

## BAZE PODATAKA

Predavač: Dalibor Bužić

## Pravila kolegija

- predavanja i auditorne vježbe: obavezno prisustvo 70% za redovne studente
- laboratorijske vježbe 100% prisustvo i položen izlazni test
- 2 kolokvija – prolaz minimalno 50% na svakom
  - uvjetni prolaz: minimalno 40% na pojedinom → ukupno mora biti 50%
    - npr. 1. kol = 45%, 2. kol = 55% → prolaz
    - npr. 1. kol = 45%, 2. kol = 50% → pad
  - najviše jedan nepoloženi kolokvij može se polagati na dva zimski ispitna roka
- na jesenskim rokovima položen kolokvij NE VRIJEDI → piše se cijeli ispit

## Pravila kolegija

- predavanja i auditorne vježbe: obavezno prisustvo 70% za redovne studente
- laboratorijske vježbe 100% prisustvo i položen izlazni test
- maksimalno 3 vježbe u nadoknadi
- dozvoljeni dolasci na termine drugih grupa labosa u **istom tjednu** u slučaju spriječenosti

## Pravila kolegija

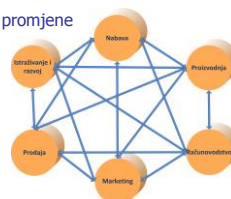
- studenti koji su dva puta uspješno odslušali kolegij, mogu najkasnije do kraja drugog tjedna nastave zatražiti oslobađanje od vježbi → poslati e-mail predavaču
- predavanja su obavezna bez obzira na to je li student već odslušao predavanja
- konačnu ocjenu sačinjava kolokvij teorije (ili rezultat ispita) 80%, izlazni testovi laboratorijskih vježbi 13% te izlazni testovi predavanja 7%

## Prije baza podataka

- zapisivanje podataka → glinene pločice, papirus, papir, bušena kartica, bušena vrpca
- magnetski disk – doba masovnih medija
- datoteke
  - datoteka načelno predstavlja skup izoliranih podataka
  - događaji u realnom svijetu, a posljedično i podaci o njima, ostvaruju međusobne veze
  - nemogućnost izražavanja veza među podacima dovodi do velikog problema - redundance

## Prije baza podataka

- dodatni problemi nastaju ako se razvoju aplikacija u tvrtki pristupa nekoordinirano
  - preklapanje podataka
  - potreba za razmjeno podataka
  - visoki troškovi razvoja
  - upitna sigurnost
  - promjene aplikacija ne prate poslovne promjene
  - otežana izrada izvještaja

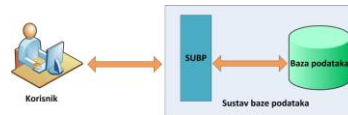


## Što je baza podataka?

- Baza podataka je organizirani skup međusobno povezanih podataka pohranjenih u vanjskoj memoriji računala bez štetne redundance.
- Podaci se spremaju neovisno o aplikacijama koje ih koriste (pretražuju, uspoređuju, sortiraju, dodaju, mijenjaju, brišu...), a sva manipulacija podacima se odvija kroz kontrolirano zajedničko sučelje
- Sustav za upravljanje bazom podataka (SUBP, eng. Database Management System, DBMS) je složeni programski sustav koji obavlja sve operacije nad bazom podataka, a služi kao sučelje između korisnika i zapisa baze podataka na disku.

## Što je baza podataka?

- Bilo kakav pristup podacima mora proći putem SUBP-a.
- Baza podataka i sustav za upravljanje bazom podataka čine sustav baze podataka



## Sustav za upravljanje bazama podataka – zadaci:

- opis i manipulacija podacima pomoću jednog ili više posebnih jezika:
  - jezik za opis podataka (eng. Data Definition Language, skr. DDL) služi administratoru baze za definiranje interne strukture baze → njime se logički definiraju podaci i veze među podacima

```
CREATE TABLE Predmet (
  PredmetID int PRIMARY KEY,
  NazivPredmeta varchar(50) NOT NULL)
```
  - jezik za manipulaciju podacima (eng. Data Manipulation Language, skr. DML) služi za tri osnovne operacije izmjene (dodavanje, promjenu i brisanje podatka) te za pretraživanje

```
INSERT INTO Predmet VALUES (22, 'Projektiranje
baza podataka');

SELECT ime, prezime FROM Student WHERE
godinaUpisa=2014;
```

Kod relacijskih baza podataka ova dva jezika su objedinjena u jedan: SQL (eng. Structured Query Language)

## Sustav za upravljanje bazama podataka – zadaci:

- visok nivo sučelja prema korisniku, što podrazumijeva skrivanje fizičke implementacije, a znači da aplikacije moraju biti neovisne o memorijskoj lokaciji i fizičkoj strukturi podataka
- efikasno izvođenje operacija nad podacima, što uključuje optimizaciju upita
- zaštitu integriteta – to znači da SUBP mora odbiti izvršiti promjenu nad podacima koja bi narušila konzistentnost baze
- definiranje dozvola korisnicima koji će moći pristupiti točno određenim dijelovima baze čime se štiti sigurnost baze.
- istovremeni rad većeg broja korisnika bez štetnog međudjelovanja
- obnovu podataka u slučaju djelomičnog ili potpunog razrušenja baze pomoću sigurnosnih kopija

## Baza podataka

- Svaki poslovni sustav ima svoj informacijski sustav.
- U središtu (informatiziranog) informacijskog sustava je baza podataka.
- Arhitektura baze podataka sastoji se od tri razine: vanjske, konceptualne i unutarnje, a opisuju se shemama.
  - Unutarnja (interna) shema odnosi se na fizičku strukturu pohranjivanja podataka, kao i na načine pristupa tim podacima. Ona ovisi o konkretnom SUBP-u.
  - Konceptualna (logička) shema odnosi se na opis tipova entiteta i tipova veza, atributa te raznih ograničenja koja vrijede u sustavu. Može se prikazati npr. EV modelom ili relacijskim modelom.
  - Vanjska (eksterna) shema odnosi se podatke u bazi kako ih vidi korisnik (ustvari različite grupe korisnika). Za razliku od unutarnje i konceptualne sheme, vanjskih shema može biti više jer se za različite grupe korisnika stvaraju različite aplikacije.

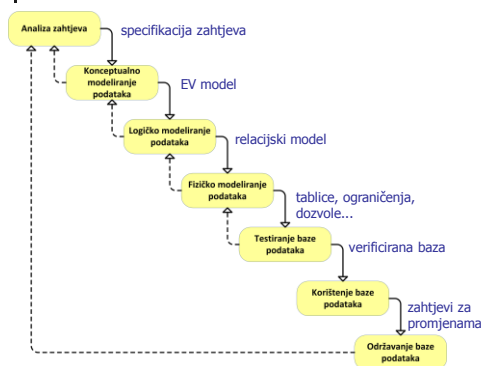
## Povijesni razvoj

- prve baze podataka početkom 60 – tih godina
- danas dominiraju relacijske baze podataka
  - 1970. E. F. Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol. 13, No. 6, lipanj 1970.
  - velika većina istraživanja od strane IBM-a
  - ranih 70-ih godina razvijen SQL (Structured Query Language), IBM
  - 1979. prva komercijalna relacijska baza podataka (Oracle)
  - 1982. IBM – SQL/Data System
  - tijekom daljnjih godina velik broj komercijalnih proizvoda
  - 1982. SQL postaje standard (ANSI)
  - 1987. SQL postaje ISO standard

### Dio proizvođača na području relacijskih baza podataka i njihovi proizvodi

Proizvođač	Proizvod
Microsoft	SQL Server
Oracle	Oracle
Sysbase	Sysbase SQL Server
IBM	DB2
Informix	Informix-SQL
Interbase (bivši Borland)	Interbase
T.c.X	mySQL

### Životni ciklus baze podataka



## MODELIRANJE PODATAKA – EV MODEL

Predavač: Dalibor Bužić

## Model podataka

- događaji u organizacijskom sustavu (npr. izdavanje računa kupcu, polaganje ispita, upis novog člana u planinarsko društvo) za sobom povlače nastajanje podataka o tim događajima
- događaji (a posljedično i podaci) ostvaruju složene međusobne odnose koji mogu biti slučajni, uvjetovani, neovisni, ovisni, isključivi...
- potrebno je osmisлити optimalan način na koji će se podaci iz stvarnog svijeta pohraniti u bazi podataka bez gubljenja odnosa koji postoje u stvarnosti
- Zato se podaci iz realnog svijeta modeliraju

## Model podataka

- model podataka definira način oblikovanja te načine rukovanja
- model podataka je skup definicija koje:
  - određuju kako valja oblikovati izapisivati podatke
  - definiraju operacije koje se mogu izvoditi s podacima
- Svaki model podataka čine tri dijela:
  - struktura – skup elemenata koji prikazuju podatke i njihove odnose
  - operacije – omogućavaju izmjenu stanja (vrijednosti) podataka čime omogućavaju dinamiku podataka
  - ograničenja – skup valjanih stanja (npr. dopuštene vrijednosti za ocjenu ispita)

## Model podataka

- Modeliranje podataka se temelji na procesima apstrakcije i klasifikacije
- Apstrakcija je jedan od temeljnih misaonih postupaka kojim se odbacuje ono što je sporedno, posebno i slučajno radi onoga što je opće, zakonito i bitno.
- Klasifikacija podataka je logički postupak obrnut od apstrakcije – to je postupak raščlambe podataka logički višeg reda u niz podataka logički nižeg reda nekih zajedničkih, ali i nekih bitno različitih svojstava

## EV model podataka

- Model Entiteti-Veze (EV model) je grafički jezik za predstavljanje struktura podataka, autora Petera Chena
- EV model je konceptualni model visoke razine koji projektantu služi za modeliranje podataka pomoću grafičkih simbola.
- Neovisan je o SUBP-u i sklopovlju.
- Gotovo nezaobilazni jezik za konceptualno modeliranje podataka
  - intuitivan i jednostavan za crtanje
- Osnovni koncepti na kojima se ovaj model zasniva su entiteti, veze i atributi.

## Entitet

- Entitet je nešto što postoji u stvarnom svijetu, ima značajke koje ga opisuju, te se o njemu žele pohraniti podaci
  - Živo biće – npr. osoba u različitim ulogama (student, sportaš, član biblioteke, rukovoditelj, planinar), životinja (mačka, pas, ptica itd.) ili biljka (oskoruša, šljivica itd.)
  - objekt – npr. bicikl, artikel, proizvod
  - dokument – npr. račun, prijavnica, putovnica
  - događaj ili pojava – npr. skijaška utrka, servisiranje perilice, zaposlenje
  - tvrtka, organizacija, ustanova – npr. Tehnika d.d., ribički klub, VSITE
- Skup entiteta iste vrste naziva se **tip entiteta**.

RADNIK

### Tip entiteta

- Tipovi entiteta se mogu podijeliti u dvije kategorije: tipovi jakog i tipovi slabog entiteta.
- Tip jakog entiteta je takav tip entiteta koji ima vlastiti primarni ključ te može postojati sam za sebe, neovisno o drugim tipovima entiteta (npr. RADNIK, FAKULTET, PROIZVOĐAČ, ARTIKL...)
- Tip slabog entiteta je takav tip entiteta koji na neki način ovisi o drugom tipu entiteta (ne nužno jakom). Ova ovisnost može biti egzistencijska ili identifikacijska.
  - Identifikacijska ovisnost znači da identifikator slabog tipa entiteta ovisi o identifikatoru jakog tipa entiteta.
  - Egzistencijska ovisnost znači da nije moguća egzistencija slabog entiteta bez postojanja tipa jakog entiteta.

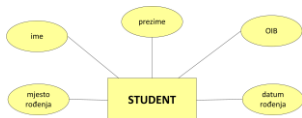
### Tip entiteta

- Tip entiteta koji je ovisan identifikacijski o jakom entitetu, ujedno je i egzistencijski ovisan.
- Međutim, egzistencijska ovisnost ne znači automatski i identifikacijsku ovisnost!
- Primjer tipa slabog entiteta je STAVKA koja ovisi o RAČUNU – ne bi imalo smisla pohranjivati informacije o količini i prodajnim cijenama artikala (koji predstavljaju stavke), a da se ne zna kojem računu pripadaju.
- Tip slabog entiteta prikazuje se u obliku dvostrukog pravokutnika

STAVKA

### Atribut

- Atribut je karakteristika (svojstvo) entiteta.
- Njime se identifikira, klasificira, kvantificira odnosno kvalificira stanje entiteta.
- Primjerice, atributi tipa entiteta OSOBA su OIB, ime, prezime, datum rođenja i mjesto rođenja.
- Atribut se na EV dijagramu prikazuje u ovalnom obliku unutar kojeg se opisuje naziv atributa
- Odabrati relevantne attribute!



### Atribut

- Složeni atribut?
- atribut ima vrijednost koju poprima iz određene domene
  - npr. atribut Ocjena ima vrijednost 4
  - domena je skup mogućih (dozvoljenih) vrijednosti, za ocjenu to je {1,2,3,4,5}
- kardinalnost atributa u entitetu je uređeni par koji govori koliko najmanje i koliko najviše atributa može pojedini entitet imati:
 
$$\text{kard}(A, E) = (\min \text{kard}(A, E), \max \text{kard}(A, E))$$
- određivanje kardinalnosti za atribut datum rođenja entiteta RADNIK:
  - $\min \text{kard}(\text{datum rođenja}, \text{RADNIK}) = 1$  jer pojedini radnik može imati najmanje jedan datum rođenja;
  - $\max \text{kard}(\text{datum rođenja}, \text{RADNIK}) = 1$  jer pojedini radnik može imati najviše jedan datum rođenja;
  - stoga je  **$\text{kard}(\text{datum rođenja}, \text{RADNIK}) = (1, 1)$** .

### Atribut

- Pojedina kardinalnost atributa može poprimiti vrijednosti 0, 1 ili n (n znači više od jedan, ili kraće: više), s time da maksimalna kardinalnost može biti 1 ili n.
- Minimalna kardinalnost je 0 ako atribut nije obavezan za unos.
- Minimalna kardinalnost je 1 ako je podatak obavezan za unos.
- Maksimalna kardinalnost atributa je 1 ako je atribut obavezan za unos
- Maksimalna kardinalnost atributa je n ako atribut može imati više vrijednosti

### Atribut

- određivanje kardinalnosti za atribut hobi entiteta RADNIK:
  - $\min \text{kard}(\text{hobi}, \text{RADNIK}) = 0$  jer pojedini radnik ne mora imati nikakav hobi;
  - $\max \text{kard}(\text{hobi}, \text{RADNIK}) = n$  jer pojedini radnik može imati više hobija (npr. plivanje, šah, planinarenje);
  - stoga je  **$\text{kard}(\text{hobi}, \text{RADNIK}) = (0, n)$  → više-vrijednosni atribut**

## Atribut

- Postoje tri vrste atributa:
  - identifikacijski
  - opisni
  - izvedeni
- Identifikacijski atribut ili identifikator (koristi se i naziv kandidat za ključ) je atribut koji jednoznačno određuje pojedini entitet.
- Ne postoje dva entiteta s istom vrijednošću identifikatora. Primjerice, ne postoje dvije osobe s istim OIB-om.
- Opisni atribut je atribut kojim se opisuju kvantitativna ili kvalitativna svojstva entiteta.
- Njegove vrijednosti se mogu ponekad mijenjati tijekom vremena (primjerice, broj telefona, adresa stanovanja, broj putovnice) budući da se i u stvarnosti ta svojstva mijenjaju.

## Atribut

- Izvedeni atribut nastaje aritmetičkim ili logičkim operacijama iz vrijednosti drugih atributa.
  - Primjer izvedenog atributa je starost u godinama koja se dobiva tako da se od današnje godine oduzme godina rođenja (ona se pak dobije izdvajanjem iz datuma rođenja).
- Pojedini tip entiteta može imati više identifikatora.
  - Primjerice, tip entiteta STUDENT s atributima OIB, ime, prezime, datum rođenja, broj indeksa ima dva identifikatora: OIB i broj indeksa.
- Svaki tip entiteta mora imati barem jedan identifikator koji će jednoznačno određivati svaki pojedini entitet.
- Ako tip entiteta nema identifikator u skupu atributa koji su dobiveni selekcijom relevantnih atributa, onda mu se dodaje jedan novi (umjetni) atribut koji će osiguravati jedinstvenost svakog pojedinog entiteta.

