

# **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.

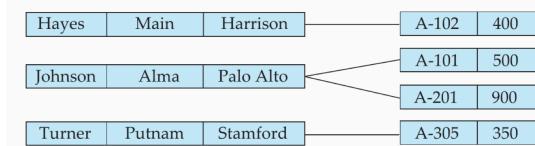
- Mrežni model
- Hijerarhijski model
- Relacioni model
- Model entiteta i odnosa
- Prošireni relacioni model
- Objektni model
- Objektno relacioni model

## 2. Objasniti osnovne koncepte mrežnog modela podataka

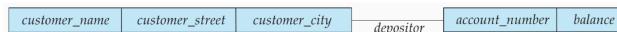
Mrežni model podataka je nalik na dijagramsko povezivanje podataka.

- 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.

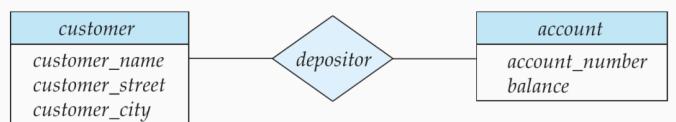
Пример података:



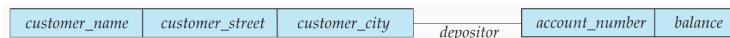
Пример модела:



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



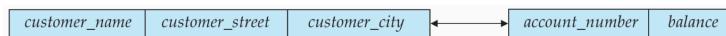
Однос више-више у мрежном моделу:



Однос један-више у мрежном моделу:



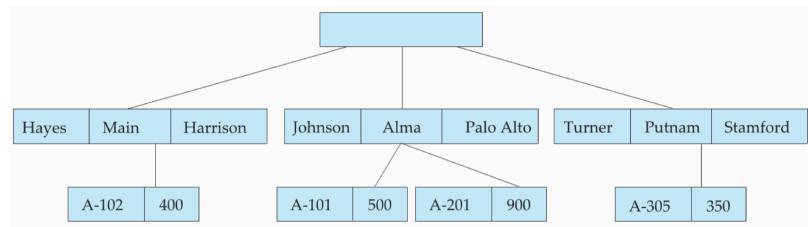
Однос један-један у мрежном моделу:



## 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.

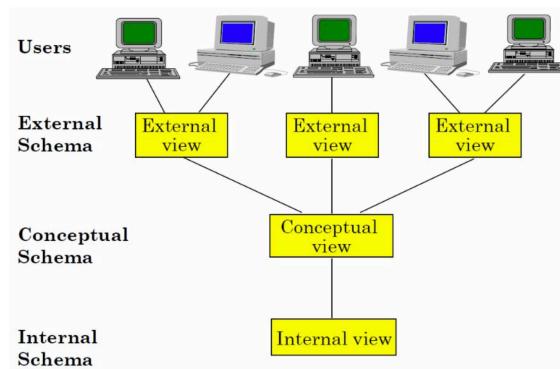
Svi pristupi se moraju kombinovati kako bi se dobio model arhitekture koji u svakoj posmatranoj tački daje dovoljno informacija o odgovarajućim aspektima.

## 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 podacima:

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

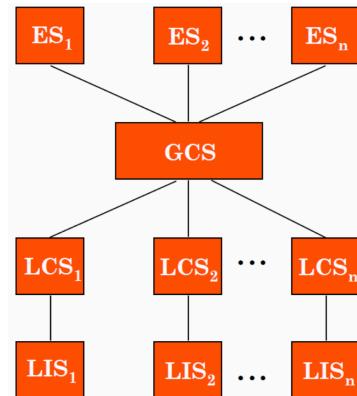
## 9. Objasniti koncept distribuiranih arhitektura na primeru arhitekture ravnopravnih čvorova

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, na primer 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 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 baze podataka.

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. Na primer, jedna opisna funkcija može biti  $ime(e_1) = Petar$ ,  $ime(e_2) = Marko$  i tako dalje.  $ime$  je znači funkcija.

Zato neka funkcija  $f_1 : E \mapsto D_1$  slika stvarni skup entiteta u domen funkcije  $f_1$ .

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

Imamo funkciju  $f_1$ , može da postoji i  $f_2$  na primer  $prezime, f_3$  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 se pravi jedna složena funkciju  $F(e) := (f_1(e), f_2(e), \dots, f_k(e)) = r$ . Ovo je opis jednog svojstva, modelira se jedno svojstvo:  $ime, visina, \dots$ . Domen složene funkcije je  $F : E \mapsto D_1 \times D_2 \times \dots \times D_k$ .

Za svaki entitet  $e$ , dobija se neka n-torka.

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

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

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  $\varphi$ , 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:

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

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  $R$ .

$A_i(e)$  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 postoji jedna relacija koja opisuje neku vrstu entiteta, na primer studente i želimo da opišemo još i profesore, učionice i ostalo, mora da se uvede skup relacija i na taj način se uvodi bazu podataka.

Odnosno, relaciona baza podataka je skup relacija.

Opis relacije čine domen relacije i nazivi atributa.

Relaciona shema je skup opisa relacija koje čine bazu podataka.

Čitavu bazu podataka opisujemo jednom relacionom 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).

Na primer, «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.

Na primer, student je upisao PBP, 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)
  - = vrednost relacije
  - = sadržaj relacione promenljive
- Promenljiva baze podataka
- Baza podataka
  - = vrednost baze podataka
  - = sadržaj promenljive baze podataka

Ažuriranje baze podataka je zamena 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  $\{(x.ime, x.prezime) \mid x \in Dosije\}$
- Restrikcija - izdvajanje podskupa redova  $\{x \mid x \in Dosije \wedge x.indeks = 2015400\}$
- Proizvod - uparivanje svih torki jedne sa svim torkama druge relacije  
 $\{(x, y) \mid x \in Dosije \wedge y \in Ispit\}$ .

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 nama treba  $(x_1, \dots, x_n, y_1, \dots, y_m)$ .

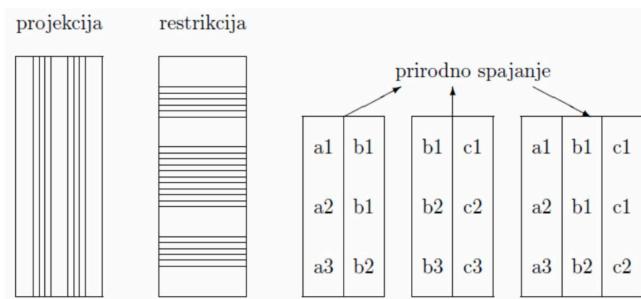
Kombinovanjem osnovnih operacija dobijaju se složeniji upiti, kao:

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

$$\{(x, y) | x \in Dosije \wedge y \in Ispit \wedge x.indeks = y.indeks\}$$

- Slobodno spajanje - spajanje sa restrikcijom po slobodnom uslovu

$$\{(x, y) | x \in Dosije \wedge y \in Ispit \wedge y.od\_godine \leq year(x.indeks) \leq y.do\_godine\}$$



## 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?

Na primer, 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 integriteta.

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

Formalnost modela olakšava formulisanje uslova integriteta.

## 23. Navesti osnovne vrste uslova integriteta u relacionoj bazi podataka

- Opšti uslovi integriteta (implicitni)
  - oni koji moraju da važe za svaku relaciju i svaku bazu podataka
    - (Na primer, 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  
(Na primer, 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 - proveravanje 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  $A_{i1}, \dots, A_{in}$  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 atribute 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 eksplisitno deklarišu i jedinstveni ključevi, koji su po svemu isti kao primarni ključevi, ali nije uobičajeno da se koriste pri referisanju.

## **27. Objasniti referencijski integritet u relacionom modelu**

Referencijski integritet predstavlja uslove o međusobnim odnosima koje moraju da zadovoljavaju torke dveju relacija.

Na primer, id\_studijskog\_programa u tabeli ispit, mora da odgovara id\_studijskog\_programa nekog studijskog programa u tabeli studijski programi.

Pravila referencijskog integrleta:

- 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.

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

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

U praksi se ostvaruje 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 ključu *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 ključu *PK* bar jedne torke relacije *B*.

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

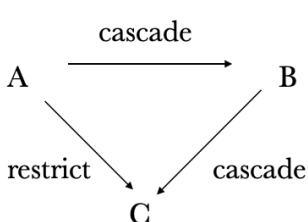
Poštovanje integriteta stranog ključa pri menjanju sadržaja baze podataka se određuju pravilima ažuriranja koja mogu da budu:

