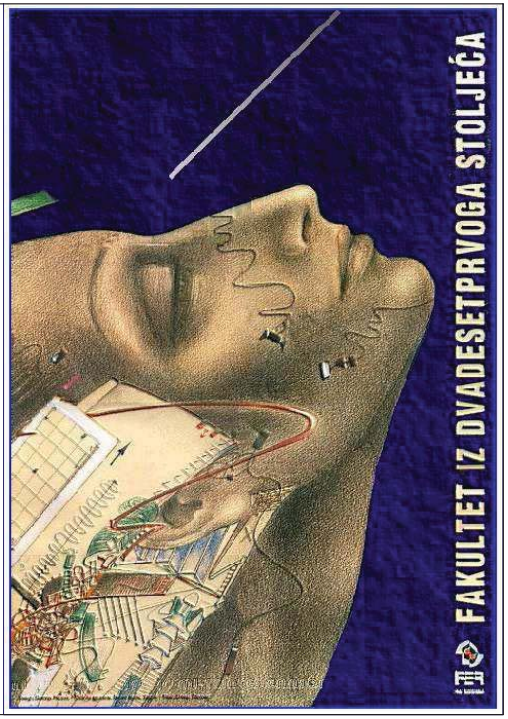
Baze podataka

Predavanja
svibanj 2009.

12. ER model baze podataka
(1. dio)



Primjer normalizacije

- Zadana je relacijska shema:
  ISPIT = { matBr, prez, ime, sifPred, nazPred, datIsp, ocj, sifNas, prezNas }
  i trenutna vrijednost relacije ispit(ISPIT):

ispit (ISPIT)

matBr	prez	ime	sifPred	nazPred	datIsp	ocj	sifNas	prezNas
1111	Novak	Ivan	1001	Mat-1	29.01.06	1	1111	Pašić
1111	Novak	Ivan	1001	Mat-1	05.02.06	3	1111	Pašić
1111	Novak	Ivan	1003	Fiz-1	28.06.06	2	3333	Horvat
1111	Novak	Ivan	1002	Mat-2	27.06.06	4	2222	Brnetić
1234	Kolar	Petar	1001	Mat-1	29.01.06	3	2222	Brnetić

- funkcijske zavisnosti odrediti na temelju značenja podataka
- odrediti primarni ključ relacije (tako da bude zadovoljen uvjet 1NF prema kojem neključni atributi funkcijski ovise o ključu)
- postupno normalizirati relacijsku shemu ISPIT na 2NF i 3NF

Primjer normalizacije

student (STUDENT)

matBr	prez	ime
1111	Novak	Ivan
1234	Kolar	Petar

$K_{STUDENT}$ = { matBr }

predmet (PREDMET)

sifPred	nazPred
1001	Mat-1
1003	Fiz-1
1002	Mat-2

$K_{PREDMET}$ = { sifPred }

nastavnik (NASTAVNIK)

sifNas	prezNas
1111	Pašić
3333	Horvat
2222	Brnetić

$K_{NASTAVNIK}$ = { sifNas }

$ispit_3$ ($ISPIT_3$)

matBr	sifPred	datIsp	ocj	sifNas
1111	1001	29.01.06	1	1111
1111	1001	05.02.06	3	1111
1111	1003	28.06.06	2	3333
1111	1002	27.06.06	4	2222
1234	1001	29.01.06	3	2222

K_{ISPIT_3} = { matBr, sifPred, datIsp }

- Shema baze podataka STUSLU:
  STUSLU = { STUDENT, PREDMET, NASTAVNIK, $ISPIT_3$ }

Implementacija: SQL

```
CREATE TABLE student (
   matBr     INTEGER
 , prez      CHAR(20)
 , ime       CHAR(20)
 , PRIMARY KEY (matBr));
```

```
CREATE TABLE predmet (
   sifPred   INTEGER
 , nazPred   CHAR(20)
 , PRIMARY KEY (sifPred));
```

```
CREATE TABLE nastavnik (
   sifNas    INTEGER
 , prezNas   CHAR(20)
 , PRIMARY KEY (sifNas));
```

```
CREATE TABLE ispit (
 , matBr     INTEGER REFERENCES student (matBr)
                       ON DELETE CASCADE
 , sifPred   INTEGER REFERENCES predmet (sifPred)
 , datIsp    DATE
 , ocj       SMALLINT CHECK (ocj BETWEEN 1 AND 5)
 , sifNas    INTEGER REFERENCES nastavnik(sifNas)
                       ON DELETE SET NULL
 , PRIMARY KEY (matBr, sifPred, datIsp));
```

OBLIKOVANJE MODELA BAZE PODATAKA

ER model *(Entity-Relationship Model)*
Model entiteti-veze

- postrelacijski model
- zadržava dobre karakteristike relacijskog modela
- omogućuje eksplicitni prikaz veza koje u sebi sadrže važne semantičke informacije

Literatura:

- P.P.Chen:
  The Entity-Relationship Model - Toward a Unified View of Data,
  ACM Transactions on Database Systems, Vol. 1, No. 1, 1976
- T. J. Teorey:
  Database Modeling & Design, Morgan Kaufmann, 1999

Entiteti, veze, uloge

Entitet

- bilo što, što ima suštinu ili bit, ima jasnoću kao činjenica ili ideja, posjeduje značajke s pomoću kojih se može razlučiti od svoje okoline

Skup entiteta E_i *(entityset)*

- Slični entiteti se grupiraju u skupove entiteta

Skup veza R_i (*relationship set*)

- matematička relacija između n entiteta:

$$R_i \subseteq E_1 \times E_2 \times E_3 \times ... \times E_n$$

ili $R_i = \{ (e_1, e_2, \ldots, e_n) \mid e_1 \in E_1 , e_2 \in E_2, ... e_n \in E_n \}$

n-torka (e_1, e_2, e_3, ... e_n), naziva se vezom.

Uloga (*role*)

- funkcija koju skup entiteta obavlja u skupu veza.

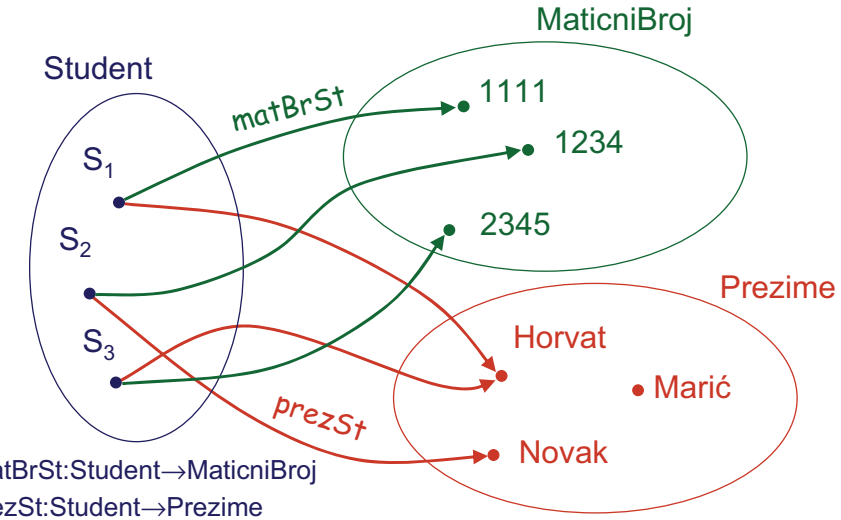
Skup vrijednosti, atribut

- Informacije o entitetu ili vezi izražavaju se s pomoću parova atribut-vrijednost
- Vrijednosti su klasificirane u skupove vrijednosti V_i.
- Atribut je funkcija koja preslikava iz skupa entiteta ili skupa veza u skup vrijednosti ili Kartezijev produkt skupova vrijednosti:

$$f : E_i \rightarrow V_i$$
$$f : E_i \rightarrow V_{i_1} \times V_{i_2} \times ... \times V_{i_n}$$
$$f : R_i \rightarrow V_i$$
$$f : R_i \rightarrow V_{i_1} \times V_{i_2} \times ... \times V_{i_n}$$

Atributi entiteta

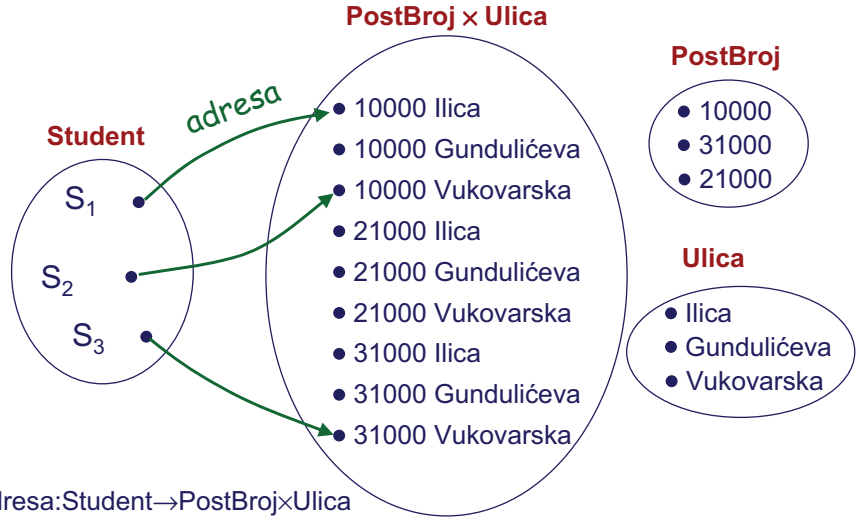
- funkcija koja preslikava sa skupa entiteta na skup vrijednosti ...



matBrSt:Student→MaticniBroj
prezSt:Student→Prezime

Atributi entiteta

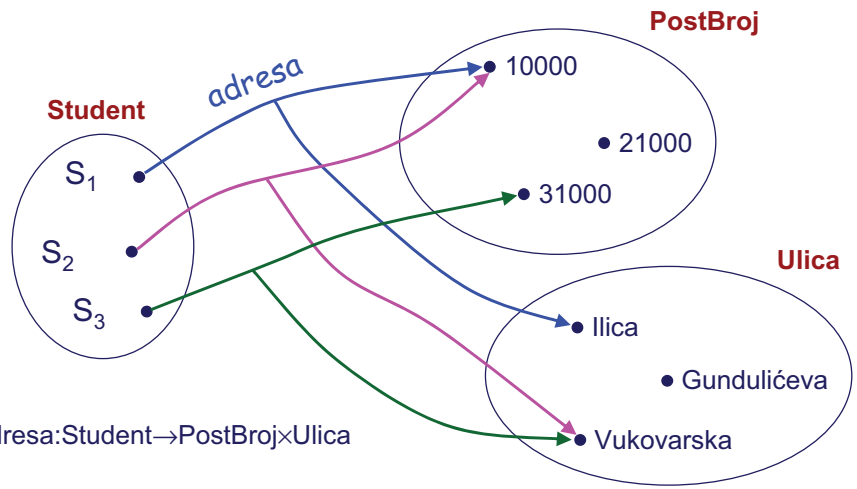
... ili na Kartezijev produkt skupova vrijednosti



adresa:Student→PostBroj×Ulica

Atributi entiteta

... ili na Kartezijev produkt skupova vrijednosti



adresa:Student→PostBroj×Ulica

Terminologija

Chen:

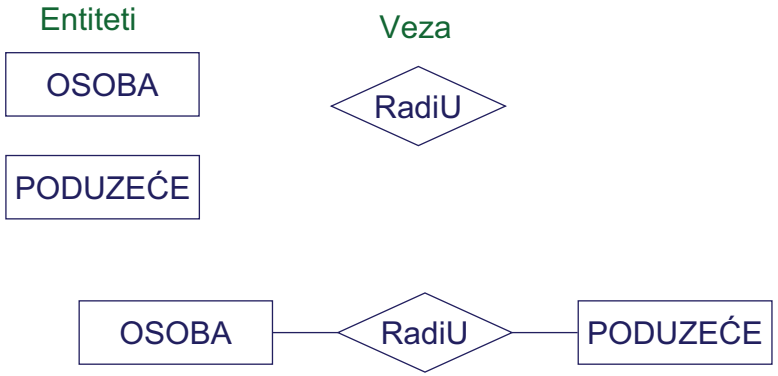
entitet, skup entiteta
veza, skup veza

Teorey:

instanca entiteta, entitet
(entity instance)
(entity occurrence)
instanca veze, veza
(relationship instance)
(relationship occurrence)

Grafički prikaz entiteta i veza

- entitet se grafički prikazuje pravokutnikom unutar kojeg se nalazi ime entiteta
- veza se grafički prikazuje rombom unutar kojeg se nalazi ime veze



Atributi entiteta

- atribut entiteta se grafički prikazuju ovalom unutar kojeg se upisuje ime atributa
- atribut (ili atributi) **primarnog** ključa se potcrtavaju

- povećanjem broja atributa, dijagram postaje nepregledan
  - atributi se tada ne prikazuju grafički - umjesto toga, uz dijagram se prilažu sheme entiteta

Shema entiteta:

NASTAVNIK = sifNast, jmbgNast, imeNast, prezNast

PK = { sifNast }

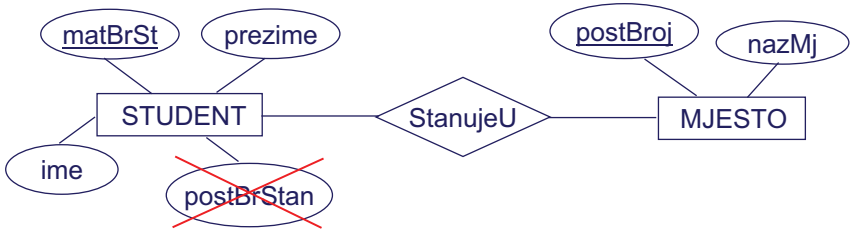
ili

NASTAVNIK
sifNast
jmbgNast
imeNast
prezNast
K_1 = { sifNast }
K_2 = { jmbgNast }
PK = K_1

Vlastiti atributi entiteta

Entiteti se opisuju samo vlastitim atributima

- vlastiti atribut entiteta je atribut koji opisuje znanja o entitetu koja se pripisuju isključivo samom entitetu, a nikako vezi s drugim entitetima



- isključivo identifikacijski slabi entiteti, osim svojih vlastitih atributa, posjeduju i atribute primarnog ključa entiteta vlasnika

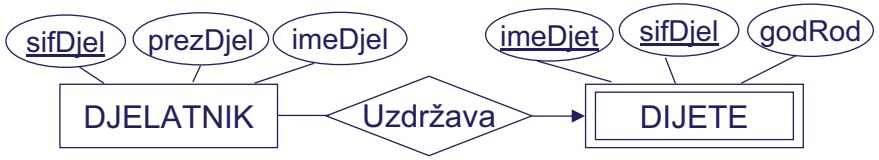
Regularni i slabi entiteti

- regularni entitet je entitet koji može postojati sam za sebe
- slabi entiteti (engl. *weak entity*) ne postoje ukoliko ne postoji i neki drugi entitet (entitet vlasnik)
- Slabi entitet se grafički prikazuje dvostruko uokvirenim pravokutnikom, sa strelicom koja dolazi iz smjera veze koja ga povezuje s entitetom vlasnikom

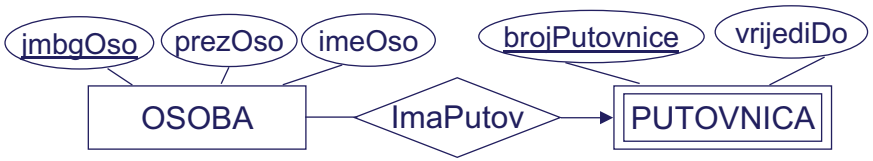
- slabi entiteti, osim što su egzistencijalno slabi, također mogu biti i identifikacijski slabi
  - kod određivanja identifikatora nisu im dovoljni vlastiti atributi
  - za identifikaciju se koriste i ključni atributi entiteta vlasnika

Identifikacijski slabi entiteti (primjer)

- entitet DIJETE, osim što je egzistencijalno slab, također je i identifikacijski slab

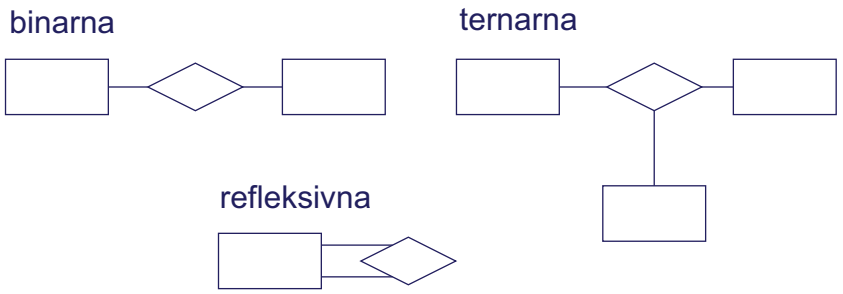


- entitet PUTOVNICA je egzistencijalno slab (nije identifikacijski slab)



Stupanj veze

- broj entiteta koje povezuje dotična veza
- veza može biti unarna(refleksivna), binarna, ternarna, itd.
  - unarna ili refleksivna veza - veza je definirana nad jednim entitetom koji u vezi ima dvije različite uloge



Spojnost veze (*connectivity*)

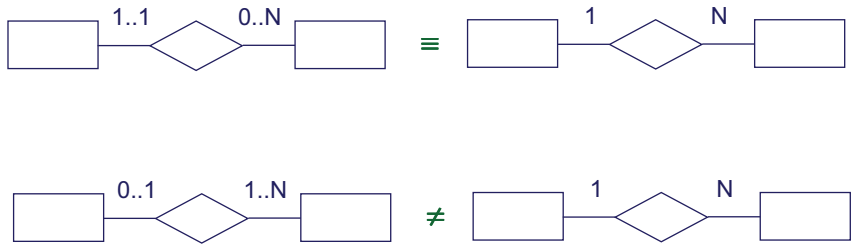
- spojnost veze opisuje ograničenje preslikavanja pojedinačnih entiteta koje veza povezuje
- vrijednosti spojnosti: jedan *(one),* više *(many)*
- koriste se oznake 1, N ili rasponi, npr. 0..1, 1..N, 1..2, itd.

- jedan djelatnik radi na jednom projektu, na jednom projektu radi N djelatnika

- jedan djelatnik radi na nula (niti jednom) ili jednom projektu, na jednom projektu radi između nula (niti jedan) i više djelatnika

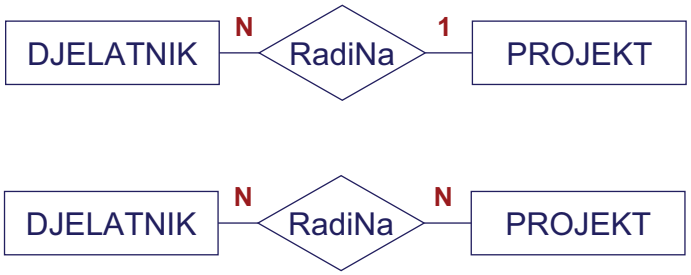
Spojnost veze (*connectivity*)

- radi pojednostavljenja
  - spojnost 0..N se često označava samo oznakom N
  - spojnost 1..1 se često označava samo oznakom 1

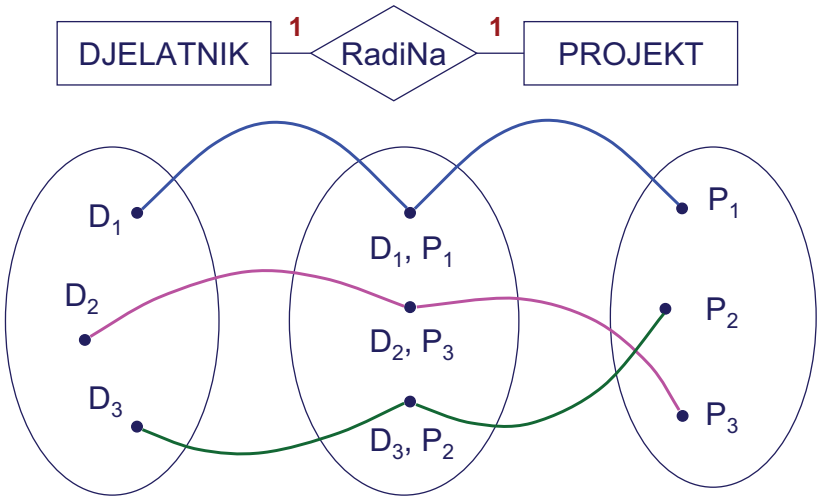


Preslikavanje (*mapping*)

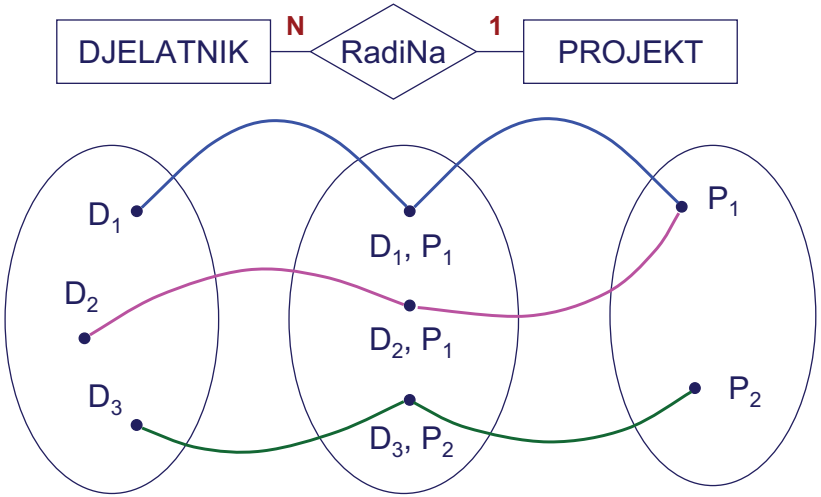
- preslikavanje - međusobni odnos entiteta u vezi
- kod binarnih veza moguća su preslikavanja 1:1 (jedan-prema-jedan), 1:N (jedan-prema-više), N:1 (više-prema-jedan), N:M (više-prema-više).



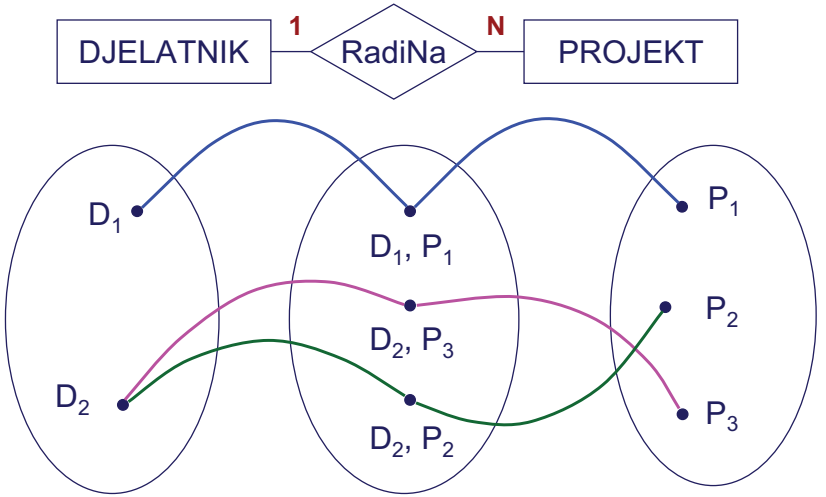
Preslikavanje 1:1



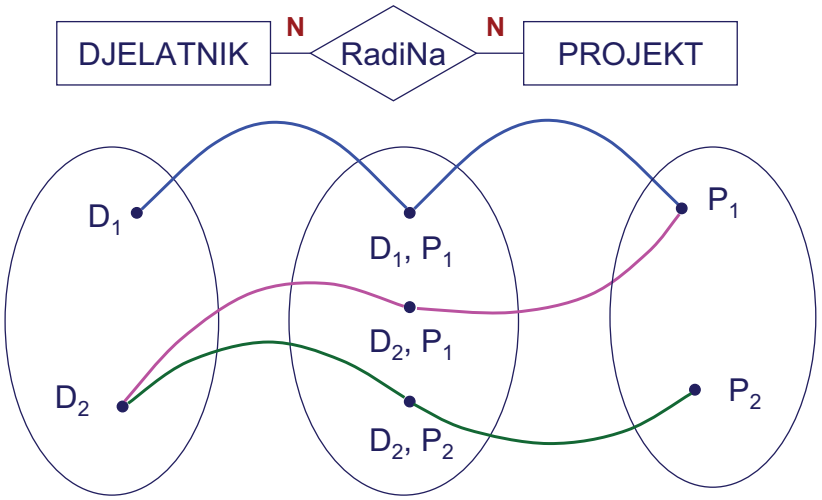
Preslikavanje N:1



Preslikavanje 1:N



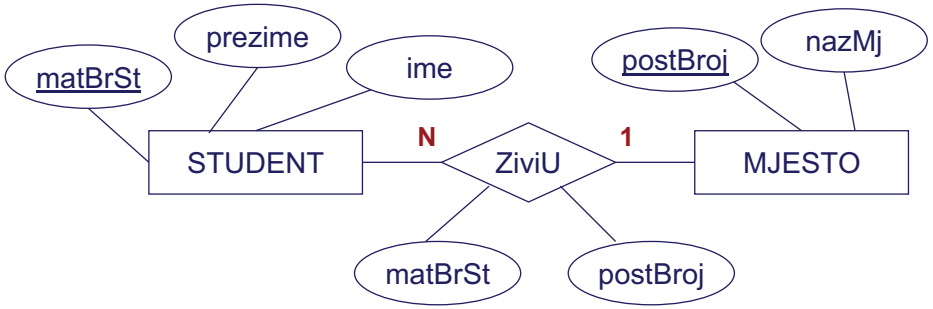
Preslikavanje N:N



Atributi veza

- Shema veze sadrži ključeve entiteta koje povezuje, te vlastite atribute
- Atribut veze se grafički prikazuje ovalom unutar kojeg se upisuje ime atributa

Atributi veza



sheme entiteta:

STUDENT = matBrSt, prezime, ime

MJESTO = postBroj, nazMj

shema veze:

ZiviU = matBrSt, postBroj

Koji atributi čine ključ veze?

Ključevi veza

- Povezanost entiteta opisuje se kao odnos među ključevima entiteta
- Ključevi veza definirani su s pomoću ključeva entiteta koje povezuju i njihovih spojnosti

Definicija 1. (Teorey)

U vezi koja povezuje entitete

$E_1, \ldots, E_k, \ldots, E_m$,

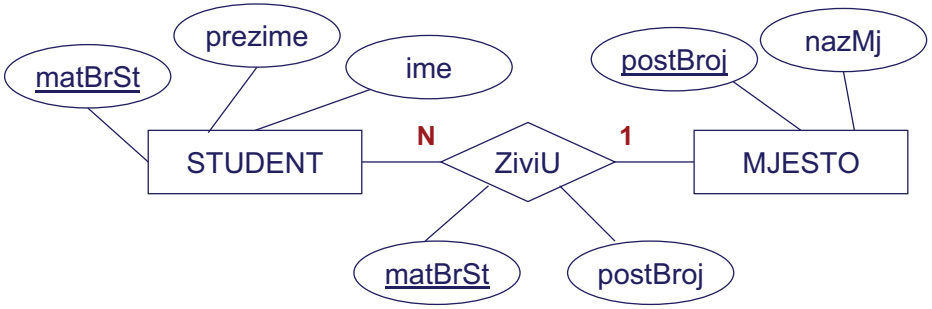
spojnost =1 entiteta E_k znači da za svaku vrijednost svih entiteta $E_1, \ldots, E_m$, osim E_k , uvijek postoji točno jedna vrijednost od E_k.

