-----------|------------|--------------|
| 1         | Uprava     | <b>NULL</b>  |
| 2         | Odjel A    | <b>1</b>     |
| 3         | Odjel B    | <b>1</b>     |
| 4         | Pododjel X | <b>2</b>     |
| 5         | Pododjel Y | <b>2</b>     |
| 6         | Pododjel Z | <b>3</b>     |

orgjed (u ulozi nadređene org.jedinice)

| sifOrgjed | nazOrgjed  | sifNadorgjed |
|-----------|------------|--------------|
| 1         | Uprava     | <b>NULL</b>  |
| 2         | Odjel A    | <b>1</b>     |
| 3         | Odjel B    | <b>1</b>     |
| 4         | Pododjel X | <b>2</b>     |
| 5         | Pododjel Y | <b>2</b>     |
| 6         | Pododjel Z | <b>3</b>     |

```
SELECT orgjed.sifOrgjed
 , orgjed.nazOrgjed
 , orgjed.sifNadorgjed
 , nadorgjed.nazOrgjed AS nazNadorgjed
 FROM orgjed, orgjed AS nadOrgjed
 WHERE orgjed.sifNadorgjed = nadOrgjed.sifOrgjed;
```

# Refleksivno spajanje

```
SELECT orgjed.sifOrgjed
 , orgjed.nazOrgjed
 , orgjed.sifNadorgjed
 , nadorgjed.nazOrgjed AS nazNadorgjed
 FROM orgjed, orgjed AS nadOrgjed
 WHERE orgjed.sifNadorgjed = nadOrgjed.sifOrgjed;
```

| sifOrgjed | nazOrgjed  | sifNadorgjed | nazNadorgjed |
|-----------|------------|--------------|--------------|
| 2         | Odjel A    | 1            | Uprava       |
| 3         | Odjel B    | 1            | Uprava       |
| 4         | Pododjel X | 2            | Odjel A      |
| 5         | Pododjel Y | 2            | Odjel A      |
| 6         | Pododjel Z | 3            | Odjel B      |

- Nema organizacijske jedinice Uprava? Kako to popraviti?

# Refleksivno spajanje

```
SELECT orgjed.sifOrgjed
 , orgjed.nazOrgjed
 , orgjed.sifNadorgjed
 , nadorgjed.nazOrgjed AS nazNadorgjed
 FROM orgjed
 LEFT OUTER JOIN orgjed AS nadOrgjed
 ON orgjed.sifNadorgjed = nadOrgjed.sifOrgjed;
```

| sifOrgjed | nazOrgjed  | sifNadorgjed | nazNadorgjed |
|-----------|------------|--------------|--------------|
| 1         | Uprava     | NULL         | NULL         |
| 2         | Odjel A    | 1            | Uprava       |
| 3         | Odjel B    | 1            | Uprava       |
| 4         | Pododjel X | 2            | Odjel A      |
| 5         | Pododjel Y | 2            | Odjel A      |
| 6         | Pododjel Z | 3            | Odjel B      |

# Refleksivno spajanje

---

- Kako dobiti sljedeći rezultat
  - uz svaku organizacijsku jedinicu ispisati nazine neposredno podređenih organizacijskih jedinica
  - ako org. jedinica ima više od jedne podređene org. jedinice, u popisu se pojavljuje više puta
  - u popisu se moraju naći i one organizacijske jedinice koje nemaju niti jednu podređenu organizacijsku jedinicu

| sifOrgjed | nazOrgjed  | nazPodorgjed |
|-----------|------------|--------------|
| 1         | Uprava     | Odjel A      |
| 1         | Uprava     | Odjel B      |
| 2         | Odjel A    | Pododjel X   |
| 2         | Odjel A    | Pododjel Y   |
| 3         | Odjel B    | Pododjel Z   |
| 4         | Pododjel X | NULL         |
| 5         | Pododjel Y | NULL         |
| 6         | Pododjel Z | NULL         |

# Refleksivno spajanje

| orgjed    |            |              |
|-----------|------------|--------------|
| sifOrgjed | nazOrgjed  | sifNadorgjed |
| 1         | Uprava     | NULL         |
| 2         | Odjel A    | 1            |
| 3         | Odjel B    | 1            |
| 4         | Pododjel X | 2            |
| 5         | Pododjel Y | 2            |
| 6         | Pododjel Z | 3            |

| orgjed (u ulozi podređene org.jedinice) |            |              |
|-----------------------------------------|------------|--------------|
| sifOrgjed                               | nazOrgjed  | sifNadorgjed |
| 1                                       | Uprava     | NULL         |
| 2                                       | Odjel A    | 1            |
| 3                                       | Odjel B    | 1            |
| 4                                       | Pododjel X | 2            |
| 5                                       | Pododjel Y | 2            |
| 6                                       | Pododjel Z | 3            |

```
SELECT orgjed.sifOrgjed
 , orgjed.nazOrgjed
 , podOrgjed.nazOrgjed AS nazPodorgjed
 FROM orgjed
 LEFT OUTER JOIN orgjed AS podOrgjed
 ON podOrgjed.sifNadorgjed = orgjed.sifOrgjed;
```

# Preimenovanje relacija unutar upita

- Još jedan primjer u kojem se koristi preimenovanje relacije
- Ispisati podatke o svim osobama čija je plaća manja od plaće osobe sa šifrom 103

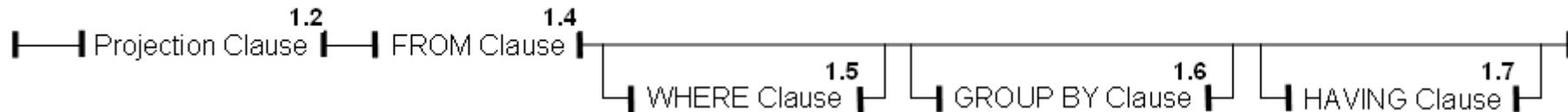
| osoba | sifra | ime  | prez  | placa |
|-------|-------|------|-------|-------|
|       | 100   | Ana  | Novak | 6000  |
|       | 101   | Ana  | Kolar | 5000  |
|       | 102   | Ivan | Kolar | 3000  |
|       | 103   | Ana  | Novak | 5000  |
|       | 104   | Jura | Ban   | 4000  |

```
SELECT osoba.*
FROM osoba
INNER JOIN osoba AS osoba103
ON osoba.placa < osoba103.placa
AND osoba103.sifra = 103;
```

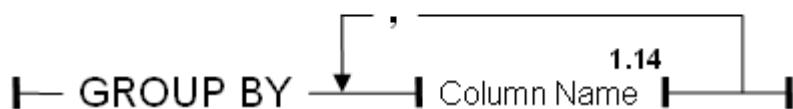
| sifra | ime  | prez  | placa |
|-------|------|-------|-------|
| 102   | Ivan | Kolar | 3000  |
| 104   | Jura | Ban   | 4000  |

# GROUP BY Clause

## 1.1. SELECT Options



## 1.6. GROUP BY Clause



- U GROUP BY dijelu naredbe se navodi jedan ili više **atributa** relacija koje su navedene u FROM dijelu naredbe

# GROUP BY Clause

ispit

| matBr | nazPredmet    | ocjena |
|-------|---------------|--------|
| 100   | Matematika    | 3      |
| 100   | Programiranje | 2      |
| 100   | Fizika        | 5      |
| 101   | Matematika    | 2      |
| 101   | Programiranje | 2      |
| 101   | Fizika        | 3      |
| 102   | Matematika    | 4      |

```
SELECT nazPredmet AS naziv
 , AVG(ocjena) AS prosjek
FROM ispit
GROUP BY nazPredmet;
```

| naziv         | prosjek |
|---------------|---------|
| Matematika    | 3       |
| Programiranje | 2       |
| Fizika        | 4       |

- u GROUP BY nije dopušteno koristiti izraze ili zamjenska imena atributa (*display\_label*)

```
SELECT nazPredmet AS naziv
 , AVG(ocjena)
FROM ispit
GROUP BY naziv;
```

# HAVING Clause

| ispit | matBr | nazPredmet    | ocjena |
|-------|-------|---------------|--------|
|       | 100   | Matematika    | 3      |
|       | 100   | Programiranje | 2      |
|       | 100   | Fizika        | 5      |
|       | 101   | Matematika    | 2      |
|       | 101   | Programiranje | 2      |
|       | 101   | Fizika        | 3      |
|       | 102   | Matematika    | 4      |

```
SELECT nazPredmet AS naziv
 , AVG(ocjena) AS prosjek
 FROM ispit
GROUP BY nazPredmet;
```

| naziv         | prosjek |
|---------------|---------|
| Matematika    | 3       |
| Programiranje | 2       |
| Fizika        | 4       |

- Kako u rezultatu prikazati samo one grupe koje zadovoljavaju neki uvjet, npr. kako u rezultatu prikazati samo one predmete za koje je prosjek ocjena veći od 2 ?

```
SELECT nazPredmet AS naziv
 , AVG(ocjena) AS prosjek
 FROM ispit
GROUP BY nazPredmet
HAVING AVG(ocjena) > 2;
```

| naziv      | prosjek |
|------------|---------|
| Matematika | 3       |
| Fizika     | 4       |

# **HAVING Clause**

- U *Condition* koji se navodi u HAVING dijelu naredbe dopušteno je u izrazima izvan agregatnih funkcija koristiti samo one attribute koji su navedeni u GROUP BY dijelu naredbe

## 1.1. SELECT Options



## 1.7. HAVING Clause



```
SELECT nazPredmet AS naziv
 , AVG(ocjena) AS prosjek
 FROM ispit
 GROUP BY nazPredmet
HAVING matBr > 104;
```

# HAVING Clause

- Primjer: ispisati nazine predmeta i njihove prosječne ocjene, ali samo za one predmete u kojima je najveća ikad dobivena ocjena bila **manja ili jednaka 4**

| ispit |               |        |
|-------|---------------|--------|
| matBr | nazPredmet    | ocjena |
| 100   | Matematika    | 3      |
| 100   | Programiranje | 2      |
| 100   | Fizika        | 5      |
| 101   | Matematika    | 2      |
| 101   | Programiranje | 2      |
| 101   | Fizika        | 3      |
| 102   | Matematika    | 4      |

```
SELECT nazPredmet AS naziv
 , AVG(ocjena) AS prosjek
 FROM ispit
 GROUP BY nazPredmet
 HAVING MAX(ocjena) <= 4 ;
```

| naziv         | prosjek |
|---------------|---------|
| Matematika    | 3       |
| Programiranje | 2       |

# HAVING Clause

- U rezultatu se pojavljuju one grupe za koje se navedeni uvjet (*Condition*) izračuna kao logička vrijednost *true*. U rezultatu se ne pojavljuju one grupe za koje se navedeni uvjet izračuna kao logička vrijednost *false* ili *unknown*

ispit

|     | matBr         | nazPredmet | ocjena |
|-----|---------------|------------|--------|
| 100 | Matematika    | 3          |        |
| 100 | Programiranje | 2          |        |
| 100 | Fizika        | NULL       |        |
| 101 | Matematika    | 2          |        |
| 101 | Programiranje | 2          |        |
| 101 | Fizika        | NULL       |        |
| 102 | Matematika    | 4          |        |

```
SELECT nazPredmet AS naziv
 , AVG(ocjena) AS prosjek
 FROM ispit
 GROUP BY nazPredmet
 HAVING AVG(ocjena) > 2;
```

| naziv      | prosjek |
|------------|---------|
| Matematika | 3       |

# **ORDER BY Clause**

---

- Koristi se za sortiranje rezultata upita
- Ispisati podatke o položenim ispitima: poredati ih prema ocjenama, tako da se bliže početku liste nalaze studenti s većim ocjenama. Studente koji imaju međusobno jednake ocjene poredati prema prezimenima, tako da se "manja" prezimena ispisuju prije "većih" prezimena (tj. po abecedi)

```
SELECT *
 FROM poloziliProg
 ORDER BY ocjena DESC
 , prez ASC;
```

| poloziliProg |        |        |
|--------------|--------|--------|
| matBr        | prez   | ocjena |
| 100          | Horvat | 3      |
| 107          | Novak  | 3      |
| 102          | Horvat | 5      |
| 101          | Kolar  | 5      |
| 103          | Kolar  | 2      |
| 104          | Horvat | 3      |

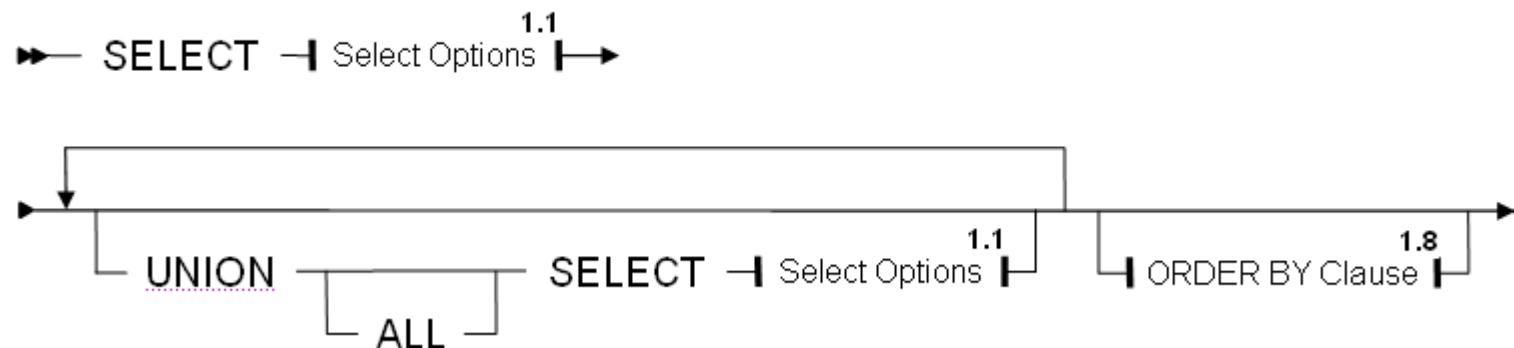
| matBr | prez   | ocjena |
|-------|--------|--------|
| 102   | Horvat | 5      |
| 101   | Kolar  | 5      |
| 104   | Horvat | 3      |
| 100   | Horvat | 3      |
| 107   | Novak  | 3      |
| 103   | Kolar  | 2      |

**DESC**  
silazno (*descending*)  
**ASC**  
uzlazno (*ascending*)

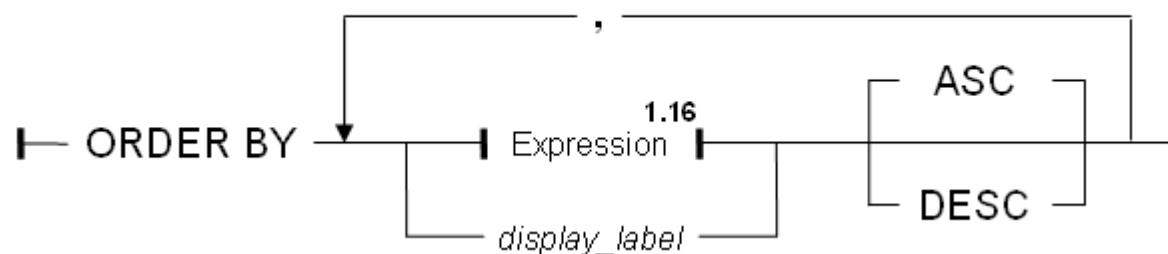
# ***ORDER BY Clause***

---

## **1. SELECT Statement**



## **1.8. ORDER BY Clause**



- Ako se smjer sortiranja ne navede, podrazumijeva se uzlazni (ASC) smjer sortiranja

## ***ORDER BY Clause***

---

- U ORDER BY dijelu naredbe mogu se koristiti i izrazi koji nisu navedeni u listi za selekciju
- ORDER BY dio naredbe je jedino mjesto u SELECT naredbi u kojem je dopušteno referencirati se na zamjensko ime atributa (*display\_label*)
- U jednoj SELECT naredbi može se pojaviti samo jedan ORDER BY dio naredbe
  - ako se u SELECT naredbi koristi UNION, ORDER BY se nalazi iza posljednjeg SELECT dijela naredbe
- SQL standard zahtijeva da se NULL vrijednosti pri sortiranju smatraju ili uvijek manjim ili uvijek većim od svih drugih vrijednosti
  - IBM Informix NULL vrijednosti pri sortiranju uvijek tretira kao da su manje od svih ostalih vrijednosti

# ***ORDER BY Clause***

---

| bodoviMat | mbr | prez   | bodLab | bodMI |
|-----------|-----|--------|--------|-------|
|           | 101 | Novak  | 20     | 30    |
|           | 103 | Horvat | NULL   | 20    |
|           | 107 | Ban    | 10     | 80    |

| bodoviProg | mbr | prez  | bodLab | bodMI |
|------------|-----|-------|--------|-------|
|            | 102 | Kolar | 12     | NULL  |
|            | 104 | Novak | 30     | 0     |

- ispisati podatke o bodovima na lab. vježbama i međuispitu, te ukupnom broju bodova svih studenata, poredati po ukupnom broju bodova: studenti s manjim ukupnim brojem bodova nalaze se bliže početku liste

```
SELECT *, bodLab + bodMI AS ukupno
 FROM bodoviMat
UNION
SELECT *, bodLab + bodMI AS ukupno
 FROM bodoviProg
 ORDER BY ukupno;
```

| mbr | prez   | bodLab | bodMI | ukupno |
|-----|--------|--------|-------|--------|
| 102 | Kolar  | 12     | NULL  | NULL   |
| 103 | Horvat | NULL   | 20    | NULL   |
| 104 | Novak  | 30     | 0     | 30     |
| 101 | Novak  | 20     | 30    | 50     |
| 107 | Ban    | 10     | 80    | 90     |

# "Redoslijed obavljanja" dijelova SELECT naredbe

---

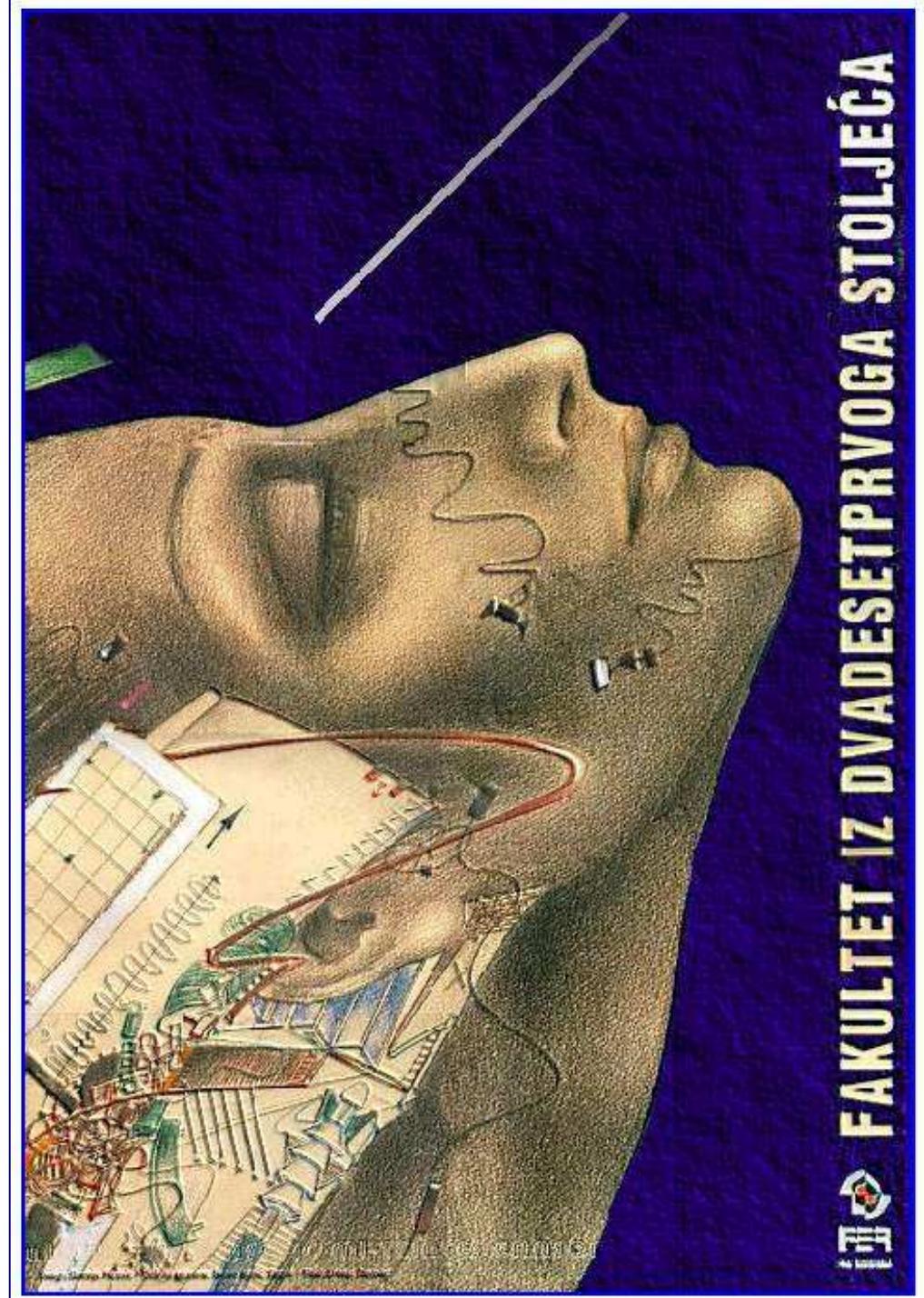


- **Logički promatrano**, tj. konačni rezultat će sigurno odgovarati rezultatu koji bi se dobio kada bi se operacije obavljale navedenim redoslijedom. Fizički promatrano, upit će se možda izvesti drugačijim redoslijedom.

# Baze podataka

Predavanja  
ožujak 2014.

## 5. SQL (2. dio)



FAKULTET IZ DVADESETPRVOGA STOLJEĆA



# Podupiti

# Podupiti (Subqueries)

| vozilo | sifVoz | nosivost |
|--------|--------|----------|
|        | 101    | 2500     |
|        | 102    | 2000     |
|        | 103    | 800      |
|        | 104    | 1000     |

| teret | sifTeret | tezina |
|-------|----------|--------|
|       | 1001     | 1800   |
|       | 1002     | 1200   |
|       | 1003     | 1000   |

- Ispisati podatke o vozilima čija je nosivost veća od težine najtežeg tereta
- Pogrešan način:** prvo obaviti upit kojim se određuje težina najtežeg tereta

```
SELECT MAX(tezina) FROM teret;
```

- Zapamtitи dobiveni rezultat (1800), te napisati novi upit:

```
SELECT *
 FROM vozilo
 WHERE nosivost > 1800;
```

| sifVoz | nosivost |
|--------|----------|
| 101    | 2500     |
| 102    | 2000     |

# Podupiti (Subqueries)

- Ispravan način:

```
SELECT *
 FROM vozilo
 WHERE nosivost > (SELECT MAX(tezina)
 FROM teret);
```

podupit

| teret    |        |
|----------|--------|
| sifTeret | tezina |
| 1001     | 1800   |
| 1002     | 1200   |
| 1003     | 1000   |

vozilo

| sifVoz | nosivost |
|--------|----------|
| 101    | 2500     |
| 102    | 2000     |
| 103    | 800      |
| 104    | 1000     |

1800

2500 > (SELECT MAX ...) → true  
2000 > (SELECT MAX ...) → true  
800 > (SELECT MAX ...) → false  
1000 > (SELECT MAX ...) → false

| sifVoz | nosivost |
|--------|----------|
| 101    | 2500     |
| 102    | 2000     |

- U navedenom primjeru je rezultat podupita jednak za svaku n-torku iz relacije vozilo, stoga je (fizički promatrano) rezultat podupita dovoljno izračunati samo jednom tijekom obavljanja upita

# Podupiti (*Subqueries*)

---

- podupit je upit koji je ugrađen u neki drugi upit
  - upit u kojeg je podupit ugrađen naziva se vanjski upit (*outer query*)
  - osim izraza **podupit** (*subquery*), u literaturi se također koristi i izraz **ugniježđeni upit** (*nested query*)
- podupit se u vanjski upit može ugraditi
  - u uvjet (*Condition*) u WHERE dijelu vanjskog upita
  - u uvjet (*Condition*) u HAVING dijelu vanjskog upita
  - u listu za selekciju (*SELECT List*) vanjskog upita
- podupit može sadržavati sve do sada spomenute dijelove SELECT naredbe osim ORDER BY dijela naredbe
- u vanjski upit se može ugraditi više podupita, u svaki od podupita se može ugraditi više podupita, itd.

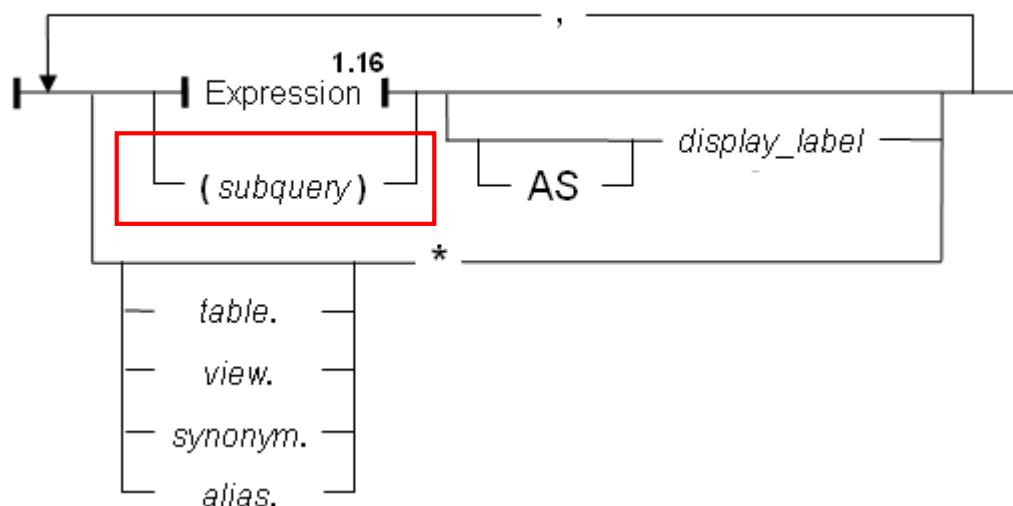
# Skalarni podupit (*Scalar subquery*)

---

- za početak, razmatrat će se najjednostavniji oblik podupita: podupit čiji je rezultat **jedna jednostavna** vrijednost (skalar)
  - npr. podatak tipa: cijeli broj, niz znakova, datum, itd.
- može se reći: rezultat skalarnog podupita je "relacija" stupnja jedan i kardinalnosti jedan
  - vrijednost atributa n-torke dotične "relacije" se u vanjskom upitu koristi kao skalarna vrijednost

# Podupiti u listi za selekciju

## 1.3 SELECT List



| vozilo | sifVoz | nosivost |
|--------|--------|----------|
| 101    | 2500   |          |
| 102    | 2000   |          |
| 103    | 800    |          |
| 104    | 1000   |          |

| teret | sifTeret | tezina |
|-------|----------|--------|
| 1001  | 1800     |        |
| 1002  | 1200     |        |
| 1003  | 1000     |        |

- Ispisati podatke o svim vozilima. Uz svako vozilo ispisati podatak o najvećoj težini tereta

```
SELECT *
, (SELECT MAX(tezina)
 FROM teret) AS maxTezina
FROM vozilo;
```

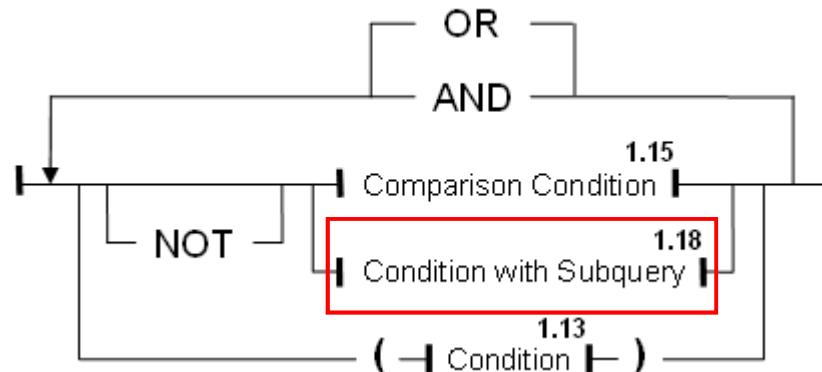
| sifVoz | nosivost | maxTezina |
|--------|----------|-----------|
| 101    | 2500     | 1800      |
| 102    | 2000     | 1800      |
| 103    | 800      | 1800      |
| 104    | 1000     | 1800      |

# Podupiti u WHERE dijelu naredbe

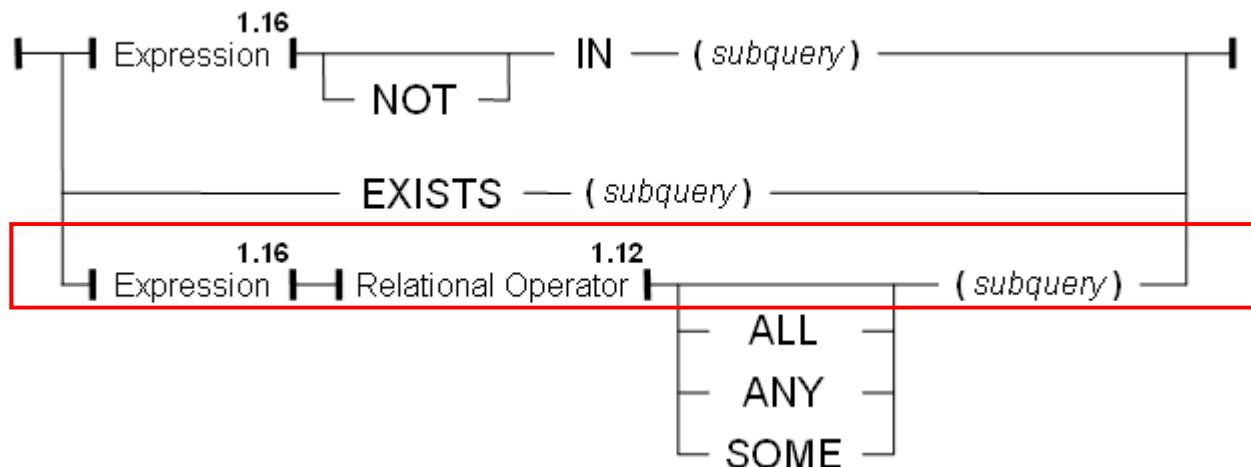
## 1.5. WHERE Clause



## 1.13. Condition



## 1.18. Condition with Subquery



- Ako se u WHERE ili HAVING dijelu naredbe koristi **ovdje** označeni oblik uvjeta, dopušteno je koristiti **isključivo skalarne podupite**

# Podupiti u WHERE dijelu naredbe

- Ispisati podatke o studentima koji stanuju u mjestu Ludbreg

| stud | mbr    | prez  | pbrSt |
|------|--------|-------|-------|
| 100  | Horvat | 42230 |       |
| 101  | Kolar  | 21000 |       |
| 102  | Novak  | 42230 |       |

| mjesto | pbr      | nazMjesto |
|--------|----------|-----------|
| 42000  | Varaždin |           |
| 42230  | Ludbreg  |           |
| 21000  | Split    |           |

```
SELECT * FROM stud
WHERE pbrSt = (SELECT pbr FROM mjesto
 WHERE nazMjesto = 'Ludbreg');
```

42230

- Često se problem može riješiti bez podupita. U konkretnom slučaju, **bolje** rješenje glasi:

```
SELECT stud.*
 FROM stud
 JOIN mjesto
 ON stud.pbrSt = mjesto.pbr
 WHERE nazMjesto = 'Ludbreg';
```

# Podupiti u WHERE dijelu naredbe

- Ispisati podatke o studentima koji su rođeni u mjestu Ludbreg, a stanuju u mjestu Varaždin

| stud | mbr    | prez  | pbrRod | pbrSt |
|------|--------|-------|--------|-------|
| 100  | Horvat | 42230 | 42230  |       |
| 101  | Kolar  | 21000 | 42000  |       |
| 102  | Novak  | 42230 | 42000  |       |

| mjesto | pbr      | nazMjesto |
|--------|----------|-----------|
| 42000  | Varaždin |           |
| 42230  | Ludbreg  |           |
| 21000  | Split    |           |

```
SELECT * FROM stud
```

```
WHERE pbrRod = (SELECT pbr FROM mjesto
 WHERE nazMjesto = 'Ludbreg')
```

```
AND pbrSt = (SELECT pbr FROM mjesto
 WHERE nazMjesto = 'Varaždin');
```

42230

42000

| mbr | prez  | pbrRod | pbrSt |
|-----|-------|--------|-------|
| 102 | Novak | 42230  | 42000 |

- Postoji bolje rješenje u kojem se koriste operacije spajanja i selekcije. Riješiti za vježbu!

# Podupiti u WHERE dijelu naredbe

| stud | mbr    | prez  | pbrSt |
|------|--------|-------|-------|
| 100  | Horvat | 42230 |       |
| 101  | Kolar  | 21000 |       |
| 102  | Novak  | 42230 |       |

| mjesto | pbr      | nazMjesto |
|--------|----------|-----------|
| 42000  | Varaždin |           |
| 42230  | Ludbreg  |           |
| 21000  | Split    |           |

- Ukoliko podupit čiji bi rezultat trebao biti skalar vrati više od jedne n-torce ili više nego jedan atribut, sustav će dojaviti pogrešku

```
SELECT * FROM stud
WHERE pbrSt = (SELECT pbr FROM mjesto
 WHERE nazMjesto LIKE '%r%');
```

→ Pogreška

- Ukoliko podupit čiji bi rezultat trebao biti skalar ne vrati niti jednu n-torku, dobivena skalarna vrijednost će biti NULL vrijednost

```
SELECT * FROM stud
WHERE pbrSt = (SELECT pbr FROM mjesto
 WHERE nazMjesto = 'Grad Split');
```

NULL

# Podupiti u HAVING dijelu naredbe

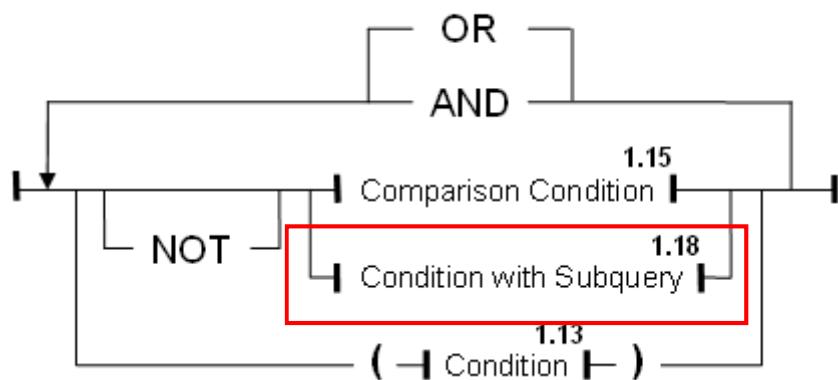
---

## 1.7. HAVING Clause



- Podupiti u HAVING dijelu naredbe koriste se na jednak način kao u WHERE dijelu naredbe

## 1.13. Condition



# Podupiti u HAVING dijelu naredbe

- Primjer:

dvorana

| oznDv | kapacitet |
|-------|-----------|
| D1    | 150       |
| D2    | 200       |
| A201  | 80        |

raspored

| predmet     | oznGr | brojSt |
|-------------|-------|--------|
| Matematika  | M1    | 200    |
| Matematika  | M2    | 50     |
| Matematika  | M3    | 50     |
| Fizika      | F1    | 250    |
| Fizika      | F2    | 150    |
| Elektronika | E1    | 50     |
| Elektronika | E2    | 50     |
| Elektronika | E3    | 100    |
| Elektronika | E4    | 150    |

- Ispisati nazive predmeta za koje se u "D dvoranama" predavanja mogu održati istovremeno za sve grupe. Uz svaki takav predmet ispisati ukupni broj studenata na predmetu

```
SELECT predmet
 , SUM(brojSt) AS ukupnoStud
 FROM raspored
 GROUP BY predmet
 HAVING SUM(brojSt) <=
 (SELECT SUM(kapacitet)
 FROM dvorana
 WHERE oznDv LIKE 'D%');
```

| predmet     | ukupnoStud |
|-------------|------------|
| Matematika  | 300        |
| Elektronika | 350        |

350

# Korelirani podupit (*Correlated subquery*)

---

- Ako se u podupitu koriste atributi iz vanjskog upita, za podupit i vanjski upit se kaže da su korelirani (*correlated*)
- Za podupit koji je koreliran s vanjskim upitom koristi se naziv korelirani podupit (*correlated subquery*)
- Najčešće se korelirani podupit mora (fizički) izvršiti po jedanput za svaku n-torku iz vanjskog upita
- Sljedeći podupit nije korelirani podupit: u podupitu se ne koriste atributi vanjskog upita. Rezultat podupita ne ovisi o vrijednostima n-torki iz vanjskog podupita, stoga se taj podupit (fizički) treba izvršiti samo jednom tijekom jednog obavljanja vanjskog upita

```
SELECT *
 FROM vozilo
 WHERE nosivost > (SELECT MAX(tezina)
 FROM teret);
```

The diagram illustrates the execution of a correlated subquery. The main query is shown in a yellow box:

```
SELECT *
 FROM vozilo
 WHERE nosivost > (SELECT MAX(tezina)
 FROM teret);
```

The subquery `(SELECT MAX(tezina) FROM teret)` is highlighted in blue and has an arrow pointing to the value `1800` in another blue box, indicating that the subquery is evaluated once per row of the outer query.

# Korelirani podupit (*Correlated subquery*)

- ispisati podatke o strojevima koji su ukupno korišteni više od dopuštenog broja radnih sati

| stroj | oznStr | dopBrSati |
|-------|--------|-----------|
|       | S1     | 1000      |
|       | S2     | 1500      |
|       | S3     | 500       |

| radStroja | oznStr | godina | brSatiRada |
|-----------|--------|--------|------------|
|           | S1     | 2002   | 700        |
|           | S1     | 2003   | 100        |
|           | S1     | 2004   | 300        |
|           | S2     | 2002   | 700        |
|           | S2     | 2003   | 500        |
|           | S3     | 2005   | 600        |

```
SELECT oznStr, dopBrSati
 FROM stroj
 WHERE dopBrSati <
 (SELECT SUM(brSatiRada)
 FROM radStroja
 WHERE oznStr = stroj.oznStr);
```

| oznStr | dopBrSati |
|--------|-----------|
| S1     | 1000      |
| S3     | 500       |

- korelirani podupit: rezultat podupita ovisi o vrijednostima atributa vanjskog upita - za svaku n-torku vanjskog upita dobiva se drugačiji rezultat podupita

# Korelirani podupit (Correlated subquery)

```
SELECT oznStr, dopBrSati
 FROM stroj
 WHERE dopBrSati <
 (SELECT SUM(brSatiRada)
 FROM radStroja
 WHERE radStroja.oznStr = stroj.oznStr) ;
```

| oznStr | dopBrSati |
|--------|-----------|
| S1     | 1000      |
| S3     | 500       |

- upit se (logički promatrano) obavlja na sljedeći način:
  - vanjski upit uzima jednu n-torku iz relacije stroj. Na temelju sadržaja te n-torke i sadržaja relacije radStroja, u podupitu se izračunava suma sati rada dotičnog stroja. Ukoliko je uvjet usporedbe zadovoljen, testirana n-torka se pojavljuje u rezultatu
  - postupak se ponavlja za svaku n-torku relacije stroj

| oznStr | dopBrSati |
|--------|-----------|
| S1     | 1000      |
| S2     | 1500      |
| S3     | 500       |

1000 < (SELECT SUM ... WHERE oznStr = 'S1'  $\Rightarrow$  1100)  $\rightarrow$  true  
1500 < (SELECT SUM ... WHERE oznStr = 'S2'  $\Rightarrow$  1200)  $\rightarrow$  false  
500 < (SELECT SUM ... WHERE oznStr = 'S3'  $\Rightarrow$  600)  $\rightarrow$  true

# Korelirani podupit (*Correlated subquery*)

- Primjer: korelirani podupit u listi za selekciju
- uz svaki stroj koji je korišten više od dopuštenog broja sati, ispisati broj sati korištenja stroja

```
SELECT oznStr
 , dopBrSati
 , (SELECT SUM(brSatiRada)
 FROM radStroja
 WHERE radStroja.oznStr = stroj.oznStr) AS koristenSati
 FROM stroj
 WHERE dopBrSati <
 (SELECT SUM(brSatiRada)
 FROM radStroja
 WHERE radStroja.oznStr = stroj.oznStr);
```

| oznStr | dopBrSati | koristenSati |
|--------|-----------|--------------|
| S1     | 1000      | 1100         |
| S3     | 500       | 600          |

# Korelirani podupit (*Correlated subquery*)

---

- Rješenje istog problema bez korištenja podupita:

```
SELECT stroj.oznStr
 , dopBrSati
 , SUM(brSatiRada) AS koristenSati
 FROM stroj
 JOIN radStroja
 ON stroj.oznStr = radStroja.oznStr
 GROUP BY stroj.oznStr, dopBrSati
 HAVING dopBrSati < SUM(brSatiRada) ;
```

| oznStr | dopBrSati | koristenSati |
|--------|-----------|--------------|
| S1     | 1000      | 1100         |
| S3     | 500       | 600          |

# Korelirani podupit (*Correlated subquery*)

- Primjer: korelirani podupit u HAVING dijelu naredbe

ispit

| mbr | predmet     | akGod | ocj |
|-----|-------------|-------|-----|
| 100 | Matematika  | 2011  | 2   |
| 101 | Matematika  | 2011  | 3   |
| 102 | Matematika  | 2011  | 4   |
| 100 | Fizika      | 2011  | 2   |
| 101 | Fizika      | 2011  | 5   |
| 100 | Elektronika | 2011  | 3   |
| 101 | Elektronika | 2011  | 3   |
| 110 | Matematika  | 2012  | 3   |
| 111 | Matematika  | 2012  | 5   |
| 110 | Fizika      | 2012  | 3   |
| 111 | Fizika      | 2012  | 3   |
| 112 | Fizika      | 2012  | 3   |
| 113 | Fizika      | 2012  | 2   |
| 111 | Elektronika | 2012  | 5   |
| 112 | Elektronika | 2012  | 4   |

- ispisati predmete čija je prosječna ocjena za 2012. godinu veća od prosječne ocjene tog istog predmeta za 2011. godinu

```
SELECT predmet
 FROM ispit AS ispit2012
 WHERE akGod = 2012
 GROUP BY predmet
 HAVING AVG(ocj) >
 (SELECT AVG(ocj)
 FROM ispit
 WHERE predmet = ispit2012.predmet
 AND ispit.akGod = 2011);
```

| predmet     |
|-------------|
| Matematika  |
| Elektronika |

# Korištenje atributa vanjskog upita u podupitu

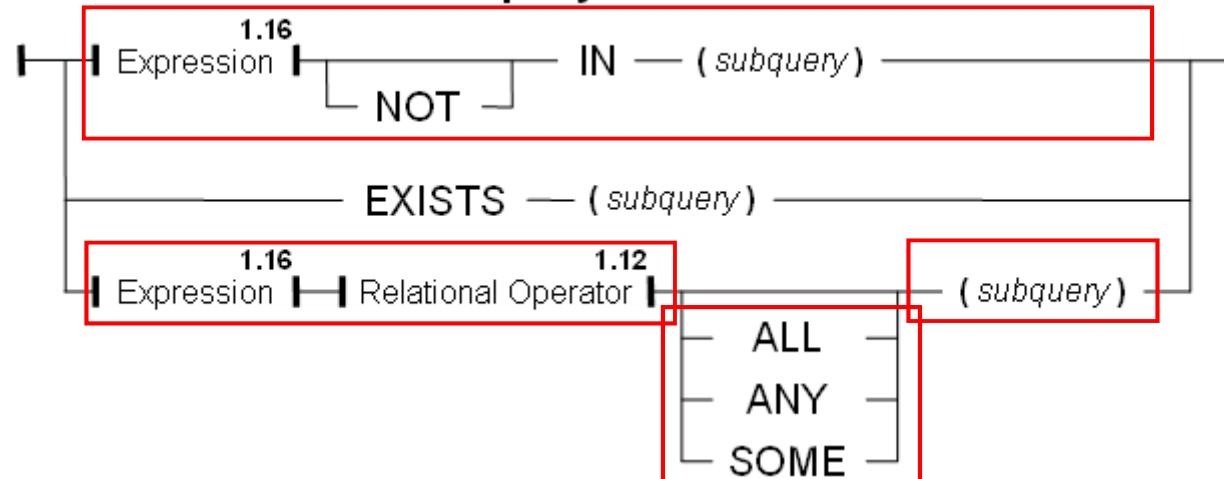
---

- u podupitu se mogu koristiti atributi iz vanjskog upita (obratno ne vrijedi)
- ukoliko se imena atributa (relacija) vanjskog upita podudaraju s imenima atributa (relacija) podupita:
  - ime atributa (relacije) navedeno u podupitu se odnosi na ime atributa (relacije) iz podupita
  - ime atributa (relacije) navedeno u vanjskom upitu se odnosi na ime atributa (relacije) vanjskog upita
- ukoliko je potrebno razriješiti dvosmislenost (npr. ista relacija se koristi u FROM dijelu vanjskog upita i FROM dijelu podupita, a u podupitu se koriste atributi relacije iz vanjskog upita), dovoljno je preimenovati relaciju u vanjskom upitu ili u podupitu
  - prethodni primjer ilustrira takav slučaj

# Jednostupčani podupit (*Single-column subquery*)

- Rezultat jednostupčanog podupita je relacija stupnja jedan, s (moguće) više n-torki
  - također je dopušteno da jednostupčani podupit vrati jednu ili niti jednu n-torku
- jednostupčani podupiti se koriste u WHERE dijelu ili HAVING dijelu vanjskog upita
- jednostupčani podupiti se ne koriste u listi za selekciju

## 1.18. Condition with Subquery



# Jednostupčani podupit (*Single-column subquery*)

- Ispisati podatke o studentima čije je prezime različito od svih prezimena nastavnika

| stud | mbr   | ime    | prez |
|------|-------|--------|------|
| 100  | Ivan  | Horvat |      |
| 101  | Ana   | Kolar  |      |
| 102  | Marko | Novak  |      |

| nastavnik | jmbg  | prez   |
|-----------|-------|--------|
|           | 12345 | Kolar  |
|           | 23456 | Ban    |
|           | 34567 | Pernar |

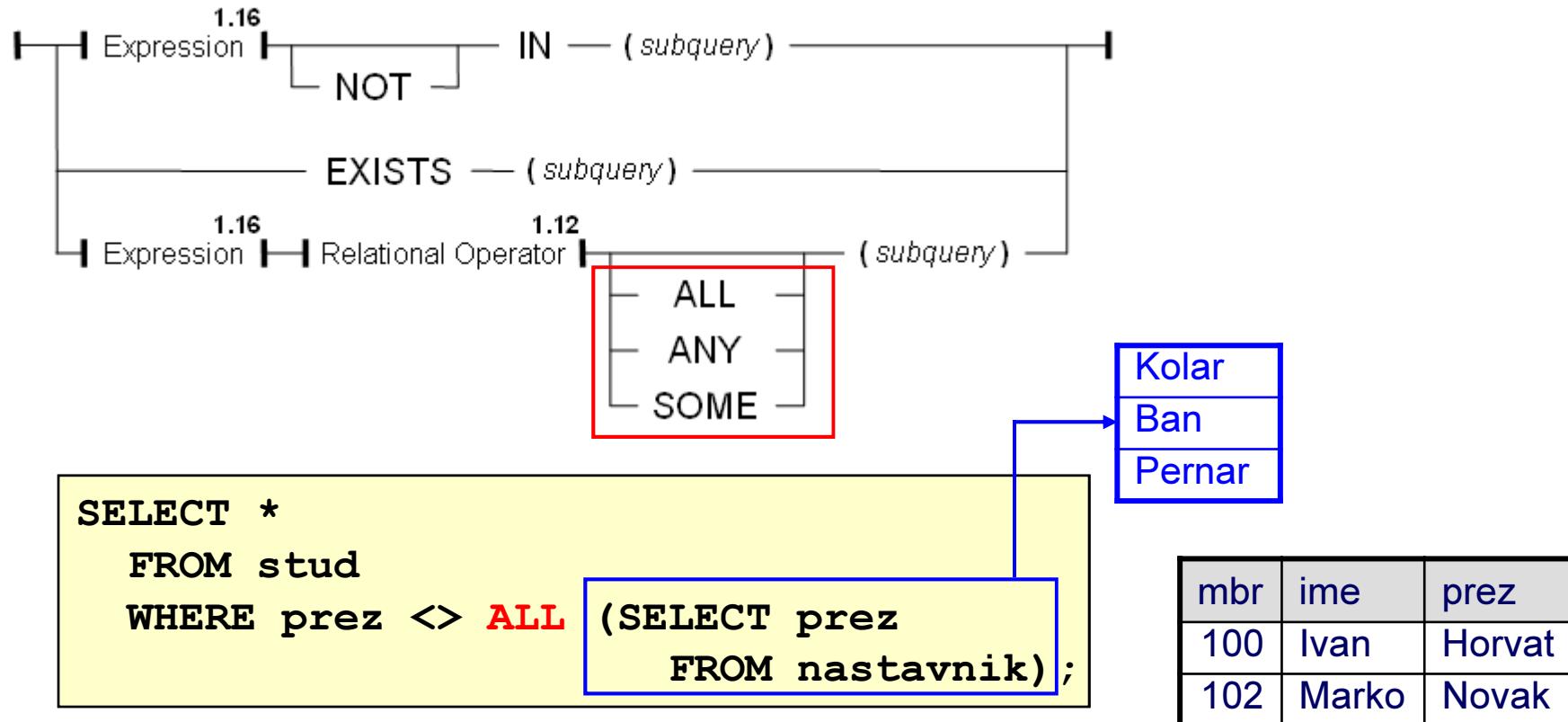
```
SELECT *
 FROM stud
 WHERE prez <> (SELECT prez
 FROM nastavnik);
```

|        |
|--------|
| Kolar  |
| Ban    |
| Pernar |

- Ovako napisan podupit **nije ispravan** - sustav će dojaviti pogrešku jer pomoću relacijskog operatora **<>** pokušavamo usporediti skalarnu vrijednost i rezultat podupita koji sadrži tri vrijednosti
- Potrebno je koristiti oblike usporedbe s IN, ALL, ANY, SOME

# Jednostupčani podupit (*Single-column subquery*)

## 1.18. Condition with Subquery



- uvjet selekcije će biti zadovoljen za one n-torce iz relacije stud čija je vrijednost atributa prez različita od vrijednosti svih članova (multi)skupa dobivenog obavljanjem podupita

# Jednostupčani podupit (*Single-column subquery*)

---

- izraz { < | <= | = | <> | > | >= } **ALL** (podupit)
  - *true* ako je izraz { < | <= | = | <> | > | >= } od svih vrijednosti dobivenih podupitom
- izraz { < | <= | = | <> | > | >= } **SOME** (podupit)
  - *true* ako je izraz { < | <= | = | <> | > | >= } od barem jedne vrijednosti dobivene podupitom
- **ANY** je sinonim za **SOME**

# Jednostupčani podupit (*Single-column subquery*)

- Ispisati podatke o dvoranama čiji je kapacitet veći od broja studenata u barem jednoj od grupa

dvorana

| oznDv | kapacitet |
|-------|-----------|
| D1    | 150       |
| D2    | 120       |
| A201  | 80        |

grupa

| oznGr | brojSt |
|-------|--------|
| M1    | 100    |
| F1    | 170    |
| E4    | 150    |

```
SELECT *
 FROM dvorana
 WHERE kapacitet > SOME (SELECT brojSt
 FROM grupa);
```

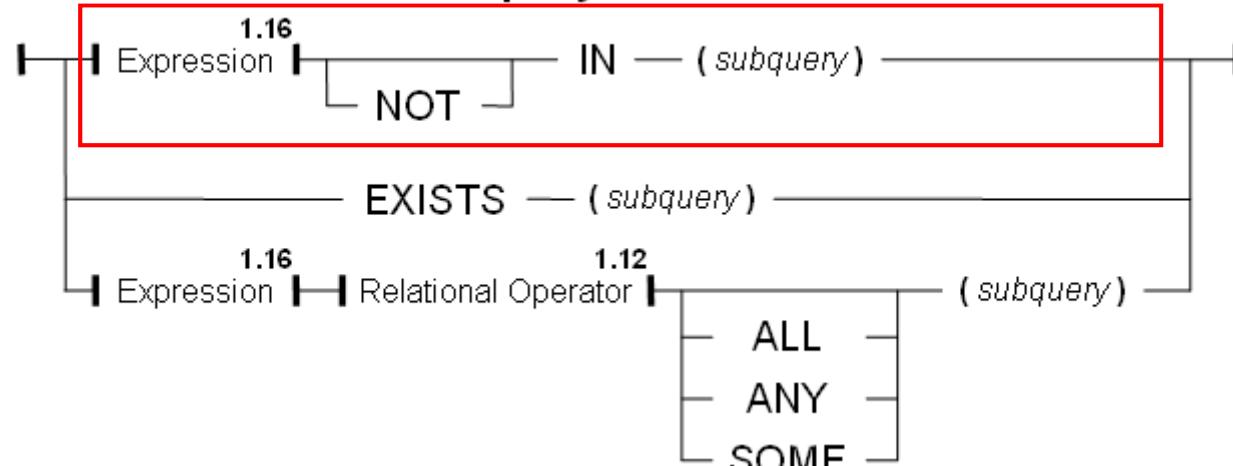
|     |
|-----|
| 100 |
| 170 |
| 150 |

| oznDv | kapacitet |
|-------|-----------|
| D1    | 150       |
| D2    | 120       |

- za vježbu riješiti bez podupita (spajanje, selekcija, projekcija)

# Jednostupčani podupit (*Single-column subquery*)

## 1.18. Condition with Subquery



- izraz **IN** (podupit)
  - *true* ako se u (multi)skupu vrijednosti dobivenih podupitom nalazi barem jedan element jednak vrijednosti izraza
  - ekvivalentno sa: izraz = **SOME** (podupit)
- izraz **NOT IN** (podupit)
  - *true* ako se u (multi)skupu vrijednosti dobivenih podupitom ne nalazi niti jedan element jednak vrijednosti izraza
  - ekvivalentno sa: izraz <> **ALL** (podupit)

# Jednostupčani podupit (*Single-column subquery*)

- Ispisati podatke o studentima koji su bilo koji predmet položili tijekom akademske godine 2010.

| stud |        |
|------|--------|
| mbr  | prez   |
| 100  | Horvat |
| 101  | Kolar  |
| 102  | Novak  |
| 103  | Ban    |

| ispit |             |       |        |
|-------|-------------|-------|--------|
| mbr   | predmet     | akGod | ocjena |
| 100   | Matematika  | 2011  | 1      |
| 100   | Elektronika | 2010  | 2      |
| 100   | Fizika      | 2010  | 3      |
| 102   | Elektronika | 2010  | 2      |
| 102   | Fizika      | 2011  | 5      |
| 103   | Elektronika | 2010  | NULL   |
| 103   | Matematika  | 2012  | 4      |

```
SELECT *
 FROM stud
 WHERE mbr IN
 (SELECT mbr
 FROM ispit
 WHERE akGod = 2010
 AND ocjena > 1);
```

|     |
|-----|
| 100 |
| 100 |
| 102 |

| mbr | prez   |
|-----|--------|
| 100 | Horvat |
| 102 | Novak  |

- za vježbu riješiti bez podupita (spajanje, selekcija, projekcija)

# NULL vrijednosti i jednostupčani podupiti

---

- izraz  $relop \text{ } \textcolor{red}{ALL}$  (podupit)
  - ako je podupitom dobiven skup vrijednosti  $\{ x_1, x_2, \dots, x_n \}$ , efektivno se uvjet izračunava na sljedeći način:
    - izraz  $relop x_1 \text{ AND } relop x_2 \text{ AND } \dots \text{ AND } relop x_n$
- izraz  $relop \text{ } \textcolor{red}{SOME}$  (podupit)
  - ako je podupitom dobiven skup vrijednosti  $\{ x_1, x_2, \dots, x_n \}$ , efektivno se uvjet izračunava na sljedeći način:
    - izraz  $relop x_1 \text{ OR } relop x_2 \text{ OR } \dots \text{ OR } relop x_n$
- izraz **IN** (podupit)
  - ekvivalentno sa: izraz = **SOME** (podupit)
- izraz **NOT IN** (podupit)
  - ekvivalentno sa: izraz  $\text{<>>} \text{ } \textcolor{red}{ALL}$  (podupit)

# Primjeri: podupiti i NULL vrijednosti

---

- naročitu pažnju pri korištenju podupita čiji rezultat može sadržavati NULL vrijednosti treba obratiti na uvjete selekcije oblika:

```
WHERE expression relationalOperator ALL (subquery)
```

```
WHERE expression NOT IN (subquery)
```

ukoliko se u rezultatu ovakvih podupita nalazi makar jedna NULL vrijednost, rezultat izračunavanja uvjeta selekcije nikad neće biti *true*

# Primjeri: podupiti i NULL vrijednosti

| uplata | rbrUpl | iznosUpl |
|--------|--------|----------|
|        | 1      | 50       |
|        | 2      | NULL     |
|        | 3      | 200      |
|        | 4      | 120      |

| isplata | rbrIspl | iznosIspl |
|---------|---------|-----------|
|         | 1       | 80        |
|         | 2       | 100       |
|         | 3       | NULL      |
|         | 4       | 150       |

```
SELECT * FROM uplata
WHERE iznosUpl >
SOME (SELECT iznosIspl FROM isplata);
```

|      |
|------|
| 80   |
| 100  |
| NULL |
| 150  |

| rbrUpl | iznosUpl |
|--------|----------|
| 1      | 50       |
| 2      | NULL     |
| 3      | 200      |
| 4      | 120      |

$50 > 80 \text{ OR } 50 > 100 \text{ OR } 50 > \text{NULL} \text{ OR } 50 > 150 \rightarrow \text{unknown}$

$\text{NULL} > 80 \text{ OR } \text{NULL} > 100 \text{ OR } \text{NULL} > \text{NULL} \text{ OR } \text{NULL} > 150 \rightarrow \text{unknown}$

$200 > 80 \text{ OR } 200 > 100 \text{ OR } 200 > \text{NULL} \text{ OR } 200 > 150 \rightarrow \text{true}$

$120 > 80 \text{ OR } 120 > 100 \text{ OR } 120 > \text{NULL} \text{ OR } 120 > 150 \rightarrow \text{true}$

| rbrUpl | iznosUpl |
|--------|----------|
| 3      | 200      |
| 4      | 120      |

# Primjeri: podupiti i NULL vrijednosti

| uplata | rbrUpl | iznosUpl |
|--------|--------|----------|
|        | 1      | 50       |
|        | 2      | NULL     |
|        | 3      | 200      |
|        | 4      | 120      |

| isplata | rbrIspl | iznosIspl |
|---------|---------|-----------|
|         | 1       | 80        |
|         | 2       | 100       |
|         | 3       | NULL      |
|         | 4       | 150       |

```
SELECT * FROM uplata
WHERE iznosUpl >
ALL (SELECT iznosIspl FROM isplata);
```

|      |
|------|
| 80   |
| 100  |
| NULL |
| 150  |

| rbrUpl | iznosUpl |
|--------|----------|
| 1      | 50       |
| 2      | NULL     |
| 3      | 200      |
| 4      | 120      |

50 > 80 AND 50 > 100 AND 50 > NULL AND 50 > 150 → *false*  
NULL > 80 AND NULL > 100 AND NULL > NULL AND NULL > 150 → *unknown*  
200 > 80 AND 200 > 100 AND 200 > NULL AND 200 > 150 → *unknown*  
120 > 80 AND 120 > 100 AND 120 > NULL AND 120 > 150 → *false*

| rbrUpl | iznosUpl |
|--------|----------|
|        |          |

# Primjeri: podupiti i NULL vrijednosti

| uplata | rbrUpl | iznosUpl |
|--------|--------|----------|
|        | 1      | 50       |
|        | 2      | NULL     |
|        | 3      | 100      |
|        | 4      | 80       |

| isplata | rbrIspl | iznosIspl |
|---------|---------|-----------|
|         | 1       | 80        |
|         | 2       | 100       |
|         | 3       | NULL      |
|         | 4       | 150       |

```
SELECT * FROM uplata
WHERE iznosUpl IN
 (SELECT iznosIspl FROM isplata);
```

|      |
|------|
| 80   |
| 100  |
| NULL |
| 150  |

| rbrUpl | iznosUpl |
|--------|----------|
| 1      | 50       |
| 2      | NULL     |
| 3      | 100      |
| 4      | 80       |

$50 = 80 \text{ OR } 50 = 100 \text{ OR } 50 = \text{NULL} \text{ OR } 50 = 150 \rightarrow \text{unknown}$

$\text{NULL} = 80 \text{ OR } \text{NULL} = 100 \text{ OR } \text{NULL} = \text{NULL} \text{ OR } \text{NULL} = 150 \rightarrow \text{unknown}$

$100 = 80 \text{ OR } 100 = 100 \text{ OR } 100 = \text{NULL} \text{ OR } 100 = 150 \rightarrow \text{true}$

$80 = 80 \text{ OR } 80 = 100 \text{ OR } 80 = \text{NULL} \text{ OR } 80 = 150 \rightarrow \text{true}$

| rbrUpl | iznosUpl |
|--------|----------|
| 3      | 100      |
| 4      | 80       |

# Primjeri: podupiti i NULL vrijednosti

| uplata | rbrUpl | iznosUpl |
|--------|--------|----------|
|        | 1      | 50       |
|        | 2      | NULL     |
|        | 3      | 100      |
|        | 4      | 80       |

| isplata | rbrIspl | iznosIspl |
|---------|---------|-----------|
|         | 1       | 80        |
|         | 2       | 100       |
|         | 3       | NULL      |
|         | 4       | 150       |

```
SELECT * FROM uplata
WHERE iznosUpl NOT IN
(SELECT iznosIspl FROM isplata);
```

|      |
|------|
| 80   |
| 100  |
| NULL |
| 150  |

| rbrUpl | iznosUpl |
|--------|----------|
| 1      | 50       |
| 2      | NULL     |
| 3      | 100      |
| 4      | 80       |

$50 \neq 80 \text{ AND } 50 \neq 100 \text{ AND } 50 \neq \text{NULL} \text{ AND } 50 \neq 150 \rightarrow \text{unknown}$

$\text{NULL} \neq 80 \text{ AND } \text{NULL} \neq 100 \text{ AND } \text{NULL} \neq \text{NULL} \text{ AND } \text{NULL} \neq 150 \rightarrow \text{unknown}$

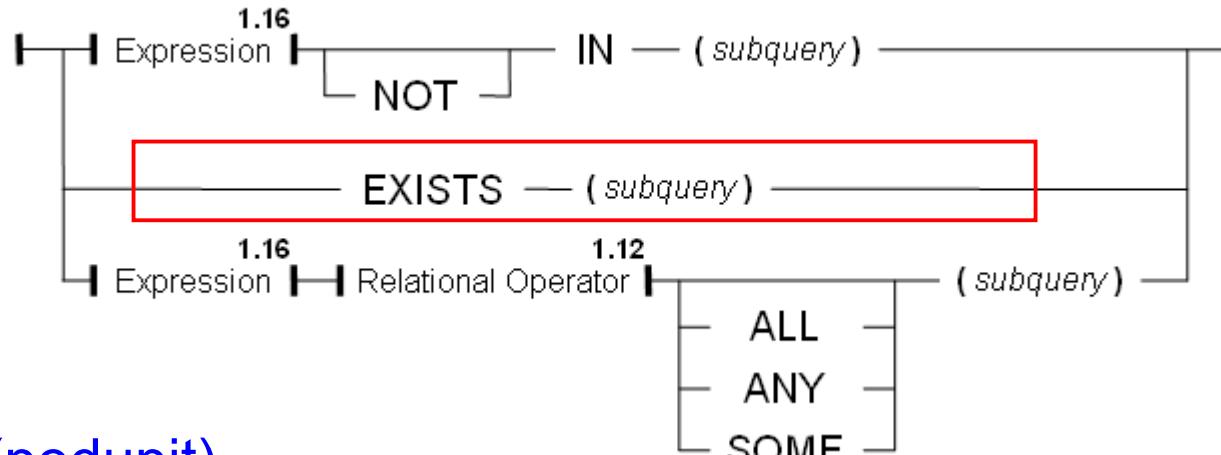
$100 \neq 80 \text{ AND } 100 \neq 100 \text{ AND } 100 \neq \text{NULL} \text{ AND } 100 \neq 150 \rightarrow \text{false}$

$80 \neq 80 \text{ AND } 80 \neq 100 \text{ AND } 80 \neq \text{NULL} \text{ AND } 80 \neq 150 \rightarrow \text{false}$

| rbrUpl | iznosUpl |
|--------|----------|
|        |          |

# Operator EXISTS

## 1.18. Condition with Subquery



- **EXISTS (podupit)**
  - *true* ako rezultat podupita sadrži barem jednu n-torku (bilo kakvu). Pri tome nije važno koliko u dobivenoj n-torci ili n-torkama ima atributa (podupit ne mora biti jednostupčan) niti koje su vrijednosti njihovih atributa
- **NOT EXISTS (podupit)**
  - *true* ako rezultat podupita ne sadrži niti jednu n-torku
- na rezultat vanjskog upita ne utječe eventualna pojava NULL vrijednosti u rezultatu podupita

# Operator EXISTS

- Ispisati podatke o studentima koji u akademskoj godini u kojoj su upisali studij nisu položili niti jedan ispit

| stud | mbr | prez   | akGodUpis |
|------|-----|--------|-----------|
|      | 100 | Horvat | 2010      |
|      | 101 | Kolar  | 2010      |
|      | 102 | Novak  | 2011      |
|      | 103 | Ban    | 2010      |

| ispit | mbr | predmet     | akGod | ocjena |
|-------|-----|-------------|-------|--------|
|       | 100 | Matematika  | 2010  | 1      |
|       | 100 | Fizika      | 2011  | 2      |
|       | 100 | Elektronika | 2012  | 4      |
|       | 100 | Matematika  | 2011  | 3      |
|       | 101 | Matematika  | 2010  | 2      |
|       | 101 | Fizika      | 2010  | 5      |
|       | 102 | Matematika  | 2011  | 4      |

```
SELECT *
 FROM stud
 WHERE NOT EXISTS
 (SELECT * FROM ispit
 WHERE ispit.mbr = stud.mbr
 AND akGod = akGodUpis
 AND ocjena > 1);
```

ovdje se npr. moglo napisati samo mbr: dobio bi se jednak rezultat

| mbr | prez   | akGodUpis |
|-----|--------|-----------|
| 100 | Horvat | 2010      |
| 103 | Ban    | 2010      |

# Podupiti u HAVING dijelu naredbe

- Svi prikazani oblici podupita mogu se također koristiti i u HAVING dijelu naredbe
- Primjer: ispisati naziv(e) predmeta s najvećim prosjekom

```
SELECT predmet
 FROM ispit
 GROUP BY predmet
 HAVING AVG(ocjena) >= ALL
 (SELECT AVG(ocjena)
 FROM ispit
 GROUP BY predmet);
```

| ispit |             |        |
|-------|-------------|--------|
| mbr   | predmet     | ocjena |
| 100   | Matematika  | 4      |
| 101   | Matematika  | 5      |
| 102   | Matematika  | 3      |
| 100   | Fizika      | 3      |
| 101   | Fizika      | 4      |
| 101   | Elektronika | 5      |
| 102   | Elektronika | 3      |

|     |
|-----|
| 4.0 |
| 3.5 |
| 4.0 |

| predmet     |
|-------------|
| Matematika  |
| Elektronika |

# Nepotrebno korištenje podupita

- Ispisati podatke o upisima predmeta svih studenata koji stanuju u Zaboku
- Loše rješenje:

```
SELECT * FROM upisanPredmet
WHERE jmbag IN (
 SELECT jmbag FROM student
 WHERE pbrStan IN (
 SELECT pbr FROM mjesto
 WHERE nazMjesto = 'Zabok'
)
);
```

- Bolje rješenje:

```
SELECT upisanPredmet.*
FROM upisanPredmet, student, mjesto
WHERE upisanPredmet.jmbag = student.jmbag
AND student.pbrStan = mjesto.pbr
AND nazMjesto = 'Zabok';
```

# Nepotrebno korištenje podupita

- Ispisati podatke o studentima i nazivima mjesta u kojima stanuju. U rezultatu trebaju biti i studenti čije je mjesto stanovanja nepoznato
- **Vrlo loše** rješenje:

```
SELECT student.*
 , (SELECT nazMjesto FROM mjesto
 WHERE pbr = pbrStan) AS nazMjesto
 FROM student;
```

- **Ispravno** rješenje:

```
SELECT student.* , nazMjesto
 FROM student
LEFT OUTER JOIN mjesto
 ON pbrStan = pbr;
```

- Zašto LEFT OUTER JOIN?

# Presjek

polozioMatem

| mbr | ime  | prez   |
|-----|------|--------|
| 100 | Ivan | Kolar  |
| 102 | Ana  | Novak  |
| 103 | Tea  | Ban    |
| 107 | Jura | Horvat |

polozioProgr

| mbr | ime  | prez   |
|-----|------|--------|
| 102 | Ana  | Novak  |
| 105 | Rudi | Kolar  |
| 107 | Jura | Horvat |

studenti koji su položili i  
Matematiku i Programiranje

**polozioMatem  $\cap$  polozioProgr**

```
SELECT *
 FROM polozioMatem
 WHERE EXISTS
 (SELECT * FROM polozioProgr
 WHERE polozioProgr.mbr = polozioMatem.mbr
 AND polozioProgr.ime = polozioMatem.ime
 AND polozioProgr.prez = polozioMatem.prez);
```

- Ispisuju se one n-torce relacije vanjskog upita za koje podupit **uspije** pronaći barem jednu n-torku koja ima jednakе vrijednosti atributa mbr, ime i prez kao n-torka iz vanjskog upita

| mbr | ime  | prez   |
|-----|------|--------|
| 102 | Ana  | Novak  |
| 107 | Jura | Horvat |

# Presjek (pomoću spajanja)

polozioMatem

| mbr | ime  | prez   |
|-----|------|--------|
| 100 | Ivan | Kolar  |
| 102 | Ana  | Novak  |
| 103 | Tea  | Ban    |
| 107 | Jura | Horvat |

polozioProgr

| mbr | ime  | prez   |
|-----|------|--------|
| 102 | Ana  | Novak  |
| 105 | Rudi | Kolar  |
| 107 | Jura | Horvat |

**polozioMatem  $\cap$  polozioProgr**

- Isti rezultat se može dobiti na sljedeći način:

```
SELECT polozioMatem.*
FROM polozioMatem, polozioProgr
WHERE polozioMatem.mbr = polozioProgr.mbr
AND polozioMatem.ime = polozioProgr.ime
AND polozioMatem.prez = polozioProgr.prez;
```

| mbr | ime  | prez   |
|-----|------|--------|
| 102 | Ana  | Novak  |
| 107 | Jura | Horvat |

# Presjek (u prisustvu NULL vrijednosti)

| polozioMatem |      |       |
|--------------|------|-------|
| mbr          | ime  | prez  |
| 100          | Ivan | Kolar |
| 102          | NULL | Novak |
| 103          | Tea  | Ban   |
| 107          | NULL | NULL  |

| polozioProgr |      |       |
|--------------|------|-------|
| mbr          | ime  | prez  |
| 102          | NULL | Novak |
| 105          | Rudi | Kolar |
| 107          | NULL | NULL  |

**polozioMatem  $\cap$  polozioProgr**

- Paziti: što je kopija n-torce?

| mbr | ime  | prez  |
|-----|------|-------|
| 102 | NULL | Novak |
| 107 | NULL | NULL  |

```
SELECT *
 FROM polozioMatem
 WHERE EXISTS
 (SELECT * FROM polozioProgr
 WHERE
 (polozioProgr.mbr = polozioMatem.mbr OR
 polozioProgr.mbr IS NULL AND polozioMatem.mbr IS NULL)
 AND
 (polozioProgr.ime = polozioMatem.ime OR
 polozioProgr.ime IS NULL AND polozioMatem.ime IS NULL)
 AND
 (polozioProgr.prez = polozioMatem.prez OR
 polozioProgr.prez IS NULL AND polozioMatem.prez IS NULL));
```

# Presjek (u prisustvu NULL vrijednosti)

| mbr | ime  | prez  |
|-----|------|-------|
| 100 | Ivan | Kolar |
| 102 | NULL | Novak |
| 103 | Tea  | Ban   |
| 107 | NULL | NULL  |

| mbr | ime  | prez  |
|-----|------|-------|
| 102 | NULL | Novak |
| 105 | Rudi | Kolar |
| 107 | NULL | NULL  |

- Isti rezultat može se dobiti pomoću spajanja

```
SELECT polozioMatem.*
FROM polozioMatem, polozioProgr
WHERE
 (polozioMatem.mbr = polozioProgr.mbr OR
 polozioMatem.mbr IS NULL AND polozioProgr.mbr IS NULL)
AND
 (polozioMatem.ime = polozioProgr.ime OR
 polozioMatem.ime IS NULL AND polozioProgr.ime IS NULL)
AND
 (polozioMatem.prez = polozioProgr.prez OR
 polozioMatem.prez IS NULL AND polozioProgr.prez IS NULL);
```

# Razlika

polozioMatem

| mbr | ime  | prez   |
|-----|------|--------|
| 100 | Ivan | Kolar  |
| 102 | Ana  | Novak  |
| 103 | Tea  | Ban    |
| 107 | Jura | Horvat |

polozioProgr

| mbr | ime  | prez   |
|-----|------|--------|
| 102 | Ana  | Novak  |
| 105 | Rudi | Kolar  |
| 107 | Jura | Horvat |

studenti koji su položili Matematiku,  
ali nisu položili Programiranje

**polozioMatem \ polozioProgr**

```
SELECT *
 FROM polozioMatem
 WHERE NOT EXISTS
 (SELECT * FROM polozioProgr
 WHERE polozioProgr.mbr = polozioMatem.mbr
 AND polozioProgr.ime = polozioMatem.ime
 AND polozioProgr.prez = polozioMatem.prez);
```

- Ispisuju se one n-torce relacije vanjskog upita za koje podupit **ne uspije** pronaći niti jednu n-torku koja ima jednakе vrijednosti atributa mbr, ime i prez kao n-torka iz vanjskog upita

| mbr | ime  | prez  |
|-----|------|-------|
| 100 | Ivan | Kolar |
| 103 | Tea  | Ban   |

# Razlika (u prisustvu NULL vrijednosti)

| polozioMatem |      |       |
|--------------|------|-------|
| mbr          | ime  | prez  |
| 100          | Ivan | Kolar |
| 102          | NULL | Novak |
| 103          | Tea  | Ban   |
| 107          | NULL | NULL  |

| polozioProgr |      |       |
|--------------|------|-------|
| mbr          | ime  | prez  |
| 102          | NULL | Novak |
| 105          | Rudi | Kolar |
| 107          | NULL | NULL  |

## polozioMatem \ polozioProgr

- Paziti: što je kopija n-torce?

| mbr | ime  | prez  |
|-----|------|-------|
| 100 | Ivan | Kolar |
| 103 | Tea  | Ban   |

```
SELECT *
 FROM polozioMatem
 WHERE NOT EXISTS
 (SELECT * FROM polozioProgr
 WHERE
 (polozioProgr.mbr = polozioMatem.mbr OR
 polozioProgr.mbr IS NULL AND polozioMatem.mbr IS NULL)
 AND
 (polozioProgr.ime = polozioMatem.ime OR
 polozioProgr.ime IS NULL AND polozioMatem.ime IS NULL)
 AND
 (polozioProgr.prez = polozioMatem.prez OR
 polozioProgr.prez IS NULL AND polozioMatem.prez IS NULL));
```

---

## SQL naredbe za izmjenu sadržaja relacije

**INSERT  
DELETE  
UPDATE**

# INSERT

```
CREATE TABLE mjesto (
 pbr INTEGER
, nazMjesto CHAR(30)
, sifZup SMALLINT
);
```



| mjesto |           |        |
|--------|-----------|--------|
| pbr    | nazMjesto | sifZup |

```
INSERT INTO mjesto
VALUES (42000, 'Varaždin', 7);

INSERT INTO mjesto
(pbr, sifZup, nazMjesto)
VALUES (52100, 4, 'Pula');

INSERT INTO mjesto
(pbr, nazMjesto)
VALUES (42230, 'Ludbreg');
```



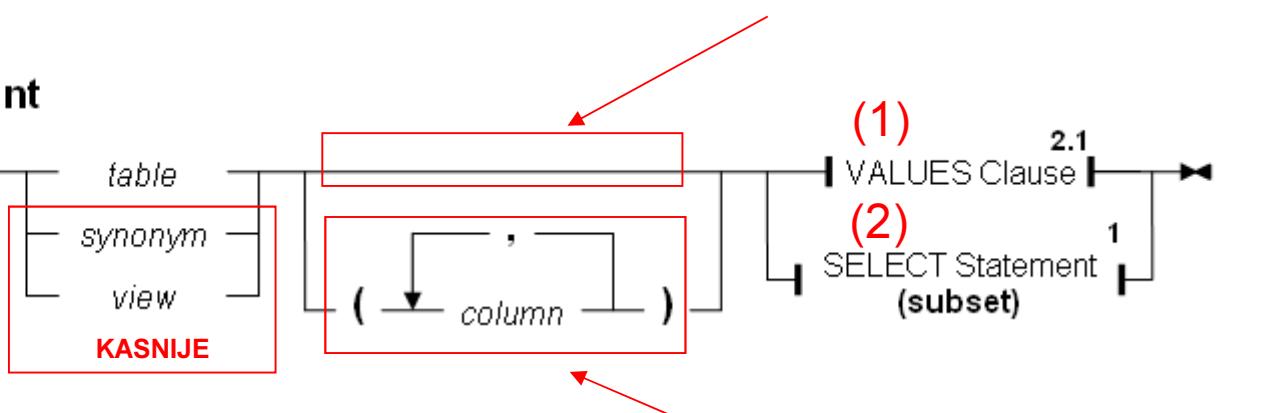
| mjesto |           |        |
|--------|-----------|--------|
| pbr    | nazMjesto | sifZup |
| 42000  | Varaždin  | 7      |
| 52100  | Pula      | 4      |
| 42230  | Ludbreg   | NULL   |

# INSERT

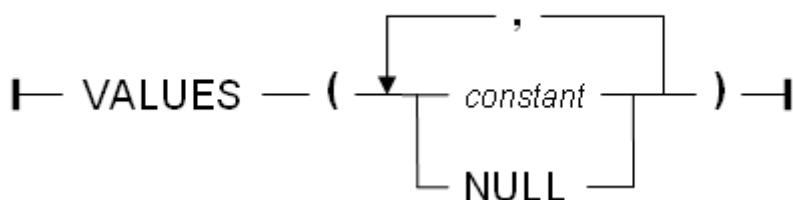
- INSERT naredba se koristi za unos jedne n-torke (1) ili skupa n-torki (2) u relaciju *table*

## 2. INSERT Statement

►► INSERT INTO



## 2.1. VALUES Clause



# INSERT (1), bez navedene liste atributa

- U relaciju se upisuje jedna n-torka pri čemu **vrijednosti svih** atributa n-torke **moraju biti navedene redoslijedom** kojim su atributi navedeni u CREATE TABLE naredbi kojom je relacija definirana

```
CREATE TABLE stud (
 mbr INTEGER
, ime NCHAR(30)
, prez NCHAR(50)
, stipend CHAR(1) DEFAULT 'N'
, pbrStan INTEGER DEFAULT 10000
);
```



Znači: ako se prilikom unosa nove n-torke ne navede vrijednost za atribut stipend, sustav će kao vrijednost tog atributa postaviti vrijednost N

```
INSERT INTO stud VALUES (
 100
, 'Ivan'
, 'Horvat'
, 'N'
, NULL
);
```



# INSERT (1), bez navedene liste atributa

---

- Opisani oblik INSERT naredbe ima neke nedostatke:
  - u slučaju kada se u relaciju upisuje n-torka čiji relativno veliki broj atributa treba postaviti na NULL vrijednost ili prepostavljenu vrijednost (*default value*)
    - ipak se moraju navesti vrijednosti **svih** atributa
  - u slučaju kada se relacijska shema promijeni (npr. promijeni se "redoslijed" atributa)
    - budući da vrijednosti atributa moraju biti navedene redoslijedom atributa u CREATE TABLE naredbi, INSERT naredbe koje su napisane prije promjene relacijske sheme više neće biti ispravne
- zbog navedenih nedostataka, preporuča se korištenje oblika INSERT naredbe s navedenom listom atributa

# INSERT (1), uz navedenu listu atributa

- U **prvom dijelu naredbe** navode se imena atributa (i njihov redoslijed) čije će vrijednosti (u odgovarajućem redoslijedu) biti navedene u drugom dijelu naredbe
  - atributi čije vrijednosti nisu navedene u INSERT naredbi, postavljaju se na prepostavljenu (*default*) vrijednost (ukoliko je takva definirana u CREATE TABLE naredbi) ili na NULL vrijednost

```
INSERT INTO stud (
 prez
 , mbr
 , pbrStan
)
VALUES (
 'Kolar'
 , 101
 , 10000
);
```

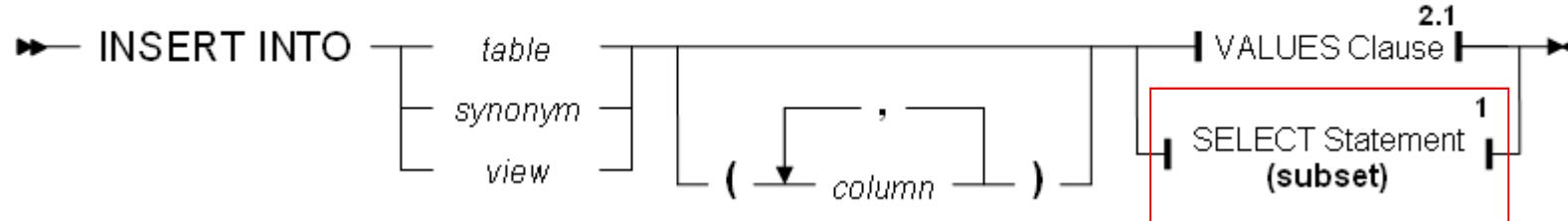
| stud |      |        |         |         |
|------|------|--------|---------|---------|
| mbr  | ime  | prez   | stipend | pbrStan |
| 100  | Ivan | Horvat | N       | NULL    |

| stud |      |        |         |         |
|------|------|--------|---------|---------|
| mbr  | ime  | prez   | stipend | pbrStan |
| 100  | Ivan | Horvat | N       | NULL    |
| 101  | NULL | Kolar  | N       | 10000   |

# INSERT (2)

## 2. INSERT Statement



- u relaciju *table* upisuju se n-torke dobivene dijelom INSERT naredbe koji je sličan SELECT naredbi - u sintaksnom dijagramu taj je dio naredbe označen sa *SELECT Statement (subset)*
  - *SELECT Statement (subset)* može sadržavati sve prethodno opisane dijelove SELECT naredbe **osim**
    - ORDER BY
    - FIRST n
    - UNION

## INSERT (2), bez navedene liste atributa

- u relaciju polozioFiz upisati podatke o studentima koji su položili predmet Fizika

| stud |      |        |       |
|------|------|--------|-------|
| mbr  | ime  | prez   | pbrSt |
| 102  | Ana  | Novak  | 10000 |
| 105  | Rudi | Kolar  | 21000 |
| 107  | Jura | Horvat | 41000 |
| 109  | Tea  | Ban    | 51000 |

| ispit |             |        |
|-------|-------------|--------|
| mbr   | predmet     | ocjena |
| 102   | Elektronika | 1      |
| 102   | Matematika  | 3      |
| 105   | Fizika      | 3      |
| 105   | Matematika  | 4      |
| 107   | Fizika      | 1      |
| 109   | Fizika      | 5      |

| polozioFiz |       |        |
|------------|-------|--------|
| mbr        | imeSt | prezSt |

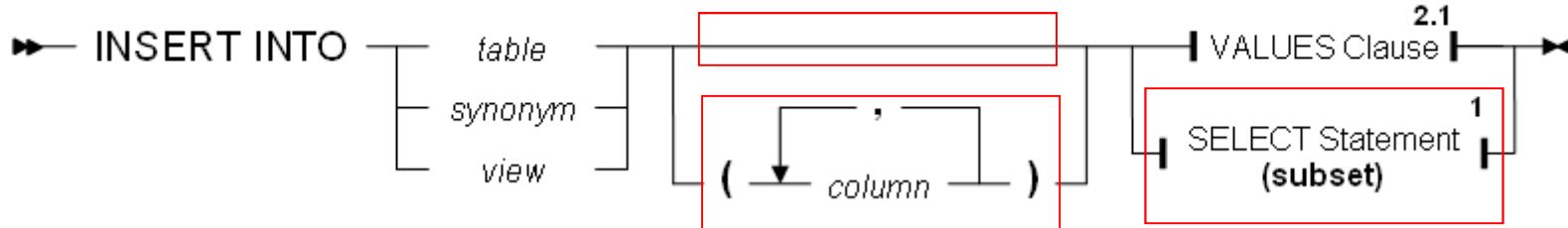
```
INSERT INTO polozioFiz
SELECT stud.mbr, ime, prez
FROM stud, ispit
WHERE stud.mbr = ispit.mbr
AND predmet = 'Fizika'
AND ocjena > 1;
```

imena atributa u SELECT listi  
ne moraju odgovarati imenima  
atributa u relaciji polozioFiz

| polozioFiz |       |        |
|------------|-------|--------|
| mbr        | imeSt | prezSt |
| 105        | Rudi  | Kolar  |
| 109        | Tea   | Ban    |

# INSERT (2)

## 2. INSERT Statement



- jednako kao kod oblika `INSERT` naredbe za unos jedne n-torke u relaciju
  - bez navedene liste imena atributa, broj i redoslijed atributa u n-torkama dobivenim obavljanjem `SELECT` dijela naredbe mora odgovarati broju i redoslijedu atributa u `CREATE TABLE` naredbi (priказано у prethodnom primjeru)
  - uz navedenu listu imena atributa, atributi čije vrijednosti nisu navedene u `INSERT` naredbi, postavljaju se na prepostavljenu (*default*) vrijednost (ukoliko je takva definirana u `CREATE TABLE` naredbi) ili na `NULL` vrijednost

## INSERT (2), uz navedenu listu atributa

- u relaciju prosjek upisati podatke o prosječnim ocjenama pojedinih predmeta. Za vrijednost atributa akGodina postaviti NULL vrijednost

| ispit |             |        |
|-------|-------------|--------|
| mbr   | predmet     | ocjena |
| 102   | Elektronika | 1      |
| 102   | Matematika  | 3      |
| 105   | Fizika      | 3      |
| 105   | Matematika  | 4      |
| 107   | Fizika      | 1      |
| 109   | Fizika      | 5      |

```
INSERT INTO prosjek (
 prosOcj
 , predmet
)
SELECT AVG(ocjena)
 , predmet
FROM ispit
GROUP BY predmet;
```



| projek  |         |       |
|---------|---------|-------|
| predmet | prosOcj | akGod |

| projek      |         |       |
|-------------|---------|-------|
| predmet     | prosOcj | akGod |
| Elektronika | 1.0     | NULL  |
| Matematika  | 3.5     | NULL  |
| Fizika      | 3.0     | NULL  |

# DELETE

- brisanje n-torki iz relacije

mjesto

| pbr   | nazMjesto | sifZup |
|-------|-----------|--------|
| 42230 | Ludbreg   | 7      |
| 42000 | VARAŽDIN  | 7      |
| 52100 | Pula      | 4      |

```
DELETE FROM mjesto
WHERE sifZup = 7;
```

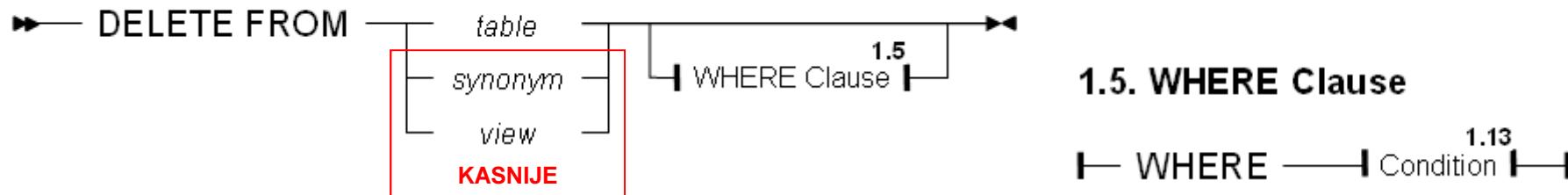


mjesto

| pbr   | nazMjesto | sifZup |
|-------|-----------|--------|
| 52100 | Pula      | 4      |

# DELETE

## 4. DELETE Statement



- DELETE naredba briše one n-torke relacije *table* za koje se uvjet naveden u WHERE dijelu naredbe izračuna kao *true*
  - ako se WHERE dio naredbe ne navede, iz relacije *table* se brišu **sve** n-torke
- u WHERE dijelu naredbe mogu se koristiti svi oblici uvjeta (*Condition*) koji se koriste u WHERE dijelu SELECT naredbe, ali u FROM dijelu podupruta nije dopušteno koristiti relaciju *table*

```
DELETE FROM mjesto
```

```
WHERE pbr IN (SELECT pbr FROM mjesto WHERE nazMjesto LIKE 'V%');
```

# DELETE

- iz relacije polozioFiz obrisati n-torce onih studenata koji (sudeći prema podacima iz relacije ispit) nisu položili predmet Fizika

| stud |      |        |       |
|------|------|--------|-------|
| mbr  | ime  | prez   | pbrSt |
| 102  | Ana  | Novak  | 10000 |
| 105  | Rudi | Kolar  | 21000 |
| 107  | Jura | Horvat | 41000 |
| 109  | Tea  | Ban    | 51000 |

| ispit |             |        |
|-------|-------------|--------|
| mbr   | predmet     | ocjena |
| 102   | Elektronika | 1      |
| 102   | Matematika  | 3      |
| 105   | Fizika      | 3      |
| 105   | Matematika  | 4      |
| 107   | Fizika      | 1      |
| 109   | Fizika      | 5      |

| polozioFiz |       |        |
|------------|-------|--------|
| mbr        | imeSt | prezSt |
| 102        | Ana   | Novak  |
| 105        | Rudi  | Kolar  |
| 107        | Jura  | Horvat |
| 109        | Tea   | Ban    |
| 111        | Ivan  | Polak  |

```
DELETE FROM polozioFiz
WHERE mbr NOT IN
(SELECT mbr
FROM ispit
WHERE predmet = 'Fizika'
AND ocjena > 1);
```



| polozioFiz |       |        |
|------------|-------|--------|
| mbr        | imeSt | prezSt |
| 105        | Rudi  | Kolar  |
| 109        | Tea   | Ban    |

# UPDATE

- Svim studentima, čija je ocjena manja od 5, ocjenu uvećati za 1, a broj bodova postaviti na NULL

```
UPDATE ispit
 SET ocjena = ocjena + 1
 , brBod = NULL
 WHERE ocjena < 5;
```

ili

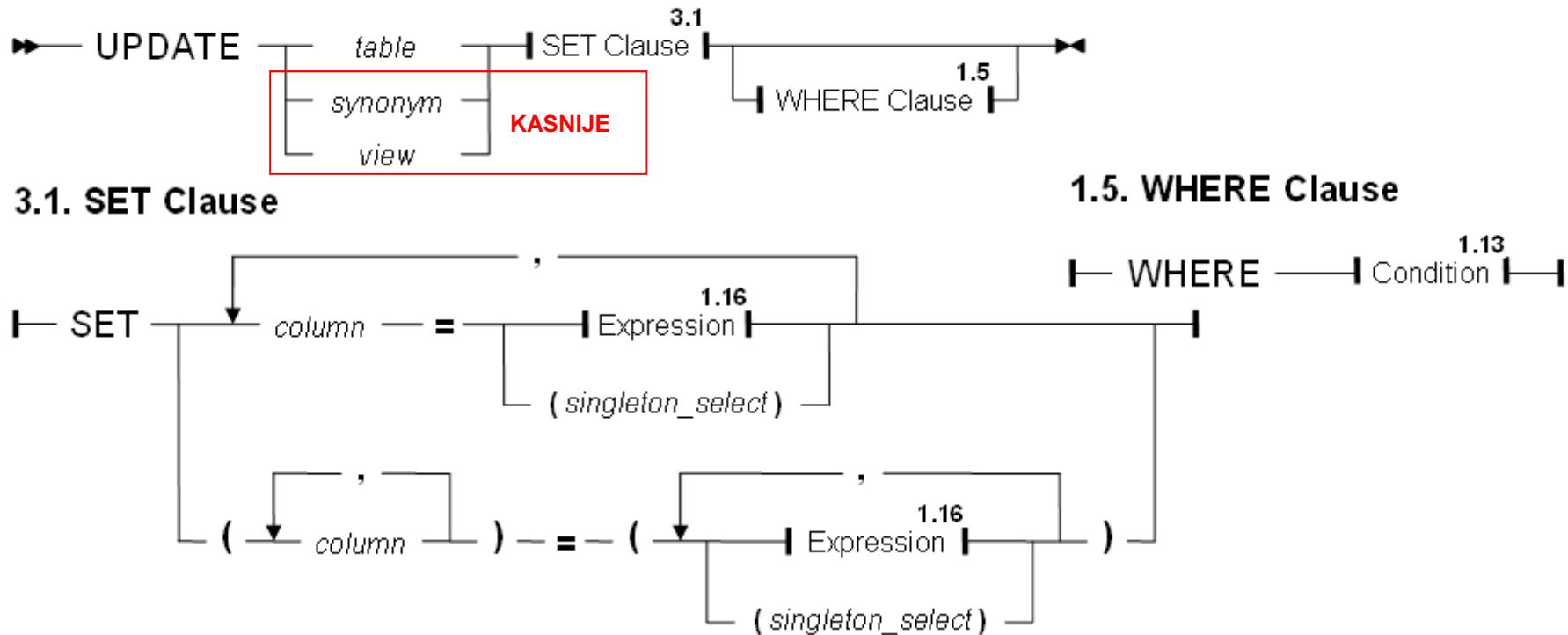
```
UPDATE ispit
 SET (ocjena, brBod) = (ocjena + 1, NULL)
 WHERE ocjena < 5;
```

| ispit |       |        |
|-------|-------|--------|
| mbrSt | brBod | ocjena |
| 1001  | 200   | 5      |
| 1002  | 150   | 4      |
| 1003  | 130   | 3      |
| 1004  | 110   | 2      |

| ispit |       |        |
|-------|-------|--------|
| mbrSt | brBod | ocjena |
| 1001  | 200   | 5      |
| 1002  | NULL  | 5      |
| 1003  | NULL  | 4      |
| 1004  | NULL  | 3      |

# UPDATE

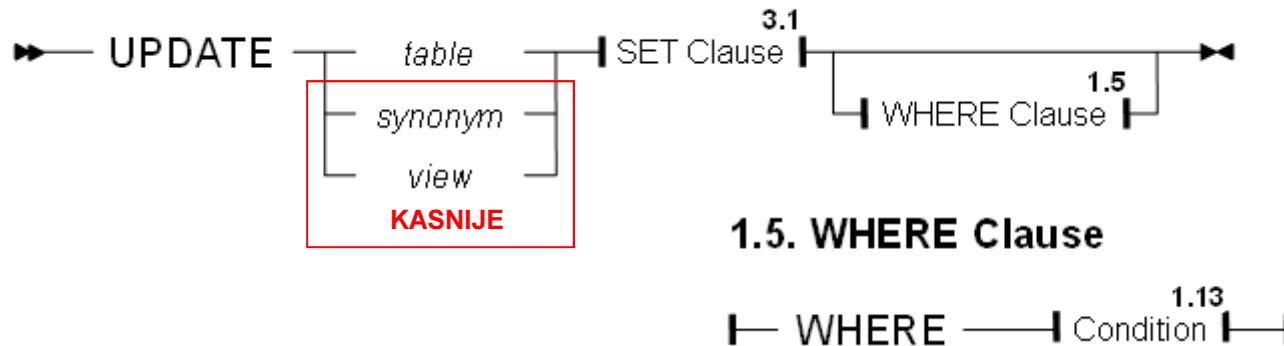
## 3. UPDATE Statement



- UPDATE naredba mijenja vrijednosti atributa postojećih n-torki relacije *table*. U WHERE dijelu naredbe opisuje se koje n-torke će biti promijenjene; u SET dijelu naredbe opisuje se koji će atributi biti postavljeni na koje vrijednosti

# UPDATE

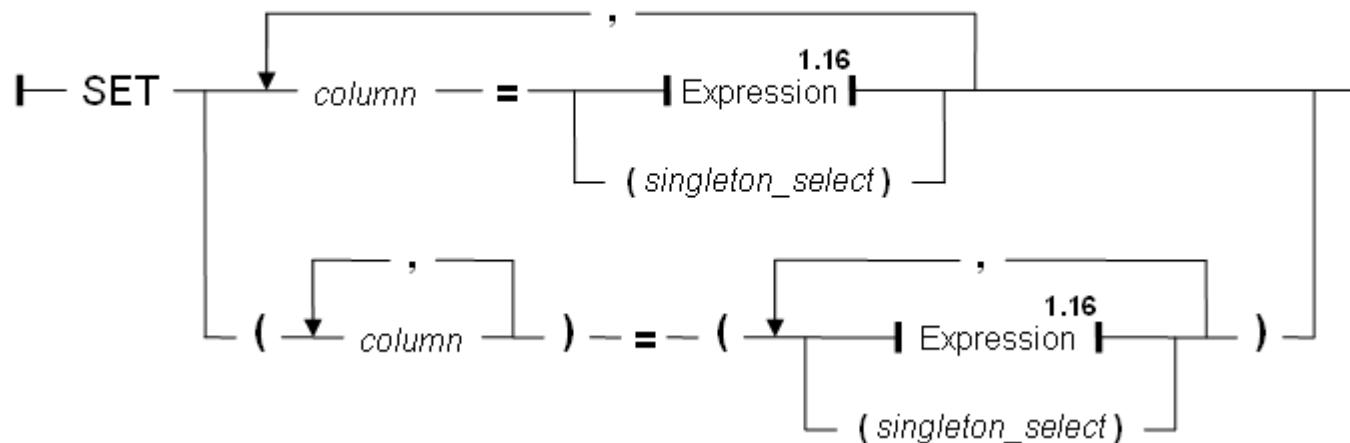
## 3. UPDATE Statement



- UPDATE naredba mijenja vrijednosti atributa onih n-torki relacije *table* za koje se uvjet naveden u WHERE dijelu naredbe izračuna kao *true*
  - ako se WHERE dio naredbe ne navede, mijenjaju se vrijednosti atributa u svim n-torkama relacije *table*
  - u WHERE dijelu naredbe mogu se koristiti svi oblici uvjeta (*Condition*) koji se koriste u WHERE dijelu SELECT naredbe, ali u FROM dijelu poduprta nije dopušteno koristiti relaciju *table*

# UPDATE

## 3.1. SET Clause



- SET dio naredbe određuje nove vrijednosti atributa. Nova vrijednost atributa može biti definirana kao:
  - *Expression* - konstante, NULL, atributi iz relacije *table*, binarni i unarni operatori, funkcije, uvjetni izraz, zgrade
  - *singleton\_select* - jednako kao skalarni podupit (korelirani ili nekorelirani)
    - relaciju *table* nije dopušteno koristiti u FROM dijelu

# UPDATE

- Bodove na međuispitu uvećati za 10 bodova onim studentima koji time ne bi stekli ukupno (bodLab+bodMI) više od 100 bodova

| bodovi | mbr | prez   | bodLab | bodMI |
|--------|-----|--------|--------|-------|
|        | 101 | Novak  | 30     | 55    |
|        | 102 | Polak  | NULL   | 20    |
|        | 103 | Kolar  | 20     | 10    |
|        | 104 | Ban    | 10     | 80    |
|        | 105 | Horvat | 50     | 49    |
|        | 106 | Seljan | 10     | NULL  |

```
UPDATE bodovi
SET bodMI = bodMI + 10
WHERE bodLab + bodMI + 10 <= 100;
```



| bodovi | mbr | prez   | bodLab | bodMI |
|--------|-----|--------|--------|-------|
|        | 101 | Novak  | 30     | 65    |
|        | 102 | Polak  | NULL   | 20    |
|        | 103 | Kolar  | 20     | 20    |
|        | 104 | Ban    | 10     | 90    |
|        | 105 | Horvat | 50     | 49    |
|        | 106 | Seljan | 10     | NULL  |

# UPDATE

- Bodove na međuispitu uvećati za 2 boda studentima koji time ne bi stekli više od 50 bodova za međuispit, bodove za laboratorij povećati za 3 boda onim studentima koji time ne bi stekli ukupno više od 40 bodova za laboratorij

```
UPDATE bodovi SET
 bodMI = CASE
 WHEN bodMI + 2 <= 50
 THEN bodMI + 2
 ELSE bodMI
 END
, bodLab = CASE
 WHEN bodLab + 3 <= 40
 THEN bodLab + 3
 ELSE bodLab
END ;
```

bodovi

| mbr | prez   | bodLab | bodMI |
|-----|--------|--------|-------|
| 101 | Novak  | 30     | 55    |
| 102 | Polak  | NULL   | 20    |
| 103 | Kolar  | 20     | 10    |
| 104 | Ban    | 10     | 80    |
| 105 | Horvat | 50     | 49    |
| 106 | Seljan | 10     | NULL  |

bodovi

| mbr | prez   | bodLab | bodMI |
|-----|--------|--------|-------|
| 101 | Novak  | 33     | 55    |
| 102 | Polak  | NULL   | 22    |
| 103 | Kolar  | 23     | 12    |
| 104 | Ban    | 13     | 80    |
| 105 | Horvat | 50     | 49    |
| 106 | Seljan | 13     | NULL  |

# UPDATE

---

- U sintaksnom dijagramu za *SET Clause* može se vidjeti da postoje dva slična, jednako vrijedna oblika:
  - SET atribut1=vrijednost1, atribut2=vrijednost2, ...
  - SET (atribut1, atribut2, ...)=(vrijednost1, vrijednost2, ...)
- Prethodna UPDATE naredba se mogla napisati na sljedeći način:

```
UPDATE bodovi SET
 (bodMI, bodLab) =
 (CASE
 WHEN bodMI + 2 <= 50
 THEN bodMI + 2
 ELSE bodMI
 END
 , CASE
 WHEN bodLab + 3 <= 40
 THEN bodLab + 3
 ELSE bodLab
 END
) ;
```

# UPDATE

- U relaciji mjesto zamijeniti stare poštanske brojeve i nazive (samo onim mjestima za koje postoji opisani novi poštanski brojevi i nazivi)

| mjesto |           | konverzija |         |           |
|--------|-----------|------------|---------|-----------|
| pbr    | nazMjesto | stariPbr   | noviPbr | noviNaziv |
| 41000  | ZAGREB    | 51400      | 52000   | Pazin     |
| 51400  | PAZIN     | 52000      | 52100   | Pula      |
| 52000  | PULA      | 54000      | 31000   | Osijek    |
| 54000  | OSIJEK    |            |         |           |

```
UPDATE mjesto
SET (pbr, nazMjesto) = (
 (SELECT noviPbr
 FROM konverzija
 WHERE konverzija.stariPbr = mjesto.pbr
)
 , (SELECT noviNaziv
 FROM konverzija
 WHERE konverzija.stariPbr = mjesto.pbr
)
)
WHERE pbr IN (SELECT stariPbr
 FROM konverzija);
```

| mjesto |           |
|--------|-----------|
| pbr    | nazMjesto |
| 41000  | ZAGREB    |
| 52000  | Pazin     |
| 52100  | Pula      |
| 31000  | Osijek    |

# UPDATE

- Nastavnicima koji su na ispitima iz BP podijelili (prosječno) najveće ocjene, povećati plaću za 20000

```
UPDATE nast SET placa = placa + 20000
WHERE sifNast IN (
 SELECT sifNast
 FROM ispitBP
 GROUP BY sifNast
 HAVING AVG(ocjena) >= ALL
 (SELECT AVG(ocjena)
 FROM ispitBP
 GROUP BY sifNast)
);
```

| nast    |        |       |
|---------|--------|-------|
| sifNast | prez   | placa |
| 101     | Novak  | 52000 |
| 102     | Kolar  | 55000 |
| 103     | Horvat | 48000 |
| 104     | Ban    | 57000 |

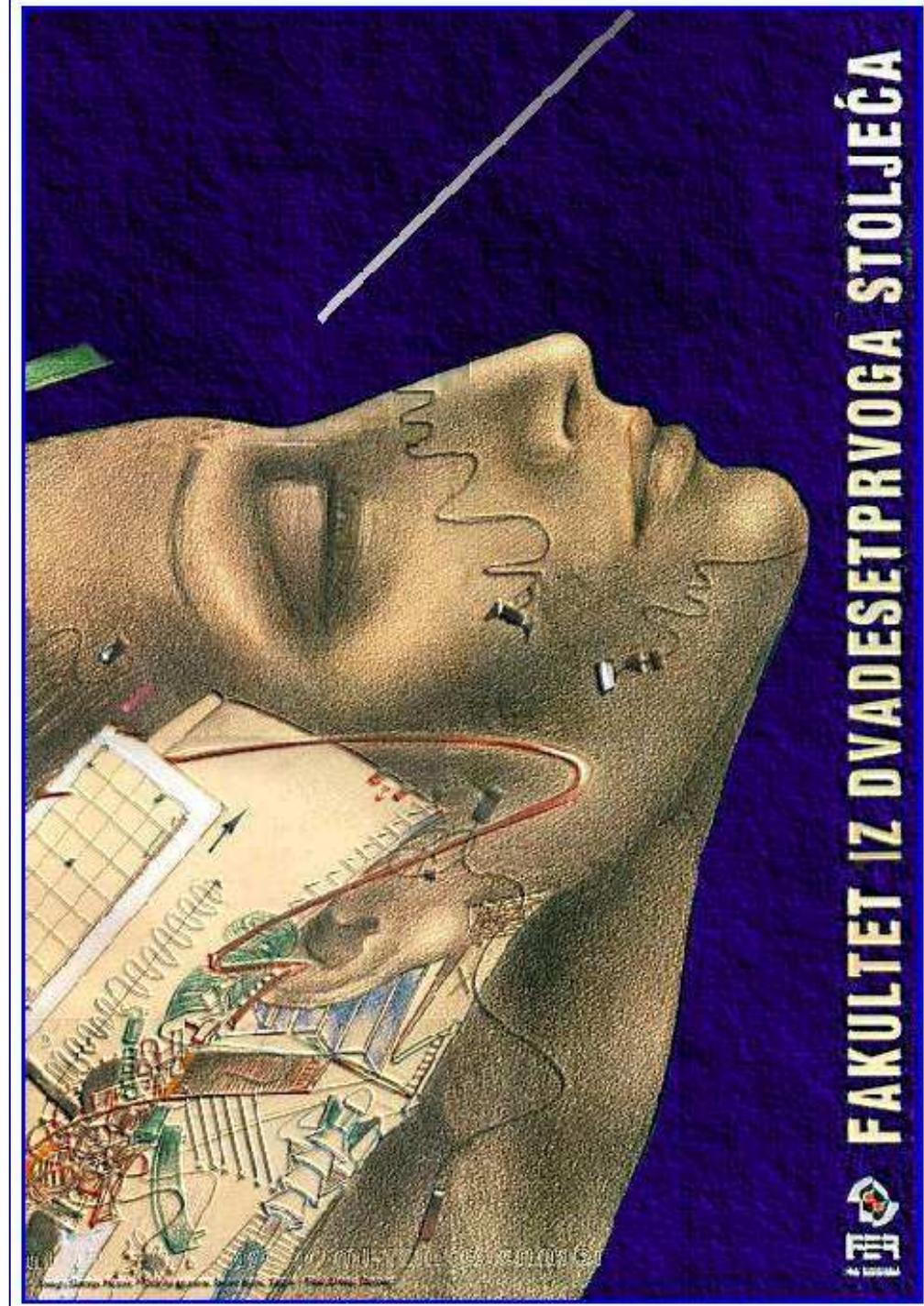
| ispitBP |        |         |
|---------|--------|---------|
| mbrSt   | ocjena | sifNast |
| 1001    | 5      | 101     |
| 1002    | 4      | 101     |
| 1003    | 3      | 101     |
| 1004    | 2      | 102     |
| 1005    | 4      | 102     |
| 1006    | 5      | 103     |
| 1007    | 5      | 103     |
| 1008    | 2      | 103     |
| 1009    | 3      | 104     |

| nast    |        |       |
|---------|--------|-------|
| sifNast | prez   | placa |
| 101     | Novak  | 72000 |
| 102     | Kolar  | 55000 |
| 103     | Horvat | 68000 |
| 104     | Ban    | 57000 |

# Baze podataka

Predavanja  
travanj 2014.

## 6. Oblikovanje sheme relacijske baze podataka (1. dio)



FAKULTET IZ DVADESETPRVOGA STOLJEĆA

# Oblikovanje sheme baze podataka

---

- cilj: oblikovati shemu baze podataka s dobrim svojstvima
- karakteristike loše koncipirane sheme baze podataka:
  - redundancija (čije su posljedice):
    - neracionalno korištenje prostora za pohranu
    - anomalija unosa
    - anomalija izmjene
    - anomalija brisanja
  - pojava lažnih n-torki

# Primjer loše koncipirane sheme baze podataka

- Prodavaonice šalju svoje narudžbe proizvođaču:

|                                                                                                                                                                                                 |                                     |                                                                                                                                                                                              |                                     |                                                                                                                                          |                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| Konzum-7<br>Ilica 20<br>10 000 Zagreb                                                                                                                                                           | Kraš<br>Ravnice bb<br>10 000 Zagreb | Diona-28<br>Bolska 7<br>21 000 Split                                                                                                                                                         | Kraš<br>Ravnice bb<br>10 000 Zagreb | Konzum-7<br>Ilica 20<br>10 000 Zagreb                                                                                                    | Kraš<br>Ravnice bb<br>10 000 Zagreb |
| <b>Narudžba</b><br>br. 13/25<br><b>datum: 1.5.2011</b><br><br>Molimo isporučite nam 1200 komada proizvoda <b>Napolitanke</b> (šifra 129) i 2000 komada proizvoda <b>Albert keks</b> (šifra 139) |                                     | <b>Narudžba</b><br>br. 43-21<br><b>datum: 7.2.2011</b><br><br>Molimo isporučite nam 1200 komada proizvoda <b>Napolitanke</b> (šifra 129) i 1800 komada proizvoda <b>Domaćica</b> (šifra 221) |                                     | <b>Narudžba</b><br>br. 41/56<br><b>datum: 4.2.2012</b><br><br>Molimo isporučite nam 1100 komada proizvoda <b>Napolitanke</b> (šifra 129) |                                     |

- proizvođač želi pohraniti podatke o narudžbama u svoju bazu podataka.  
Svi podaci se pohranjuju u relaciju **narudzbaArtikla**

**narudzbaArtikla**

| nazProd | pbr | nazMjesto | adresa | brNar | datNar | sifArtikl | nazArtikl | kolicina |
|---------|-----|-----------|--------|-------|--------|-----------|-----------|----------|
|         |     |           |        |       |        |           |           |          |

# Neracionalno korištenje prostora za pohranu

- Sadržaj relacije nakon unosa podataka iz prispjelih narudžbi:

narudzbaArtikla

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1100     |

- na više mjesta se ponavlja isti (redundantan) podatak:
  - Konzum-7 je prodavaonica u Zagrebu
  - adresa prodavaonice Konzum-7 je Ilica 20
  - naziv artikla sa šifrom 129 je Napolitanke
  - naziv mjesta s poštanskim brojem 10000 je Zagreb
  - datum narudžbe s brojem 13/25 je 1.5.2011
  - itd.

# Anomalija unosa

narudzbaArtikla

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1100     |

- ne mogu se unijeti podaci o artiklima koje nitko nije naručio
- ne mogu se unijeti podaci o prodavaonicama koje ništa nisu naručile
- ...
- svaki put kad se unosi novi podatak o narudžbi nekog artikla, mora se ponovno upisivati i naziv i mjesto i adresa prodavaonice koja taj artikl naručuje
  - pri tome treba paziti da se podaci za istu prodavaonicu uvijek jednako unesu da bi se zadržala konzistentnost podataka

# Anomalija izmjena

- ako neka prodavaonica promijeni adresu, promjenu adrese potrebno je obaviti na više mjesta da bi se zadržala konzistentnost podataka

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1100     |

- npr. prodavaonica Konzum-7 se preseli jedan kućni broj dalje od centra

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 22 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 22 | 13/25 | 1.5.2011 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 22 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1100     |

# Anomalija brisanja

- brisanjem svih narudžbi za neki artikl gube se podaci o artiklu

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1100     |

- npr. ako se obriše posljednja n-torka o narudžbama artikla Domaćica, podatke o tom artiklu više nećemo imati u bazi podataka

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1100     |

# Pokušaj (neuspješni) popravka sheme baze podataka

- Podaci o narudžbama će se pohranjivati u dvije relacije

$\text{narudzba} = \pi_{\text{nazProd}, \text{pbr}, \text{nazMjesto}, \text{adresa}, \text{brNar}, \text{datNar}, \text{sifArtikl}}(\text{narudzbaArtikla})$

$\text{artikl} = \pi_{\text{sifArtikl}, \text{nazArtikl}, \text{kolicina}}(\text{narudzbaArtikla})$

| narudzba | nazProd | pbr    | nazMjesto | adresa | brNar    | datNar | sifArtikl |
|----------|---------|--------|-----------|--------|----------|--------|-----------|
| Konzum-7 | 10000   | Zagreb | Ilica 20  | 13/25  | 1.5.2011 |        | 129       |
| Konzum-7 | 10000   | Zagreb | Ilica 20  | 13/25  | 1.5.2011 |        | 139       |
| Diona-28 | 21000   | Split  | Bolska 7  | 43-21  | 7.2.2011 |        | 129       |
| Diona-28 | 21000   | Split  | Bolska 7  | 43-21  | 7.2.2011 |        | 221       |
| Konzum-7 | 10000   | Zagreb | Ilica 20  | 41/56  | 4.2.2012 |        | 129       |

| artikl | sifArtikl | nazArtikl   | kolicina |
|--------|-----------|-------------|----------|
|        | 129       | Napolitanke | 1200     |
|        | 139       | Albert keks | 2000     |
|        | 221       | Domaćica    | 1800     |
|        | 129       | Napolitanke | 1100     |

- Ovakva shema baze podataka uzrokovat će pojavu lažnih (*spurious*) n-torki
- dolazi do gubitka informacije!

# Pojava lažnih n-torki

- Obavljanjem operacije narudzba  $\bowtie$  artikl dobije se **više n-torki** nego ih je bilo u relaciji narudzbaArtikla (neke n-torke u rezultatu su "lažne" - označene su zvjezdicom)

narudzbaArtikla<sub>2</sub>

narudzbaArtikla<sub>2</sub> = narudzba  $\bowtie$  artikl  $\neq$  narudzbaArtikla

|   | nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|---|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| * | Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1200     |
|   | Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 129       | Napolitanke | 1100     |
|   | Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2011 | 139       | Albert keks | 2000     |
| * | Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1200     |
|   | Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 129       | Napolitanke | 1100     |
|   | Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2011 | 221       | Domaćica    | 1800     |
| * | Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1100     |
|   | Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2012 | 129       | Napolitanke | 1200     |

- Što bi se dogodilo ako se na temelju relacija narudzba i artikl pokuša izračunati ukupni broj naručenih proizvoda Napolitanke

```
SELECT SUM(kolicina)
 FROM narudzba, artikl
 WHERE narudzba.sifArtikl = artikl.sifArtikl
 AND nazArtikl = 'Napolitanke';
```

# Ispravna shema baze podataka

---

| mjesto |           |
|--------|-----------|
| pbr    | nazMjesto |
| 10000  | Zagreb    |
| 21000  | Split     |

| prodavaonica |       |          |
|--------------|-------|----------|
| nazProd      | pbr   | adresa   |
| Konzum-7     | 10000 | Ilica 20 |
| Diona-28     | 21000 | Bolska 7 |

| artikl    |             |
|-----------|-------------|
| sifArtikl | nazArtikl   |
| 129       | Napolitanke |
| 139       | Albert keks |
| 221       | Domaćica    |

| narudzba |          |          |
|----------|----------|----------|
| brNar    | nazProd  | datNar   |
| 13/25    | Konzum-7 | 1.5.2011 |
| 43-21    | Diona-28 | 7.2.2011 |
| 41/56    | Konzum-7 | 4.2.2012 |

| stavkaNarudzbe |           |          |
|----------------|-----------|----------|
| brNar          | sifArtikl | kolicina |
| 13/25          | 129       | 1200     |
| 13/25          | 139       | 2000     |
| 43-21          | 129       | 1200     |
| 43-21          | 221       | 1800     |
| 41/56          | 129       | 1100     |

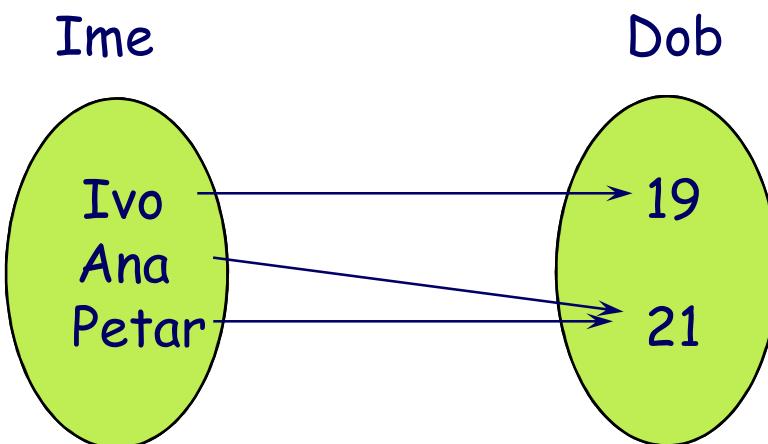
- Za vježbu provjerite
  - postoji li redundancija u ovoj bazi podataka ?
  - je li moguća pojava lažnih n-torki ?

## Kako odrediti zamjenu za loše koncipiranu relacijsku shemu?

---

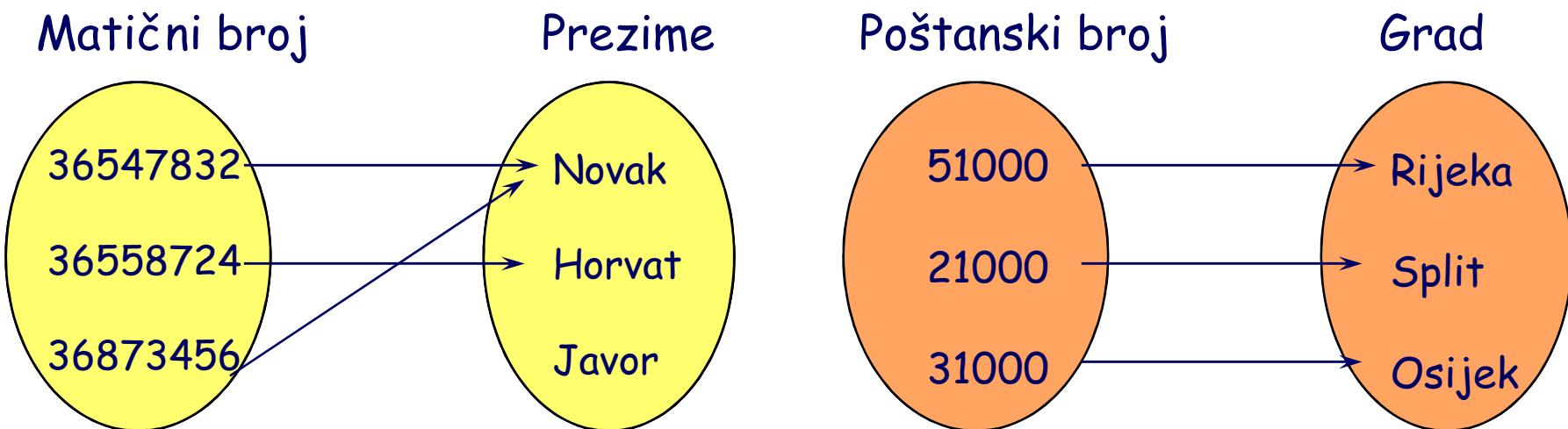
- proučavanjem značenja podataka (semantike)
- proučavanjem zavisnosti među podacima
- uvođenjem ograničenja koja su ovisna o semantici podataka
- najvažnije su **FUNKCIJSKE ZAVISNOSTI**

# Funkcija



## Preslikavanje kod kojeg vrijedi:

svakom članu skupa **Ime**  
pridružen je jedan i samo  
jedan član skupa **Dob**



# Ponavljanje: X-vrijednost n-torke

- Neka je  $X \subseteq R$ . n-torka t reducirana na skup atributa X naziva se X-vrijednost n-torke t i označava s  $t(X)$
- Primjer:

$t = \{ \text{matBr:102, prez:Novak, ime: Marko} \}$

$X = \{ \text{matBr, prez} \} \quad X \subseteq R$

$t(X) = t(\{ \text{matBr, prez} \}) = \{ \text{matBr:102, prez:Novak} \}$

| osoba | matBr | prez  | ime   |
|-------|-------|-------|-------|
|       | 101   | Kolar | Josip |
| $t$   | 102   | Novak | Marko |

A diagram illustrating the reduction of a relation  $t$  to its  $X$ -value. A blue arrow points from the label  $t$  to the second row of the table. A red arrow points from the label  $t(X)$  to the third row of the table, which contains the values for attributes matBr and prez only.