- Pravila brisanja (bira se tačno jedno)
- Pravila ažuriranja (bira 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 ključu 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 ključeve 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 opšti 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 (na primer, atribut mora da ima jednu od nekoliko vrednosti).
  - Može da se koristi za proveru ispravnosti složenih tipova podataka (na primer, 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 (na primer, 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 karaktera, 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 (na primer, 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: Pre svakog dodavanja ispita u tabelu ispit izvršiti proveru vrednosti ocene.

```
create trigger provera_ocena before insert on ispit
for each row
begin
    if (new.ocena > 10 || new.ocena < 5)
        then SIGNAL SQLSTATE '45000' SET message_text = 'Greska!'
    end if ;
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 na primer 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 na primer 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. Na primer, 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. Na primer 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. (na primer, 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.

Na primer, 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

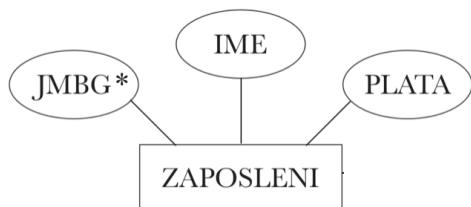
ER-dijagrami su sastavni deo modela, predstavljaju osnovni način zapisivanja semantičkih znanja o informacijama.

Skupovi entiteta se predstavljaju pravougaoncima.

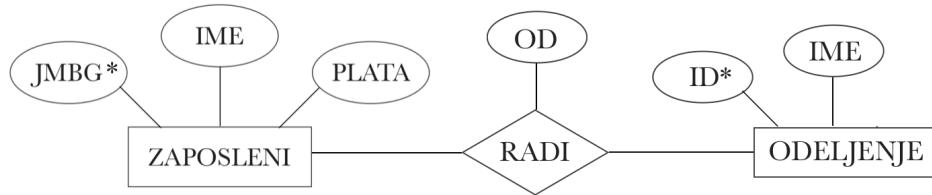
Atributi se predstavljaju elipsama.

Primenjivost atributa na skup entiteta se predstavlja linijom.

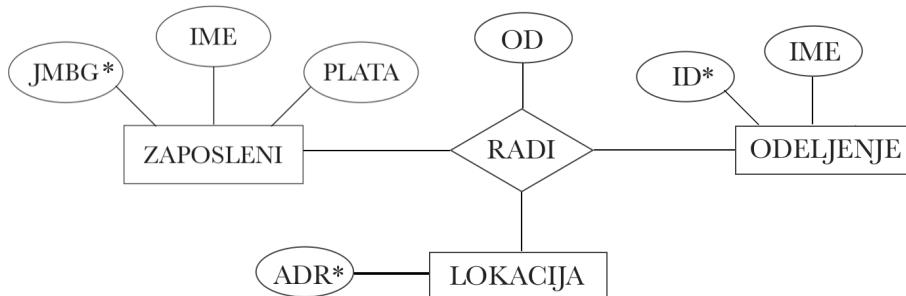
Primarni ključ označavamo zvezdicom.



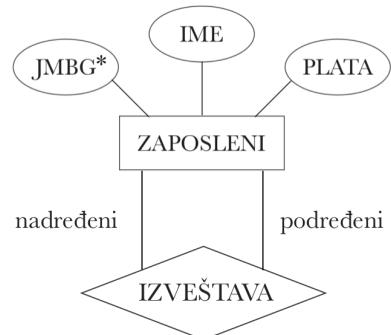
Odnosi se predstavljaju rombovima i mogu imati opisne attribute. (Na vežbama takve odnose još i uokvirujemo!)



Odnosi mogu da uključuju više entiteta.



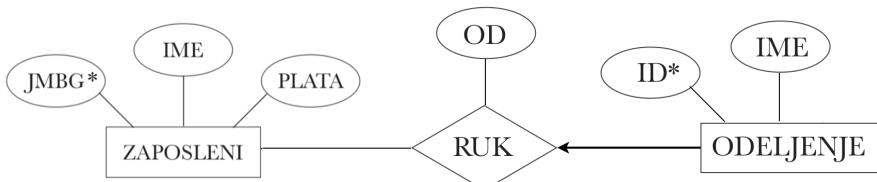
Odnosi mogu da budu i između više instanci istog skupa entiteta. U tom slučaju moraju da se imenuju uloge koje entiteti imaju u odnosima.



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

Uslovi ključa:

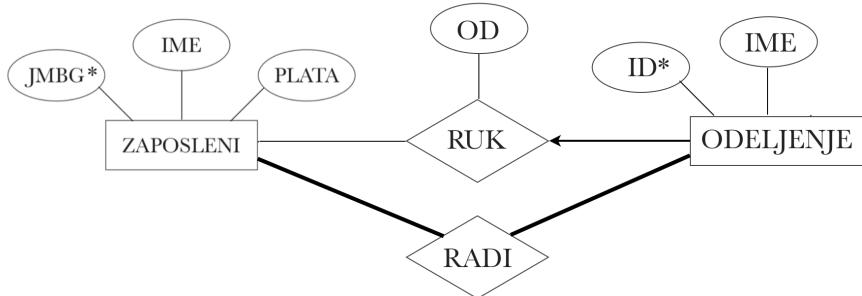
- Ako na osnovu entiteta može da se jednoznačno odredi odnos u kome učestvuje, onda je to ključni entitet odnosa. To se označava strelicom od entiteta prema odnosu!



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?

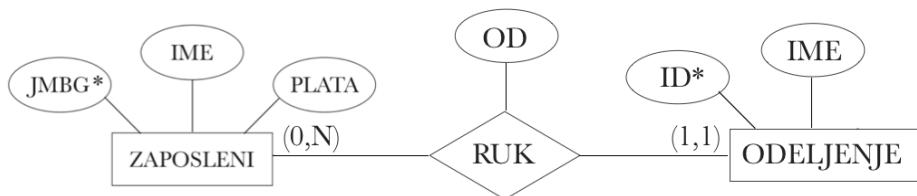
Kardinalnost odnosa opisuje u koliko različitih odnosa tog tipa može da učestvuje svaki od entiteta.

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:



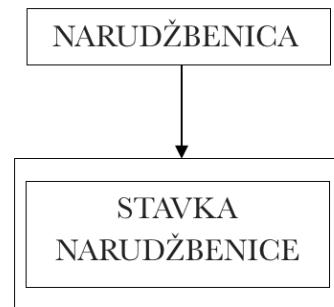
## 42. Kako se na ER-dijagramima označavaju slabti entiteti?

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.



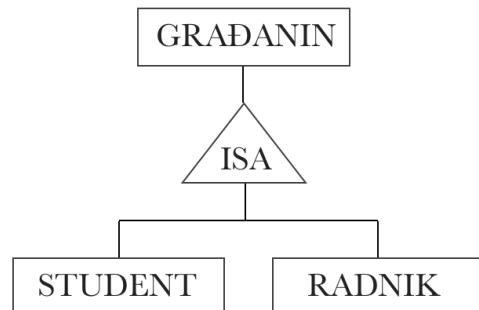
Primer:

Stavka narudžbenice ne može da postoji bez narudžbenice!

## 43. Kako se na ER-dijagramima označavaju hijerarhije?

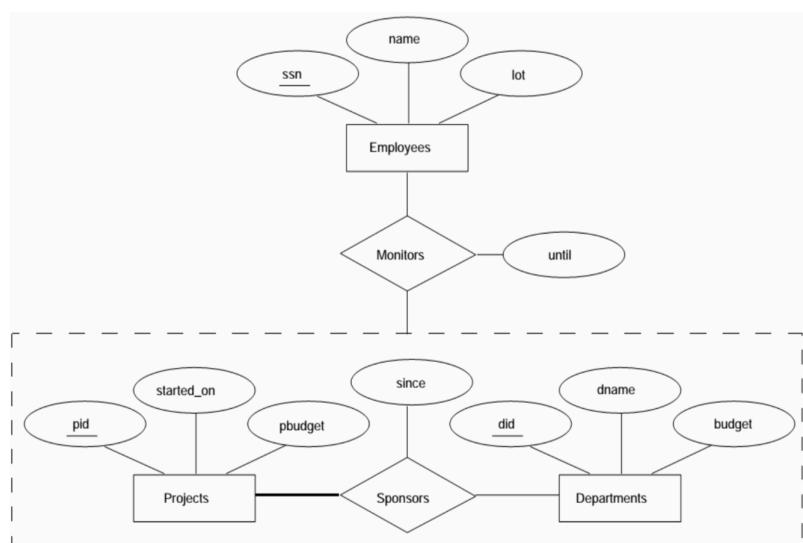
Hijerarhije klase su tzv. «ISA» hijerarhije.

- 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 isprekidanom 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 (na primer 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. Objasniti 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. Objasniti 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? ...

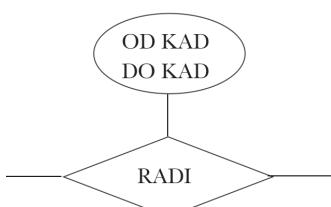
## 51. Objasniti dilemu entitet ili atribut

Ako imamo složen atribut, možemo da ga modeliramo kao:

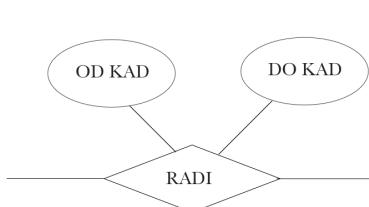
- Jedan složen atribut
- Više prostih atributa
- Jeden entitet

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

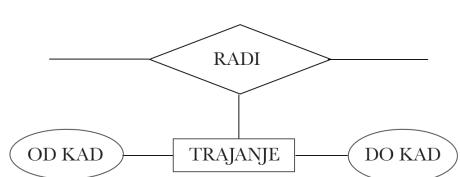
Jedan složen atribut



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.

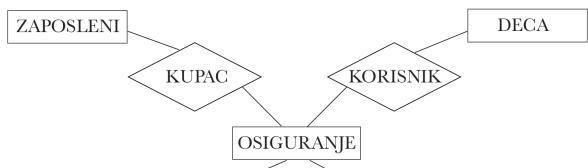
Pitanje je da li ih je bolje podeliti ili ostaviti?

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! Na primer, 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).

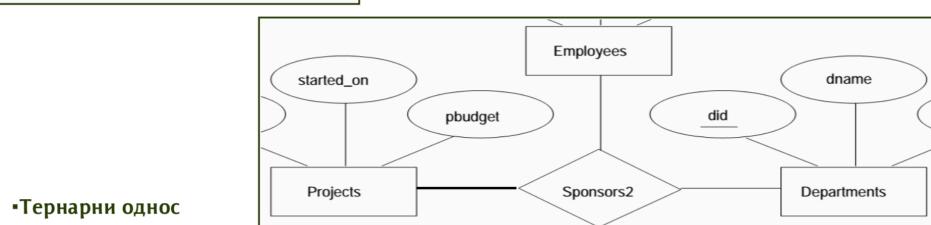
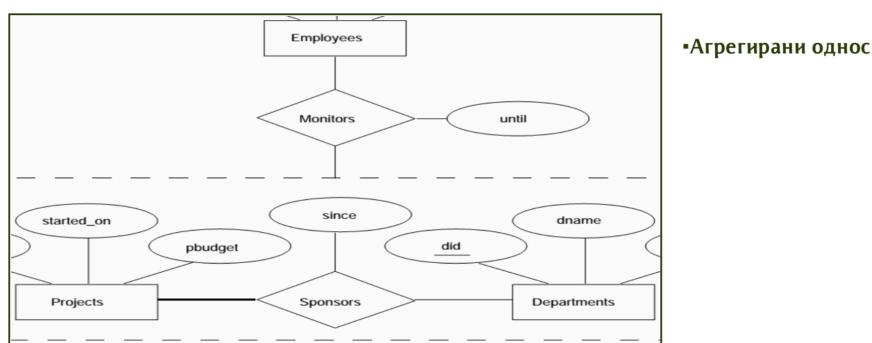
)

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 agregirani 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?

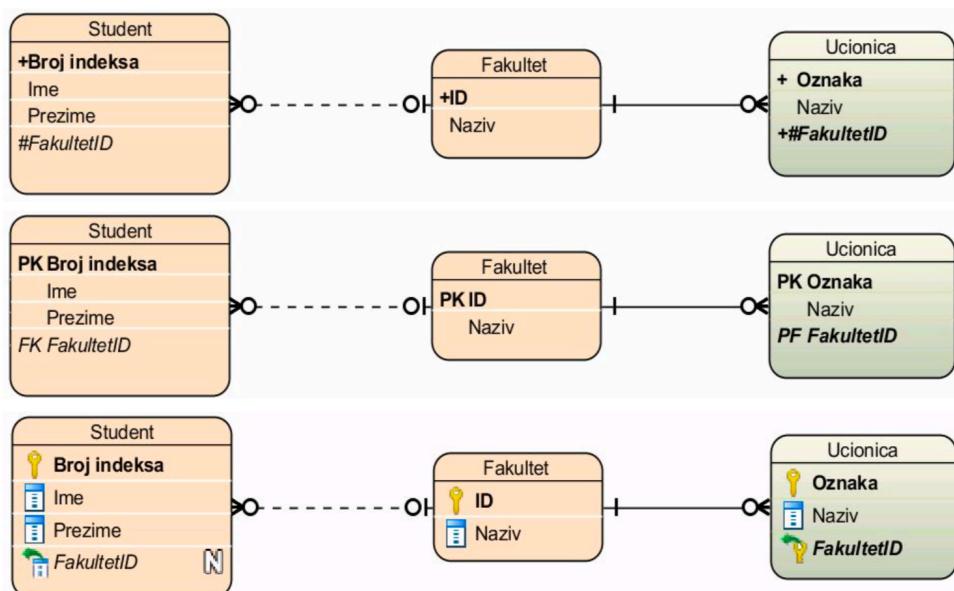
Dijagrami tabela su vrlo bliski fizičkom nivou, dok su ER dijagrami na konceptualnom nivou. Ne govore nam ništa o semantici, za razliku od ER modela.

## 60. Kako se označavaju ključevi u dijagramima 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 dijagramima 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, na primer odnos Student - Fakultet kaže: jedan fakultet može imati 0 ili više studenata.

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

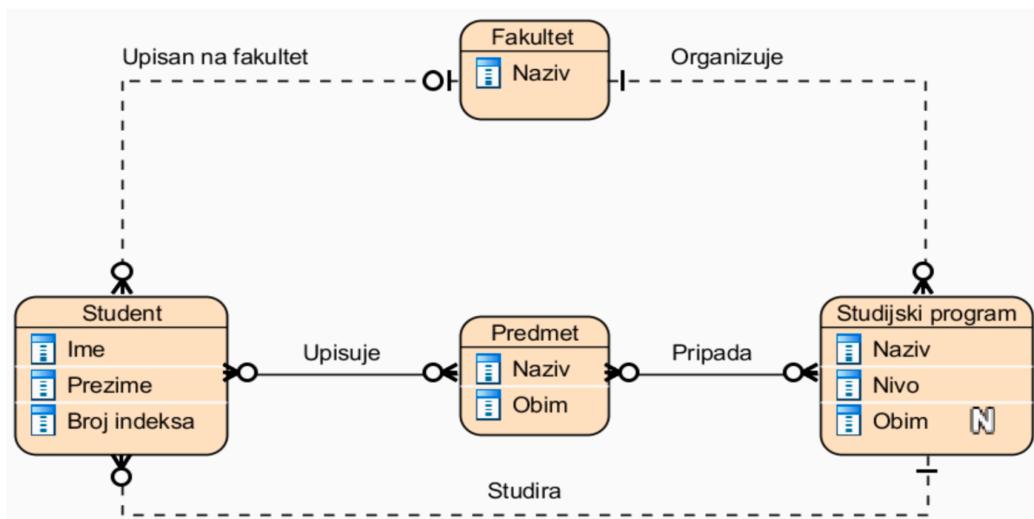
Osnovni oblik dijagrama najviše odgovara logičkom i posebno fizičkom nivou:

- Na konceptualnom nivou se prilagođava
- Na logičkom nivou može da se prilagođava ali ne mora (na primer 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 surogat ključeva, dodatne kolone stranih ključeva, oznake ključeva, tipove kolona.

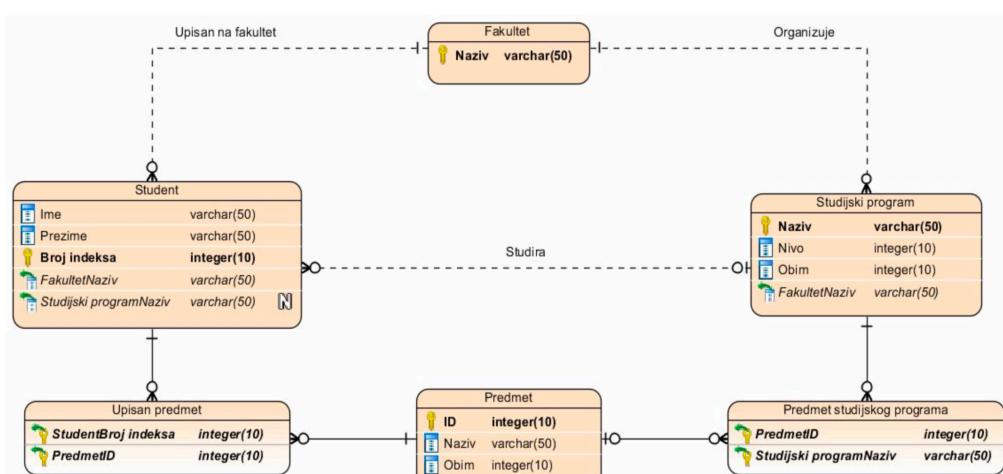
I na logičkom nivou može da se koristi ovakav dijagram.



Na

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

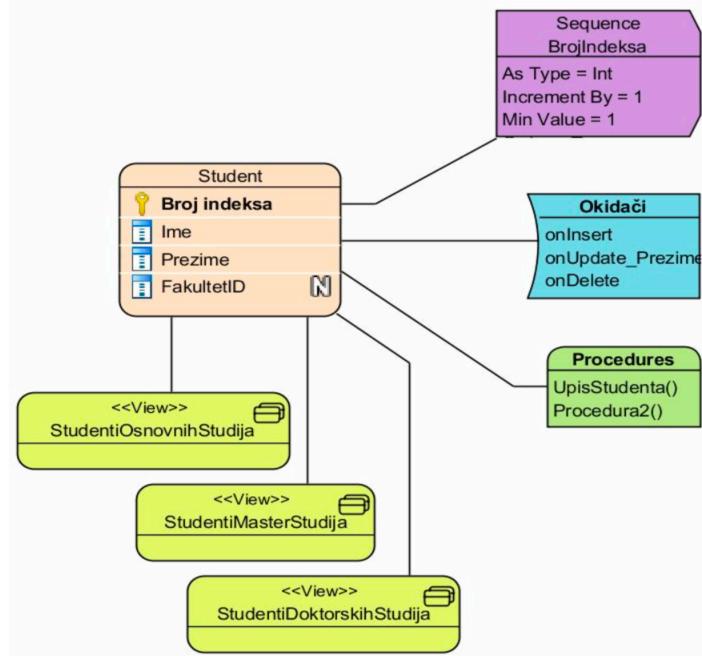
- kolone surogat ključeva
- kolone vezanih tabela koji čine strane ključeve
- oznake ključeva i vrsta ključeva
- tipovi kolona
- odnosi više-više se često zamenjuju tabelama koje ih modeliraju.



### 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 tabela 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 klase.

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 na primer 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!

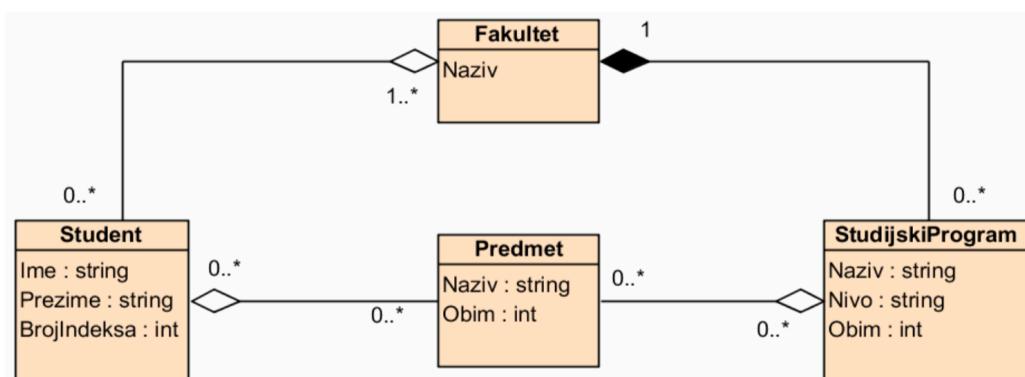
- Relacije (tabele) su skupovi podataka
- Klase su tipovi podataka

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 zagradica {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.

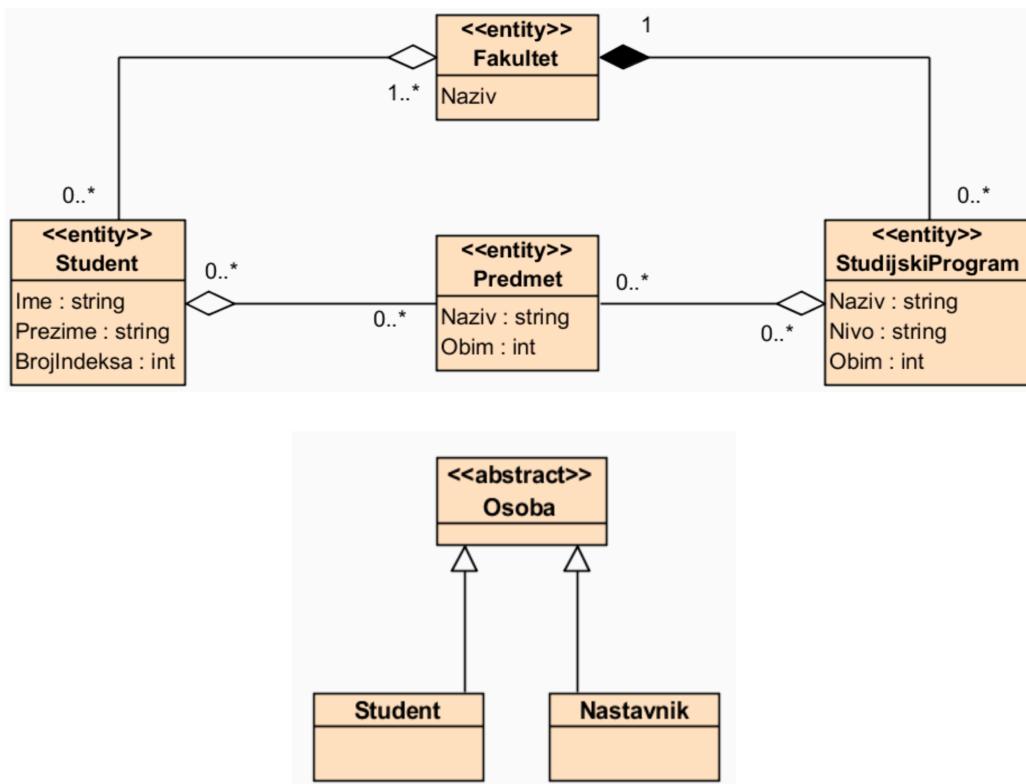
Podrazumevano označavanje je navođenjem naziva između dvostrukih izlomljenih zagrada:

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

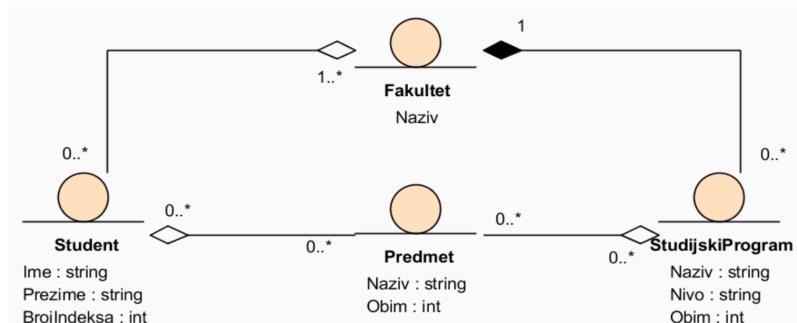
Obično se navodi ispod ili iznad naziva klase, može i ispred.

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 (na primer 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:

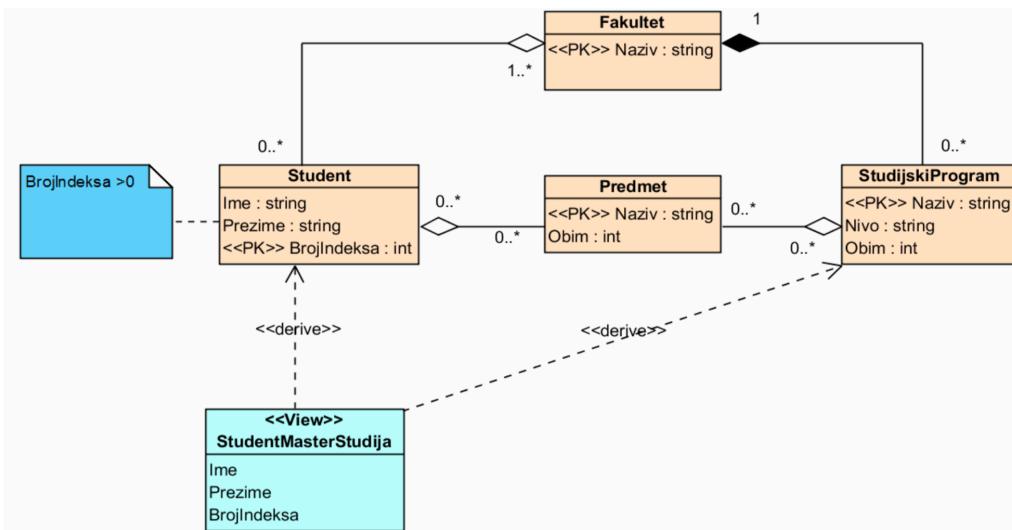
- Dodatne kolone surogat ključeva
- Dodatne kolone stranih ključeva
- Oznake ključeva