## Identifikator i primarni ključ

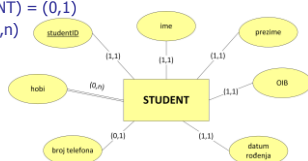
- Dodavanje novog identifikacijskog atributa je u praksi učestalo čak i ako identifikator već postoji.
  - većina sustava dodjeljuje svoje interne šifre entitetima, a šifra je ništa drugo nego identifikator.
    - Primjerice, biblioteka dodjeljuje identifikator članski broj prilikom upisa novog člana, iako on već ima jedan identifikator – OIB.
    - Tako dodijeljeni identifikator najčešće dobiva ime tako što se iza naziva tipa entiteta doda nastavak ID (npr. radnikID, studentID, proizvodID, računID)
- Za svaki tip entiteta jedan od identifikatora se proglašava primarnim ključem (eng. primary key).
- Primarni ključ (PK) tipa entiteta je skup atributa koji jedinstveno identificira (određuje) svaki pojedini entitet.
  - Skup atributa često se sastoji od samo jednog atributa, naime primarni ključ treba biti što jednostavniji (treba se sastojati od najmanjeg broja atributa koji zadovoljavaju načelo jedinstvenosti).

## Dijagram entiteta

- EV metoda podatke prikazuje grafički – u obliku dijagrama.
- Dosad izloženi koncepti mogu se prikazati dijagramom entiteta koji je dio dijagrama entiteti-veze.
- Pravila crtanja dijagrama entiteta su:
  - Tip entiteta se crta u obliku pravokutnika u koji se velikim štampanim slovima upisuje naziv tipa entiteta u jednini. Ako se želi prikazati (odnosno naglasiti) tip slabog entiteta, on se tada crta u obliku dvostrukog pravokutnika.
  - Svaki pojedini atribut se crta u obliku ovala unutar kojeg se upisuje naziv atributa.
  - Atribut se crtom povezuje s tipom entiteta kojem pripada. Pored crte se u obliku uređenog para upisuje kardinalnost atributa.
  - Viševrijednosni atribut (kardinalnosti (0,n)) se povezuje dvostrukom crtom s tipom entiteta.
  - Atribut koji je primarni ključ se podcrtava.

## Dijagram entiteta

- Nacrtati dijagram entiteta za tip entiteta STUDENT koji ima attribute studentID, ime, prezime, OIB, datum rođenja, broj telefona, hobi sa slijedećim kardinalnostima:
  - kard(studentID, STUDENT) = (1,1)
  - kard(ime, STUDENT) = (1,1)
  - kard(prezime, STUDENT) = (1,1)
  - kard(OIB, STUDENT) = (1,1)
  - kard(datum rođenja, STUDENT) = (1,1)
  - kard(broj telefona, STUDENT) = (0,1)
  - kard(hobi, STUDENT) = (0,n)



## Veze

- Entiteti su međusobno povezani logičkim vezama, npr:
  - Student predaje prijavnicu
  - Dobavljač dostavlja proizvod
  - Prodavač izdaje račun kupcu
- Veze se imenuju glagolima ili glagolskim imenicama (npr. naručuje, dostavlja, polaže, živi, je u braku).
- Tip veze je skup pojedinačnih veza.
- Na EV dijagramu se ne crtaju pojedinačne veze, već tip veze i to u obliku romba unutar kojeg se malim slovima upisuje naziv veze



## Veze

- Entiteti su međusobno povezani logičkim vezama, npr:
  - Student predaje prijavnicu
  - Dobavljač dostavlja proizvod
  - Prodavač izdaje račun kupcu
- Veze se imenuju glagolima ili glagolskim imenicama (npr. naručuje, dostavlja, polaže, živi, je u braku).
- Tip veze je skup pojedinačnih veza.
- Na EV dijagramu se ne crtaju pojedinačne veze, već tip veze i to u obliku romba unutar kojeg se malim slovima upisuje naziv veze



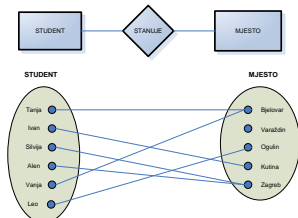
## Veze

Veza između entitetima može biti

- ⇒ 1–naprama–1 (ili 1:1),
- ⇒ 1–naprama–mnogo (ili 1:N)
- ⇒ mnogo–naprama–mnogo (ili N:M)

## Veze (relationships)

- entiteti su međusobno povezani logičkim vezama.
- STUDENT i MJESTO** svaki student živi u nekom mjestu
- entitet **MJESTO** ima **atribute** MjestoID, Naziv, PoštanskiBroj
- u jednom mjestu može živjeti veći broj studenata, između entiteta postoji **logička veza**



## Veze (relationships)

**Kardinalitet entiteta** u relaciji definiran je brojem veza pojedinog entiteta s entitetima s kojim je povezan.

Primjer:

relacijska veza tipa entiteta **STUDENT i MJESTO**.

**STUDENT**

student može živjeti u samo jednom mjestu

**MJESTO**

U jednom mjestu može živjeti više studenata, ali ne mora niti jedan.

Vrijedi

**min card** (STUDENT, STANUJE)=1

**max card** (STUDENT, STANUJE)=1

Odnosno

**card** (STUDENT, STANUJE)=(1,1)

**min card** (MJESTO, STANUJE)=0

**max card** (MJESTO, STANUJE)=n

**card** (MJESTO, STANUJE)=(0,n)

## Veze (relationships)

relacija **1xn** odnosno **jedan prema više (one-to-many)**.

**def. 6** Ako dva entiteta **E i F** ostvaruju relaciju **R**, pri čemu vrijedi

**min card** (E,R)=0

**max card** (E,R)=n

**card** (E,R)=(0,n)

**min card** (F,R)=1

**max card** (F,R)=1

**card** (F,R)=(1,1)

takva relacija je **tipa 1xn (jedan-prema-više)**.

**Napomena:**

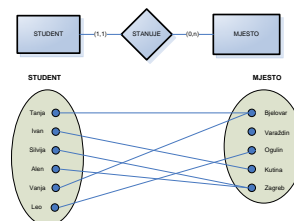
Pri određivanju **tipa relacije**, ključan je **maksimalni kardinalitet entiteta u relaciji**.

To znači da su entiteti **E i F** povezani relacijom **R** sa definiranim kardinalitetima,

**card**(E,R)=(0,n) i **card**(F,R)=(0,1)

također u vezi jedan prema više.

## Veze (relationships)



## Veze (relationships)

relacija **1x1** odnosno **jedan prema jedan (one-to-one)**.

Ako dva tipa entiteta **E** i **F** ostvaruju relaciju **R** pri čemu vrijedi

$\min \text{ card } (E, R) = 1$        $\min \text{ card } (F, R) = 1$   
 $\max \text{ card } (E, R) = 1$        $\max \text{ card } (F, R) = 1$

takva relacija je **tipa 1x1 (jedan-na-jedan)**.

Svaki entitet iz tipa entiteta **E** može biti povezan samo sa jednim entitetom iz tipa entiteta **F**, ali i ne mora.

Isto vrijedi za entitete iz tipa entiteta **F**.

Primjer:

Svaki **student** može imati samo jednu **osobnu iskaznicu**.

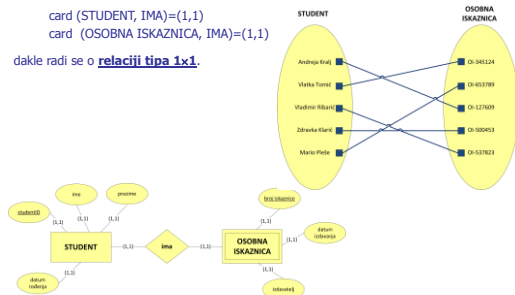
Svaka **osobna iskaznica** može pripadati samo jednom **studentu**.

## Veze (relationships)

Za relaciju **IMA** koja povezuje entitete **STUDENT** i **OSOBNJA ISKAZNICA** vrijedi

$\text{card } (\text{STUDENT}, \text{IMA}) = (1,1)$   
 $\text{card } (\text{OSOBNJA ISKAZNICA}, \text{IMA}) = (1,1)$

dakle radi se o **relaciji tipa 1x1**.



## Veze (relationships)

relacija **nxm** odnosno **više prema više (many-to-many)**.

def. Ako dva tipa entiteta **E** i **F** ostvaruju relaciju **R** pri čemu vrijedi

$\min \text{ card } (E, R) = 0$        $\min \text{ card } (F, R) = 0$   
 $\max \text{ card } (E, R) = n$        $\max \text{ card } (F, R) = m$

takva relacija je **tipa nxn (više-prema-više)**.

Svaki entitet iz tipa entiteta **E** može biti povezan sa više entiteta iz tipa entiteta **F**, ali i ne mora.

Isto vrijedi za entitete iz tipa entiteta **F**.

Primjer:

Da bi se definirao proces polaganja ispita potrebno je definirati novi tip entiteta **KOLEGIJ**.

Tip entiteta **KOLEGIJ** je skup svih kolegija koji se predaju.

Tip entiteta **KOLEGIJ** definiran je atributima:

KolegijID, NazivKolegija, SatiPredavanja, SatiLabosa itd.

## Veze (relationships)

Odnos između ova dva entiteta **STUDENT** i **KOLEGIJ**.

Student može polagati više kolegija (to se od njega i očekuje ☺)

No student ne mora nužno polagati kolegij – na prvoj godini dok još nije odslušao kolegije

Svaki kolegij može polagati više studenata.

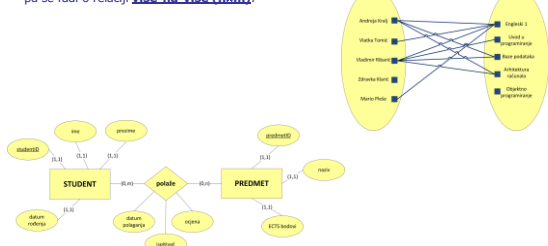
Ali se može dogoditi da neki od kolegija ne polaže niti jedan student – npr. izborni kolegij kojeg nitko nije slušao.

## Veze (relationships)

Za relaciju **POLAŽE** koja definira odnos između entiteta **STUDENT** i **KOLEGIJ**, vrijedi

$\min \text{ card } (\text{STUDENT}, \text{POLAŽE}) = 0$        $\min \text{ card } (\text{KOLEGIJ}, \text{POLAŽE}) = 0$   
 $\max \text{ card } (\text{STUDENT}, \text{POLAŽE}) = n$        $\max \text{ card } (\text{KOLEGIJ}, \text{POLAŽE}) = m$

pa se radi o relaciji **više-na-više (nxm)**.

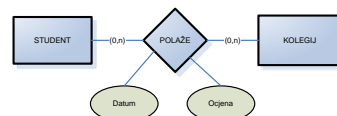


## Veze (relationships)

Za relacije više-na-više svojstveno je da sa sobom može donijeti određene attribute koji su posljedica veze.

U našem primjeru, javljaju se atributi **OCJENA** i **DATUM** (datum polaganja).

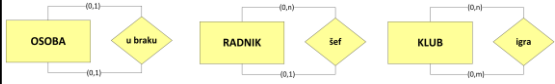
Ovi atributi nisu dio niti jednog od entiteta koji sudjeluju u relaciji, već su posljedica veze, u ovom slučaju tipa entiteta **STUDENT** i **KOLEGIJ**.





### Red (stupanj) veze

- Veza prema redu (stupnju) može biti:
  - unarna – u vezi sudjeluju dva entiteta istog tipa entiteta
  - binarna – u vezi sudjeluju dva entiteta različitog tipa entiteta
  - veza višeg reda – u vezi sudjeluju najmanje tri entiteta različitog tipa entiteta
- Sve dosad izložene veze su bile binarne.
- Za unarnu vezu postoje još i sljedeći nazivi: rekurzivna veza, ring struktura, prstenasta struktura, involucijska veza, refleksivna veza.



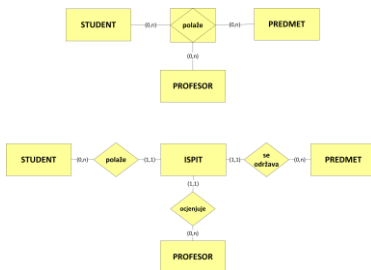
### Agregirani tip entiteta

- Neke metodike koriste poseban simbol za specifične oblike veze – romb unutar pravokutnika.
- To se čini u sljedećim situacijama:
  - ako u vezi sudjeluje više od dva entiteta (veza višeg reda)
  - ako tip veze ima vlastite atribute
  - ako se radi o vezi n:m (više na više)
- U ova tri slučaja govori se o agregiranom (mješovitom) tipu entiteta ili agregaciji



### Agregirani tip entiteta

- Agregirani tip entiteta može se izraziti pomoću novog tipa entiteta



### EV dijagram

Proširenjem grafičkog prikaza entiteta, na međusobne veze među entitetima, dobija se **EV dijagram** (dijagram ENTITETI-VEZE).

- Veza se simbolički ucrtava deltoidnim znakom (◊) između entiteta koji sudjeluju u relaciji, te se povezuje s njima.
- Na poveznici tipa entiteta i veze upisuje se kardinalitet entiteta u relaciji u obliku uredenog para.
- Atributi koji su posljedica veze ucrtaavaju se ovalnim simbolom kao i atributi tipa entiteta, te se povezuju sa vezom iz koje proizilaze.

### Postupak izrade EV dijagrama

- Iako ne postoji standardni postupak izrade EV dijagrama, mogući su sljedeći koraci u njegovom oblikovanju (ne nužno tim redoslijedom):
  - Prepoznavanje tipova entiteta
  - Prepoznavanje atributa
  - Definiranje kardinalnosti atributa u tipu entiteta
  - Određivanje ključeva
  - Prepoznavanje tipova veza između tipova entiteta (koje veze postoje između entiteta)
  - Određivanje vrste veze (1:1, 1:n ili m:n).
  - Prepoznavanje eventualnih atributa veze

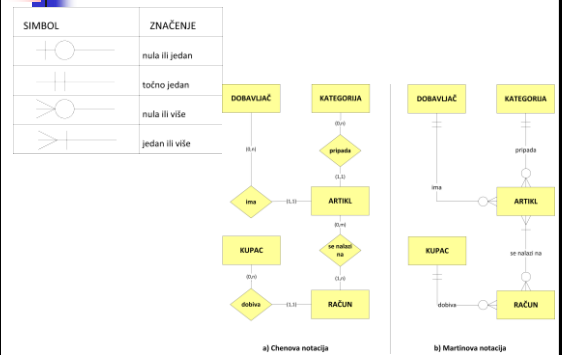
### Postupak izrade EV dijagrama

- nekoliko dodatnih napomena:
  - Imenice ukazuju na entitete (npr. TRGOVAC izdaje RAČUN)
  - Jedan dokument postaje entitet, no u jednom dokumentu može se uočiti i više entiteta (npr. iz dokumenta RAČUN će nastati dva entiteta: RAČUN i KUPAC ukoliko se na dokumentu nalaze podaci o kupcu)
  - Kod utvrđivanja da li je nešto entitet ili atribut pomaže sljedeći kriterij: entitet ima svojstva, dok ih atribut nema.
  - Atribut koji ima svoje atribute je entitet.
  - Glagoli ukazuju na veze (npr. Trgovac IZDAJE račun).
  - Veza više na više uglavnom ima svoje atribute. Datum je jedan od njezinih najčešćih atributa.
  - Ponekad je minimalna/maksimalna kardinalnost konkretan broj. Primjerice minimalni broj igrača jednog tima na utakmici je 7, a maksimalni 11. Taj konkretan broj se može upisati na dijagramu umjesto oznake n.

### EV dijagram - zadatak

- Potrebno je evidentirati podatke o studentima (studentID, ime, prezime, datum rođenja i hobi (student može imati više hobija), zatim o osobnim iskaznicama (broj iskaznice, datum izdavanja i izdavatelj), mjestima (mjestoID, naziv i poštanski broj) te o predmetima (predmetID, naziv, ECTS bodovi).
- Također je potrebno evidentirati koja osobna iskaznica pripada kojem studentu, u kojem mjestu je student rođen, ali i u kojem mjestu student stanuje.
- Pored toga, potrebno je evidentirati koje kolegije je student polagao i pri tome treba ostati zapis o datumu polaganja te ocjeni.