# Funkcijske zavisnosti - definicija

---

- Neka je  $r$  relacija sa shemom  $R$  i neka su  $X$  i  $Y$  skupovi atributa,  $X \subseteq R$ ,  $Y \subseteq R$

**Funkcijska zavisnost  $X \rightarrow Y$  vrijedi na shemi  $R$  ukoliko**  
u svim dopuštenim stanjima relacije  $r(R)$  svaki par n-torki  $t_1$  i  $t_2$  koje imaju jednake  $X$ -vrijednosti, također imaju jednake  $Y$ -vrijednosti, odnosno:

$$t_1(X) = t_2(X) \Rightarrow t_1(Y) = t_2(Y)$$

Kratica za funkciju zavisnost je FZ.

# Funkcijske zavisnosti - primjer

---

- relacija osoba(OSOBA)

| osoba | matBr | prezime | ime   | postBr | grad   |
|-------|-------|---------|-------|--------|--------|
|       | 11234 | Novak   | Josip | 21000  | Split  |
|       | 12345 | Horvat  | Ivan  | 10000  | Zagreb |
|       | 22211 | Kolar   | Ante  | 21000  | Split  |
|       | 33345 | Ban     | Tomo  | 31000  | Osijek |
|       | 23456 | Kolar   | Ana   | 31000  | Osijek |

- Funkcijska zavisnost **postBr → grad** vrijedi na shemi OSOBA jer svaki par n-torki koje imaju jednake vrijednosti atributa **postBr** također imaju jednake vrijednosti atributa **grad** (i to vrijedi ne samo za trenutačno stanje relacije, nego za sva dopuštena stanja relacije)
- Vrijedi li funkcija zavisnost **prezime → postBr?**

# Funkcijske zavisnosti

---

- **Funkcijske zavisnosti proizlaze iz značenja podataka (semantike), a ne iz trenutačnog stanja relacije!**
- Primjer: relacija osoba(OSOBA)

| osoba | matBr | prezime | ime   | pbr |
|-------|-------|---------|-------|-----|
| 11234 | Kolar | Ante    | 21000 |     |
| 22211 | Kolar | Ante    | 31000 |     |
| 33345 | Ban   | Tomo    | 10000 |     |

- promatranjem samo trenutačnog stanja relacije mogli bismo (**pogrešno!**) zaključiti da vrijedi FZ **prezime → ime**
- **međutim**, poznavanjem značenja podataka u relaciji možemo zaključiti da je u gore prikazanu relaciju dopušteno unijeti n-torku **< 76555, Kolar, Zrinka, 51000 >**
- ⇒ FZ **prezime → ime** ne vrijedi na shemi OSOBA

# Priroda funkcijskih zavisnosti

---

- Postojanje funkcijске zavisnosti ne može se dokazati na temelju postojećih podataka u relaciji.
- Analizom postojećih podataka u relaciji moguće je tek **pretpostaviti** da bi funkcijска zavisnost mogla vrijediti.
- Dokaz za postojanje FZ treba tražiti u **značenju** pojedinih atributa.

# Priroda funkcijskih zavisnosti

$$R = \{ A, B, C \}$$

| r(R) | A | B        | C |
|------|---|----------|---|
|      | a | $\alpha$ | 1 |
|      | b | $\gamma$ | 1 |
|      | b | $\alpha$ | 1 |
|      | c | $\alpha$ | 3 |
|      | a | $\alpha$ | 1 |

| r(R) | A | B        | C |
|------|---|----------|---|
|      | a | $\alpha$ | 1 |
|      | b | $\gamma$ | 1 |
|      | b | $\alpha$ | 1 |
|      | c | $\alpha$ | 3 |
|      | a | $\alpha$ | 1 |

Vrijedi li FZ  $AB \rightarrow C$  na shemi R?

- moguće je da vrijedi, ali to ne možemo sa sigurnošću tvrditi
- bez poznavanja značenja atributa A, B i C, ne možemo zaključiti koje funkcijске zavisnosti zaista vrijede na shemi R

Vrijedi li FZ  $BC \rightarrow A$  na shemi R?

**Sa sigurnošću možemo tvrditi: NE**

# Priroda funkcijskih zavisnosti

---

- Ako u relacijskoj shemi R vrijedi FZ  $X \rightarrow Y$ , relacija  $r(R)$  ne može sadržavati dvije n-torce koje imaju jednake X-vrijednosti i različite Y-vrijednosti
- Primjer: ako u relacijskoj shemi  
 $R = \{ \text{matBr}, \text{ prezime}, \text{ grad}, \text{ telefon} \}$   
vrijedi FZ  $\text{matBr} \rightarrow \text{ prezime}$  tada relacija  $r(R)$  ne smije sadržavati dvije n-torce s istim matičnim brojem i različitim prezimenom

# Priroda funkcijskih zavisnosti - primjer

| ispit | mbr | sifPred | datIspit  | sifNast | ocjena |
|-------|-----|---------|-----------|---------|--------|
|       | 101 | 10      | 30.1.2006 | 1003    | 1      |
|       | 101 | 10      | 15.1.2007 | 1002    | 4      |
|       | 102 | 10      | 30.1.2006 | 1001    | 3      |
|       | 102 | 11      | 15.1.2006 | 1002    | 5      |

- studenta  $mbr$  je na ispitu iz predmeta  $sifPred$  na datum  $datIspit$  nastavnik  $sifNast$  ocijenio ocjenom  $ocjena$
- vrijedi li FZ  $mbr \ sifNast \rightarrow ocjena$ 
  - ne, jer bi to značilo da nastavnik  $x$  studentu  $y$  uvijek mora dati istu ocjenu
- vrijedi li FZ  $mbr \ sifPred \rightarrow ocjena$ 
  - ne, jer bi to značilo da student  $x$  iz predmeta  $y$  mora dobiti uvijek istu ocjenu
- vrijedi li FZ  $mbr \ datIspit \rightarrow ocjena$ 
  - ne, jer bi to značilo da student  $x$  na datum  $y$  mora uvijek mora dobiti sve jednake ocjene
- vrijedi li FZ  $mbr \ sifPred \ sifNast \rightarrow ocjena$  NE (Zašto?)
- vrijedi li FZ  $mbr \ sifPred \ datIspit \rightarrow ocjena$  DA (Zašto?)

# Funkcijske zavisnosti - SQL primjer

| ispit | mbr | sifPred | datIspit  | sifNast | ocjena |
|-------|-----|---------|-----------|---------|--------|
|       | 101 | 10      | 30.1.2006 | 1003    | 1      |
|       | 101 | 10      | 15.1.2007 | 1002    | 4      |
|       | 102 | 10      | 30.1.2006 | 1001    | 3      |
|       | 102 | 11      | 15.1.2006 | 1002    | 5      |

- pomoću SELECT naredbe ispitati bi li u relaciji ispit eventualno mogla vrijediti FZ  
 $mbr \ sifNast \rightarrow ocjena \ datIspit$
- ispituju se svi parovi n-torki  $t_1, t_2$  koje imaju jednake X-vrijednosti (u primjeru  $X = \{ mbr, sifNast \}$ )
- ako postoji par n-torki  $t_1$  i  $t_2$  koje imaju iste X-vrijednosti, a različite Y-vrijednosti (u primjeru  $Y = \{ ocjena, datIspit \}$ ), tada FZ sigurno ne vrijedi

```
SELECT *
 FROM ispit AS t1, ispit AS t2
 WHERE t1.mbr = t2.mbr
 AND t1.sifNast = t2.sifNast
 AND (t1.ocjena <> t2.ocjena
 OR t1.datIspit <> t2.datIspit);
```

n-torke  $t_1$  i  $t_2$   
koje imaju  
jednake  
X-vrijednosti ... }  
... a različite  
Y-vrijednosti }

- ako takve n-torke ne postoje, onda FZ **možda** vrijedi

# Armstrongovi aksiomi

---

- Projektant sheme baze podataka specificira FZ koje su mu semantički očite, no obično vrijede i brojne druge FZ koje mogu biti izvedene iz početnih FZ. Korištenjem Armstrongovih aksioma izvode se nove FZ.

## ARMSTRONGOVI AKSIOMI

Neka je  $R$  relacijska shema, neka su  $X, Y, Z$  skupovi atributa i neka vrijedi:

$$X \subseteq R, Y \subseteq R, Z \subseteq R$$

### A-1 REFLEKSIVNOST

- Ako je  $Y \subseteq X$ , tada vrijedi  $X \rightarrow Y$

### A-2 UVEĆANJE

- Ako u shemi  $R$  vrijedi  $X \rightarrow Y$ , tada vrijedi i  $XZ \rightarrow Y$

### A-3 TRANZITIVNOST

- Ako u shemi  $R$  vrijedi  $X \rightarrow Y$  i  $Y \rightarrow Z$ , tada vrijedi i  $X \rightarrow Z$

# Armstrongovi aksiomi

---

## A-1 REFLEKSIVNOST

- **Ako je  $Y \subseteq X$ , tada vrijedi  $X \rightarrow Y$** 
  - uvijek vrijedi  $X \rightarrow X$

PRIMJER:

osoba(OSOBA)

|       | matBr  | prezime | ime   | postBr | grad |
|-------|--------|---------|-------|--------|------|
| 11234 | Novak  | Josip   | 21000 | Split  |      |
| 12345 | Horvat | Ivan    | 10000 | Zagreb |      |
| 23456 | Kolar  | Ana     | 31000 | Osijek |      |
| 34567 | Novak  | Josip   | 31000 | Osijek |      |

$$X = \{ \text{prezime, ime} \} \quad Y = \{ \text{prezime} \}$$

$Y \subseteq X \Rightarrow$  u relaciji osoba vrijedi i FZ **prezime ime  $\rightarrow$  prezime**

$X \subseteq X \Rightarrow$  u relaciji osoba vrijedi i FZ **prezime ime  $\rightarrow$  prezime ime**

# Armstrongovi aksiomi

---

## A-2 UVEĆANJE

- **Ako u shemi R vrijedi  $X \rightarrow Y$ , tada vrijedi i  $XZ \rightarrow Y$** 
  - možemo uvećati lijevu stranu funkcijске zavisnosti

PRIMJER:

| osoba(OSOBA) | matBr | prezime | ime   | postBr | grad   |
|--------------|-------|---------|-------|--------|--------|
|              | 11234 | Novak   | Josip | 21000  | Split  |
|              | 12345 | Horvat  | Ivan  | 10000  | Zagreb |
|              | 23456 | Kolar   | Ana   | 31000  | Osijek |
|              | 34567 | Novak   | Josip | 31000  | Osijek |

U relaciji *osoba* vrijedi FZ **matBr  $\rightarrow$  ime**

$\Rightarrow$  u relaciji *osoba* vrijedi i FZ **matBr prezime  $\rightarrow$  ime**

$\Rightarrow$  u relaciji *osoba* vrijedi i FZ **matBr prezime grad  $\rightarrow$  ime**

# Armstrongovi aksiomi

---

## A-3 TRANZITIVNOST

- **Ako u shemi R vrijedi  $X \rightarrow Y$  i  $Y \rightarrow Z$ , tada vrijedi i  $X \rightarrow Z$** 
  - $X \rightarrow Z$  je tranzitivna zavisnost

PRIMJER:

| osoba(OSOBA) | matBr | prezime | ime   | postBr | grad   |
|--------------|-------|---------|-------|--------|--------|
|              | 11234 | Novak   | Josip | 21000  | Split  |
|              | 12345 | Horvat  | Ivan  | 10000  | Zagreb |
|              | 23456 | Kolar   | Ana   | 31000  | Osijek |
|              | 34567 | Novak   | Josip | 31000  | Osijek |

U relaciji *osoba* vrijede FZ **matBr → postBr** i **postBr → grad**

⇒ u relaciji *osoba* vrijedi i FZ **matBr → grad**

# Pravila koja proizlaze iz Armstrongovih aksioma

---

Neka je  $R$  relacijska shema, neka su  $X, Y, Z, V$  skupovi atributa i neka vrijedi:

$$X \subseteq R, Y \subseteq R, Z \subseteq R, V \subseteq R$$

## P-1 PRAVILO UNIJE (pravilo o aditivnosti)

- Ako u shemi  $R$  vrijedi  $X \rightarrow Y$  i  $X \rightarrow Z$ , tada vrijedi i  $X \rightarrow YZ$

## P-2 PRAVILO DEKOMPOZICIJE (pravilo o projektivnosti)

- Ako u shemi  $R$  vrijedi  $X \rightarrow YZ$ , tada vrijedi i  $X \rightarrow Y$

## P-3 PRAVILO O PSEUDOTRANZITIVNOSTI

- Ako u shemi  $R$  vrijedi  $X \rightarrow Y$  i  $Y \rightarrow Z$ , tada vrijedi i  $X \rightarrow Z$

# Pravila koja proizlaze iz Armstrongovih aksioma

---

## P-1 PRAVILO UNIJE (pravilo o aditivnosti)

- Ako u shemi R vrijedi  $X \rightarrow Y$  i  $X \rightarrow Z$ , tada vrijedi i  $X \rightarrow YZ$

PRIMJER:

osoba(OSOBA)

| matBr | prezime | ime   | postBr | grad   |
|-------|---------|-------|--------|--------|
| 11234 | Novak   | Josip | 21000  | Split  |
| 12345 | Horvat  | Ivan  | 10000  | Zagreb |
| 23456 | Kolar   | Ana   | 31000  | Osijek |
| 34567 | Novak   | Josip | 31000  | Osijek |

U relaciji *osoba* vrijede FZ **matBr  $\rightarrow$  ime** i **matBr  $\rightarrow$  prezime**

$\Rightarrow$  u relaciji *osoba* vrijedi i FZ **matBr  $\rightarrow$  ime prezime**

# Pravila koja proizlaze iz Armstrongovih aksioma

---

## P-2 PRAVILO DEKOMPOZICIJE (pravilo o projektivnosti)

- **Ako u shemi R vrijedi  $X \rightarrow YZ$ , tada vrijedi i  $X \rightarrow Y$**

PRIMJER:

| osoba(OSOBA) | matBr | prezime | ime   | postBr | grad   |
|--------------|-------|---------|-------|--------|--------|
|              | 11234 | Novak   | Josip | 21000  | Split  |
|              | 12345 | Horvat  | Ivan  | 10000  | Zagreb |
|              | 23456 | Kolar   | Ana   | 31000  | Osijek |
|              | 34567 | Novak   | Josip | 31000  | Osijek |

U relaciji *osoba* vrijedi FZ **matBr → ime prezime**

- ⇒ u relaciji *osoba* vrijedi i FZ **matBr → ime**
- ⇒ u relaciji *osoba* vrijedi i FZ **matBr → prezime**

# Pravila koja proizlaze iz Armstrongovih aksioma

---

## P-3 PRAVILO PSEUDOTRANZITIVNOSTI

- Ako u shemi R vrijedi  $X \rightarrow Y$  i  $Y \rightarrow Z$ , tada vrijedi i  $X \rightarrow Z$

PRIMJER:

zaposlenje(ZAPOSLENJE)

| jmbg | strSprema | funkcija | zaposlOd | zaposlDo   | placa |
|------|-----------|----------|----------|------------|-------|
| 101  | VSS       | direktor | 1.1.2006 | 31.12.2007 | 10000 |
| 101  | VSS       | tajnik   | 1.1.2008 | 31.12.2008 | 8000  |
| 102  | VŠS       | direktor | 1.1.2009 | 31.12.2009 | 9000  |
| 102  | VŠS       | tajnik   | 1.1.2006 | 31.12.2007 | 7000  |
| 103  | VSS       | direktor | 1.1.2010 | 31.12.2010 | 10000 |
| 101  | VSS       | direktor | 1.1.2011 | 31.12.2011 | 10000 |

U relaciji zaposlenje vrijede FZ

**jmbg → strSprema** i **funkcija strSprema → placa**

⇒ u relaciji zaposlenje vrijedi i FZ **jmbg funkcija → placa**

# Primjer korištenja aksioma i pravila

---

Uz pretpostavku da na relacijskoj shemi  $R = \{ A, B, C, D, E \}$  vrijedi skup funkcijskih zavisnosti  $F = \{ A \rightarrow BD, B \rightarrow C, D \rightarrow E \}$ , dokazati da vrijedi FZ  $AE \rightarrow AC$ .

Dokaz:

- $A \rightarrow BD$  (P2: dekompozicija)  $\Rightarrow A \rightarrow B$
- $A \rightarrow B \wedge B \rightarrow C$  (A3: tranzitivnost)  $\Rightarrow A \rightarrow C$
- (A1: refleksivnost)  $\Rightarrow A \rightarrow A$
- $A \rightarrow A \wedge A \rightarrow C$  (P1: unija)  $\Rightarrow A \rightarrow AC$
- $A \rightarrow AC$  (A2: uvećanje)  $\Rightarrow AE \rightarrow AC$

# Pravilo o akumulaciji

---

Sljedeće dodatno pravilo omogućuje "algoritamski" pristup rješavanju sličnih zadataka

## PRAVILO O AKUMULACIJI

- **Ako u shemi R vrijedi**
  - **$X \rightarrow VZ$  i  $Z \rightarrow W$ , tada vrijedi i  $X \rightarrow VZW$**

# Primjer korištenja pravila o akumulaciji

---

Uz pretpostavku da na relacijskoj shemi  $R = \{ A, B, C, D, E \}$  vrijedi skup funkcijskih zavisnosti  $F = \{ A \rightarrow BD, B \rightarrow C, D \rightarrow E \}$ , dokazati da vrijedi FZ  $AE \rightarrow AC$ .

Označimo lijevu stranu FZ s  $X$  ( $X=AE$ ), a desnu stranu FZ s  $Y$  ( $Y=AC$ ).

Dokaz (primjenom A-1, pravila o akumulaciji i P-2):

1. korak:  $X \rightarrow X$

- (A1: refleksivnost)  $\Rightarrow AE \rightarrow AE$

u sljedećim koracima pomoći pravila akumulacije "uvećavati desnu stranu FZ" sve dok desna strana ne sadrži  $Y$

2. { ▪  $AE \rightarrow AE \wedge A \rightarrow BD$  (akumulacija)  $\Rightarrow AE \rightarrow AE BD$

3. { ▪  $AE \rightarrow AE BD \wedge B \rightarrow C$  (akumulacija)  $\Rightarrow AE \rightarrow AE BDC$

u zadnjem koraku, kad (i ako) desna strana FZ sadrži  $Y$

4. { ▪  $AE \rightarrow AE BDC$  (P2: dekompozicija)  $\Rightarrow AE \rightarrow AC$

## Primjer korištenja pravila o akumulaciji (za vježbu 1)

---

$R = \{ L, M, N, P, Q, R \}$ ,  $F = \{ Q \rightarrow R, M \rightarrow PQ, PQL \rightarrow N \}$   
dokazati da vrijedi FZ  $MLR \rightarrow QN$ .

- (A1: refleksivnost)  $\Rightarrow MLR \rightarrow MLR$
- $MLR \rightarrow MLR \wedge M \rightarrow PQ$  (akumulacija)  $\Rightarrow MLR \rightarrow MLRPQ$
- $MLR \rightarrow MLRPQ \wedge PQL \rightarrow N$  (akumulacija)  $\Rightarrow MLR \rightarrow MLRPQN$
- $MLR \rightarrow MLRPQN$  (P2: dekompozicija)  $\Rightarrow MLR \rightarrow QN$

## Primjer korištenja pravila o akumulaciji (za vježbu 2)

---

$R = \{ L, M, N, P, Q, R \}$ ,  $F = \{ Q \rightarrow R, M \rightarrow PQ, PQL \rightarrow N \}$   
dokazati da vrijedi FZ  $MQ \rightarrow LN$ .

- (A1: refleksivnost)  $\Rightarrow MQ \rightarrow MQ$
- $MQ \rightarrow MQ \wedge Q \rightarrow R$  (akumulacija)  $\Rightarrow MQ \rightarrow MQR$
- $MQ \rightarrow MQR \wedge M \rightarrow PQ$  (akumulacija)  $\Rightarrow MQ \rightarrow MQRP$
- ne postoji FZ kojom bi se moglo nastaviti "uvećavati desnu stranu"
- $\Rightarrow MQ \rightarrow LN$  ne vrijedi

# Ključ entiteta, ključ relacije

---

- entitet je bilo što, što ima suštinu ili bit i posjeduje značajke s pomoću kojih se može razlučiti od svoje okoline
- ključ entiteta sadrži one atributi koji omogućuju da se pojedini entiteti mogu razlučiti od okoline
- relacijom se opisuje skup entiteta

Ključ relacije je skup atributa koji nedvosmisleno određuje n-torke relacije.

- Ključ relacije ima svojstvo da funkcijски određuje atribute u preostalom dijelu relacije

# Ključ relacije

---

- ključ relacijske sheme  $R$  je skup atributa  $K$ ,  $K \subseteq R$ , koji ima sljedeća svojstva:
  1.  $K \rightarrow (R \setminus K)$  (također vrijedi i  $K \rightarrow R$ )
    - ključ funkcijски određuje attribute u preostalom dijelu relacijske sheme
  2. ne postoji  $K' \subset K$  za kojeg vrijedi  $K' \rightarrow R$ 
    - ključ je minimalan skup atributa koji funkcijски određuje attribute u preostalom dijelu relacijske sheme

# Ključ relacije - primjer

---

| osoba | matBr | prezime | ime   | postBr | grad   |
|-------|-------|---------|-------|--------|--------|
|       | 11234 | Novak   | Josip | 21000  | Split  |
|       | 12345 | Horvat  | Ivan  | 10000  | Zagreb |
|       | 23456 | Kolar   | Ana   | 31000  | Osijek |
|       | 34567 | Novak   | Josip | 10000  | Zagreb |

Ključ:  $K_{\text{OSOBA}} = \{ \text{matBr} \}$

$\text{matBr} \rightarrow \text{prezime}$

$\text{matBr} \rightarrow \text{ime}$

$\text{matBr} \rightarrow \text{postBr}$

$\text{matBr} \rightarrow \text{grad}$

Za  $K = \{ \text{matBr}, \text{prezime} \}$  također vrijedi

$K \rightarrow \{ \text{ime}, \text{postBr}, \text{grad} \},$

ali  $K$  **nije** ključ jer postoji  $K' = \{ \text{matBr} \}$ ,  $K' \subset K$ , za kojeg vrijedi

$K' \rightarrow \{ \text{prezime}, \text{ime}, \text{postBr}, \text{grad} \}$

# Ključevi relacije

---

- mogući ključevi (*candidate key*)
- primarni ključ (*primary key*) odabire se jedan od mogućih ključeva
- alternativni ključevi (*alternate key*) ostali mogući ključevi

PRIMJER:

| djelatnik | matBr | prezime | ime   | JMBG          |
|-----------|-------|---------|-------|---------------|
|           | 11234 | Novak   | Josip | 1403970330103 |
|           | 12345 | Horvat  | Ivan  | 2812964310267 |
|           | 23456 | Kolar   | Ana   | 0111959335208 |
|           | 34567 | Novak   | Josip | 0301949320319 |

- mogući ključevi:
  - { matBr }
  - { JMBG }
- primarni ključ: { matBr }
- alternativni ključ: { JMBG }

# Struktura relacije

---

- Relacijska shema sastoji se od:
  - atributa koji su dio ključa (**ključni atributi, ključni dio relacije**)
  - atributa iz zavisnog dijela relacije (**neklučni atributi, neključni dio relacije**)

PRIMJER:

| djelatnik | matBr | prezime | ime   | JMBG          |
|-----------|-------|---------|-------|---------------|
|           | 11234 | Novak   | Josip | 1403970330103 |
|           | 12345 | Horvat  | Ivan  | 2812964310267 |
|           | 23456 | Kolar   | Ana   | 0111959335208 |
|           | 34567 | Novak   | Josip | 0301949320319 |

- primarni ključ: { matBr }
- alternativni ključ: { JMBG }

- ključni atributi, ključni dio relacije:
  - matBr
  - JMBG
- neključni atributi, neključni dio relacije:
  - prezime
  - ime

# Zadatak:

---

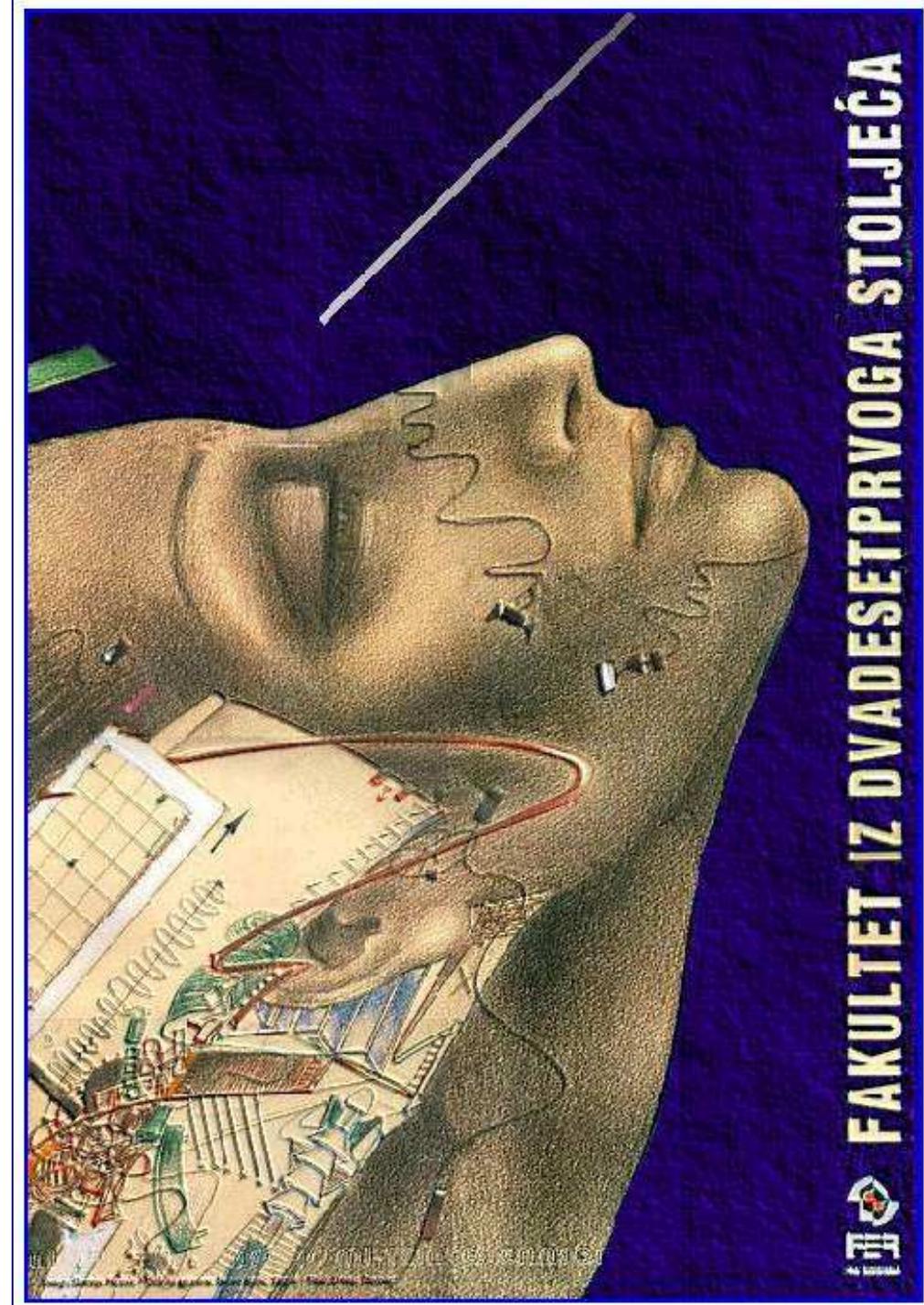
- Odrediti moguće ključeve, primarni ključ, alternativne ključeve, ključni dio relacije, neključni dio relacije
  - uzeti u obzir da klub tijekom istog dana može igrati najviše jednu utakmicu

| utakmicaPrvenstva | domaci    | gosti     | datum      | rezultat |
|-------------------|-----------|-----------|------------|----------|
|                   | Arsenal   | Liverpool | 12.06.2007 | 2:1      |
|                   | Arsenal   | Liverpool | 08.03.2008 | 2:1      |
|                   | Newcastle | Everton   | 08.03.2008 | 3:3      |
|                   | Everton   | Liverpool | 22.03.2008 | 4:0      |
|                   | Liverpool | Everton   | 05.04.2008 | 5:2      |

# Baze podataka

Predavanja  
travanj 2014.

## 7. Oblikovanje sheme relacijske baze podataka (2. dio)



FAKULTET IZ DVADESETPRVOGA STOLJEĆA



# Normalizacija

# Postupci normalizacije

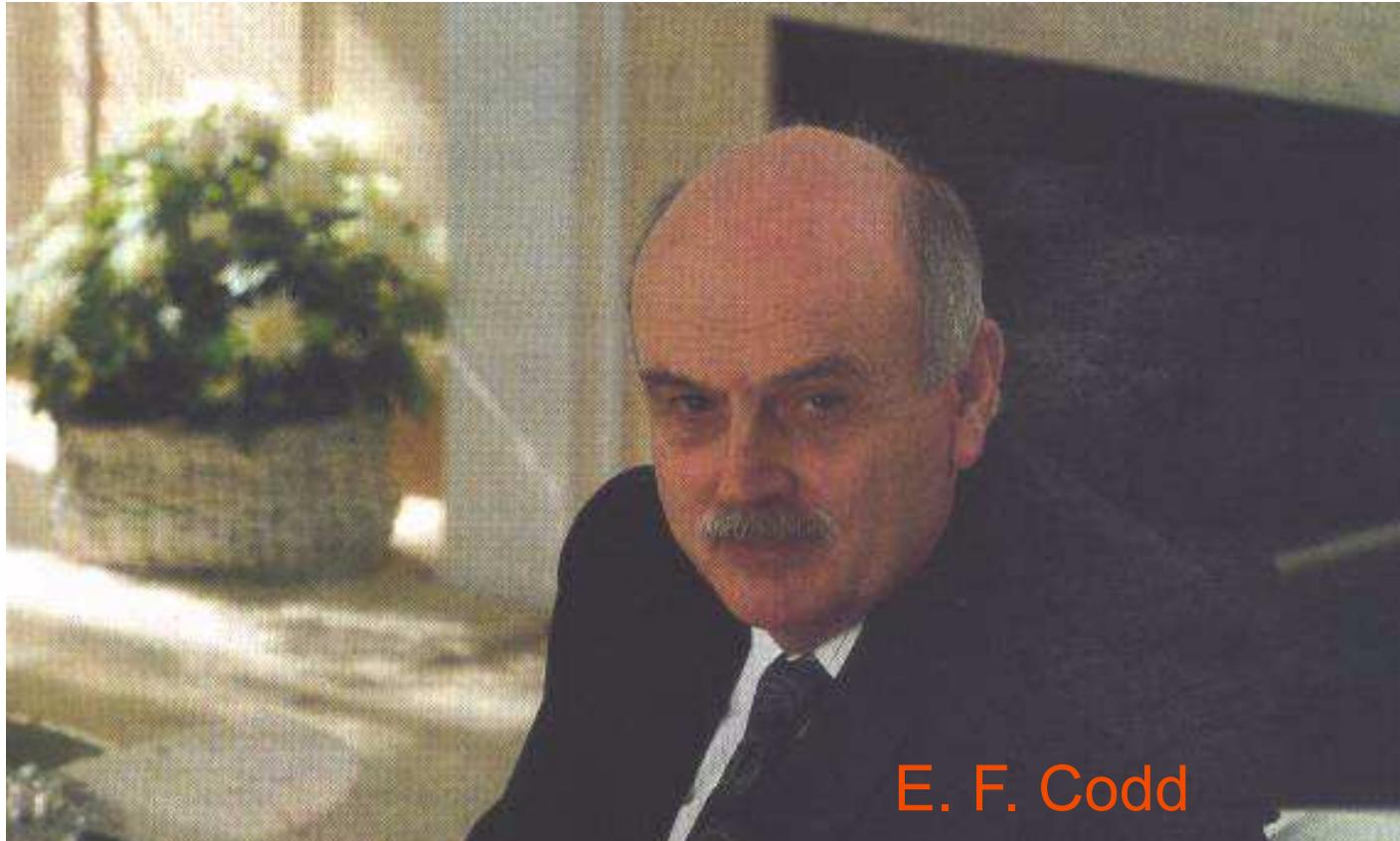
---

- Uočena znanja o međusobnim **funkcijskim zavisnostima** atributa relacije koriste se u postupcima normalizacije.
- **Cilj:**
  - **ukloniti redundanciju**
    - anomalije unosa, izmjene i brisanja
    - neracionalno korištenje prostora za pohranu
  - **spriječiti pojavu lažnih n-torki**
- Postupci normalizacije omogućavaju da se postupno, **točno definiranom metodom**, odredi dobra zamjena za loše koncipiranu relacijsku shemu

**E. F. Codd:**

**“Normalized data base structure: A brief tutorial”**

*Proc. ACM SIGFIDET Workshop on Data Description, Access and Control, 1971*



E. F. Codd

"I called it normalization because then-President Nixon was talking a lot about normalizing relations with China. I figured that if he could normalize relations, so could I."  
('A "FIRESIDE" CHAT', DBMS, Dec. 1993)

# Normalne forme

---

- Prva normalna forma - 1 NF
- Druga normalna forma - 2 NF
- Treća normalna forma - 3 NF
- Boyce-Coddova normalna forma - BCNF

Temelje se na FUNKCIJSKIM ZAVISNOSTIMA

---

- Četvrta normalna forma - 4NF  
Temelji se na VIŠEZNAČNIM ZAVISNOSTIMA
- Projekcijsko-spojna normalna forma - PJNF  
Temelji se na SPOJNIM ZAVISNOSTIMA

# Postupci normalizacije

---

- Dekompozicija
  - početne relacije (relacijske sheme) se dekomponiraju na temelju uočenih funkcijskih zavisnosti
- Sinteza
  - zadan je skup atributa i nad njima skup funkcijskih zavisnosti iz kojih se sintetiziraju relacijske sheme koje zadovoljavaju 3NF

# Dekompozicija relacijske sheme (relacije)

---

- Dekompozicijom (razlaganjem) relacijska shema  $R$  zamjenjuje se shemama  $R_1, R_2, \dots, R_n$ ,  $R_i \subseteq R$ , pri čemu vrijedi  $R = R_1 R_2 \dots R_n$
- Dekompozicijom se relacija  $r(R)$  zamjenjuje relacijama  $r_1(R_1), r_2(R_2), \dots, r_n(R_n)$ , pri čemu je  $r_i(R_i) = \pi_{R_i}(r)$ , za  $i = 1, \dots, n$
- Relacija  $r(R)$  se dekomponira na relacije  $r_1(R_1), r_2(R_2), \dots, r_n(R_n)$  **bez gubitaka informacija (lossless decomposition)** ako vrijedi:

$$r_1(R_1) \bowtie r_2(R_2) \bowtie \dots \bowtie r_n(R_n) = r(R)$$

odnosno

$$\pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \dots \pi_{R_n}(r) = r(R)$$

# Dekompozicija relacije - primjer

- Zadana je relacija:

| r(R) |    |    |    |
|------|----|----|----|
| A    | B  | C  | D  |
| a1   | b1 | c1 | d1 |
| a2   | b2 | c1 | d1 |

- Relaciju  $r(R)$  dekomponirati na relacije

- $r_1(R_1), R_1 = \{ A, C \}$
- $r_2(R_2), R_2 = \{ B, C \}$
- $r_3(R_3), R_3 = \{ C, D \}$

| r <sub>1</sub> (R <sub>1</sub> ) |    | r <sub>2</sub> (R <sub>2</sub> ) |    | r <sub>3</sub> (R <sub>3</sub> ) |    |
|----------------------------------|----|----------------------------------|----|----------------------------------|----|
| A                                | C  | B                                | C  | C                                | D  |
| a1                               | c1 | b1                               | c1 | c1                               | d1 |
| a2                               | c1 | b2                               | c1 |                                  |    |

- Je li dekompozicija obavljena bez gubitaka informacija?

$r_1(R_1) \bowtie r_2(R_2) \bowtie r_3(R_3)$

| A  | B  | C  | D  |
|----|----|----|----|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c1 | d1 |
| a2 | b1 | c1 | d1 |
| a2 | b2 | c1 | d1 |

$\Rightarrow NE$

# Razlaganje relacije bez gubitaka na dvije projekcije

- Relacija se bez gubitaka razlaže na svoje dvije projekcije ako:
  - projekcije imaju zajedničke attribute
  - zajednički atributi su ključ u barem jednoj od projekcija

PRIMJER:

| osoba | matBr  | prez  | ime   | postBr | nazMj |
|-------|--------|-------|-------|--------|-------|
| 11234 | Novak  | Josip | 21000 | Split  |       |
| 12345 | Horvat | Ivan  | 10000 | Zagreb |       |
| 23456 | Kolar  | Ana   | 31000 | Osijek |       |
| 34567 | Novak  | Josip | 10000 | Zagreb |       |

$$osoba_1 = \pi_{\text{matBr}, \text{prez}, \text{ime}, \text{postBr}} (\text{osoba})$$

$$mjesto = \pi_{\text{postBr}, \text{nazMj}} (\text{osoba})$$

- Hoće li se relacija osoba dekomponirati bez gubitaka informacija na relacije  $osoba_1$  i  $mjesto$ ? Odnosno, vrijedi li:

$$osoba \equiv osoba_1 \triangleright\triangleleft mjesto$$

# Primjer razlaganja relacije na dvije projekcije

osoba<sub>1</sub>

| matBr | prez   | ime   | postBr |
|-------|--------|-------|--------|
| 11234 | Novak  | Josip | 21000  |
| 12345 | Horvat | Ivan  | 10000  |
| 23456 | Kolar  | Ana   | 31000  |
| 34567 | Novak  | Josip | 10000  |

$$\text{OSOBA}_1 = \{ \text{matBr}, \text{prez}, \text{ime}, \text{postBr} \}$$

$$K_{\text{OSOBA}_1} = \{ \text{matBr} \}$$

mjesto

| postBr | nazMj  |
|--------|--------|
| 21000  | Split  |
| 10000  | Zagreb |
| 31000  | Osijek |

$$\text{MJESTO} = \{ \text{postBr}, \text{nazMj} \}$$

$$K_{\text{MJESTO}} = \{ \text{postBr} \}$$

$$\text{OSOBA}_1 \cap \text{MJESTO} = \{ \text{postBr} \}$$

$$\Rightarrow \text{osoba} \equiv \text{osoba}_1 \bowtie \text{mjesto}$$

osoba

| matBr | prez   | ime   | postBr | nazMj  |
|-------|--------|-------|--------|--------|
| 11234 | Novak  | Josip | 21000  | Split  |
| 12345 | Horvat | Ivan  | 10000  | Zagreb |
| 23456 | Kolar  | Ana   | 31000  | Osijek |
| 34567 | Novak  | Josip | 10000  | Zagreb |

# Prva normalna forma (1NF)

---

- Definicija:

Relacijska shema je u **1NF** ako:

- domene atributa sadrže samo jednostavne (nedjeljive) vrijednosti
- vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa
- neključni atributi relacije funkcijски ovise o ključu relacije
- Shema baze podataka  $R = \{ R_1, R_2, \dots, R_n \}$  je u 1NF ako je svaka relacijska shema  $R_1, R_2, \dots, R_n$  u 1NF

# Prva normalna forma - primjer

---

- Poduzeće evidentira podatke o radnicima
  - $\text{RADNIK} = \{ \text{matBr}, \text{prezime}, \text{ime}, \text{datRod}, \text{sifOdjel} \}$

| radnik (RADNIK) | matBr  | prezime | ime        | datRod | sifOdjel |
|-----------------|--------|---------|------------|--------|----------|
| 1111            | Novak  | Ivan    | 28.12.1970 | 50     |          |
| 1121            | Kolar  | Iva     | 16.10.1965 | 30     |          |
| 1133            | Horvat | Krešo   | 19.03.1978 | 50     |          |

$$K_{\text{RADNIK}} = \{ \text{matBr} \}$$

- domene svih atributa sadrže jednostavne (nedjeljive) vrijednosti
- vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa
- neklučni atributi relacije funkcijски ovise o ključu relacije

⇒ Relacijska shema RADNIK je u 1NF

# Prva normalna forma - primjer

---

- Poduzeće evidentira podatke o radnicima i njihovoј djeci - korisnicima zdravstvenog osiguranja.
  - $\text{RADNIK}_1 = \{ \text{matBr}, \text{prezime}, \text{ime}, \text{imenaDjece} \}$

$K_{\text{RADNIK}_1} = \{ \text{matBr} \}$

| radnik <sub>1</sub> (RADNIK <sub>1</sub> ) | matBr | prezime | ime   | imenaDjece       |
|--------------------------------------------|-------|---------|-------|------------------|
|                                            | 1111  | Novak   | Ivan  | Jasna, Vedran    |
|                                            | 1121  | Kolar   | Iva   | Ivan             |
|                                            | 1133  | Horvat  | Krešo | Petar, Ana, Ivan |

- domena atributa *imenaDjece* ne sadrži jednostavne (nedjeljive vrijednosti)

⇒ Relacijska shema RADNIK<sub>1</sub> nije u 1NF

# Prva normalna forma - primjer

- Poduzeće evidentira podatke o radnicima i njihovoј djeci - korisnicima zdravstvenog osiguranja.
  - $\text{RADNIK}_2 = \{ \text{matBr}, \text{prezime}, \text{ime}, \text{imeDj}, \text{datRodDj} \}$

| $\text{radnik}_2(\text{RADNIK}_2)$         | $\text{matBr}$ | $\text{prezime}$ | $\text{ime}$ | $\text{imeDj}$       | $\text{datRodDj}$                      |
|--------------------------------------------|----------------|------------------|--------------|----------------------|----------------------------------------|
| $K_{\text{RADNIK}_2} = \{ \text{matBr} \}$ | 1111           | Novak            | Ivan         | Jasna<br>Vedran      | 21.01.1995<br>13.12.1997               |
|                                            | 1121           | Kolar            | Iva          | Ivan                 | 23.03.2000                             |
|                                            | 1133           | Horvat           | Krešo        | Petar<br>Ana<br>Ivan | 22.02.1998<br>19.09.2000<br>05.11.2002 |

- domene sadrže jednostavne vrijednosti, ali vrijednost atributa  $\text{imeDj}$  nije uvijek samo jedna vrijednost iz domene tog atributa (isto vrijedi i za atribut  $\text{datRodDj}$ )

⇒ Relacijska shema  $\text{RADNIK}_2$  nije u 1NF

# Normalizacija na 1NF - izdvajanjem atributa u posebnu relaciju

- u posebnu relaciju izdvaja se skup atributa koji se ponavlja s jednakom kratnošću, zajedno s ključem originalne relacije

$$RADNIK_2 = \{ \text{matBr}, \text{prezime}, \text{ime}, \text{imeDj}, \text{datRodDj} \} \quad K_{RADNIK_2} = \{ \text{matBr} \}$$

| radnik <sub>3</sub> (RADNIK <sub>3</sub> ) | matBr | prezime | ime   |
|--------------------------------------------|-------|---------|-------|
|                                            | 1111  | Novak   | Ivan  |
|                                            | 1121  | Kolar   | Iva   |
|                                            | 1133  | Horvat  | Krešo |

$$RADNIK_3 = \{ \text{matBr}, \text{prezime}, \text{ime} \}$$

$$K_{RADNIK_3} = \{ \text{matBr} \}$$

| dijete (DIJETE) | matBr | imeDj  | datRodDj   |
|-----------------|-------|--------|------------|
|                 | 1111  | Jasna  | 21.01.1995 |
|                 | 1111  | Vedran | 13.12.1997 |
|                 | 1121  | Ivan.  | 23.03.2000 |
|                 | 1133  | Petar  | 22.02.1998 |
|                 | 1133  | Ana    | 19.09.2000 |
|                 | 1133  | Ivan   | 05.11.2002 |

$$DIJETE = \{ \text{matBr}, \text{imeDj}, \text{datRodDj} \}$$

$$K_{DIJETE} = \{ \text{matBr}, \text{imeDj} \}$$

- operacija je izvedena bez gubitaka informacija - relacijske sheme imaju zajedničke attribute (matBr), zajednički attribute su ključ u RADNIK<sub>3</sub>

## Normalizacija na 1NF - promjenom ključa

---

$RADNIK_2 = \{ matBr, prezime, ime, imeDj, datRodDj \}$      $K_{RADNIK_2} = \{ matBr \}$

$RADNIK_4 = \{ matBr, prezime, ime, imeDj, datRodDj \}$

$K_{RADNIK_4} = \{ matBr, imeDj \}$

| radnik <sub>4</sub> (RADNIK <sub>4</sub> ) | matBr  | prezime | ime    | imeDj      | datRodDj |
|--------------------------------------------|--------|---------|--------|------------|----------|
| 1111                                       | Novak  | Ivan    | Jasna  | 21.01.1995 |          |
| 1111                                       | Novak  | Ivan    | Vedran | 13.12.1997 |          |
| 1121                                       | Kolar  | Iva     | Ivan   | 23.03.2000 |          |
| 1133                                       | Horvat | Krešo   | Petar  | 22.02.1998 |          |
| 1133                                       | Horvat | Krešo   | Ana    | 19.09.2000 |          |
| 1133                                       | Horvat | Krešo   | Ivan   | 05.11.2002 |          |

## Druga normalna forma (2NF)

---

- Definicija:

Relacijska shema  $R$  je u **2NF** ako je u **1NF** i ako je

- svaki atribut iz zavisnog dijela potpuno funkcijски ovisan o svakom ključu relacije

- Shema baze podataka  $R = \{ R_1, R_2, \dots, R_n \}$  je u **2NF** ako je svaka relacijska shema  $R_1, R_2, \dots, R_n$  u **2NF**

# Potpuna funkcionalna zavisnost

---

- Skup atributa Y **potpuno je funkcionalno ovisan** o skupu atributa X relacijske sheme R ako:
  - Y funkcionalno ovisi o X
  - i
  - ne postoji pravi podskup od X koji funkcionalno određuje Y

PRIMJER:

Zadan je skup FZ  $F = \{ ABC \rightarrow DE, E \rightarrow F \}$ .

Je li  $\{ D, E \}$  potpuno funkcionalno ovisan o  $\{ A, B, C \}$  ?

Da, jer ne postoji skup  $Z \subset \{ A, B, C \}$  takav da  $Z \rightarrow \{ D, E \}$

# Nepotpuna funkcionalna zavisnost

---

Zadana je relacijska shema R i skupovi atributa X i Y iz R, tj.  $X \subseteq R$ ,  $Y \subseteq R$ . Neka u R vrijedi FZ  $X \rightarrow Y$ .

FZ  $X \rightarrow Y$  je **nepotpuna** ako postoji skup atributa  $Z$  koji je pravi podskup od  $X$ , za koji vrijedi  $Z \rightarrow Y$

odnosno

FZ  $X \rightarrow Y$  je **nepotpuna** ako  $(\exists Z) (Z \subset X) : Z \rightarrow Y$

PRIMJER:

Zadan je skup FZ  $F = \{ ABC \rightarrow D, BC \rightarrow E, E \rightarrow D \}$ .

Je li  $\{ D \}$  potpuno funkcionalski ovisan o  $\{ A, B, C \}$  ?

Ne, jer postoji skup  $\{ B, C \} \subset \{ A, B, C \}$  takav da  $\{ B, C \} \rightarrow \{ D \}$

## Druga normalna forma - primjer

$\text{RADNIK}_4 = \{ \text{matBr}, \text{prezime}, \text{ime}, \text{imeDj}, \text{datRodDj} \}$        $K_{\text{RADNIK}_4} = \{ \text{matBr}, \text{imeDj} \}$

| radnik <sub>4</sub> ( $\text{RADNIK}_4$ ) | matBr  | prezime | ime    | imeDj      | datRodDj |
|-------------------------------------------|--------|---------|--------|------------|----------|
| 1111                                      | Novak  | Ivan    | Jasna  | 21.01.1995 |          |
| 1111                                      | Novak  | Ivan    | Vedran | 13.12.1997 |          |
| 1121                                      | Kolar  | Iva     | Ivan.  | 23.03.2000 |          |
| 1133                                      | Horvat | Krešo   | Petar  | 22.02.1998 |          |
| 1133                                      | Horvat | Krešo   | Ana    | 19.09.2000 |          |
| 1133                                      | Horvat | Krešo   | Ivan   | 05.11.2002 |          |

- relacijska shema  $\text{RADNIK}_4$  zadovoljava 1NF
- **postoji FZ:**  $\text{matBr} \rightarrow \text{prezime } \text{ime}$
- $\text{matBr } \text{imeDj} \rightarrow \text{prezime } \text{ime}$  je nepotpuna FZ!

⇒ Relacijska shema  $\text{RADNIK}_4$  nije u 2NF

# Normalizacija na 2NF

- Normalizacijom na 2NF nastaju:
  - relacijska shema koja sadrži skup atributa koji su bili nepotpuno funkcijски ovisni o ključu i dio ključa o kojem su potpuno funkcijски ovisni
  - relacijska shema koja sadrži ključ originalne relacije i skup atributa koji su potpuno funkcijski ovisni o ključu

$RADNIK_5 = \{ \text{matBr}, \text{ prezime}, \text{ ime} \}$

$K_{RADNIK_5} = \{ \text{matBr} \}$

| radnik <sub>5</sub> (RADNIK <sub>5</sub> ) |         |       |
|--------------------------------------------|---------|-------|
| matBr                                      | prezime | ime   |
| 1111                                       | Novak   | Ivan  |
| 1121                                       | Kolar   | Iva   |
| 1133                                       | Horvat  | Krešo |

$DIJETE = \{ \text{matBr}, \text{ imeDj}, \text{ datRodDj} \}$

$K_{DIJETE} = \{ \text{matBr}, \text{ imeDj} \}$

| dijete (DIJETE) |        |            |
|-----------------|--------|------------|
| matBr           | imeDj  | datRodDj   |
| 1111            | Jasna  | 21.01.1995 |
| 1111            | Vedran | 13.12.1997 |
| 1121            | Ivan.  | 23.03.2000 |
| 1133            | Petar  | 22.02.1998 |
| 1133            | Ana    | 19.09.2000 |
| 1133            | Ivan   | 05.11.2002 |

# Normalizacija na 2NF

---

Neka su  $X, Y, Z, V$  atributi ili skupovi atributa. Zadana je relacijska shema  $R = XYZV$  i na njoj skup funkcijskih zavisnosti  $F = \{ XY \rightarrow ZV, X \rightarrow Z \}$ .

Ključ relacije  $K_R = XY$ . R je u 1NF. **Zadovoljava li R 2NF?**

- funkcijска zavisnost  $XY \rightarrow Z$  je nepotpuna  
**R ne zadovoljava 2NF**

Normalizacijom na 2NF shema R se zamjenjuje shemama:

$$R_1 = XZ$$

$$R_2 = XYV$$

$$K_{R_1} = X$$

$$K_{R_2} = XY$$

Relacija  $r(R)$  se normalizacijom na 2NF zamjenjuje projekcijama:

$$r_1 = \pi_{XZ}(r) \quad r_2 = \pi_{XYV}(r)$$

- operacija je izvedena bez gubitaka informacija - relacijske sheme imaju zajedničke attribute (X), zajednički atributi su ključ u  $R_1$ .

## Treća normalna forma (3NF)

---

- Definicija:

Relacijska shema je u 3NF ako je u 1NF i ako:

- niti jedan atribut iz zavisnog dijela nije tranzitivno funkcijski ovisan o bilo kojem ključu relacije
- Shema baze podataka  $R = \{ R_1, R_2, \dots, R_n \}$  je u 3NF ako je svaka relacijska shema  $R_1, R_2, \dots, R_n$  u 3NF

# Tranzitivna funkcionalna zavisnost

---

Zadano je:

- relacijska shema  $R$ ,
- skupovi atributa  $X \subseteq R$ ,  $Y \subseteq R$ ,  $Z \subseteq R$

Skup atributa  $Z$  je tranzitivno ovisan o  $X$  ako vrijedi:

- $X \rightarrow Y$ ,  $Y \not\rightarrow X$  i  $Y \rightarrow Z$

## Tranzitivna funkcionalna zavisnost - primjer

---

Zadana je relacijska shema  $R = \{ A, B, C, D, E, F, G \}$  i skup FZ  
 $F = \{ AB \rightarrow CD, D \rightarrow EF, CD \rightarrow ABG \}$

EF je tranzitivno funkcijski ovisan o AB, jer

- $AB \rightarrow D, D \not\rightarrow AB, D \rightarrow EF$

G nije tranzitivno funkcijski ovisan o AB, jer iako

- $AB \rightarrow CD, CD \rightarrow G$
- nije zadovoljen uvjet  $CD \not\rightarrow AB$

## Treća normalna forma - primjer

$\text{OSOBA} = \{ \text{matBr}, \text{prez}, \text{ime}, \text{postBr}, \text{nazMjesto} \}$     $K_{\text{OSOBA}} = \{ \text{matBr} \}$

| osoba (OSOBA) | matBr  | prez  | ime   | postBr | nazMjesto |
|---------------|--------|-------|-------|--------|-----------|
| 1111          | Novak  | Ivan  | 10000 | Zagreb |           |
| 1121          | Kolar  | Iva   | 31000 | Osijek |           |
| 1133          | Horvat | Krešo | 10000 | Zagreb |           |

- Relacijska shema OSOBA zadovoljava 1NF.
  - vrijedi FZ:  $\text{matBr} \rightarrow \text{postBr}$
  - vrijedi FZ:  $\text{postBr} \rightarrow \text{nazMjesto}$
  - ne vrijedi FZ:  $\text{postBr} \rightarrow \text{matBr}$
- $\text{matBr} \rightarrow \text{nazMjesto}$  je tranzitivna zavisnost !
- Relacijska shema OSOBA ne zadovoljava 3NF.

# Normalizacija na 3NF

$\text{OSOBA} = \{ \text{matBr}, \text{prez}, \text{ime}, \text{postBr}, \text{nazMjesto} \}$     $K_{\text{OSOBA}} = \{ \text{matBr} \}$

Normalizacijom na 3NF nastaju:

- relacijska shema koja sadrži skup atributa relacijske sheme OSOBA koji su tranzitivno ovisni o ključu (*nazMjesto*) te srednji skup atributa uočene tranzitivne zavisnosti (*postBr*)
- relacijska shema koja sadrži ključ relacijske sheme OSOBA (*matBr*) i neključne attribute relacijske sheme OSOBA koji nisu tranzitivno ovisni o ključu

$\text{MJESTO} = \{ \text{postBr}, \text{nazMjesto} \}$

$K_{\text{MJESTO}} = \{ \text{postBr} \}$

| mjesto (MJESTO) |           |
|-----------------|-----------|
| postBr          | nazMjesto |
| 10000           | Zagreb    |
| 31000           | Osijek    |

$\text{OSOBA}_1 = \{ \text{matBr}, \text{prezime}, \text{ime}, \text{postBr} \}$

$K_{\text{OSOBA}_1} = \{ \text{matBr} \}$

| osoba <sub>1</sub> (OSOBA <sub>1</sub> ) |         |       |        |
|------------------------------------------|---------|-------|--------|
| matBr                                    | prezime | ime   | postBr |
| 1111                                     | Novak   | Ivan  | 10000  |
| 1121                                     | Kolar   | Iva   | 31000  |
| 1133                                     | Horvat  | Krešo | 10000  |

# Normalizacija na 3NF

---

Neka su  $X, Y, Z, V$  atributi ili skupovi atributa. Zadana je relacijska shema  $R = XYZV$  i na njoj skup funkcijskih zavisnosti  $F = \{ X \rightarrow YZV, Z \rightarrow V \}$ .

Ključ relacije  $K_R = X$ . R je u 1NF. **Zadovoljava li R 3NF?**

- funkcijска зависност  $X \rightarrow V$  је транзитивна  
 $R$  не задоволjava 3NF

Normalizацијом на 3NF shema R se zamjenjuje shemama:

$$R_1 = XYZ$$

$$K_{R1} = X$$

$$R_2 = ZV$$

$$K_{R2} = Z$$

Relacija  $r(R)$  se normalizацијом на 3NF zamjenjuje пројекцијама:

$$r_1 = \pi_{XYZ}(r) \quad r_2 = \pi_{ZV}(r)$$

- операција је изведена без губитака информација - релацијске схеме имају заједничке атрибуте ( $Z$ ), заједнички атрибути су клjuč у  $R_2$ .

## Treća normalna forma - komentar

---

Normalizacija na 2NF nije nužni preuvjet za provođenje normalizacije na 3NF jer se nepotpune FZ mogu promatrati kao tranzitivne FZ.

**Primjer:** zadana je shema  $R = XYZV$  i na njoj skup funkcijskih zavisnosti  $F = \{ XY \rightarrow ZV, X \rightarrow Z \}$ . Ključ relacije  $K_R = XY$ . R je u 1NF, ali nije u 2NF jer postoji nepotpuna FZ  $XY \rightarrow Z$ . Međutim, postoji i tranzitivna funkcionska zavisnost  $XY \rightarrow Z$  ( $XY \rightarrow X \wedge X \rightarrow Z$ ).

Normalizacijom na 3NF shema R se zamjenjuje shemama:

$$R_1 = XZ$$

$$K_{R1} = X$$

$$R_2 = XYV$$

$$K_{R2} = XY$$

$R_1$  i  $R_2$  su u 2NF i 3NF

Preporuka: normalizaciju ipak obavljati postupno

$1NF \Rightarrow 2NF \Rightarrow 3NF$

# Normalizacija na 3NF - primjer

---

$\text{OSOBA}_2 = \{ \text{matBr}, \text{prez}, \text{ime}, \text{JMBG} \}$

| osoba2 (OSOBA2) | matBr | prez   | ime   | JMBG          |
|-----------------|-------|--------|-------|---------------|
|                 | 1111  | Novak  | Ivan  | 1403970330103 |
|                 | 1121  | Kolar  | Iva   | 2812968310267 |
|                 | 1133  | Horvat | Krešo | 0301979320319 |

- postoji FZ:  $\text{matBr} \rightarrow \text{prez} \text{ } \text{ime} \text{ } \text{JMBG}$
- postoje FZ:  $\text{JMBG} \rightarrow \text{prez} \text{ } \text{ime}$  i  $\text{JMBG} \rightarrow \text{matBr}$
  
- $\text{matBr}$  i  $\text{JMBG}$  su mogući ključevi
- Relacijska shema  $\text{OSOBA}_2$  zadovoljava 3NF.

$$K_1_{\text{OSOBA2}} = \{ \text{matBr} \}$$

$$K_2_{\text{OSOBA2}} = \{ \text{JMBG} \}$$

## Normalizacija na 3NF - dodatna razmatranja

---

Neka su  $X, Y, Z, V$  atributi ili skupovi atributa. Zadana je relacijska shema  $R = XYZV$  i na njoj skup funkcijskih zavisnosti

$F = \{ X \rightarrow YZV, Z \rightarrow V, Z \rightarrow X \}$ . R je u 1NF. Neka je ključ  $K_R = X$ .

- vrijedi  $X \rightarrow Z$  i  $Z \rightarrow V$ , ali  $X \rightarrow V$  nije tranzitivna FZ jer vrijedi i  $Z \rightarrow X$
- Zbog  $X \rightarrow Z$  i  $Z \rightarrow X$  funkcijsku zavisnost  $X \rightarrow V$  nije potrebno ukloniti jer u tom slučaju nema redundancije.
- $Z$  je također mogući ključ u R

$$K_{R1} = X$$

$$K_{R2} = Z$$

$X$  i  $Z$  su mogući ključevi.

- Relacijska shema R zadovoljava 3NF.

# 1. primjer normalizacije

---

- Zadana je relacijska shema:

ISPIT = { matBr, prez, ime, sifPred, nazPred, datlsp, ocj, sifNas, prezNas }

i trenutna vrijednost relacije **ispit**(ISPIT):

| ispit (ISPIT) |       |       |         |         |          |     |        |         |
|---------------|-------|-------|---------|---------|----------|-----|--------|---------|
| matBr         | prez  | ime   | sifPred | nazPred | datlsp   | ocj | sifNas | prezNas |
| 1111          | Novak | Ivan  | 1001    | Mat-1   | 29.01.06 | 1   | 1111   | Pašić   |
| 1111          | Novak | Ivan  | 1001    | Mat-1   | 05.02.06 | 3   | 1111   | Pašić   |
| 1111          | Novak | Ivan  | 1003    | Fiz-1   | 28.06.06 | 2   | 3333   | Horvat  |
| 1111          | Novak | Ivan  | 1002    | Mat-2   | 27.06.06 | 4   | 2222   | Brnetić |
| 1234          | Kolar | Petar | 1001    | Mat-1   | 29.01.06 | 3   | 2222   | Brnetić |

- funkcijske zavisnosti odrediti na temelju značenja podataka
- odrediti primarni ključ relacije (tako da bude zadovoljen uvjet 1NF prema kojem neključni atributi funkcijски ovise o ključu)
- postupno normalizirati relacijsku shemu ISPIT na 2NF i 3NF

# 1. primjer normalizacije - 1NF

ispit (ISPIT)

| matBr | prez  | ime   | sifPred | nazPred | datlsp   | ocj | sifNas | prezNas |
|-------|-------|-------|---------|---------|----------|-----|--------|---------|
| 1111  | Novak | Ivan  | 1001    | Mat-1   | 29.01.06 | 1   | 1111   | Pašić   |
| 1111  | Novak | Ivan  | 1001    | Mat-1   | 05.02.06 | 3   | 1111   | Pašić   |
| 1111  | Novak | Ivan  | 1003    | Fiz-1   | 28.06.06 | 2   | 3333   | Horvat  |
| 1111  | Novak | Ivan  | 1002    | Mat-2   | 27.06.06 | 4   | 2222   | Brnetić |
| 1234  | Kolar | Petar | 1001    | Mat-1   | 29.01.06 | 3   | 2222   | Brnetić |

- Određivanje ključa: ako se (pogrešno) prepostavi da je  $K = \{ \text{matBr} \}$   
Bi li tada postojali neključni atributi koje ključ funkcijski ne određuje?
- $\text{matBr} \rightarrow \text{prez } \text{ime}$   
**međutim:**
- $\text{matBr} \not\rightarrow \text{sifPred}$        $\text{matBr} \not\rightarrow \text{nazPred}$   
 $\text{matBr} \not\rightarrow \text{datlsp}$        $\text{matBr} \not\rightarrow \text{ocj}$   
 $\text{matBr} \not\rightarrow \text{sifNas}$        $\text{matBr} \not\rightarrow \text{prezNas}$

# 1. primjer normalizacije - 1NF

ispit (ISPIT)

| matBr | prez  | ime   | sifPred | nazPred | datlsp   | ocj | sifNas | prezNas |
|-------|-------|-------|---------|---------|----------|-----|--------|---------|
| 1111  | Novak | Ivan  | 1001    | Mat-1   | 29.01.06 | 1   | 1111   | Pašić   |
| 1111  | Novak | Ivan  | 1001    | Mat-1   | 05.02.06 | 3   | 1111   | Pašić   |
| 1111  | Novak | Ivan  | 1003    | Fiz-1   | 28.06.06 | 2   | 3333   | Horvat  |
| 1111  | Novak | Ivan  | 1002    | Mat-2   | 27.06.06 | 4   | 2222   | Brnetić |
| 1234  | Kolar | Petar | 1001    | Mat-1   | 29.01.06 | 3   | 2222   | Brnetić |

- Ako se pretpostavi  $K = \{ \text{matBr}, \text{sifPred}, \text{datlsp} \}$   
Bi li tada postojali **neklučni atributi koje ključ funkcijski ne određuje?**
- $\text{matBr } \text{sifPred } \text{datlsp} \rightarrow \text{prez } \text{ime } \text{nazPred } \text{ocj } \text{sifNas } \text{prezNas}$   
postoji li skup  $X \subset \{ \text{matBr}, \text{sifPred}, \text{datlsp} \}$  za kojeg vrijedi  $X \rightarrow R$  ?  
 $\Rightarrow \text{NE} \Rightarrow \{ \text{matBr}, \text{sifPred}, \text{datlsp} \}$  je mogući ključ
- $K_{\text{ISPIT}} = \{ \text{matBr}, \text{sifPred}, \text{datlsp} \}$
- **zadovoljen je uvjet 1NF prema kojem neključni atributi funkcijski ovise o ključu**

# 1. primjer normalizacije - 2NF

ispit (ISPIT)

| matBr | prez  | ime   | sifPred | nazPred | datlsp   | ocj | sifNas | prezNas |
|-------|-------|-------|---------|---------|----------|-----|--------|---------|
| 1111  | Novak | Ivan  | 1001    | Mat-1   | 29.01.06 | 1   | 1111   | Pašić   |
| 1111  | Novak | Ivan  | 1001    | Mat-1   | 05.02.06 | 3   | 1111   | Pašić   |
| 1111  | Novak | Ivan  | 1003    | Fiz-1   | 28.06.06 | 2   | 3333   | Horvat  |
| 1111  | Novak | Ivan  | 1002    | Mat-2   | 27.06.06 | 4   | 2222   | Brnetić |
| 1234  | Kolar | Petar | 1001    | Mat-1   | 29.01.06 | 3   | 2222   | Brnetić |

- Postoje li neključni atributi koji ovise o dijelu ključa?
  - $\text{matBr} \rightarrow \text{prez, ime}$

$\text{student} = \pi_{\text{matBr, prez, ime}}(\text{ispit})$

$\text{ispit}_1 = \pi_{\text{matBr, sifPred, nazPred, datlsp, ocj, sifNas, prezNas}}(\text{ispit})$

$K_{\text{STUDENT}} = \{ \text{matBr} \}$

| student (STUDENT) |       |       |
|-------------------|-------|-------|
| matBr             | prez  | ime   |
| 1111              | Novak | Ivan  |
| 1234              | Kolar | Petar |

2NF, 3NF: O.K.

ispit<sub>1</sub> (ISPIT<sub>1</sub>)

$K_{\text{ISPIT}_1} = \{ \text{matBr, sifPred, datlsp} \}$

| matBr | sifPred | nazPred | datlsp   | ocj | sifNas | prezNas |
|-------|---------|---------|----------|-----|--------|---------|
| 1111  | 1001    | Mat-1   | 29.01.06 | 1   | 1111   | Pašić   |
| 1111  | 1001    | Mat-1   | 05.02.06 | 3   | 1111   | Pašić   |
| 1111  | 1003    | Fiz-1   | 28.06.06 | 2   | 3333   | Horvat  |
| 1111  | 1002    | Mat-2   | 27.06.06 | 4   | 2222   | Brnetić |
| 1234  | 1001    | Mat-1   | 29.01.06 | 3   | 2222   | Brnetić |

# 1. primjer normalizacije - 2NF (nastavak)

ispit<sub>1</sub> (ISPIT<sub>1</sub>)

- Postoje li neključni atributi koji ovise o dijelu ključa?

| matBr | sifPred | nazPred | datlsp   | ocj | sifNas | prezNas |
|-------|---------|---------|----------|-----|--------|---------|
| 1111  | 1001    | Mat-1   | 29.01.06 | 1   | 1111   | Pašić   |
| 1111  | 1001    | Mat-1   | 05.02.06 | 3   | 1111   | Pašić   |
| 1111  | 1003    | Fiz-1   | 28.06.06 | 2   | 3333   | Horvat  |
| 1111  | 1002    | Mat-2   | 27.06.06 | 4   | 2222   | Brnetić |
| 1234  | 1001    | Mat-1   | 29.01.06 | 3   | 2222   | Brnetić |

- sifPred → nazPred

$$\text{predmet} = \pi_{\text{sifPred}, \text{nazPred}}(\text{ispit}_1)$$

$$\text{ispit}_2 = \pi_{\text{matBr}, \text{sifPred}, \text{datlsp}, \text{ocj}, \text{sifNas}, \text{prezNas}}(\text{ispit}_1)$$

K<sub>PREDMET</sub> = { sifPred }

| predmet (PREDMET) |         |
|-------------------|---------|
| sifPred           | nazPred |
| 1001              | Mat-1   |
| 1003              | Fiz-1   |
| 1002              | Mat-2   |

2NF, 3NF: O.K.

ispit<sub>2</sub> (ISPIT<sub>2</sub>)

K<sub>ISPIT<sub>2</sub></sub> = { matBr, sifPred, datlsp }

| matBr | sifPred | datlsp   | ocj | sifNas | prezNas |
|-------|---------|----------|-----|--------|---------|
| 1111  | 1001    | 29.01.06 | 1   | 1111   | Pašić   |
| 1111  | 1001    | 05.02.06 | 3   | 1111   | Pašić   |
| 1111  | 1003    | 28.06.06 | 2   | 3333   | Horvat  |
| 1111  | 1002    | 27.06.06 | 4   | 2222   | Brnetić |
| 1234  | 1001    | 29.01.06 | 3   | 2222   | Brnetić |

2NF: O.K.

# 1. primjer normalizacije - 3NF

ispit<sub>2</sub> (ISPIT<sub>2</sub>)

| matBr | sifPred | datlsp   | ocj | sifNas | prezNas |
|-------|---------|----------|-----|--------|---------|
| 1111  | 1001    | 29.01.06 | 1   | 1111   | Pašić   |
| 1111  | 1001    | 05.02.06 | 3   | 1111   | Pašić   |
| 1111  | 1003    | 28.06.06 | 2   | 3333   | Horvat  |
| 1111  | 1002    | 27.06.06 | 4   | 2222   | Brnetić |
| 1234  | 1001    | 29.01.06 | 3   | 2222   | Brnetić |

- Postoje li neključni atributi koji tranzitivno ovise o ključu?
- matBr sifPred datlsp → sifNas      sifNas → prezNas

nastavnik =  $\pi_{\text{sifNas}, \text{prezNas}}(\text{ispit}_2)$

ispit<sub>3</sub> =  $\pi_{\text{matBr}, \text{sifPred}, \text{datlsp}, \text{ocj}, \text{sifNas}}(\text{ispit}_2)$

K<sub>NASTAVNIK</sub> = { sifNas }

| nastavnik (NASTAVNIK) |         |
|-----------------------|---------|
| sifNas                | prezNas |
| 1111                  | Pašić   |
| 3333                  | Horvat  |
| 2222                  | Brnetić |

3NF: O.K.

ispit<sub>3</sub> (ISPIT<sub>3</sub>)

K<sub>ISPIT<sub>3</sub></sub> = { matBr, sifPred, datlsp }

| matBr | sifPred | datlsp   | ocj | sifNas |
|-------|---------|----------|-----|--------|
| 1111  | 1001    | 29.01.06 | 1   | 1111   |
| 1111  | 1001    | 05.02.06 | 3   | 1111   |
| 1111  | 1003    | 28.06.06 | 2   | 3333   |
| 1111  | 1002    | 27.06.06 | 4   | 2222   |
| 1234  | 1001    | 29.01.06 | 3   | 2222   |

3NF: O.K.

# 1. primjer normalizacije - 3NF

| student (STUDENT) |       |       |
|-------------------|-------|-------|
| matBr             | prez  | ime   |
| 1111              | Novak | Ivan  |
| 1234              | Kolar | Petar |

$$K_{\text{STUDENT}} = \{ \text{matBr} \}$$

| predmet (PREDMET) |         |
|-------------------|---------|
| sifPred           | nazPred |
| 1001              | Mat-1   |
| 1003              | Fiz-1   |
| 1002              | Mat-2   |

$$K_{\text{PREDMET}} = \{ \text{sifPred} \}$$

| nastavnik (NASTAVNIK) |         |
|-----------------------|---------|
| sifNas                | prezNas |
| 1111                  | Pašić   |
| 3333                  | Horvat  |
| 2222                  | Brnetić |

$$K_{\text{NASTAVNIK}} = \{ \text{sifNas} \}$$

| ispit <sub>3</sub> (ISPIT <sub>3</sub> ) |         |          |     |        |
|------------------------------------------|---------|----------|-----|--------|
| matBr                                    | sifPred | datlsp   | ocj | sifNas |
| 1111                                     | 1001    | 29.01.06 | 1   | 1111   |
| 1111                                     | 1001    | 05.02.06 | 3   | 1111   |
| 1111                                     | 1003    | 28.06.06 | 2   | 3333   |
| 1111                                     | 1002    | 27.06.06 | 4   | 2222   |
| 1234                                     | 1001    | 29.01.06 | 3   | 2222   |

$$K_{\text{ISPIT}_3} = \{ \text{matBr}, \text{sifPred}, \text{datlsp} \}$$

- Shema baze podataka STUSLU:  
$$\text{STUSLU} = \{ \text{STUDENT}, \text{PREDMET}, \text{NASTAVNIK}, \text{ISPIT}_3 \}$$
- Shema baze podataka STUSLU zadovoljava 3NF

## 2. primjer normalizacije

---

Zadana je relacijska shema  $R = ABCDEFGH$  i na njoj skup funkcijskih zavisnosti

$$F = \{ ABC \rightarrow DEFGH, A \rightarrow D, BC \rightarrow FGH, FG \rightarrow H \}.$$

Domene atributa sadrže samo jednostavne vrijednosti, vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa.

Odrediti primarni ključ relacije (tako da bude zadovoljen uvjet 1NF prema kojem neključni atributi funkcijski ovise o ključu), te shemu postupno normalizirati na 2NF i 3NF.

## 2. primjer normalizacije

---

$R = ABCDEFGH$

$F = \{ ABC \rightarrow DEFGH, A \rightarrow D, BC \rightarrow FGH, FG \rightarrow H \}$

- **Odrediti primarni ključ relacije**

Vrijedi li  $ABC \rightarrow DEFGH$  ?

**DA**

postoji li skup  $X \subset ABC$  za kojeg vrijedi  $X \rightarrow R$  ?

**NE**

⇒  $ABC$  je mogući ključ i može se odabratи kao primarni ključ sheme  $R$ .

$R = ABCDEFGH$

$K_R = ABC$

$R$  je u 1NF

## 2. primjer normalizacije - 2NF

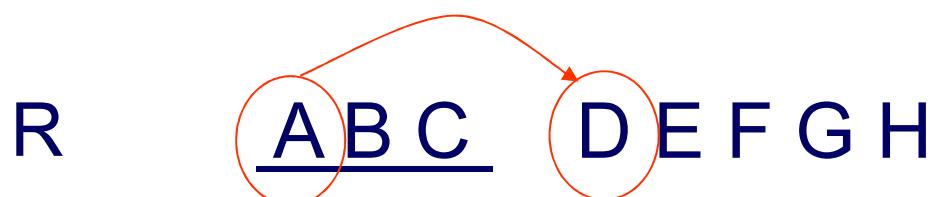
---

$$R = ABCDEFGH \quad K_R = ABC$$

$$F = \{ ABC \rightarrow DEFGH, A \rightarrow D, BC \rightarrow FGH, FG \rightarrow H \}$$

- **Normalizacija na 2NF**

Svi atributi iz zavisnog dijela moraju biti **potpuno** funkcijski ovisni o ključu.



- $ABC \rightarrow D$  je nepotpuna FZ, jer vrijedi  $A \rightarrow D$        $R$  nije u 2NF

Normalizacijom na 2NF se  $R$  zamjenjuje shemama:

$$R_1 = AD$$

$$K_{R1} = A$$

$R_1$  je u 2NF

$$R_2 = ABCEFGH$$

$$K_{R2} = ABC$$

$R_2$  nije u 2NF

## 2. primjer normalizacije - 2NF (nastavak)

---

$$R_1 = AD$$

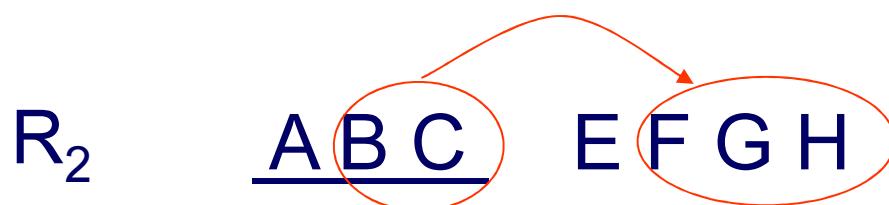
$$K_{R_1} = A$$

$$R_2 = ABC E F G H$$

$$K_{R_2} = ABC$$

$$F = \{ ABC \rightarrow DEFGH, A \rightarrow D, BC \rightarrow FGH, FG \rightarrow H \}$$

Svi atributi iz zavisnog dijela moraju biti **potpuno** funkcijski ovisni o ključu.



- $ABC \rightarrow FGH$  je nepotpuna FZ, jer vrijedi  $BC \rightarrow FGH$      $R_2$  nije u 2NF

Normalizacijom na 2NF se  $R_2$  zamjenjuje shemama:

$$R_{21} = BCFGH$$

$$K_{R_{21}} = BC$$

$R_{21}$  je u 2NF

$$R_{22} = ABCE$$

$$K_{R_{22}} = ABC$$

$R_{22}$  je u 2NF

## 2. primjer normalizacije - 3NF

$$R_1 = AD$$

$$K_{R_1} = A$$

$R_1$  je u 3NF

$$R_{21} = BCFGH$$

$$K_{R_{21}} = BC$$

$R_{21}$  nije u 3NF

$$R_{22} = ABCE$$

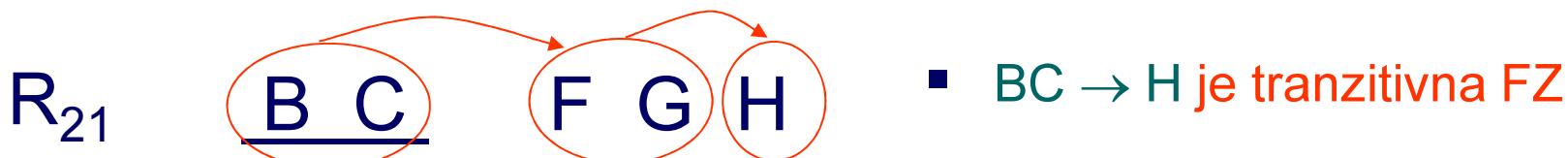
$$K_{R_{22}} = ABC$$

$R_{22}$  je u 3NF

$$F = \{ ABC \rightarrow DEFGH, A \rightarrow D, BC \rightarrow FGH, FG \rightarrow H \}$$

- **Normalizacija na 3NF**

Niti jedan atribut iz zavisnog dijela ne smije biti tranzitivno ovisan o ključu.



Normalizacijom na 3NF se  $R_{21}$  zamjenjuje shemama:

$$R_{211} = BCFG$$

$$K_{R_{211}} = BC$$

$R_{211}$  je u 3NF

$$R_{212} = FGH$$

$$K_{R_{212}} = FG$$

$R_{212}$  je u 3NF

**Shema baze podataka u 3NF sastoji se od relacijskih shema:**

$R_1, R_{22}, R_{211}$  i  $R_{212}$

# Boyce-Coddova normalna forma - BCNF

---

- Definicija:

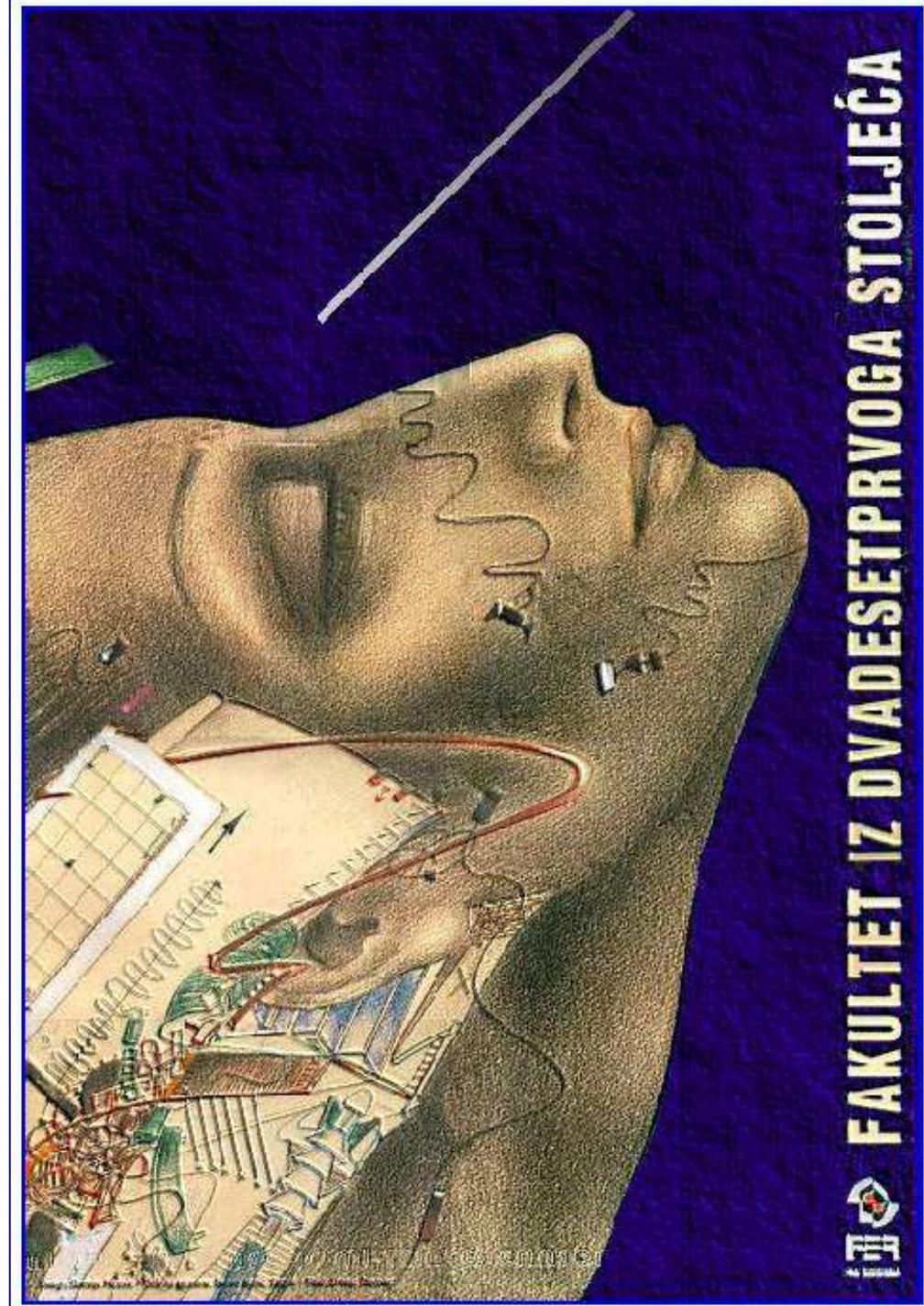
Relacijska shema R je u BCNF ako je u 1NF i ako niti jedan atribut nije tranzitivno funkcijски ovisan o bilo kojem ključu relacije

- Iako je BCNF stroža od 3NF, rijetki su slučajevi da je relacijska shema u 3NF, a da istovremeno nije i u BCNF.
- Normalizaciju na BCNF nije nužno provoditi.
- **Smatra se da shema baze podataka ima dobra svojstva ako zadovoljava 3NF.**

# Baze podataka

Predavanja  
travanj 2014.

## 8. Oblikovanje sheme relacijske baze podataka (3. dio - primjeri)



FAKULTET IZ DVADESETPRVOGA STOLJEĆA

# Zadatak 1

- Prodavaonice šalju svoje narudžbe proizvođaču:

|                                                                                                                                                                                                 |                                                                                                                                                                                              |                                                                                                                                          |                                     |                                       |                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|---------------------------------------|-------------------------------------|
| Konzum-7<br>Ilica 20<br>10 000 Zagreb                                                                                                                                                           | Kraš<br>Ravnice bb<br>10 000 Zagreb                                                                                                                                                          | Diona-28<br>Bolska 7<br>21 000 Split                                                                                                     | Kraš<br>Ravnice bb<br>10 000 Zagreb | Konzum-7<br>Ilica 20<br>10 000 Zagreb | Kraš<br>Ravnice bb<br>10 000 Zagreb |
| <b>Narudžba</b><br>br. 13/25<br><b>datum: 1.5.2006</b><br><br>Molimo isporučite nam 1200 komada proizvoda <b>Napolitanke</b> (šifra 129) i 2000 komada proizvoda <b>Albert keks</b> (šifra 139) | <b>Narudžba</b><br>br. 43-21<br><b>datum: 7.2.2006</b><br><br>Molimo isporučite nam 1200 komada proizvoda <b>Napolitanke</b> (šifra 129) i 1800 komada proizvoda <b>Domaćica</b> (šifra 221) | <b>Narudžba</b><br>br. 41/56<br><b>datum: 4.2.2007</b><br><br>Molimo isporučite nam 1100 komada proizvoda <b>Napolitanke</b> (šifra 129) |                                     |                                       |                                     |

- proizvođač želi pohraniti podatke o narudžbama u svoju bazu podataka.  
Svi podaci se pohranjuju u relaciju **narudzbaArtikla**

**narudzbaArtikla**

| nazProd | pbr | nazMjesto | adresa | brNar | datNar | sifArtikl | nazArtikl | kolicina |
|---------|-----|-----------|--------|-------|--------|-----------|-----------|----------|
|         |     |           |        |       |        |           |           |          |

# Zadatak 1

---

- Sadržaj relacije nakon unosa podataka iz prispjelih narudžbi:

narudzbaArtikla

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2006 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2006 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2006 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2006 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2007 | 129       | Napolitanke | 1100     |

- Normalizirajte relaciju narudzbaArtikla na 1NF, 2NF, 3NF ako vrijedi da je svaki broj narudžbe jedinstven (ne može se desiti da brojevi narudžbi prispjelih iz različitih prodavaonica budu jednaki)

$\text{brNar} \rightarrow \text{nazProd}$

# Zadatak 1

---

**NARUDZBAARTIKLA = { nazProd, pbr, nazMjesto, adresa, brNar, datNar, sifArtikl, nazArtikl, kolicina}**

- trenutna vrijednost relacije **narudzbaArtikla (NARUDZBAARTIKLA)** :

narudzbaArtikla

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2006 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2006 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2006 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2006 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2007 | 129       | Napolitanke | 1100     |

- odrediti funkcione zavisnosti na temelju značenja podataka
- odrediti primarni ključ relacije (tako da bude zadovoljen uvjet **1NF** prema kojem neključni atributi funkcionalno ovise o ključu)
- postupno normalizirati relacijsku shemu NARUDZBAARTIKLA na **2NF** i **3NF**

# Zadatak 1 - 1NF

narudzbaArtikla

| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   | sifArtikl | nazArtikl   | kolicina |
|----------|-------|-----------|----------|-------|----------|-----------|-------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2006 | 129       | Napolitanke | 1200     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2006 | 139       | Albert keks | 2000     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2006 | 129       | Napolitanke | 1200     |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2006 | 221       | Domaćica    | 1800     |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2007 | 129       | Napolitanke | 1100     |

- Određivanje ključa: bi li brNar bio dobar odabir za ključ?
- Postoje li neključni atributi koji ne ovise o broju narudžbe (brNar)?
- $\text{brNar} \rightarrow \text{nazProd}$   $\text{pbr}$   $\text{nazMjesto}$   $\text{adresa}$   $\text{datNar}$   
međutim:
  - $\text{brNar} \not\rightarrow \text{sifArtikl}$        $\text{brNar} \not\rightarrow \text{nazArtikl}$        $\text{brNar} \not\rightarrow \text{kolicina}$
  - O kojim atributima funkcijски ovisi atribut  $\text{nazArtikl}$ ?       $\text{sifArtikl} \rightarrow \text{nazArtikl}$
  - O kojim atributima funkcijски ovisi atribut  $\text{kolicina}$ ?       $\text{brNar}$   $\text{sifArtikl} \rightarrow \text{kolicina}$   
 $\text{sifArtikl} \not\rightarrow \text{kolicina}$
- **KLJUČ?**

# Zadatak 1 - 1NF

| narudzbaArtikla |       |           |          |              |          |                  |             |          |
|-----------------|-------|-----------|----------|--------------|----------|------------------|-------------|----------|
| nazProd         | pbr   | nazMjesto | adresa   | <u>brNar</u> | datNar   | <u>sifArtikl</u> | nazArtikl   | kolicina |
| Konzum-7        | 10000 | Zagreb    | Ilica 20 | 13/25        | 1.5.2006 | 129              | Napolitanke | 1200     |
| Konzum-7        | 10000 | Zagreb    | Ilica 20 | 13/25        | 1.5.2006 | 139              | Albert keks | 2000     |
| Diona-28        | 21000 | Split     | Bolska 7 | 43-21        | 7.2.2006 | 129              | Napolitanke | 1200     |
| Diona-28        | 21000 | Split     | Bolska 7 | 43-21        | 7.2.2006 | 221              | Domaćica    | 1800     |
| Konzum-7        | 10000 | Zagreb    | Ilica 20 | 41/56        | 4.2.2007 | 129              | Napolitanke | 1100     |

- Prepostavimo  $K = \{ \text{brNar}, \text{sifArtikl} \}$   
Provjerite postoje li neključni atributi koje ključ funkcijски ne određuje.
- $\text{brNar } \text{sifArtikl} \rightarrow \text{nazProd } \text{pbr } \text{nazMjesto } \text{adresa } \text{datNar } \text{nazArtikl } \text{kolicina}$   
postoji li skup  $X \subset \{ \text{brNar}, \text{sifArtikl} \}$  za kojeg vrijedi  $X \rightarrow R$  ?  
 $\Rightarrow \text{NE} \Rightarrow \{ \text{brNar}, \text{sifArtikl} \}$  je mogući ključ

$$K_{\text{NARUDZBAARTIKLA}} = \{ \text{brNar}, \text{sifArtikl} \}$$

- zadovoljen je uvjet 1NF prema kojem neključni atributi funkcijski ovise o ključu**

# Zadatak 1 - 2NF

narudzbaArtikla

| <u>nazProd</u> | <u>pbr</u> | <u>nazMjesto</u> | <u>adresa</u> | <u>brNar</u> | <u>datNar</u> | <u>sifArtikl</u> | <u>nazArtikl</u> | <u>kolicina</u> |
|----------------|------------|------------------|---------------|--------------|---------------|------------------|------------------|-----------------|
| Konzum-7       | 10000      | Zagreb           | Ilica 20      | 13/25        | 1.5.2006      | 129              | Napolitanke      | 1200            |
| Konzum-7       | 10000      | Zagreb           | Ilica 20      | 13/25        | 1.5.2006      | 139              | Albert keks      | 2000            |
| Diona-28       | 21000      | Split            | Bolska 7      | 43-21        | 7.2.2006      | 129              | Napolitanke      | 1200            |
| Diona-28       | 21000      | Split            | Bolska 7      | 43-21        | 7.2.2006      | 221              | Domaćica         | 1800            |
| Konzum-7       | 10000      | Zagreb           | Ilica 20      | 41/56        | 4.2.2007      | 129              | Napolitanke      | 1100            |

- Postoje li neključni atributi koji ovise o dijelu ključa?
  - vrijedi:  $\text{brNar} \rightarrow \text{nazProd pbr nazMjesto adresa datNar}$   
 $\Rightarrow$  Na koje relacije treba razložiti relaciju narudzbaArtikla?  
 Koji su ključevi novonastalih relacija?

$\text{narudzba} = \pi_{\text{nazProd, pbr, nazMjesto, adresa, brNar, datNar}} (\text{narudzbaArtikla})$   
 $K_{\text{NARUDZBA}} = \{ \text{brNar} \}$

$\text{stavkaNarudzbe} = \pi_{\text{brNar, sifArtikl, nazArtikl, kolicina}} (\text{narudzbaArtikla})$   
 $K_{\text{STAVKANARUDZBE}} = \{ \text{brNar, sifArtikl} \}$

# Zadatak 1 - 2NF

brNar → nazProd pbr nazMjesto adresa datNar

narudzba

| nazProd  | pbr   | nazMjesto | adresa   | <u>brNar</u> | datNar   |
|----------|-------|-----------|----------|--------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25        | 1.5.2006 |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21        | 7.2.2006 |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56        | 4.2.2007 |

- narudzba ima jednostavan ključ  
⇒ **2NF OK**

stavkaNarudzbe

| <u>brNar</u> | <u>sifArtikl</u> | nazArtikl   | kolicina |
|--------------|------------------|-------------|----------|
| 13/25        | 129              | Napolitanke | 1200     |
| 13/25        | 139              | Albert keks | 2000     |
| 43-21        | 129              | Napolitanke | 1200     |
| 43-21        | 221              | Domaćica    | 1800     |
| 41/56        | 129              | Napolitanke | 1100     |

Jesu li relacije narudzba i stavkaNarudzbe u 2NF?

# Zadatak 1 - 2NF

- Je li stavkaNarudzbe u 2NF?

(postoje li neključni atributi koji ovise o dijelu ključa?)

- Vrijedi:  $sifArtikl \rightarrow nazArtikl$
- ⇒ Na koje relacije treba razložiti relaciju stavkaNarudzbe?
- Koji su ključevi novonastalih relacija?

stavkaNarudzbe

| <u>brNar</u> | <u>sifArtikl</u> | <u>nazArtikl</u> | kolicina |
|--------------|------------------|------------------|----------|
| 13/25        | 129              | Napolitanke      | 1200     |
| 13/25        | 139              | Albert keks      | 2000     |
| 43-21        | 129              | Napolitanke      | 1200     |
| 43-21        | 221              | Domaćica         | 1800     |
| 41/56        | 129              | Napolitanke      | 1100     |

$$artikl = \pi_{sifArtikl, nazArtikl}(stavkaNarudzbe)$$

$$K_{ARTIKL} = \{ sifArtikl \}$$

$$stavkaNarudzbe_1 = \pi_{brNar, sifArtikl, kolicina}(stavkaNarudzbe)$$

$$K_{STAVKANARUDZBE1} = \{ brNar, sifArtikl \}$$

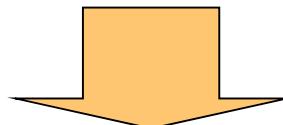
# Zadatak 1 - 2NF

stavkaNarudzbe

| <u>brNar</u> | <u>sifArtikl</u> | <u>nazArtikl</u> | <u>kolicina</u> |
|--------------|------------------|------------------|-----------------|
| 13/25        | 129              | Napolitanke      | 1200            |
| 13/25        | 139              | Albert keks      | 2000            |
| 43-21        | 129              | Napolitanke      | 1200            |
| 43-21        | 221              | Domaćica         | 1800            |
| 41/56        | 129              | Napolitanke      | 1100            |

$sifArtikl \rightarrow nazArtikl$

Jesu li relacije artikl i  
stavkaNarudzbe<sub>1</sub> u 2NF?



| artikl           |                  |
|------------------|------------------|
| <u>sifArtikl</u> | <u>nazArtikl</u> |
| 129              | Napolitanke      |
| 139              | Albert keks      |
| 221              | Domaćica         |

2NF O.K.

stavkaNarudzbe<sub>1</sub>

| <u>brNar</u> | <u>sifArtikl</u> | <u>kolicina</u> |
|--------------|------------------|-----------------|
| 13/25        | 129              | 1200            |
| 13/25        | 139              | 2000            |
| 43-21        | 129              | 1200            |
| 43-21        | 221              | 1800            |
| 41/56        | 129              | 1100            |

2NF O.K.

# Zadatak 1 - 3NF

- Postoje li neključni atributi koji tranzitivno ovise o ključu?

narudzba

| nazProd  | pbr   | nazMjesto | adresa   | <u>brNar</u> | datNar   |
|----------|-------|-----------|----------|--------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25        | 1.5.2006 |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21        | 7.2.2006 |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56        | 4.2.2007 |

Je li  
relacija  
narudzba  
u 3NF?

stavkaNarudzbe<sub>1</sub>

| <u>brNar</u> | <u>sifArtikl</u> | kolicina |
|--------------|------------------|----------|
| 13/25        | 129              | 1200     |
| 13/25        | 139              | 2000     |
| 43-21        | 129              | 1200     |
| 43-21        | 221              | 1800     |
| 41/56        | 129              | 1100     |

3NF O.K.

artikl

| <u>sifArtikl</u> | nazArtikl   |
|------------------|-------------|
| 129              | Napolitanke |
| 139              | Albert keks |
| 221              | Domaćica    |

3NF O.K.

# Zadatak 1 - 3NF

- Postoje li u relaciji narudzba neključni atributi koji tranzitivno ovise o ključu?

| narudzba |       |           |          |       |          |
|----------|-------|-----------|----------|-------|----------|
| nazProd  | pbr   | nazMjesto | adresa   | brNar | datNar   |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25 | 1.5.2006 |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21 | 7.2.2006 |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56 | 4.2.2007 |

- Vrijedi:  
 $\text{brNar} \rightarrow \text{nazProd}$        $\text{nazProd} \rightarrow \text{pbr nazMjesto adresa}$   
 $\text{nazProd} \not\rightarrow \text{brNar}$

⇒ Na koje relacije treba razložiti relaciju narudzba?

Koji su ključevi novonastalih relacija?

$$\text{prodavaonica} = \pi_{\text{nazProd}, \text{pbr}, \text{nazMjesto}, \text{adresa}}(\text{narudzba})$$

$$K_{\text{PRODAVAONICA}} = \{ \text{nazProd} \}$$

$$\text{narudzba}_1 = \pi_{\text{brNar}, \text{nazProd}, \text{datNar}}(\text{narudzba})$$

$$K_{\text{NARUDZBA}_1} = \{ \text{brNar} \}$$

# Zadatak 1 - 3NF

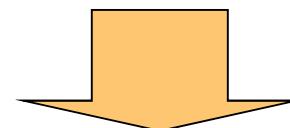
narudzba

| nazProd  | pbr   | nazMjesto | adresa   | <u>brNar</u> | datNar   |
|----------|-------|-----------|----------|--------------|----------|
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 13/25        | 1.5.2006 |
| Diona-28 | 21000 | Split     | Bolska 7 | 43-21        | 7.2.2006 |
| Konzum-7 | 10000 | Zagreb    | Ilica 20 | 41/56        | 4.2.2007 |

$\text{brNar} \rightarrow \text{nazProd}$   
 $\text{nazProd} \not\rightarrow \text{brNar}$

$\text{nazProd} \rightarrow \text{pbr} \text{ } \text{nazMjesto} \text{ } \text{adresa}$

Jesu li relacije prodavaonica i  
narudzba<sub>1</sub> u 3NF?



prodavaonica

| <u>nazProd</u> | pbr   | nazMjesto | adresa   |
|----------------|-------|-----------|----------|
| Konzum-7       | 10000 | Zagreb    | Ilica 20 |
| Diona-28       | 21000 | Split     | Bolska 7 |

3NF?

narudzba<sub>1</sub>

| <u>brNar</u> | <u>nazProd</u> | datNar   |
|--------------|----------------|----------|
| 13/25        | Konzum-7       | 1.5.2006 |
| 43-21        | Diona-28       | 7.2.2006 |
| 41/56        | Konzum-7       | 4.2.2007 |

3NF: O.K.

# Zadatak 1 - 3NF

- Postoje li neključni atributi koji tranzitivno ovise o ključu u relaciji prodavaonica?

| prodavaonica   |       |                  |          |
|----------------|-------|------------------|----------|
| <u>nazProd</u> | pbr   | <u>nazMjesto</u> | adresa   |
| Konzum-7       | 10000 | Zagreb           | Ilica 20 |
| Diona-28       | 21000 | Split            | Bolska 7 |

- $\text{nazProd} \rightarrow \text{pbr}$        $\text{pbr} \rightarrow \text{nazMjesto}$   
 $\text{pbr} \not\rightarrow \text{nazProd}$

⇒ Na koje relacije treba razložiti relaciju prodavaonica?

Koji su ključevi novonastalih relacija?

$$\text{mjesto} = \pi_{\text{pbr}, \text{nazMjesto}}(\text{prodavaonica})$$

$$K_{\text{MJESTO}} = \{ \text{pbr} \}$$

$$\text{prodavaonica}_1 = \pi_{\text{nazProd}, \text{pbr}, \text{adresa}}(\text{prodavaonica})$$

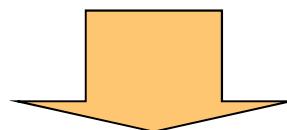
$$K_{\text{PRODAVAONICA1}} = \{ \text{nazProd} \}$$

# Zadatak 1 - 3NF

prodavaonica

| <u>nazProd</u> | pbr   | nazMjesto | adresa   |
|----------------|-------|-----------|----------|
| Konzum-7       | 10000 | Zagreb    | Ilica 20 |
| Diona-28       | 21000 | Split     | Bolska 7 |

- $\text{nazProd} \rightarrow \text{pbr}$        $\text{pbr} \rightarrow \text{nazMjesto}$
- $\text{pbr} \not\rightarrow \text{nazProd}$



Jesu li relacije mjesto i  
prodavaonica<sub>1</sub> u 3NF?

mjesto

| <u>pbr</u> | nazMjesto |
|------------|-----------|
| 10000      | Zagreb    |
| 21000      | Split     |

3NF: O.K.

prodavaonica<sub>1</sub>

| <u>nazProd</u> | pbr   | adresa   |
|----------------|-------|----------|
| Konzum-7       | 10000 | Ilica 20 |
| Diona-28       | 21000 | Bolska 7 |

3NF: O.K.

# Zadatak 1 - 3NF

| mjesto |           |
|--------|-----------|
| pbr    | nazMjesto |
| 10000  | Zagreb    |
| 21000  | Split     |

| prodavaonica <sub>1</sub> |       |          |
|---------------------------|-------|----------|
| nazProd                   | pbr   | adresa   |
| Konzum-7                  | 10000 | Ilica 20 |
| Diona-28                  | 21000 | Bolska 7 |

| artikl           |             |
|------------------|-------------|
| <u>sifArtikl</u> | nazArtikl   |
| 129              | Napolitanke |
| 139              | Albert keks |
| 221              | Domaćica    |

narudzba<sub>1</sub>

| <u>brNar</u> | nazProd  | datNar   |
|--------------|----------|----------|
| 13/25        | Konzum-7 | 1.5.2006 |
| 43-21        | Diona-28 | 7.2.2006 |
| 41/56        | Konzum-7 | 4.2.2007 |

stavkaNarudzbe<sub>1</sub>

| <u>brNar</u> | <u>sifArtikl</u> | kolicina |
|--------------|------------------|----------|
| 13/25        | 129              | 1200     |
| 13/25        | 139              | 2000     |
| 43-21        | 129              | 1200     |
| 43-21        | 221              | 1800     |
| 41/56        | 129              | 1100     |

- Shema baze podataka u 3NF sastoji se od relacijskih shema:  
*mjesto, prodavaonica<sub>1</sub>, artikl, narudzba<sub>1</sub>, stavkaNarudzbe<sub>1</sub>*

## Zadatak 2

---

Zadana je relacijska shema  $R = ABCDEF$  i na njoj skup funkcijskih zavisnosti:

$$F = \{ AB \rightarrow CD, AB \rightarrow EF, A \rightarrow F, D \rightarrow E \}.$$

Domene atributa sadrže samo jednostavne vrijednosti, vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa.

Odrediti primarni ključ relacijske sheme (tako da bude zadovoljen uvjet 1NF prema kojem neključni atributi funkcijski ovise o ključu), te shemu postupno normalizirati na 2NF i 3NF.

## Zadatak 2 – 1NF

---

$R = ABCDEF$

$F = \{ AB \rightarrow CD, AB \rightarrow EF, A \rightarrow F, D \rightarrow E \}$

- Odrediti primarni ključ relacije.

$AB \rightarrow CD$

$AB \rightarrow EF$

$\Rightarrow AB \rightarrow CDEF$  (P-1: unija)

postoji li skup  $X \subset AB$  za kojeg vrijedi  $X \rightarrow R$  ?

NE

$\Rightarrow R = ABCDEF$

$K_R = AB$

$R$  je u 1NF

## Zadatak 2 - 2NF

R = ABCDEF

$$K_R = AB$$

$$F = \{ AB \rightarrow CD, AB \rightarrow EF, A \rightarrow F, D \rightarrow E \}$$

Postoje li atributi iz zavisnog dijela koji nisu potpuno funkcijски ovisni o ključu?



- $AB \rightarrow F$  je nepotpuna FZ, jer vrijedi  $A \rightarrow F$   $R$  nije u 2NF

Odredite relacijske sheme kojima treba zamijeniti relacijsku shemu R.

# Odredite ključeve.

$$R_1 = AF$$

$$K_{R1} = A$$

## R<sub>1</sub> je u 2NF

$$R_2 = ABCDE$$

$$K_{R^2} = AB$$

## R<sub>2</sub> je u 2NF

## Zadatak 2 - 3NF

---

$$R_1 = AF$$

$$K_{R1} = A$$

$$R_2 = ABCDE$$

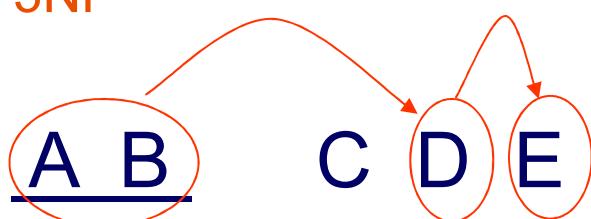
$$K_{R2} = AB$$

$$F = \{ AB \rightarrow CD, AB \rightarrow EF, A \rightarrow F, D \rightarrow E \}$$

- Jesu li  $R_1$  i  $R_2$  u 3NF?
- Postoje li u  $R_1$  i  $R_2$  neključni atributi koji tranzitivno ovise o ključu?

$R_1$  je u 3NF

$R_2$



- $AB \rightarrow E$  je tranzitivna FZ

Odredite relacijske sheme kojima treba zamijeniti relacijsku shemu  $R_2$ .

Odredite ključeve.

$$R_{21} = DE$$

$$K_{R21} = D$$

$$R_{22} = ABCD$$

$$K_{R22} = AB$$

## Zadatak 2 - 3NF

---

- Jesu li  $R_{21}$  i  $R_{22}$  u 3NF?

$$F = \{ AB \rightarrow CD, AB \rightarrow EF, A \rightarrow F, D \rightarrow E \}$$

$$R_1 = AF$$

$$K_{R1} = A$$

$R_1$  je u 3NF

$$R_{21} = DE$$

$$K_{R21} = D$$

$R_{21}$  je u 3NF

$$R_{22} = ABCD$$

$$K_{R22} = AB$$

$R_{22}$  je u 3NF

Shema baze podataka u 3NF sastoji se od relacijskih shema:

$$R_1, R_{21}, R_{22}$$

## Zadatak 3

---

U biblioteci se evidentiraju **posudbe (primjeraka) knjiga**.

Relacijska shema **POSUDBAPRIMJ** sastoji se od sljedećih atributa:

sifCln - šifra člana

prezCln - prezime člana

imeCln - ime člana

pbr - poštanski broj mjesta stanovanja člana

nazMj - naziv mjesta stanovanja člana

adrCln - adresa člana

invBrPrim – inventarski broj primjerka

datPos - datum posudbe

datVr – datum vraćanja (datum kad je primjerak vraćen)

sifKnj - šifra knjige

nazKnj - naziv knjige

sifIzd - šifra izdavača

nazIzd - naziv izdavača

Vrijede sljedeća pravila:

- jedan član istoga dana može posuditi više primjeraka
- jedan član isti primjerak može posuditi više puta, ali ne istog dana
- jedna knjiga ima jednog izdavača

# Zadatak 3

| posudbaPrimj |         |        |       |        |         | posud                                                                                                                        |
|--------------|---------|--------|-------|--------|---------|------------------------------------------------------------------------------------------------------------------------------|
| sifCln       | prezCln | imeCln | pbr   | nazMj  | adrCln  | invBrPrim, datPos, datVr, sifKnj, nazKnj, siflzd, nazlzd                                                                     |
| 123          | Novak   | Jasna  | 10000 | Zagreb | Unska 6 | <11234,3.1.2007,,567,Kiklop,12,AGM><br><21345,3.1.2007,2.2.2007,351,Geto,12,AGM<br>><br><19435,29.1.2007,, 459,Bajke,15,VBZ> |
| 124          | Horvat  | Krešo  | 10020 | Zagreb | Siget 8 | <19435,2.1.2007,9.1.2007,459,Bajke,15,VBZ><br><23414, 2.1.2007,,398,Svila, 15, VBZ>                                          |
| 234          | Grgić   | Ana    | 10000 | Zagreb | Krčka 1 | <21345,3.2.2007,,351,Geto,12,AGM>                                                                                            |

- $K_{\text{PosudbaPrimj}} = \{\text{sifCln}\}$   
 $\text{sifCln} \rightarrow \text{prezCln } \text{imeCln } \text{pbr } \text{nazMj } \text{adrCln } \text{posud}$
- Vrijednosti atributa *posud* su n-torke koje sadrže vrijednosti atributa:  
*invBrPrim, datPos, datVr, sifKnj, nazKnj, siflzd, nazlzd*
- Normalizirajte relacijsku shemu **POSUDBAPRIMJ** na 1NF (izdvajanjem atributa u novu relaciju), 2NF i 3NF

## Zadatak 3 – 1NF

| clan | <u>sifCln</u> | prezCln | imeCln | pbr   | nazMj  | adrCln  |
|------|---------------|---------|--------|-------|--------|---------|
|      | 123           | Novak   | Jasna  | 10000 | Zagreb | Unska 6 |
|      | 124           | Horvat  | Krešo  | 10020 | Zagreb | Siget 8 |
|      | 234           | Grgić   | Ana    | 10000 | Zagreb | Krčka 1 |

$\text{CLAN} = \{ \text{sifCln}, \text{prezCln}, \text{imeCln}, \text{pbr}, \text{nazMj}, \text{adrCln} \}$     $K_{\text{CLAN}} = \{ \text{sifCln} \}$

$\text{POSUDBA} = \{ \text{sifCln}, \text{invBrPrim}, \text{datPos}, \text{datVr}, \text{sifKnj}, \text{nazKnj}, \text{siflzd}, \text{nazlzd} \}$

Odredite ključ za relacijsku shemu POSUDBA tako da ona zadovoljava 1NF.

$K_{\text{POSUDBA}} = \{ \text{sifCln}, \text{invBrPrim}, \text{datPos} \}$

| posudba | <u>sifCln</u> | <u>invBrPrim</u> | <u>datPos</u> | datVr    | <u>sifKnj</u> | <u>nazKnj</u> | <u>siflzd</u> | <u>nazlzd</u> |
|---------|---------------|------------------|---------------|----------|---------------|---------------|---------------|---------------|
|         | 123           | 11234            | 3.1.2007      | NULL     | 567           | Kiklop        | 12            | AGM           |
|         | 123           | 21345            | 3.1.2007      | 2.2.2007 | 351           | Geto          | 12            | AGM           |
|         | 123           | 19435            | 29.1.2007     | NULL     | 459           | Bajke         | 15            | VBZ           |
|         | 124           | 19435            | 2.1.2007      | 9.1.2007 | 459           | Bajke         | 15            | VBZ           |
|         | 124           | 23414            | 2.1.2007      | NULL     | 398           | Svila         | 15            | VBZ           |
|         | 234           | 21345            | 3.2.2007      | NULL     | 351           | Geto          | 12            | AGM           |

## Zadatak 3 – 2NF

clan

| sifCln | prezCln | imeCln | pbr   | nazMj  | adrCln  |
|--------|---------|--------|-------|--------|---------|
| 123    | Novak   | Jasna  | 10000 | Zagreb | Unska 6 |
| 124    | Horvat  | Krešo  | 10020 | Zagreb | Siget 8 |
| 234    | Grgić   | Ana    | 10000 | Zagreb | Krčka 1 |

CLAN zadovoljava 2NF  
- ZAŠTO?

Zadovoljava li POSUDBA 2NF?

Postoje li neključni atributi koji ovise o dijelu ključa?

Normalizirajte relacijsku shemu POSUDBA na 2NF.

posudba

| sifCln | invBrPrim | datPos    | datVr    | sifKnj | nazKnj | siflzd | nazlzd |
|--------|-----------|-----------|----------|--------|--------|--------|--------|
| 123    | 11234     | 3.1.2007  | NULL     | 567    | Kiklop | 12     | AGM    |
| 123    | 21345     | 3.1.2007  | 2.2.2007 | 351    | Geto   | 12     | AGM    |
| 123    | 19435     | 29.1.2007 | NULL     | 459    | Bajke  | 15     | VBZ    |
| 124    | 19435     | 2.1.2007  | 9.1.2007 | 459    | Bajke  | 15     | VBZ    |
| 124    | 23414     | 2.1.2007  | NULL     | 398    | Svila  | 15     | VBZ    |
| 234    | 21345     | 3.2.2007  | NULL     | 351    | Geto   | 12     | AGM    |

## Zadatak 3 – 2NF

posudba1

| <u>sifCln</u> | <u>invBrPrim</u> | <u>datPos</u> | <u>datVr</u> |
|---------------|------------------|---------------|--------------|
| 123           | 11234            | 3.1.2007      | NULL         |
| 123           | 21345            | 3.1.2007      | 2.2.2007     |
| 123           | 19435            | 29.1.2007     | NULL         |
| 124           | 19435            | 2.1.2007      | 9.1.2007     |
| 124           | 23414            | 2.1.2007      | NULL         |
| 234           | 21345            | 3.2.2007      | NULL         |

2NF?  
OK

primjerak

| <u>invBrPrim</u> | <u>sifKnj</u> | <u>nazKnj</u> | <u>sifLzd</u> | <u>nazLzd</u> |
|------------------|---------------|---------------|---------------|---------------|
| 11234            | 567           | Kiklop        | 12            | AGM           |
| 21345            | 351           | Geto          | 12            | AGM           |
| 19435            | 459           | Bajke         | 15            | VBZ           |
| 23414            | 398           | Svila         | 15            | VBZ           |

2NF?  
OK

## Zadatak 3 – 3NF

| clan          |         |        |       |        |         |
|---------------|---------|--------|-------|--------|---------|
| <u>sifCln</u> | prezCln | imeCln | pbr   | nazMj  | adrCln  |
| 123           | Novak   | Jasna  | 10000 | Zagreb | Unska 6 |
| 124           | Horvat  | Krešo  | 10020 | Zagreb | Siget 8 |
| 234           | Grgić   | Ana    | 10000 | Zagreb | Krčka 1 |



Zadovoljava li CLAN 3NF?

Postoje li neključni atributi koji tranzitivno ovise o ključu?

Normalizirajte relacijsku shemu CLAN na 3NF.

| clan <sub>1</sub> |         |        |       |         |  |
|-------------------|---------|--------|-------|---------|--|
| <u>sifCln</u>     | prezCln | imeCln | pbr   | adrCln  |  |
| 123               | Novak   | Jasna  | 10000 | Unska 6 |  |
| 124               | Horvat  | Krešo  | 10020 | Siget 8 |  |
| 234               | Grgić   | Ana    | 10000 | Krčka 1 |  |

3NF?

OK

| mjesto |        |
|--------|--------|
| pbr    | nazMj  |
| 10000  | Zagreb |
| 10020  | Zagreb |

3NF?

OK

## Zadatak 3 – 3NF

posudba<sub>1</sub>

| sifCln | invBrPrim | datPos    | datVr    |
|--------|-----------|-----------|----------|
| 123    | 11234     | 3.1.2007  | NULL     |
| 123    | 21345     | 3.1.2007  | 2.2.2007 |
| 123    | 19435     | 29.1.2007 | NULL     |
| 124    | 19435     | 2.1.2007  | 9.1.2007 |
| 124    | 23414     | 2.1.2007  | NULL     |
| 234    | 21345     | 3.2.2007  | NULL     |

POSUDBA<sub>1</sub> zadovoljava 3NF  
- ZAŠTO?

primjerak

| invBrPrim | sifKnj | nazKnj | sifLzd | nazLzd |
|-----------|--------|--------|--------|--------|
| 11234     | 567    | Kiklop | 12     | AGM    |
| 21345     | 351    | Geto   | 12     | AGM    |
| 19435     | 459    | Bajke  | 15     | VBZ    |
| 23414     | 398    | Svila  | 15     | VBZ    |

Zadovoljava li PRIMJERAK 3NF?

Postoje li neključni atributi koji tranzitivno ovise o ključu?

Normalizirajte relacijsku shemu PRIMJERAK na 3NF.

## Zadatak 3 – 3NF

primjerak<sub>1</sub>

| <u>invBrPrim</u> | sifKnj |
|------------------|--------|
| 11234            | 567    |
| 21345            | 351    |
| 19435            | 459    |
| 23414            | 398    |

3NF?  
OK

knjiga

| <u>sifKnj</u> | nazKnj | siflzd | nazlzd |
|---------------|--------|--------|--------|
| 567           | Kiklop | 12     | AGM    |
| 351           | Geto   | 12     | AGM    |
| 459           | Bajke  | 15     | VBZ    |
| 398           | Svila  | 15     | VBZ    |



Zadovoljava li KNJIGA 3NF?

Postoje li u relacijskoj shemi KNJIGA atributi u zavisnom dijelu koji su tranzitivno ovisni o ključu?

Normalizirajte relacijsku shemu KNJIGA na 3NF.

knjiga<sub>1</sub>

| <u>sifKnj</u> | nazKnj | siflzd |
|---------------|--------|--------|
| 567           | Kiklop | 12     |
| 351           | Geto   | 12     |
| 459           | Bajke  | 15     |
| 398           | Svila  | 15     |

3NF?  
OK

izdavac

| <u>siflzd</u> | nazlzd |
|---------------|--------|
| 12            | AGM    |
| 15            | VBZ    |

3NF?  
OK

## Zadatak 3 – Shema baze podataka u 3NF

---

$CLAN_1 = \{ sifCln, prezCln, imeCln, pbr, adrCln \}$        $K_{CLAN1} = \{ sifCln \}$

$MJESTO = \{ pbr, nazMj \}$        $K_{MJESTO} = \{ pbr \}$

$PRIMJERAK_1 = \{ invBrPrim, sifKnj \}$        $K_{PRIMJERAK1} = \{ invBrPrim \}$

$KNJIGA_1 = \{ sifKnj, nazKnj, siflzd \}$        $K_{KNJIGA1} = \{ sifKnj \}$

$IZDAVAC = \{ siflzd, nazlzd \}$        $K_{IZDAVAC} = \{ siflzd \}$

$POSUDBA_1 = \{ sifCln, invBrPrim, datPos, datVr \}$   
 $K_{POSUDBA} = \{ sifCln, invBrPrim, datPos \}$

## Zadatak 4

---

Zadana je relacijska shema  $R = ABCDEF$  i na njoj skup funkcijskih zavisnosti:

$$F = \{ AB \rightarrow CDE, B \rightarrow EF, F \rightarrow B \} .$$

Domene atributa sadrže samo jednostavne vrijednosti, vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa.

Odrediti primarni ključ relacijske sheme (tako da bude zadovoljen uvjet 1NF prema kojem neključni atributi funkcijski ovise o ključu), te shemu postupno normalizirati na 2NF i 3NF.

## Zadatak 4 – 1NF

---

$R = ABCDEF$

$F = \{ AB \rightarrow CDE, B \rightarrow EF, F \rightarrow B \}$

- **Odrediti primarni ključ relacije.**

$AB \rightarrow CDE$

$B \rightarrow EF \Rightarrow AB \rightarrow EF$  (A-2: uvećanje)

$\Rightarrow AB \rightarrow CDEF$  (P-1: unija)

postoji li skup  $X \subset AB$  za kojeg vrijedi  $X \rightarrow R$  ?

NE

$\Rightarrow R = ABCDEF$

$K_R = AB$

$R$  je u 1NF

## Zadatak 4 - 2NF

---

$$R = ABCDEF$$

$$K_R = AB$$

$$F = \{ AB \rightarrow CD, B \rightarrow EF, F \rightarrow B \}$$

Postoje li atributi iz zavisnog dijela koji nisu potpuno funkcijski ovisni o ključu?



- $AB \rightarrow EF$  je nepotpuna FZ, jer vrijedi  $B \rightarrow EF$        $R$  nije u 2NF

Odredite relacijske sheme kojima treba zamijeniti relacijsku shemu  $R$ .

Odredite ključeve.

$$R_1 = BEF$$

$$K_{R1} = B$$

$R_1$  je u 2NF

$$R_2 = ABCD$$

$$K_{R2} = AB$$

$R_2$  je u 2NF

## Zadatak 4 - 3NF

---

$$R_1 = BEF$$

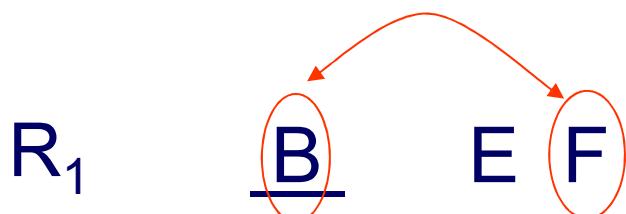
$$K_{R1} = B$$

$$R_2 = ABCD$$

$$K_{R2} = AB$$

$$F = \{ AB \rightarrow CD, B \rightarrow EF, F \rightarrow B \}$$

- Jesu li  $R_1$  i  $R_2$  u 3NF?
- Postoje li u  $R_1$  i  $R_2$  neključni atributi koji tranzitivno ovise o ključu?



- Nema tranzitivnih funkcijskih ovisnosti.
- $B \rightarrow F$  i  $F \rightarrow B$

$\Rightarrow F$  je također mogući ključ u  $R_1$

$$K1_{R1} = B$$

$$K2_{R1} = F$$

$R_1$  i  $R_2$  su u 3NF

# Zadatak 5

---

Zadane su relacijske sheme UREDJAJ i KVAR:

$$\text{UREDJAJ} = \{ \text{mbrUr}, \text{oznVrUr}, \text{nazVrUr}, \text{oznPr}, \text{nazPr} \}$$

$$K_{\text{UREDJAJ}} = \{ \text{mbrUr} \}$$

$$\text{KVAR} = \{ \text{mbrUr}, \text{datKv}, \text{oznVrKv}, \text{opVrKv}, \text{napKv} \}$$

$$K_{\text{KVAR}} = \{ \text{mbrUr}, \text{datKv}, \text{oznVrKv} \}$$

i vrijedi da se za jedan uređaj istog dana može evidentirati više različitih kvarova.

