

Projektovanje baza podataka

Odgovori na pitanja (sređeni i dopunjeni)
profesor: Saša Malkov
2018/2019

Zorana Gajić, 400/2016

1. Navesti najvažnije modele podataka kroz istoriju računarstva do danas.

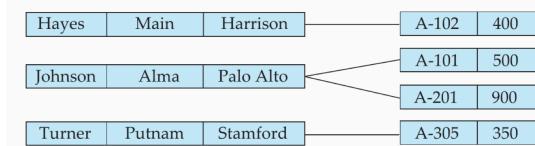
- Mrežni model
- Hijerarhijski model
- Relacioni model
- Model entiteta i odnosa
- Prošireni relacioni model
- Objektni model
- Objektno relacioni model

2. Objasniti osnovne koncepte mrežnog modela podataka

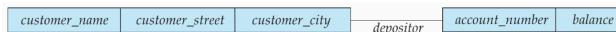
Mrežni model podataka je nalik na dijagramsko povezivanje podataka.

- Struktura podataka - nalik na slogove u programskim jezicima. Slog sadrži podatke jedne instance entiteta, sastoji se od polja.
- Strukture se povezuju «vezama» (link) - nalik na pokazivače.

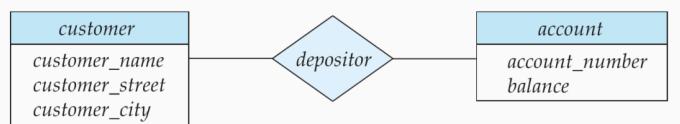
Пример података:



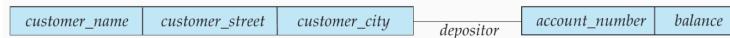
Пример модела:



Повезујемо клијента и банковни рачун (нотација ЕР, касније детаљније):



Однос више-више у мрежном моделу:



Однос један-више у мрежном моделу:



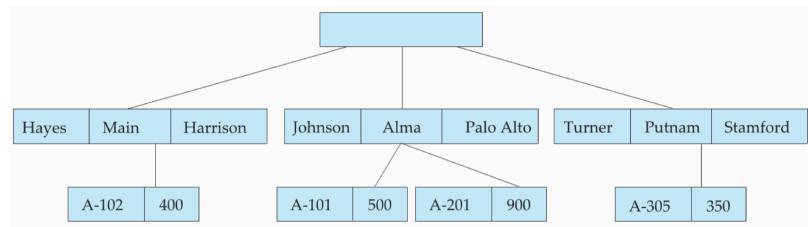
Однос један-један у мрежном моделу:



3. Objasniti osnovne koncepte hijerarhijskog modela podataka

Skup «slogova» povezanih «vezama» tako da grade «hijerarhiju».

- Slično mrežnom modelu, ali se zahteva hijerarhija. Kod mrežnog sloja je hijerarhija bila implicirana smerom veza, a ovde je eksplisitna i ima smer jedan-više.
- Nije dopušteno višestruko vezivanje čvorova - do svakog čvora vodi tačno jedan put od korena.
- Skup slogova u kolekciji započinje «praznim» (dummy) čvorom.



4. Objasniti ukratko osnovne nivoje apstrakcije kod savremenih baza podataka

Kod savremenih baza podataka razlikujemo 4 nivoja apstrakcije:

- Spoljašnja shema - što korisnik vidi
- Konceptualna (logička) shema - sve što čini logički model podataka
- Fizička shema - fizička organizacija podataka u softverskom sistemu
- Fizički uređaj - fizička organizacija podataka na fizičkim uređajima.

5. Objasniti uglove posmatranja arhitekture baze podataka

Referentni model i arhitektura se mogu posmatrati iz 3 ugla:

- iz ugla komponenti
 - Komponente sistema se definišu zajedno sa njihovim međusobnim odnosima. Sistem za upravljanje bazom podataka (SUBP) se sastoji od skupa komponenti, koje obavljaju određene funkcije.
 - Putem definisanih interakcija komponenti ostvaruje se puna funkcionalnost sistema.
 - Posmatranje komponenti je neophodno pri implementaciji.
 - Ali nije dovoljno posmatrati samo komponente, da bi se odredila funkcionalnost sistema kao celine!
- iz ugla funkcija
 - Prepoznaju se različite klase korisnika i funkcije koje sistem za njih obavlja.
 - Uobičajeno se prave hijerarhije korisnika.
 - Prednost pristupa je u jasnoći predstavljanja funkcija i ciljeva sistema.
 - Slabost je u nedovoljnem uvidu u način ostvarivanja funkcija i ciljeva.
- iz ugla podataka
 - Prepoznaju se različite vrste podataka.
 - Arhitektura se određuje tako da definiše funkcionalne jedinice koje koriste podatke na različite načine.
 - Često najpoželjniji.
 - Prednost je u isticanju centralne pozicije podataka u sistemu.
 - Slabost je u nemogućnosti da se arhitektura u potpunosti odredi ako nisu opisane i funkcionalne celine.

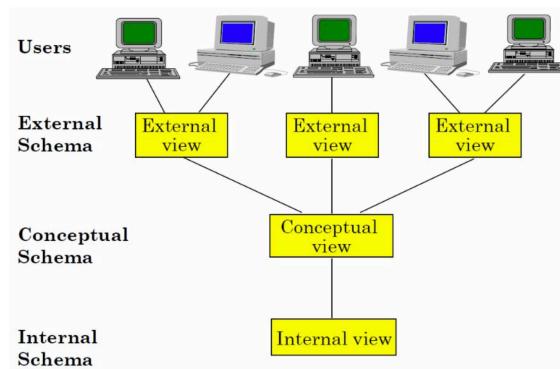
Svi pristupi se moraju kombinovati kako bi se dobio model arhitekture koji u svakoj posmatranoj tački daje dovoljno informacija o odgovarajućim aspektima.

6. Objasniti standardizovanu arhitekturu ANSI/SPARC

ANSI/SPARC (American National Standards Institute / Standard Planning and Requirements Committee) je jedan od najvažnijih standarda baza podataka koji je nastao 1977.godine.

Prepoznaju se 3 nivoa (pogleda) podataka:

- Spoljašnji nivo - kako korisnici (uključujući programere) vide podatke. Može biti više domena, gde svaki obuhvata samo podatke značajne jednoj klasi korisnika.
- Konceptualni nivo - kako podaci čine celinu iz ugla poslovnog okruženja, apstraktna definicija baze podataka.
- Interni nivo - kako su podaci implementirani na računaru. Obuhvata elemente fizičke implementacije.



7. Kakav je odnos relacionih baza podataka i standardizovane arhitekture ANSI/SPARC

Na primeru relacionih baza podataka nivoi se mogu (okvirno) predstaviti na sledeći način:

- Spoljašnji nivo čine pogledi koji pružaju denormalizovanu sliku dela domena.
- Konceptualni nivo čini shema baze podataka - obuhvata opise atributa, ključeva i ograničenja, uključujući i strane ključeve.
- Interni nivo čine fizički aspekti - indeksi, prostori za tabele, razne optimizacije.

8. Objasniti primer arhitekture klijent-server

Osnovna ideja je razdvojiti funkcionalnosti servera i klijenta. Server pruža usluge, a klijent koristi te usluge.

Klijent je zadužen za upravljanje podacima:

- obrada upita
- optimizacija
- izvođenje transakcija.

Server je zadužen za upravljanje podatkovim sistemom:

- ostvarivanje komunikacije između aplikacije i servera
- upravljanje podacima koji su keširani na strani klijenta - podaci, katanci, provera konzistentnosti transakcija.

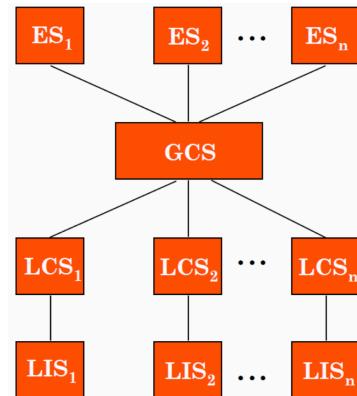
9. Objasniti koncept distribuiranih arhitektura na primeru arhitekture ravnopravnih čvorova

Distribuirane arhitekture imaju više servera, koji imaju različite ili deljene uloge.

Primer: arhitektura ravnopravnih čvorova, federativne baze podataka.

Arhitektura ravnopravnih čvorova:

- Svaki od čvorova može imati sopstvenu fizičku organizaciju podataka, koja se naziva lokalna interna shema (LIS - local internal schema)
- Poslovno viđenje tih podataka na konceptualnom nivou je opisano globalnom konceptualnom shemom (GCS - global conceptual schema)
- Zbog fragmentacije i replikacije podataka, na svakom čvoru je potrebno da postoji logički opis podataka, koji se naziva lokalna konceptualna shema (LCS - local conceptual schema)



Globalna konceptualna shema je zapravo unija svih lokalnih konceptualnih shema.

- Korisnici na spoljašnjem nivou imaju odgovarajuće spoljašnje sheme (ES - external schema)

10. Objasniti osnovne koncepte relacionog modela podataka

Objedinjeno modeliranje:

- I entiteti i odnosi se modeliraju relacijom
- Relacija se sastoji od atributa koji imaju imena i domene
- Skup imena i domena atributa jedne relacije predstavlja shemu relacije
- Torka je skup imenovanih vrednosti
- Torka koja ima isti broj vrednosti, njihove nazive i domene kao atributi jedne relacije predstavlja instancu domena (sheme) relacije
- Vrednost (sadržaj) relacije je skup instanci njenog domena

Integritet se obezbeđuje ključevima.

Relacioni model čine:

- Strukturni deo - način modeliranja podataka
- Manipulativni deo - način rukovanja modeliranim podacima (U relacionom modelu se ovaj deo odnosi se na jezike za postavljanje upita, naprimjer neke formalne, kao što su relacioni račun i relaciona algebra)
- Integritetni deo - način obezbeđivanja valjanosti podataka, odnosno kako da se postaramo da podaci i njihovi odnosi uvek budu ispravni.

Relacioni model je u **potpunosti formalno matematički zasnovan**.

Objašnjenje matematičke formulacije:

Neka nam je dat skup studenata u oznaci E i neki student $e \in E$. To je ono što zovemo entitetima stvarnog sveta, to je ono što predstavlja stvarni domen naše baze podataka.

Mi želimo da napravimo model tog stvarnog sveta i postavlja se pitanje kako?

Modeliranje stvarnog sveta u relacionom modelu počinje od funkcija, odnosno od preslikavanja objekata stvarnog sveta odgovarajućim opisnim funkcijama. Naprimer, jedna opisna funkcija može biti $ime(e_1) = Petar$, $ime(e_2) = Marko$ i tako dalje. Ime je znači funkcija.

Zato možemo reći da neka funkcija $f_1 : E \rightarrow D_1$ slika stvarni skup entiteta u domen funkcije f_1 .

Ova funkcija mora biti uvek definisana, odnosno da bude definisana za sve elemente skupa E .

Imamo funkciju f_1 , možemo da imamo i f_2 naprimjer *prezime*, f_3 i tako dalje. Što znači da imamo skup funkcija. Svaka od tih funkcija ima svoj domen. Da bismo dobro opisali jedan skup entiteta potrebno nam je nekoliko takvih funkcija, zato ćemo napraviti jednu složenu funkciju

$F(e) := (f_1(e), f_2(e), \dots, f_k(e)) = r$. Ovo je opis jednog svojstva, modeliramo jedno svojstvo *ime*, *visinu*, Domen složene funkcije je $F : E \rightarrow D_1 \times D_2 \times \dots \times D_k$.

Za svaki entitet e , dobićemo neku n-torku.

$F(E) := \{F(e) | e \in E\}$.

Šta znači da ako neko $r \in F(E)$? To znači da postoji $e \in E : F(e) = r$.

Naša funkcija je dovoljno dobra i dovoljno dobro opisuje podatke iz skupa entiteta kada postoji jedinstveno $e \in E$, odnosno $F(e_1) = F(e_2) \Rightarrow e_1 = e_2$, odnosno, funkcija F je 1 – 1.

Znači funkcija je dovoljno dobra ako možemo sve studente predstaviti sa različitim r .

U osnovi relacionog modela je definisanje dovoljno dobre funkcije kojom preslikavamo jedan skup entiteta u odgovarajući skup kojim modelira taj skup entiteta.

Zašto se ovo naziva relacijom?

Ako je relacija $\varphi(x, y)$ neka relacija za koju važi da je $x > y$, njen model je skup svih parova (x, y) za koje važi da je $x > y$!