➡ može se reći da tada vrijedi funkcijska zavisnost:

$$\bigcup_{j=1}^{m} K_j \setminus K_k \rightarrow K_k$$

gdje su skupovi K_j, (j = 1, ..., m) ključevi entiteta $E_1, \ldots, E_m$

Ključevi veza



sheme entiteta:

STUDENT = matBrSt, prezime, ime

MJESTO = postBroj, nazMj

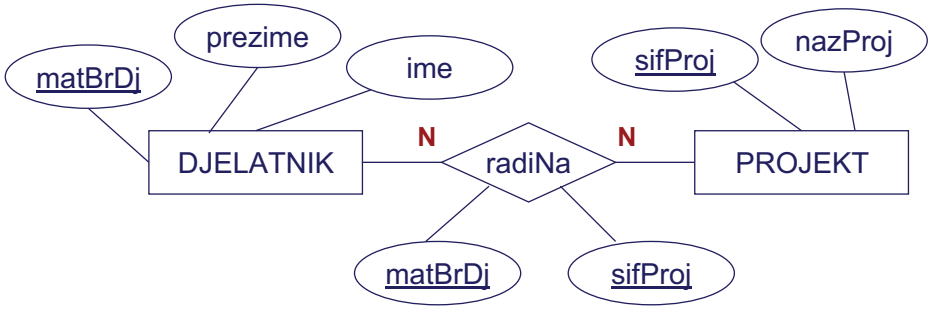
shema veze:

ZiviU = matBrSt, postBroj

Iz definicije 1:

matBrSt → postBroj

Ključevi veza

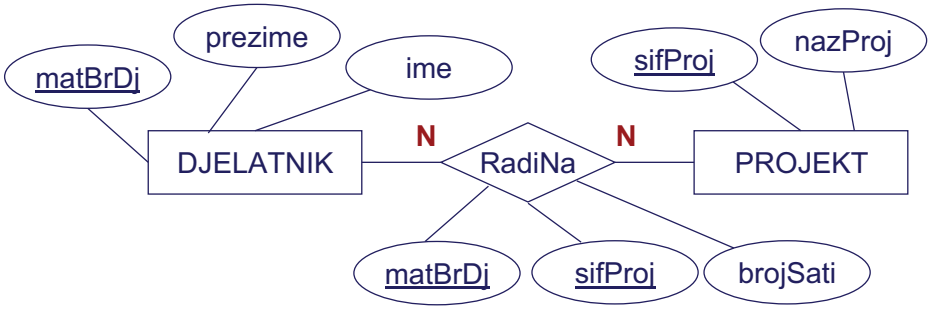


DJELATNIK = matBrDj, prezime, ime

PROJEKT = sifProj, nazProj

RadiNa = matBrDj, sifProj

Vlastiti atributi veza



DJELATNIK = matBrDj, prezime, ime

PROJEKT = sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

ključ veze funkcijski određuje vlastite atribute veze:
matBrDj, sifProj → brojSati

Ključ veze - dodatna razmatranje

- iz definicije 1. proizlazi da se ključ veze sastoji isključivo od ključeva entiteta koje povezuje (svih ili samo nekih, ovisno o spojnostima)

  Međutim, u nekim slučajevima ključ može sadržavati i neke druge atribute.

STUDENT = matBrSt, prezime, ime

PREDMET = sifPred, nazPred

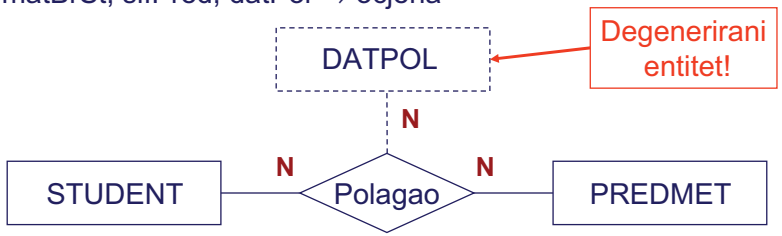
Položio = matBrSt, sifPred, ocjena

Ključ veze - dodatna razmatranje

- ako se želi evidentirati sva polaganja ispita

  matBrSt, sifPred $\not\rightarrow$ ocjena
- potrebno je uvesti atribut datPol (datum polaganja):

  matBrSt, sifPred, datPol $\rightarrow$ ocjena



STUDENT = matBrSt, prezime, ime

PREDMET = sifPred, nazPred

Polagao = matBrSt, sifPred, datPol, ocjena

Ključ veze - dodatna razmatranje

- druga mogućnost - veza postaje entitet:

STUDENT = matBrSt, prezime, ime

PREDMET = sifPred, nazPred

ISPIT = matBrSt, sifPred, datPol, ocjena

StudIsp = matBrSt, sifPred, datPol

PredIsp = matBrSt, sifPred, datPol

Veza 1:N $\rightarrow$ preslikavanje u relacijski model

DJELATNIK = matBrDj, prezime, ime
MJESTO = postBr, nazMjesto
Stanuje = matBrDj, postBr, adresa

Relacijske sheme opisuju entitete (veze postaju entiteti)

DJELATNIK = matBrDj, prezime, ime
MJESTO = postBr, nazMjesto
Stanuje = matBrDj, postBr, adresa

Unija relacijskih shema s jednakim ključevima

DJELATNIK = matBrDj, prezime, ime, postBr, adresa
MJESTO = postBr, nazMjesto

Veza N:N → preslikavanje u relacijski model

DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

Relacijske sheme opisuju entitete (veze postaju entiteti)

DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

Primjer: zašto je važno ispravno odrediti vlastite atribute entiteta i veza?

- Entiteti se opisuju samo vlastitim atributima: vlastiti atribut entiteta je atribut koji opisuje znanja o entitetu koja se pripisuju isključivo samom entitetu, a nikako vezi s drugim entitetima

DJELATNIK= matBrDj, prezime, ime, brojSati ← nije vlastiti atribut

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj

- Ako se preslikavanje promijeni u N:N

matBrDj ↛ brojSati

atribut brojSati se iz entiteta DJELATNIK mora premjestiti u vezu RadiNa

Primjer: zašto je važno ispravno odrediti vlastite atribute entiteta i veza?

DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati ← vlastiti atribut

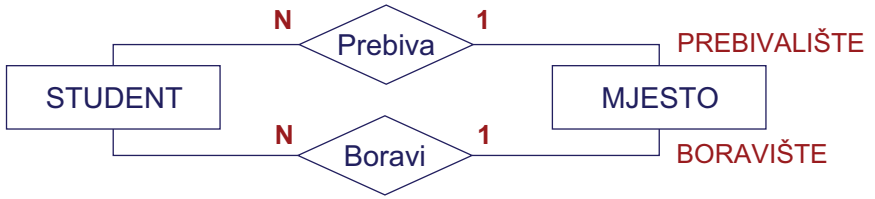
- Ako se preslikavanje promijeni u N:N

DJELATNIK= matBrDj, prezime, ime

PROJEKT= sifProj, nazProj

RadiNa = matBrDj, sifProj, brojSati

Paralelne veze



STUDENT = matBrSt, prezime, ime

MJESTO = postBroj, nazMjesto

Uloge: PREBIVALIŠTE
BORAVIŠTE

Prebiva = matBrSt, postBroj PostBrojPreb

Boravi = matBrSt, postBroj PostBrojBor

Paralelne veze → relacijski model

- Unija shema s jednakim ključevima:

  MJESTO = postBroj, nazMjesto

  STUDENT = matBrSt, prezime, ime, ~~postBroj, postBroj~~

  STUDENT = matBrSt, prezime, ime, postBrojBor, postBrojPreb
  + pravila integriteta

Zadatak: Ispisati prezime i ime studenta, poštanski broj i naziv mjesta boravka te poštanski broj i naziv mjesta prebivališta

```
SELECT student.*
     , boraviste.nazMjesto AS nazMjestoBoraviste
     , prebivaliste.nazMjesto AS nazMjestoPrebivaliste
  FROM student
       INNER JOIN mjesto AS boraviste
             ON boraviste.postBroj = student.postBrojBor
       INNER JOIN mjesto AS prebivaliste
             ON prebivaliste.postBroj = student.postBrojPreb
```

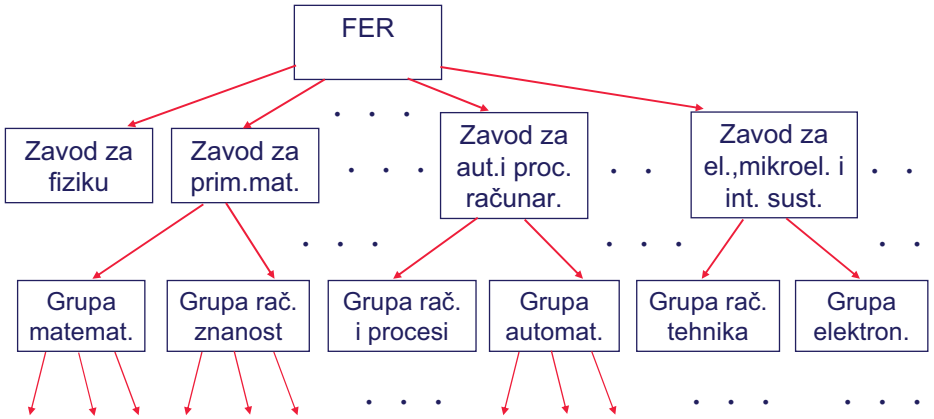

Problem

Kako opisati organizacijsku strukturu poduzeća?

- Organizacijske jedinice opisane su svojom šifrom i nazivom
- Organizacijske jedinice međusobno su povezane
  - **kako?**
    - ➡ među njima postoji hijerarhijski odnos!
  - **kolika je dubina stabla (broj razina)?**
    - ➡ promjenjiva!
- Kako opisati hijerarhiju - stablo promjenjive dubine?
- Čvorovi stabla su opisani na isti način (šifra, naziv)

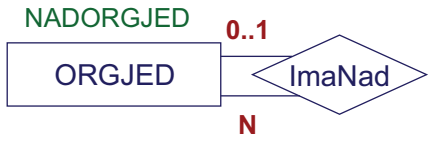
Homogeno stablo

Primjer: Organizacijska struktura



Čvorovi stabla imaju jednaku strukturu: ORGJED= sifOrgJed, nazOrgJed

Refleksivne veze - preslikavanje 1:N



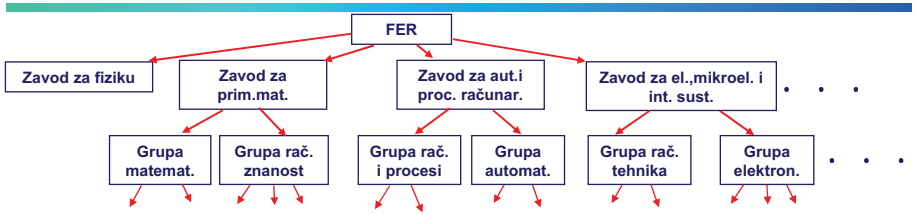
ORGJED = sifOrgJed, nazOrgJed

ImaNad = ~~sifOrgJed, sifOrgJed~~

ImaNad = sifOrgJed, sifNadOrgJed

Preimenovati jedan od atributa !

Homogeno stablo



ORGJED

sifOrgJed	nazOrgJed
1	FER
9	Zavod za prim.mat.
21	Grupa Matematika
33	Grupa Rač. Znanost
49	Zavod za aut. i proc. rač.
53	Grupa Automatika
67	Grupa RASIP
73	Zavod za el.mikroel. i int.
89	Grupa Rač. tehnika

imaNad

sifOrgJed	sifNadOrgJed
9	1
21	9
33	9
49	1
53	49
67	49
73	1
89	73

Refleksivne veze 1:N → relacijski model

- Unija shema s jednakim ključevima:

ORGJED = sifOrgJed, nazOrgJed
imaNad = sifOrgJed, sifNadOrgJed

ORGJED = sifOrgJed, nazOrgJed, sifNadOrgJed
+ pravila integriteta

Zadatak: Ispisati naziv organizacijske jedinice i naziv njezine nadređene organizacijske jedinice (ukoliko postoji)

```
SELECT orgjed.nazOrgJed, nadorgjed.nazOrgJed
   FROM orgjed
        LEFT OUTER JOIN orgjed AS nadorgjed
          ON orgjed.sifNadOrgJed = nadorgjed.sifOrgJed
```


Što je šifra organizacijske jedinice?

- Govoreća šifra – šifra koja označava poziciju organizacijske jedinice unutar poduzeća??

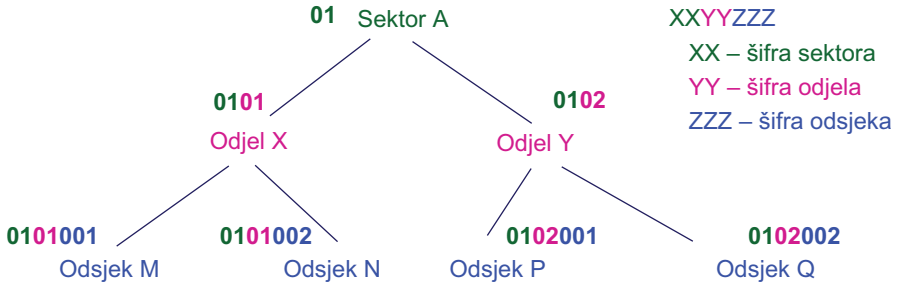
  npr. XXYYZZZ
  - XX – šifra sektora
  - YY – šifra odjela
  - ZZZ – šifra odsjeka

  ➡ **što se dešava prilikom reorganizacije?**
  - ➡ moraju se promijeniti šifre organizacijskih jedinica!

  ➡ **što se dešava kada broj odjela preraste 100??**
  - ➡ moraju se promijeniti šifre organizacijskih jedinica!

- ➡ Šifra organizacijske jedinice NE SMIJE BITI GOVOREĆA!
- ➡ To vrijedi i za sve ostale šifre i identifikatore!!!

Što je šifra organizacijske jedinice?



ORGJED

sifOrgJed	nazOrgJed
01	Sektor A
0101	Odjel X
0102	Odjel Y
0101001	Odsjek M
0101002	Odsjek N
0102001	Odsjek P
0102002	Odsjek Q

Što kada Odsjek P zbog reorganizacije iz Odjela Y preseli u Odjel X?

Što kada broj odjela preraste broj 99?

Oblikovanje ER modela

Oblikovanje ER modela

- **definiranje entiteta**
  - ime, opis, komentar
- **definiranje veza**
  - ime, opis, komentar, entiteti koje povezuje, preslikavanje
- **definiranje atributa entiteta**
  - za svaki atribut: ime, opis, komentar, domena
  - definirati ključeve, provjeriti da li zadovoljava 3NF
- **definiranje atributa veza**
  - za svaki atribut: ime, opis, komentar, domena
    - definirati ključeve, provjeriti da li zadovoljava 3NF

POSTUPAK JE ITERATIVAN!

Model baze podataka

SADRŽI OPISE
- entiteta
- veza
- atributa entiteta
- atributa veza

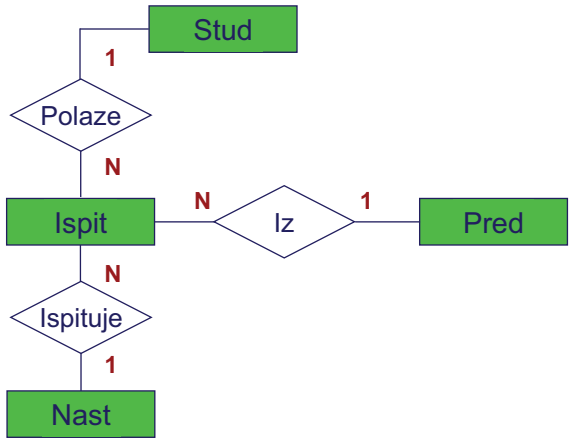
KARAKTERISTIKE DOBROG MODELA
- opisuje suštinu, prirodu stvari, neovisan o postojećem stanju
- sveobuhvatan
- neredundantan
- fleksibilan
- razumljiv - korisnicima i informatičarima

POSEBNO OBRATITI PAŽNJU NA:
- različito shvaćanje istih stvari - kupac, dobavljač ➡ poslovni partner
- praćenje promjena u vremenu - stipendist, djelatnik, penzioner
- jednakost - uopćavanje - različiti odjeli i pojedinci mogu iste ili slične stvari shvaćati različito

Primjer: Model baze podataka za studentsku službu

- Oblikovati model baze podataka koja će omogućiti praćenje podataka o studentima, predmetima, nastavnicima i polaganjima ispita

Primjer (nastavak)

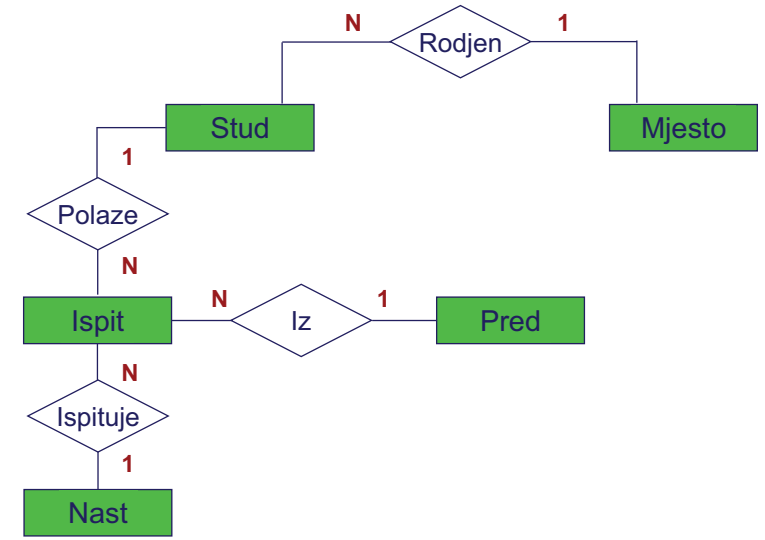


Primjer (nastavak)

Stud = matBrStud, prezStud, imeStud, datRodStud

MJESTO ROĐENJA STUDENTA ???

Primjer (nastavak)



Primjer (nastavak)

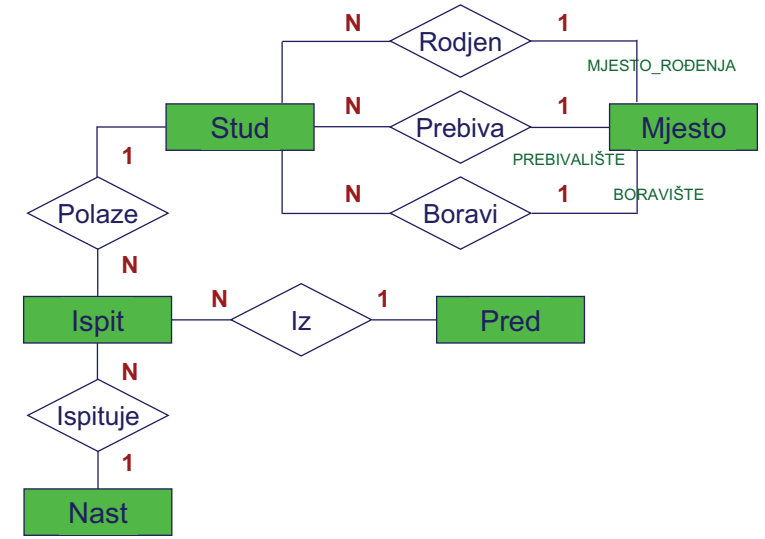
Stud = matBrStud, prezStud, imeStud, datRodStud

Mjesto = pbrMjesto, nazMjesto

PREBIVALIŠTE STUDENTA ???

BORAVIŠTE STUDENTA ???

Primjer (nastavak)



Primjer (nastavak)

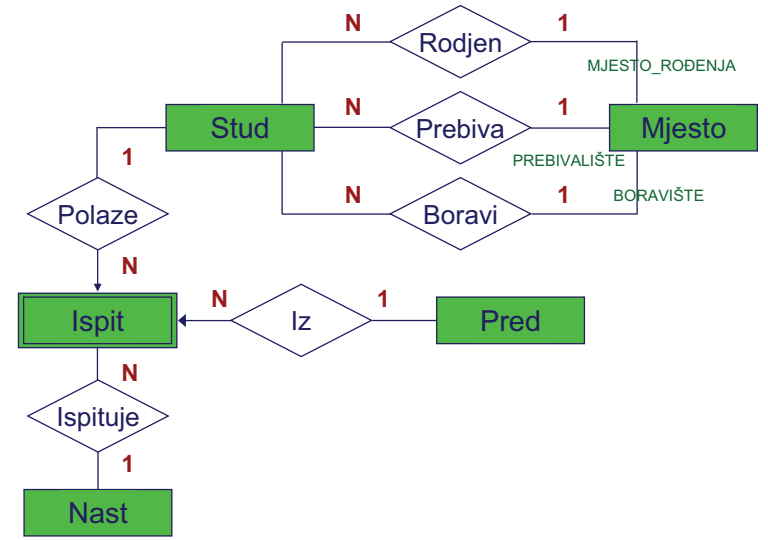
Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud, rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

SLABI ENTITET !!!!

Primjer (nastavak)



Primjer (nastavak)

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud, rangKlasIspitStud, eMailStud

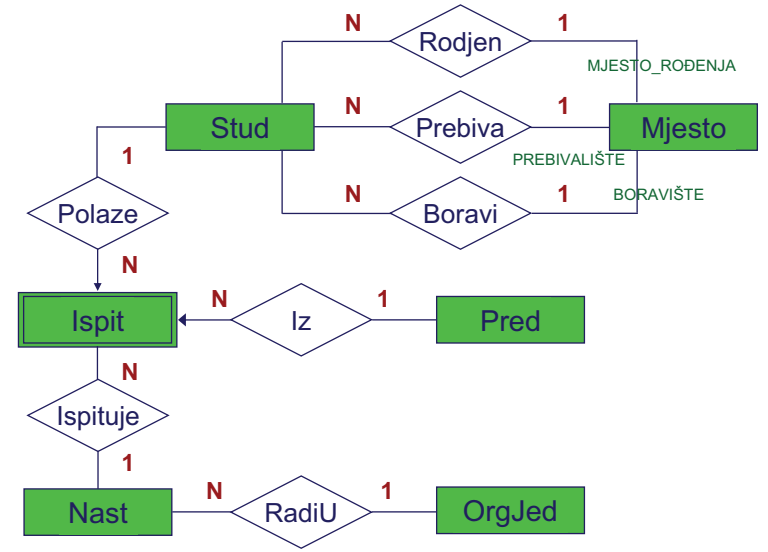
Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

Nast = sifraNast, prezNast, imeNast

ORGANIZACIJSKA JEDINICA ???

Primjer (nastavak)



Primjer (nastavak)

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud, rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

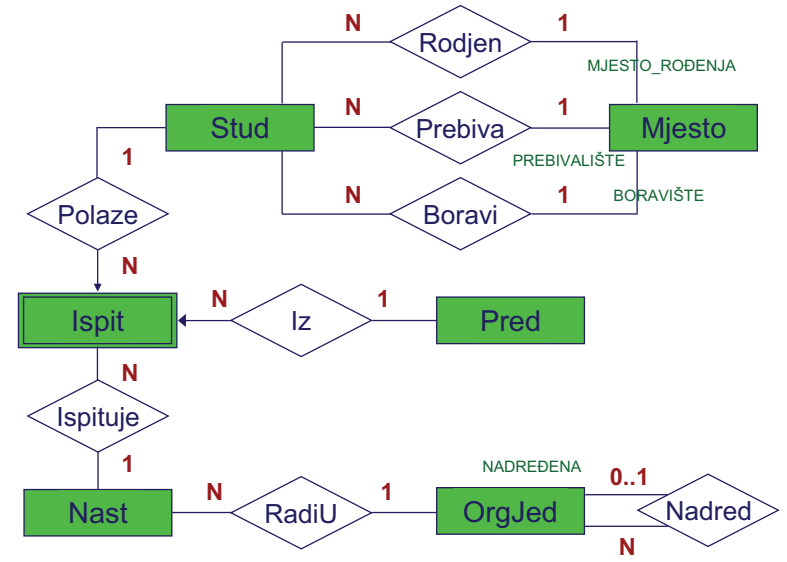
Ispit = matBrStud, sifraPred, datumIspit, ocjena

Nast = sifraNast, prezNast, imeNast

OrgJed = sifraOrgJed, nazivOrgJed

NADREĐENA ORGANIZACIJSKA JEDINICA ???

Primjer (nastavak)



Primjer (nastavak)

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud, rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

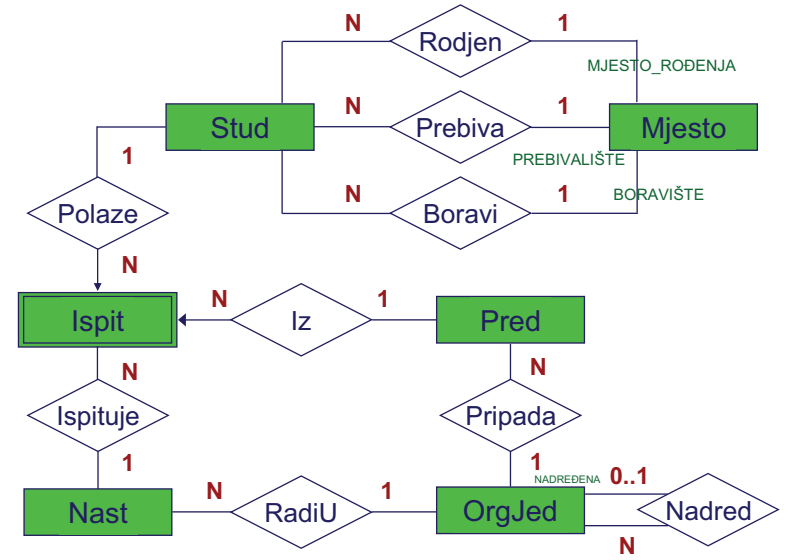
Nast = sifraNast, prezNast, imeNast, eMailNast, URLNast

OrgJed = sifraOrgJed, nazivOrgJed

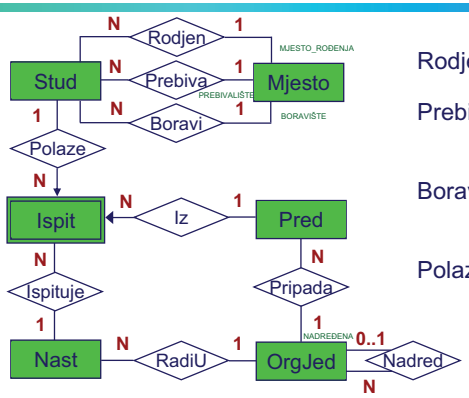
Pred = sifraPred, kraticaPred, nazivPred, URLPred

PREDMET PRIPADA ORGANIZACIJSKOJ JEDINICI ???