- mbrUr – matični broj uređaja
  - oznVrUr – oznaka vrste uređaja
  - nazVrUr – naziv vrste uređaja
  - oznPr - oznaka proizvođača
  - nazPr – naziv proizvođača
  - datKv – datum kvara
  - oznVrKv – oznaka vrste kvara
  - opisVrKv – opis vrste kvara
  - napKv – napomena uz kvar (napomena uz konkretni kvar na određenom uređaju određenog datuma)
- Relacijske sheme UREDJAJ i KVAR su u 1NF (provjerite!). Normalizirati te relacijske sheme na 2NF i 3NF.

## Zadatak 5 – 2NF

---

2NF?    UREDJAJ = { mbrUr, oznVrUr, nazVrUr, oznPr, nazPr }  
          K<sub>UREDJAJ</sub> = { mbrUr }

UREDJAJ zadovoljava 2NF (zašto?).

KVAR = { mbrUr, datKv, oznVrKv, opVrKv, napKv }  
          K<sub>KVAR</sub> = { mbrUr, datKv, oznVrKv }

Postoje li neključni atributi koji ne ovise o čitavom ključu nego samo o dijelu ključa?  
Normalizirajte relacijsku shemu KVAR na 2NF.

VRSTAKVARA = { oznVrKv, opVrKv }  
          K<sub>VRSTAKVARA</sub> = { oznVrKv }

KVAR<sub>1</sub> = { mbrUr, datKv, oznVrKv, napKv }  
          K<sub>KVAR1</sub> = { mbrUr, datKv, oznVrKv }

## Zadatak 5 – 3NF

---

Postoje li neključni atributi koji tranzitivno ovise o ključu?

$UREDJAJ = \{ mbrUr, oznVrUr, nazVrUr, oznPr, nazPr \}$

$K_{UREDJAJ} = \{ mbrUr \}$



Normalizirajte relacijsku shemu UREDJAJ na 3NF.

$VRSTAUREDJ = \{ oznVrUr, nazVrUr \}$        $K_{VRSTAUREDJ} = \{ oznVrUr \}$

$PROIZVODJAC = \{ oznPr, nazPr \}$        $K_{PROIZVODJAC} = \{ oznPr \}$

$UREDJAJ_1 = \{ mbrUr, oznVrUr, oznPr \}$        $K_{UREDJAJ1} = \{ mbrUr \}$

3NF?

OK

## Zadatak 5 – 3NF

---

$VRSTAKVARA = \{ oznVrKv, opVrKv \}$       3NF OK

$K_{VRSTAKVARA} = \{ oznVrKv \}$

$KVAR1 = \{ mbrUr, datKv, oznVrKv, napKv \}$       3NF OK

$K_{KVAR1} = \{ mbrUr, datKv, oznVrKv \}$

Shema baze podataka u 3NF sastoji se od relacijskih shema:  
 $VRSTAUREDJ$ ,  $PROIZVODJAC$ ,  $UREDJAJ_1$ ,  $VRSTAKVARA$ ,  $KVAR_1$

## Zadatak 6

---

Zadane su relacijske sheme LINIJA i PROMET:

LINIJA = { lin, sifOdr, nazOdr, vrijPol, trVoz }

$K_{LINIJA} = \{ \text{lin} \}$

PROMET = { lin, sifPrij, nazPrij, sifAut, tipAut, datPol, brSjed, brKart }

$K_{PROMET} = \{ \text{lin}, \text{sifPrij}, \text{sifAut}, \text{datPol} \}$

- lin – broj linije na kojoj se odvija promet
- sifPrij – šifra prijevoznika (poduzeća)
- nazPrij – naziv prijevoznika
- sifAut - šifra autobusa – određuje je prijevoznik
- tipAut – tip autobusa
- brSjed – broj sjedala
- sifOdr – šifra mjesta - odredišta
- nazOdr – naziv mjesta - odredišta
- datPol – datum polaska
- vrijPol – vrijeme polaska
- trVoz – trajanje vožnje
- brKart – broj prodanih karata

Relacijske sheme PROMET i LINIJA su u 1NF (provjeriti!)

## Zadatak 6

---

$LINIJA = \{ \text{lin}, \text{sifOdr}, \text{nazOdr}, \text{vrijPol}, \text{trVoz} \}$

$K_{LINIJA} = \{ \text{lin} \}$

$PROMET = \{ \text{lin}, \text{sifPrij}, \text{nazPrij}, \text{sifAut}, \text{tipAut}, \text{datPol}, \text{brSjed}, \text{brKart} \}$

$K_{PROMET} = \{ \text{lin}, \text{sifPrij}, \text{sifAut}, \text{datPol} \}$

Normalizirati navedene relacijske sheme na 2NF i 3NF ako vrijedi:

- linija određuje odredište, vrijeme polaska i trajanje vožnje
- istog dana na istoj liniji može prometovati više autobusa (istog ili različitih prijevoznika)
- šifru autobusa određuje prijevoznik – mogu postojati različiti autobusi različitih prijevoznika koji imaju istu šifru
- autobusi istog tipa imaju jednak broj sjedala

## Zadatak 6 – 2NF

---

2NF?

LINIJA = { lin, sifOdr, nazOdr, vrijPol, trVoz }

2NF OK

PROMET = { lin, sifPrij, nazPrij, sifAut, tipAut, datPol, brSjed, brKart }

- šifru autobusa određuje prijevoznik – mogu postojati različiti autobusi različitih prijevoznika koji imaju istu šifru

sifPrij sifAut → tipAut brSjed

Normalizirajte relacijsku shemu PROMET na 2NF.

PRIJEVOZNIK = { sifPrij, nazPrij }

K<sub>PRIJEVOZNIK</sub> = { sifPrij }

AUTOBUS = { sifPrij, sifAut, tipAut, brSjed }

K<sub>AUTOBUS</sub> = { sifPrij, sifAut }

PROMET<sub>1</sub> = { lin, sifPrij, sifAut, datPol, brKart }

K<sub>PROMET1</sub> = { lin, sifPrij, sifAut, datPol }

## Zadatak 6 – 3NF

---

LINIJA = { lin, sifOdr, nazOdr, vrijPol, trVoz }       $K_{LINIJA} = \{ lin \}$     3NF?

Normalizirajte relacijsku shemu LINIJA na 3NF.

ODREDISTE = { sifOdr, nazOdr }       $K_{ODREDISTE} = \{ sifOdr \}$

$LINIJA_1 = \{ lin, sifOdr, vrijPol, trVoz \}$        $K_{LINIJA1} = \{ lin \}$

PRIJEVOZNIK = { sifPrij, nazPrij }       $K_{PRIJEVOZNIK} = \{ sifPrij \}$     3NF OK

$PROMET_1 = \{ lin, sifPrij, sifAut, datPol, brKart \}$

$K_{PROMET1} = \{ lin, sifPrij, sifAut, datPol \}$

3NF?

OK

## Zadatak 6 – 3NF

---

$\text{AUTOBUS} = \{ \text{sifPrij}, \text{sifAut}, \text{tipAut}, \text{brSjed} \}$

3NF?

$K_{\text{AUTOBUS}} = \{ \text{sifPrij}, \text{sifAut} \}$

- autobusi istog tipa imaju jednak broj sjedala

Normalizirajte relacijsku shemu **AUTOBUS** na 3NF.

$\text{TIPAUTOB} = \{ \text{tipAut}, \text{brSjed} \}$

$K_{\text{TIPAUTOB}} = \{ \text{tipAut} \}$

$\text{AUTOBUS}_1 = \{ \text{sifPrij}, \text{sifAut}, \text{tipAut} \}$

$K_{\text{AUTOBUS}_1} = \{ \text{sifPrij}, \text{sifAut} \}$

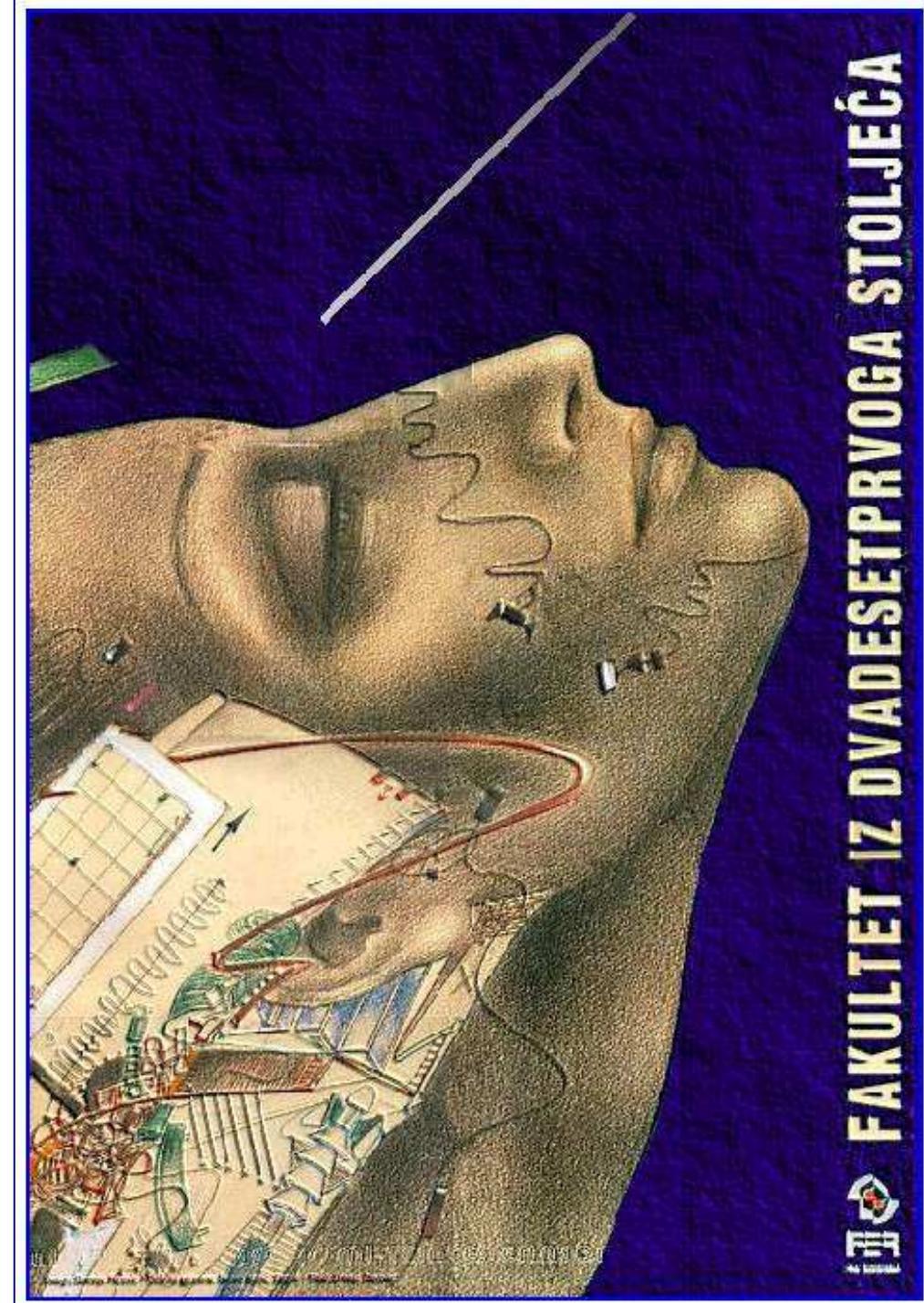
3NF OK

Shema baze podataka u 3NF sastoji se od relacijskih shema:  
**ODREDISTE, LINIJA<sub>1</sub>, PRIJEVOZNIK,**  
**PROMET<sub>1</sub>, TIPAUTOB, AUTOBUS<sub>1</sub>**

# Baze podataka

Predavanja  
travanj 2014.

## 9. Fizička organizacija podataka



# UVOD - Fizička organizacija

---

- Pojam fizičke organizacije podataka odnosi se na:
  - strukture podataka primjenjene pri pohrani podataka u sekundarnoj memoriji
  - metode pristupa (*access methods*): postupci koji se primjenjuju pri obavljanju operacija nad podacima
- Fizička organizacija podataka ne utječe na rezultate operacija s podacima, ali ima vrlo veliki utjecaj na učinkovitost sustava za upravljanje bazama podataka
  - važna zadaća sustava za upravljanje bazama podataka: obavljati operacije nad velikim količinama podataka **na učinkovit način**
- SUBP skriva od korisnika detalje fizičke organizacije podataka jer za većinu korisnika sustava nisu značajni

# Pohrana baze podataka u sekundarnoj memoriji

---

- Glavna memorija (radni spremnik, *main memory*)
  - velike brzina pristupa podacima (10-100 ns), relativno skupa, kapacitet  $\sim$  GB
  - neprikladna za pohranu baze podataka jer:
    - ima kapacitet nedovoljan za pohranu baze podataka (previsoka cijena za pohranu velikih količina podataka)
    - nepostojana (*volatile*) memorija: sadržaj memorije se gubi pri gubitku napajanja ili pri pogrešci sustava

# Pohrana baze podataka u sekundarnoj memoriji

---

- karakteristike medija za pohranu podataka uvjetuju da se većina današnjih baza podataka pohranjuje u **sekundarnoj memoriji** (uobičajeno: magnetski diskovi)
  - podaci se između sekundarne i primarne memorije prenose u blokovima (tipično 512 B, 1 kB, 2kB, 4kB)
- ⇒ **dominiraju troškovi U/I (ulazno/izlaznih) operacija:** vrijeme potrebno za obavljanje U/I operacije radi prijenosa bloka podataka između sekundarne i primarne memorije znatno je veće od vremena koje će biti utrošeno za obavljanje operacija nad podacima u primarnoj memoriji

# Važniji ciljevi fizičke organizacije

---

- minimizirati broj U/I operacija pri pohrani i dohvatu podataka, minimizirati utrošak prostora za pohranu
  - u koji fizički blok pohraniti logički zapis odnosno n-torku
  - koje je dodatne informacije potrebno pohraniti da bi se omogućio učinkovit pristup podacima
- omogućiti različite metode pristupa koje se koriste za pronalaženje fizičke pozicije zapisa (ili fizičke pozicije bloka u kojem se taj zapis nalazi) na temelju vrijednosti ključa pretrage
  - ključ pretrage (*search key*) ne mora nužno biti primarni ili alternativni ključ. Ključ pretrage može biti bilo koji atribut ili skup atributa relacije ("sekundarni ključ").
  - primjenjivost pojedinih metoda pristupa podacima ovisi o primijenjenim strukturama podataka

# Strukture podataka i metode pristupa podacima

---

- Primjena različitih struktura podataka omogućava različite metode pristupa podacima
- Ne postoji "najbolja" metoda fizičke organizacije, ali dvije se u današnjim sustavima za upravljanje bazama podataka koriste najčešće
  - **Neporedana (*heap*) datoteka**
  - Poredana datoteka (*sorted file*)
  - Raspršena datoteka (*hash file*)
  - Indeksno-slijedna organizacija (*index-sequential file*)
  - **B-stablo (*B-tree*)**

# Neporedana (*heap*) datoteka

---

- zapis se upisuje na bilo koje slobodno mjesto u datoteci
- pristup podacima (dohvat podatka sa zadanom vrijednošću ključa pretrage) moguć je isključivo linearnim pretraživanjem



- koristi se za relacije s malim brojem n-torki ili u relacijama čiji se podaci uvijek obrađuju slijedno

# Neporedana (*heap*) datoteka

---

- Dohvat zapisa prema ključu pretrage
  - prema primarnom ili alternativnom ključu
    - u prosjeku je potrebno obaviti  $n/2$  U/I operacija (n predstavlja broj fizičkih blokova u kojima su pohranjeni logički zapisi odnosno n-torce)
    - još gore: u slučaju kada traženi zapis ne postoji, sustav će morati obaviti n U/I operacija
  - prema ostalim ključevima pretrage ili prema zadanim granicama intervala
    - potrebno je obaviti prijenos n fizičkih blokova

---

# B-Stabla

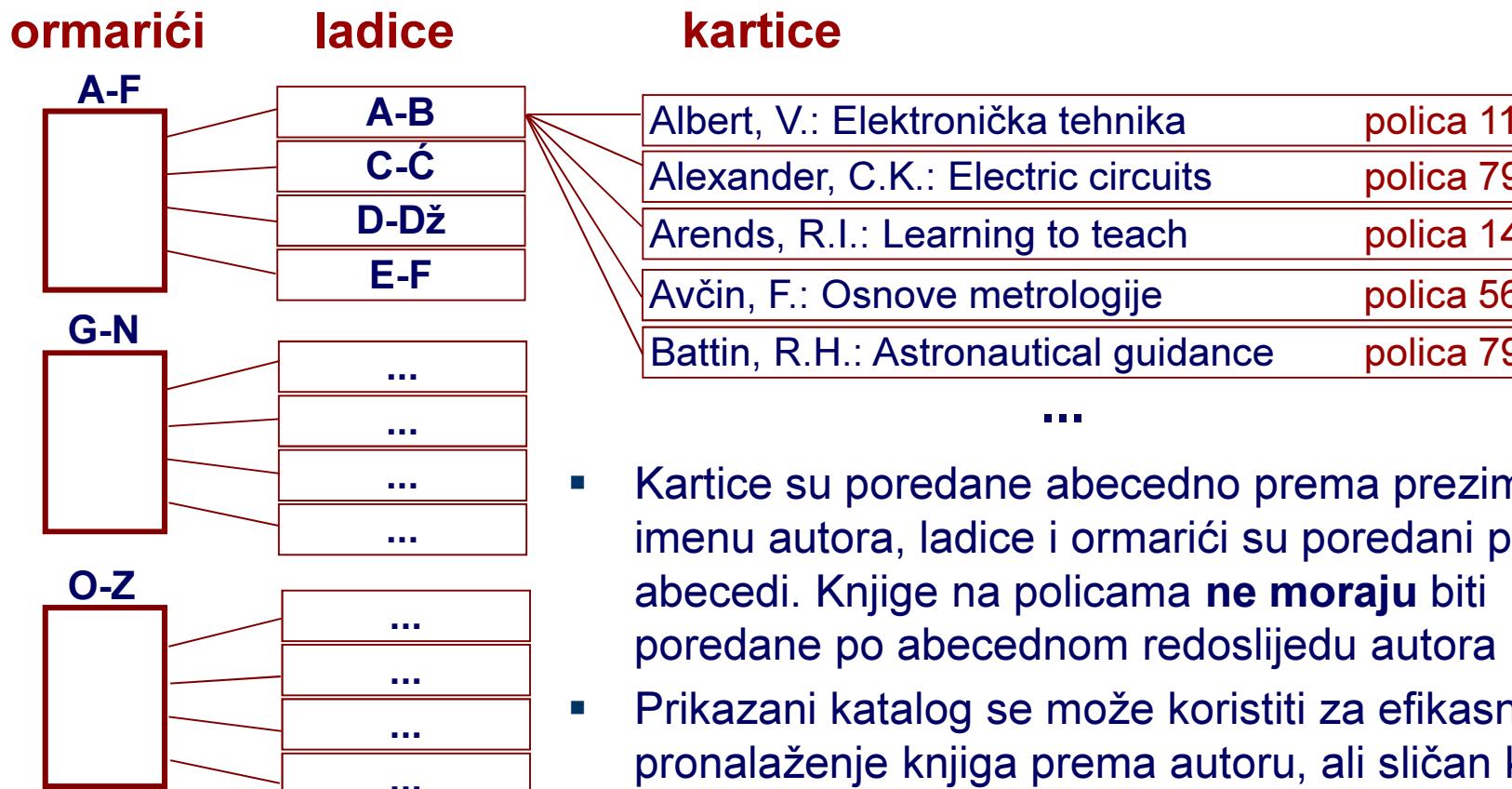
Literatura:

1. **Silberschatz, Korth, Sudarshan:** Database System Concepts
- 2 .**Garcia-Molina, Ullman, Widom:** Database Systems -The Complete Book

Ovdje će biti opisana varijanta B-stabla koja se naziva **B<sup>+</sup>- stablo**. Opisi ostalih varijanti B-stabala (B\*-stablo, B-stablo, ...) mogu se pronaći u literaturi.

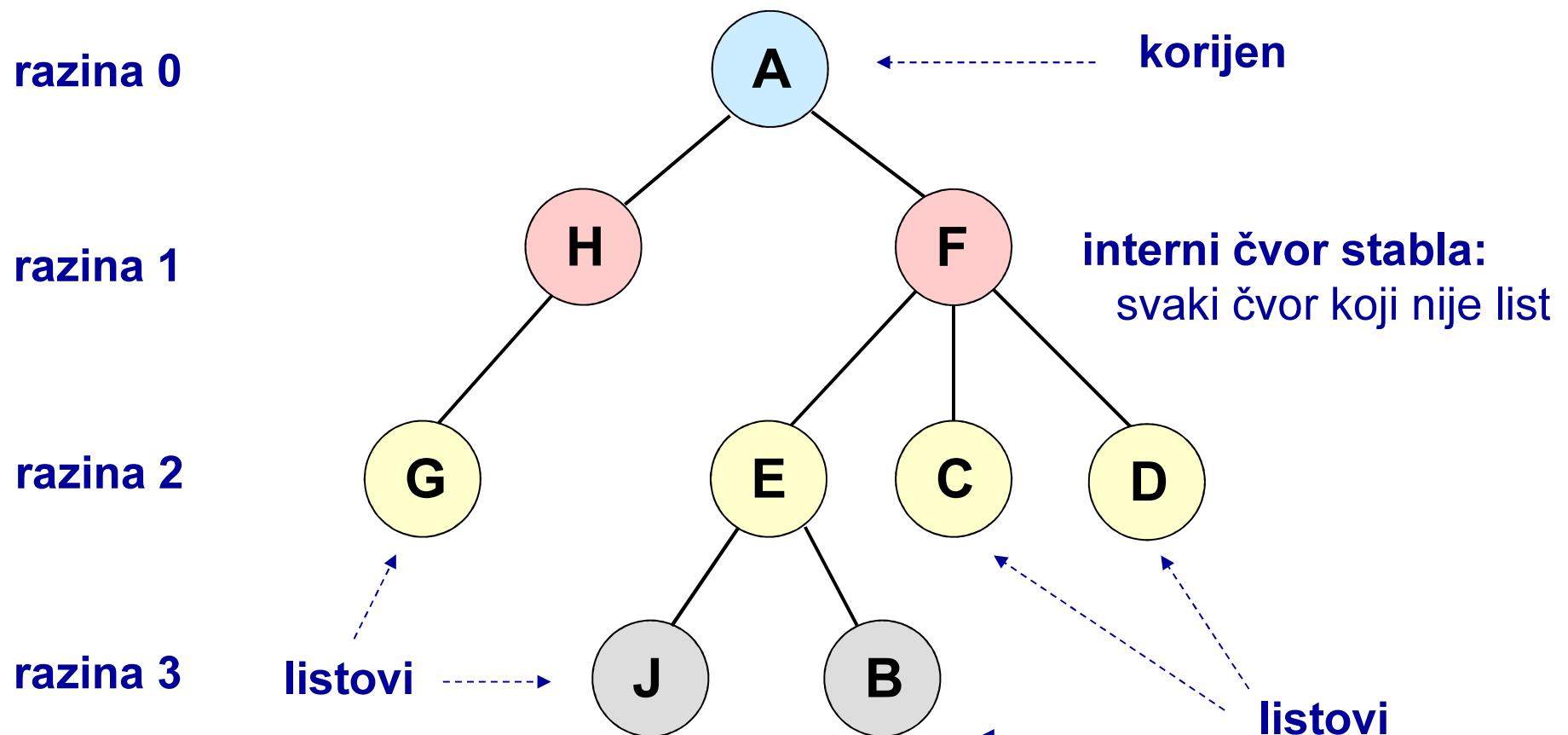
# B-stabla

- Ideja se temelji na izgradnji indeksnog kazala na više razina: slično kao kod kataloga u papirnatom obliku u knjižnicima (danас se rijetko koriste)



- Kartice su poredane abecedno prema prezimenu i imenu autora, ladice i ormarići su poredani prema abecedi. Knjige na policama **ne moraju** biti poredane po abecednom redoslijedu autora
- Prikazani katalog se može koristiti za efikasno pronalaženje knjiga prema autoru, ali sličan katalog se može izgraditi i za brzo pronalaženje knjiga prema naslovu ili prema nekim drugim pojmovima.

# Stablo kao struktura podataka



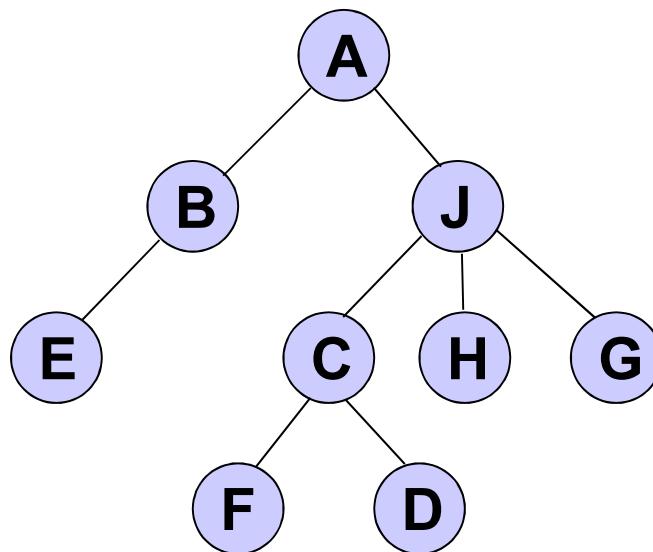
- **razina čvora (level):** duljina puta od korijena do čvora
- **dubina stabla (depth):** najveća duljina puta od korijena do lista
- **red stabla (order):** najveći broj djece koje čvor može imati

# Stablo kao struktura podataka

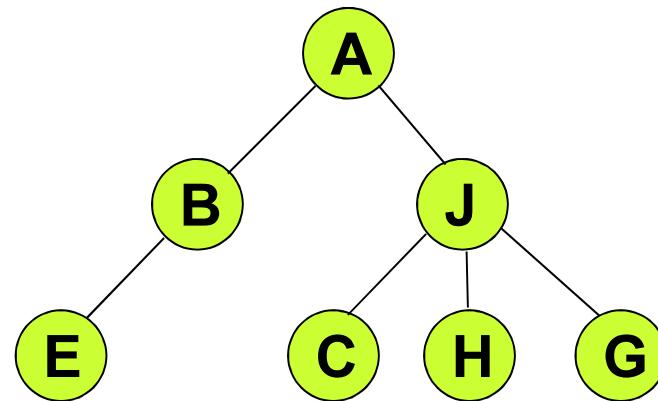
---

Stablo je **balansirano** (*balanced*) ukoliko je duljina puta od korijena do lista jednaka za svaki list u stablu

stablo nije balansirano

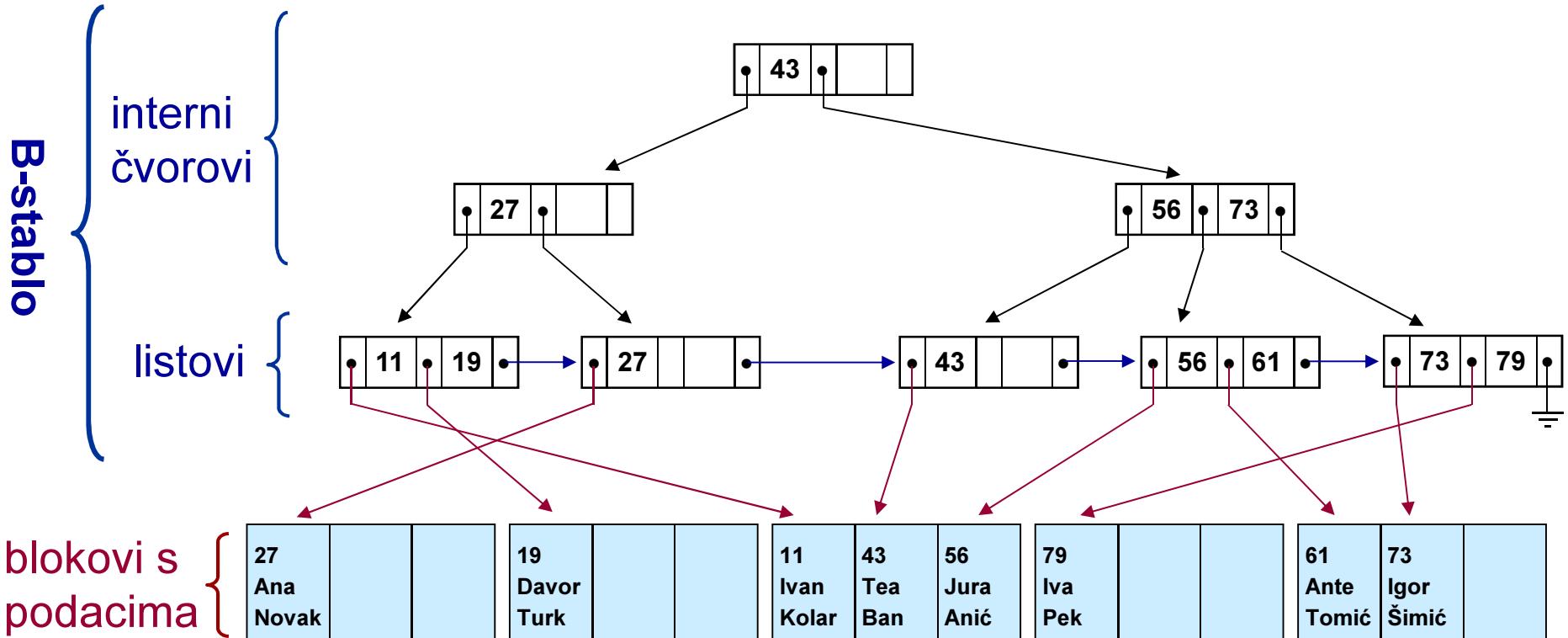


stablo je balansirano



Oznaka **B** u B-stablo znači "**balansirano**"!

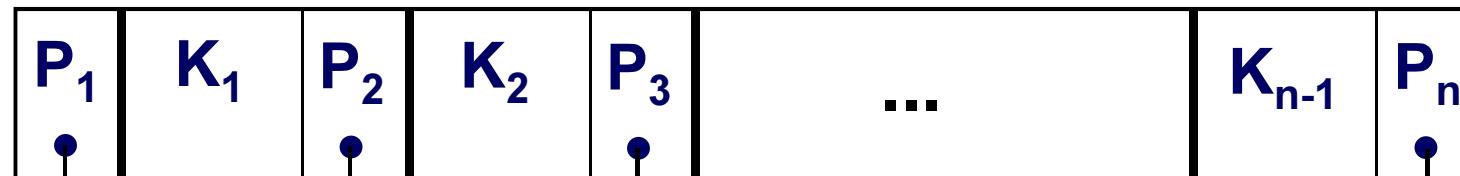
# Struktura B<sup>+</sup>-stabla



- Shema: mbr, ime, prezime; B<sup>+</sup>-stablo je izgrađeno za atribut mbr
- Moguće metode pristupa podacima (za bilo koji ključ pretrage):
  - linearnim pretraživanjem (kao kod neporedane datoteke)
  - ako je ključ pretrage mbr, može se koristiti B-stablo

# Struktura internog čvora $B^+$ -stabla

- U  $B^+$ -stablu reda  $n$ , **interni čvor** sadrži:
  - najviše  $n$  kazaljki
  - najmanje  $\lceil n/2 \rceil$  kazaljki  $\rightarrow \lceil a \rceil$  je najmanji cijeli broj  $\geq a$ 
    - ovo ograničenje ne vrijedi za korijen (najmanji broj kazaljki je 2)
  - uz  $p$  kazaljki u čvoru, broj pripadnih vrijednosti  $K_i$  u čvoru je  $p-1$ 
    - $K_i$  je vrijednost ključa



podstablo s  
vrijednostima  
ključa  $k < K_1$

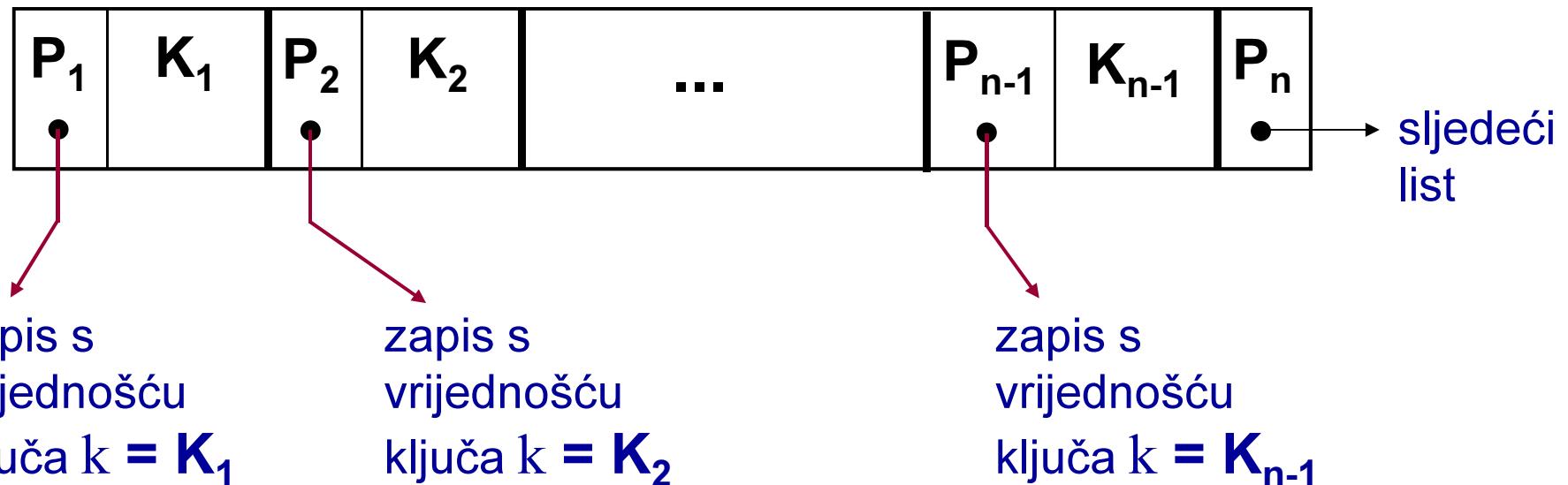
podstablo s  
vrijednostima  
ključa  $k$   
 $K_1 \leq k < K_2$

podstablo s  
vrijednostima  
ključa  $k$   
 $K_2 \leq k < K_3$

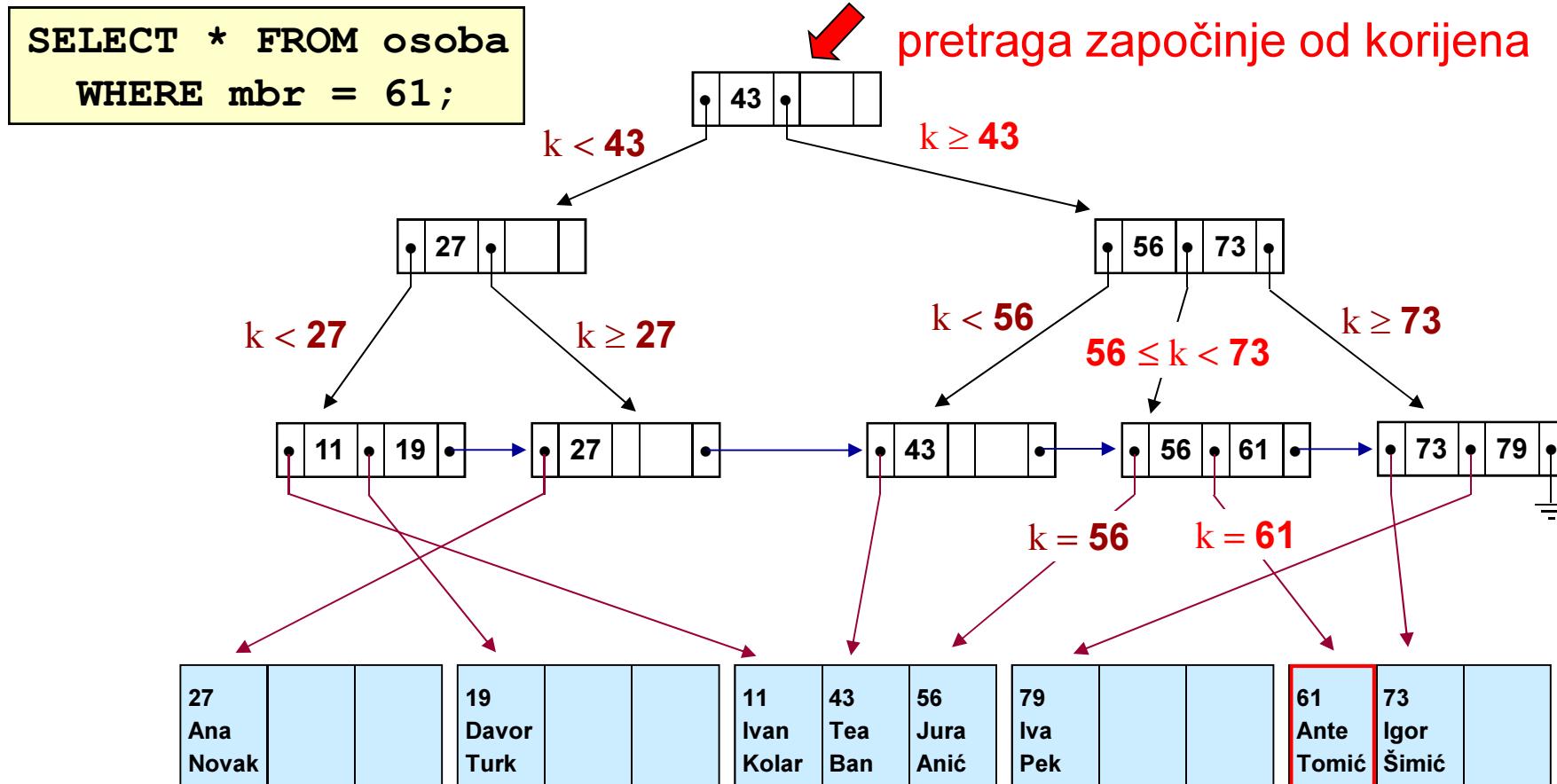
podstablo s  
vrijednostima  
ključa  $k \geq K_{n-1}$

# Struktura lista B<sup>+</sup>-stabla

- U B<sup>+</sup>-stablu reda n, list sadrži:
  - najviše  $n-1$  vrijednosti  $K_i$  i pripadnih kazaljki na zapise
  - najmanje  $\lceil (n-1)/2 \rceil$  vrijednosti  $K_i$  i pripadnih kazaljki na zapise
  - svi listovi sadrže kazaljku na sljedeći list
    - omogućava upite tipa od-do (prema zadanim granicama intervala)



# Algoritam za pronalaženje zapisa putem B<sup>+</sup>-stabla



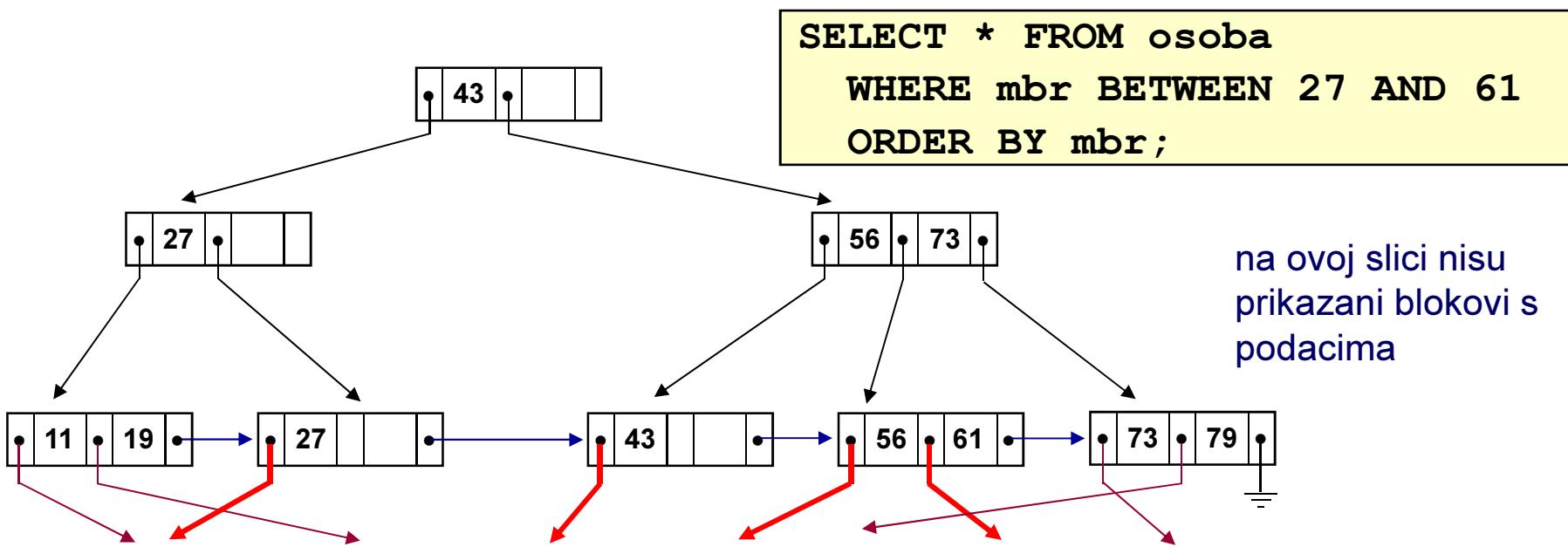
- slijediti odgovarajuću kazaljku do sljedeće razine
- postupak se ponavlja dok se ne dođe do lista u kojem će se naći kazaljka na zapis u bloku s podacima

# Algoritam za pronalaženje zapisa putem $B^+$ -stabla

---

- algoritam za traženje zapisa s ključem vrijednosti  $k$  je rekursivan
  - cilj je u svakom koraku rekurzije (pretraga i-te razine) pronaći čvor na nižoj,  $(i+1)$ -voj razini, koji će voditi prema listu u kojem se nalazi ključ čija je vrijednost  $k$
- traženje zapisa započinje od korijena (0-te razine)
- u čvoru i-te razine potrebno je pronaći najveću vrijednost ključa koja je manja ili jednaka traženoj vrijednosti  $k$ 
  - za prvu kazaljku internog čvora nije navedena vrijednost ključa, pa ona "pokriva" sve vrijednosti ključeva manje od prve vrijednosti ključa ( $K_1$ ) navedene u čvoru
- nakon pronalaženja odgovarajuće vrijednosti ključa, slijedi se pripadna kazaljka i time se obavlja pozicioniranje na  $(i+1)$ -vu razinu
- postupak se ponavlja rekursivno sve dok se ne dođe do lista. U njemu se mora nalaziti, ukoliko postoji, ključ čija je vrijednost  $k$ , te pripadna kazaljka prema traženom zapisu (n-torki)

# Dohvat podataka iz intervala, sortiranje



- u listu pronaći kazaljku na zapis s ključem **27**
- redom dohvaćati kazaljke i pripadne zapise dok se ne dođe do kazaljke na zapis s ključem **61**
  - taj postupak omogućavaju kazaljke među listovima
- dobiveni su svi traženi zapisi, pri tome su poredani prema mbr
- Ako se obavlja **SELECT \* ... ORDER BY mbr DESC**
  - pronađene zapise jednostavno ispisati obrnutim redoslijedom

# Dodavanje i brisanje zapisa

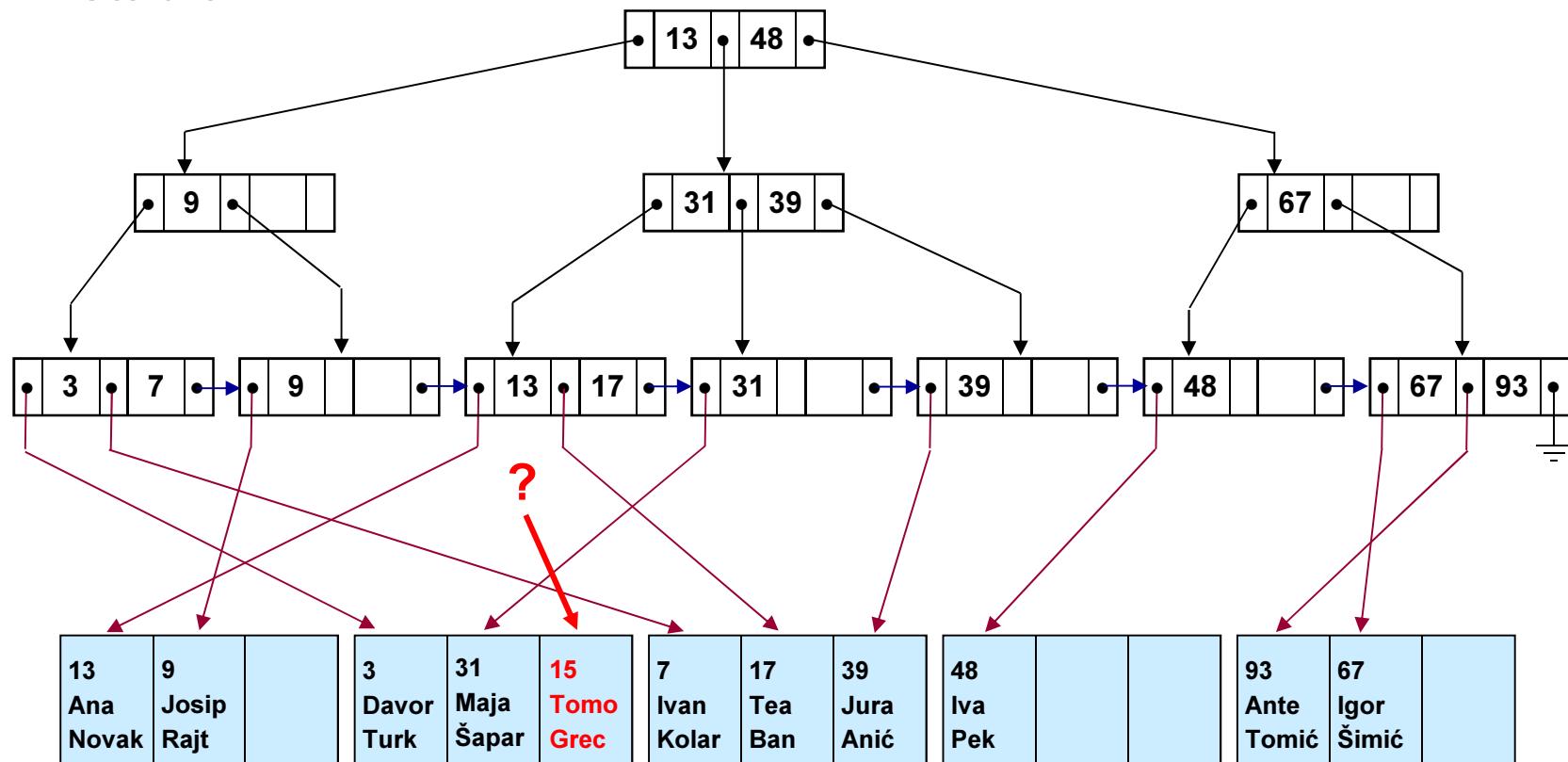
---

- nakon dodavanja ili brisanja zapisa u bloku s podacima, mijenja se i sadržaj  $B^+$ - stabla
  - koriste se algoritmi za dodavanje i brisanje zapisa u  $B^+$  - stablu
  - algoritmi osiguravaju ispravnu popunjenošću internih čvorova i listova  $B^+$ - stabla
  - pri tome se može dogoditi da stablo promijeni dubinu
- operacija izmjene
  - ukoliko se ne mijenjaju vrijednosti atributa za koje je izgrađeno  $B^+$ -stablo, u  $B^+$ -stablu nisu potrebne izmjene
  - ukoliko se mijenjaju vrijednosti atributa za koje je izgrađeno  $B^+$ -stablo, u  $B^+$ -stablu se obavlja algoritam za brisanje zapisa i algoritam za dodavanje zapisa
- **Važno dobro svojstvo algoritama B-stabla:** dubina stabla se automatski prilagođava broju zapisa - čvorovi stabla (osim korijena) su uvijek barem 50% popunjeni

# Primjer dodavanja zapisa

- dodavanje zapisa u  $B^+$ -stablu:

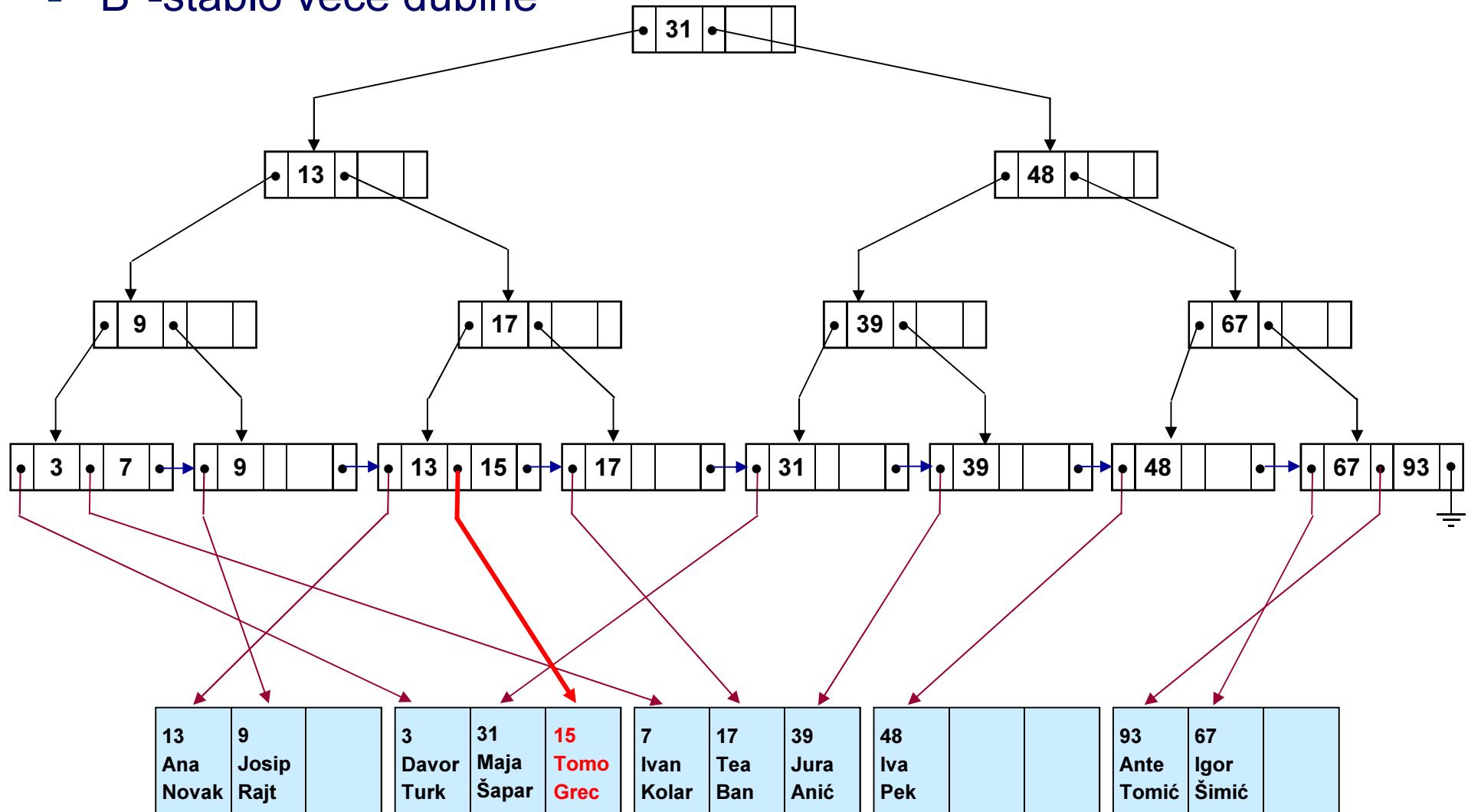
```
INSERT INTO osoba
VALUES (15, 'Tomo', 'Grec');
```



- rezultat dodavanja zapisa u  $B^+$ -stablu je na sljedećoj slici:

# Rezultat dodavanja zapisa

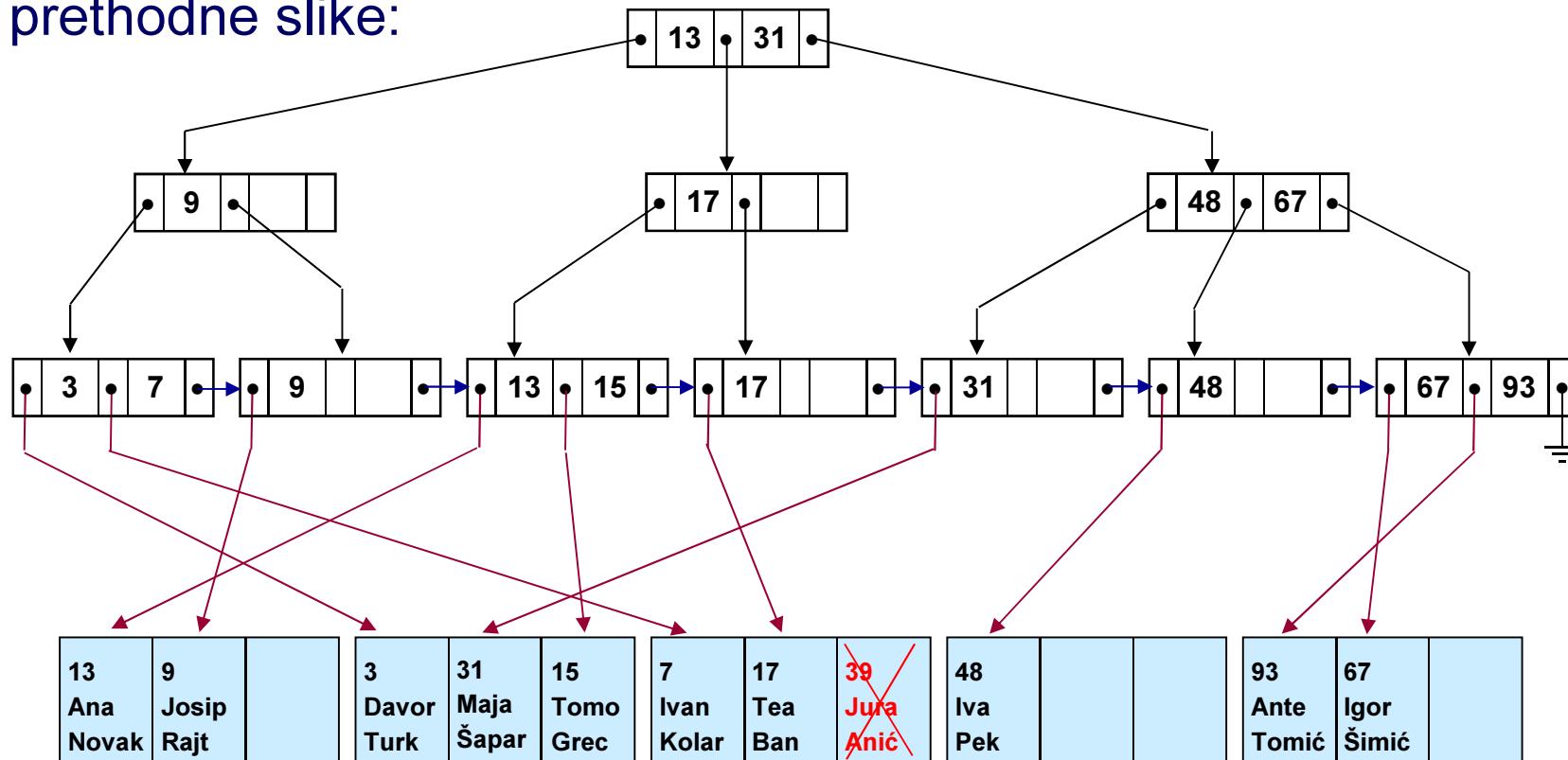
- $B^+$ -stablo veće dubine



# Primjer brisanja zapisa

- obavljanjem operacije nad  $B^+$ -stablom s prethodne slike:

```
DELETE FROM osoba WHERE mbr = 39;
```



- $B^+$ -stablo manje dubine

# Učinkovitost operacije pretrage u B<sup>+</sup>-stablu

---

- pretpostavka: stablo reda **n** sadrži kazaljke na **m** zapisa podataka
- **n** se odabire tako da se sadržaj čvora može smjestiti u jedan fizički blok
  - ⇒ za dohvat **jednog** čvora potrebna je **jedna** U/I operacija
- broj U/I operacija u stablu pri traženju zapisa ovisi o broju razina u stablu jer se pri dohvatu zapisa mora obaviti po jedna U/I operacija za svaki čvor B-stabla na putu od korijena do lista
- B-stablo ima najveći broj razina onda kada su čvorovi najmanje popunjeni
  - ⇒ moguće je odrediti koliko će U/I operacija biti potrebno obaviti u najlošijem slučaju

# Učinkovitost operacije pretrage u B<sup>+</sup>-stablu

---

- **Primjer:** za broj n-torki  $m = 1\ 000\ 000$ , za **red** stabla  $n = 70$ , ukupni broj razina (uključujući i razinu korijena) u najlošijem slučaju je 4:

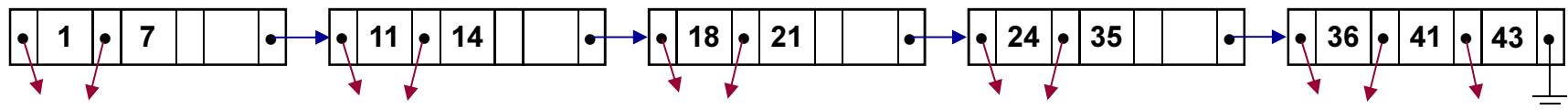
1.  točno 1 čvor, najmanje 2 kazaljke
2.  najmanje 2 čvora, najmanje 70 kazaljki
3.  najmanje 70 čvorova, najmanje 2 450 kazaljki
4.  najmanje 2 450 čvorova, najmanje 85 750 kazaljki
5.  najmanje 85 750 čvorova, najmanje 3 001 250 kazaljki

- B<sup>+</sup>-stablo koje bi imalo ukupno 5 razina, moralo bi imati **najmanje** 3 001 250 kazaljki na zapise. To znači da B-stablo reda 70 čije kazaljke u listovima pokazuju na 1 000 000 n-torki može imati najviše 4 razine.

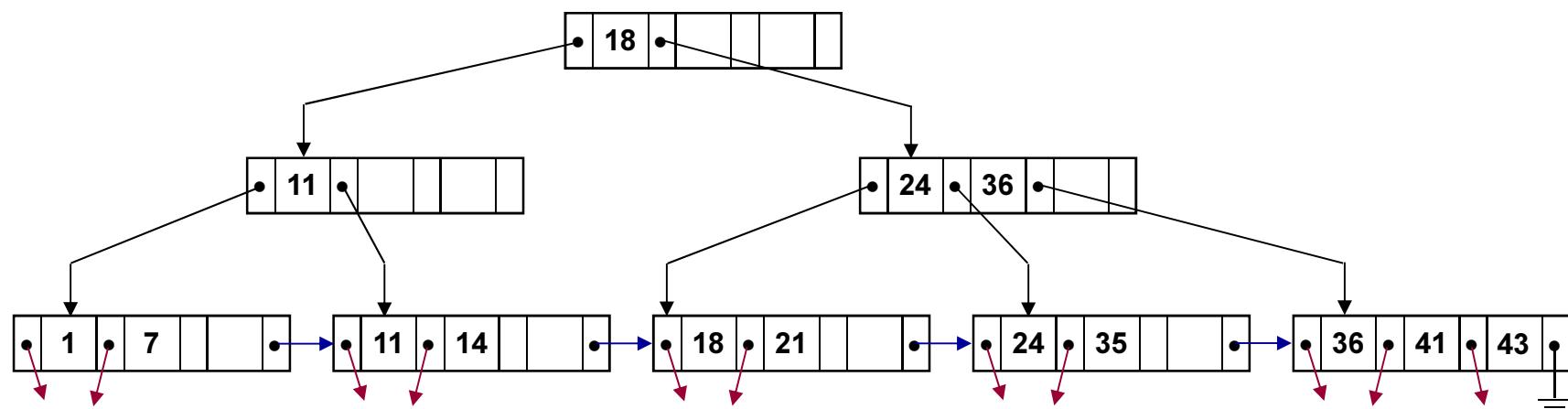
⇒ Za dohvatzanja prema vrijednosti ključa potrebno je najviše 5 U/I operacija (4 U/I operacije za dohvatzanje lista u kojem se nalazi kazaljka na zapis + 1 U/I operacija za dohvatzanje bloka s podacima)

# Zadatak 1.

- Relacija *stud* (*mbr*,  *prez*, *ime*) sadrži n-torke sa sljedećim vrijednostima atributa *mbr*: 1, 7, 11, 14, 18, 21, 24, 35, 36, 41, 43. Nacrtati  $B^+$ -stablo reda 4 za atribut *mbr* tako da popunjeno stablo bude minimalna.
- min. broj kazaljki na zapise (n-torke) u jednom listu je  $\lceil (4 - 1) / 2 \rceil = 2$

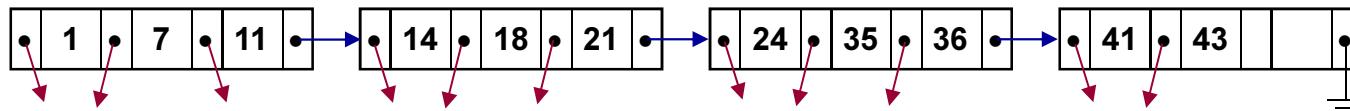


- min. broj kazaljki u jednom internom čvoru (osim korijena) je  $\lceil 4 / 2 \rceil = 2$

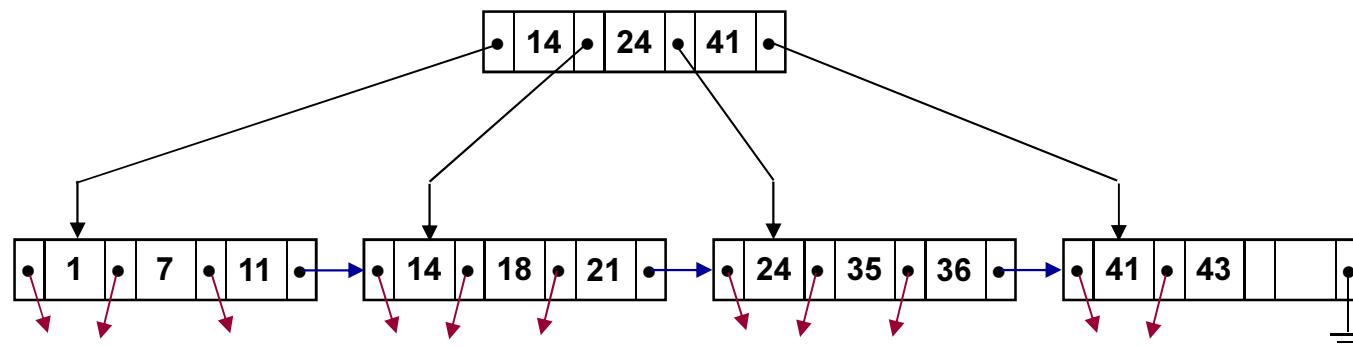


## Zadatak 2.

- Relacija *stud* (*mbr*,  *prez*, *ime*) sadrži n-torce sa sljedećim vrijednostima atributa *mbr*: 1, 7, 11, 14, 18, 21, 24, 35, 36, 41, 43. Nacrtati  $B^+$ -stablo reda 4 za atribut *mbr* tako da popunjenočvorova u stablu bude maksimalna.
- maksimalni broj kazaljki na zapise (n-torce) u jednom listu je  $4 - 1 = 3$



- maksimalni broj kazaljki u jednom internom čvoru je 4



## Zadatak 3.

---

- Koliko n-torki sadrži relacija ako je nad njom izgrađeno B<sup>+</sup>-stablo reda 101, s ukupno 5 razina, s minimalno dopuštenom popunjenošću svih čvorova
- min. broj kazaljki u jednom listu je  $\lceil (101 - 1) / 2 \rceil = 50$
- min. broj kazaljki u jednom internom čvoru (osim korijena) je  $\lceil 101 / 2 \rceil = 51$
- min. broj kazaljki u korijenu je 2
- relacija sadrži  $2 \cdot 51 \cdot 51 \cdot 51 \cdot 50 \approx 1.33 \cdot 10^7$  n-torki
- **ZAKLJUČAK:** ako je B-stablo reda 101, do svake n-torke u relaciji koja sadrži  $\approx 1.33 \cdot 10^7$  n-torki može se pristupiti, u najlošijem slučaju, korištenjem tek 6 U/I operacija (5 U/I za dohvati lista, 1 U/I za dohvati fizičkog bloka u kojem se nalazi n-torka)

## Zadatak 4.

---

- Koliko n-torki sadrži relacija ako je nad njom izgrađeno  $B^+$ -stablo reda 101, s ukupno 5 razina, s maksimalno popunjениm **svim** čvorovima
- max. broj kazaljki u jednom listu je  $101 - 1 = 100$
- max. broj kazaljki u internom čvoru je **101**
- relacija sadrži  $101 \cdot 101 \cdot 101 \cdot 101 \cdot 100 \approx 1.04 \cdot 10^{10}$  n-torki
  
- **ZAKLJUČAK:** ako je B-stablo reda 101, u najboljem slučaju, korištenjem tek 6 U/I operacija može se dohvatiti blok s n-torkom koja se nalazi u relaciji koja sadrži čak  $\approx 1.04 \cdot 10^{10}$  n-torki

# SQL: Indeksi

## 7. CREATE INDEX Statement

```
CREATE [UNIQUE] INDEX index ON table [synonym] (column [ASC | DESC])
```

- Obavljanjem naredbe za kreiranje indeksa nad relacijom, nad blokovima s podacima relacije formira se struktura B-stabla

1

```
CREATE TABLE osoba (
 mbr INTEGER,
 ime NCHAR(20),
 prez NCHAR(20));
INSERT INTO ...;
```



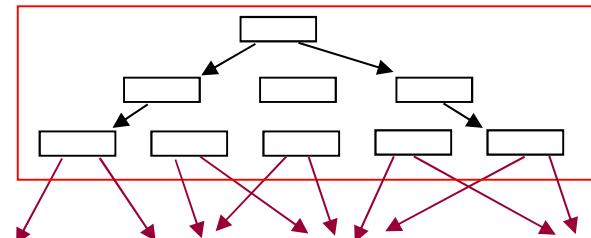
|                    |                  |                    |                  |  |  |                     |                     |  |
|--------------------|------------------|--------------------|------------------|--|--|---------------------|---------------------|--|
| 7<br>Ivan<br>Kolar | 17<br>Tea<br>Ban | 39<br>Jura<br>Anić | 48<br>Iva<br>Pek |  |  | 93<br>Ante<br>Tomić | 67<br>Igor<br>Šimić |  |
|--------------------|------------------|--------------------|------------------|--|--|---------------------|---------------------|--|

2

```
CREATE INDEX osoba_pres
ON osoba (pres);
```



B-stablo za osoba.pres



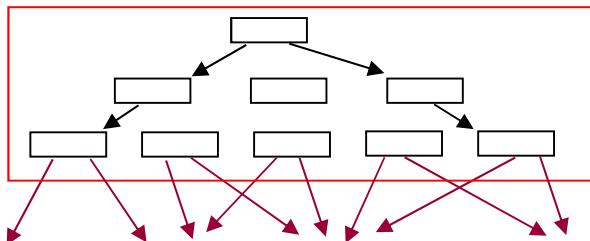
# SQL: Indeksi

- Kreiranjem indeksa uz navođenje rezervirane riječi UNIQUE osigurava se jedinstvenost vrijednosti navedenog atributa

```
CREATE UNIQUE INDEX osoba_mbr
ON osoba (mbr);
```



B-stablo za osoba.mbr



|                    |                  |                    |                  |  |  |                     |                     |  |
|--------------------|------------------|--------------------|------------------|--|--|---------------------|---------------------|--|
| 7<br>Ivan<br>Kolar | 17<br>Tea<br>Ban | 39<br>Jura<br>Anić | 48<br>Iva<br>Pek |  |  | 93<br>Ante<br>Tomić | 67<br>Igor<br>Šimić |  |
|--------------------|------------------|--------------------|------------------|--|--|---------------------|---------------------|--|

- ukoliko se indeks pokuša kreirati nad relacijom u kojoj već postoji duplikat vrijednosti atributa mbr, sustav će odbiti kreirati indeks i dojaviti pogrešku
- pokuša li se nakon kreiranja ovog indeksa unijeti n-torka s vrijednošću atributa mbr koja već postoji u nekoj n-torci, sustav će odbiti operaciju i dojaviti pogrešku

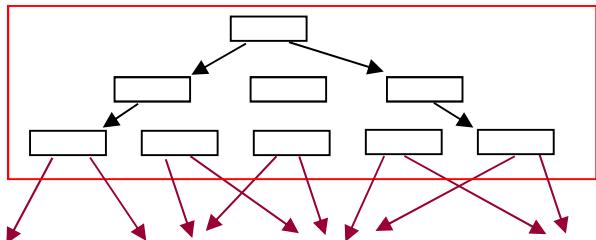
- Uništavanje indeksa - primjer:

```
DROP INDEX osoba_mbr;
```

# Više indeksa nad istom relacijom

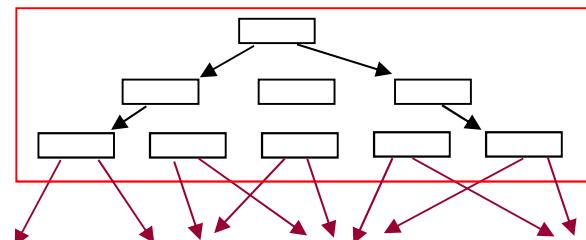
- nad istom relacijom može se izgraditi više indeksa

B-stablo za osoba.mbr



|                     |                  |                    |                  |  |  |  |
|---------------------|------------------|--------------------|------------------|--|--|--|
| 11<br>Ivan<br>Kolar | 43<br>Tea<br>Ban | 56<br>Jura<br>Anić | 79<br>Iva<br>Pek |  |  |  |
|---------------------|------------------|--------------------|------------------|--|--|--|

B-stablo za osoba.prez



|                     |                     |  |
|---------------------|---------------------|--|
| 61<br>Ante<br>Tomić | 73<br>Igor<br>Šimić |  |
|---------------------|---------------------|--|

- B-stabla (indeksi) prikazani u primjeru omogućuju efikasno obavljanje upita s uvjetima ( $=$ ,  $>$ ,  $\geq$ ,  $<$ ,  $\leq$ , BETWEEN) i efikasno sortiranje (ASC, DESC)
  - za atribut mbr
  - za atribut prez
- prema uvjetima koji sadrže atribut ime, podacima se može pristupati jedino linearnom pretragom svih blokova

# Više indeksa nad istom relacijom

---

- Ako indeksi omogućuju efikasan pristup do n-torki, znači li to da bi indekse trebalo kreirati za svaki atribut u relaciji?

**NE !!! Zašto?**

- indeksi zauzimaju prostor
- operacija unosa ili brisanja n-torke
  - uvijek rezultira promjenama (manjim ili većim) B-stabla
  - npr. ako je nad relacijom izgrađeno 10 različitih indeksa, unosom jedne n-torke u blokove s podacima morat će se unijeti zapisi i u 10 različitih indeksa
- operacija izmjene n-torke
  - izmjena vrijednosti atributa A jedne n-torke rezultirat će brisanjem i dodavanjem zapisa u svim B-stablima za indekse u kojima se koristi atribut A

# Za koje atribute treba kreirati indeks?

---

- za atribute koji se često koriste za postavljanje uvjeta selekcije (zašto?)
- za atribute prema kojima se obavlja spajanje relacija (zašto?)
  - primarni i alternativni ključevi relacije
  - strani ključevi
- za atribute prema kojima se često obavlja sortiranje ili grupiranje (zašto?)

# Za koje atribute treba kreirati indeks?

- **Primjer:**

```
CREATE TABLE stud (
 mbr INTEGER
, ime NCHAR(20)
, prez NCHAR(20)) ;
```

- Često se postavljaju upiti oblika:

```
SELECT * FROM stud
WHERE mbr = 12345;
```

```
SELECT * FROM stud
WHERE prez > 'Kolar'
ORDER BY prez;
```

⇒ Kreirati indeks za mbr i indeks za prez

- Što se nakon kreiranja navedenih indeksa dešava pri obavljanju:

```
UPDATE stud SET prez = UPPER(prez)
WHERE ime = 'Ivan';
```

- n-torce se pronalaze linearnom pretragom (loše), a zbog izmjene vrijednosti atributa prez mora se izmijeniti sadržaj B-stabla za indeks nad atributom prez (loše)

```
UPDATE stud SET ime = UPPER(ime)
WHERE prez = 'Horvat';
```

- n-torce se pronalaze pomoću B-stabla (dobro), nad atributom ime nije izgrađen indeks, stoga ne postoji B-stablo čiji se sadržaj mora izmijeniti (dobro)

# Za koje atribute ne treba kreirati indeks?

---

- ako vrijednosti atributa imaju relativno mali broj različitih vrijednosti
  - npr. atribut spolOsobe s dozvoljenim vrijednostima M, Ž
- ako relaciji predstoji velik broj upisa, izmjena ili brisanja n-torki. Preporuča se u takvim slučajevima postojeće indekse izbrisati, te ih ponovo izgraditi tek nakon obavljenih promjena nad podacima
- ako relacija sadrži relativno mali broj n-torki (sve n-torke su pohranjene u nekoliko blokova). U takvim slučajevima B-stablo ne pridonosi efikasnosti pretrage
  - npr. relacija zupanija

# Složeni indeksi

```
CREATE TABLE stud (
 mbr INTEGER
, ime NCHAR(20)
, prez NCHAR(20));
```

```
CREATE INDEX stud_ime ON stud (ime);
CREATE INDEX stud_pres ON stud (pres);
```

- efikasno se obavljaju upiti oblika:

```
SELECT * FROM stud
WHERE prez = 'Kolar';
```

```
SELECT * FROM stud
WHERE ime = 'Ivan';
```

- upit oblika:

```
SELECT * FROM stud
WHERE prez = 'Kolar'
AND ime = 'Ivan';
```

- pomoću indeksa nad atributom prez dohvatit će se n-torce studenata čije je prezime 'Horvat', ali će se u dobivenom skupu n-torki linearnom pretragom morati pronaći oni čije je ime 'Ivan' (ili indeksom po imenu, a onda linearno po prezimenu)

# Složeni indeksi

- Prethodni upit se efikasnije obavlja ako se umjesto posebnih indeksa za atribute ime i prezime, kreira složeni indeks:

```
CREATE INDEX stud_pres_ime ON stud (pres, ime);
```

- problem: ovaj indeks se koristi za upite oblika:

```
SELECT * FROM stud
WHERE prez = 'Kolar'
AND ime = 'Ivan';
```

=

```
SELECT * FROM stud
WHERE ime = 'Ivan'
AND prez = 'Kolar';
```

- također i za upite oblika:

```
SELECT * FROM stud
WHERE prez = 'Kolar';
```

- ali se ne može koristiti za upite oblika:

```
SELECT * FROM stud
WHERE ime = 'Ivan';
```

# Složeni indeksi

- Kako upotreba složenih indeksa utječe na sortiranje:

```
CREATE INDEX stud_pres_ime1 ON stud (pres, ime);
```

- indeks osoba\_pres\_ime1 se efikasno koristi za sortiranje oblika:

```
SELECT * FROM stud
ORDER BY prez, ime;
```

```
SELECT * FROM stud
ORDER BY prez DESC, ime DESC;
```

- ali ne i za:

```
SELECT * FROM stud
ORDER BY prez DESC, ime;
```

- ako se kreira indeks:

```
CREATE INDEX stud_pres_ime2 ON stud (pres DESC, ime);
```

- indeks osoba\_pres\_ime2 se efikasno koristi za sortiranje oblika:

```
SELECT * FROM stud
ORDER BY prez DESC, ime;
```

```
SELECT * FROM stud
ORDER BY prez, ime DESC;
```

# Složeni indeksi

---

- Ako je nad relacijom r (ABCD) kreiran složeni indeks r\_abc1

```
CREATE INDEX r_abc1 ON r (A, B, C);
```

tada sljedeće indekse nije potrebno kreirati:

```
CREATE INDEX r_abc2 ON r (A DESC, B DESC, C DESC);
CREATE INDEX r_a1 ON r (A);
CREATE INDEX r_a2 ON r (A DESC);
CREATE INDEX r_ab1 ON r (A, B);
CREATE INDEX r_ab2 ON r (A DESC, B DESC);
```

- Indekse r\_abc2, r\_a1, r\_a2, r\_ab1 i r\_ab2 ne treba kreirati jer SUBP može koristiti indeks r\_abc1 u svim slučajevima u kojima bi se koristili indeksi r\_abc2, r\_a1, r\_a2, r\_ab1 i r\_ab2.

## Zadatak 5. zadana je relacija stud (mbr ime prez pbrStan)

---

- Za relaciju stud kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje (pomoću  $B^+$ -stabla) svih navedenih upita:

1. SELECT \* FROM stud ORDER BY ime DESC, prez;
  2. SELECT \* FROM stud ORDER BY ime DESC, prez DESC;
  3. SELECT \* FROM stud ORDER BY ime, prez, pbrStan;
  4. SELECT \* FROM stud WHERE prez = 'Novak' AND ime = 'Ivo';
  5. SELECT \* FROM stud WHERE pbrStan > 51000 ORDER BY pbrStan DESC;
- 

1. (ime DESC, prez)
2. (ime DESC, prez DESC)
3. (ime, prez, pbrStan) - ali sada više nije potreban indeks pod 2.
4. može se koristiti indeks pod 3.
5. (pbrStan)

**Konačno rješenje** - kreirati indekse za: (ime DESC, prez)

(ime, prez, pbrStan)

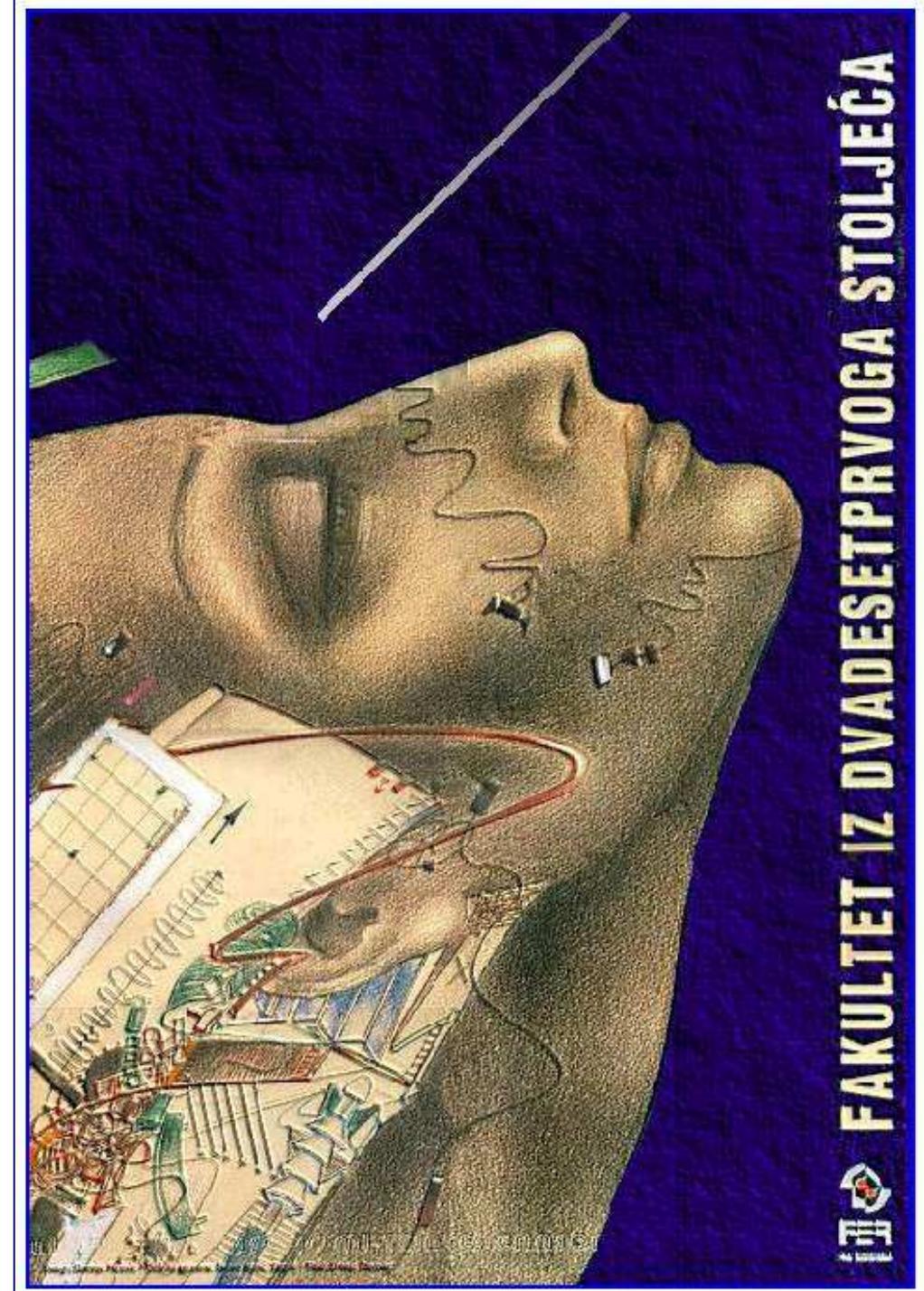
(pbrStan)

SQL naredbe za  
kreiranje indeksa  
napisati za vježbu!

# Baze podataka

Predavanja  
travanj 2014.

## 10. Integritet baze podataka



# UVOD - Integritet baze podataka

---

- Pojam integriteta baze podataka odnosi se na konzistentnost (ispravnost) podataka sadržanih u bazi podataka
- Integritet baze podataka može biti narušen zbog:
  - slučajne pogreške korisnika kod unosa ili izmjene podataka
  - pogreške aplikacijskog programa ili sustava
- **Integritetska ograničenja** osiguravaju da izmjene podataka koje obavljaju korisnici ne narušavaju konzistentnost podataka
  
- O problemu djelovanja neautoriziranih korisnika, diverzije ili sabotaže brine poseban dio SUBP koji je zadužen za **sigurnost baze podataka**

# UVOD - Shema i instanca baze podataka

---

- Shema baze podataka sastoji se od:

- skupa relacijskih shema

$$\mathcal{R} = \{ R_1, R_2, \dots, R_n \}$$

- i skupa integritetskih ograničenja (*integrity constraints*)

$$IC = \{ IC_1, IC_2, \dots, IC_m \}$$

- Instanca baze podataka (stanje baze podataka) definirana na shemi baze podataka  $\mathcal{R} = \{ R_1, R_2, \dots, R_n \}$  je skup instanci relacija (stanja relacija)

$$r = \{ r_1(R_1), r_2(R_2), \dots, r_n(R_n) \}$$

- Ispravna instanca baze podataka je ona instanca koja zadovoljava **sva** definirana integritetska ograničenja

## Primjer: definirati integritetska ograničenja za zadalu bazu podataka

**student**

| mbr  | prez  | ime   | postBrPreb |
|------|-------|-------|------------|
| 1111 | Novak | Ivan  | 10000      |
| 1234 | Kolar | Petar | 31000      |

**ispit**

| mbr  | sifPred | datISP   | ocj | sifNast |
|------|---------|----------|-----|---------|
| 1111 | 1001    | 29.01.05 | 1   | 1111    |
| 1111 | 1001    | 05.02.05 | 1   | 2222    |
| 1111 | 1001    | 01.04.05 | 3   | 1111    |
| 1111 | 1003    | 03.02.05 | 2   | 3333    |
| 1111 | 1002    | 15.06.05 | 4   | 2222    |
| 1234 | 1001    | 29.01.05 | 3   | 2222    |

**predmet**

| sifPred | nazPred | ECTS | sifOrgJed |
|---------|---------|------|-----------|
| 1001    | Mat-1   | 6    | 102       |
| 1002    | Mat-2   | 6    | 102       |
| 1003    | Fiz-1   | 5    | 101       |

**nastavnik**

| sifNast | prezNast | postBr |
|---------|----------|--------|
| 1111    | Pašić    | 10020  |
| 2222    | Brnetić  | 10000  |
| 3333    | Horvat   | 10430  |

**mjesto**

| postBr | nazMjesto |
|--------|-----------|
| 10000  | Zagreb    |
| 10020  | Zagreb    |
| 10430  | Samobor   |
| 21000  | Split     |
| 31000  | Osijek    |

**orgJed**

| sifOrgJed | kratOrgJed | sifNadOrg |
|-----------|------------|-----------|
| 100       | FER        | NULL      |
| 101       | ZPF        | 100       |
| 102       | ZPM        | 100       |
| 103       | ZOEEM      | 100       |
| 104       | GOE        | 103       |
| 105       | GEM        | 103       |

# UVOD - Integritetska ograničenja

---

- **Primjer:** Shema baze podataka PODUZECE
- relacijske sheme:
  - RADNIK = { mbr, prez, ime, pbrStan, datRod, datZap }
    - $K_{RADNIK} = \{ mbr \}$
  - MJESTO = { pbr, nazMjesto }
    - $K_{MJESTO} = \{ pbr \}$
- integritetska ograničenja:
  - vrijednost atributa mbr je iz skupa cijelih brojeva iz intervala [1000, 9999]
  - vrijednost atributa pbr je iz skupa cijelih brojeva iz intervala [10000, 99999]
  - ista vrijednost atributa mbr ne smije se pojaviti u dvije ili više n-torki relacije radnik(RADNIK) - vrijednost atributa mbr je jedinstvena
  - vrijednost atributa mbr ne smije poprimiti NULL vrijednost
  - razlika između datZap (datum zaposlenja) i datRod(datum rođenja) ne smije biti manja od 16 godina niti veća od 65 godina
  - itd.

# UVOD - Integritetska ograničenja

---

- definicije integritetskih ograničenja su sastavni dio sheme baze podataka
- definicije integritetskih ograničenja se pohranjuju u **rječnik podataka** baze podataka
  - na taj način pravila definirana integritetskim ograničenjima postaju nezaobilazna za svakog korisnika sustava
  - SUBP provjerava integritetska ograničenja pri obavljanju svake operacije koja mijenja sadržaj baze podataka
    - u trenutku **završetka** operacije nad podacima, baza podataka mora biti u stanju u kojem su zadovoljena **sva** integritetska ograničenja
    - SUBP **odbija** obaviti operacije koje nemaju to svojstvo ili obavlja kompenzacijске akcije koje osiguravaju da su u konačnici sva integritetska ograničenja zadovoljena

# (Rječnik podataka)

---

- *Data dictionary, Catalogue, Repository*
- Opisi podataka (metapodaci) su pohranjeni u rječnik podataka. Prikazani su na isti način i može im se pristupiti na isti način kao i "običnim" podacima.
  - korisnici s pravom pristupa nad rječnikom podataka mogu primijeniti relacijski upitni jezik (npr. SQL)
- Rječnik podataka sadrži:
  - opis relacijskih shema
  - opis pravila integriteta
  - opis pravila pristupa - korisnik-objekt-dozvoljena akcija
  - opis pohranjenih procedura, poslovnih pravila
  - opis okidača (*triggers*)
  - ...

# (Rječnik podataka) - primjer

- baza podataka u IBM Informix SUBP sadrži nekoliko desetaka "sistemske" relacija koje čine rječnik podataka. Te relacije se kreiraju automatski, prilikom kreiranja baze podataka
- npr, u relacijama *systables* i *syscolumns* pohranjeni su metapodaci o relacijama i atributima

```
SELECT *
 FROM systables, syscolumns
 WHERE systables.tabid = syscolumns.tabid
 AND tabname = 'mjesto'
 ORDER BY colno;
```

| systables |         |       |       |       |            |     | syscolumns  |       |       |         |           |     |  |
|-----------|---------|-------|-------|-------|------------|-----|-------------|-------|-------|---------|-----------|-----|--|
| tabname   | owner   | tabid | ncols | nrows | created    | ... | colname     | tabid | colno | coltype | collength | ... |  |
| mjesto    | bpadmin | 101   | 3     | 275   | 15.02.2012 | ... | pbr         | 101   | 1     | 258     | 4         | ... |  |
| mjesto    | bpadmin | 101   | 3     | 275   | 15.02.2012 | ... | nazmjesto   | 101   | 2     | 271     | 40        | ... |  |
| mjesto    | bpadmin | 101   | 3     | 275   | 15.02.2012 | ... | sifzupanija | 101   | 3     | 1       | 2         | ... |  |

# Integritetska ograničenja

---

- Entitetski integritet (*Entity integrity*)
- Integritet ključa (*Key integrity*)
- Domenski integritet (*Domain integrity*)
- Ograničenja NULL vrijednosti (*Constraints on NULL*)
- Referencijski integritet (*Referential integrity*)
- Opća integritetska ograničenja (*General integrity constraints*)

# Entitetski integritet

---

- Niti jedan atribut **primarnog ključa** ne smije poprimiti NULL vrijednost
- Primjer:

$NASTAVNIK = \{ \text{sifNast}, \text{jmbgNast}, \text{prezNast} \}$

$\text{PK}_{NASTAVNIK} = \{ \text{sifNast} \}$        $\text{K2}_{NASTAVNIK} = \{ \text{jmbgNast} \}$

Primarni ključ označen je s PK

$ISPIT = \{ \text{mbrStud}, \text{sifPred}, \text{datlsp}, \text{ocj}, \text{sifNas} \}$

$\text{PK}_{ISPIT} = \{ \text{mbrStud}, \text{sifPred}, \text{datlsp} \}$

⇒ atribut *sifNast* ne smije poprimiti NULL vrijednost niti u jednoj n-torci relacije *nastavnik(NASTAVNIK)*

⇒ atributi *mbrStud*, *sifPred*, *datlsp* ne smiju poprimiti NULL vrijednost niti u jednoj n-torci relacije *ispit(ISPIT)*

# Integritet ključa

---

- U relaciji ne smiju postojati dvije n-torce s jednakim vrijednostima ključa (vrijedi za **sve moguće** ključeve)
- Primjer:

$NASTAVNIK = \{ \text{sifNast}, \text{jmbgNast}, \text{prezNast} \}$

$$PK_{NASTAVNIK} = \{ \text{sifNast} \} \quad K2_{NASTAVNIK} = \{ \text{jmbgNast} \}$$

$ISPIT = \{ \text{mbrStud}, \text{sifPred}, \text{datlsp}, \text{ocj}, \text{sifNas} \}$

$$PK_{ISPIT} = \{ \text{mbrStud}, \text{sifPred}, \text{datlsp} \}$$

⇒ u relaciji *nastavnik(NASTAVNIK)* ne smiju postojati dvije n-torce koje imaju jednake vrijednosti atributa *sifNast*

⇒ u relaciji *nastavnik(NASTAVNIK)* ne smiju postojati dvije n-torce koje imaju jednake vrijednosti atributa *jmbgNast*

⇒ u relaciji *ispit(ISPIT)* ne smiju postojati dvije n-torce koje imaju jednake vrijednosti (istu kombinaciju vrijednosti) atributa *mbrStud*, *sifPred* i *datlsp*

# Domenski integritet

---

- Atribut može poprimiti samo jednu vrijednost iz domene atributa
- Primjer:  $MJESTO = \{ pbr, nazMjesto \}$   
 $PK_{MJESTO} = \{ pbr \}$
- domena atributa *pbr* je skup cijelih brojeva iz intervala [10000, 99999]

⇒ vrijednost atributa *pbr* u svakoj n-tortki relacije *mjesto(MJESTO)* mora biti cijeli broj iz intervala [10000, 99999]

# Ograničenja NULL vrijednosti

---

- Za određene atributе se može definirati ograničenje prema kojem vrijednost atributa ne smije poprimiti NULL vrijednost
- Primjer:  $\text{STUDENT} = \{ \text{mbrStud}, \text{imeStud}, \text{ prezStud}, \text{adresa} \}$   
 $\text{PK}_{\text{STUDENT}} = \{ \text{mbrStud} \}$   
⇒ vrijednost atributa *imeStud* ne smiju poprimiti NULL vrijednosti niti u jednoj n-torci relacije *student(STUDENT)*  
⇒ vrijednost atributa *prezStud* ne smiju poprimiti NULL vrijednosti niti u jednoj n-torci relacije *student(STUDENT)*

atribut *mbrStud* ?

# Strani ključ (*Foreign key*) i referencijski integritet

- Referencijski integritet se odnosi na konzistentnost među n-torkama dviju relacija (ili n-torkama iste relacije). Neformalno: n-torka iz jedne relacije koja se poziva (referencira) na drugu relaciju se može pozivati (referencirati) samo na postojeće n-torke (primarne ključeve) u toj relaciji
- Primjer:       $\text{OSOBA} = \{ \text{mbr}, \text{prez}, \text{pbrStan} \}$   
 $\text{PK}_{\text{OSOBA}} = \{ \text{mbr} \}$

$\text{MJESTO} = \{ \text{pbr}, \text{nazMjesto} \}$

$\text{PK}_{\text{MJESTO}} = \{ \text{pbr} \}$

| osoba(OSOBA) | mbr    | prez  | pbrStan |
|--------------|--------|-------|---------|
| 100          | Horvat | 10000 |         |
| 107          | Kolar  | 10000 |         |
| 109          | Novak  | 51000 |         |

| mjesto(MJESTO) | pbr    | nazMjesto |
|----------------|--------|-----------|
| 10000          | Zagreb |           |
| 51000          | Rijeka |           |

- Skup atributa { pbrStan } je **strani ključ** u relaciji osoba koji se poziva (referencira) na relaciju *mjesto*. Relacija *osoba* je **pozivajuća**, a relacija *mjesto* je **pozivana** relacija

# Strani ključ (*Foreign key*) i referencijski integritet

---

- Zadane su relacije  $r(R)$  s primarnim ključem  $PK_R$  i  $s(S)$  s primarnim ključem  $PK_S$ . Skup atributa  $FK$ ,  $FK \subseteq R$ , je **strani ključ** u relaciji  $r(R)$  koji se poziva na relaciju  $s(S)$  ukoliko vrijedi:
  - atributi u skupu  $FK$  imaju domene jednake domenama korespondentnih atributa u skupu  $PK_S$
  - za svaku n-torku  $t_1 \in r(R)$ 
    - postoji n-torka  $t_2 \in s(S)$  takva da je  $t_2[PK_S] = t_1[FK]$
    - ili
    - vrijednost barem jednog atributa iz  $t_1[FK]$  je NULL
- Relacija  $r(R)$  se naziva pozivajuća, a relacija  $s(S)$  se naziva pozivana relacija
  - (relacije  $r(R)$  i  $s(S)$  ne moraju nužno biti različite relacije)
- Referencijski integritet se odnosi na ograničenje koje proizlazi iz definicije stranog ključa

# Strani ključ (*Foreign key*) i referencijski integritet

- Primjer:  $\text{OSOBA} = \{ \text{mbr}, \text{prez}, \text{pbrStan} \}$

$$\text{PK}_{\text{OSOBA}} = \{ \text{mbr} \}$$

$$\text{MJESTO} = \{ \text{pbr}, \text{nazMjesto} \}$$

$$\text{PK}_{\text{MJESTO}} = \{ \text{pbr} \}$$

osoba(OSOBA)

| mbr | prez   | pbrStan |
|-----|--------|---------|
| 100 | Horvat | 10000   |
| 107 | Kolar  | NULL    |
| 109 | Novak  | 31000   |

mjesto(MJESTO)

| pbr   | nazMjesto |
|-------|-----------|
| 10000 | Zagreb    |
| 51000 | Rijeka    |

- Skup atributa { pbrStan } je strani ključ u relaciji osoba koji se poziva na relaciju mjesto
- Relacije osoba i mjesto ne zadovoljavaju pravilo referencijskog integriteta jer u relaciji osoba postoji n-torka  $t_1 = <109, \text{Novak}, 31000>$  za koju u relaciji mjesto ne postoji odgovarajuća n-torka  $t_2$  (s vrijednošću atributa pbr jednakoj 31000)
  - n-torka  $<107, \text{Kolar}, \text{NULL}>$  ne narušava referencijski integritet!

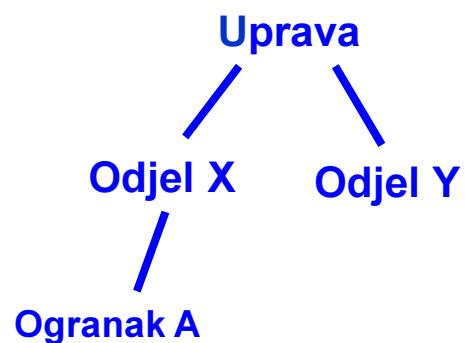
# Strani ključ (*Foreign key*) i referencijski integritet

|                | osoba(OSOBA) |        |        |       | mjesto(MJESTO) |            |              | drzava(DRZAVA) |               |
|----------------|--------------|--------|--------|-------|----------------|------------|--------------|----------------|---------------|
|                | <u>mbr</u>   | prez   | oznDrz | pbr   | <u>oznDrz</u>  | <u>pbr</u> | <u>nazMj</u> | <u>oznDrz</u>  | <u>nazDrz</u> |
| t <sub>1</sub> | 101          | Horvat | HR     | 10000 | HR             | 10000      | Zagreb       | HR             | Hrvatska      |
| t <sub>2</sub> | 102          | Jones  | GB     | 51000 | GB             | 51000      | Leeds        | GB             | V. Britanija  |
| t <sub>3</sub> | 103          | Kolar  | NULL   | NULL  | HR             | 51000      | Rijeka       |                |               |
| t <sub>4</sub> | 104          | Smith  | GB     | NULL  | GB             | 10000      | Bristol      |                |               |
| t <sub>5</sub> | 105          | Novak  | NULL   | 47000 |                |            |              |                |               |
| t <sub>6</sub> | 106          | Clark  | USA    | NULL  |                |            |              |                |               |
| t <sub>7</sub> | 107          | Adams  | GB     | 47000 |                |            |              |                |               |
| t <sub>8</sub> | 108          | Wilson | USA    | 10000 |                |            |              |                |               |

- $\text{FK}_1 = \{ \text{oznDrz}, \text{pbr} \}$  je strani ključ u relaciji *osoba* koji se poziva na relaciju *mjesto*
- $\text{FK}_2 = \{ \text{oznDrz} \}$  je strani ključ u relaciji *osoba* koji se poziva na relaciju *drzava*
- $\text{FK}_3 = \{ \text{oznDrz} \}$  je strani ključ u relaciji *mjesto* koji se poziva na relaciju *drzava*
  
- n-torce  $t_1, t_2, t_3, t_4, t_5$  **zadovoljavaju** pravila referencijskog integriteta za  $\text{FK}_1$  i  $\text{FK}_2$
- n-torka  $t_6$  ne zadovoljava pravilo referencijskog integriteta za  $\text{FK}_2$
- n-torka  $t_7$  ne zadovoljava pravilo referencijskog integriteta za  $\text{FK}_1$
- n-torka  $t_8$  ne zadovoljava pravila referencijskog integriteta za  $\text{FK}_1$  i  $\text{FK}_2$

# Strani ključ (*Foreign key*) i referencijski integritet

- Primjer:  $\text{ORGJED} = \{ \text{sifOrgjed}, \text{nazOrgjed}, \text{sifNadOrgjed} \}$   
 $\text{PK}_{\text{ORGJED}} = \{ \text{sifOrgjed} \}$



$\text{orgjed(ORGJED)}$

|   | $\text{sifOrgjed}$ | $\text{nazOrgjed}$ | $\text{sifNadOrgjed}$ |
|---|--------------------|--------------------|-----------------------|
| 1 | Uprava             |                    | NULL                  |
| 2 | Odjel X            |                    | 1                     |
| 3 | Odjel Y            |                    | 1                     |
| 4 | Odjel Z            |                    | 6                     |
| 5 | Ogranak A          |                    | 2                     |

- Skup atributa  $\{ \text{sifNadOrgjed} \}$  je strani ključ u relaciji  $\text{orgjed}$  koji se poziva na relaciju  $\text{orgjed}$
- Relacija  $\text{orgjed(ORGJED)}$  **ne zadovoljava** pravilo referencijskog integriteta jer u relaciji postoji n-torka  $t_1 = <4, \text{Odjel Z}, 6>$  za koju u istoj relaciji ne postoji odgovarajuća n-torka  $t_2$  (s vrijednošću atributa  $\text{sifOrgjed}$  jednakoj 6)
  - n-torka  $<1, \text{Uprava}, \text{NULL}>$  **ne** narušava referencijski integritet!

# Strani ključ (*Foreign key*) i referencijski integritet

---

- Postoje slučajevi u kojima strani ključ iz  $r(R)$  ne smije biti NULL.  
To vrijedi za slučaj kad se pravila referencijskog integriteta primjenjuju na atribute za koje su definirana neka druga pravila integriteta (npr. pravilo entitetskog integriteta, ograničenje NULL vrijednosti)
  - strani ključ relacije  $r(R)$  koji je ujedno **dio primarnog ključa** relacije  $r(R)$  ne smije poprimiti NULL vrijednost!
- Primjer:  
 $STUD = \{ mbrStud, imeStud, prezStud \}$   
 $PK_{STUD} = \{ mbrStud \}$   
 $ISPIT = \{ mbrStud, sifPred, datIsp, ocj, sifNast \}$   
 $PK_{ISPIT} = \{ mbrStud, sifPred, datIsp \}$

Skup atributa  $\{ mbrStud \}$  je strani ključ u relaciji *ispit*( $ISPIT$ ) koji se poziva na relaciju *stud*( $STUD$ ), ali je ujedno dio prim. ključa u relaciji *ispit*  
⇒ atribut *mbrStud* u relaciji *ispit* ne smije poprimiti NULL vrijednost!

# Opća integritetska ograničenja

---

- Opća integritetska ograničenja su ograničenja općeg (generalnog) oblika
  - npr. poslovna pravila
- Primjer: ograničenje odnosa među vrijednostima atributa

$\text{RADNIK} = \{ \text{mbr}, \text{prez}, \text{ime}, \text{pbrStan}, \text{datRod}, \text{datZap} \}$   
 $\text{PK}_{\text{RADNIK}} = \{ \text{mbr} \}$

⇒ razlika između vrijednosti atributa *datZap* (datum zaposlenja) i *datRod* (datum rođenja) ne smije biti manja od 16 godina niti veća od 65 godina. To integritetsko ograničenje proizlazi iz (za ovaj primjer izmišljenog) zakonskog ograničenja da se osobe mlađe od 16 godina ili starije od 65 godina ne smiju zapošljavati.

# Implementacija integritetskih ograničenja u SQL-u

---

- integritetska ograničenja se mogu definirati
  - u okviru naredbe za kreiranje relacije ili
  - naknadnim definiranjem pomoću naredbe ALTER TABLE

# Integritetska ograničenja - za atribut

---

```
CREATE TABLE tableName
({columnName dataType [DEFAULT defaultExpr]
 [columnConstraint] [, ...] } | tableConstraint [, ...])
```

## Column constraints:

```
[CONSTRAINT constraintName]
{ NOT NULL | UNIQUE | PRIMARY KEY |
CHECK (condition) |
REFERENCES reftable [(refcolumn)] [ON DELETE action]
[ON UPDATE action] }
```

action: NO ACTION, CASCADE, SET NULL, SET DEFAULT  
condition - odnosi se samo na dotičnu kolonu

# Integritetska ograničenja - za tablicu

---

```
CREATE TABLE tableName
({ columnName dataType [DEFAULT defaultExpr]
 [columnConstraint] [, ...] } | tableConstraint [, ...])
```

## Table constraints:

```
[CONSTRAINT constraintName]
{ UNIQUE (columnName [, ...]) |
 PRIMARY KEY (columnName [, ...]) |
 CHECK (condition) |
 FOREIGN KEY (columnName [, ...]) REFERENCES
 reftable [(refcolumn [, ...])] [ON DELETE action]
 [ON UPDATE action] }
```

# SQL: PRIMARY KEY

- Primjer: ISPIT = { mbrStud, sifPred, datIsp, ocj, sifNast }

$$PK_{ISPIT} = \{ mbrStud, sifPred, datIsp \}$$

```
CREATE TABLE ispit (
 mbrStud INTEGER
, sifPred INTEGER
, datIsp DATE
, ocj SMALLINT
, sifNast INTEGER
, PRIMARY KEY (
 mbrStud, sifPred, datIsp)
);
```

- SUBP osigurava:  
entitetski integritet i  
integritet ključa

vidjeti sintaksne  
dijagrame 5, 5.1 i 5.5

- entitetski integritet i integritet ključa za primarni ključ se uvijek osigurava pomoću PRIMARY KEY

# Definiranje int. ograničenja pri definiciji atributa

- U slučajevima kad se integritetsko ograničenje odnosi na samo jedan atribut, može se definirati neposredno uz definiciju atributa (to vrijedi za sve vrste ograničenja)
- Primjer:

NASTAVNIK (sifNast, jmbgNast, prezNast)      nastavnik(NASTAVNIK)

$\text{PK}_{\text{NASTAVNIK}} = \text{sifNast}$        $\text{K2}_{\text{NASTAVNIK}} = \text{jmbgNast}$

```
CREATE TABLE nastavnik (
 sifNast INTEGER PRIMARY KEY
, jmbgNast CHAR(13)
, prezNast CHAR(40)
);
```

vidjeti sintaksne  
dijagrame 5, 5.1,  
5.2 i 5.4

# SQL: *UNIQUE*

---

- Primjer:

NASTAVNIK (sifNast, jmbgNast, prezNast)    nastavnik(NASTAVNIK)

$\text{PK}_{\text{NASTAVNIK}} = \text{sifNast}$      $\text{K2}_{\text{NASTAVNIK}} = \text{jmbgNast}$

```
CREATE TABLE nastavnik (
 sifNast INTEGER
, jmbgNast CHAR(13)
, prezNast CHAR(40)
, PRIMARY KEY (sifNast)
, UNIQUE (jmbgNast)
);
```

- SUBP osigurava:  
**integritet ključa**

vidjeti sintaksne  
dijagrame 5, 5.1 i 5.5

```
CREATE TABLE nastavnik (
 sifNast INTEGER PRIMARY KEY
, jmbgNast CHAR(13) UNIQUE
, prezNast CHAR(40)
...);
```

vidjeti sintaksne dijagrame  
5, 5.1, 5.2 i 5.4

# SQL: CHECK

- Domenski integritet je djelomično osiguran samom definicijom tipa podatka za atribut
  - npr. definiranjem podatka tipa SMALLINT određena je njegova domena kao skup cijelih brojeva u intervalu -32767 do 32767
- Moguće je postići točnije određenje domene atributa:

```
CREATE TABLE ispit (
 mbrStud INTEGER
, sifPred INTEGER
, datIsp DATE
, ocj SMALLINT
, sifNast INTEGER
, PRIMARY KEY (mbrStud, sifPred, datIsp)
, CHECK (ocj BETWEEN 1 AND 5)
);
```

- SUBP osigurava:  
**domenski integritet**

vidjeti sintaksne dijagrame  
5, 5.1, 5.5 i 5.7

```
CREATE TABLE ispit (
 ...
, ocj SMALLINT CHECK (ocj BETWEEN 1 AND 5)
, sifNast INTEGER
);
```

vidjeti sintaksne dijagrame 5, 5.1, 5.2, 5.4 i 5.7

# SQL: CHECK

- Također se može koristiti za definiranje ograničenja odnosa među vrijednostima atributa u istoj n-torci (vidjeti primjer za opće pravilo integriteta)

**Primjer:** razlika između vrijednosti atributa *datZap* (datum zaposlenja) i *datRod* (datum rođenja) ne smije biti manja od 16 godina niti veća od 65 godina.

```
CREATE TABLE radnik (
 mbr INTEGER
, ime CHAR (40)
, prez CHAR (40)
, datRod DATE
, datZap DATE
, CHECK (datZap - datRod >= 16*365
 AND datZap - datRod <= 65*365)
) ;
```

U rješenju je zanemareno  
da prestupne godine  
broje po 366 dana

- Ograničenje koje se tiče odnosa među vrijednostima atributa se ne može napisati neposredno uz definiciju atributa

# SQL: NOT NULL

---

- Ograničenje NULL vrijednosti se postiže navođenjem rezerviranih riječi *NOT NULL* iza tipa podatka pri definiciji atributa
- Primjer:     ORGJED = { sifOrgjed, nazOrgjed, sifNadOrgjed }  
                  PK<sub>ORGJED</sub> = { sifOrgjed }

```
CREATE TABLE orgjed (
 sifOrgjed INTEGER
, nazOrgjed CHAR(40) NOT NULL
, sifNadOrgjed INTEGER
, PRIMARY KEY (sifOrgjed)
, FOREIGN KEY (sifNadOrgjed)
 REFERENCES orgjed (sifOrgjed)
);
```

- Hoće li SUBP dopustiti da vrijednost atributa sifOrgjed bude NULL?
- Hoće li SUBP dopustiti da vrijednost atributa sifNadOrgjed bude NULL?

# SQL: FOREIGN KEY

- Primjer:
  - primarni ključ u relaciji *student* je { *mbrStud* }
  - primarni ključ u relaciji *predmet* je { *sifPred* }
  - primarni ključ u relaciji *nastavnik* je { *sifNast* }

```
CREATE TABLE ispit (
 mbrStud INTEGER,
 sifPred INTEGER,
 datIsp DATE,
 ocj SMALLINT,
 sifNast INTEGER,
 PRIMARY KEY (mbrStud, sifPred, datIsp),
 FOREIGN KEY (mbrStud) REFERENCES student(mbrStud),
 FOREIGN KEY (sifPred) REFERENCES predmet(sifPred),
 FOREIGN KEY (sifNast) REFERENCES nastavnik(sifNast)
);
```

vidjeti sintaksne dijagrame  
5, 5.1, 5.5 i 5.6

SUBP osigurava ref. integritet: strani ključ u relaciji *ispit* (skup atributa { *sifNast* }) poziva se na primarni ključ u relaciji *nastavnik* (skup atributa { *sifNast* })

Podrazumijeva se da su u relacijama *student*, *predmet* i *nastavnik* pomoću PRIMARY KEY definirana ograničenja (ent. integritet i integritet ključa)

# SQL: FOREIGN KEY

```
CREATE TABLE ispiti (
 mbrStud INTEGER REFERENCES student(mbrStud)
 , sifPred INTEGER REFERENCES predmet(sifPred)
 , datIsp DATE
 , ocj SMALLINT
 , sifNast INTEGER REFERENCES nastavnik(sifNast)
 , PRIMARY KEY (mbrStud, sifPred, datIsp)
);
```

vidjeti sintaksne dijagrame  
5, 5.1, 5.2, 5.4 i 5.6

# SQL: FOREIGN KEY

| student | mbr  | prez  | ime   |
|---------|------|-------|-------|
|         | 1111 | Novak | Ivan  |
|         | 1234 | Kolar | Petar |

| predmet | sifPred | nazPred |
|---------|---------|---------|
|         | 1001    | Mat-1   |
|         | 1002    | Mat-2   |
|         | 1003    | Fiz-1   |

| nastavnik |          |
|-----------|----------|
| sifNast   | prezNast |
| 1111      | Pašić    |
| 2222      | Brnetić  |
| 3333      | Horvat   |

| ispit | mbr  | sifPred | datIsp   | ocj | sifNast |
|-------|------|---------|----------|-----|---------|
|       | 1111 | 1001    | 29.01.12 | 1   | 1111    |
|       | 1111 | 1001    | 05.02.11 | 1   | 2222    |
|       | 1111 | 1001    | 01.04.11 | 3   | 1111    |
|       | 1111 | 1003    | 03.02.12 | 2   | 3333    |
|       | 1111 | 1002    | 15.06.11 | 4   | 2222    |
|       | 1234 | 1001    | 29.01.12 | 3   | 2222    |

## Operacije koje bi narušile referencijski integritet:

- unos ispita za nepostojećeg studenta
- unos ispita iz nepostojećeg predmeta
- unos ispita kod nepostojećeg nastavnika
- izmjene u tablici ispit:
  - mbr se mijenja na neku vrijednost koja ne postoji u tablici student
  - sifPred se mijenja na neku vrijednost koja ne postoji u tablici predmet
  - sifNast se mijenja na neku vrijednost koja ne postoji u tablici nastavnik
- **SUBP ODBIJA OBAVITI OVE AKCIJE !!!**

Odgovara li nam to? DA

# SQL: FOREIGN KEY

| student | mbr  | prez  | ime   |
|---------|------|-------|-------|
|         | 1111 | Novak | Ivan  |
|         | 1234 | Kolar | Petar |

| nastavnik |          |
|-----------|----------|
| sifNast   | prezNast |
| 1111      | Pašić    |
| 2222      | Brnetić  |
| 3333      | Horvat   |

| ispit |         |          |     |         |
|-------|---------|----------|-----|---------|
| mbr   | sifPred | datlsp   | ocj | sifNast |
| 1111  | 1001    | 29.01.12 | 1   | 1111    |
| 1111  | 1001    | 05.02.11 | 1   | 2222    |
| 1111  | 1001    | 01.04.11 | 3   | 1111    |
| 1111  | 1003    | 03.02.12 | 2   | 3333    |
| 1111  | 1002    | 15.06.11 | 4   | 2222    |
| 1234  | 1001    | 29.01.12 | 3   | 2222    |

## Operacije koje bi također narušile referencijski integritet:

- brisanje podataka o studentu koji se ispisao s fakulteta (npr. 1111 Novak Ivan) iz tablice student
- brisanje podataka o predmetu koji više ne postoji u novom nastavnom programu
- brisanje nastavnika koji je otišao u mirovinu
- SUBP ODBIJA OBAVITI I OVE AKCIJE !!!**

Primjedbe?

Željeli bismo arhivirati podatke o studentima/nastavnicima  
koji su napustili fakultet i izbrisati ih iz aktualne baze podataka !!!

# SQL: FOREIGN KEY

---

- pri definiciji ograničenja referencijskog integriteta moguće je specificirati da li će SUBP pri pokušaju narušavanja ograničenja **brisanjem pozivane n-torke**:
  - odbiti operaciju brisanja pozivane n-torke
    - ON DELETE NO ACTION
  - obaviti operaciju brisanja pozivane n-torke, ali pri tome obaviti i kompenzacijске akcije koje će rezultirati time da integritetsko ograničenje u konačnici bude zadovoljeno. Moguće akcije su:
    - vrijednosti stranog ključa u n-torkama koje se pozivaju na obrisanu n-torku postaviti na NULL vrijednosti
      - ON DELETE SET NULL
    - vrijednosti stranog ključa u n-torkama koje se pozivaju na obrisanu n-torku postaviti na *default* vrijednosti
      - ON DELETE SET DEFAULT
    - obrisati pozivajuće n-torke
      - ON DELETE CASCADE

# SQL: FOREIGN KEY

---

- pri definiciji ograničenja referencijskog integriteta također je moguće specificirati da li će SUBP pri pokušaju narušavanja ograničenja **izmjenom primarnog ključa u pozivanoj n-torci**:
  - odbiti operaciju izmjene pozivane n-torke
    - ON UPDATE NO ACTION
  - obaviti operaciju izmjene pozivane n-torke, ali pri tome obaviti i kompenzacijске akcije koje će rezultirati time da integritetsko ograničenje u konačnici bude zadovoljeno. Moguće akcije su:
    - vrijednosti stranog ključa u n-torkama koje se pozivaju na izmijenjenu n-torku postaviti na NULL vrijednosti
      - ON UPDATE SET NULL
    - vrijednosti stranog ključa u n-torkama koje se pozivaju na izmijenjenu n-torku postaviti na *default* vrijednosti
      - ON UPDATE SET DEFAULT
    - vrijednosti stranog ključa u n-torkama koje se pozivaju na izmijenjenu n-torku postaviti na novu vrijednost primarnog ključa pozivane n-torke
      - ON UPDATE CASCADE

# SQL: FOREIGN KEY

- Različite reakcije na pokušaj narušavanja referencijskog integriteta brisanjem pozivanih n-torki:
  - odbijanje operacije (za strani ključ sifPred)
  - obavljanje kompenzacijskih akcija
    - uz kaskadno brisanje (za strani ključ mbr)
    - uz postavljanje na NULL vrijednosti (za strani ključ sifNast)

```
CREATE TABLE ispit (
 mbr INTEGER
, sifPred INTEGER
, datIsp DATE
, ocj SMALLINT
, sifNast INTEGER
, PRIMARY KEY (mbr, sifPred, datIsp)
, FOREIGN KEY (mbr) REFERENCES student (mbr)
 ON DELETE CASCADE
, FOREIGN KEY (sifPred) REFERENCES predmet (sifPred)
, FOREIGN KEY (sifNast) REFERENCES nastavnik (sifNast)
 ON DELETE SET NULL
);
```

# Ref. integritet definiran uz odbijanje operacije

Ukoliko se pokušaju obrisati n-torce iz tablice predmet na čije se šifre predmeta pozivaju n-torce iz tablice ispit

```
CREATE TABLE ispit (
 mbr INTEGER
, sifPred INTEGER
, datIsp DATE
, ocj SMALLINT
, sifNast INTEGER
, PRIMARY KEY (mbr, sifPred, datIsp)
, FOREIGN KEY (mbr) REFERENCES student (mbr)
 ON DELETE CASCADE
, FOREIGN KEY (sifPred) REFERENCES predmet (sifPred)
, FOREIGN KEY (sifNast) REFERENCES nastavnik (sifNast)
 ON DELETE SET NULL
);
```

Operacija brisanja n-torki iz tablice  
predmet će biti odbijena - korisnik ili  
aplikacija će dobiti poruku o pogrešci

## Ref. integritet definiran uz kaskadno brisanje

Ukoliko se pokušaju obrisati n-torke iz tablice student na čije se matične brojeve pozivaju n-torke iz tablice ispit

```
CREATE TABLE ispit (
 mbr INTEGER
, sifPred INTEGER
, datIsp DATE
, ocj SMALLINT
, sifNast INTEGER
, PRIMARY KEY (mbr, sifPred, datIsp)
, FOREIGN KEY (mbr) REFERENCES student (mbr)
 ON DELETE CASCADE
, FOREIGN KEY (sifPred) REFERENCES predmet (sifPred)
, FOREIGN KEY (sifNast) REFERENCES nastavnik (sifNast)
 ON DELETE SET NULL
);
```

Obrisat će se n-torke iz tablice student i sve n-torke iz tablice ispit koje se pozivaju na obrisane n-torke iz tablice student



## Ref. int. definiran uz postavljanje na NULL vrijednosti

Ukoliko se pokušaju obrisati n-torce iz tablice nastavnik na čije se šifre nastavnika pozivaju n-torce iz tablice ispit

```
CREATE TABLE ispit (
 mbr INTEGER
, sifPred INTEGER
, datIsp DATE
, ocj SMALLINT
, sifNast INTEGER
, PRIMARY KEY (mbr, sifPred, datIsp)
, FOREIGN KEY (mbr) REFERENCES student (mbr)
 ON DELETE CASCADE
, FOREIGN KEY (sifPred) REFERENCES predmet (sifPred)
, FOREIGN KEY (sifNast) REFERENCES nastavnik (sifNast)
 ON DELETE SET NULL
);
```

Obrisat će se n-torce iz tablice nastavnik,  
a vrijednosti stranog ključa (sifNast) u  
tablici ispit koje se pozivaju na obrisane  
n-torce će se postaviti na NULL



# SQL: Imenovanje integritetskih ograničenja

- naziv integritetskog ograničenja (**CONSTRAINT constraint**) se navodi opcionalno: ako se navede, korisnik (ili aplikacija) će pri pokušaju obavljanja naredbe koja narušava integritetsko ograničenje dobiti informaciju o kojem se točno integritetskom ograničenju radi

```
CREATE TABLE ispit (
 mbrStud INTEGER
, sifPred INTEGER
, datIsp DATE
, ocj SMALLINT
 NOT NULL CONSTRAINT ocjNotNull
 CHECK (ocj BETWEEN 1 AND 5) CONSTRAINT chkOcj
, sifNast INTEGER NOT NULL
, PRIMARY KEY(mbrStud, sifPred, datIsp) CONSTRAINT pkIspit
, FOREIGN KEY(mbrStud) REFERENCES stud(mbrStud) CONSTRAINT fkIspitStud
, FOREIGN KEY(sifPred) REFERENCES pred(sifPred) CONSTRAINT fkIspitPred
, FOREIGN KEY(sifNast) REFERENCES nast(sifNast) CONSTRAINT fkIspitNast
);
```

- Primjer uklanjanja definiranog integritetskog ograničenja:

```
ALTER TABLE ispit DROP CONSTRAINT ocjNotNull;
```

# SQL: Napomene

---

- **Isključivo** u onim slučajevima kada SUBP ne podržava mogućnost definiranja ograničenja tipa *PRIMARY KEY* i *UNIQUE*
  - entitetski integritet se može osigurati specificiranjem ograničenja *NOT NULL* uz atribute primarnog ključa
  - integritet ključa se može osigurati kreiranjem indeksa (*UNIQUE INDEX*) nad ključem
    - po jedan takav indeks se kreira za svaki mogući ključ (ne svaki atribut ključa)
- Većina današnjih sustava za upravljanje bazama podataka podržava mogućnost definiranja tih tipova ograničenja

# SQL: Napomene

---

- Većina sustava za upravljanje bazama podataka automatski kreira *UNIQUE* indekse pri definiranju sljedećih ograničenja:
  - PRIMARY KEY (a, b, c)
    - SUBP automatski kreira UNIQUE INDEX za (a, b, c)
  - UNIQUE (a, b, c)
    - SUBP automatski kreira UNIQUE INDEX za (a, b, c)
- Neki sustavi (npr. IBM Informix) također automatski kreiraju indekse pri definiranju ograničenja referencijskog integriteta:
  - FOREIGN KEY (a, b, c) REFERENCES *reftable* (e, f, g)
    - SUBP automatski kreira INDEX za (a, b, c)

# SQL: Napomene

---

- IBM Informix: definicija referencijskog integriteta
  - **nisu podržane** opcije:
    - ON UPDATE SET NULL
    - ON UPDATE SET DEFAULT
    - ON UPDATE CASCADE
    - ON DELETE SET NULL
    - ON DELETE SET DEFAULT
  - **podržane su** opcije:
    - ON DELETE CASCADE
    - ON DELETE NO ACTION se podrazumijeva u slučaju kad nije navedena opcija ON DELETE CASCADE
    - ON UPDATE NO ACTION se podrazumijeva

# Kreirati integritetska ograničenja u bazi *studadmin*

---

```
ALTER TABLE mjesto
 ADD CONSTRAINT PRIMARY KEY (pbr) CONSTRAINT mjestoPK;

ALTER TABLE student
 ADD CONSTRAINT PRIMARY KEY (jmbag) CONSTRAINT studentPK,
 ADD CONSTRAINT CHECK (spol IN ('M', 'Ž')) CONSTRAINT studentCHKspol,
 ADD CONSTRAINT FOREIGN KEY (pbrrodstudent)
 REFERENCES mjesto(pbr) CONSTRAINT studentFKpbrrod,
 ADD CONSTRAINT FOREIGN KEY (pbrstanstudent)
 REFERENCES mjesto(pbr) CONSTRAINT studentFKpbrstan;
```

Sljedeća naredba se na postojećoj bazi podataka *studadmin* neće uspješno izvršiti.

```
ALTER TABLE student ADD CONSTRAINT UNIQUE (jmbg) CONSTRAINT studentUQ;
```

**Zašto?**

Ukidanje integritetskog ograničenja – primjer:

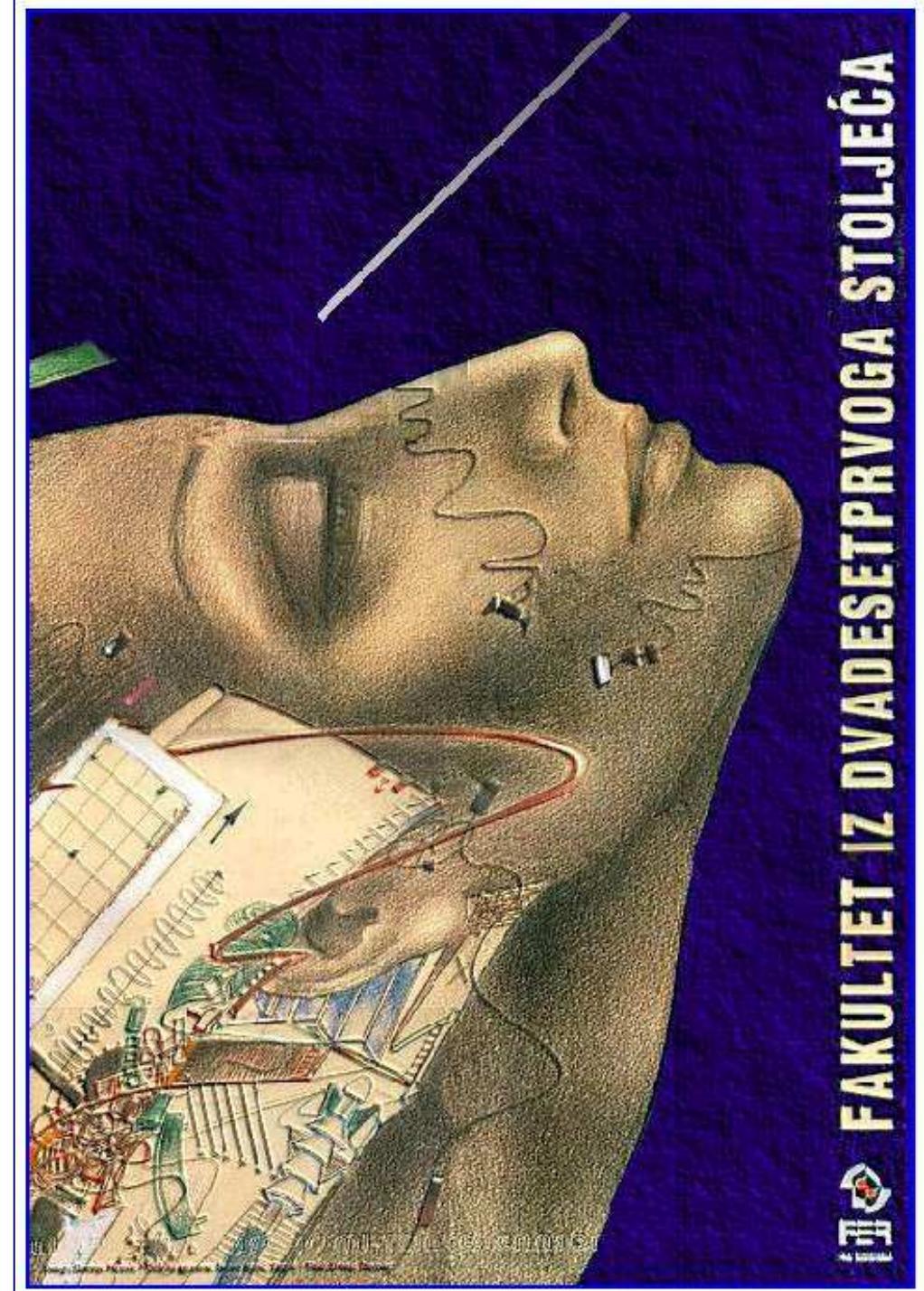
```
ALTER TABLE student DROP CONSTRAINT studentPK
```

Za vježbu napisati i izvesti sve preostale naredbe potrebne za očuvanje integriteta baze *studadmin*.