### EV dijagram Martinova notacija



### Relacijski model podataka

- Relacijski model je najkompletniji teorijski model podataka, a doživio je i najveći broj komercijalnih implementacija.
- Temeljna načela relacijskog modela postavio je 1970. godine dr. Edgar F. Codd.
- Ovaj općeprihvaćeni model se zasniva na matematičkim principima teorije skupova i predikatne logike.
- Relacijski model definira oblik u kojem se podaci predstavljaju (strukturu), pravila i načine kojima se osigurava ispravnost podataka u bazi (ograničenja) te operacije koje se mogu izvršavati nad podacima.

### Relacijska struktura

	StudentID	Ime	Prezime	DatumRođenja
n-torke	1	Dunja	Kralj	5.11.1988
	2	Jagoda	Levanić	11.2.1992
	3	Zvezdan	Smetiško	29.6.1986
	4	Jela	Galunić	30.7.1990
	5	Lana	Petrinec	15.12.1992
vrijednost	6	Trpimir	Mikac	1.8.1993

### Svojstva relacije

- Unutar jedne relacije ne smiju postojati dvije iste n-torke (ili: u istoj tablici ne smiju biti dva potpuno ista retka).
- Unutar jedne relacije ne smiju postojati dva atributa istog imena (ili: u istoj tablici ne smiju postojati dva istoimena stupca).
- Redoslijed n-torki je neuređen (ili: redoslijed redaka u tablici je potpuno nebitan).
- Redoslijed atributa je neuređen (ili: redoslijed stupaca u tablici je potpuno nebitan).
- Pojedinačne vrijednosti u relaciji su atomarne.

StudentID	Ime	Prezime	Predmet
1	Dunja	Kralj	Baze, Mreže
2	Jagoda	Levanić	Programiranje, Baze, Mreže
3	Zvezdan	Smetiško	Informatizacija, Matematika, Mreže

### Primarni ključ relacije

- Primarni ključ je atribut ili skup atributa koji jedinstveno određuje svaku pojedinu n-torku (svaki pojedini redak u tablici).
- Primarni ključ mora zadovoljiti tri osnovna uvjeta:
  - Jedinstvenost
    - U relaciji (tablici) ne smiju postojati dvije iste n-torke (retka) s istom vrijednošću primarnog ključa
  - Minimalnost
    - Ako je primarni ključ složen, tada se ne može ukloniti niti jedna njegova komponenta (atribut), a da se pri tom ne naruši pravilo jedinstvenosti.
  - Integritet
    - Niti jedan atribut koji je dio primarnog ključa ne smije poprimiti null vrijednost, što znači da kardinalnost svakog atributa koji čini primarni ključ mora biti (1,1).

### Primarni ključ relacije

- U praksi se primarni ključ često sastoji od samo jednog atributa te ga se naziva jednostavnim ključem (eng. simple key).
- Ako se sastoji od većeg broja atributa, naziva se složenim primarnim ključem (eng. composite key).
- Svaka relacija ima primarni ključ, jer ako ne postoji pravi podskup atributa relacije koji zadovoljava uvjete za primarni ključ, onda to mora biti skup svih atributa relacije (proizlazi iz prvog svojstva relacije)

### NULL vrijednost

- NULL predstavlja neodređenu vrijednost atributa, te nije ovisna o tipu podataka. NULL vrijednost se razlikuje od
  - praznog niza znakova (stringa) "",
  - znaka bjeline (razmaka) " ",
  - niza znakova "NULL",
  - broja 0 ili bilo kojeg drugog broja.
- NULL vrijednost se može odnositi na:
  - **trenutno nepoznatu ili nepostojeću** vrijednost (npr. atribut ProdajnaCijena za proizvod kome cijena još nije definirana)
  - **neprimjenjivu** vrijednost atributa za danu n-torku (npr. RadniObujamMotora nije primjenjiv za elektromotor, dok je za diesel i benzinski motor primjenjiv)

### Strani ključ

- Pojednostavljeno govoreći, strani ključ (eng. *foreign key*) je atribut ili skup atributa koji nije primarni ključ u promatranoj relaciji, ali je primarni ključ neke druge relacije
- Strani ključ FK koji se nalazi u relaciji R mora slijedeće uvjete:
  - Svaka vrijednost stranog ključa različita od null mora biti identična jednoj vrijednosti primarnog ključa relacije S s kojom je relacija R u logičkoj vezi.
  - Svaka pojedina vrijednost stranog ključa mora biti potpuno null ili potpuno ne-null.

StudentID	Ime	Prezime	MjestoID
1	Dunja	Kralj	24
2	Jagoda	Levančić	23
3	Zvezdan	Šmetlićko	23

MjestoID	Naziv	Pbr	BrojStanovnika	Drzava
23	Zaprešić	10290	18000	HR
24	Križevci	48250		HR
26	Zupanja	35170	10000	HR
27	Zlatar	49250	6500	HR

### Strani ključ

StudentID	Ime	Prezime	MjestoIDKod	MjestoIDStan
1	Dunja	Kralj	24	24
2	Jagoda	Levančić	23	26
3	Zvezdan	Šmetlićko	23	27

MjestoID	Naziv	Pbr	BrojStanovnika	Drzava
23	Zaprešić	10290	18000	HR
24	Križevci	48250		HR
26	Zupanja	35170	10000	HR
27	Zlatar	49250	6500	HR

RadnikID	Ime	Prezime	ŠefID
1	Izabela	Zvagač	
2	Mutimir	Riharić	1
3	Srebrnko	Miliac	2
4	Ogriznita	Petrinac	1
5	Častimir	Fundek	2

### Integritetska ograničenja

- Integritet podataka znači potpunost (cjelovitost) i točnost podataka u bazi.
- Relacijski model podataka sadrži načela koja određuju klasu valjanih stanja baze podataka, čime se ujedno ograničavaju i mogućnosti mijenjanja sadržaja baze podataka.
- Takva načela se nazivaju uvjetima integriteta ili integritetnim ograničenjima.
- Integritetna ograničenja mogu se podijeliti na opća i specifična.
- Opća ograničenja vrijede za sam relacijski model, a samim time i za svaku relacijsku bazu podataka.
- Specifična ograničenja ovise o konkretnim relacijama odnosno o pojedinoj predmetnoj bazi podataka.

### Integritetska ograničenja

- Relacijski model definira dva opća ograničenja: entitetski integritet i referencijalni integritet.
- Oba ograničenja proizlaze iz definicije relacije i ključeva.
- Entitetska ograničenja ne govore ništa novo, već naglašavaju ono što je ranije rečeno o primarnom i stranom ključu.
- Pravilo entitetskog integriteta zahtijeva da atributi koji čine primarni ključ ne smiju poprimiti ili sadržavati null vrijednost. Ovo pravilo je identično trećem uvjetu kojeg mora zadovoljiti primarni ključ

### Integritetska ograničenja

- Pravilo referencijalnog integriteta zahtijeva da niti jedna vrijednost stranog ključa ne smije biti neuparena.
- Drugim riječima, u relaciji ne smije postojati vrijednost stranog ključa za koju ne postoji identična vrijednost primarnog ključa u logički povezanoj tablici.
- Ovo pravilo je identično prvom uvjetu kojeg mora zadovoljiti strani ključ

### Osiguravanje referencijalnog integriteta

- Kod unosa se referencijalni integritet osigurava na samo jedan način: zabranom unosa stranog ključa ako ta vrijednost stranog ključa ne poklapa s vrijednošću primarnog ključa u logički povezanoj tablici.
- Kod promjene primarnog ključa referencijalni integritet se može osigurati na jedan od tri načina:
  - zabranom promjene vrijednosti primarnog ključa, ako ta vrijednost postoji u nekoj povezanoj tablici kao strani ključ;
  - kaskadnom promjenom vrijednosti primarnog ključa, ako ta vrijednost postoji u nekoj povezanoj tablici kao strani ključ i to tako da se promjenom vrijednosti u primarnom ključu, promijene i sve vrijednosti u stranom ključu kako bi se održala veza među podacima; ili
  - postavljanjem na null stranog ključa prilikom promjene primarnog ključa.

### Osiguravanje referencijalnog integriteta

- Kod brisanja primarnog ključa (brisanjem retka u tablici) referencijalni integritet se može osigurati na jedan od tri načina:
  - zabranom brisanja retka, ako vrijednost primarnog ključa u tom retku postoji u nekoj povezanoj tablici kao strani ključ;
  - kaskadnim brisanjem redaka u tablici u kojima je vrijednost stranog ključa jednaka vrijednosti primarnog ključa u obrisanoj retku; ili
  - postavljanjem na null stranog ključa prilikom brisanja retka s primarnim ključem.

### Prevođenje EV modela u relacijski model podataka

#### 1. Tip entiteta

- Tip entiteta postaje relacija.
- Naziv relacije treba biti istovjetan nazivu tipa entiteta. Pojedinačni entiteti postaju n-torke relacije.
- Atributi tipa entiteta postaju atributi relacije.
- Primarni ključ tipa entiteta postaje primarni ključ relacije.
- Kad se relacijski model implementira, relacija je ustvari tablica, atributi su stupci (kolone), n-torke redovi u tablici, a primarni ključ relacije je primarni ključ tablice

EV model	Relacijski model	Implementirana baza podataka
tip entiteta	relacija	tablica
entitet	n-torka	redak
atribut	atribut	stupac (kolona)
primarni ključ tipa entiteta	primarni ključ relacije	primarni ključ tablice

### Prevođenje EV modela u relacijski model podataka

#### 2. Viševrijednosni atribut

- Viševrijednosni atributi se ne smiju prevoditi kao obični (jednovrijednosni) atributi, jer bi u krajnoj fazi za njih postojao samo jedan stupac u tablici.
- Prikazivanje više vrijednosti unutar jednog stupca je zabranjeno.
- viševrijednosni atribut se prevodi novom relacijom koja će se sastojati od dva atributa: jedan je primarni ključ tipa entiteta, a drugi je viševrijednosni atribut.
- Oba atributa će sačinjavati primarni ključ relacije.
- Dakle, primarni ključ relacije nastale prevođenjem viševrijednosnog atributa je uvijek složen.

### Prevođenje EV modela u relacijski model podataka

#### 2. Viševrijednosni atribut



Sheme relacija su slijedeće:  
 PROFESOR (profesorID, ime, prezime, OIB)  
 TELEFON (profesorID, broj telefona)

### Prevođenje EV modela u relacijski model podataka

#### 3. Veza 1:n

- Veza 1:n se prevodi u relacijski model proširivanjem jedne relacije primarnim ključem druge relacije.
- Nije svejedno koju relaciju se proširuje: potrebno je proširiti onu relaciju **koja u vezi sudjeluje samo jedanput** primarnim ključem relacije koja u vezi sudjeluje više puta.
- Ako dva tipa entiteta A i B sudjeluju u vezi V tako da vrijedi  $\max \text{kard}(A,V) = 1$  i  $\max \text{kard}(B,V) = n$ , gdje je atribut B\_id primarni ključ tipa entiteta B, tada se u relaciju A ubacuje atribut B\_id.
- Atribut B\_id je u relaciji A strani ključ

### Prevođenje EV modela u relacijski model podataka

#### 3. Veza 1:n



VLASNIK (vlasnikID, ime, prezime, adresa)  
 LJUBIMAC (ljubimacID, naziv, vrsta, vlasnikID)

### Prevođenje EV modela u relacijski model podataka

#### 4. Veza m:n

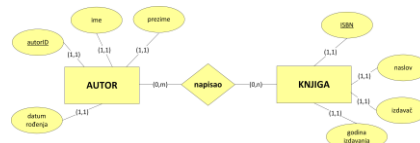
- Ako dva tipa entiteta A i B sudjeluju u vezi V tako da vrijedi:  
 $\max \text{kard}(A,V) = m$   
 $\max \text{kard}(B,V) = n$   
 $\text{PK}(A) = A\_id$   
 $\text{PK}(B) = B\_id$

veza se prevodi novom relacijom C čija je shema  
 C (A\_id, B\_id)

a primarni ključ  $\text{PK}(C) = \{A\_id, B\_id\}$   
 pri čemu treba imati na umu da su A\_id i B\_id strani ključevi u relaciji C.

### Prevođenje EV modela u relacijski model podataka

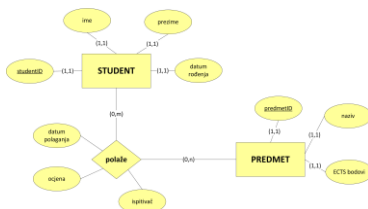
#### 4. Veza m:n



AUTOR (autorID, ime, prezime, datum rođenja)  
 KNJIGA (ISBN, naslov, izdavač, godina izdavanja)  
 NAPISAO (autorID, ISBN)

### Prevođenje EV modela u relacijski model podataka

#### 4. Veza m:n



?

### Prevođenje EV modela u relacijski model podataka

#### 5. Veza 1:1

- Veza u kojoj dva tipa entiteta A i B mogu sudjelovati najviše jednom može se prevesti na tri načina:
  - umetanjem primarnog ključa relacije B (B\_id) u relaciju A
  - umetanjem primarnog ključa relacije A (A\_id) u relaciju B
  - proširivanjem jedne relacije svim atributima druge relacije (pri čemu druga relacija potpuno nestaje!)

## Prevođenje EV modela u relacijski model podataka

### 5. Veza 1:1



DRŽAVA (državaID, ime, površina, međunarodna oznaka, glavniID)  
GLAVNI GRAD (glavniID, naziv, broj stanovnika, državaID)

ili:

DRŽAVA (državaID, ime, površina, međunarodna oznaka)  
GLAVNI GRAD (glavniID, naziv, broj stanovnika, državaID)

## Prevođenje EV modela u relacijski model podataka

### 6. Tip slabog entiteta

- Tip slabog entiteta također postaje relacija.
- Sve rečeno za prvo pravilo vrijedi i u ovom slučaju, jedino se razlikuje formiranje primarnog ključa: u slučaju da se radi o identifikacijskoj vezi primarni ključ relacije bit će složen od primarnog ključa jakog tipa entiteta i primarnog ključa slabog tipa entiteta.

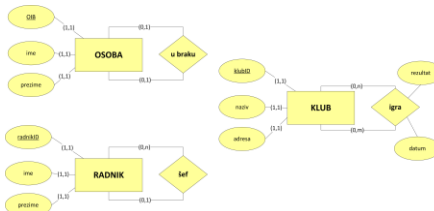


HOTEL (hotelID, naziv, adresa, kategorija)  
SOBA (hotelID, broj sobe, tip sobe, zauzeta)

## Prevođenje EV modela u relacijski model podataka

### 7. Rekurzivne veze

- Kod prevođenja unarnih veza potrebno je ovisno o vrsti veze primijeniti odgovarajuće pravilo uz obavezno preimenovanje stranog ključa.



## Relacijska algebra

- podrazumijeva definirane operacije nad entitetima (tablicama) i podacima koji im pripadaju

### Operacije teorije skupova (Set-theory operations)

**Unija** ( $T := R \cup S$ )

**Presjek** ( $T := R \cap S$ )

**Razlika** ( $T := R - S$ )

**Produkt** ( $T := R \times S$ )

### Prirodne relacijske operacije (Native-relation operations)

**Projekcija** ( $T := R [a]$ )

**Selekcija** ( $T := R \text{ where } a=12$ )

**Spajanje** ( $T := R \bowtie S$ )

**Dijeljenje** ( $T := R \div S$ )

## Relacijska algebra

### Definicija

Kompatibilnim tablicama smatramo tablice koje

- ☐ Sadrže atribute jednakog naziva i isti broj atributa
- ☐ Atributi istog naziva definirani su nad jednakim domenama

## Unija

### UNIJA ( $T := R \cup S$ )

- može se provoditi samo nad kompatibilnim tablicama
- tablica T je unija tablica R i S,
- ima isto zaglavlje kao tablice R i S, a sadrži sve redove koji se nalaze bilo u tablici R ili S.