Ako je ova funkcija F injektivna onda možemo reći da je r torka, skup svih torki $F(e)$ predstavlja model jedne relacije, odnosno φ , a stvarna relacija je relacija koja kaže da postoji jedinstven student koji ima ime ovo, prezime ovo, indeks taj i taj.

Terminološki ćemo obično izjednačavati pojам model i relacija, ali je dobro znati razliku.

Najgrublje receno, relacijom smatramo sliku našeg skupa entiteta funkcijom 1-1 u skup torki.

Relacija predstavlja odnos, a model je skup!

11. Šta je strukturni deo relacionog modela? Objasniti ukratko.

Strukturni deo relacionog modela - način modeliranja podataka.

Osnovna ideja je da se i entiteti i odnosi predstavljaju (modeliraju) relacijama.

12. Šta je manipulativni deo relacionog modela? Objasniti ukratko.

Pitanje 19.

13. Šta je integritetni deo relacionog modela? Objasniti ukratko.

Pitanje 22.

14. Navesti primer modeliranja skupa iz posmatranog domena odgovarajućom relacijom.

Neka su nam date 4 kutije: crvena, plava, bela i zelena. U tim kutijama se nalaze lopte, kocke i pločice.

Binarna relacija $kutijaSadrži(K, P)$ je zadovoljena ako kutija K sadrži predmet P .

Njen domen je $Dom(kutijaSadrži) = \{crvena, plava, bela, zelena\} \times \{lopte, kocke, plocice\}$.

Model te relacije je skup, koji predstavlja podskup dekartovog proizvoda kutija i predmeta i neka naredni model opisuje šta se nalazi u kojoj kutiji:

$$kutijaSadrži = \{(plava, lopte), (plava, kocke), (zelena, plocice), (crvena, kocke)\}$$

Iskaz $kutijaSadrži(plava, kocke)$ je tačan.

Iskaz $kutijaSadrži(zelena, kocke)$ nije tačan.

Iskaz $kutijaSadrži(žuta, kocke)$ nije definisan, jer argument nije u domenu relacije.

Relacije sa konačnim brojem elemenata mogu da se predstave pomoću tabele.

kutijaSadrži	
BOJA	SADRŽAJ
plava	lopte
plava	kocke
zelena	pločice
crvena	kocke

15. Šta su entiteti? Kako se formalno definišu atributi i relacije?

Entitetima nazivamo neke objekte «stvarnog» sveta koje modeliramo i opisujemo nekim skupom podataka.

Kažemo da se skup entiteta karakteriše konačnim skupom atributa A_1, A_2, \dots, A_n u oznaci $E(A_1, A_2, \dots, A_n)$ akko:

- Svaki atribut A_i predstavlja funkciju koja slika entitete u odgovarajući domen atributa D_i

$$A_i : E \mapsto D_i$$

- Svaki atribut A_i ima jedinstven naziv t_i

Za svaki entitet $e \in E$, vrednost funkcije $A_i(e) \in D_i$ predstavlja vrednost atributa A_i .

Primer:

Neka je E skup radnika koji se karakteriše atributima: ime, prezime, osnovna_plata

Skup svih atributa određuje funkciju:

IME	PREZIME	OSNOVA_PLATE
Dragana	Pantić	45000
Marko	Marković	40000
Dragana	Pantić	45000
Jelena	Popović	43000
Dragiša	Đukić	47000

$$f(x) = (ime(x), prezime(x), osnovna_plata(x))$$

Skup svih atributa određuje funkciju:

$$f(x) = (A_1(x), \dots, A_n(x))$$

Slika skupa entiteta funkcijom f je skup $R = f(E)$.

Prisetimo se, skup atributa dobro karakteriše skup entiteta ako funkcija f predstavlja injektivno preslikavanje.

Ako je f injektivna funkcija, tada kažemo da je slika $R = f(E)$

relacija R sa atributima A_1, A_2, \dots, A_n , **domenom relacije** $\text{Dom}(R) = D_1 \times \dots \times D_n$ i **nazivima atributa** $\text{Kol}(R) = (t_1, \dots, t_n)$ i tada skup entiteta E sa atributima A_1, A_2, \dots, A_n modeliramo relacijom R .

$A_i(e)$ zapisujemo i kao $e.t_i$.

Relaciju $R = f(E)$ često predstavljamo i nazivamo tabelom:

- Kolone tabele odgovaraju atributima A_1, A_2, \dots, A_n
- Nazivi kolona odgovaraju atributima t_1, \dots, t_n
- Vrste tabele odgovaraju torkama relacije, tj entitetima.

16. Šta je relaciona baza podataka? Šta je relaciona shema?

Ako imamo jednu relaciju koja opisuje neku vrstu entiteta, npr studente i želimo da opišemo još i profesore, učionice i drugo, znači moramo uvesti skup relacija i na taj način smo uveli bazu podataka.

Odnosno, reaciona baza podataka je skup relacija.

Opis relacije čine domen relacije i nazivi atributa.

Reaciona shema je skup opisa relacija koje čine bazu podataka.

Čitavu bazu podataka opisujemo jednom reacionom shemom.

Primer:

RADNIK	
RADNIK_ID	N(4)
ORG_JED_ID	N(3)
IME	C(30)
PREZIME	C(50)
POL	C(1)
DAT_ZAPOSLENJA	D
OSNOVA_PLATE	N(10,2)

REŠENJE	
RAD_ID	N(4)
PROJ_ID	N(3)
DAT_POČETKA	D
DAT_PRESTANKA	D
OPIS	C(60)

ORG_JEDINICA	
ORG_JED_ID	N(3)
NAD_ORG_JED_ID	N(3)
NAZIV	C(60)

PROJEKAT	
PROJEKAT_ID	N(3)
PROJ_BONUS	N(10,2)
NAZIV	C(60)

KAT_STAŽA	
OD_GODINE	N(3)
DO_GODINE	N(3)
STAŽ_BONUS	N(10,2)
NAZIV	C(60)

Shema

$N(a)$ - skup svih celih brojeva sa najviše a dekadnih cifara

$N(a, b)$ - skup svih brojeva koji u dekadnom zapisu sa leve strane decimalne zapete imaju najviše a , a sa desne strane najviše b cifara

$C(a)$ - skup svih niski dužine do a znakova.

D - skup svih datuma

$$\text{Kol}(\text{ORG_JEDINICA}) = (\text{'org_jed_id'}, \text{'nad_org_jed_id'}, \text{'naziv'})$$

$$\text{Dom}(\text{ORG_JEDINICA}) = N(3) \times N(3) \times C(60)$$

I slično za ostale relacije.

Primer sadržaja

ORG_JEDINICA		
ORG_JED_ID	NAD_ORG_JED_ID	NAZIV
40	-	Softver
30	40	Projektovanje
11	10	Dokumentacija
10	40	Planiranje
50	40	Analiza
60	40	Dizajn i modeliranje
70	40	Kodiranje

17. Kako se modeliraju entiteti posmatranog domena u relacionom modelu?

(Iz pitanja 15)

Skup svih atributa određuje funkciju:

$$f(x) = (A_1(e), \dots, A_n(e))$$

Slika skupa entiteta funkcijom f je skup $R = f(E)$.

Prisetimo se, skup atributa dobro karakteriše skup entiteta ako funkcija f predstavlja injektivno preslikavanje.

Ako je f injektivna funkcija, tada kažemo da je slika $R = f(E)$

relacija R sa atributima A_1, A_2, \dots, A_n , **domenom relacije** $\text{Dom}(R) = D_1 \times \dots \times D_n$ i **nazivima atributa** $\text{Kol}(R) = (t_1, \dots, t_n)$ i tada skup entiteta E sa atributima A_1, A_2, \dots, A_n modeliramo relacijom R .

18. Kako se modeliraju odnosi u posmatranom domenu u relacionom modelu?

Odnosi se modeliraju na isti način kao i entiteti - relacijama.

Osnovna ideja:

- Ako imamo dva entiteta $e \in E$ i $f \in F$ u nekom odnosu, onda to možemo opisati novom relacijom $\varphi(e, f)$ čiji je domen $\text{Dom}(\varphi)$.
- Ako su entiteti $e \in E$ i $f \in F$ modelirani kao neke torke, odnosno $e = (x_1, \dots, x_n)$ i $f = (y_1, \dots, y_m)$ onda njihov odnos može da se modelira kao $\varphi(e, f) = (x_1, \dots, x_n, y_1, \dots, y_m)$.

Ako postoji neki podskup atributa $A_{i_1}, \dots, A_{i_{nk}}$ takav da postoji preslikavanje k koje za svaki entitet $e \in E$ jednoznačno preslikava izabrani podskup atributa $(x_{i_1}, \dots, x_{i_{nk}})$ entiteta e u kompletan model entiteta $e(x_1, \dots, x_n)$ onda u modelu relacije taj skup atributa može da se koristi umesto punog skupa atributa A_1, \dots, A_n .

19. Šta čini manipulativni deo relacionog modela?

Manipulativni deo relacionog modela je način rukovanja modeliranim podacima. Ključno mesto u manipulativnom delu modela je pojam upita (izvođenje zaključaka iz postojećih podataka).

Naprimjer, «Koliko je studenata polozilo PBP?»

Upit je definicija nove relacije na osnovu već poznatih relacija baze podataka.

Najpoznatiji formalni upitni jezici u relacionom modelu su:

- Relaciona algebra
- Relacioni račun

Relaciona algebra i relacioni račun su dve formalizacije koje su međusobno ekvivalentne

Pored upita važno mesto zauzima ažuriranje baze podataka, odnosno promena modela.

Zašto menjamo model? Da bismo održali stanje modela sa stanjem «stvarnog» sveta.

Naprimjer, student je upisao PBP, mi moramo u naš skup dodati odgovarajući red.

Ovde počinju problemi, jer se matematika ne snalazi sa promenom podataka.

Matematička definicija jednog fiksnog stanja je savršena, čim imamo menjanje podataka to je problem.

Ima više pristupa, ali suština je da zahteva proširenja i nove pojmove:

- Relaciona promenljiva (= promenljiva relacije)
- Relacija
 - = vrednost relacije
 - = sadržaj relacione promenljive
- Promenljiva baze podataka
- Baza podataka
 - = vrednost baze podataka
 - = sadržaj promenljive baze podataka

Ažuriranje baze podataka je zamjenjivanje vrednosti promenljive baze podataka novom vrednošću baze podataka.

20. Objasniti ukratko relacioni račun

Primena predikatskog računa na relacije.

Ekvivalentan relacionoj algebri.

Jedna od razlika od relacione algebre je u korišćenje kvantifikatora «postoji» i «za svako».

Primer 1: Naredni upit izdvaja muškarce koji imaju osnovnu platu manju od 32000

$$\{x \mid x \in Radnik \wedge x.pol =' M' \wedge x.osnovna_plata < 32000\}$$

Primer 2: Imena i prezimena zaposlenih u «Planiranju», čiji staž ulazi u kategoriju «Srednja»

$$\{(x.ime, x.prezime) \mid x \in Radnik$$

$$\begin{aligned} &\wedge (\exists y \in Org_Jedinica) \\ &(y.naziv =' Planiranje' \wedge x.org_jed_id = y.org_jed_id) \\ &\wedge (\exists z \in Kat_Staza) \\ &(z.naziv =' Srednja' \wedge z.od_godine \leq staz(x) \leq z.do_godine) \end{aligned}$$

21. Objasniti ukratko relacionu algebru

U relacionom modelu imamo skupove. Rad sa skupovima se obavlja putem skupovne algebre, ali ovo nisu obični skupovi, pa će nam trebati neke nove operacije i to ćemo zvati relacionom algebrrom.

Osnovne dodatne operacije su:

- Projekcija - izdvajanje podskupa atributa $\{(x.ime, x.prezime) \mid x \in Radnik\}$
- Restrikcija - izdvajanje podskupa redova $\{x \mid x \in Radnik \wedge x.org_jed_id = 40\}$
- Proizvod - uparivanje svih torki jedne sa svim torkama druge relacije
 $\{(x, y) \mid x \in Radnik \wedge y \in Org_Jedinica\}.$

Ovaj proizvod nije Dekartov jer za $x = (x_1, \dots, x_n)$ i $y = (y_1, \dots, y_m)$ Dekartov proizvod daje $x \times y = ((x_1, \dots, x_n), (y_1, \dots, y_m))$. A mi želimo $(x_1, \dots, x_n, y_1, \dots, y_m)$.