# Baze podataka

Predavanja  
svibanj 2014.

## 11. Privremene i virtualne relacije



# Vrste relacija (tablica)

---

- **Temeljna relacija (*base relation*)**
  - relacija korespondentna skupu entiteta u konceptualnoj shemi, čija su shema i sadržaj trajno pohranjeni u bazi podataka
- **Privremena relacija (*temporary relation*)**
  - relacija čija su shema i sadržaj u bazu podataka pohranjeni privremeno
- **Virtualna relacija (*virtual relation, view*)**
  - relacija kojoj su shema i sadržaj definirani izrazom relacijske algebre čiji su operandi temeljne ili virtualne relacije
    - u praksi, shema i sadržaj virtualne relacije opisuju se u obliku SQL upita
  - sadržaj virtualne relacije dinamički se određuje u trenutku obavljanja operacije nad virtualnom relacijom: ovisi o trenutačnom stanju temeljnih relacija

# Temeljna relacija

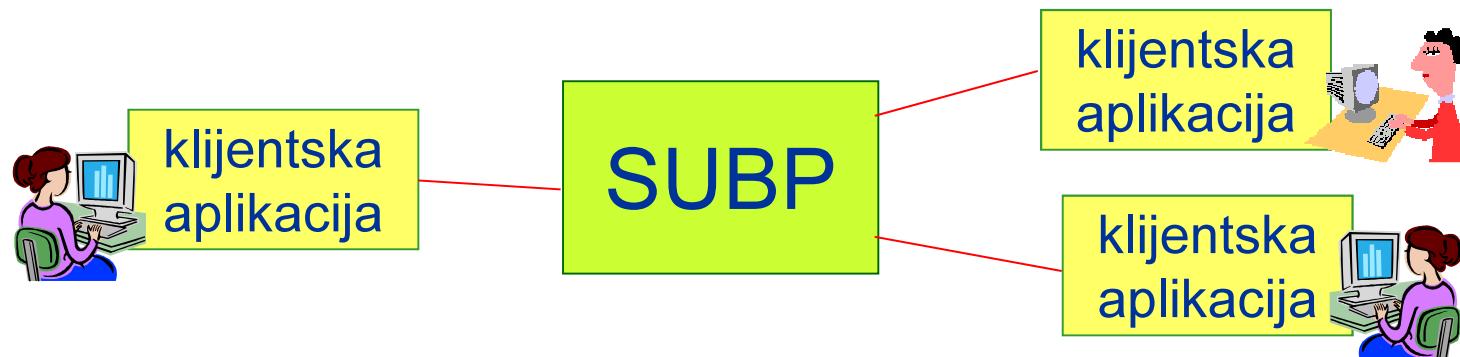
---

- Obavljanjem naredbe CREATE TABLE u rječnik podataka se pohranjuju metapodaci
  - naziv relacije
  - nazivi i tipovi atributa
  - integritetska ograničenja
  - ostali metapodaci (vrijeme kreiranja, vlasnik, primjenjena fizička organizacija, itd.)
- shema i sadržaj temeljne relacije su **postojani**: pohranjeni su u bazi podataka na neograničeno vrijeme
  - mijenjaju se tek u slučaju obavljanja eksplicitnih operacija za izmjenu sadržaja (UPDATE, DELETE, INSERT) ili sheme relacije (ALTER TABLE)

# (SQL-sjednica)

---

- SQL-sjednica (*SQL-session*) je kontekst u kojem jedan korisnik obavlja niz SQL naredbi putem jedne veze (*SQL-Connection*) prema sustavu za upravljanje bazama podataka
  - SQL-sjednica započinje u trenutku kada korisnik ostvari vezu (*connect*) sa sustavom za upravljanje bazama podataka
    - npr. u trenutku kada korisnik uporabom klijentske aplikacije Server Studio ostvari vezu s IBM Informix sustavom za upravljanje bazama podataka
  - SQL-sjednica završava u trenutku kada korisnik prekine vezu (*disconnect*) prema sustavu za upravljanje bazama podataka



# Privremena relacija

---

- Privremena relacija se kreira obavljanjem naredbe CREATE TEMP TABLE
  - sintaksa preostalog dijela naredbe je identična sintaksi naredbe CREATE TABLE, uz određena ograničenja
    - npr. nije moguće definirati ograničenje referencijskog integriteta
- Privremena relacija je u dosegu ("vidljiva je") isključivo u okviru SQL-sjednice tijekom koje je kreirana
  - "svaka SQL-sjednica koristi svoje privremene relacije"
- Privremene relacije se koriste kao pomoćni objekti, npr. za pohranu međurezultata pri obavljanju složenijih upita
  - **zašto temeljne relacije nisu prikladne za tu namjenu?**
- Privremena relacija se uklanja iz baze podataka:
  - obavljanjem naredbe DROP TABLE *nazivPrivremeneRelacije* ili
  - završetkom SQL-sjednice tijekom koje je ta privremena relacija kreirana

# Primjer

- Ispisati najmanju i najveću prosječnu ocjenu predmeta u obliku:

| minPros | maksPros |
|---------|----------|
| 3.00    | 4.00     |

| polozeniIspit |         |     |
|---------------|---------|-----|
| mbr           | sifPred | ocj |
| 100           | 1001    | 5   |
| 101           | 1001    | 4   |
| 102           | 1001    | 3   |
| 100           | 1002    | 2   |
| 101           | 1002    | 5   |
| 100           | 1003    | 3   |
| 101           | 1003    | 3   |

```
CREATE TEMP TABLE prosjek (
 sifPred INTEGER
, prosOcj DECIMAL(3,2));
```

projek

sifPred prosOcj

```
INSERT INTO prosjek
SELECT sifPred, AVG(ocj)
FROM polozeniIspit
GROUP BY sifPred;
```

projek

| sifPred | prosOcj |
|---------|---------|
| 1001    | 4.00    |
| 1002    | 3.50    |
| 1003    | 3.00    |

```
SELECT MIN(prosOcj) AS minPros
, MAX(prosOcj) AS maksPros
FROM prosjek;
```

| minPros | maksPros |
|---------|----------|
| 3.00    | 4.00     |

Za vježbu riješiti bez korištenja privremene relacije!

# Primjer (nastavak)

- Treba primijetiti: sadržaj privremene relacije *projek* neće se "automatski" promijeniti nakon upisa još jedne n-torke u temeljnu relaciju *polozeniIspit*

```
INSERT INTO polozeniIspit VALUES(102, 1003, 2);
```

| polozeniIspit | mbr | sifPred | ocj |
|---------------|-----|---------|-----|
|               | 100 | 1001    | 5   |
|               | 101 | 1001    | 4   |
|               | 102 | 1001    | 3   |
|               | 100 | 1002    | 2   |
|               | 101 | 1002    | 5   |
|               | 100 | 1003    | 3   |
|               | 101 | 1003    | 3   |
|               | 102 | 1003    | 2   |

Sadržaj  
privremene  
relacije se  
time nije  
promijenio:

| projek  |         |
|---------|---------|
| sifPred | prosOcj |
| 1001    | 4.00    |
| 1002    | 3.50    |
| 1003    | 3.00    |

projek za predmet  
1003 bi trebao biti 2.67

```
SELECT MIN(prosOcj) AS minPros
 , MAX(prosOcj) AS maksPros
FROM projek;
```

| minPros | maksPros |
|---------|----------|
| 3.00    | 4.00     |

neispravan  
rezultat

# Virtualna relacija (primjer)

- Problem "zastarijevanja" podataka u privremenim relacijama može se izbjeći uporabom virtualnih relacija

| polozeniIspit |         |     |
|---------------|---------|-----|
| mbr           | sifPred | ocj |
| 100           | 1001    | 5   |
| 101           | 1001    | 4   |
| 102           | 1001    | 3   |
| 100           | 1002    | 2   |
| 101           | 1002    | 5   |
| 100           | 1003    | 3   |
| 101           | 1003    | 3   |

```
CREATE VIEW prosjek (sifPred
, prosOcj) AS
SELECT sifPred, AVG(ocj)
FROM polozeniIspit
GROUP BY sifPred;
```

| prosjek |         |
|---------|---------|
| sifPred | prosOcj |
| ?       |         |

- Tek u trenutku obavljanja upita, SUBP dinamički određuje sadržaj virtualne relacije *prosjek*

```
SELECT MIN(prosOcj) AS minPros
, MAX(prosOcj) AS maksPros
FROM prosjek;
```

| prosjek |         |
|---------|---------|
| sifPred | prosOcj |
| 1001    | 4.00    |
| 1002    | 3.50    |
| 1003    | 3.00    |

| minPros | maksPros |
|---------|----------|
| 3.00    | 4.00     |

# Primjer (nastavak)

- Sadržaj virtualne relacije se ponovno određuje pri izvršavanju svakog upita koji koristi tu virtualnu relaciju

```
INSERT INTO polozeniIspit VALUES(102, 1003, 2);
```

polozeniIspit

| mbr | sifPred | ocj |
|-----|---------|-----|
| 100 | 1001    | 5   |
| 101 | 1001    | 4   |
| 102 | 1001    | 3   |
| 100 | 1002    | 2   |
| 101 | 1002    | 5   |
| 100 | 1003    | 3   |
| 101 | 1003    | 3   |
| 102 | 1003    | 2   |

```
SELECT MIN(prosOcj) AS minPros
 , MAX(prosOcj) AS maksPros
FROM prosjek;
```

projek

| sifPred | prosOcj |
|---------|---------|
| 1001    | 4.00    |
| 1002    | 3.50    |
| 1003    | 2.67    |

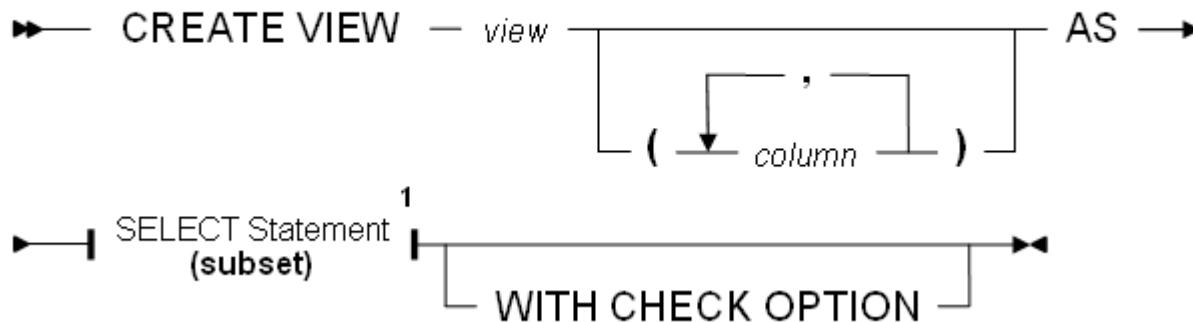
| minPros | maksPros |
|---------|----------|
| 2.67    | 4.00     |

# Naredba za kreiranje virtualne relacije

---

- Virtualna relacija se kreira naredbom CREATE VIEW

## 6. CREATE VIEW Statement



- SELECT Statement (subset)*** može sadržavati sve prethodno opisane dijelove SELECT naredbe osim
  - ORDER BY
  - FIRST n
- Virtualna relacija se uklanja iz baze podataka naredbom:
  - **DROP VIEW nazivVirtualneRelacije;**

# Svojstva virtualne relacije

---

- Obavljanjem naredbe CREATE VIEW u rječnik podataka se pohranjuje samo definicija virtualne relacije
  - sadržaj virtualne relacije se određuje tek za vrijeme izvršavanja upita koji koristi virtualnu relaciju
  - odnosno, sadržaj virtualne relacije uvijek odražava sadržaj temeljnih relacija u trenutku izvršavanja upita u kojem se virtualna relacija koristi
- virtualne relacije se u upitima mogu koristiti na svim mjestima gdje se mogu koristiti temeljne relacije
  - između ostalog i za kreiranje novih virtualnih relacija
- za razliku od privremene relacije
  - definicija virtualne relacije je trajno pohranjena u bazi podataka
  - virtualna relacija je u dosegu ("vidljiva je") u svim SQL-sjednicama

# Atributi virtualne relacije

- Ukoliko se nazivi atributa u definiciji virtualne relacije ne navedu, nazivi atributa virtualne relacije određeni su nazivima atributa u SELECT naredbi kojom se definira sadržaj virtualne relacije
- tipovi podataka za atribute virtualne relacije proizlaze iz tipova podataka atributa temeljnih relacija koje se koriste u definiciji virtualne relacije

```
CREATE VIEW zadrani1 AS
 SELECT mbr, ime, prez
 FROM osoba
 WHERE pbrStan = 23000;
SELECT * FROM zadrani1;
```

| mbr | ime | prez  |
|-----|-----|-------|
| 101 | Ana | Kolar |
| 103 | Tea | Ban   |

```
CREATE VIEW zadrani2 (matBr
 , imeSt
 , prezSt) AS
 SELECT mbr, ime, prez
 FROM osoba
 WHERE pbrStan = 23000;
SELECT * FROM zadrani2;
```

| osoba |      |       |         |
|-------|------|-------|---------|
| mbr   | ime  | prez  | pbrStan |
| 101   | Ana  | Kolar | 23000   |
| 102   | Tomo | Novak | 21000   |
| 103   | Tea  | Ban   | 23000   |

| matBr | imeSt | prezSt |
|-------|-------|--------|
| 101   | Ana   | Kolar  |
| 103   | Tea   | Ban    |

# Atributi virtualne relacije

- Ukoliko se u listi za selekciju pri definiciji virtualne relacije koriste izrazi, nazivi atributa virtualne relacije se moraju eksplicitno navesti

| polozeniIspit |         |     |
|---------------|---------|-----|
| mbr           | sifPred | ocj |
| 100           | 1001    | 2   |
| 101           | 1001    | 4   |
| 102           | 1001    | 3   |
| 100           | 1002    | 2   |
| 101           | 1002    | 5   |
| 100           | 1003    | 3   |
| 101           | 1003    | 3   |

```
CREATE VIEW prosjek (sifPred
, prosOcj) AS
SELECT sifPred, AVG(ocj)
FROM polozeniIspit
GROUP BY sifPred;
```

ispravno

```
CREATE VIEW prosjek AS
SELECT sifPred
, AVG(ocj)
FROM polozeniIspit
GROUP BY sifPred;
```

neispravno

```
CREATE VIEW prosjek AS
SELECT sifPred
, AVG(ocj) AS prosOcj
FROM polozeniIspit
GROUP BY sifPred;
```

neispravno

# Implementacija virtualnih relacija

---

- Kako sustavi za upravljanje bazama podataka izvršavaju upite koji sadrže virtualne relacije?
  - **modifikacijom upita**
    - SUBP ugrađuje elemente definicije virtualne relacije u originalni SQL upit koji koristi virtualnu relaciju - umjesto originalnog SQL upita izvršava se modificirani SQL upit
  - **korištenjem materijalizirane virtualne relacije**
    - SUBP fizički pohranjuje sadržaj virtualne relacije. Kada se promjeni sadržaj neke od temeljnih relacija pomoću kojih je virtualna relacija definirana, SUBP automatski mijenja i sadržaj materijalizirane virtualne relacije
    - prednost: virtualne relacije koje se vrlo često koriste, a čiji se sadržaj određuje složenim upitim, ne moraju se svaki puta kada neki korisnik koristi tu virtualnu relaciju ponovno izračunavati
    - nedostatak: ukoliko se temeljne relacije pomoću kojih je virtualna relacija definirana često mijenjaju, pri svakoj izmjeni temeljnih relacija troši se dodatno vrijeme radi izmjene sadržaja virtualne relacije

# Implementacija virtualnih relacija modifikacijom upita

- Primjer:

| ispit |             |     |
|-------|-------------|-----|
| mbr   | predmet     | ocj |
| 100   | Elektronika | 3   |
| 100   | Fizika      | 2   |
| 101   | Elektronika | 5   |
| 101   | Fizika      | 2   |
| 102   | Fizika      | 1   |
| 103   | Fizika      | 5   |

| stud |      |        |         |
|------|------|--------|---------|
| mbr  | ime  | prez   | pbrStan |
| 100  | Ivan | Kolar  | 52100   |
| 101  | Ana  | Horvat | 42230   |
| 102  | Jura | Novak  | 52100   |
| 103  | Ana  | Ban    | 52100   |

| mjesto |           |
|--------|-----------|
| pbr    | nazMjesto |
| 42000  | Varaždin  |
| 52100  | Pula      |
| 42230  | Ludbreg   |

studenti koji su položili predmet Fizika

```
CREATE VIEW polFiz AS
 SELECT stud.* , ocj
 FROM ispit, stud
 WHERE ispit.mbr = stud.mbr
 AND predmet = 'Fizika'
 AND ocj > 1;
```

korisnik obavlja:

```
SELECT * FROM polFiz;
```

↓ SUBP modificira upit

```
SELECT stud.* , ocj
 FROM ispit, stud
 WHERE ispit.mbr = stud.mbr
 AND predmet = 'Fizika'
 AND ocj > 1;
```



| mbr | ime  | prez   | pbrStan | ocj |
|-----|------|--------|---------|-----|
| 100 | Ivan | Kolar  | 52100   | 2   |
| 101 | Ana  | Horvat | 42230   | 2   |
| 103 | Ana  | Ban    | 52100   | 5   |

## Primjer (nastavak)

- Ispisati prezime, ime i dobivenu ocjenu iz Fizike za studente koji su položili Fiziku, a stanuju u Puli korisnik obavlja:

```
SELECT polFiz.prez, polFiz.ime, polFiz.ocj
 FROM polFiz, mjesto
 WHERE polFiz.pbrStan = mjesto.pbr
 AND nazMjesto = 'Pula';
```

```
CREATE VIEW polFiz AS
 SELECT stud.* , ocj
 FROM ispit, stud
 WHERE ispit.mbr = stud.mbr
 AND predmet = 'Fizika'
 AND ocj > 1;
```



SUBP modificira upit

```
SELECT stud.prez, stud.ime, ispit.ocj
 FROM ispit, stud, mjesto
 WHERE ispit.mbr = stud.mbr
 AND predmet = 'Fizika'
 AND ocj > 1
 AND stud.pbrStan = mjesto.pbr
 AND nazMjesto = 'Pula';
```



| prez  | ime  | ocj |
|-------|------|-----|
| Kolar | Ivan | 2   |
| Ban   | Ana  | 5   |

# Virtualna relacija: INSERT, UPDATE, DELETE

- virtualne relacije se također mogu koristiti u naredbama INSERT, UPDATE i DELETE

```
CREATE VIEW splitStud AS
 SELECT mbr, ime, prez, pbrStan
 FROM stud
 WHERE pbrStan = 21000;
```

```
INSERT INTO splitStud
VALUES (102, 'Jure', 'Novak', 21000);

SELECT * FROM splitStud;
```

| stud |      |        |         |
|------|------|--------|---------|
| mbr  | ime  | prez   | pbrStan |
| 100  | Ivan | Kolar  | 31000   |
| 101  | Ana  | Horvat | 21000   |



| mbr | ime  | prez   | pbrStan |
|-----|------|--------|---------|
| 101 | Ana  | Horvat | 21000   |
| 102 | Jure | Novak  | 21000   |

```
INSERT INTO splitStud
VALUES (103, 'Tea', 'Ban', 10000);

SELECT * FROM splitStud;
```



| mbr | ime  | prez   | pbrStan |
|-----|------|--------|---------|
| 101 | Ana  | Horvat | 21000   |
| 102 | Jure | Novak  | 21000   |

n-torka jest unesena u temeljnu relaciju,  
ali se "ne vidi" u virtualnoj relaciji

```
SELECT * FROM stud;
```



| mbr | ime  | prez   | pbrStan |
|-----|------|--------|---------|
| 100 | Ivan | Kolar  | 31000   |
| 101 | Ana  | Horvat | 21000   |
| 102 | Jure | Novak  | 21000   |
| 103 | Tea  | Ban    | 10000   |

# Virtualna relacija: INSERT, UPDATE, DELETE

- SUBP ne može promijeniti "sadržaj virtualne relacije" - umjesto toga mora promijeniti sadržaj temeljnih relacija koje se koriste u definiciji te virtualne relacije

ispit

| mbr | predmet     | ocj |
|-----|-------------|-----|
| 100 | Elektronika | 1   |
| 100 | Fizika      | 5   |
| 101 | Elektronika | 1   |
| 101 | Fizika      | 3   |

```
CREATE VIEW prosli AS
SELECT * FROM ispit
WHERE ocj > 1;
```

```
CREATE VIEW pali AS
SELECT * FROM ispit
WHERE ocj = 1;
```

korisnik  
obavlja:

```
UPDATE prosli SET ocj = 4
WHERE mbr = 100
AND predmet = 'Fizika';
```

↓ SUBP modificira upit

```
UPDATE ispit SET ocj = 4
WHERE ocj > 1
AND mbr = 100
AND predmet = 'Fizika';
```

# Virtualna relacija: problem migrirajućih n-torki

- n-torka se pojavljuje u virtualnoj relaciji onda kada zadovoljava uvjet iz definicije virtualne relacije
  - n-torka unesena u virtualnu relaciju ili izmijenjena u virtualnoj relaciji može "nestati" iz te virtualne relacije (i eventualno se "pojaviti" u nekoj drugoj virtualnoj relaciji)

ispit

|                | mbr | predmet     | ocj |
|----------------|-----|-------------|-----|
| t <sub>1</sub> | 100 | Elektronika | 1   |
| t <sub>2</sub> | 100 | Fizika      | 5   |
| t <sub>3</sub> | 101 | Elektronika | 1   |
| t <sub>4</sub> | 101 | Fizika      | 3   |

```
CREATE VIEW prosli AS
SELECT * FROM ispit
WHERE ocj > 1;
```

```
CREATE VIEW pali AS
SELECT * FROM ispit
WHERE ocj = 1;
```

korisnik  
obavlja:

```
UPDATE prosli SET ocj = 1
WHERE mbr = 100
AND predmet = 'Fizika';
INSERT INTO prosli
VALUES (102, 'Elektronika', 1);
```

- n-torka t<sub>2</sub> je "nestala" iz prosli i "pojavila" se u pali
- nova n-torka <102, Elektronika, 1> unesena preko prosli se "pojavila" u pali

# Virtualna relacija: problem migrirajućih n-torki

- **Rješenje:** virtualne relacije koje se koriste u naredbama koje mijenjaju podatke obavezno se kreiraju uz opciju **WITH CHECK OPTION**
  - SUBP tada ne dopušta izmjenu ili unos n-torke putem virtualne relacije ukoliko n-torka nakon obavljanja operacije više ne bi pripadala virtualnoj relaciji putem koje je izmijenjena ili unesena

| ispit | mbr | predmet     | ocj |
|-------|-----|-------------|-----|
|       | 100 | Elektronika | 1   |
|       | 100 | Fizika      | 5   |
|       | 101 | Elektronika | 1   |
|       | 101 | Fizika      | 3   |

```
CREATE VIEW prosli AS
SELECT * FROM ispit
WHERE ocj > 1
WITH CHECK OPTION;
```

```
CREATE VIEW pali AS
SELECT * FROM ispit
WHERE ocj = 1
WITH CHECK OPTION;
```

```
UPDATE prosli SET ocj = 1
WHERE mbr = 100 pogreška
AND predmet = 'Fizika';
```

```
UPDATE prosli SET ocj = 4
WHERE mbr = 100 O.K.
AND predmet = 'Fizika';
```

```
INSERT INTO prosli pogreška
VALUES (102, 'Fizika', 1);
```

```
INSERT INTO prosli O.K.
VALUES (102, 'Fizika', 3);
```

```
INSERT INTO pali pogreška
VALUES (102, 'Fizika', 3);
```

# Neizmjenjive virtualne relacije

- SUBP ne može promijeniti "sadržaj virtualne relacije" - umjesto toga mora promijeniti sadržaj temeljnih relacija koje se koriste u definiciji te virtualne relacije
  - ako je virtualna relacija definirana tako da SUBP nije u stanju **jednoznačno** odrediti koje operacije treba obaviti na temeljnim relacijama, tada je virtualna relacija **neizmjenjiva** (*non-updatable*)

polozeniIspit

| mbr | sifPred | ocj |
|-----|---------|-----|
| 100 | 1001    | 5   |
| 101 | 1001    | 4   |
| 102 | 1001    | 3   |
| 100 | 1002    | 2   |
| 101 | 1002    | 5   |
| 100 | 1003    | 3   |
| 101 | 1003    | 3   |

```
CREATE VIEW projek (sifPred
, prosOcj) AS
SELECT sifPred, AVG(ocj)
FROM polozeniIspit
GROUP BY sifPred;
```

SELECT \* FROM projek;

| sifPred | prosOcj |
|---------|---------|
| 1001    | 4.00    |
| 1002    | 3.50    |
| 1003    | 3.00    |

```
UPDATE projek SET prosOcj = 4.5
WHERE sifPred = 1001;
```

?

```
INSERT INTO projek VALUES (1004, 2.5);
```

?

# Izmjenjive virtualne relacije

---

- Virtualna relacija je izmjenjiva ukoliko u glavnom SELECT dijelu definicije virtualne relacije koristi atribute iz samo jedne temeljne relacije r(R) i pri tome:
  - ne sadrži eliminaciju duplikata pomoću DISTINCT
  - ne sadrži izraze u listi za selekciju (osim trivijalnih izraza koji sadrže samo ime atributa)
  - izostavljeni atributi ne smiju imati NOT NULL ograničenje ili moraju imati pretpostavljenu (*default*) vrijednost
  - ne sadrži spajanje ili uniju
  - ne sadrži grupiranje i postavljanje uvjeta nad grupom (GROUP BY i HAVING)
- Prethodno navedena ograničenja se ne odnose na eventualne podupite koji se koriste unutar WHERE dijela SELECT naredbe koja se koristi za definiciju virtualne relacije

# Primjeri izmjenjivih virtualnih relacija

| ispit | matBr | sifPred | datIsp     | ocj | sifNas |
|-------|-------|---------|------------|-----|--------|
|       | 1111  | 1001    | 29.01.2011 | 1   | 101    |
|       | 1111  | 1001    | 05.02.2011 | 3   | 101    |
|       | 1111  | 1003    | 28.06.2011 | 2   | 303    |
|       | 1111  | 1002    | 27.06.2011 | 4   | 202    |
|       | 1234  | 1001    | 29.01.2011 | 3   | 202    |

| stud | matBr | prez  | ime   | pbrSt |
|------|-------|-------|-------|-------|
|      | 1111  | Novak | Ivan  | 10000 |
|      | 4444  | Ban   | Marko | 51000 |
|      | 1234  | Kolar | Petar | 23000 |

```
CREATE VIEW poloziliNista AS
 SELECT * FROM stud
 WHERE NOT EXISTS
 (SELECT * FROM ispit
 WHERE ispit.matBr = stud.matBr
 AND ocj > 1)
 WITH CHECK OPTION;
```

```
CREATE VIEW poloziliBaremDva AS
 SELECT * FROM stud
 WHERE
 (SELECT COUNT(*) FROM ispit
 WHERE ispit.matBr = stud.matBr
 AND ocj > 1) >= 2
 WITH CHECK OPTION;
```

```
CREATE VIEW ispitiZadranal AS
 SELECT matBr
 , sifPred
 , datIsp
 , ocj
 FROM ispit
 WHERE matBr IN (
 SELECT matBr FROM stud
 WHERE pbrSt = 23000)
 WITH CHECK OPTION;
```

# Primjeri neizmjenjivih virtualnih relacija

| ispit | matBr | sifPred | datIsp     | ocj | sifNas |
|-------|-------|---------|------------|-----|--------|
|       | 1111  | 1001    | 29.01.2011 | 1   | 1111   |
|       | 1111  | 1001    | 05.02.2011 | 3   | 1111   |
|       | 1111  | 1003    | 28.06.2011 | 2   | 3333   |
|       | 1111  | 1002    | 27.06.2011 | 4   | 2222   |
|       | 1234  | 1001    | 29.01.2011 | 3   | 2222   |

| stud | matBr | prez  | ime   | pbrSt |
|------|-------|-------|-------|-------|
|      | 1111  | Novak | Ivan  | 10000 |
|      | 1234  | Kolar | Petar | 21000 |

```
CREATE VIEW ispitiZadrana2 AS
 SELECT ispit.matBr
 , sifPred
 , datIsp
 , ocj
 FROM ispit, stud
 WHERE ispit.matBr = stud.matBr
 AND pbrSt = 23000;
```



usporediti s izmjenjivom virtualnom  
relacijom **ispitiZadrana1** S  
prethodne stranice!

```
CREATE VIEW projek
 (matBr, prosOcj) AS
 SELECT matBr, AVG(ocj)
 FROM ispit
 GROUP BY matBr;
```

```
CREATE VIEW stud1 (ime prez) AS
 SELECT ime || prez
 FROM stud;
```

```
CREATE VIEW poloziliNesto AS
 SELECT DISTINCT matBr
 FROM ispit
 WHERE ocj > 1;
```

# Implementacija eksternih shema pomoću virtualnih relacija

- konceptualna shema: baza podataka u banci

| klijent | jmbg   | ime    | prez    | uplatalsplata | brRac | vrijeme        | valuta | iznos   |
|---------|--------|--------|---------|---------------|-------|----------------|--------|---------|
|         | 123456 | Ana    | Horvat  |               | 1001  | 7.8.2007 08:20 | HRK    | 15.00   |
|         | 654321 | Ivan   | Novak   |               | 1002  | 9.4.2006 12:31 | EUR    | -100.21 |
|         | 123654 | Tea    | Kolar   |               | 1001  | 6.5.2007 14:15 | HRK    | 452.15  |
| racun   | brRac  | jmbgVI | tipRac  | datRac        | 1004  | 5.5.2007 16:42 | HRK    | 1200.00 |
|         | 1001   | 123456 | kunski  | 7.2.2007      | 1004  | 9.9.2005 10:15 | HRK    | -350.50 |
|         | 1002   | 123456 | devizni | 1.3.2006      | 1002  | 7.2.2007 15:01 | EUR    | 235.20  |
|         | 1003   | 654321 | devizni | 4.8.2004      | 1003  | 1.4.2005 12:44 | USD    | 2750.00 |
|         | 1004   | 123654 | kunski  | 8.9.2005      | 1001  | 1.9.2007 12:19 | HRK    | -250.35 |
|         |        |        |         |               | 1004  | 8.2.2006 11:55 | HRK    | 420.00  |

- za različite kategorije korisnika (aplikacija) definiraju se različite eksterne sheme
  - aplikacija za otvaranje računa
  - aplikacija za deviznu upлату/isплату
  - aplikacija za kunsku upлату/isплату
  - aplikacija za pregled trenutačnog stanja sredstava u banci

# Implementacija eksternih shema pomoću virtualnih relacija

```
CREATE VIEW devRacun AS
 SELECT * FROM racun
 WHERE tipRac = 'devizni'
 WITH CHECK OPTION;
```

```
CREATE VIEW devUplIspl AS
 SELECT * FROM uplataIsplata
 WHERE valuta <> 'HRK'
 WITH CHECK OPTION;
```

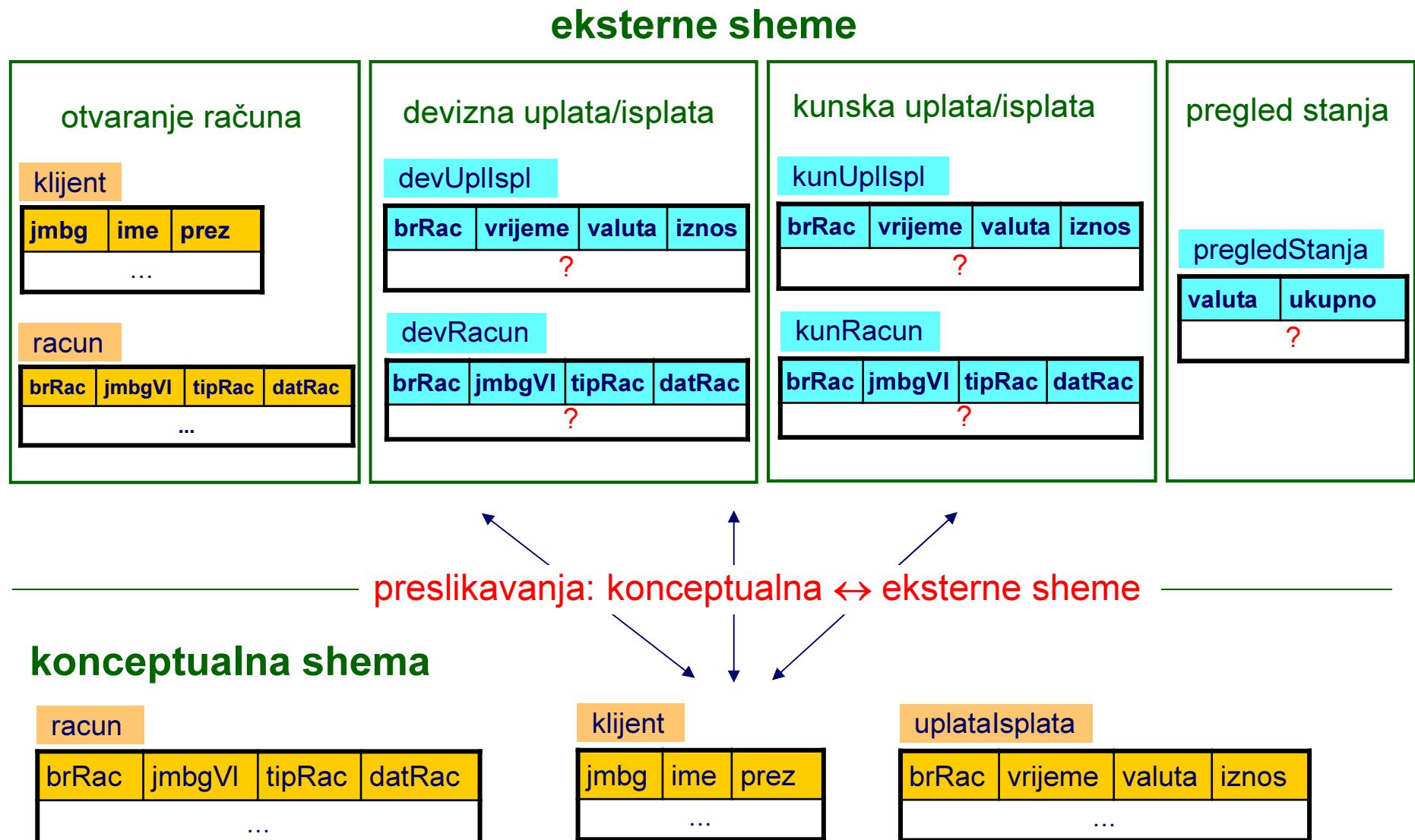
```
CREATE VIEW kunRacun AS
 SELECT * FROM racun
 WHERE tipRac = 'kunski'
 WITH CHECK OPTION;
```

```
CREATE VIEW kunUplIspl AS
 SELECT * FROM uplataIsplata
 WHERE valuta = 'HRK'
 WITH CHECK OPTION;
```

```
CREATE VIEW pregledStanja
 (valuta, ukupno) AS
 SELECT valuta, SUM(iznos)
 FROM uplataIsplata
 GROUP BY valuta;
```

- eksterne sheme za aplikacije
  - za otvaranje računa: **klijent**, **racun**
  - za deviznu uplatu/isplatu: **devRacun**, **devUplIspl**
  - za kunsku uplatu/isplatu: **kunRacun**, **kunUplIspl**
  - za pregled trenutačnog stanja sredstava: **pregledStanja**

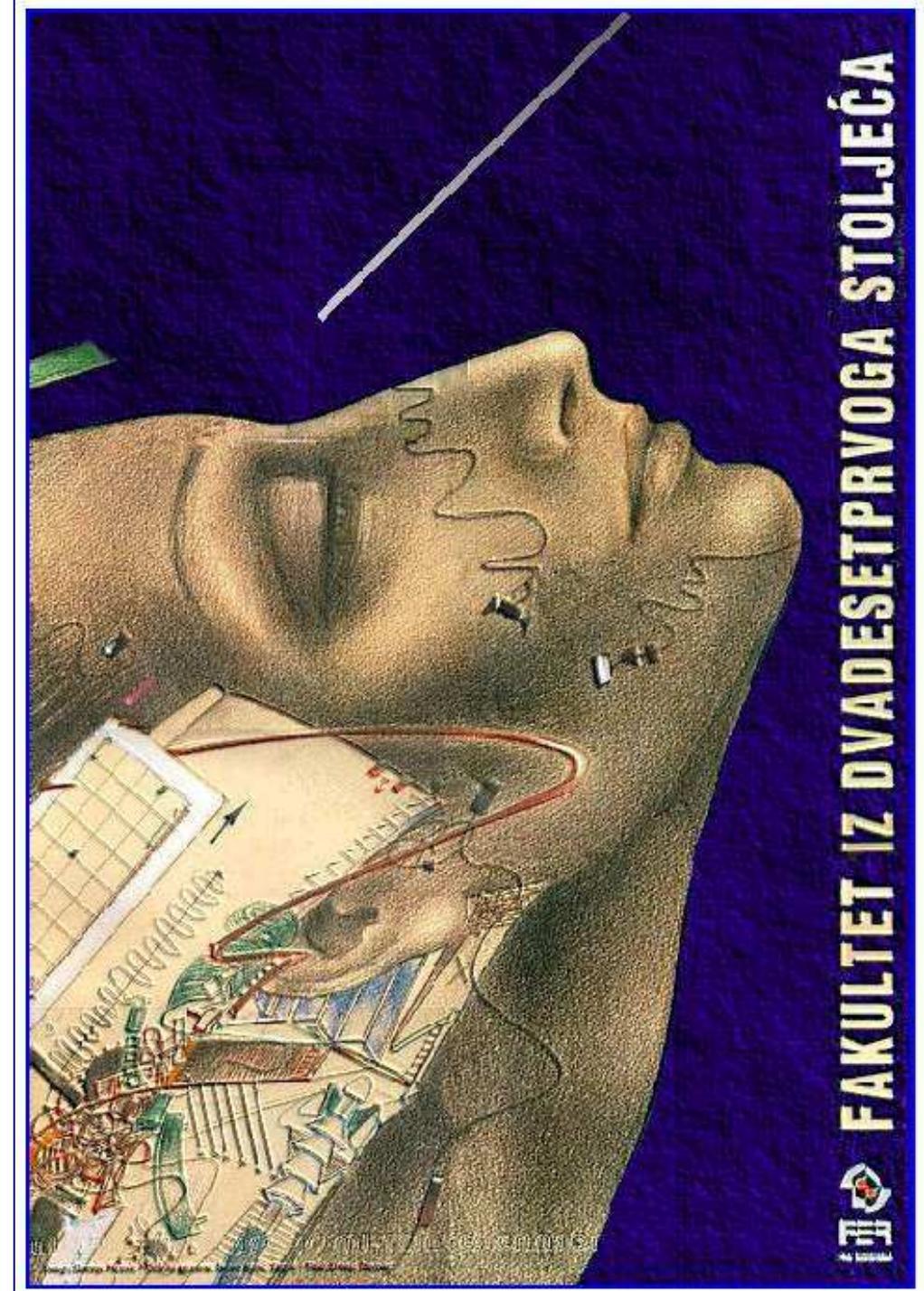
# Implementacija eksternih shema pomoću virtualnih relacija



# Baze podataka

Predavanja  
svibanj 2014.

## 12. Pohranjene procedure i okidači



# **Pohranjene procedure**

# Primjer 1:

|       | CHAR (13)       | CHAR (30) |
|-------|-----------------|-----------|
| osoba | jmbg            | prez      |
|       | 2203979622876   | Horvat    |
|       | 1712 12871211   | Kolar     |
|       | 2707986736233   | Še5fer    |
|       | 03AB621 . 22876 | Novak     |

- smatra se da su ispravne one vrijednosti atributa JMBG u kojima postoji točno 13 znamenaka
- smatra se da su ispravna ona prezimena u kojima ne postoji niti jedna znamenka
- ispisati podatke o osobama s neispravnim JMBG-om ili prezimenom
- kad bi barem postojala SQL funkcija CountDigits(nizZnakova)**

```
SELECT * FROM osoba
WHERE CountDigits(jmbg) <> 13
OR CountDigits(prez) > 0;
```

# Pohranjene procedure (pohranjene funkcije)

---

- Pohranjena procedura ili pohranjena funkcija je potprogram koji je pohranjen u rječniku podataka i koji se izvršava u kontekstu sustava za upravljanje bazama podataka
  - može se promatrati kao procedura ili funkcija kojom se proširuje skup SQL funkcija ugrađenih u SUBP
    - procedura je potprogram koji u pozivajući program ne vraća rezultat
    - funkcija je potprogram koji u pozivajući program vraća rezultat

## Primjer 1 (nastavak):

---

- Funkcija koja u zadanom nizu znakova broji koliko ima znakova koji su znamenke (broji znakove iz intervala '0' ... '9'). Pretpostavlja se da duljina zadanog niza znakova ne premašuje 255 bajtova

```
CREATE FUNCTION brojZnamenki (niz CHAR(255))
 RETURNING SMALLINT AS broj
DEFINE brojac, i SMALLINT;
LET brojac = 0;
FOR i = 1 TO CHAR_LENGTH(niz)
 IF SUBSTRING(niz FROM i FOR 1) BETWEEN '0' AND '9' THEN
 LET brojac = brojac + 1;
 END IF;
END FOR;
RETURN brojac;
END FUNCTION;

GRANT EXECUTE ON brojZnamenki TO PUBLIC;
```

- funkciju brojZnamenki svaki (sadašnji i budući) korisnik može koristiti na jednak način kao što se koriste standardne SQL funkcije

## Primjer 1 (nastavak):

---

|       | CHAR (13)     | CHAR (30) |
|-------|---------------|-----------|
| osoba | jmbg          | prez      |
|       | 2203979622876 | Horvat    |
|       | 1712 12871211 | Kolar     |
|       | 2707986736233 | Še5fer    |
|       | 03AB621.22876 | Novak     |

- funcija brojZnamenki se može iskoristiti za ispis onih osoba u čijem JMBG-u nema točno 13 znamenaka ili u prezimenu postoje znamenke

```
SELECT *, brojZnamenki(jmbg) AS br1, brojZnamenki(prez) AS br2
FROM osoba
WHERE brojZnamenki(jmbg) <> 13 OR brojZnamenki(prez) > 0;
```

| jmbg          | prez   | br1 | br2 |
|---------------|--------|-----|-----|
| 1712 12871211 | Kolar  | 12  | 0   |
| 2707986736233 | Še5fer | 13  | 1   |
| 03AB621.22876 | Novak  | 10  | 0   |

## Primjer 1 (nastavak):

- Pohranjena funkcija se iz interaktivnih alata (npr. Server Studio) može pozvati na sljedeći način:

```
EXECUTE FUNCTION brojZnamenki('abc123efg456');
```

|      |
|------|
| broj |
| 6    |

```
CREATE FUNCTION brojZnamenki (niz CHAR(255))
 RETURNING SMALLINT AS broj
```

```
...
```

## Primjer 2:

---

- Korisnik novak je službenik u banci kojem je potrebno omogućiti obavljanje **isključivo** jedne vrste bankovne transakcije: prebacivanje iznosa s jednog na drugi račun

| racun | brRacun | stanje  |
|-------|---------|---------|
|       | 1001    | 1250.15 |
|       | 1002    | -300.00 |
|       | 1003    | 10.25   |

- Zadatak se ne može riješiti dodjelom dozvole za obavljanje operacije UPDATE nad relacijom racun korisniku novak (**zašto?**)

# Dozvole za pohranjene procedure/funkcije

---

- SQL naredbe za dodjeljivanje i ukidanje dozvola za izvršavanje procedura
- **GRANT EXECUTE ON {procName | funName}**  
    **TO {PUBLIC | userList | roleList}**  
    **[WITH GRANT OPTION]**
- **REVOKE EXECUTE ON {procName | funName}**  
    **FROM {PUBLIC | userList | roleList}**  
    **[ CASCADE | RESTRICT ]**

## Primjer 2 (nastavak):

---

```
CREATE PROCEDURE prebaci (saRacunaBr LIKE racun.brRacun
 , naRacunBr LIKE racun.brRacun
 , iznos LIKE racun.stanje)
 -- prenesi zadani iznos
 UPDATE racun SET stanje = stanje - iznos
 WHERE brRacun = saRacunaBr;
 UPDATE racun SET stanje = stanje + iznos
 WHERE brRacun = naRacunBr;
END PROCEDURE;
GRANT EXECUTE ON prebaci TO novak;
```

## Primjer 2 (nastavak):

| racun | brRacun | stanje  |
|-------|---------|---------|
|       | 1001    | 1250.15 |
|       | 1002    | -300.00 |
|       | 1003    | 10.25   |

novak UPDATE racun SET stanje = stanje - 60.30  
WHERE brRacun = 1001;

[Error] No UPDATE permission

novak EXECUTE PROCEDURE prebaci (1001, 1002, 60.30);

| racun | brRacun | stanje  |
|-------|---------|---------|
|       | 1001    | 1189.85 |
|       | 1002    | -239.70 |
|       | 1003    | 10.25   |

- Problem: što će se dogoditi ako korisnik pri pozivu procedure kao broj prvog računa zada postojeći, a kao broj drugog računa zada nepostojeći broj računa?  
`EXECUTE PROCEDURE prebaci(1001, 1005, 30.15);`

# Iznimke (Exceptions)

---

- ukoliko SUBP tijekom obavljanja operacije utvrdi da se dogodila pogreška (*error condition*), obavljanje operacije se prekida, a stanje pogreške se signalizira iznimkom (*exception*)

```
SELECT (stanje/(stanje-10.25)) FROM racun;
```

[Error] An attempt was made to divide by zero.

```
SELECT * FROM ispit;
```

[Error] No SELECT permission.

- pogreške koje SUBP nije u stanju prepoznati (jer ih ne smatra pogreškama), mogu se signalizirati naredbom **RAISE EXCEPTION**. Npr, u poboljšanoj proceduri **prebaci** signalizira se pogreška u slučaju kad ne postoji neki od zadanih brojeva računa

```
EXECUTE PROCEDURE prebaci(1001, 1005, 30.15);
```

[Error] Ne postoji drugi račun

## Primjer 2 (nastavak):

```
CREATE PROCEDURE prebaci (saRacunaBr LIKE racun.brRacun
 , naRacunBr LIKE racun.brRacun
 , iznos LIKE racun.stanje)
 -- provjeri postoje li zadani brojevi računa
 IF (SELECT COUNT(*) FROM racun
 WHERE brRacun = saRacunaBr) = 0 THEN
 RAISE EXCEPTION -746, 0, 'Ne postoji prvi račun';
 END IF;
 IF (SELECT COUNT(*) FROM racun
 WHERE brRacun = naRacunBr) = 0 THEN
 RAISE EXCEPTION -746, 0, 'Ne postoji drugi račun';
 END IF;
 -- prenesi zadani iznos
 UPDATE racun SET stanje = stanje - iznos
 WHERE brRacun = saRacunaBr;
 UPDATE racun SET stanje = stanje + iznos
 WHERE brRacun = naRacunBr;
END PROCEDURE;
GRANT EXECUTE ON prebaci TO novak;
```

# SPL (*Stored Procedure Language*)

---

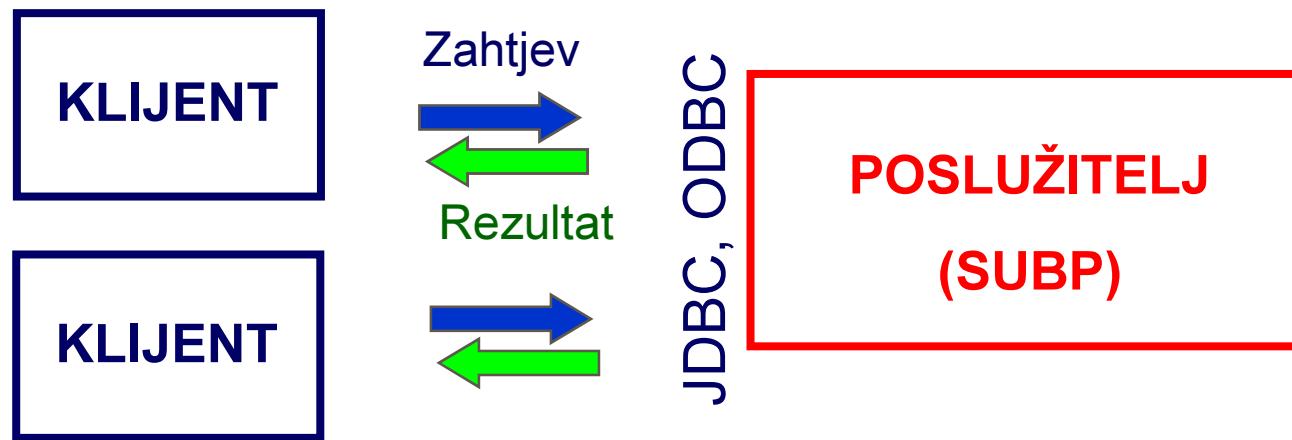
- Proizvođači SUBP koriste vlastite inačice jezika za definiranje pohranjenih procedura (standard postoji, ali je rijetko gdje implementiran)
  - IBM Informix: SPL (Stored Procedure Language)
  - Oracle: PL/SQL (Procedural Language/Structured Query Language)
  - Microsoft SQL Server: Transact-SQL
- Navedeni jezici proširuju mogućnosti SQL jezika proceduralnim elementima koji se koriste u strukturiranim jezicima (C, Java, ...). Osim SQL naredbi, pohranjene procedure omogućuju korištenje
  - varijabli
  - naredbi za kontrolu toka programa (*if, for, while, ...*)
  - naredbi za rukovanje iznimkama (*exception handling*)

# Prednosti uporabe pohranjenih procedura

---

- proširenje mogućnosti SQL jezika
- omogućena je zaštita podataka na razini funkcije (a ne samo objekta)
- omogućena je uporaba klijent-poslužitelj arhitekture oslonjene na poslužitelj:
  - postiže se veća učinkovitost SUBP
    - SUBP ne mora ponavljati prevođenje i optimiranje SQL upita
  - postiže se veća produktivnost programera i smanjuje mogućnost pogreške
    - programski kôd potreban za obavljanje nekog postupka koji čini logičku cjelinu implementira se i testira na samo jednom mjestu

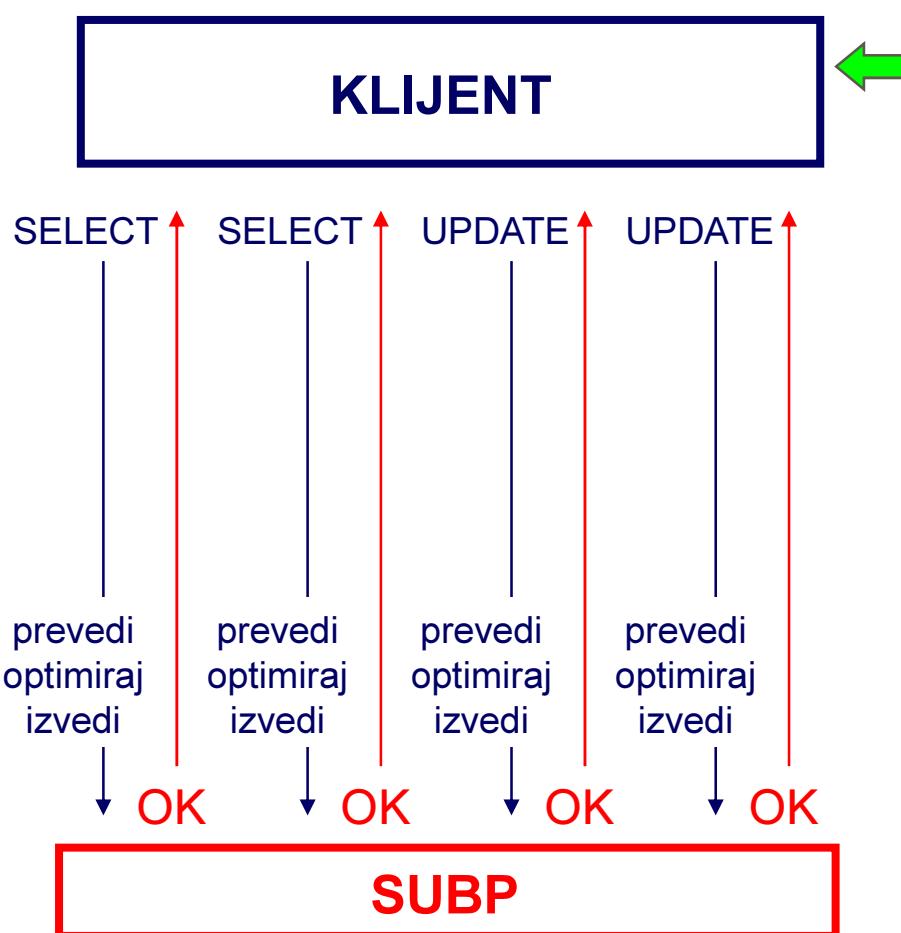
# (Klijent-poslužitelj arhitektura)



- sustav obuhvaća dvije komponente
  - klijent i poslužitelj (*client-server*)
- koncept zahtjev-odgovor (*request-response*): klijent postavlja zahtjev, poslužitelj odgovara
- komunikacija između klijenta i poslužitelja se odvija preko dobro definiranih, standardnih programskih sučelja: npr. ODBC (*Open Database Connectivity*), JDBC (*Java Database Connectivity*)

# (Klijent-poslužitelj arhitektura - oslonjena na klijenta)

- provjeri postoje li zadani brojevi računa, ako postoje, prebaci iznos s jednog na drugi račun

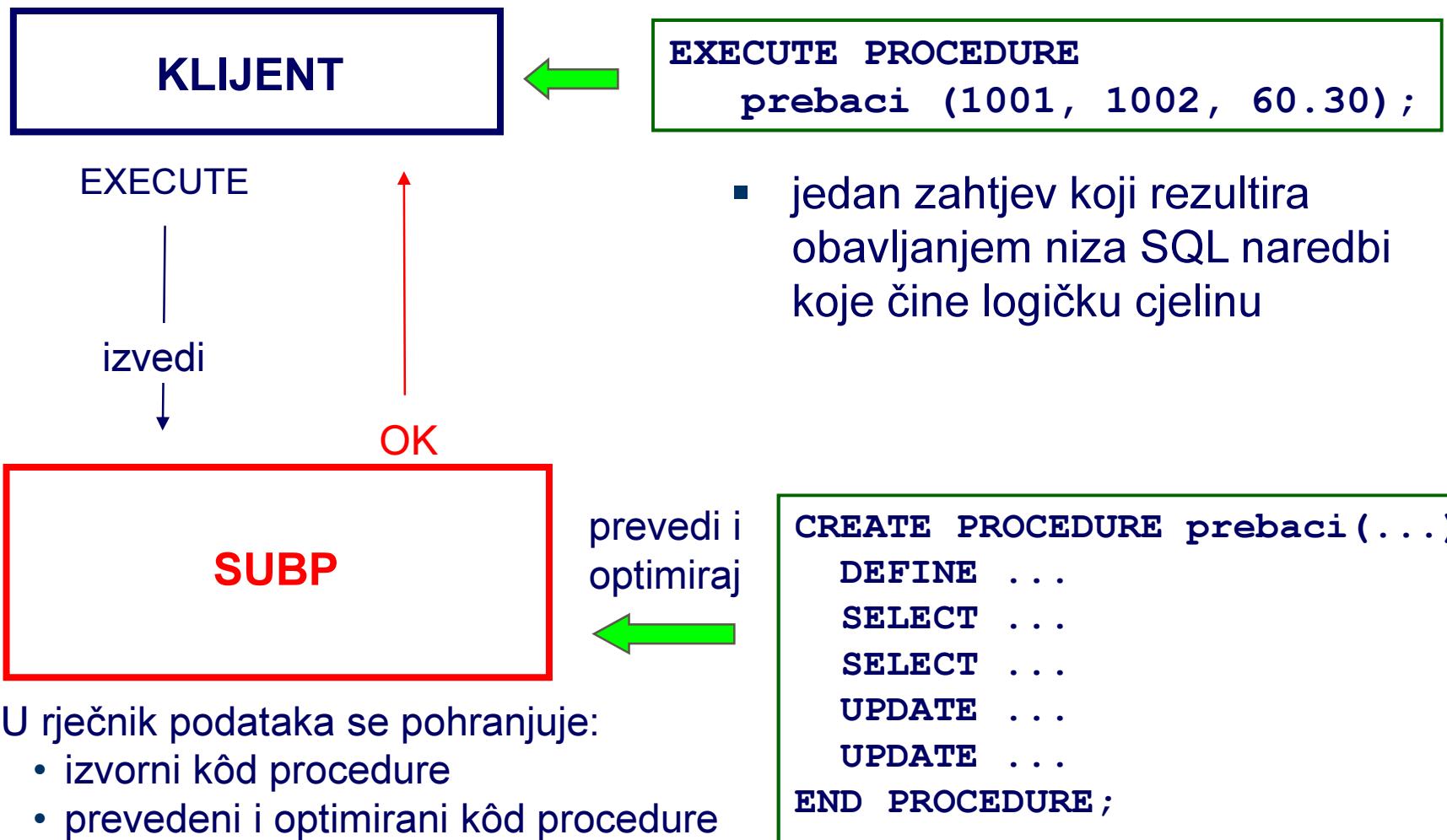


```
SELECT COUNT(*) FROM racun
WHERE brRacun = 1001;
?
SELECT COUNT(*) FROM racun
WHERE brRacun = 1002;
?
UPDATE racun
SET stanje = stanje - 60.30
WHERE brRacun = 1001;
UPDATE racun
SET stanje = stanje + 60.30
WHERE brRacun = 1002;
```

- niz zahtjeva za obavljanje jedne po jedne SQL naredbe

# (Klijent-poslužitelj arhitektura - oslonjena na poslužitelj)

- provjeri postoje li zadani brojevi računa, ako postoje, prebaci iznos s jednog na drugi račun



**Okidači**

## Primjer 3:

| racun | brRac | sifKlijent | stanje |
|-------|-------|------------|--------|
| 1001  | 98281 | 216.80     |        |
| 1002  | 89734 | 134.99     |        |
| 1003  | 23232 | 2750.00    |        |
| 1004  | 63443 | 849.50     |        |
| ...   |       |            | ...    |

| uplatalsplata | brRac | vrijeme        | iznos   |
|---------------|-------|----------------|---------|
|               | 1001  | 7.8.2007 08:20 | 15.00   |
|               | 1002  | 9.4.2006 12:31 | -100.21 |
|               | 1001  | 6.5.2007 14:15 | 452.15  |
|               | 1004  | 5.5.2007 16:42 | 1200.00 |
|               | 1004  | 9.9.2005 10:15 | -350.50 |
|               | 1002  | 7.2.2007 15:01 | 235.20  |
|               | 1003  | 1.4.2005 12:44 | 2750.00 |
|               | 1001  | 1.9.2007 12:19 | -250.35 |
|               | ...   | ...            | ...     |

- u relaciju **uplatalsplata** upisuju se promjene na računima
- tijekom godina evidentiran je vrlo veliki broj uplata i isplata
- stanje na određenom računu moglo bi se izračunati zbrajanjem iznosa u relaciji **uplatalsplata**, koji se odnose na dotični račun
- u ovom primjeru, uz svaki račun se redundantno pohranjuje trenutno stanje računa, koje u svakom trenutku mora odgovarati stanju koje bi se dobilo zbrajanjem iznosa u relaciji **uplatalsplata**
- kako osigurati da se pri svakoj relevantnoj promjeni podataka (unos, brisanje, izmjena iznosa) u relaciji **uplatalsplata** izmjeni i odgovarajuće stanje u relaciji **racun**?

# Aktivne baze podataka

---

- konvencionalni SUBP je pasivan
  - operacije nad podacima se izvršavaju isključivo na temelju eksplicitnog zahtjeva korisnika/aplikacije
- aktivni SUBP i aktivne baze podataka
  - aktivni SUBP autonomno reagira na određene događaje (*events*)
  - u aktivnim bazama podataka neke operacije nad podacima se izvršavaju automatski, reakcijom na određeni događaj ili stanje
- željeno ponašanje sustava postiže se definiranjem aktivnih pravila (*active rules*)
- najčešće korištena paradigma za opisivanje aktivnih pravila u današnjim SUBP je događaj-uvjet-akcija (*ECA: Event-Condition-Action*)
  - okidači (*triggers*)

# ECA

---

**on *event***

**if *condition* then *action***

- **događaj (*event*):** ako se dogodi, izračunava se uvjet
  - općenito, događaji mogu biti:
    - unos, izmjena ili brisanje podatka
    - čitanje podatka
    - uspostavljanje SQL-sjednice
    - protok određene količine vremena, dostizanje trenutka u vremenu, ...
- **uvjet (*condition*):** ako je rezultat izračunavanja uvjeta istina, obavljaju se akcije
  - zadaje se u obliku predikata (slično kao u WHERE dijelu SQL naredbi)
- **akcije (*action*):** niz operacija, najčešće operacije nad podacima
  - SQL naredbe INSERT, UPDATE, DELETE, poziv procedure, ...

## Primjer 3 (nastavak):

- kako osigurati da se pri svakoj relevantnoj promjeni podataka (unos, brisanje, izmjena iznosa) u relaciji **uplatalsplata** izmijeni i odgovarajuće stanje u relaciji **racun**?

| racun | brRac | sifKlijent | stanje | uplatalsplata | brRac | vrijeme        | iznos   |
|-------|-------|------------|--------|---------------|-------|----------------|---------|
|       | 1001  | 98281      | 216.80 |               | 1001  | 7.8.2007 08:20 | 15.00   |
|       | 1002  | 89734      | 134.99 |               | 1002  | 9.4.2006 12:31 | -100.21 |
| ...   |       |            | ...    |               | 1001  | 6.5.2007 14:15 | 452.15  |
|       |       |            |        |               | ...   | ...            | ...     |

- potrebno je utvrditi koji događaji mogu uzrokovati neispravnu vrijednost atributa stanje u relaciji racun, te pod kojim uvjetima treba obaviti koje akcije kako bi se očuvao integritet podataka, npr.
  - događaj: obavljanje operacije **INSERT** nad relacijom **uplatalsplata**
  - uvjet: **iznos <> 0.00**
  - akcija: pribrojiti vrijednost atributa **iznos** unesene n-torke u odgovarajuće stanje

## Primjer 3 (nastavak):

- događaj: obavljanje operacije INSERT nad relacijom uplatalsplata
- uvjet:iznos <> 0.00
- akcija: pribrojiti vrijednost atributa iznos unesene n-torke u odgovarajuće stanje

```
CREATE TRIGGER insUplataIsplata
 INSERT ON uplataIsplata
 REFERENCING NEW AS novaUplataIsplata
 FOR EACH ROW
 WHEN (novaUplataIsplata.iznos <> 0)
 (UPDATE racun SET stanje = stanje + novaUplataIsplata.iznos
 WHERE brRac = novaUplataIsplata.brRac);
```

- kad god se obavi naredba INSERT nad relacijom uplatalsplata SUBP obavlja
  - nakon unosa svake n-torke (jednom INSERT naredbom može se unijeti više n-torki) provjerava uvjet `novaUplataIsplata.iznos <> 0`
  - na sadržaj unesene n-torke može se referencirati koristeći "ime" n-torke koje je zadano pomoću `REFERENCING NEW AS novaUplataIsplata`
  - ako je uvjet zadovoljen (za dotičnu n-torku), obavlja izmjenu stanja u relaciji racun

## Primjer 3 (nastavak):

- događaj: brisanje n-torke iz relacije uplatalsplata
- uvjet: iznos <> 0.00
- akcija: oduzeti vrijednost atributa iznos unesene n-torke od odgovarajućeg stanja

```
CREATE TRIGGER delUplataIsplata
 DELETE ON uplataIsplata
 REFERENCING OLD AS brisanaUplataIsplata
 FOR EACH ROW
 WHEN (brisanaUplataIsplata.iznos <> 0)
 (UPDATE racun SET stanje = stanje - brisanaUplataIsplata.iznos
 WHERE brRac = brisanaUplataIsplata.brRac);
```

- ukoliko je potrebno, moguće je navesti više SQL naredbi, međusobno odijeljenih zarezima
- SQL naredbe koje se mogu koristiti za opisivanje akcije:
  - INSERT
  - UPDATE
  - DELETE
  - EXECUTE PROCEDURE

## Primjer 3 (nastavak):

- događaj: izmjena vrijednosti atributa iznos u relaciji uplatalsplata
- uvjet:nova vrijednost iznosa <> stara vrijednost iznosa
- akcija: u odgovarajuće stanje pribrojiti razliku između nove i stare vrijednosti atributa iznos

```
CREATE TRIGGER updIznosUplataIsplata
 UPDATE OF iznos ON uplataIsplata
 REFERENCING OLD AS staraUplataIsplata NEW AS novaUplataIsplata
 FOR EACH ROW
 WHEN (novaUplataIsplata.iznos <> staraUplataIsplata.iznos)
 (UPDATE racun SET stanje = stanje +
 novaUplataIsplata.iznos - staraUplataIsplata.iznos
 WHERE brRac = staraUplataIsplata.brRac);
```

- **UPDATE OF iznos ON uplataIsplata:** događaj izmjene vrijednosti atributa iznos u relaciji uplatalsplata
- **UPDATE OF a, b, c ON relacija:** događaj izmjene vrijednosti bilo kojeg od atributa a, b, c u relaciji
- **UPDATE ON relacija:** događaj izmjene vrijednosti bilo kojeg atributa u relaciji

# Naredba CREATE TRIGGER

---

- oblik naredbe za kreiranje okidača propisan je SQL standardom, ali SUBP koriste uglavnom vlastite inačice
- jedna od važnijih mogućnosti koje su na raspolaganju pri definiciji okidača:
  - moguće je specificirati da li se akcije navedene u okidaču obavljaju:
    - po jednom za svaku n-torku na koju je djelovala operacija koja je aktivirala okidač (operacija koja je uzrokovala događaj)
      - FOR EACH ROW
    - samo jednom, nakon što se obavi operacija koja je aktivirala okidač
      - AFTER INSERT, AFTER UPDATE, AFTER DELETE
    - samo jednom, prije nego se obavi operacija koja je aktivirala okidač
      - BEFORE INSERT, BEFORE UPDATE, BEFORE DELETE
- uništavanje okidača: DROP TRIGGER *imeOkidača*

# Primjena okidača

---

- implementacija integritetskih ograničenja
  - okidače treba koristiti onda kada integritetska ograničenja nije moguće opisati na drugi način (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, ...)
    - obavljanjem korektivne akcije koja bazu podataka dovodi u konzistentno stanje (primjer 3)
    - odbijanjem operacije koja narušava integritetsko ograničenje (primjer 4)
- praćenje rada korisnika (primjer 5)
- sustavi obavještavanja (primjer 6)
- itd.

## Primjer 4:

---

- u relaciji ispit osigurati integritetsko ograničenje prema kojem je promjena ocjena dopuštena samo ako se mijenja na nižu ocjenu, npr.
  - dopušteno je ocjenu izvrstan promijeniti u dobar
  - nije dopušteno ocjenu dovoljan promijeniti u vrlo dobar

| ispit | matBr | sifPred | datIsp     | ocj | sifNast |
|-------|-------|---------|------------|-----|---------|
| 100   | 1001  |         | 29.06.2006 | 3   | 1111    |
| 100   | 1001  |         | 05.02.2006 | 1   | 3333    |
| 101   | 1002  |         | 27.06.2006 | 2   | 2222    |
| 102   | 1001  |         | 29.01.2006 | 1   | 2222    |

- očito je da ne postoji korektivna akcija koja bi bazu podataka mogla dovesti u konzistentno stanje nakon što korisnik obavi naredbu:

```
UPDATE ispit SET ocjena = 2
WHERE ocjena = 1;
```

- jedini način na koji se može osigurati navedeno integritetsko ograničenje jest: odbiti izvršavanje takve naredbe

## Primjer 4 (nastavak):

```
CREATE PROCEDURE dojaviPogreskuUvecanjeOcjene ()
 RAISE EXCEPTION -746, 0, 'Ocjena se ne smije uvećati';
END PROCEDURE;

CREATE TRIGGER updOcjIspit
 UPDATE OF ocj ON ispit
 REFERENCING OLD AS stariIspit NEW AS noviIspit
 FOR EACH ROW
 WHEN (noviIspit.ocj > stariIspit.ocj)
 (EXECUTE PROCEDURE dojaviPogreskuUvecanjeOcjene());
```

- što se dešava pri izvršavanju naredbe 

UPDATE ispit SET ocj = 2 WHERE matBr = 100;
- nakon promjene prve n-torke, akcije iz okidača se neće obaviti jer uvjet za obavljanje akcije nije ispunjen
- nakon promjene druge n-torke, aktivirat će se akcija iz okidača
  - poziva se procedura
  - procedura signalizira pogrešku
  - budući da se naredba mora obaviti u cijelosti ili uopće ne, sustav poništava i promjenu prve n-torke, a korisniku prikazuje opis pogreške

## Primjer 5:

- prepostavi li se da je izmjena podataka u relaciji racun naročito osjetljiva operacija - potrebno je pratiti rad korisnika (*audit trail*)

```
CREATE TABLE auditTrailZaRacun (
 korisnik CHAR(32)
, vrijeme DATETIME YEAR TO SECOND
, brRac1 ...
, sifKlijent1 ...
, stanje1 ...
, brRac2 ...
, sifKlijent2 ...
, stanje2 ...
);
```

```
CREATE TRIGGER updRacun
 UPDATE ON racun
 REFERENCING OLD AS stari NEW AS novi
 FOR EACH ROW
 -- uvjet se može ispuštiti
 (INSERT INTO auditTrailZaRacun VALUES (
 USER, CURRENT, stari.brRac, stari.sifKlijent, stari.stanje,
 novi.brRac, novi.sifKlijent, novi.stanje));
```

## Primjer 5 (nastavak):

```
...
FOR EACH ROW
(INSERT INTO auditTrailZaRacun VALUES (
 USER, CURRENT, stari.brRac, stari.sifKlijent, stari.stanje,
 novi.brRac, novi.sifKlijent, novi.stanje));
```

- što se dešava obavljanjem naredbe

novak

```
UPDATE racun
SET stanje = stanje + 10
WHERE brRac BETWEEN 1002 AND 1003;
```

| racun |            |         |
|-------|------------|---------|
| brRac | sifKlijent | stanje  |
| 1001  | 98281      | 216.80  |
| 1002  | 89734      | 134.99  |
| 1003  | 23232      | 2750.00 |
| 1004  | 63443      | 849.50  |

- osim promjene u relaciji racun, u relaciju auditTrailZaRacun bit će dodane dvije n-torce

auditTrailZaRacun

| korisnik | vrijeme             | brRac1 | sifKlijent1 | stanje1 | brRac2 | sifKlijent2 | stanje2 |
|----------|---------------------|--------|-------------|---------|--------|-------------|---------|
| ...      | ...                 | ...    | ...         | ...     | ...    | ...         | ...     |
| novak    | 2007.02.27 14:13:47 | 1002   | 89734       | 134.99  | 1002   | 89734       | 144.99  |
| novak    | 2007.02.27 14:13:47 | 1003   | 23232       | 2750.00 | 1003   | 23232       | 2760.00 |

## Primjer 6:

- postoji pohranjena procedura saljiPostu(adresa, tekst)
- u relaciji artikel nalaze se podaci o artiklima na skladištu. Za svaki artikel prati se trenutno stanje (količina) artikla
- kada stanje artikla padne ispod optimalne količine, potrebno je na e-mail adresu djelatnika zaduženog za nabavu tog artikla poslati poruku

| artikel | sifArt | stanje | optKol | adresaZaduzenog |
|---------|--------|--------|--------|-----------------|
|         | 1001   | 250    | 150    | pero@tvrtka.hr  |
|         | 1002   | 400    | 200    | joza@tvrtka.hr  |
|         | 1003   | 450    | 350    | jura@tvrtka.hr  |

```
CREATE TRIGGER updArtikel
 UPDATE OF stanje ON artikel
 REFERENCING OLD AS stari NEW AS novi
 FOR EACH ROW
 WHEN (stari.stanje >= stari.optKol
 AND novi.stanje < stari.optKol)
 (EXECUTE PROCEDURE saljiPostu(stari.adresaZaduzenog
 , 'Nabavi artikl: ' || stari.sifArt));
```

## Primjer 6 (nastavak):

| artikl | sifArt | stanje | optKol | adresaZaduzenog |
|--------|--------|--------|--------|-----------------|
|        | 1001   | 250    | 150    | pero@tvrtka.hr  |
|        | 1002   | 400    | 200    | joza@tvrtka.hr  |
|        | 1003   | 450    | 350    | jura@tvrtka.hr  |

- rezultat obavljanja naredbe

```
UPDATE artikl
SET stanje = stanje - 150;
```

artikl

| artikl | sifArt | stanje | optKol | adresaZaduzenog |
|--------|--------|--------|--------|-----------------|
|        | 1001   | 100    | 150    | pero@tvrtka.hr  |
|        | 1002   | 250    | 200    | joza@tvrtka.hr  |
|        | 1003   | 300    | 350    | jura@tvrtka.hr  |

- + dvije poruke

pero@tvrtka.hr: Nabavi artikl: 1001  
jura@tvrtka.hr: Nabavi artikl: 1003

- ako se nakon toga obavi naredba

```
UPDATE artikl
SET stanje = stanje - 100;
```

artikl

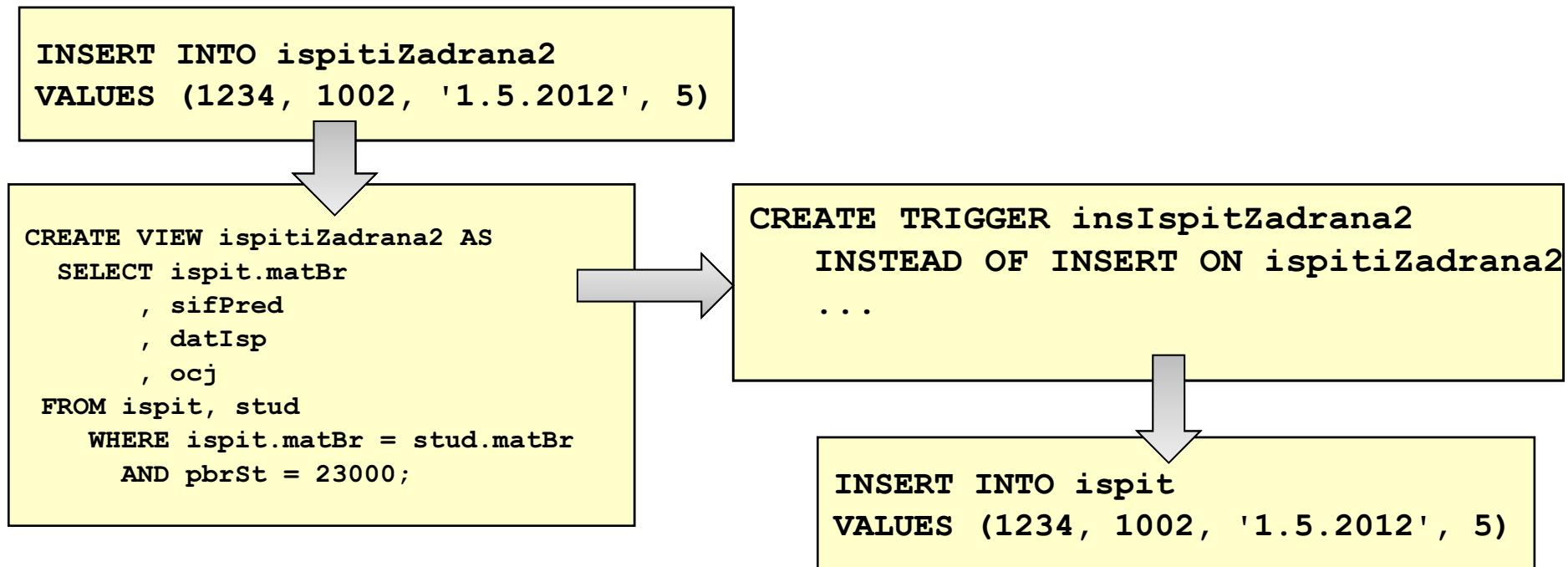
| artikl | sifArt | stanje | optKol | adresaZaduzenog |
|--------|--------|--------|--------|-----------------|
|        | 1001   | 0      | 150    | pero@tvrtka.hr  |
|        | 1002   | 150    | 200    | joza@tvrtka.hr  |
|        | 1003   | 200    | 350    | jura@tvrtka.hr  |

- + poruka

joza@tvrtka.hr: Nabavi artikl: 1002

# Dodatno: INSTEAD OF okidači

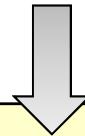
- Mehanizam kojim se svaka virtualna relacija može učiniti izmjenjivom
  - definira se koje operacije treba napraviti u temeljnim relacijama umjesto zadanih operacija nad virtualnom relacijom
- Nije u skladu sa SQL standardom, ali je podržan u mnogim sustavima
- Npr. za neizmjenjivu virtualnu relaciju `ispitiZadrana2`:



## Dodatno: INSTEAD OF okidači

---

- Posebni okidači se pišu za INSERT, UPDATE i DELETE
- Djelovanje okidača mora biti u skladu s definicijom virtualne relacije
- Npr. je li prije unosa n-torke s prethodne strane obavljena provjera da li je student 1234 iz Zadra?



```
INSERT INTO ispit
VALUES (1234, 1002, '1.5.2012', 5)
```

# Dodatno: INSTEAD OF okidači

```
CREATE PROCEDURE insIspitZad(matBr int, sifPred int, datIsp DATE, ocj SMALLINT)
 IF ((SELECT pbrSt FROM stud WHERE matBr = matBr)= 23000) THEN
 INSERT INTO ispit VALUES (matBr, sifPred, datIsp, ocj);
 ELSE
 RAISE EXCEPTION -746, 0, 'Student ne postoji ili nije iz Zadra.';
 END IF;
END PROCEDURE;
```

```
CREATE TRIGGER insIspitZadrana2
INSTEAD OF INSERT ON ispitiZadrana2
REFERENCING NEW AS noviIspit
FOR EACH ROW
(EXECUTE PROCEDURE
 insIspitZad(noviIspit.matBr, noviIspit.sifPred, noviIspit.datIsp, noviIspit.ocj)
)
```

- IBM Informix ne podržava WHEN u INSTEAD OF okidaču

# KRATKI PRIRUČNIK ZA SPL

# Kreiranje pohranjene procedure

---

- Procedura se kreira SQL naredbom oblika:

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
 tijelo procedure
END PROCEDURE ;
```

```
CREATE FUNCTION imeFunkcije (eventualni argumenti)
 tijelo funkcije
END FUNCTION ;
```

- Eventualne pogreške u sintaksi naredbe sustav će dojaviti za vrijeme obavljanja naredbe (na isti način kao i pogreške za vrijeme obavljanja ostalih SQL naredbi).
- Brisanje (uništavanje) procedure

```
DROP PROCEDURE imeProcedure ;
DROP FUNCTION imeFunkcije ;
```

- Izmjena procedure: brisanjem starog objekta i definiranjem novog objekta pod istim imenom, npr.

```
DROP PROCEDURE imeProcedure ;
CREATE PROCEDURE imeProcedure . . . ;
```

# Struktura pohranjene procedure

---

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
 definicija varijabli
 naredba;
 naredba;
 ...
END PROCEDURE;
```

```
CREATE FUNCTION imeProcedure (eventualni argumenti)
 definicija varijabli
 naredba;
 naredba;
 ...
END FUNCTION;
```

- Naredbe procedure završavaju znakom ; (točka-zarez)

# Definicija varijabli

---

- Definicije varijabli se navode na početku procedure. Sadržaj varijable je nedefiniran dok mu se ne pridruži neka vrijednost.
- Tipovi varijabli mogu biti definirani eksplisitno:

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
 DEFINE ime CHAR(20);
 DEFINE ocjena, brojIzlazaka SMALLINT;
 ...
```

- ili implicitno, prema tipovima atributa u relacijama baze podataka

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
 DEFINE ime LIKE student.imeStud;
 DEFINE ocjena LIKE ispit.ocjena;
 ...
```

- kad god je moguće, tipove varijabli treba definirati implicitno
  - u slučaju promjene tipa podatka nekog atributa u relaciji, sve što je potrebno obaviti jest ponovo prevesti procedure

# Naredba LET

---

- koristi se za pridruživanje vrijednosti varijablama

```
CREATE PROCEDURE ...
 DEFINE r, povrsina DECIMAL(10,5);
 DEFINE brojIspita SMALLINT;
 DEFINE sumaOcjena INTEGER;
 DEFINE prosjek DECIMAL(3,2);
 DEFINE brojZnam SMALLINT;

 LET r = 10;
 LET povrsina = 3.14159 * r * r;

 LET brojIspita = (SELECT COUNT(*) FROM ispit);
 LET sumaOcjena = (SELECT SUM(ocjena) FROM ispit);
 LET prosjek = sumaOcjena/brojIspita;
 LET brojZnam = brojZnamenki('123abc');

 ...
```

- rezultat obavljanja SELECT naredbe koja vraća jednu jednostavni vrijednost (skalar) može se koristiti na svim mjestima na kojima se koriste izrazi. SELECT naredba mora biti unutar okruglih zagrada

# Naredbe IF, WHILE, FOR

- naredba za jednostranu, dvostranu ili višestranu selekciju
- naredba za realizaciju petlje s ispitivanjem uvjeta na početku
- naredba za realizaciju petlje s unaprijed utvrđenim brojem ponavljanja

```
IF uvjet THEN
 naredbe
ELIF uvjet THEN
 naredbe
ELIF uvjet THEN
 naredbe ...
ELSE
 naredbe
END IF;
```

```
WHILE uvjet
 naredbe
 ...
 EXIT WHILE; ----->
 CONTINUE WHILE; ----->
END WHILE;
```

```
FOR i = m TO n STEP k
 naredbe
 ...
 EXIT FOR; ----->
 CONTINUE FOR; ----->
END FOR;
```

kao break u jeziku C  
kao continue u jeziku C

kao break u jeziku C  
kao continue u jeziku C

# SQL naredbe u pohranjenim procedurama

- U pohranjenim procedurama mogu se koristiti (gotovo) sve do sada prikazane SQL naredbe (izuzetak je npr. DROP DATABASE)

```
...
DELETE FROM stud
 WHERE prezStud LIKE 'Z%';
UPDATE stud SET pbrMjestoStan = 10000
 WHERE pbrMjestoStan = 41000;
INSERT INTO mjesto VALUES (31000, 'Osijek');
...
```

- Rezultat SELECT naredbe može se pohraniti u varijable, npr.

```
DEFINE v_imeStud LIKE student.imeStud;
DEFINE v_presStud LIKE student.presStud;
...
SELECT imeStud, prezStud
 INTO v_imeStud, v_presStud
 FROM student
 WHERE mbrStud = 12345;
```

- broj i tipovi varijabli moraju odgovarati broju i tipovima izraza iz liste za selekciju
- SELECT naredba smije vratiti samo jednu n-torku

# Uporaba varijabli u SQL naredbama

- varijable se slobodno mogu koristiti na svim mjestima na kojim se u SQL naredbama koriste izrazi, npr.
  - u izrazima u SELECT listi, u WHERE dijelu SQL naredbe
  - u VALUES listi INSERT naredbe
  - u izrazima u SET dijelu UPDATE naredbe, ...

```
CREATE PROCEDURE ...
 DEFINE iznos, koef DECIMAL (3,2);
 DEFINE datum DATE;
 DEFINE s INTEGER;
 DEFINE n CHAR(20);
 LET koef = (SELECT MAX(koef) FROM nastavnik);
 LET s = 100; LET n = 'Primorsko-goranska';
 LET datum = TODAY - 365*20;
 SELECT AVG(ocjena) * koef INTO iznos FROM ispiti
 WHERE datIspit = datum;
 UPDATE stud SET datRodStud = datum
 WHERE datRodStud <> datum;
 INSERT INTO zupanija VALUES(s, n);
 ...
```

# Argumenti pohranjene procedure

---

```
CREATE PROCEDURE imeProcedure (imeArg tip, imeArg tip, ...)
```

- tipovi podataka ulaznih argumenata procedure mogu, kao i varijable, biti definirani eksplisitno ...

```
CREATE FUNCTION povrsina (sirina INTEGER, visina INTEGER)
...
```

- ili implicitno ...

```
CREATE PROCEDURE postaviAdresu (p_mbrStud LIKE stud.mbrStud
, p_adresa LIKE stud.adresa)
...
```

- jednako kao u drugim programskim jezicima, argumenti se u tijelu procedure/funkcije mogu koristiti na jednak način kao i varijable

# Rezultati funkcije

---

- tipovi rezultata koje funkcija vraća moraju se deklarirati

```
CREATE FUNCTION imeFunkcije (imeArg tip, imeArg tip, ...)
 RETURNING INTEGER AS ime, CHAR(20) AS ime, DATE AS ime
 DEFINE ...
 ...
END FUNCTION;
```

- tipovi rezultata mogu se deklarirati jedino eksplisitno (nije moguće koristiti oblik "LIKE atribut" kao pri definiciji argumenata ili varijabli)
- "ime" rezultata se navodi opcionalno: korisno je deklarirati ime rezultata jer se npr. pri pozivu funkcije iz interaktivnog alata rezultat prikazuje zajedno s deklariranim imenom

# Povrat rezultata funkcije u pozivajući program

- koristi se naredba RETURN slična naredbi RETURN u ostalim programskim jezicima. RETURN naredba se u tijelu procedure može pojaviti više puta. Naredbom je u pozivajući program moguće vratiti jednu ili više vrijednosti

```
CREATE FUNCTION opsegPovrsina(radijus DECIMAL(10,5))
 RETURNING DECIMAL(10,5) AS opseg
 , DECIMAL(10,5) AS povrsina
 DEFINE o, p DECIMAL(10,5);
 LET o = 2 * radijus * 3.14159;
 LET p = radijus * radijus * 3.14159;
 RETURN o, p;
END FUNCTION;
```

```
EXECUTE FUNCTION opsegPovrsina(4.5);
```

| opseg    | povrsina |
|----------|----------|
| 28.27431 | 63.61720 |

# Načini poziva procedure (funkcije)

---

- Iz interaktivnih alata, npr. Server Studio

```
EXECUTE PROCEDURE prebaci (1001, 1002, 60.30);
```

- procedura ne vraća rezultat (eventualno signalizira pogrešku)

```
EXECUTE FUNCTION opsegPovrsina(4.5);
```

- funkcija vraća rezultat (eventualno signalizira pogrešku)

| opseg    | povrsina |
|----------|----------|
| 28.27431 | 63.61720 |

# Načini poziva procedure (funkcije)

---

- Iz pohranjene procedure ili funkcije

```
CREATE PROCEDURE x (...)
 DEFINE brojZnam ...
 DEFINE opseg, povrsina ...
 ...
 -- procedure ne vraćaju rezultat
 EXECUTE PROCEDURE prebaci (1001, 1002, 60.30);
 ...
 -- funkcije koje vraćaju jednu vrijednost
 LET brojZnam = brojZnamenki('abc123');
 CALL brojZnamenki('abc123') RETURNING brojZnam;
 ...
 -- funkcije koje vraćaju više vrijednosti
 CALL opsegPovrsina(4.5) RETURNING opseg, povrsina;
```

# Načini poziva funkcije

---

- Korištenje funkcija u SQL naredbama
  - funkcije koje vraćaju točno jednu vrijednost mogu se u SQL naredbama koristiti na svim mjestima na kojima se mogu koristiti ugrađene SQL funkcije

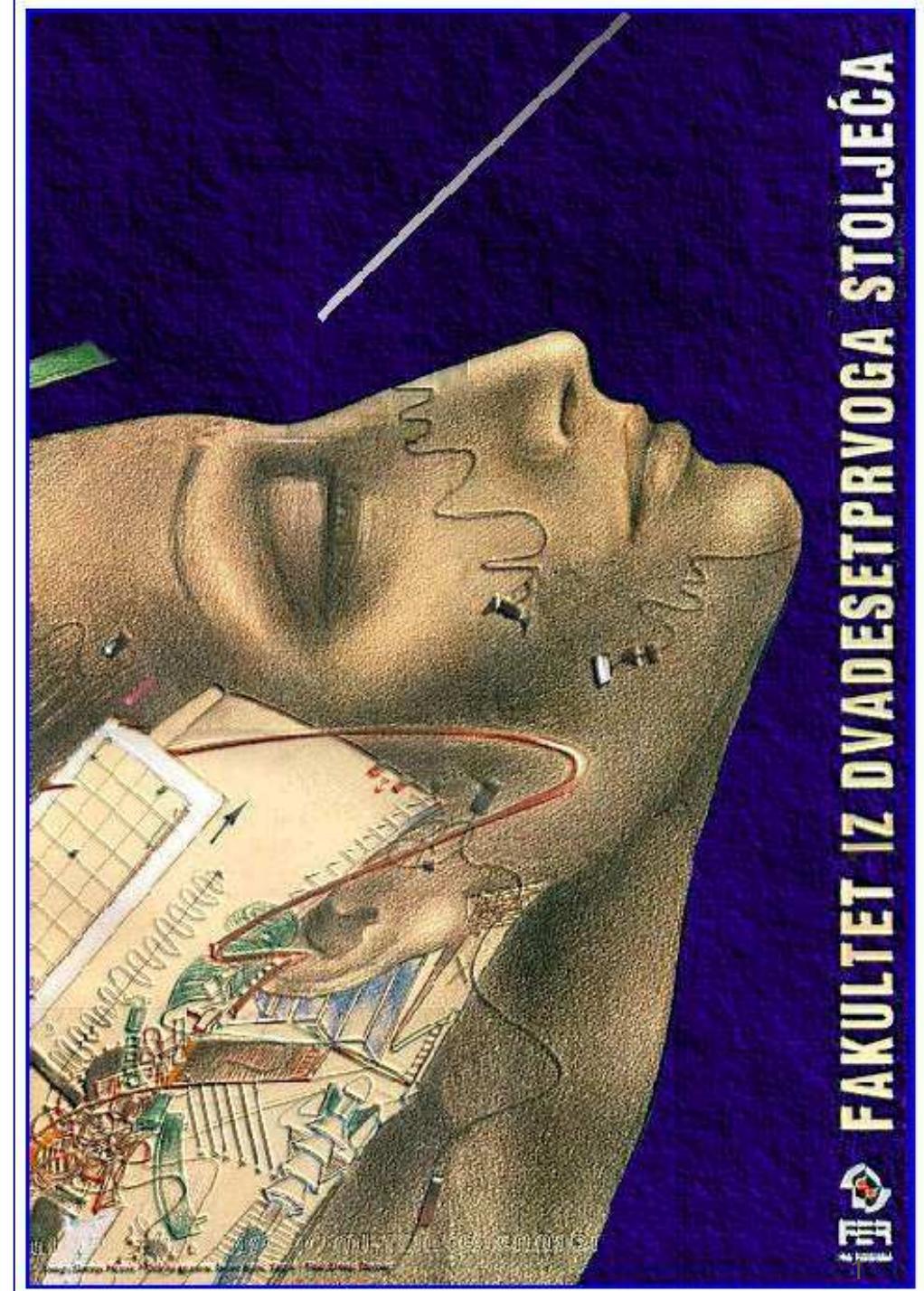
```
SELECT * , brojZnamenki(adresa) AS brojZnam
 FROM osoba
 WHERE brojZnamenki(adresa) > 0;

DELETE FROM osoba
 WHERE brojZnamenki(jmbg) <> 13;
```

# Baze podataka

Predavanja  
Svibanj 2014.

## 13. Optimiranje upita



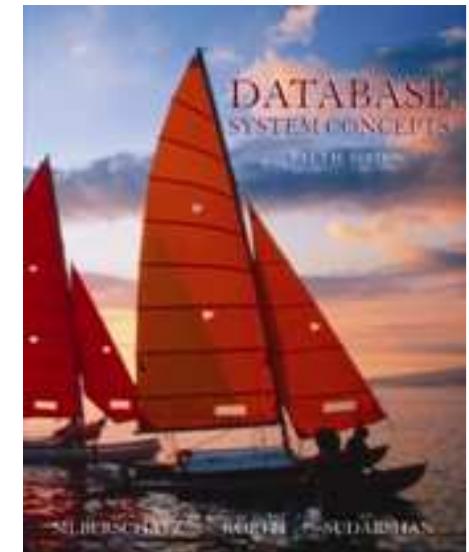
# Obavljanje upita (engl. *Query Processing*)

---

- niz aktivnosti potrebnih da bi se dohvatio podatke iz baze podataka
  - translacija upita pisanih u jeziku visoke razine u izraze koji se mogu primijeniti na fizičku razinu
  - optimiranje upita koje uključuje različite transformacije
  - izvršavanje upita

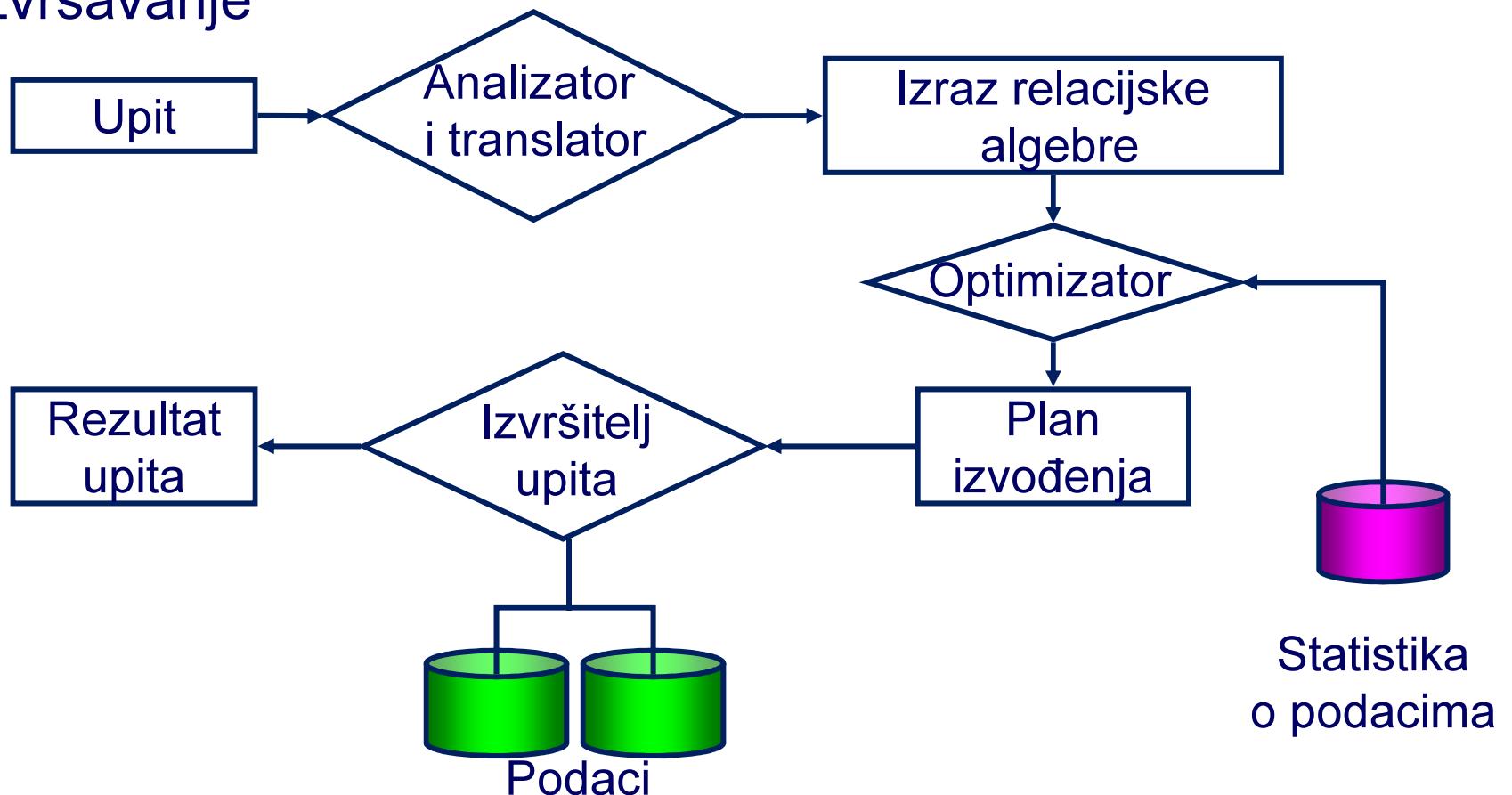
Literatura:

A. Silberschatz, H. Korth, S. Sudarshan,  
*Database System Concepts*, McGraw-Hill, 2005



# Osnovni koraci pri obavljanju upita

1. Analiza (parsiranje) i translacija
2. Optimiranje (engl. Query Optimization)
3. Izvršavanje



# Procjena troškova - uloga rječnika podataka

---

Optimizator upita mora na raspolaganju imati određene podatke o relacijama koje sudjeluju u upitu.

- To su npr.
  - $N(r)$  - broj n-torki u relaciji
  - informacija o tome da li atribut sadrži jedinstvene vrijednosti
  - koji indeksi su izgrađeni nad relacijom, radi li se o uzlazno ili silazno poredanom indeksu, radi li se o *Cluster indexu* (podaci u podatkovnim blokovima su poredani prema ključu)
  - dubina B-stabla za indeks
  - $V(A, r)$  - broj različitih vrijednosti atributa A u relaciji r
    - $V(A, r) = G_{COUNT(*)}(\pi_A(r))$
    - A može biti skup atributa
  - broj jedinstvenih vrijednosti u indeksu
  - distribucija vrijednosti u pojedinim atributima relacije

# Procjena troškova - uloga rječnika podataka

---

- Optimiranje bi postalo izuzetno neefikasno kada bi se ovi statistički podaci prikupljali iz baze podataka prilikom optimiranja svakog upita.
- Postoji posebna naredba kojom se pokreće prikupljanje ovih podataka, a rezultati se zapišu u rječnik podataka
  - IBM IDS: administrator obavlja naredbu: **UPDATE STATISTICS**
- Odgovornost je administratora baze podataka da uvijek kada se bitno promijene podaci koji bi mogli utjecati na rad optimizatora podataka (npr. u relaciju se upiše relativno veliki broj zapisa) izvede naredbu za ažuriranje rječnika podataka.
- U suprotnom, moglo bi se dogoditi da optimizator, temeljeći svoje odluke na pogrešnim statističkim podacima, generira neoptimalne planove obavljanja.
- Danas svi sustavi imaju ugrađenu i automatiku (koja se može i isključiti) za periodičko ili *on-event* aktiviranje ažuriranja statistike.

# Analiza (parsiranje) i translacija

---

- Pretvorba upita pisanog u upitnom jeziku (SQL) u izraz relacijske algebre
- Zamjena virtualnih relacija s temeljnim relacijama koje proizlaze iz definicije virtualne relacije

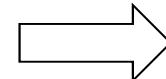
**Primjer:**

FILMSKA ZVIJEZDA = {ime, adresa, spol, datrod}  $K_{FILMSKAZVIJEZDA} = \{ime\}$

GLUMIU = {naslov, godina, imeZvijezda}  $K_{GLUMIU} = \{naslov, godina, imeZvijezda\}$

```
SELECT *
 FROM filmskaZvijezda, glumiU
 WHERE imeZvijezda = ime
 AND godina = 2012
 AND spol = 'M'
```

analiza i translacija u početni izraz relacijske algebre



# Translacija u početni izraz relacijske algebre

FILMSKA ZVIJEZDA = {ime, adresa, spol, datrod}  $K_{FILMSKA ZVIJEZDA}$  = {ime}

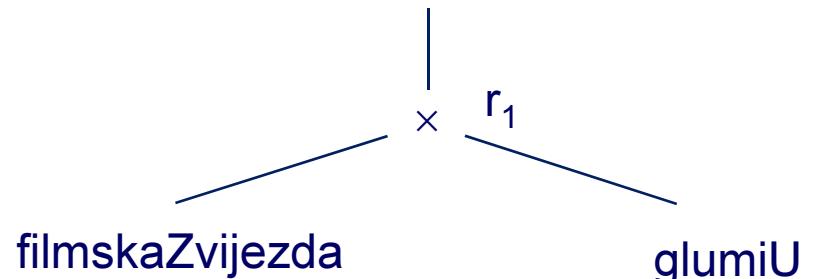
GLUMIU = {naslov, godina, imeZvijezda}  $K_{GLUMIU}$  = {naslov, godina, imeZvijezda}

```
SELECT *
 FROM filmskaZvijezda, glumiU
 WHERE imeZvijezda = ime
 AND godina = 2012
 AND spol = 'M'
```

izraz relacijske algebre:

$$\sigma_{\text{godina}=2012 \text{ AND } \text{spol}='M' \text{ AND } \text{imeZvijezda} = \text{ime}} (\text{filmskaZvijezda} \times \text{glumiU})$$

- stablo upita (*query tree*) je način prikaza izraza relacijske algebre
- listovi stabla su relacije, a ostali čvorovi su operacije relacijske algebre

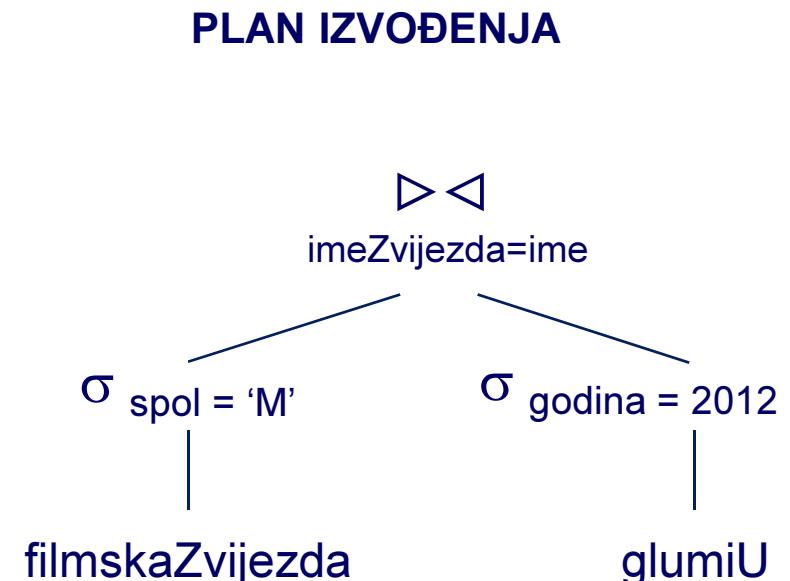
$$\sigma_{\text{godina} = 2012 \text{ AND } \text{spol} = 'M' \text{ AND } \text{imeZvijezda} = \text{ime}}$$


$$\text{card}(r_1) = \text{card}(\text{filmskaZvijezda}) * \text{card}(\text{glumiU})$$

# Plan izvođenja (engl. Execution Plan)

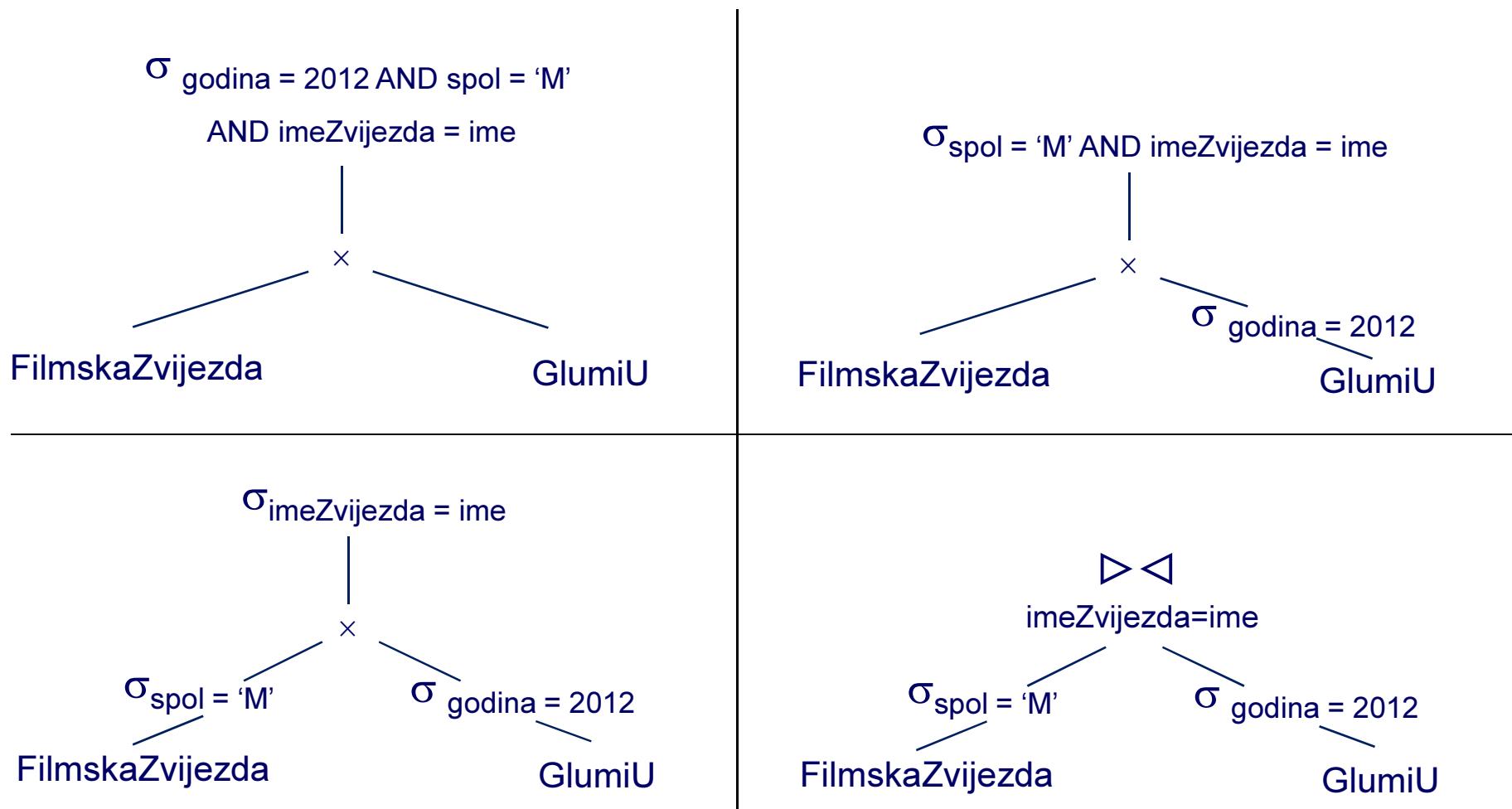
---

- Odabir operanada i operacija ( $\sigma$ ,  
     $\bowtie$ , ...)
- Redoslijed izvođenja operacija
- Detaljni plan izvođenja operacija
  - metode pristupa podacima
  - redoslijed spajanja
  - metode spajanja
  - stvaranje privremenih indeksa?
  - sortiranje privremenih rezultata
  - ....



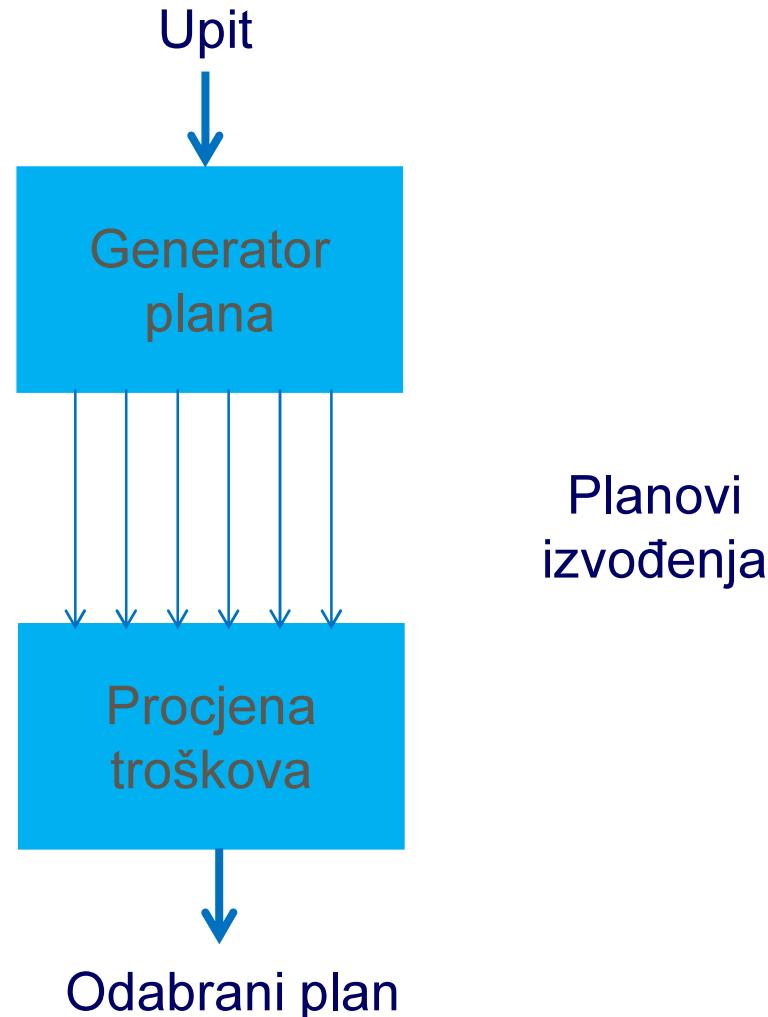
# Plan izvođenja nije jednoznačno određen upitom

U općem slučaju se izraz relacijske algebre može zamijeniti ekvivalentnim, alternativnim izrazom relacijske algebre (jednim od mnogih):



# Optimiranje upita

Proces odabira *najprikladnijeg* od mogućih planova izvršavanja.



# Ekvivalentni izrazi relacijske algebre

---

- Različiti planovi izvođenja dobiju se za
  - različiti odabir operacija
  - različiti redoslijed obavljanja operacija
  - različit odabir metoda izvršavanja pojedinih operacija
  - ...
- Dva izraza relacijske algebre su ekvivalentni ako primijenjeni nad svakom instancom baze podataka daju jednaki rezultat
- Alternativni izrazi se određuju na temelju pravila za transformaciju izraza relacijske algebre

# Algebarske transformacije

---

- Prirodno spajanje
  - komutativno:  $r \bowtie s = s \bowtie r$
  - asocijativno:  $(r \bowtie s) \bowtie t = r \bowtie (s \bowtie t)$
- Operacije  $\times$ ,  $\cup$ ,  $\cap$  su komutativne i asocijativne
- $\theta$ -spajanje nije uvijek asocijativno! Primjer: relacije  $r(A,B)$ ,  $s(B,C)$ ,  $t(C,D)$ 
$$(r \bowtie_{\theta} s) \bowtie t \neq r \bowtie_{\theta} (s \bowtie t) \rightarrow A \text{ nije atribut iz } s, \text{ niti iz } t !!!$$
$$\begin{array}{c} r.B > s.B \\ A < D \end{array} \quad \begin{array}{c} r.B > s.B \\ A < D \end{array}$$
- $\theta$ -spajanje  $(r \bowtie_{\theta_1} s) \bowtie_{\theta_2} t$  može biti asocijativno  
ako se uvjet  $\theta_2$  podijeli na  $\theta_3$  i  $\theta_4$  tako da  $\theta_3$  uključuje samo attribute iz  $s$  i  $t$  :

$$(r \bowtie_{\theta_1} s) \bowtie_{\theta_2} t = r \bowtie_{\theta_1 \wedge \theta_4} (s \bowtie_{\theta_3} t)$$

# Pravila koja se odnose na selekciju

---

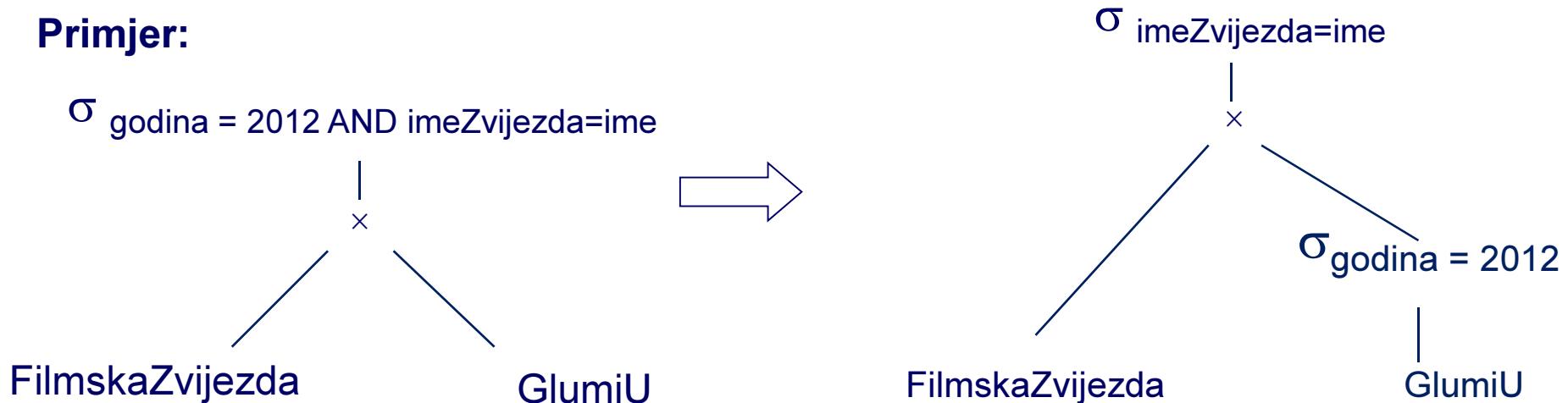
- Podjela:
  - $\sigma_{C1 \text{ AND } C2}(r) = \sigma_{C1}(\sigma_{C2}(r)) = \sigma_{C2}(\sigma_{C1}(r))$
  - $\sigma_{C1 \text{ OR } C2}(r) = \sigma_{C1}(r) \cup \sigma_{C2}(r)$
- Potiskivanje selekcije
  - **ako je uvjet C primjenjiv samo na r:**
$$\sigma_C(r \bowtie s) = (\sigma_C(r)) \bowtie s$$
  - **ako je uvjet C primjenjiv samo na s:**
$$\sigma_C(r \bowtie s) = r \bowtie (\sigma_C(s))$$
  - **ako je uvjet C primjenjiv na r i na s:**
$$\sigma_C(r \bowtie s) = (\sigma_C(r)) \bowtie (\sigma_C(s))$$
- Na isti se način selekcija može potiskivati u odnosu na Kartezijev produkt i spajanje uz uvjet
- U ovim predavanjima su navedena samo neka od pravila algebarskih transformacija (projekcija, grupiranje,...)

# Heuristička pravila

Nije moguće generirati i analizirati sve moguće planove izvođenja (preskupo). Broj planova koji će se procjenjivati reducira se pomoću heurističkih pravila.