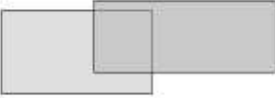


Diagram illustrating the union operation. Two overlapping rectangles represent sets R and S. The intersection is shaded gray, and the union is outlined with a thicker border.

A	B	C
s	1	4
d	1	5
e	2	5

A	B	C
d	1	5
e	2	4
f	4	4

A	B	C
s	1	4
d	1	5
e	2	5
e	2	4
f	4	4

## Unija

### UNIJA ( $T := R \cup S$ )

SELECT \* FROM R UNION [ALL] SELECT \* FROM S;

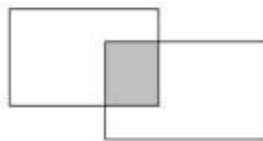
ALL se stavlja ako ne želimo izbaci duple n-torke.



## Presjek

### **PRESJEK ( $T := R \cap S$ )**

- može se provoditi samo nad kompatibilnim tablicama
- presjek dvaju skupova R i S je skup T
- sastoji se od svih elemenata koji pripadaju i skupu R i skupu S



A	B	C
s	1	4
d	1	5
e	2	5

A	B	C
d	1	5
c	2	4
f	4	4

A	B	C
d	1	5

## Presjek

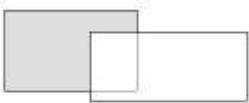
### **PRESJEK ( $T := R \cap S$ )**

- *SELECT R.\* FROM R, S  
WHERE R.pbr=S.pbr;*
- *SELECT R.\* FROM R INNER JOIN S  
ON R.pbr=S.pbr;*
- *SELECT \* FROM R  
WHERE R.pbr IN (SELECT pbr FROM S);*
- *SELECT \* FROM R  
WHERE EXISTS (SELECT S.pbr FROM S  
WHERE S.pbr=R.pbr);*

## Razlika

### RAZLIKA ( $T := R - S$ )

- razlike se može provoditi samo nad kompatibilnim tablicama
- razlika dvaju skupova R i S je skup T
- sastoji se od svih elemenata koji pripadaju skupu R i ne pripadaju skupu S



A	B	C
s	1	4
d	1	5
e	2	5

A	B	C
d	1	5
e	2	4
f	4	4

A	B	C
s	1	4
e	2	5

## Razlika

### RAZLIKA ( $T := R - S$ )

- *SELECT \* FROM R  
WHERE R.pbr NOT IN (SELECT pbr FROM S);*
- *SELECT \* FROM R  
WHERE NOT EXISTS (SELECT S.pbr FROM S  
WHERE S.pbr=R.pbr);*

## Produkt

### PRODUKT ( $T := R \times S$ )

-operacija produkta nad entitetima (tablicama),  
temelji se na skupovnoj operaciji Kartezijevog produkta.

-tablica T je produkt tablica R i S, čije je zaglavlje (header) definirano:

-**head( $R \times S$ ) = head(R) U head(S)**,

-elementi(redovi) te tablice nastaju spajanjem redova iz tablice R i redova iz tablice S.

R

A	B	C
s	1	4
d	1	5
e	2	5

S

D	E	F
d	1	5
e	2	4

$T = R \times S$

A	B	C	D	E	F
s	1	4	d	1	5
s	1	4	e	2	4
d	1	5	d	1	5
d	1	5	e	2	4
e	2	5	d	1	5
e	2	5	e	2	4

## Produkt

Općenito za produkt tablica vrijedi

R ( $m \times n$ ) - m redova i n kolona

S ( $k \times l$ ) - k redova i l kolona,

onda je  $T := R \times S$  tablica sa ( $m \cdot k$ ) redova i ( $n+l$ ) kolona

Za operatore unije, presjeka i produkta vrijede pravila asocijativnosti i komutativnosti.

$$(R \cup S) \cup T = R \cup (S \cup T) = R \cup S \cup T$$

$$(R \cap S) \cap T = R \cap (S \cap T) = R \cap S \cap T$$

$$(R \times S) \times T = R \times (S \times T) = R \times S \times T$$

Međuviznost operacija

$$R \cap S = R - (R - S) = S - (S - R)$$

$$R - S = R - (R \cap S)$$

## Produkt

**PRODUKT** ( $T := R \times S$ )

⊙ *SELECT R.\*, S.\* FROM R, S;*

## Projekcija

**PROJEKCIJA**  $T := R[a]$

-izdvajaju se atributi tablice na kojima se vrši projekcija.

-projekcija tablice R nad atributima X,Y,Z jest tablica T

-T sa zaglavljem  $\text{head}(T) = \{X, Y, Z\}$

-T sadržava sve redove koji su sadržani u tablici R, nova tablica predstavlja vertikalni podskup zadane tablice.


R		
A	B	C
s	1	4
d	1	5
c	2	5

T:=R[A]	A
	s
	d
	c

## Projekcija

**PROJEKCIJA**  $T := R[a]$

- *SELECT naziv FROM R;*
- *SELECT R.naziv FROM R;*

## Selekcija

**SELEKCIJA - IZDVAJANJE** ( $T := R \text{ where } a = \text{vrijednost}$ )

- selekcija nad tablicom R izdvaja skup redova koji zadovoljavaju uvjet
- tablica koja se dobija selekcijom sadrži sve atribute kao i izvorna tablica
- sadrži samo one redove koji zadovoljavaju traženi uvjet
- dobijena tablica predstavlja horizontalni podskup izvorne tablice


R

A	B	C
s	1	4
d	1	5
s	4	6
c	2	5

R where A=s

A	B	C
s	1	4
s	4	6

## Selekcija

**SELEKCIJA - IZDVAJANJE** ( $T := R$  where  $a=X$ )

- *SELECT \* FROM R  
WHERE ocjena = 5;*
- *SELECT \* FROM R  
WHERE R.ocjena = 5;*

## Spajanje - join

**SPAJANJE – JOIN**

Operacija spajanja ima nekoliko podvrsta:

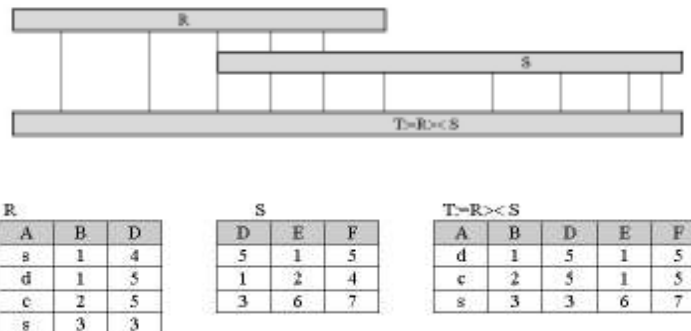
- INNER JOIN** - unutrašnja veza  $T := R \bowtie S$
- LEFT OUTER JOIN** - lijeva vanjska veza  $T := R \bowtie_{LO} S$
- RIGHT OUTER JOIN** - desna vanjska veza  $T := R \bowtie_{RO} S$
- OUTER JOIN** - vanjska veza  $T := R \bowtie_O S$

## Unutrašnja veza – inner join

### Inner join- unutrašnja veza $T = R \bowtie S$

-spajaju se redovi tablica po istim vrijednostima zajedničkog atributa

-spajaju se redovi koji u kolonama istog naziva u obje tablice imaju istu vrijednost



## Unutrašnja veza – inner join

### Inner join- unutrašnja veza $T = R \bowtie S$

Student (R)			Mjesto (S)		$T = R \bowtie S$			
stud ID	ime prez	pbr rod	pbr	naziv	stud ID	ime prez	pbr rod	naziv
1234	Pero Perić	31000	21000	Split	1234	Pero Perić	31000	Osijek
1422	Ive Marić	22000	31000	Osijek	1283	Ana Sršen	21000	Split
1765	Mia Koch	10000	23000	Zadar				
1283	Ana Sršen	21000	51000	Rijeka				

**SELECT R.\*, naziv FROM R INNER JOIN S  
ON R.pbr\_rod=S.pbr;**

**SELECT R.\*, naziv FROM R, S  
WHERE R.pbr\_rod=S.pbr;**

## Lijeva vanjska veza – left outer join

### Left outer join- lijeva vanjska veza $T = R \bowtie_{LO} S$

-lijeva vanjska veza je proširenje unutrašnje veze

-dodaju se oni elementi tablice, koji su u relacijskom izrazu sa lijeve strane, a koji ne sudjeluju u vezi

-u stvaranju  $R \bowtie_{LO} S$  najprije se realizira  $R \bowtie S$  (time se dobiju prva tri reda tablice T)

-dodaju svi redovi tablice R, koji ne sudjeluju u inner-join vezi.

R	A	B	D
s	1	4	
d	1	5	
c	2	5	
s	3	3	

S	D	E	F
5	1	5	
1	2	4	
3	6	7	

$T = R \bowtie_{LO} S$	A	B	D	E	F
d	1	5	1	5	
c	2	5	1	5	
s	3	3	6	7	
s	1	4	null	null	

## Lijeva vanjska veza – left outer join

### Left outer join- lijeva vanjska veza $T = R \bowtie_{LO} S$

Student (R)			Mjesto (S)		$T = R \bowtie_{LO} S$			
stud ID	ime prez	pbr	pbr	naziv	stud ID	ime prez	pbr	naziv
1234	Pero Perić	31000	21000	Split	1234	Pero Perić	31000	Osijek
1422	Ive Marić	22000	31000	Osijek	1286	Ana Sršen	21000	Split
1765	Mia Koch	10000	23000	Zadar	1422	Ive Marić	22000	NULL
1283	Ana Sršen	21000	51000	Rijeka	1756	Mia Koch	10000	NULL

**SELECT R.\*, naziv FROM R LEFT JOIN S  
ON R.pbr=S.pbr;**



### Desna vanjska veza – right outer join

**Right outer join- desna vanjska veza  $T = R \bowtie_{RO} S$**

-desna vanjska veza je proširenje unutrašnje veze.

-dodaju se oni elementi tablice, koji su u relacijskom izrazu sa desne strane , a koji ne sudjeluju u vezi

-u stvaranju  $T = R \bowtie_{RO} S$  najprije se realizira  $R \bowtie S$  (time se dobiju prva tri reda tablice T)

-dodaju svi redovi tablice S, koji ne sudjeluju u inner-join vezi.

R			S			$T = R \bowtie_{RO} S$				
A	B	D	D	E	F	A	B	D	E	F
s	1	4	5	1	5	d	1	5	1	5
d	1	5	1	2	4	e	2	5	1	5
c	2	5	3	6	7	s	3	3	6	7
s	3	3				null	null	1	2	4

### Desna vanjska veza – right outer join

**Right outer join- desna vanjska veza  $T = R \bowtie_{RO} S$**

Student (R)			Mjesto (S)		$T = R \bowtie_{RO} S$			
stud_ID	ime_prez	pbr	pbr	naziv	stud_ID	ime_prez	pbr	naziv
1234	Pero Perić	31000	21000	Split	1283	Ana Sršen	21000	Split
1422	Ive Marić	22000	31000	Osijek	1234	Pero Perić	31000	Osijek
1765	Mia Koch	10000	23000	Zadar	NULL	NULL	23000	Zadar
1283	Ana Sršen	21000	51000	Rijeka	NULL	NULL	51000	Rijeka

**SELECT R.stud\_ID, R.ime\_prez, S.pbr, naziv FROM R RIGHT JOIN S  
ON R.pbr=S.pbr;**

## Vanjska veza – outer join

**Outer join- vanjska veza  $T = R \bowtie_o S$**

- vanjska veza je proširenje unutrašnje veze.
- dodaju se elementi obje tablice koji ne sudjeluju u unutrašnjoj vezi

-vrijedi  $T = R \bowtie_o S = (R \bowtie_{LO} S) \cup (R \bowtie_{RO} S)$

R			S			$T = R \bowtie_o S$				
A	B	D	D	E	F	A	B	D	E	F
s	1	4	5	1	5	d	1	5	1	5
d	1	5	1	2	4	c	2	5	1	5
e	2	5	3	6	7	s	3	3	6	7
s	3	3				null	null	1	2	4
						s	1	4	null	null

Vrijedi  $T = R \bowtie_o S = (R \bowtie_{LO} S) \cup (R \bowtie_{RO} S)$

## Vanjska veza – outer join

**Outer join- vanjska veza  $T = R \bowtie_o S$**

Student (R)			Mjesto (S)		$T = R \bowtie_o S$			
stud_ID	ime_prez	pbr	pbr	naziv	stud_ID	ime_prez	pbr	naziv
1234	Pero Perić	31000	21000	Split	1283	Ana Sršen	21000	Split
1422	Ive Marić	22000	31000	Osijek	1234	Pero Perić	31000	Osijek
1765	Mia Koch	10000	23000	Zadar	1422	Ive Marić	22000	NULL
1283	Ana Sršen	21000	51000	Rijeka	1756	Mia Koch	10000	NULL
					NULL	NULL	23000	Zadar
					NULL	NULL	51000	Rijeka

```
SELECT R.*, S.naziv FROM R RIGHT JOIN S ON R.pbr = S.pbr
UNION
SELECT R.stud_ID, R.ime_prez, S.pbr, naziv FROM R LEFT JOIN S ON
R.pbr = S.pbr
```

## Dijeljenje

### Dijeljenje $T := R \div S$

- tablica T koja se dobije dijeljenjem R i S
- T je najveća tablica za koju vrijedi da se **svi redovi produkta**  $T \times S$  nalaze u tablici R.

R		
A	B	D
s	1	4
s	1	5
c	2	5

S	
A	B
s	1

$T := R \div S$	
D	
4	
5	

### Prioritet logičkih operatora

Projekcija  
 Selekcija  
 Produkt  
 Spajanje, Dijeljenje  
 Razlika  
 Unija, Presjek

## Logičke operacije

AND	T	F
T	T	F
F	F	F

OR	T	F
T	T	T
F	T	F

NOT	
T	F
F	T

**Unutrašnju vezu može se izraziti kombinacijom produkta, selekcije i projekcije**

A	B
1	2
3	4

B	C	D
2	5	6
4	7	8
9	10	11

A	B	C	D
1	2	5	6
3	4	7	8

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

- **inner-join** podrazumijeva samo one redove iz produkta gdje atributi istog naziva imaju iste vrijednosti
- potrebno je primijeniti selekciju tj. izdvojiti redove gdje atributi istog naziva u obje tablice imaju istu vrijednost

**Unutrašnju vezu može se izraziti kombinacijom produkta, selekcije i projekcije**

**D:= C where R.B=S.B =(R×S) where R.B= S.B**

- tablica D ima jedan atribut više u odnosu na rezultat inner-join operacije

A	R.B	S.B	C	D
1	2	2	5	6
3	4	4	7	8

- izvršimo projekciju: **T:=D[A,R.B,C,D]**

A	B	C	D
1	2	5	6
3	4	7	8



## SQL

Tipovi podataka (format) mogu se podijeliti u 4 ključne kategorije

- znakovni tipovi podataka
- binarni podaci
- numerički tipovi podataka
- podaci formata datum- vrijeme

### ZNAKOVNI TIPOVI PODATAKA

*Character*

*Char (n)* – string duljine točno n znakova. Svaki znak prikazuje se jednim bajtom. Znakovi mogu biti alfanumerički i specijalni.

*Character varying*

*Varchar (n)* – string promjenive duljine, maksimalno n karaktera a najčešće iznosi 255. Pri unosu podataka, duljina stringa nije fiksna već je određena veličinom podatka.

*Text*

*Text* - String neograničene veličine.



## SQL

### BINARNI PODACI

***Bit***

*Bit*– podatak koji predstavlja binarnu vrijednost ( 1 ili 0). Veličina rezerviranog memorijskog prostora je 1 bajt. Ukoliko se bit tipu podataka pokuša pridjeliti cjelobrojna vrijednost, ona se interpretira kao 1.

### NUMERIČKI TIPOVI PODATAKA

***Binary(n)***

Sadržava maksimalno 255 bajta binarnih podataka, te se koristi za spremanje binarnih i heksadecimalnih vrijednosti.

***INTEGER – int***

cjelobrojni format s predznakom ili bez njega. Zauzima 4 bajta i sadržava cijele brojeve od  $-2^{31}$  (-2,147,483,648) do  $(2^{31} - 1)$  (2,147,483,647).