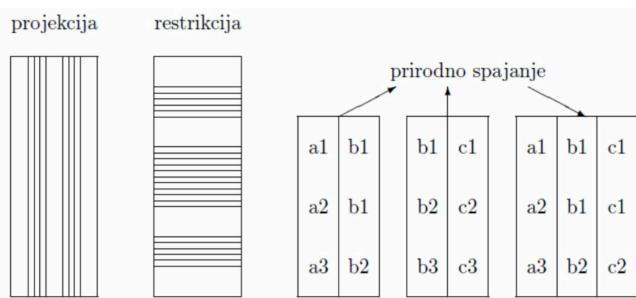
Kombinovanjem osnovnih operacija dobijaju se složeniji upiti, kao:

- Prirodno spajanje - proizvod, pa restrikcija po uslovu jednakosti atributa sa istim imenom, pa projekcija na različite attribute.

$$\{(x, y) | x \in Radnik \wedge y \in Org_Jedinica \wedge x.org_jed_id = y.org_jed_id\}$$

- Slobodno spajanje - spajanje sa restrikcijom po slobodnom uslovu

$$\{(x, y) | x \in Radnik \wedge y \in Kat_Staza \wedge y.od_godine \leq staz(x) \leq y.do_godine\}$$



22. Šta čini integritetni deo relacionog modela?

Kada ažuriramo bazu podataka mi menjamo nešto u bazi. Koje izmene smemo, a koje ne smemo da napravimo, a da baza ostane ispravna?

Naprimjer, da li u bazi podataka smemo da promenimo indeks studenta, a da ne promenimo taj podatak u ostalim tabelama? Odgovor je ne!

Integritetni deo modela se odnosi na to da sprečimo izmene koje dovode bazu u neispravna stanja.

Integritetni deo modela čine koncepti i mehanizmi koji omogućavaju da se automatizuje proveravanja zadovoljenosti određenih uslova.

Stanje baze je konzistentno, tj. ispravno, ako sadržaj baze podataka ispunjava sve uslove integrateta.

Promena sadržaja (vrednosti) baze podataka je dopuštena ako i samo ako prevodi bazu iz jednog konzistentnog stanja u drugo.

Baza podataka (tj. njena vrednost) se opisuje relacijama.

Uslovi integrateta u relacionom modelu se opisuju predikatima (istinitosnim formulama) nad relacijom ili bazom podataka.

Formalnost modela olakšava formulisanje uslova integrateta.

23. Navesti osnovne vrste uslova integrateta u relacionoj bazi podataka

- Opšti uslovi integrateta (implicitni)
 - oni koji moraju da važe za svaku relaciju i svaku bazu podataka
(Naprimjer, svaka tabela mora da ima primarni ključ)
 - podrazumevaju se na nivou modela i implementacije SUBP
 - ne definišu se eksplicitno za svaku relaciju ili bazu podataka

- Specifični uslovi integriteta (eksplicitni)
 - oni koji se odnose na pojedinu relaciju, atribut ili bazu podataka
(Naprimjer, broj cipela studenta ne može biti veći od 45)
 - određuju se pri određivanju strukture baze podataka

Specifični uslovi integriteta se odnose na različite vrste pravila koje možemo proveravati. Pravila proveravamo na različitim nivoima:

- Integritet domena
- Integritet ključa
- Referencijalni integritet
- Integritet stranog ključa
- Opšti uslovi integriteta
- Aktivno održavanje integriteta

24. Objasniti integritet domena u relacionom modelu

Integritet domena - proveravanja domena atributa, da li je vrednost atributa u dopuštenom opsegu vrednosti. Tj. određuje da svaki atribut može da ima samo neku od vrednosti iz unapred izabranog domena.

Domen je neki od tipova podataka, a može da obuhvata i:

- Dužinu podatka
- Opcionu deklaraciju jedinstvenosti
- Opcionu deklaraciju podrazumevane vrednosti

Svaka relacija mora da ima tačno određen domen! (Jedan od opštih uslova integriteta)

25. Objasniti integritet ključa u relacionom modelu

Odnosi se na jednu relaciju.

Određuje se uslovom ključa - uslov ključa određuje minimalan podskup atributa relacije koji predstavlja jedinstveni identifikator torke relacije.

Podskup X atributa A_{i1}, \dots, A_{in_k} relacije R je ključ (ili ključ kandidat) relacije R akko su ispunjeni sledeći uslovi:

- X funkcionalno određuje sve attribute relacije R
- Nijedan pravi podskup od X nema prethodno svojstvo.

Ili prostije rečeno, kandidat ključ je skup atributa koji jednoznačno identificiše redove u tabeli.

Jedna relacija može imati više ključeva.

Jedan od ključeva se proglašava za primarni ključ i upotrebljava za jedinstveno identifikovanje torki relacije.

Podskup X skupa atributa relacije R predstavlja nadključ ako zadovoljava samo prvi od uslova ključa:

- X funkcionalno određuje sve attribute relacije R

Svaka relacija mora da ima primarni ključ - jedan od opštih uslova integriteta, ali u praksi neke tabele neće zahtevati.

Torke relacije se po pravilu referišu pomoću primarnog ključa, zbog efikasnosti.

Kada masovno učitamo podatke u tabelu, postavlja se pitanje kada napraviti primarni ključ u tabeli, kada da napravite indekse. Nikad ne praviti unapred! Zato što je proveravanje uslova pri dodavanju svakog pojedinačnog reda jako sporo, nego uradimo masovni unos podataka i onda pravimo ključeve i tek onda proveravamo zbirno da li su ispunjeni svi uslovi.

26. Objasniti integritet jedinstvenosti u relacionom modelu

Naziva se i «integritet entiteta».

Primarni ključ ne sme da sadrži nedefinisane vrednosti, niti dve torke jedne relacije smeju imati iste vrednosti primarnih ključeva - jedan od opštih uslova integriteta nedoslednih implementacija.

U doslednim relacionim modelima:

- Integritet jedinstvenosti = integritet ključa
- Ne obuhvataju nedefinisane vrednosti
- Jedinstvenost vrednosti primarnog ključa je implicirana jedinstvenošću torki u relaciji.

Pored primarnog ključa, mogu da se eksplicitno deklarišu i jedinstveni ključevi, koji su po svemu isti kao primarni ključevi, ali nije uobičajeno da se koriste pri referisanju.

27. Objasniti referencijalni integritet u relacionom modelu

Referencijalni integritet predstavlja uslove o međusobnim odnosima koje moraju da zadovoljavaju torke dve relacije.

Naprimjer, id_studijskog_programa u tabeli ispit, mora da odgovara id_studijskog_programa nekog studjiskog programa u tabeli studjiski programi.

Pravila referencijalnog integriteta:

- Ne sme se obrisati torka na koju se odnosi neka torka neke relacije u bazi podataka, niti se sme tako izmeniti da referenca postane neispravna.

(Naprimjer, da li smemo da obrišemo neki studijski program, dok imamo studente na njemu? Ne.)

- Ne sme se dodati torka sa neispravnom referencom (takov da ne postoji torka na koju se odnosi) (Ovde ne bismo smeli dodati studenta sa nepostojećim studentskim programom.)

U praksi se ostvarjuje putem integriteta stranog ključa.

Ova pravila se menjaju ukoliko imamo nedefinisane ili nedostajuće vrednosti:

- Važe prva dva pravila od pre!
- Referenca koja sadrži nedefinisane vrednosti je ispravna (i ne referiše ništa) akko je u potpunosti nedefinisana.

28. Objasniti integritet stranog ključa u relacionom modelu

Skup *FK* atributa relacije *R* je njen strani ključ koji se odnosi na baznu relaciju *B* akko važe sledeće:

- Relacija *B* ima primarni ključ *PK*
- Domen ključa *FK* je identičan domenu ključa *PK*
- Svaka vrednost ključa *FK* u torkama relacije *R* je identična kluču *PK* bar jedne torke relacije *B*.

Za relaciju *R* se kaže da je zavisna od bazne relacije *B*.

Bazna relacija *B* se naziva i roditeljskom relacijom.

Ova pravila se menjaju ukoliko imamo nedefinisane ili nedostajuće vrednosti:

- Važe prva dva pravila od pre!
- Za svaku vrednost ključa *FK* u torkama relacije *R* važi da ili sve vrednosti atributa imaju nedefinisanu vrednost ili nijedna vrednost atributa nije nedefinisana
- Svaka vrednost ključa *FK* u torkama relacije *R* je ili u potpunosti nedefinisana ili je identična kluču *PK* bar jedne torke relacije *B*.

29. Objasniti pravila brisanja i ažuriranja kod integriteta stranog ključa u relacionom modelu

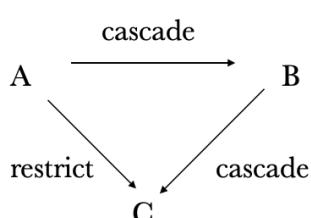
Poštovanje integriteta stranog ključa pri menjanju sadržaja baze podataka se određuju pravilima ažuriranja koja mogu da budu:

- Pravila brisanja (biraj se tačno jedno)
- Pravila ažuriranja (biraj se tačno jedno)

Pravila brisanja: Ako se pokuša brisanje torke bazne relacije *B*, za koju postoji zavisna torka relacije *R* (ona čiji je strani ključ jednak primarnom kluču torke koja se pokušava obrisati) onda:

- Aktivna zabrana brisanja - RESTRICT - zabranjuje se brisanje torke iz relacije *B*
- Pasivna zabrana brisanja - NO ACTION - slično kao aktivna, ali se odlaže provera do samog kraja brisanja, zato što možda postoji linija kaskadnih pravila koja će obrisati zavisnu torku
- Kaskadno brisanje - CASCADE - brišu se sve odgovarajuće zavisne torke *R*
- Postavljanje nedefinisanih vrednosti - SET NULL - u svim zavisnim torkama relacije *R* atributi stranog ključa se postavljaju na nedefinisane vrednosti
- Postavljanje podrazumevanih vrednosti - SET DEFAULT - u svim zavisnim torkama relacije *R* atributi stranog ključa se postavljaju na podrazumevane vrednosti.

Primer: Ako imamo 3 tabele *A*, *B*, *C* i imamo raspoređene strane klučeve kao na slici ispod. Želimo da obrišemo red iz tabele *C*.



Prvo što se proverava je da li postoji red koji se referiše preko pravila RESTRICT u tabeli *A*. Pošto postoji, brisanje nije dozvoljeno. Nezavisno što postoji CASCADE do *B* i *A* preko *B*.

Pravila ažuriranja: Ako se pokuša menjanje primarnog ključa torke relacije B , za koju postoji zavisna torka relacije R (ona čiji je strani ključ jednak primarnom ključu torke koja se pokušava obrisati).

- Aktivna zabrana menjanja - RESTRICT - zabranjuje se menjanje torke relacije B
- Pasivna zabrana menjanja - NO ACTION - slično kao aktivna, ali se odlaže provera do samog kraja menjanja, zato što možda postoji linija kaskadnih pravila koja će promeniti zavisnu torku
- Kaskadno menjanje - CASCADE - na isti način se menjaju atributi stranog ključa svim zavisnim torkama relacije R
- Postavljanje nedefinisanih vrednosti - SET NULL - u svim zavisnim torkama relacije R atributi stranog ključa se postavljaju na nedefinisane vrednosti
- Postavljanje podrazumevanih vrednosti - SET DEFAULT - u svim zavisnim torkama relacije R atributi stranog ključa se postavljaju na podrazumevane vrednosti.

30. Objasniti opšte uslove integriteta relacionog modela

Pored posebnih uslova integriteta, mogu da se odrede i specifični uslovi integriteta:

- Na atributu - Pri definisanju domena atributa mogu da se propisu i dodatni uslovi integriteta atributa.
 - To je uslov na atributu koji mora uvek da važi, obično vrlo lokalnog karaktera (odnosi se na vrednost jednog atributa jedne torke).
 - Uslov može da zavisi samo od vrednosti atributa.
 - Može da se koristi za dodatno sužavanje domena (naprimjer, atribut mora da ima jednu od nekoliko vrednosti).
 - Može da se koristi za proveru ispravnosti složenih tipova podataka (naprimjer, datum se predstavlja tekstrom, ali se onda proverava da li tekstualni zapis predstavlja ispravan zapis datuma).

Primer:

```
create table ... (
    attr1 int check (attr1 in (1,2,3)),
    ...
)
```

- Na torki - Pri definisanju relacije mogu da se propisu i dodatni uslovi integriteta relacije.
 - Lokalnog karaktera, odnosi se na vrednost jedne torke
 - Uslov može da zavisi od vrednosti svih atributa torke
 - Koristi se za proveru ispravnosti složenijih saglasnosti atributa u okviru jedne torke (naprimjer, ako se navodi neki interval, može da se proverava da li je početna vrednost intervala manja od krajnje vrednosti intervala).