**1. Potiskivanje selekcije:** operaciju selekcije obaviti u čim ranijoj fazi.

**Primjer:**

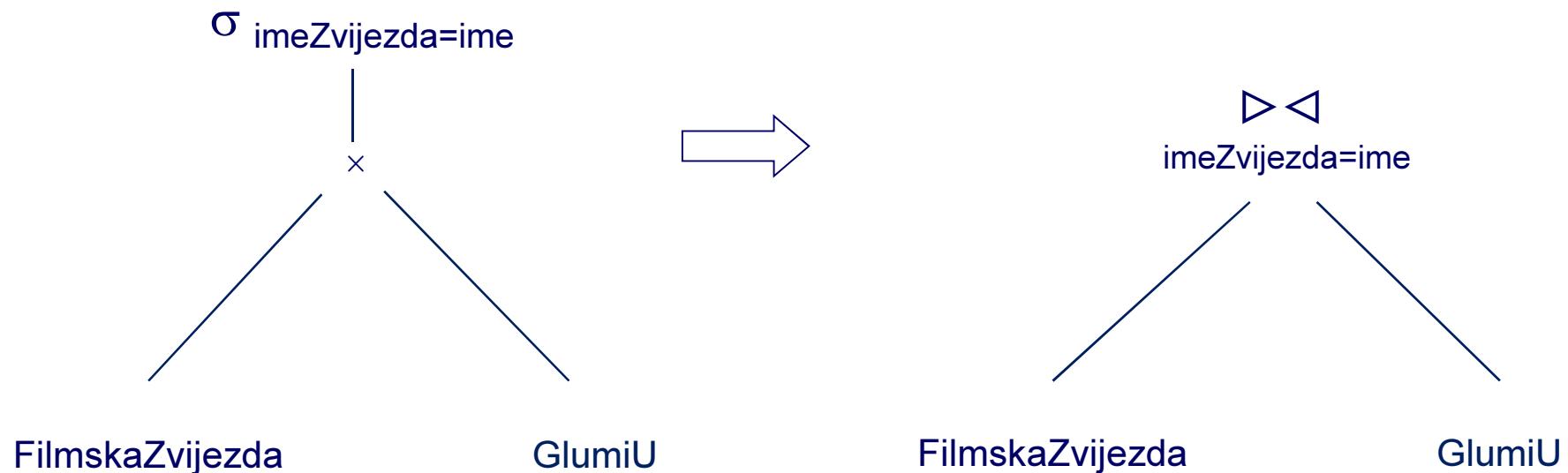


- **Potiskivanje selekcije je jedno od heurističkih pravila koja se koriste pri optimizaciji.**
- **Heuristička pravila**
  - proizlaze iz iskustva, temelje se na ekvivalentnim izrazima relacijske algebre
  - njihova primjena u većini slučajeva rezultira efikasnijim planom izvršavanja

# Heuristička pravila

## 2. Kombiniranje operacije selekcije i Kartezijevog produkta

Primjer:



- u ovim predavanjima navedena su tek dva od mnogih heurističkih pravila

# Izračunavanje troška izvršavanja operacija

---

- Primjenom heurističkih pravila ne dobiva se uvijek jednoznačan i/ili uvijek najbolji mogući plan
- Za svaki od dobivenih planova izvođenja procjenjuje se ukupni trošak izvršavanja. Odabire se plan s najmanjim procijenjenim troškom (*cost based optimization*)
- Najveći dio troška izvršavanja upita odnosi se na trošak obavljanja U/I operacija
- Važna je veličina međurezultata!
- Veličina međurezultata ovisi o
  - broju n-torki u međurezultatu (može se samo procijeniti)
  - veličini n-torke u međurezultatu (dobije se izravno iz metapodataka)

# Procjena veličine rezultata selekcije

---

- Jednostavna pretpostavka:
  - Jednolika razdioba vrijednosti atributa A

- Selekcija uz uvjet  $A = c$ :

$$N(\sigma_{A=c}(r)) = N(r) / V(A, r)$$

- ako vrijednost  $c$  uopće ne postoji???

- Selekcija koja uključuje nejednakost  $\sigma_{A < c}(r)$ 
  - pretpostavka - 1/3 n-torki zadovoljava uvjet:

$$N(\sigma_{A < c}(r)) = N(r) / 3$$

- Složeni uvjeti:  $\sigma_{A=c1 \text{ AND } B < c2}(r)$ :

$$\begin{aligned} N(\sigma_{A=c1 \text{ AND } B < c2}(r)) &= N(\sigma_{A=c1}(\sigma_{B < c2}(r))) \\ &= (N(r) / 3) / V(A, r) = N(r) / (3 * V(A, r)) \end{aligned}$$

# Procjena veličine rezultata spajanja

---

- Neka je  $t = r \triangleright \triangleleft s$ . Neka su  $X$  i  $Y$  skupovi atributa iz  $r$ , odnosno  $s$
- 1.  $X \cap Y = \emptyset$  Spajanje je produkt -  $N(t) = N(r) * N(s)$ 
  - nema eliminacije duplikata!
- 2.  $X \cap Y$  je ključ od  $r$  - svaka  $n$ -torka iz  $s$  može se spojiti s najviše jednom  $n$ -torkom iz  $r$  :  $N(t) = N(s)$
- 3.  $X \cap Y$  nije prazan skup, nije ključ od  $r$  ili  $s$ , neka je  $A = X \cap Y$ 
  - Svaka  $n$ -torka iz  $r$  spaja se s  $N(s) / V(A, s)$   
$$N(t) = N(r) * N(s) / V(A, s)$$
  - ili ako krenemo s  $n$ -torkama iz  $s$   $N(t) = N(r) * N(s) / V(A, r)$
  - Ako je  $V(A, s) \neq V(A, r)$ , tada se računa s većim od mogućih nazivnika:  
$$N(t) = N(r) * N(s) / \max(V(A, r), V(A, s))$$

## Primjer:

- Važna primjena procjene veličine je procjena planova u kojima se operandi spajaju različitim poretkom

$$r(A, B)$$

$$N(r) = 1000$$

$$V(B, r) = 20$$

$$s(B, C)$$

$$N(s) = 2000$$

$$V(B, s) = 50$$

$$V(C, s) = 100$$

$$t(C, D)$$

$$N(t) = 5000$$

$$V(C, t) = 500$$

Obavlja se operacija:  $r \triangleright\triangleleft s \triangleright\triangleleft t$

- optimizator ne raspolaže informacijom o ključevima relacija

1. Prvo se spajaju **r** i **s**:

$$N(r \triangleright\triangleleft s) = 1000 \times 2000 / \max(20, 50) = 40.000$$

$$V(C, r \triangleright\triangleleft s) = 100$$

$$N((r \triangleright\triangleleft s) \triangleright\triangleleft t) = 40.000 \times 5000 / \max(100, 500) = \mathbf{400.000}$$

|                |                 |                 |
|----------------|-----------------|-----------------|
| $r(A, B)$      | $s(B, C)$       | $t(C, D)$       |
| $N(r) = 1000$  | $N(s) = 2000$   | $N(t) = 5000$   |
| $V(B, r) = 20$ | $V(B, s) = 50$  | $V(C, t) = 500$ |
|                | $V(C, s) = 100$ |                 |

2. Prvo se spajaju **s** i **t**:

$$N(s \triangleright\!\!< t) = 2000 \times 5000 / \max(100, 500) = 20.000$$

$$V(B, s \triangleright\!\!< t) = 50$$

$$N(r \triangleright\!\!< (s \triangleright\!\!< t)) = 1000 \times 20.000 / \max(20, 50) = \mathbf{400.000}$$

3. Prvo se spajaju **r** i **t**:

$$N(r \triangleright\!\!< t) = 1000 \times 5000 = 5.000.000$$

$$V(B, r \triangleright\!\!< t) = 20$$

$$V(C, r \triangleright\!\!< t) = 500$$

$$N((r \triangleright\!\!< t) \triangleright\!\!< s) = 5.000.000 \times 2000 / (\max(20, 50) \times \max(100, 500)) = \mathbf{400.000}$$

## Redoslijed spajanja

---

- procijenjena veličina je 400,000 n-torki bez obzira na redoslijed spajanja → Slučajnost?
- za stvaranje i rukovanje međurezultatima potrebni su značajni resursi
- **kriterij odabira redoslijeda spajanja operanada je veličina međurezultata**

# Metode pristupa podacima u relaciji (*access plan*)

---

- Čitanje n-torki:
  - slijednim čitanjem blokova podataka (*table-scan, sequential scan*)
  - korištenjem indeksa
    - *key-only index scan*
    - *index scan*

# Metode pristupa podacima u relaciji

Slijedno čitanje podataka iz relacije (*table scan*)

- koristi se kad nema 'upotrebljivog' indeksa prema kojem bi se obavljala selekcija ili kada ionako treba pročitati sve ili vrlo velik broj n-torki iz relacije

```
CREATE TABLE stud (
 mbrStud INTEGER
, prezStud CHAR(25)
, imeStud CHAR(25)
);
CREATE INDEX prez_ime ON stud (prezStud, imeStud)
```

```
SELECT * FROM stud WHERE prezStud = 'Horvat'
```

- za selekciju se koristi indeks *prez\_ime*

```
SELECT * FROM stud WHERE imeStud = 'Ivo'
```

- indeks *prez\_ime* nije upotrebljiv

# Čitanje pomoću indeksa (*index scan*)

---

```
CREATE INDEX prez_ime ON stud (prezStud, imeStud)
```

- **key-only index scan**

ako su svi podaci koji se čitaju iz relacije dijelovi JEDNOG indeksa, sve potrebne vrijednosti se mogu pronaći u indeksnim blokovima

```
SELECT imeStud, prezStud
 FROM stud
 WHERE prezStud = 'Horvat'
```

- **index scan**

ako se svi potrebni podaci ne mogu naći u indeksu, **mora se pristupati podatkovnim blokovima**

```
SELECT mbrStud, imeStud, prezStud
 FROM stud
 WHERE prezStud = 'Horvat'
```

# Metode spajanja relacija (*join*)

---

- Prikazat će se samo neke metode spajanja relacija:
  - Spajanje ugniježđenim petljama (*nested loop join*)
  - Raspršeno spajanje (*hash join*)

# Spajanje ugniježđenim petljama

---

## Postupak:

- relacije koje se spajaju se nazivaju vanjska relacija (outer table) i unutarnja relacija (inner table). (Ovi pojmovi nemaju veze s vanjskim spajanjem (outer join) u relacijskoj algebri!)
- iz vanjske relacije se čita svaka n-torka
  - ako postoji uvjet selekcije, čitaju se samo one n-torke koje zadovoljavaju uvjet. Pri tome, ukoliko postoji upotrebljiv indeks za obavljanje uvjeta selekcije, kao metoda za pristup n-torkama iz vanjske relacije koristi se index scan ili key-only index scan
- za svaku pročitanu n-torku iz vanjske relacije traže se n-torke iz unutarnje relacije koje zadovoljavaju uvjet spajanja

# Spajanje ugniježđenim petljama

---

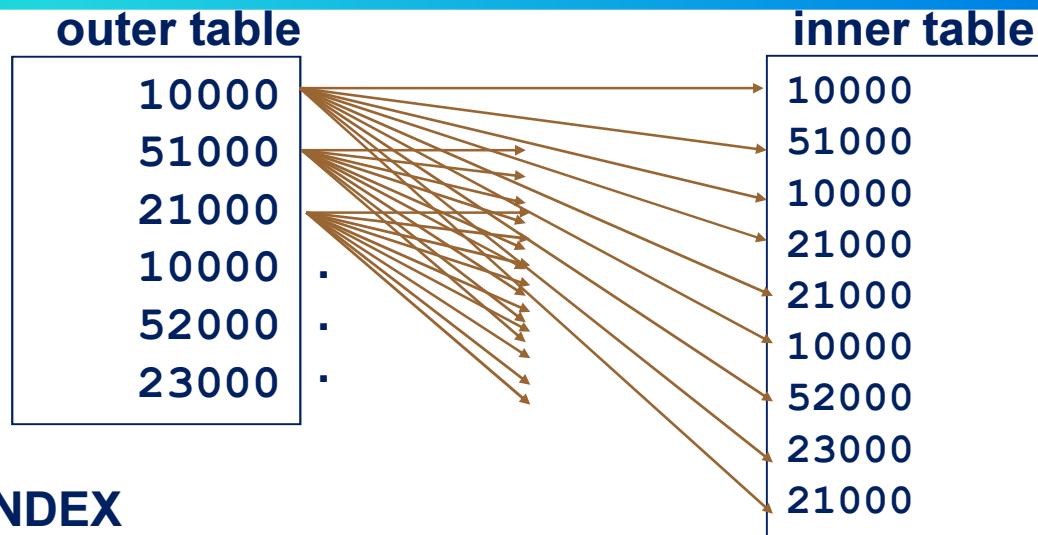
**SUBP mora za svaku n-torku iz vanjske relacije po jednom pretražiti cijelu unutarnju relaciju**

**Način pristupa podacima iz unutarnje relacije:**

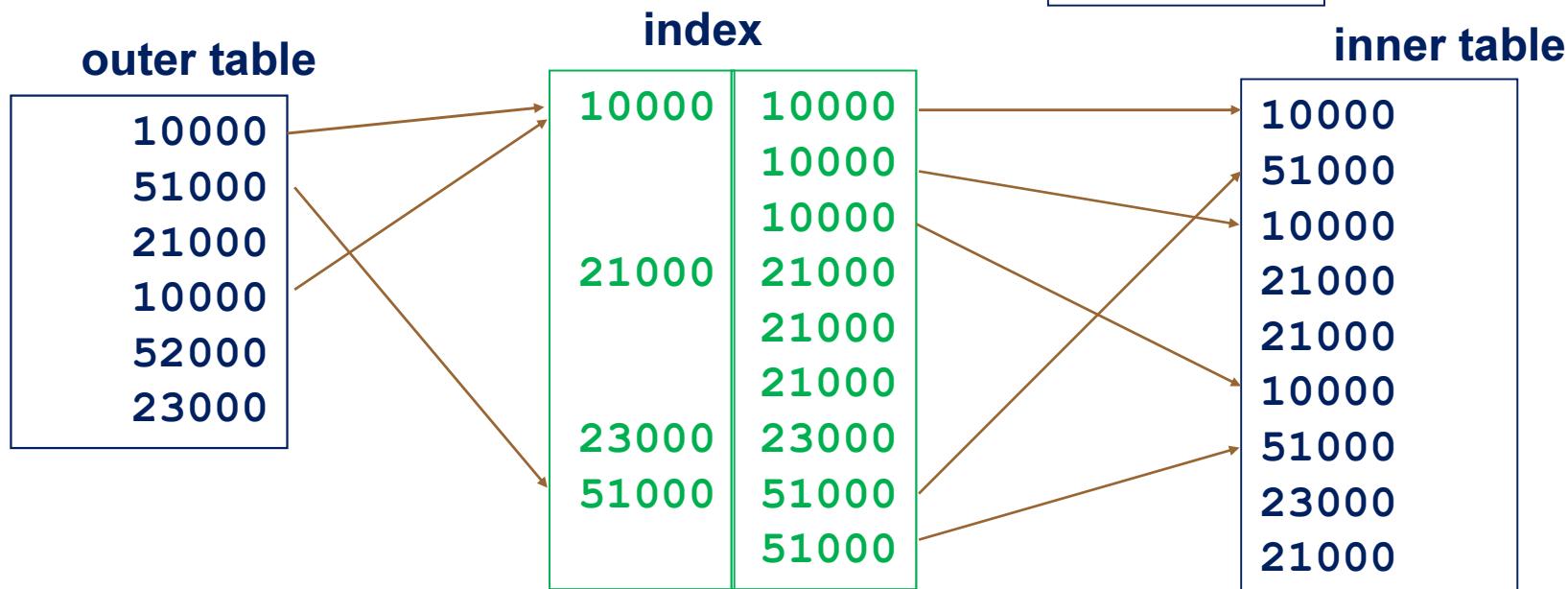
- a) ako se u unutarnjoj relaciji nalazi mali broj n-torki može se koristiti slijedno pretraživanje
- b) ukoliko postoji upotrebljivi indeks nad unutarnjom relacijom, za pretragu se koristi taj indeks
- c) ukoliko upotrebljivi indeks za pretragu unutarnje relacije ne postoji, a relacija sadrži veliki broj n-torki, optimizator može procijeniti da je isplativije potrošiti određeno vrijeme na izgradnju privremenog indeksa nego veliki broj puta slijedno pretraživati relaciju. U planu obavljanja se takav način pristupa označava kao **autoindex-path**

# Spajanje ugniježđenim petljama

SLIJEDNO



INDEX ili AUTO-INDEX



# Raspršeno spajanje

---

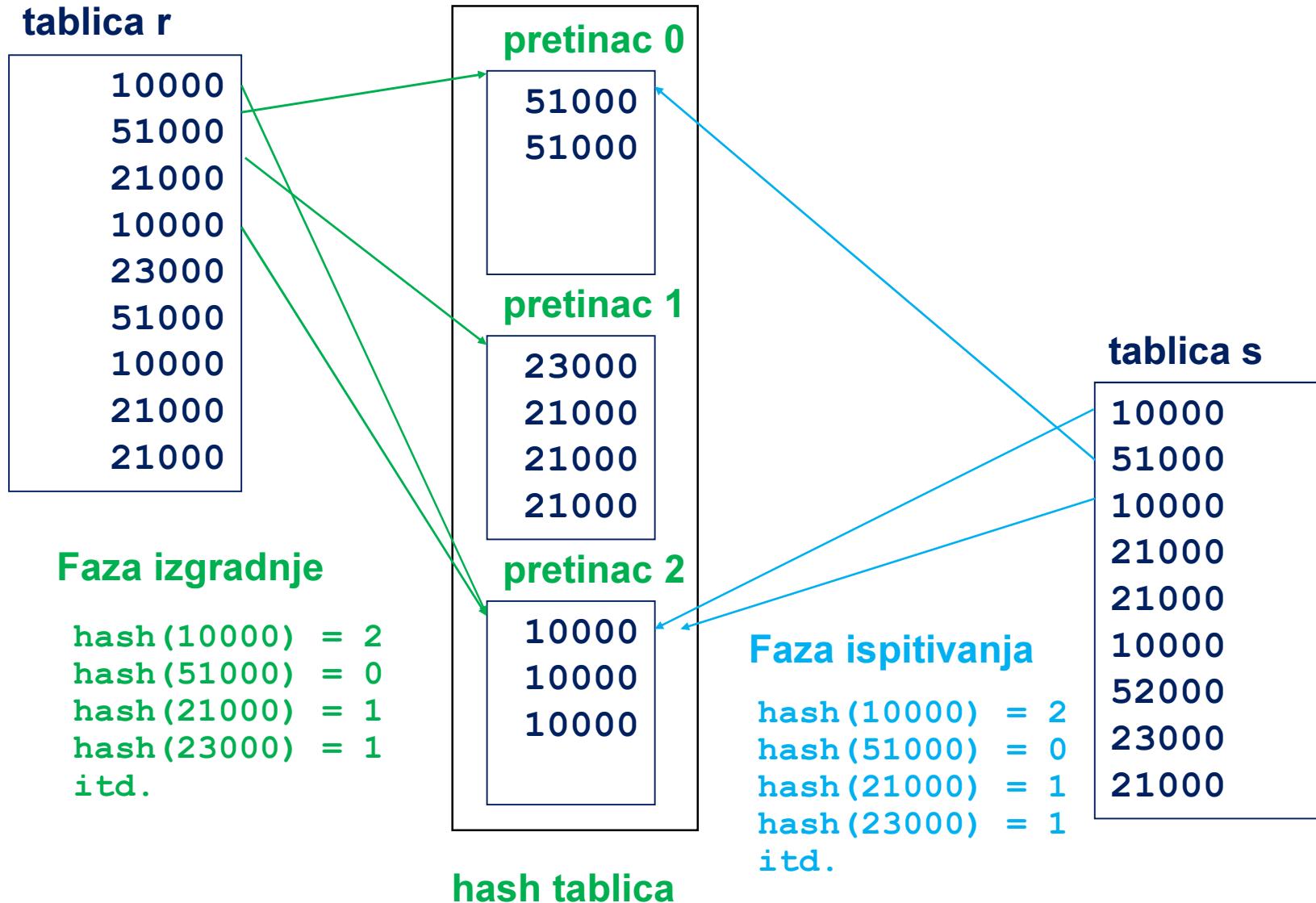
## Faza izgradnje:

- Raspršena tablica se izgrađuje za samo jednu relaciju iz para kojeg treba spojiti (npr. relaciju  $r$  iz para relacija  $r$  i  $s$ ).
- Funkcija raspršenja se primjenjuje na vrijednosti atributa ili vrijednosti skupa atributa prema kojima se obavlja spajanje.
- U obzir se uzimaju samo one n-torce iz relacije  $r$  koje zadovoljavaju eventualno postavljeni uvjet selekcije
- pri tome se za njihovo izdvajanje, ako je moguće, primjenjuje indeks, slično kao kod čitanja n-torki iz vanjske relacije kod spajanja s ugniježđenom petljom.

## Faza ispitivanja:

- Čita sadržaj relacije  $s$  (uzimaju se samo n-torce koje zadovoljavaju eventualni uvjet selekcije, ukoliko je moguće koristi se indeks)
- Za svaku pročitanu n-torku pomoću funkcije raspršenja, a na temelju vrijednosti atributa iz relacije  $s$  prema kojima se obavlja spajanje, izračuna se u kojem se džepu raspršene tablice nalaze zapisi iz relacije  $r$  s kojima se ta n-torka treba spojiti.

# Raspršeno spajanje



# Primjer

Zadane su relacije *mjesto*, *stud* i *ispit* sa sljedećim shemama:

$$MJESTO = \{pbr, nazMjesto\}$$

$$N(mjesto) = 500$$

$$V(nazMjesto, mjesto) = 500$$

$$STUD = \{mbrStud, prezStud, pbr\}$$

$$N(stud) = 10\,000$$

$$V(prezStud, stud) = 1000$$

$$K_{MJESTO} = \{pbr\}$$

$$K_{STUD} = \{mbrStud\}$$

$$ISPIT = \{mbrStud, sifPred, datRok, ocjena\}$$

$$N(ispit) = 100\,000$$

$$V(ocjena, ispit) = 5$$

$$K_{ISPIT} = \{mbrStud, sifPred, datRok\}$$

Indeks je kreiran samo nad atributom *prezStud* relacije *stud*.

Optimizator ima na raspolaganju sljedeće metode:

- pristup podacima u relaciji bez indeksa (*sequential scan*)
- pristup podacima u relaciji preko indeksa (*index path*)
  
- raspršeno spajanje (*hash join*)
- spajanje pomoću ugniježđene petlje (*nested-loop join*)

# Primjer

---

Za upit

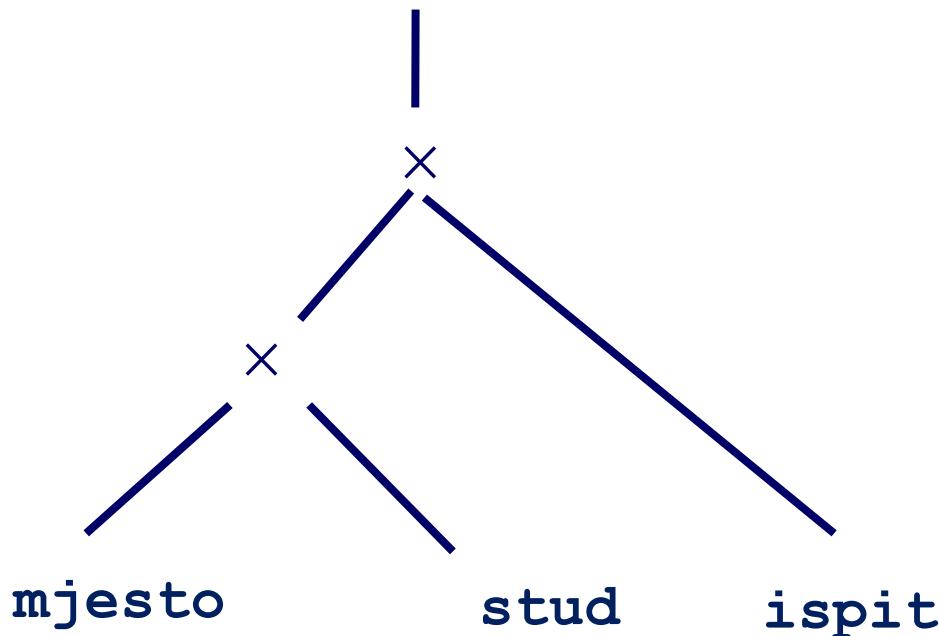
```
SELECT * FROM mjesto, stud, ispit
 WHERE mjesto.pbr = stud.pbr
 AND stud.mbrStud = ispit.mbrStud
 AND stud.prezStud = 'Horvat'
 AND ispit.ocjena = 5
```

- Nacrtati stablo upita za početni plan izvođenja upita pri čemu je redoslijed spajanja relacija određen redoslijedom kojim su relacije navedene u FROM dijelu SELECT naredbe.
- Nacrtati stablo upita nakon provedene heurističke optimizacije. Procijeniti broj n-torki u međurezultatima. U stablu upita naznačiti korištene metode pristupa podacima.
- Procijeniti broj n-torki za različite moguće redoslijede spajanja međurezultata.

## Primjer – a)

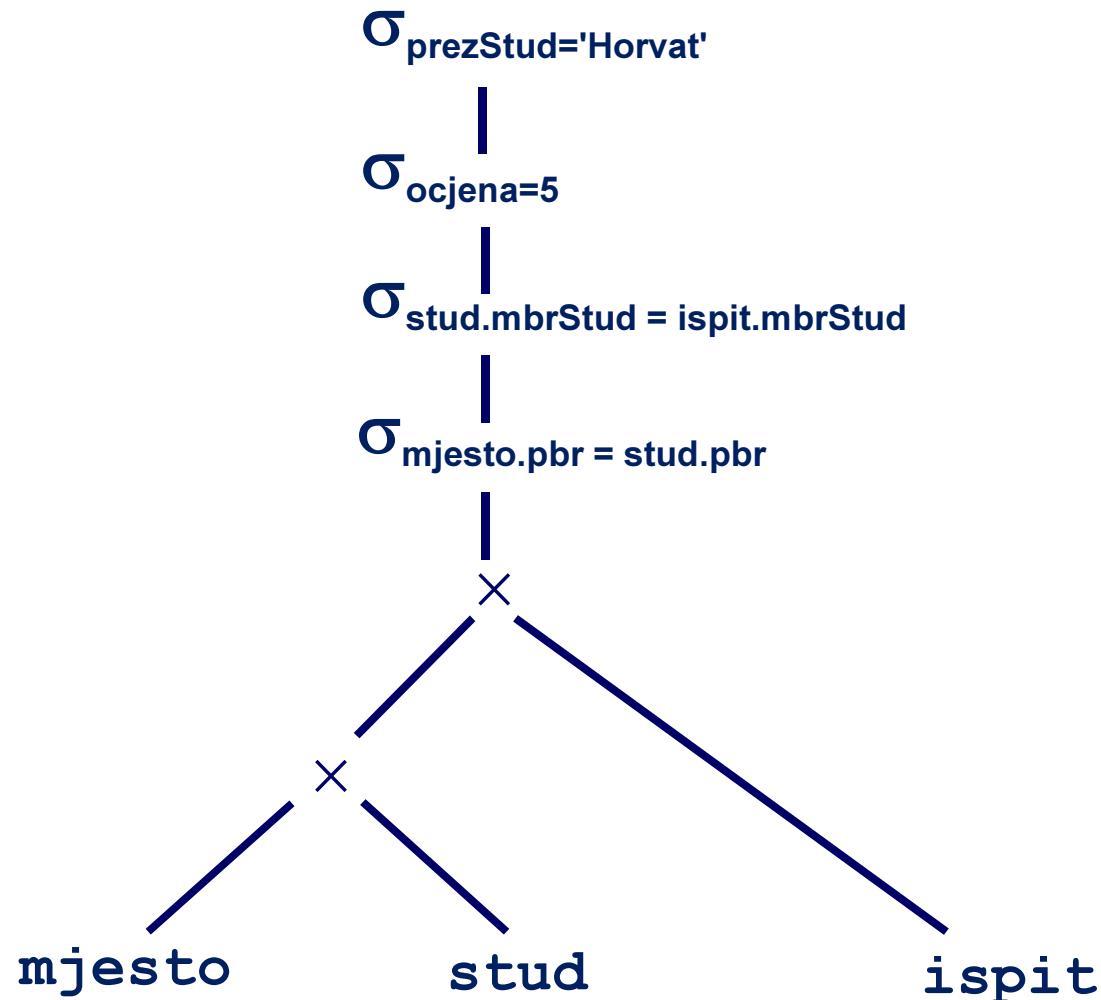
```
SELECT * FROM mjesto, stud, ispit
 WHERE mjesto.pbr = stud.pbr
 AND stud.mbrStud = ispit.mbrStud
 AND stud.prezStud = 'Horvat'
 AND ispit.ocjena = 5
```

$\sigma_{\text{prezStud}='Horvat' \text{ AND } \text{ocjena}=5 \text{ AND } \text{mjesto.pbr} = \text{stud.pbr} \text{ AND } \text{stud.mbrStud} = \text{ispit.mbrStud}}$



## Primjer – b)

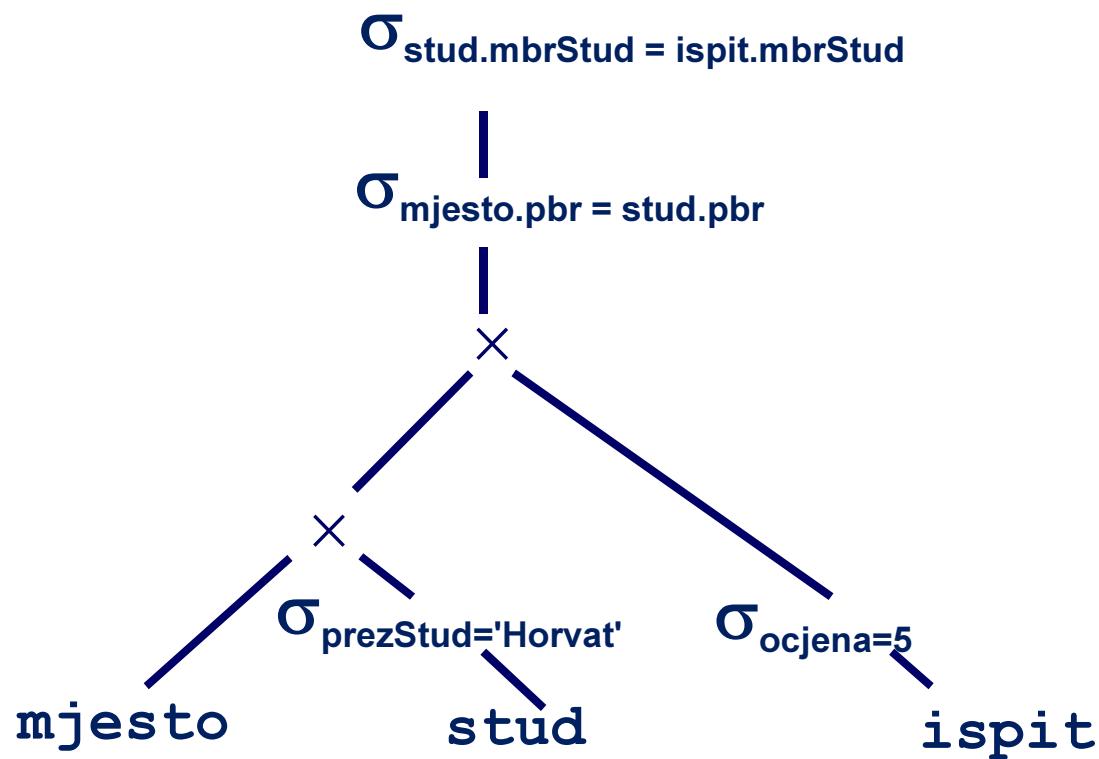
Rastavljanje uvjeta selekcije:



## Primjer – b)

---

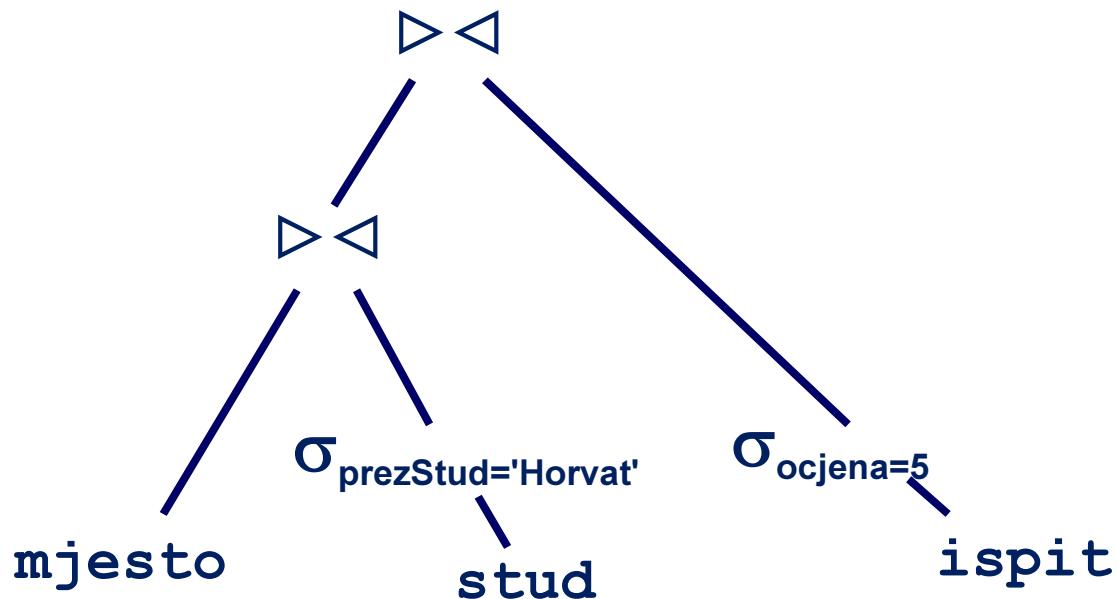
Potiskivanje selekcije:



## Primjer – b)

---

Kombiniranje operacije selekcije i Kartezijskog produkta



## Primjer – b)

---

Procjena broja n-torki u međurezultatima:

$$r_1 = \sigma_{\text{prezStud}='Horvat'}(\text{stud})$$

$$N(r_1) = N(\text{stud}) / V(\text{prezStud}, \text{stud}) = 10000/1000 = 10$$

$$r_2 = \sigma_{\text{ocjena}=5}(\text{ispit})$$

$$N(r_2) = N(\text{ispit}) / V(\text{ocjena}, \text{ispit}) = 100000/5 = 20000$$

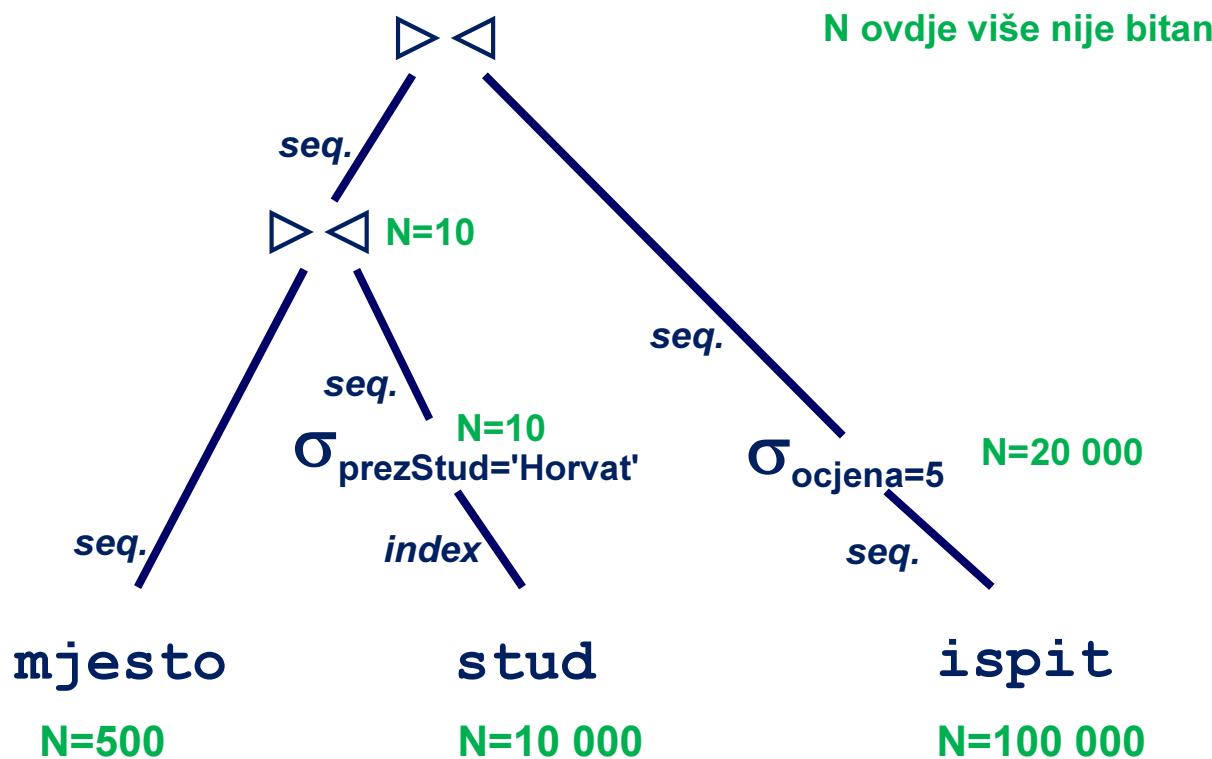
$$r_3 = \text{mjesto} \bowtie r_1$$

$$N(r_3) = N(r_1) = 10 \text{ (mjesto} \cap r_1 \text{ je ključ relacije mjesto)}$$

## Primjer – b)

---

Korištene metode pristupa podacima



## Primjer – c)

---

### Procjena broja n-torki u međurezultatu za različite redoslijede spajanja

$$r_1 = \sigma_{\text{prezStud}='Horvat'}(\text{stud}) \quad N(r_1) = 10$$

$$r_2 = \sigma_{\text{ocjena}=5}(\text{ispit}) \quad N(r_2) = 20000$$

$$N(\text{mjesto} \bowtie r_1) \leq N(\text{stud1}) = 10 \quad (\text{mjesto} \cap r_1 \text{ je ključ relacije mjesto})$$

$$N(\text{mjesto} \bowtie r_2) = 500 * 20\,000 \quad (\text{Kartezijev produkt})$$

$$N(r_1 \bowtie r_2) = N(r_2) = 20\,000 \quad (r_1 \cap r_2 \text{ je ključ relacije stud})$$

Kriterij za određivanje redoslijeda spajanja: veličina međurezultata

$$\Rightarrow (\text{mjesto} \bowtie r_1) \bowtie r_2$$

# Optimiranje upita - analiza plana obavljanja

## IBM Informix

---

Svaki korisnik koji pokrene SQL naredbu može dozнати koji je plan obavljanja upotrijebljen za obavljanje naredbe.

Korisnik postavlja zahtjev SUBP-u da za svaku iduću SQL naredbu ispiše plan obavljanja naredbom:

**SET EXPLAIN ON**

SUBP će prestati ispisivati plan obavljanja nakon što se obavi naredba:

**SET EXPLAIN OFF**

# Optimiranje upita - analiza plana obavljanja

## IBM Informix

---

Nakon što se obavi naredba SET EXPLAIN ON, plan obavljanja svake sljedeće naredbe će biti zapisan u datoteku **sqexplain.out** u *home* kazalu korisnika na računalu na kojem se nalazi SUBP.

Npr. ako ste na bazu **studadmin** spojeni kao korisnik **bpadmin**, datoteka će se nalaziti u kazalu **/usr/bpadmin/** (*home* kazalo korisnika **bpadmin** na virtualnom linux računalu)

**sqexplain.out** je obična tekstualna datoteka čiji sadržaj možete pregledati upotrebom naredbe **cat**, npr. "cat sqexplain.out | more" ili u nekom od instaliranih editora (npr. **vi** ili **joe**)

Zapisi se u ovu datoteku nadopunjuju, tako da opisi starih planova obavljanja ostaju sačuvani.

# Optimiranje upita - analiza plana obavljanja

## IBM Informix

---

- Sadržaj SQL naredbe (prijepis zadane SQL naredbe)
- Procjenu troška obavljanja upita (u apstraktnim jedinicama koje mogu poslužiti za procjenu relativne uspješnosti jednog plana obavljanja u odnosu na drugi plan)
- Procjenu broja n-torki koje se evaluiraju kao rezultat
- Redoslijed pristupa relacijama
- Način pristupa pojedinoj relaciji
  - **SEQUENTIAL SCAN** - bez upotrebe indeksa
  - **INDEX PATH** - pomoću jednog ili više indeksa
  - **AUTOINDEX PATH** - kreira se privremeni indeks
- Popis atributa koji se koriste za uvjet selekcije, uz informaciju da li se selekcija obavlja uz pomoć indeksa

## Sadržaj plana obavljanja – IBM Informix

---

Oznaka **DYNAMIC HASH JOIN** označava da se na prethodno isписаном пару relacija (или пару prethodno dobivenih međurezultata) obavlja raspršeno spajanje.

Oznaka (Build outer) u dijelu specifikacije koja se odnosi na **DYNAMIC HASH JOIN** označava da se raspršena tablica gradi za n-torce iz prve relacije (или međurezultata) iz пара. U suprotnom, raspršena tablica se gradi za n-torce iz druge relacije (или međurezultata) u paru.

Ako se ne spominje **HASH JOIN**, radi se o spajanju ugnježdenim petljama (nested loop join). Ako se za pristup drugoj relaciji koristi:

- **INDEX PATH** - za spajanje se koristi postojeći indeks,
- **AUTO-INDEX PATH** - treba razmisliti treba li kreirati takav indeks, kako se privremeni indeks ne bi kreirao svaki put kad se obavlja ovaj upit.

# Primjer plana obavljanja – IBM Informix

QUERY:

-----

```
SELECT * FROM ispit
 , stud
 , mjesto
 WHERE mjesto.pbr = stud.pbrStan
 AND stud.mbrStud = ispit.mbrStud
```

Estimated Cost: 482

Estimated # of Rows Returned: 2960

1) stud: SEQUENTIAL SCAN

2) mjesto: INDEX PATH

(1) Index Keys: pbr

Lower Index Filter: mjesto.pbr = stud.pbrStan

3) ispit: SEQUENTIAL SCAN

DYNAMIC HASH JOIN (Build Outer)

Dynamic Hash Filters: stud.mbrStud = ispit.mbrStud

# Primjer plana obavljanja – IBM Informix

---

**Procijenjeni trošak: 482**

**Procijenjeni broj n-torki u rezultatu: 2960**

1) **stud**: **SEQUENTIAL SCAN** – slijedno se čitaju sve n-torke iz relacije stud, jer ne postoji nikakav uvjet selekcije

2) **mjesto**: **INDEX PATH**

(1) **Index Keys**: **pbr**

**Lower Index Filter**: **mjesto.pbr = stud.pbrStan**

relacije stud i mjesto spajaju se metodom ugniježđene petlje. Pri tome se za svaku n-torku iz stud, pomoću indeksa dohvaćaju odgovarajuće n-torke u relaciji mjesto

3) **ispit**: **SEQUENTIAL SCAN**

**DYNAMIC HASH JOIN (Build Outer)**

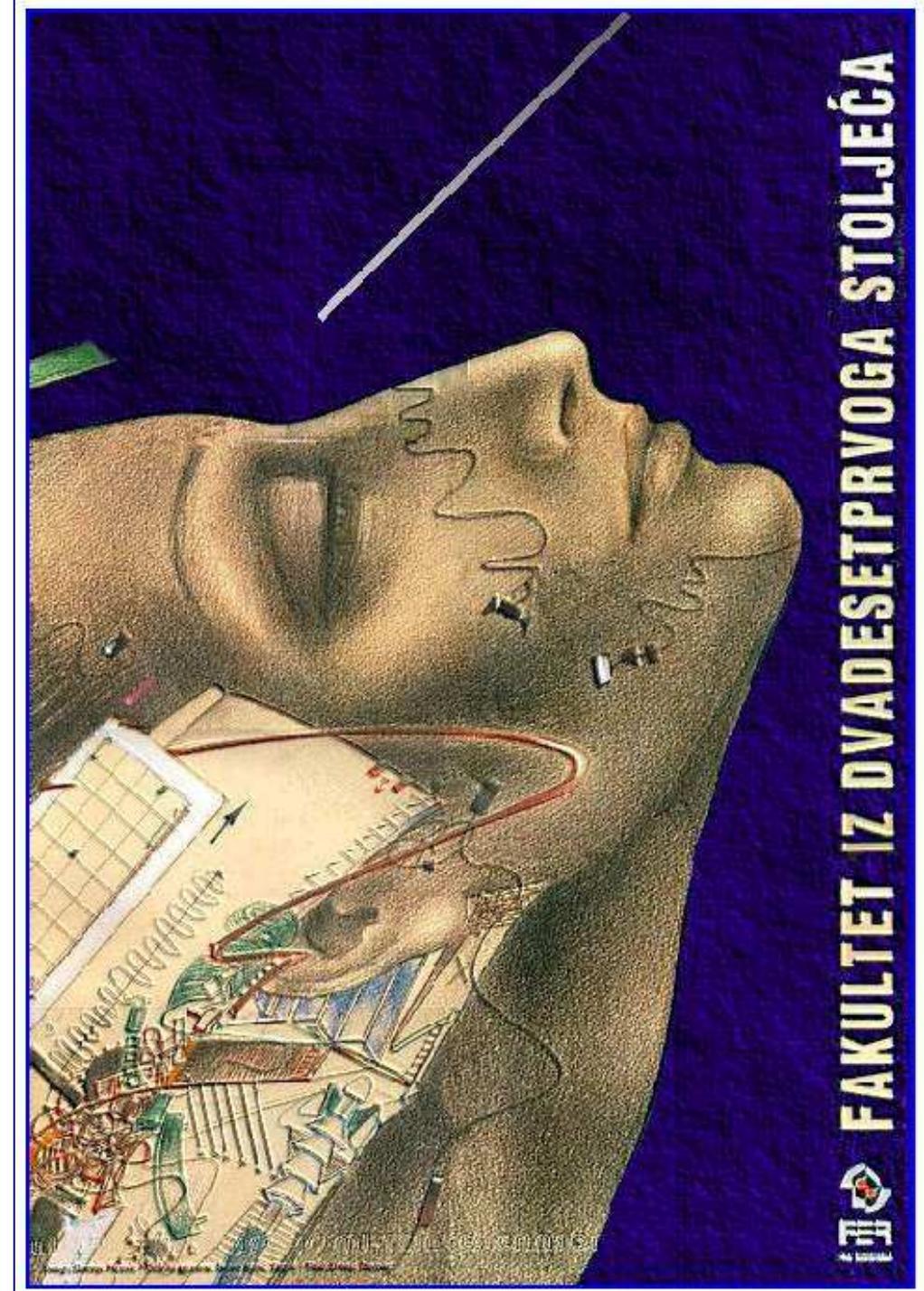
**Dynamic Hash Filters**: **stud.mbrStud=ispit.mbrStud**

Za rezultat iz koraka 2) se napravi raspršena tablica. Slijedno se čitaju zapisi iz relacije ispit (jer ne postoji nikakav uvjet) te se preko hash funkcije dolazi do odgovarajućih zapisa iz koraka 2)

# Baze podataka

Predavanja  
svibanj 2014.

## 14. ER model baze podataka (1. dio)



# Primjer normalizacije

---

- Zadana je relacijska shema:

ISPIT = { matBr, prez, ime, sifPred, nazPred, datlsp, ocj, sifNas, prezNas }

i trenutna vrijednost relacije **ispit**(ISPIT):

| ispit (ISPIT) |       |       |         |         |            |     |        |         |
|---------------|-------|-------|---------|---------|------------|-----|--------|---------|
| matBr         | prez  | ime   | sifPred | nazPred | datlsp     | ocj | sifNas | prezNas |
| 1111          | Novak | Ivan  | 1001    | Mat-1   | 29.01.2011 | 1   | 1111   | Pašić   |
| 1111          | Novak | Ivan  | 1001    | Mat-1   | 05.02.2011 | 3   | 1111   | Pašić   |
| 1111          | Novak | Ivan  | 1003    | Fiz-1   | 28.06.2011 | 2   | 3333   | Horvat  |
| 1111          | Novak | Ivan  | 1002    | Mat-2   | 27.06.2011 | 4   | 2222   | Brnetić |
| 1234          | Kolar | Petar | 1001    | Mat-1   | 29.01.2011 | 3   | 2222   | Brnetić |

- funkcijske zavisnosti odrediti na temelju značenja podataka
- odrediti primarni ključ relacije (tako da bude zadovoljen uvjet 1NF prema kojem neključni atributi funkcijски ovise o ključu)
- postupno normalizirati relacijsku shemu ISPIT na 2NF i 3NF

# Primjer normalizacije

| student (STUDENT) |       |       |
|-------------------|-------|-------|
| matBr             | prez  | ime   |
| 1111              | Novak | Ivan  |
| 1234              | Kolar | Petar |

$$K_{\text{STUDENT}} = \{ \text{matBr} \}$$

| predmet (PREDMET) |         |
|-------------------|---------|
| sifPred           | nazPred |
| 1001              | Mat-1   |
| 1003              | Fiz-1   |
| 1002              | Mat-2   |

$$K_{\text{PREDMET}} = \{ \text{sifPred} \}$$

| nastavnik (NASTAVNIK) |         |
|-----------------------|---------|
| sifNas                | prezNas |
| 1111                  | Pašić   |
| 3333                  | Horvat  |
| 2222                  | Brnetić |

$$K_{\text{NASTAVNIK}} = \{ \text{sifNas} \}$$

| ispit <sub>3</sub> (ISPIT <sub>3</sub> ) |         |            |     |        |
|------------------------------------------|---------|------------|-----|--------|
| matBr                                    | sifPred | datlsp     | ocj | sifNas |
| 1111                                     | 1001    | 29.01.2011 | 1   | 1111   |
| 1111                                     | 1001    | 05.02.2011 | 3   | 1111   |
| 1111                                     | 1003    | 28.06.2011 | 2   | 3333   |
| 1111                                     | 1002    | 27.06.2011 | 4   | 2222   |
| 1234                                     | 1001    | 29.01.2011 | 3   | 2222   |

$$K_{\text{ISPIT}_3} = \{ \text{matBr}, \text{sifPred}, \text{datlsp} \}$$

- Shema baze podataka STUSLU:

$$\text{STUSLU} = \{ \text{STUDENT}, \text{PREDMET}, \text{NASTAVNIK}, \text{ISPIT}_3 \}$$

# Implementacija: SQL

```
CREATE TABLE student (
 matBr INTEGER
 , prez CHAR(20)
 , ime CHAR(20)
 , PRIMARY KEY (matBr));
```

```
CREATE TABLE predmet (
 sifPred INTEGER
 , nazPred CHAR(20)
 , PRIMARY KEY (sifPred));
```

```
CREATE TABLE nastavnik (
 sifNas INTEGER
 , prezNas CHAR(20)
 , PRIMARY KEY (sifNas));
```

```
CREATE TABLE ispit (
 , matBr INTEGER REFERENCES student (matBr)
 ON DELETE CASCADE
 , sifPred INTEGER REFERENCES predmet (sifPred)
 , datIsp DATE
 , ocj SMALLINT CHECK (ocj BETWEEN 1 AND 5)
 , sifNas INTEGER REFERENCES nastavnik(sifNas)
 ON DELETE SET NULL
 , PRIMARY KEY (matBr, sifPred, datIsp));
```

# OBLIKOVANJE MODELA BAZE PODATAKA

---

## ER model (*Entity-Relationship Model*) Model entiteta-veze

- postrelacijski model
- zadržava dobre karakteristike relacijskog modela
- omogućuje eksplisitni prikaz veza koje u sebi sadrže važne semantičke informacije

## Literatura:

---

- P.P.Chen:  
The Entity-Relationship Model - Toward a Unified View of Data,  
ACM Transactions on Database Systems, Vol. 1, No. 1, 1976
- T. J. Teorey:  
Database Modeling & Design, Morgan Kaufmann, 1999

# Entiteti, veze, uloge

---

## Entitet

- bilo što, što ima suštinu ili bit, ima jasnoću kao činjenica ili ideja, posjeduje značajke s pomoću kojih se može razlučiti od svoje okoline

## Skup entiteta $E_i$ (*entityset*)

- Slični entiteti se grupiraju u skupove entiteta

## Skup veza $R_i$ (*relationship set*)

- matematička relacija između n entiteta:

$$R_i \subseteq E_1 \times E_2 \times E_3 \times \dots \times E_n$$

ili  $R_i = \{ (e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n \}$

n-torka ( $e_1, e_2, e_3, \dots, e_n$ ), naziva se vezom.

## Uloga (*role*)

- funkcija koju skup entiteta obavlja u skupu veza.

# Skup vrijednosti, atribut

---

- Informacije o entitetu ili vezi izražavaju se s pomoću parova **atribut-vrijednost**
- Vrijednosti su klasificirane u skupove vrijednosti  $V_i$ .
- **Atribut** je funkcija koja preslikava iz skupa entiteta ili skupa veza u skup vrijednosti ili Kartezijev produkt skupova vrijednosti:

$$f : E_i \rightarrow V_i$$

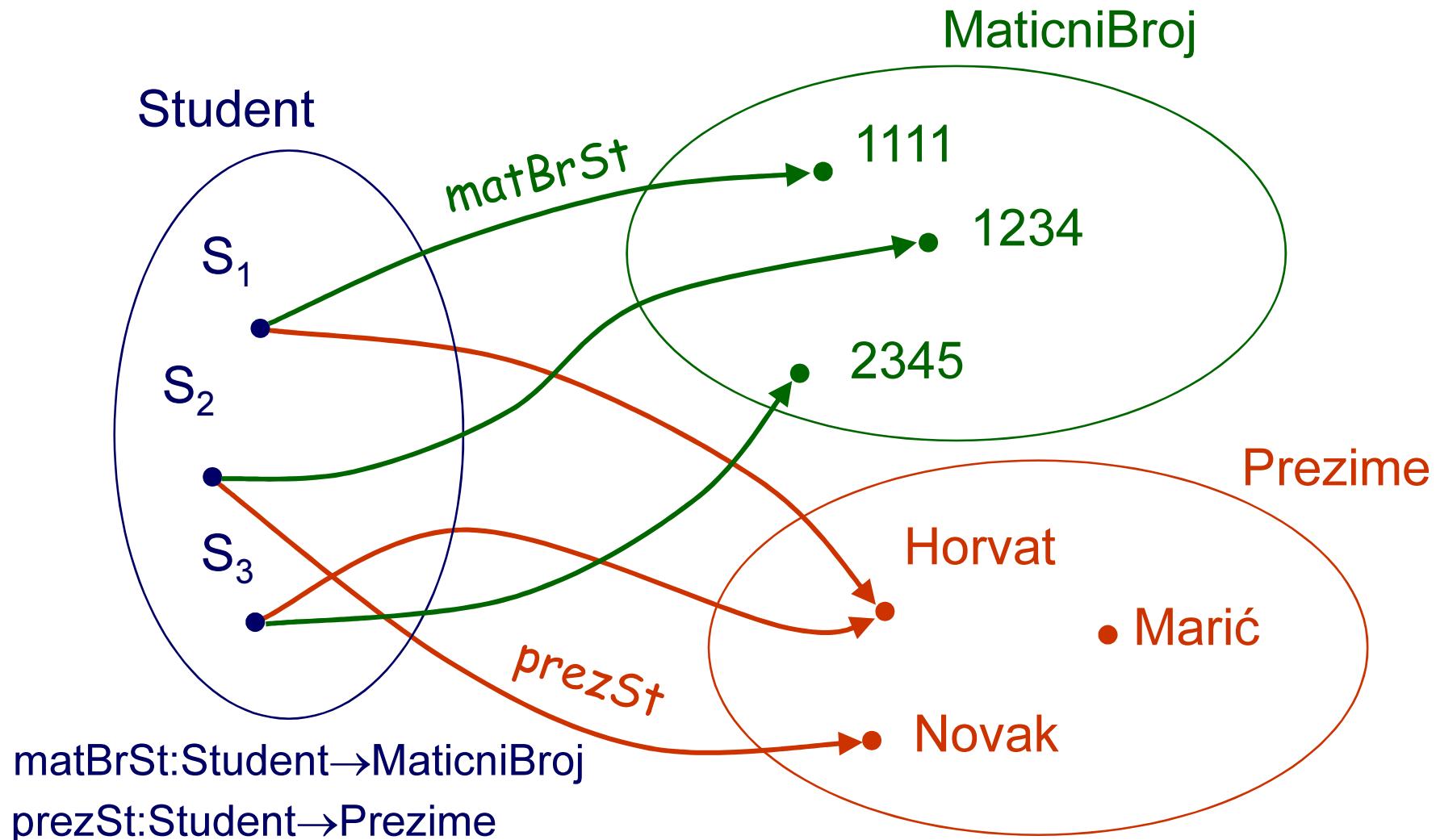
$$f : E_i \rightarrow V_{i_1} \times V_{i_2} \times \dots \times V_{i_n}$$

$$f : R_i \rightarrow V_i$$

$$f : R_i \rightarrow V_{i_1} \times V_{i_2} \times \dots \times V_{i_n}$$

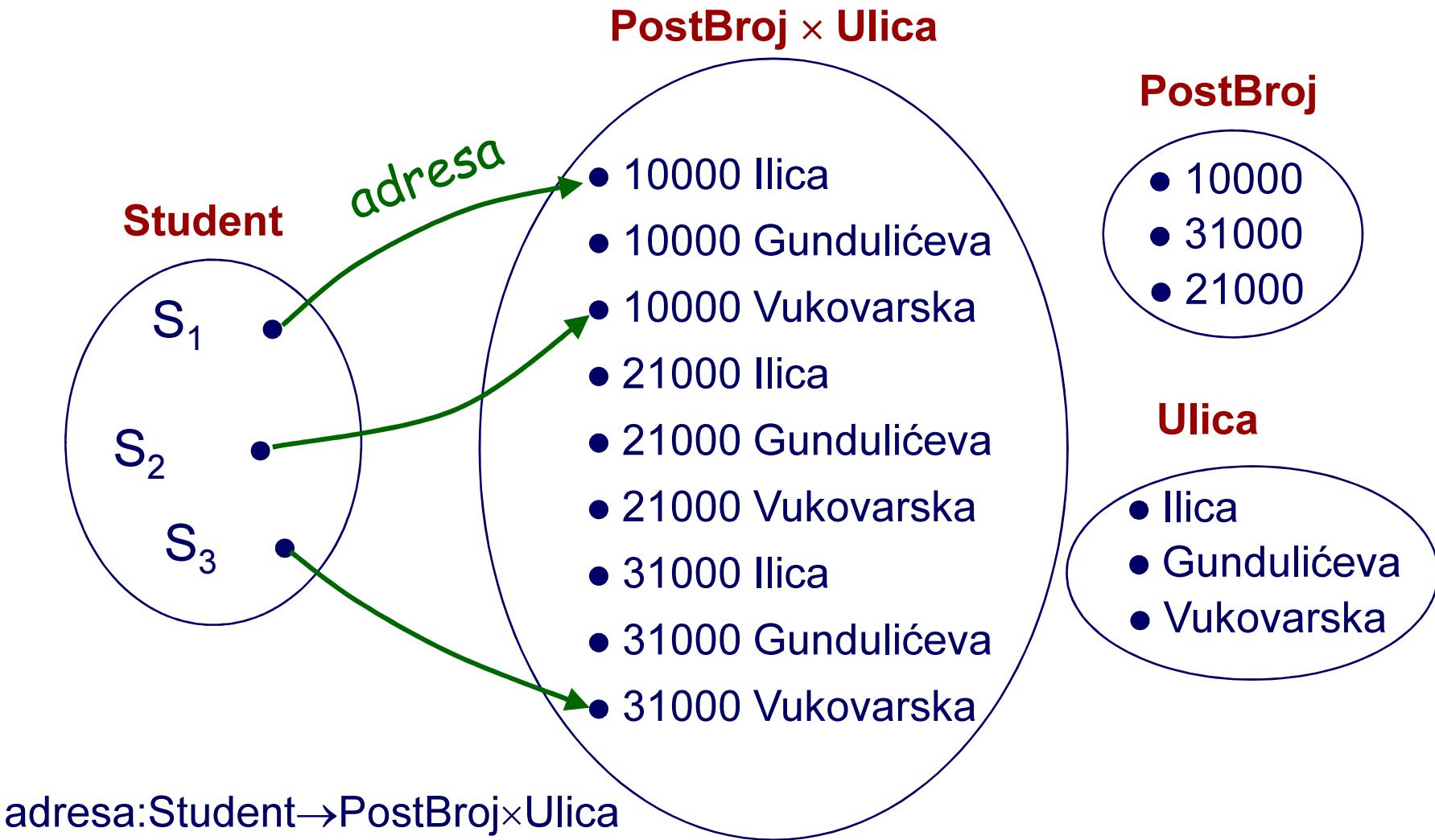
# Atributi entiteta

- funkcija koja preslikava sa skupa entiteta na skup vrijednosti ...



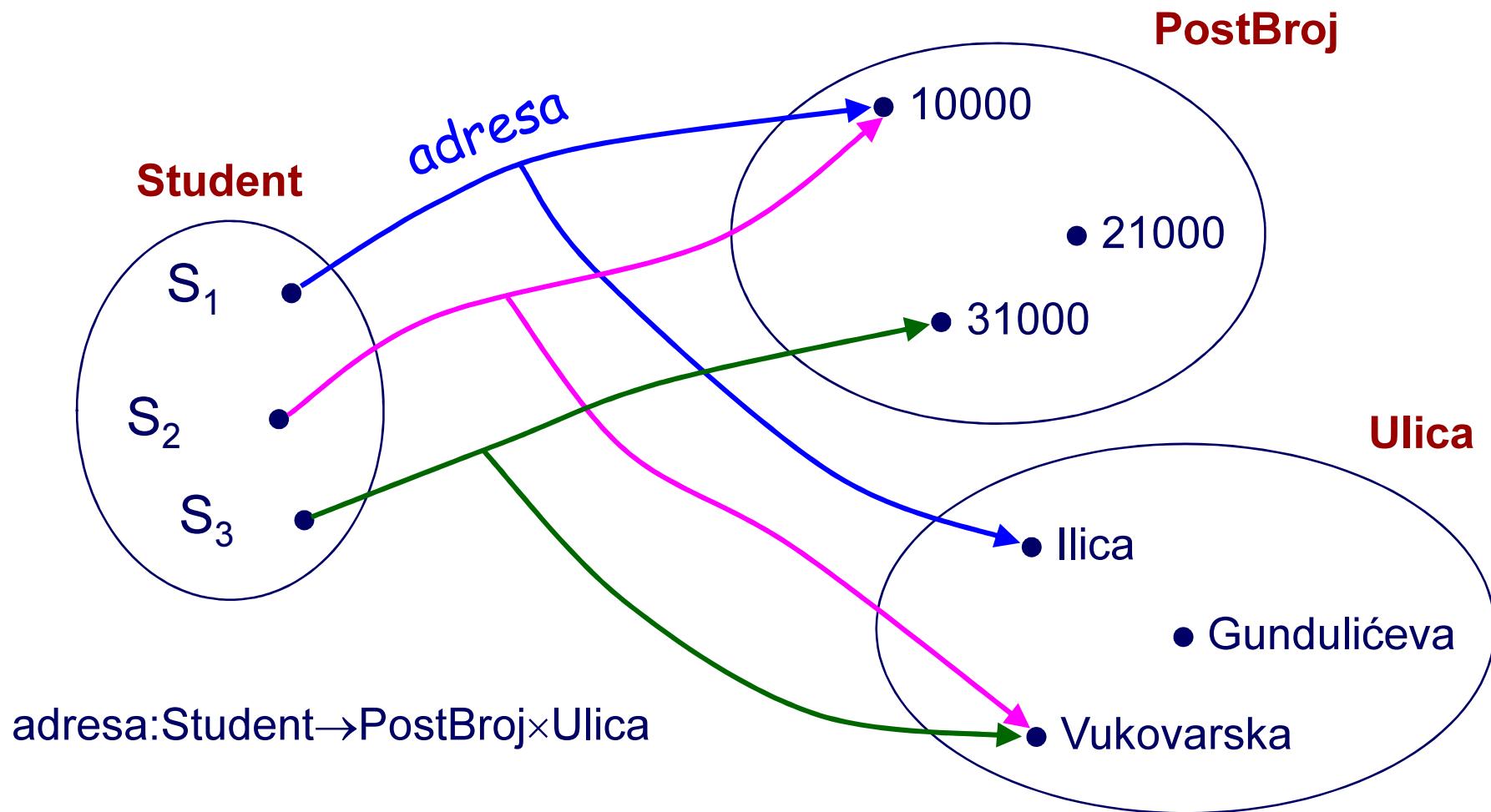
# Atributi entiteta

... ili na Kartezijev produkt skupova vrijednosti



# Atributi entiteta

... ili na Kartezijev produkt skupova vrijednosti



# Terminologija

---

Chen:

entitet, skup entiteta  
veza, skup veza

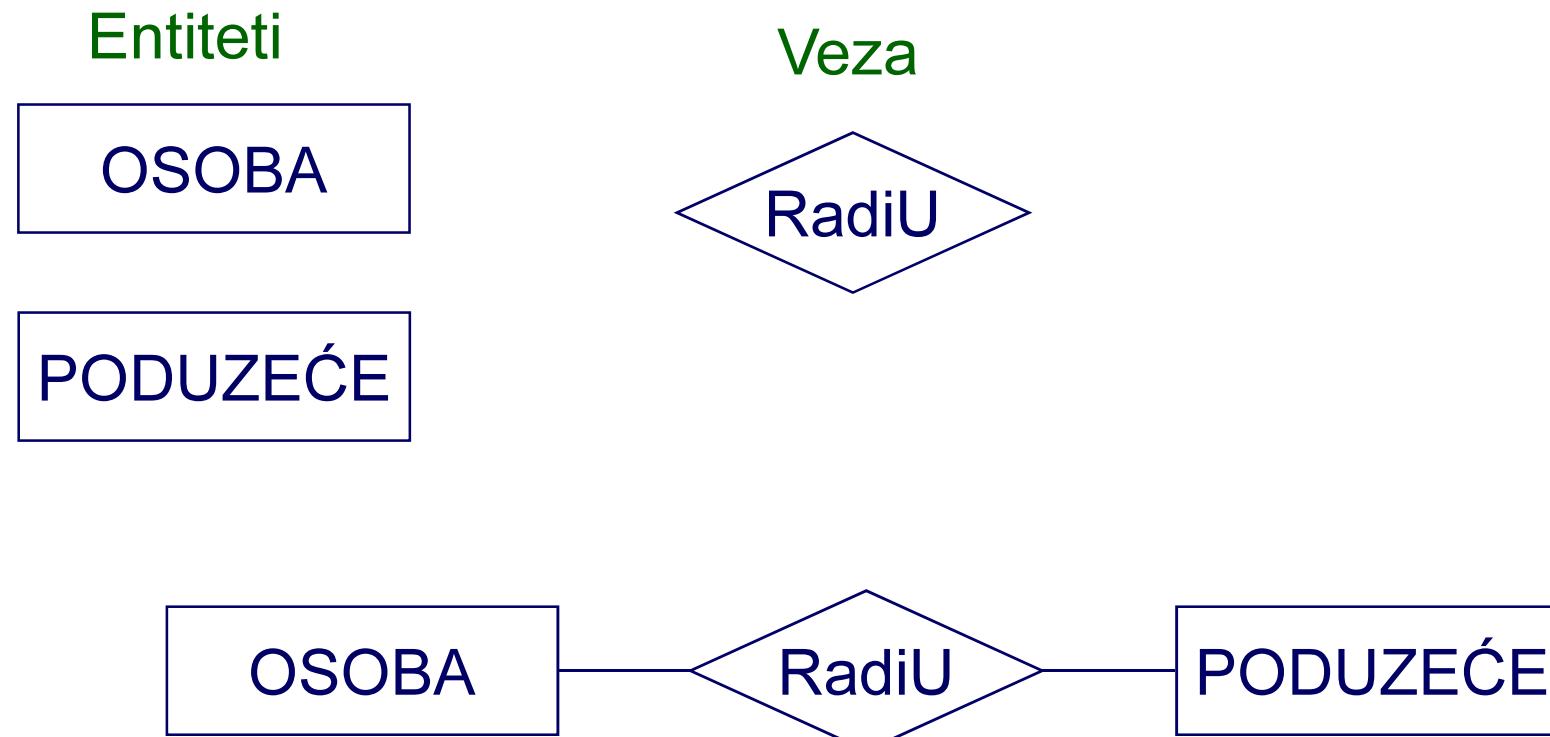
Teorey:

instanca entiteta, entitet  
(*entity instance*)  
(*entity occurrence*)  
instanca veze, veza  
(*relationship instance*)  
(*relationship occurrence*)

# Grafički prikaz entiteta i veza

---

- entitet se grafički prikazuje pravokutnikom unutar kojeg se nalazi ime entiteta
- veza se grafički prikazuje rombom unutar kojeg se nalazi ime veze



# Atributi entiteta

- atribut entiteta se grafički prikazuju ovalom unutar kojeg se upisuje ime atributa
- atribut (ili atributi) **primarnog ključa** se potcrtavaju



- povećanjem broja atributa, dijagram postaje nepregledan
  - atributi se tada ne prikazuju grafički - umjesto toga, uz dijagram se prilaže **sheme entiteta**

Shema entiteta:

**NASTAVNIK** = **sifNast**, **jmbgNast**, **imeNast**, **prezNast**

**PK** = { **sifNast** }

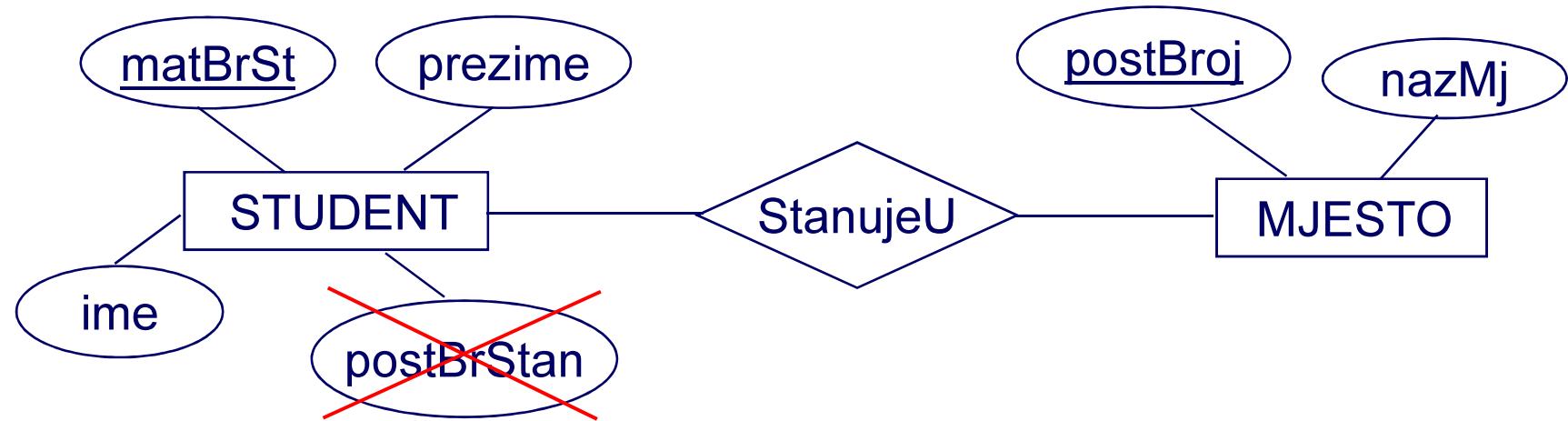
ili

|                               |
|-------------------------------|
| <b>NASTAVNIK</b>              |
| <b>sifNast</b>                |
| <b>jmbgNast</b>               |
| <b>imeNast</b>                |
| <b>prezNast</b>               |
| $K_1 = \{ \text{sifNast} \}$  |
| $K_2 = \{ \text{jmbgNast} \}$ |
| $\text{PK} = K_1$             |

# Vlastiti atributi entiteta

Entiteti se opisuju samo vlastitim atributima

- **vlastiti atribut entiteta** je atribut koji opisuje znanja o entitetu koja se pripisuju isključivo samom entitetu, a nikako vezi s drugim entitetima



- isključivo identifikacijski slabi entiteti, osim svojih **vlastitih** atributa, posjeduju i attribute primarnog ključa entiteta vlasnika

# Regularni i slabi entiteti

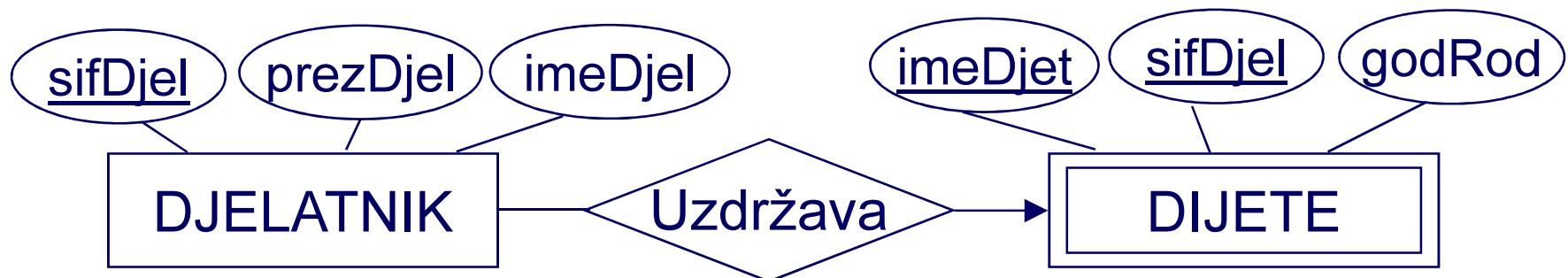
- regularni entitet je entitet koji može postojati sam za sebe
- slabi entiteti (engl. *weak entity*) ne postoji uokolo ne postoji i neki drugi entitet (entitet vlasnik)
- Slabi entitet se grafički prikazuje dvostruko uokvirenim pravokutnikom, sa strelicom koja dolazi iz smjera veze koja ga povezuje s entitetom vlasnikom



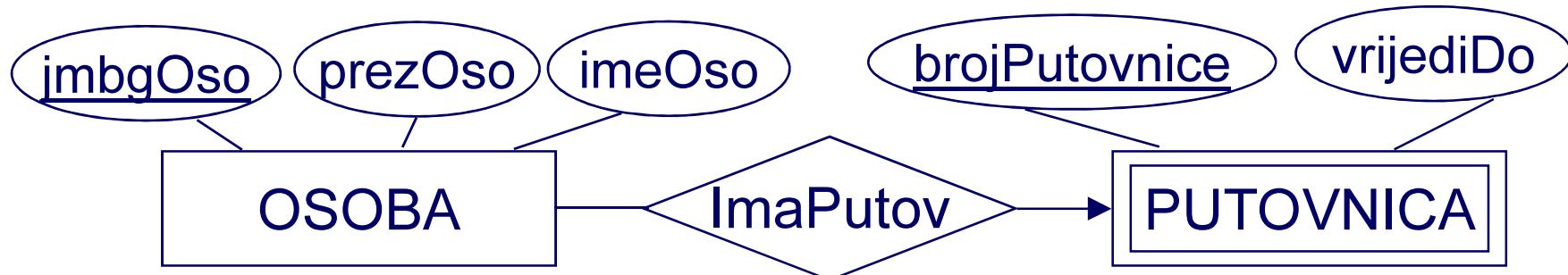
- slabi entiteti, osim što su **egzistencijalno slabi**, također mogu biti i **identifikacijski slabi**
  - kod određivanja identifikatora nisu im dovoljni vlastiti atributi
  - za identifikaciju se koriste i ključni atributi entiteta vlasnika

# Identifikacijski slabi entiteti (primjer)

- entitet DIJETE, osim što je egzistencijalno slab, također je i identifikacijski slab



- entitet PUTOVNICA je egzistencijalno slab (nije identifikacijski slab)

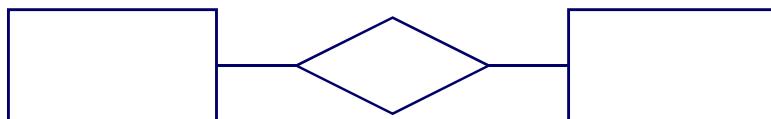


# Stupanj veze

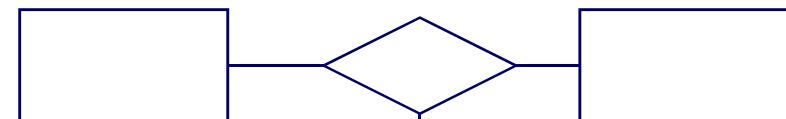
---

- broj entiteta koje povezuje dotična veza
- veza može biti unarna(refleksivna), binarna, ternarna, itd.
  - unarna ili refleksivna veza - veza je definirana nad jednim entitetom koji u vezi ima dvije različite uloge

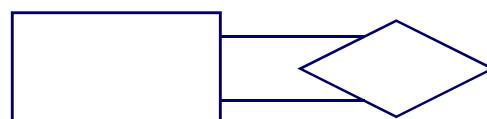
binarna



ternarna



refleksivna



# Spojnost veze (*connectivity*)

- spojnost veze opisuje ograničenje preslikavanja pojedinačnih entiteta koje veza povezuje
- vrijednosti spojnosti: jedan (*one*), više (*many*)
- koriste se oznake 1, N ili rasponi, npr. 0..1, 1..N, 1..2, itd.



- jedan djelatnik radi na jednom projektu, na jednom projektu radi N djelatnika



- jedan djelatnik radi na nula (niti jednom) ili jednom projektu, na jednom projektu radi između nula (niti jedan) i više djelatnika

# Spojnost veze (*connectivity*)

---

- radi pojednostavljenja
  - spojnost 0..N se često označava samo oznakom N
  - spojnost 1..1 se često označava samo oznakom 1



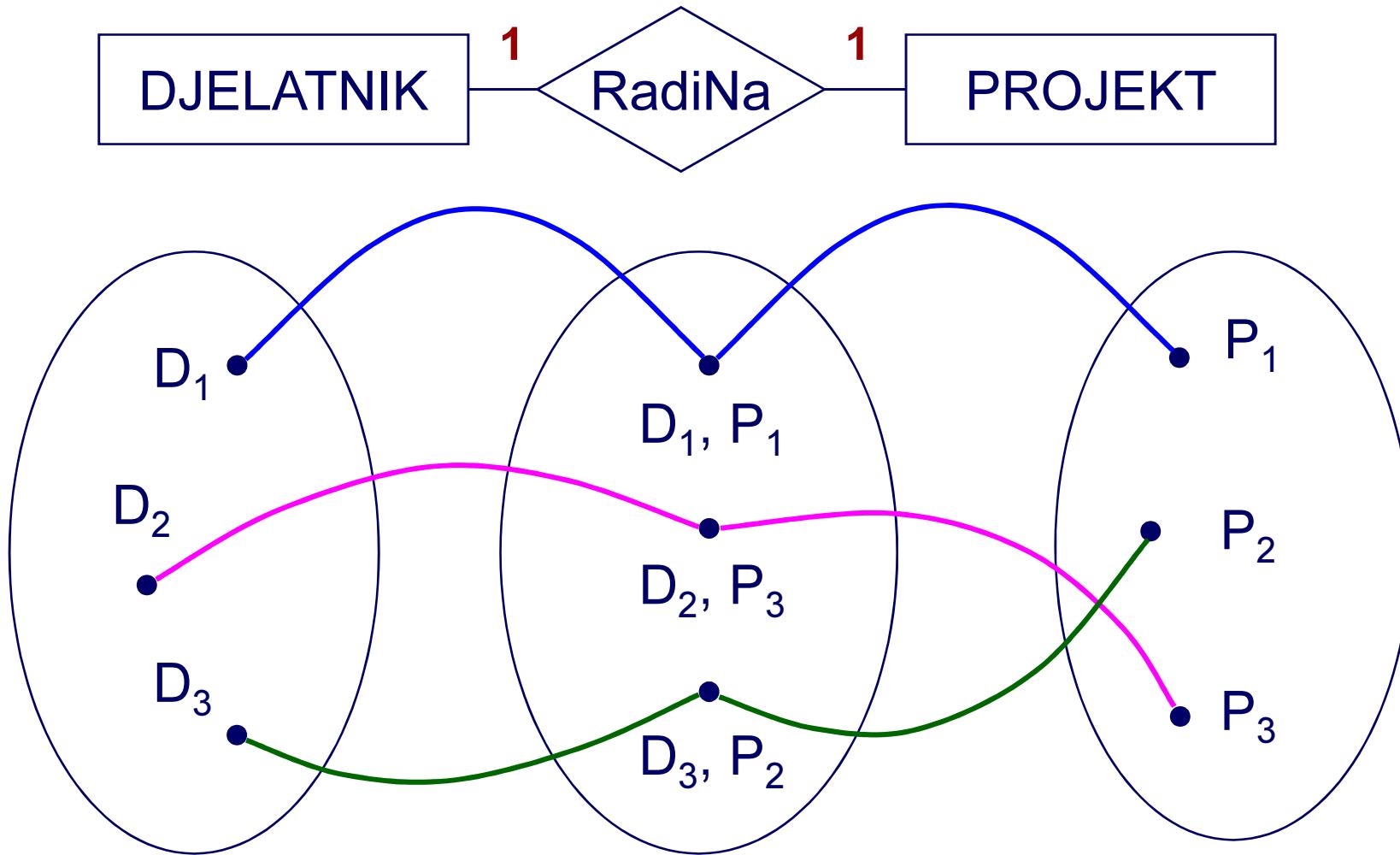
# Preslikavanje (*mapping*)

- preslikavanje - međusobni odnos entiteta u vezi
- kod binarnih veza moguća su preslikavanja **1:1** (jedan-prema-jedan), **1:N** (jedan-prema-više), **N:1** (više-prema-jedan), **N:N** (više-prema-više).



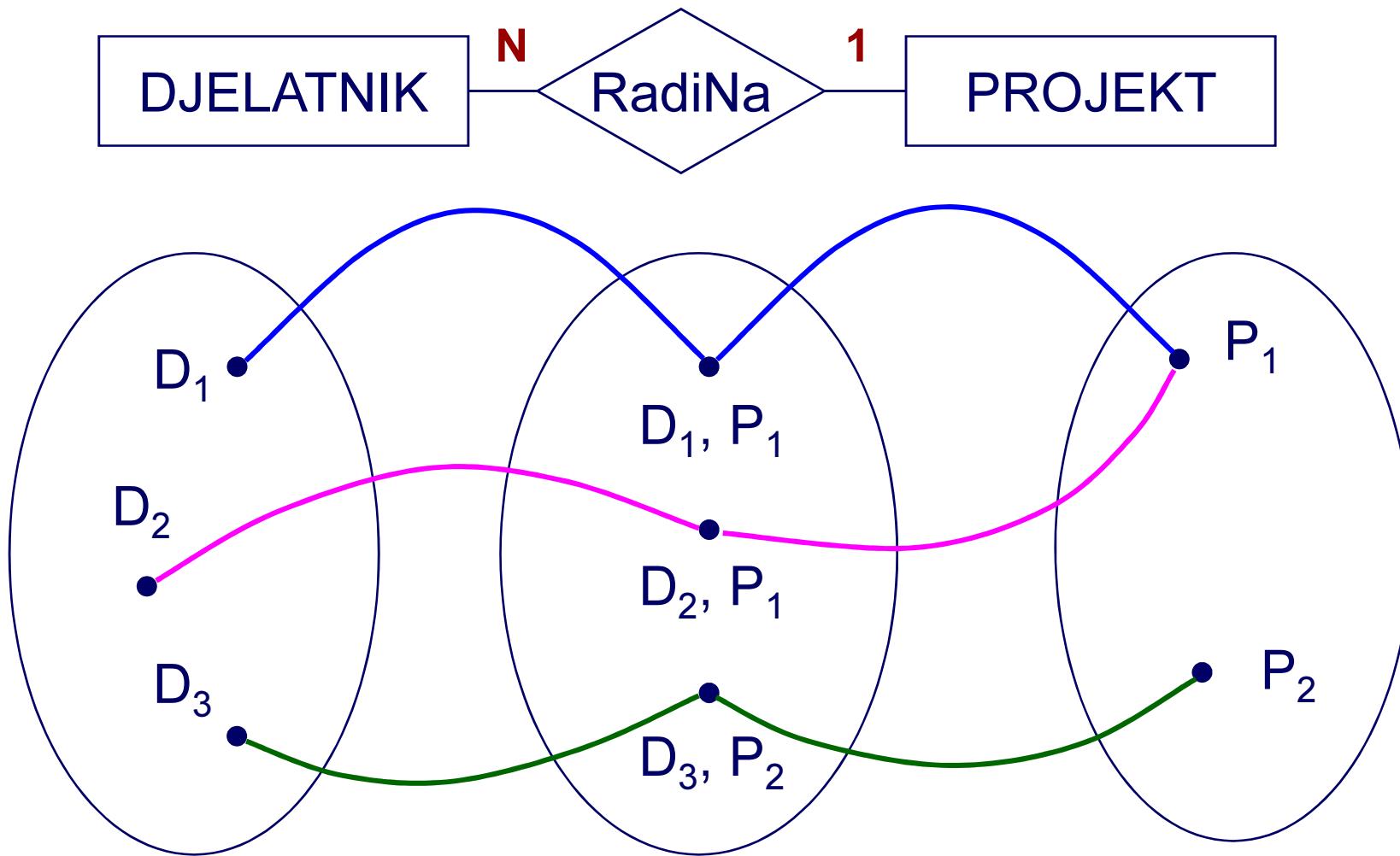
# Preslikavanje 1:1

---



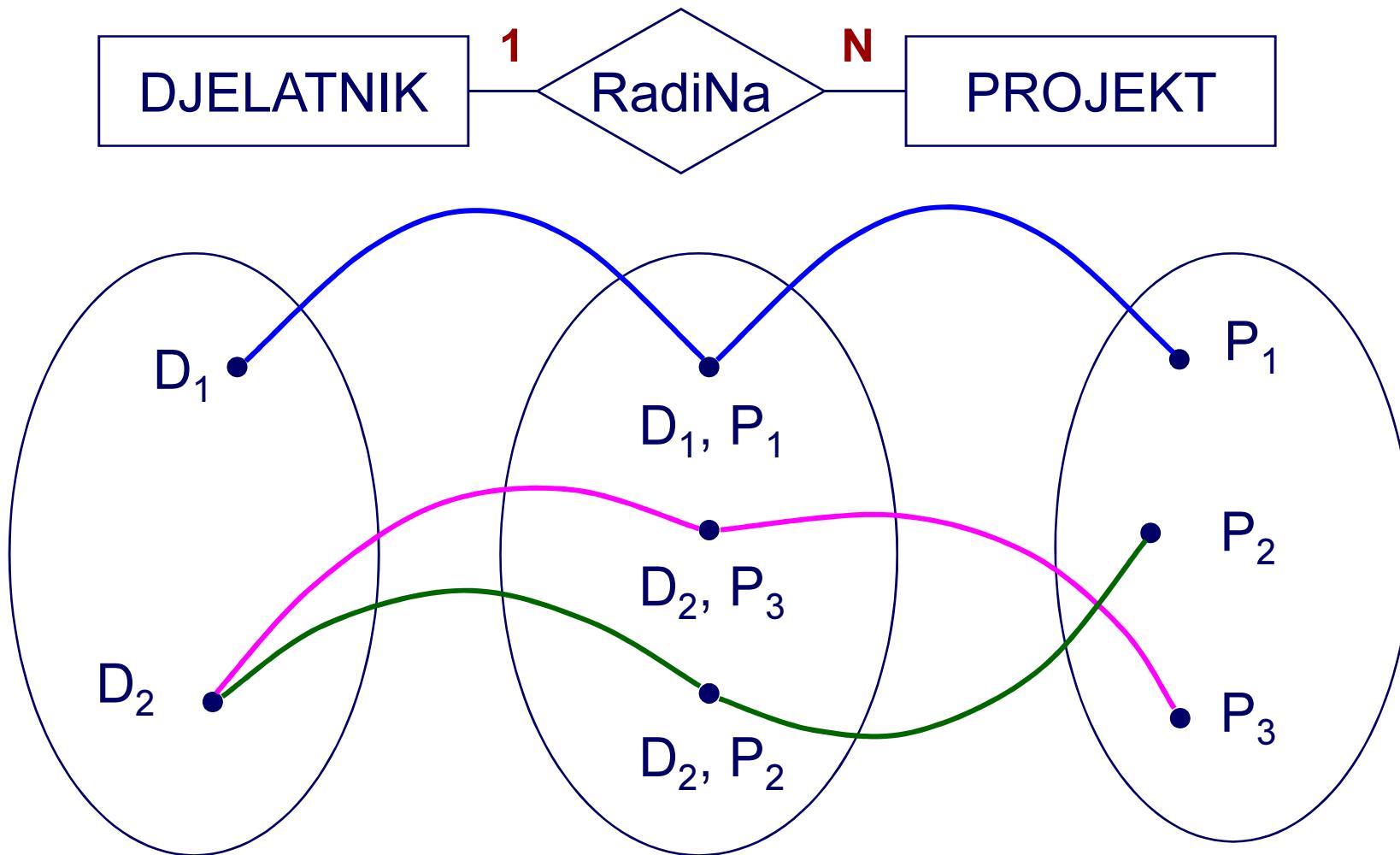
## Preslikavanje N:1

---

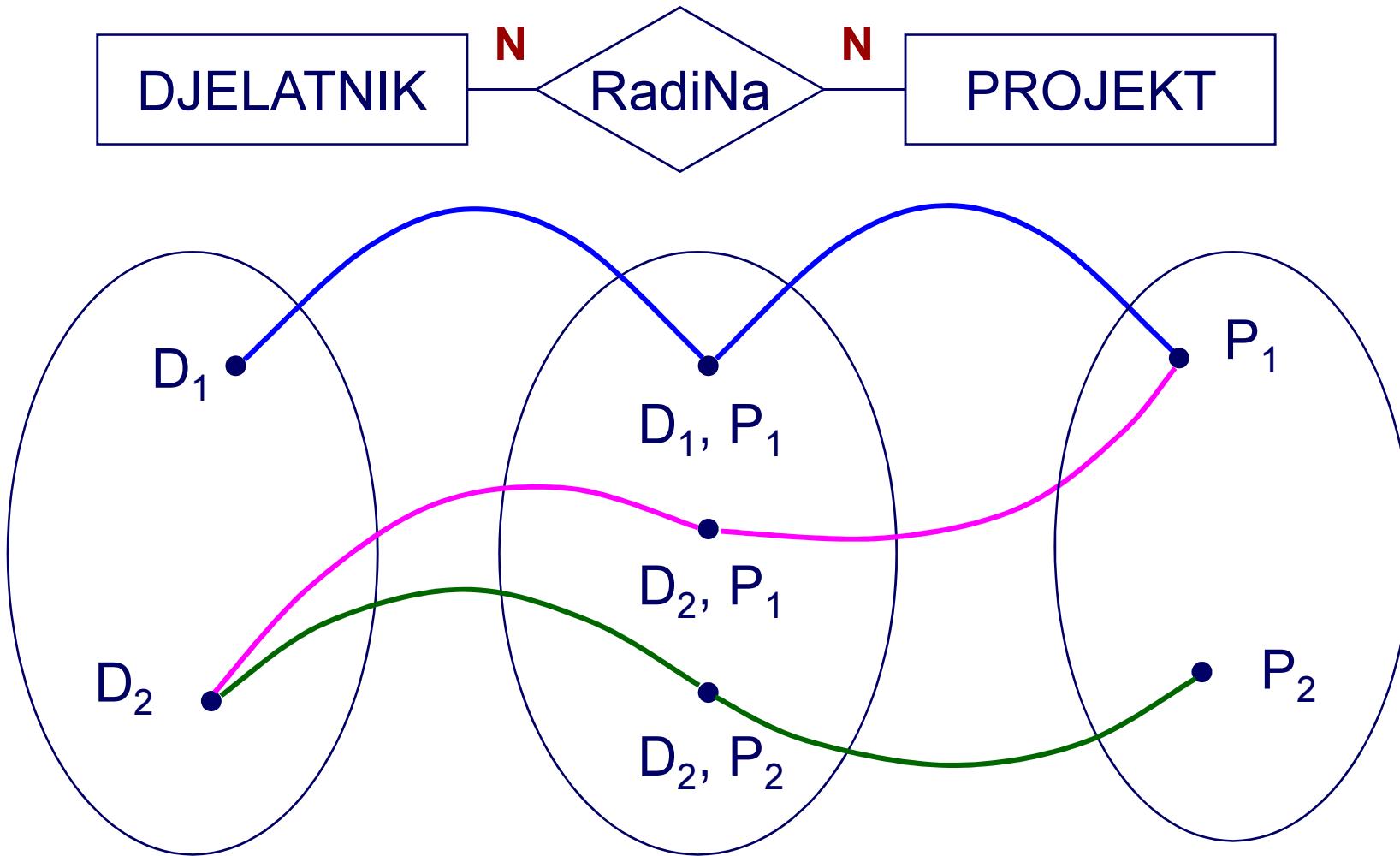


# Preslikavanje 1:N

---



# Preslikavanje N:N



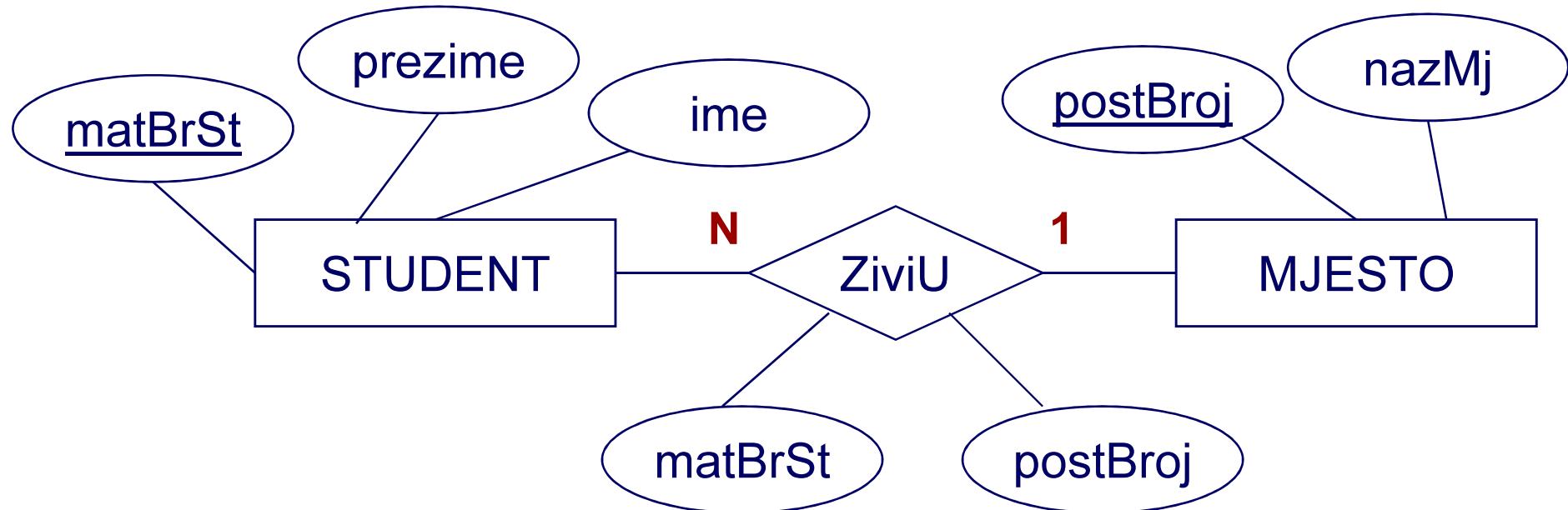
## Atributi veza

---

- Shema veze sadrži ključeve entiteta koje povezuje, te vlastite attribute
- Atribut veze se grafički prikazuje ovalom unutar kojeg se upisuje ime atributa

# Atributi veza

---



sheme entiteta:

**STUDENT** = matBrSt, prezime, ime

**MJESTO** = postBroj, nazMj

shema veze:

**ZiviU** = matBrSt, postBroj

Koji atributi čine ključ veze?

# Ključevi veza

---

- Povezanost entiteta opisuje se kao odnos među ključevima entiteta
- Ključevi veza definirani su s pomoću **ključeva entiteta** koje **povezuju i njihovih spojnosti**

# Definicija 1. (Teorey)

---

U vezi koja povezuje entitete

$E_1, \dots, E_k, \dots, E_m$ ,

spojnost = 1 entiteta  $E_k$  znači da za svaku vrijednost svih entiteta  $E_1, \dots, E_m$ , osim  $E_k$ , uvijek postoji točno jedna vrijednost od  $E_k$ .

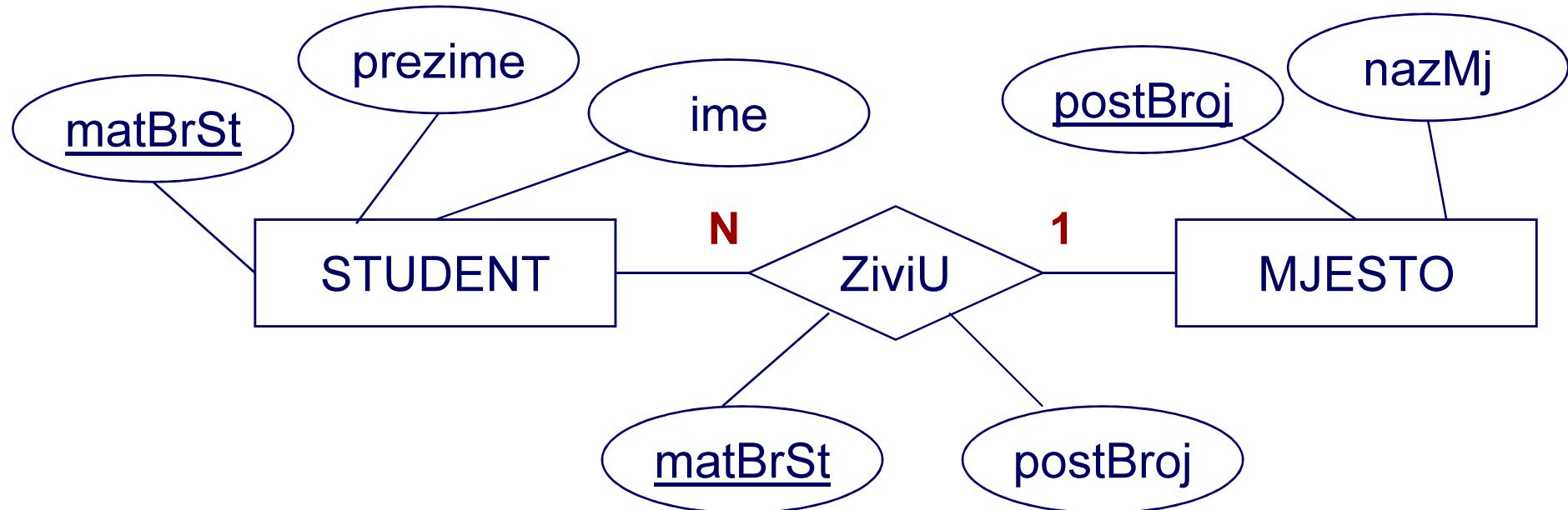
→ može se reći da tada vrijedi funkcionska zavisnost:

$$\bigcup_{j=1}^m K_j \setminus K_k \rightarrow K_k$$

gdje su skupovi  $K_j$ , ( $j = 1, \dots, m$ ) ključevi entiteta  $E_1, \dots, E_m$

# Ključevi veza

---



sheme entiteta:

**STUDENT** = matBrSt, prezime, ime

**MJESTO** = postBroj, nazMj

shema veze:

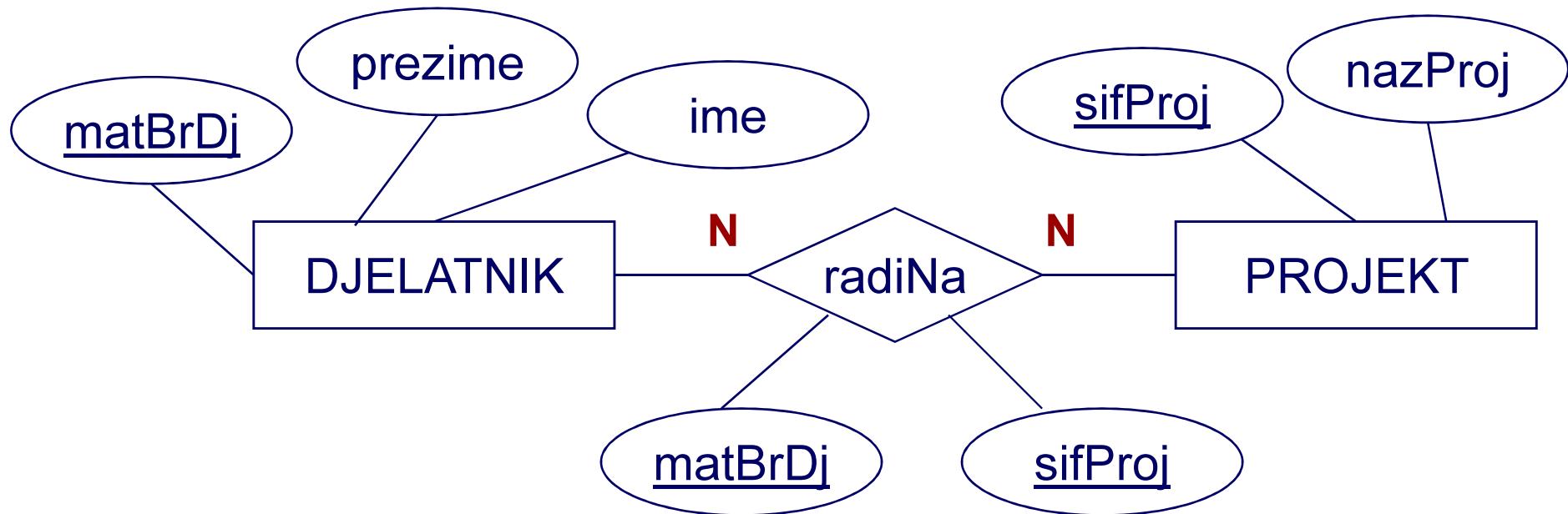
**ZiviU** = matBrSt, postBroj

Iz definicije 1:  
 $\text{matBrSt} \rightarrow \text{postBroj}$



# Ključevi veza

---



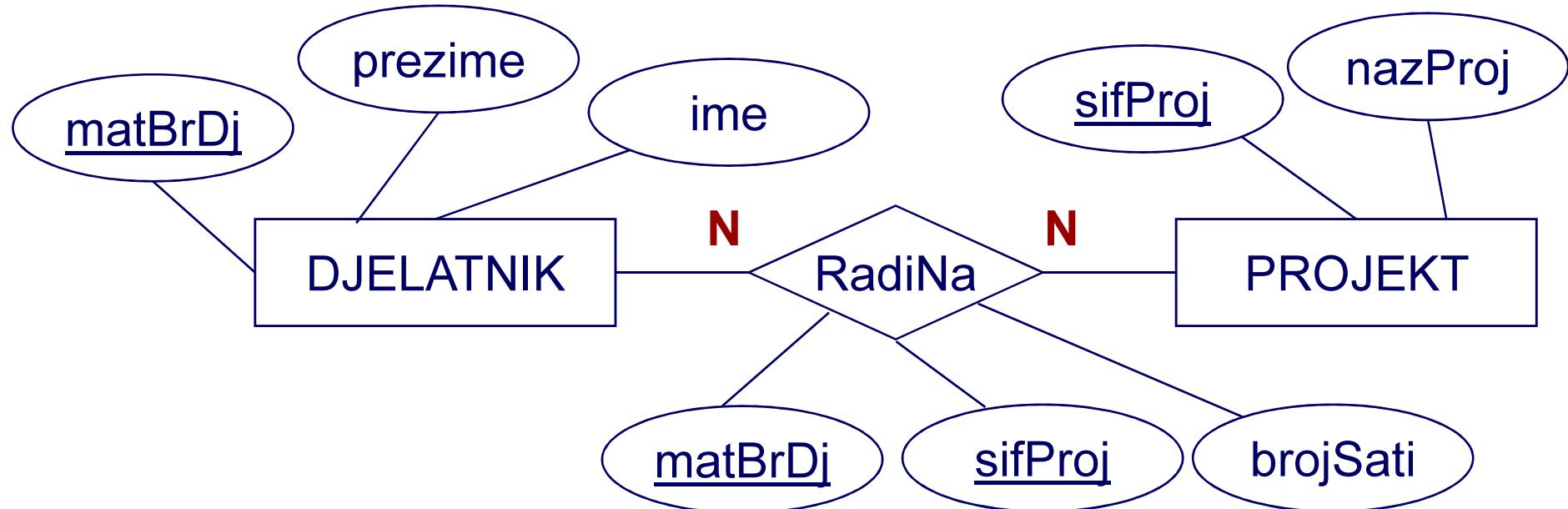
**DJELATNIK** = matBrDj, prezime, ime

**PROJEKT** = sifProj, nazProj

**RadiNa** = matBrDj, sifProj

# Vlastiti atributi veza

---



DJELATNIK = matBrDj, prezime, ime

PROJEKT = sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

ključ veze funkcijски одређује  
властите атрибуте везе:  
matBrDj, sifProj → brojSati

## Ključ veze - dodatna razmatranje

- iz definicije 1. proizlazi da se ključ veze sastoji isključivo od ključeva entiteta koje povezuje (svih ili samo nekih, ovisno o spojnostima)

Međutim, u nekim slučajevima ključ može sadržavati i neke druge atribute.



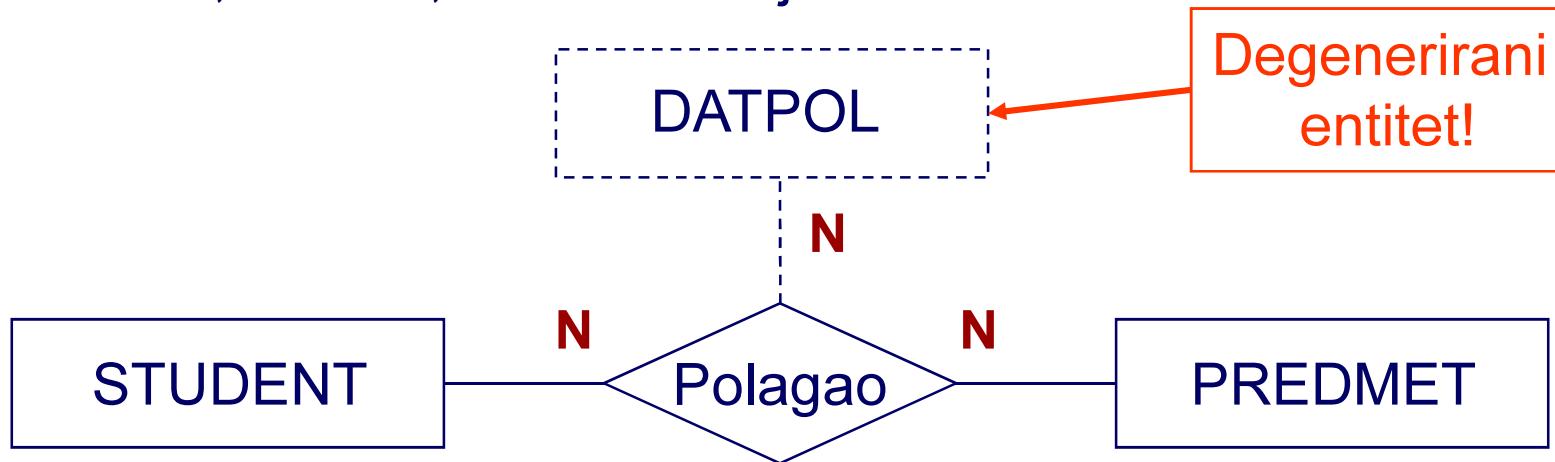
STUDENT = matBrSt, prezime, ime

PREDMET = sifPred, nazPred

Položio = matBrSt, sifPred, ocjena

# Ključ veze - dodatna razmatranje

- ako se želi evidentirati sva polaganja ispita  
matBrSt, sifPred  $\not\rightarrow$  ocjena
- potrebno je uvesti atribut datPol (datum polaganja):  
matBrSt, sifPred, datPol  $\rightarrow$  ocjena



STUDENT = matBrSt, prezime, ime

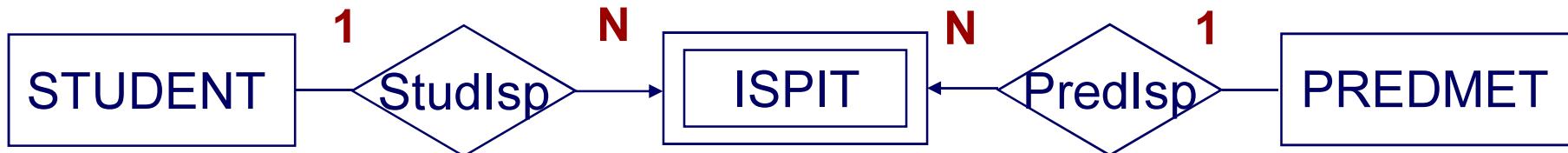
PREDMET = sifPred, nazPred

Polagao = matBrSt, sifPred, datPol, ocjena

# Ključ veze - dodatna razmatranje

---

- druga mogućnost - veza postaje entitet:



STUDENT = matBrSt, prezime, ime

PREDMET = sifPred, nazPred

ISPIT = matBrSt, sifPred, datPol, ocjena

StudIsp = matBrSt, sifPred, datPol

PredIsp = matBrSt, sifPred, datPol

# Veza 1:N → preslikavanje u relacijski model

---



DJELATNIK = matBrDj, prezime, ime

MJESTO = postBr, nazMjesto

Stanuje = matBrDj, postBr, adresa

Relacijske sheme opisuju entitete (veze postaju entiteti)

DJELATNIK = matBrDj, prezime, ime

MJESTO = postBr, nazMjesto

Stanuje = matBrDj, postBr, adresa

Unija relacijskih shema s jednakim ključevima

DJELATNIK = matBrDj, prezime, ime, postBr, adresa

MJESTO = postBr, nazMjesto

# Veza N:N → preslikavanje u relacijski model

---



DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

Relacijske sheme opisuju entitete (veze postaju entiteti)

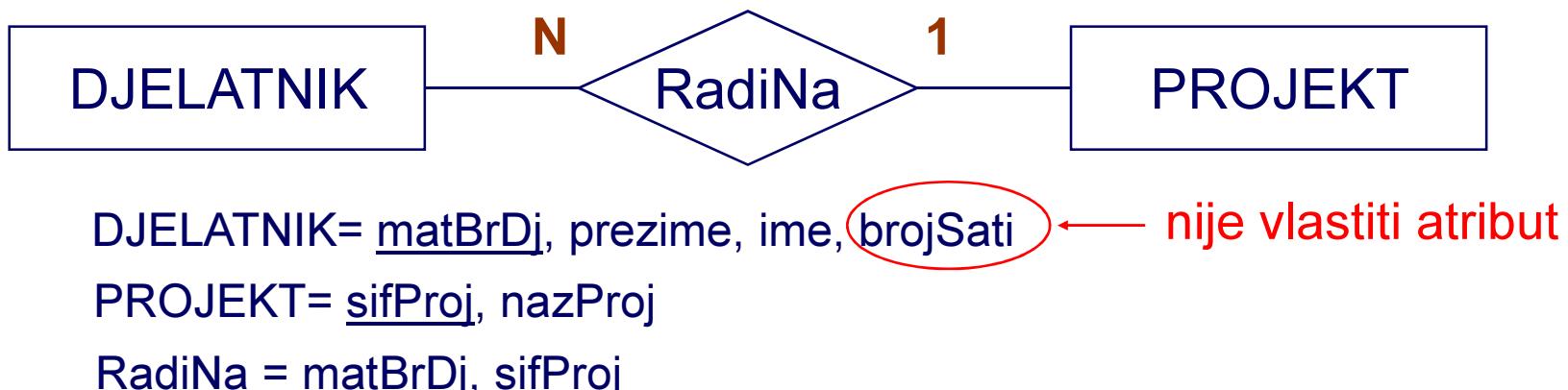
DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

## Primjer: zašto je važno ispravno odrediti vlastite attribute entiteta i veza?

- Entiteti se opisuju samo vlastitim atributima: **vlastiti atribut entiteta** je atribut koji opisuje znanja o entitetu koja se pripisuju isključivo samom entitetu, a nikako vezi s drugim entitetima



- Ako se preslikavanje promijeni u N:N



## Primjer: zašto je važno ispravno odrediti vlastite attribute entiteta i veza?



DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati ← vlastiti atribut

- Ako se preslikavanje promijeni u N:N

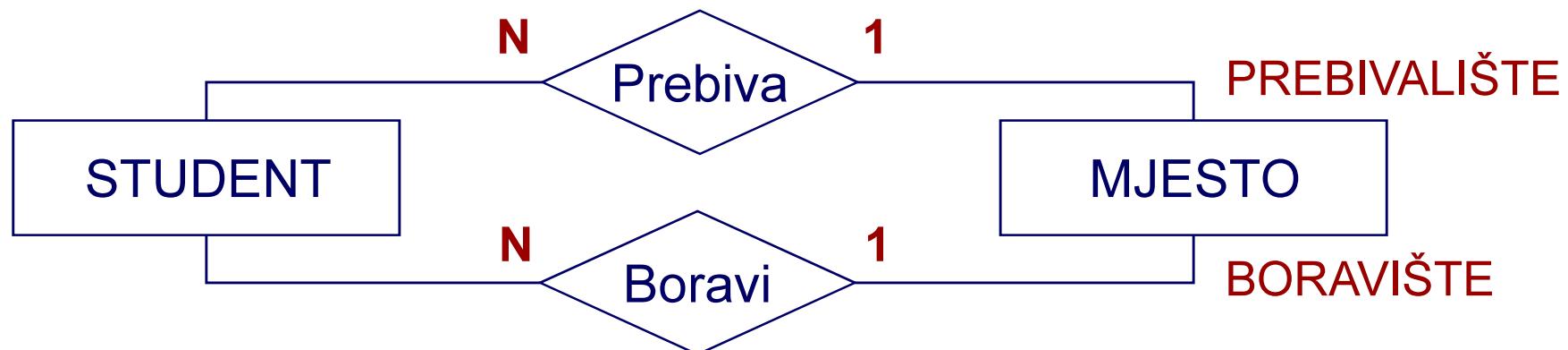


DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

# Paralelne veze



STUDENT = matBrSt, prezime, ime

MJESTO = postBroj, nazMjesto

Uloge: PREBIVALIŠTE  
BORAVIŠTE

Prebiva = matBrSt, postBroj      PostBrojPreb

Boravi = matBrSt, postBroj      PostBrojBor

# Paralelne veze → relacijski model

- Unija shema s jednakim ključevima:

MJESTO = postBroj, nazMjesto

STUDENT = matBrSt, prezime, ime, ~~postBroj~~, ~~postBroj~~

STUDENT = matBrSt, prezime, ime, **postBrojBor**, **postBrojPreb**  
+ pravila integriteta

**Zadatak:** Ispisati prezime i ime studenta, poštanski broj i naziv mesta boravka te poštanski broj i naziv mesta prebivališta

```
SELECT student.*
 , boraviste.nazMjesto AS nazMjestoBoraviste
 , prebivaliste.nazMjesto AS nazMjestoPrebivaliste
FROM student
 INNER JOIN mjesto AS boraviste
 ON boraviste.postBroj = student.postBrojBor
 INNER JOIN mjesto AS prebivaliste
 ON prebivaliste.postBroj = student.postBrojPreb
```

# Problem

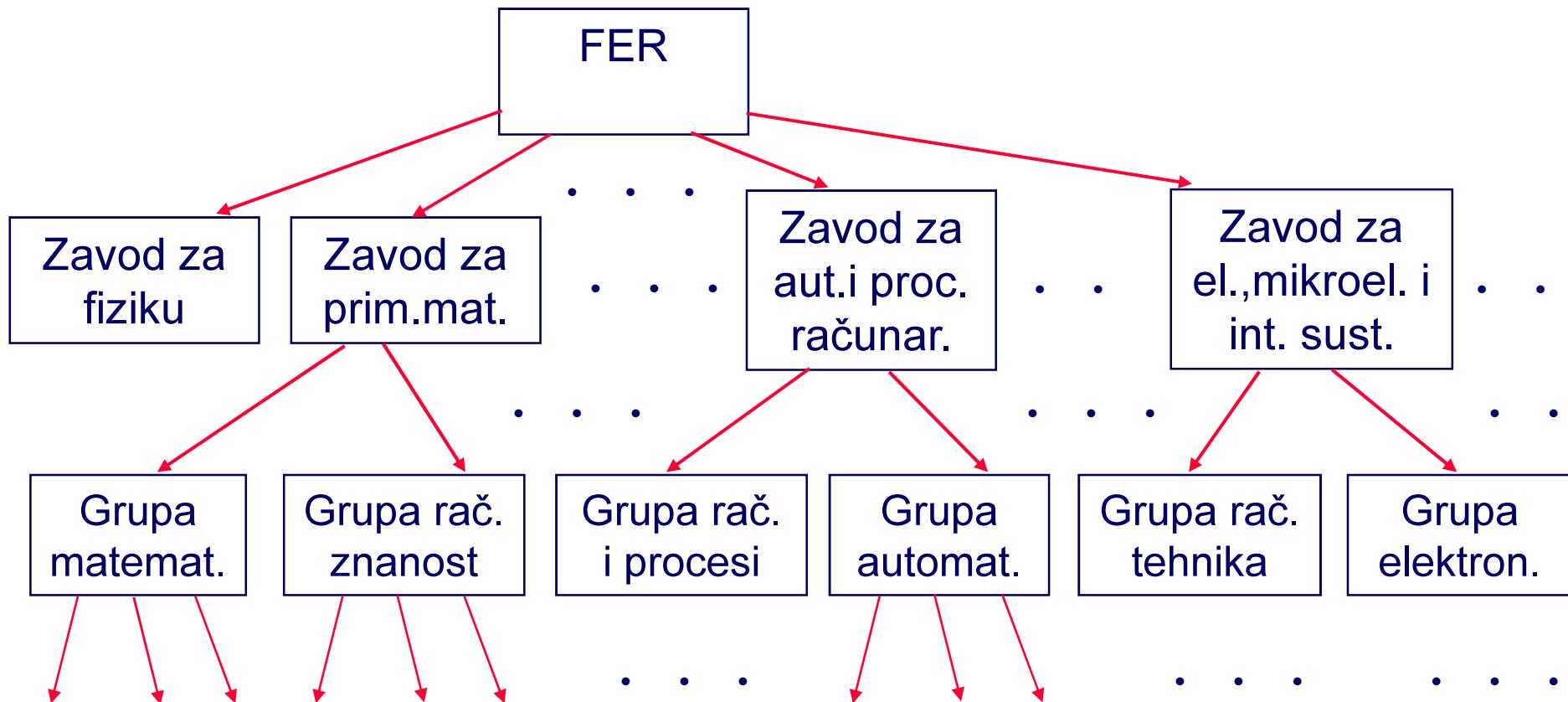
---

## Kako opisati organizacijsku strukturu poduzeća?

- Organizacijske jedinice opisane su svojom šifrom i nazivom
- Organizacijske jedinice međusobno su povezane
  - **kako?**  
→ među njima postoji hijerarhijski odnos!
  - **kolika je dubina stabla (broj razina)?**  
→ promjenjiva!
- Kako opisati hijerarhiju - stablo promjenjive dubine?
- Čvorovi stabla su opisani na isti način (šifra, naziv)

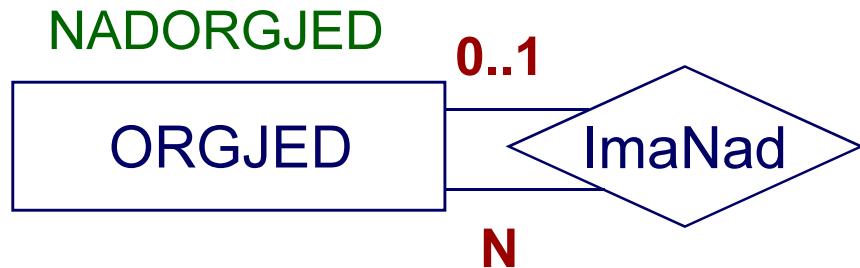
# Homogeno stablo

Primjer: Organizacijska struktura



Čvorovi stabla imaju jednaku strukturu: ORGJED= sifOrgJed, nazOrgJed

# Refleksivne veze - preslikavanje 1:N



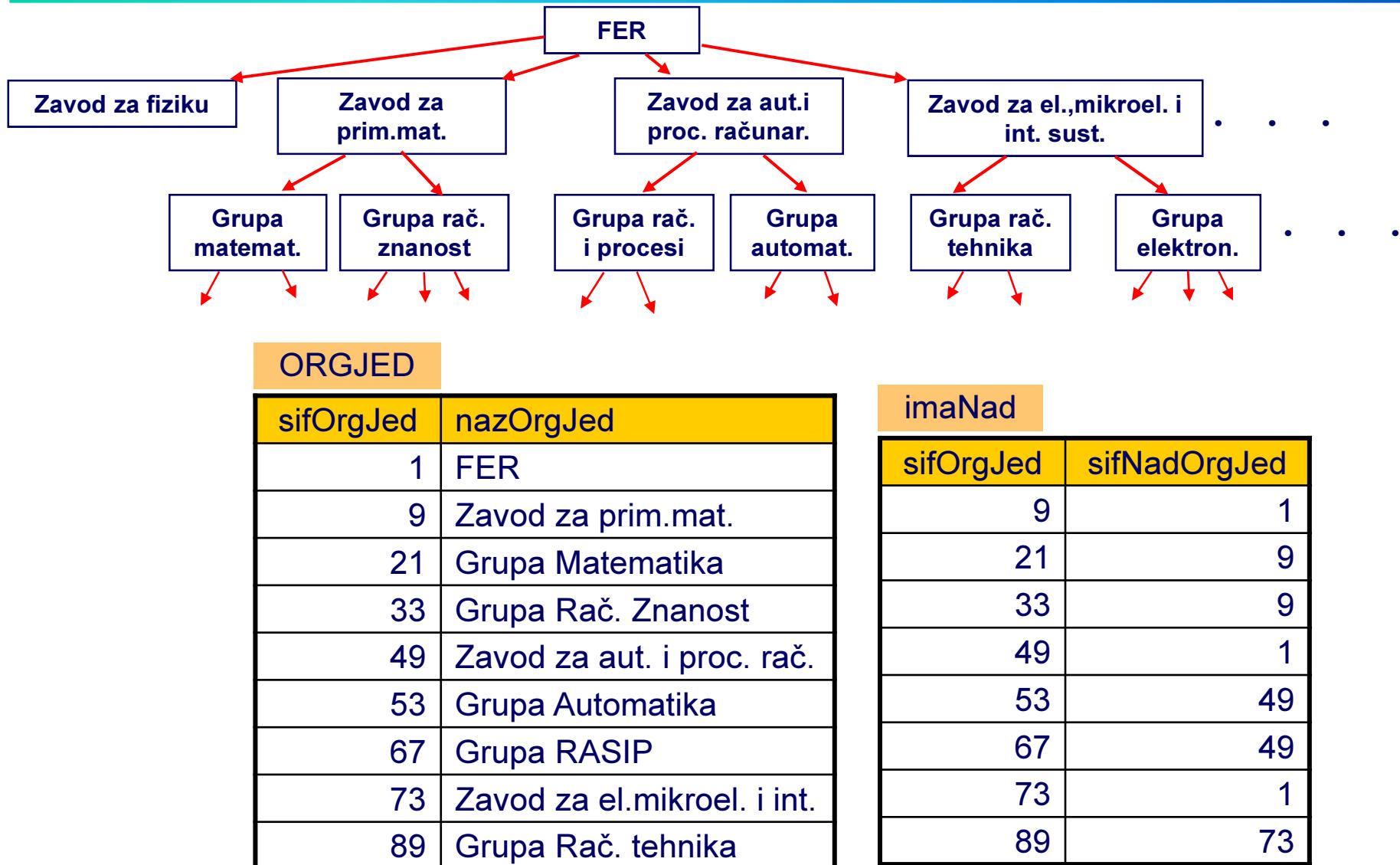
**ORGJED** = sifOrgJed, nazOrgJed

**ImaNad** = sifOrgJed, sifOrgJed

**ImaNad** = sifOrgJed, **sifNadOrgJed**

Preimenovati  
jedan od atributa !