Primjer (nastavak)



Primjer (nastavak) - Opis veza



Rodjen = matBrStud, postBrMjRodStud

Prebiva = matBrStud, postBrMjPrebStud, adresaMjPrebStud

Boravi = matBrStud, postBrMjBorStud, adresaMjBorStud

Polaze = matBrStud, sifraPred, datumIspit

Iz = matBrStud, sifraPred, datumIspit

Ispituje = matBrStud, sifraPred, datumIspit, sifraNast

RadiU = sifraNast, sifraOrgJed

Pripada = sifraPred, sifraOrgJed

Nadred = sifraOrgJed ,sifraNadOrgJed

→ Relacijski model

Stud = matBrStud, prezStud, imeStud, datRodStud, datUpisFERStud, rangKlasIspitStud, eMailStud

Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena

Nast = sifraNast, prezNast, imeNast, eMailNast, URLNast

OrgJed = sifraOrgJed, nazivOrgJed

Pred = sifraPred, kraticaPred, nazivPred, URLPred

Rodjen = matBrStud, postBrMjRodStud

Prebiva = matBrStud, postBrMjPrebStud, adresaMjPrebStud

Boravi = matBrStud, postBrMjBorStud, adresaMjBorStud

Polaze = matBrStud, sifraPred, datumIspit

Iz = matBrStud, sifraPred, datumIspit

Ispituje = matBrStud, sifraPred, datumIspit ,sifraNast

RadiU = sifraNast ,sifraOrgJed

Pripada = sifraPred ,sifraOrgJed

Nadred = sifraOrgJed ,sifraNadOrgJed

→ Relacijski model

Unija shema s jednakim ključevima

Stud = matBrStud, prezStud, imeStud, datRodStud,datUpisFERStud, rangKlasIspitStud, eMailStud, postBrMjRodStud, postBrMjPrebStud, adresaMjPrebStud, postBrMjBorStud, adresaMjBorStud

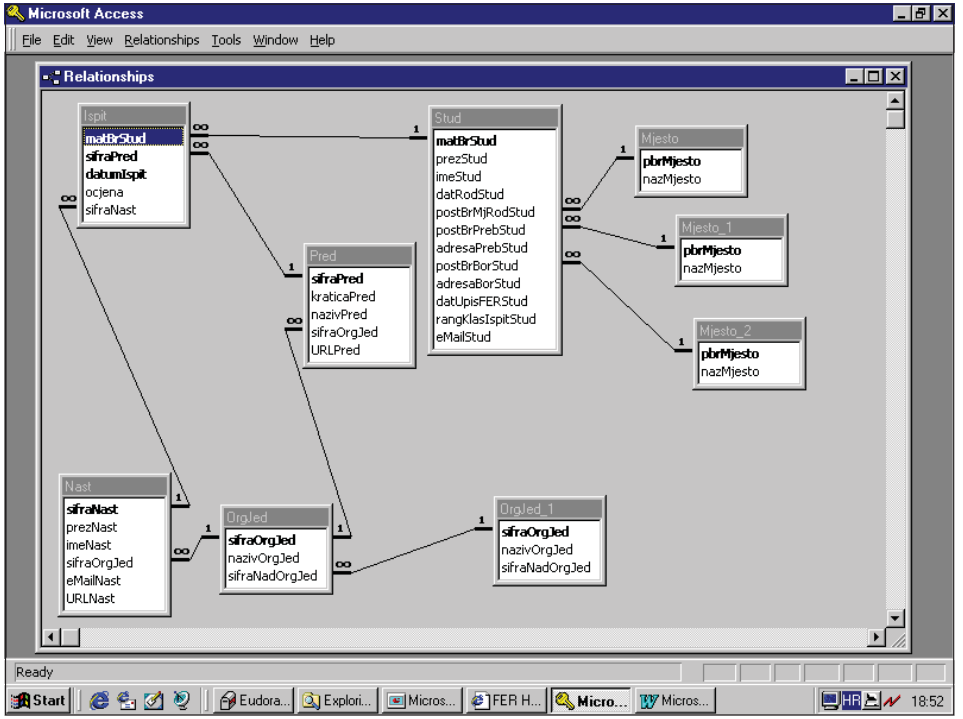
Mjesto = pbrMjesto, nazMjesto

Ispit = matBrStud, sifraPred, datumIspit, ocjena, sifraNast

Nast = sifraNast, prezNast, imeNast, eMailNast, URLNast, sifraOrgJed

OrgJed = sifraOrgJed, nazivOrgJed, sifraNadOrgJed

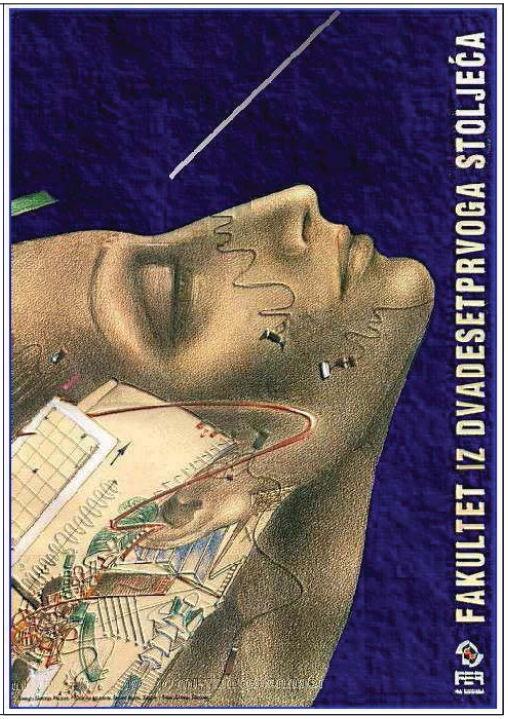
Pred = sifraPred, kraticaPred, nazivPred, URLPred, sifraOrgJed



Baze podataka

Predavanja
lipanj 2009.

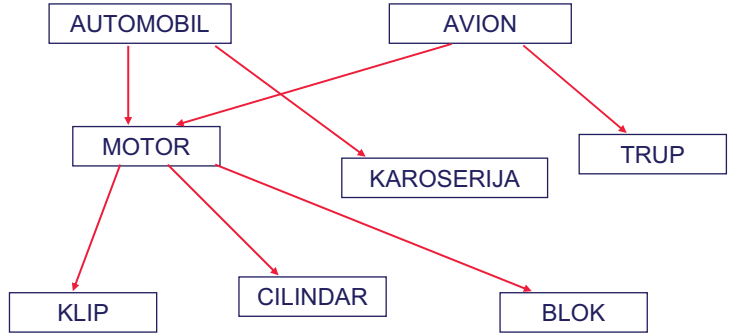
13. ER model baze podataka
(2. dio)



Homogena mreža

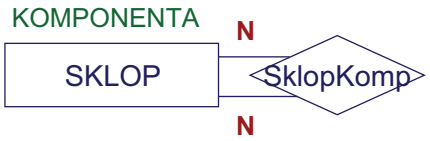
Primjer: Sastavnica

AUTOMOBIL: MOTOR, KAROSERIJA
MOTOR: KLIP, CILINDAR, BLOK
AVION: MOTOR, TRUP



Čvorovi u mreži imaju jednaku strukturu: SKLOP= sifSklop, nazSklop

Refleksivne veze - preslikavanje N:N



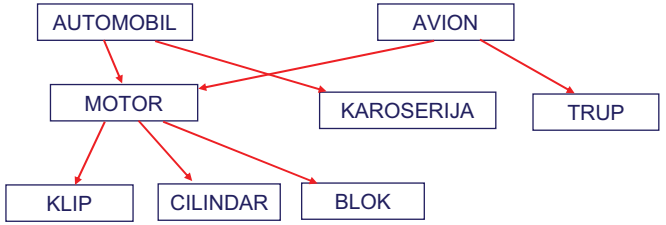
SKLOP = sifSklop, nazSklop

SklopKomp = ~~sifSklop, sifSklop~~

SklopKomp = sifSklop, sifKomp

Preimenovati jedan od atributa !

Refleksivne veze - preslikavanje N:N



Sklop

sifSklop	nazSklop
17	Automobil
19	Motor
21	Karoserija
37	Klip
49	Cilindar
52	Blok
64	Avion
82	Trup

sklopKomp

sifSklop	sifKomp
17	19
17	21
19	37
19	49
19	52
64	19
64	82

Refleksivne veze N:N → relacijski model

SKLOP = sifSklop, nazSklop

SklopKomp = sifSklop, sifKomp

+ pravila integriteta

Zadatak: Ispisati naziv sklopa i naziv komponenti od kojih se sklop sastoji (ukoliko komponente sklopa postoje)

```
SELECT sklop.nazSklop
     , komponenta.nazSklop AS nazKomponenta
  FROM sklop AS komponenta
       INNER JOIN sklopKomp
         ON komponenta.sifSklop = sklopKomp.sifKomp
       RIGHT OUTER JOIN sklop
         ON sklopKomp.sifSklop = sklop.sifSklop;
```

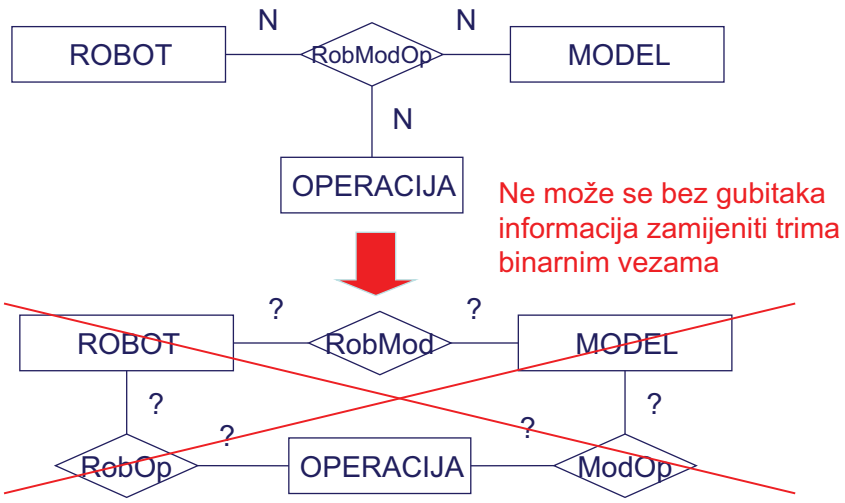

Problem

Model proizvodnje:

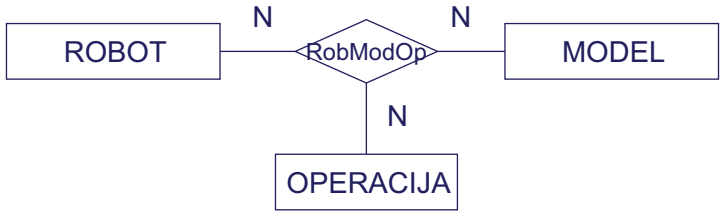
- robot R1 montira prednja lijeva vrata na modelu automobila Volvo S40 za 45 sekundi i pri tome utroši 0.8 kWh energije
- robot R2 oboji poklopac motora na modelu automobila Volvo S40 za 28 sekundi i pri tome utroši 0.4 kWh energije
- robot R1 montira prednja lijeva vrata na modelu automobila Volvo S60 za 52 sekunde i pri tome utroši 0.9 kWh energije
- robot R1 montira poklopac motora na modelu automobila Volvo S40 za 25 sekundi i pri tome utroši 0.75 kWh energije
- robot R2 montira prednja lijeva vrata na modelu automobila Volvo S40 za 40 sekundi i pri tome utroši 0.6 kWh energije
- robot R2 montira poklopac motora na modelu automobila Volvo S40 za 18 sekundi i pri tome utroši 0.7 kWh energije

Ternarne veze

Ternarnom vezom prikazuje se istovremeni odnos triju entiteta.



Ternarne veze - preslikavanje N:N:N



ROBOT = sifRobot, nazRobot, …

MODEL = sifModel, nazModel, …

OPERACIJA = sifOper, nazOper, …

RobModOp = sifRobot, sifModel, sifOper, utrVrijeme, utrEnergija

Ternarne veze N:N:N → relacijski model

ROBOT = sifRobot, nazRobot, …

MODEL = sifModel, nazModel, …

OPERACIJA = sifOper, nazOper, …

+ pravila integriteta

RobModOp = sifRobot, sifModel, sifOper, utrVrijeme, utrEnergija

Zadatak: Ispisati naziv robota, naziv modela automobila, naziv operacije, utrošak vremena i energije, za sve operacije koje roboti mogu obaviti

```
SELECT nazRobot, nazModel, nazOper, utrVrijeme, utrEnergija
  FROM robModOp
       INNER JOIN robot
         ON robModOp.sifRobot = robot.sifRobot
       INNER JOIN model
         ON robModOp.sifModel = model.sifModel
       INNER JOIN operacija
         ON robModOp.sifOper = operacija.sifOper;
```


Definicija 1. (Teorey)

U vezi koja povezuje entitete

$E_1, \ldots, E_k, \ldots, E_m$,

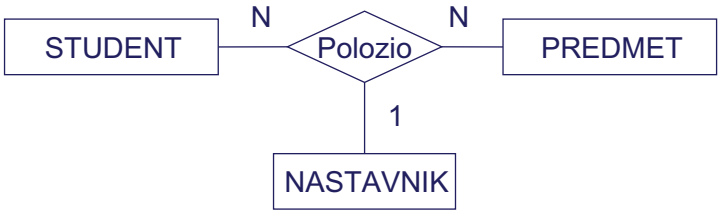
spojnost =1 entiteta E_k znači da za svaku vrijednost svih entiteta $E_1, \ldots, E_m$, osim E_k , uvijek postoji točno jedna vrijednost od E_k.

➡ može se reći da tada vrijedi funkcijska zavisnost:

$$\bigcup_{j=1}^{m} K_j \setminus K_k \rightarrow K_k$$

gdje su skupovi K_j, (j = 1, ..., m) ključevi entiteta $E_1, \ldots, E_m$

Ternarne veze - preslikavanje N:N:1



STUDENT = matBrSt, prezSt, imeSt

PREDMET = sifPred, nazPred

NASTAVNIK = sifNast, prezNast, imeNast

Polozio = matBrSt, sifPred, sifNast, ocjena

Ternarne veze N:N:1 → relacijski model

STUDENT = matBrSt, prezSt, imeSt

PREDMET = sifPred, nazPred

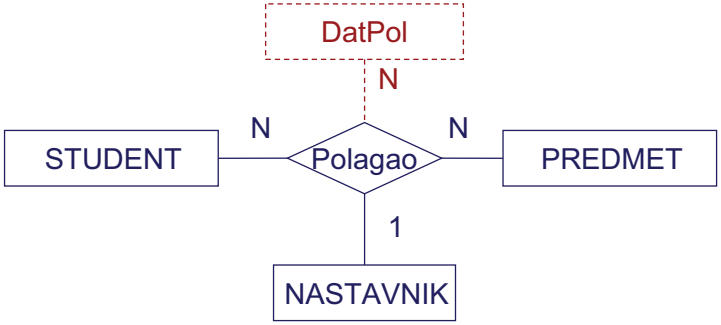
NASTAVNIK = sifNast, prezNast, imeNast

Polozio = matBrSt, sifPred, sifNast, ocjena

+ pravila integriteta

```
SELECT prezSt, imeSt, nazPred, prezNast, imeNast, ocjena
  FROM polozio
       INNER JOIN student
         ON polozio.matBrSt = student.matBrSt
       INNER JOIN predmet
         ON polozio.sifPred = predmet.sifPred
       INNER JOIN nastavnik
         ON polozio.sifNast = nastavnik.sifNast;
```

Ternarne veze - preslikavanje N:N:1



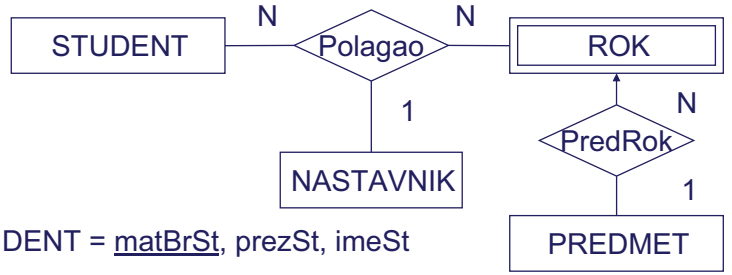
STUDENT = matBrSt, prezSt, imeSt

PREDMET = sifPred, nazPred

NASTAVNIK = sifNast, prezNast, imeNast

Polagao = matBrSt, sifPred, datPol, sifNast, ocjena

Ternarne veze - preslikavanje N:N:1



STUDENT = matBrSt, prezSt, imeSt

PREDMET = sifPred, nazPred

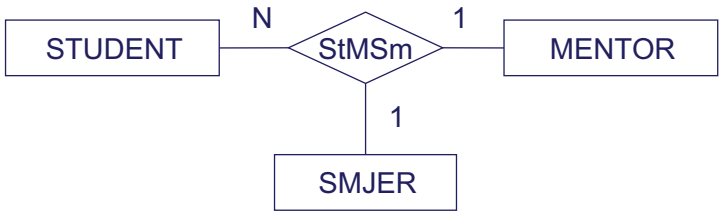
ROK = sifPred, datRok, vrstaRok

NASTAVNIK = sifNast, prezNast, imeNast

PredRok = sifPred, datRok

Polagao = matBrSt, sifPred, datRok, sifNast, ocjena

Ternarne veze - preslikavanje N:1:1



Student može studirati na više smjerova, ali na svakom smjeru mora imati različitog mentora. Student na svakom smjeru ima samo jednog mentora.

STUDENT = matBrSt, prezSt, imeSt

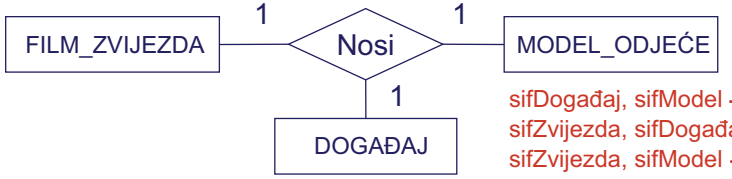
SMJER = sifSmjer, nazSmjer

MENTOR = sifMentor, prezMentor, imeMentor

StMSm = sifSmjer, matBrSt, sifMentor

Ternarne veze - preslikavanje 1:1:1

Na jednom događaju (npr. Dodjela Oscara 2008.) ne smiju dvije ili više filmskih zvijezda nositi isti model odjeće. Tijekom jednog događaja, jedna zvijezda nosi samo jedan model odjeće. Jedna zvijezda smije jedan model odjeće nositi na samo jednom događaju.



sifDogađaj, sifModel → sifZvijezda
sifZvijezda, sifDogađaj → sifModel
sifZvijezda, sifModel → sifDogađaj

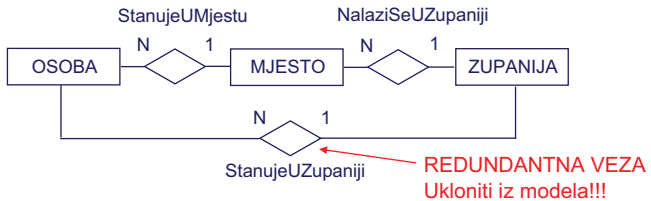
FILM_ZVIJEZDA = sifZvijezda, ime, prezime

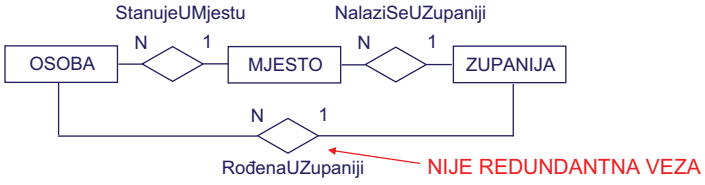
MODEL_ODJEĆE = sifModel, nazModel

DOGAĐAJ = sifDogađaj, nazDogađaj, datumDogađaj

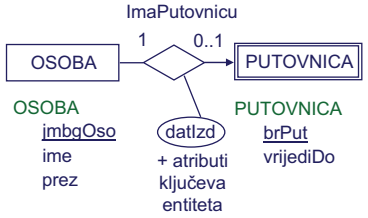
Nosi = sifZvijezda, sifDogađaj, sifModel

Redundantne veze





Primjeri preslikavanja u relacijski model

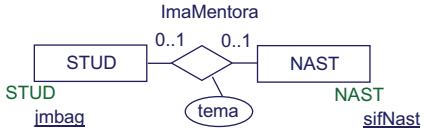



```
CREATE TABLE osoba (
  jmbgOso   ...
, ime       ...
, prez      ...
, PRIMARY KEY (jmbgOso));
```

```
CREATE TABLE putovnica (
  brPut      ...
, vrijediDo  ...
, datIzd     ...
, jmbgOso    ... NOT NULL
, PRIMARY KEY (brPut)
, UNIQUE (jmbgOso)
, FOREIGN KEY (jmbgOso)
    REFERENCES osoba (jmbgOso));
```

ImaPutovnicu
brPut
jmbgOso – alternativni ključ
datIzd

Primjeri preslikavanja u relacijski model



ImaMentora
jmbag
sifNast – alternativni ključ
tema

Pretpostavlja se da jedan nastavnik može biti mentor najviše jednom studentu (ili niti jednom), te da student može imati najviše jednog mentora (ili niti jednog).

ILI

ImaMentora
sifNast
jmbag – alternativni ključ
tema

```
CREATE TABLE stud (
  jmbag      ...
, imeS       ...
, prezS      ...
, sifNast    ...
, tema       ...
, PRIMARY KEY (jmbag)
, FOREIGN KEY (sifNast)
    REFERENCES nast(sifNast));
```

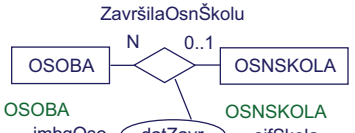
```
CREATE TABLE nast (
  sifNast    ...
, imeN       ...
, prezN      ...
, PRIMARY KEY (sifNast));
```

ILI

```
CREATE TABLE stud (
  jmbag      ...
, imeS       ...
, prezS      ...
, PRIMARY KEY (jmbag));
```

```
CREATE TABLE nast (
  sifNast    ...
, imeN       ...
, prezN      ...
, jmbag      ...
, tema       ...
, PRIMARY KEY (sifNast)
, FOREIGN KEY (jmbag)
    REFERENCES stud (jmbag));
```


Primjeri preslikavanja u relacijski model

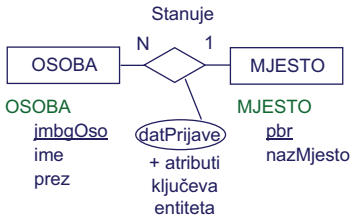


ZavršilaOsnŠkolu
jmbgOso
sifSkola
datZavr

```
CREATE TABLE osoba (
  jmbgOso   ...
, ime       ...
, prez      ...
, sifSkola  ...
, datZavr   ...
, PRIMARY KEY (jmbgOso)
, FOREIGN KEY (sifSkola)
    REFERENCES osnSkola(sifSkola));
```

```
CREATE TABLE osnSkola (
  sifSkola   ...
, nazSkola   ...
, adrSkola   ...
, PRIMARY KEY (sifSkola));
```

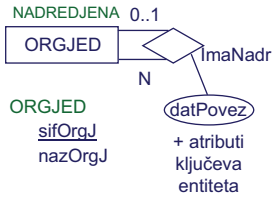
Primjeri preslikavanja u relacijski model




```
CREATE TABLE mjesto (
  pbr        ...
, nazMjesto ...
, PRIMARY KEY (pbr));
```

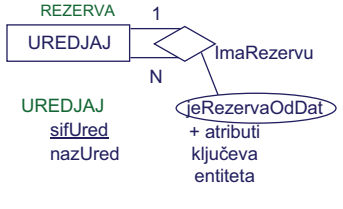
```
CREATE TABLE osoba (
  jmbgOso     ...
, ime         ...
, prez        ...
, pbrStan     ... NOT NULL
, datPrijave ...
, PRIMARY KEY (jmbgOso)
, FOREIGN KEY (pbrStan)
    REFERENCES mjesto(pbr));
```


Primjeri preslikavanja u relacijski model




```
CREATE TABLE orgJed (
  sifOrgJ     ...
, nazOrgJ     ...
, sifNadOrgJ ...
, datPovez    ...
, PRIMARY KEY (sifOrgJ)
, FOREIGN KEY (sifNadOrgJ)
    REFERENCES orgjed (sifOrgJ));
```

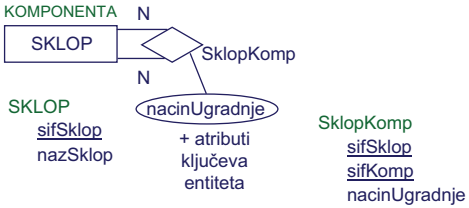
Za svaki uređaj mora biti evidentiran točno jedan rezervni uređaj. Jedan uređaj može biti rezerva za nekoliko uređaja. Evidentira se datum kad je uređaj postao rezerva nekog drugog uređaja




```
CREATE TABLE uredjaj (
  sifUred         ...
, nazUred         ...
, sifRezUred      ... NOT NULL
, jeRezervaOdDat ...
, PRIMARY KEY (sifUred)
, FOREIGN KEY (sifRezUred)
    REFERENCES uredjaj (sifUred));
```

Zadatak: kako osigurati da uređaj ne bude sam sebi rezerva?