Primer:

```
create table ... (
    attr1 ... ,
    attr2 ... ,
    constraint check ( attr1 < attr2 )
    ...
)
```

- Na relaciji - Pri definisanju relacije mogu da se propisu i dodatni uslovi integriteta relacije.
 - Globalnog karatkera, može da se odnosi na sve torke jedne relacije
 - Uslov može da zavisi od vrednosti svih atributa torke
 - Koristi se za proveru ispravnosti složenijih saglasnosti vrednosti u okviru jedne relacije (naprimer, da li je suma po nekom atributu zadovoljava određeni uslov)

Primer:

```
create table T1 (
    attr1 ... ,
    attr2 ... ,
    constraint check (
        select sum(attr1*attr2) from T1 < 1000
    )
    ...
)
```

- Na bazi podataka - Na nivou baze podataka mogu da se propisu i dodatni uslovi integriteta između više relacija, nisu vezani za konkretnu tabelu.
 - Globalnog karaktera, odnosi se na vrednosti torki u različitim relacijama
 - Koristi se za proveru složenijih uslova integriteta

Primer: Da li je broj redova u tabeli T1 manji od broja redova u tabeli T2

```
create assertion asrt_1 check (
    select count(*) from T1 < select count(*) from T2
)
```

31. Objasniti aktivno održavanje integriteta u relacionim bazama podataka

Osnovno sredstvo aktivnog održavanja integriteta su okidači.

Aktivno održavanje integriteta podrazumeva da nakon što se nešto dogodi, nakon što se izmeni neki podatak ili pre nego što izmenimo podataka želimo da proverimo da li su ispunjeni neki uslovi.

- Okidači su izvorno uvedeni za automatizaciju reagovanja na promene podataka u relacijama.
- Izvršavaju se pre ili posle naredbe za dodavanje, menjanje ili brisanje torki.
- Ako na relaciji postoji okidač, koji se izvršava posle naredbe dodavanja, onda će on biti automatski pokrenut posle svake naredbe dodavanja torki toj relaciji.

32. Objasniti ulogu i princip rada okidača na tabelama relacione baze podataka

Okidači se mogu izvršavati pre ili posle naredbe za dodavanje, menjanje ili brisanje torki:

- Na jednoj relaciji može da bude više okidača
- Nekad se mora reagovati pre ili posle, a nekad je svejedno.

Okidači se mogu izvršavati na 2 osnovna načina: za svaki pojedinačno izmenjeni red ili zbirno za celu naredbu koja menja podatke.

Rekli smo da možemo da reagujemo pre ili posle naredbe (dodavanje, menjanje, brisanje torki). Nije svejedno da li sve izvršava pre ili posle u slučaju brisanja i dodavanja! Ako izvršavamo posle brisanja, ne možemo da pristupamo starim vrednostima, jer su obrisane! A ako izvršavamo pre dodavanja, ne možemo da pristupamo novim vrednostima, još nisu ubačene.

Primer: Kad se dodaje novi zaposleni, pre nego što se doda, izvršavamo za svaki red koji se dodaje instrukciju u kojoj za n označavamo novi red koji se dodaje. Postavićemo u tom n platu u zavisnosti od kvalifikacije zaposlenog.

```
create or replace trigger insert_set_salary
    no cascade before insert on employee
        referencing new as n for each row
        begin
            set n.salary = case n.edlevel when 18 then 50000
                                when 16 then 40000
                                else 25000
            end;
        end
```

33. Objasniti ulogu i princip rada okidača na pogledima relacione baze podataka

Šta je pogled u bazi? Kada napravimo pogled, koje izmene smemo da napravimo na pogledu? Da li smemo da radimo insert, update, ... Samo ako je pogled definisan nad jednom tabelom i ima još nekih dodatnih uslova.

Osnovno sredstvo razdvajanja nivoa shema u relacionom modelu su pogledi.

- Na nižem nivou (interna ili konceptualna shema) relacije su normalizovane radi stabilnosti i neredudantnosti
- Na višem nivou (konceptualna ili spoljašnja shema) relacije mogu da budu denormalizovane, tj. spajaju se radi lakšeg postavljanja upita i pristupanja podacima
- Privilegije nad pogledima mogu da se potpuno razlikuju od privilegija na relacijama.

Osnovno ograničenje pogleda: Pogledi nad više relacija ne mogu da se ažuriraju!

Okidači nad pogledima nam omogućavaju da radimo nad njima šta god hoćemo. Jer praktično mi definišemo da umesto naprimer naredbe update na našem pogledu se obavlja neka druga operacija. Ovo je zgodno jer onda pogledi u potpunosti postaju tabele za nekog korisnika, ne zanima ga šta je iza.

Savremeni SUBP nude okidače nad pogledima:

- Izvršavaju se umesto naredbe za dodavanje, menjanje ili brisanje torki iz pogleda
- Omogućavaju preusmeravanje izmena na relacije na kojima počiva pogled, ali i dalje od toga, na sasvim druge relacije.
- Omogućavaju skrivanje veoma složenih operacija kojima se spoljašnja shema razdvaja od konceptualne ili konceptualna od interne.

Primer okidača na pogledu: Ako pogled V1 počiva na relacijama T1 i T2, okidačem se rešava kako se menjaju tabele «kroz pogled».

```
create trigger V1_update instead of update on V1
```

```
referencing new as n old as o  
for each row mode db2sql  
update T1 set (c1, c2) = (n.c1, n.c2)  
where c1 = o.c1 and c2 = o.c2
```

34. Objasniti motivaciju za pravljenje modela entiteta i odnosa

Pun naziv «Entity-Relationship Model», skraćeno ER model.

Model entiteta i odnosa je predložio Peter Čen 1976. godine. U svom radu navodi nedostatke mrežnog, hijerarhijskog i relacionog modela i pokušava da definiše jedan bolji model pronalazeći pravo mesto za njega u procesu projektovanja baza podataka.

«Rad predstavlja model entiteta i odnosa, koji je napredniji od navedena 3 modela. Model entiteta i odnosa usvaja prirodni pristup da se stvarni svet sastoji od entiteta i odnosa».

Kada posmatramo relationalni model, koliko god da je dobro napravljen, kada gledamo relacije, postavlja se pitanje da li iz predstavljenog dijagrama možemo da shvatamo semantiku, da razumemo šta je koja relacija, šta je koja tabela. I vrlo često ne možemo.

Osnovna kritika postojećih modela:

- Ne čuvaju meta informacije o entitetima i odnosima među njima.
- Relacioni model može da izgubi neke važne semantičke informacije o stvarnom svetu (koji je entitet važniji, ko je stariji, ko je mlađi, da li je u pitanju nasleđivanje, agregacija...)
- Rad Čena koristi model entiteta i odnosa kao okvir iz koga mogu da se razviju 3 postojeća modela.

35. Objasniti osnove koncepte i pretpostavke modela entiteta i odnosa

Model entiteta i odnosa ima 2 različita osnovna koncepta: entitete i odnose.

Teži da očuva sve semantičke informacije.

Prepoznajemo 4 nivoa pogleda na podatke:

1. «Informacije koje se tiču entiteta i odnosa, koje postoje u našem umu»

- Konceptualni model, tj. apstraktne semantičke informacije

Imamo onu sliku informacija o entitetima i odnosima, koju mi možemo da razumemo. Kada posmatramo neki stvaran svet mi razumemo naprimer šta je fakultet, šta je ispit, šta je predmet. Jedan krajnje konceptualni prikaz.

2. «Strukturu informacija, tj način organizovanja informacija u kome se entiteti i odnosi predstavljaju podacima»

- Logički model, tj. strukture podataka koje čuvamo u bazi podataka

Na ovom nivou pokušavamo da uđemo u neki tehnički model i da ograničenjima tehničkog modela predstavimo taj isti domen. Gubimo malo apstrakcije. Pokušavamo da vidimo kako da organizujemo informacije. Naprimer, sad znamo da imamo fakultet i imamo studenta. Mi to možemo da organizujemo tako što nam je student jedan skup, fakultet drugi skup, svaki student je vezan za neki fakultet itd. Ovde ćemo reći da postoji odnos «studiranje» koji kaže da taj student studira neki fakultet.

3. «Strukture podataka nezavisne od pristupnog puta, tj. koje ne zahtevaju sheme pretraživanja, sheme indeksiranja i slično»

- Fizički model sa određenim nivoom apstrakcije

Rad sa podacima ćemo imati na apstraktnom nivou. Naprimer u relacionom modelu, kažemo imamo tabele i indekse, ali radimo sa podacima na sql-u, na jednom apstraktnom nivou! Ne silazimo na nivo gde bismo morali da čitamo podatke, vrednosti i sl.

4. «Strukture podataka koje zavise od pristupnog puta»

- Niski fizički modeli, praktično bez apstrakcije

Kada imamo eksplicitno praćenje pristupnog puta, kada tačno pratimo kojim putem čitamo podatke, kao u mrežnom, hijerarhijskom modelima onda kažemo da su to strukture podataka koje zavise od pristupnog puta.

Nivo implementacije.

Kojim nivoima odgovaraju koji modeli?

Mrežni i hijerarhijski modeli odgovaraju najnižim, radimo navigaciju, kuda i kako idemo kroz podatke. Relacioni model se bavi prvenstveno nivoima 2 i 3.

Model entiteta i odnosa se bavi prvenstveno nivoima 1 i 2.

U praksi nemamo nijednu bazu koja je napravljena na osnovu modela entiteta i odnosa.

36. Objasniti kako se ER-model uklapa u nivoe 1 i 2 pogleda na podatke

ER-model nivo 1:

- Entitet je stvar koja može da se jednoznačno identificuje - osoba, preduzeće, događaj.
- Odnos je neko međusobno pridruživanje entiteta - otac-sin, radnik-preduzeće.
- Entiteti se klasifikuju u različite skupove entiteta. Svakom skupu entiteta odgovara predikat koji proverava da li mu neki entitet pripada. Skupovi entiteta ne moraju biti disjunktni. (naprimer, skup studenata I smera, skup studenata na fakultetu....).
- Skup odnosa je matematička relacija između N entiteta koji pripadaju nekim skupovima entiteta, a čiji su elementi odnosi.
- Uloga entiteta u odnosu je funkcija koju on obavlja u odnosu.
- Informacije o entitetima i odnosima se izražavaju skupom parova atribut-vrednost. Vrednosti se klasifikuju u skup vrednosti. Skupovi vrednosti odgovaraju domenu atributa.
- Atribut može da se definiše kao funkcija koja preslikava skup entiteta u skup vrednosti.

ER-model nivo 2:

- Primarni ključ, mora da postoji sredstvo za razlikovanje i jednoznačno referisanje elemenata skupova entiteta i odnosa.
- Relacije entiteta i odnosa. Skupovi entiteta i skupovi odnosa se modeliraju relacijama (tabelama).

37. Objasniti razliku između entiteta i odnosa u ER-modelu

Pitanje 36.

38. Šta su slabi i jaki entiteti?

ER-model razlikuje slabe i jake entitete.

Slabe entitetske relacije - u identifikovanju entiteta učestvuju odnosi sa drugim entitetima. Neki odnosi među entitetima koji su uslovljeni postojanjem jedne strane u tom odnosu.

Naprimjer, neka student studira neki fakultet. Da li student može da studira ako ne postoji fakultet? Ne, ali fakultet može da postoji bez studenta, bez svih ne, ali bez nekog konkretnog može. Ovo je slab entitetski odnos, jer entitet student bez entiteta fakultet ne postoji.

U praksi, u primarnom ključu učestvuje referenca na drugi entitet.

Obične (regularne) entitetske relacije - u identifikovanju entiteta ne učestvuju odnosi sa drugim entitetima (jak entitet)

U praksi, u primarnom ključu ne učestvuje referenca na drugi entitet.

Znači, **jaki entiteti**:

- identificuju se sami za sebe
- uglavnom nezavisni od drugih entiteta u bazi
- njihovo postojanje nije uslovljeno postojanjem drugih entiteta.

Slabi entiteti:

- identificuju se samo kroz odnos sa nekim drugim entitetom
- uglavnom ne postoje samostalno, bez nekih drugih entiteta
- obično predstavljaju sastavni deo ili opis nekog drugog entiteta.