# Homogeno stablo



# Refleksivne veze 1:N → relacijski model

---

- Unija shema s jednakim ključevima:

ORGJED = sifOrgJed, nazOrgJed

imaNad = sifOrgJed, sifNadOrgJed

ORGJED = sifOrgJed, nazOrgJed, sifNadOrgJed

+ pravila integriteta

**Zadatak:** Ispisati naziv organizacijske jedinice i naziv njezine nadređene organizacijske jedinice (ukoliko postoji)

```
SELECT orgjed.nazOrgJed, nadorgjed.nazOrgJed
 FROM orgjed
 LEFT OUTER JOIN orgjed AS nadorgjed
 ON orgjed.sifNadOrgJed = nadorgjed.sifOrgJed
```

# Što je šifra organizacijske jedinice?

---

- Govoreća šifra – šifra koja označava poziciju organizacijske jedinice unutar poduzeća??

npr. XXYYZZZ

XX – šifra sektora

YY – šifra odjela

ZZZ – šifra odsjeka

→ **što se dešava prilikom reorganizacije?**

→ moraju se promijeniti šifre organizacijskih jedinica!

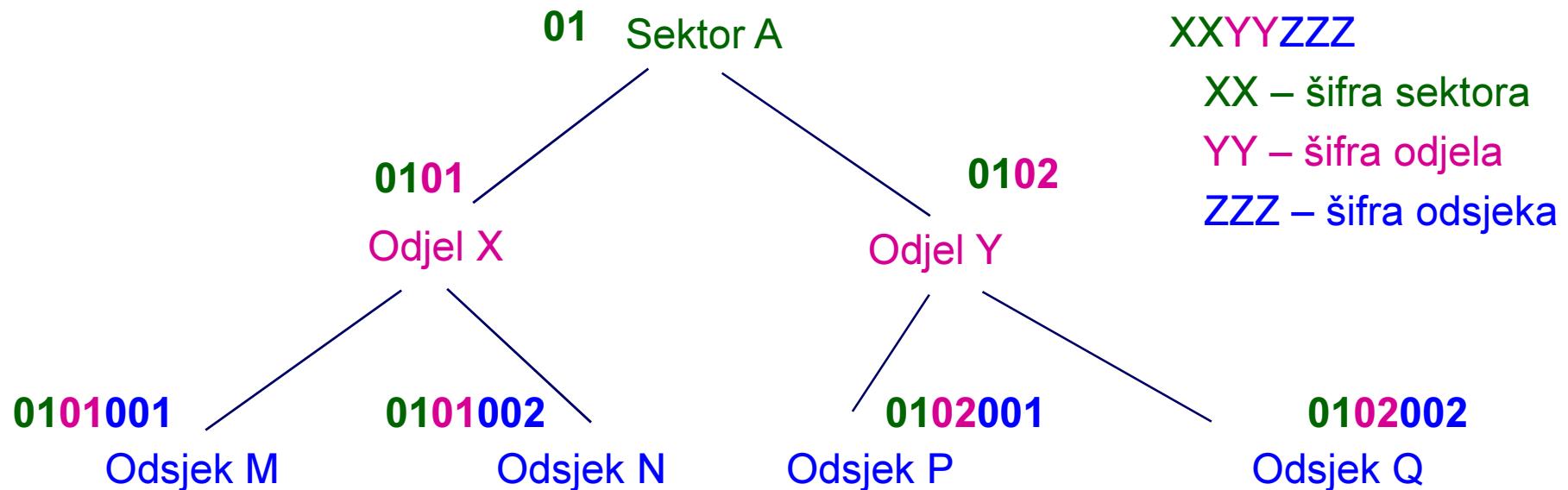
→ **što se dešava kada broj odjela preraste 100??**

→ moraju se promijeniti šifre organizacijskih jedinica!

→ Šifra organizacijske jedinice **NE SMIJE BITI GOVOREĆA!**

→ To vrijedi i za sve ostale šifre i identifikatore!!!

# Što je šifra organizacijske jedinice?



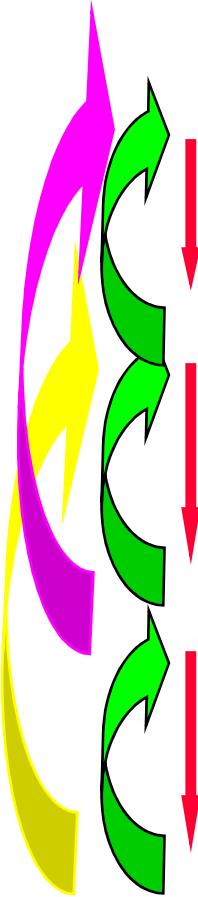
| ORGJED | sifOrgJed | nazOrgJed |
|--------|-----------|-----------|
|        | 01        | Sektor A  |
|        | 0101      | Odjel X   |
|        | 0102      | Odjel Y   |
|        | 0101001   | Odsjek M  |
|        | 0101002   | Odsjek N  |
|        | 0102001   | Odsjek P  |
|        | 0102002   | Odsjek Q  |

Što kada Odsjek P zbog reorganizacije iz Odjela Y preseli u Odjel X?

Što kada broj odjela preraste broj 99?

# Oblikovanje ER modela

---



## Oblikovanje ER modela

- **definiranje entiteta**
  - ime, opis, komentar
- **definiranje veza**
  - ime, opis, komentar, entiteti koje povezuje, preslikavanje
- **definiranje atributa entiteta**
  - za svaki atribut: ime, opis, komentar, domena
  - definirati ključeve, provjeriti da li zadovoljava 3NF
- **definiranje atributa veza**
  - za svaki atribut: ime, opis, komentar, domena
  - definirati ključeve, provjeriti da li zadovoljava 3NF

**POSTUPAK JE ITERATIVAN!**

# Model baze podataka

---

## SADRŽI OPISE

- entiteta
- veza
- atributa entiteta
- atributa veza

## KARAKTERISTIKE DOBROG MODELA

- opisuje suštinu, prirodu stvari, neovisan o postojećem stanju
- sveobuhvatan
- neredundantan
- fleksibilan
- razumljiv - korisnicima i informatičarima

## POSEBNO OBRATITI PAŽNJU NA:

- različito shvaćanje istih stvari - kupac, dobavljač → poslovni partner
- praćenje promjena u vremenu - stipendist, djelatnik, umirovljenik
- jednakost - uopćavanje - različiti odjeli i pojedinci mogu iste ili slične stvari shvaćati različito

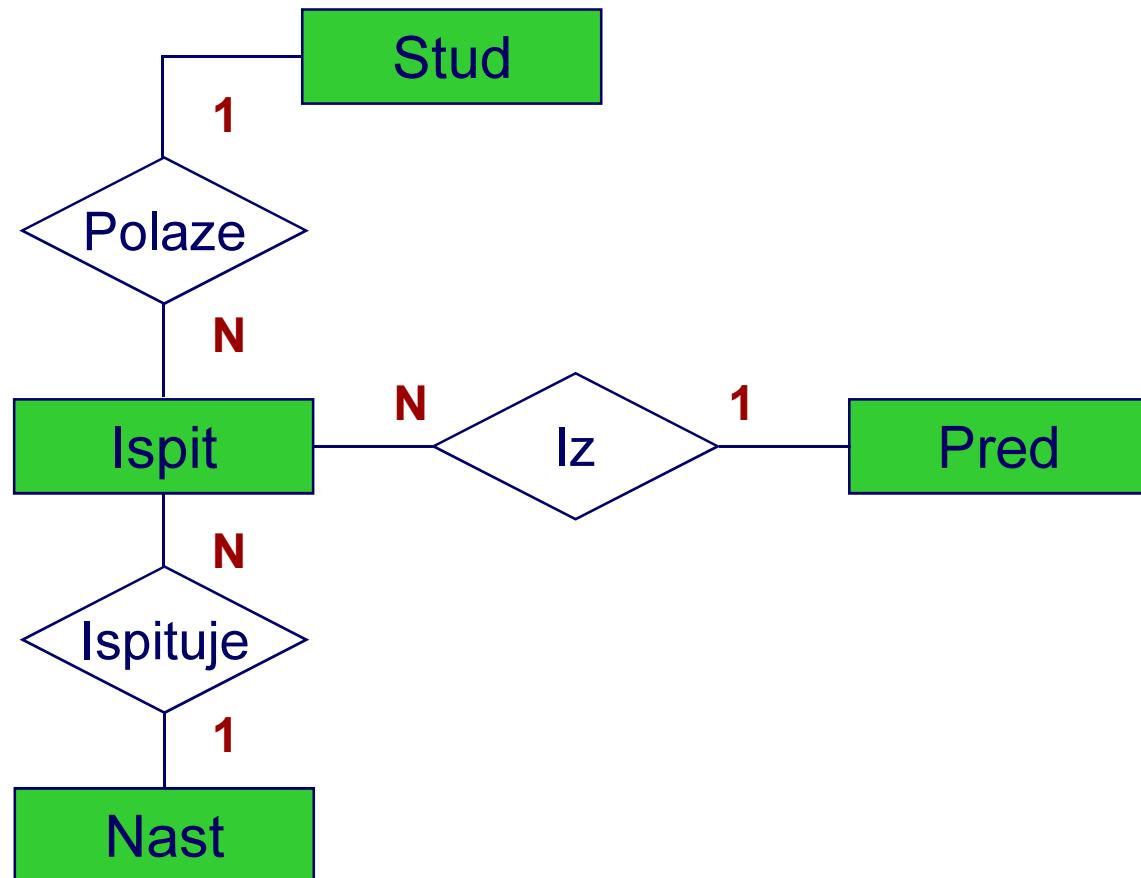
## Primjer: Model baze podataka za studentsku službu

---

- Oblikovati model baze podataka koja će omogućiti praćenje podataka o studentima, predmetima, nastavnicima i polaganjima ispita

# Primjer (nastavak)

---



## Primjer (nastavak)

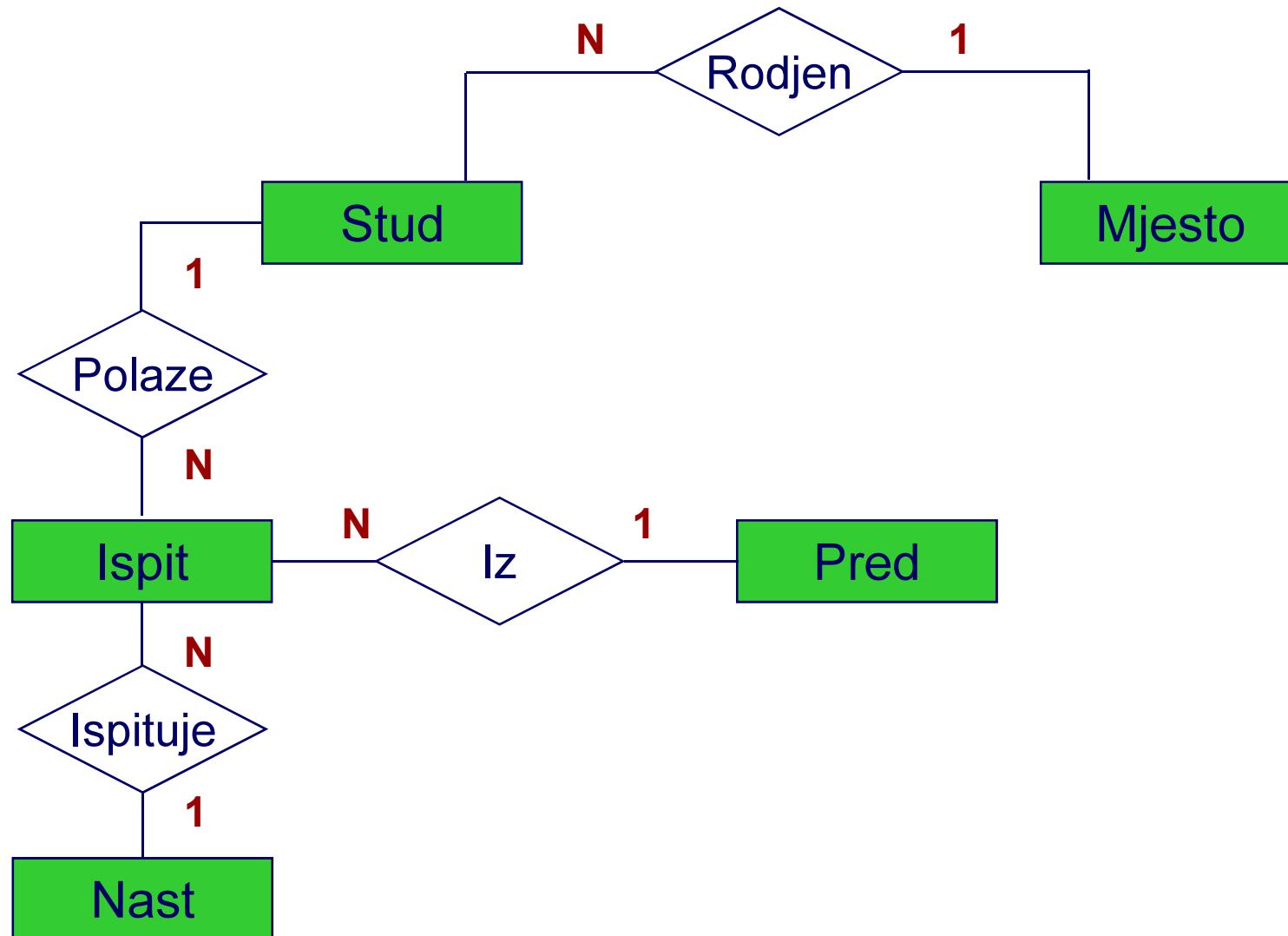
---

Stud = matBrStud, prezStud, imeStud, datRodStud

MJESTO ROĐENJA STUDENTA ???

# Primjer (nastavak)

---



## Primjer (nastavak)

---

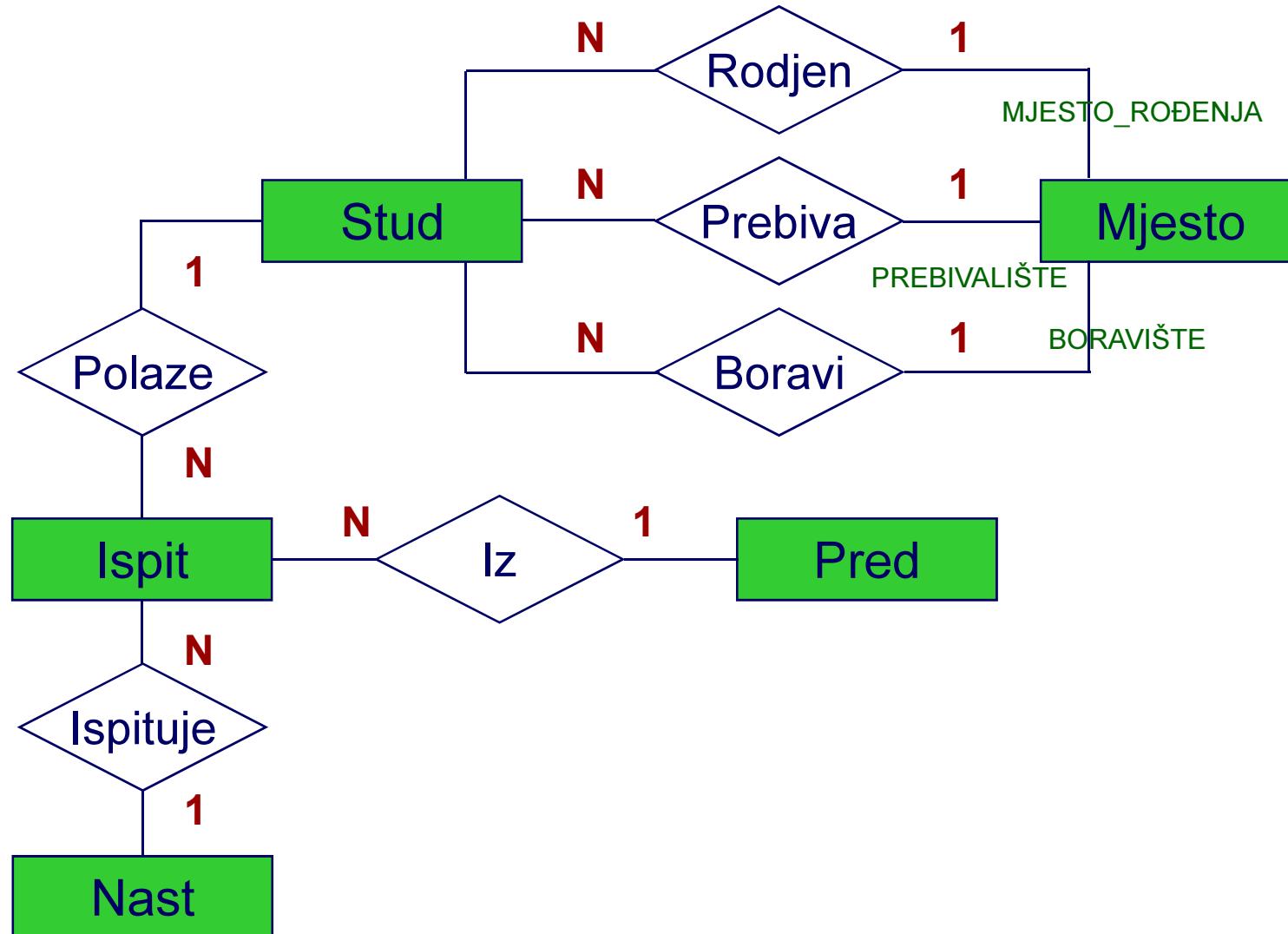
Stud = matBrStud, prezStud, imeStud, datRodStud

Mjesto = pbrMjesto, nazMjesto

PREBIVALIŠTE STUDENTA ???

BORAVIŠTE STUDENTA ???

# Primjer (nastavak)



## Primjer (nastavak)

---

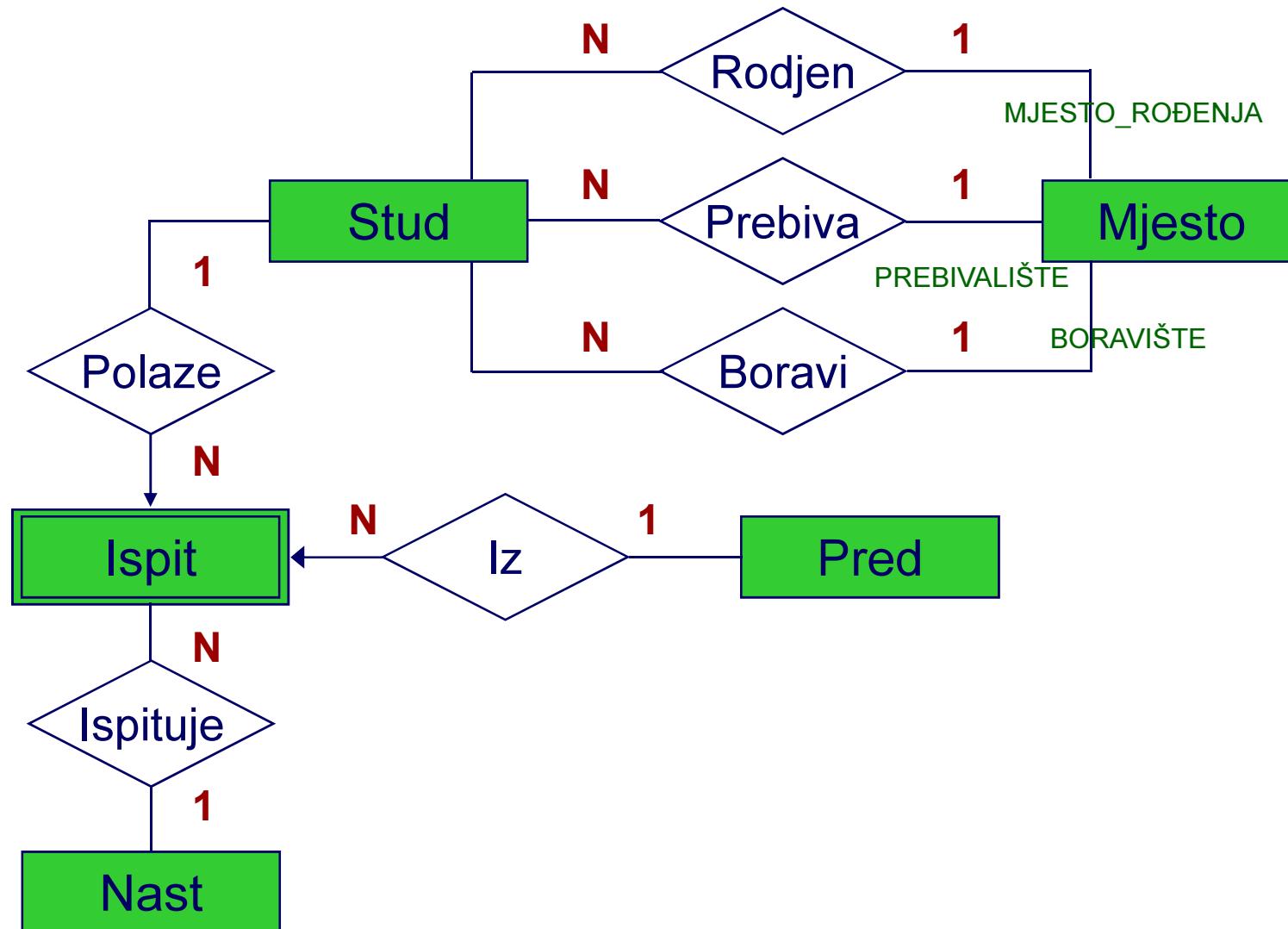
Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud,  
rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

SLABI ENTITET !!!!

# Primjer (nastavak)



## Primjer (nastavak)

---

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud,  
rangKlasIspitStud, eMailStud

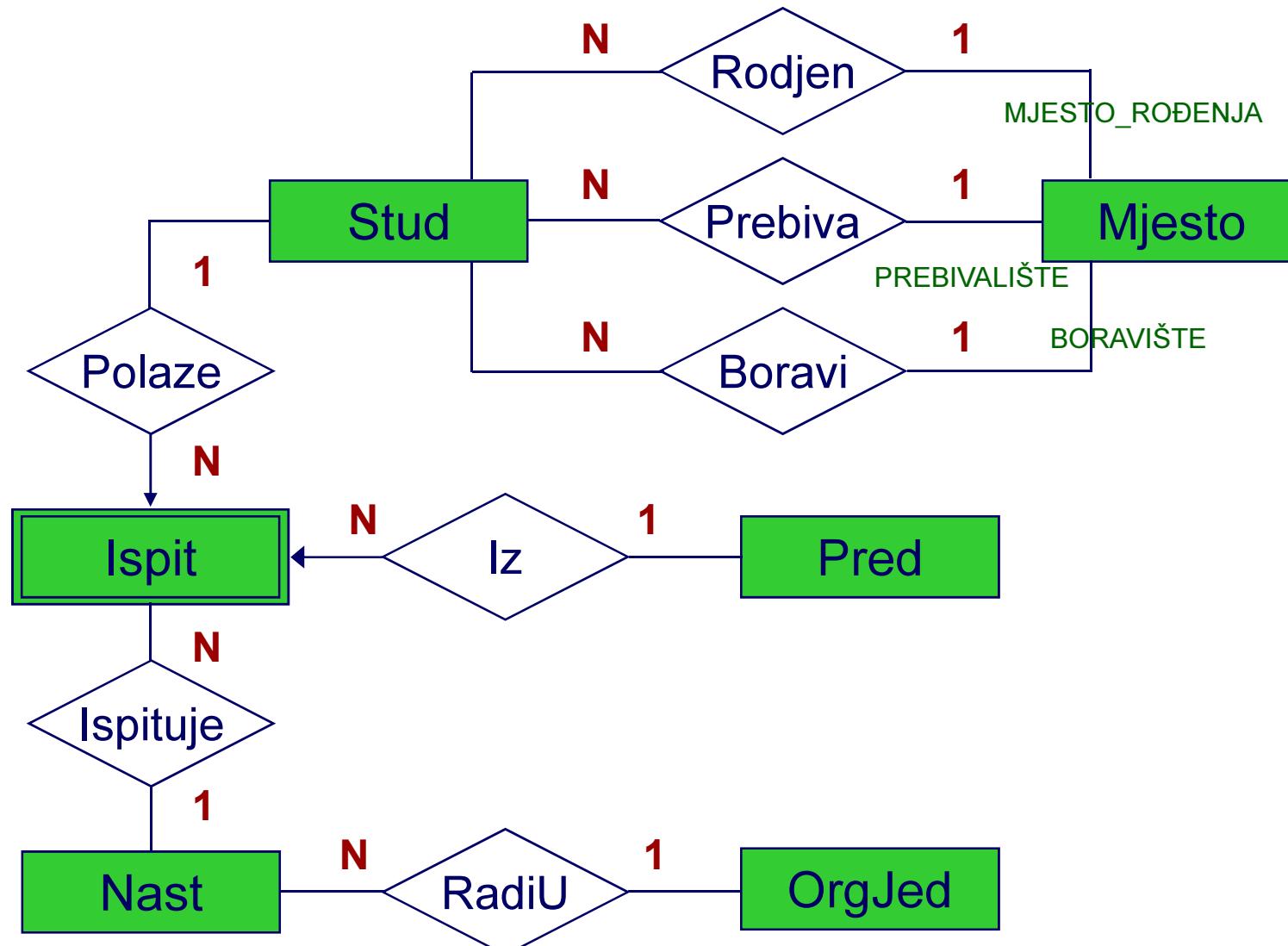
Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

Nast = sifraNast, prezNast, imeNast

ORGANIZACIJSKA JEDINICA ???

## Primjer (nastavak)



## Primjer (nastavak)

---

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud,  
rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

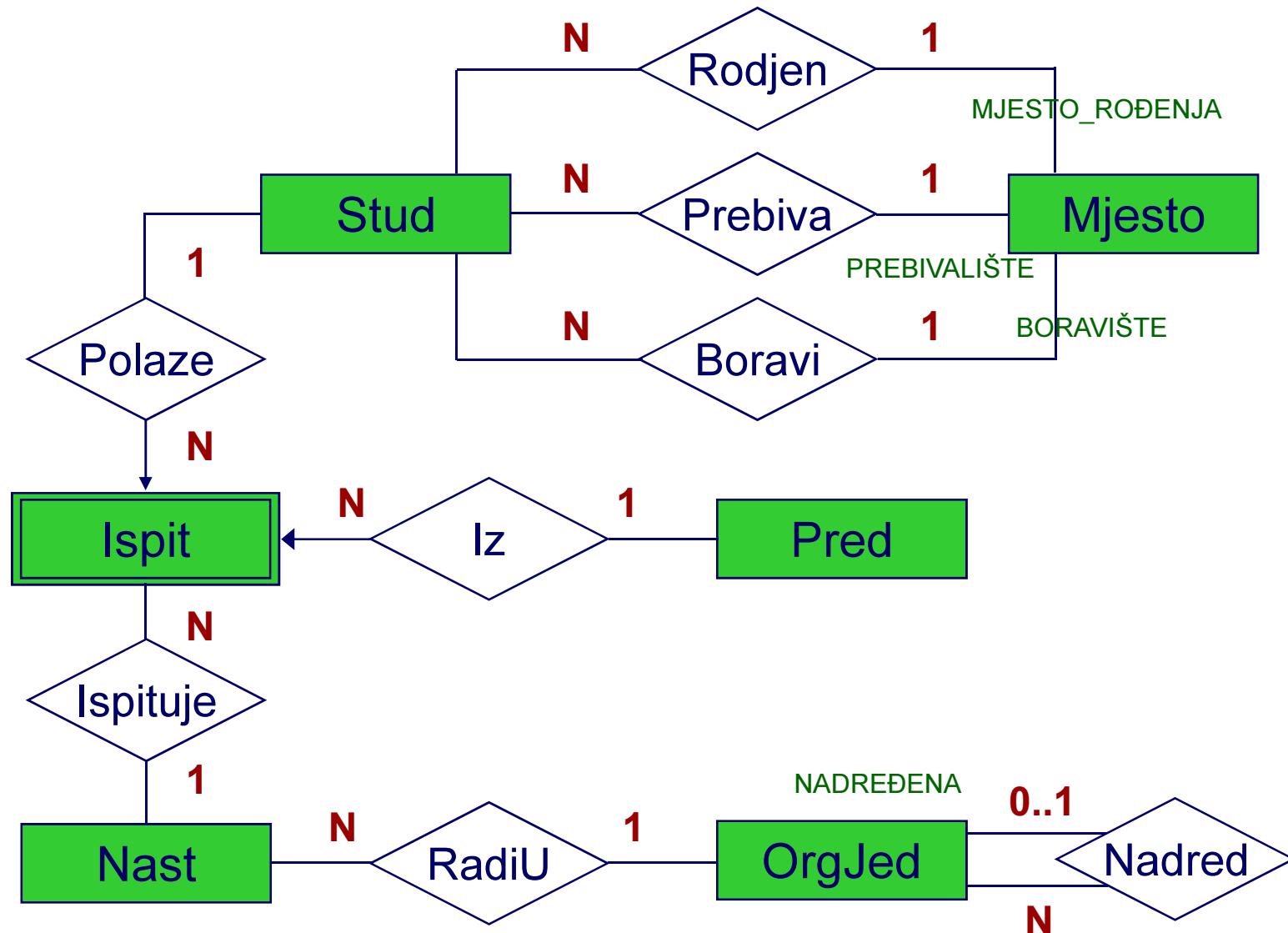
Ispit = matBrStud, sifraPred, datumIspit, ocjena

Nast = sifraNast, prezNast, imeNast

OrgJed = sifraOrgJed, nazivOrgJed

NADREĐENA ORGANIZACIJSKA JEDINICA ???

# Primjer (nastavak)



## Primjer (nastavak)

---

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud,  
rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

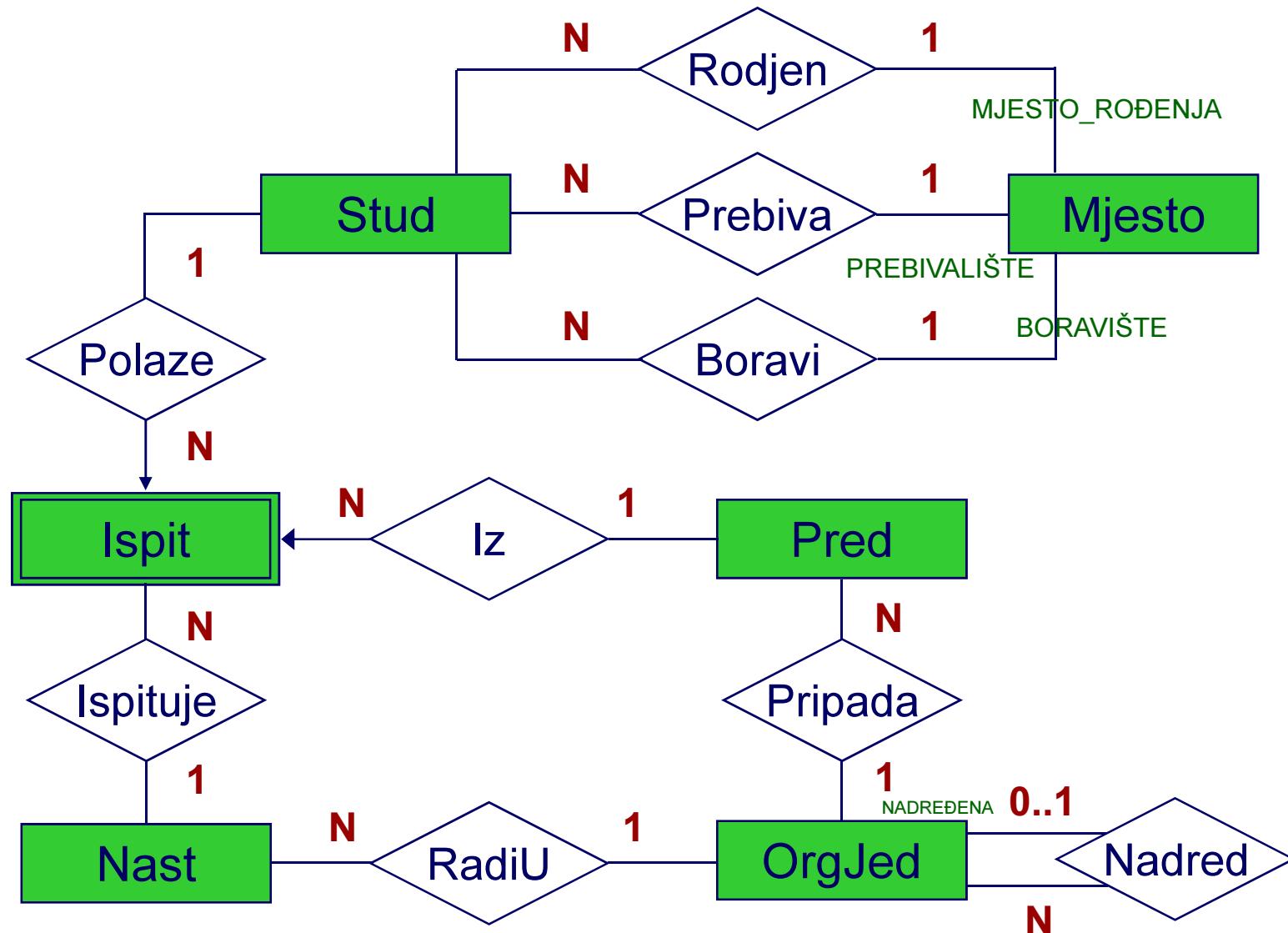
Nast = sifraNast, prezNast, imeNast, eMailNast, URLNast

OrgJed = sifraOrgJed, nazivOrgJed

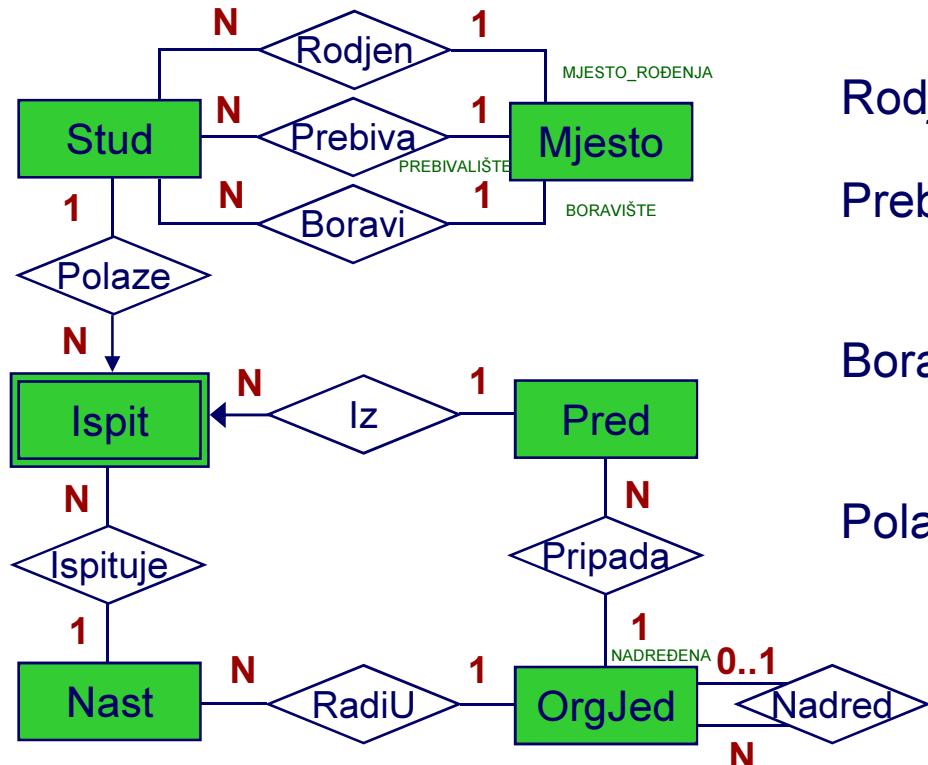
Pred = sifraPred, kraticaPred, nazivPred, URLPred

PREDMET PRIPADA ORGANIZACIJSKOJ JEDINICI ???

# Primjer (nastavak)



# Primjer (nastavak) - Opis veza



**Rodjen** = matBrStud, postBrMjRodStud

**Prebiva** = matBrStud, postBrMjPrebStud,  
adresaMjPrebStud

**Boravi** = matBrStud, postBrMjBorStud,  
adresaMjBorStud

**Polaze** = matBrStud, sifraPred, datumlispit

**Iz** = matBrStud, sifraPred, datumlispit

**Ispituje** = matBrStud, sifraPred, datumlispit, sifraNast

**RadiU** = sifraNast, sifraOrgJed

**Pripada** = sifraPred, sifraOrgJed

**Nadred** = sifraOrgJed, sifraNadOrgJed

## → Relacijski model

---

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud,  
rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

Nast = sifraNast, prezNast, imeNast, eMailNast, URLNast

OrgJed = sifraOrgJed, nazivOrgJed

Pred = sifraPred, kraticaPred, nazivPred, URLPred

Rodjen = matBrStud, postBrMjRodStud

Prebiva = matBrStud, postBrMjPrebStud, adresaMjPrebStud

Boravi = matBrStud, postBrMjBorStud, adresaMjBorStud

Polaze = matBrStud, sifraPred, datumIspit

Iz = matBrStud, sifraPred, datumIspit

Ispituje = matBrStud, sifraPred, datumIspit, sifraNast

RadiU = sifraNast, sifraOrgJed

Pripada = sifraPred, sifraOrgJed

Nadred = sifraOrgJed, sifraNadOrgJed

## → Relacijski model

---

### Unija shema s jednakim ključevima

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud,  
rangKlasIspitStud, eMailStud, postBrMjRodStud,  
postBrMjPrebStud, adresaMjPrebStud, postBrMjBorStud,  
adresaMjBorStud

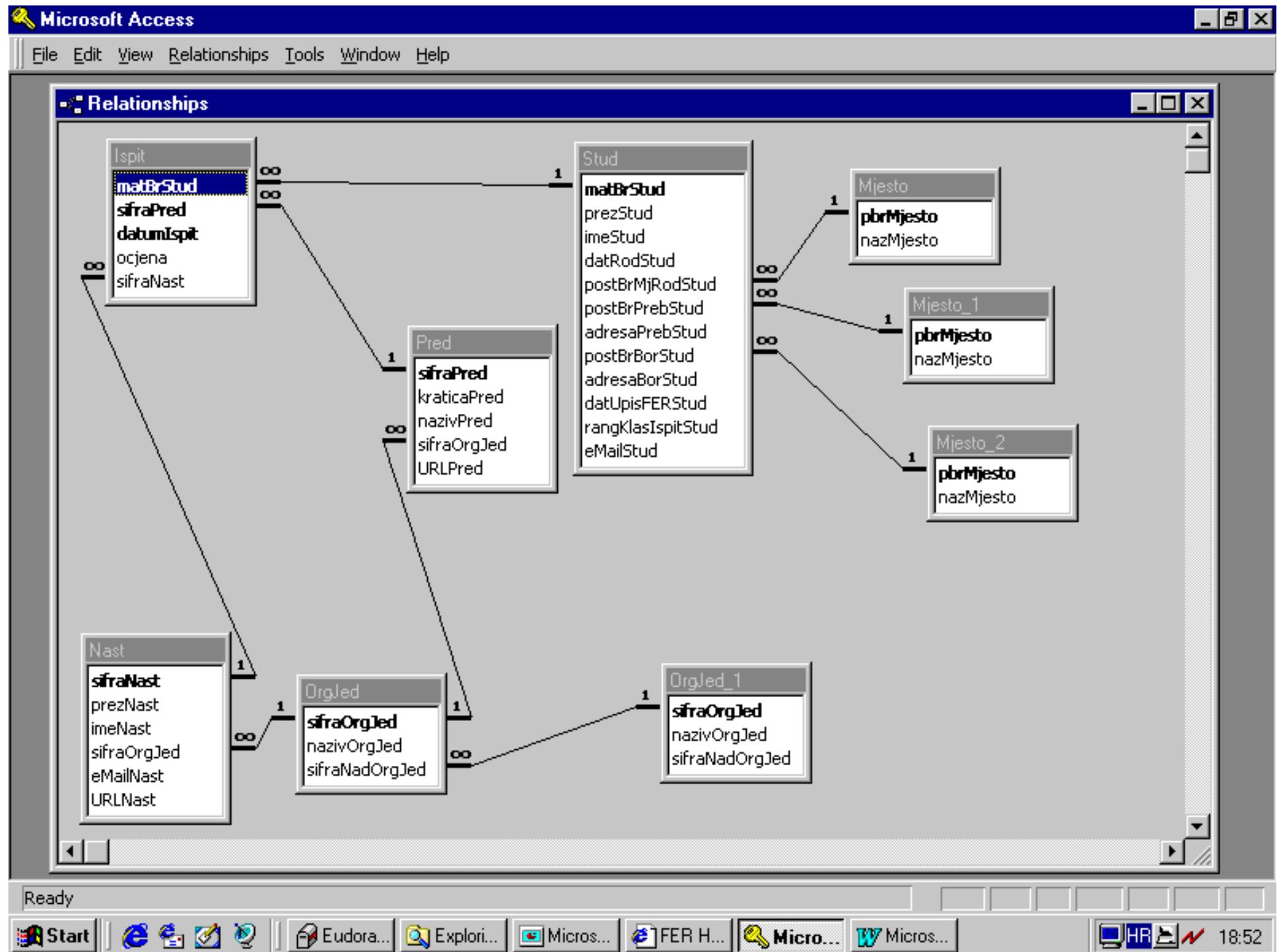
Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena, **sifraNast**

Nast = sifraNast, prezNast, imeNast, eMailNast, URLNast, **sifraOrgJed**

OrgJed = sifraOrgJed, nazivOrgJed, **sifraNadOrgJed**

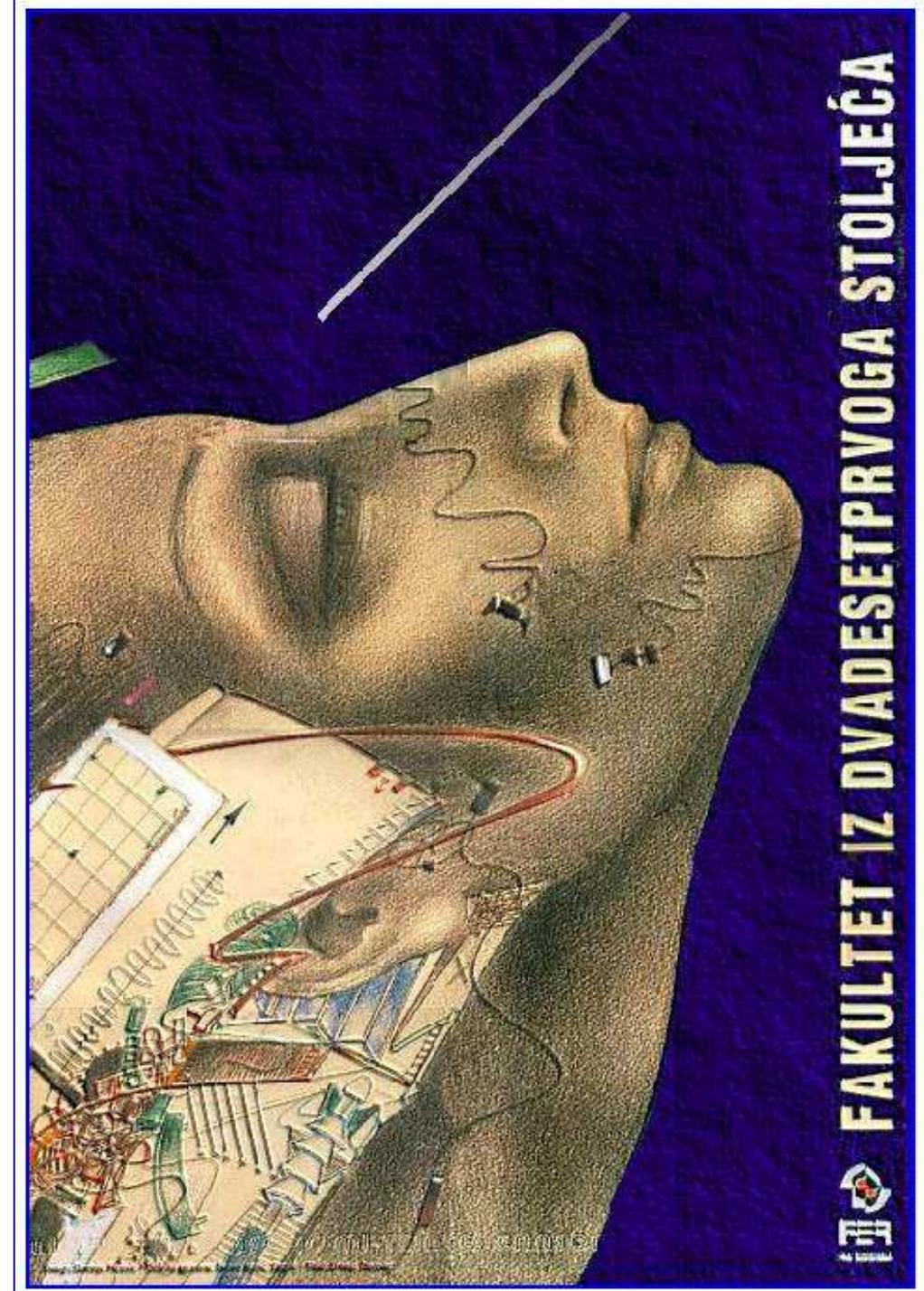
Pred = sifraPred, kraticaPred, nazivPred, URLPred, **sifraOrgJed**



# Baze podataka

Predavanja  
svibanj 2014.

## 15. ER model baze podataka (2. dio)



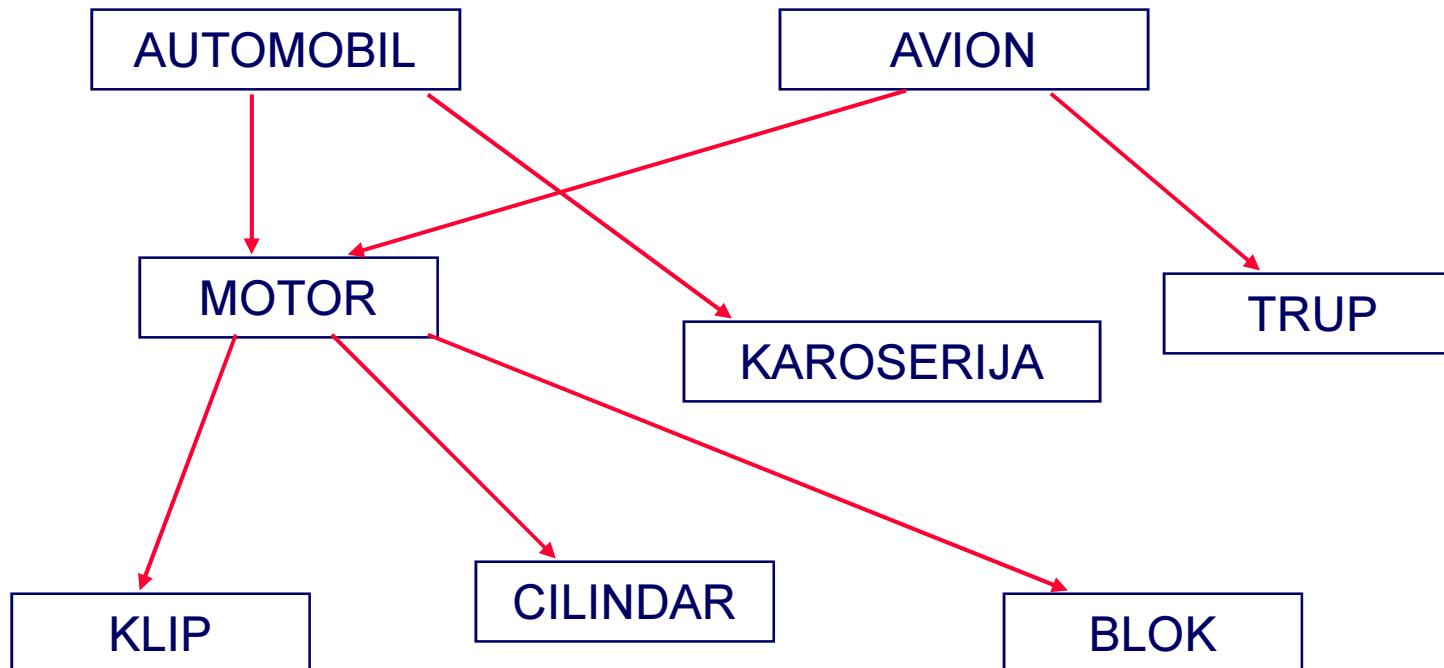
# Homogena mreža

Primjer: Sastavnica

AUTOMOBIL: MOTOR, KAROSERIJA

MOTOR: KLIP, CILINDAR, BLOK

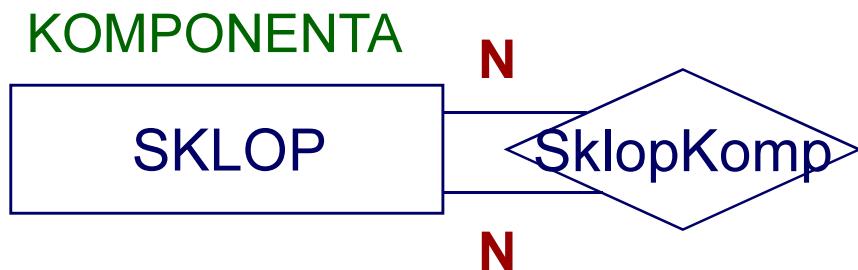
AVION: MOTOR, TRUP



Čvorovi u mreži imaju jednaku strukturu: SKLOP= sifSklop, nazSklop

# Refleksivne veze - preslikavanje N:N

---



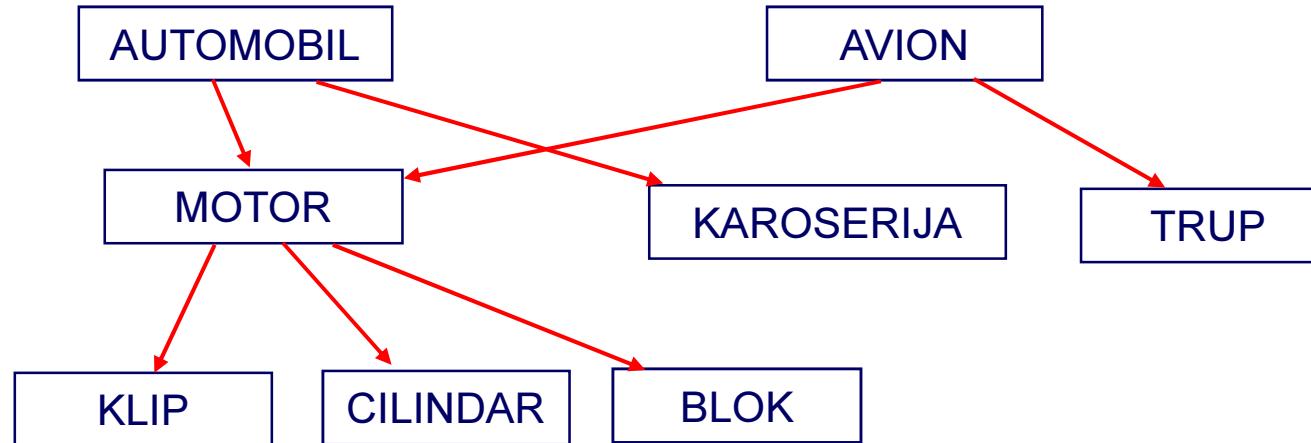
SKLOP = sifSklop, nazSklop

SklopKomp = ~~sifSklop, sifSklop~~

SklopKomp = sifSklop, sifKomp

Preimenovati  
jedan od atributa !

# Refleksivne veze - preslikavanje N:N



| Sklop    |            |
|----------|------------|
| sifSklop | nazSklop   |
| 17       | Automobil  |
| 19       | Motor      |
| 21       | Karoserija |
| 37       | Klip       |
| 49       | Cilindar   |
| 52       | Blok       |
| 64       | Avion      |
| 82       | Trup       |

| sklopKomp |         |
|-----------|---------|
| sifSklop  | sifKomp |
| 17        | 19      |
| 17        | 21      |
| 19        | 37      |
| 19        | 49      |
| 19        | 52      |
| 64        | 19      |
| 64        | 82      |

# Refleksivne veze N:N → relacijski model

---

SKLOP = sifSklop, nazSklop

SklopKomp = sifSklop, sifKomp

+ pravila integriteta

**Zadatak:** Ispisati naziv sklopa i naziv komponenti od kojih se sklop sastoji (ukoliko komponente sklopa postoje)

```
SELECT sklop.nazSklop
 , komponenta.nazSklop AS nazKomponenta
 FROM sklop AS komponenta
 INNER JOIN sklopKomp
 ON komponenta.sifSklop = sklopKomp.sifKomp
 RIGHT OUTER JOIN sklop
 ON sklopKomp.sifSklop = sklop.sifSklop;
```

# Problem

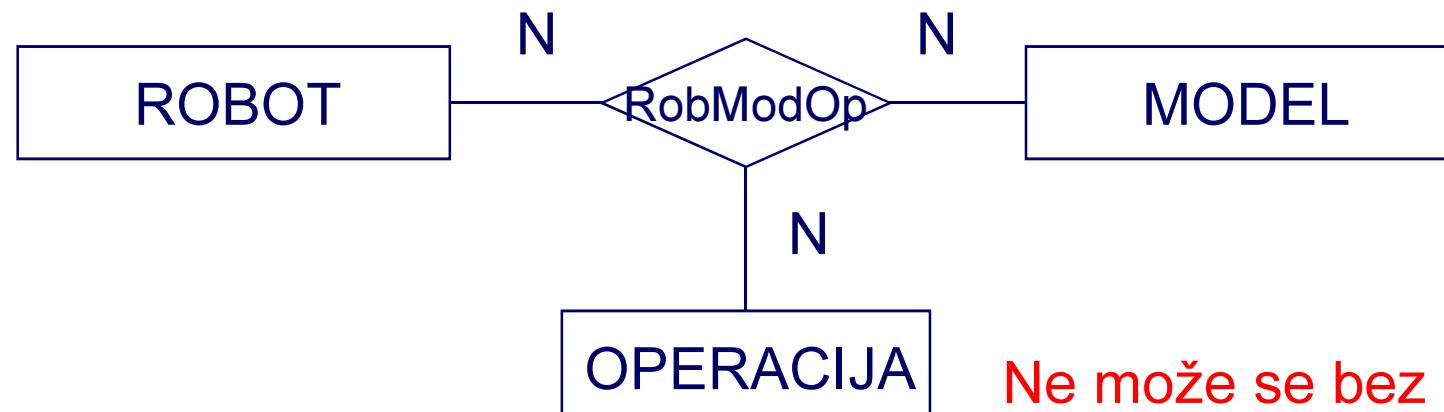
---

## Model proizvodnje:

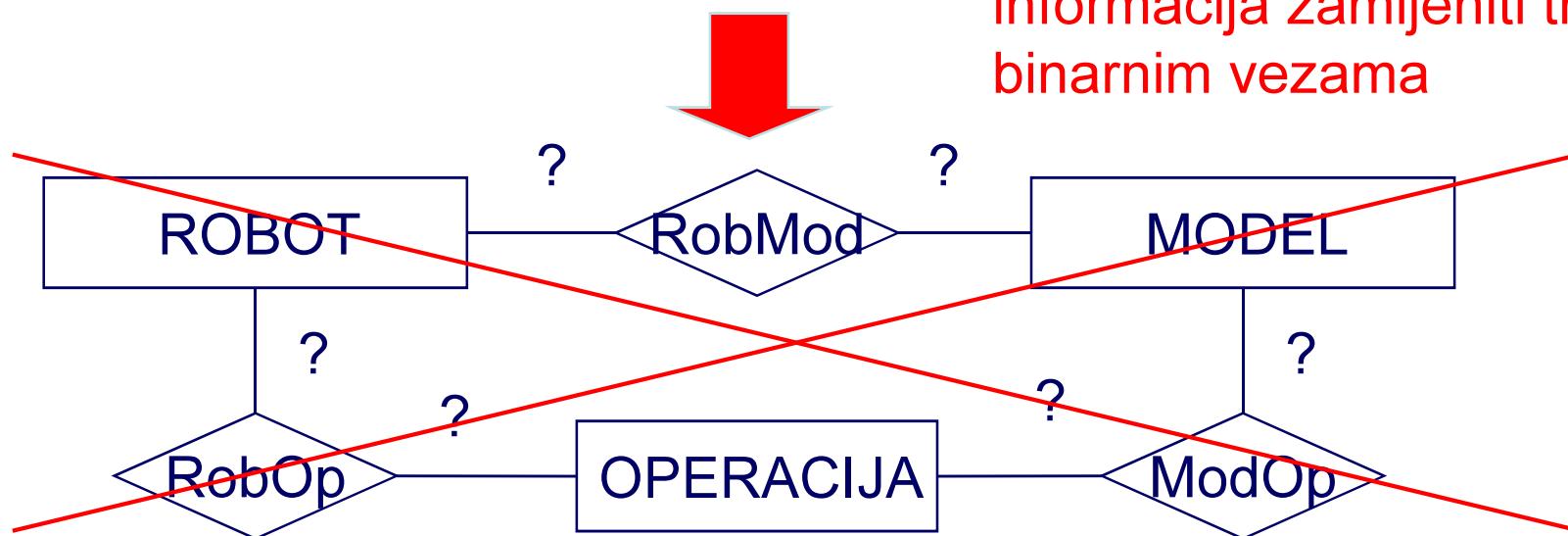
- robot R1 montira prednja lijeva vrata na modelu automobila Volvo S40 za 45 sekundi i pri tome utroši 0.8 kWh energije
- robot R2 oboji poklopac motora na modelu automobila Volvo S40 za 28 sekundi i pri tome utroši 0.4 kWh energije
- robot R1 montira prednja lijeva vrata na modelu automobila Volvo S60 za 52 sekunde i pri tome utroši 0.9 kWh energije
- robot R1 montira poklopac motora na modelu automobila Volvo S40 za 25 sekundi i pri tome utroši 0.75 kWh energije
- robot R2 montira prednja lijeva vrata na modelu automobila Volvo S40 za 40 sekundi i pri tome utroši 0.6 kWh energije
- robot R2 montira poklopac motora na modelu automobila Volvo S40 za 18 sekundi i pri tome utroši 0.7 kWh energije

# Ternarne veze

Ternarnom vezom prikazuje se istovremeni odnos triju entiteta.

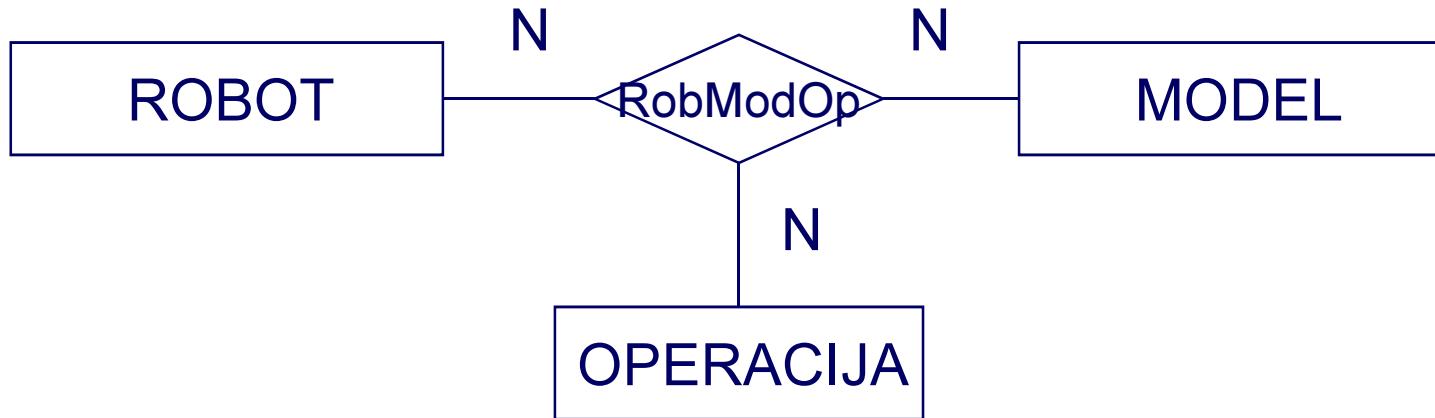


Ne može se bez gubitaka informacija zamijeniti trima binarnim vezama



# Ternarne veze - preslikavanje N:N:N

---



ROBOT = sifRobot, nazRobot, ...

MODEL = sifModel, nazModel, ...

OPERACIJA = sifOper, nazOper, ...

RobModOp = sifRobot, sifModel, sifOper, utrVrijeme, utrEnergija

# Ternarne veze N:N:N → relacijski model

ROBOT = sifRobot, nazRobot, ...

MODEL = sifModel, nazModel, ...

OPERACIJA = sifOper, nazOper, ...

+ pravila integriteta

RobModOp = sifRobot, sifModel, sifOper, utrVrijeme, utrEnergija

**Zadatak:** Ispisati naziv robota, naziv modela automobila, naziv operacije, utrošak vremena i energije, za sve operacije koje roboti mogu obaviti

```
SELECT nazRobot, nazModel, nazOper, utrVrijeme, utrEnergija
 FROM robModOp
 INNER JOIN robot
 ON robModOp.sifRobot = robot.sifRobot
 INNER JOIN model
 ON robModOp.sifModel = model.sifModel
 INNER JOIN operacija
 ON robModOp.sifOper = operacija.sifOper;
```

# Definicija 1. (Teorey)

---

U vezi koja povezuje entitete

$E_1, \dots, E_k, \dots, E_m$ ,

spojnost = 1 entiteta  $E_k$  znači da za svaku vrijednost svih entiteta  $E_1, \dots, E_m$ , osim  $E_k$ , uvijek postoji točno jedna vrijednost od  $E_k$ .

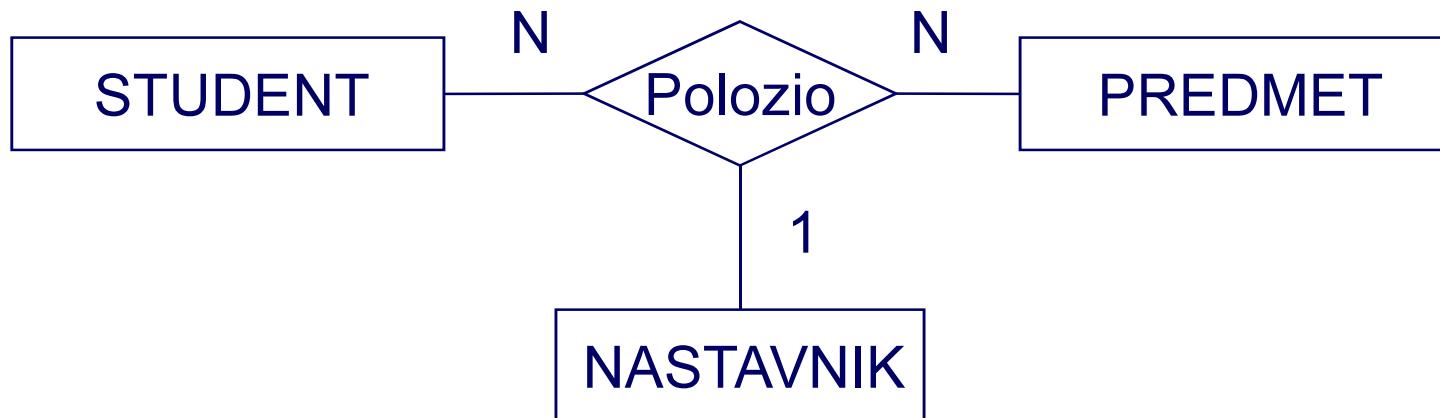
→ može se reći da tada vrijedi funkcija zavisnost:

$$\bigcup_{j=1}^m K_j \setminus K_k \rightarrow K_k$$

gdje su skupovi  $K_j$ , ( $j = 1, \dots, m$ ) ključevi entiteta  $E_1, \dots, E_m$

# Ternarne veze - preslikavanje N:N:1

---



STUDENT = matBrSt, prezSt, imeSt

PREDMET = sifPred, nazPred

NASTAVNIK = sifNast, prezNast, imeNast

Polozio = matBrSt, sifPred, sifNast, ocjena

# Ternarne veze N:N:1 → relacijski model

---

STUDENT = matBrSt, prezSt, imeSt

PREDMET = sifPred, nazPred

NASTAVNIK = sifNast, prezNast, imeNast

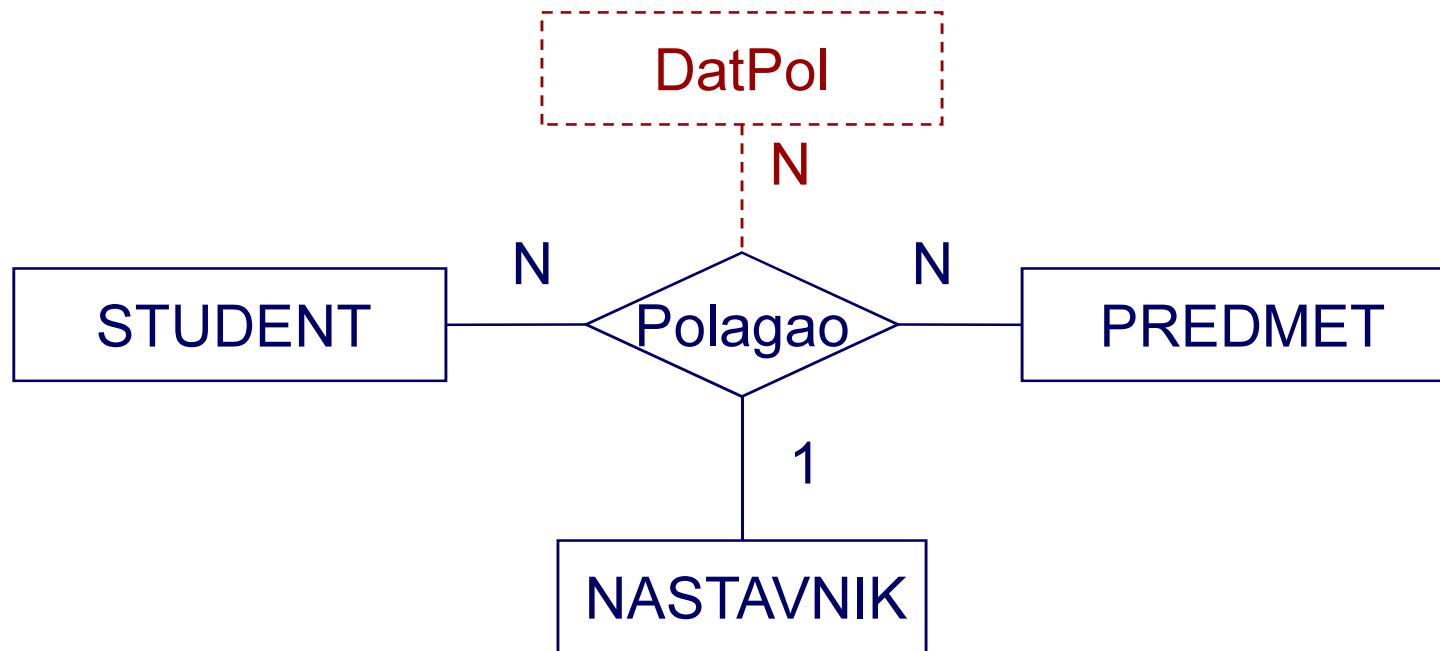
Položio = matBrSt, sifPred, sifNast, ocjena

+ pravila integriteta

```
SELECT prezSt, imeSt, nazPred, prezNast, imeNast, ocjena
 FROM polozio
 INNER JOIN student
 ON polozio.matBrSt = student.matBrSt
 INNER JOIN predmet
 ON polozio.sifPred = predmet.sifPred
 INNER JOIN nastavnik
 ON polozio.sifNast = nastavnik.sifNast;
```

# Ternarne veze - preslikavanje N:N:1

---



**STUDENT** = matBrSt, prezSt, imeSt

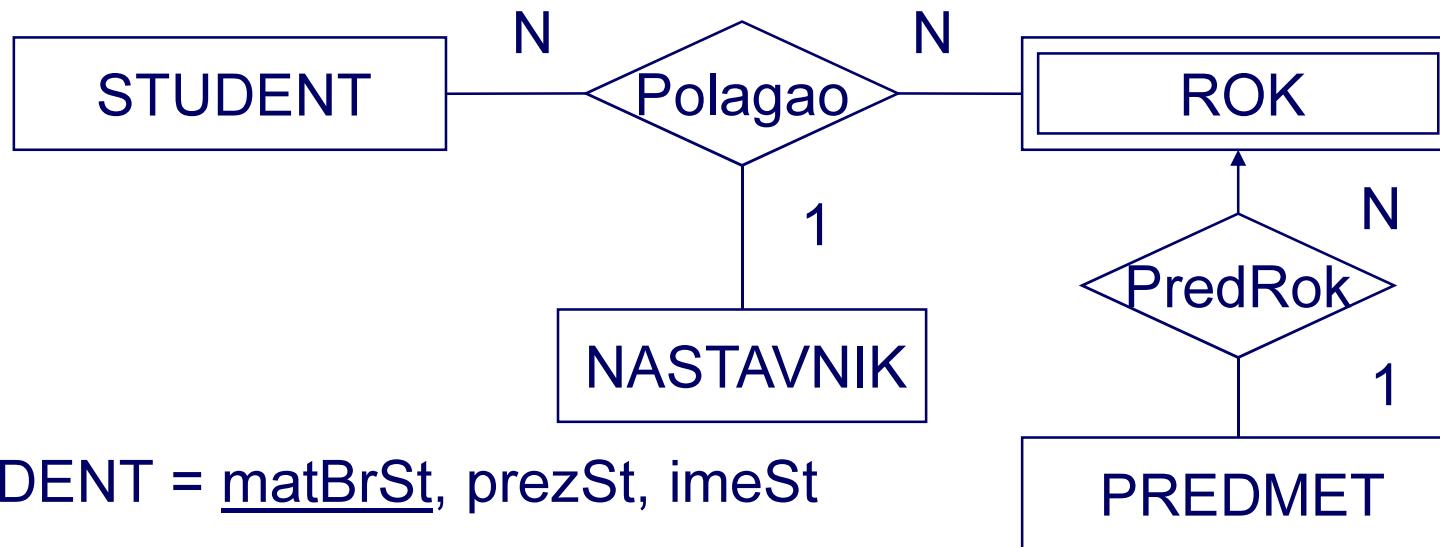
**PREDMET** = sifPred, nazPred

**NASTAVNIK** = sifNast, prezNast, imeNast

**Polagao** = matBrSt, sifPred, datPol, sifNast, ocjena

# Ternarne veze - preslikavanje N:N:1

---



STUDENT = matBrSt, prezSt, imeSt

PREDMET = sifPred, nazPred

ROK = sifPred, datRok, vrstaRok

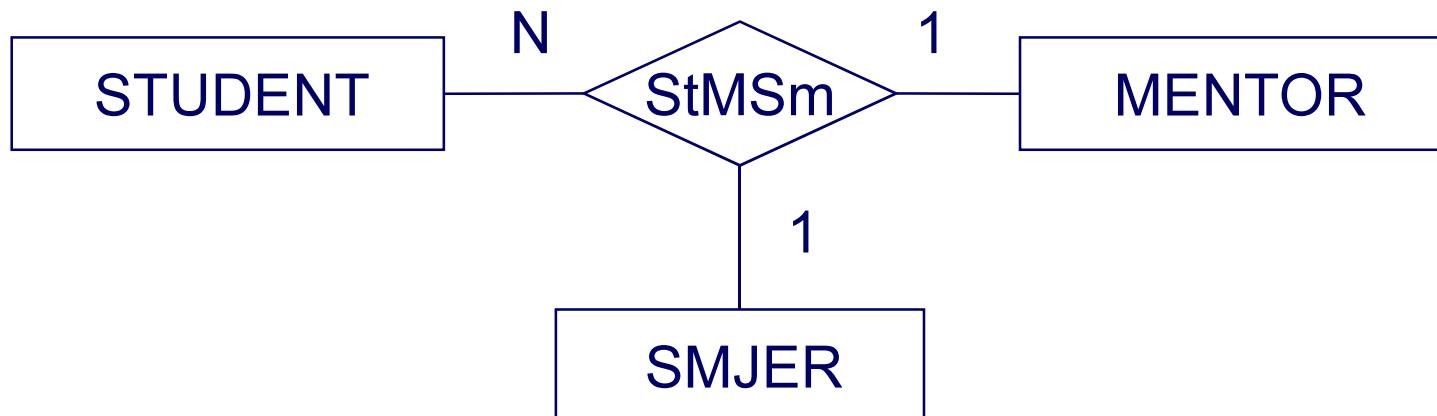
NASTAVNIK = sifNast, prezNast, imeNast

PredRok = sifPred, datRok

Polagao = matBrSt, sifPred, datRok, sifNast, ocjena

# Ternarne veze - preslikavanje N:1:1

---



Student može studirati na više smjerova, ali na svakom smjeru mora imati različitog mentora. Student na svakom smjeru ima samo jednog mentora.

STUDENT = matBrSt, prezSt, imeSt

SMJER = sifSmjer, nazSmjer

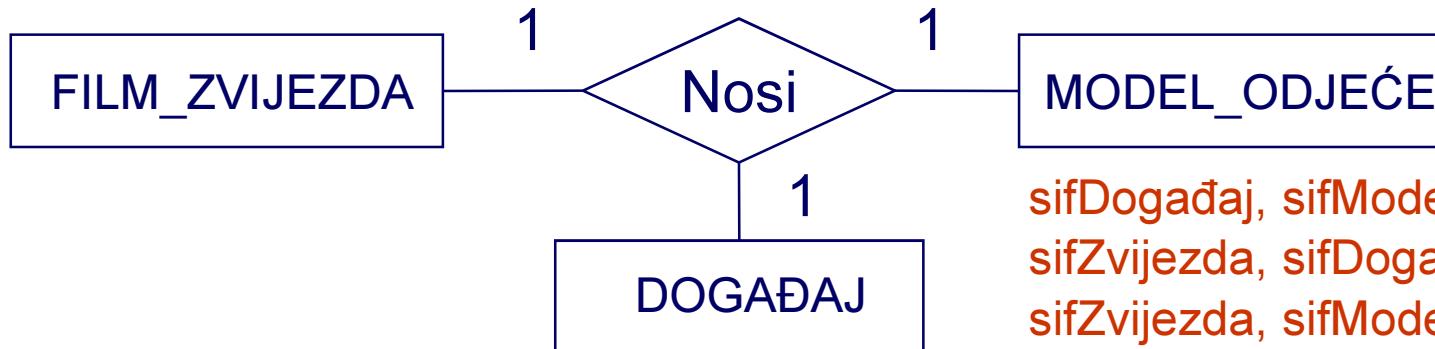
MENTOR = sifMentor, prezMentor, imeMentor

StMSm = sifSmjer, matBrSt, sifMentor

---

# Ternarne veze - preslikavanje 1:1:1

Na jednom događaju (npr. Dodjela Oscara 2008.) ne smiju dvije ili više filmskih zvijezda nositi isti model odjeće. Tijekom jednog događaja, jedna zvijezda nosi samo jedan model odjeće. Jedna zvijezda smije jedan model odjeće nositi na samo jednom događaju.



sifDogađaj, sifModel → sifZvijezda  
sifZvijezda, sifDogađaj → sifModel  
sifZvijezda, sifModel → sifDogađaj

FILM\_ZVIJEZDA = sifZvijezda, ime, prezime

MODEL\_ODJEĆE = sifModel, nazModel

DOGAĐAJ = sifDogađaj, nazDogađaj, datumDogađaj

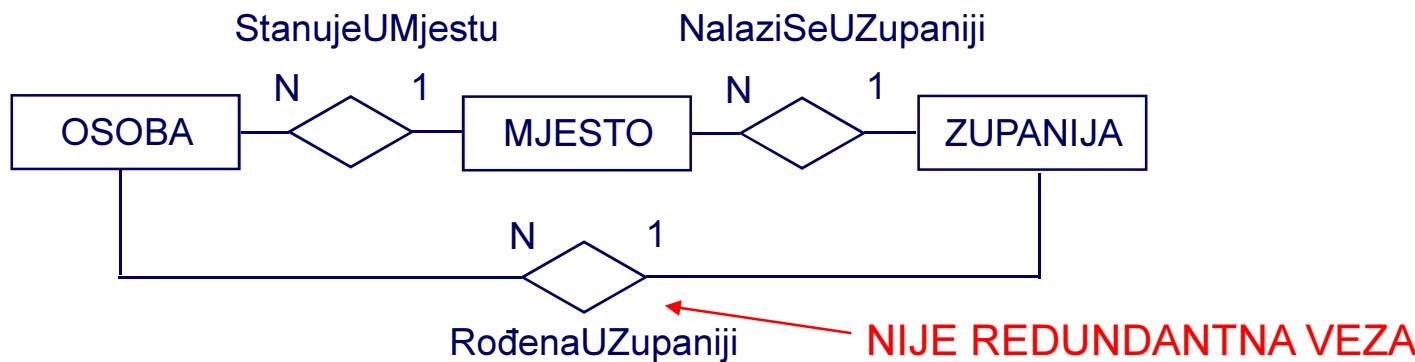
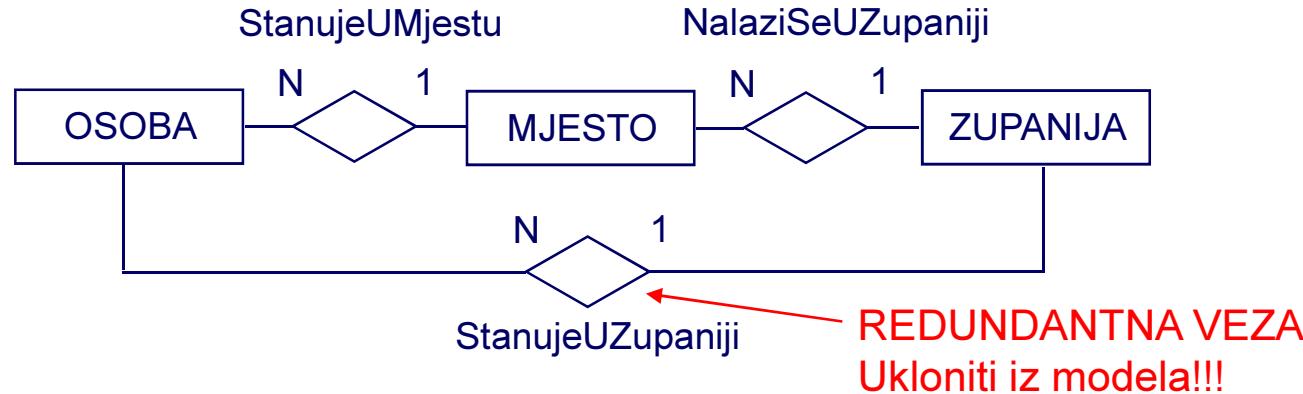
Nosi = sifZvijezda, sifDogađaj, sifModel

---

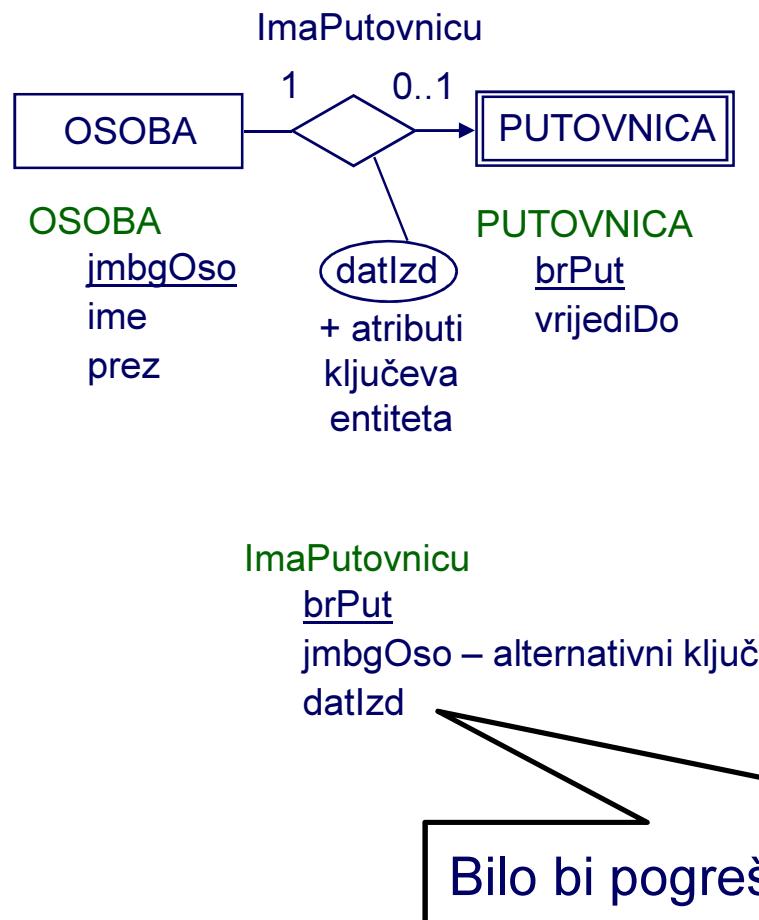
---

# Redundantne veze

---



# Primjeri preslikavanja u relacijski model

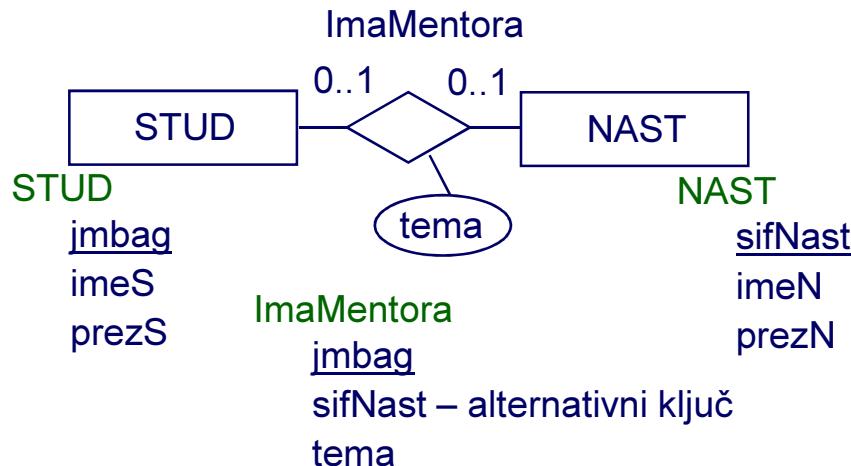


```
CREATE TABLE osoba (
 jmbgOso ...
 , ime ...
 , prez ...
 , PRIMARY KEY (jmbgOso));
```

```
CREATE TABLE putovnica (
 brPut ...
 , vrijediDo ...
 , datIzd ...
 , jmbgOso ... NOT NULL
 , PRIMARY KEY (brPut)
 , UNIQUE (jmbgOso)
 , FOREIGN KEY (jmbgOso)
 REFERENCES osoba (jmbgOso));
```

Bilo bi pogrešno kao **primarni ključ** odabrati **jmbgOso**.

# Primjeri preslikavanja u relacijski model



```
CREATE TABLE stud (
 jmbag ...
 , imeS ...
 , prezS ...
 , sifNast ...
 , tema ...
 , PRIMARY KEY (jmbag)
 , FOREIGN KEY (sifNast)
 REFERENCES nast(sifNast));

```

```
CREATE TABLE nast (
 sifNast ...
 , imeN ...
 , prezN ...
 , PRIMARY KEY (sifNast));

```

ILI

ILI

Pretpostavlja se da jedan nastavnik može biti mentor najviše jednom studentu (ili niti jednom), te da student može imati najviše jednog mentora (ili niti jednog).

ImaMentora  
sifNast  
jmbag – alternativni ključ  
tema

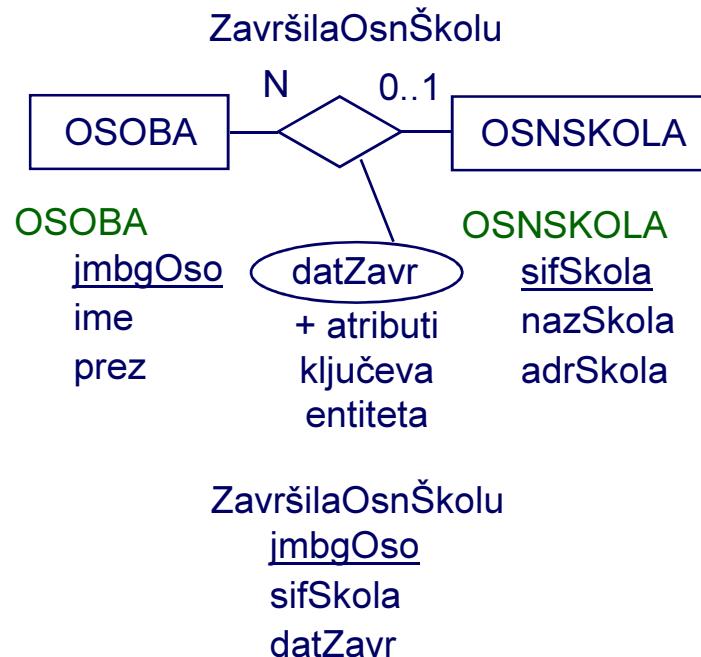
```
CREATE TABLE stud (
 jmbag ...
 , imeS ...
 , prezS ...
 , PRIMARY KEY (jmbag));

```

```
CREATE TABLE nast (
 sifNast ...
 , imeN ...
 , prezN ...
 , jmbag ...
 , tema ...
 , PRIMARY KEY (sifNast)
 , FOREIGN KEY (jmbag)
 REFERENCES stud (jmbag));

```

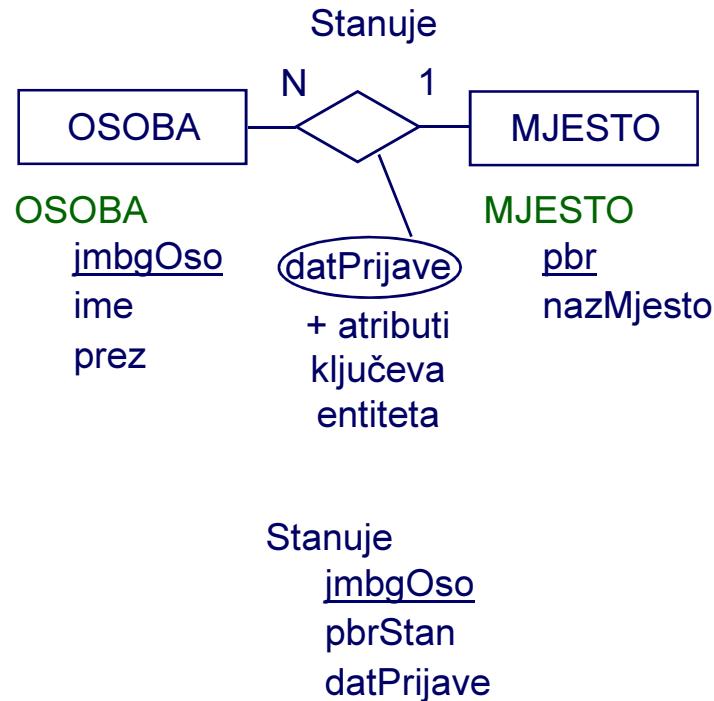
# Primjeri preslikavanja u relacijski model



```
CREATE TABLE osoba (
 jmbgOso ...
 , ime ...
 , prez ...
 , sifSkola ...
 , datZavr ...
 , PRIMARY KEY (jmbgOso)
 , FOREIGN KEY (sifSkola)
 REFERENCES osnSkola(sifSkola));
```

```
CREATE TABLE osnSkola (
 sifSkola ...
 , nazSkola ...
 , adrSkola ...
 , PRIMARY KEY (sifSkola));
```

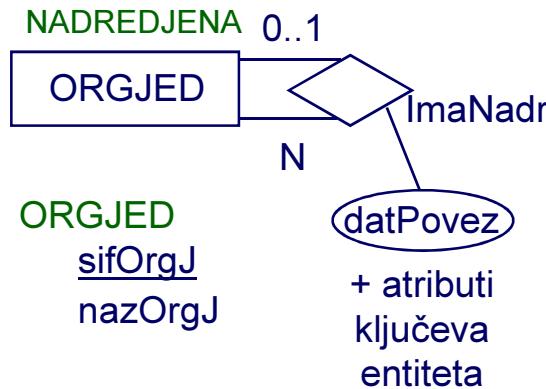
# Primjeri preslikavanja u relacijski model



```
CREATE TABLE mjesto (
 pbr ...
 , nazMjesto ...
 , PRIMARY KEY (pbr));
```

```
CREATE TABLE osoba (
 jmbgOso ...
 , ime ...
 , prez ...
 , pbrStan ... NOT NULL
 , datPrijave ...
 , PRIMARY KEY (jmbgOso)
 , FOREIGN KEY (pbrStan)
 REFERENCES mjesto(pbr));
```

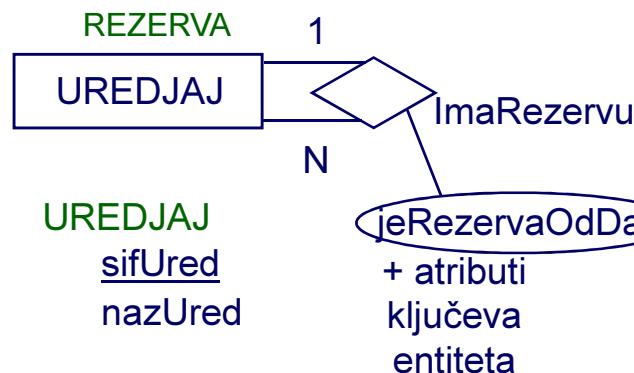
# Primjeri preslikavanja u relacijski model



imaNadr  
  sifOrgJ  
  sifNadOrgJ  
  datPovez

```
CREATE TABLE orgJed (
 sifOrgJ ...
, nazOrgJ ...
, sifNadOrgJ ...
, datPovez ...
, PRIMARY KEY (sifOrgJ)
, FOREIGN KEY (sifNadOrgJ)
 REFERENCES orgjed (sifOrgJ));
```

Za svaki uređaj mora biti evidentiran točno jedan rezervni uređaj. Jedan uređaj može biti rezerva za nekoliko uređaja. Evidentira se datum kad je uređaj postao rezerva nekog drugog uređaja

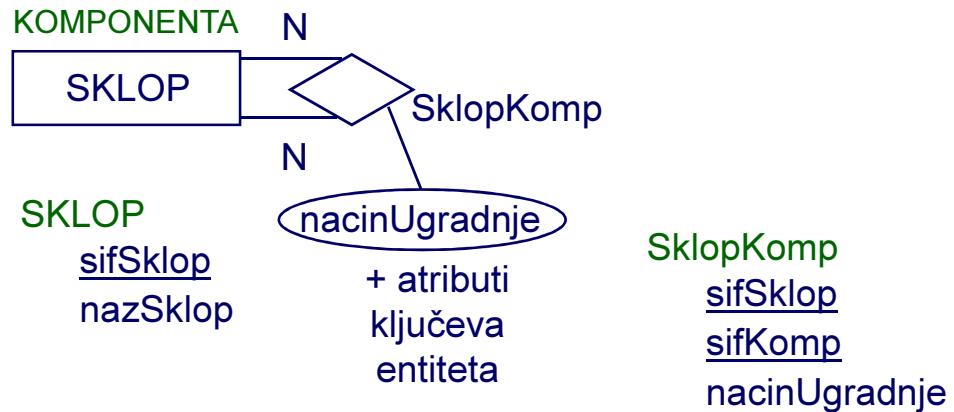


imaRezervu  
  sifUred  
  sifRezUred  
  jeRezervaOdDat

```
CREATE TABLE uredjaj (
 sifUred ...
, nazUred ...
, sifRezUred ... NOT NULL
, jeRezervaOdDat ...
, PRIMARY KEY (sifUred)
, FOREIGN KEY (sifRezUred)
 REFERENCES uredjaj (sifUred));
```

Zadatak: kako osigurati da uređaj ne bude sam sebi rezerva?

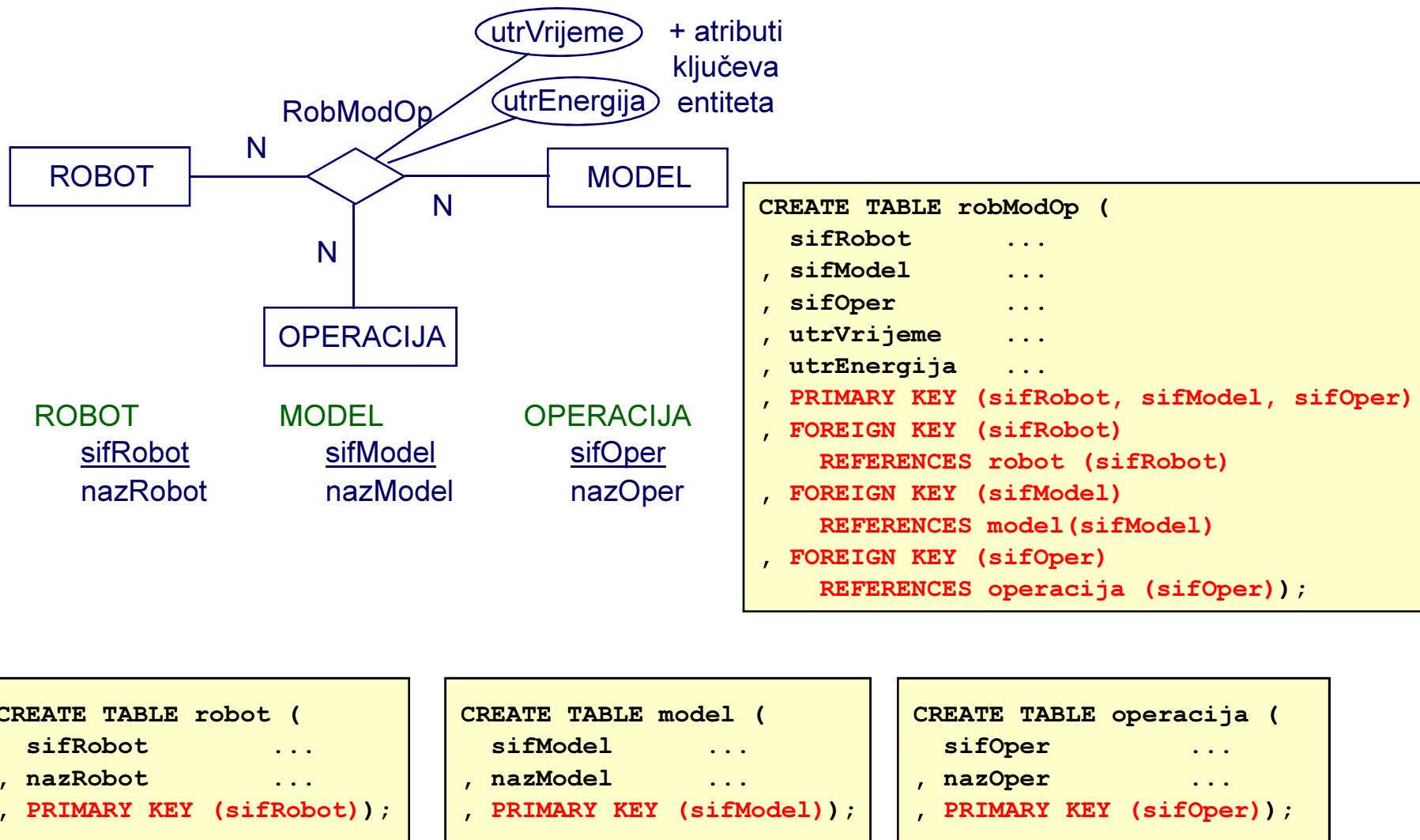
# Primjeri preslikavanja u relacijski model



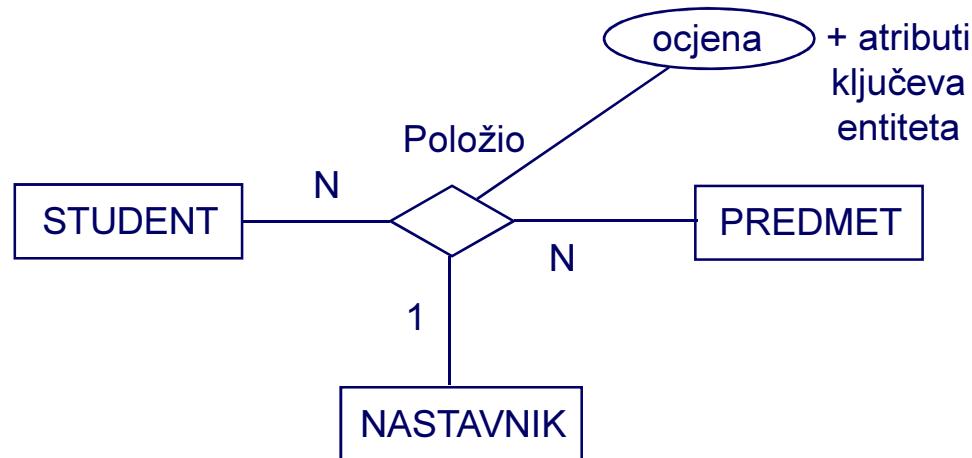
```
CREATE TABLE sklop (
 sifSklop ...
 , nazSklop ...
 , PRIMARY KEY (sifSklop));
```

```
CREATE TABLE sklopKomp (
 sifSklop ...
 , sifKomp ...
 , nacinUgradnje ...
 , PRIMARY KEY (sifSklop, sifKomp)
 , FOREIGN KEY (sifSklop)
 REFERENCES sklop(sifSklop)
 , FOREIGN KEY (sifKomp)
 REFERENCES sklop (sifSklop));
```

# Primjeri preslikavanja u relacijski model



# Primjeri preslikavanja u relacijski model



STUDENT  
mbrSt  
prezSt  
imeSt

PREDMET  
sifPred  
nazPred

NASTAVNIK  
sifNast  
prezNast  
imeNast

```
CREATE TABLE polozio (
 mbrSt ...
 , sifPred ...
 , sifNast ... NOT NULL
 , ocjena ...
 , PRIMARY KEY (mbrSt, sifPred)
 , FOREIGN KEY (mbrSt)
 REFERENCES student (mbrSt)
 , FOREIGN KEY (sifPred)
 REFERENCES predmet (sifPred)
 , FOREIGN KEY (sifNast)
 REFERENCES nastavnik (sifNast));

```

```
CREATE TABLE student (
 mbrSt ...
 , prezSt ...
 , imeSt ...
 , PRIMARY KEY (mbrSt));

```

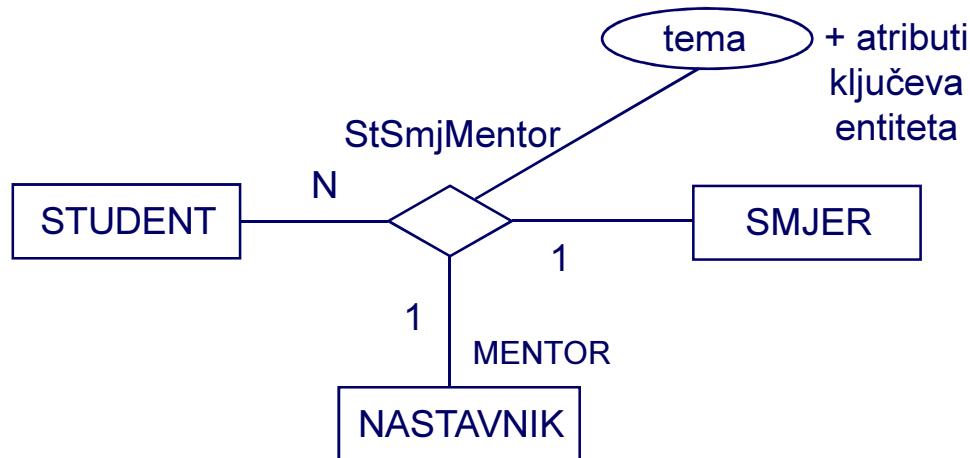
```
CREATE TABLE predmet (
 sifPred ...
 , nazPred ...
 , PRIMARY KEY (sifPred));

```

```
CREATE TABLE nastavnik (
 sifNast ...
 , prezNast ...
 , imeNast ...
 , PRIMARY KEY (sifNast));

```

# Primjeri preslikavanja u relacijski model



**STUDENT**  
mbrSt  
prezSt  
imeSt

**SMJER**  
sifSmjer  
nazSmjer

**NASTAVNIK**  
sifNast  
prezNast  
imeNast

```
CREATE TABLE stSmjMentor (
 mbrSt ...
 , sifSmjer ...
 , sifNast ... NOT NULL
 , tema ...
 , PRIMARY KEY (mbrSt, sifSmjer)
 , UNIQUE (mbrSt, sifNast)
 , FOREIGN KEY (mbrSt)
 REFERENCES student(mbrSt)
 , FOREIGN KEY (sifSmjer)
 REFERENCES smjer(sifSmjer)
 , FOREIGN KEY (sifNast)
 REFERENCES nastavnik(sifNast));
```

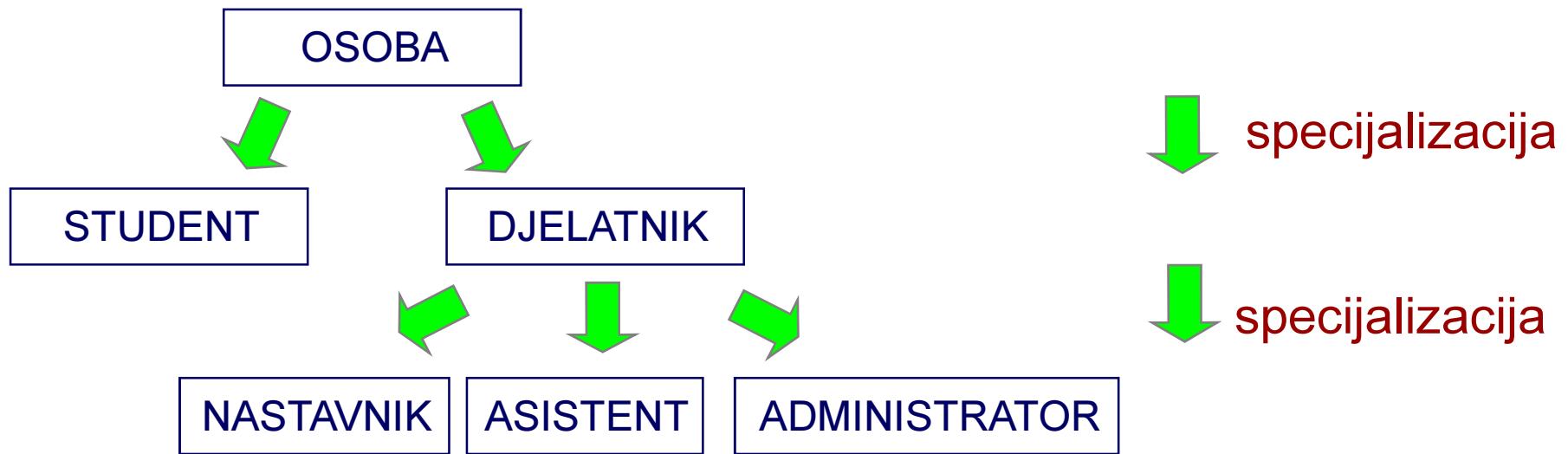
```
CREATE TABLE student (
 mbrSt ...
 , prezSt ...
 , imeSt ...
 , PRIMARY KEY (mbrSt));
```

```
CREATE TABLE smjer (
 sifSmjer ...
 , nazSmjer ...
 , PRIMARY KEY (sifSmjer));
```

```
CREATE TABLE nastavnik (
 sifNast ...
 , prezNast ...
 , imeNast ...
 , PRIMARY KEY (sifNast));
```

# Specijalizacija

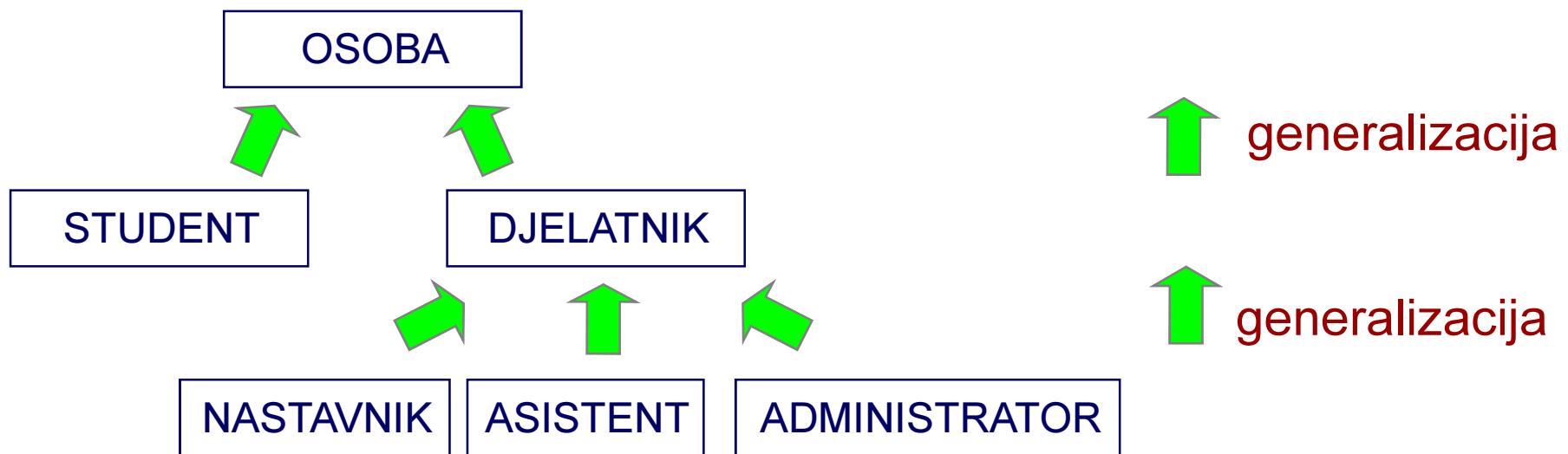
Entiteti jednog skupa entiteta mogu se temeljem njihovih karakterističnih svojstava klasificirati u zasebne skupove entiteta, postupkom koji se naziva specijalizacija



Skupovi entiteta dobiveni postupkom specijalizacije nazivaju se podklase (*subclasses*) ili specijalizacije

# Generalizacija

Entiteti iz nekoliko skupova entiteta sa sličnim svojstvima mogu se grupirati u zajednički skup entiteta, postupkom koji se naziva generalizacija

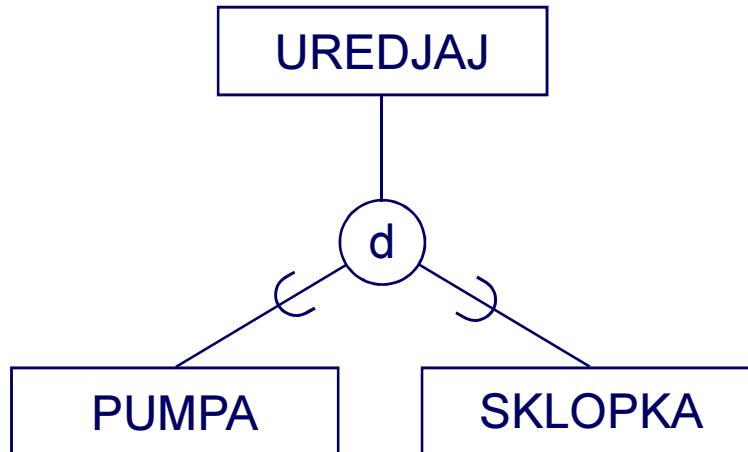


Skupovi entiteta dobiveni postupkom generalizacije nazivaju se nadklase (*superclasses*) ili generalizacije

Postupak generalizacije je inverzan postupku specijalizacije

# Generalizacija i specijalizacija - ER dijagram

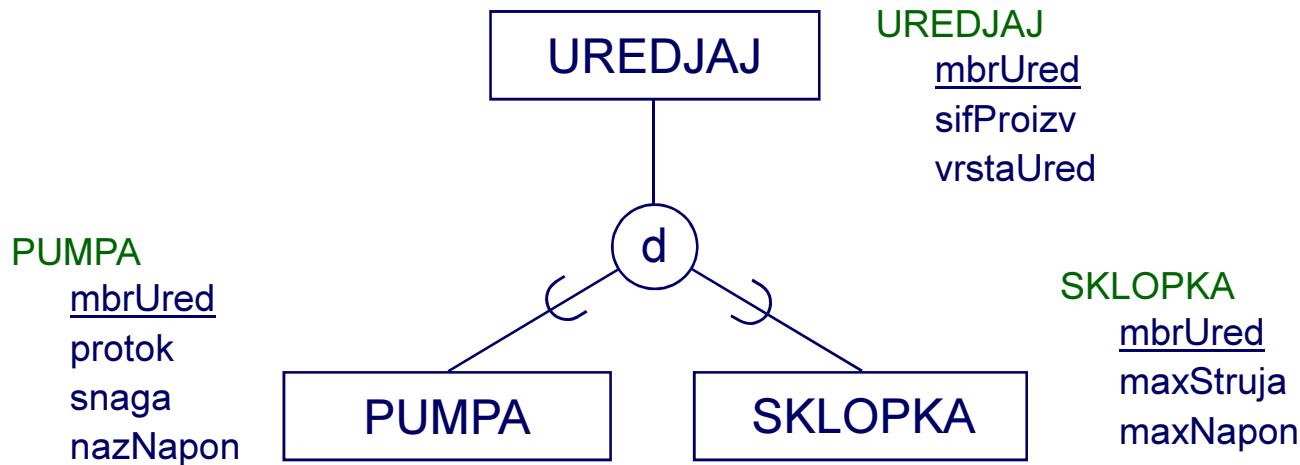
---



- u kružnicu se upisuje slovo **d** (*disjoint*) ukoliko se radi o ekskluzivnoj generalizaciji/specijalizaciji
  - uređaj može biti **ili** pumpa **ili** sklopka (ekskluzivni ili)
- pomoću vrijednosti atributa *vrstaUred* (npr. 'p' ili 's') može se odrediti podklasa (specijalizacija) kojoj entitet pripada

# Preslikavanje u relacijski model

- specijalizacije nemaju vlastite ključeve



UREDJAJ = mbrUred, sifProizv, vrstaUred

PUMPA = mbrUred, protok, snaga, nazNapon

SKLOPKA = mbrUred, maxStruja, maxNapon

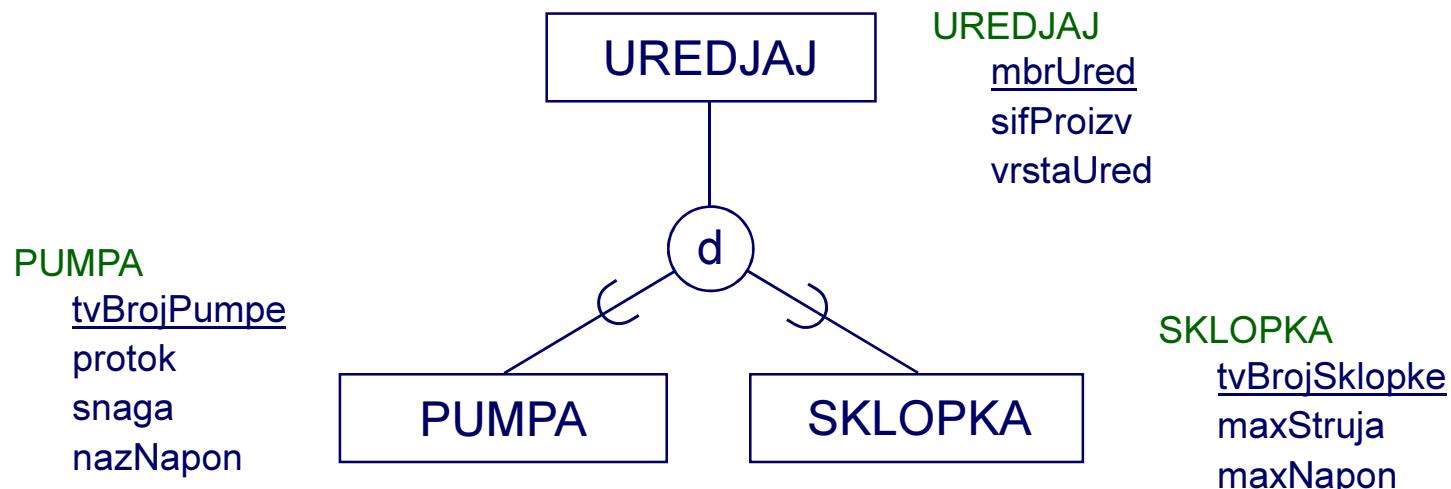
ILI

UREDJAJ = mbrUred, sifProizv, vrstaUred, protok, snaga, nazNapon, maxStruja, maxNapon

Za vježbu: napisati SQL naredbe za kreiranje relacija. Voditi računa o primarnim i stranim ključevima.

# Preslikavanje u relacijski model

- specijalizacije imaju vlastite ključeve



UREDJAJ = mbrUred, sifProizv, vrstaUred

PUMPA = tvBrojPumpe, protok, snaga, nazNapon, mbrUred

SKLOPKA = tvBrojSklopke, maxStruja, maxNapon, mbrUred

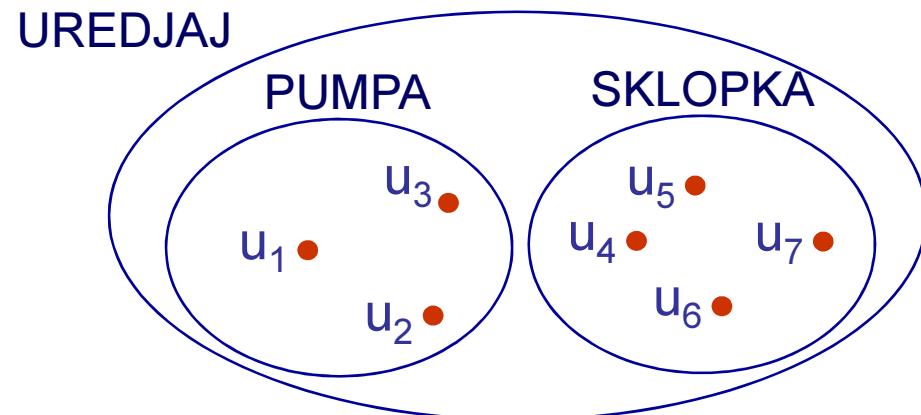
Alternativni  
ključevi

Za vježbu: napisati SQL naredbe za kreiranje relacija. Voditi računa o primarnim, alternativnim i stranim ključevima.