- Tipove kolona

Fizički dijagram sadrži i:

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

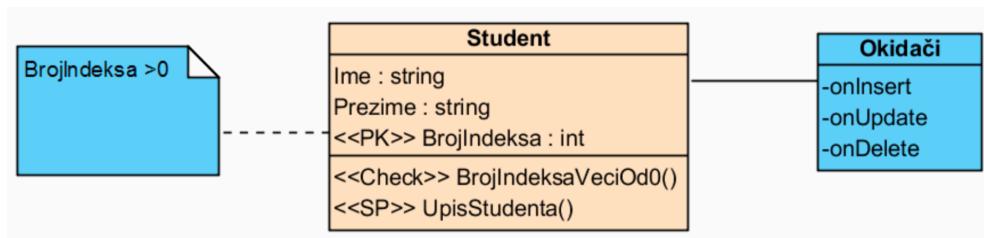
## 69. Kako se u dijagramima klasa označavaju ključevi?

Simbolima ili tekstrom



## 70. Kako se u dijagramima klase 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.



## 71. Navesti i objasniti po jednom rečenicom korake pri pravljenju konceptualnog modela BP.

— Malo o konceptualnom modelu —

Konceptualni model opisuje prepoznatu semantiku posmatranog domena i posebno je važan za razumevanje celine sistema, za razmenu informacija sa implementatorima aplikacija...

Dakle, suština konceptualnog modela je:

1. Da razumemo celinu sistema koji projektujemo.
2. Da vidimo koji su objekti koji postoje
3. Da sliku tog sistema zapišemo na način na koji će razumeti oni što razvijaju i BP i aplikaciju.

Model se zapisuje primenom:

- ER modeliranja
- UML modeliranja (objektnim pristupom, pravljenjem dijagrama klase)

Ključna karakteristika je apstraktnost modela, koja smanjuje nivo detaljnosti, izostavlja suvišne informacije (bez nebitnih atributa, bez tabela neophodnih za fizičku implementaciju). Apstraktnost modela uopštava predstavljanje odnosa. Ideja je da se lakše razume.

Tradicionalni pristup pravljenja konceptualnog modela BP - modeliranje odozdo-naviše.

Polazi se od pojedinačnih atributa, atributi se grupišu u normalizovane relacije (entite...), vodi se računa o funkcionalnim zavisnostima. Pristup je uspešan je za manje baze, kada uglavnom neposredno iz analize zahteva mogu da se ustanove potrebni podaci i njihovi odnosi.

Ispravan pristup pravljenja konceptualnog modela počiva na sledećim koracima:

- Analiza zahteva
- Konceptualno modeliranje podataka
- Integrisanje pogleda (lokalnih shema)
- Grupisanje entiteta

## **72. Objasniti korak *Analiza zahteva pri pravljenju konceptualnog modela BP***

- Zahtevan i važan korak
- Prepoznajemo osnovne delove celine
- Prepoznajemo ko se sve pojavljuje kao korisnik podataka i na koji način koristi podatke i iz tih različitih uglova modeliramo one delove sistema koji su bitni za različite vrste korisnika

Da bi se razumele potrebe za podacima često mora da se detaljno razumeju svi procesi koji se odvijaju u sistemu.

## **73. Navesti i objasniti ciljeve koraka *Analiza zahteva pri pravljenju konceptualnog modela BP***

- Prevodenje funkcionalnih zahteva u kontekst trajnih podataka
- Razumevanje i opisivanje
  - Skupova potrebnih podataka i njihovih odnosa
  - Strukture potrebnih podataka
  - Skup pojedinačnih transakcija koje se izvršavaju nad podacima
- Definisanje svih nefunkcionalnih zahteva: integritet, performanse, bezbednost... (nefunkcionalni zahtevi ne čine deo strukture podataka)
- Određivanje i oblikovanje dodatnih uslova implementacije: tehnologije, hardverske i softverske pretpostavke, aplikativni i korisnički interfejs...
- Izrada iscrpne dokumentacije za sve navedeno.

## **74. Objasniti ukratko korak *Konceptualno modeliranje podataka pri pravljenju konceptualnog modela BP***

Pokušavamo da napravimo neku sliku naše baze.

Glavni koraci:

- Klasifikacija skupova podataka - prepoznavanje entiteta / klasa i atributa
- Prepoznavanje odnosa između entitetima/klasama
  - Generalizacija / specijalizacija i hijerarhije
  - Asocijacije i složeniji odnosi

## **75. Koje poslove obuhvata konceptualno modeliranje podataka?**

### **76. Šta obuhvata *klasifikacija skupova podataka* ?**

Svaka imenica je kandidat za entitet ili atribut i često nije očigledno šta je šta.

Klasifikacija skupova podataka je prepoznavanje entiteta / klasa i atributa.

Smernice:

- Entiteti bi trebalo da sadrže opisne informacije
  - Ako imamo opisne informacije o nekom podatku, onda bi to trebalo da bude entitet (na primer, predmet ima «naziv» i «espb» pa bi predmet trebalo da bude entitet)

- Ako neki podatak ima samo identifikator (može se jednostavno opisati) onda je to verovatno atribut (na primer, ocena se identificuje svojom vrednošću i nisu potrebni dodatni opisi, pa je zato atribut)
- Ako nešto ima višestruku ili složenu vrednost (a ne skalaru) onda je verovatno entitet
- Atributi bi trebalo da pripadaju onim entitetima koje najneposrednije opisuju
  - Obično može da se ustanovi iz funkcionalnih zavisnosti

## **77. Kako se ustanavljava da li bi nešto trebalo da bude atribut ili entitet?**

Pogledati prethodno pitanje.

Da li nešto atribut ili entitet zavisi od toga šta je podatak. Ako je podatak jednostavan skalar to ćemo najverovatnije posmatrati kao atribut, ali ako je u pitanju skup podataka onda je bolje da bude skup atributa, relacija, entitet.

U prethodnom pitanju smo rekli šta može biti kandidat za entitet, dodatno podatak najverovatnije mora biti entitet ako taj podatak potencijalno može da menja vrednost.

Primer: Da li je mesto\_rođenja entitet ili atribut?

Skroz je validno da bude atribut, ali to može da smeta ako želimo da lakše i brže grupišemo podatke po nekom atributu, na primer ako želimo da nađemo sve podatke tako da zadovoljava neki konkretni grad, to će biti brže ako je entitet a ne atribut (odnosno niska). Ali ako ne radimo to često, nije strašno da bude atribut. Međutim, šta ako se promeni ime grada? Postavlja se pitanje da li nam je jednostavno da posle ažuriramo ovakvu promenu? Ako se podatak o nazivu grada sadrži u samo jednoj tabeli, onda samo zamenimo u toj tabeli vrednosti. Ali šta ako se pojavljuje u više tabela? To je problem.

## **78. Koji elementi odnosa moraju da se ustanove pri modeliranju odnosa?**

Jedan od osnovnih odnosa koji trebamo da prepoznamo ako je tesno vezan sa raspoređivanjem atributa u entitete jeste odnos generalizacije ili specijalizacije.

Na primer, da li treba da pravimo hijerarhiju klasa student, student osnovnih, student master, student doktorskih studija. Ako treba da pravimo, onda moramo da prepoznamo koji atributi idu gore a koji dole. Uvek se trudimo da što više ide gore.

Znači, što apstraktnije to bolje:

- Atribute i odnose je potrebno popeti što više uz hijerarhiju
- Ključne atrbute je potrebno vezati za baznu klasu (entitet)

Za svaki odnos je potrebno prepoznati:

- Učesnike i red odnosa, tj broj učesnika (najčešće 2 strane učestvuju u odnosu)
- Kardinalnost i opcionost/obaveznost
- Dodatne atrbute koji opisuju odnos
- Jasan naziv i semantiku uloge odnosa

## 79. Objasniti problem redundantnih odnosa. Kako se opčavaju redundantni odnosi?

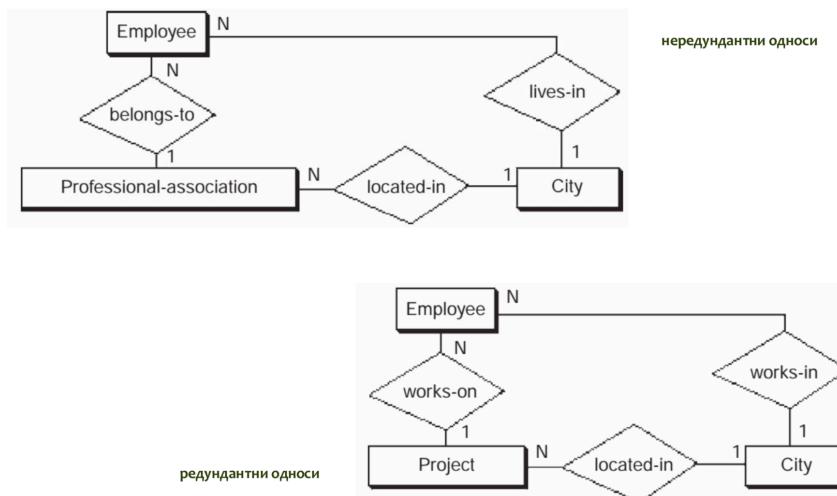
Redudantni odnosi

- imaju za rezultat teško otklonjivu redundantnost u modelu
- ometaju normalizaciju

Potrebno ih je eliminisati jer omogućavaju greške u podacima.