39. Nacrtati primer ER-dijagrama i objasniti njegove osnovne elemente

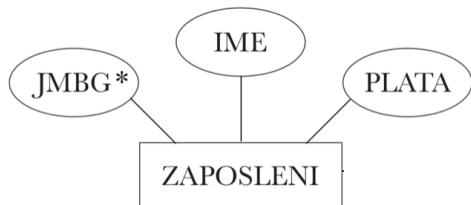
ER-dijagrami su sastavni deo modela, predstavljaju osnovni način zapisivanja semantičkih znanja o informacijama.

Skupovi entiteta se predstavljaju pravougaoncima.

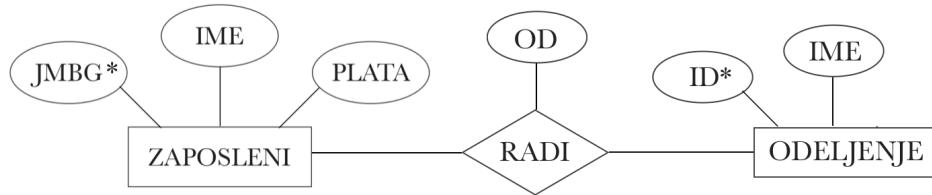
Atributi se predstavljaju elipsama.

Primenjivost atributa na skup entiteta se predstavlja linijom.

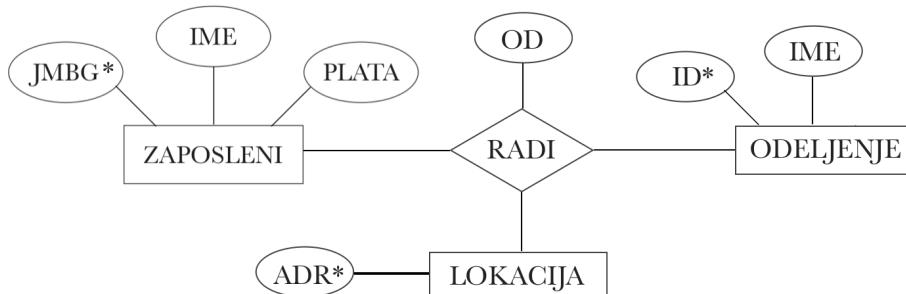
Primarni ključ označavamo zvezdicom.



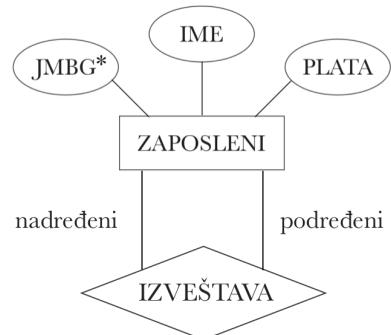
Odnosi se predstavljaju rombovima i mogu imati opisne attribute. (Na vežbama takve odnose još i uokvirujemo!)



Odnosi mogu da uključuju više entiteta.



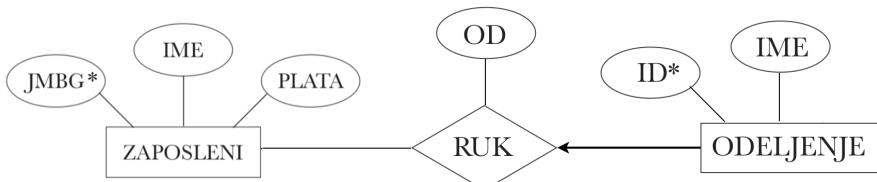
Odnosi mogu da budu i između više instanci istog skupa entiteta. U tom slučaju moraju da se imenuju uloge koje entiteti imaju u odnosima.



40. Šta su i kako se na ER-dijagramu označavaju uslov ključa, uslov učešća i puno učešće?

Uslovi ključa:

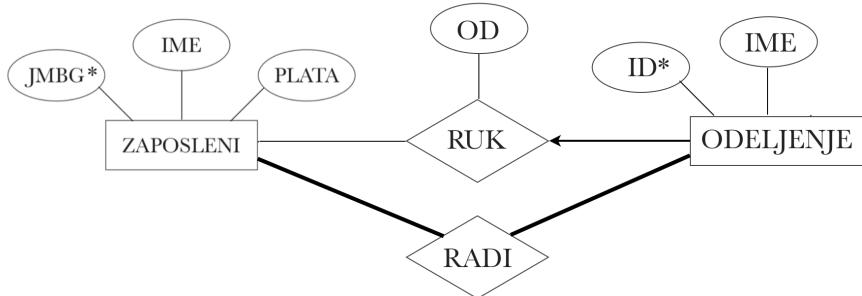
- Ako na osnovu entiteta može da se jednoznačno odredi odnos u kome učestvuje, onda je to ključni entitet odnosa. To se označava strelicom od entiteta prema odnosu!



Odeljenje ima najviše jednog rukovodioca.

Uslovi učešća:

- Ako svaki entitet skupa učestvuje u bar jednom odnosu, onda je to puno učešće u odnosu, a inače parcijalno učešće. Puno učešće u odnosu se označava debljom linijom.



Svaki zaposleni radi u bar jednom odeljenju.

Svako odeljenje ima bar jednog zaposlenog.

Svakao odeljenje ima tačno jednog rukovodioca.

41. Kako se na ER-dijagramima označava kardinalnost i šta tačno označava?

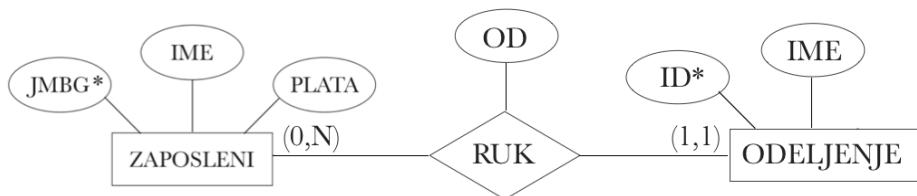
Kardinalnost odnosa opisuje u koliko različitih odnosa tog tipa može da učestvuje svaki od entiteta.

Ima više notacija, navodimo izvornu:

- Označava se brojem na svakoj od linija koje vode od entiteta prema odnosu.
- Broj označava u koliko takvih odnosa učestvuje jedan entitet.
- Umesto broja može stajati opseg vrednosti (A,B) ili $A..B$ gde je $A \leq B, A \geq 0, B \geq 1$.

Primeri ispravnih oznaka kardinalnosti:

- 1
- 0..1
- 2..5
- 1..*
- 0..*



Za jednog zaposlenog važi da ne mora da rukovodi nijednim odeljenjem ili može rukovoditi najviše N odeljenja.

Za svako odeljenje važi da ima tačno jednog rukovodioca.

Alternativna notacija:



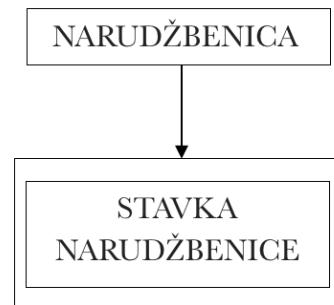
42. Kako se na ER-dijagramima označavaju slabti entiteti?

Elementi slabog skup entiteta se identificuju samo u sklopu odnosa sa nekim drugim entitetom. Taj odnos mora biti «jedan-više» i naziva se identificujući odnos.

Slab entitet mora imati puno učešće u identificujućem odnosu i identificujući odnos i učešće slabog entiteta u njemu se označavaju debljim (ili dvostrukim) linijama.

Na vežbama spajamo strelicom ka slabom entitetu, kod profesora je obrnuto.

Ovde su entiteti spojeni direktno, ali može i preko veze.



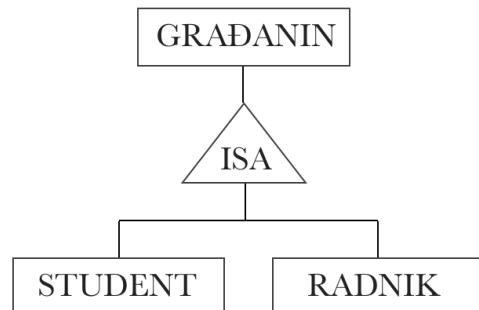
Primer:

Stavka narudžbenice ne može da postoji bez narudžbenice!

43. Kako se na ER-dijagramima označavaju hijerarhije?

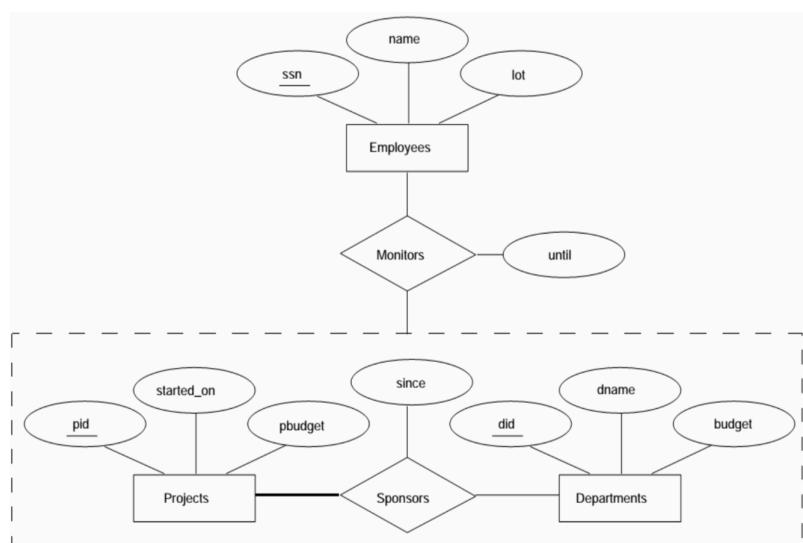
Hijerarhije klase su tzv. «ISA» hijerarhije.

- Uslov preklapanja: da li isti entitet može da pripada dvema potklasama ili ne (povezano sa konceptom višestrukog nasleđivanja)
- Uslov prekrivanja: da li svaki entitet natklase pripada nekoj od potklasa ili ne (povezano sa konceptom apstraktnih klasa)



44. Šta je i kako se na ER-dijagramima označava «agregacija»?

Agregacije su odnosi koji učestvuju u drugim odnosima. Takve odnose uokvirujemo isprekidanim linijom. Nije isto kao ternarne relacije!



45. Objasniti osnovne slabosti ER-modela

Postoje određene nedoslednosti.

Počiva na pretpostavci da postoje 2 različita koncepta entiteta i odnosa:

- Ali ne nudi objektivne kriterijume za njihovo razumevanje
- Način predstavljanja agregacija na dijagramima dodatno potvrđuje da odnosi mogu da imaju neke karakteristike entiteta.

Složeni atributi su dodatni problem, da li složeni atributi imaju osobine entiteta?

Osnovni konceptualni problem ER-modela uviđa i sam autor: «Moguće je da neki ljudi vide nešto (naprimjer brak) kao entitet, dok neki drugi ljudi to vide kao odnos. Mišljenja smo da odluku o tome mora doneti administrator preduzeća. On bi trebao da definiše šta su entiteti a šta odnosi tako da razlika bude odgovarajuća za njegovo okruženje».

Sam autor ovime priznaje da su osnovni koncepti modela samo subjektivno a ne i suštinski različiti.

Bez objektivnih kriterijuma za njihovo razlikovanje, dva osnovna koncepta se stapaju u jedan. Time nastaje suštinska razlika (na nivoima 2 i 3) između ER-modela i relacionog modela.

Svođenje razlika samo na nivo 1, ER-model se u teoriji svodi na alat za projektovanje, umesto da sveobuhvatni model podataka.

Danas se ER najčešće i ne koristi kao model, već kao dijagramska tehnika relacionog modela.

Dijagrami su PRETRPANI.

46. Navesti i ukratko objasniti osnovne korake u projektovanju baza podataka

Koraci:

1. Analiza zahteva
2. Konceptualno projektovanje
3. Logičko projektovanje
4. Prečišćavanje sheme
5. Fizičko projektovanje
6. Projektovanje bezbednosti

47. Objasniti korake Analiza zahteva i Konceputalno projektovanje pri projektovanju baza podataka

Analize zahteva obuhvata:

- Razumevanje podataka - strukture podataka, obima podataka, odnosa među podacima (koliko su složeni)
- Razumevanje aplikacija - potrebe za podacima, učestalost upita i transakcija, performanse (koliko sme trajati upit, koje sve upite imamo)

Kada sve infomacije budemo imali, onda smo analizirali zahtev i dalje pravimo konceptualni model. Težimo da napravimo apstraktnu sliku, da prepoznamo entitete, da prepoznamo odnose i semantiku (zašto je entitet tu, od čega se sastoji, koji su mu atributi i odnosi i da sve to predstavimo na jedan dijagram). Konceptualni model možemo da zamislimo kao ER-dijagram, uz koji idu dodatne informacije, koje određuju svojstva entiteta, procenat veličine i sl.