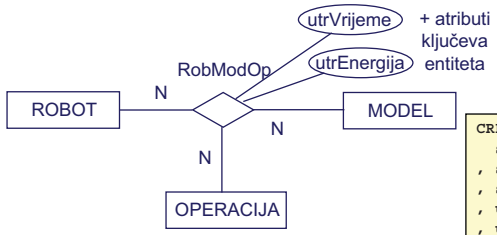
Primjeri preslikavanja u relacijski model




```
CREATE TABLE sklop (
  sifSklop    ...
, nazSklop    ...
, PRIMARY KEY (sifSklop));
```

```
CREATE TABLE sklopKomp (
  sifSklop      ...
, sifKomp       ...
, nacinUgradnje ...
, PRIMARY KEY (sifSklop, sifKomp)
, FOREIGN KEY (sifSklop)
    REFERENCES sklop(sifSklop)
, FOREIGN KEY (sifKomp)
    REFERENCES sklop (sifSklop));
```


Primjeri preslikavanja u relacijski model



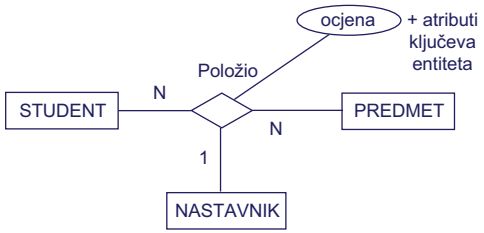

```
CREATE TABLE robModOp (
  sifRobot      ...
, sifModel      ...
, sifOper       ...
, utrVrijeme    ...
, utrEnergija   ...
, PRIMARY KEY (sifRobot, sifModel, sifOper)
, FOREIGN KEY (sifRobot)
    REFERENCES robot (sifRobot)
, FOREIGN KEY (sifModel)
    REFERENCES model(sifModel)
, FOREIGN KEY (sifOper)
    REFERENCES operacija (sifOper));
```

```
CREATE TABLE robot (
  sifRobot        ...
, nazRobot        ...
, PRIMARY KEY (sifRobot));
```

```
CREATE TABLE model (
  sifModel      ...
, nazModel      ...
, PRIMARY KEY (sifModel));
```

```
CREATE TABLE operacija (
  sifOper         ...
, nazOper         ...
, PRIMARY KEY (sifOper));
```

Primjeri preslikavanja u relacijski model



STUDENT
mbrSt
prezSt
imeSt

PREDMET
sifPred
nazPred

NASTAVNIK
sifNast
prezNast
imeNast

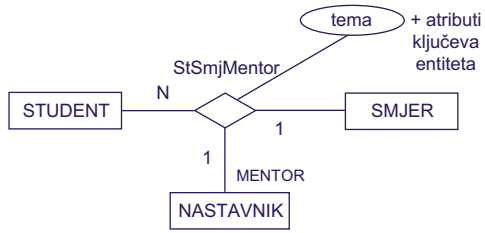
```
CREATE TABLE polozio (
  mbrSt         ...
, sifPred       ...
, sifNast       ... NOT NULL
, ocjena        ...
, PRIMARY KEY (mbrSt, sifPred)
, FOREIGN KEY (mbrSt)
    REFERENCES student (mbrSt)
, FOREIGN KEY (sifPred)
    REFERENCES predmet (sifPred)
, FOREIGN KEY (sifNast)
    REFERENCES nastavnik (sifNast));
```

```
CREATE TABLE student (
  mbrSt         ...
, prezSt        ...
, imeSt
, PRIMARY KEY (mbrSt));
```

```
CREATE TABLE predmet (
  sifPred       ...
, nazPred       ...
, PRIMARY KEY (sifPred));
```

```
CREATE TABLE nastavnik (
  sifNast       ...
, prezNast      ...
, imeNast
, PRIMARY KEY (sifNast));
```


Primjeri preslikavanja u relacijski model



STUDENT
mbrSt
prezSt
imeSt

SMJER
sifSmjer
nazSmjer

NASTAVNIK
sifNast
prezNast
imeNast

```
CREATE TABLE stSmjMentor (
  mbrSt         ...
, sifSmjer      ...
, sifNast       ... NOT NULL
, tema          ...
, PRIMARY KEY (mbrSt, sifSmjer)
, UNIQUE (mbrSt, sifNast)
, FOREIGN KEY (mbrSt)
    REFERENCES student(mbrSt)
, FOREIGN KEY (sifSmjer)
    REFERENCES smjer(sifSmjer)
, FOREIGN KEY (sifNast)
    REFERENCES nastavnik(sifNast));
```

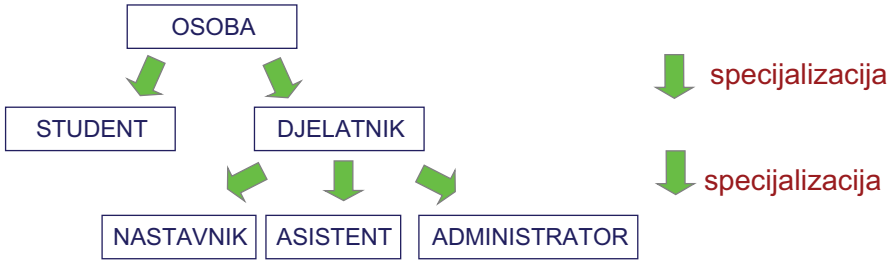
```
CREATE TABLE student (
  mbrSt         ...
, prezSt        ...
, imeSt
, PRIMARY KEY (mbrSt));
```

```
CREATE TABLE smjer (
  sifSmjer      ...
, nazSmjer      ...
, PRIMARY KEY (sifSmjer));
```

```
CREATE TABLE nastavnik (
  sifNast       ...
, prezNast      ...
, imeNast
, PRIMARY KEY (sifNast));
```


Specijalizacija

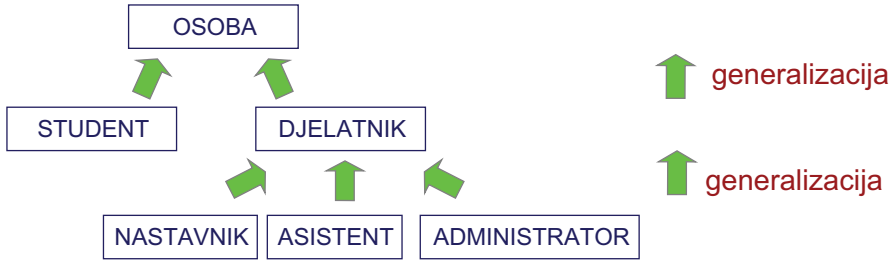
Entiteti jednog skupa entiteta mogu se temeljem njihovih karakterističnih svojstava klasificirati u zasebne skupove entiteta, postupkom koji se naziva specijalizacija



Skupovi entiteta dobiveni postupkom specijalizacije nazivaju se podklase (*subclasses*) ili specijalizacije

Generalizacija

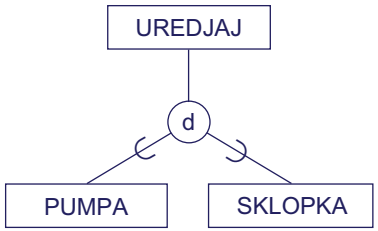
Entiteti iz nekoliko skupova entiteta sa sličnim svojstvima mogu se grupirati u zajednički skup entiteta, postupkom koji se naziva generalizacija



Skupovi entiteta dobiveni postupkom generalizacije nazivaju se nadklase (*superclasses*) ili generalizacije

Postupak generalizacije je inverzan postupku specijalizacije

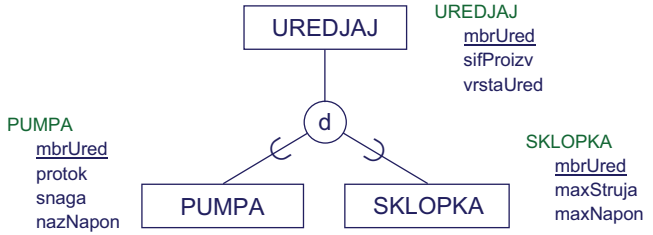
Generalizacija i specijalizacija - ER dijagram



- u kružnicu se upisuje slovo **d** (*disjoint*) ukoliko se radi o ekskluzivnoj generalizaciji/specijalizaciji
  - uređaj može biti **ili** pumpa **ili** sklopka (ekskluzivni ili)
- pomoću vrijednosti atributa *vrstaUred* (npr. 'p' ili 's') može se odrediti podklasa (specijalizacija) kojoj entitet pripada

Preslikavanje u relacijski model

- specijalizacije nemaju vlastite ključeve



UREDJAJ = mbrUred, sifProizv, vrstaUred
PUMPA = mbrUred, protok, snaga, nazNapon
SKLOPKA = mbrUred, maxStruja, maxNapon

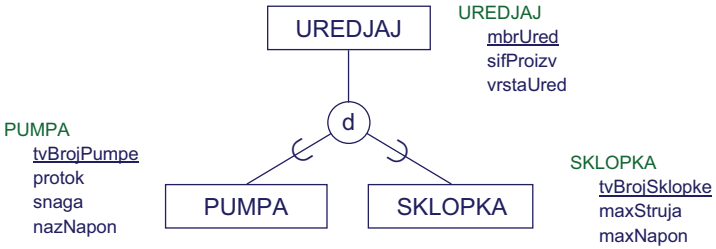
Za vježbu: napisati SQL naredbe za kreiranje relacija. Voditi računa o primarnim i stranim ključevima.

ILI

UREDJAJ = mbrUred, sifProizv, vrstaUred, protok, snaga, nazNapon, maxStruja, maxNapon

Preslikavanje u relacijski model

- specijalizacije imaju vlastite ključeve



UREDJAJ = mbrUred, sifProizv, vrstaUred
PUMPA = tvBrojPumpe, protok, snaga, nazNapon, mbrUred
SKLOPKA = tvBrojSklopke, maxStruja, maxNapon, mbrUred

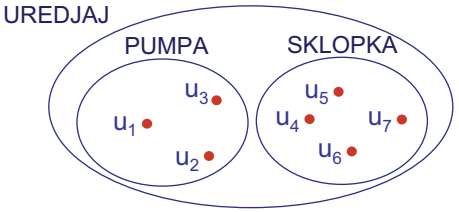
Alternativni ključevi

Za vježbu: napisati SQL naredbe za kreiranje relacija. Voditi računa o primarnim, alternativnim i stranim ključevima.

Neekskluzivna generalizacija/specijalizacija

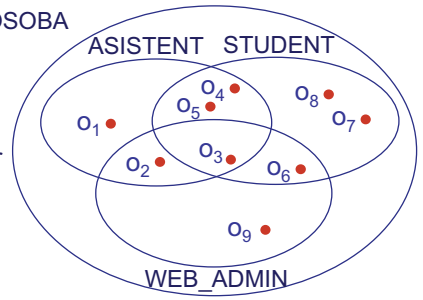
- ekskluzivna generalizacija/specijalizacija

Uređaj može biti ili pumpa ili sklopka

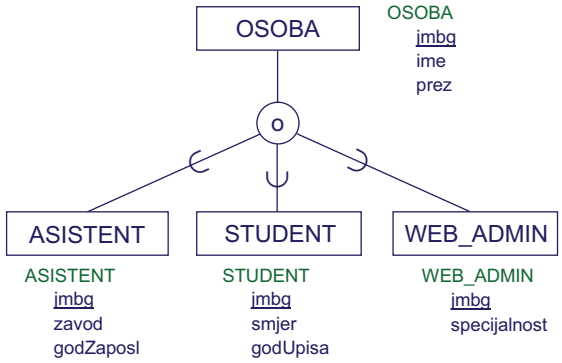


- neekskluzivna generalizacija/specijalizacija

Osoba može biti asistent i/ili student na poslijediplomskom studiju i/ili administrator web stranica fakulteta

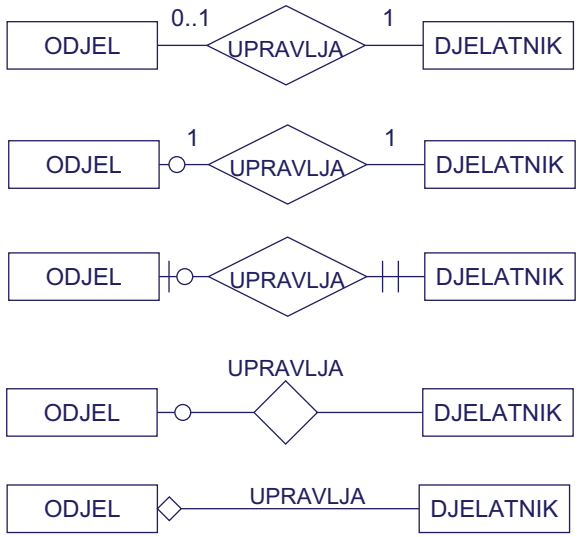


Neekskluzivna generalizacija/specijalizacija

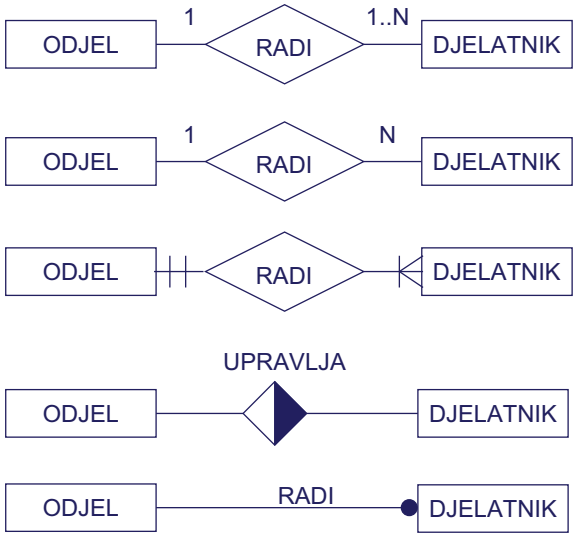


- ukoliko se radi o neekskluzivnoj generalizaciji/specijalizaciji u kružnicu se upisuje slovo **o** (*overlapping*)
  - osoba može biti asistent **i/ili** student **i/ili** administrator web stranica
- preslikavanje u relacijski model: jednako kao kod ekskluzivne generalizacije/specijalizacije (također su moguće specijalizacije s vlastitim i bez vlastitih ključeva)

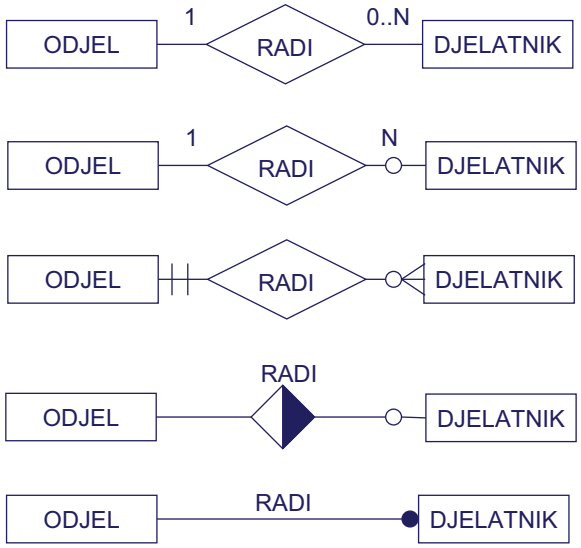
ER model - različita notacija



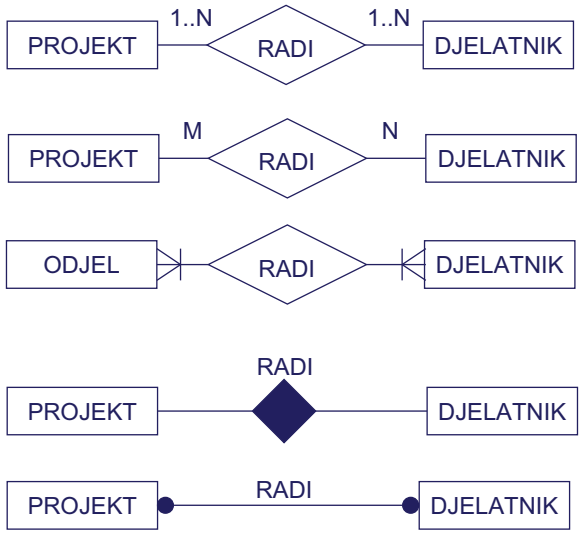
ER model - različita notacija



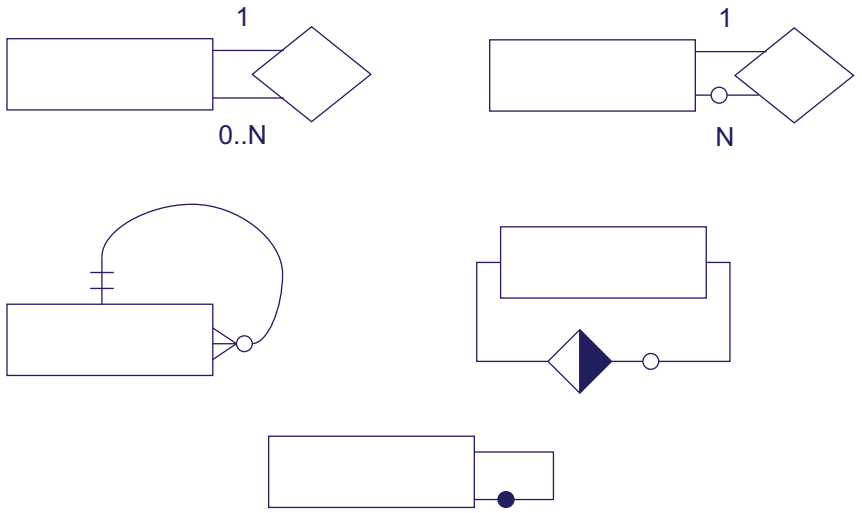
ER model - različita notacija



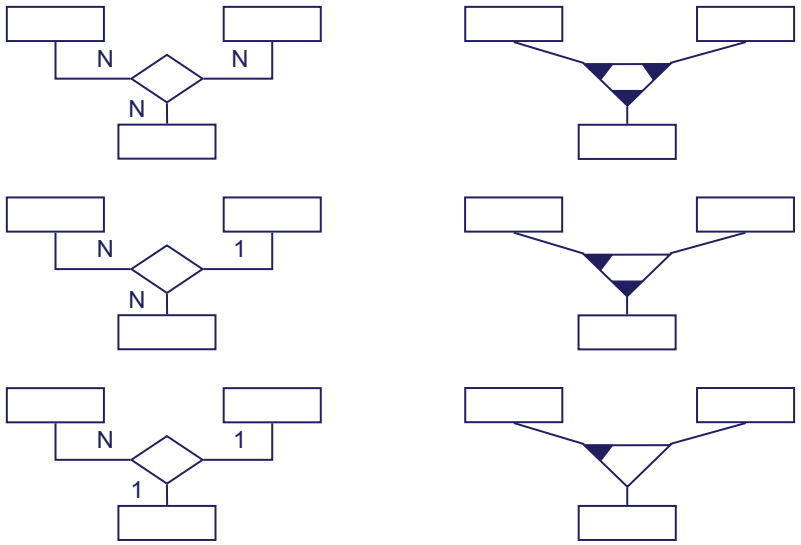
ER model - različita notacija



ER model - različita notacija



ER model - različita notacija

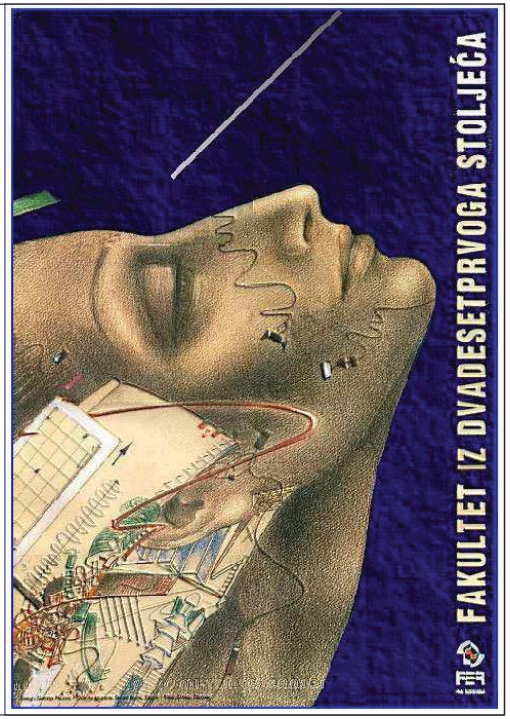


Baze podataka

Predavanja
lipanj 2009.

14. ER model baze podataka

(3. dio - primjeri)



1. Model baze podataka za razredbeni ispit

Potrebno je evidentirati podatke o kandidatima: JMBG, prezime, ime, završenu srednju školu, mjesto rođenja i mjesto stanovanja. Pretpostavlja se da je kandidat završio samo jednu srednju školu. Za svaku srednju školu treba evidentirati šifru koja ju jedinstveno identificira, naziv, adresu i mjesto u kojem se škola nalazi. Za mjesto treba evidentirati poštanski broj, naziv mjesta i županiju u kojoj se mjesto nalazi. Županija ima svoju šifru i naziv.

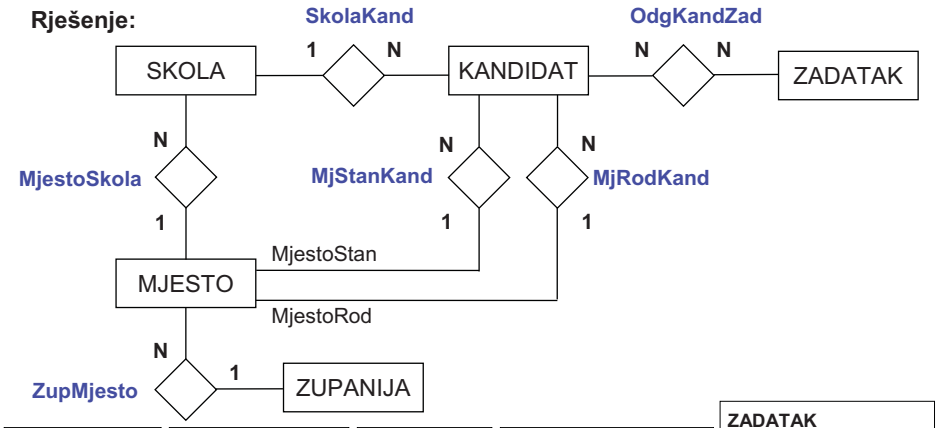
Treba evidentirati podatke o zadacima na testu: redni broj zadatka, tekst zadatka, oznaku točnog odgovora (može biti A, B, C, D ili E).

Za svakog kandidata evidentirati odgovore koje je dao na zadatke (mogući odgovori kandidata su A, B, C, D, E ili ništa).

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Opisati relacijski model u obliku SQL naredbi za kreiranje relacija s opisanim integritetskim ograničenjima. Odabrati prikladne tipove podataka.

Rješenje:



KANDIDAT
jmbg
imeKand
prezKand
K = { jmbg }

SKOLA
sifSkola
nazSkola
K = { sifSkola }

MJESTO
pbr
nazMjesto
K = { pbr }

ZUPANIJA
sifZupanija
nazZupanija
K = { sifZupanija }

ZADATAK
rbrZadatak
tekstZadatak
oznTocnogOdg
K = { rbrZadatak }

SkolaKand
jmbg
sifSkola
K = { jmbg }

MjestoSkola
sifSkola
pbr
adresa
K = { sifSkola }

ZupMjesto
pbr
sifZupanija
K = { pbr }

MjStanKand
jmbg
pbr
K = { jmbg }

MjRodKand
jmbg
pbr
K = { jmbg }

OdgKandZad
jmbg
rbrZadatak
odgKand
K = { jmbg, rbrZadatak }

KANDIDAT
jmbg
imeKand
prezKand
K = { jmbg }

Ako bi neki entitet imao više mogućih ključeva, shema entiteta bi se mogla opisati npr. ovako:

KANDIDAT
jmbg
sifKand
imeKand
prezKand
PK = K_1 = { jmbg }
K_2 = { sifKand }

Relacijski model u obliku SQL naredbi za kreiranje relacija:

```
CREATE TABLE zupanija (
   sifZupanija    SMALLINT
 , nazZupanija    CHAR(40)
 , PRIMARY KEY (sifZupanija));

CREATE TABLE mjesto (
   pbr          INTEGER
 , nazMjesto    CHAR(20)
 , sifZupanija SMALLINT  NOT NULL
 , PRIMARY KEY (pbr)
 , FOREIGN KEY (sifZupanija) REFERENCES zupanija(sifZupanija));

CREATE TABLE skola (
   sifSkola  INTEGER
 , nazSkola  CHAR(40)
 , pbr       INTEGER   NOT NULL
 , adresa    CHAR(40)
 , PRIMARY KEY (sifSkola)
 , FOREIGN KEY (pbr) REFERENCES mjesto(pbr));

CREATE TABLE zadatak (
   rbrZadatak    INTEGER
 , tekstZadatak  CHAR(512)
 , oznTocnogOdg  CHAR(1)
 , PRIMARY KEY (rbrZadatak));
```

Relacijski model u obliku SQL naredbi za kreiranje relacija (nastavak):

```
CREATE TABLE kandidat (
   jmbg       CHAR(13)
 , imeKand    CHAR(20)
 , prezKand   CHAR(20)
 , pbrRod     INTEGER  NOT NULL
 , pbrStan    INTEGER  NOT NULL
 , sifSkola   INTEGER  NOT NULL
 , PRIMARY KEY (jmbg)
 , FOREIGN KEY (pbrRod) REFERENCES mjesto (pbr)
 , FOREIGN KEY (pbrStan) REFERENCES mjesto (pbr)
 , FOREIGN KEY (sifSkola) REFERENCES skola (sifSkola));


CREATE TABLE odgKandZad (
   jmbg        CHAR(13)
 , rbrZadatak INTEGER
 , odgKand     CHAR(1)
 , PRIMARY KEY (jmbg, rbrZadatak)
 , FOREIGN KEY (jmbg) REFERENCES kandidat (jmbg)
 , FOREIGN KEY (rbrZadatak) REFERENCES zadatak (rbrZadatak));
```


2. Model baze podataka za videoteku

Za film se evidentira šifra (identificira film), naslov filma, te osobe i njihove funkcije u filmu.

Funkcije koje osoba može imati u filmu predstavljene su kraticom i nazivom (npr. GL, glumac; RED, redatelj; SC, scenarist, itd.). Za svaku se osobu evidentira šifra osobe (identificira osobu), prezime i ime.

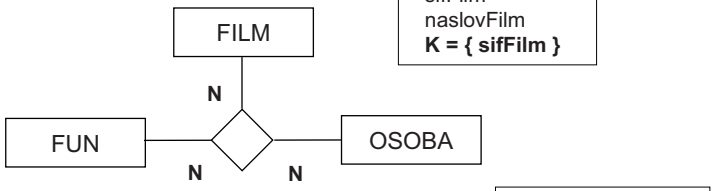
Treba uočiti da ista osoba može u istom filmu imati različite funkcije, npr:

relacija VIDEOTEKA

sif Film	naslovFilm	ozn Fun	nazFun	sif Osoba	ime	prezime
1	Nepomirljivi	RED	redatelj	10	Clint	Eastwood
1	Nepomirljivi	GL	glumac	10	Clint	Eastwood
1	Nepomirljivi	GL	glumac	20	Morgan	Freeman
2	Mostovi okruga Madison	RED	redatelj	10	Clint	Eastwood
2	Mostovi okruga Madison	GL	glumac	40	Meryl	Streep
2	Mostovi okruga Madison	GL	glumac	10	Clint	Eastwood
3	Prljavi Harry	RED	redatelj	30	Don	Siegel
3	Prljavi Harry	GL	glumac	10	Clint	Eastwood

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Rješenje:



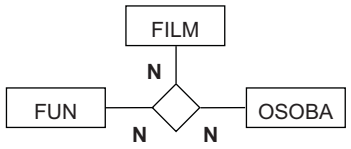
FILM
sifFilm
naslovFilm
K = { sifFilm }

OSOBA
sifOsoba
imeOsoba
prezOsoba
K = { sifOsoba }

FUN
oznFun
nazFun
K = { oznFun }

FilmFunOsoba
sifFilm
sifFun
sifOsoba
K = { sifFilm, sifFun, sifOsoba }

Diskusija:



FilmFunOsoba

Relacije koje nastaju transformacijom ER modela na slici:

ključevi relacija su podcrtani

FILM

sifFilm	***naslovFilm***
1	Nepomirljivi
2	Mostovi okruga M.
3	Prljavi Harry

OSOBA

sifOsoba	***ime, prezime***
10	C. Eastwood
20	M. Freeman
30	D. Siegel
40	M. Streep

FUN

oznFun	***nazFun***
RED	redatelj
GL	glumac

FilmFunOsoba

sifFilm	***oznFun***	***sifOsoba***
1	RED	10
1	GL	10
1	GL	20
2	RED	10
2	GL	40
2	GL	10
3	RED	30
3	GL	10

SQL upit kojim se dohvaćaju osobe i njihove funkcije u filmu "Prljavi Harry"

```
SELECT osoba.*, fun.*
   FROM osoba, film, fun, filmFunOsoba
   WHERE osoba.sifOsoba = filmFunOsoba.sifOsoba
     AND film.sifFilm = filmFunOsoba.sifFilm
     AND fun.oznFun = filmFunOsoba.oznFun
     AND film.naslovFilm = 'Prljavi Harry'
```

sifOsoba	ime	prezime	oznFun	nazFun
10	Clint	Eastwood	GL	glumac
30	Don	Siegel	RED	redatelj

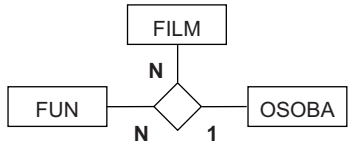
relacija VIDEOTEKA

sif Film	naslovFilm	ozn Fun	nazFun	sif Osoba	ime	prezime
1	Nepomirljivi	RED	redatelj	10	Clint	Eastwood
1	Nepomirljivi	GL	glumac	10	Clint	Eastwood
1	Nepomirljivi	GL	glumac	20	Morgan	Freeman
2	Mostovi okruga Madison	RED	redatelj	10	Clint	Eastwood
2	Mostovi okruga Madison	GL	glumac	40	Meryl	Streep
2	Mostovi okruga Madison	GL	glumac	10	Clint	Eastwood
3	Prljavi Harry	RED	redatelj	30	Don	Siegel
3	Prljavi Harry	GL	glumac	10	Clint	Eastwood

(Teorey): U vezi koja povezuje entitete E_1, .., E_k, .., E_m, spojnost =1 entiteta E_k znači da (...) odnosno, vrijedi funkcijska zavisnost

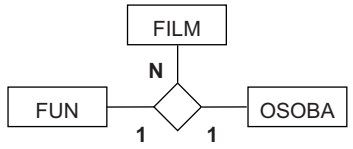
$$\bigcup_{j=1}^{m} K_j \setminus K_k \rightarrow K_k$$ gdje su skupovi K_j , (j = 1, .., m), ključevi entiteta E_1, .., E_m

ZAŠTO OVI MODELI NISU ISPRAVNI?



sifFilm,oznFun → sifOsoba

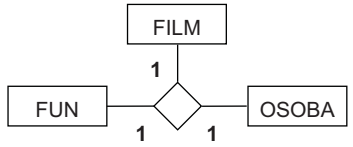
u filmu može glumiti samo jedan glumac, film može režirati samo jedan redatelj, scenarij za film može pisati samo jedan scenarist, ...



sifFilm, oznFun → sifOsoba

sifFilm, sifOsoba → oznFun

dodatno: u jednom filmu osoba može imati samo jednu funkciju



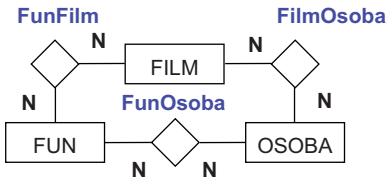
sifFilm, oznFun → sifOsoba

sifFilm, sifOsoba → oznFun

sifOsoba, oznFun → sifFilm

dodatno: ...