Obratiti pažnju da isti odnosi mogu biti i redundantni i neredundantni, redundantnost zavisi od semantike.

Primer



(Pogrešne kardinalnosti!)

Na gornjem dijagramu imamo zaposlenog koji može da pripada većem broju udruženja. Asocijacija ili udruženje može da se nalazi u nekom gradu. Zaposleni živi u jednom gradu, ali može da bude u udruženju koje se nalazi u drugom gradu. Ovde nemamo redundantne odnose, postoje višestruke veze do grada, ali nije ista informacija u pitanju!

Dok na donjem dijagramu zaposleni radi u nekom gradu, a sa druge strane imamo da zaposleni radi na nekom projektu koji se nalazi u nekom drugom gradu, to već jeste redundantna informacija. Ne može zaposleni da radi u jednom gradu, a da radi na projektu u nekom drugom gradu!

## 80. Kako se postupa sa odnosima sa više od 2 učesnika?

Odnos je višeg reda ako obuhvata više od 2 učesnika.

Uvek kada imamo ovakve odnose, moramo dobro razmislići da li nam zaista trebaju! Vrlo često je bolje da takve odnose podelimo na više jednostavnijih odnosa.

Uputstvo:

- Uvek pokušati sa više binarnih odnosa
- Ako se ispostavi da je nemoguće, tek onda modelirati odnos višeg reda.

Primer:

U rasporedu časova složen odnos je «nastavnik N drži čas iz predmeta P u učionici U u terminu T». Međutim, odnos nastavnika i predmeta je važan i nezavisno od rasporeda i izdvaja se u koncept grupe. Pa je onda grupa G = «nastavnik N drži predmet P». Početni odnos je onda jednostavniji, tj. grupa G ima čas u učionici U u terminu T.

## **81. Objasniti razliku između lokalnih i globalnih shema pri modeliranju BP. Zašto su obično neophodni lokalni pogledi?**

Konceptualno modeliranje se često odvija po delovima odvojeno za različite aplikacije ili delove aplikacije jer je tako jednostavnije i razumljivije.

Svaki pojedinačan model se naziva pogledom ili lokalnom shemom.

Ovaj pogled nije isto što i pogled na tabelu, ovde je pogled na domen.

Primer razdvojenih pogleda:

Informaciona služba fakulteta

- Iz ugla studenta - potrebno je da se zna da je neki nastavnik zadužen za neki predmet i nije bitno kada je ko izabran u koje zvanje
- Iz ugla kadrovske službe - nije bitno ko je zadužen za koji predmet već kada je ko izabran u koje zvanje.

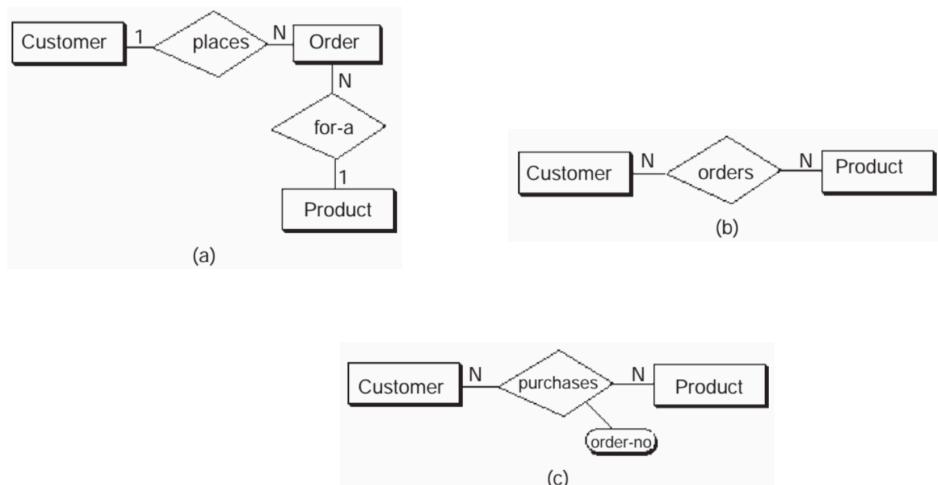
Ovo su dve lokalne sheme koje želimo objediniti u jednu globalnu shemu integracijom pogleda.

## **82. Objasniti ukratko korak Integrisanje pogleda pri pravljenju konceptualnog modela BP i navesti osnovne postupke**

Integracija pogleda je postupak spajanja lokalnih shema u jednu globalnu shemu.

Predstavlja sastavni deo konceptualnog modeliranja, ali se često opisuje kao poseban korak.

Primer različitih modela za isti domen



(Enumeracija je pogrešna)

Osnovni koraci:

- Poređenje shema i prepoznavanje konflikata - posmatranje shema i prepoznajemo gde su svi mogući konflikti (na primer, na prethodnom dijagramu pod a) vidimo da imamo više odnosa (places i for-a) dok na drugim nemamo, na dijagramu pod c) imamo atribut odnosa purchases, dok kod drugih nemamo ... )
- Preuređivanje shema i razrešavanje konflikata - preoblikovanje pojedinačnih shema tako da budu međusobno saglasne
- Spajanje i restrukturiranje shema.

### **83. Koji su osnovni problemi pri integrisanju pogleda?**

### **84. Navesti i objasniti vrste konflikata koji mogu da nastanu pri integrisanju pogleda**

Vrste konflikata:

- Konflikti imena - neujednačeno imenovanje entiteta i odnosa. Imamo na 2 moguća načina:
  - Konflikt sinonima - različiti nazivi za iste koncepte, prvi i najlakše rešiv problem
  - Konflikt homonima - isti naziv se koristi za različite koncepte, može da bude veliki problem ako je različito tumačena specifikacija zahteva. Teže rešivo od sinonima i zato ima prioritet pri rešavanju.
- Strukturni konflikti - različiti strukturni elementi se upotrebljavaju za modeliranje istih koncepcata. Da bi pogledi mogli da se spoje, odgovarajući koncepti moraju da imaju identične strukture.(na primer, da li je nešto odnos ili entitet)
- Konflikti ključeva - istom entitetu u različitim pogledima dodeljeni različiti ključevi, neophodno je da se ključevi ujednače (na primer, tipovi ključeva su različiti, atributi koji čine ključeve nisu isti)
- Konfliktni zavisnosti - različito prepoznate funkcionalne zavisnosti.

### **85. Objasniti detaljno konflikte imena pri integrisanju podataka**

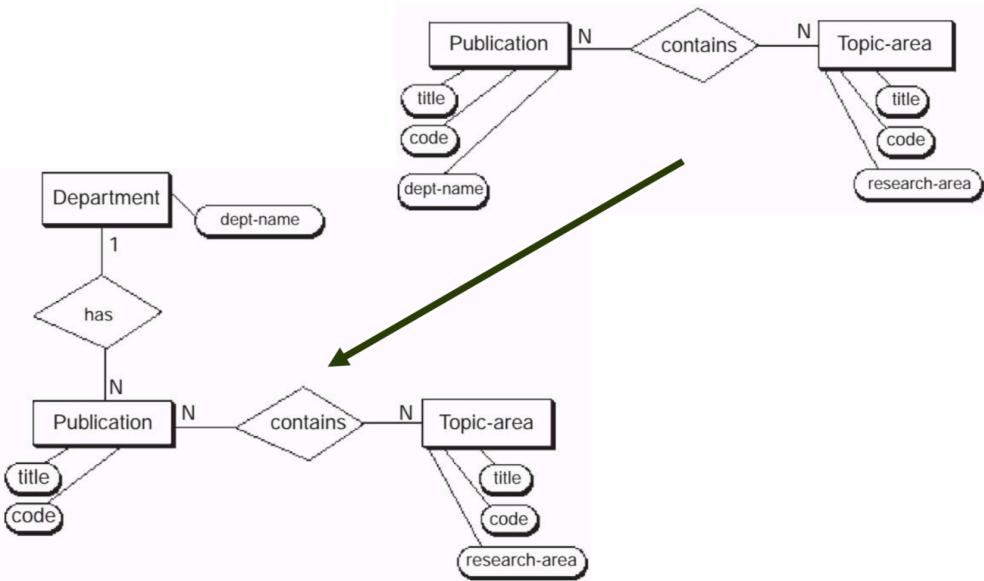
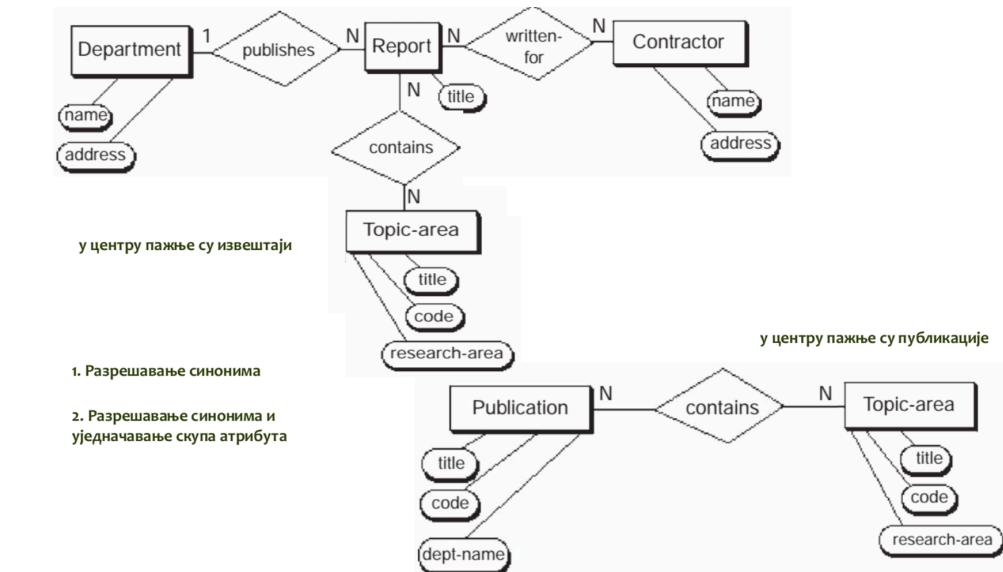
Deo pitanja 84.

### **86. Kako se razrešavaju konflikti pri integrisanju podataka**

Razrešavanje konflikta

- Uključuje projektante svih konfliktnih pogleda
- Može da zahteva dodatnu ili ponovljenu analizu zahteva
- Može da zahteva ozbiljno modifikovanje pogleda
- Tehnike razrešavanja konflikata su iste kao i za pravljenje konceptualnog modela

Primer konceptualnog konflikta



## 87. Kojim principima se rukovodi pri spajanju i restruktuiranju lokalnih shema? Objasniti ih.

Spajanje i restruktuiranje shema nastupa nakon usaglašavanja konflikata.

Rukovodi se principima:

- Potpunosti - svi koncepti iz lokalnih shema moraju u potpunosti očuvano da se prenesu u globalnu shemu.
- Minimalnosti - svi redundatni odnosi (i entiteti i odnosi) moraju da se uklone iz globalne sheme, uvođenjem generalizacije
- Razumljivosti - globalna shema mora da bude razumljiva svim korisnicima

## 88. Šta je grupisanje entiteta pri pravljenju konceptualnog modela BP i navesti osnovne postupke? Zašto je važno?

Nakon integrisanja podataka može da se dobije velika shema, koja nije pregledna zbog veličine.

Cilj grupisanja entiteta je grupisanje delova modela radi predstavljanja jednog velikog modela pomoću više manjih dijagrama.

Tehnika grupisanja:

- Detalji predstavljanja entiteta se obrišu i entitet se zaokruži dvostrukom linijom
- Grupa entiteta se zamenjuje simbolom grupe sa podebljanim okvirom

Važno je jer nam olakšava sagledanje sheme.

## **89. Navesti i objasniti principe grupisanja entiteta pri pravljenju konceptualnog modela BP**

Principi grupisanja:

- Grupisanje prema dominantnosti - dominantni entiteti se uočavaju na osnovu odnosa, jer učestvuju u većem broju odnosa i posredno povezuju veće delove dijagrama. Jedan dominantan entitet se grupiše sa svim pripadajućim nedominantnim entitetima.
  - Slabi entiteti se pridružuju jakom
  - Delovi dijagrama koji opisuju odnose višeg reda se grupišu
  - Ako postoji agregacija, delovi se pridružuju celini
- Grupisanje prema apstraktnosti - vrši se ako postoje hijerarhije. Hijerarhija se predstavlja jednim baznim entitetom, ako postoje bitni specifični odnosi nekih elemenata hijerarhije, onda se celi hijerarhija deli na više grupa.