Konceptualno projektovanje obuhvata:

- Pravljenje modela podataka visokog nivoa
- Opisivanje - struktura podataka, odnosa među strukturama podataka, uslova integriteta
- Često se koristi ER-model ili bar ER-dijagram.

48. Objasnitи korake Logičko projektovanje i Prečišćavanje sheme pri projektovanju baza podataka

Logičko projektovanje je korak u kome težimo da konceptualni model prilagodimo konkretnom sistemu na kojem će biti baza podataka. Ideja je da entitete i odnose iz konceptualnog modela pokušamo da prevedemo sada na jezik logičkog modela, odnosno platforme na kojoj radimo.

Neku ćemo apstraktnost izgubiti i neku ćemo semantiku izgubiti, ali težimo da zadržimo što više toga.

Logičko projektovanje obuhvata:

- Prilagođavamo se nekom konkretnom modelu, odnosno bira se konkretna vrsta, a često i implementacija SUBP.
- Konceptualni model se prilagođava konkretnom implementacionom modelu podataka
- Najčešće koristi relacioni SUBP, pa je ključni posao prevođenje konceptualnog modela na relacioni model - obično ERM na RM, često objektni model na RM, ako je na konceptualnom nivou korišćen RM, onda ovaj korak može da bude trivijalan.
- Obično se radi parcijalno, po delovima, jer uglavnom imamo prevelike baze.

Prečišćavanje sheme obuhvata:

- Analizira se logički model.
- Prepoznaju se potencijalni problemi u implementaciji i rešavaju se (otklanjanje svih sitnih neispravnosti u logičkom modelu)
- U slučaju relacionog modela obično se svodi na normalizaciju.

49. Objasnitи korake Fizičko projektovanje i Projektovanje bezbednosti pri projektovanju baza podataka

Fizičko projektovanje obuhvata:

- Razmatra se očekivano opterećenje: uobičajeno, vršno.
- Dodatno se popravlja model BP tako da zadovolji postavljene kriterijume vezano za performanse.
- Uobičajeni koraci: projektovanje indeksa, denormalizacija, projektovanja distribuiranja...
(dodavanje nekih stvari koje nisu bile bitne na logičkom nivou, kako da organizujemo memoriju...)

Projektovanje bezbednosti obuhvata:

- Prepoznaju se vrste/uloge korisnika - i vrste aplikacija.
- Za svaku ulogu i za svaku vrstu aplikacija definišu se korisnička grupa i odgovarajući minimalan skup privilegija za obavljanje posla.
- Prepoznaju se delovi baze podataka koje su posebno osetljivi i kojima je potrebno dodatno redukovati pristup.
- Definišu se i implementiraju odgovarajući bezbednosni mehanizmi.

50. U kojoj fazi projektovanja baza podataka se najviše koristi ER-model?

ER-model se najviše koristi u koracima:

- 2 - Konceptualno projektovanje jer često za rezultat ima upravo ER-model.
- 3 - Logičko projektovanje. Često najveći posao predstavlja prevođenje konceptualnog projekta sa ER-modela na relacioni model.

Osnovni koraci pri pravljenju ER-modela:

- Određivanje tipova entiteta
- Prepoznavanje odnosa među entitetima
- Profinjavanje definicije odnosa

Uobičajen scenario:

Polazi se od tekstualnog problema, imenice predstavljaju kandidate za entitete (neki od njih će možda biti odnosi!)

Ima ozbiljnih nedoumica:

Da li je nešto bolje modelirati entitetom ili atributom? Da li je nešto bolje modelirati entitetom ili odnosom? Koji su odgovarajući skupovi entiteta i skupovi odnosa? ...

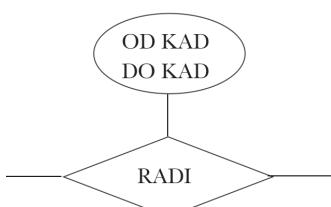
51. Objasniti dilemu entitet ili atribut

Ako imamo složen atribut, možemo da ga modeliramo kao:

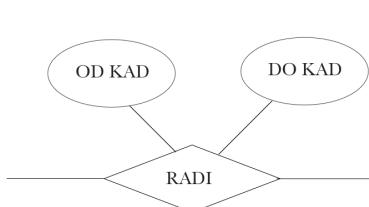
- Jedan složen atribut
- Više prostih atributa
- Jeden entitet

Na osnovu čega donosimo odluku? Čak i za proste attribute se može postaviti pitanje da li su oni možda ipak entiteti.

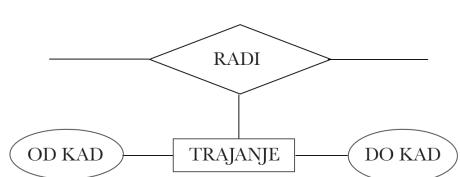
Jedan složen atribut



Dva prosta atributa



Poseban entitet



Argumenti za «običan atribut»:

- Ako jednom entitetu odgovara tačno jedna vrednost «atributa»
- Ako predstavlja jednostavnu skalarnu vrednost koja opisuje entitet
- Ako u više entiteta može da ima iste vrednosti, a da one nisu međusobno uslovljene (tj. menjanje vrednosti za jedan entitet ne povlači njeno menjanje za drugi)
- Ako važi slično i kada je to «atribut» u različitim skupovima entiteta.

Složen atribut može da ima smisla:

- Ako želimo da sačuvamo internu strukturu, ali ona ima značaja samo interno
- Ako u više entiteta može da ima iste vrednosti, a da one nisu međusobno uslovljene (npr ako «adresa» obuhvata grad, ulicu i broj, onda više zaposlenih može imati istu adresu, ali da ne žive zajedno)
- Ako važi slično i kada je to atribut u različitim skupovima entiteta.

Atribut mora da preraste u entitet:

- Ako nekom drugom entitetu odgovara istovremeno više vrednosti tog atributa
- Ako postoji međuzavisnost vrednosti atributa (npr, ako više entiteta ima istu vrednost atributa i on se promeni za jedan entitet, onda mora da se promeni i za ostale)

52. Objasniti dilemu entitet ili odnos

Odnos može da bude «odnos»:

- Ako se svi njegovi atributi odnose striktno na jednu instancu odnosa (tj. nema redundantnosti)

Odnos mora da preraste u entitet:

- Ako se neki od njegovih atributa odnosi istovremeno na više instanci odnosa (redundantnost povlači uočavanje novog entiteta i podizanje složenosti odnosa)

Inače: U velikom broju slučajeva je «jednako ispravno» da nešto bude bilo odnos bilo entitet



53. Objasniti dilemu složeni odnos ili više binarnih odnosa

Većina složenih odnosa može da se «podeli» na više binarnih agregiranih odnosa.

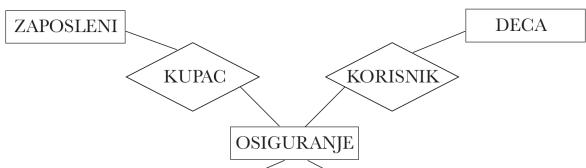
Pitanje je da li ih je bolje podeliti ili ostaviti?

Ako odnos uvek mora da obuhvati 3 entiteta, ako je svugde kardinalnost ne 0 onda ternarni odnos predstavlja bolje rešenje. Sa druge strane, ako nam treba 2 entiteta i kao opcioni može da se doda treći, onda su bolji binarni odnosi.

Ternarni odnos



Binarni odnosi



(Bolje je ovo desno, zato što u levom slučaju imamo redundantnost! Naprimer, student polaže PBP kod profesora SM. Ako imam levi dijagram onda bi to izgledalo ovako:

Student1, PBP, SM.

Student2, PBP, SM.

Dok bi desnom odgovaralo:

Student1 i Student2 odgovaraju paru (PBP, SM).

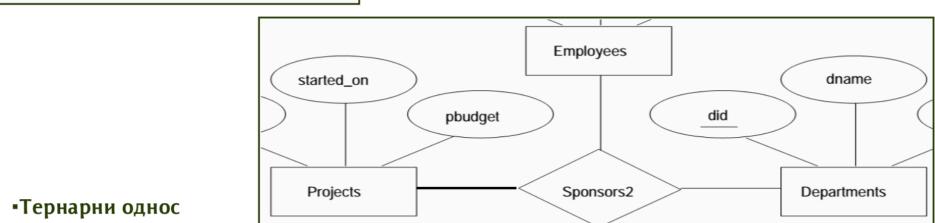
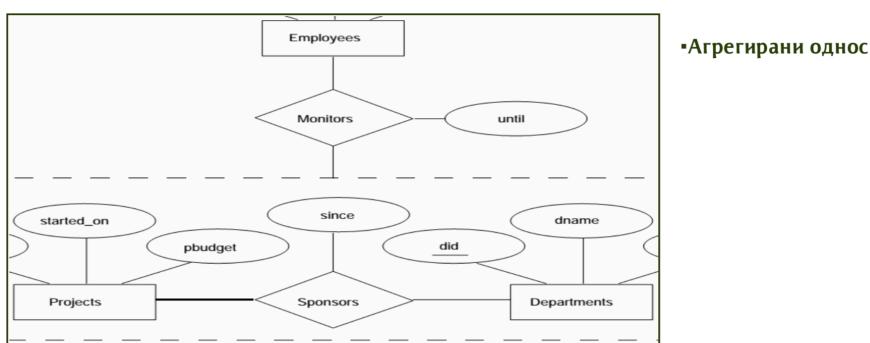
)

Složene odnose ima smisla podeliti na više jednostavnih isključivo ako svaki od dobijenih odnosa ima svoj semantički smisao u posmatranom domenu.

Čak i tada, ako je semantika očiglednija iz složenog odnosa, onda bi taj složeni odnos trebalo da ostane.

54. Objasniti odnos agregacija ili ternarni odnos

Agregirani odnos podrazumeva da se aggregirani odnos može posmatrati kao skup entiteta. Ali to može biti i veštački napravljeno!



Agregirani odnosi su bolje rešenje:

- Ako nekada postoji samo deo odnosa, a nekada ceo
- Ako se deo atributa odnosa ne tiče celine odnosa već samo njegovog dela, koji uključuje manje entiteta.

U ostalim slučajevima je bolje da se koriste složeni odnosi.

55. Kako se entiteti i odnosi ER-modela prevode u relacioni model?

Prevođenje ER-modela na relacioni model je u nekim elementima pravolinjski, može da zavisi i od semantike i od konteksta.

Neka jednostavna pravila:

- Svaki entitet se prevodi u relaciju
- Skoro svaki odnos se prevodi u relaciju
- Atributi koji predstavljaju skupove podataka prevode se u relacije

Svaki entitet se prevodi u relaciju:

- Atributi entiteta se prevode u atribute relacije (složeni atributi se prevode u više prostih atributa)
- Ključni atributi se prevode u primarni ključ

Skoro svaki odnos se prevodi u relaciju:

- Atributi odnosa se prevode u atribute relacije (dodaju se i ključni atributi uključenih entiteta)
- Primarni ključ dobijene relacije čine primarni ključevi relacija koje odgovaraju uključenim entitetima
- Primarnom ključu se dodaju i svi ključni atributi odnosa
- Strani ključevi su primarni ključevi učesnika.

Ako je odnos **1 prema (0,1)**:

- Onda ne mora da se pravi dodatna relacija za odnos
- Atributi odnosa se dodaju relaciji koja odgovara entitetu sa kardinalnošću 1
- AKO ODNOS NIJE 1 prema (0,1) već (0,1) prema (0,1) onda se dobija potencijalno redundantno rešenje

Atributi koji predstavljaju skupove podataka prevode se u entitete, pa u relacije.

56. Kako se hijerarhije ER-modela prevode u relacioni model?

Hijerarhije mogu da se reše na tri osnovna načina:

- Cela hijerarhija u jednu relaciju
- Svaki entitet u posebnu relaciju (tj. kao da se radi o običnom odnosu entiteta)
- Svaki entitet-list u posebnu relaciju, ali tako da uključi sve nasleđene atribute.