ZAŠTO OVAJ MODEL NIJE ISPRAVAN?



Relacije koje nastaju transformacijom ER modela na slici:

FILM

sifFilm	***naslovFilm***
1	Nepomirljivi
2	Mostovi okruga M.
3	Prljavi Harry

OSOBA

sifOsoba	***ime, prezime***
10	C. Eastwood
20	M. Freeman
30	D. Siegel
40	M. Strep

FUN

oznFun	***nazFun***
RED	redatelj
GL	glumac

ključevi relacija su podcrtani

FunFilm

oznFun	***sifFilm***
RED	1
RED	2
RED	3
GL	1
GL	2
GL	3

FilmOsoba

sifFilm	***sifOsoba***
1	10
1	20
2	10
2	40
3	30
3	10

FunOsoba

oznFun	***sifOsoba***
GL	10
GL	20
GL	40
RED	10
RED	30

SQL upit kojim se dohvaćaju osobe i njihove funkcije u filmu "Prljavi Harry"

```
SELECT osoba.*, fun.*
   FROM osoba, film, fun, filmOsoba, funOsoba, funFilm
   WHERE osoba.sifOsoba = filmOsoba.sifOsoba
     AND filmOsoba.sifFilm = film.sifFilm
     AND film.sifFilm = funFilm.sifFilm
     AND funFilm.oznFun = fun.oznFun
     AND fun.oznFun = funOsoba.oznFun
     AND funOsoba.sifOsoba = osoba.sifOsoba
     AND film.naslovFilm = 'Prljavi Harry'
```

sifOsoba	ime	prezime	oznFun	nazFun
10	Clint	Eastwood	GL	glumac
10	Clint	Eastwood	RED	redatelj
30	Don	Siegel	RED	redatelj

Clint Eastwood nije redatelj filma "Prljavi Harry"!
→ "gubitak informacije"!!!

Dekompozicija relacije VIDEOTEKA **nije** obavljena bez gubitka informacije. Uvjet za dekompoziciju bez gubitka informacija opisan je u predavanjima.

3. Model baze podataka za poduzeće za održavanje plinskih instalacija

Uređaji koje poduzeće evidentira su brojila, ventili i reduktori. Za svaki pojedini uređaj treba evidentirati vrstu uređaja ('B', 'V' ili 'R'), proizvođača uređaja, tvornički broj i godinu proizvodnje uređaja. Za proizvođače uređaja evidentiraju se njihove šifre i nazivi. Ne postoje dva uređaja istog proizvođača koji imaju jednake tvorničke brojeve.

Dodatno, ovisno o vrsti uređaja, treba evidentirati njima svojstvene, posebne ili specijalističke podatke.

za brojila:	za ventile:	za reduktore:
razred točnosti	promjer navoja	ulazni tlak plina
max. protok plina	način zatvaranja	izlazni tlak plina
		promjer navoja

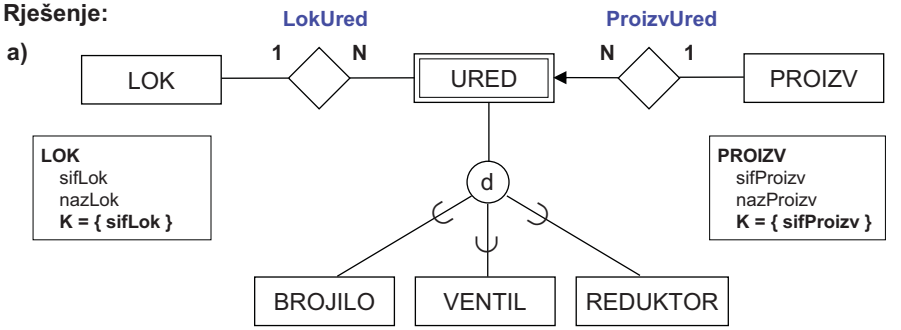
Potrebno je evidentirati popis lokacija (šifra i naziv) na kojima uređaji mogu biti instalirani. Evidentirati trenutnu lokaciju na kojoj je uređaj instaliran.

a) Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF. Opisati relacijski model u obliku SQL naredbi za kreiranje relacija s ugrađenim pravilima integriteta.

b) Što treba promijeniti u ER modelu iz a) kako bi se omogućilo evidentiranje povijesti premještanja uređaja. Kakve su posljedice na relacijski model?

Rješenje:

a)



LOK
sifLok
nazLok
K = { sifLok }

PROIZV
sifProizv
nazProizv
K = { sifProizv }

URED
tvBroj
sifProizv
godProizv
vrUred
K = { tvBroj, sifProizv }

ProizvUred
tvBroj
sifProizv
K = { tvBroj, sifProizv }

LokUred
tvBroj
sifProizv
sifLok
K = { tvBroj, sifProizv }

BROJILO
tvBroj
sifProizv
razrTocn
maxProtok
K = { tvBroj, sifProizv }

REDUKTOR
tvBroj
sifProizv
ulTlak
izlTlak
promjNavoja
K = { tvBroj, sifProizv }

VENTIL
tvBroj
sifProizv
promjNavoja
nacZatvar
K = { tvBroj, sifProizv }

Rješenje: Relacijski model

```
CREATE TABLE proizv (
   sifProizv   INTEGER
 , nazProizv   CHAR(20)
 , PRIMARY KEY (sifProizv));
```

```
CREATE TABLE lok (
   sifLok    INTEGER
 , nazLok    CHAR(40)
 , PRIMARY KEY (sifLok));
```

```
CREATE TABLE ured (
   tvBroj    CHAR(20)
 , sifProizv INTEGER
 , godProizv SMALLINT
 , vrUred    CHAR(1)
 , sifLok    INTEGER   NOT NULL
 , PRIMARY KEY (tvBroj, sifProizv)
 , FOREIGN KEY (sifProizv) REFERENCES proizv (sifProizv)
 , FOREIGN KEY (sifLok) REFERENCES lok (sifLok));
```

```
CREATE TABLE brojilo (
   tvBroj    CHAR(20)
 , sifProizv INTEGER
 , razrTocn  DECIMAL(3,1)
 , maxProtok DECIMAL(5,4)
 , PRIMARY KEY (tvBroj, sifProizv)
 , FOREIGN KEY (tvBroj, sifProizv)
     REFERENCES ured (tvBroj, sifProizv));
```

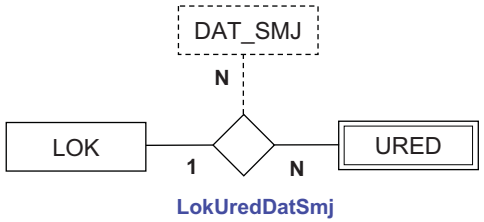

Rješenje: Relacijski model (nastavak)

```
CREATE TABLE ventil (
   tvBroj      CHAR(20)
 , sifProizv   INTEGER
 , promjNavoja DECIMAL(3,1)
 , nacZatvar   DECIMAL(5,4)
 , PRIMARY KEY (tvBroj, sifProizv)
 , FOREIGN KEY (tvBroj, sifProizv)
     REFERENCES ured (tvBroj, sifProizv));
```

```
CREATE TABLE reduktor (
   tvBroj      CHAR(20)
 , sifProizv   INTEGER
 , ulTlak      DECIMAL(6,2)
 , izlTlak     DECIMAL(6,2)
 , promjNavoja DECIMAL(3,1)
 , PRIMARY KEY (tvBroj, sifProizv)
 , FOREIGN KEY (tvBroj, sifProizv)
     REFERENCES ured (tvBroj, sifProizv));
```

Rješenje:

b) Pretpostavi li se da jedan uređaj ne može biti premješten više nego jedan puta na dan, segment ER modela će se promijeniti na sljedeći način:



U odnosu na rješenje pod a), u novom relacijskom modelu potrebno je izbaciti atribut sifLok iz relacije ured, te dodati novu relaciju lokUredDatSmj

```
CREATE TABLE lokUredDatSmj (
   tvBroj      CHAR(20)
 , sifProizv   INTEGER
 , datSmjestaj DATE
 , sifLok      INTEGER NOT NULL
 , PRIMARY KEY (tvBroj, sifProizv, datSmjestaj)
 , FOREIGN KEY (tvBroj, sifProizv)
     REFERENCES ured (tvBroj, sifProizv)
 , FOREIGN KEY (sifLok)
     REFERENCES lok(sifLok));
```


4. Model baze podataka automehaničarske radionice

Evidentirati podatke o automobilima. Automobil je identificiran tvorničkim brojem (ne postoje dva automobila s istim tvorničkim brojem). Za automobil treba evidentirati godinu proizvodnje i model automobila. Modeli automobila identificirani su proizvođačem i nazivom modela (međusobno različiti proizvođači mogu svoje modele nazivati istim imenom - npr. Renault može imati svoj model naziva Europa, a Opel može imati sasvim drugi model koji se također naziva Europa). Za model automobila evidentira se godina u kojoj je model prvi puta proizveden. Proizvođač ima naziv, a identificiran je svojom šifrom.

U radionici je napravljen popis vrsta poslova koji se mogu obavljati na automobilima. Vrste poslova su šifrirane, a osim šifre i opisa vrste posla (npr. "Izmjena ulja", "Podešavanje ventila", itd.), za svaku vrstu posla se evidentira normativom zadano trajanje izraženo u minutama (koliko bi vremena mehaničar trebao utrošiti obavljajući posao te vrste). Vrste poslova su nezavisne od modela automobila - npr. "Izmjena ulja" je uvijek jednak posao neovisno od modela automobila na kojem se obavlja.

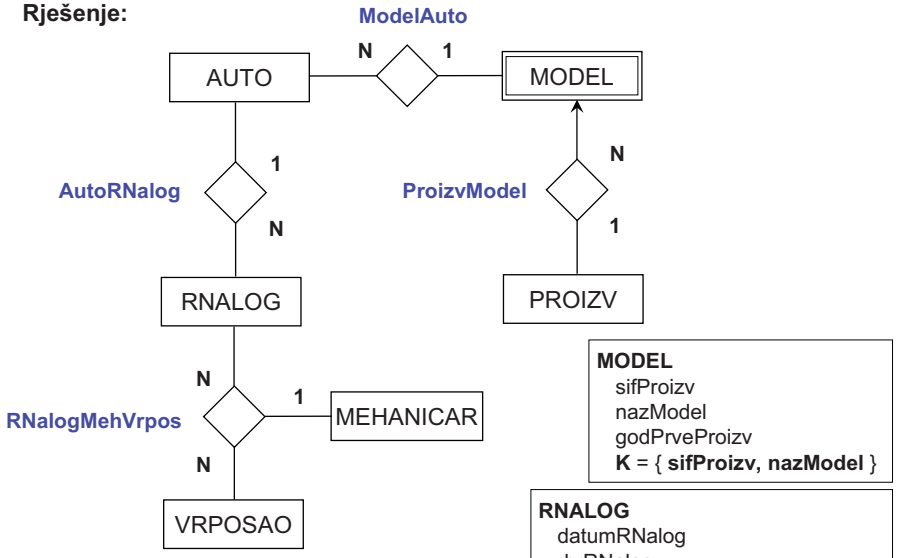
Za mehaničare zaposlene u radionici evidentira se jmbg, prezime i ime.

Za svaki dolazak automobila u radionicu otvara se jedan Radni nalog na kojem se evidentira automobil i datum dolaska automobila u radionicu. Isti automobil može biti primljen u radionicu više puta (čak i istog dana), ali se svaki put otvara novi Radni nalog. Radni nalog nema šifru. Radni nalog pri otvaranju dobiva svoj redni broj, pri čemu svakog dana redni brojevi naloga započinju ponovo s brojem jedan. Za isti datum ne postoje dva Radna naloga s istim brojem.

Uz Radni nalog se evidentira koji mehaničari će obaviti koje vrste poslova na automobilu. Poslove koji su zadani na Radnom nalogu može obaviti jedan ili nekoliko mehaničara, ali jedan zadani posao će jedan mehaničar obaviti sam od početka do kraja. Mehaničari na raznim Radnim nalozima mogu obavljati različite vrste poslova. Mehaničar odmah po obavljenom poslu na nekom automobilu evidentira koliko je vremena u minutama zaista utrošio na obavljanje tog posla (to se vrijeme može razlikovati od normativom zadanog vremena).

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Rješenje:



Za vježbu opisati atribute i ključeve preostalih entiteta i veza.

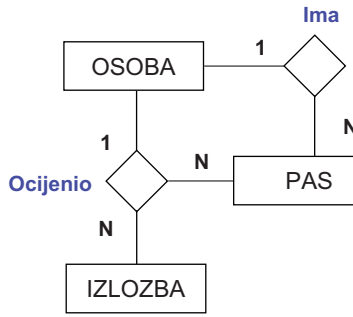
5. Model baze podataka za izložbe pasa

Za svaku se osobu evidentira jmbg, prezime i ime. Za psa se evidentira broj markice koja identificira psa, ime psa, datum okota i osoba koja je vlasnik tog psa. Pretpostavlja se da jedna osoba može imati više pasa, a pas pripada samo jednoj osobi.

Neki vlasnici vode svoje pse na izložbe pasa. Za izložbu se evidentira šifra izložbe koja ju jedinstveno identificira i datum izložbe. Za jednog psa na jednoj izložbi treba evidentirati samo jednu ocjenu i osobu koja ga je ocjenjivala. Ista osoba na jednoj izložbi može ocijeniti više pasa. Ista osoba može ocjenjivati istog psa na više različitih izložbi. Za osobe koje ocjenjuju pse također se evidentiraju jmbg, prezime i ime. Te osobe mogu istovremeno biti i vlasnici pasa.

Nacrtati ER model i opisati entitete i veze. Sve sheme moraju zadovoljavati 3NF.

Rješenje:



OSOBA
jmbg
imeOso
prezOso
K = { **jmbg** }

PAS
brMarkice
imePas
datOkota
K = { **brMarkice** }

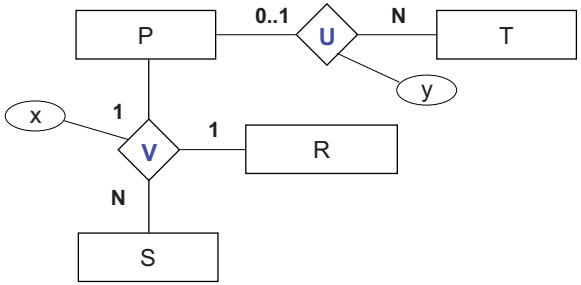
IZLOZBA
siflzlozba
datIzlozba
K = { **siflzlozba** }

Ima
brMarkice
jmbg
K = { **brMarkice** }

Ocijenio
brMarkice
siflzlozba
jmbg
ocjena
K = { **siflzlozba, brMarkice** }

6. Zadan je ER model i pripadne sheme entiteta. Na slici su prikazani samo vlastiti atributi veza.

Definirati sheme veza. Napisati SQL naredbe za kreiranje relacija relacijskog modela. Tipove podataka ne treba navoditi. Naredbe moraju sadržavati definicije integritetskih ograničenja.



P
a
b
c
d
PK = K_1 = { **a, b** }
K_2 = { **c** }

R
e
f
PK = { **e** }

S
g
h
PK = { **g** }

T
i
j
k
PK = { **i, j** }

Rješenje: **Sheme veza**

V
a
b
e
g
x
PK = K_1 = { **a, b, g** }
K_2 = { **e, g** }

U
i
j
a
b
y
PK = { **i, j** }

Relacijski model

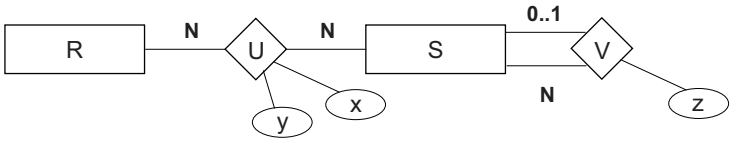
```
CREATE TABLE p (
   a   ...
 , b   ...
 , c   ...
 , d   ...
 , PRIMARY KEY (a, b)
 , UNIQUE (c));

CREATE TABLE r (
   e   ...
 , f   ...
 , PRIMARY KEY (e));

CREATE TABLE s (
   g   ...
 , h   ...
 , PRIMARY KEY (g));
```

```
CREATE TABLE t (
   i   ...
 , j   ...
 , k   ...
 , y   ...
 , a   ...
 , b   ...
 , PRIMARY KEY (i, j)
 , FOREIGN KEY (a, b)
     REFERENCES p (a, b));
```

```
CREATE TABLE v (
   a   ...
 , b   ...
 , e   ...
 , g   ...
 , x   ...
 , PRIMARY KEY (a, b, g)
 , UNIQUE (e, g)
 , FOREIGN KEY (a, b)
     REFERENCES p (a, b)
 , FOREIGN KEY (e)
     REFERENCES r (e)
 , FOREIGN KEY (g)
     REFERENCES s (g));
```

7. Zadan je ER model i pripadne sheme entiteta. Na slici su prikazani samo vlastiti atributi veza.

Definirati sheme veza. Napisati SQL naredbe za kreiranje relacija relacijskog modela. Tipove podataka ne treba navoditi. Naredbe moraju sadržavati definicije integritetskih ograničenja.



R
a
b
c
PK = { **a, b** }

S
d
e
PK = { **d** }

Rješenje:

Sheme veza

U
a
b
d
x
y
PK = { **a, b, d** }

V
d
~~d~~ d1 ← Preimenovati jedan od atributa
z
PK = { **d** }

Relacijski model

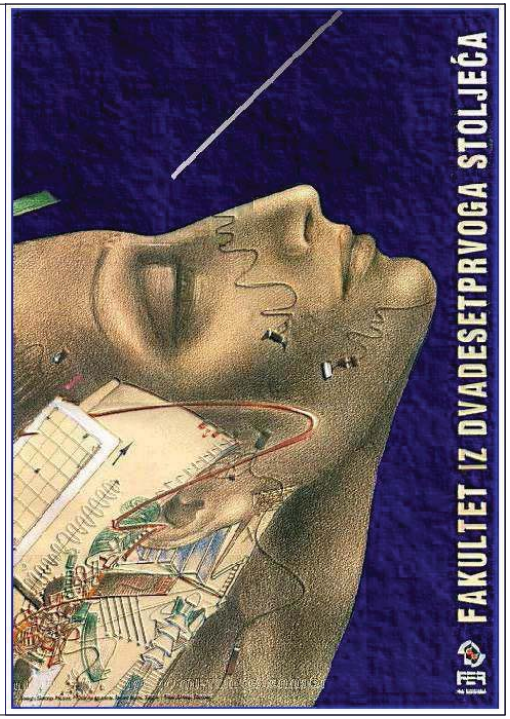
```
CREATE TABLE r (
   a   ...
 , b   ...
 , c   ...
 , PRIMARY KEY (a, b));

CREATE TABLE u (
   a   ...
 , b   ...
 , d   ...
 , x   ...
 , y   ...
 , PRIMARY KEY (a, b, d)
 , FOREIGN KEY (a, b) REFERENCES r (a, b)
 , FOREIGN KEY (d) REFERENCES s (d));
```

```
CREATE TABLE s (
   d   ...
 , d1  ...
 , e   ...
 , z   ...
 , PRIMARY KEY (d)
 , FOREIGN KEY (d1) REFERENCES s (d));
```


Baze podataka

Predavanja
lipanj 2009.

15. Sigurnost baze podataka



Integritet i sigurnost baze podataka

- Pojmovi integritet i sigurnost baze podataka se često spominju zajedno, međutim radi se o dva različita aspekta zaštite podataka
  - Integritet baze podataka (*database integrity*) - operacije nad podacima koje korisnici obavljaju **su ispravne** (tj. uvijek rezultiraju konzistentnim stanjem baze podataka)
    - "podaci se štite od ovlaštenih korisnika"
  - Sigurnost baze podataka (*database security*) - korisnici koji obavljaju operacije nad podacima **su ovlašteni** za obavljanje tih operacija
    - "podaci se štite od neovlaštenih korisnika"

Među ovim pojmovima postoje i sličnosti. U oba slučaja:

- moraju biti definirana **pravila** koja korisnici ne smiju narušiti
- pravila se pohranjuju u rječnik podataka
- SUBP nadgleda rad korisnika - osigurava poštivanje pravila

Oblici narušavanja sigurnosti i moguće posljedice

- Oblici narušavanja sigurnosti baze podataka su:
  - neovlašteno čitanje podataka
  - neovlaštena izmjena podataka
  - neovlašteno uništavanje podataka
- Moguće posljedice su:
  - krađa ili prijevara
  - gubitak tajnosti
    - odnosi se na podatke kritične za funkcioniranje organizacije
    - npr. krađa recepture - rezultira gubitkom konkurentnosti na tržištu
  - gubitak privatnosti
    - odnosi se na osobne podatke
    - npr. krađa podataka o zdravstvenom stanju osobe - rezultira sudskim procesom protiv vlasnika baze podataka
  - gubitak raspoloživosti
    - npr. uništenjem dijela podataka

Protumjere

- sigurnost baze podataka se osigurava zaštitom na nekoliko razina
  - **zaštita na razini SUBP**
    - spriječiti pristup bazama podataka ili onim dijelovima baza podataka za koje korisnici nisu ovlašteni
  - **zaštita na razini operacijskog sustava**
    - spriječiti pristup radnoj memoriji računala ili datotekama u kojima SUBP pohranjuje podatke
  - **zaštita na razini računalne mreže**
    - spriječiti presretanje poruka (*sniffing*) na internetu i intranetu
  - **fizička zaštita**
    - fizički zaštititi lokaciju računalnog sustava
  - **zaštita na razini korisnika**
    - spriječiti da ovlašteni korisnici nepažnjom ili namjerno (npr. u zamjenu za mito ili druge usluge) omoguće pristup podacima neovlaštenim osobama

Aspekti zaštite podataka

- **zakonski, socijalni i etički aspekt**
  - ima li vlasnik baze podataka zakonsko pravo na prikupljanje i korištenje podataka
  - npr. smije li zdravstvena ustanova koja, u skladu sa zakonom prikuplja podatke o pacijentima, te iste podatke koristiti pri donošenju odluke hoće li svog bivšeg pacijenta zaposliti
- **strategijski aspekt**
  - tko definira pravila pristupa - tko određuje kakve ovlasti ima pojedini korisnik baze podataka, ...
- **operativni aspekt**
  - kako osigurati poštivanje pravila - kojim mehanizmima se osigurava poštivanje definiranih pravila, na koji način su lozinke zaštićene, koliko često se mijenjaju, ...

Ustav RH - Članak 37.

Svakom se jamči sigurnost i tajnost osobnih podataka. Bez privole ispitanika, osobni se podaci mogu prikupljati, obrađivati i koristiti samo uz uvjete određene zakonom.

Zakonom se uređuje zaštita podataka te nadzor nad djelovanjem informatičkih sustava u Republici.

Zabranjena je uporaba osobnih podataka suprotna utvrđenoj svrsi njihovoga prikupljanja.

- Zakon o zaštiti osobnih podataka

Korisnici SUBP i ovjera autentičnosti

- administrator sustava (operacijskog sustava ili SUBP) omogućuje korisniku pristup sustavu (operacijskom sustavu ili SUBP) definiranjem jedinstvenog identifikatora korisnika (*user name*, *user ID*, *login ID*) i pripadne lozinke (*password*) koja je poznata samo dotičnom korisniku i sustavu
- korisnik koji pristupa sustavu (operacijskom sustavu ili SUBP) poznavanjem lozinke ovjerava svoju autentičnost (*authentication*)
- za ovjeru autentičnosti korisnika SUBP može koristiti
  - mehanizme operacijskog sustava
    ili
  - vlastite mehanizme

Autorizacija i modeli kontrole pristupa

- Autorizacija je postupak kojim se određenom korisniku dodjeljuje dozvola za obavljanje određenih vrsta operacija (čitanje, izmjena, brisanje, ...) nad određenim objektima baze podataka (relacija, pogled, atribut, ...)
  - podaci o dodijeljenim dozvolama pohranjuju se u rječnik podataka
- Prije obavljanja svake operacije, SUBP provjerava ima li korisnik dozvolu za obavljanje operacije nad objektom
  - kontrola pristupa (*access control*)
- Današnji SUBP podržavaju dva različita modela kontrole pristupa podacima
  - **mandatna kontrola pristupa** (MAC-*Mandatory Access Control*)
  - **diskrecijska kontrola pristupa** (DAC-*Discretionary Access Control*)

Mandatna kontrola pristupa

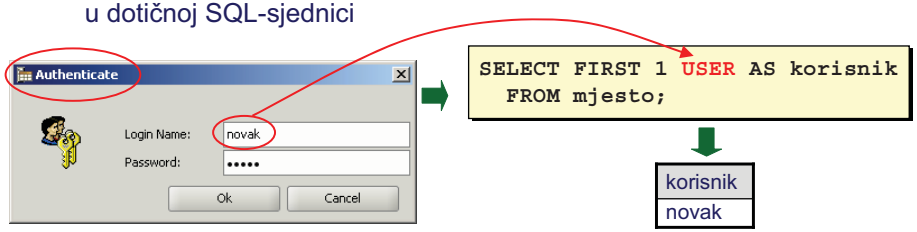
- manji broj SUBP podržava mandatnu kontrolu pristupa
  - koristi se relativno rijetko u odnosu na diskrecijsku kontrolu pristupa
- mandatna kontrola pristupa je primjenjiva u sustavima u kojima se dozvole dodjeljuju na temelju pozicije korisnika u hijerarhiji neke organizacije (vojska, državna uprava, ...)
- svaki **objekt** dobiva oznaku klasifikacijske razine (*classification level*), npr. povjerljivo, tajno, vrlo tajno, ...
- svakom **korisniku** dodjeljuje se oznaka razine ovlasti (*clearance level*)
  - korisnici mogu obavljati operacije nad onim objektima za koje imaju odgovarajuću razinu ovlasti

Diskrecijska kontrola pristupa

- većina današnjih SUBP podržava diskrecijsku kontrolu pristupa
  - diskrecijska kontrola pristupa je podržana SQL standardom
- određenom korisniku se eksplicitno dodjeljuje dozvola za obavljanje određene operacije nad određenim objektom
  - dozvole su opisane trojkama <korisnik, objekt, vrsta operacije>
    <horvat, ispit, čitanje>
    <horvat, ispit, izmjena>
    <horvat, predmet, čitanje>
    <novak, predmet, čitanje>
  - kada korisnik novak pokuša obaviti operaciju čitanja objekta (relacije) predmet, SUBP provjerava postoji li dozvola u obliku trojke <novak, predmet, čitanje>
- u preostalom dijelu predavanja razmatrat će se diskrecijska kontrola pristupa

Korisnici u SQL-u

- korisnik s određenom identifikacijskom oznakom (*userID*)
  - pri uspostavljanju SQL-sjednice korisnik se prijavljuje svojim identifikatorom korisnika, te lozinkom ovjerava svoju autentičnost
  - funkcija USER vraća vrijednost identifikatora korisnika koji se koristi u dotičnoj SQL-sjednici



- bilo koji korisnik (PUBLIC)
  - dodjelom dozvole "korisniku" PUBLIC, dozvolu za obavljanje operacije dobivaju svi sadašnji i budući korisnici