# Neekskluzivna generalizacija/specijalizacija

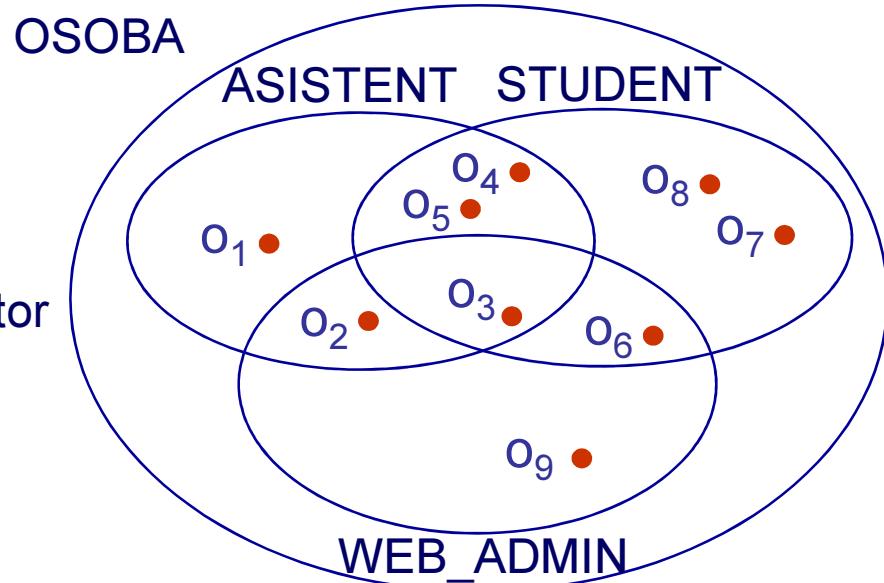
- ekskluzivna generalizacija/specijalizacija

Uredaj može biti ili pumpa ili sklopka

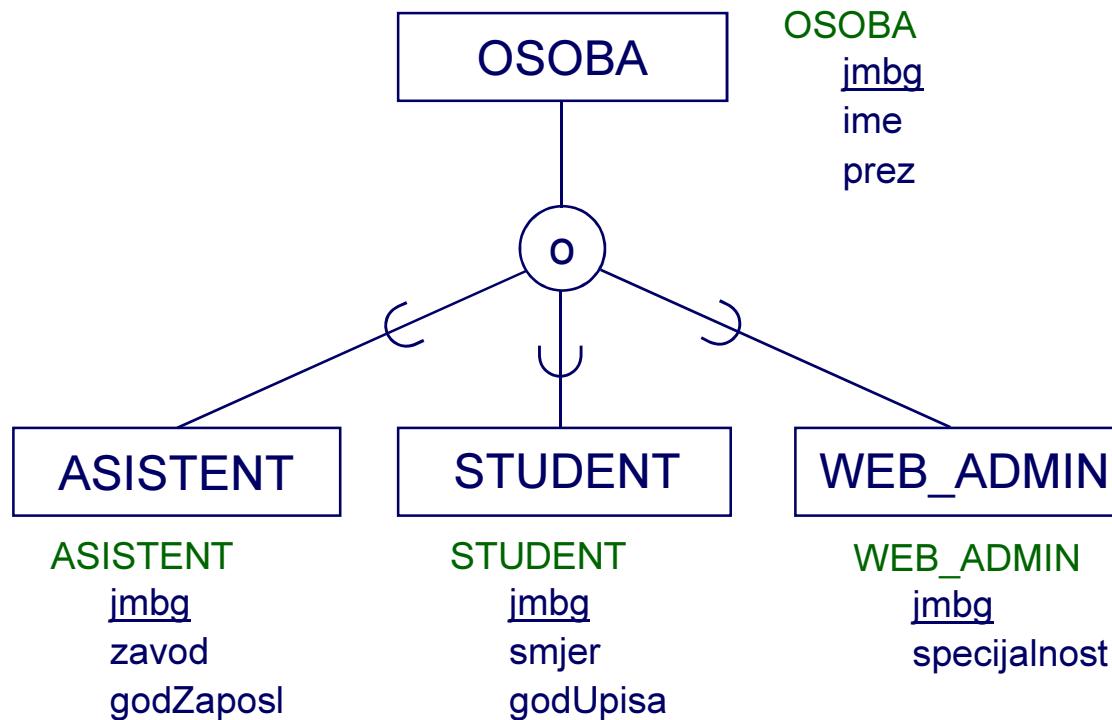


- neekskluzivna generalizacija/specijalizacija

Osoba može biti asistent i/ili student na poslijediplomskom studiju i/ili administrator web stranica fakulteta



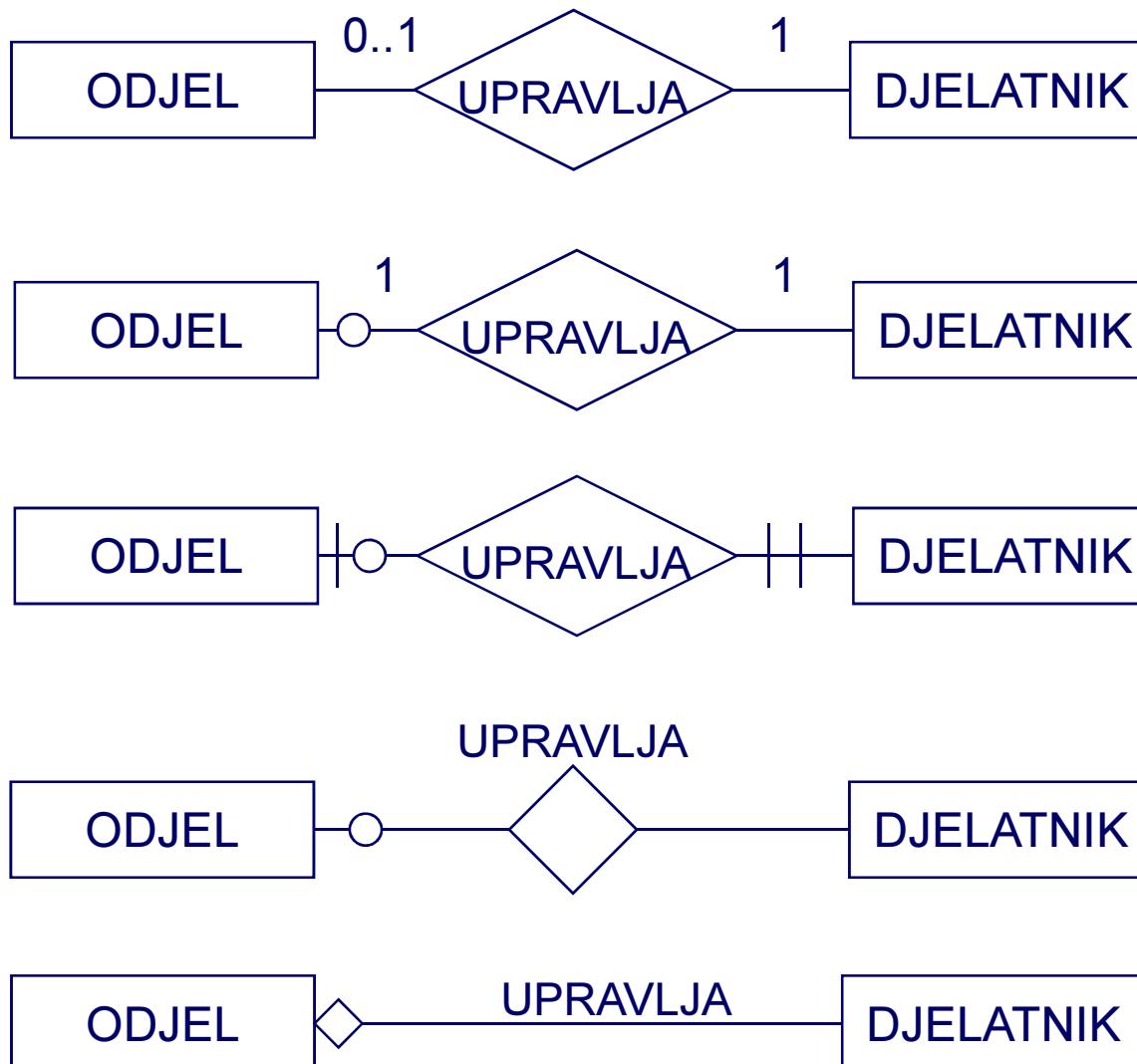
# Neekskluzivna generalizacija/specijalizacija



- ukoliko se radi o neekskluzivnoj generalizaciji/specijalizaciji u kružnicu se upisuje slovo **o** (*overlapping*)
  - osoba može biti asistent **i/ili** student **i/ili** administrator web stranica
- preslikavanje u relacijski model: jednako kao kod ekskluzivne generalizacije/specijalizacije (također su moguće specijalizacije s vlastitim i bez vlastitih ključeva)

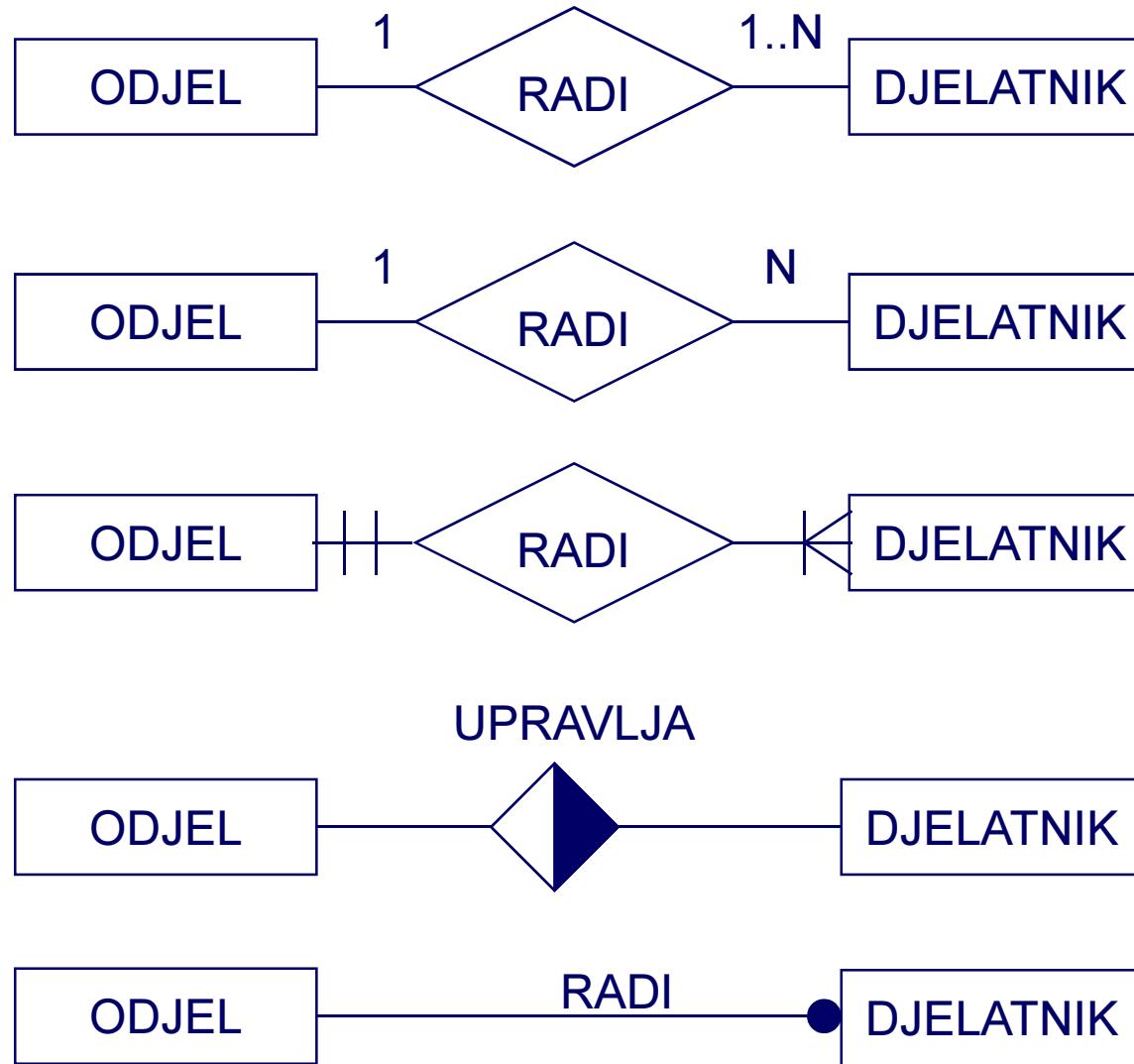
# ER model - različita notacija

---



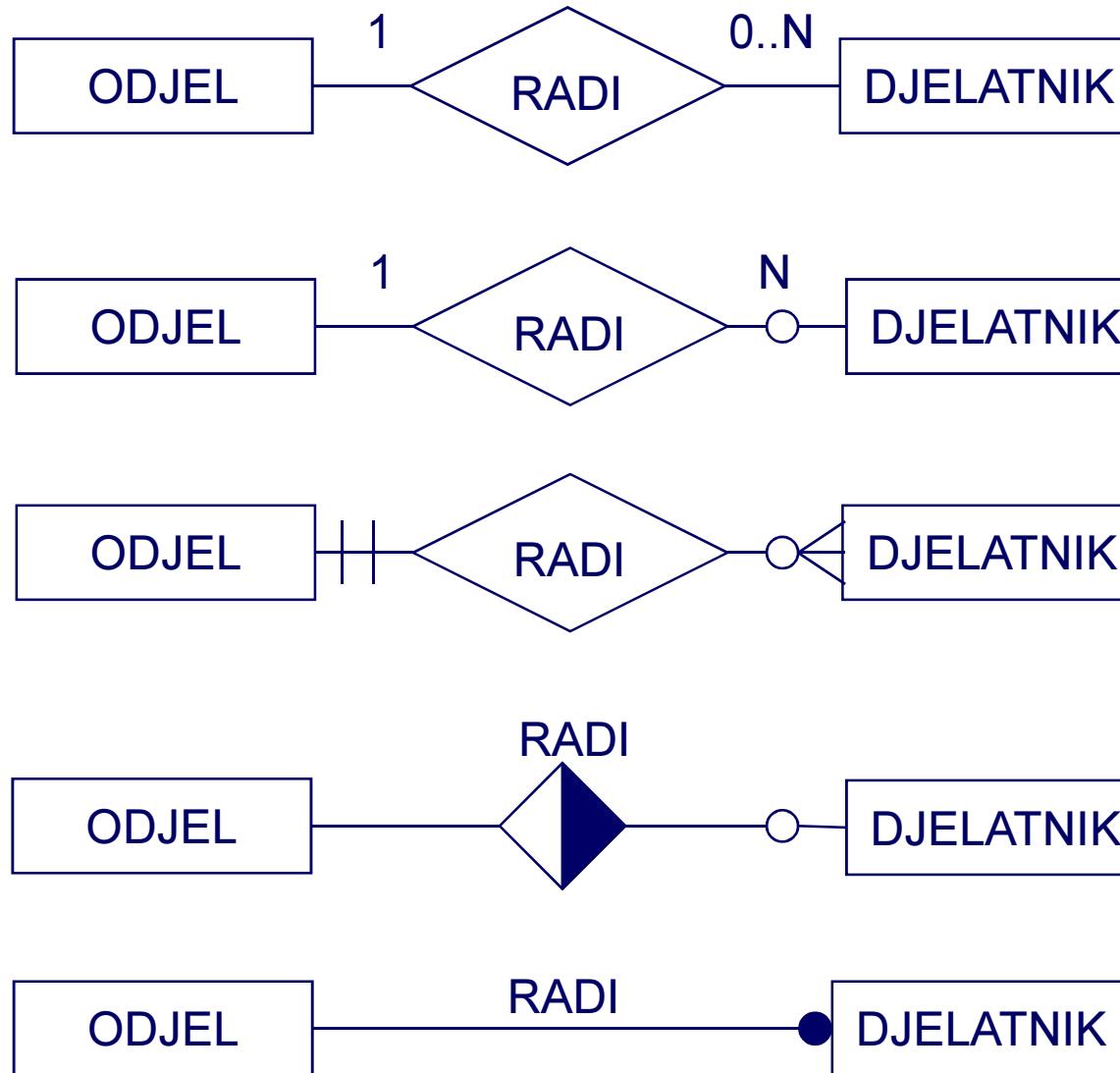
# ER model - različita notacija

---



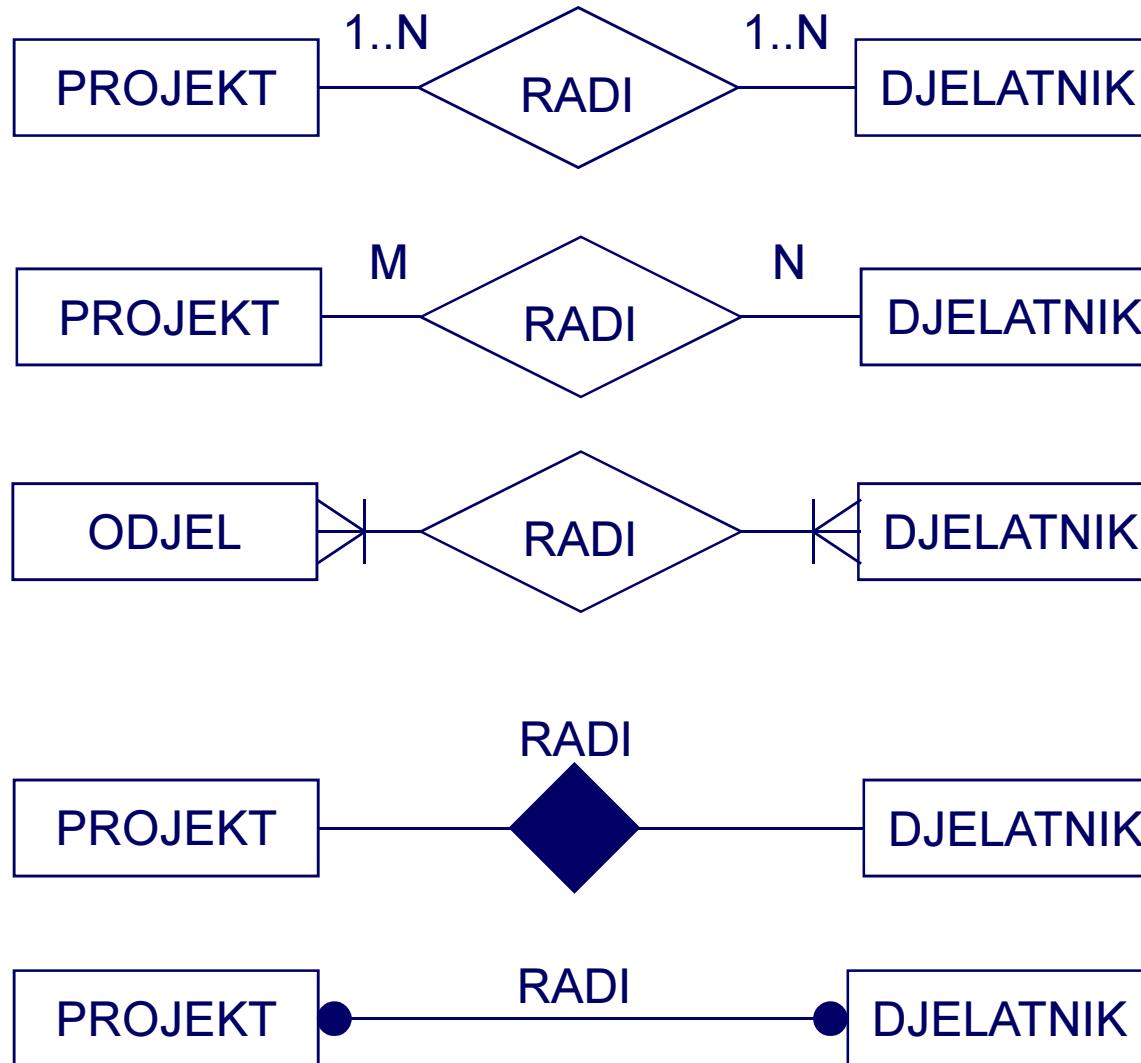
# ER model - različita notacija

---



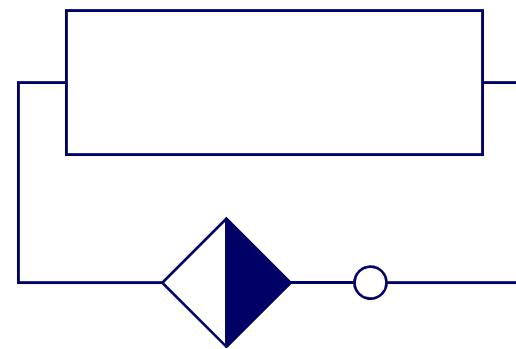
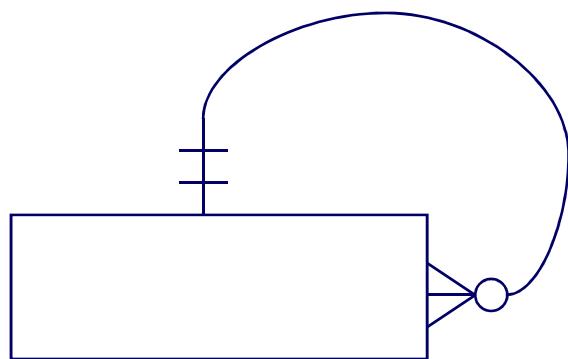
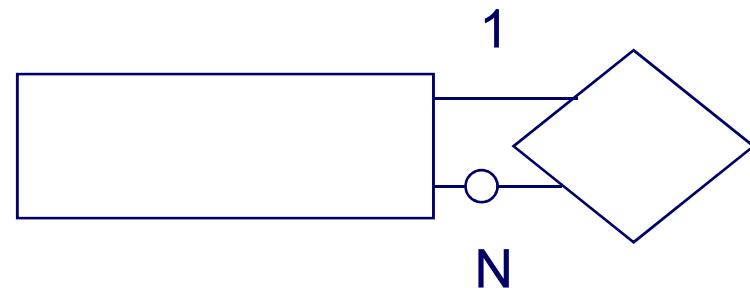
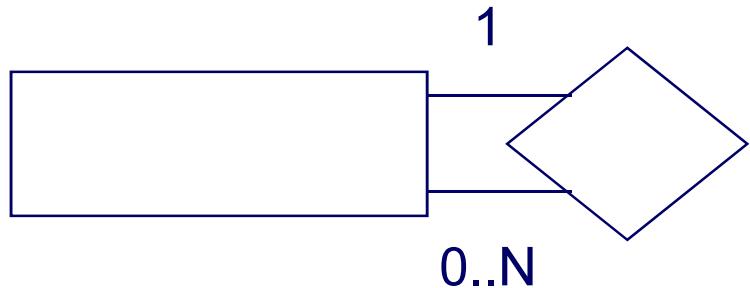
# ER model - različita notacija

---



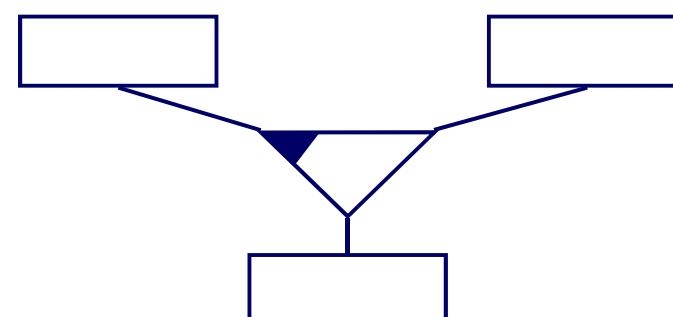
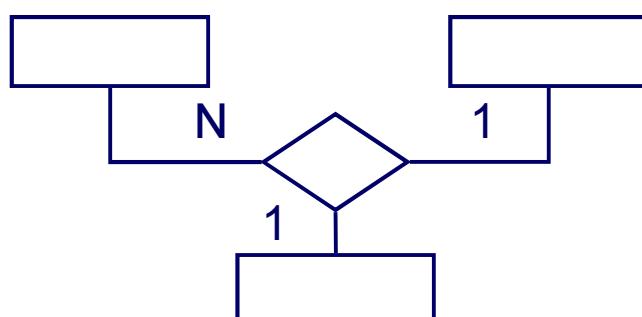
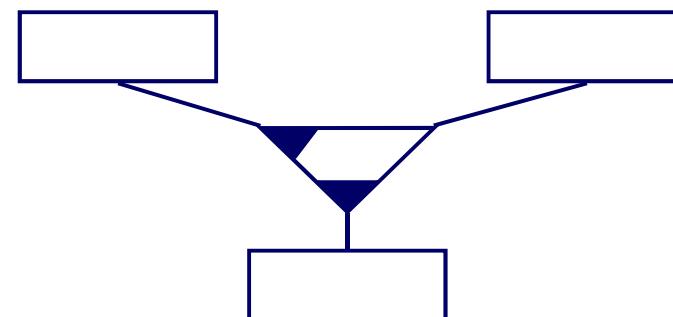
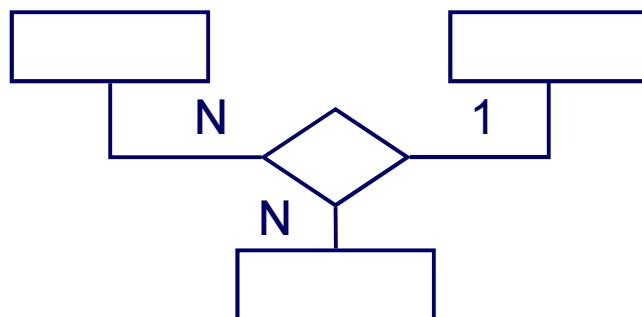
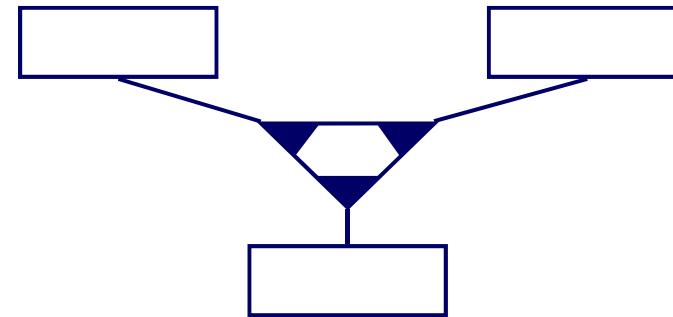
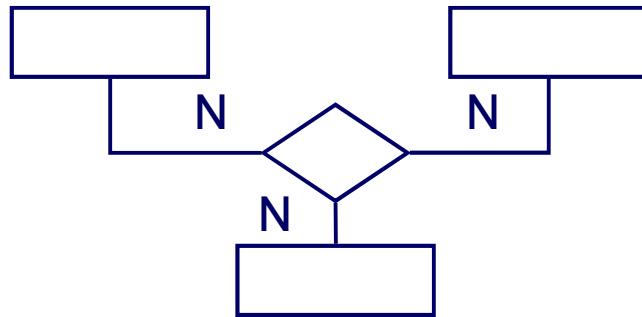
# ER model - različita notacija

---



# ER model - različita notacija

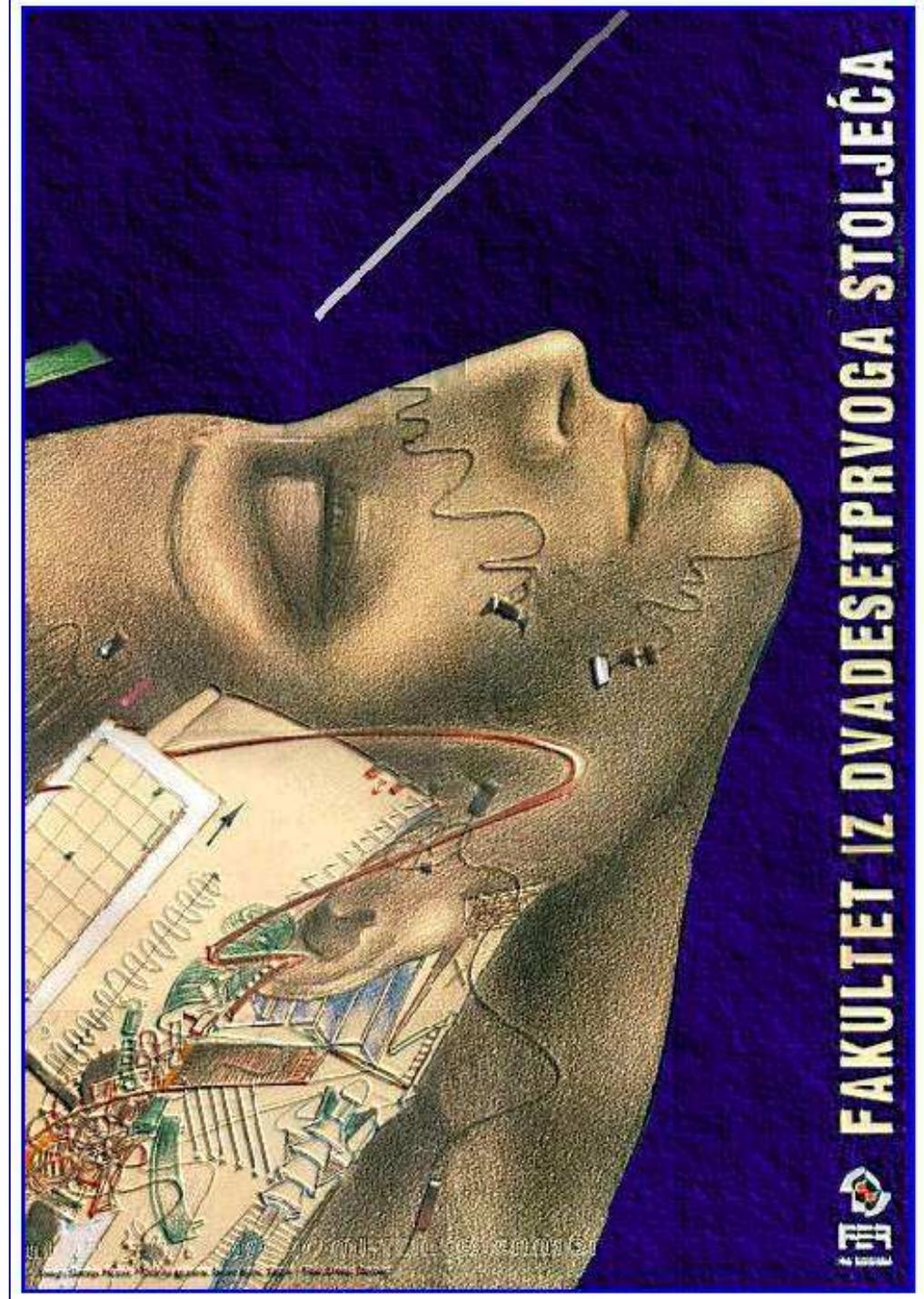
---



# Baze podataka

Predavanja  
svibanj 2014.

## 16. ER model baze podataka (3. dio - primjeri)



FAKULTET IZ DVADESETPRVOGA STOLJEĆA



## **1. Model baze podataka za razredbeni ispit**

Potrebno je evidentirati podatke o kandidatima: JMBG, prezime, ime, završenu srednju školu, mjesto rođenja i mjesto stanovanja. Pretpostavlja se da je kandidat završio samo jednu srednju školu. Za svaku srednju školu treba evidentirati šifru koja ju jedinstveno identificira, naziv, adresu i mjesto u kojem se škola nalazi. Za mjesto treba evidentirati poštanski broj, naziv mjesta i županiju u kojoj se mjesto nalazi. Županija ima svoju šifru i naziv.

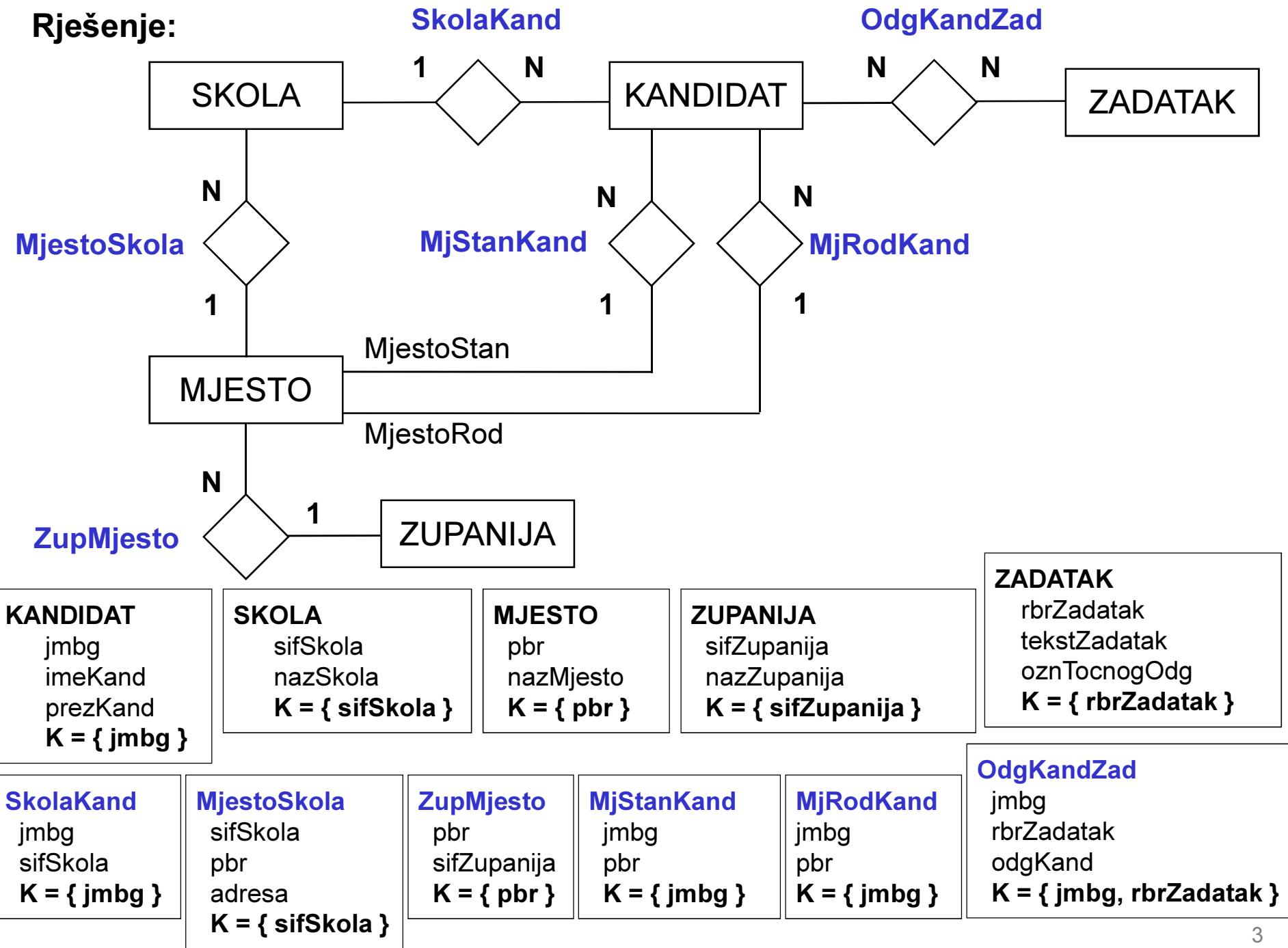
Treba evidentirati podatke o zadacima na testu: redni broj zadatka, tekst zadatka, oznaku točnog odgovora (može biti A, B, C, D ili E).

Za svakog kandidata evidentirati odgovore koje je dao na zadatke (mogući odgovori kandidata su A, B, C, D, E ili ništa).

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Opisati relacijski model u obliku SQL naredbi za kreiranje relacija s opisanim integritetskim ograničenjima. Odabratи prikladne tipove podataka.

Rješenje:



**KANDIDAT**

jmbg

imeKand

prezKand

**K = { jmbg }**

Ako bi neki entitet imao više mogućih ključeva, shema entiteta bi se mogla opisati npr. ovako:

**KANDIDAT**

jmbg

sifKand

imeKand

prezKand

**PK = K<sub>1</sub> = { jmbg }****K<sub>2</sub> = { sifKand }**

## Relacijski model u obliku SQL naredbi za kreiranje relacija:

```
CREATE TABLE zupanija (
 sifZupanija SMALLINT
 , nazZupanija CHAR(40)
 , PRIMARY KEY (sifZupanija)) ;

CREATE TABLE mjesto (
 pbr INTEGER
 , nazMjesto CHAR(20)
 , sifZupanija SMALLINT NOT NULL
 , PRIMARY KEY (pbr)
 , FOREIGN KEY (sifZupanija) REFERENCES zupanija(sifZupanija)) ;

CREATE TABLE skola (
 sifSkola INTEGER
 , nazSkola CHAR(40)
 , pbr INTEGER NOT NULL
 , adresa CHAR(40)
 , PRIMARY KEY (sifSkola)
 , FOREIGN KEY (pbr) REFERENCES mjesto(pbr)) ;

CREATE TABLE zadatak (
 rbrZadatak INTEGER
 , tekstZadatak CHAR(512)
 , oznTocnogOdg CHAR(1)
 , PRIMARY KEY (rbrZadatak)) ;
```

## Relacijski model u obliku SQL naredbi za kreiranje relacija (nastavak):

```
CREATE TABLE kandidat (
 jmbg CHAR(13)
, imeKand CHAR(20)
, prezKand CHAR(20)
, pbrRod INTEGER NOT NULL
, pbrStan INTEGER NOT NULL
, sifSkola INTEGER NOT NULL
, PRIMARY KEY (jmbg)
, FOREIGN KEY (pbrRod) REFERENCES mjesto (pbr)
, FOREIGN KEY (pbrStan) REFERENCES mjesto (pbr)
, FOREIGN KEY (sifSkola) REFERENCES skola (sifSkola)) ;
```

```
CREATE TABLE odgKandZad (
 jmbg CHAR(13)
, rbrZadatak INTEGER
, odgKand CHAR(1)
, PRIMARY KEY (jmbg, rbrZadatak)
, FOREIGN KEY (jmbg) REFERENCES kandidat (jmbg)
, FOREIGN KEY (rbrZadatak) REFERENCES zadatak (rbrZadatak)) ;
```

## 2. Model baze podataka za videoteku

Za film se evidentira šifra (identificira film), naslov filma, te osobe i njihove funkcije u filmu.

Funkcije koje osoba može imati u filmu predstavljene su kraticom i nazivom (npr. GL, glumac; RED, redatelj; SC, scenarist, itd.). Za svaku se osobu evidentira šifra osobe (identificira osobu), prezime i ime.

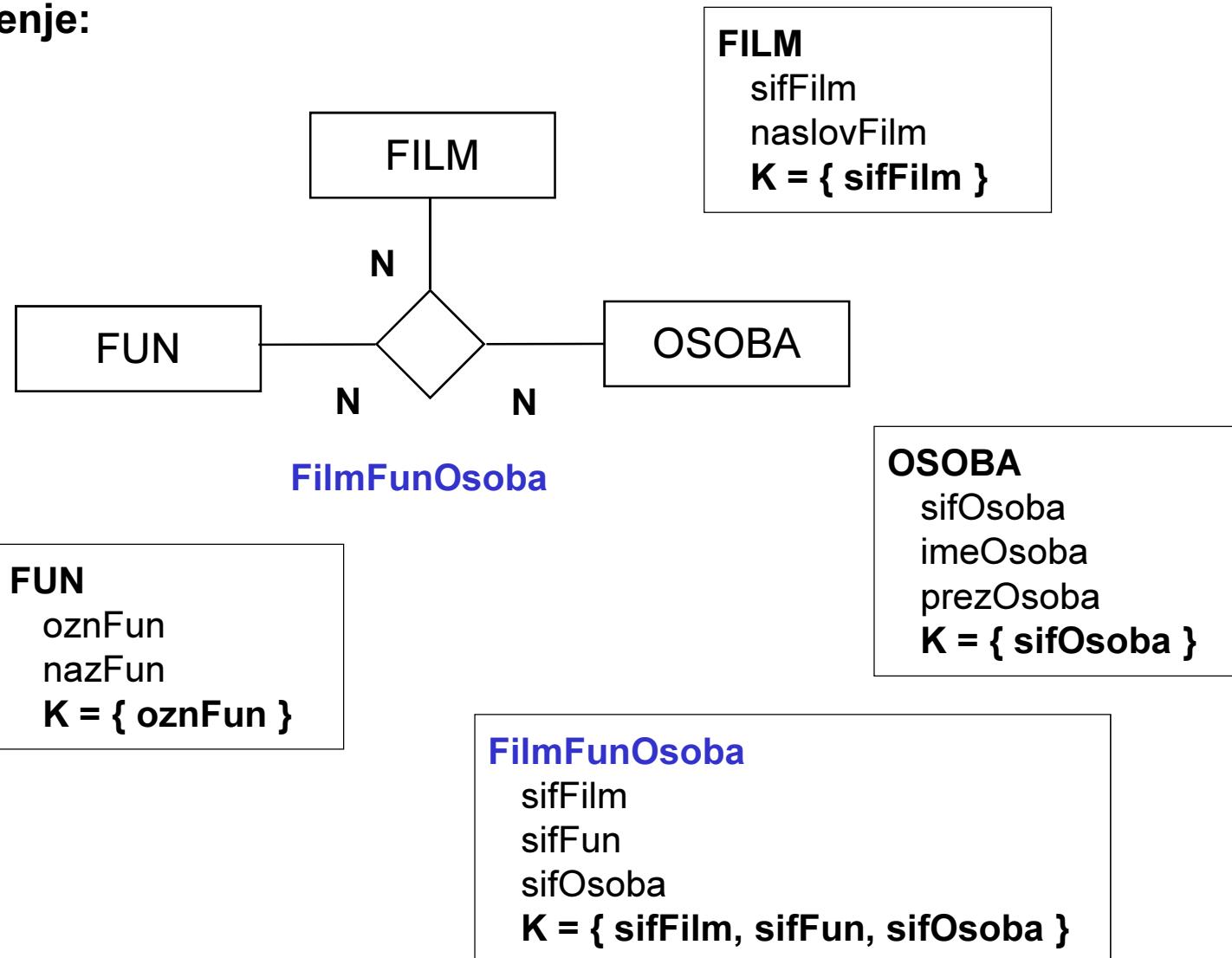
Treba uočiti da ista osoba može u istom filmu imati različite funkcije, npr:

### relacija VIDEOTEKA

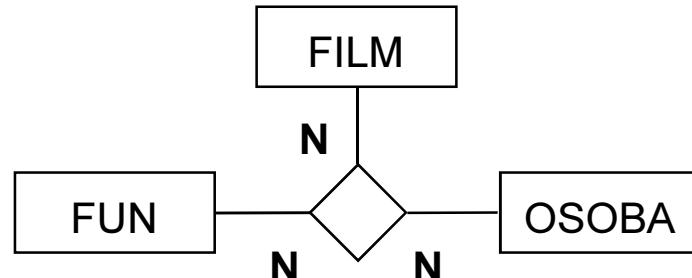
| sif<br>Film | naslovFilm             | ozn<br>Fun | nazFun   | sif<br>Osoba | ime    | prezime  |
|-------------|------------------------|------------|----------|--------------|--------|----------|
| 1           | Nepomirljivi           | RED        | redatelj | 10           | Clint  | Eastwood |
| 1           | Nepomirljivi           | GL         | glumac   | 10           | Clint  | Eastwood |
| 1           | Nepomirljivi           | GL         | glumac   | 20           | Morgan | Freeman  |
| 2           | Mostovi okruga Madison | RED        | redatelj | 10           | Clint  | Eastwood |
| 2           | Mostovi okruga Madison | GL         | glumac   | 40           | Meryl  | Streep   |
| 2           | Mostovi okruga Madison | GL         | glumac   | 10           | Clint  | Eastwood |
| 3           | Prljavi Harry          | RED        | redatelj | 30           | Don    | Siegel   |
| 3           | Prljavi Harry          | GL         | glumac   | 10           | Clint  | Eastwood |

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Rješenje:



## Diskusija:



FilmFunOsoba

Relacije koje nastaju transformacijom ER modela na slici:

ključevi relacija su podcrtani

|   | FILM           |                   |
|---|----------------|-------------------|
|   | <u>sifFilm</u> | <u>naslovFilm</u> |
| 1 |                | Nepomirljivi      |
| 2 |                | Mostovi okруга М. |
| 3 |                | Prljavi Harry     |

|    | OSOBA           |                     |
|----|-----------------|---------------------|
|    | <u>sifOsoba</u> | <u>ime, prezime</u> |
| 10 |                 | C. Eastwood         |
| 20 |                 | M. Freeman          |
| 30 |                 | D. Siegel           |
| 40 |                 | M. Streep           |

|     | FUN           |               |
|-----|---------------|---------------|
|     | <u>oznFun</u> | <u>nazFun</u> |
| RED |               | redatelj      |
| GL  |               | glumac        |

|   | FilmFunOsoba   |               |                 |
|---|----------------|---------------|-----------------|
|   | <u>sifFilm</u> | <u>oznFun</u> | <u>sifOsoba</u> |
| 1 | 1              | RED           | 10              |
| 2 | 1              | GL            | 10              |
| 3 | 1              | GL            | 20              |
|   | 2              | RED           | 10              |
|   | 2              | GL            | 40              |
|   | 2              | GL            | 10              |
|   | 3              | RED           | 30              |
|   | 3              | GL            | 10              |

SQL upit kojim se dohvaćaju osobe i njihove funkcije u filmu "Prljavi Harry"

```
SELECT osoba.*, fun.*
 FROM osoba, film, fun, filmFunOsoba
 WHERE osoba.sifOsoba = filmFunOsoba.sifOsoba
 AND film.sifFilm = filmFunOsoba.sifFilm
 AND fun.oznFun = filmFunOsoba.oznFun
 AND film.naslovFilm = 'Prljavi Harry'
```

| sifOsoba | ime   | prezime  | oznFun | nazFun   |
|----------|-------|----------|--------|----------|
| 10       | Clint | Eastwood | GL     | glumac   |
| 30       | Don   | Siegel   | RED    | redatelj |

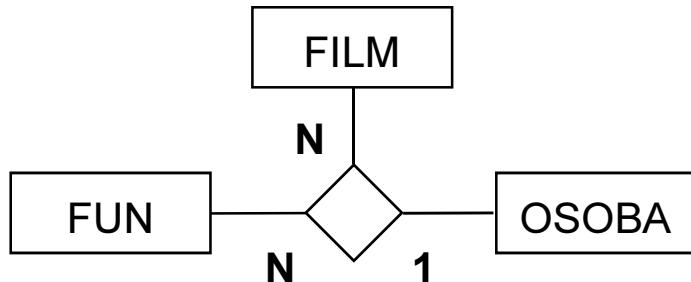
#### relacija VIDEOTEKA

| sif<br>Film | naslovFilm             | ozn<br>Fun | nazFun   | sif<br>Osoba | ime    | prezime  |
|-------------|------------------------|------------|----------|--------------|--------|----------|
| 1           | Nepomirljivi           | RED        | redatelj | 10           | Clint  | Eastwood |
| 1           | Nepomirljivi           | GL         | glumac   | 10           | Clint  | Eastwood |
| 1           | Nepomirljivi           | GL         | glumac   | 20           | Morgan | Freeman  |
| 2           | Mostovi okruga Madison | RED        | redatelj | 10           | Clint  | Eastwood |
| 2           | Mostovi okruga Madison | GL         | glumac   | 40           | Meryl  | Streep   |
| 2           | Mostovi okruga Madison | GL         | glumac   | 10           | Clint  | Eastwood |
| 3           | Prljavi Harry          | RED        | redatelj | 30           | Don    | Siegel   |
| 3           | Prljavi Harry          | GL         | glumac   | 10           | Clint  | Eastwood |

**(Teorey):** U vezi koja povezuje entitete  $E_1, \dots, E_k, \dots, E_m$ , spojnost =1 entiteta  $E_k$  znači da (...) odnosno, vrijedi funkcijačka zavisnost

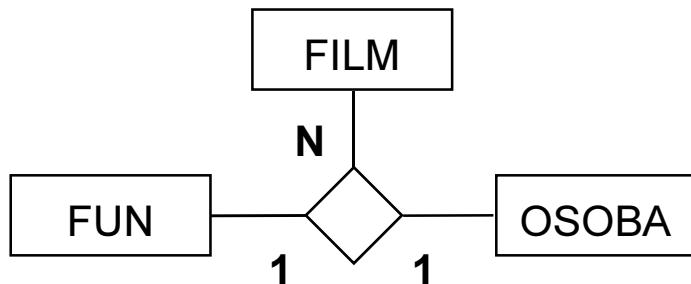
$$\bigcup_{j=1}^m K_j \setminus K_k \rightarrow K_k \quad \text{gdje su skupovi } K_j, (j = 1, \dots, m), \text{ ključevi entiteta } E_1, \dots, E_m$$

## ZAŠTO OVI MODELI NISU ISPRAVNI?



sifFilm, oznFun → sifOsoba

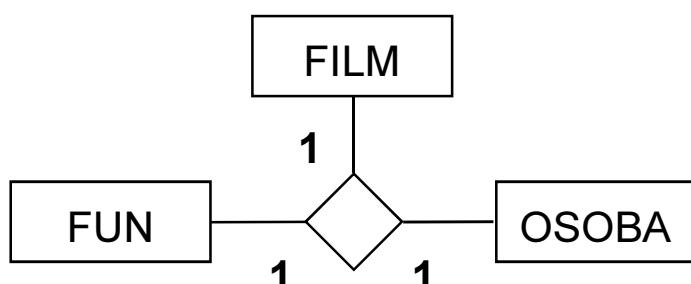
u filmu može glumiti samo jedan glumac, film može režirati samo jedan redatelj, scenarij za film može pisati samo jedan scenarist, ...



sifFilm, oznFun → sifOsoba

sifFilm, sifOsoba → oznFun

dodatno: u jednom filmu osoba može imati samo jednu funkciju



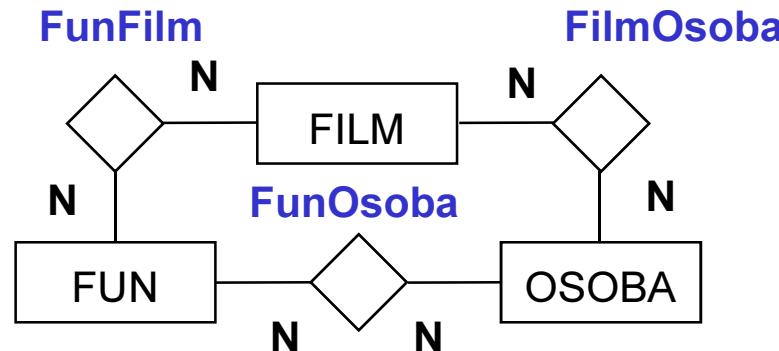
sifFilm, oznFun → sifOsoba

sifFilm, sifOsoba → oznFun

sifOsoba, oznFun → sifFilm

dodatno: ...

# ZAŠTO OVAJ MODEL NIJE ISPRAVAN?



Relacije koje  
nastaju  
transformacijom  
ER modela na  
slici:

FILM

sifFilm

*naslovFilm*

1

Nepomirljivi

2

Mostovi okruga M.

3

Prljavi Harry

ključevi  
relacija su  
podcrtani

OSOBA

sifOsoba

*ime, prezime*

10

C. Eastwood

20

M. Freeman

30

D. Siegel

40

M. Strep

FUN

oznFun

*nazFun*

RED

redatelj

GL

glumac

FunFilm

oznFun

sifFilm

RED

1

RED

2

RED

3

GL

1

GL

2

GL

3

GL

3

FilmOsoba

sifFilm

sifOsoba

1

10

1

20

2

10

2

40

3

30

3

10

FunOsoba

oznFun

sifOsoba

GL

10

GL

20

GL

40

RED

10

RED

30

SQL upit kojim se dohvaćaju osobe i njihove funkcije u filmu "Prljavi Harry"

```
SELECT osoba.*, fun.*
 FROM osoba, film, fun, filmOsoba, funOsoba, funFilm
 WHERE osoba.sifOsoba = filmOsoba.sifOsoba
 AND filmOsoba.sifFilm = film.sifFilm
 AND film.sifFilm = funFilm.sifFilm
 AND funFilm.oznFun = fun.oznFun
 AND fun.oznFun = funOsoba.oznFun
 AND funOsoba.sifOsoba = osoba.sifOsoba
 AND film.naslovFilm = 'Prljavi Harry'
```

| sifOsoba | ime   | prezime  | oznFun | nazFun   |
|----------|-------|----------|--------|----------|
| 10       | Clint | Eastwood | GL     | glumac   |
| 10       | Clint | Eastwood | RED    | redatelj |
| 30       | Don   | Siegel   | RED    | redatelj |

**Clint Eastwood nije redatelj filma "Prljavi Harry"!  
→ "gubitak informacije"!!!**

Dekompozicija relacije VIDEOTEKA nije obavljena bez gubitka informacije. Uvjet za dekompoziciju bez gubitka informacija opisan je u predavanjima.

### 3. Model baze podataka za poduzeće za održavanje plinskih instalacija

Uređaji koje poduzeće evidentira su brojila, ventili i reduktori. Za svaki pojedini uređaj treba evidentirati vrstu uređaja ('B', 'V' ili 'R'), proizvođača uređaja, tvornički broj i godinu proizvodnje uređaja. Za proizvođače uređaja evidentiraju se njihove šifre i nazivi. Ne postoje dva uređaja istog proizvođača koji imaju jednake tvorničke brojeve.

Dodatno, ovisno o vrsti uređaja, treba evidentirati njima svojstvene, posebne ili specijalističke podatke.

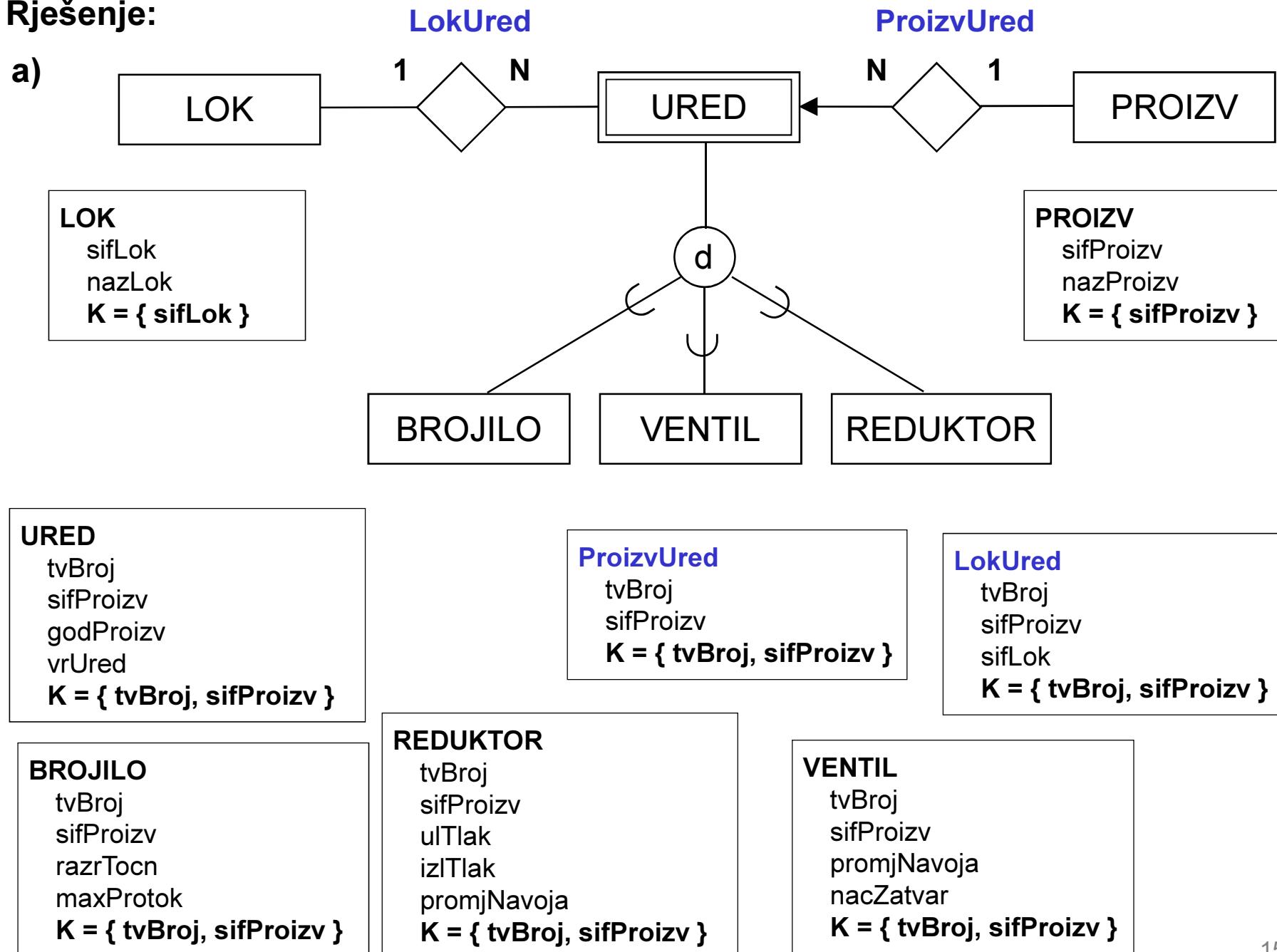
| za brojila:       | za ventile:      | za reduktore:      |
|-------------------|------------------|--------------------|
| razred točnosti   | promjer navoja   | ulazni tlak plina  |
| max. protok plina | način zatvaranja | izlazni tlak plina |
|                   |                  | promjer navoja     |

Potrebno je evidentirati popis lokacija (šifra i naziv) na kojima uređaji mogu biti instalirani. Evidentirati trenutnu lokaciju na kojoj je uređaj instaliran.

- Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF. Opisati relacijski model u obliku SQL naredbi za kreiranje relacija s ugrađenim pravilima integriteta.
- Što treba promijeniti u ER modelu iz a) kako bi se omogućilo evidentiranje povijesti premještanja uređaja. Kakve su posljedice na relacijski model?

## Rješenje:

a)



## Rješenje: Relacijski model

```
CREATE TABLE proizv (
 sifProizv INTEGER
, nazProizv CHAR(20)
, PRIMARY KEY (sifProizv));
```

```
CREATE TABLE lok (
 sifLok INTEGER
, nazLok CHAR(40)
, PRIMARY KEY (sifLok));
```

```
CREATE TABLE uređ (
 tvBroj CHAR(20)
, sifProizv INTEGER
, godProizv SMALLINT
, vrUred CHAR(1)
, sifLok INTEGER NOT NULL
, PRIMARY KEY (tvBroj, sifProizv)
, FOREIGN KEY (sifProizv) REFERENCES proizv (sifProizv)
, FOREIGN KEY (sifLok) REFERENCES lok (sifLok));
```

```
CREATE TABLE brojilo (
 tvBroj CHAR(20)
, sifProizv INTEGER
, razrTocn DECIMAL(3,1)
, maxProtok DECIMAL(5,4)
, PRIMARY KEY (tvBroj, sifProizv)
, FOREIGN KEY (tvBroj, sifProizv)
 REFERENCES uređ (tvBroj, sifProizv));
```

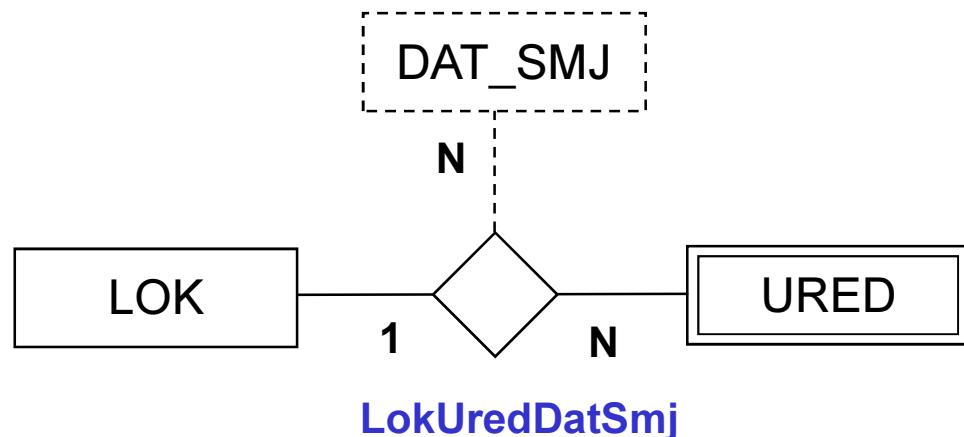
## Rješenje: Relacijski model (nastavak)

```
CREATE TABLE ventil (
 tvBroj CHAR(20)
, sifProizv INTEGER
, promjNavoja DECIMAL(3,1)
, nacZatvar DECIMAL(5,4)
, PRIMARY KEY (tvBroj, sifProizv)
, FOREIGN KEY (tvBroj, sifProizv)
 REFERENCES ured (tvBroj, sifProizv));
```

```
CREATE TABLE reduktor (
 tvBroj CHAR(20)
, sifProizv INTEGER
, ulTlak DECIMAL(6,2)
, izlTlak DECIMAL(6,2)
, promjNavoja DECIMAL(3,1)
, PRIMARY KEY (tvBroj, sifProizv)
, FOREIGN KEY (tvBroj, sifProizv)
 REFERENCES ured (tvBroj, sifProizv));
```

## Rješenje:

- b) Prepostavi li se da jedan uređaj ne može biti premješten više nego jedan put na dan, segment ER modela će se promijeniti na sljedeći način:



U odnosu na rješenje pod a), u novom relacijskom modelu potrebno je izbaciti atribut sifLok iz relacije ured, te dodati novu relaciju lokUredDatSmj

```
CREATE TABLE lokUredDatSmj (
 tvBroj CHAR(20)
, sifProizv INTEGER
, datSmjestaj DATE
, sifLok INTEGER NOT NULL
, PRIMARY KEY (tvBroj, sifProizv, datSmjestaj)
, FOREIGN KEY (tvBroj, sifProizv)
 REFERENCES ured (tvBroj, sifProizv)
, FOREIGN KEY (sifLok)
 REFERENCES lok(sifLok)) ;
```

#### **4. Model baze podataka automehaničarske radionice**

Evidentirati podatke o automobilima. Automobil je identificiran tvorničkim brojem (ne postoje dva automobila s istim tvorničkim brojem). Za automobil treba evidentirati godinu proizvodnje i model automobila. Modeli automobila identificirani su proizvođačem i nazivom modela (međusobno različiti proizvođači mogu svoje modele nazivati istim imenom - npr. Renault može imati svoj model naziva Europa, a Opel može imati sasvim drugi model koji se također naziva Europa). Za model automobila evidentira se godina u kojoj je model prvi puta proizведен. Proizvođač ima naziv, a identificiran je svojom šifrom.

U radionici je napravljen popis vrsta poslova koji se mogu obavljati na automobilima. Vrste poslova su šifrirane, a osim šifre i opisa vrste posla (npr. "Izmjena ulja", "Podešavanje ventila", itd.), za svaku vrstu posla se evidentira normativom zadano trajanje izraženo u minutama (koliko bi vremena mehaničar trebao utrošiti obavljajući posao te vrste). Vrste poslova su nezavisne od modela automobila - npr. "Izmjena ulja" je uvijek jednak posao neovisno od modela automobila na kojem se obavlja.

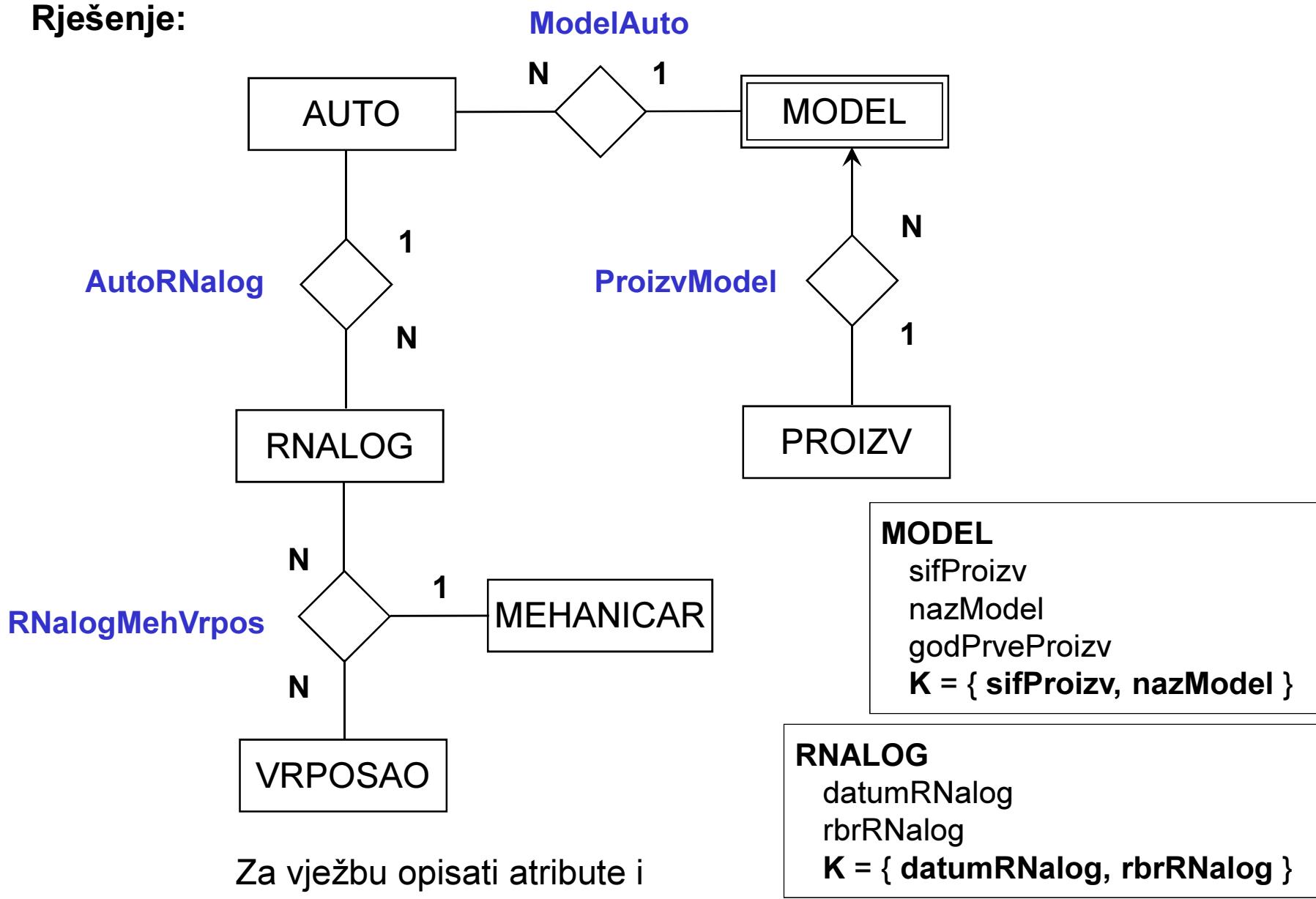
Za mehaničare zaposlene u radionici evidentira se jmbg, prezime i ime.

Za svaki dolazak automobila u radionicu otvara se jedan Radni nalog na kojem se evidentira automobil i datum dolaska automobila u radionicu. Isti automobil može biti primljen u radionicu više puta (čak i istog dana), ali se svaki put otvara novi Radni nalog. Radni nalog nema šifru. Radni nalog pri otvaranju dobiva svoj redni broj, pri čemu svakog dana redni brojevi naloga započinju ponovo s brojem jedan. Za isti datum ne postoje dva Radna naloga s istim brojem.

Uz Radni nalog se evidentira koji mehaničari će obaviti koje vrste poslova na automobilu. Poslove koji su zadani na Radnom nalogu može obaviti jedan ili nekoliko mehaničara, ali jedan zadani posao će jedan mehaničar obaviti sam od početka do kraja. Mehaničari na raznim Radnim nalozima mogu obavljati različite vrste poslova. Mehaničar odmah po obavljenom poslu na nekom automobilu evidentira koliko je vremena u minutama zaista utrošio na obavljanje tog posla (to se vrijeme može razlikovati od normativom zadanoj vremenu).

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Rješenje:



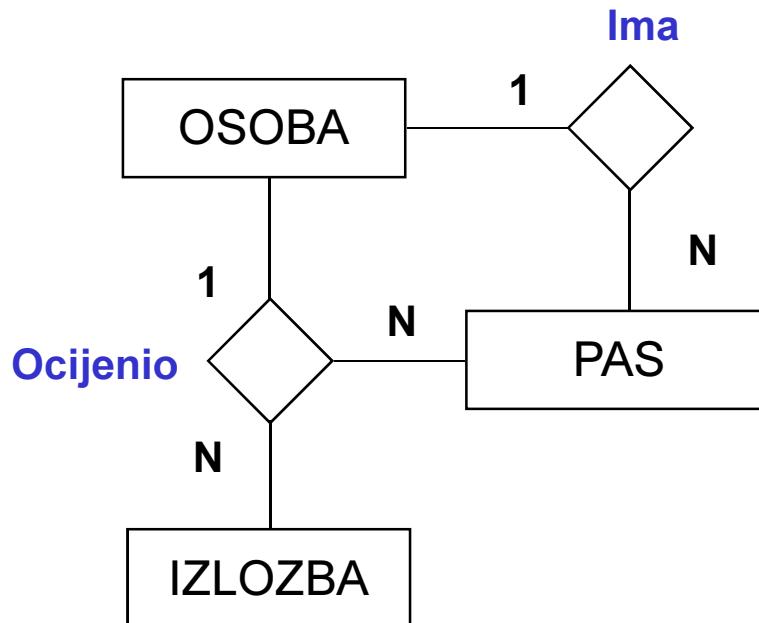
## **5. Model baze podataka za izložbe pasa**

Za svaku se osobu evidentira jmbg, prezime i ime. Za psa se evidentira broj markice koja identificira psa, ime psa, datum okota i osoba koja je vlasnik tog psa. Pretpostavlja se da jedna osoba može imati više pasa, a pas pripada samo jednoj osobi.

Neki vlasnici vode svoje pse na izložbe pasa. Za izložbu se evidentira šifra izložbe koja ju jedinstveno identificira i datum izložbe. Za jednog psa na jednoj izložbi treba evidentirati samo jednu ocjenu i osobu koja ga je ocjenjivala. Ista osoba na jednoj izložbi može ocijeniti više pasa. Ista osoba može ocjenjivati istog psa na više različitih izložbi. Za osobe koje ocjenjuju pse također se evidentiraju jmbg, prezime i ime. Te osobe mogu istovremeno biti i vlasnici pasa.

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Rješenje:



**OSOBA**  
jmbg  
imeOso  
prezOso  
**K = { jmbg }**

**PAS**  
brMarkice  
imePas  
datOkota  
**K = { brMarkice }**

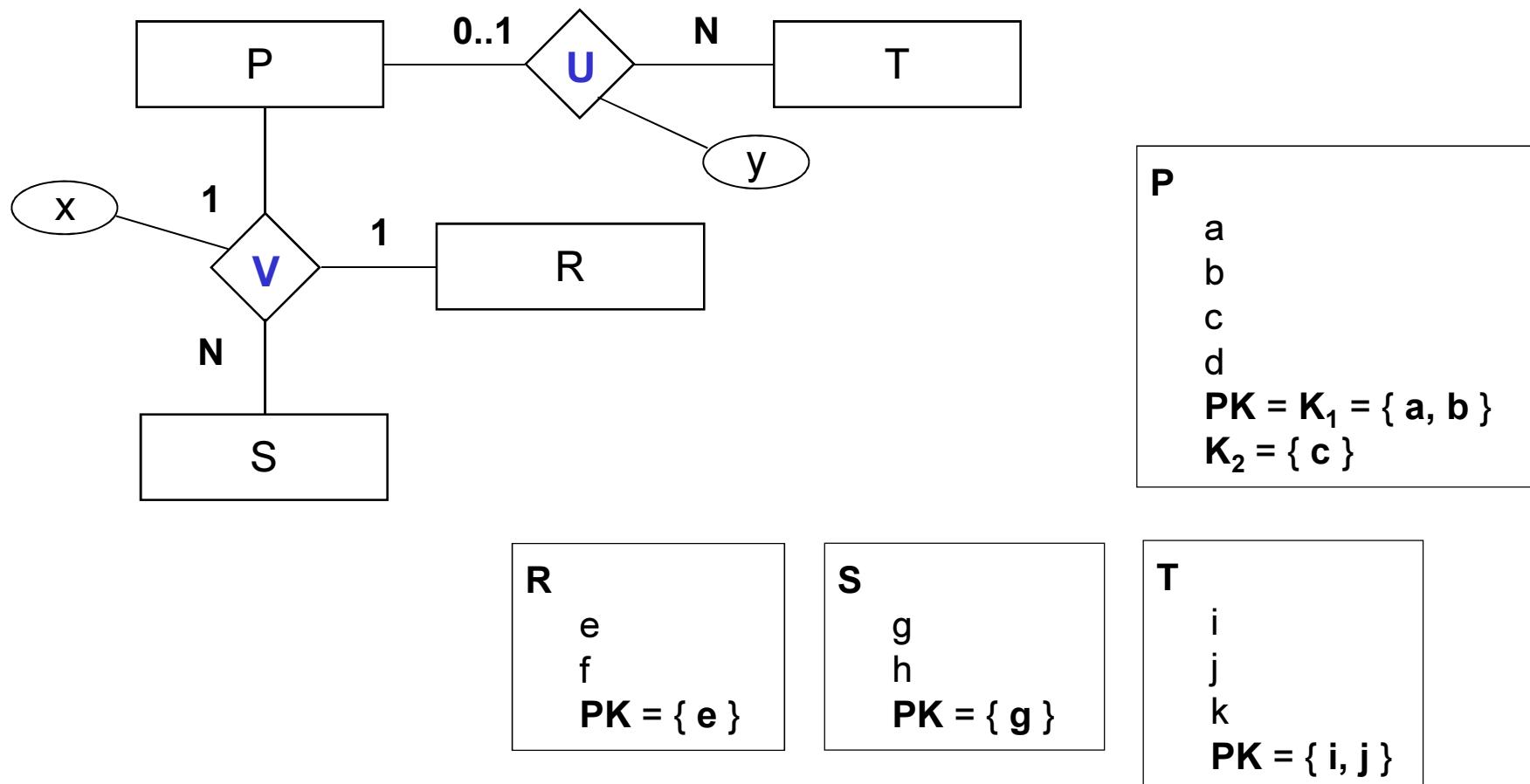
**IZLOZBA**  
siflzlozba  
datlzlozba  
**K = { siflzlozba }**

**Ima**  
brMarkice  
jmbg  
**K = { brMarkice }**

**Ocijenio**  
brMarkice  
siflzlozba  
jmbg  
ocjena  
**K = { siflzlozba, brMarkice }**

6. Zadan je ER model i pripadne sheme entiteta. Na slici su prikazani samo vlastiti atributi veza.

Definirati sheme veza. Napisati SQL naredbe za kreiranje relacija relacijskog modela. Tipove podataka ne treba navoditi. Naredbe moraju sadržavati definicije integritetskih ograničenja.



Rješenje:      **Sheme veza**

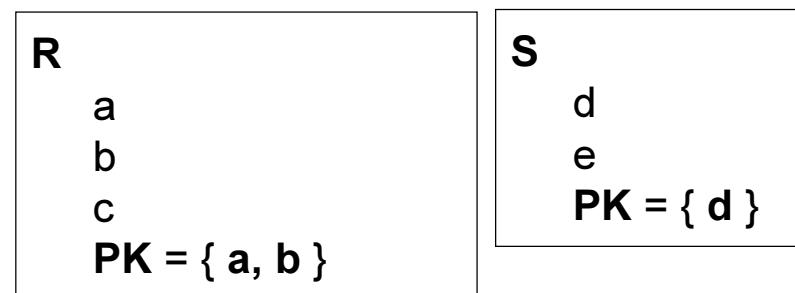
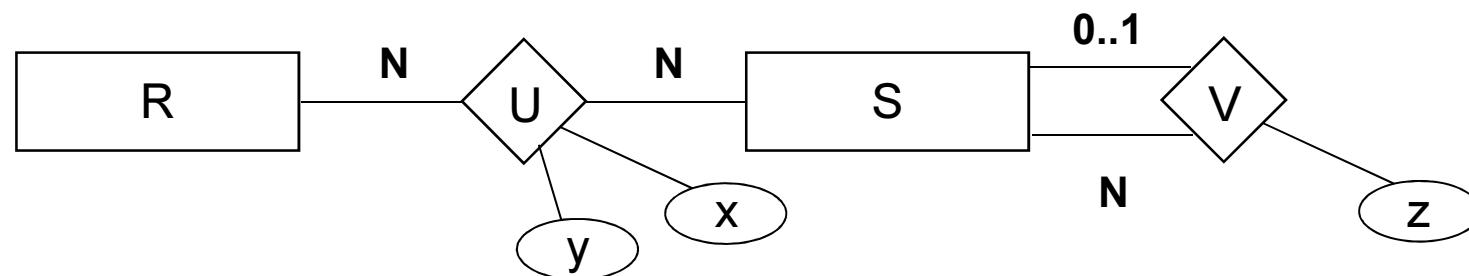
| V                                 | U |
|-----------------------------------|---|
| a                                 | i |
| b                                 | j |
| e                                 | a |
| g                                 | b |
| x                                 | y |
| PK = K <sub>1</sub> = { a, b, g } |   |
| K <sub>2</sub> = { e, g }         |   |

## Relacijski model

```
CREATE TABLE p (
 a ...
 , b ...
 , c ...
 , d ...
 , PRIMARY KEY (a, b)
 , UNIQUE (c));
CREATE TABLE r (
 e ...
 , f ...
 , PRIMARY KEY (e));
CREATE TABLE s (
 g ...
 , h ...
 , PRIMARY KEY (g));
CREATE TABLE t (
 i ...
 , j ...
 , k ...
 , y ...
 , a ...
 , b ...
 , PRIMARY KEY (i, j)
 , FOREIGN KEY (a, b)
 REFERENCES p (a, b));
CREATE TABLE v (
 a ...
 , b ...
 , e ...
 , g ...
 , x ...
 , PRIMARY KEY (a, b, g)
 , UNIQUE (e, g)
 , FOREIGN KEY (a, b)
 REFERENCES p (a, b)
 , FOREIGN KEY (e)
 REFERENCES r (e)
 , FOREIGN KEY (g)
 REFERENCES s (g));
```

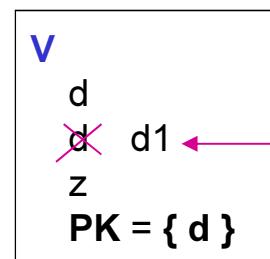
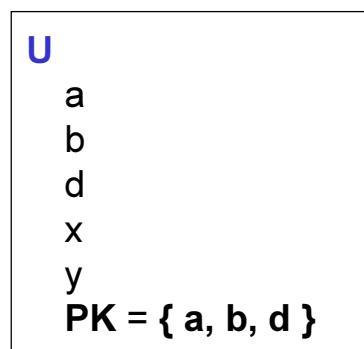
7. Zadan je ER model i pripadne sheme entiteta. Na slici su prikazani samo vlastiti atributi veza.

Definirati sheme veza. Napisati SQL naredbe za kreiranje relacija relacijskog modela. Tipove podataka ne treba navoditi. Naredbe moraju sadržavati definicije integritetskih ograničenja.



Rješenje:

Sheme veza



Preimenovati jedan  
od atributa

Relacijski model

```
CREATE TABLE r (
 a ...
 , b ...
 , c ...
 , PRIMARY KEY (a, b));
```

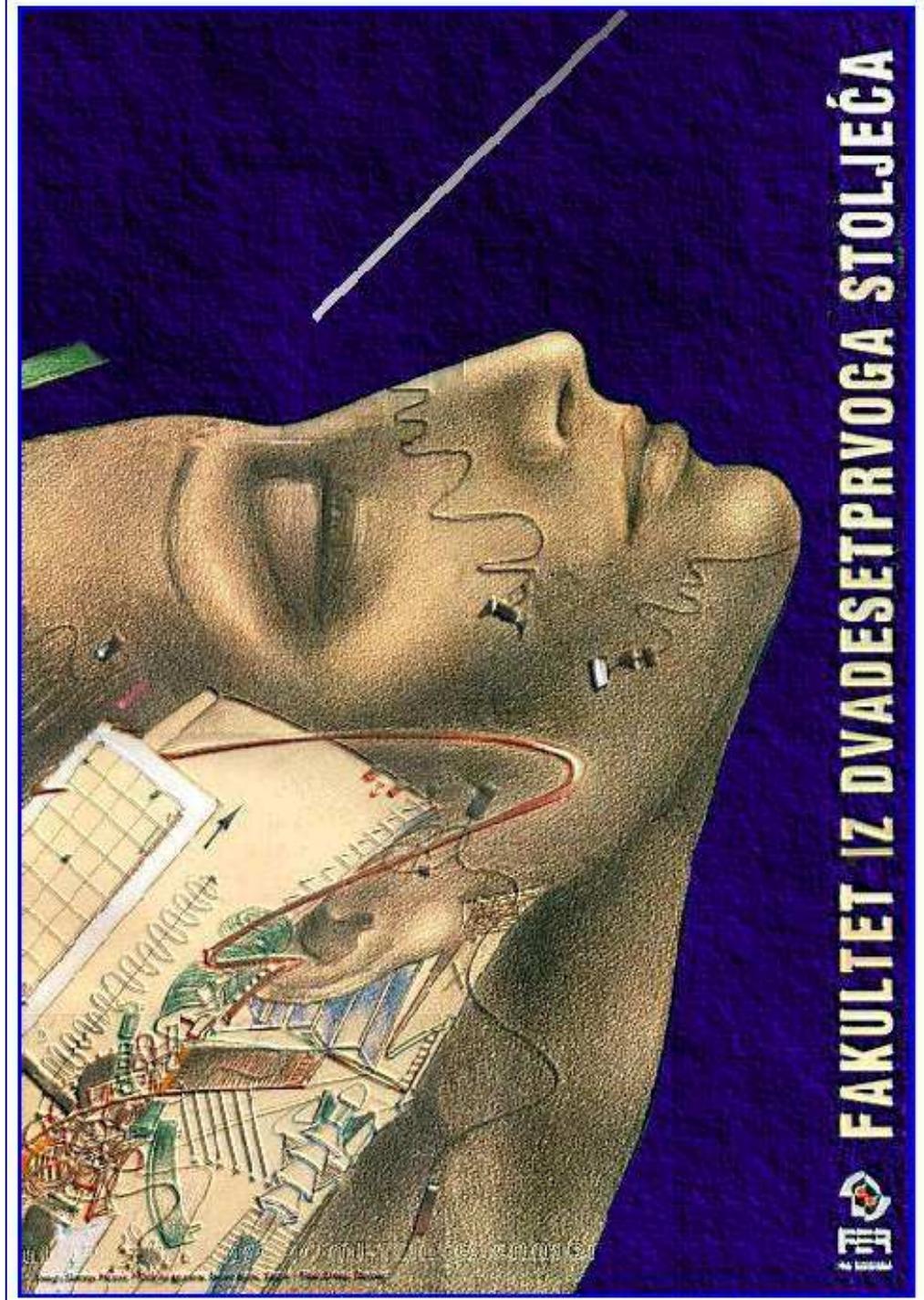
```
CREATE TABLE u (
 a ...
 , b ...
 , d ...
 , x ...
 , y ...
 , PRIMARY KEY (a, b, d)
 , FOREIGN KEY (a, b) REFERENCES r (a, b)
 , FOREIGN KEY (d) REFERENCES s (d));
```

```
CREATE TABLE s (
 d ...
 , d1 ...
 , e ...
 , z ...
 , PRIMARY KEY (d)
 , FOREIGN KEY (d1) REFERENCES s (d));
```

# Baze podataka

Predavanja  
svibanj 2014.

## 17. Transakcije i obnova baze podataka u slučaju razrušenja



FAKULTET IZ DVADESETPRVOGA STOLJEĆA

# Sustav za upravljanje bazama podataka

---

- *Database Management System*
  - skriva od korisnika detalje fizičke pohrane podataka
  - omogućuje definiciju i rukovanje s podacima
  - obavlja optimiranje upita
- obavlja funkciju zaštite podataka
  - integritet podataka
  - pristup podacima - autorizacija, sigurnost
  - **osigurava potporu za upravljanje transakcijama**
    - obnova u slučaju pogreške ili uništenja baze podataka
    - kontrola paralelnog pristupa

# 1. TRANSAKCIJA

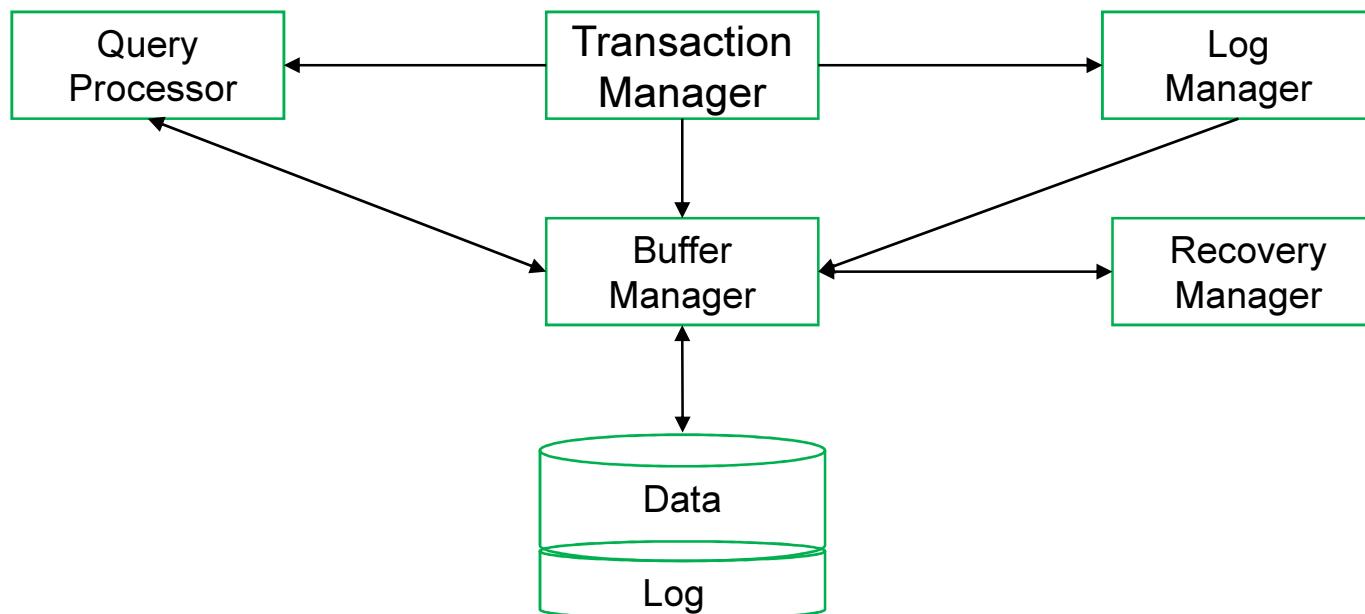
---

- jedinica rada nad bazom podataka
- sastoji se od niza logički povezanih izmjena
- početak transakcije - **BEGIN WORK**
- završetak transakcije:
  - **COMMIT WORK** - uspješan završetak - potvrđivanje transakcije
  - **ROLLBACK WORK** - neuspješan završetak - poništavanje transakcije - poništavanje svih izmjena koje je transakcija obavila

# Upravljanje transakcijama

---

- *Transaction Management*
- Upravljač transakcijama (*transaction manager*, *transaction processing monitor* – *TP monitor*) - dio sustava koji brine o obavljanju transakcija i osigurava zadovoljavanje svih poznatih pravila integriteta.



# Terminologija

---

- Transaction Management
- Transaction Manager
- Query Processor
- Log
- Log Manager
- Recovery Manager
- Buffer Manager
- Upravljanje transakcijama
- Upravljač transakcijama
- Procesor upita
- Dnevnik
- Upravljač dnevnicima
- Upravljač obnovom
- Upravljač međuspremnicima

# Primjer transakcije

---

```
CREATE PROCEDURE prijenos (s_racuna INTEGER
 , na_racun INTEGER
 , iznos DECIMAL (8,2))
BEGIN WORK;
 UPDATE racun SET saldo = saldo - iznos
 WHERE br_racun = s_racuna;
 UPDATE racun SET saldo = saldo + iznos
 WHERE br_racun = na_racun;
 SELECT saldo INTO pom_saldo FROM racun
 WHERE br_racun = s_racuna;
 IF pom_saldo < 0 THEN
 ROLLBACK WORK;
 ELSE
 COMMIT WORK;
 END IF
END PROCEDURE
```

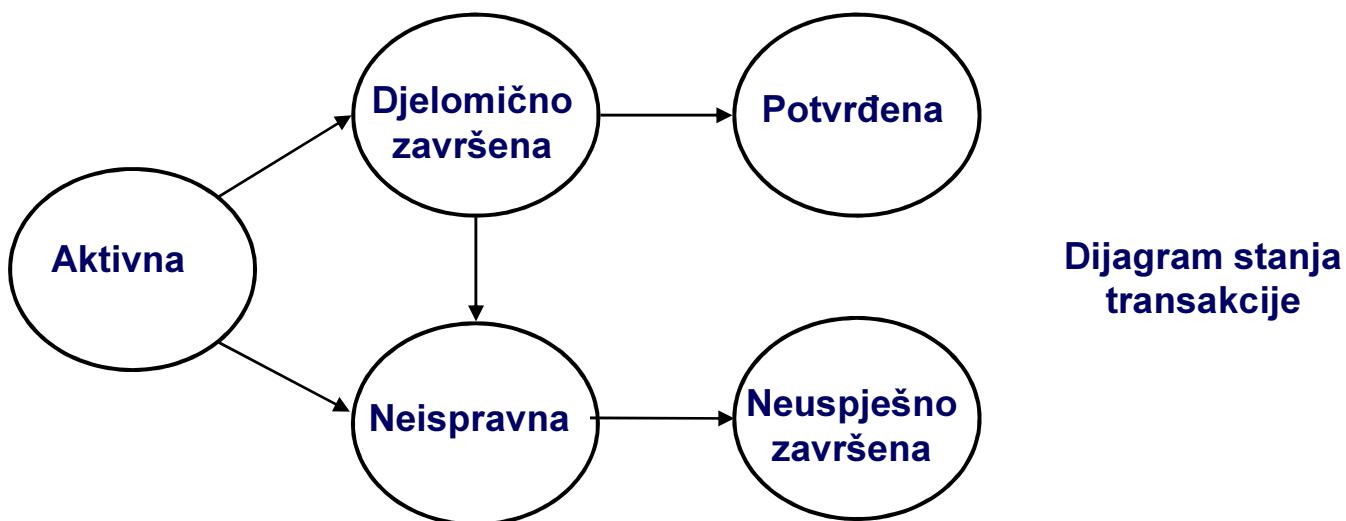
# Implicitne granice transakcija

---

- Ako granice transakcije nisu eksplicitno definirane naredbama BEGIN/COMMIT/ROLLBACK, tada se granice transakcije određuju implicitno:
  - svaka SQL naredba se smatra transakcijom za sebe
    - naročito važno: UPDATE, DELETE, INSERT u slučajevima kada djeluju nad skupom n-torki
- Neki SUBP-ovi (npr. *Oracle*, *SQL Server*, ...) podržavaju način rada u kojem nije potrebno eksplicitno zadati početak transakcije
  - tada se početkom transakcije smatra prva naredba izvedena u okviru sjednice
  - potrebno je eksplicitno zadati COMMIT ili ROLLBACK - nakon čega opet implicitno počinje nova transakcija

# Stanja transakcije

- Aktivna (*active*) – tijekom izvođenja
- Djelomično završena – (*partially committed*) – nakon što je obavljena njezina posljednja operacija
- Neispravna (*failed*) – nakon što se ustanovi da nije moguće nastaviti njezino normalno izvođenje
- Neuspješno završena (*aborted*) – nakon što su poništeni njezini efekti i baza podataka vraćena u stanje kakvo je bilo prije nego što je započela
- Potvrđena (*committed*) – uspješno završena



# Potvrđivanje transakcije

---

- Točka potvrđivanja (*commit point*) – trenutak u kojem sve izmjene koje je transakcija napravila postaju trajne
- Sve izmjene koje je transakcija načinila prije točke potvrđivanja mogu se smatrati tentativnima (*tentative* = privremeno, provizorno)
- U točki potvrđivanja otpuštaju se svi ključevi
- Potvrđena izmjena nikad ne može biti poništena - sustav garantira da će njezine izmjene biti trajno pohranjene u bazi podataka, čak i ako kvar nastane neposredno nakon njezinog potvrđivanja

# Svojstva transakcije

---

## ▪ ACID

- *Atomicity* - nedjeljivost transakcije (atomarnost) - transakcija se mora obaviti u cijelosti ili se uopće ne smije obaviti
- *Consistency* - konzistentnost - transakcijom baza podataka prelazi iz jednog konzistentnog stanja u drugo konzistentno stanje
- *Isolation* - izolacija - kada se paralelno obavljaju dvije ili više transakcija, njihov učinak mora biti jednak kao da su se obavljale jedna iza druge
- *Durability* - izdržljivost - ako je transakcija obavila svoj posao, njezini efekti ne smiju biti izgubljeni ako se dogodi kvar sustava, čak i u situaciji kada se kvar desi neposredno nakon završetka transakcije

# Nedjeljivost transakcije

```
CREATE PROCEDURE prijenos (s_racuna INTEGER, na_racun INTEGER
 , iznos DECIMAL (8,2))
 DEFINE pom_saldo DECIMAL (8,2);
 BEGIN WORK;
 UPDATE racun SET saldo = saldo - iznos
 WHERE br_racun = s_racuna;
 UPDATE racun SET saldo = saldo + iznos
 WHERE br_racun = na_racun;
 SELECT saldo INTO pom_saldo FROM racun
 WHERE br_racun = s_racuna;
 ...
```



Kvar sustava

Kvar se dogodio za vrijeme obavljanja druge UPDATE naredbe

- sustav mora osigurati poništavanje efekata prve UPDATE naredbe!

- Sa stanovišta krajnjeg korisnika transakcija je nedjeljiva
  - nije bitno što se moraju obaviti dvije ili više zasebnih operacija nad bazom podataka
- Korisnik mora biti siguran da je zadatak **obavljen potpuno i samo jednom** (ili ništa nije obavljeno)

# Izdržljivost transakcije

---

```
...
BEGIN WORK;
 UPDATE racun SET saldo = saldo - iznos
 WHERE br_racun = s_racuna;
 UPDATE racun SET saldo = saldo + iznos
 WHERE br_racun = na_racun;
 SELECT saldo INTO pom_saldo FROM racun
 WHERE br_racun = s_racuna;
 IF pom_saldo < 0 THEN
 ROLLBACK WORK;
 ELSE
 COMMIT WORK;
 END IF
```



Kvar sustava

- Kvar se dogodio nakon potvrđivanja transakcije
  - efekti transakcije ne smiju biti izgubljeni
- Bez obzira u kojem se trenutku nakon potvrđivanja transakcije dogodio kvar, sustav mora osigurati da su njezini efekti trajno pohranjeni

## 2. OBNOVA BAZE PODATAKA (*Database Recovery*)

---

- dovesti bazu podataka u najnovije stanje za koje se pouzdano zna da je bilo ispravno
- Velike baze podataka – dijeljene, višekorisničke – nužno moraju posjedovati mehanizme obnove
- Male, jednokorisničke baze podataka obično imaju malu ili uopće nemaju potporu obnovi – obnova se prepušta korisnikovoj odgovornosti – podrazumijeva se da korisnik periodički stvara arhivsku (*backup*) kopiju pomoću koje u slučaju potrebe obnavlja bazu podataka

# Uzroci pogrešaka

---

- pogreške opreme
- pogreške operacijskog sustava
- pogreške sustava za upravljanje bazama podataka
- Pogreške aplikacijskog programa
- pogreške operatera
- kolebanje izvora energije
- požar, sabotaža, ...

# Općenito pravilo koje omogućuje obnovu

---

- Redundancija - svaki se podatak mora moći rekonstruirati iz nekih drugih informacija redundantno pohranjenih negdje drugdje u sustavu (na traci, na drugom disku, na zrcalnom disku, ...)
- Redundancija se postiže:
  - zrcaljenjem podataka (*mirroring*)
  - sigurnosnim kopijama (*backup*)
  - dnevnicima izmjena (*logical log*) koji služe za:
    - poništavanje transakcija
    - ponovno obavljanje transakcija

# Općeniti opis postupka koji omogućuje obnovu

- ① Periodičko kopiranje sadržaja baze podataka na arhivski medij (drugi disk, diskovni automat (*jukebox*) specijaliziranih sustava za sigurnosne kopije; nekad su to bile mag. trake)  
( $1 \times$  dnevno,  $1 \times$  tjedno - ovisno o učestalosti promjena)
- ② Svaka izmjena u bazi podataka evidentira se u **logičkom dnevniku izmjena** (*logical log, journal*)
  - stara vrijednost zapisa, nova vrijednost zapisa
  - korisnik, vrijeme, ...
  - izmjena se **prvo zapisuje u dnevnik, a tek se onda provodi!**
  - dnevnići izmjena omogućuju
    - poništavanje transakcija (važno radi svojstva nedjeljivosti)
    - ponovno obavljanje transakcija (važno radi svojstva izdržljivosti)

Zašto?

# Kad nastane kvar ...

---



- **Ako je baza je potpuno uništena:**
  - Učitava se najsježija arhivska kopija (naredbom ROLLFORWARD DATABASE) – time se baza podataka dovodi u stanje kakvo je bilo u trenutku kad je napravljena posljednja arhivska kopija
  - Koristeći dnevnik izmjena ponovno se obavljaju izmjene koje su dogodile u međuvremenu - nakon izrade arhive
- **Baza nije uništena - sadržaj je nepouzdan -** program je prekinut tijekom obavljanja niza logički povezanih izmjena – potrebno je vratiti bazu podataka u ispravno stanje
  - pomoću podataka sadržanih u dnevniku izmjena poništavaju se sve izmjene koje su načinile nezavršene transakcije

# Dnevnik izmjena

---

Transakcija A

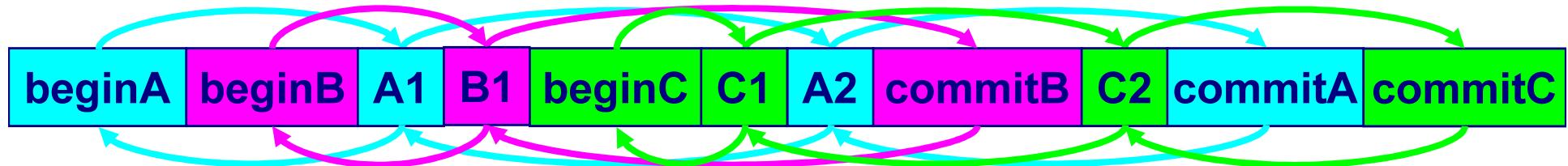
|        |    |    |         |
|--------|----|----|---------|
| beginA | A1 | A2 | commitA |
|--------|----|----|---------|

Transakcija B

|        |    |         |
|--------|----|---------|
| beginB | B1 | commitB |
|--------|----|---------|

Transakcija C

|        |    |    |         |
|--------|----|----|---------|
| beginC | C1 | C2 | commitC |
|--------|----|----|---------|



# Tipovi pogrešaka

---

- ① Pogreške transakcija (*transaction failure*) - pogreške koje su posljedica neplaniranog prekida transakcije
  - ② Pogreška računalskog sustava (*system failure*) - baza podataka nije fizički uništена
  - ③ Kvar medija za pohranu (*media failure*) - baza podataka je fizički uništena
- 

Slučaj ① - pomoću dnevnika izmjena poništavaju se efekti transakcije, kao da transakcija nikada nije započela s radom

Slučaj ② - transakcije koje su se obavljale u trenutku prekida se nakon ponovnog pokretanja poništavaju

Slučaj ③ - baza podataka se obnavlja pomoću arhivske kopije i pripadnog dnevnika izmjena

# Pogreške transakcija

---

- U slučaju pogreške transakcije SUBP će poništiti sve efekte transakcije
- Dovesti bazu u stanje kao da transakcija nije nikada započela s radom
- **Poništavanje izmjena** - pretragom dnevnika unatrag - nova vrijednost zapisa zamjenjuje se sa starom vrijednošću - sve dok se ne dođe do početka transakcije, odnosno do zapisa BEGIN WORK

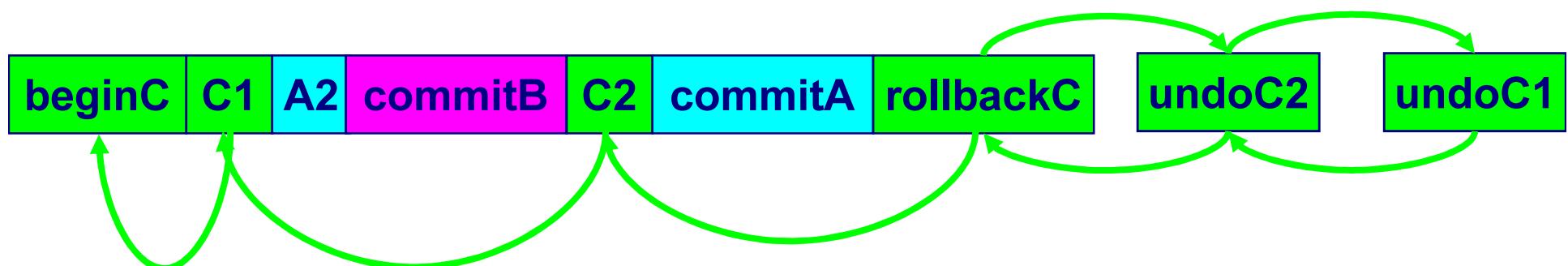
# Poništavanje transakcije pomoću dnevnika izmjena

---

Transakcija A    beginA | A1 | A2 | commitA

Transakcija B    beginB | B1 | commitB

Transakcija C    beginC | C1 | C2 | rollbackC



# Pogreške koje otkriva aplikacija

---

- Slučajevi u kojima aplikacija predviđa obavljanje naredbe  
**ROLLBACK WORK**

```
...
IF pom_saldo < 0 THEN
 ROLLBACK WORK;
ELSE
 COMMIT WORK;
END IF
...
```

# Pogreške koje ne otkriva aplikacija

- po završetku sjednice SUBP automatski poništava sve nepotvrđene transakcije
  - npr. ako se dogodi pogreška za koju program nema prepostavljenu reakciju, program (u konačnici i sjednica) završava na neplanirani način, te SUBP automatski obavlja **ROLLBACK WORK**

Primjer: pokušaj unosa zapisa čiji ključ već postoji u bazi:

početak programa

BEGIN WORK;

INSERT INTO osoba VALUES (1, 'Djetlić', 'Pero');

INSERT INTO osoba VALUES (2, 'Marić', 'Maro');

**INSERT INTO osoba VALUES (1, 'Katić', 'Kata');**

**INSERT INTO osoba VALUES (4, 'Matić', 'Mato');**

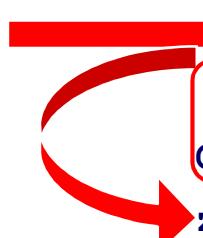
COMMIT WORK;

završetak programa

```
CREATE TABLE osoba (
 mbr INTEGER,
 prezime CHAR(20),
 PRIMARY KEY (mbr));
```

Pogreška!

neće se obaviti



# Način zapisivanja

---

- Koriste se spremnici (*buffer*):
  - Spremnik dnevnika
  - Spremnik baze podataka
- Sadržaj spremnika zapisuje se u dnevnik/bazu podataka:
  - Kad je spremnik popunjen ili
  - Kada SUBP izda nalog

# Očuvanje izmjena u slučaju razrušenja neposredno nakon završetka transakcije

---

- Postizanje izdržljivosti transakcije (*durability*)

Ako kvar nastane:

- nakon potvrđivanja i
- prije nego što su izmjene iz memorijskih spremnika prebačene u bazu podataka

izmjene bi u času kvara bile izgubljene, ALI:

- Procedura za ponovno pokretanje provest će promjene u bazi podataka
- Vrijednosti koje je potrebno zapisati u bazu podataka pronalaze se u odgovarajućim zapisima u dnevniku izmjena
- Sadržaj spremnika dnevnika mora biti zapisan u dnevnik na disku prije nego što završi procedura potvrđivanja transakcije (***write-ahead log rule***)

# Pogreške računalnog sustava

---

- Baza nije uništена
  - sve transakcije koje su se odvijale u trenutku kvara moraju biti poništene jer nisu završene!
  - pretraživanjem dnevnika od početka identificiraju se transakcije za koje postoji **BEGIN** i ne postoji **COMMIT** ili **ROLLBACK**
    - takav postupak bi predugo trajao
  - u određenim intervalima (obično svakih 5 minuta) određuje se **kontrolna točka** (*checkpoint*)

# Aktivnosti u kontrolnoj točki

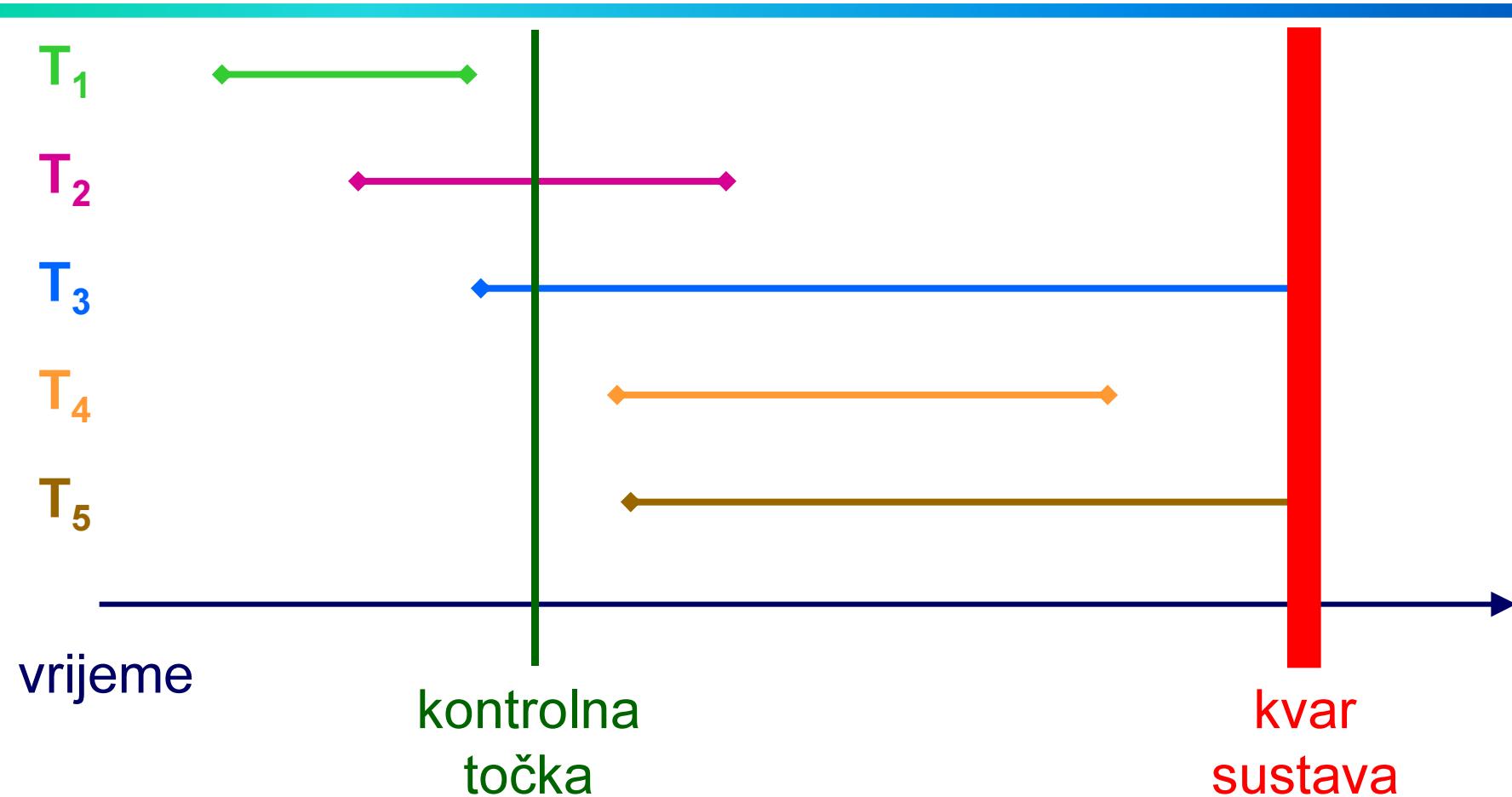
---

- ① pohrana sadržaja spremnika dnevnika (*log buffer*) u datoteku dnevnika
- ② zapisivanje zapisa kontrolne točke u datoteku dnevnika
- ③ zapisivanje adrese zapisa kontrolne točke iz datoteke dnevnika u datoteku za ponovno pokretanje (*restart file*)
- ④ pohrana sadržaja spremnika baze podataka (*database buffer*) u bazu podataka

Zapis kontrolne točke sadrži:

- listu svih aktivnih transakcija
- za svaku transakciju - adresu najnovijeg zapisa u datoteci dnevnika

## Primjer:



Transakcije  $T_3$  i  $T_5$  treba poništitи

Transakcije  $T_2$  i  $T_4$  treba ponovo obaviti

Zašto?

# Proces obnove

---

- Iz datoteke za ponovno pokretanje pročita se adresa posljednje kontrolne točke
- Iz datoteke dnevnika pročita se zapis kontrolne točke - lista transakcija koje su bile aktivne u kontrolnoj točki i adrese njihovih zadnjih zapisa
- Stvara se:
  - **lista za poništavanje** - na početku sadrži transakcije iz zapisa kontrolne točke
  - **lista za ponovo obavljanje** - na početku je prazna
- Pretražuje se dnevnik od kontrolne točke
  - transakcija za koju se pronađe **BEGIN** dodaje se u listu za poništavanje
  - transakcija za koju se pronađe **COMMIT** prebacuje se iz liste za poništavanje u listu za ponovo obavljanje
- Ponovo se obavljaju transakcije iz liste za ponovo obavljanje
- Poništavaju se transakcije iz liste za poništavanje

**SUBP ne može prihvatiti niti jedan zahtjev dok se ne završi proces obnove!**

# Ponovno obavljanje i poništavanje transakcija

---

## Logika ponovnog obavljanja

- Učinak ponovnog obavljanja, bez obzira koliko se puta obavljalo mora biti isti kao da je operacija obavljena **točno jednom!**

$$\text{redo}(\text{redo}(\text{redo}(\dots \text{redo}(x)))) = \text{redo}(x)$$

- Ponovno obavljanje = Obnova unaprijed (*forward recovery*)

## Logika poništavanja

- Učinak poništavanja, bez obzira koliko se puta obavljalo mora biti isti kao da je operacija obavljena **točno jednom!**

$$\text{undo}(\text{undo}(\text{undo}(\dots \text{undo}(x)))) = \text{undo}(x)$$

- Poništavanje = Obnova unatrag (*backward recovery*)

# Kvar medija za pohranu

---

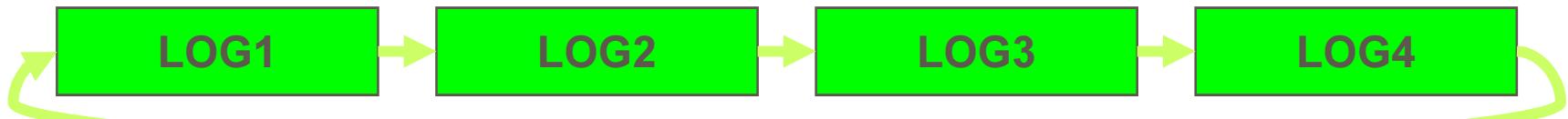
- **baza je fizički uništena - npr. zbog kvara diska**
- obnova sadržaja baze pomoću najnovije arhivske kopije
- pomoću najnovijeg dnevnika obavljaju se transakcije koje su bile provedene od trenutka arhiviranja
  - ako je najnovija arhivska kopija “pokvarena”
    - uzima se predzadnja arhivska kopija
    - dnevnik izmjena od predzadnje arhive do zadnje arhive
    - dnevnik izmjena nastalih nakon zadnje arhive

## PREPORUKE:

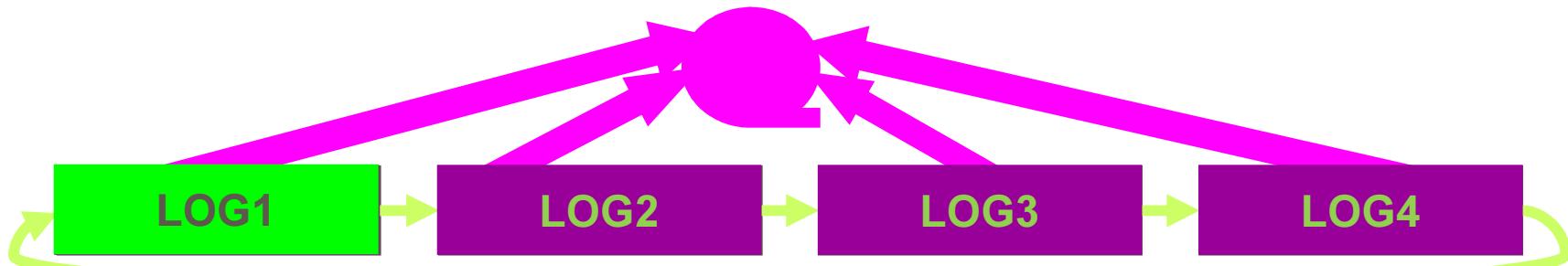
- čuvati najmanje tri posljednje arhive i pripadne dnevниke
- dnevnik se ne nalazi na istom disku na kojem je baza podataka
  - što ako je dnevnik “pokvaren”?

# Ciklička izmjena logičkih dnevnika

- Dnevnik može biti vrlo velik – započinje u času pokretanja arhiviranja i aktivan je do sljedećeg arhiviranja
- Dnevnići su ključni za obnovu - kako ih očuvati?
  - ➔ Čim prije treba njihov sadržaj pohraniti na „sigurno mjesto“
- Dnevnik se dijeli na manje odsječke koji se ciklički izmjenjuju



- Čim se jedan dnevnik popuni - kopira se na arhivski medij
- Dnevnik se mora nalaziti na disku sve dok su transakcije sadržane u njemu aktivne - da bi se omogućilo poništavanje (ROLLBACK)



- Čim se završe sve transakcije iz LOG1, on se oslobađa i može se ponovo koristiti

# Vremenski preduge i prevelike transakcije

---

- Vremenski preduge transakcije
  - Korisnik započne transakciju i „ode na kavu“
    - Podaci koje je transakcija zaključala nedostupni su ostalim korisnicima
    - Nepotrebno se zadržavaju dnevnički
      - ➔ U programskom sustavu (aplikaciji) ograničiti trajanje neaktivnih transakcija (pažljivo odabrati vremensku jedinicu!)
        - ➔ upozoriti korisnika (*time-out warning*)
        - ➔ ako korisnik ne reagira - prekid i poništavanje transakcije
  - Prevelike transakcije
    - Transakcija stvara vrlo mnogo izmjena
    - U slučaju prekida rada mnogo će izmjena biti izgubljeno
      - ➔ Velike transakcije gdje god je to moguće treba podijeliti na manje

## Ostali mehanizmi obnove

---

- Zrcaljenje podataka (*mirroring*)
- Arhiviranje tijekom obavljanja transakcija (*on-line backup*)
- Inkrementalno arhiviranje (*incremental backup*)

# Zrcaljenje

---

- Postoje dva jednakih područja – primarno područje i zrcalno područje
- Promjene se provode istovremeno u primarnom i zrcalnom području
- U slučaju pogreške u jednom od područja
  - ➔ nastavak rada na ispravnom području
  - ➔ nalog za ponovo kreiranje (“popravljanje”) zrcalnog područja
  - ➔ nakon „popravka“ područja se sinkroniziraju
- Zrcaljenje se može provoditi pod kontrolom sklopolja
  - RAID - Redundant Arrays of Independent (Inexpensive) Disks
- Zrcaljenje pod kontrolom SUBP-a – na razini baze podataka određuje se koji dio baze podataka će se zrcaliti

# Zrcaljenje na razini baze podataka

---

- Omogućuje finiju granulaciju zrcaljenog područja
  - mogu se zrcaliti samo dijelovi baze podataka - tablice koje uvijek moraju biti dostupne
- Visoka dostupnost – u slučaju kvara jednog područja dostupni su podaci u drugom
- Skalabilnost - podjela opterećenja
  
- Zrcaljenje ne može pomoći pri poništavanju transakcija niti kod kvara čitavog sustava
- Zrcaljenje je samo dodatni mehanizam za postizanje visoke dostupnosti
- Ne može zamijeniti arhiviranje i vođenje dnevnika

# Arhiviranje tijekom rada korisnika

---

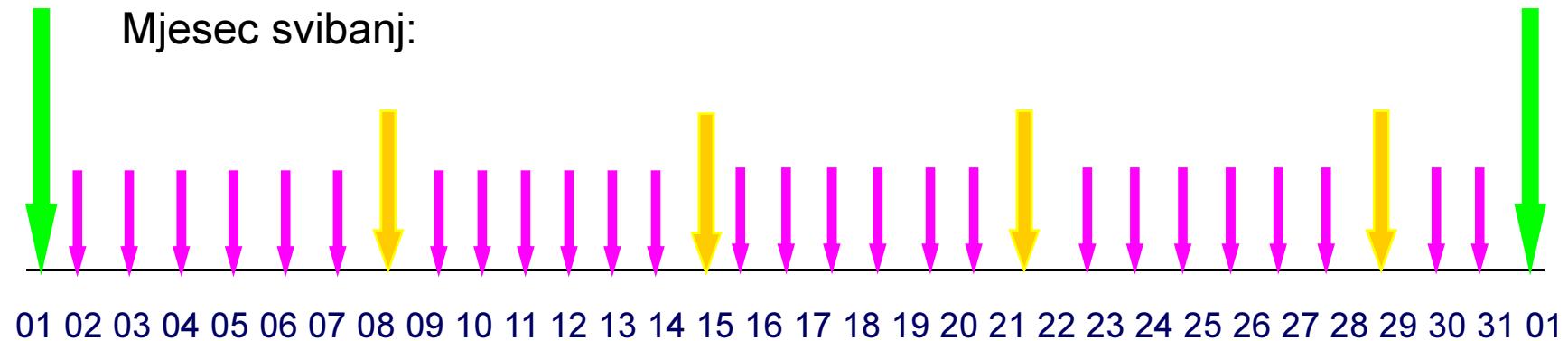
- *On-line backup*
- U klasičnim sustavima arhiviranje se obavljalo na sustavu na kojem nije bilo korisnika (npr. u petak na kraju radnog vremena)
- Današnji sustavi omogućuju da se arhiviranje obavlja tijekom rada korisnika, odnosno izvođenja transakcija
- Stanje baze podataka pohranjeno u arhivskoj kopiji je konzistentno i odgovara stanju kakvo je bilo u bazi podataka u času pokretanja arhiviranja.

# Inkrementalno arhiviranje

---

- Arhiviranje baze podataka dodatno opterećuje sustav i usporava rad korisnika
  - želi se skratiti trajanje i obim arhiviranja
- Inkrementalno arhiviranje omogućuje stvaranje arhiva različitih razina
- Na primjer (u različitim SUBP-ovima se koristi različita terminologija):
- **razina 0** – kopija čitave baze podataka – npr. jednom mjesечно
- **razina 1** – tjedna arhiva – sadrži promjene nastale nakon arhive razine 0
- **razina 2** – dnevna arhiva - sadrži promjene nastale nakon arhive razine 1

# ... Inkrementalno arhiviranje



→ arhiva razine 0

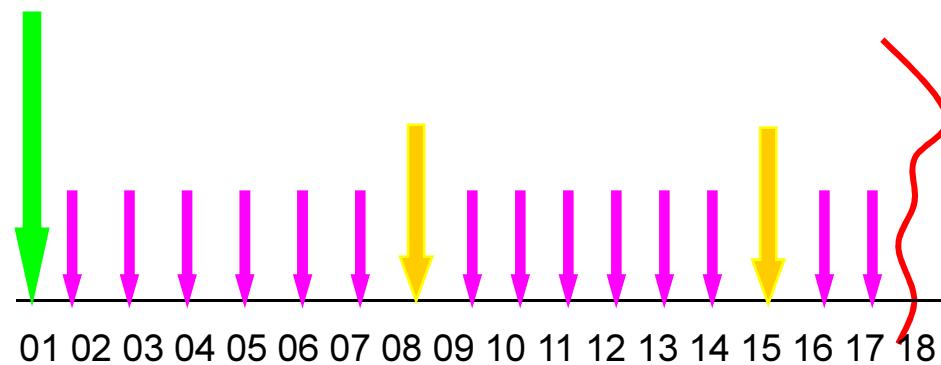
→ arhiva razine 1

→ arhiva razine 2

Ako se 18. svibnja dogodi kvar diska:

Pri obnovi se koriste se:

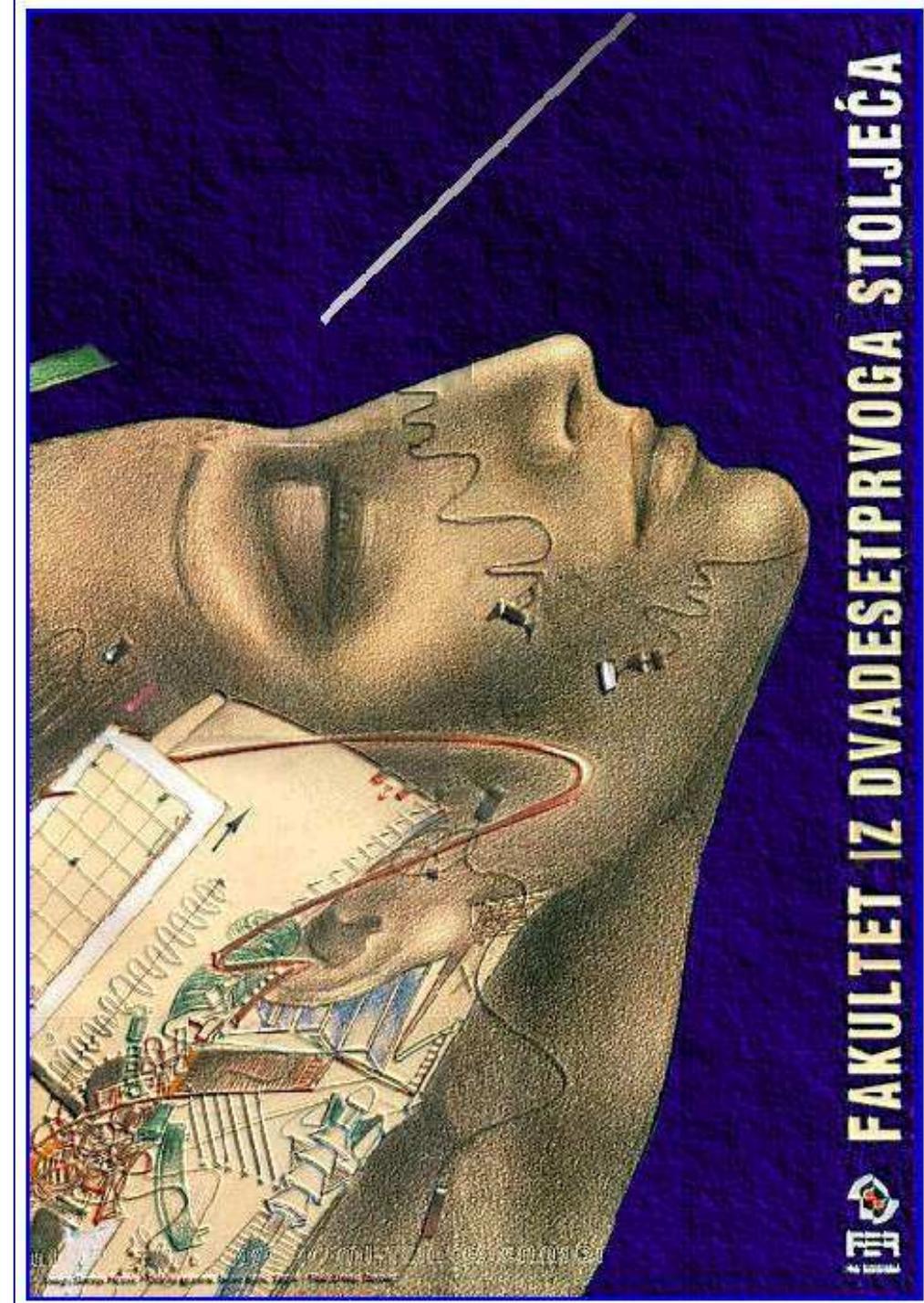
1. Arhiva razine 0 od 1. svibnja
2. Arhiva razine 1 od 15. svibnja
3. Arhiva razine 2 od 17. svibnja
4. Logički dnevnik koji je započeo nakon arhive razine 2 od 17. svibnja



# Baze podataka

Predavanja  
svibanj 2014.