Sve u jednu relaciju:

- Jedna relacija koja obuhvata sve atribute koji postoje u hijerarhiji
 - za svaki entitet se koristi samo deo atributa, ostali imaju nedefinisane vrednosti
- Efikasna upotreba
- Neefikasno zauzeće prostora
- Ako je hijerarhija sa preklapanjem onda mora da postoji način da se za svaki konkretni skup entiteta hijerarhije proveri da li mu entitet pripada
 - ili po jedan dodatan binarni atribut za svaki entitet hijerarhije
 - ili se proverava da li su ostali odgovarajući atributi neprazni
- Ako je hijerarhija sa pokrivanjem onda je potreban uslov da svaka torka relacije pripada bar jednom konkretnom skupu entiteta

Svaki entitet posebno:

- Za svaki entitet se pravi po relacija, sa stranim ključem na neposrednu baznu relaciju
 - Svaka relacija obuhvata samo one atribute koji su za nju specifični (i još atribute ključa bazne relacije koji se koriste kao strani ključ)
- Upotreba zahteva česta spajanja i potencijalno neefikasno
 - Prihvatljivo na nivou konceptualnog i logičkog modela
- Ako je hijerarhija sa pokrivanjem mora da se obezbedi dodatno sredstvo za proveru pokrivenosti
 - tj. da svaki red baznog entiteta ima odgovarajući red u nekom listu

Listovi posebno:

- Za svaki entitet se pravi po relacija, sa stranim ključem na neposrednu baznu relaciju
 - Svaka relacija obuhvata atribute koji su za nju definisani (i sve atribute svih baznih klasa, atribute ključa bazne relacije se koriste kao strani ključ)
- Upotreba pojedinačnih entiteta je efikasna
- Pretraživanje baznog skupa entiteta je neefikasno (unija)
- Ako je hijerarhija sa preklapanjem onda svako rešenje može da napravi probleme
 - Delovi entiteta mogu da se redundantno ponavljaju u više listova.

57. Kako se slabi entiteti ER-modela prevode u relacioni model?

Slab entitet i odgovarajući odnos se modeliraju jednom relacijom.

Dodaje se i ključni atribut vezanog jakog entiteta, kao deo primarnog ključa.

58. Šta su dijagrami tabela (relacija) ?

Dijagrami tabela nisu ER dijagrami! Prilagođeni su i logičkom i fizičkom nivou relacionog modela. Semantika odnosa je predstavljena u meri u kojoj to dopušta relacioni model.

59. U čemu se dijagrami tabela suštinski razlikuju od ER dijagrama? Koji se kada koriste?

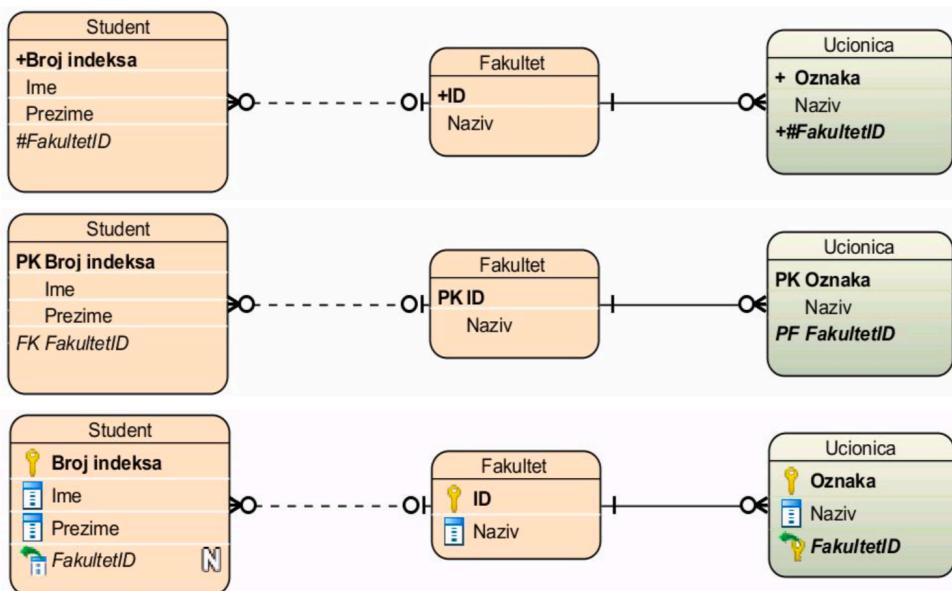
Dijagrami tabela su vrlo bliski fizičkom nivou, dok su ER dijagrami na konceptualnom nivou. Ne govore nam ništa o semantici, za razliku od ER modela.

60. Kako se označavaju ključevi u dijigramima tabela?

Ako kolona pripada primarnom ključu: simbol +, oznaka PK, crtež ključa.

Ako kolona pripada stranom ključu: simbol #, oznaka FK, crtež ključa sa strelicom, obično iskošen naziv.

Dijagram tabela



61. Kako se označavaju različite vrste odnosa i kardinalnosti odnosa u dijigramima tabela?

Odnosi se predstavljaju isprekidanim linijama. Puna linija predstavlja odnos slabog entiteta sa matičnim.

Kardinalnost se predstavlja na kraju linije i govori nam broj entiteta (uz koji oznaka stoji) koji mogu da budu u odnosu sa jednim entitetom koji je na drugom kraju linije.

Nula se predstavlja kružićem, jedinica uspravnom linijom, a linija koja se grana u tri kratke linije prema tabeli je * (ili N sa vezbi).

Sa gornje slike, naprimjer odnos Student - Fakultet kaže: jedan fakultet može imati 0 ili više studenata.

62. Objasniti potencijalne razlike u dijagramima tabela na konceptualnom/ logičkom/fizičkom nivou

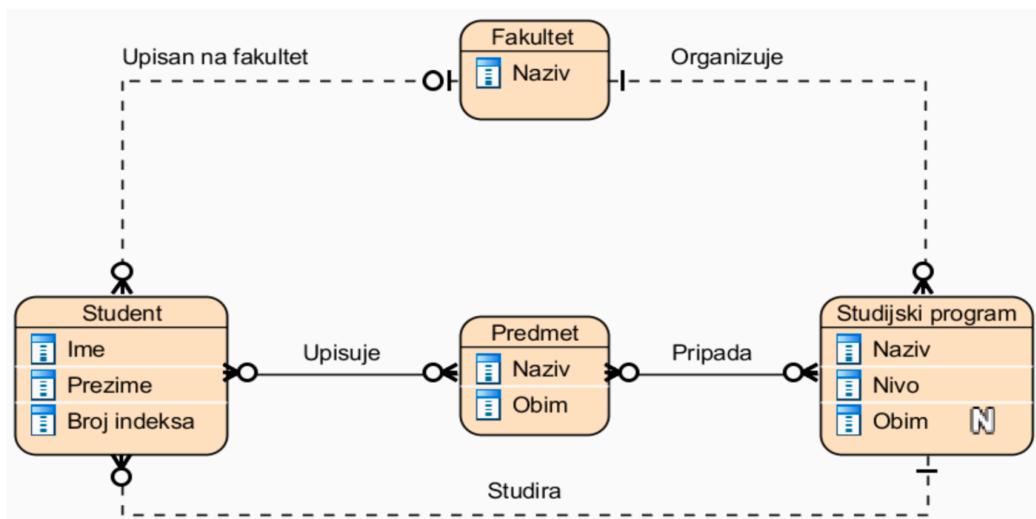
Osnovni oblik dijagrama najviše odgovara logičkom i posebno fizičkom nivou:

- Na konceptualnom nivou se prilagođava
- Na logičkom nivou može da se prilagođava ali ne mora (npr. da izbacimo neke atribute, radi vizuelnog olakšavanja)
- Na fizičkom nivou se koristi u osnovnom obliku.

Na **konceptualnom nivou** dijagram sadrži samo strukturu podataka i odnose.

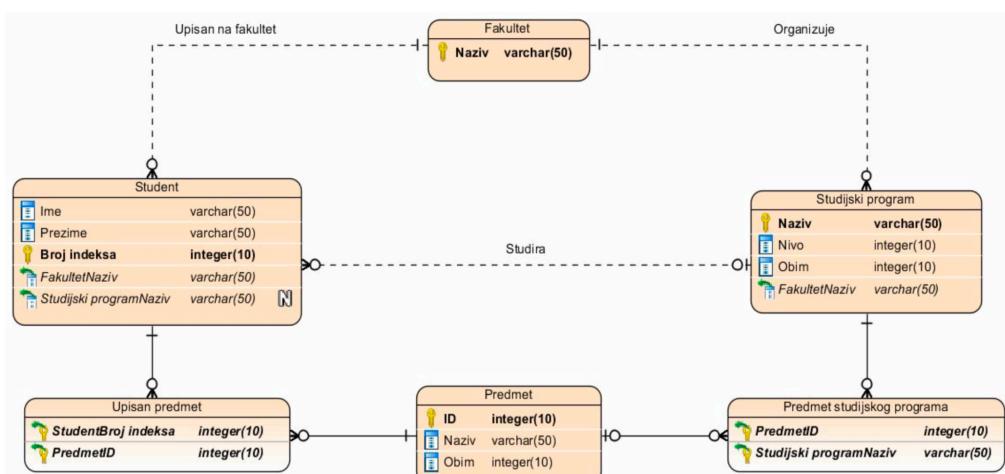
Ne sadrži: dodatne kolone surrogat ključeva, dodatne kolone stranih ključeva, oznake ključeva, tipove kolona.

I na logičkom nivou može da se koristi ovakav dijagram.



Na fizičkom nivou dijagramu se dodaju svi ostali podaci neophodni za preslikavanje u relacioni model:

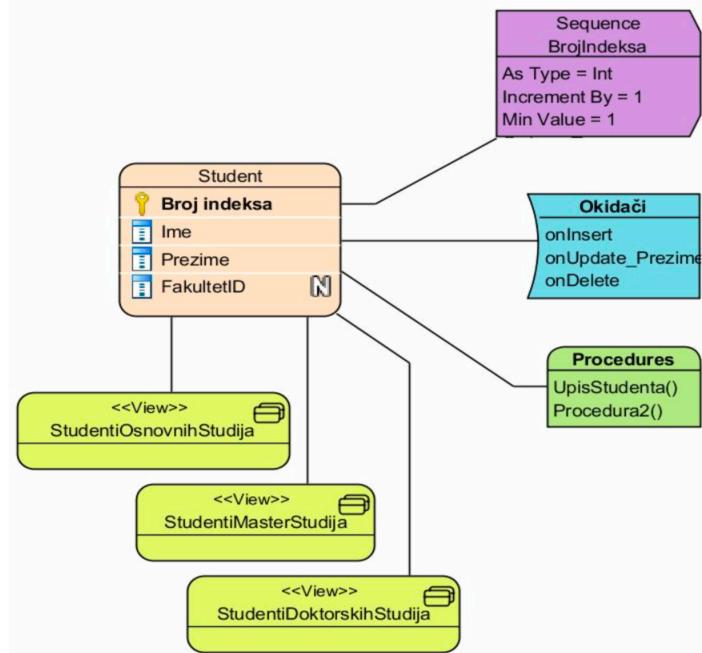
- kolone surrogat ključeva
- kolone vezanih tabela koji čine strane ključeve
- oznake ključeva
i vrsta ključeva
- tipovi kolona
- odnosi više-više
se često
zamenjuju
tabelama koje ih
modeliraju.



63. Kako se u dijagramima tabela označavaju pogledi, okidači i drugi elementi?

Poglede možemo dodati i na logičkom nivou.

Procedure, sekvence mogu biti vezane za više tabele istovremeno. Okidači uglavnom za jednu, pogledi zavisi.



64. Kako se dijagrami klase UML-a koriste u projektovanju baza podataka? Objasniti razlike u odnosu na uobičajene dijagrame klase.

UML dijagram klasa opisuje strukturu podataka klasa.

Dijagrami klasa imaju mogućnost da predstave različite semantike odnosa. Znači možemo da prepoznamo vrstu odnosa: hijerarhije, agregacije, asocijacije...

Ideja je da se isti dijagram iskoristi za modeliranje podataka:

- Bolje predstavlja semantiku odnosa od dijagrama tabela
- Bolje predstavlja semantiku odnosa čak i od pravog ER dijagrama (vizuelno jednostavnije)

Dijagrame klase možemo da koristimo za konceptualno projektovanje otprilike jednako dobro kao i kod modela entiteta i odnosa.

Kada pravimo dijagram klasa, imaćemo jednu klasu koja označava naprimer studenta, u dijagramu tabela možemo da napravimo tri tabele za studenta: student osnovih studija, student master studija i student doktorskih studija. Ove tri tabele imale bi potpuno istu strukturu, zato će u dijagramu klasa to biti jedna tabela.

Zato na dijagramu klasa predstavljamo tipove, a na dijagramu tabela skupove.

Modeli klasa i relacija se konceptualno razlikuju: BITNO!