Objekti i vlasnici objekata u SQL-u

- **Objekti**
  - relacija (tablica, *table*)
  - atribut (stupac tablice, *column*)
  - virtualna relacija (pogled, *view*)
  - baza podataka
- **Vlasnik objekta (*object owner*)**
  - vlasnik objekta je korisnik koji je kreirao objekt, npr:
    - vlasnik baze podataka je korisnik koji je kreirao bazu podataka
    - vlasnik relacije je korisnik koji je kreirao relaciju
  - vlasnik objekta implicitno dobiva dozvole za obavljanje **svih** vrsta operacija nad objektom, uključujući dozvole za:
    - dodjeljivanje svih vrsta dozvola nad tim objektom drugim korisnicima
    - uništavanje objekta

Vrste dozvola u SQL-u na razini baze podataka (*dbPrivilege*)

- Različiti SUBP imaju različita rješenja za dodjeljivanje dozvola na razini baze podataka. Ovdje je prikazano rješenje koje se koristi u sustavu IBM Informix:
  - **CONNECT**
    - uspostavljanje SQL-sjednice i obavljanje operacija nad objektima za koje je korisnik dobio dozvolu od vlasnika objekta ili je njihov vlasnik, kreiranje virtualnih i privremenih relacija
  - **RESOURCE**
    - CONNECT + kreiranje **novih** relacija u bazi podataka
  - **DBA**
    - RESOURCE + neovisno o vlasništvu i dozvolama nad objektima u bazi podataka: sve vrste operacija nad svim objektima, uništavanje svih objekata (uključujući i bazu podataka)
    - korisnik koji kreira bazu podataka je vlasnik te baze podataka i implicitno dobiva DBA (*Database administrator*) dozvolu

Vrste dozvola u SQL-u na razini [virtualne] relacije (*tablePrivilege*)

- **SELECT [(*columnList*)]**
  - čitanje n-torki (ili vrijednosti navedenih atributa) [virtualne] relacije
- **UPDATE [(*columnList*)]**
  - izmjena n-torki (ili vrijednosti navedenih atributa) [virtualne] relacije
- **INSERT**
  - unos n-torki [virtualne] relacije
- **DELETE**
  - brisanje n-torki [virtualne] relacije
- **REFERENCES [(*columnList*)]**
  - korištenje **relacije** (ili samo navedenih atributa kao pozivane relacije pri definiranju stranog ključa)
- **INDEX**
  - kreiranje indeksa nad **relacijom**
- **ALTER**
  - izmjena strukture **relacije** i definiranje integritetskih ograničenja
- **ALL PRIVILEGES**
  - sve do sada navedene vrste operacija nad [virtualnom] relacijom

SQL naredbe za dodjeljivanje i ukidanje dozvola

- GRANT *dbPrivilege* TO { PUBLIC | *userList* }
- REVOKE *dbPrivilege* FROM { PUBLIC | *userList* }
- GRANT *tablePrivilegeList* ON { *tableName* | *viewName* }
  TO { PUBLIC | *userList* | *roleList* }
  [WITH GRANT OPTION]
- REVOKE *tablePrivilegeList* ON { *tableName* | *viewName* }
  FROM { PUBLIC | *userList* | *roleList* }
  [CASCADE | RESTRICT]

Primjer 1:

student

matBr	ime	prez	pbr	adresa
100	Ana	Ivić	51000	Korzo 2
102	Ivan	Perić	10000	Ilica 20
105	Matija	Matić	31000	Unska 7
107	Tea	Bilić	10000	Vlaška 5

ispit

matBr	nazPred	datIsp	ocj
100	Fizika	1.5.2004	3
102	Matematika	7.9.2003	1
102	Matematika	9.2.2004	5
107	Fizika	5.4.2006	4

- kreirati bazu podataka studBaza i relacije student i ispit
  - vlasnik baze podataka i relacija treba biti korisnik bpadmin
- korisnik horvat treba dobiti dozvole:
  - pregled svih podataka u relacijama student i ispit
  - unos, izmjena, brisanje svih podataka u relaciji ispit
- korisnik novak treba dobiti dozvole:
  - pregled svih podataka u relaciji student
  - izmjena poštanskog broja i adrese u relaciji student
- korisnik kolar treba dobiti dozvolu:
  - pregled svih podataka u relaciji student, osim adrese

Primjer 1 (nastavak):

bpadmin ← **naredbe obavlja korisnik bpadmin**

```
CREATE DATABASE studBaza;
CREATE TABLE student (...);
CREATE TABLE ispit (...);

GRANT CONNECT TO horvat;
GRANT CONNECT TO novak;
GRANT CONNECT TO kolar;

GRANT SELECT ON student
  TO horvat;
GRANT SELECT, INSERT
    , UPDATE, DELETE ON ispit
  TO horvat;

GRANT SELECT ON student
  TO novak;
GRANT UPDATE(pbr, adresa)
  ON student TO novak;

GRANT SELECT(matBr, ime
           , prez, pbr)
  ON student TO kolar;
```

⇨ korisnik bpadmin je vlasnik baze podataka studBaza i relacija student i ispit. Posjeduje DBA dozvolu na razini baze podataka

⇨ dozvole za uspostavljanje SQL-sjednice

⇨ dozvole korisniku horvat za pregled podataka u relaciji student

⇨ dozvole korisniku horvat za pregled, unos, izmjenu i brisanje podataka u relaciji ispit

⇨ dozvola korisniku novak za pregled podataka u relaciji student

⇨ dozvola korisniku novak za izmjenu vrijednosti atributa u relaciji student

⇨ dozvola korisniku kolar za pregled svih podataka u relaciji student, osim adrese

Primjer 2:

bpadmin

```
CREATE DATABASE studBaza;
GRANT RESOURCE TO horvat;
GRANT CONNECT TO novak;
```

⇨ korisnik bpadmin kreira bazu podataka studBaza. Kao vlasnik baze podataka implicitno dobiva DBA dozvolu na razini baze podataka

horvat

```
CREATE TABLE zupanija (
  sifZup INTEGER
, nazZup CHAR(30)
, PRIMARY KEY(sifZup));
GRANT SELECT, INSERT, UPDATE
  ON zupanija TO novak;
```

⇨ može jer ima RESOURCE dozvolu

⇨ može jer je vlasnik relacije zupanija

novak

```
SELECT * FROM zupanija;
INSERT INTO zupanija ...;
UPDATE zupanija ...;
```

⇨ može jer ima barem CONNECT dozvolu (bez CONNECT dozvole ne bi mogao uspostaviti SQL-sjednicu), te dozvole koje je dobio od vlasnika relacije zupanija

Primjer 2 (nastavak):

novak
```
DROP TABLE zupanija;
```
➡ ne može jer nije vlasnik objekta niti ima DBA dozvolu

kolar
```
SELECT * FROM zupanija;
```
➡ ne može jer nema niti CONNECT dozvolu (ne može uspostaviti SQL-sjednicu)

horvat
```
GRANT CONNECT TO kolar;
```
➡ ne može jer nema DBA dozvolu

bpadmin
```
GRANT CONNECT TO kolar;
```
➡ može jer ima DBA dozvolu

horvat
```
GRANT SELECT
  ON zupanija TO kolar;
```
➡ može jer je vlasnik relacije zupanija

kolar
```
SELECT * FROM zupanija;
```
➡ može jer ima barem CONNECT dozvolu, te dozvolu za obavljanje operacije SELECT nad relacijom zupanija

Primjer 2 (nastavak):

novak
```
CREATE TABLE mjesto ...;
```
➡ ne može jer nema RESOURCE dozvolu

horvat
```
GRANT RESOURCE TO novak;
```
➡ ne može jer nema DBA dozvolu

bpadmin
```
GRANT DBA TO horvat;
```
➡ može jer ima DBA dozvolu

horvat
```
GRANT RESOURCE TO novak;
```
➡ može jer ima DBA dozvolu

novak
```
CREATE TABLE mjesto (...
  REFERENCES zupanija ...);
```
➡ ne može jer nema dozvolu za kreiranje stranog ključa koji se poziva na primarni ključ relacije zupanija (mogao bi kreirati relaciju bez stranog ključa jer ima RESOURCE dozvolu)

Primjer 2 (nastavak):

horvat
```
GRANT REFERENCES
  ON zupanija TO novak;
```
➡ može jer ima DBA dozvolu (ali čak i da nema DBA dozvolu, vlasnik je relacije zupanija)

novak
```
CREATE TABLE mjesto (...
  REFERENCES zupanija ...);
```
➡ može jer ima RESOURCE dozvolu i dozvolu za kreiranje stranog ključa koji se poziva na primarni ključ relacije zupanija

horvat
```
GRANT CONNECT TO PUBLIC;
```
➡ može jer ima DBA dozvolu

- sada svaki korisnik (sadašnji ili budući) koji uspije ovjeriti svoju autentičnost može uspostaviti SQL-sjednicu s bazom podataka studBaza

novak
```
GRANT SELECT
  ON mjesto TO PUBLIC;
```
➡ može jer je vlasnik relacije mjesto

- sada svaki korisnik (sadašnji ili budući) koji uspostavi SQL-sjednicu s bazom podataka (uz prethodnu ovjeru autentičnosti) može obavljati operaciju SELECT nad relacijom mjesto

Dodjeljivanje prenosivih dozvola

- ukoliko se korisniku dozvola dodijeli uz navođenje opcije WITH GRANT OPTION, korisnik će moći dodjeljivati tu istu dozvolu ostalim korisnicima (unatoč tome što nije vlasnik objekta)

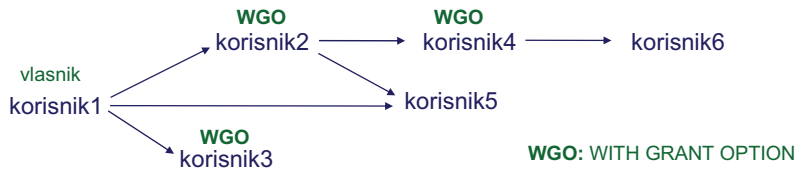
Primjer:

korisnik1
```
CREATE TABLE ispit (...);
GRANT SELECT ON ispit TO korisnik2 WITH GRANT OPTION;
GRANT SELECT ON ispit TO korisnik3 WITH GRANT OPTION;
```

korisnik2
```
GRANT SELECT ON ispit TO korisnik4 WITH GRANT OPTION;
GRANT SELECT ON ispit TO korisnik5;
```

korisnik4
```
GRANT SELECT ON ispit TO korisnik6;
```

korisnik1
```
GRANT SELECT ON ispit TO korisnik5;
```



Ukidanje dozvola

- korisnik koji je dozvolu dodijelio, tu istu dozvolu može ukinuti naredbom REVOKE

Primjer:
- vlasnik baze podataka studBaza je korisnik bpadmin
- vlasnik relacije mjesto je korisnik horvat

horvat
```
GRANT SELECT, UPDATE ON mjesto TO novak WITH GRANT OPTION;
```

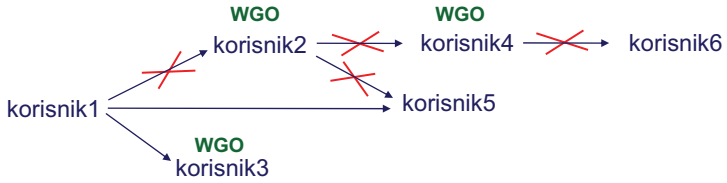
novak
```
GRANT SELECT, UPDATE ON mjesto TO kolar;
```

- npr. naredbu:
```
REVOKE UPDATE ON mjesto FROM kolar;
```
- može obaviti korisnik novak jer je novak korisnik koji je dozvolu dodijelio

Ukidanje dozvola dodijeljenih temeljem WITH GRANT OPTION

- ukidanjem dozvole korisniku x (koji je dozvole dalje dodjeljivao temeljem ovlasti stečene pomoću WITH GRANT OPTION) uz primjenu opcije CASCADE, dozvola se ukida i svim ostalim korisnicima koji su dotičnu dozvolu stekli od korisnika x (neposredno ili posredno)

Primjer: korisnik1
```
REVOKE SELECT ON ispit FROM korisnik2 CASCADE;
```

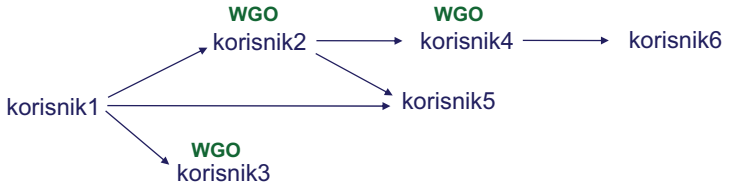


- obavljanjem naredbe dozvolu gube korisnik2, korisnik4 i korisnik6
- korisnik5 će izgubiti dozvolu koju je dobio od korisnika2, ali će zadržati dozvolu koju je dobio od korisnika1
- ukoliko se opcija CASCADE ne navede, naredba REVOKE djeluje na jednak način kao kada je opcija CASCADE navedena

Ukidanje dozvola dodijeljenih temeljem WITH GRANT OPTION

- ukidanjem dozvole korisniku x uz primjenu opcije RESTRICT, dozvola će biti ukinuta jedino u slučaju kada korisnik x nije dalje dodjeljivao ovlasti temeljem ovlasti stečene pomoću WITH GRANT OPTION

Primjer:



korisnik1
```
REVOKE SELECT ON ispit FROM korisnik2 RESTRICT;
```
SUBP odbija obaviti naredbu (dojavljuje pogrešku)

korisnik2
```
REVOKE SELECT ON ispit FROM korisnik4 RESTRICT;
```
SUBP odbija obaviti naredbu (dojavljuje pogrešku)

korisnik1
```
REVOKE SELECT ON ispit FROM korisnik3 RESTRICT;
```
SUBP obavlja naredbu (korisnik3 ostaje bez dozvole)

Primjena virtualnih relacija

ispit

mbrSt	nazPred	datIsp	ocj
100	Fizika	1.5.2004	3
102	Matematika	7.9.2003	1
102	Matematika	9.2.2004	5
107	Fizika	5.4.2006	4

- vlasnik relacije ispit je korisnik horvat
- korisniku novak omogućiti pregled samo prosječnih ocjena po predmetima
- korisniku kolar omogućiti pregled, unos, izmjenu i brisanje samo za ispite iz predmeta Fizika

horvat
```
CREATE VIEW prosjek (nazPred, prosOcj) AS
  SELECT nazPred, AVG(ocj)
    FROM ispit
    GROUP BY nazPred;
GRANT SELECT ON prosjek TO novak;
CREATE VIEW ispitFizika AS
  SELECT * FROM ispit
    WHERE nazPred = 'Fizika'
  WITH CHECK OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE
  ON ispitFizika TO kolar;
```

- zašto je nužno virtualnu relaciju ispitFizika kreirati uz opciju WITH CHECK OPTION?!

Dodjeljivanje kontekstno ovisnih dozvola

ispit

mbrSt	sifPred	datIsp	ocj
100	100	1.5.2004	3
102	200	7.9.2003	1
102	200	9.2.2004	5
107	300	5.4.2006	4

nast

sifNast	imeN	prezN	userId
1001	Slavko	Kolar	kolar
1002	Ivo	Ban	ban
1003	Ana	Novak	novak

predaje

sifNast	sifPred
1001	100
1001	200
1002	200
1003	200
1003	300

- vlasnik relacija je korisnik horvat
- svakom nastavniku (korisnicima kolar, ban, novak) omogućiti pregled i izmjenu ispita samo iz predmeta koje predaju

LOŠE RJEŠENJE!

horvat

```
CREATE VIEW kolarIspiti AS
  SELECT * FROM ispit
    WHERE sifPred IN (
      SELECT sifPred FROM predaje
        WHERE sifNast = 1001) WITH CHECK OPTION;
GRANT SELECT, UPDATE ON kolarIspiti TO kolar;
```

- ponoviti za svakog nastavnika: banIspiti, novakIspiti, ...
- nova virtualna relacija za svakog novog nastavnika (≈150 na FER-u)
- svaki nastavnik upit nad relacijom ispit mora pisati na drugačiji način

Dodjeljivanje kontekstno ovisnih dozvola

ispit

mbrSt	sifPred	datIsp	ocj
100	100	1.5.2004	3
102	200	7.9.2003	1
102	200	9.2.2004	5
107	300	5.4.2006	4

nast

sifNast	imeN	prezN	userId
1001	Slavko	Kolar	kolar
1002	Ivo	Ban	ban
1003	Ana	Novak	novak

predaje

sifNast	sifPred
1001	100
1001	200
1002	200
1003	200
1003	300

ISPRAVNO RJEŠENJE!

horvat

```
CREATE VIEW ispitiZaNastavnike AS
  SELECT * FROM ispit
    WHERE sifPred IN (
      SELECT sifPred FROM predaje, nast
       WHERE predaje.sifNast = nast.sifNast
         AND userId = USER) WITH CHECK OPTION;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO kolar;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO ban;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO novak;
```

- "sadržaj" virtualne relacije ovisit će o identifikatoru nastavnika koji je ostvario SQL-sjednicu
- smije li se nastavnicima dozvoliti izmjena vrijednosti atributa userId u relaciji nast ili sadržaj relacije predaje?!

Upotreba sinonima

PROBLEM:

- nastavnici (odnosno aplikativni ili primjenski programi koje nastavnici koriste) moraju u upitima o ispitima koristiti virtualnu relaciju ispitiZaNastavnike

```
SELECT * FROM ispitiZaNastavnike WHERE ocj = 1;
```

- dekan (npr. korisnik s identifikatorom novosel), za razliku od nastavnika, dobiva sve dozvole nad relacijom ispit. U upitima o ispitima mora koristiti relaciju ispit

```
SELECT * FROM ispit WHERE ocj = 1;
```

- kada korisnik novosel prestane biti dekan, ukinut će mu se dozvola nad relacijom ispit, a dodijeliti dozvola nad virtualnom relacijom ispitiZaNastavnike. U svojim upitima morat će koristiti virtualnu relaciju ispitiZaNastavnike

```
SELECT * FROM ispitiZaNastavnike WHERE ocj = 1;
```


Upotreba sinonima

RJEŠENJE:

- Kreirati sinonime: alternativna imena za relacije ili virtualne relacije

korisnik s DBA dozvolom

```
CREATE PRIVATE SYNONYM kolar.ispitiZaSve FOR ispitiZaNastavnike;
CREATE PRIVATE SYNONYM ban.ispitiZaSve FOR ispitiZaNastavnike;
... sinonimi za ostale nastavnike i sinonim za dekana
CREATE PRIVATE SYNONYM novosel.ispitiZaSve FOR ispit;
```

- sada i dekan i nastavnici mogu koristiti isto ime objekta kada postavljaju upite o ispitima

```
SELECT * FROM ispitiZaSve WHERE ocj = 1;
```

- kada korisnik novosel prestane biti dekan

horvat

```
REVOKE SELECT, UPDATE ON ispit FROM novosel;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO novosel;
```

korisnik s DBA dozvolom

```
DROP SYNONYM novosel.ispitiZaSve;
CREATE PRIVATE SYNONYM novosel.ispitiZaSve FOR ispitiZaNastavnike;
```

- korisnik novosel će i dalje u svojim upitima moći koristiti ime objekta ispitiZaSve, ali će kao rezultat dobivati samo one podatke na koje, sada u svojstvu nastavnika, ima pravo

Dodjeljivanje istih dozvola velikom broju korisnika

PROBLEM:

- svakom nastavniku treba dodijeliti dozvole za
  - pregled, unos i izmjenu podataka o ispitima za predmete koje predaje, pregled podataka iz relacije nast, iz relacije predaje, itd.
  - 150 nastavnika⇒150 puta treba obaviti niz naredbi za dodjelu dozvola:

```
GRANT SELECT, INSERT, UPDATE ON ispitiZaNastavnike TO kolar;
GRANT SELECT ON predmet TO kolar;
GRANT SELECT ON nast TO kolar;
...
-- ponoviti za svakog od 150 nastavnika
```

- za svakog novog zaposlenog nastavnika ponoviti postupak
- kada nastavnik ode u mirovinu, mora se obaviti niz REVOKE naredbi
- ako se promijene pravila pristupa (npr. odluči se da nastavnici mogu brisati "svoje" ispite), promjena se mora provesti za svakog nastavnika posebno:

```
GRANT DELETE ON ispitiZaNastavnike TO kolar;
-- ponoviti za svakog od 150 nastavnika
```


Dodjeljivanje istih dozvola velikom broju korisnika

RJEŠENJE:

- definira se uloga (*role*), npr. nastavnik
- dozvole se, umjesto direktno korisnicima-nastavnicima, dodjeljuju ulozi

```
CREATE ROLE nastavnik;
GRANT SELECT, INSERT, UPDATE ON ispitiZaNastavnike TO nastavnik;
GRANT SELECT ON nast TO nastavnik;
GRANT SELECT ON predaje TO nastavnik;
...
```

- svakom nastavniku, umjesto cijelog niza dozvola, dovoljno je dodijeliti dozvolu za korištenje uloge nastavnik

```
GRANT nastavnik TO kolar;
GRANT nastavnik TO ban;
...
```

- ako nastavnik s identifikatorom korisnika ban ode u mirovinu

```
REVOKE nastavnik FROM ban;
```

- ako nastavnici trebaju dobiti dozvolu za brisanje "svojih" ispita

```
GRANT DELETE ON ispitiZaNastavnike TO nastavnik;
```


Korištenje dozvola dobivenih putem uloga

- nakon uspostavljanja SQL-sjednice, korisnik posjeduje sljedeće dozvole:
  1. sve dozvole koje su dodijeljene PUBLIC "korisniku"
  2. sve dozvole koje su dodijeljene izravno dotičnom korisniku
  3. sve dozvole nad objektima kojima je dotični korisnik vlasnik
  4. dozvole na razini baze podataka (npr. ako korisnik ima DBA dozvolu, dopušteno mu je obavljanje svih operacija nad svim objektima)
- ako korisnik namjerava koristiti i dozvole dodijeljene nekoj ulozi, mora obaviti naredbu (npr.): `SET ROLE nastavnik;`
  - od tog trenutka, korisnik će (osim dozvola navedenih pod 1-4) imati i dozvole dodijeljene ulozi nastavnik.
- korisniku može biti dodijeljena više nego jedna uloga, ali u jednom trenutku može koristiti samo jednu od njih. Npr. nakon obavljanja naredbe:

```
SET ROLE studentskiSavjetnik;
```

  - korisnik će (osim dozvola navedenih pod 1-4) imati i dozvole dodijeljene ulozi studentskiSavjetnik (ali ne i ulozi nastavnik).
- naredbu `SET ROLE NONE;` korisnik koristi onda kad ne želi koristiti niti jednu ulogu

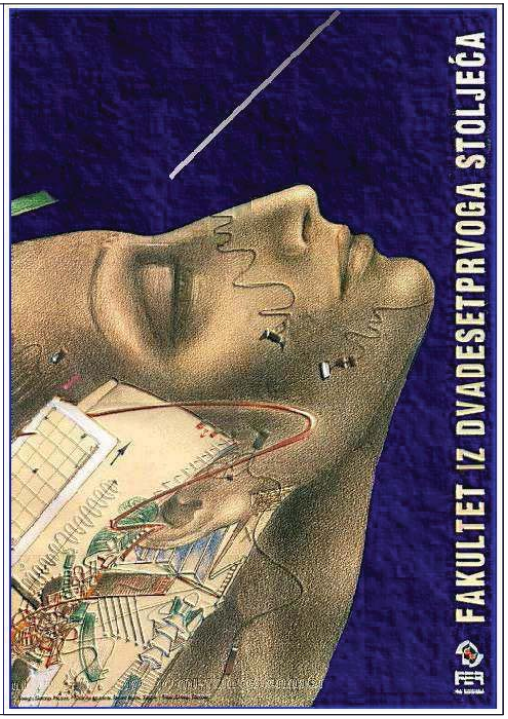
Praćenje rada korisnika (*auditing*)

- evidentirati svaki pristup osjetljivim podacima u posebnoj datoteci za praćenje rada korisnika (*Audit Trail*)
- tipičan zapis datoteke sadrži sljedeće informacije:
  - SQL naredba koja se izvršava (*statement source*)
  - mjesto s kojeg je upućen zahtjev (terminal, IP adresa računala)
  - identifikator korisnika koji je pokrenuo operaciju
  - datum i vrijeme operacije
  - n-torke, atributi na koje se zahtjev odnosi
  - stara vrijednost n-torke
  - nova vrijednost n-torke
- sama činjenica da se prati "trag" obavljenih operacija nad podacima, često je dovoljna za sprečavanje zloporabe

Baze podataka

Predavanja
lipanj 2009.

16. Pohranjene procedure i okidači



Pohranjene procedure

Primjer 1:

	CHAR(13)	CHAR(30)
osoba	jmbg	prez
	2203979622876	Horvat
	1712 12871211	Kolar
	2707986736233	Še5fer
	03AB621.22876	Novak

- smatra se da su ispravne one vrijednosti atributa jmbg u kojima postoji točno 13 znamenaka
- smatra se da su ispravna ona prezimena u kojima ne postoji niti jedna znamenka
- ispisati podatke o osobama s neispravnim jmbg ili prezimenom
- kad bi barem postojala SQL funkcija CountDigits(nizZnakova)

```
SELECT * FROM osoba
  WHERE CountDigits(jmbg) <> 13
     OR CountDigits(prez) > 0;
```


Pohranjene procedure (pohranjene funkcije)

- Pohranjena procedura ili pohranjena funkcija je potprogram koji je pohranjen u rječniku podataka i koji se izvršava u kontekstu sustava za upravljanje bazama podataka
  - može se promatrati kao procedura ili funkcija kojom se proširuje skup SQL funkcija ugrađenih u SUBP
    - procedura je potprogram koji u pozivajući program ne vraća rezultat
    - funkcija je potprogram koji u pozivajući program vraća rezultat

Primjer 1 (nastavak):

- Funkcija koja u zadanom nizu znakova broji koliko ima znakova koji su znamenke (broji znakove iz intervala '0' ... '9'). Pretpostavlja se da duljina zadanog niza znakova ne premašuje 255 bajtova

```
CREATE FUNCTION brojZnamenki (niz CHAR(255))
                    RETURNING SMALLINT AS broj
  DEFINE brojac, i SMALLINT;

  LET brojac = 0;
  FOR i = 1 TO CHAR_LENGTH(niz)
    IF SUBSTRING(niz FROM i FOR 1) BETWEEN '0' AND '9' THEN
      LET brojac = brojac + 1;
    END IF;
  END FOR;
  RETURN brojac;
END FUNCTION;

GRANT EXECUTE ON brojZnamenki TO PUBLIC;
```

- funkciju brojZnamenki svaki (sadašnji i budući) korisnik može koristiti na jednak način kao što se koriste standardne SQL funkcije

Primjer 1 (nastavak):

osoba

jmbg (CHAR(13))	prez (CHAR(30))
2203979622876	Horvat
1712 12871211	Kolar
2707986736233	Še5fer
03AB621.22876	Novak

- funkcija brojZnamenki se može iskoristiti za ispis onih osoba u čijem jmbg nema točno 13 znamenaka ili u prezimenu postoje znamenke

```
SELECT *, brojZnamenki(jmbg) AS br1, brojZnamenki(prez) AS br2
  FROM osoba
  WHERE brojZnamenki(jmbg) <> 13 OR brojZnamenki(prez) > 0;
```

jmbg	prez	br1	br2
1712 12871211	Kolar	12	0
2707986736233	Še5fer	13	1
03AB621.22876	Novak	10	0

Primjer 1 (nastavak):

- Pohranjena funkcija se iz interaktivnih alata (npr. Server Studio) može pozvati na sljedeći način:

```
EXECUTE FUNCTION brojZnamenki('abc123efg456');
```

broj
6

```
CREATE FUNCTION brojZnamenki (niz CHAR(255))
                    RETURNING SMALLINT AS broj
  ...
```


Primjer 2:

- Korisnik novak je službenik u banci kojem je potrebno omogućiti obavljanje isključivo jedne vrste bankovne transakcije: prebacivanje iznosa s jednog na drugi račun

racun

brRacun	stanje
1001	1250.15
1002	-300.00
1003	10.25

- Zadatak se ne može riješiti dodjelom dozvole za obavljanje operacije UPDATE nad relacijom racun korisniku novak (zašto?)