## SQL

### NUMERIČKI TIPOVI PODATAKA

***SMALL INTEGER - smallint***

cjelobrojni format s predznakom ili bez njega. Zauzima 2 bajta i sadržava cijele brojeve od  $-2^{15}$  (-32,768) do  $(2^{15}-1)$  (32,767).

***TINY INTEGER (BYTE) - tinyint***

cjelobrojni format koji prikazuje brojeve od 0 do 255. Zauzima 1 bajt.

***NUMERIC - numeric(p,q)***

decimalni broj, ukupno p znamenki i predznak, q znamenki iza decimalnog zareza.

***DECIMAL - decimal(p,q)***

decimalni broj, koji se interno bilježi sa m znamenki, pri čemu vrijedi  $0 < p < q < m$ . Decimal (7,2) dopušta unos broja koji ima točno dvije znamenke iza decimalnog zareza, a ukupan broj znamenaka mu je najmanje 7. Stvarni broj znamenaka koji se rezervira za interni prikaz u bazi zavisi od operacijskog sustava.



## SQL

### NUMERIČKI TIPOVI PODATAKA

***FLOAT – Float(n)***

Realni broj u formatu pomičnog zareza (floating point). Zauzima 4 bajta memorije.

***DOUBLE – Double(n)***

Realni broj dvostruke preciznosti u formatu pomičnog zareza (floating point). Zauzima 8 bajta memorije.



## SQL

### DATUM I VRIJEME FORMAT

Datum i vrijeme sastavljeni su od alfanumeričkih podataka koji predstavljaju podatke o datumu i vremenu. Uobičajeni format prikaza je "dan-mjesec-godina sat:minut: sekunda AM"

#### **datetime**

Je tip podatka koji je predstavljen sa 8 bajtova ili dva 4-bajtna cjela broja: 4 bajta za prikaz datuma i 4 bajta za prikaz vremena (sati). Moguće je prikazati datume od 1. siječnja 1753 do 31. prosinca 9999, sa točnošću od 3.33 milisekunde.

#### **smalldatetime** – skraćeni format datum- vrijeme

Je tip podatka koji je predstavljen sa 4 bajta: 2 bajta za prikaz datuma i 2 bajta za prikaz vremena (sati). Moguće je prikazati datume od 1. siječnja 1900 do 6. lipnja 2079, sa točnošću od minute.



## SQL: FORMIRANJE TABLICE

```
CREATE TABLE ime_tablice  
(  
    naziv_kolone1 / svojstva / column_constraint ,  
    naziv_kolone2 / svojstva / column_constraint,  
    ....  
    table constraints  
)
```

#### **ime\_tablice**

Ime svake tablice mora biti jedinstveno u bazi podataka. Dužina naziva tablice može imati maksimalno 128 znakova.

#### **naziv\_kolone**

Predstavlja naziv pojedinačne kolone u tablici. Naziv pojedine kolone mora biti jedinstven u tablici.

#### **svojstva**

Određuju tip podataka, null vrijednosti, identity – svojstvo za kolonu.



## SQL: FORMIRANJE TABLICE

### **IDENTITY**[(*seed*, *increment*)]

Generira inkrementalnu vrijednost za svaki novi red podataka u tablici na osnovu *seed* i *increment* parametara. Ako je navedena, *seed* vrijednost označava početak brojača i bit će dodijeljena prvom redu u tablici, a svaki slijedeći red dobit će jedinstvenu identity vrijednost, koja je jednaka posljednjoj identity vrijednosti uvećanoj za vrijednost parametra *increment*. Ako vrijednosti *seed* i *increment* nisu navedene smatra se da je njihova vrijednost 1. Pridjeljivanje IDENTITY svojstva koloni podrazumijeva njezino svojstvo NOT NULL. Samo jedna kolona u tablici može biti definirana sa svojstvom identity. Svojstvo identity može se dodijeliti onoj koloni koja je definirana kao numerički tip podataka.

### **NULL | NOT NULL**

Svojstvo nuliranja kolone, koje određuje da li je navedena kolona može sadržavati null vrijednosti. Ukoliko se u definiciji kolone ovo svojstvo ne navede izričito, smatra se NULL.



## SQL: FORMIRANJE TABLICE

### **CONSTRAINT** *ime\_ograničenja*

Constraint – ograničenje, definira ograničenja i dodatna relacijska svojstva unutar tablice. Ako *ime\_ograničenja* nije navedeno u naredbi CREATE TABLE, sistem sam generira naziv za pojedina ograničenja.

Pri tome je bitno razlikovati postavljanje ograničenja na pojedine kolone, od postavljanja ograničenja na nivou tablice (*table constraints*). Ukoliko su ograničenja vezana za pojedinu kolonu ona se pri definiranju tablice navode u definiciji kolone. Ako se radi o ograničenjima koja obuhvaćaju više kolona (npr. složeni primarni ključ), njih je potrebno definirati kao ograničenja na nivou tablice, nakon što su definirane sve kolone u tablici.

### **PRIMARY KEY** [CLUSTERED | NONCLUSTERED] (*col\_1*, *col\_2*, ..., *col\_n*)

Osigurava integritet i jedinstvenost podataka u određenom atributu (koloni), gdje su *col\_1*, ..., *col\_n* kolone (atributi) koje čine primarni ključ. Sve kolone koje čine primarni ključ moraju imati svojstvo NOT NULL. Ako to nije posebno navedeno, sustav sam postavlja NOT NULL, za sve kolone koje čine primarni ključ tablice. Ukoliko se ograničenje primarnog ključa definira na nivou kolone, lista kolona se izostavlja. Za osiguranje jedinstvenosti primarnog ključa, SQL automatski formira jedinstveni (unique) index na zadane kolone. Unutar tablice može postojati samo jedan primarni ključ.



## SQL: FORMIRANJE TABLICE

**UNIQUE [CLUSTERED | NONCLUSTERED] (col\_1, col\_2, ..., col\_n)**

Osigurava jedinstvenost podataka u navedenim kolonama), gdje su col\_1, ..., col\_n kolone (atributi) obuhvaćene ograničenjem. Ukoliko se ograničenje jedinstvenosti definira na nivou kolone, lista kolona se izostavlja. Unutar tablice može se definirati više UNIQUE ograničenja. SQL sustav automatski stvara jedinstveni(unique) index za kolonu ograničenu UNIQUE.

**FOREIGN KEY (col\_1, ..., col\_n) REFERENCES ime\_tablice (ref\_col\_1, ..., ref\_col\_m)**

Definira referencijalni integritet – vezu stranog ključa. Broj kolona i tip podataka u svakoj koloni koja se navodi u FOREIGN KEY col\_n izrazu mora se poklapati sa onima navedenim u REFERENCES dijelu ref\_col\_n.

**DEFAULT konstanta / izraz**

Određuje uobičajenu vrijednost (default) za kolonu, kada se prilikom unosa podataka u tablicu ne navede vrijednost za taj atribut. DEFAULT constraint može se primijeniti na kolone bilo kojeg tipa podataka, osim one kolone kojoj je dodijeljeno IDENTITY svojstvo.

**CHECK (expression)**

Ograničava moguće vrijednosti koje se unose u pojedine kolone tablice.

## SQL: PROMJENA STRUKTURE TABLICE

Promjene na već formiranoj tablici obavljaju se naredbom **ALTER TABLE**

**Dodavanje kolona ili ograničenja (constraints) u tablicu**

**ALTER TABLE ime\_tablice**

**ADD naziv\_kolone/ svojstva/ constraints, naziv\_kolone/ svojstva/ constraints**

ALTER TABLE student ADD CONSTRAINT UQ\_JMBG UNIQUE (jmbg)

ALTER TABLE mjesto ADD CONSTRAINT DF\_DRZAVA DEFAULT 'Hrvatska' FOR drzava

ALTER TABLE student ADD spol char (1) NOT NULL

CONSTRAINT CK\_SPOL check (spol='M' OR spol='Ž')



## SQL: PROMJENA STRUKTURE TABLICE

Brisanje kolona ili ograničenja (constraints) iz tablice

ALTER TABLE *ime\_tablice*

DROP COLUMN *naziv\_kolone*, CONSTRAINT *naziv\_ograničenja*

ALTER TABLE student DROP COLUMN ime\_oca

Promjene svojstava postojećih kolona u tablici

ALTER TABLE *ime\_tablice* ALTER COLUMN *naziv\_kolone* *svojstva*

ALTER TABLE student ALTER COLUMN prezime varchar(20) NOT NULL



## SQL: PROMJENA STRUKTURE TABLICE

### **Dodavanje novih kolona**

Ako tablica već sadrži neke podatke, svaki od već postojećih redova u tablici proširuje se sa novom kolonom, za koju prethodno nije definirana njegova vrijednost. U ovakvim slučajevima, moguća su dva rješenja :

1. Kolona koja se dodaje u tablicu treba se definirati sa svojstvom NULL. Na taj način svim već postojećim redovima dodaje se nova kolona, a vrijednosti u toj koloni za svaki redak u tablici bit će null.
2. Ako se kolona ne želi definirati sa svojstvom null, za nju je potrebno navesti default vrijednost. U takvim slučajevima svi postojeći redovi u novostvorenoj koloni poprimaju vrijednost koja je određena kao default.

ALTER TABLE student ADD ime\_majke varchar (20)

ili ALTER TABLE student ADD ime\_majke varchar (20) NOT NULL DEFAULT 'Nije poznato'

### **Dodavanje novih ograničenja**

Kod dodavanja novih ograničenja treba voditi računa da svako ograničenje

## SQL: PROMJENA STRUKTURE TABLICE

### Dodavanje novih ograničenja

Ukoliko se nekoj tablici, koja već sadrži određene podatke (određeni broj redova) želi dodati ograničenje tipa CHECK ili FOREIGN KEY, potrebno je definirati kako se novo ograničenje odnosi prema već postojećim podacima.

```
CREATE TABLE upisni_list
(
    ulist_id int IDENTITY
    CONSTRAINT PK_UPISNI PRIMARY KEY,
    student_id int NOT NULL
    CONSTRAINT FK_ST_UPIS FOREIGN KEY REFERENCES student(student_id),
    sem tinyint NOT NULL,
    sk_god varchar(8) NOT NULL,
    obr_prog_id int NOT NULL
    CONSTRAINT FK_OBR_UPIS FOREIGN KEY REFERENCES obr_prog(obr_prog_id)
)
```

```
ALTER TABLE upisni_list ADD CONSTRAINT CHK_SEM CHECK (sem>0
AND sem<6)
```

## SQL: PROMJENA STRUKTURE TABLICE

### Brisanje postojećih kolona

Nije moguće ukloniti kolone

- koje čine primarni ključ ili dio primarnog ključa
- koje predstavljaju strani ključ
- koje su vezane ograničenjem tipa CHECK
- jedinstvene kolone (UNIQUE)
- kolone koje imaju određenu default vrijednost.
- kolone po kojima postoji indeks.



## SQL: PROMJENA STRUKTURE TABLICE

### Promjena svojstava postojećih kolona

Promjena svojstava neke od postojećih kolona vrši se instrukcijom oblika

**ALTER TABLE** ime\_tablice **ALTER COLUMN** naziv\_kolone| svojstva

Na tablici je moguće promijeniti samo svojstva kolone, a ne i pojedina ograničenja. Kod promjene pojedinih ograničenja, potrebno je najprije ukloniti postojeće ograničenje, a nakon toga definirati novo. Stoga je instrukcijom ALTER TABLE moguće samo mijenjanje svojstava vezanih za pojedinu kolonu (tip podataka i null (not null svojstvo)).

Primjenom ove instrukcije nije moguće promijeniti svojstva kolona:

- koje čine primarni ključ ili dio primarnog ključa
- koje predstavljaju strani ključ
- koje su vezane ograničenjem tipa CHECK ili UNIQUE. Jedina promjena koja se može izvršiti jest ako su te kolone definirane kao niz znakova varchar, dozvoljeno je mijenjanje duljine niza.
- kolone po kojima postoji indeks.



## SQL: BRISANJE TABLICE

Tablica se briše naredbom **DROP TABLE** ime\_tablice

## SQL: UNOS PODATAKA U TABLICE

**INSERT [INTO] *ime\_tablice* (*kolone*) VALUES (*vrijednosti*)**

*ime\_tablice*

Opisuje tablicu u koju se unose podaci.

*kolone*

Lista kolona u koje se unose vrijednosti. Kolone se mogu navesti u proizvoljnom redoslijedu, ali podaci koji se unose (vrijednosti) moraju biti u istom redoslijedu kao i navedene kolone. Navođenje kolona potrebno je samo ako se novi red tablice popunjava sa samo nekim kolonama, a ne svim.

VALUES

Ključna riječ koja se koristi za navođenje vrijednosti koje se unose u novi red tablice.

*vrijednosti*

Vrijednosti koje se unose u novi red tablice. Tip podataka pojedinih vrijednosti mora odgovarati tipu podataka koji je definiran za pojedinu kolonu.

## SQL: UNOS PODATAKA U TABLICE

```
CREATE TABLE student
(
    student_id int IDENTITY
    CONSTRAINT PK_STUDENT PRIMARY KEY,
    ime varchar(20) NOT NULL,
    prezime varchar(30) NOT NULL,
    jmbg char(13),
    spol char(1),
    dat_rođenja smalldatetime NOT NULL,
    mjesto_rod int not null
    CONSTRAINT FK_MJESTO_ST FOREIGN KEY REFERENCES mjesto(mjesto_id)
)
```

head(student)={student\_id, ime, prezime,jmbg,spol, dat\_rođenja, mjesto\_rod}

**A. INSERT – Unos svih kolona tablice**

**INSERT student VALUES('Marko', 'Bilić', '1110980370086', 'M', '11.10.1980',11)**

**B. INSERT sa nazivima kolona**

**INSERT student(spol, mjesto\_rod, dat\_rođenja, ime, prezime)  
VALUES ('M', 11, '11.10.1980', 'Marko', 'Bilić')**

## SQL: UNOS PODATAKA U TABLICE

svojstvo kolone	nema ulaza, nema default	nema ulaza, postoji default	ulaz null nema default	ulaz null postoji default
null	null	Default	null	null
not null	greška – error	Default	greška- error	greška- error

## SQL: AŽURIRANJE PODATAKA U TABLICI

**UPDATE naziv\_tablice SET**  
**naziv\_kolone = nova\_vrijednost,**  
**naziv\_kolone2 = nova\_vrijednost2**  
**[FROM tablica\_1, tablica\_2,...]**  
**WHERE uvjetni izraz**

*UPDATE STUDENT SET ime='Marko' where ime='Darko' and prezime='Y'*

*UPDATE UPISNI\_LIST SET sem=3 FROM STUDENT, UPISNI\_LIST WHERE ime='Darko' AND prezime='Y' AND STUDENT.STUDENT\_ID=UPISNI\_LIST.STUDENT\_ID AND sem=1*

**UPDATE STUDENT SET mjesto\_id=MJESTO.mjesto\_id FROM MJESTO, STUDENT**  
**WHERE MJESTO.ime\_mjesta = 'Trogir' and student\_id=21**

## SQL: BRISANJE PODATAKA IZ TABLICE

Brisanje podataka iz tablice provodi se naredbom DELETE prema sintaksi  
**DELETE** *naziv\_tablice*  
**[FROM** *tablica\_1, tablica\_2,...***]**  
**WHERE** *uvjetni izraz*

*DELETE STUDENT where ime='Darko' and prezime='Y'*

*DELETE UPISNI\_LIST FROM STUDENT, UPISNI\_LIST WHERE ime='Darko' AND prezime='Y'  
AND STUDENT.STUDENT\_ID=UPISNI\_LIST.STUDENT\_ID AND sem=1*

ili

*DELETE UPISNI\_LIST FROM STUDENT INNER JOIN UPISNI\_LIST ON  
STUDENT.STUDENT\_ID=UPISNI\_LIST.STUDENT\_ID WHERE ime='Darko' AND prezime='Y'  
AND sem=1*

## SQL: DATUMSKE FUNKCIJE

• **NOW – trenutno vrijeme dobiveno od operacijskog sustava**  
NOW se evaluira kao vrijeme "5.11.2002. 10:30:01" ako je današnji datum 5. studeni 2002. i vrijeme je 10:30:01.