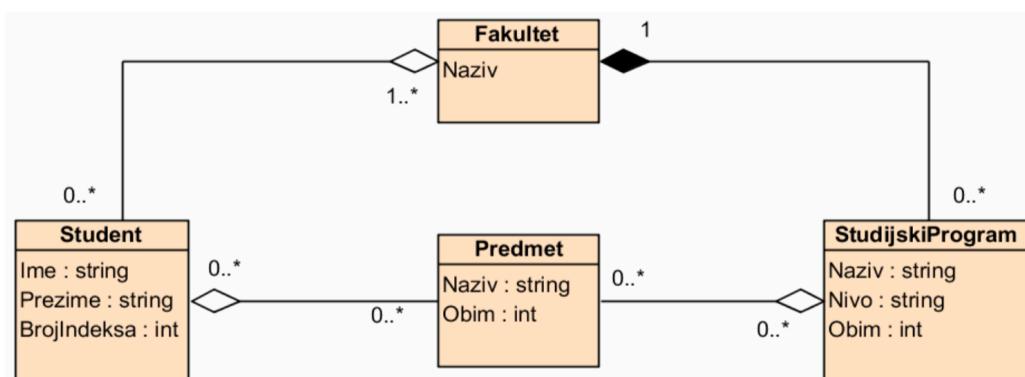
- Relacije (tabele) su skupovi podataka
- Klase su tipovi podataka

Može da se prevaziđe u praksi:

- Ako je potrebno više skupova istog tipa, uvodimo više naslednika klase koja određuje tip
 - nije sasvim semantički ispravno, ali eksplisitno se vidi «sličnost» tipova skupova entiteta
 - bazna klasa, koja služi samo kao tip, može da se označi npr kao apstraktna.
- Jedan pristup je da se svaka «klasa» označi kao «type» ili «persistant»
 - tipovi/klase se označavaju sa «type»
 - skupovi entiteta sa «persistant»
- Ignoriše se i smatra da je klasa skup, a ne tip.

Za razliku od uobičajenog dijagrama klasa:

- U prvom planu su atributi i odnosi
- Ponašanje (metodi) se skoro potpuno zanemaruje
- I enkapsulacija je u drugom planu



65. Objasniti dopune UML dijagrama koje se koriste u specifičnim oblastima primene

UML sadrži standardizovane koncepte koji omogućavaju uvođenje novih načina označavanja

- Stereotipovi - pogledati sledeće pitanje.
- Označene vrednosti - opštiji slučaj stereotipova. Navode se kao lista parova ime-vrednost između vitičastih zagrada {isPersistent = true, minCount = 5}.
- Proširenja - element koji se povezuje sa osnovnim elementom strelicom sa popunjениm trouglom na kraju (strelica usmerena od osnovnog elementa ka proširenju). Može da se koristi za dodavanje osobina ili za opisivanje osnovnog elementa. Obično je opcionalno, ali može da bude i neophodno, što se označava sa {required} iznad strelice.

66. Šta su stereotipovi UML-a i kako se označavaju?

Stereotip predstavlja vrstu šablonu:

- Apstrakciju opštijeg slučaja nečega
- Neki predefinisani skup osobina
- Neko predefinisano ponašanje

Koriste se za navođenje dodatnih deklaracija ili napomena.

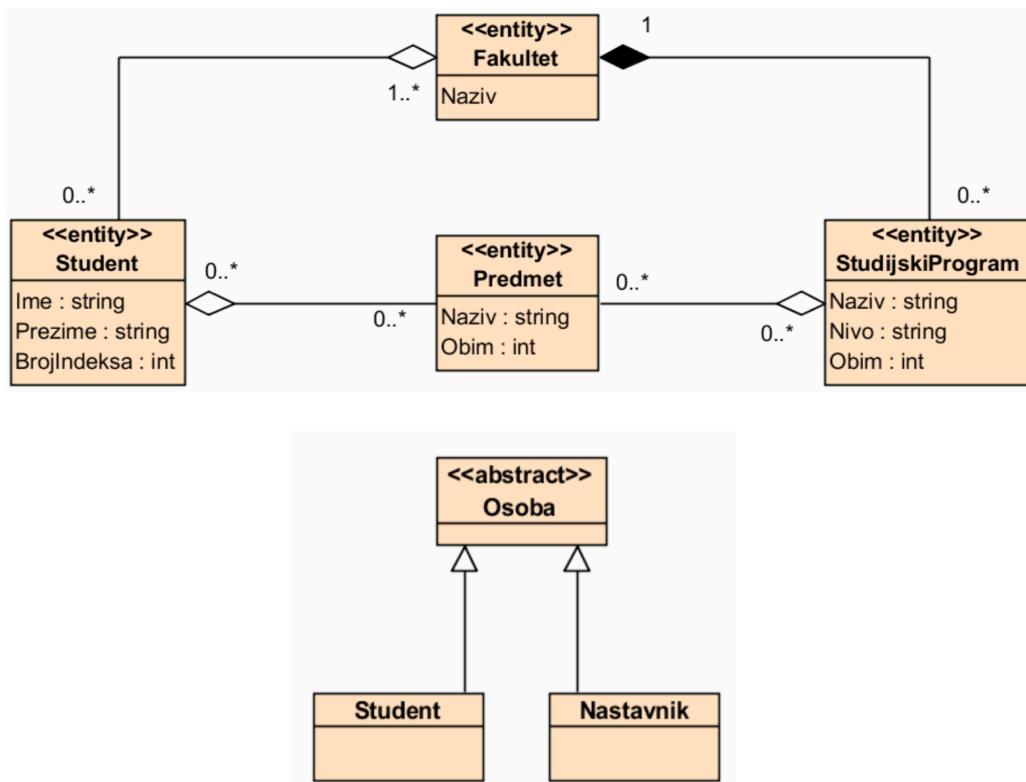
Podrazumevano označavanje je navođenjem naziva između dvostrukih izlomljenih zagrada:

- «actor»
- «entity»
- «abstract»
- «persistent»

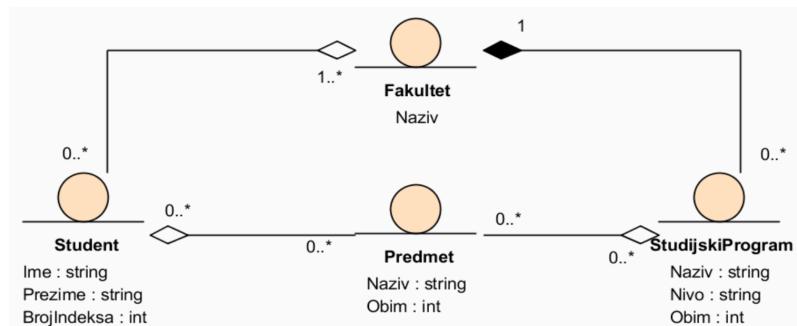
Obično se navodi ispod ili iznad naziva klase, može i ispred.

U primeru fakultet bi trebao biti tipa «persistent».

Označavanje stereotipova



Za neke uobičajene stereotipove kasnije su uvedene i grafičke oznake. One obično stoje kao dopuna u desnom gornjem uglu uobičajenog simbola za tu vrstu objekta. Mogu čak i da zamene osnovni grafički simbol neke vrste objekta.



67. Objasniti osnovne odnose u dijagramima klasa podataka

Asocijacija - odnos između dve klase. Označava da bar jedna od klase «zna» za neke objekte druge klase i na neki način upravlja njima.

Može biti:

- Funkcionalna - «uradi nešto za mene» (objekat A koristi usluge objekta B da bi nešto uradio)
- Strukturalna - «budi nešto za mene» (objekat B predstavlja deo objekta A)

U kontekstu modela podataka strukturalna je važnija.

Agregacija - posebna vrsta asocijacije. Implicitira da se jedna klasa «sastoji» od (jednog ili više) objekata druge klase.

Predstavlja slabiji oblik strukturalne asocijacije:

- Isti deo može da bude «sadržan» u više složenih objekata
- Ako se složeni objekat obriše, delovi mogu da nastave da postoje.

Praktično svaka strukturalna asocijacija, čija je kardinalnost 1-više, može da se posmatra kao forma agregacije.

Kompozicija - jači vid agregacije.

Predstavlja jači oblik strukturalne asocijacije:

- Uvek je binaran odnos
- Odnos celina/deo
- Jedan deo može da pripada samo jednom složenom objektu
- Ako se složeni objekat obriše uobičajeno ponašanje je da se obrišu i svi delovi

Nasleđivanje - odnos koji je suštinski za OO modeliranje, pa time i za model klasa podataka.

Osnovna klasa predstavlja opšći slučaj izvedenih klasa - generalizacija

Izvedene klase predstavljaju posebne slučajeve bazne klase - specijalizacija.

Relacioni model nema, a model entiteta i odnosa ima odgovarajući koncept.

Navigacija - Model klasa podataka podrazumeva da se za referisanje drugih objekata koriste neki vidovi jedinstvenih identifikatora, tzv. OID (načelno odgovara konceptu primarnog ključa).

Većina OOBP podrazumeva da identifikator objekta nema nikakvo semantičko značenje i njegova vrednost često ne može da se vidi i promeni.

- Takvi ključevi identificuju promenljive, a ne podatke.

Ako ne prejudiciramo implementaciju, onda možemo da ne navodimo surrogat ključeve: prihvatljivo na konceptualnom i logičkom nivoima, nije prihvatljiv na fizičkom.

68. Kako se dijagrami klase podataka koriste na različitim nivoima modeliranja?

Iste odnose često možemo da modeliramo na više ispravnih načina (naprimer da li birati asocijaciju ili agregaciju).

Izbor često zavisi od nivoa modela:

- Na konceptualnom nivou, cilj je da očuvamo semantiku odnosa iz domena problema.
- Na logičkom modelu, cilj je da ostvarimo stabilnu i jednoznačnu strukturu
- Na fizičkom nivou, cilj je (pored prethodnog) da omogućimo dobre performanse.

Razlike u načinu predstavljanja modela na konceptualnom/logičkom/fizičkom nivou slične su kao kod dijagrama tabela.

Konceptualni dijagram ne mora da sadrži:

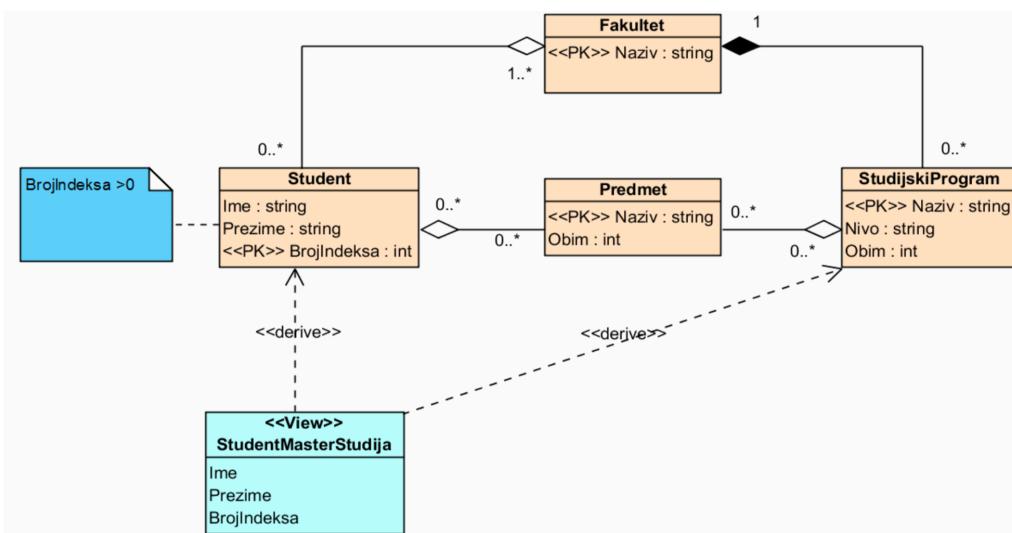
- Dodatne kolone surogat ključeva
- Dodatne kolone stranih ključeva
- Oznake ključeva
- Tipove kolona

Fizički dijagram sadrži i:

- Kolone surogat ključeva (ako su potrebne)
- Kolone vezanih tabela koji čine strane ključeve
- Oznake ključeva i vrsta ključeva
- Tipove kolona

69. Kako se u dijagramima klasa označavaju ključevi?

Simbolima ili tekstrom



70. Kako se u dijagramima klasa označavaju uslovi integriteta?

Razne druge stvari kao što su uslovi integriteta, indeksi, okidači, serverske procedure itd, mogu da se navode u okviru klase u odeljku ponašanja. Kao povezane klase ili komentari.