Dozvole za pohranjene procedure/funkcije

- SQL naredbe za dodjeljivanje i ukidanje dozvola za izvršavanje procedura
- `GRANT EXECUTE ON {procName | funName}`
  `TO {PUBLIC | userList | roleList}`
  `[WITH GRANT OPTION]`
- `REVOKE EXECUTE ON {procName | funName}`
  `FROM {PUBLIC | userList | roleList}`
  `[ CASCADE | RESTRICT ]`

Primjer 2 (nastavak):

```
CREATE PROCEDURE prebaci (saRacunaBr LIKE racun.brRacun
                        , naRacunBr  LIKE racun.brRacun
                        , iznos      LIKE racun.stanje)
  -- prenesi zadani iznos
  UPDATE racun SET stanje = stanje - iznos
    WHERE brRacun = saRacunaBr;
  UPDATE racun SET stanje = stanje + iznos
    WHERE brRacun = naRacunBr;
END PROCEDURE;
GRANT EXECUTE ON prebaci TO novak;
```


Primjer 2 (nastavak):

racun

brRacun	stanje
1001	1250.15
1002	-300.00
1003	10.25

novak
```
UPDATE racun SET stanje = stanje - 60.30
  WHERE brRacun = 1001;
```
`[Error] No UPDATE permission`

novak
```
EXECUTE PROCEDURE prebaci (1001, 1002, 60.30);
```

racun

brRacun	stanje
1001	1189.85
1002	-239.70
1003	10.25

- Problem: što će se dogoditi ako korisnik pri pozivu procedure kao broj prvog računa zada postojeći, a kao broj drugog računa zada nepostojeći broj računa?

```
EXECUTE PROCEDURE prebaci(1001, 1005, 30.15);
```


Iznimke (*Exceptions*)

- ukoliko SUBP tijekom obavljanja operacije utvrdi da se dogodila pogreška (*error condition*), obavljanje operacije se prekida, a stanje pogreške se signalizira iznimkom (*exception*)

```
SELECT (stanje/(stanje-10.25)) FROM racun;
```
`[Error] An attempt was made to divide by zero.`

```
SELECT * FROM ispit;
```
`[Error] No SELECT permission.`

- pogreške koje SUBP nije u stanju prepoznati (jer ih ne smatra pogreškama), mogu se signalizirati naredbom RAISE EXCEPTION. Npr, u poboljšanoj proceduri **`prebaci`** signalizira se pogreška u slučaju kad ne postoji neki od zadanih brojeva računa

```
EXECUTE PROCEDURE prebaci(1001, 1005, 30.15);
```
`[Error] Ne postoji drugi račun`

Primjer 2 (nastavak):

```
CREATE PROCEDURE prebaci (saRacunaBr LIKE racun.brRacun
                        , naRacunBr  LIKE racun.brRacun
                        , iznos      LIKE racun.stanje)
  -- provjeri postoje li zadani brojevi računa
  IF (SELECT COUNT(*) FROM racun
        WHERE brRacun = saRacunaBr) = 0 THEN
    RAISE EXCEPTION -746, 0, 'Ne postoji prvi račun';
  END IF;
  IF (SELECT COUNT(*) FROM racun
        WHERE brRacun = naRacunBr) = 0 THEN
    RAISE EXCEPTION -746, 0, 'Ne postoji drugi račun';
  END IF;
  -- prenesi zadani iznos
  UPDATE racun SET stanje = stanje - iznos
    WHERE brRacun = saRacunaBr;
  UPDATE racun SET stanje = stanje + iznos
    WHERE brRacun = naRacunBr;
END PROCEDURE;
GRANT EXECUTE ON prebaci TO novak;
```

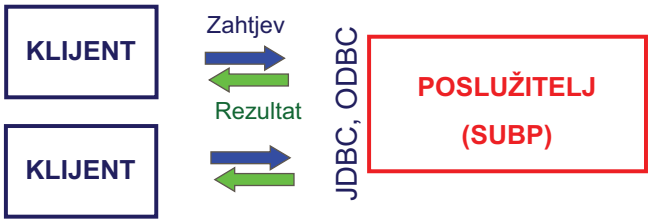

SPL (*Stored Procedure Language*)

- Proizvođači SUBP koriste vlastite inačice jezika za definiranje pohranjenih procedura (standard postoji, ali je rijetko gdje implementiran)
  - IBM Informix: SPL (Stored Procedure Language)
  - Oracle: PL/SQL (Procedural Language/Structured Query Language)
  - Microsoft SQL Server: Transact-SQL
- Navedeni jezici proširuju mogućnosti SQL jezika proceduralnim elementima koji se koriste u strukturiranim jezicima (C, Java, ...). Osim SQL naredbi, pohranjene procedure omogućuju korištenje
  - varijabli
  - naredbi za kontrolu toka programa (*if*, *for*, *while*, ...)
  - naredbi za rukovanje iznimkama (*exception handling*)

Prednosti uporabe pohranjenih procedura

- proširenje mogućnosti SQL jezika
- omogućena je zaštita podataka na razini funkcije (a ne samo objekta)
- omogućena je uporaba klijent-poslužitelj arhitekture oslonjene na poslužitelj:
  - postiže se veća učinkovitost SUBP
    - SUBP ne mora ponavljati prevođenje i optimiranje SQL upita
  - postiže se veća produktivnost programera i smanjuje mogućnost pogreške
    - programski kôd potreban za obavljanje nekog postupka koji čini logičku cjelinu implementira se i testira na samo jednom mjestu

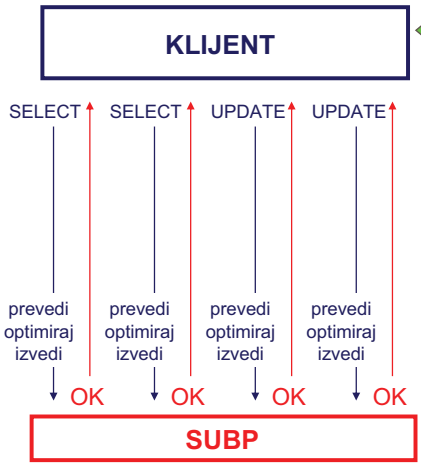
(Klijent-poslužitelj arhitektura)



- sustav obuhvaća dvije komponente
  - klijent i poslužitelj (*client-server*)
- koncept zahtjev-odgovor (*request-response*): klijent postavlja zahtjev, poslužitelj odgovara
- komunikacija između klijenta i poslužitelja se odvija preko dobro definiranih, standardnih programskih sučelja: npr. ODBC (*Open Database Connectivity*), JDBC (*Java Database Connectivity*)

(Klijent-poslužitelj arhitektura - oslonjena na klijenta)

- provjeri postoje li zadani brojevi računa, ako postoje, prebaci iznos s jednog na drugi račun

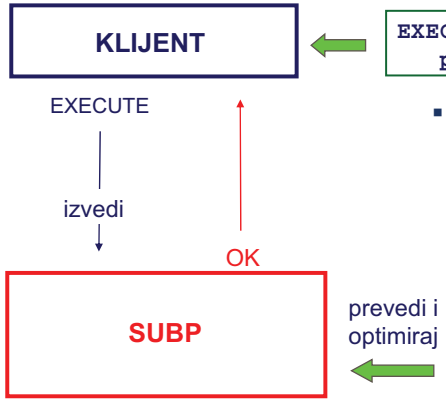



```
SELECT COUNT(*) FROM racun
  WHERE brRacun = 1001;
  ?
SELECT COUNT(*) FROM racun
  WHERE brRacun = 1002;
  ?
UPDATE racun
  SET stanje = stanje - 60.30
  WHERE brRacun = 1001;
UPDATE racun
  SET stanje = stanje + 60.30
  WHERE brRacun = 1002;
```

- niz zahtjeva za obavljanje jedne po jedne SQL naredbe

(Klijent-poslužitelj arhitektura - oslonjena na poslužitelj)

- provjeri postoje li zadani brojevi računa, ako postoje, prebaci iznos s jednog na drugi račun




```
EXECUTE PROCEDURE
   prebaci (1001, 1002, 60.30);
```

- jedan zahtjev koji rezultira obavljanjem niza SQL naredbi koje čine logičku cjelinu

- U rječnik podataka se pohranjuje:
  - izvorni kôd procedure
  - prevedeni i optimirani kôd procedure

```
CREATE PROCEDURE prebaci(...)
  DEFINE ...
  SELECT ...
  SELECT ...
  UPDATE ...
  UPDATE ...
END PROCEDURE;
```


Okidači

Primjer 3:

racun

brRac	sifKlijent	stanje
1001	98281	216.80
1002	89734	134.99
1003	23232	2750.00
1004	63443	849.50
...		...

uplataIsplata

brRac	vrijeme	iznos
1001	7.8.2007 08:20	15.00
1002	9.4.2006 12:31	-100.21
1001	6.5.2007 14:15	452.15
1004	5.5.2007 16:42	1200.00
1004	9.9.2005 10:15	-350.50
1002	7.2.2007 15:01	235.20
1003	1.4.2005 12:44	2750.00
1001	1.9.2007 12:19	-250.35
...	...	...

- u relaciju uplataIsplata upisuju se promjene na računima
- tijekom godina evidentiran je vrlo veliki broj uplata i isplata
- stanje na određenom računu moglo bi se izračunati zbrajanjem iznosa u relaciji uplataIsplata, koji se odnose na dotični račun
- u ovom primjeru, uz svaki račun se redundantno pohranjuje trenutno stanje računa, koje u svakom trenutku mora odgovarati stanju koje bi se dobilo zbrajanjem iznosa u relaciji uplataIsplata
- kako osigurati da se pri svakoj relevantnoj promjeni podataka (unos, brisanje, izmjena iznosa) u relaciji uplataIsplata izmijeni i odgovarajuće stanje u relaciji racun?

Aktivne baze podataka

- konvencionalni SUBP je pasivan
  - operacije nad podacima se izvršavaju isključivo na temelju eksplicitnog zahtjeva korisnika/aplikacije
- aktivni SUBP i aktivne baze podataka
  - aktivni SUBP autonomno reagira na određene događaje (*events*)
  - u aktivnim bazama podataka neke operacije nad podacima se izvršavaju automatski, reakcijom na određeni događaj ili stanje
- željeno ponašanje sustava postiže se definiranjem aktivnih pravila (*active rules*)
- najčešće korištena paradigma za opisivanje aktivnih pravila u današnjim SUBP je događaj-uvjet-akcija (*ECA: Event-Condition-Action*)
  - okidači (*triggers*)

ECA

```
on event
   if condition then action
```

- događaj (*event*): ako se dogodi, izračunava se uvjet
  - općenito, događaji mogu biti:
    - unos, izmjena ili brisanje podatka
    - čitanje podatka
    - uspostavljanje SQL-sjednice
    - protok određene količine vremena, dostizanje trenutka u vremenu, ...
- uvjet (*condition*): ako je rezultat izračunavanja uvjeta istina, obavljaju se akcije
  - zadaje se u obliku predikata (slično kao u WHERE dijelu SQL naredbi)
- akcije (*action*): niz operacija, najčešće operacije nad podacima
  - SQL naredbe INSERT, UPDATE, DELETE, poziv procedure, ...

Primjer 3 (nastavak):

- kako osigurati da se pri svakoj relevantnoj promjeni podataka (unos, brisanje, izmjena iznosa) u relaciji uplataIsplata izmijeni i odgovarajuće stanje u relaciji racun?

racun

brRac	sifKlijent	stanje
1001	98281	216.80
1002	89734	134.99
...		...

uplataIsplata

brRac	vrijeme	iznos
1001	7.8.2007 08:20	15.00
1002	9.4.2006 12:31	-100.21
1001	6.5.2007 14:15	452.15
...	...	...

- potrebno je utvrditi koji događaji mogu uzrokovati neispravnu vrijednost atributa stanje u relaciji racun, te pod kojim uvjetima treba obaviti koje akcije kako bi se očuvao integritet podataka, npr.
  - događaj: obavljanje operacije INSERT nad relacijom uplataIsplata
  - uvjet:iznos <> 0.00
  - akcija: pribrojiti vrijednost atributa iznos unesene n-torke u odgovarajuće stanje

Primjer 3 (nastavak):

- događaj: obavljanje operacije INSERT nad relacijom uplataIsplata
- uvjet:iznos <> 0.00
- akcija: pribrojiti vrijednost atributa iznos unesene n-torke u odgovarajuće stanje

```
CREATE TRIGGER insUplataIsplata
  INSERT ON uplataIsplata
  REFERENCING NEW AS novaUplataIsplata
  FOR EACH ROW
    WHEN (novaUplataIsplata.iznos <> 0)
      (UPDATE racun SET stanje = stanje + novaUplataIsplata.iznos
        WHERE brRac = novaUplataIsplata.brRac);
```

- kad god se obavi naredba INSERT nad relacijom uplataIsplata SUBP obavlja
  - nakon unosa svake n-torke (jednom INSERT naredbom može se unijeti više n-torki) provjerava uvjet `novaUplataIsplata.iznos <> 0`
  - na sadržaj unesene n-torke može se referencirati koristeći "ime" n-torke koje je zadano pomoću `REFERENCING NEW AS novaUplataIsplata`
  - ako je uvjet zadovoljen (za dotičnu n-torku), obavlja izmjenu stanja u relaciji racun

Primjer 3 (nastavak):

- događaj: brisanje n-torke iz relacije uplataIsplata
- uvjet:iznos <> 0.00
- akcija: oduzeti vrijednost atributa iznos unesene n-torke od odgovarajućeg stanja

```
CREATE TRIGGER delUplataIsplata
  DELETE ON uplataIsplata
  REFERENCING OLD AS brisanaUplataIsplata
  FOR EACH ROW
    WHEN (brisanaUplataIsplata.iznos <> 0)
      (UPDATE racun SET stanje = stanje - brisanaUplataIsplata.iznos
       WHERE brRac = brisanaUplataIsplata.brRac);
```

- ukoliko je potrebno, moguće je navesti više SQL naredbi, međusobno odijeljenih zarezima
- SQL naredbe koje se mogu koristiti za opisivanje akcije:
  - INSERT
  - UPDATE
  - DELETE
  - EXECUTE PROCEDURE

Primjer 3 (nastavak):

- događaj: izmjena vrijednosti atributa iznos u relaciji uplataIsplata
- uvjet:nova vrijednost iznosa <> stara vrijednost iznosa
- akcija: u odgovarajuće stanje pribrojiti razliku između nove i stare vrijednosti atributa iznos

```
CREATE TRIGGER updIznosUplataIsplata
  UPDATE OF iznos ON uplataIsplata
  REFERENCING OLD AS staraUplataIsplata NEW AS novaUplataIsplata
  FOR EACH ROW
    WHEN (novaUplataIsplata.iznos <> staraUplataIsplata.iznos)
      (UPDATE racun SET stanje = stanje +
              novaUplataIsplata.iznos - staraUplataIsplata.iznos
       WHERE brRac = staraUplataIsplata.brRac);
```

- **`UPDATE OF iznos ON uplataIsplata`**: događaj izmjene vrijednosti atributa iznos u relaciji uplataIsplata
- **`UPDATE OF a, b, c ON relacija`**: događaj izmjene vrijednosti bilo kojeg od atributa a, b, c u relaciji
- **`UPDATE ON relacija`**: događaj izmjene vrijednosti bilo kojeg atributa u relaciji

Naredba CREATE TRIGGER

- oblik naredbe za kreiranje okidača propisan je SQL standardom, ali SUBP koriste uglavnom vlastite inačice
- jedna od važnijih mogućnosti koje su na raspolaganju pri definiciji okidača:
  - moguće je specificirati da li se akcije navedene u okidaču obavljaju:
    - po jednom za svaku n-torku na koju je djelovala operacija koja je aktivirala okidač (operacija koja je uzrokovala događaj)
      - FOR EACH ROW
    - samo jednom, nakon što se obavi operacija koja je aktivirala okidač
      - AFTER INSERT, AFTER UPDATE, AFTER DELETE
    - samo jednom, prije nego se obavi operacija koja je aktivirala okidač
      - BEFORE INSERT, BEFORE UPDATE, BEFORE DELETE
- uništavanje okidača: DROP TRIGGER *imeOkidača*

Primjena okidača

- implementacija integritetskih ograničenja
  - okidače treba koristiti onda kada integritetska ograničenja nije moguće opisati na drugi način (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, ...)
    - obavljanjem korektivne akcije koja bazu podataka dovodi u konzistentno stanje (primjer 3)
    - odbijanjem operacije koja narušava integritetsko ograničenje (primjer 4)
- praćenje rada korisnika (primjer 5)
- sustavi obavještavanja (primjer 6)
- itd.

Primjer 4:

- u relaciji ispit osigurati integritetsko ograničenje prema kojem je promjena ocjena dopuštena samo ako se mijenja na nižu ocjenu, npr.
  - dopušteno je ocjenu izvrstan promijeniti u dobar
  - nije dopušteno ocjenu dovoljan promijeniti u vrlo dobar

ispit

matBr	sifPred	datIsp	ocj	sifNast
100	1001	29.06.2006	3	1111
100	1001	05.02.2006	1	3333
101	1002	27.06.2006	2	2222
102	1001	29.01.2006	1	2222

- očito je da ne postoji korektivna akcija koja bi bazu podataka mogla dovesti u konzistentno stanje nakon što korisnik obavi naredbu:

```
UPDATE ispit SET ocjena = 2
  WHERE ocjena = 1;
```

- jedini način na koji se može osigurati navedeno integritetsko ograničenje jest: odbiti izvršavanje takve naredbe

Primjer 4 (nastavak):

```
CREATE PROCEDURE dojaviPogreskuUvecanjeOcjene ()
   RAISE EXCEPTION -746, 0, 'Ocjena se ne smije uvećati';
END PROCEDURE;

CREATE TRIGGER updOcjIspit
  UPDATE OF ocj ON ispit
  REFERENCING OLD AS stariIspit NEW AS noviIspit
  FOR EACH ROW
    WHEN (noviIspit.ocj > stariIspit.ocj)
      (EXECUTE PROCEDURE dojaviPogreskuUvecanjeOcjene());
```

- što se dešava pri izvršavanju naredbe

```
UPDATE ispit SET ocj = 2
  WHERE matBr = 100;
```

- nakon promjene prve n-torke, akcije iz okidača se neće obaviti jer uvjet za obavljanje akcije nije ispunjen
- nakon promjene druge n-torke, aktivirat će se akcija iz okidača
  - poziva se procedura
  - procedura signalizira pogrešku
  - budući da se naredba mora obaviti u cjelosti ili uopće ne, sustav poništava i promjenu prve n-torke, a korisniku prikazuje opis pogreške

Primjer 5:

- pretpostavi li se da je izmjena podataka u relaciji racun naročito osjetljiva operacija - potrebno je pratiti rad korisnika (*audit trail*)

```
CREATE TABLE auditTrailZaRacun (
  korisnik    CHAR(32)
, vrijeme     DATETIME YEAR TO SECOND
, brRac1      ...
, sifKlijent1 ...
, stanje1     ...
, brRac2      ...
, sifKlijent2 ...
, stanje2     ...
);
```

```
CREATE TRIGGER updRacun
  UPDATE ON racun
  REFERENCING OLD AS stari NEW AS novi
  FOR EACH ROW
     -- uvjet se može ispustiti
    (INSERT INTO auditTrailZaRacun VALUES (
       USER, CURRENT, stari.brRac, stari.sifKlijent, stari.stanje,
       novi.brRac, novi.sifKlijent, novi.stanje));
```


Primjer 5 (nastavak):

```
...
FOR EACH ROW
  (INSERT INTO auditTrailZaRacun VALUES (
     USER, CURRENT, stari.brRac, stari.sifKlijent, stari.stanje,
     novi.brRac, novi.sifKlijent, novi.stanje));
```

- što se dešava obavljanjem naredbe

novak

```
UPDATE racun
  SET stanje = stanje + 10
  WHERE brRac BETWEEN 1002 AND 1003;
```

racun

brRac	sifKlijent	stanje
1001	98281	216.80
1002	89734	134.99
1003	23232	2750.00
1004	63443	849.50

- osim promjene u relaciji racun, u relaciju auditTrailZaRacun bit će dodane dvije n-torke

auditTrailZaRacun

korisnik	vrijeme	brRac1	sifKlijent1	stanje1	brRac2	sifKlijent2	stanje2
...	...	...	...	...	...	...	...
novak	2007.02.27 14:13:47	1002	89734	134.99	1002	89734	144.99
novak	2007.02.27 14:13:47	1003	23232	2750.00	1003	23232	2760.00

Primjer 6:

- postoji pohranjena procedura saljiPostu(adresa, tekst)
- u relaciji artikl nalaze se podaci o artiklima na skladištu. Za svaki artikl prati se trenutno stanje (količina) artikla
- kada stanje artikla padne ispod optimalne količine, potrebno je na e-mail adresu djelatnika zaduženog za nabavu tog artikla poslati poruku

artikl

sifArt	stanje	optKol	adresaZaduzenog
1001	250	150	pero@tvrtka.hr
1002	400	200	joza@tvrtka.hr
1003	450	350	jura@tvrtka.hr

```
CREATE TRIGGER updArtikl
  UPDATE OF stanje ON artikl
  REFERENCING OLD AS stari NEW AS novi
  FOR EACH ROW
    WHEN (stari.stanje >= stari.optKol
          AND novi.stanje < stari.optKol)
   (EXECUTE PROCEDURE saljiPostu(stari.adresaZaduzenog
                              , 'Nabavi artikl: ' || stari.sifArt));
```


Primjer 6 (nastavak):

artikl

sifArt	stanje	optKol	adresaZaduzenog
1001	250	150	pero@tvrtka.hr
1002	400	200	joza@tvrtka.hr
1003	450	350	jura@tvrtka.hr

- rezultat obavljanja naredbe

```
UPDATE artikl
  SET stanje = stanje - 150;
```

artikl

sifArt	stanje	optKol	adresaZaduzenog
1001	100	150	pero@tvrtka.hr
1002	250	200	joza@tvrtka.hr
1003	300	350	jura@tvrtka.hr

  - + dvije poruke

pero@tvrtka.hr: Nabavi artikl: 1001
jura@tvrtka.hr: Nabavi artikl: 1003

- ako se nakon toga obavi naredba

```
UPDATE artikl
  SET stanje = stanje - 100;
```

artikl

sifArt	stanje	optKol	adresaZaduzenog
1001	0	150	pero@tvrtka.hr
1002	150	200	joza@tvrtka.hr
1003	200	350	jura@tvrtka.hr

  - + poruka

joza@tvrtka.hr: Nabavi artikl: 1002

KRATKI PRIRUČNIK ZA SPL

Kreiranje pohranjene procedure

- Procedura se kreira SQL naredbom oblika:

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
   tijelo procedure
END PROCEDURE;
```

```
CREATE FUNCTION imeFunkcije (eventualni argumenti)
   tijelo funkcije
END FUNCTION;
```

- Eventualne pogreške u sintaksi naredbe sustav će dojaviti za vrijeme obavljanja naredbe (na isti način kao i pogreške za vrijeme obavljanja ostalih SQL naredbi).
- Brisanje (uništavanje) procedure

```
DROP PROCEDURE imeProcedure;
DROP FUNCTION imeFunkcije;
```

- Izmjena procedure: brisanjem starog objekta i definiranjem novog objekta pod istim imenom, npr.

```
DROP PROCEDURE imeProcedure;
CREATE PROCEDURE imeProcedure ...;
```

Struktura pohranjene procedure

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
   definicija varijabli
   naredba;
   naredba;
   ...
END PROCEDURE;
```

```
CREATE FUNCTION imeProcedure (eventualni argumenti)
   definicija varijabli
   naredba;
   naredba;
   ...
END FUNCTION;
```

- Naredbe procedure završavaju znakom ; (točka-zarez)

Definicija varijabli

- Definicije varijabli se navode na početku procedure. Sadržaj varijable je nedefiniran dok mu se ne pridruži neka vrijednost.
- Tipovi varijabli mogu biti definirani eksplicitno:

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
   DEFINE ime CHAR(20);
   DEFINE ocjena, brojIzlazaka SMALLINT;
   ...
```

- ili implicitno, prema tipovima atributa u relacijama baze podataka

```
CREATE PROCEDURE imeProcedure (eventualni argumenti)
   DEFINE ime LIKE student.imeStud;
   DEFINE ocjena LIKE ispit.ocjena;
   ...
```

- kad god je moguće, tipove varijabli treba definirati implicitno
  - u slučaju promjene tipa podatka nekog atributa u relaciji, sve što je potrebno obaviti jest ponovo prevesti procedure

Naredba LET

- koristi se za pridruživanje vrijednosti varijablama

```
CREATE PROCEDURE ...
  DEFINE r, povrsina DECIMAL(10,5);
  DEFINE brojIspita SMALLINT;
  DEFINE sumaOcjena INTEGER;
  DEFINE prosjek DECIMAL(3,2);
  DEFINE brojZnam SMALLINT;
  LET r = 10;
  LET povrsina = 3.14159 * r * r;
  LET brojIspita = (SELECT COUNT(*) FROM ispit);
  LET sumaOcjena = (SELECT SUM(ocjena) FROM ispit);
  LET prosjek = sumaOcjena/brojIspita;
  LET brojZnam = brojZnamenki('123abc');
  ...
```

- rezultat obavljanja SELECT naredbe koja vraća jednu jednostavni vrijednost (skalar) može se koristiti na svim mjestima na kojima se koriste izrazi. SELECT naredba mora biti unutar okruglih zagrada

Naredbe IF, WHILE, FOR

- naredba za jednostranu, dvostranu ili višestranu selekciju

```
IF uvjet THEN
  naredbe
ELIF uvjet THEN
  naredbe
ELIF uvjet THEN
  naredbe ...
ELSE
  naredbe
END IF;
```

- naredba za realizaciju petlje s ispitivanjem uvjeta na početku

```
WHILE uvjet
  naredbe
  ...
  EXIT WHILE;
  CONTINUE WHILE;
END WHILE;
```

kao break u jeziku C

kao continue u jeziku C

- naredba za realizaciju petlje s unaprijed utvrđenim brojem ponavljanja

```
FOR i = m TO n STEP k
  naredbe
  ...
  EXIT FOR;
  CONTINUE FOR;
END FOR;
```

kao break u jeziku C

kao continue u jeziku C

SQL naredbe u pohranjenim procedurama

- U pohranjenim procedurama mogu se koristiti (gotovo) sve do sada prikazane SQL naredbe (izuzetak je npr. DROP DATABASE)

```
...
DELETE FROM stud
  WHERE prezStud LIKE 'Z%';
UPDATE stud SET pbrMjestoStan = 10000
  WHERE pbrMjestoStan = 41000;
INSERT INTO mjesto VALUES (31000, 'Osijek');
...
```

- Rezultat SELECT naredbe može se pohraniti u varijable, npr.

```
DEFINE v_imeStud LIKE student.imeStud;
DEFINE v_prezStud LIKE student.prezStud;
...
SELECT imeStud, prezStud
  INTO v_imeStud, v_prezStud
  FROM student
  WHERE mbrStud = 12345;
```

- broj i tipovi varijabli moraju odgovarati broju i tipovima izraza iz liste za selekciju
- SELECT naredba smije vratiti samo jednu n-torku

Uporaba varijabli u SQL naredbama

- varijable se slobodno mogu koristiti na svim mjestima na kojim se u SQL naredbama koriste izrazi, npr.
  - u izrazima u SELECT listi, u WHERE dijelu SQL naredbe
  - u VALUES listi INSERT naredbe
  - u izrazima u SET dijelu UPDATE naredbe, ...