• **DATE – trenutni datum dobiven od operacijskog sustava.**  
Evaluiira se kao "5.11.2002." ako je današnji datum 5. studeni 2002.

• **DAY – broj dana za zadani datum**  
DAY(21.03.2002) je cijeli broj 21

• **DATESERIAL – evaluira datum iz 3 INT vrijednosti (godina, mjesec, dan)**  
DATESERIAL(2002, 3, 21) je datumska vrijednost 21.03.2002.

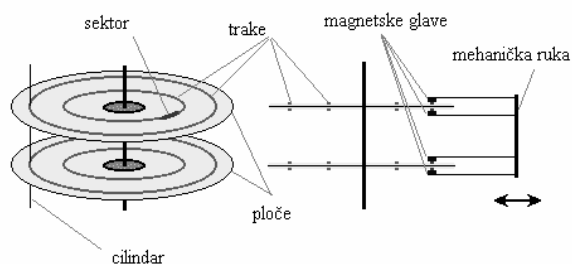
• **MONTH – za zadani datum vraća broj mjeseca**  
MONTH(21.03.2002) je cijeli broj 3

• **WEEKDAY – redni broj dana u tjednu za zadani datum (1-nedjelja)**  
WEEKDAY(NOW) 2 ako je danas ponedjeljak

• **YEAR – redni broj godine za zadani datum**  
YEAR(21.03.2002) je cijeli broj 2002

## Indeksiranje baze podataka

- **podaci u bazi podataka** (tablice i ostale strukture) moraju biti trajno smješteni na mediju koji omogućava njihovo **normalno održavanje**.
- **radna memorija računala**, prestankom **napajanja** ili sklopovskog problema dolazi do **gubitka** svih podataka
- **disk** je rotirajući magnetski medij , sastavljen od nekoliko ploča, mehaničke ruke i magnetske glave
- **disk** je sastavljen od niza **traka, sektora i cilindara**



## Indeksiranje baze podataka

**Za pristup određenom podatku na disku, potrebno je da:**

- mehanička ruka zauzima položaj na određenom cilindru,
- čeka se da bi traženi sektor bio smješten ispod magnetske glave
- vrši se učitavanje podataka.

### Nedostaci

- prve dvije faze su znatno trajnije i čine najveći dio vremena pristupa podacima
- pristup podacima na disku je spor, što čini glavni problem smještaja podataka na disk.



## Indeksiranje baze podataka

**Indeks** za određeni atribut jest tablica

- sa **dva stupca**
- i sa **brojem redova** kao i tablica **nad kojom je indeks** definiran.
- **prva kolona** u indeksu je atribut za koji je indeks definiran i naziva se **ključ indeksa** (index key)
- u drugoj koloni je **disk pointer** (pokazivač na mjesto na disku gdje se nalazi podatak sa određenom vrijednošću indeksnog ključa).
- tablica indeksa **uvijek je uređena po vrijednostima ključa**, bilo u rastućem ili padajućem redoslijedu

INDEX (Prezime)	
Prezime (Key)	D.P.
Bilić	P1
Milić	P2

STUDENT				
id	Ime	Prezime	JMBG	Datum rod.
1	Marko	Milić	xxxxxx	
2	Ante	Bilić	xxxxxx	

## Indeksiranje baze podataka

- **prostor** diska za smještaj **baze podataka** podijeljen je na stranice
- uobičajene **veličine** stranica su **2Kb** (2048 byte) i **4Kb** (4096 byte).
- **podaci** iz tablice smještaju su na stranice tvrdog diska **u slotove**
- broj slotova na pojedinoj stranici zavisi od veličine podatka u jednom redu tablice.

### Primjer:

- memorijska **stranica** je **2Kb**,
- za smještaj **jednog reda** tablice treba **128 byte-a**,
- onda **jedna stranica** za **smještaj podataka** iz navedene tablice sadrži **16 slotova**.

## Indeksiranje baze podataka

slot 1	Page 1
slot 2	
....	
slot 16	
slot 1	Page 2
slot 2	
....	
slot 16	

- **stranica** predstavlja **osnovnu jedinicu** za učitavanje podataka sa diska.
- najveći dio **vremena** za učitavanje podataka čini **smještaj** magnetske glave
- **učitavaju** se čitave **stranice** kako bi se što više  **smanjila potreba** za ponovnim pozicioniranjem glave
- **uvođenje indeksa** u bazu ima za cilj ubrzavanje pretraživanja, smanjenje broja pristupa disku.
- indeks definiran nad tablicom vezuje se za određeni atribut (kolonu ) u tablici

## Indeksiranje baze podataka

indeksiranja <----->katalogizacijom knjiga u biblioteci

- za **svaku knjigu** postoji odgovarajuća **kartica** sa **osnovnim** podacima o knjizi i podatkom **gdje je smještena**.
- pri tome može postojati nekoliko **odvojenih kataloga**,
- u jednom su knjige sortirane prema naslovu, u drugom **prema imenu autora** itd.
- kad korisnik zatraži knjigu, npr. od željenog autora, potrebno je **pogledati u katalog** gdje su knjige sortirane po autoru, te se nađe pripadajuća kartica i pogleda **u kojem je dijelu knjižnice ta knjiga**.
- ako se podacima **pristupa** po vrijednostima atributa nad kojima postoji **indeks**
- **sustav** ne pretražuje **cijelu** tablicu, **pretražuje se tablica indeksa** koja je mnogo manja
- **nije nužno** pretražiti **cijelu** tablicu indeksa, već samo **do zadane** vrijednosti indeksnog ključa
- indeks je uređen po vrijednostima ključa, **ostatak tablice indeksa** ne sadrži podatke sa zadanom vrijednošću ključa.



## Indeksiranje baze podataka

### **prednosti indeksiranja:**

- ako upiti ispituju samo postojanje određenog podatka
- dovoljno je pretraživanje indeksa, bez pretraživanja osnovne tablice.

### **nedostaci indeksiranja:**

- smanjuje se brzina ažuriranja i dodavanja novih podataka.
- uz promjene u samoj tablici, potrebno je promijeniti i tablicu indeksa.



## Indeksiranje baze podataka

### **Višestruki indeksi**

- predstavljaju indeksiranje bazne tablice po većem broju atributa.
- indeksna tablica zadržava istu strukturu,
- ključ indeksa složen je od vrijednosti atributa koji su uključeni u indeks.

INDEX (Prezime,JMBG)

(Key)	D.P.
Bilić;2222	P1
Milić;1111	P2

STUDENT

id	Ime	Prezime	JMBG	Datum rod.
1	Marko	Milić	1111	
2	Ante	Bilić	2222	

- pretraživanje po jednoj ili obje vrijednosti atributa
- pretraživanje po jednoj vrijednosti atributa identično je pretraživanju indeksa sa jednim atributom
- pretraživanje po više vrijednosti daje rezultat samo za one vrijednosti gdje oba atributa zadovoljavaju tražene uvjete.

## Indeksiranje baze podataka – clustered indeks

- uvođenjem indeksa formira se **indeksna tablica**
- **redoslijed** podataka u **osnovnoj** tablici ostaje **nepromijenjen**
- pretraživanjem **indeksa** dobije se podatak o **smještaju** traženih podataka
- ali su oni u **osnovnoj** tablici **raspršeni**.
- ako se indeks **definira** svojstvom **CLUSTERED**,
- **bitno** se mijenja **raspored** podataka u **osnovnoj** tablici.
- CLUSTERED indeks podrazumijeva smještaj podataka u osnovnoj tablici u uređenom redoslijedu prema vrijednostima ključa.

INDEX (Prezime)

Prezime (Key)	D.P.
Antić	P1
Bilić	P2
Katić	P3
Milić	P4

STUDENT

id	Ime	Prezime	JMBG	Datum rođ.
1	Ivo	Antić	xxxxx	
2	Ante	Bilić	xxxxx	
3	Jure	Katić		
4	Marko	Milić		

**U jednoj tablici može postojati samo jedan CLUSTERED indeks.**

## Indeksiranje baze podataka – unique indeks

- **indeks** sa svojstvom jedinstvenosti (UNIQUE)
- u indeksnoj tablici **ne mogu** postojati dvije iste vrijednosti indeksnog ključa
- **vrijednost** atributa ne može biti **null**
- atribut po kojem se stvara jedinstveni indeks, mora imati minimalni kardinalitet 1.
- za višestruke indekse, koji predstavljaju kombinaciju više atributa,
- jedinstveni indeks podrazumijeva jedinstvenost kombinacije atributa koji su obuhvaćeni indeksom

### Primjer:

- definira se jedinstveni indeks na tablici STUDENT po atributima ime i prezime,
- u tablici ne mogu postojati dva studenta koji imaju jednaka oba atributa (ime i prezime).

## Indeksiranje baze podataka

### KOMPRESIJA INDEKSA

- pretpostavlja se **postojanje jednog retka u tablici indeksa za svaki red u osnovnoj tablici**.
- ukoliko **indeks nije jedinstven**, moguć je veći broj **istih** vrijednosti indeksnog ključa
- **ključ** indeksa podatak je **duži** od **pokazivača**
- grupiranjem pokazivača po istim vrijednostima ključa provodi se kompresija indeksa

Prx	key	D.P.	D.P.	D.P.
-----	-----	------	------	------

Prx	key	D.P.	D.P.
-----	-----	------	------

- na početku bloka nalazi se **Prefiks bloka** (Prx), koji sadrži podatak o
- početku prethodnog i slijedećeg bloka (druge vrijednosti ključa) i
- broju vrijednosti tekućeg ključa (broju pokazivača u tekućem bloku).

## Indeksiranje baze podataka

### B-TREE STRUKTURA INDEKS-a

- indeks je po definiciji tablica, nalazi se smješten na disku računala
- smješten je po memorijskim stranicama u slotovima.

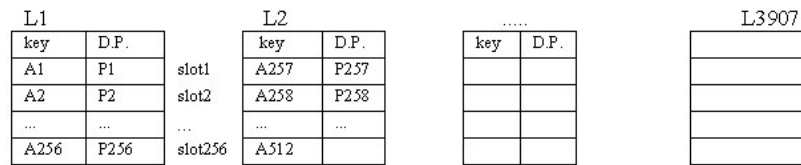
#### Primjer:

- **tablica sa milijun redova podataka** indeks je definiran nad **cjelobrojnim atributom**

- jedan red u tablici indeksa sadržava vrijednost **indeksnog ključa** (*integer – 4byte*)
- i **disk pointer** (*integer 4-byte*).

- za svaku vrijednost indeksnog ključa *8 byte*                      veličina memorijske stranice *2Kb*
- na jednu stranicu dolazi    *2Kb:8byte=256 indeksnih redova.*
- za smještaj milijun redova u tablici indeksa potrebno je *1.000.000:256=3907 stranica.*

## Indeksiranje baze podataka

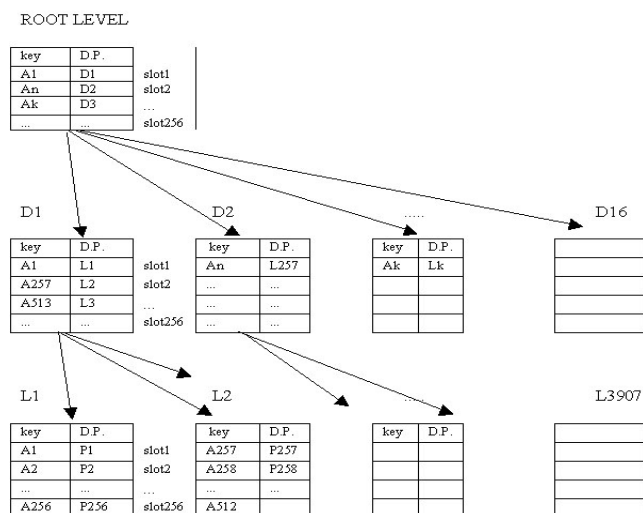


L1-L3907...LEAF pages na kojima su smješteni osnovni indeksi

- pri tome su vrijednosti ključa uređene po **rastućem rasporedu**
- vrijedi  **$A_k < A_m$** , za svaki  **$k < m$**
- radi ubrzavanja pristupa indeksima
- formira se viši nivo indeksnih stranica
- **početni indeks sa svake leaf stranice** uključuje **se u nove stranice** te služi kao pokazivač
- za smještaj 3907 indeksnih redova, potrebno je  $3907:256=16$  stranica
- organizira se i najviši indeksni nivo, tzv.root level, po identičnom principu

## Indeksiranje baze podataka

**Primjer:** pronaći vrijednost indeksnog ključa A380  $2Kb:8byte=256$  indeksnih redova.





## Normalizacija baze podataka

- **Normalizacija** predstavlja primjenu:

**matematičkih**

**i formalnih pravila**

- osigurava se **ispravno postavljanje modela** podataka i njihova **logička povezanost**.

### 1. NORMALNA FORMA

*Tablica se nalazi u 1. normalnoj formi ako su svi neključni atributi funkcijски ovisni o primarnom ključu.*



## Normalizacija baze podataka

**Def:**

**Funkcijska zavisnost atributa**

*Za tablicu **R** koja sadrži attribute **X** i **Y** koji mogu biti i složeni vrijedi **funkcijska zavisnost atributa** **Y** o atributu **X**,*

*tj.  **$X \rightarrow Y$** ,*

*ako je svaka pojedina vrijednost atributa **X** povezana sa samo jednom vrijednošću atributa **Y**.*

ILI

*Vrijedi  **$X \rightarrow Y$***

*ako u tablici **R** ne postoje dva reda sa istom vrijednošću atributa **X**, a različitim vrijednostima atributa **Y**.*

- pravilo je jednostavno i proizlazi iz definicije primarnog ključa

- naglašava transformacijsko pravilo prikaza viševrijednosnih atributa.

## Normalizacija baze podataka

### Primjer:

prikaz viševrijednosnog atributa **sport** za entitet **STUDENT**

**head(STUDENT)={Student\_id,Ime,Prezime,Dat.rođenja, Mj.rođenja,JMBG,Sport}**

**card(Ime,STUDENT) = (1,1),**

**card(Prezime,STUDENT) = (1,1),**

**card(Dat.rođenja,STUDENT) = (1,1),**

**card(Mj.rođenja,STUDENT) = (1,1),**

**card(JMBG,STUDENT) = (1,1),**

**card(Sport,STUDENT) = (0,n)**

STUDENT

Student_id	Ime	Prezime	Dat.rođenja	Mj. rođenja	JMBG	Sport
1	Ante	Rožić	11.10. 1980	Osijek	1110980370071	-
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Odbojka
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Košarka
3	....	....	....	....		

- tablica u skladu sa pravilom

da nije dozvoljen unos više vrijednosti pojedinog atributa u jednom reduku.

- tablica nije u prvoj normalnoj formi,

jer svi neključni atributi nisu funkcijski zavisni o ključu.

**Student\_id ↗ Sport**

## Normalizacija baze podataka

-osim posljedice da ne možemo na **jedinstveni način** doći do zapisa

tj. da je **повrijeđena je jedinstvenost primarnog ključa**

javljaju se **razne anomalije**,

problemi koji se javljaju pri **unosu** i **manipuliranju** podacima.

### Anomalija unosa:

za svaki sport kojim se bavi pojedini student, potrebno je ponovno unijeti vrijednost svih atributa vezanih za tog studenta: Ime, Prezime, itd.

### Anomalija ažuriranja/promjene podataka:

Kod promjene vrijednosti nekog atributa potrebno je taj atribut promijeniti u svim redovima.



## Normalizacija baze podataka

STUDENT

Student id	Ime	Prezime	Mj. rođenja	JMBG
1	Ante	Rožić	Osiijek	1110980370071
2	Stipe	Anić	Split	0307980380025
3	....	....	....	

SPORT

Student id	Sport	Dat.rođenja
2	odbojka	03.07. 1980
2	košarka	03.07. 1980
...	...	...

- problem viševrijednosnog atributa sada je u skladu sa transformacijskim pravilom, tj. atribut je prikazan novom tablicom **PK(SPORT)=(Student\_id,Sport)**

- tablica sadrži i atribut Dat.rođenja  
- datum rođenja je vezan za studenta, neovisno o tome da li se bavi sportom ili ne vrijedi  
**Student\_id → Dat.rođenja,**

**Student\_id** je podskup primarnog ključa u tablici SPORT

## Normalizacija baze podataka

Anomalije koje se javljaju kada **neključni atributi ovise samo o dijelu ključa**

### Anomalija unosa:

- Ne može se unijeti datum rođenja za studenta koji se ne bavi sportom.