- Grupisanje prema uslovima - ako postoje neki složeni uslovi koji moraju da važe u nekim odnosima, odgovarajući entiteti se grupišu
  - Cilj je sklanjanje složenih uslova iz velikog dijagrama, radi povećanja čitljivosti
  - Ako ti složeni uslovi imaju visok značaj za celinu dijagrama, onda ovakvo grupisanje nije poželjno
- Grupisanje prema odnosima - odnosi višeg reda i odgovarajući entiteti mogu da se grupišu, ovakva grupa predstavlja odnos kao jednu celinu.

Primenjuju se iterativno i/ili rekurzivno.

## **90. Objasniti postupak grupisanja entiteta pri pravljenju konceptualnog modela BP**

Postupak grupisanja:

- Prepoznati elemente koji se grupišu u okviru funkcionalnih oblasti
  - Funkcionalne oblasti su poslovne jedinice ili grupe podataka koje se najčešće koriste zajedno
- Grupisati entitete
  - Svaka grupa mora da u potpunosti pripada jednoj funkcionalnoj oblasti
  - Ako postoje konflikti između više alternativnih grupisanja, ne praviti takve grupe
- Napraviti grupe višeg reda
  - Rekurzivnom primenom postupka

- Proveriti ispravnost dijagrama
  - Svi interfejsi između entiteta/objekata i drugih delova/nivoa dijagrama moraju da budu konzistentni

## **91. Šta je logički model baze podataka?**

Logički model bi trebalo da reši sve nedoumice oko načina modeliranja nekih delova oko prevođenja konceptualnog modela na jezik neke baze koju pravimo. Treba da sadrži sve suštinske elemente konceptualnog modela: neredundantnosti, konzistentnosti podataka i ostalo.

Konceptualni model je fokusiran na semantiku i na odnose.

Logički model silazi bliže izabranom implementacionom modelu.

Apstraktni koncepti konceptualnog modela se prevode u konkretne logičke modele.

Izborom konkretne vrste baze podataka (npr. relacione) i dalje ostajemo u domenu logičkog projektovanja.

Ulaz za postupak logičkog modeliranja je konceptualni model.

Izlaz je detaljan (npr. relacioni) model:

- Logička shema svih trajnih objekata (relacija)
- Specifikacija svih uslova i ograničenja
- Poznati su svi ključevi i odgovarajući surrogat-atributi kao i načini implementacije svih odnosa

Logički model uzima u obzir dodatne nefunkcionalne zahteve, koji su u fazi konceptualnog projektovanja možda mogli da ostanu zanemareni. Vodi se računa o fleksibilnosti, proširivosti.

Logički model tačno određuje koji se podaci čuvaju i kako će se njima rukovati.

## **92. U čemu je osnovna razlika između konceptualnog i logičkog modeliranja?**

Deo iz pitanja 91

## **93. Šta je očekivani rezultat logičkog modeliranja?**

Deo iz pitanja 91.

Logičko modeliranje može da ima za rezultat i neke značajne odluke:

- Može da se ne pravi jedna nego više baza podataka
- Može da se struktura prilagodi očekivanim promenama
- Mogu se donositi neke odluke u skladu sa specifičnostima izabrane implementacije SUBP

## **94. Zašto nije dobro preskočiti logički model i praviti fizički model na osnovu konceptualnog?**

Preskakanje logičkog modela nosi rizik da se izgube neke od poželjnih karakteristika baze podataka:

- Kompletност

- Integritet
- Fleksibilnost
- Efikasnot
- Upotrebljivost

Kada su male baze u pitanju moguće je odmah konceptualni model prevesti u fizički, međutim teško da ćemo moći da izvršimo neku promenu na fizičkom nivou, jer bismo morali prvo opet da pravimo konceptualni. A ako imamo logički model, nekad možemo različite promene vršiti na njemu.

## **95. Kako teče postupak pravljenja logičkog modela?**

Logički model se pravi iterativno.

Prva iteracija se pravi na osnovu konceptualnog modela, tj. prevođenje konceptualnog modela u logički model - oba modela mogu da se izražavaju na ER-u ili UML-u, pa je prevođenje često transformisanje ili dopunjavanje i nadograđivanje. Pokušavamo da to prevođenje opišemo u formi skupa pravila.

Svaka naredna se pravi menjanjem prethodne.

Logički model ćemo najčešće da predstavljamo jezikom relacionog modela:

- Entiteti se predstavljaju kao relacije
- Složeni odnosi se predstavljaju kao relacije
- Jednostavni odnosi se predstavljaju stranim ključevima - strani ključ iz vezane relacije B prema baznoj relaciji A ima vrlo ograničenu semantiku kardinalnosti:
  - Kardinalnost A je uobičajeno 0..1 ako ključ može da sadrži NULL
  - Kardinalnost A može da bude 1, ako ključ ne sme da sadrži NULL
  - Kardinalnost B je uobičajeno 0..\*
  - Kardinalnost B može da bude 0..1 ako se doda uslov jedinstvenosti ključa uz dopušteno ponavljanje vrednosti NULL

## **96. Na osnovu kojih kriterijuma se pristupa menjanju logičkog modela?**

Kriterijumi za menjanje su željene karakteristike:

- Da li model ispunjava funkcionalne zahteve?
- Da li model ispunjava nefunkcionalne zahteve?
- Da li je model kompletan?
- Da li model garantuje integritet podataka?
- Da li model pruža potrebnu fleksibilnost?
- Da li model omogućava efikasan rad?
- Da li je model upotrebljiv?

## **97. Kako se konceptualni model prevodi u logički?**

- Prevesti svaki entitet u relaciju sa ključem i neključnim atributima.
  - Ako postoji odnos 1-N, dodati ključ entiteta sa strane 1 u relaciju sa strani N kao strani ključ

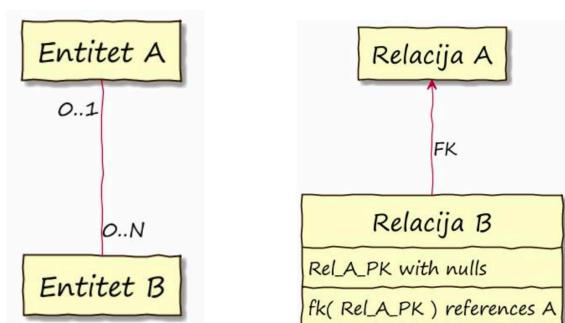
- Ako postoji odnos 1-1 dodati ključ jednog entiteta u drugu relaciju kao strani ključ
    - Ako postoji prirodan odnos roditelj-dete, dodajemo ključ u dete
    - U suprotnom dodajemo ključ u relaciju sa manje redova
  - Svaki entitet u hijerarhiji prevesti u relaciju
  - Svaka relacija mora da ima primarni ključ
  - Suvišni ključevi će se eliminisati pri prečišćavanju sheme
- Prevesti svaki binarni (ili rekurzivan) odnos N-N u relaciju koja sadrži ključeve učesnika u odnosu
    - Svaki odnos se prevodi u relaciju koja kao strane ključeve sadrži ključeve entiteta koji učestvuju u odnosu
    - Voditi računa o usklađivanju ograničenja za primarne i strane ključeve sa funkcionalnim zavisnostima
- Prevesti svaki odnos sa 3 ili više učesnika u relaciju
    - Svaki odnos se prevodi u relaciju koja kao strane ključeve sadrži ključeve entiteta koji učestvuju u odnosu
    - Dodati jedinstvene ključeve u skladu sa funkcionalnim zavisnostima
    - Voditi računa o usklađivanju ograničenja za primarne i strane ključeve sa funkcionalnim zavisnostima

## 98. Kakva je izražajnost logičkog modela u odnosu na konceptualni? Šta može da se vidi u konceptualnom modelu a obično ne može u logičkom?

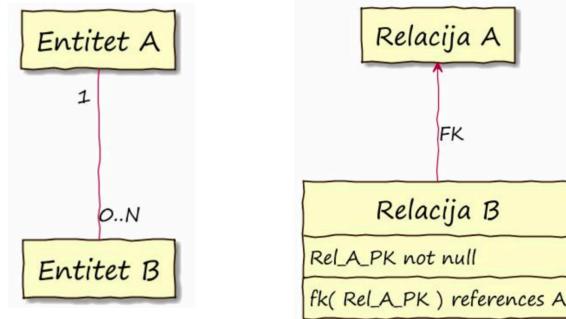
### 99. Kako se u logičkom modelu modeliraju odnosi 0..1 - 0..N , 1 - 0..N, 0..1 - 1 , 0..1 - 0..1 ?

**Obratiti pažnju** da se ovde drugačije čitaju kardinalnosti nego na vežbama. Na prvom sledećem dijagramu na primer, za entitet A važi da ima 0 ili najviše N entiteta B, dok B može da ima 0 ili jedan entitet A.

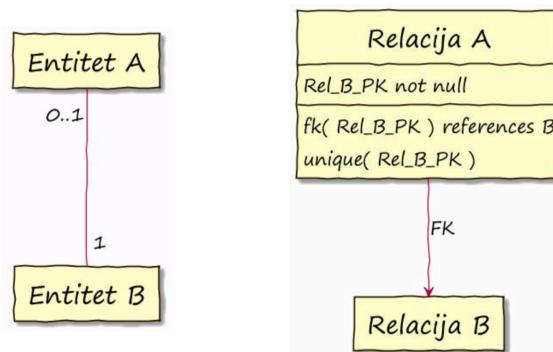