```
CREATE PROCEDURE ...
  DEFINE iznos, koef DECIMAL (3,2);
  DEFINE datum DATE;
  DEFINE s INTEGER;
  DEFINE n CHAR(20);
  LET koef = (SELECT MAX(koef) FROM nastavnik);
  LET s = 100; LET n = 'Primorsko-goranska';
  LET datum = TODAY - 365*20;
  SELECT AVG(ocjena) * koef INTO iznos FROM ispit
    WHERE datIspit = datum;
  UPDATE stud SET datRodStud = datum
    WHERE datRodStud <> datum;
  INSERT INTO zupanija VALUES(s, n);
  ...
```


Argumenti pohranjene procedure

```
CREATE PROCEDURE imeProcedure (imeArg tip, imeArg tip, ...)
```

- tipovi podataka ulaznih argumenata procedure mogu, kao i varijable, biti definirani eksplicitno ...

```
CREATE FUNCTION povrsina (sirina INTEGER, visina INTEGER)
  ...
```

- ili implicitno ...

```
CREATE PROCEDURE postaviAdresu (p_mbrStud LIKE stud.mbrStud
                              , p_adresa LIKE stud.adresa)
  ...
```

- jednako kao u drugim programskim jezicima, argumenti se u tijelu procedure/funkcije mogu koristiti na jednak način kao i varijable

Rezultati funkcije

- tipovi rezultata koje funkcija vraća moraju se deklarirati

```
CREATE FUNCTION imeFunkcije (imeArg tip, imeArg tip, ...)
  RETURNING INTEGER AS ime, CHAR(20) AS ime, DATE AS ime
  DEFINE ...
  ...
END FUNCTION;
```

- tipovi rezultata mogu se deklarirati jedino eksplicitno (nije moguće koristiti oblik "LIKE atribut" kao pri definiciji argumenata ili varijabli)
- "ime" rezultata se navodi opcionalno: korisno je deklarirati ime rezultata jer se npr. pri pozivu funkcije iz interaktivnog alata rezultat prikazuje zajedno s deklariranim imenom

Povrat rezultata funkcije u pozivajući program

- koristi se naredba RETURN slična naredbi RETURN u ostalim programskim jezicima. RETURN naredba se u tijelu procedure može pojaviti više puta. Naredbom je u pozivajući program moguće vratiti jednu ili više vrijednosti

```
CREATE FUNCTION opsegPovrsina(radijus DECIMAL(10,5))
       RETURNING DECIMAL(10,5) AS opseg
                 , DECIMAL(10,5) AS povrsina
  DEFINE o, p DECIMAL(10,5);
  LET o = 2 * radijus * 3.14159;
  LET p = radijus * radijus * 3.14159;
  RETURN o, p;
END FUNCTION;
```

```
EXECUTE FUNCTION opsegPovrsina(4.5);
```

opseg	povrsina
28.27431	63.61720

Načini poziva procedure (funkcije)

- Iz interaktivnih alata, npr. Server Studio

```
EXECUTE PROCEDURE prebaci (1001, 1002, 60.30);
```

  - procedura ne vraća rezultat (eventualno signalizira pogrešku)

```
EXECUTE FUNCTION opsegPovrsina(4.5);
```

  - funkcija vraća rezultat (eventualno signalizira pogrešku)

opseg	povrsina
28.27431	63.61720

Načini poziva procedure (funkcije)

- Iz pohranjene procedure ili funkcije

```
CREATE PROCEDURE x (...)
  DEFINE brojZnam ...
  DEFINE opseg, povrsina ...
  ...
  -- procedure ne vraćaju rezultat
  EXECUTE PROCEDURE prebaci (1001, 1002, 60.30);
  ...
  -- funkcije koje vraćaju jednu vrijednost
  LET brojZnam = brojZnamenki('abc123');
  CALL brojZnamenki('abc123') RETURNING brojZnam;
  ...
  -- funkcije koje vraćaju više vrijednosti
  CALL opsegPovrsina(4.5) RETURNING opseg, povrsina;
```


Načini poziva funkcije

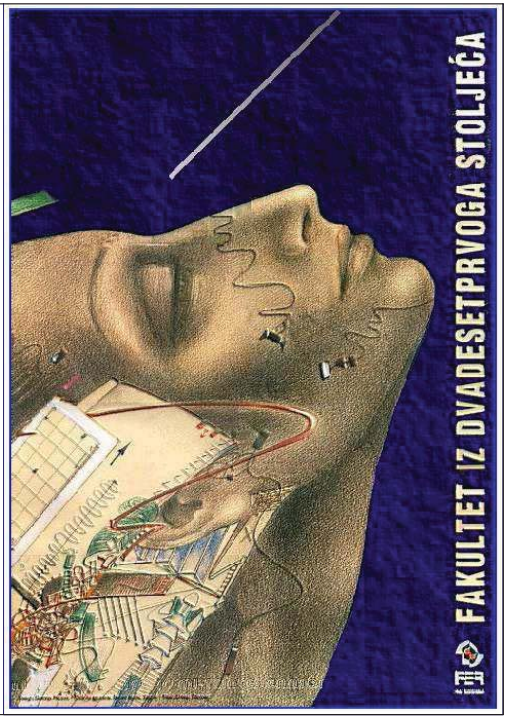
- Korištenje funkcija u SQL naredbama
  - funkcije koje vraćaju točno jednu vrijednost mogu se u SQL naredbama koristiti na svim mjestima na kojima se mogu koristiti ugrađene SQL funkcije

```
SELECT *, brojZnamenki(adresa) AS brojZnam
  FROM osoba
  WHERE brojZnamenki(adresa) > 0;

DELETE FROM osoba
  WHERE brojZnamenki(jmbg) <> 13;
```

Baze podataka

Predavanja
lipanj 2009.

17. Sustav za upravljanje bazama podataka



Sustav za upravljanje bazama podataka

- *Database Management System*
  - skriva od korisnika detalje fizičke pohrane podataka
  - omogućuje definiciju i rukovanje s podacima
  - obavlja optimiranje upita
  - obavlja funkciju zaštite podataka
    - integritet podataka
    - pristup podacima - autorizacija, sigurnost
    - osigurava potporu za upravljanje transakcijama
      - obnova u slučaju pogreške ili uništenja baze podataka
      - kontrola paralelnog pristupa

1. TRANSAKCIJA

- jedinica rada nad bazom podataka
- sastoji se od niza logički povezanih izmjena
- početak transakcije - **BEGIN WORK**
- završetak transakcije:
  - **COMMIT WORK** - uspješan završetak - potvrđivanje transakcije
  - **ROLLBACK WORK** - neuspješan završetak - poništavanje transakcije - poništavanje svih izmjena koje je transakcija obavila

Primjer transakcije

```
CREATE PROCEDURE prijenos (s_racuna INTEGER
                         , na_racun INTEGER
                         , iznos DECIMAL (8,2))
  DEFINE pom_saldo DECIMAL (8,2);
  BEGIN WORK;
     UPDATE racun SET saldo = saldo - iznos
        WHERE br_racun = s_racuna;
     UPDATE racun SET saldo = saldo + iznos
        WHERE br_racun = na_racun;
     SELECT saldo INTO pom_saldo FROM racun
        WHERE br_racun = s_racuna;
     IF pom_saldo < 0 THEN
        ROLLBACK WORK;
     ELSE
        COMMIT WORK;
     END IF
END PROCEDURE
```

Implicitne granice transakcija

- Ukoliko granice transakcije nisu eksplicitno definirane naredbama BEGIN/COMMIT/ROLLBACK, tada se granice transakcije određuju implicitno:
  - početkom transakcije smatra se početak programa
    - uspješan završetak programa - potvrda transakcije
    - neuspješan završetak programa - poništavanje transakcije

  ili

  - svaka SQL naredba se smatra transakcijom za sebe
    - naročito važno: UPDATE, DELETE, INSERT u slučajevima kada djeluju nad skupom n-torki

Svojstva transakcije

- **ACID**
  - *Atomicity* - nedjeljivost transakcije (atomarnost) - transakcija se mora obaviti u cijelosti ili se uopće ne smije obaviti
  - *Consistency* - konzistentnost - transakcijom baza podataka prelazi iz jednog konzistentnog stanja u drugo konzistentno stanje
  - *Isolation* - izolacija - kada se paralelno obavljaju dvije ili više transakcija, njihov učinak mora biti jednak kao da su se obavljale jedna iza druge
  - *Durability* - izdržljivost - ukoliko je transakcija obavila svoj posao, njezini efekti ne smiju biti izgubljeni ako se dogodi kvar sustava, čak i u situaciji kada se kvar desi neposredno nakon završetka transakcije

Nedjeljivost transakcije

```
CREATE PROCEDURE prijenos (s_racuna INTEGER, na_racun INTEGER
                         , iznos DECIMAL (8,2))
   DEFINE pom_saldo DECIMAL (8,2);
   BEGIN WORK;
     UPDATE racun SET saldo = saldo - iznos
        WHERE br_racun = s_racuna;
     UPDATE racun SET saldo = saldo + iznos
        WHERE br_racun = na_racun;
     SELECT saldo INTO pom_saldo FROM racun
        WHERE br_racun = s_racuna;
     ...
```

Kvar sustava

Kvar se dogodio za vrijeme obavljanja druge UPDATE naredbe
- sustav mora osigurati poništavanje efekata prve UPDATE naredbe!

- Sa stanovišta krajnjeg korisnika transakcija je nedjeljiva
  - nije bitno što se moraju obaviti dvije ili više zasebnih operacija nad bazom podataka
- Korisnik mora biti siguran da je zadatak **obavljen potpuno i samo jednom** (ili ništa nije obavljeno)

Izdržljivost transakcije

```
   ...
BEGIN WORK;
  UPDATE racun SET saldo = saldo - iznos
     WHERE br_racun = s_racuna;
  UPDATE racun SET saldo = saldo + iznos
     WHERE br_racun = na_racun;
  SELECT saldo INTO pom_saldo FROM racun
     WHERE br_racun = s_racuna;
  IF pom_saldo < 0 THEN
     ROLLBACK WORK;
  ELSE
     COMMIT WORK;
  END IF
```

Kvar sustava

Kvar se dogodio nakon potvrđivanja transakcije
- efekti transakcije ne smiju biti izgubljeni

- Bez obzira u kojem se trenutku nakon potvrđivanja transakcije dogodio kvar, sustav mora osigurati da su njezini efekti trajno pohranjeni

2. OBNOVA BAZE PODATAKA (*Database Recovery*)

- dovesti bazu podataka u najnovije stanje za koje se pouzdano zna da je bilo ispravno

- Velike baze podataka – dijeljene, višekorisničke – nužno moraju posjedovati mehanizme obnove
- Male, jednokorisničke baze podataka obično imaju malu ili uopće nemaju potporu obnovi – obnova se prepušta korisnikovoj odgovornosti – podrazumijeva se da korisnik periodički stvara "*backup*" kopiju pomoću koje u slučaju potrebe obnavlja bazu podataka

Uzroci pogrešaka

- pogreške opreme
- pogreške operacijskog sustava
- pogreške sustava za upravljanje bazama podataka
- pogreške operatera
- kolebanje izvora energije
- požar, sabotaža, ...

Općenito pravilo koje omogućuje obnovu

- Redundancija - svaki se podatak mora moći rekonstruirati iz nekih drugih informacija redundantno pohranjenih negdje drugdje u sustavu (na traci, na drugom disku, na zrcalnom disku, ...)

Općeniti opis postupka koji omogućuje obnovu

❶ Periodičko kopiranje sadržaja baze podataka na arhivski medij (traka)

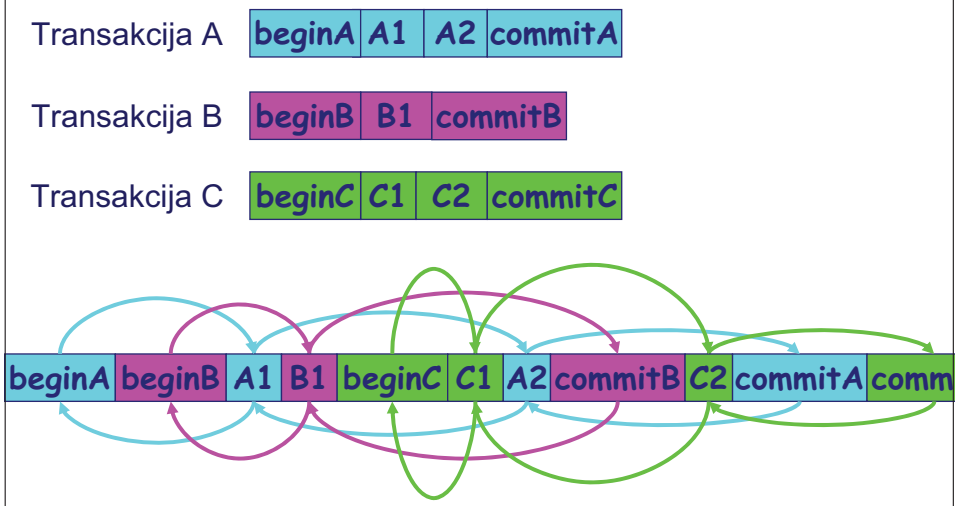
(1 × dnevno, 1 × tjedno - ovisno o učestalosti promjena)

❷ Svaka izmjena u bazi podataka evidentira se u logičkom dnevniku izmjena (*logical log*, *journal*)

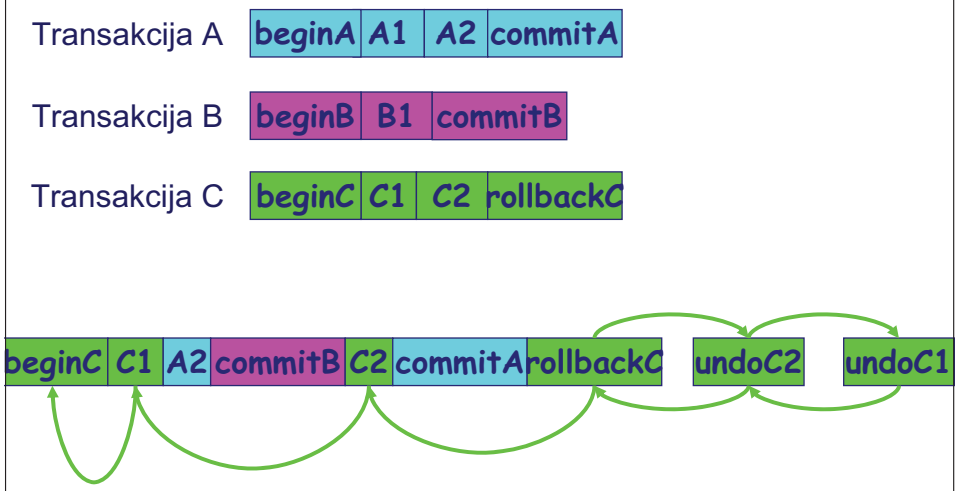
- stara vrijednost zapisa, nova vrijednost zapisa
- korisnik, vrijeme, …
- izmjena se prvo zapisuje u dnevnik, a tek se onda provodi!

- dnevnici izmjena omogućuju
  - poništavanje transakcija (važno radi svojstva nedjeljivosti)
  - ponovno obavljanje transakcija (važno radi svojstva izdržljivosti)

Dnevnik izmjena



Poništavanje transakcije pomoću dnevnika izmjena



Tipovi pogrešaka

1. Pogreške transakcija (*transaction failure)* - pogreške koje otkriva sama aplikacija ili pogreške koje su posljedica neplaniranog prekida transakcije
2. Pogreška računalskog sustava (*system failure*) - baza podataka nije fizički uništena
3. Kvar medija za pohranu (*media failure*) - baza podataka je fizički uništena

Slučaj 1 - pomoću dnevnika izmjena poništavaju se efekti transakcije, kao da transakcija nikada nije započela s radom

Slučaj 2 - transakcije koje su se obavljale u trenutku prekida se nakon ponovnog pokretanja poništavaju

Slučaj 3 - baza podataka se obnavlja pomoću arhivske kopije i pripadnog dnevnika izmjena

Pogreške transakcija koje otkriva aplikacija

- Slučajevi u kojima aplikacija predviđa obavljanje naredbe **ROLLBACK WORK**

```
...
 IF pom_saldo < 0 THEN
    ROLLBACK WORK;
 ELSE
    COMMIT WORK;
 END IF
...
```

Pogreške transakcija koje ne otkriva aplikacija

- ako se dogodi pogreška za koju program nema pretpostavljenu reakciju, program završava na neplanirani način, SUBP automatski obavlja ROLLBACK WORK
- Primjer: pokušaj unosa zapisa čiji ključ već postoji u bazi:

```
CREATE TABLE osoba (
   mbr          INTEGER,
   prezime      CHAR(20),
   PRIMARY KEY (mbr));
```

```
početak programa
BEGIN WORK;
INSERT INTO osoba VALUES (1,'Djetlić', 'Pero');
INSERT INTO osoba VALUES (2,'Marić', 'Maro');
INSERT INTO osoba VALUES (1,'Katić', 'Kata');
INSERT INTO osoba VALUES (4,'Matić', 'Mato');
COMMIT WORK;
završetak programa
```

Pogreška!

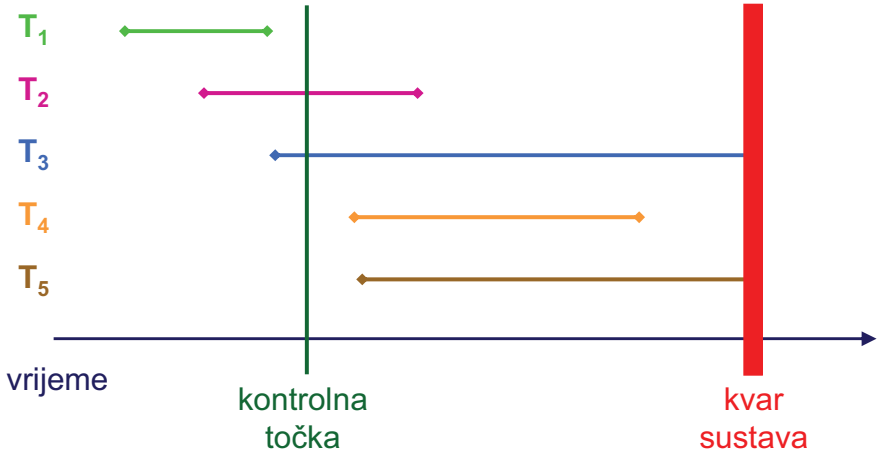
Pogreške računalskog sustava

- **baza nije uništena**
  - **sve transakcije koje su se odvijale u trenutku kvara moraju biti poništene** jer nisu kompletne!
  - pretraživanjem dnevnika od početka identificiraju se transakcije za koje postoji **BEGIN** i ne postoji **COMMIT ili ROLLBACK**
    - takav postupak bi predugo trajao
    - u određenim intervalima (obično svakih 5 minuta) određuje se **kontrolna točka** *(checkpoint)*

Zapis kontrolne točke sadrži:

- listu svih aktivnih transakcija
- za svaku transakciju - adresu najnovijeg zapisa u datoteci dnevnika

Primjer:



Transakcije T_3 i T_5 treba poništiti

Transakcije T_2 i T_4 treba ponovo obaviti

Proces obnove

- Stvara se:
  - lista za poništavanje - na početku sadrži sve transakcije koje su bile aktivne u kontrolnoj točci
  - lista za ponovo obavljanje - na početku je prazna
- Pretražuje se dnevnik od kontrolne točke
  - transakcija za koju se pronađe **BEGIN** dodaje se u listu za poništavanje
  - transakcija za koju se pronađe **COMMIT** prebacuje se iz liste za poništavanje u listu za ponovo obavljanje
- Ponovo se obavljaju transakcije iz liste za ponovo obavljanje
- Poništavaju se transakcije iz liste za poništavanje

SUBP ne može prihvatiti niti jedan zahtjev dok se ne završi proces obnove!

Kvar medija za pohranu

- **baza je fizički uništena - npr. zbog kvara diska**
- obnova sadržaja baze pomoću najnovije arhivske kopije
- pomoću najnovijeg dnevnika obavljaju se transakcije koje su bile provedene od trenutka arhiviranja
  - ako je najnovija arhivska kopija "pokvarena"
    - uzima se predzadnja arhivska kopija
    - dnevnik izmjena od predzadnje arhive do zadnje arhive
    - dnevnik izmjena nastalih nakon zadnje arhive

PREPORUKE:

- čuvati najmanje tri posljednje arhive i pripadne dnevnike
- dnevnik se ne nalazi na istom disku na kojem je baza podataka

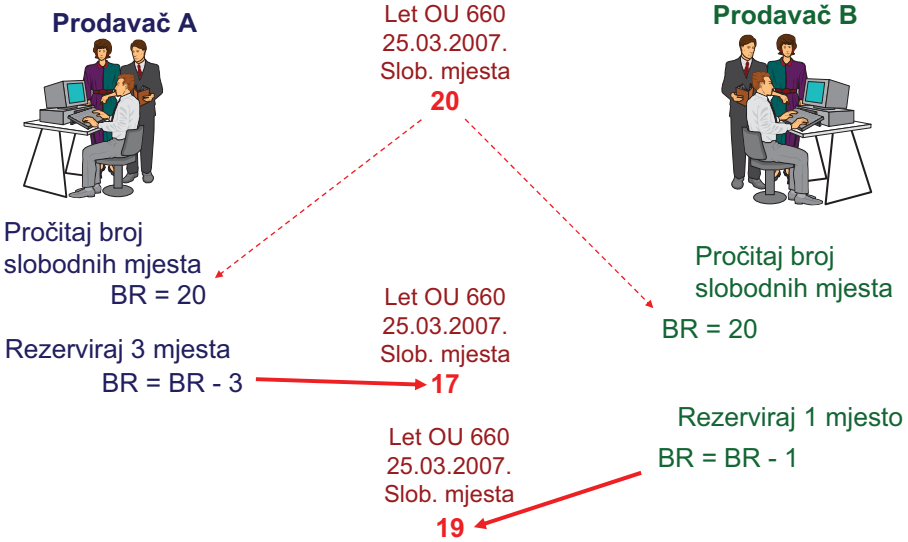
3. KONTROLA PARALELNOG PRISTUPA

- u višekorisničkom radu više programa može istovremeno pristupiti istim podacima
- Rezultat transakcije ne smije ovisiti o tome da li se istodobno odvijaju i neke druge transakcije!!!

- SUBP mora spriječiti sljedeće:
  - dva (ili više) programa "istovremeno" mijenjaju isti podatak – problem: izgubljene izmjene *(lost update)* - posljednji pobjeđuje
  - neki programi pregledavaju podatak dok ga neki drugi program mijenja – problemi: prljavo čitanje *(dirty read)*, neponovljivo čitanje *(nonrepeatable read)*, sablasti *(phantoms)*

Primjer: Izgubljena izmjena (*Lost update*)

Rezervacija zrakoplovnih karata



Prljavo čitanje (*Dirty read*)

osoba

mbr	prez	ime
1111	Novak	Ivan
2222	Kolar	Iva

Transakcija A

```
.
INSERT INTO osoba
 VALUES (3333,
         'Jurić',
         'Ana');
.
.
.
.
.
.
ROLLBACK WORK;
```

Transakcija B

```
.
.
SELECT * FROM osoba;
    1111 Novak  Ivan
    2222 Kolar  Iva
    3333 Jurić  Ana
.
```

n-torka koja nikad nije stvarno postojala u bazi podataka

Neponovljivo čitanje i sablasne n-torke

- Ista transakcija obavljanjem istog upita mora dobiti uvijek isti rezultat (osim ako sama nije promijenila podatke čije čitanje ponavlja)

Transakcija A

```
SELECT saldo FROM racun
  WHERE brRacun = 2;
            .
-- A ne mijenja saldo za
-- racun s brojem 2
            .
            .
SELECT saldo FROM racun
  WHERE brRacun = 2;
```

→ Rezultat: 400.00

→ Rezultat mora (opet) biti: 400.00

Primjer: Neponovljivo čitanje (*Nonrepeatable read*)

Transakcija A

```
SELECT saldo FROM racun
  WHERE brRacun = 2;
            .
-- A ne mijenja saldo za
-- racun s brojem 2
            .
            .
SELECT saldo FROM racun
  WHERE brRacun = 2;
```

→ Rezultat: 400.00

Transakcija B

```
UPDATE racun
   SET saldo = saldo * 1.1
   WHERE brRacun = 2
```

→ Rezultat: 440.00

- Ista transakcija obavljanjem istog upita dobije drugačiji rezultat

Primjer: Sablasne n-torke (*Phantom rows*)

Transakcija A

```
SELECT COUNT(*)
   FROM racun
   WHERE saldo > 100
            .
-- A ne mijenja racun
            .
SELECT COUNT(*)
   FROM racun
   WHERE saldo > 100
```

→ Rezultat: 2

Transakcija B

```
INSERT INTO racun
   VALUES (3, 400.00)
```

→ Rezultat: 3 !!!

- Ista transakcija obavljanjem istog upita dobije drugačiji rezultat - zbog toga što je u međuvremenu transakcijom B unesena n-torka koja zadovoljava kriterij upita

3.1. Zaključavanje (*Locking*)

- transakcija može zaključati podatak (podatke)
  - sprečava druge transakcije da pristupe podatku dok ga ona ne otključa
- podaci koji su se mijenjali tijekom transakcije **ostaju zaključani do kraja transakcije**
- dio SUBP (*locking manager*) zaključava zapise i prosuđuje u slučajevima kad postoji više zahtjeva za zaključavanjem istog podatka

Zaključavanje

Rezervacija zrakoplovnih karata

Prodavač A

Let OU 660
25.03.2007.
Slob. mjesta
20

Prodavač B

Zaključaj n-torku: OK
Pročitaj broj slob.mjesta
BR = 20
Rezerviraj 3 mjesta
BR = BR - 3
Otključaj n-torku

Let OU 660
25.03.2007.
Slob. mjesta
17

Zaključaj n-torku: čekaj

čekaj → OK
Pročitaj broj slob.mjesta
BR = 17
Rezerviraj 1 mjesto
BR = BR - 1
Otključaj n-torku

Let OU 660
25.03.2007.
Slob. mjesta
16

3.2. Potpuni zastoj (*Deadlock*)

Transakcija 1	Transakcija 2
Zaključaj A	Zaključaj B
Zaključaj B	Zaključaj A

---------- POTPUNI ZASTOJ ----------

- izbjegavanje potpunog zastoja:
  - transakcija zatraži sva zaključavanja odjednom (npr. na početku) - **zaključa sve ili ništa!**
  - zahtijeva se da transakcije zaključavaju podatke u nekom **određenom poretku**
- u slučaju da se dogodi potpuni zastoj:
  - barem jedna transakcija se mora prekinuti - poništavaju se njezini efekti

3.3. Vrste zaključavanja

- ključ za pisanje/izmjenu
  - **WRITE LOCK, EXCLUSIVE LOCK**
- ključ za čitanje
  - **READ LOCK, SHARED LOCK**

3.4. Granulacija zaključavanja

- Veličina objekta koji se zaključava
  - baza podataka
  - tablica/relacija
  - memorijska stranica
  - n-torka
- Zaključavanje većih objekata
  - manji broj ključeva ⇒ manji utrošak proc. vremena i memorije
  - manja dostupnost podataka (često se zaključa više nego što je potrebno)
- Zaključavanje manjih objekata
  - veći broj ključeva ⇒ veći utrošak proc. vremena i memorije
  - veća dostupnost podataka (zaključavaju se samo objekti koje je zaista potrebno zaključati)

4. Zaključak

- Zaštita integriteta i sigurnost baze podataka temelji se na pravilima pohranjenim u rječniku podataka
- Pravila pohranjena u rječniku podataka
  - nezaobilazna su za sve korisnike
  - ne opterećuju aplikacijske programe
- Obnova baze podataka bez gubitka informacija moguća je jedino ako se:
  - redovito izrađuju arhivske kopije
  - vodi briga o dnevnicima izmjena