### Anomalija promjene:

- Promjene datuma rođenja uzrokuje promjenu u svim redovima koji su vezani za tog studenta.

### Anomalija brisanja:

- Brisanjem posljednjeg sporta kojim se student bavi, gubi se i podatak o njegovom datumu rođenja

## Normalizacija baze podataka

### 2. NORMALNA FORMA

**Tablica se nalazi u 2. normalnoj formi ako se nalazi u 1. normalnoj formi, i ako su svi neključni atributi potpuno funkcijski zavisni o ključu.**

Pravilo druge normalne forme vrijedi za složeni primarni ključ koji se sastoji od više atributa.

#### DEF: Potpuna funkcijska zavisnost atributa

*U tablici **R** koja sadrži attribute **X** i **Y** koji mogu biti i složeni, vrijedi da je **Y potpuno funkcijski zavisan o atributu X**, ako je **Y** funkcijski zavisan o **X** i nije funkcijski zavisan niti o jednom manjem podskupu atributa X.*

*Drugim riječima,*

*kada vrijedi **X**→**Y**, tada ne smije postojati niti jedan podskup **Z (Z⊂X)**, za koji bi vrijedilo **Z**→**Y**.*

## Normalizacija baze podataka

STUDENT

Student id	Ime	Prezime	Mj. rođenja	JMBG
1	Ante	Rožić	Osijek	1110980370071
2	Stipe	Anić	Split	0307980380025
3	....	....	....	

SPORT

Student id	Sport	Dat.rođenja
2	odbojka	03.07. 1980
2	košarka	03.07. 1980
...	...	...

Problem se rješava :

- da se atribut koji je funkcijski zavisan o podskupu primarnog ključa (student\_id), prebacuje u drugu tablicu u kojoj je taj atribut-podskup primarni ključ, a to je u ovom slučaju tablica SPORT.

## Normalizacija baze podataka

STUDENT

Student id	Ime	Prezime	JMBG
1	Ante	Rožić	1110980370071
2	Stipe	Anić	0307980380025
3	....	....	

UPISNI LIST

ulist_id	student_id	semestar	šk.godina	obr.program	Mj.rođenja
1	1	1	1999/00	Elektronika	Osijek
2	1	2	1999/00	Elektronika	Osijek

### Anomalija unosa:

Nije moguće unijeti mjesto rođenja za pojedinog studenta, dok se ne unese njegov upisni list.

### Anomalija ažuriranja/promjene podataka:

Kod promjene mjesta rođenja pojedinog studenta, taj podatak treba mijenjati u svim upisnim listovima, koji su vezani za tog studenta.

### Anomalija brisanja:

Brisanjem upisnog lista briše se podatak o mjestu rođenja

## Normalizacija baze podataka

STUDENT

Student id	Ime	Prezime	JMBG
1	Ante	Rožić	1110980370071
2	Stipe	Anić	0307980380025
3	....	....	

UPISNI LIST

ulist_id	student_id	semestar	šk.godina	obr.program	Mj.rođenja
1	1	1	1999/00	Elektronika	Osijek
2	1	2	1999/00	Elektronika	Osijek

vrijedi     ulist\_id→student\_id (svaki upisni list vezan je samo za jednog studenta)  
              student\_id↔ulist\_id (jedan student ima više upisnih listova)  
              student\_id →Mj.rođenja (student je rođen u jednom određenom mjestu)  
              Mj.rođenja↔student\_id (u jednom mjestu može biti rođeno više studenata)

Kako vrijedi

              ulist\_id→student\_id →Mj.rođenja, atribut mjesto rođenja je tranzitivno  
(posredno) funkcijski zavisao o primarnom ključu tablice (ulist\_id).

## Normalizacija baze podataka

### 3. NORMALNA FORMA

**Tablica se nalazi u 3. normalnoj formi ako se nalazi u 2. normalnoj formi i ako niti jedan neključni atribut nije tranzitivno funkcijski zavisian o primarnom ključu.**

**Definicija:**

**Tranzitivna funkcijska zavisnost atributa**

U tablici R koja sadrži attribute  $X, Y$  i  $A$ , vrijedi da je  $A$  tranzitivno funkcijski zavisian o atributu  $X$ , ako je  $X \rightarrow Y, Y \rightarrow A$  i  $X \rightarrow A$ .

Simbolički tranzitivna zavisnost se prikazuje  $X \rightarrow Y \rightarrow A$

**Svođenja tablice na treću normalnu formu rješava se tako da:**

**atribut koji je tranzitivno zavisian o ključu, preseli u se tablicu, u kojoj je atribut koji posreduje u tranzitivnoj vezi (u ovom slučaju student\_id) primarni ključ.**

## Normalizacija baze podataka

STUDENT

Student id	Ime	Prezime	JMBG
1	Ante	Rožić	1110980370071
2	Stipe	Anić	0307980380025
3	....	....	

UPISNI LIST

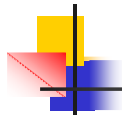
ulist id	student id	semestar	šk.godina	obr.program	Mj.rođenja
1	1	1	1999/00	Elektronika	Osijek
2	1	2	1999/00	Elektronika	Osijek

STUDENT

Student id	Ime	Prezime	Mj. rođenja	JMBG
1	Ante	Rožić	Osijek	1110980370071
2	Stipe	Anić	Split	0307980380025
3	....	....	....	

UPISNI LIST

ulist id	student id	semestar	šk.godina	obr.program
1	1	1	1999/00	Elektronika
2	1	2	1999/00	Elektronika



## Normalizacija baze podataka

### BOYCE-CODDOVA NORMALNA FORMA (BCNF)

#### Def:

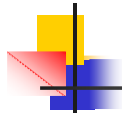
**Determinant** je **atribut** o kojem je **neki** atribut potpuno funkcijski zavisan.

*Tablica se nalazi u BCN formi ako svaki determinant ima jedinstvenu vrijednost u cijeloj tablici.*

#### Primjer:

Tablice **STUDENT** i **MJESTO**

Vrijedi da je  $PK(STUDENT) = Student\_id$   
 $PK(MJESTO) = Mjesto\_ID = FK(STUDENT)$



## Normalizacija baze podataka

### MJESTO

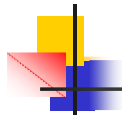
Mjesto_ID	IME MJESTA	OPĆINA	DRŽAVA
1	Split	Split	Hrvatska
2	Smokvica	Pag	Hrvatska
3	Smokvica	Smokvica	Hrvatska
4	Osijek	Osijek	Hrvatska
5	Bonn	-	Njemačka

### STUDENT

Student_id	Ime	Prezime	Dat.rođenja	Mjesto_ID	Pošt.broj
1	Ante	Rožić	11.10. 1980	4	31000
2	Stipe	Anić	03.07. 1980	1	21000
3	...	...	...	...	...

Za tablicu STUDENT vrijedi

$Student\_id \rightarrow Mjesto\_ID$ ,  $Mjesto\_ID \rightarrow Pošt.broj$ ,  $Mjesto\_ID \not\rightarrow Student\_id$ ,  $Pošt.broj \rightarrow Mjesto\_ID$



## Normalizacija baze podataka

Atribut **Mjesto\_Id** je determinant

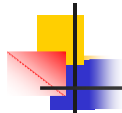
jer je atribut **Pošt.broj** potpuno funkcijski zavisan o oznaci mjesta (**Mjesto\_id**)

U tablici se **ne mogu** pojaviti dva reda  
sa *istom vrijednošću* **Mjesto\_id**, a *različitom vrijednošću* atributa **Pošt.broj**.

**Mjesto\_ID** jest **determinant**, ali nije **jedinstven**.

U tablici se mogu pojaviti *studenti* koji su *rođeni u istom mjestu*,  
tj. u raznim redovima **može se pojaviti ista vrijednost atributa Mjesto\_ID**.

Tablica dakle nije u BCN formi.



## Normalizacija baze podataka

**Nepravilnosti** koje proizlaze iz ovakve strukture podataka:

### **Anomalija unosa:**

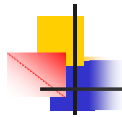
Nije moguće unijeti poštanski broj za mjesto dok god nije unešen prvi student koji je rođen u tom mjestu.

### **Anomalija promjene:**

Ako se promijeni poštanski broj pojedinog mjesta, treba promijeniti sve zapise koji pripadaju studentima rođenim u tom mjestu

### **Anomalija brisanja:**

Ako se iz tablice ukloni zadnji student koji je rođen u nekom mjestu, gubi se podatak o poštanskom broju



## Normalizacija baze podataka

### Svođenja tablice na BCN formu:

Atribut koji je potpuno funkcijski ovisan o **determinantu** koji nije jedinstven preseli se u tablicu, u kojoj je taj **determinant** (u ovom slučaju **Mjesto\_ID**) primarni ključ.

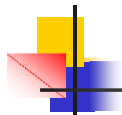
atribut **Pošt.broj** preseli u tablicu u kojoj je primarni ključ atribut **Mjesto\_ID**.

MJESTO

Mjesto_ID	IME MJESTA	POŠT. BROJ	OPĆINA	DRŽAVA
1	Split	21000	Split	Hrvatska
2	Smokvica	23249	Pag	Hrvatska
3	Smokvica	20272	Smokvica	Hrvatska
4	Osijek	31000	Osijek	Hrvatska
5	Bonn	-	-	Njemačka

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mjesto_ID
1	Ante	Rožić	11.10. 1980	4
2	Stipe	Anić	03.07. 1980	1
3	....	....	....	....



## Normalizacija baze podataka

### 4. NORMALNA FORMA

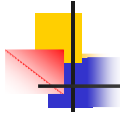
*Tablica se nalazi u 4. normalnoj formi ako i samo ako vrijedi da postojanje višeznačne zavisnosti atributa  $A \rightarrow B$  povlači za sobom postojanje funkcijske ovisnosti svih atributa u toj tablici o atributu A.*

**Def:**

**Višeznačna zavisnost atributa**

*U tablici **R** koja sadrži attribute **A, B** i **C**, vrijedi da je **B** višeznačno ovisan o atributu **A**, ako vrijedi: za svaki **B** koji odgovara vrijednostima atributa **A** i **C**, **B** je ovisan samo o **A**, ali ne i o **C**.*

Predavač	Predmet	Poglavljje
Ilić	Fizika	Optika
Ilić	Fizika	Mehanika
Ilić	Fizika	Toplina
Matić	Fizika	Optika
Matić	Fizika	Mehanika
Matić	Fizika	Toplina



## Normalizacija baze podataka

Vrijedi:

### **Predmet**→**Poglavlje**

kako je sadržaj predmeta vezan isključivo za predmet,

a ne ovisi od toga koji nastavnik predaje taj predmet.

Da bi tablica bila u **4. normalnoj formi**

treba vrijediti **Predmet**→**Predavač**,

to ovdje ne vrijedi jer za jedan isti predmet postoji više predavača.

Tablica se svodi na **4. normalnu formu** rastavljanjem postojeće tablice na dvije nove



## Normalizacija baze podataka

Predavač	Predmet
Ilić	Fizika
Matić	Fizika

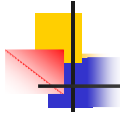
Predmet	Poglavlje
Fizika	Optika
Fizika	Mehanika
Fizika	Toplina

Pri **dekompoziciji tablica**, mora vrijediti da se **postupkom razdvajanja tablica ne izgubi** niti jedna informacija, niti se pojave informacije koje ne postoje u polaznoj tablici.

Postoje dva pravila:

- 1) Svaka funkcijska zavisnost tablice **T** može biti logički izvedena iz funkcijskih zavisnosti u tablicama **R** i **S**, koje nastanu dekompozicijom tablice **T**.
- 2) Zajednički atribut tablica **R** i **S** mora biti ključ u barem jednoj od tih tablica.





## Sortiranje izlaznih rezultata

Za sortiranje izlaznih rezultata koristi se izraz **ORDER BY**, kao dio *select* instrukcije.

*Sintaksa:*

**ORDER BY** naziv\_kolone tip\_sortiranja, naziv\_kolone tip\_sortiranja,...

**SELECT** [ALL | DISTINCT] nazivi\_kolona  
**FROM** nazivi\_tablica  
**WHERE** uvjetni izraz  
**ORDER BY** kolone\_sortiranja tip\_sortiranja ,...

Sortiranje može biti:

rastuće (ascending **ASC**)  
ili  
padajuće (descending **DESC**)

Ako nije naveden tip sortiranja podrazumijeva se rastuće (**ASC**) sortiranje.



## Sortiranje izlaznih rezultata

head(STUDENT)={student\_id, ime, prezime,jmbg,spol, dat\_rođenja, mjesto\_id}

Primjer:

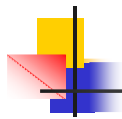
**SELECT** \*  
**FROM** student  
**ORDER BY** prezime, ime

Umjesto naziva kolona u izrazu **order by**

može se koristiti redni broj kolone u select listi.

Primjer:

**SELECT** \*  
**FROM** student  
**ORDER BY** 3 asc , ime desc



## Grupni upiti

### GROUP BY

Određuje **grupe** po kojima se dijele izlazni rezultati, i primjenu agregatnih funkcija nad pojedinim grupama.

Primijenom GROUP BY izraza, rezultati dobijeni dijelom izraza

**SELECT** [ALL / DISTINCT] nazivi kolona, agregatne kolone  
**FROM** nazivi\_tablica| join veze  
**[WHERE]** uvjetni izraz  
**[GROUP BY]** nazivi kolona  
**[HAVING]** uvjetni izraz  
**[ORDER BY]** sort\_izraz

grupiraju se na način da oni redovi izlaznog rezultata koji u svim kolonama navedenim u GROUP BY izrazu imaju iste vrijednosti spadaju u istu grupu, nad kojom se može zadati **određena agregatna funkcija**.



## Grupni upiti

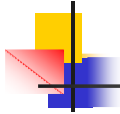
Primjer:

**SELECT** student.student\_id , ime, prezime, sem  
**FROM** student , upisni\_list  
**WHERE** student.student\_id=upisni\_list.student\_id

student_id	ime	prezime	sem
1	A	B	1
2	C	D	1
...	...	...	...
1	A	B	2
2	C	D	2
...	...	...	...
1	A	B	3

**SELECT** student.student\_id , ime, prezime, broj=count(sem)  
**FROM** student , upisni\_list  
**WHERE** student.student\_id=upisni\_list.student\_id  
**GROUP by** student.student\_id , ime, prezime

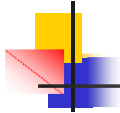
student_id	ime	prezime	broj
1	A	B	3
2	C	D	2
...	...	...	...



## Having uvjetni izrazi

Predstavlja ograničenje izlaznih rezultata po zadanom uvjetu  
nakon izvršenja **GROUP BY** operacije.

Osnovna razlika između **WHERE** i **HAVING** izraza jest u činjenici  
da **WHERE** izraz ograničava **izlazne rezultate prije njihovog grupiranja**  
tj. uvjetuje broj redova koji ulaze u grupiranje,  
dok **HAVING** izraz **postavlja uvjet na izlazne rezultate nakon što je grupiranje završeno**.

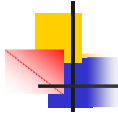


## Having uvjetni izrazi

Primjer:

```
SELECT student.student_id , ime, prezime, broj=count(sem)
FROM student , upisni_list
WHERE student.student_id=upisni_list.student_id
AND sk_god>'1997/98'
GROUP BY student.student_id , ime, prezime
```

Navedeni upit prikazuje podatke o studentima u formatu *student\_id, ime, prezime, broj*, gdje *broj* predstavlja broj upisnih listova za svakog studenta za školske godine iznad 1997/98.

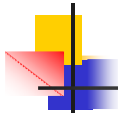


## Having uvjetni izrazi

```
SELECT student.student_id, ime, prezime, broj=count(sem)
FROM student, upisni_list
WHERE student.student_id=upisni_list.student_id
GROUP BY student.student_id, ime, prezime
HAVING sk_god>'1997/98'
```

*neispravano*, budući da operator **HAVING** djeluje nakon što završi grupiranje rezultata, kada izlazni rezultati imaju oblik **gdje se gubi podatak o školskoj godini**

student_id	ime	prezime	broj
1	A	B	3
2	C	D	2
...	...	...	...