## 18. Kontrola istodobnog pristupa



# Kontrola istodobnog pristupa (*Concurrency Control*)

---

- višekorisnički (*multiuser*) SUBP
  - ispravni i maksimalno dostupni podaci u uvjetima istodobnog pristupa velikog broja korisnika
  - komponenta SUBP-a zadužena za kontrolu istodobnog pristupa

## Jednostavan, ali neefikasan način korištenja višekorisničkog SUBP-a:

- korisnici obavljaju transakcije jednu za drugom (serijsko izvršavanje transakcija) - sustav počinje izvršavati sljedeću transakciju tek kad je u potpunosti završio prethodnu
  - slaba ukupna iskoristivost sustava (npr. CPU čeka završetak U/I operacije)
  - korisnik koji pokreće relativno kratku transakciju može (nepredvidivo) dugo čekati na završetak neke relativno duge transakcije

# Zbog čega je istodobni pristup nužan?

---

- budući da transakcije obuhvaćaju U/I i CPU operacije, njihovo istodobno obavljanje omogućilo bi istodobno korištenje različitih resursa računala
  - uvećava se broj transakcija obavljen u jedinici vremena (*throughput*), čime se uvećava ukupna iskoristivost sustava (*utilization*)
  - prosječno vrijeme koje protekne između aktiviranja i završetka transakcije (*average response time*) se smanjuje
- današnji sustavi su (većinom) višekorisnički, stoga serijsko izvršavanje transakcija (izvršavanje jedne po jedne transakcije) predstavlja neracionalno raspolaganje računalnim resursima
  - potrebno je omogućiti istodobno (ili prividno istodobno) izvršavanje transakcija

# Istodobni pristup i transakcija

---

Kontrola istodobnog pristupa (kao i postupak obnove baze podataka) usko su povezani s pojmom transakcije.

Transakcija je niz logički povezanih operacija koje se izvršavaju kao cjelina i prevode bazu podataka iz jednog u drugo **konzistentno stanje**.

**Rezultat transakcije ne smije ovisiti o tome odvijaju li se istodobno i neke druge transakcije!**

ACID svojstva transakcije:

- atomarnost, konzistentnost i izdržljivost
  - nisu ugroženi istodobnim pristupom
- izolacija - kada se istodobno obavljaju dvije ili više transakcija, njihov učinak mora biti jednak kao da su se obavljale jedna iza druge
  - problem kada više korisnika pristupa **istom** podatku/podacima njihove se aktivnosti (čitanje i/ili pisanje) **isprepliću**

# Primjer: Istodobni pristup i transakcija

---

- dva objekta u bazi podataka ( $x = 100$ ,  $y = 100$ )
- integritetsko ograničenje:  $x = y$
- operacije čitanja ili pisanja (*database operations*)
  - **pročitaj (x, p)** u varijablu p učitaj vrijednost elementa x
  - **zapiši (y, p)** u element y upiši vrijednost variable p

$T_1$

```
pročitaj (x,p)
p ← p + 100
zapiši (x, p)
pročitaj (y, p)
p ← p + 100
zapiši (y, p)
```

$T_2$

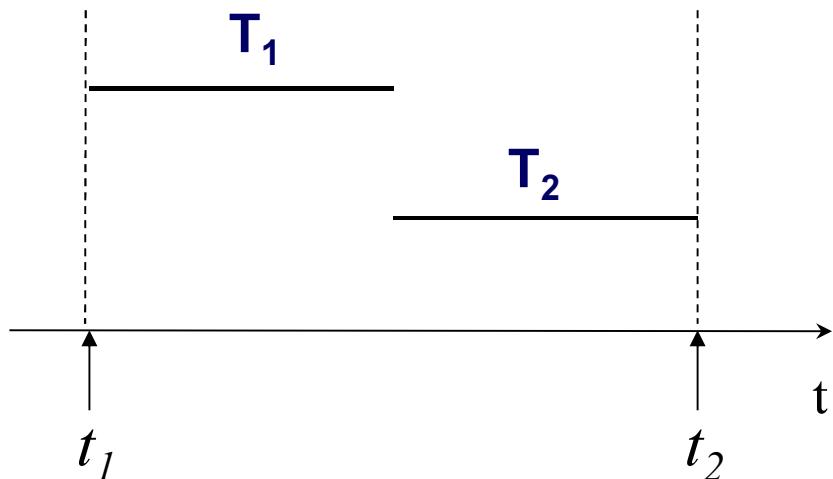
```
pročitaj(x, p)
p ← p * 2
zapiši(x, p)
pročitaj(y, p)
p ← p * 2
zapiši(y, p)
```

- transakcije su **korektne**

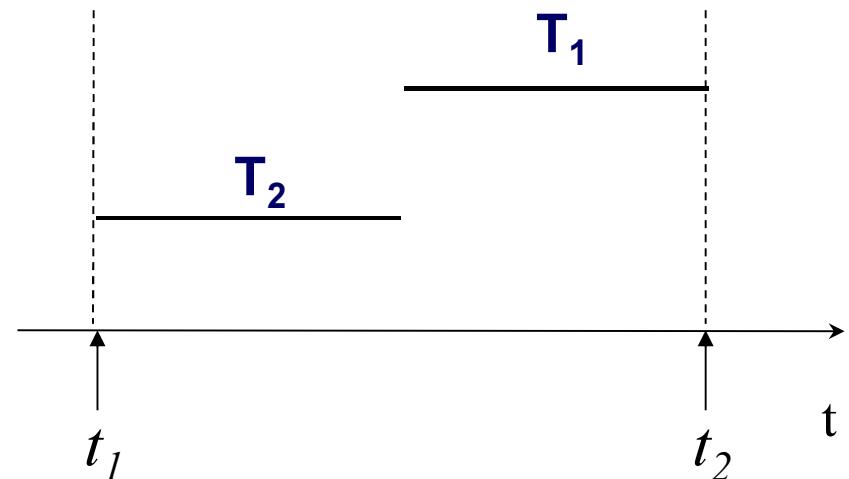
- korektna transakcija prevodi bazu podataka iz jednog konzistentnog u drugo konzistentno stanje
- ako se korektne transakcije izvršavaju međusobno izolirano (to znači jedna iza druge, serijski), neće narušiti konzistentnost baze podataka

# Istodobno izvršavanje transakcija

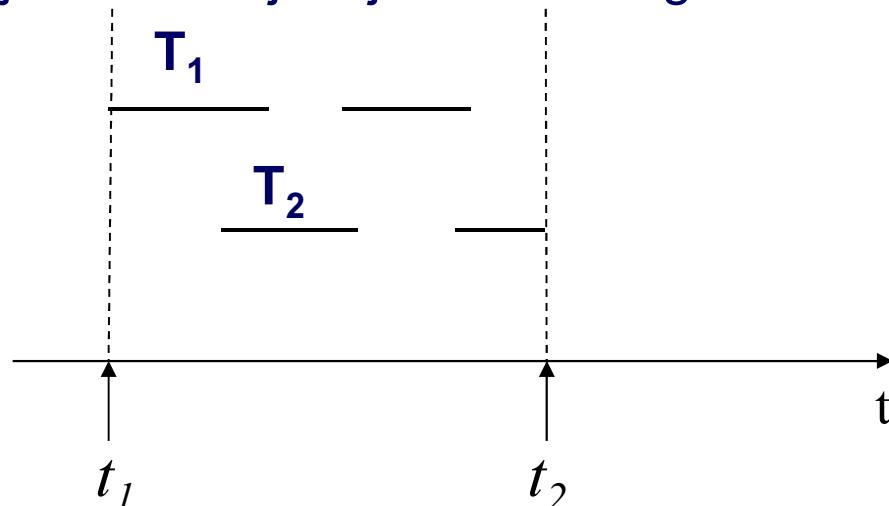
serijsko izvršavanje transakcija



ili



istodobno izvršavanje transakcija – jedan od mogućih redoslijeda



# Serijsko izvršavanje transakcija

Primjer:

a) redoslijed  $T_1, T_2$

| $T_1$                  | $T_2$ | x   | y   |
|------------------------|-------|-----|-----|
|                        |       | 100 | 100 |
| pročitaj(x, p)         |       |     |     |
| $p \leftarrow p + 100$ |       |     |     |
| zapiši (x, p)          |       | 200 |     |
| pročitaj(y, p)         |       |     |     |
| $p \leftarrow p + 100$ |       |     |     |
| zapiši (y, p)          |       |     | 200 |
| pročitaj(x, p)         |       |     |     |
| $p \leftarrow p * 2$   |       |     |     |
| zapiši (x, p)          |       | 400 |     |
| pročitaj(y, p)         |       |     |     |
| $p \leftarrow p * 2$   |       |     |     |
| zapiši (y, p)          |       |     | 400 |

b) redoslijed  $T_2, T_1$

| $T_1$                  | $T_2$ | x   | y   |
|------------------------|-------|-----|-----|
|                        |       | 100 | 100 |
| pročitaj(x, p)         |       |     |     |
| $p \leftarrow p * 2$   |       |     |     |
| zapiši (x, p)          |       | 200 |     |
| pročitaj(y, p)         |       |     |     |
| $p \leftarrow p * 2$   |       |     |     |
| zapiši (y, p)          |       |     | 200 |
| pročitaj(x, p)         |       |     |     |
| $p \leftarrow p + 100$ |       |     |     |
| zapiši (x, p)          |       | 300 |     |
| pročitaj(y, p)         |       |     |     |
| $p \leftarrow p + 100$ |       |     |     |
| zapiši (y, p)          |       |     | 300 |

primjer iz [Garcia-Molina]

# Istodobno izvršavanje transakcija

c) redoslijed izvršavanja koji narušava konzistentnost baze podataka

| $T_1$                  | $T_2$                | x   | y   |
|------------------------|----------------------|-----|-----|
|                        |                      | 100 | 100 |
| pročitaj(x, p)         |                      |     |     |
| $p \leftarrow p + 100$ |                      |     |     |
| zapiši (x, p)          |                      | 200 |     |
|                        | pročitaj(x, p)       |     |     |
|                        | $p \leftarrow p * 2$ |     |     |
|                        | zapiši (x, p)        | 400 |     |
|                        | pročitaj(y, p)       |     |     |
|                        | $p \leftarrow p * 2$ |     |     |
|                        | zapiši (y, p)        | 200 |     |
| pročitaj(y, p)         |                      |     |     |
| $p \leftarrow p + 100$ |                      |     |     |
| pročitaj(y, p)         |                      |     |     |

# Istodobno izvršavanje transakcija

d) redoslijed izvršavanja koji ne narušava konzistentnost baze podataka

| $T_1$                  | $T_2$                | x   | y   |
|------------------------|----------------------|-----|-----|
|                        |                      | 100 | 100 |
| pročitaj(x, p)         |                      |     |     |
| $p \leftarrow p + 100$ |                      |     |     |
| zapiši (x, p)          |                      | 200 |     |
|                        | pročitaj(x, p)       |     |     |
|                        | $p \leftarrow p * 2$ |     |     |
|                        | zapiši (x, p)        | 400 |     |
| pročitaj(y, p)         |                      |     |     |
| $p \leftarrow p + 100$ |                      |     |     |
| zapiši (y, p)          |                      |     | 200 |
|                        | pročitaj(y, p)       |     |     |
|                        | $p \leftarrow p * 2$ |     |     |
|                        | zapiši (y, p)        | 400 |     |

Redoslijed izvršavanja nije serijski ali je učinak izvršavanja jednak učinku serijskog izvršavanja.

Svaki takav redoslijed ne narušava konzistentnost baze podataka – za njega se kaže da je **serijalizabilan**.

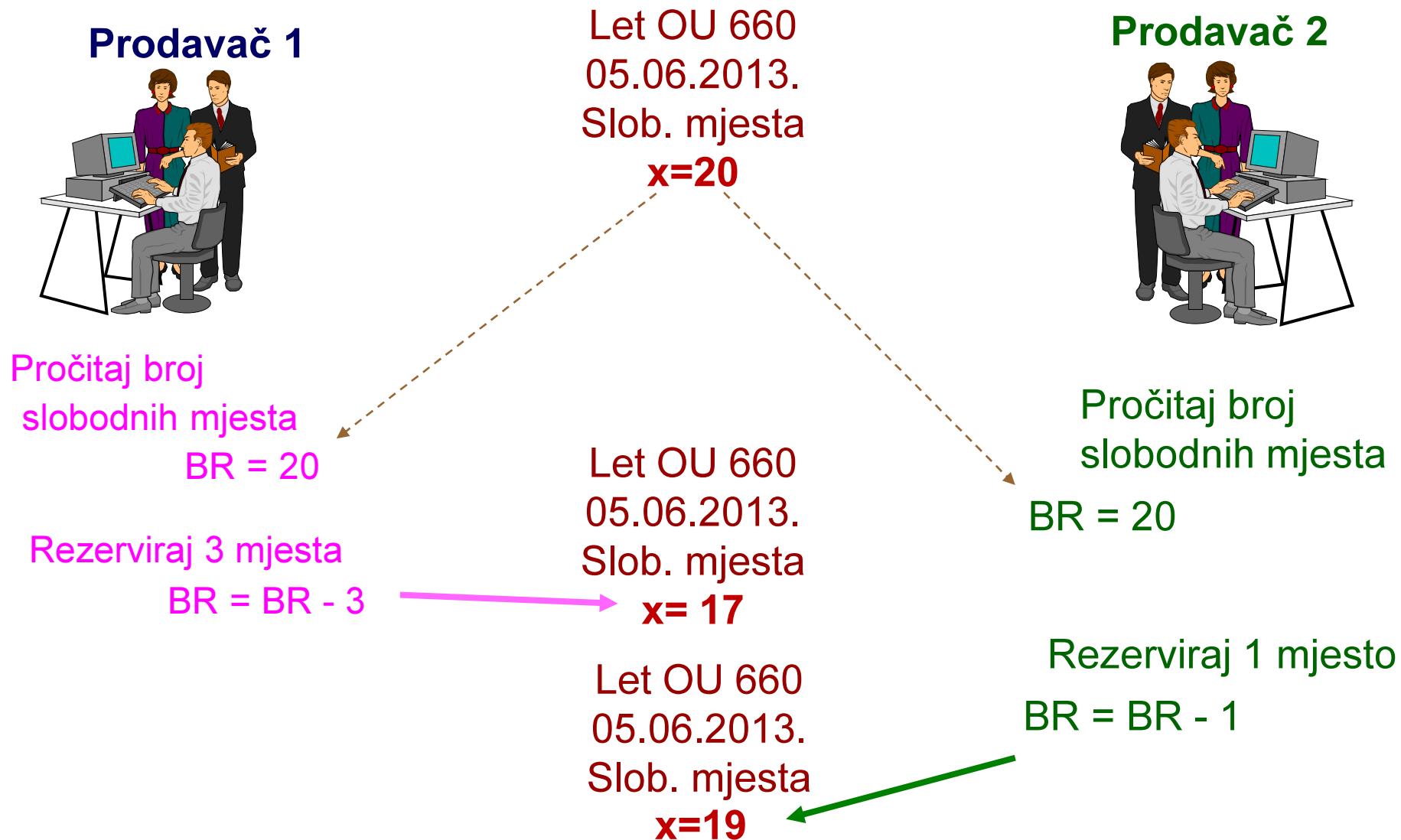
# Karakteristični problemi istodobnog pristupa

Prema SQL standardu neki od karakterističnih problema istodobnog pristupa su:

- P1 – prljavo čitanje *(dirty read)*
  - P2 – neponovljivo čitanje *(nonrepeatable read)*
  - P3 – sablasne n-torke *(phantom rows)*
  - P4 – izgubljena izmjena *(lost update)*

# Izgubljena izmjena (Lost update)

## Primjer: Rezervacija zrakoplovnih karata



# Prljavo čitanje (*Dirty read*)

## Transakcija 1

...

```
UPDATE osoba
SET prez = 'Horvat'
WHERE mbr = 1111
```

...

```
ROLLBACK WORK;
```

osoba

| mbr  | prez  | ime  |
|------|-------|------|
| 1111 | Novak | Ivan |
| 2222 | Kolar | Iva  |

## Transakcija 2

.

.

```
SELECT * FROM osoba;
```

```
1111 Horvat Ivan
2222 Kolar Iva
```

n-torka koja s prikazanim vrijednostima atributa nikad nije stvarno postojala u bazi podataka

„prljavi podaci“ – podaci izmijenjeni nepotvrđenom transakcijom (čiji efekti će naknadno biti poništeni)

# Neponovljivo čitanje i sablasne n-torke

- Ista transakcija obavljanjem istog upita mora dobiti uvijek isti rezultat (osim ako sama nije promijenila podatke čije čitanje ponavlja)

## Transakcija 1

```
SELECT saldo FROM racun
WHERE brRacun = 2;
```

...

```
-- A ne mijenja saldo za
-- racun s brojem 2
```

...

```
SELECT saldo FROM racun
WHERE brRacun = 2;
```

Rezultat: 400.00

Rezultat mora (opet) biti:  
400.00

## Primjer: Neponovljivo čitanje (*Nonrepeatable read*)

---

### Transakcija 1

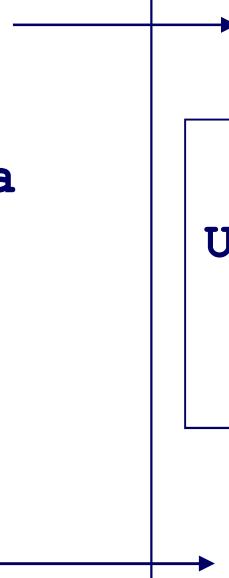
```
SELECT saldo FROM racun
WHERE brRacun = 2;
```

...

```
-- A ne mijenja saldo za
-- racun s brojem 2
```

...

```
SELECT saldo FROM racun
WHERE brRacun = 2;
```



Rezultat: 400.00

### Transakcija 2

```
UPDATE racun
SET saldo = saldo * 1.1
WHERE brRacun = 2
```

- Ista transakcija obavljanjem istog upita dobije drugačiji rezultat

## Primjer: Sablasne n-torke (*Phantom rows*)



### Transakcija 1

```
SELECT COUNT(*)
 FROM racun
 WHERE saldo > 100
.
-- A ne mijenja racun
.
SELECT COUNT(*)
 FROM racun
 WHERE saldo > 100
```

Rezultat: 2

### Transakcija 2

```
INSERT INTO racun
VALUES (3, 400.00)
```

Rezultat: 3 !!!

- Ista transakcija obavljanjem istog upita dobije drugačiji rezultat - zbog toga što je u međuvremenu transakcijom B unesena n-torka koja zadovoljava kriterij upita

## Kontrola istodobnog pristupa u SUBP - rješenja

---

- protokol zasnovan na zaključavanju
- protokol korištenja vremenskih oznaka
- protokol zasnovan na validaciji
- protokol temeljen na grafovima
- ...

## Protokol zasnovan na zaključavanju

---

- transakcija može zaključati podatak (podatke)
  - sprečava druge transakcije da pristupe podatku dok ga ona ne otključa
- dio SUBP-a (*locking manager*) zaključava zapise i prosuđuje u slučajevima kad postoji više zahtjeva za zaključavanjem istog podatka

## Prodavač 1



Zaključaj x

Pročitaj broj  
slobodnih mesta

BR = 20

Rezerviraj 3 mesta

BR = BR - 3

Otključaj x

## Zaključavanje

## Prodavač 2



Let OU 660  
10.05.2012.  
Slob. mesta  
 $x=20$

Zaključaj x

Let OU 660  
10.05.2012.  
Slob. mesta

$x=17$

Pročitaj broj  
slobodnih mesta

BR = 17

Rezerviraj 1 mjesto

BR = BR - 1

Otključaj x

$x=16$

## Vrste zaključavanja

---

- ključ za pisanje/izmjenu - **WRITE LOCK, EXCLUSIVE LOCK**
  - transakcija  $T_1$  zaključa objekt za pisanje
  - niti jedna druga transakcija ga **ne može zaključati** (niti za čitanje niti za pisanje) **dok ga  $T_1$  ne otključa**
  - svaka operacija **izmjene** (SQL naredbe INSERT, UPDATE, DELETE) postavlja ključ za pisanje
  
- ključ za čitanje - **READ LOCK, SHARED LOCK**
  - transakcija  $T_1$  (SQL naredbom SELECT) zaključa objekt za čitanje
  - bilo koja druga transakcija ga također **može zaključati za čitanje**
  - niti jedna ga transakcija **ne može zaključati za pisanje**

# Matrica kompatibilnosti ključeva

---

Proces 2  
pokušava  
postaviti na  
isti objekt  
ključ:

Proces 1 postavio je na objekt ključ:

|       | READ | WRITE | NO LOCK |
|-------|------|-------|---------|
| READ  | ✓    | ✗     | ✓       |
| WRITE | ✗    | ✗     | ✓       |

# Serijalizabilnost – je li zaključavanje dovoljno?

| $T_1$                  | $T_2$                | x   | y   |
|------------------------|----------------------|-----|-----|
|                        |                      | 100 | 100 |
| zaključaj (x)          |                      |     |     |
| pročitaj(x, p)         |                      |     |     |
| $p \leftarrow p + 100$ |                      |     |     |
| zapiši (x, p)          |                      | 200 |     |
| otključaj (x)          |                      |     |     |
|                        | zaključaj (x)        |     |     |
|                        | pročitaj(x, p)       |     |     |
|                        | $p \leftarrow p * 2$ |     |     |
|                        | zapiši (x, p)        | 400 |     |
|                        | otključaj (x)        |     |     |
|                        | zaključaj (y)        |     |     |
|                        | pročitaj(y, p)       |     |     |
|                        | $p \leftarrow p * 2$ |     |     |
|                        | zapiši (y, p)        | 200 |     |
|                        | otključaj (y)        |     |     |
| zaključaj (y)          |                      |     |     |
| pročitaj(y, p)         |                      |     |     |
| $p \leftarrow p + 100$ |                      |     |     |
| zapiši (y, p)          |                      |     |     |
| otključaj (y)          |                      | 300 |     |

# Protokol dvofaznog zaključavanja

## *Two-phase locking protocol (2PL)*

---

Serijalizabilni redoslijed izvršavanja osigurava svojstvo izolacije (I iz ACID-a).

Serijalizabilnost redoslijeda izvršavanja je osigurana ako sve transakcije poštuju protokol dvofaznog zaključavanja:

- ① prije obavljanja operacije nad objektom (npr. n-torkom iz baze), transakcija mora za taj objekt zatražiti ključ
- ② nakon otpuštanja ključa transakcija ne smije više zatražiti nikakav ključ
- transakcije koje poštuju 2PL protokol imaju 2 faze - fazu pribavljanja ključeva (faza rasta - ***growing phase***) i fazu otpuštanja ključeva (fazu sužavanja - ***shrinking phase***)

# Protokol dvofaznog zaključavanja – otpuštanje ključeva

- prema protokolu dvofaznog zaključavanja ključevi se otpuštaju u fazi sužavanja
- faza otpuštanja ključeva najčešće je stiješnjena u jednu operaciju (COMMIT ili ROLLBACK na kraju transakcije)

```
BEGIN WORK;
INSERT INTO osoba
VALUES (4567, "Jurić", "Ana");
...

UPDATE osoba
SET ime = "Anita"
WHERE mbr= 1111;
...

COMMIT WORK;
```

osoba

| mbr  | prez  | ime  |
|------|-------|------|
| 1111 | Novak | Ivan |
| 2222 | Kolar | Iva  |

Za n-torku se postavlja ključ za pisanje

Za n-torku se postavlja ključ za pisanje

Otpuštaju se ključevi za obje n-torke

# Granulacija podataka

---

Granulacija podataka je određena relativnom veličinom objekta koji će biti zaključan:

- n-torka
  - fizička stranica
  - relacija
  - baza podataka
- ↑ finija granulacija  
↓ grublja granulacija
- granulacija podataka pri zaključavanju utječe na performance sustava
    - odabirom finije granulacije uvećava se konkurentnost i troškovi postavljanja ključeva
    - odabirom grublje granulacije smanjuje se konkurentnost i troškovi postavljanja ključeva
  - koja je granulacija "najbolja"?
    - ovisi o konkretnim operacijama transakcije

# Primjer: Granulacija podataka

| osoba | mbr  | prez  | ime  |
|-------|------|-------|------|
|       | 1111 | Novak | Ivan |
|       | ...  | ...   | ...  |
|       | 2222 | Kolar | Iva  |

} 100 000 n-torki

$T_1$

```
SELECT ime, prez FROM osoba
WHERE mbr = '1111';
```

$T_2$

```
SELECT ime, prez FROM osoba;
```

- uz granulaciju određenu relacijom  
⇒ slaba konkurentnost, nepotrebno se ograničava pristup svim n-torkama relacije
  - koristiti granulaciju određenu n-torkom!
- 
- uz granulaciju određenu n-torkom  
⇒ loše performance, pojedinačno se postavlja 100 000 ključeva
  - koristiti granulaciju određenu relacijom!
- 
- očito, sustav mora podržavati zaključavanje na više razina granulacije

# Granulacija zaključavanja - IBM Informix

---

- baza podataka - DATABASE
  - DATABASE *dbname* EXCLUSIVE
- relacija (datoteka, tablica) - RELATION, TABLE
  - LOCK TABLE *tabname* IN { SHARE | EXCLUSIVE} MODE
- memorijska stranica - MEMORY PAGE
- n-torka (redak) - ROW
  - CREATE TABLE ( ... ) LOCK MODE { ROW | PAGE}
- indeks - INDEX, KEY
  - pod kontrolom SUBP - npr. čuvanje "mjesta" za ključ čiji je zapis obrisan - za slučaj poništavanja transakcije

Primjer:

```
CREATE TABLE racun (
 brRac INTEGER ...
) LOCK MODE ROW;
```

```
CREATE TABLE stavkaRacuna (
 brRac INTEGER
 rbrStavka ...
) LOCK MODE PAGE;
```

## Razina izolacije - motiv

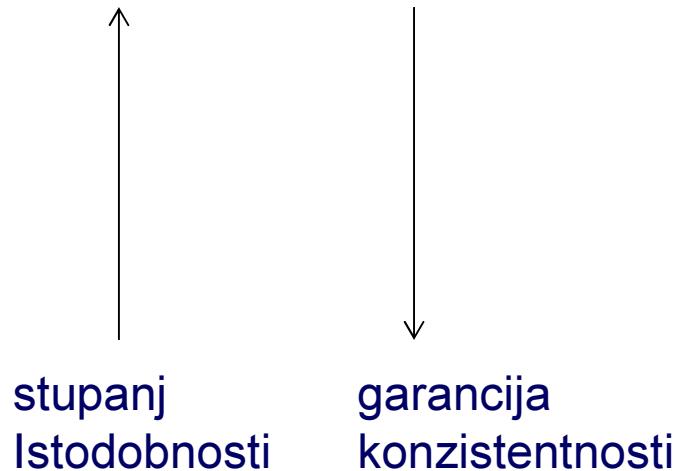
---

- serijalizabilnost se osigurava protokolom dvofaznog zaključavanja
  - ključevi se zadržavaju barem dok se ne postave svi transakciji potrebni ključevi
  - što se ključevi dulje zadržavaju povećava se vjerojatnost da će druga transakcija „zatražiti“ ključ nad već zaključanim objektom
- za neke primjene serijalizabilnost nije nužna
  - npr. transakcija koja agregira velik broj n-torki će tolerirati nekonzistentnost u zamjenu za poboljšane performanse
- koncept **razina izolacije** je razvijen da bi se omogućilo transakcijama balansiranje između konzistentnosti i istovremenosti

## Razina izolacije: SQL-92

---

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE



- definira se na razini transakcije
- za različite transakcije moguće je definirati različite razine izolacije
- promjenom razine izolacije mijenja se ponašanje transakcije pri postavljanju ključeva za čitanje

# Razina izolacije: SQL-92

## READ UNCOMMITTED

- podaci se čitaju bez zaključavanja i bez provjere da li su možda zaključani
  - mogu se pojaviti n-torce koje nikada nisu potvrđene u bazi podataka (zapravo nisu ni postojale u bazi podataka)

## READ COMMITTED

- čitaju se isključivo potvrđene n-torce
- provjerava se da li je trenutno pročitani podatak zaključan za pisanje
- provjera se obavlja postavljanjem „kratkotrajnog” ključa za čitanje
- ključ je postavljen samo za vrijeme čitanja podatka

## REPEATABLE READ

- osigurava ponovljivo čitanje podataka u okviru transakcije
- podatak se zaključava i ostaje zaključan ključem za čitanje do kraja transakcije
- ne sprječava pojavu sablasnih n-torki

## SERIALIZABLE

- čitanjem se podatak zaključava ključem za čitanje i ostaje zaključan do kraja transakcije
- sprječava probleme: prljavo čitanje, neponovljivo čitanje, sablasne n-torce i izgubljena izmjena

# Razina izolacije: SQL standard – IBM Informix

| SQL-92           | IBM Informix     |                                 |
|------------------|------------------|---------------------------------|
|                  | SET ISOLATION TO | SET TRANSACTION ISOLATION LEVEL |
| READ UNCOMMITTED | DIRTY READ       | READ UNCOMMITTED                |
| READ COMMITTED   | COMMITTED READ   | READ COMMITTED                  |
| REPEATABLE READ  |                  |                                 |
| SERIALIZABLE     | REPEATABLE READ  | SERIALIZABLE                    |

- moguće koristiti i izvan eksplisitno zadanih granica transakcije
- traje do kraja korisničke sjednice ili do sljedeće SET ISOLATION TO naredbe

```
BEGIN WORK;
SET ISOLATION TO DIRTY READ;
SELECT ... ;
SET ISOLATION TO REPEATABLE READ;
INSERT INTO...;
COMMIT WORK; -- Executes without error
```

- moguće koristiti isključivo u okviru eksplisitno zadanih granica transakcije
- nije moguće mijenjati unutar iste transakcije

```
BEGIN WORK;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT ... ;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
-- error 876: Cannot issue SET TRANSACTION -- in an active
transaction.
```

# Primjer: READ UNCOMMITTED – prljavo čitanje

Program A:  
BEGIN WORK;

UPDATE osoba SET prezime = "Jurić"  
WHERE sifOsoba = 2345;

ROLLBACK WORK;

SELECT \* FROM osoba;

| rezultat:   |       |
|-------------|-------|
| 1111 Novak  | Ivan  |
| 2345 Kolar  | Iva   |
| 3456 Horvat | Krešo |

Program B:

BEGIN WORK;  
SET TRANSACTION ISOLATION  
LEVEL READ UNCOMMITTED;

SELECT \* FROM osoba;

| rezultat:   |       |
|-------------|-------|
| 1111 Novak  | Ivan  |
| 2345 Jurić  | Iva   |
| 3456 Horvat | Krešo |

SELECT \* FROM osoba;

| rezultat:   |       |
|-------------|-------|
| 1111 Novak  | Ivan  |
| 2345 Kolar  | Iva   |
| 3456 Horvat | Krešo |

## Primjer: READ UNCOMMITTED – sablasne n-torke

Program A:  
BEGIN WORK;

INSERT INTO osoba VALUES  
(4567, "Jurić", "Ana");  
...

ROLLBACK WORK;

SELECT \* FROM osoba;

| rezultat:   |       |
|-------------|-------|
| 1111 Novak  | Ivan  |
| 2345 Kolar  | Iva   |
| 3456 Horvat | Krešo |

| rezultat:   |       |
|-------------|-------|
| 1111 Novak  | Ivan  |
| 2345 Kolar  | Iva   |
| 3456 Horvat | Krešo |
| 4567 Jurić  | Ana   |

| rezultat:   |       |
|-------------|-------|
| 1111 Novak  | Ivan  |
| 2345 Kolar  | Iva   |
| 3456 Horvat | Krešo |

Program B:

BEGIN WORK;  
SET TRANSACTION ISOLATION  
LEVEL READ UNCOMMITTED;

SELECT \* FROM osoba;

SELECT \* FROM osoba;

# Primjer: READ COMMITTED – sprječavanje prljavog čitanja

---

```
SELECT * FROM osoba;
```

rezultat:

|             |       |
|-------------|-------|
| 1111 Novak  | Ivan  |
| 2345 Kolar  | Iva   |
| 3456 Horvat | Krešo |

Program A:

```
BEGIN WORK;
```

```
UPDATE osoba
SET ime = "Ana"
WHERE sifOsoba = 2345;
```

.

.

.

```
ROLLBACK WORK;
```

Program B:

```
BEGIN WORK;
SET TRANSACTION ISOLATION
LEVEL READ COMMITTED;
```

```
SELECT * FROM osoba;
```

POGREŠKA: Zapis zaključan!

# Problemi koji se javljaju kod različitih razina izolacije SQL-92

|                  | Prljavo čitanje   | Neponovljivo čitanje | Sablasne n-torke  |
|------------------|-------------------|----------------------|-------------------|
| UNCOMMITTED READ | Da <sup>(1)</sup> | Da                   | Da <sup>(2)</sup> |
| COMMITTED READ   | Ne <sup>(3)</sup> | Da                   | Da                |
| REPEATABLE READ  | Ne                | Ne                   | Da                |
| SERIALIZABLE     | Ne                | Ne                   | Ne                |

(1) Primjer: READ UNCOMMITTED – prljavo čitanje

(2) Primjer: READ UNCOMMITTED – sablasne n-torke

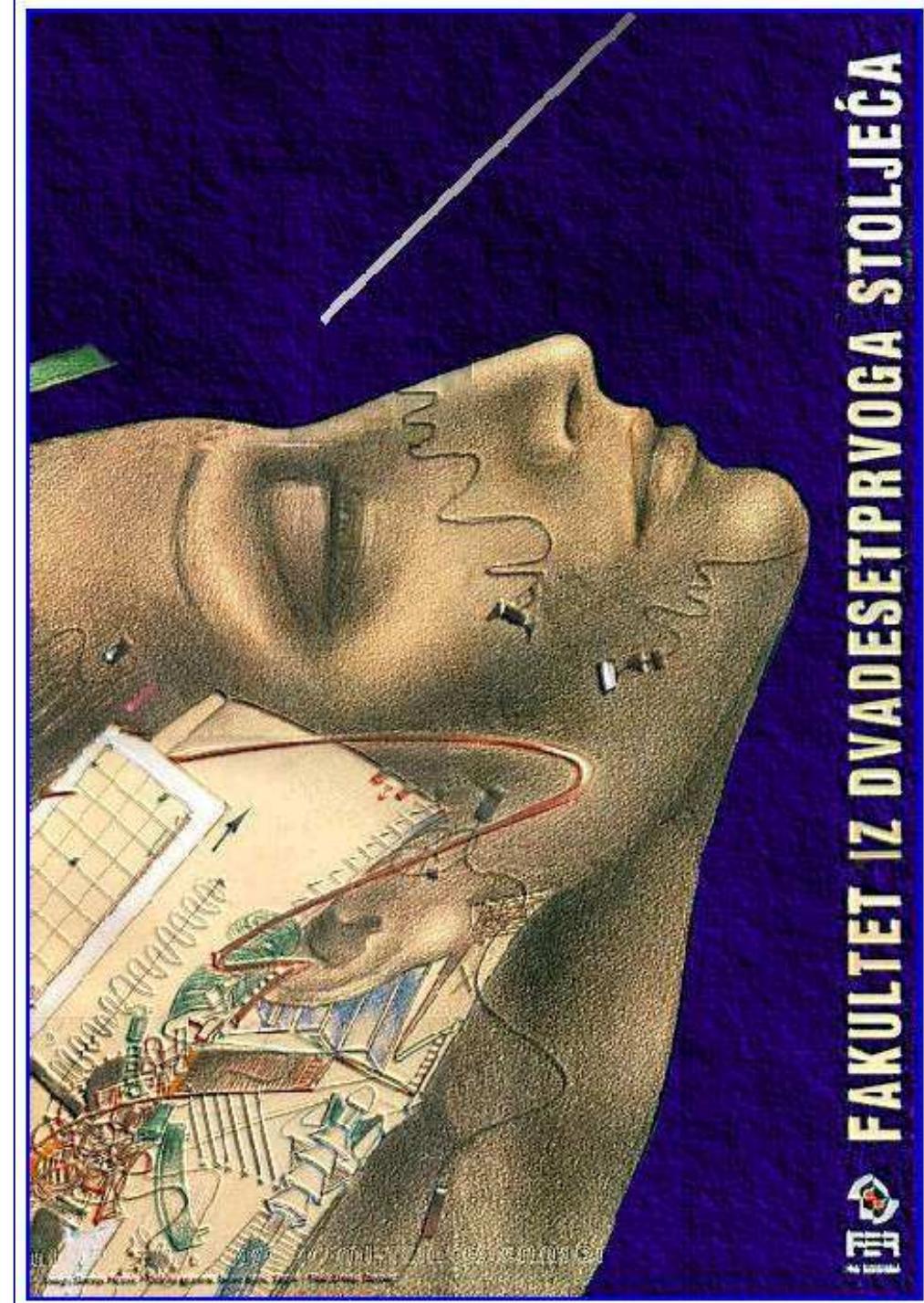
(3) Primjer: READ COMMITTED – sprječavanje prljavog čitanja

- READ UNCOMMITTED
  - prljavo čitanje
  - neponovljivo čitanje
  - sablasne n-torke
- READ COMMITTED
  - neponovljivo čitanje
  - sablasne n-torke
- REPEATABLE READ
  - sablasne n-torke
- SERIALIZABLE
  - -

# Baze podataka

Predavanja  
lipanj 2014.

## 19. Sigurnost baze podataka



# Integritet i sigurnost baze podataka

---

- Pojmovi integritet i sigurnost baze podataka se često spominju zajedno, međutim radi se o dva različita aspekta zaštite podataka
  - Integritet baze podataka (*database integrity*) - operacije nad podacima koje korisnici obavljaju **su ispravne** (tj. uvijek rezultiraju konzistentnim stanjem baze podataka)
    - "podaci se štite od ovlaštenih korisnika"
  - Sigurnost baze podataka (*database security*) - korisnici koji obavljaju operacije nad podacima **su ovlašteni** za obavljanje tih operacija
    - "podaci se štite od neovlaštenih korisnika"

Među ovim pojmovima postoje i sličnosti. U oba slučaja:

- moraju biti definirana **pravila** koja korisnici ne smiju narušiti
- pravila se pohranjuju u rječnik podataka
- SUBP nadgleda rad korisnika - osigurava poštivanje pravila

# Oblici narušavanja sigurnosti i moguće posljedice

---

- Oblici narušavanja sigurnosti baze podataka su:
  - neovlašteno čitanje podataka
  - neovlaštena izmjena podataka
  - neovlašteno uništavanje podataka
- Moguće posljedice su:
  - krađa ili prijevara
  - gubitak tajnosti
    - odnosi se na podatke kritične za funkcioniranje organizacije
    - npr. krađa recepture - rezultira gubitkom konkurentnosti na tržištu
  - gubitak privatnosti
    - odnosi se na osobne podatke
    - npr. krađa podataka o zdravstvenom stanju osobe - rezultira sudskim procesom protiv vlasnika baze podataka
  - gubitak raspoloživosti
    - npr. uništenjem dijela podataka

# Protumjere

---

- sigurnost baze podataka se osigurava zaštitom na nekoliko razina
  - **zaštita na razini SUBP**
    - spriječiti pristup bazama podataka ili onim dijelovima baza podataka za koje korisnici nisu ovlašteni
  - **zaštita na razini operacijskog sustava**
    - spriječiti pristup radnoj memoriji računala ili datotekama u kojima SUBP pohranjuje podatke
  - **zaštita na razini računalne mreže**
    - spriječiti presretanje poruka (*sniffing*) na internetu i intranetu
  - **fizička zaštita**
    - fizički zaštititi lokaciju računalnog sustava
  - **zaštita na razini korisnika**
    - spriječiti da ovlašteni korisnici nepažnjom ili namjerno (npr. u zamjenu za mito ili druge usluge) omoguće pristup podacima neovlaštenim osobama

# Aspekti zaštite podataka

---

- **zakonski, socijalni i etički aspekt**
  - ima li vlasnik baze podataka zakonsko pravo na prikupljanje i korištenje podataka
  - npr. smije li zdravstvena ustanova koja, u skladu sa zakonom prikuplja podatke o pacijentima, te iste podatke koristiti pri donošenju odluke hoće li svog bivšeg pacijenta zaposliti
- **strategijski aspekt**
  - tko definira pravila pristupa - tko određuje kakve ovlasti ima pojedini korisnik baze podataka, ...
- **operativni aspekt**
  - kako osigurati poštivanje pravila - kojim mehanizmima se osigurava poštivanje definiranih pravila, na koji način su lozinke zaštićene, koliko često se mijenjaju, ...

# Ustav RH - Članak 37.

---

Svakom se jamči sigurnost i tajnost osobnih podataka. Bez privole ispitanika, osobni se podaci mogu prikupljati, obrađivati i koristiti samo uz uvjete određene zakonom.

Zakonom se uređuje zaštita podataka te nadzor nad djelovanjem informatičkih sustava u Republici.

Zabranjena je uporaba osobnih podataka suprotna utvrđenoj svrsi njihovoga prikupljanja.

- Zakon o zaštiti osobnih podataka

# Korisnici SUBP i ovjera autentičnosti

---

- administrator sustava (operacijskog sustava ili SUBP) omogućuje korisniku pristup sustavu (operacijskom sustavu ili SUBP) definiranjem jedinstvenog identifikatora korisnika (*user name*, *user ID*, *login ID*) i pripadne lozinke (*password*) koja je poznata samo dotičnom korisniku i sustavu
- korisnik koji pristupa sustavu (operacijskom sustavu ili SUBP) poznavanjem lozinke ovjerava svoju autentičnost (*authentication*)
- za ovjeru autentičnosti korisnika SUBP može koristiti
  - mehanizme operacijskog sustava  
ili
  - vlastite mehanizme

# Autorizacija i modeli kontrole pristupa

---

- Autorizacija je postupak kojim se određenom korisniku dodjeljuje dozvola za obavljanje određenih vrsta operacija (čitanje, izmjena, brisanje, ...) nad određenim objektima baze podataka (relacija, pogled, atribut, ...)
  - podaci o dodijeljenim dozvolama pohranjuju se u rječnik podataka
- Prije obavljanja svake operacije, SUBP provjerava ima li korisnik dozvolu za obavljanje operacije nad objektom
  - kontrola pristupa (*access control*)
- Današnji SUBP podržavaju dva različita modela kontrole pristupa podacima
  - **mandatna kontrola pristupa** (*MAC-Mandatory Access Control*)
  - **diskrecijska kontrola pristupa** (*DAC-Discretionary Access Control*)

# Mandatna kontrola pristupa

---

- manji broj SUBP podržava mandatnu kontrolu pristupa
  - koristi se relativno rijetko u odnosu na diskrecijsku kontrolu pristupa
- mandatna kontrola pristupa je primjenjiva u sustavima u kojima se dozvole dodjeljuju na temelju pozicije korisnika u hijerarhiji neke organizacije (vojska, državna uprava, ...)
- svaki **objekt** dobiva oznaku klasifikacijske razine (*classification level*), npr. povjerljivo, tajno, vrlo tajno, ...
- svakom **korisniku** dodjeljuje se oznaka razine ovlasti (*clearance level*)
  - korisnici mogu obavljati operacije nad onim objektima za koje imaju odgovarajuću razinu ovlasti

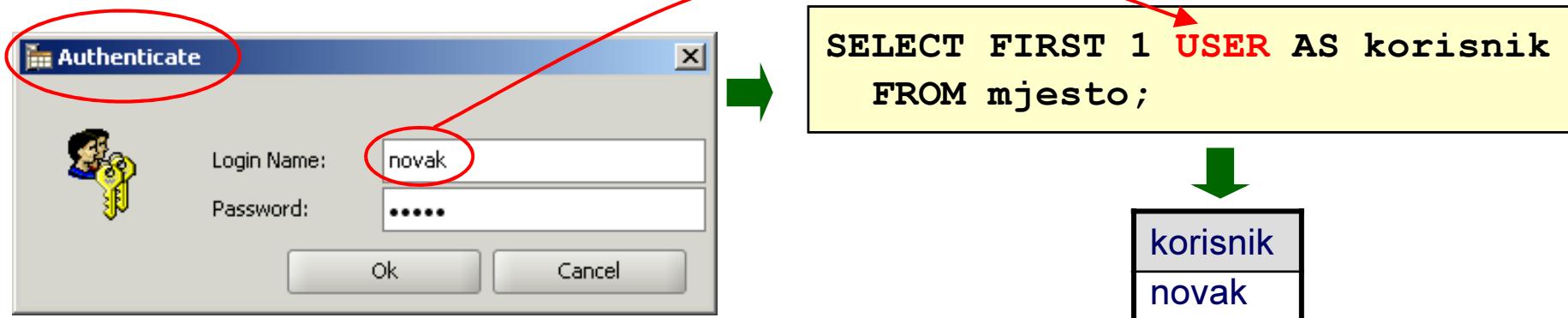
# Diskrecijska kontrola pristupa

---

- većina današnjih SUBP podržava diskrecijsku kontrolu pristupa
  - diskrecijska kontrola pristupa je podržana SQL standardom
- određenom korisniku se eksplicitno dodjeljuje dozvola za obavljanje određene operacije nad određenim objektom
  - dozvole su opisane trojkama <korisnik, objekt, vrsta operacije>
    - <horvat, ispit, čitanje>
    - <horvat, ispit, izmjena>
    - <horvat, predmet, čitanje>
    - <novak, predmet, čitanje>
  - kada korisnik novak pokuša obaviti operaciju čitanja objekta (relacije) predmet, SUBP provjerava postoji li dozvola u obliku trojke <novak, predmet, čitanje>
- u preostalom dijelu predavanja razmatrat će se diskrecijska kontrola pristupa

# Korisnici u SQL-u

- korisnik s određenom identifikacijskom oznakom (*userID*)
  - pri uspostavljanju SQL-sjednice korisnik se prijavljuje svojim identifikatorom korisnika, te lozinkom ovjerava svoju autentičnost
  - funkcija USER vraća vrijednost identifikatora korisnika koji se koristi u dotičnoj SQL-sjednici



- bilo koji korisnik (PUBLIC)
  - dodjelom dozvole "korisniku" PUBLIC, dozvolu za obavljanje operacije dobivaju svi sadašnji i budući korisnici

# Objekti i vlasnici objekata u SQL-u

---

- **Objekti**
  - relacija (tablica, *table*)
  - atribut (stupac tablice, *column*)
  - virtualna relacija (pogled, *view*)
  - baza podataka
- **Vlasnik objekta (*object owner*)**
  - vlasnik objekta je korisnik koji je kreirao objekt, npr:
    - vlasnik baze podataka je korisnik koji je kreirao bazu podataka
    - vlasnik relacije je korisnik koji je kreirao relaciju
  - vlasnik objekta implicitno dobiva dozvole za obavljanje **svih** vrsta operacija nad objektom, uključujući dozvole za:
    - dodjeljivanje svih vrsta dozvola nad tim objektom drugim korisnicima
    - uništavanje objekta

# Vrste dozvola u SQL-u na razini baze podataka (*dbPrivilege*)

---

- Različiti SUBP imaju različita rješenja za dodjeljivanje dozvola na razini baze podataka. Ovdje je prikazano rješenje koje se koristi u sustavu IBM Informix:
  - **CONNECT**
    - uspostavljanje SQL-sjednice i obavljanje operacija nad objektima za koje je korisnik dobio dozvolu od vlasnika objekta ili je njihov vlasnik, kreiranje virtualnih i privremenih relacija
  - **RESOURCE**
    - CONNECT + kreiranje **novih** relacija u bazi podataka
  - **DBA**
    - RESOURCE + neovisno o vlasništvu i dozvolama nad objektima u bazi podataka: sve vrste operacija nad svim objektima, uništavanje svih objekata (uključujući i bazu podataka)
    - korisnik koji kreira bazu podataka je vlasnik te baze podataka i implicitno dobiva DBA (*Database administrator*) dozvolu

# Vrste dozvola u SQL-u na razini [virtualne] relacije (*tablePrivilege*)

---

- **SELECT [(*columnList*)]**
  - čitanje n-torki (ili vrijednosti navedenih atributa) [virtualne] relacije
- **UPDATE [(*columnList*)]**
  - izmjena n-torki (ili vrijednosti navedenih atributa) [virtualne] relacije
- **INSERT**
  - unos n-torki [virtualne] relacije
- **DELETE**
  - brisanje n-torki [virtualne] relacije
- **REFERENCES [(*columnList*)]**
  - korištenje **relacije** (ili samo navedenih atributa kao pozivane relacije pri definiranju stranog ključa)
- **INDEX**
  - kreiranje indeksa nad **relacijom**
- **ALTER**
  - izmjena strukture **relacije** i definiranje integritetskih ograničenja
- **ALL PRIVILEGES**
  - sve do sada navedene vrste operacija nad [virtualnom] relacijom

# SQL naredbe za dodjeljivanje i ukidanje dozvola

---

- GRANT *dbPrivilege* TO { PUBLIC | *userList* }
  
- REVOKE *dbPrivilege* FROM { PUBLIC | *userList* }
  
- GRANT *tablePrivilegeList* ON { *tableName* | *viewName* }  
    TO { PUBLIC | *userList* | *roleList* }  
    [ WITH GRANT OPTION ]
  
- REVOKE *tablePrivilegeList* ON { *tableName* | *viewName* }  
    FROM { PUBLIC | *userList* | *roleList* }  
    [ CASCADE | RESTRICT ]

# Primjer 1:

student

| matBr | ime    | prez  | pbr   | adresa   |
|-------|--------|-------|-------|----------|
| 100   | Ana    | Ivić  | 51000 | Korzo 2  |
| 102   | Ivan   | Perić | 10000 | Ilica 20 |
| 105   | Matija | Matić | 31000 | Unska 7  |
| 107   | Tea    | Bilić | 10000 | Vlaška 5 |

ispit

| matBr | nazPred    | datIsp   | ocj |
|-------|------------|----------|-----|
| 100   | Fizika     | 1.5.2010 | 3   |
| 102   | Matematika | 7.9.2009 | 1   |
| 102   | Matematika | 9.2.2010 | 5   |
| 107   | Fizika     | 5.4.2012 | 4   |

- kreirati bazu podataka studBaza i relacije student i ispit
  - vlasnik baze podataka i relacija treba biti korisnik badmin
- korisnik horvat treba dobiti dozvole:
  - pregled svih podataka u relacijama student i ispit
  - unos, izmjena, brisanje svih podataka u relaciji ispit
- korisnik novak treba dobiti dozvole:
  - pregled svih podataka u relaciji student
  - izmjena poštanskog broja i adrese u relaciji student
- korisnik kolar treba dobiti dozvolu:
  - pregled svih podataka u relaciji student, osim adrese

## Primjer 1 (nastavak):

**badmin** ← naredbe obavlja korisnik badmin

```
CREATE DATABASE studBaza;
CREATE TABLE student (...);
CREATE TABLE ispit (...);

GRANT CONNECT TO horvat;
GRANT CONNECT TO novak;
GRANT CONNECT TO kolar;

GRANT SELECT ON student
 TO horvat;
GRANT SELECT, INSERT
 , UPDATE, DELETE ON ispit
 TO horvat;

GRANT SELECT ON student
 TO novak;
GRANT UPDATE(pbr, adresa)
 ON student TO novak;

GRANT SELECT(matBr, ime
 , prez, pbr)
 ON student TO kolar;
```

- korisnik badmin je vlasnik baze podataka studBaza i relacija student i ispit. Posjeduje DBA dozvolu na razini baze podataka
- dozvole za uspostavljanje SQL-sjednice
- dozvole korisniku horvat za pregled podataka u relaciji student
- dozvole korisniku horvat za pregled, unos, izmjenu i brisanje podataka u relaciji ispit
- dozvola korisniku novak za pregled podataka u relaciji student
- dozvola korisniku novak za izmjenu vrijednosti atributa u relaciji student
- dozvola korisniku kolar za pregled svih podataka u relaciji student, osim adrese

## Primjer 2:

badmin

```
CREATE DATABASE studBaza;
GRANT RESOURCE TO horvat;
GRANT CONNECT TO novak;
```

korisnik badmin kreira bazu podataka studBaza. Kao vlasnik baze podataka implicitno dobiva DBA dozvolu na razini baze podataka

horvat

```
CREATE TABLE zupanija (
 sifZup INTEGER
, nazZup CHAR(30)
, PRIMARY KEY(sifZup));
GRANT SELECT, INSERT, UPDATE
ON zupanija TO novak;
```

može jer ima RESOURCE dozvolu

može jer je vlasnik relacije zupanija

novak

```
SELECT * FROM zupanija;
INSERT INTO zupanija ...;
UPDATE zupanija ...;
```

može jer ima barem CONNECT dozvolu (bez CONNECT dozvole ne bi mogao uspostaviti SQL-sjednicu), te dozvole koje je dobio od vlasnika relacije zupanija

## Primjer 2 (nastavak):

**novak**

```
DROP TABLE zupanija;
```

→ ne može jer nije vlasnik objekta niti ima DBA dozvolu

**kolar**

```
SELECT * FROM zupanija;
```

→ ne može jer nema niti CONNECT dozvolu (ne može uspostaviti SQL-sjednicu)

**horvat**

```
GRANT CONNECT TO kolar;
```

→ ne može jer nema DBA dozvolu

**bpadmin**

```
GRANT CONNECT TO kolar;
```

→ može jer ima DBA dozvolu

**horvat**

```
GRANT SELECT
ON zupanija TO kolar;
```

→ može jer je vlasnik relacije zupanija

**kolar**

```
SELECT * FROM zupanija;
```

→ može jer ima barem CONNECT dozvolu, te dozvolu za obavljanje operacije SELECT nad relacijom zupanija

## Primjer 2 (nastavak):

---

**novak**

```
CREATE TABLE mjesto ...;
```

→ ne može jer nema RESOURCE dozvolu

**horvat**

```
GRANT RESOURCE TO novak;
```

→ ne može jer nema DBA dozvolu

**bpadmin**

```
GRANT DBA TO horvat;
```

→ može jer ima DBA dozvolu

**horvat**

```
GRANT RESOURCE TO novak;
```

→ može jer ima DBA dozvolu

**novak**

```
CREATE TABLE mjesto (...
 REFERENCES zupanija ...);
```

→ ne može jer nema dozvolu za kreiranje stranog ključa koji se poziva na primarni ključ relacije zupanija (mogao bi kreirati relaciju bez stranog ključa jer ima RESOURCE dozvolu)

## Primjer 2 (nastavak):

horvat

```
GRANT REFERENCES
ON zupanija TO novak;
```



može jer ima DBA dozvolu (ali čak i da nema DBA dozvolu, vlasnik je relacije zupanija)

novak

```
CREATE TABLE mjesto (...
REFERENCES zupanija ...);
```



može jer ima RESOURCE dozvolu i dozvolu za kreiranje stranog ključa koji se poziva na primarni ključ relacije zupanija

horvat

```
GRANT CONNECT TO PUBLIC;
```



može jer ima DBA dozvolu

- sada svaki korisnik (sadašnji ili budući) koji uspije ovjeriti svoju autentičnost može uspostaviti SQL-sjednicu s bazom podataka studBaza

novak

```
GRANT SELECT
ON mjesto TO PUBLIC;
```



može jer je vlasnik relacije mjesto

- sada svaki korisnik (sadašnji ili budući) koji uspostavi SQL-sjednicu s bazom podataka (uz prethodnu ovjeru autentičnosti) može obavljati operaciju SELECT nad relacijom mjesto

# Dodjeljivanje prenosivih dozvola

- ukoliko se korisniku dozvola dodijeli uz navođenje opcije WITH GRANT OPTION, korisnik će moći dodjeljivati tu istu dozvolu ostalim korisnicima (unatoč tome što nije vlasnik objekta)

Primjer:

korisnik1

```
CREATE TABLE ispit (...);
GRANT SELECT ON ispit TO korisnik2 WITH GRANT OPTION;
GRANT SELECT ON ispit TO korisnik3 WITH GRANT OPTION;
```

korisnik2

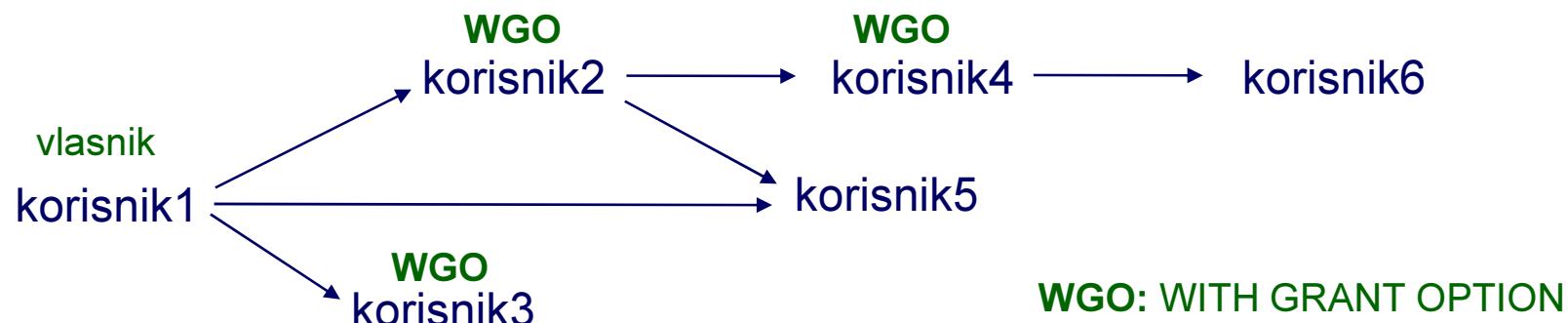
```
GRANT SELECT ON ispit TO korisnik4 WITH GRANT OPTION;
GRANT SELECT ON ispit TO korisnik5;
```

korisnik4

```
GRANT SELECT ON ispit TO korisnik6;
```

korisnik1

```
GRANT SELECT ON ispit TO korisnik5;
```



# Ukidanje dozvola

---

- korisnik koji je dozvolu dodijelio, tu istu dozvolu može ukinuti naredbom REVOKE

Primjer:

- vlasnik baze podataka studBaza je korisnik badmin
- vlasnik relacije mjesto je korisnik horvat

horvat

```
GRANT SELECT, UPDATE ON mjesto TO novak WITH GRANT OPTION;
```

novak

```
GRANT SELECT, UPDATE ON mjesto TO kolar;
```

- npr. naredbu:  

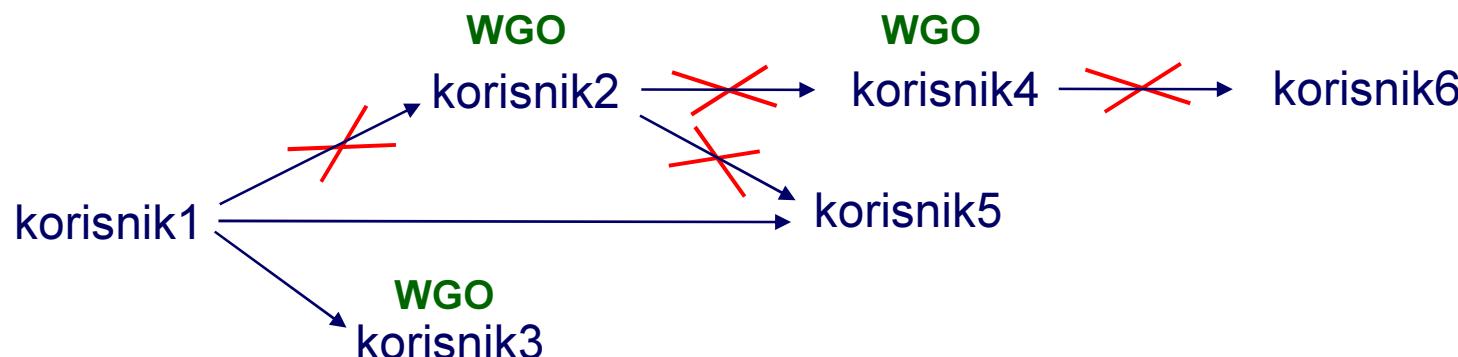
```
REVOKE UPDATE ON mjesto FROM kolar;
```
- može obaviti korisnik novak jer je novak korisnik koji je dozvolu dodijelio

# Ukidanje dozvola dodijeljenih temeljem WITH GRANT OPTION

- ukidanjem dozvole korisniku x (koji je dozvole dalje dodjeljivao temeljem ovlasti stečene pomoću WITH GRANT OPTION) **uz primjenu opcije CASCADE**, dozvola se ukida i svim ostalim korisnicima koji su dotičnu dozvolu stekli od korisnika x (neposredno ili posredno)

Primjer: **korisnik1**

```
REVOKE SELECT ON ispit FROM korisnik2 CASCADE ;
```

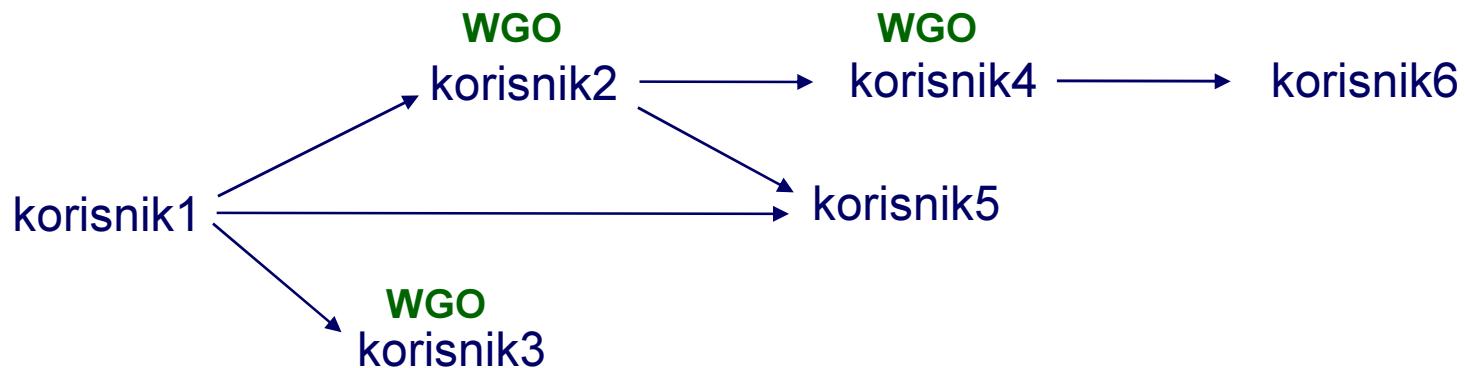


- obavljanjem naredbe dozvolu gube korisnik2, korisnik4 i korisnik6
- korisnik5 će izgubiti dozvolu koju je dobio od korisnika2, ali će zadržati dozvolu koju je dobio od korisnika1
- ukoliko se opcija CASCADE ne navede**, naredba REVOKE djeluje na jednak način kao kada je opcija CASCADE navedena

# Ukidanje dozvola dodijeljenih temeljem WITH GRANT OPTION

- ukidanjem dozvole korisniku x uz primjenu opcije **RESTRICT**, dozvola će biti ukinuta jedino u slučaju kada korisnik x nije dalje dodjeljivao ovlasti temeljem ovlasti stečene pomoću WITH GRANT OPTION

Primjer:



**korisnik1**

```
REVOKE SELECT ON ispit FROM korisnik2 RESTRICT;
```

SUBP odbija obaviti naredbu (dojavljuje pogrešku)

**korisnik2**

```
REVOKE SELECT ON ispit FROM korisnik4 RESTRICT;
```

SUBP odbija obaviti naredbu (dojavljuje pogrešku)

**korisnik1**

```
REVOKE SELECT ON ispit FROM korisnik3 RESTRICT;
```

SUBP obavlja naredbu (korisnik3 ostaje bez dozvole)

# Primjena virtualnih relacija

ispit

| mbrSt | nazPred    | datIsp   | ocj |
|-------|------------|----------|-----|
| 100   | Fizika     | 1.5.2010 | 3   |
| 102   | Matematika | 7.9.2009 | 1   |
| 102   | Matematika | 9.2.2010 | 5   |
| 107   | Fizika     | 5.4.2012 | 4   |

- vlasnik relacije ispit je korisnik horvat
- korisniku novak omogućiti pregled samo prosječnih ocjena po predmetima
- korisniku kolar omogućiti pregled, unos, izmjenu i brisanje samo za ispite iz predmeta Fizika

horvat

```
CREATE VIEW prosjek (nazPred, prosOcj) AS
 SELECT nazPred, AVG(ocj)
 FROM ispit
 GROUP BY nazPred;
GRANT SELECT ON prosjek TO novak;

CREATE VIEW ispitFizika AS
 SELECT * FROM ispit
 WHERE nazPred = 'Fizika'
 WITH CHECK OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE
 ON ispitFizika TO kolar;
```

- zašto je nužno virtualnu relaciju ispitFizika kreirati uz opciju WITH CHECK OPTION?!

# Dodjeljivanje kontekstno ovisnih dozvola

| ispit |         |          |     |
|-------|---------|----------|-----|
| mbrSt | sifPred | datIsp   | ocj |
| 100   | 100     | 1.5.2010 | 3   |
| 102   | 200     | 7.9.2009 | 1   |
| 102   | 200     | 9.2.2010 | 5   |
| 107   | 300     | 5.4.2012 | 4   |

| nast    |        |       |        |
|---------|--------|-------|--------|
| sifNast | imeN   | prezN | userId |
| 1001    | Slavko | Kolar | kolar  |
| 1002    | Ivo    | Ban   | ban    |
| 1003    | Ana    | Novak | novak  |

| predaje |         |
|---------|---------|
| sifNast | sifPred |
| 1001    | 100     |
| 1001    | 200     |
| 1002    | 200     |
| 1003    | 200     |
| 1003    | 300     |

- vlasnik relacija je korisnik horvat
- svakom nastavniku (korisnicima kolar, ban, novak) omogućiti pregled i izmjenu ispita samo iz predmeta koje predaju

horvat

LOŠE RJEŠENJE!

```
CREATE VIEW kolarIspiti AS
 SELECT * FROM ispit
 WHERE sifPred IN (
 SELECT sifPred FROM predaje
 WHERE sifNast = 1001) WITH CHECK OPTION;
GRANT SELECT, UPDATE ON kolarIspiti TO kolar;
```

- ponoviti za svakog nastavnika: banIspiti, novakIspiti, ...
- nova virtualna relacija za svakog novog nastavnika ( $\approx 150$  na FER-u)
- svaki nastavnik upit nad relacijom ispit mora pisati na drugačiji način

# Dodjeljivanje kontekstno ovisnih dozvola

| ispit |         |          |     | nast    |        |       |        | predaje |         |
|-------|---------|----------|-----|---------|--------|-------|--------|---------|---------|
| mbrSt | sifPred | datlsp   | ocj | sifNast | imeN   | prezN | userId | sifNast | sifPred |
| 100   | 100     | 1.5.2010 | 3   | 1001    | Slavko | Kolar | kolar  | 1001    | 100     |
| 102   | 200     | 7.9.2009 | 1   | 1002    | Ivo    | Ban   | ban    | 1001    | 200     |
| 102   | 200     | 9.2.2010 | 5   | 1003    | Ana    | Novak | novak  | 1002    | 200     |
| 107   | 300     | 5.4.2012 | 4   |         |        |       |        | 1003    | 200     |
|       |         |          |     |         |        |       |        | 1003    | 300     |

horvat

```
CREATE VIEW ispitiZaNastavnike AS
 SELECT * FROM ispit
 WHERE sifPred IN (
 SELECT sifPred FROM predaje, nast
 WHERE predaje.sifNast = nast.sifNast
 AND userId = USER) WITH CHECK OPTION;
 GRANT SELECT, UPDATE ON ispitiZaNastavnike TO kolar;
 GRANT SELECT, UPDATE ON ispitiZaNastavnike TO ban;
 GRANT SELECT, UPDATE ON ispitiZaNastavnike TO novak;
```

ISPRAVNO  
RJEŠENJE!

- "sadržaj" virtualne relacije ovisit će o identifikatoru nastavnika koji je ostvario SQL-sjednicu
- smije li se nastavnicima dozvoliti izmjena vrijednosti atributa userId u relaciji nast ili sadržaj relacije predaje?!

# Upotreba sinonima

PROBLEM:

- nastavnici (odnosno aplikativni ili primjenski programi koje nastavnici koriste) moraju u upitima o ispitima koristiti virtualnu relaciju `ispitiZaNastavnike`

```
SELECT * FROM ispitiZaNastavnike WHERE ocj = 1;
```

- dekan (npr. korisnik s identifikatorom novosel), za razliku od nastavnika, dobiva sve dozvole nad relacijom ispit. U upitima o ispitima mora koristiti relaciju ispit

```
SELECT * FROM ispit WHERE ocj = 1;
```

- kada korisnik novosel prestane biti dekan, ukinut će mu se dozvola nad relacijom ispit, a dodijeliti dozvola nad virtualnom relacijom `ispitiZaNastavnike`. U svojim upitima morat će koristiti virtualnu relaciju `ispitiZaNastavnike`

```
SELECT * FROM ispitiZaNastavnike WHERE ocj = 1;
```

# Upotreba sinonima

RJEŠENJE:

- Kreirati sinonime: alternativna imena za relacije ili virtualne relacije

korisnik s  
DBA  
dozvolom

```
CREATE PRIVATE SYNONYM kolar.ispitiZaSve FOR ispitiZaNastavnike;
CREATE PRIVATE SYNONYM ban.ispitiZaSve FOR ispitiZaNastavnike;
... sinonimi za ostale nastavnike i sinonim za dekana
CREATE PRIVATE SYNONYM novosel.ispitiZaSve FOR ispit;
```

- sada i dekan i nastavnici mogu koristiti isto ime objekta kada postavljaju upite o ispitima

```
SELECT * FROM ispitiZaSve WHERE ocj = 1;
```

- kada korisnik novosel prestane biti dekan

horvat  
  
korisnik s  
DBA  
dozvolom

```
REVOKE SELECT, UPDATE ON ispit FROM novosel;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO novosel;

DROP SYNONYM novosel.ispitiZaSve;
CREATE PRIVATE SYNONYM novosel.ispitiZaSve FOR ispitiZaNastavnike;
```

- korisnik novosel će i dalje u svojim upitimima moći koristiti ime objekta ispitiZaSve, ali će kao rezultat dobivati samo one podatke na koje, sada u svojstvu nastavnika, ima pravo

# Dodjeljivanje istih dozvola velikom broju korisnika

## PROBLEM:

- svakom nastavniku treba dodijeliti dozvole za
  - pregled, unos i izmjenu podataka o ispitimima za predmete koje predaje, pregled podataka iz relacije `nast`, iz relacije `predaje`, itd.
  - 150 nastavnika  $\Rightarrow$  150 puta treba obaviti niz naredbi za dodjelu dozvola:

```
GRANT SELECT, INSERT, UPDATE ON ispitizaNastavnike TO kolar;
GRANT SELECT ON predmet TO kolar;
GRANT SELECT ON nast TO kolar;
...
-- ponoviti za svakog od 150 nastavnika
```

- za svakog novog zaposlenog nastavnika ponoviti postupak
- kada nastavnik ode u mirovinu, mora se obaviti niz REVOKE naredbi
- ako se promijene pravila pristupa (npr. odluci se da nastavnici mogu brisati "svoje" ispite), promjena se mora provesti za svakog nastavnika posebno:

```
GRANT DELETE ON ispitizaNastavnike TO kolar;
-- ponoviti za svakog od 150 nastavnika
```

# Dodjeljivanje istih dozvola velikom broju korisnika

## RJEŠENJE:

- definira se uloga (*role*), npr. nastavnik
- dozvole se, umjesto direktno korisnicima-nastavnicima, dodjeljuju ulozi

```
CREATE ROLE nastavnik;
GRANT SELECT, INSERT, UPDATE ON ispitiZaNastavnike TO nastavnik;
GRANT SELECT ON nast TO nastavnik;
GRANT SELECT ON predaje TO nastavnik;
...
```

- svakom nastavniku, umjesto cijelog niza dozvola, dovoljno je dodijeliti dozvolu za korištenje uloge nastavnik

```
GRANT nastavnik TO kolar;
GRANT nastavnik TO ban;
...
```

- ako nastavnik s identifikatorom korisnika ban ode u mirovinu

```
REVOKE nastavnik FROM ban;
```

- ako nastavnici trebaju dobiti dozvolu za brisanje "svojih" ispita

```
GRANT DELETE ON ispitiZaNastavnike TO nastavnik;
```

# Korištenje dozvola dobivenih putem uloga

- nakon uspostavljanja SQL-sjednice, korisnik posjeduje sljedeće dozvole:
  1. sve dozvole koje su dodijeljene PUBLIC "korisniku"
  2. sve dozvole koje su dodijeljene izravno dotičnom korisniku
  3. sve dozvole nad objektima kojima je dotični korisnik vlasnik
  4. dozvole na razini baze podataka (npr. ako korisnik ima DBA dozvolu, dopušteno mu je obavljanje svih operacija nad svim objektima)
- ako korisnik namjerava koristiti i dozvole dodijeljene nekoj ulozi, mora obaviti naredbu (npr.): **SET ROLE nastavnik;**
  - od tog trenutka, korisnik će (osim dozvola navedenih pod 1-4) imati i dozvole dodijeljene ulozi nastavnik.
- korisniku može biti dodijeljena više nego jedna uloga, ali u jednom trenutku može koristiti samo jednu od njih. Npr. nakon obavljanja naredbe:  
**SET ROLE studentskiSavjetnik;**
  - korisnik će (osim dozvola navedenih pod 1-4) imati i dozvole dodijeljene ulozi studentskiSavjetnik (ali ne i ulozi nastavnik).
- naredbu **SET ROLE NONE;** korisnik koristi onda kad ne želi koristiti niti jednu ulogu

# Praćenje rada korisnika (auditing)

---

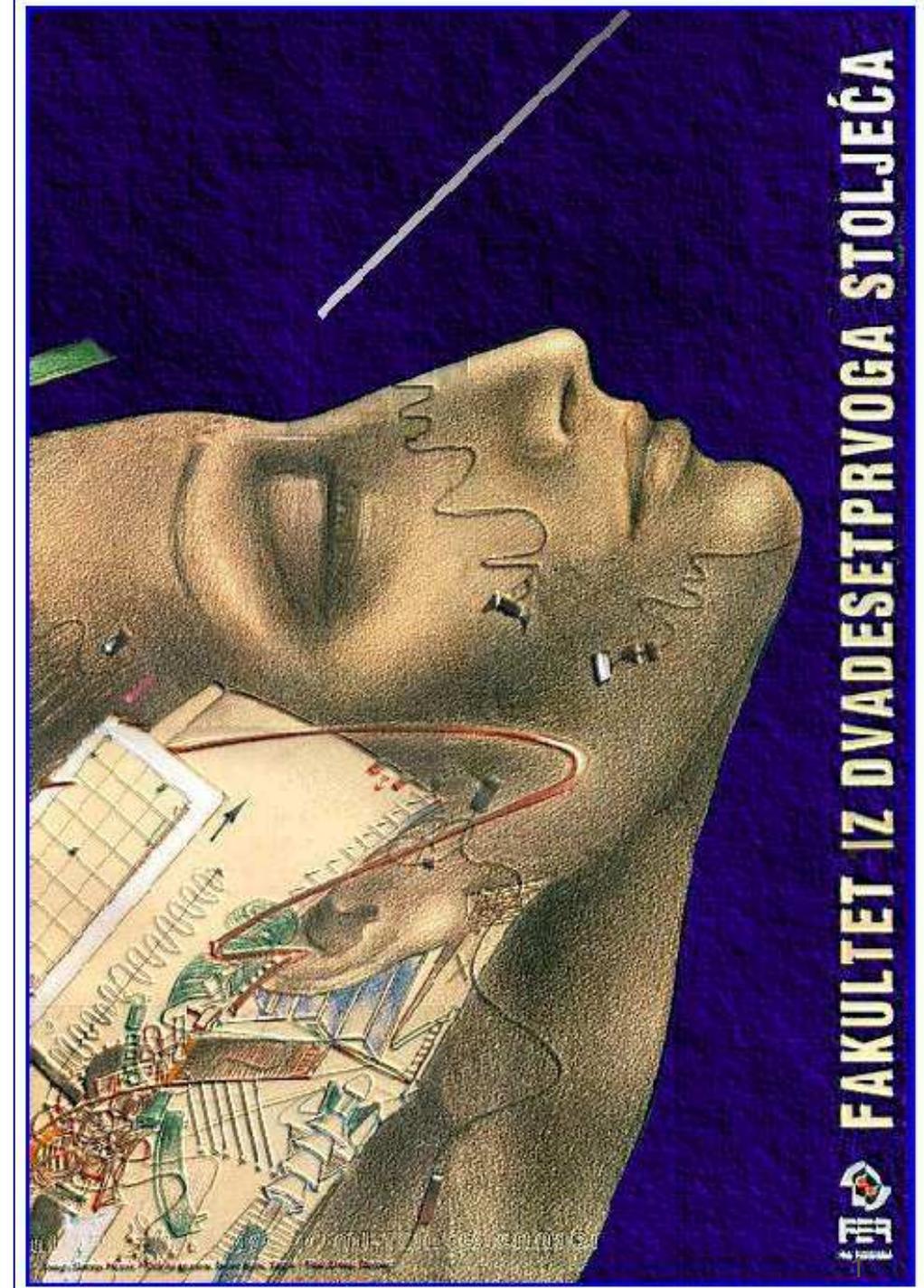
- evidentirati svaki pristup osjetljivim podacima u posebnoj datoteci za praćenje rada korisnika (*Audit Trail*)
- tipičan zapis datoteke sadrži sljedeće informacije:
  - SQL naredba koja se izvršava (*statement source*)
  - mjesto s kojeg je upućen zahtjev (terminal, IP adresa računala)
  - identifikator korisnika koji je pokrenuo operaciju
  - datum i vrijeme operacije
  - n-torke, atributi na koje se zahtjev odnosi
  - stara vrijednost n-torke
  - nova vrijednost n-torke
- sama činjenica da se prati "trag" obavljenih operacija nad podacima, često je dovoljna za sprečavanje zloporabe

# Baze podataka

Predavanja

20.  
Distribuirane  
i  
NoSQL  
baze podataka

lipanj 2014.

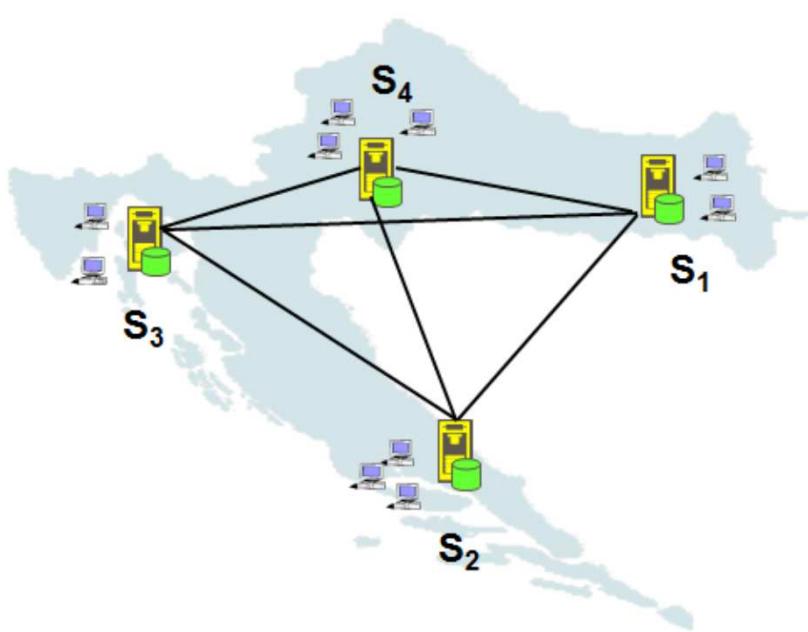


FAKULTET IZ DVADESETPRVOGA STOLJEĆA

# Distribuirana baza podataka i distribuirani SUBP

---

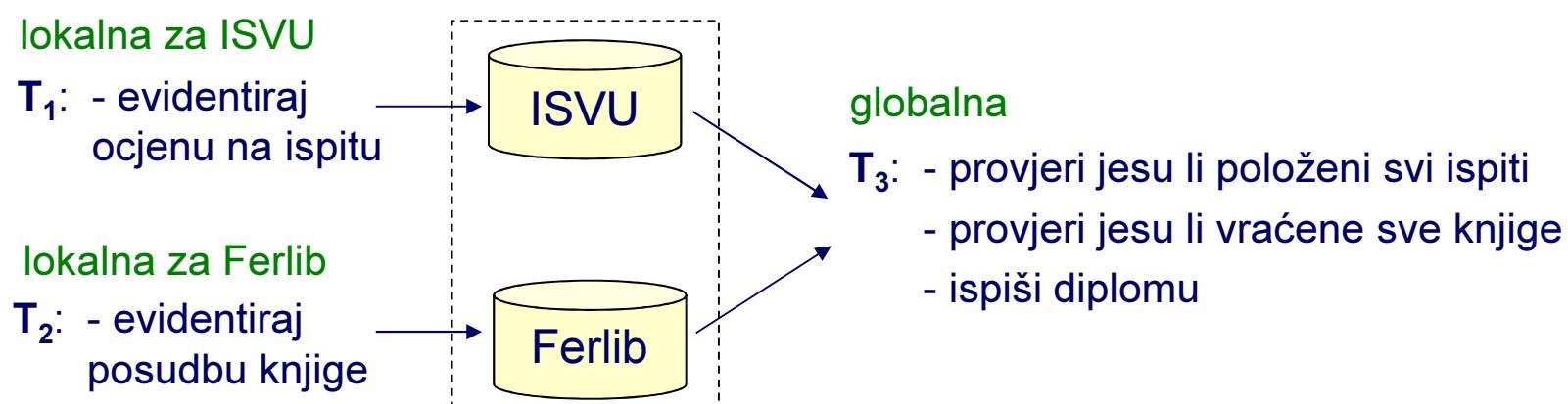
- **Distribuirana baza podataka** (DBP) je skup logički povezanih baza podataka razmještenih u različitim čvorovima računalne mreže (LAN, MAN, WAN)
- **Distribuirani sustav za upravljanje bazama podataka** (DSUBP) je programski sustav koji upravlja distribuiranom bazom podataka na takav način da je distribuiranost sustava transparentna prema korisnicima



- DSUBP obuhvaća n lokalnih SUBP-ova.
- Svaki lokalni SUBP, označen sa  $S_i$ , ( $i = 1, \dots, n$ ) predstavlja jedan čvor (site, node) distribuiranog sustava
- Svaki čvor  $S_i$  može direktno ili indirektno komunicirati sa svakim čvorom  $S_j$ , tj. postoji dvosmjerna veza između svaka dva čvora
- Čvorovi distribuiranog sustava za upravljanje bazama podataka ne dijele zajedničke fizičke komponente (disk, memorija, procesor)

# Distribuirana baza podataka i distribuirani SUBP

- Čvorovi su sposobni obavljati transakcije koje zahtijevaju isključivo lokalni pristup podacima (lokalne transakcije), ali također i transakcije koje zahtijevaju pristup podacima iz različitih čvorova (globalne transakcije)
  - čvorovi posjeduju određeni stupanj lokalne autonomije
- lokalne aplikacije (transakcije)
- globalne aplikacije (transakcije)
- baza podataka je distribuirana ako podržava barem jednu globalnu aplikaciju



# Prednosti DSUBP u odnosu na centralizirani SUBP

---

- Prilagodljivost organizacijskoj strukturi (organizacije su po prirodi "distribuirane")
  - decentralizacija javne uprave
  - elektronička trgovina
  - bankarski i telekomunikacijski sustavi
  - sustavi za upravljanje proizvodnjom, ...
- Snižavanje komunikacijskih troškova
  - približavanje podataka mjestu njihovog nastanka i korištenja
- Moguće povećanje dostupnosti podataka
  - kvar jednog ili više čvorova ne znači prestanak funkcioniranja cijelog sustava (u centraliziranom sustavu postoji *single-point-of-failure*)
- Moguće povećanje performansi
  - raspodjela opterećenja (*transaction load*) u više čvorova
  - paralelno obavljanje dijelova istog upita u nekoliko čvorova
- Omogućavanje modularnog razvoja sustava
  - dodavanje novih čvorova: održiv porast veličine BP, uvećanje procesorske snage

# Nedostaci DSUBP u odnosu na centralizirani SUBP

---

- Bitno veća složenost sustava
- Povećanje troškova, npr.
  - skuplja programska podrška
  - potreba angažiranja većeg broja administratora sustava
- Veći problemi sigurnosti
- Veći troškovi u osiguravanju integriteta podataka
- Nedostatak standarda
- Nedostatak iskustva
- Povećanje složenosti postupka projektiranja baze podataka
- Loša implementacija distribuirane baze podataka može uzrokovati
  - povećanje komunikacijskih troškova
  - smanjenje dostupnosti podataka
  - smanjenje performansi

Funkcionalnost i tehnike DSUBP, koje su rezultat mnogobrojnih provedenih istraživanja, nisu u cijelosti implementirane niti u jednom danas raspoloživom komercijalnom sustavu.

# Oblikovanje distribuirane baze podataka

---

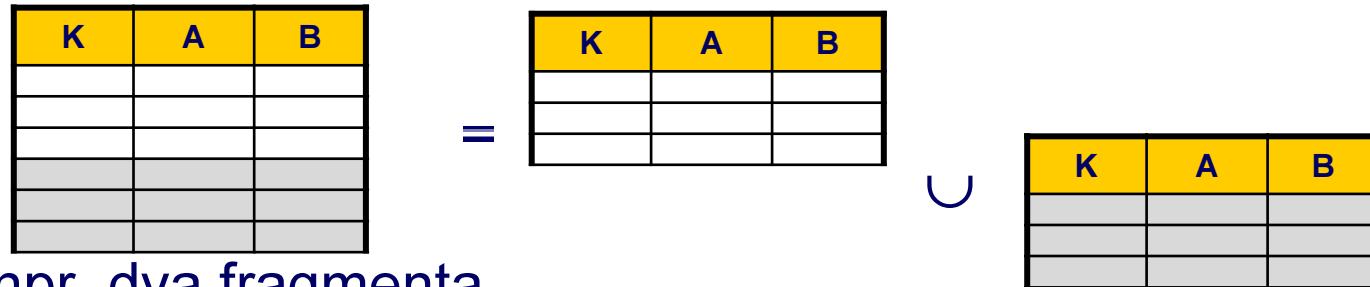
- Važan dio postupka oblikovanja distribuirane baze podataka jest određivanje načina distribucije podataka.
- podaci se smještaju u čvorove u kojima se najčešće koriste
  - minimalizira se mrežni promet

**oblikovanje distribucije = fragmentacija + alokacija**

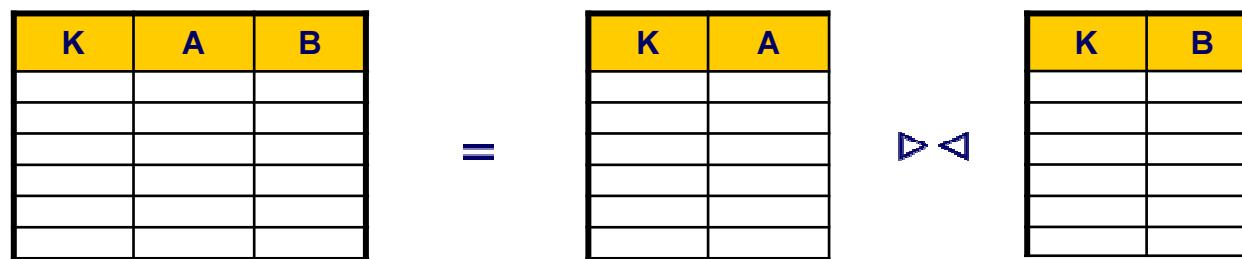
- Shema fragmentacije
  - podjela baze podataka u **disjunktni** skup fragmenata koji obuhvaćaju sve podatke u bazi podataka uz zadovoljenje pravila da se baza podataka može rekonstruirati iz tih fragmenata bez gubitka informacije
  - relacije mogu biti razdijeljene u fragmente horizontalno ili vertikalno (ili horizontalno i vertikalno)
- Shema alokacije
  - shema kojom se opisuje koji je fragment pridružen kojem čvoru distribuiranog sustava

# Fragmentacija

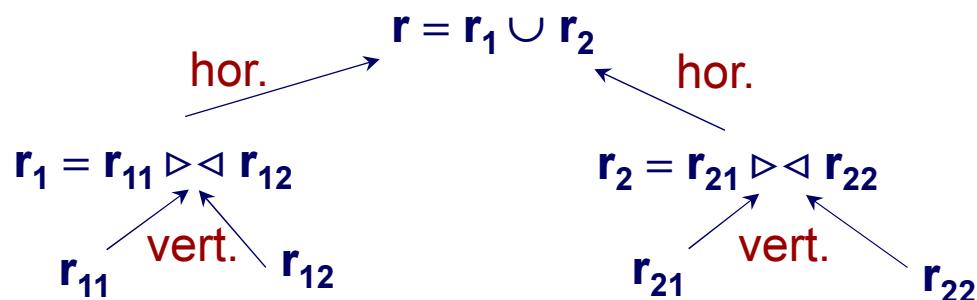
- Horizontalna, npr. dva fragmenta



- Vertikalna, npr. dva fragmenta



- Hibridna



# Alokacija

---

## Stupanj replikacije fragmenta

- broj čvorova u kojima je fragment alociran
  
- Svaki fragment mora biti alociran u barem jednom čvoru!
  
- **Particionirana (ili nereplicirana) baza podataka**
  - svaki od fragmenata alociran je u točno jednom čvoru, tj. stupanj replikacije svakog fragmenta = 1
- **Potpuno replicirana baza podataka**
  - svaki od fragmenata alociran je u svim čvorovima - svaki čvor sadrži repliku cijele baze podataka, tj. stupanj replikacije svakog fragmenta = n (broj čvorova u DSUBP)
- **Parcijalno replicirana baza podataka**
  - baza podataka nije niti particionirana niti potpuno replicirana (svaki od fragmenata može biti alociran u jednom, više ili svim čvorovima)

# Primjer alokacije

- Čvorovi  $S_1, S_2, S_3$
  - Fragmenti:
    - $\text{student}_1, \text{student}_2, \text{student}_3$
    - $\text{fakultet}_1$
- $\text{student}_1 = \sigma_{\text{sifFakultet} = 36}$  (student)
- $\text{student}_2 = \sigma_{\text{sifFakultet} = 102}$  (student)
- $\text{student}_3 = \sigma_{\text{sifFakultet} = 81}$  (student)

parcijalno replicirana baza podataka

$S_2$



- $\text{student}_2$
- $\text{fakultet}_1$

$S_1$



- $\text{student}_1$
- $\text{fakultet}_1$

$S_3$



- $\text{student}_3$
- $\text{fakultet}_1$

# **Transparentnost podataka**

---

- od korisnika DSUBP-a ne smije se zahtijevati znanje o tome gdje su podaci fizički smješteni niti na koji im način treba pristupati. Svojstvo se naziva transparentnost podataka i ima tri važna aspekta:

## **Transparentnost fragmentacije:**

- korisnici (ili aplikacije) ne trebaju voditi računa o načinu na koji je relacija fragmentirana

## **Transparentnost lokacije:**

- korisnici (ili aplikacije) ne trebaju znati u kojem je čvoru alociran koji fragment. Svakom podatku moraju moći pristupiti ukoliko im je poznat identifikator podatka

## **Transparentnost replikacije:**

- korisnici (ili aplikacije) ne trebaju voditi računa o postojanju kopija fragmenata (replikama)

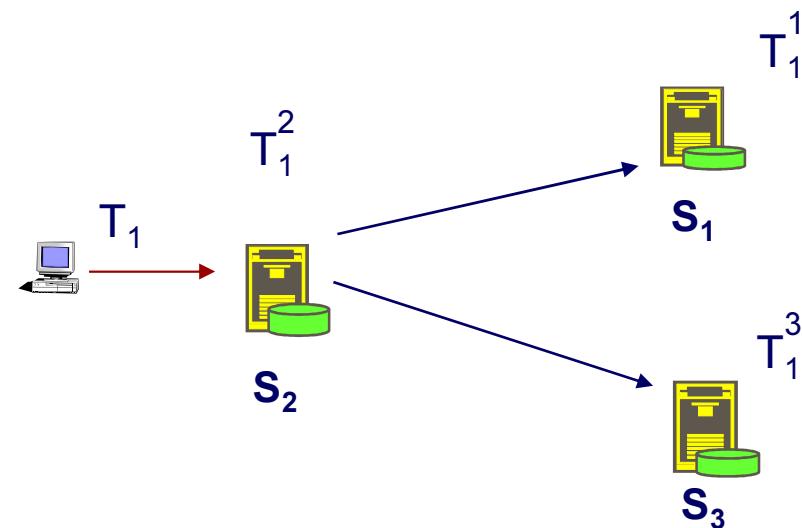
## **Transakcije u DSUBP-u**

# Globalne transakcije, subtransakcije i lokalne transakcije

Primjer: čvorovi DSUBP-a:  $S_1, S_2, S_3$

- Korisnik inicira globalnu transakciju  $T_1$  u čvoru  $S_2$
- Transakcija  $T_1$  se preslikava u skup subtransakcija:  $T_1^1, T_1^2, T_1^3$
- Svaka od subtransakcija sadrži operacije koje se obavljaju u pripadnom čvoru

- oznake:  $T_i^j$  subtransakcija globalne transakcije  $T_i$  koja se izvršava u čvoru  $S_j$



# Transakcija u DSUBP-u

---

- U svakom čvoru se nalazi zasebni, potpuno funkcionalan SUBP
- Transakcija se više ne može promatrati kao (samo) niz logički povezanih operacija koje se izvršavaju u jednom SUBP-u
- Globalna transakcija je skup subtransakcija koje koordinirano izvršavaju SUBP-ovi u više čvorova i pri tome prevode *distribuiranu* bazu podataka iz jednog u drugo konzistentno stanje
- **ACID?**
  - Svojstvo **Consistency** se relativno jednostavno ostvaruje uobičajenim mehanizmima
  - Svojstvo **Durability** osiguravaju SUBP-ovi u čvorovima
    - jer svaki čvor garantira izdržljivost svoje subtransakcije
  - Znatno teži problem: svojstva **Atomicity** i **Isolation**

# Atomarnost

---

- Atomarnost subtransakcija osiguravaju lokalni čvorovi
- Kako osigurati atomarnost globalne transakcije?
  - za vrijeme obavljanja globalne transakcije može se dogoditi prekid veze između jednog ili više čvorova ili se u jednom ili više čvorova može dogoditi kvar
- Atomarnost globalne transakcije znači da SUBP u svim čvorovima u kojima se izvršavaju pripadne subtransakcije moraju donijeti i provesti jednaku odluku o ishodu obavljanja transakcije: ili su sve subtransakcije globalne transakcije obavljene, ili nije niti jedna
- DSUBP provodi protokol potvrđivanja globalne transakcije
  - 2PC - *two-phase commit* (protokol dvofaznog potvrđivanja)

# Kvarovi u DSUBP-u

---

- Upravljanje kvarovima u DSUBP-u je složenije nego u centraliziranim sustavima
  - centralizirani sustav funkcionira u cijelosti ili ne funkcionira
  - dijelovi DSUBP-a mogu biti u kvaru, a dijelovi nastaviti s radom
- Osim kvarova koji su karakteristični za centralizirane sisteme (npr. pogreške programske podrške, sklopolja, uništenja diska), u DSUBP-u su moguće dodatne vrste kvarova:
  - prestanak rada jednog ili više čvorova
  - gubitak veze među čvorovima
  - gubitak poruka
  - podjela mreže: mreža je podijeljena (particionirana) kad je podijeljena u nekoliko podsustava koji međusobno ne mogu komunicirati. Naročiti problem: čvor  $S_i$  ne može utvrditi je li se dogodila podjela mreže ili je neki čvor  $S_j$  prestao raditi

---

# NoSQL baze podataka

# Što se promijenilo?

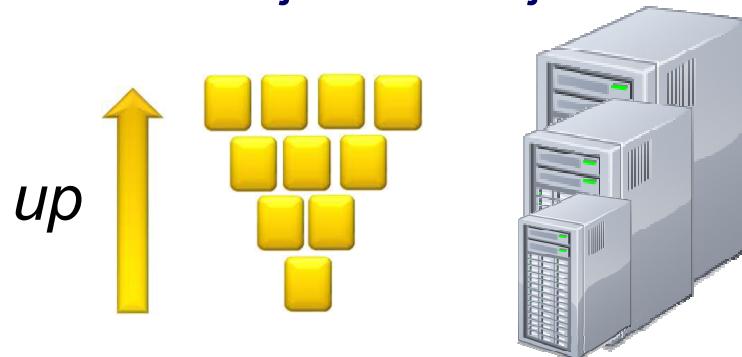
- Ogromne količine podataka
- Jeftin hardver, puno jeftinih računala
- **Uporaba isključivo relacijskih baze podataka nije najbolje rješenje za sve podatkovne probleme**

| IEC prefix |        | Representations |           |                                   |                               |       | Customary prefix |  |
|------------|--------|-----------------|-----------|-----------------------------------|-------------------------------|-------|------------------|--|
| Name       | Symbol | Base 2          | Base 1024 | Value                             | Base 10                       | Name  | Symbol           |  |
| kibi       | Ki     | $2^{10}$        | $1024^1$  | 1 024                             | $\approx 1.02 \times 10^3$    | kilo  | k, K             |  |
| mebi       | Mi     | $2^{20}$        | $1024^2$  | 1 048 576                         | $\approx 1.05 \times 10^6$    | mega  | M                |  |
| gibi       | Gi     | $2^{30}$        | $1024^3$  | 1 073 741 824                     | $\approx 1.07 \times 10^9$    | giga  | G                |  |
| tebi       | Ti     | $2^{40}$        | $1024^4$  | 1 099 511 627 776                 | $\approx 1.10 \times 10^{12}$ | tera  | T                |  |
| pebi       | Pi     | $2^{50}$        | $1024^5$  | 1 125 899 906 842 624             | $\approx 1.13 \times 10^{15}$ | peta  | P                |  |
| exbi       | Ei     | $2^{60}$        | $1024^6$  | 1 152 921 504 606 846 976         | $\approx 1.15 \times 10^{18}$ | exa   | E                |  |
| zebi       | Zi     | $2^{70}$        | $1024^7$  | 1 180 591 620 717 411 303 424     | $\approx 1.18 \times 10^{21}$ | zetta | Z                |  |
| yobi       | Yi     | $2^{80}$        | $1024^8$  | 1 208 925 819 614 629 174 706 176 | $\approx 1.21 \times 10^{24}$ | yotta | Y                |  |

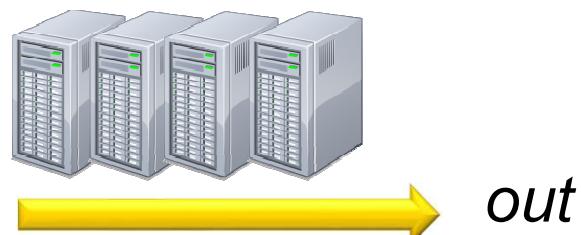
# Skalabilnost

---

- Mogućnost sustava da se nosi s rastućom količinom podataka
- Zadržavanje prihvatljivih performansi
- Vertikalna skalabilnost (engl. *scale up*):
  - Dodavanje resursa jednom čvoru (memorija, procesor, procesi, ...)



- Horizontalna skalabilnost (engl. *scale out*)
  - Dodavanje čvorova u sustav



Relacijski SUBP-ovi imaju problema s horizontalnom skalabilnošću

# NoSQL baze podataka

---

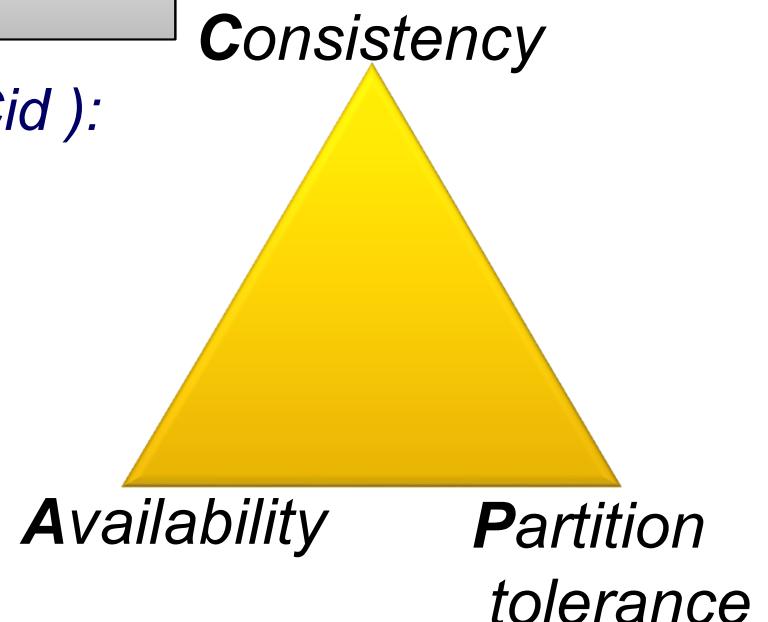
- Naziv potječe od (relacijskog!) SUBP-a koji je 1998. razvio *Carlo Strozzi*
- Naziv ponovo upotrijebljen 2009. godine na skupu „distribuiranih, ne-relacijskih baza podataka, otvorenog koda” kojeg je organizirao *Johan Oskarsson*
- No + SQL:
  - „SQL” se odnosi na „tradicionalne” relacijske SUBP-ove
  - Inicialno tumačeno kao „ne koristi SQL”, odnosno ne koristi relacijske SUBP-ove
  - Danas, *Not Only SQL* – „ne samo SQL”, rješenja koja nisu zasnovana isključivo na relacijskim SUBP-ovima
- Neformalna definicija (preuzeto s <http://nosql-database.org/>):  
Baze podataka novije generacije koje uglavnom imaju sljedeće značajke: ne-relacijske, distribuirane, otvorenog koda i horizontalno skalabilne... često posjeduju i dodatna svojstva: nema podatkovnog modela, lagana replikacija, jednostavan API, BASE (nisu ACID), rad s velikom količinom podataka i sl.

# CAP teorem

- Brewer 2000.: *Towards Robust Distributed Systems*, dokazan 2002.
- **Consistency, Availability, Partition tolerance**

U distribuiranim sustavima je moguće ostvariti samo dva od tri navedena svojstva.

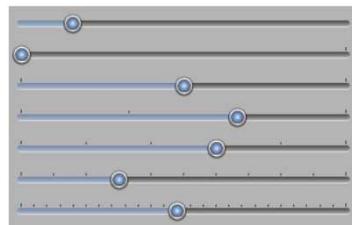
- **Consistency (dosljednost, Cap <> aCid ):**
  - Svaki odgovor poslan klijentu je točan
- **Availability (dostupnost)**
  - Svi zahtjevi uvijek odgovoreni
  - 1 čvor = 0 ili 100%
  - 2 čvora = 0%, 50% ili 100%
- **Partition tolerance (toleriranje particija)**
  - Rad sustava i u uvjetima kada nastanu izolirane skupine računala



# BASE

- Za mnoge primjene, dostupnost i toleriranje particija su važnije od stroge dosljednosti (npr. (velike) web aplikacije, tražilice, ...)
- BASE:
  - **Basically Available** – aplikacija je praktički uvijek dostupna (unatoč povremenim kvarovima)
  - **Soft-state** – ne mora uvijek biti dosljedan („mekano stanje”), sustav se stalno mijenja, protočan je
  - **Eventual consistency** – biti će, u konačnici, u nekom znanom stanju (izmjene će se, u konačnici, propagirati i svi će ih vidjeti)

| ACID                         |
|------------------------------|
| Jaka dosljednost             |
| Izolacija                    |
| Dostupnost?                  |
| Pesimistično (konzervativno) |
| ...                          |

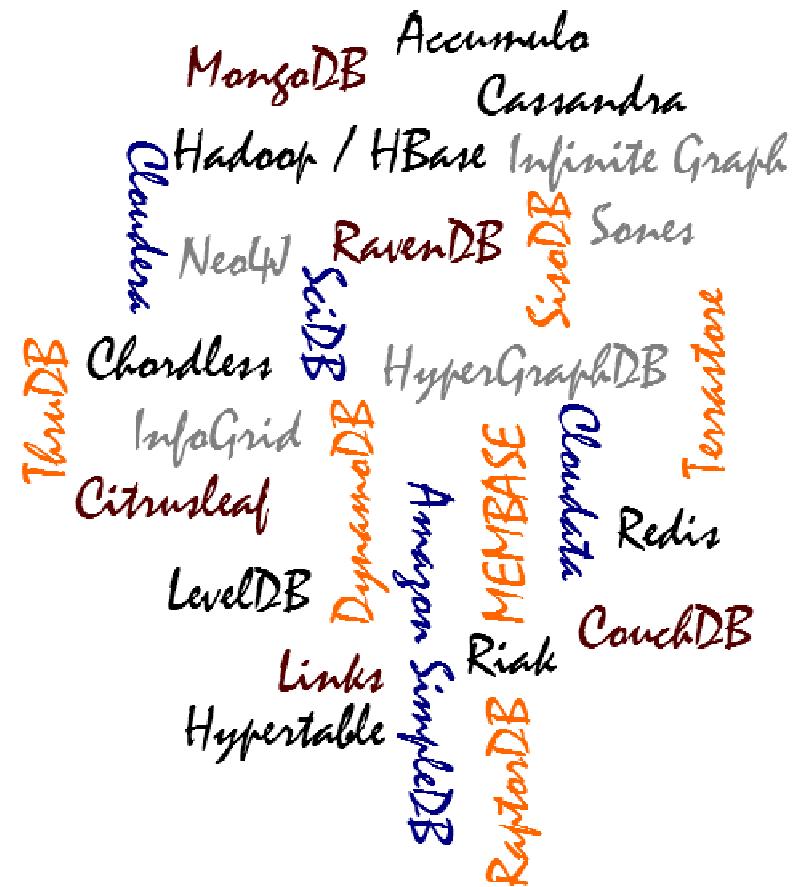


nije ili-ili

| BASE                                  |
|---------------------------------------|
| Slaba dosljednost (zastarjeli podaci) |
| Prvo dostupnost                       |
| Prihvatljivi približni odgovori       |
| Agresivno (optimistično)              |
| ...                                   |

# Vrste NoSQL baza podataka

- Neformalna podjela prema modelu podataka koji koriste:
  - Ključ-vrijednost
  - Dokumenti
  - Graf
  - Column-family
  - (Objektne baze podataka)



# Ključ-vrijednost

---

- Model podataka: (ključ, vrijednost) parovi
- Operacije:
  - Unos( $k, v$ )
  - Dohvat( $k$ )
  - Ažuriranje( $k, v$ )
  - Brisanje( $k$ )
  - Neki podražavaju određenu **strukturu** vrijednosti, **attribute** vrijednosti
  - Neki podržavaju dohvata na temelju raspona ključeva
- Podržavaju ogromne sustave, učinkoviti, vrlo skalabilni, otporni na pogreške:
  - Zаписи су расподијељени по систему с обзиром на ključ
  - Replikација
  - Трансакције од једног записа
  - Доследни у коначници
- Neki primjeri: *Riak, Amazon Dynamo...*

# Dokumenti

---

- Nalik ključ-vrijednost, pri čemu je vrijednost dokument
- Model podataka: (ključ, dokument)
- Dokument: JSON, BSON, XML, YAML, neki drugi polustrukturirani format, binarni podaci
- Osnovne operacije:
  - Unos( $k, d$ )
  - Dohvat( $k$ )
  - Ažuriranje( $k, d$ )
  - Brisanje( $k$ )
  - **Dohvat na temelju sadržaja dokumenta!** (nije standardizirano, nema upitnog jezika)
- Neki sustavi omogućuju i indeksiranje
- Neki primjeri: *CouchDB*, *MongoDB*, *SimpleDB*, ...

# Primjer: MongoDB dokumenti

---

Relacijska  
baza podataka:  
**relacija**

| Ime  | Prezime | DatRodj     | MjestoRodj |
|------|---------|-------------|------------|
| Ivan | Car     | 11.11.1971. |            |
| Iva  | Kralj   |             | Šibenik    |

MongoDB:  
**kolekcija**

```
{
 "_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
 "Ime" : "Ivan",
 "Prezime" : "Car",
 "DatRodj" : "11.11.1971."
},
{
 "_id" : ObjectId("4efa8d2b7d284dad101e4bc7"),
 "Ime" : "Iva",
 "Prezime" : "Kralj",
 "MjestoRodj" : "Šibenik"
}
```

# Primjer: MongoDB upiti

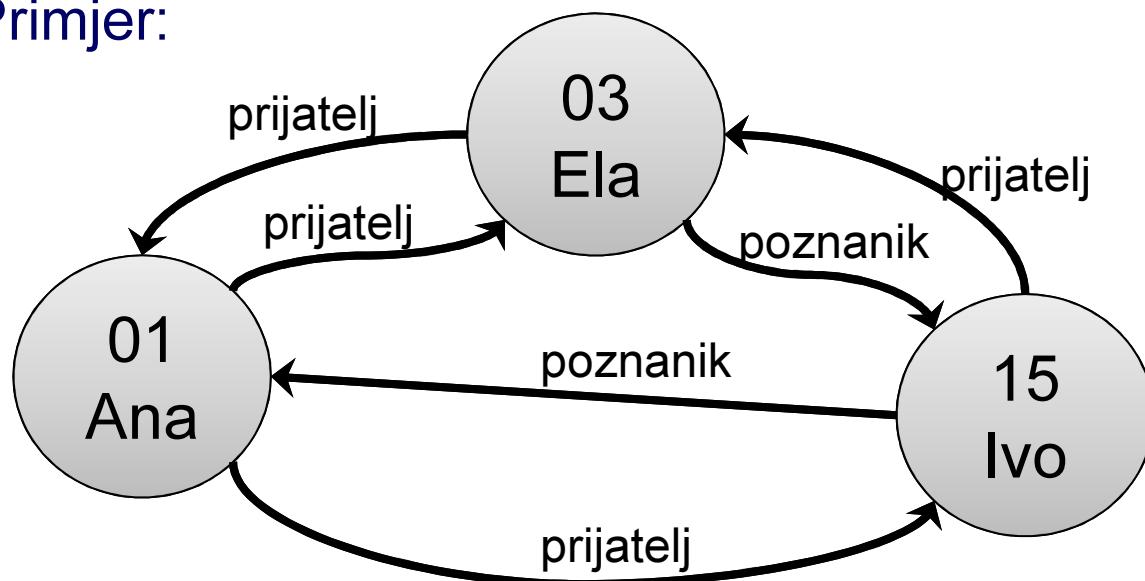
- Mongo upiti se formiraju kao JSON (BSON) objekti

| SQL                                                                          | MongoDB                                                                                                                |
|------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <pre>CREATE TABLE student(     mbr INT,     prezime CHAR(50) )</pre>         | Implicitno - unosom prvog dokumenta u kolekciju.<br>Može i eksplicitno:<br><code>db.createCollection("student")</code> |
| <pre>ALTER TABLE student ADD...</pre>                                        | Implicitno, svaki dokument je moguće promijeniti bez mijenjanja sheme                                                  |
| <pre>INSERT INTO student (     100,     'Šostaković');</pre>                 | <code>db.student.insert(     {mbr:100, prezime: 'Šostaković'} )</code>                                                 |
| <pre>SELECT * FROM student;</pre>                                            | <code>db.student.find();</code>                                                                                        |
| <pre>SELECT prezime   FROM student  WHERE mbr = 200  ORDER BY prezime;</pre> | <code>db.student.find(     {mbr:100},     {prezime:1} ).sort({prezime:1});</code>                                      |
| <pre>UPDATE student SET prezime = 'Shostakovich'  WHERE mbr = 100;</pre>     | <code>db.student.update(     { mbr: 100 },     {\$set : { prezime : 'Shostakovich' } } );</code>                       |
| <pre>DELETE FROM student WHERE mbr = 100;</pre>                              | <code>db.student.remove(     { mbr: 100 } );</code>                                                                    |

# Graf

---

- Model podataka: čvorovi i bridovi:
  - Čvorovi mogu imati svojstva (ID, ...)
  - Bridovi mogu imati oznake ili uloge
- Sučelja i upitni jezici nisu standardizirani
- Primjer:



- Neki primjeri: *Neo4j, GraphDB, DEX, FlockDB, InfoGrid, OrientDB, Pregel, ...*

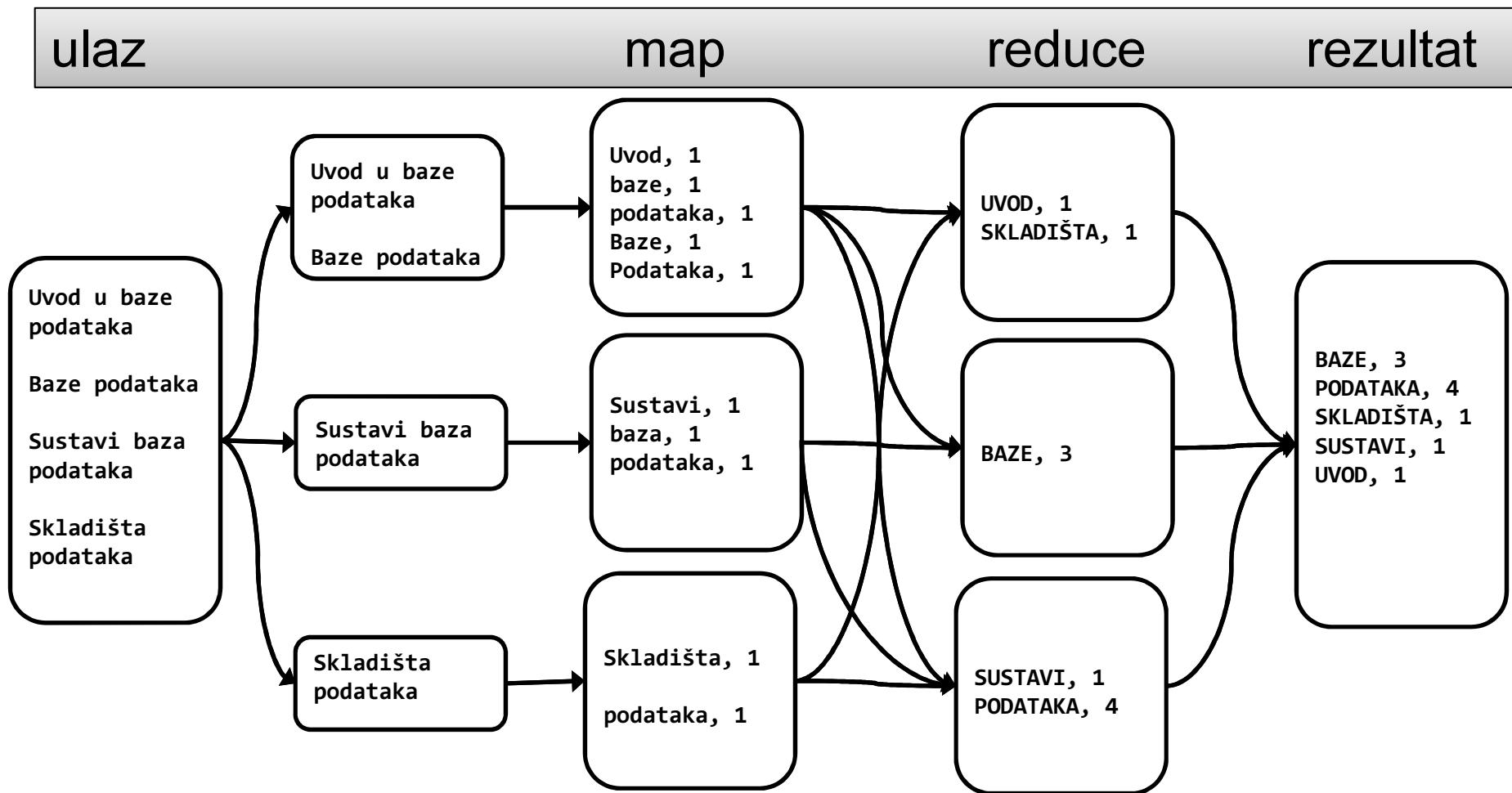
# MapReduce

---

- Mnoge NoSQL baze podataka, bez obzira na tip, omogućuju MapReduce algoritam
- Razvio 2004. i patentirao Google, npr. *Google File System*
-  **hadoop** – (među ostalim i) implementacija *MapReduce*
- Nema modela podataka, podaci pohranjeni u datotekama
- Pogodan samo za određenu vrstu (paralelnih) problema
  - npr. pretraživanje velike količine teksta, izgradnja indeksa riječi, brojanje pristupa web stranicama, itd.
- Podaci su organizirani u (ključ, vrijednost) parove
- Dvije glavne faze:
  - Map: `map(k, v) → 0 ili više (k,v) parova (lista)`
  - Reduce: `reduce(k, lista vrijednosti) → lista vrijednosti`
- Korisnik mora implementirati `map()` i `reduce()` funkcije
- Sustav orkestrira paralelno obavljanje, oporavak od pogrešaka, ...

# MapReduce pojednostavljeni primjer

- Prebrojati broj pojavljivanja riječi



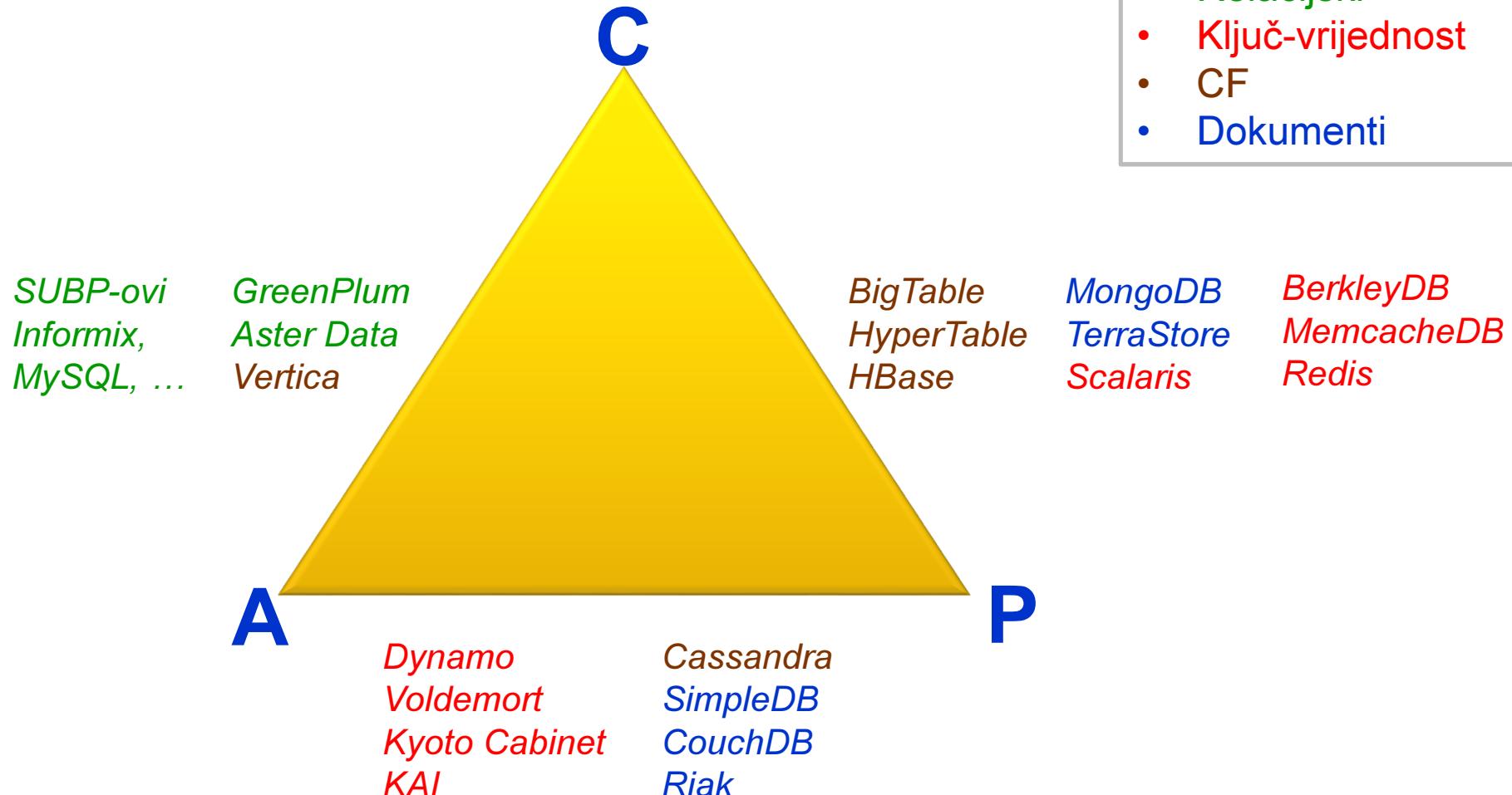
# MapReduce/Hadoop

---

- Premošćuje jaz između specifikacije paralelnog algoritma na visokoj razini i same implementacije
- Obično se implementacija bazira na specijaliziranom datotečnom sustavu optimiranom za distribuirani pristup (GFS, HDFS)
- Najpoznatiji softver: Hadoop = HDFS + M/R
- Projekti koji nadograđuju osnovnu funkcionalnost:
  - **Pig**  
[http://en.wikipedia.org/wiki/Pig\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Pig_(programming_language)): *Pig is a high-level platform for creating MapReduce programs used with Hadoop. The language for this platform is called Pig Latin.*
  - **Hive** (donosi shemu, HiveQL - upitni jezik nalik SQL-u)  
[http://en.wikipedia.org/wiki/Apache\\_Hive](http://en.wikipedia.org/wiki/Apache_Hive) : *Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.*

# Vodič za NoSQL sustave

- <http://blog.nahurst.com/visual-guide-to-nosql-systems>



## NoSQL, zaključno:

---

- Nisu zamjena za relacijske sustave, već namjenski softver za određene probleme
- Najčešće: povećanje dostupnosti (*availability*) i učinkovitosti na račun dosljednosti (*consistency*)
- BASE a ne ACID
- Velike količine podataka
- Fleksibilna shema podataka
- Horizontalna skalabilnost
- Nije zrela tehnologija, nedostatak standarda
- Najčešće besplatno/otvorenog koda
- Relativno lako za uspostaviti (za testiranje)