- **0..1 - 0..N** - Uobičajeno za odnose roditelj - potomak, celina - deo (ako celina **može** da bude bez delova, ako deo **može** da bude bez celine).  
Često kod agregacija. Relaciji koja je deo:  
dodaje se strani ključ u odnosu na celinu, koji **može** biti nedefinisan.



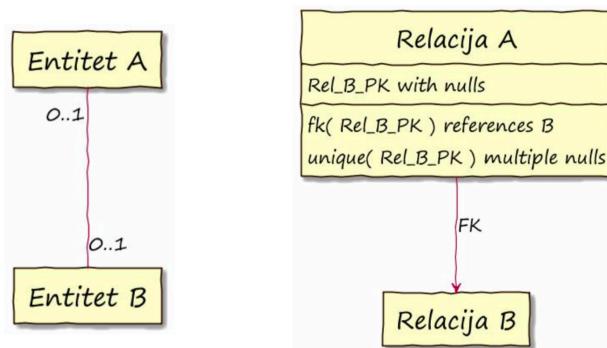
- **1 - 0..N** - Uobičajeno za odnose roditelj - potomak, celina - deo (ako celina **može** da bude bez delova, ako deo **ne može** da bude bez celine). Često kod kompozicija. Relaciji koja je deo: dodaje se strani ključ u odnosu na celinu, **ne sme** biti nedefinisan.



- **0..1 - 1** - Uobičajeno za odnose nadređeni - podređeni, celina - opcioni deo. Relaciji koja je opcioni deo: dodaje se strani ključ u odnosu na celinu, koji **ne sme** da bude nedefinisan, vrednost ključa mora da bude jedinstvena.



- **0..1 - 0..1** - Uobičajeno za dvosmerne opcione odnose. Jednoj od relacija: dodaje se strani ključ u odnosu na drugu, koji **može** da bude nedefinisan, vrednost ključa mora biti jedinstvena (nedefinisane vrednosti mogu da se ponavljaju zato pišemo multiple nulls)



Ili se pravi nova vezna relacija:

- Sadrži samo strane ključeve u odnosu na obe vezane relacije
- Ključevi ne smeju da budu nedefinisani

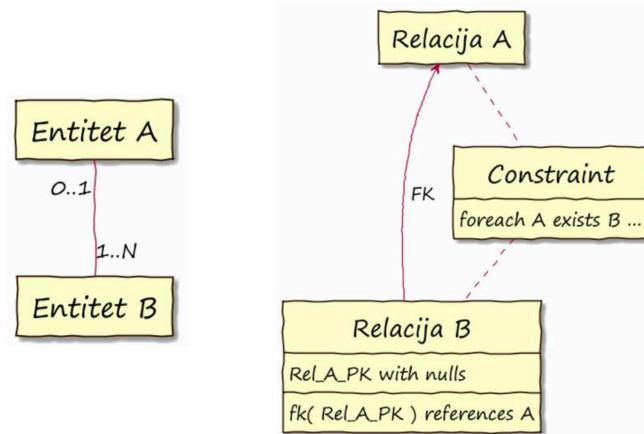
- Svi atributi čine primarni ključ
- Vrednost svakog od ključeva mora biti jedinstvena



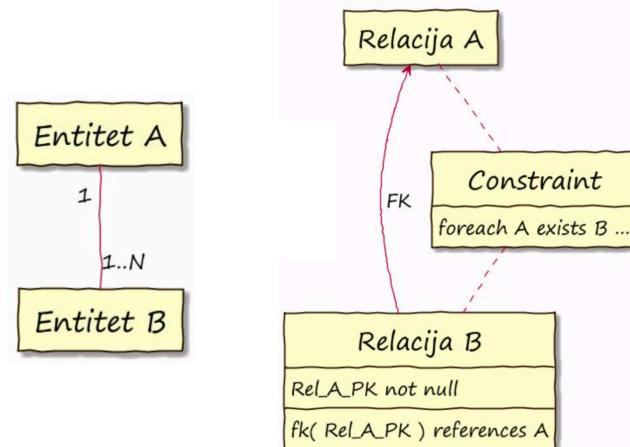
## 100. Kako se u logičkom modelu modeliraju odnosi

**0..1 - 1..N , 1 - 1..N , 1 - 1 ?**

- **0..1 - 1..N** - Uobičajeno za odnose agregacije sa obaveznim delovima. Relaciji koja je deo: dodaje se strani ključ u odnosu na celinu, koji **može** da bude nedefinisan, dodaje se uslov BP «za svaki entitet A mora da postoji bar jedan odgovarajući entitet B»

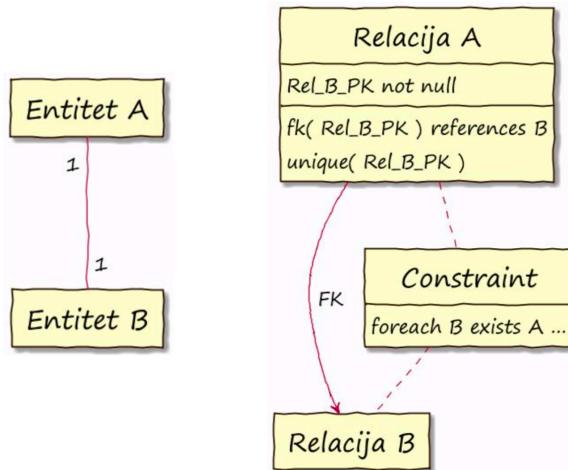


- **1 - 1..N** - Uobičajeno za odnose kompozicije sa obaveznim delovima. Relaciji koja je deo: dodaje se strani ključ u odnosu na celinu, koji **ne sme** da bude nedefinisan, dodaje se uslov BP «za svaki entitet A mora da postoji bar jedan odgovarajući entitet B»

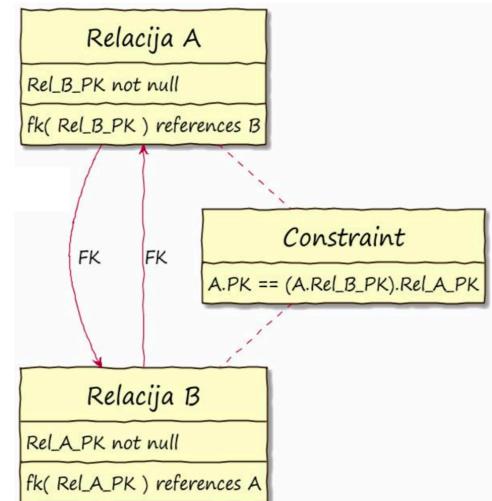


- **1 - 1** - Uobičajeno za odnose obostranog ekskluzivnog pridruživanja. Jednoj od relacija: dodaje se strani ključ u odnosu na drugu, koji **ne sme** da bude nedefinisan, vrednost kluča mora biti

jedinstvena, dodaje se uslov BP «za svaki entitet A mora da postoji bar jedan odgovarajući entitet B»



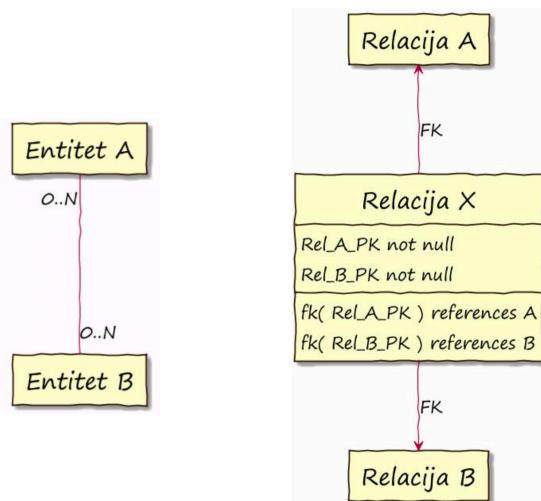
Alternativa, u svakoj od relacija se dodaje strani ključ u odnosu na drugu, atributi ključa ne smeju da budu nedefinisani, dodaje se uslov kojim se proverava uzajamnost veze.



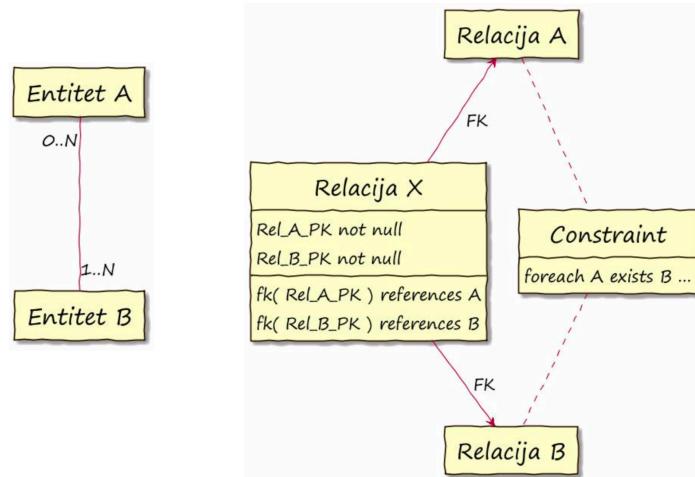
## 101. Kako se u logičkom modelu modeliraju odnosi

### 0..N - 0..N , 0..N - 1..N , 1..N - 1..N ?

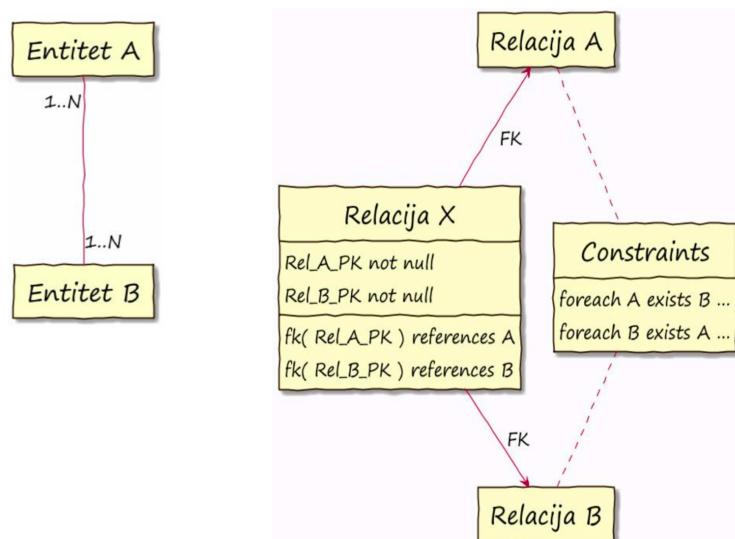
- 0..N - 0..N** - Uobičajeno za odnose asocijacije bez posebnih ograničenja. Pravi se nova vezna relacija: sadrži samo strane ključeve u odnosu na obe vezne relacije, ključevi **ne smeju** da budu nedefinisani, svi atributi čine primarni ključ



- **0..N - 1..N** - Uobičajeno za odnose asocijacija slabih i jakih entiteta, agregacije kod kojih deo može da čini više celina, ali ne može da postoji samostalno. Pravi se nova vezna relacija: sadrži samo strane ključeve u odnosu na obe vezane relacije, ključevi **ne smeju** da budu nedefinisani, svi atributi čine primarni ključ, dodaje se uslov BP «za svaki entitet A mora da postoji bar jedan odgovarajući entitet B»

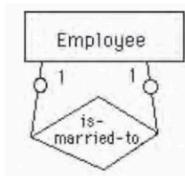


- **1..N - 1..N** - Uobičajeno za odnose agregacije kod kojih deo može da čini više celina, ali ne može da postoji samostalno, celina ne može da postoji bez delova. Pravi se nova vezna relacija: sadrži samo strane ključeve u odnosu na obe vezane relacije, ključevi **ne smeju** da budu nedefinisani, svi atributi čine primarni ključ, dodaje se uslov BP «za svaki entitet A mora da postoji bar jedan odgovarajući entitet B», dodaje se uslov BP «za svaki entitet B mora da postoji bar jedan odgovarajući entitet A»

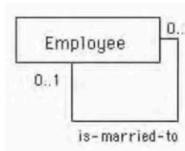
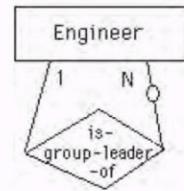


## 102. Kako se u logičkom modelu modeliraju binarni ciklički odnosi?

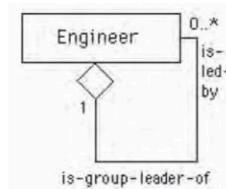
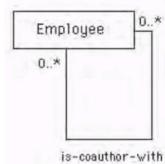
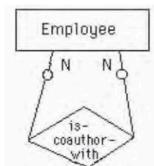
Binarni ciklični odnosi **0..1 - 0..1** - bilo da su opcioni ili ne predstavljaju se dodatnim atributima stranog ključa, ako je opcioni, sme da bude NULL.



Binarni ciklični odnosi **1 - 0..N** - strani ključ se uvodi na strani N.



Binarni ciklični odnosi **0..N - 0..N** - predstavlja se novom relacijom, kao i obični binarni odnosi.



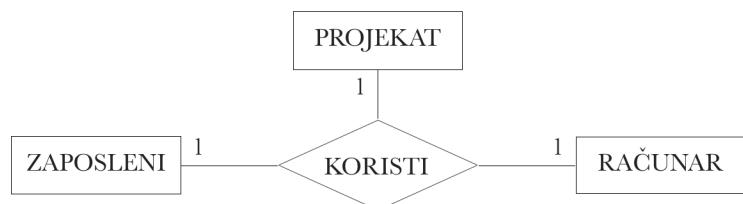
## 103. Kako se u logičkom modelu modeliraju odnosi sa više učesnika?

**1-1-1** - Nova relacija sa stranim ključevima, zavisnosti se uređuju dopuštanjem NULL i jedinstvenim ključevima.

Primer: Tehničar na svakom projektu koristi tačno jedan računar. Svaki računar pripada tačno jednom tehničaru za jedan projekat.

Zavisnosti:

- zaposleni, projekat → računar
- zaposleni, računar → projekat
- projekat, računar → zaposleni

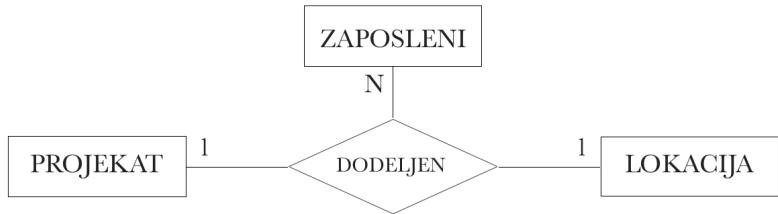


**1-1-N** - Nova relacija sa stranim ključevima, zavisnosti se uređuju dopuštanjem NULL i jedinstvenim ključevima.

Primer: Svaki zaposleni radi na svakom projektu na jednoj lokaciji, ali na raznim projektima možda na više. Na svakoj lokaciji zaposleni radi na tačno jednom projektu. Na svakoj lokaciji više službenika može da radi na svakom od projekata.

Zavisnosti:

- zaposleni, lokacija → projekat
- zaposleni, projekat → lokacija

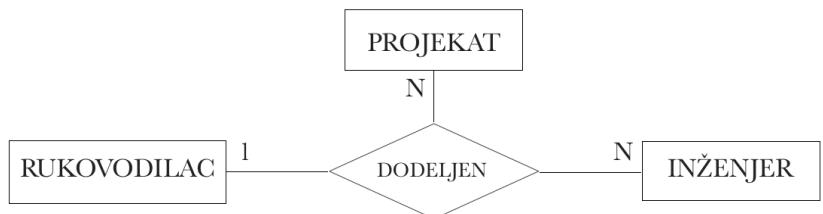


**1-N-N** - Nova relacija sa stranim ključevima, zavisnosti se uređuju izborom primarnog ključa.

Primer: Svaki inženjer na svakom projektu ima tačno jednog rukovodioca. Projekat može da ima više rukovodioca. Rukovodilac može da vodi više projekata. Inženjer može da ima više rukovodilaca na različitim projektima.

Zavisnosti:

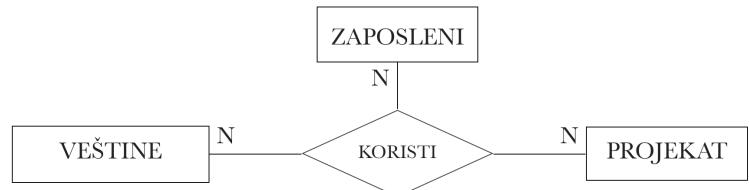
- zaposleni, projekat → rukovodilac



**N-N-N** - Nova relacija sa stranim ključevima

Primer: Zaposleni može da koristi različite veštine na svakom od projekata. Svaki projekat može da ima više zaposlenih sa više različitih ili ponovljenih veština.

Zavisnosti: nema.



#### 104. Na koje se sve načine u logičkom modelu može predstaviti hijerarhijski odnos (generalizacija, specijalizacija...) ?

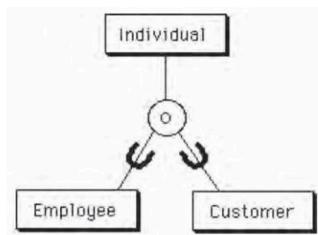
Generalizacija - već smo videli da može da se predstavi na različine načine:

- 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 attribute.