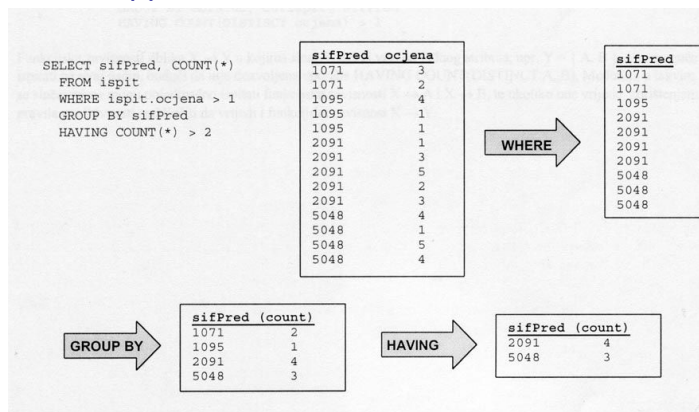


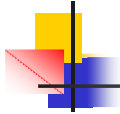
## Having uvjetni izrazi

Primjer:

Ispisati one predmete koje je prošlo više od dva studenta

```
SELECT sifPred, count (*) FROM ispit
WHERE ocjena > 1
GROUP BY sifPred
HAVING count (*)>2
```





## Primjer

Neka su definirane tablice

```
head(STUDENT)={student_id, ime, prezime,jmbg,spol, dat_rođenja, mjesto_id}  
head(MJESTO)={mjesto_id, ime_mjesta, post_br}  
head(UPISNI_LIST)={ulist_id, student_id,obr_prog_id,sk_god,sem}  
head(OBR_PROG)={obr_prog_id, obr_prog_ime}
```

Primjer:

Prikazati studente upisane u 1. sem šk.god 1999/00 sortirane po mjestu rođenja

```
SELECT mjesto.mjesto_id, ime_mjesta, broj=count(*)  
FROM student INNER JOIN upisni_list ON student.student_id=upisni_list.student_id  
        INNER JOIN mjesto ON student.mjesto_id=mjesto.mjesto_id  
WHERE sk_god='1999/00' and sem=1  
GROUP BY mjesto.mjesto_id, ime_mjesta  
ORDER BY ime_mjesta
```

ili....

## Alias

ReferentID	Ime	GodPlan	SefID
1	Zvezdoslav	285000	2
2	Izabela	150000	
3	Trpko	312000	
4	Slawica	252000	2
5	Mirko	100000	2
6	Slavko		3

• ~~SELECT ime, ime FROM referenti WHERE ReferentID=SefID~~

- SELECT ref.ime, sef.ime FROM referenti ref inner join referenti sef on ref.SefID=sef.ReferentID
- SELECT ref.ime, sef.ime FROM referenti ref, referenti sef WHERE ref.SefID=sef.ReferentID

## Podupiti

- Podupit (Subquery) je SELECT izraz uključen u glavni SELECT, INSERT, UPDATE ili DELETE upit.
- Podupit se uključuje u glavni upit u uvjetnom dijelu izraza tj. u dijelu WHERE ili HAVING.
- **SELECT [ALL | DISTINCT] nazivi kolona, agregatne kolone  
FROM nazivi\_tablica | join veze  
[WHERE uvjet=subquery]  
[GROUP BY nazivi kolona]  
[HAVING uvjet=subquery]**
- Prilikom izvršenja SQL izraza koji uključuje podupit, najprije se izvršava podupit, a rezultati koje podupit daje postaju dio uvjetnog izraza glavnog upita.

## Podupiti

- Tri osnovna oblika oblika podupita:
  - podupiti liste na koje se primijenjuje IN predikat (liste)
  - podupiti sa operatorom usporedbe
  - podupiti kojima se ispituje postojanje određenih podataka
- PODUPITI LISTE
- Prikazati studente koji nemaju niti jedan upisni list
  - *SELECT student\_id , ime, prezime FROM student WHERE student\_id not in (select student\_id from upisni\_list) ORDER BY prezime,ime*

## Podupiti

### PODUPITI SA OPERATOROM USPOREDBE

- Primjer: Prikazati sve studente koji su rođeni u istom mjestu kao i student 'x y'
  - *SELECT STUDENT\_ID , IME, PREZIME FROM STUDENT WHERE MJESTO\_ID = (SELECT MJESTO\_ID FROM STUDENT WHERE IME='X' AND PREZIME='Y') ORDER BY PREZIME,IME*
- Primjer: Izbrisati upisni list 1 semestra za studenta 'x y'
  - *DELETE UPISNI\_LIST WHERE STUDENT\_ID =(SELECT STUDENT\_ID FROM STUDENT WHERE IME='X' AND PREZIME='Y') AND SEM=1*

## Podupiti

### **PODUPITI SA OPERATOROM USPOREDBE**

- **Primjer: Prikazati sve upisne listove za studenta 'x y' osim posljednjega.**

➔ ***SELECT SK\_GOD,SEM FROM UPISNI\_LIST  
WHERE STUDENT\_ID = (SELECT STUDENT\_ID  
FROM STUDENT WHERE IME='X' AND  
PREZIME='Y') AND SEM<(SELECT MAX(SEM)  
FROM UPISNI\_LIST WHERE STUDENT\_ID=(  
SELECT STUDENT\_ID FROM STUDENT WHERE  
IME='X' AND PREZIME='Y'))***

## Podupiti

### **KORELIRANI POD-UPITI (CROSS-CORELLATED QUERIES)**

- ***SELECT \* from STUDENT where mjesto\_id=  
(SELECT mjesto\_id from MJESTO where IME\_MJESTA='Osijek')***
- ***SELECT \* from STUDENT where 'Osijek'=(SELECT  
IME\_MJESTA from MJESTO where  
STUDENT.MJESTO\_ID=MJESTO.MJESTO\_ID)***
- ***SELECT \* from STUDENT where 1 IN (SELECT sem FROM  
upisni\_list where  
STUDENT.STUDENT\_ID=UPISNI\_LIST.STUDENT\_ID AND  
sk\_god='1999/00')***



## Podupiti

### **PODUPITI SA PREDIKATOM POSTOJANJA (EXISTS, NOT EXISTS)**

- *Primjer: Prikazati sve studente koji nemaju niti jedan upisni list.*
  - ➔ *SELECT STUDENT\_ID, IME, PREZIME FROM STUDENT WHERE NOT EXISTS (SELECT \* FROM UPISNI\_LIST WHERE STUDENT.STUDENT\_ID=UPISNI\_LIST.STUDENT\_ID)*
- *Primjer: Prikazati sva mjesta u kojima je rođen neki student*
  - ➔ *SELECT \* FROM MJESTO WHERE EXISTS (SELECT \* FROM STUDENT WHERE MJESTO.MJESTO\_ID=STUDENT.MJESTO\_ID)*

## VIŠEKORISNIČKI RAD

- Većina baza podataka je višekorisnička
- Zahtjev nad DBMS-om ⇒ istovremeni pristup
- Transakcija – unaprijed definirana procedura, za korisnika nedjeljiva cjelina, obično se sastoji od nekoliko elementarnih zahvata u bazi
- Transakcija mora prevesti bazu iz jednog konzistentnog stanja u drugo

**BEGIN TRAN1**  
*naredba 1 ;*  
*naredba 2 ;*  
*..... ;*  
*naredba k ;*  
**COMMIT TRAN1**

**BEGIN TRAN2**  
*naredba 1 ;*  
*naredba 2 ;*  
*..... ;*  
*(greska) ;*  
**ROLLBACK TRAN2 ;**

- U višekorisničkoj bazi ⇒ nekoliko transakcija izvodi se paralelno
- Problem ispreplitanja osnovnih operacija koje pripadaju različitim transakcijama

## TRANSAKCIJE

### • **Problem "pamćenja" prethodnog stanja**

→ *BEGIN TRAN*  
*UPDATE P1*  
...  
*ROLLBACK TRAN*

## TRANSAKCIJE

### • **Problem zavisnosti o nepotvrđenoj promjeni**

Transakcija T1	vrijeme	Transakcija T2
AŽURIRANJE P1	t1	-----
-----	t2	SELECT P1
ROLLBACK	t3	



## VIŠEKORISNIČKI RAD

1. T1 učitava iz baze broj slobodnih sjedala.
2. T1 smanjuje pročitane vrijednosti s 1 na 0. Promjena je za sada učinjena samo u radnoj memoriji računala.
3. T2 učitava iz baze broj slobodnih sjedala. Budući da je stanje baze još uvijek nepromijenjeno, učitani broj je neažuran, dakle 1.
4. T2 smanjuje pročitane vrijednosti s 1 na 0. Promjena je opet učinjena samo u radnoj memoriji.
5. T1 unosi u bazu promijenjenu vrijednost za broj slobodnih sjedala. Dakle u bazi sada piše da nema slobodnih sjedala.
6. Slično i T2 unosi u bazu svoju promijenjenu vrijednost za broj slobodnih sjedala. Dakle u bazu se ponovo upisuje da nema slobodnih sjedala.
7. Budući da je na početku svog rada naišla na broj slobodnih sjedala veći od 0, T1 izdaje putniku kartu.
8. Iz istih razloga i T2 izdaje drugom putniku kartu.



## VIŠEKORISNIČKI RAD

### LOKOTI I ZALJUČAVANJE

**Lokot je pomoćni podatak koji služi za koordinaciju konfliktnih radnji**

1. T1 zaključa podatak o sjedalu 1 jer mu misli pristupiti.
2. Iz istih razloga T2 zaključa podatak o sjedalu 2.
3. T1 traži lokot za sjedalo 2, ali mora čekati jer je T2 već zaključala dotični podatak.
4. Slično T2 traži lokot za sjedalo 1, ali mora čekati jer je T1 već zaključala taj podatak.



## VIŠEKORISNIČKI RAD

### LOKOTI I ZALJUČAVANJE

1. T1 zaključa sjedalo 1, čita ime Ivan i pamti ga kao prvo ime, te otključa sjedalo 1.
2. T1 zaključa sjedalo 2, čita ime Marko i pamti ga kao drugo ime, te otključa sjedalo 2.
3. T1 ponovo zaključa sjedalo 1, upisuje mu zapamćeno drugo ime (dakle Marko), te otključa sjedalo 1.
4. T2 zaključa sjedalo 1, čita ime Marko i pamti ga kao svoje prvo ime, te otključa sjedalo
5. T2 zaključa sjedalo 2, čita ime Marko i pamti ga kao svoje drugo ime, te otključa sjedalo 2.
6. T1 ponovo zaključa sjedalo 2, upisuje mu zapamćeno prvo ime (dakle Ivan), te otključa sjedalo 2.
7. T2 ponovo zaključa sjedalo 1, upisuje mu svoje zapamćeno drugo ime (dakle Marko), te otključa sjedalo 1.
8. T2 ponovo zaključa sjedalo 2, upisuje mu svoje zapamćeno prvo ime (dakle opet Marko), te otključa sjedalo 2.



## VIŠEKORISNIČKI RAD

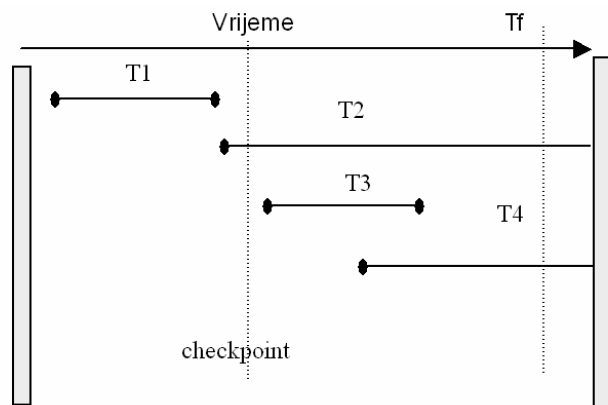
Lokoti korišteni na prethodni način nisu dovoljni!

Potrebno postrožiti pravila!

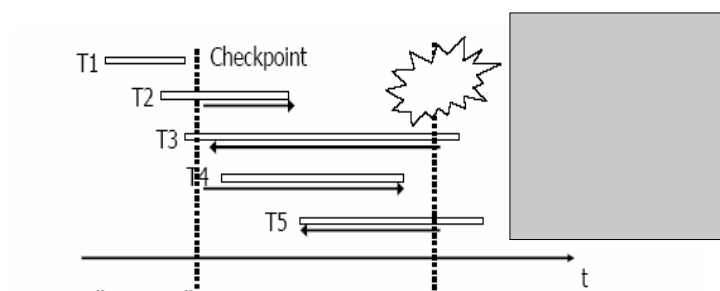
Sva zaključavanja moraju slijediti prije prvog otključavanja ⇒ dvofazni protokol zaključavanja

## VIŠEKORISNIČKI RAD

### OPORAVAK



## VIŠEKORISNIČKI RAD





## VIŠEKORISNIČKI RAD

### ZAŠTITA OD NEOVLAŠTENOG PRISTUPA

#### 1. Identifikacija korisnika

##### **ADD USER ime\_korisnika: šifra**

*ime\_korisnika (login)* – korisničko ime koje mora biti jedinstveno u bazi podataka, tj. nije dozvoljeno da postoje dva različita korisnika sa istim imenom

*šifra (password)* – sigurnosna (tajna) šifra, kojom svaki korisnik potvrđuje svoj identitet



## VIŠEKORISNIČKI RAD

### ZAŠTITA OD NEOVLAŠTENOG PRISTUPA

#### 2. Ovlaštenja

Prava pristupa pojedinim korisnicima dodjeljuju se naredbom

**GRANT nazivi\_prava ON naziv\_objekta TO ime\_korisnika**

*nazivi\_prava* – lista prava pristupa podacima

SELECT – pravo čitanja podataka

UPDATE – pravo promjene (ažuriranja) podataka

DELETE – pravo brisanja podataka

INSERT - pravo unosa podataka

*naziv\_objekta* – ime objekta (tablice ili pogleda) u bazi na koji se prava odnose

*ime\_korisnika* – naziv korisnika kojem se prava dodjeljuju



## VIŠEKORISNIČKI RAD

### ZAŠTITA OD NEOVLAŠTENOG PRISTUPA

#### 3. Pogledi kao mehanizam zaštite

RADNIK (RADNIK\_ID, IMEPREZ, ADRESA, PLACA, ODJEL\_ID)

ODJEL (ODJEL\_ID, NAZIV, MENADZER\_ID)

```
CREATE VIEW POGLED_RAD1
AS SELECT RADNIK_ID, IMEPREZ, ADRESA, ODJEL_ID
FROM RADNIK
```

```
CREATE VIEW POGLED_RAD2
AS SELECT *
FROM RADNIK
WHERE ODJEL_ID=4
```



## VIŠEKORISNIČKI RAD

### ZAŠTITA OD NEOVLAŠTENOG PRISTUPA

#### 3. Pogledi kao mehanizam zaštite

RADNIK (RADNIK\_ID, IMEPREZ, ADRESA, PLACA, ODJEL\_ID)

ODJEL (ODJEL\_ID, NAZIV, MENADZER\_ID)

```
CREATE VIEW POGLED_RAD1
AS SELECT RADNIK_ID, IMEPREZ, PLACA, ADRESA,
ODJEL.ODJEL_ID, NAZIV
FROM RADNIK, ODJEL
WHERE RADNIK.ODJEL_ID=ODJEL.ODJEL_ID
AND ODJEL.MENADZER_ID=9
```



## VIŠEKORISNIČKI RAD

### ZAŠTITA OD NEOVLAŠTENOG PRISTUPA

#### 3. Pogledi kao mehanizam zaštite

- Mijenjanje, unos i brisanje podataka u pogledu nije dozvoljeno
- ako je pogled formiran dohvaćanjem podataka iz više od jedne tablice
- ako je pogled formiran grupnim upitom (group by)
- ako je bilo koji atribut pogleda stvoren pomoću agregatne funkcije
- ako se pri formiranju pogleda koristi opcija DISTINCT
- ako je pogled formiran kao UNION query
- ako je pogled formiran pomoću složenog upita sa podupitom