```

create table individual (
    indiv_id char(10),
    indiv_name char(20),
    indiv_addr char(20)
    primary key( indiv_id )
);
create table employee (
    emp_id char(10),
    job_title char(15),
    primary key( emp_id ),
    foreign key( emp_id )
        references individual
        on delete cascade
        on update cascade
);
create table customer (
    cust_no char(10),
    cust_credit char(12),
    primary key( emp_id ),
    foreign key( emp_id )
        references individual
        on delete cascade
        on update cascade
);

```



**Agregacija:** svodi se na obične odnose, obično 1..N

**Odnosi sa više od 3 učesnika:** slično kao odnosi sa 3 učesnika, mora da se vodi računa o funkcionalnim zavisnostima.

**Slabi entiteti:** obično ne zahteva dodatno postupanje, prevođenje odnosa sa jakim entitetima rešava problem.

## 105. Šta je fizički model baze podataka?

Fizički model je najniži model BP.

- Opisuje konkretnu implementaciju
- Uzima u obzir mnoge tehničke aspekte implementacije

Često se naziva modeliranje podataka.

## 106. Šta čini fizički model podataka?

- Strukture podataka - uobičajeno je da se na fizičkom nivou govori o tabelama i kolonama, a ne o relacijama i atributima.
- Interna (fizička) organizacija podataka - prostori za tabelom, stranice, baferi...
- Pomoćne komponente - indeksi

## 107. Kako se procenjuje opterećenje baze podataka? Koji su podaci potrebni?

Procena opterećenja je nešto što se na konceptualnom nivou ne razmatra. Na konceptualnom nivou mi na primer kažemo ovo je student, ovo je ispit. Međutim kada treba da isplaniramo fizičko opterećenje baze mi treba da kažemo da naša baza u ovom trenutku ima na primer 7000 studenata, očekujemo da se svake godine broj studenata povećava za 500 i da se za otprilike 400 smanjuje broj aktivnih studenata. Zanima nas kad najviše dolazimo do opterećenja. Na primer broj studenata u toku godine se ne menja, ali u junu upisujemo 500 novih studenata. Ovo je za nas najveće opterećenje.

Da bismo mogli da procenimo opterećenje i performanse, moramo da imamo:

- Model obrade podataka - sadrže sve aspekte, kada, kako i šta radimo sa podacima, koliko složene operacije...
- Nestrukturne zahteve

- Matricu entiteta i procesa
- Zahtevane performanse
- Prepozname ciljne implementacije SUBP
- Prepozna eventualna ograničenja prostora
- Prepozname eventualne razvojne probleme

**108. Šta obuhvata model obrade podataka, koji se koristi radi procene opterećenja pri pravljenju fizičkog modela?**

- Okolnosti dodavanja novih redova
  - Koliko redova u proseku (na primer dnevno)
  - Koliko redova pri najvećem opterećenju (na primer u sekundi)
  - Da li su primarni ključevi slični i da li zavise od vremena
- Okolnosti ažuriranja postojećih redova
  - Koliko redova u proseku
  - Koliko redova pri najvećem opterećenju
  - Kolika je verovatnoća da se redovi sa sličnim primarnim ključem istovremeno koriste (zbog zaključavanja, jer se zaključavanje vrši po stranicama.. biće reči o ovome kasnije)
- Okolnosti brisanja redova
  - Koliko redova u proseku
  - Koliko redova pri najvećem opterećenju
  - Da li se brišu pojedinačno ili u grupama
- Okolnosti čitanja redova
  - Učestalost čitanja
  - Koliko redova se čita jednim upitom
  - Koje kolone se koriste za odabir redova

**109. Koji nestrukturni zahtevi se razmatraju pri pravljenju fizičkog modela baze podataka?**

- Trajanje podataka - koliko se dugo podaci zadržavaju u tabeli pre brisanja ili arhiviranja
- Obim podataka - koliko će redova biti u tabeli pri puštanju u rad i kako će se broj redova menjati tokom vremena
- Raspoloživost podataka - da li su podaci potrebni stalno ili povremeno, koliko često i dugo podaci mogu da budu nedostupni korisnicima
- Ažurnost podataka - koliko ažurni moraju da budu podaci koji se koriste, da li statistike na primer raditi nad podacima koji su potpuno ažurni
- Bezbednosni zahtevi

**110. Koji su osnovni metodi optimizacije baze podataka? Objasniti ukratko**

- Optimizacija na nivou interne organizacije podataka
  - Ostvaruje se kroz upravljanje internom organizacijom podataka, pomoćnim komponentama i resursima (kako ćemo fizički da smestimo bazu, na koji disk...)

- Fizički model se ne razlikuje u odnosu na logički
  - Fizička organizacija podataka - prostori za tabele, stranice, baferi stranica
  - Upravljanje resursima - pre svega memorija
  - Pomoćne komponente - indeksi
- Optimizacija na nivou upita
  - Vrši se pisanje upita na način koji omogućava njihovo efikasnije izvršavanje - izborom indeksa, nekad samo zamenimo redosled kolona...
  - Fizički model se ne razlikuje u odnosu na logički
- Optimizacija na nivou strukture podataka
  - Fizička struktura podataka se menja u odnosu na logički model (odnosno treba uvideti da pre nego što prebacimo iz logičkog modela u fizički moramo da promenimo logički model kako bi efikasnije bilo u fizičkom)

## **111. Koji su osnovni elementi fizičke organizacije podataka (na primer SUBP DB2)**

Logička organizacija kao osnovno mesto čuvanja podataka. Fizička organizacija ide dalje od toga.

- Prostor za tabele
- Stranice
- Bafer za stranice
- Particionisane tabele
- Kompresija podataka

## **112. Šta je prostor za tabele? Čemu služi?**

Najviši logički nivo organizacije BP i osnovni skladišni prostor se naziva prostor za tabele (engl. *table space*). Sve što pravimo moramo da usmerimo u neki prostor za tabele, postoji podrazumevani. Podrazumevano je da jedna baza podataka ima jedan prostor za tabele (glavni) i jedan privremeni u kojem se čuvaju neki privremeni podaci izračunavanja. Ali fizički ovo nije najniži nivo, već postoji kontejner. Jedan prostor za tabele se može sastojati od više kontejnera.

- Jedan prostor za tabele može da sadrži više tabele.
- U nekim sistemima jedna tabela može da bude u više prostora za tabele (horizontalna fragmentacija)

Na nivou prostora tabela definišu se:

- Veličina fizičke stranice
- Način i uslovi baferisanja stranica
- Fizički uređaji (diskovi, particije, direktorijumi, fajlovi) koji čine taj prostor za tabele (tzv. kontejnери)

Prostor za tabele može sadržati više kontejnera da se ne desi da ponestane mesta, međutim u praksi se ovo ne radi jer želimo da se svi podaci čuvaju sa istom efikasnošću, jer ako imamo različite fizičke uređaje neće biti ista brzina dohvatanja fajlova i ostalih aktivnosti, a to ne želimo.

## **113. Šta je stranica baze podataka? Čemu služi?**

Ceo prostor za tabele se deli na stranice i ta stranica u prostoru je jedinica čitanja i menjanja podataka.

Stranica je osnovni element fizičkog zapisa tabele

- Sve operacije se fizički odvijaju nad stranicama
- Uvek se čita ili piše cela stranica
- Svaka tabela kao i svaki indeks se sastoji od stranica

Veličina stranice je jedna od karakteristika prostora za tabele - sve stranice jednog prostora za tabele su iste veličine!

Ako su stranice velike:

- Mogu da sadrže više redova, odnosno više podataka
- Veća iskorišćenost prostora na stranici
- Manje se traži po disku, jer imamo manje stranica
- Manja dubina indeksa
- Redovi tabele mogu da budu veći

Ako su stranice male:

- Veća je iskorišćenost prostora na disku
- Manje je nepotrebnog čitanja i pisanja
- Veća dubina indeksa

## **114. Šta je bafer za stranice? Čemu služi?**

Bafer za stranice (engl. *buffer pool*) je memorijski prostor predviđen za čuvanje kopija jednog broja stranica radi omogućavanja bržeg pristupa podacima. Određuje se na nivou prostora za tabele.

Sve mora da prođe kroz ovaj bafer (čitanje podataka, privremena izračunavanja...) !!!

Što je bafer za stranice veći, to je broj pristupa disku manji.

- U idealnom slučaju cela baza podataka je u memoriji
- Ako to nije moguće, teži se da u memoriji budu bar indeksi i manje tabele (šifarnici)

Dobro konfiguriranje prostora za tabele i bafera za stranice može da bude od presudnog uticaja na performanse. Ako imamo veliki bafer za stranice naši veliki upiti će raditi efikasno!

## **115. Šta su katanici? Kako se i kada koriste? Šta je eskalacija katanaca?**

Mehanizam katanaca je uobičajeno sredstvo ostvarivanja izolovanosti transakcija.

Svaki katanac ima:

- Objekat koji se zaključava
  - Vrednost (atribut)
  - Red tabele
  - Stranica tabele

- Grupa stranica
- Cela tabela
- Prostor za tabele
- Indeks

Veličina objekta određuje granularnost katanca.

- Trajanje - idealno trajanje svakog katanca je do kraja transakcije
- Vrstu katanca - deljivi i ekskluzivni

Eskalacija katanca:

- Veličina katanca
  - Katanac može da zaključava jedan red ili više redova
  - Ili jednu stranicu ili više stranica
- Veliki broj katanaca može da značajno uspori rad
  - Zbog mnogo više provera pri radu
  - Zato je broj katanaca ograničen (brojem katanaca ili memorijom rezervisanom za katanace)
- Kada se prevaziđe dopušten broj katanaca dolazi do eskalacije katanca
  - Više manjih katanaca se zamenjuje jednim velikim
  - Na primer, više katanaca na redovima se zamenjuje katancem na stranici ili više katanaca na stranicama se zamenjuje katancem na tabeli

## **116. Šta su indeksi? Kada se i kako koriste?**

Indeksi su pomoćne strukture podataka koje omogućavaju brže pristupanje podacima, tj. brže pretraživanje po unapred izabranom ključu. Svaka tabela može da ima više indeksa sa različitim ključevima.

Potrebno je da indeksi budu uređeni:

- Traženje podatka - ako indeksi nisu uređeni onda imamo linearno vreme traženja, ako su uređeni onda logaritamsko

Definicija indeksa obuhvata:

- Kolone koje čine uslov uređivanja, tj ključ pristupanja
- Da li je indeks (tj. odgovarajući ključ) jedinstven ili nije
- Da li je uređujući (grupišući) ili ne
- Vrstu (strukturu) indeksa

## **117. Nавести познате врсте индекса и укратко их објаснити**

Indeksi:

- Jedinstveni - indeksi koji ne dozvoljavaju ponavljanje istog ključa, koriste se kao sredstvo za implementiranje integriteta ključa (jedinstvenosti). Tri vrste:
  - Zabranjene nedefinisane vrednosti
  - Dozvoljene nedefinisane vrednosti, smatramo da su jednake

- Dozvoljene nedefinisane vrednosti, smatramo da su različite
- Grupišući
- Indeksi sa strukturom B-stabla
- Bit-mapirani indeksi
- Heš indeksi

### **118. Šta su grupišući indeksi? Implementacija? Karakteristike? Prednosti i slabosti u odnosu na ne-grupišuće indekse?**

Grupišući indeksi su dodavanje uslova u tabelu, definiše se kriterijum po kome se uređuju podaci u tabeli.

- Podrazumevaju da su redovi u tabeli poređani u odgovarajućem poretku - **najviše jedan** takav indeks po tabeli. Ali može jedan grupišući i neki drugi indeksi.
- Ubrzavaju izdvajanje podsekvence redova u datom poretku - na primer *where ocena between 7 and 10*
- Usporavaju održavanje - ne menja se samo indeks nego i fizički zapis redova
- Zovu se i uređujući
- Kandidati za grupišuće indekse su kolone:
  - Koje se često traže u opsezima
  - Po kojima se često uređuje rezultat
  - Koje pripadaju stranom ključu po kome se najčešće vrši spajanje
  - Kolone primarnog ključa

Prednosti:

- Višestruko ubrzano čitanje nizova redova po uslovu

Slabosti:

- Dodatno usporeno održavanje
- Ne ubrzava pristupanje pojedinačnim redovima

### **119. Šta su indeksi sa strukturom B-stabla? Implementacija? Karakteristike? Prednosti i slabosti?**

B-stabla - balansirana stabla.

Ako koristimo neku strukturu za pristupanje podacima, bez obzira na to koji podatak tražimo, želimo da bude garantovano vreme pristupa. Ako imamo bilo koje drugo drvo, to vreme nije garantovano.

Podaci se čuvaju u balansiranom drvetu

- Svaki podatak je sadržan u listu jednake dubine
- Čvorovi i listovi sadrže po više podataka
- Veličina čvorova odgovara stranici prostora za tabele

Želimo da ovo drvo bude uvek iste dubine. U bilo kom trenutku su listovi ti koji sadrže pokazivače ka podacima, dok čvorovi pre njih služe za organizaciju.

Koristeći B-stabla se radi sa diskom. Jedinica čitanja sa diska je stranica. Što znači da se ne isplati da jedan čvor bude manji od jedne stranice. U veću stranicu staje više pokazivača za naredni nivo, tj. manje čvorova u svakom nivou, što znači manja dubina stabla. Interno su podaci organizovani kao uređena lista. Svaki put se čvor pretražuje serijski. Ako je stranica mala, serijsko pretraživanje čvora je efikasno.

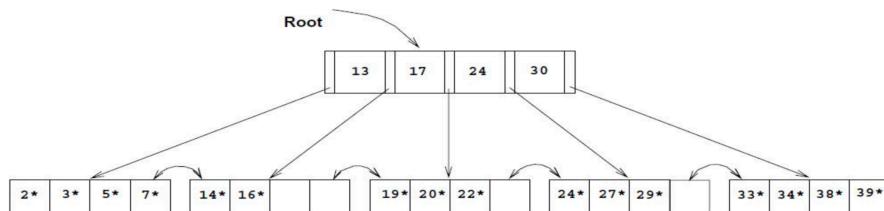
Ako se ne navede drugačije, obično se podrazumeva da se pravi ovakav indeks.

Prednosti:

- Jednostavni i efikasni algoritmi za održavanje

Slabosti:

- Ne radi sasvim dobro ako je mnogo redova a malo različitih vrednosti ključa, tj. ako ima mnogo ponavljanja ključa



## 120. Šta su bit-mapirani indeksi? Implementacija? Karakteristike? Prednosti i slabosti?

Indeks je organizovan kao niz vrednosti ključa.

Uz svaku vrednost ključa sledi niz bitova, za svaki red tabele po jedan

- Vrednost bita je jedan akko odgovarajući red ima baš tu vrednost ključa

Prednosti:

- Ako ima relativno malo različitih vrednosti ključa, onda je ovakva struktura efikasnija od B-stabla
- Efikasno se kombinuje upotreba više indeksa

Slabosti:

- U slučaju mnogo različitih vrednosti ključeva, indeks postaje veoma veliki, a time i slabo efikasan
- Održavanje indeksa je često značajno skuplje nego u slučaju B-stabla

## 121. Šta su heš-indeksi? Implementacija? Karakteristike? Prednosti i slabosti?

Alternativa klasičnim indeksima

- Računaju se heš vrednosti na osnovu ključnih atributa, a onda se pomoću dobijene vrednosti neposredno pristupa podacima. Alternativa je da se indeksira po heš vrednostima.
- Efikasne za pristupanje pojedinačnim redovima sa tačno zadatim ključem
- Nisu dobre za sekvensijalno pristupanje većem broju redova ili ako operator poređenja nije jednakost, ako je opseg u where
  - Na primer ako je  $X < 20$ , onda heš tabela nije korisna

## **122. Kada pravimo indekse, zašto i koliko? Da li uvek moramo da imamo indekse?**

Prednosti indeksa:

- Omogućavaju brži pristup konkretnim redovima u tabeli
- Ako čitanje zahteva samo kolone sadržane u indeksu, onda tabeli ne mora ni da se pristupa (indeksi sa dodatnim kolonama)

Slabosti indeksa:

- Indeksi moraju da se održavaju - svaka operacija dodavanja, ažuriranja, brisanja kolona koje čine ključ indeka, zahtevaju ažuriranje indeksa
- Zauzimaju prostor
- Mogu da povećaju broj zaključavanja ili granularnost katanca
- Podižu cenu reorganizovanja ili prenošenja tabela

Da li neki indeks donosi više koristi ili štete zavisi od toga:

- Da li se indeks uopšte koristi u upitima
- Koliko upotreba indeksa doprinosi performansama upita u kojima se koristi
- Koliko su takvi upiti česti
- Koliko je često i skupo ažuriranje indeksa

Idealan broj i vrsta indeksa zavisi od vrste, namene i strukture tabele i baze podataka. Obično se preporučuje:

- Za transakcione tabele 3-5 indeksa
- Za analitičke tabele bez ograničenja

## **123. Šta je prečišćavanje sheme? Kako se odvija?**

Kada prebacujemo podatke iz konceptualnog u logički model može da se desi da ne uočimo neke redundantnosti ili da smo propustili neku važnu osobinu da prenesemo.

Prečišćavanje sheme: (koraci)

- Analiziranje logičkog modela
- Dosledna primena pravila integriteta - na koji način se funkcionalne zavisnosti odražavaju na pravila integriteta, na primarne ključeve pre svega
- Osnovno sredstvo primene su funkcionalne zavisnosti

**Ključno** - eliminacija redundantnosti

## **124. Objasniti probleme koji nastaju usled redundantnih podataka**

- Redundantno čuvanje podataka - neki podaci se ponovljeno čuvaju

Ne volimo da imamo ponovljene podatke, jer to proizvodi naredne posledice:

- Anomalije ažuriranja - ako se jedna kopija ponovljenog podatka ažurira, baza će biti nekonzistentna ako se ne ažuriraju i ostale kopije

- Anomalije dodavanje - zapisivanje nekih podataka može da bude nemoguće ako se ne zapišu još neki podaci
- Anomalije brisanja - brisanje nekih podataka može da bude nemoguće bez brisanja još nekih podataka

## 125. Uloga nedefinisanih vrednosti u rešavanju problema redundantnosti

Jedan način rešavanja redundantnosti ali ne i najtačniji su nedefinisane vrednosti.

Nedefinisane vrednosti mogu da pomognu u rešavanju nekih anomalija dodavanja i brisanja:

- Ako moramo da dodamo podatak a ne znamo druge potrebne podatke, možemo da navedemo nedefinisane vrednosti
- Ako hoćemo da obrišemo neke redundantne podatke, a da ne brišemo cele redove, možemo da navedemo nedefinisane vrednosti (na primer ako želimo da obrišemo podatak o nastavniku na kursu a ne želimo da obrišemo podatke o studentima koji su upisali kurs)

Ali u praksi ovakav način donosi nove probleme...

## 126. Objasniti postupak dekompozicije kao alat za otklanjanje redundantnosti

Redundantnost se pojavljuje na mestima na kojima postoje ne sasvim prirodne veze između atributa. Dekompozicija je **jedini** tačan način za rešavanje redundantnosti.

**Ideja** je da se jedna relacija sa mnogo atributa zameni sa više relacija sa po manje atributa.

Primer:

UpisanKurs (indeks, ime\_prezime\_studenta, id\_predmeta, naziv\_predmeta, ime\_nastavnika)

Ovde postoji potencijalan problem sa redundantnim podacima o nastavnicima, studentima i predmetima.

Rešenje dekompozicijom:

UpisanKurs(indeks, ime\_prezime\_studenta, id\_predmeta, naziv\_predmeta)

NastavnikKursa(id\_predmeta, ime\_nastavnika)

- Dekompozicija je potpuna ako sadrži dovoljno informacija da se na osnovu dobijenih relacija uvek može rekonstruisati polazna relacija.

## 127. Objasniti funkcionalne zavisnosti

Svaki put kada razmišljamo o dekompoziciji moramo da se pitamo: koji problem želimo rešiti dekompozicijom, da li smo ga rešili i da li proizvodimo nove probleme. Odgovor na ova pitanja daju: analiza funkcionalnih zavisnosti i teorijske osnove normalnih formi relacija.

Funkcionalna zavisnost (FZ) je vrsta uslova integriteta koja uopštava koncept ključa.

- Neka je  $R$  relacija i neka su  $X$  i  $Y$  neprazni skupovi atributa relacije  $R$ . Kažemo da skup torki  $r$  relacije  $R$  zadovoljava funkcionalnu zavisnost  $X \rightarrow Y$  ako za svaki par torki  $t_1, t_2$  iz  $r$  važi:  

$$t_1 . X = t_2 . X \iff t_1 . Y = t_2 . Y$$

Iz prethodnog primera možemo da uočimo sledeće FZ:

indeks → ime\_prezime\_studenta

id\_predmeta → naziv\_predmeta

id\_predmeta →ime\_nastavnika

### **Integritet ključa**

Na osnovu FZ definišemo integritet ključa. Uslovu ključa odgovara FZ oblika:

{atributi ključa} → {svi ostali atributi}

Ovo je problematično ako ključ sadrži atribute koji mogu da imaju nedefinisane vrednosti, jer ne znamo kako se porede.

### **Integritet jedinstvenosti entiteta**

Integritet jedinstvenosti entiteta predstavlja dodatno proširenje i formalizaciju integriteta ključa

- {atributi primarnog ključa} → {svi ostali atributi}
- Atributi primarnog ključa ne smeju da imaju nedefinisane vrednosti

### **Integritet jedinstvenog ključa**

Integritet jedinstvenosti ključa je slabiji oblik jedinstvenosti. Imamo tri oblika:

1. {atributi jedinstvenog ključa} → {svi ostali atributi}
2.
  - 2.1. Atributi jedinstvenog ključa ne smeju da imaju nedefinisane vrednosti (isto kao i jedinstvenost entiteta)
  - 2.2. Ako atributi jedinstvenog ključa imaju nedefinisane vrednosti, smatra se da su one međusobno identične (različiti redovi ne mogu da sadrže iste vrednosti nedefinisanih atributa i kombinacije nedefinisanih atributa)
  - 2.3. Ako atributi jedinstvenog ključa imaju nedefinisane vrednosti , smatra se da su one međusobno različite (različiti redovi mogu da sadrže iste vrednosti definisanih atributa i kombinacije nedefinisanih atributa)

Osnovne osobine (Armstrongove aksiome) su kompletne i zatvorene na skupu svih FZ na nekoj relaciji:

- Refleksivnost:  $X \supseteq Y \implies X \rightarrow Y$
- Proširivost:  $X \rightarrow Y \implies (\forall Z)XZ \rightarrow YZ$
- Tranzitivnost:  $X \rightarrow Y \wedge Y \rightarrow Z \implies X \rightarrow Z$
- Unija:  $X \rightarrow Y \wedge X \rightarrow Z \implies X \rightarrow YZ$
- Dekompozicija:  $X \rightarrow YZ \implies X \rightarrow Y \wedge X \rightarrow Z$

Skup zavisnih atributa  $X^+$  nekog skupa atributa  $X$  je skup svih atributa koji (posredno/neposredno) FZ od atributa iz  $X$ . Na primer, skup atributa  $X$  relacije  $A$  je ključ ako je  $X^+ = A$

## **128. Šta su normalne forme? Objasniti suštinu i navesti najvažnije normalne forme**

Normalne forme su specifični oblici relacija koji zadovoljavaju određena pravila u vezi FZ, koji zato garantuju da neće biti redundantnosti određenog tipa.

Normalne forme:

- 1. Normalna forma (1NF)
- 2. Normalna forma (2NF)
- 3. Normalna forma (3NF)
- Bojs Kodova normalna forma (BCNF)

Svaka sledeća podrazumeva prethodnu.

### **1. Normalna forma**

- Relacija je u 1NF ako svaki atribut može da ima samo atomične vrednosti
- Relacioni model to već prepostavlja
- ER omogućava i složene vrednosti atributa

### **2. Normalna forma**

- Relacija je u 2NF:
  - Ako je u 1NF
  - Ako nijedan atribut, koji ne pripada nijednom kandidat ključu, nije FZ od pravog podskupa atributa nekog kandidat ključa
- Suština je da u slučaju složenog ključa svi ostali atributi zavise od celog ključa a ne od nekog njegovog dela, tj. kandidat ključevi su minimalni, nema tranzitivnih zavisnosti u okviru ključeva

### **3. Normalna forma**

Ako je  $R$  relacija i  $X$  neki podskup njenih atributa i  $A$  neki atribut relacije  $R$ , onda je  $R$  u 3NF ako za svaku FZ  $X \rightarrow A$  na relaciji  $R$  važi jedno od:

- $A \in X$  trivijalna zavisnost
- $X$  je natključ
- $A$  je deo nekog ključa relacije

Moguća narušavanja 3NF:

- $X$  je podskup nekog ključa  $K$ , tj imamo zavisnost od dela ključa (narušena i 2NF)
- $X$  nije ni podskup ni nadskup nijednog ključa, tj imamo tranzitivnu zavisnost

### **BCNF**

Ako je  $R$  relacija i  $X$  neki podskup njenih atributa i  $A$  neki atribut relacije  $R$ , onda je  $R$  u BCNF ako za svaku FZ  $X \rightarrow A$  na relaciji  $R$  važi jedno od:

- $A \in X$  trivijalna zavisnost
- $X$  je natključ

Tj svaki atribut relacije je ili deo ključa ili zavisi od celog ključa, ne postoje tranzitivne zavisnosti.

## **129. Šta je normalizacija? Kada se primenjuje, zašto i kako?**

Normalizacija logičkog modela baze podataka je svodenje na skup relacija koje su sve u BCNF formi (ili eventualno u 3NF)

Efektivan algoritam:

- Neka je  $R$  relacija koja nije u BCNF i neka je  $X$  pravi podskup njenih atributa i  $A$  jedan atribut koji zavisi od  $X$  i tako narušava BCNF. Dekompozicijom je potrebno podeliti  $R$  na relacije sa atributima  $R - A$  i  $XA$
- Ako  $R - A$  ili  $XA$  nisu u BCNF dekomponovati ih dalje rekursivno

Slabost: u nekim slučajevima ne postoji dekompozicija u BCNF koja čuva sve zavisnosti i tada smo prinuđeni ili da ostanemo u nižoj NF ili da izgubimo neke zavisnosti.

## **130. Šta su distribuirane baze podataka i distribuirani SUBP?**

Distribuirano izvršavanje je kada se dva posla izvršavaju u fizički razdvojenim adresnim prostorima (ne dele radnu memoriju).

DBP (*Distribuirana baza podataka*) - je skup više logički međuzavisnih baza podataka koje su distribuirane na računarskoj mreži.

DSUBP (*Distribuirani sistem za upravljanje bazama podataka*) - je softverski sistem koji omogućava upravljanje DBP tako da je distribuiranost transparentna za korisnika.

Distribuirane komponente su na različitim računarima. Sistem je distribuiran ako je lociran na više različitih čvorova u mreži. Prepostavlja se da je mreža jedini deljeni resurs.

## **131. Koji su osnovni doprinosi distribuiranih baza podataka?**

Osnovni doprinosi:

- Transparentno upravljanje distribuiranim i repliciranim podacima
- Pouzdanost distribuiranih transakcija
- Unapređenje performansi
- Lakše proširivanje sistema

## **132. Isto pitanje**

## **133. Objasniti šta znači transparentno upravljanje distribuiranim i repliciranim podacima**

Transparentnost podrazumeva razdvajanje semantike visokog nivoa od problema koji nastaju pri implementaciji na niskom nivou. Nas ne zanima kako su organizovani podaci (gde se čuvaju, da li su distribuirani ili ne...), tj. kako god da su oni organizovani pristup podacima se vrši na univerzalan način.

## **134. Isto pitanje**

### **135. Koji su aspekti transparentnosti upravljanja distribuiranim i repliciranim podacima?**

- Nezavisnost podataka
- Mrežna transparentnost
- Transparentnost replikacije
- Transparentnost fragmentacije

### **136. Objasniti nezavisnost podataka u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima**

Nezavisnost podataka podrazumeva:

- Logička nezavisnost podataka - otpornost aplikacije na promene logičke strukture podataka (aplikacije dobro podnose dodavanje novih elemenata strukturi baze podataka)
- Fizička nezavisnost podataka - potpuno skrivanje fizičke strukture podataka od aplikacije (aplikacije ne sme da trpi nikakve posledice u slučaju promene fizičke strukture podataka, makar ona uključivala i promene u načinu distribuiranja podataka)

### **137. Objasniti mrežnu transparentnost u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima**

Mrežna transparentnost:

- U centralizovanim bazama podataka jedini resurs o kome se stara transparentnost jesu podaci - korisniku nije potrebno da zna kako se upravlja podacima
- U distribuiranim sistemima, mrežna struktura je potrebno da bude sklonjena od očiju korisnika i transparentno upravljana - korisniku nije potrebno da zna gde su podaci

### **138. Objasniti transparentnost replikacije u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima**

Transparentnost replikacije:

- Replikacija je čuvanje istih podataka na više lokacija (više servera) - preduzima se radi performansi, raspoloživosti i pouzdanosti
- Korisnici ne bi trebalo da budu svesni činjenice da se podaci repliciraju - bilo replicirani ili ne oni se moraju koristiti na isti način

### **139. Objasniti transparentnost fragmentacije u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima**

Transparentnost fragmentacije:

- Fragmentacija je čuvanje različitih delova iste kolekcije podataka na više lokacija - preduzima se radi performansi, raspoloživosti i pouzdanosti
- Korisnici ne bi trebalo da budu svesni činjenice da su podaci fragmentisani - bilo da su fragmentisani ili ne, oni se moraju koristiti na isti način (obično se upiti, tzv. globalni upiti, prevode u izvršavanje kolekcije manjih, tzv. lokalnih upita, i rezultati se uniraju).

## **140. Šta i kako može biti nosilac transparentnosti upravljanja distribuiranim i repliciranim podacima?**

Nosilac transparentnosti može da bude:

- Upitni jezik - svaki upit se automatski prevodi na odgovarajući način, u zavisnosti od stvarne organizacije i distribuiranosti podataka (na ovaj način pri pisanju upita moramo reći na kom serveru se nalaze potrebni podaci)
- Operativni sistem - operativni sistem može da se stara o povezivanju različitih modula bez obzira na njihovu fizičku lokaciju
- DSUBP - sva pitanja strukture i distribucije podataka se razrešavaju na nivou SUBP-a, uobičajen pristup

## **141. U čemu se ogleda unapređenje performansi usled distribuiranja?**

Doprinos DBP performansama obično se ogleda u tome što:

- Fragmentisanjem podaci se čuvaju bliže mestu upotrebe
  - Ravnomerno je raspodeljeno opterećenje na više servera na različitim lokacijama
  - Manje se vremena troši na prenos podataka
- Globalni upiti se paralelizuju
  - Podaci su distribuirani, pa se globalni upiti dele na manje, koji se odnose na lokalne podatke

## **142. Koji su osnovni otežavajući faktori pri implementaciji DBP? Objasniti**

Najvažniji otežavajući faktori su:

- Složenost
  - Distribuirani sistemi su sve samo ne jednostavni
  - Distribuirani SUBP mora da reši sve probleme koje rešava centralizovani SUBP i još mnoge druge
- Cena
  - Distribuirani sistemi zahtevaju dodatni hardver (na primer komunikacioni)
  - Softverski aspekti sistema su mnogo složeniji i skupljii za razvijanje
  - Na svakoj lokaciji gde postoje serveri potrebno je i održavanje
- Distribuirana kontrola
  - Otežani su sinhronizacija i koordinacija komponenti
- Bezbednost
  - Kod centralizovanih BP staranje o bezbednosti je centralizovano
  - Ovde je distribuirano i uključuje dodatne aspekte bezbednosti računarskih mreža

## **143. Navesti najvažnije probleme i teme istraživanja u oblasti DSUBP**

Najvažniji problemi koji se rešavaju u DSUBP su:

- Projektovanje distribuiranih baza podataka
- Distribuirano izvršavanje upita
- Distribuirano upravljanje metapodacima
- Distribuirana kontrola konkurentnosti
- Distribuirano upravljanje mrtvim petljama
- Pouzdanost distribuiranih SUBP
- Podrška u operativnim sistemima
- Heterogene baze podataka

#### **144. Šta su heterogene baze podataka?**

DBP često nastaje spajanjem više postojećih rešenja. Tada je obično narušena:

- Homogenost softvera - jer različite baze rade pod različitim SUBP
- Homogenost strukture - jer struktura različitih baza podataka nije usaglašena

I tada je potrebno rešavati neke od opisanih problema:

- Distribuirano izvršavanje upita
- Upravljanje metapodacima
- Upravljanje konkurentnošću

Ovakvi sistemi se nazivaju sistemi sa više baza podataka.

## 145. Objasniti izolovanje neispravnosti i toleranciju neispravnosti

Postoje dva pristupa staranja o neispravnostima:

- Izolovanje neispravnosti - obezbeđivanje da neispravnost jedne komponente ne utiče na funkcionalnost ostalih komponenti (tj. da kvar na jednom čvoru, uređaju ne utiče na rad drugih). Posledica: Funkcija koju je obezbeđivala neispravna komponenta nije raspoloživa dok se problemi ne otklone i komponenta ne postane operativna
- Tolerancija neispravnosti - u slučaju neispravnosti neke od komponenti, druge komponente preuzimaju njene funkcije. Posledica: povećana fleksibilnost i pouzdanost sistema, povećana složenost i cena sistema.

## 146. Objasniti osnovne aspekte pouzdanosti sistema

Pouzdanost sistema čine:

- Raspoloživost sistema
  - Sposobnost sistema da primi zahtev
  - Relativno nisko trajanje perioda kada sistem ne radi
  - Ili retko dolazi do neispravnosti ili se one brzo otklanjaju
- Odgovornost sistema
  - Sposobnost sistema da odgovori na zahtev
  - Sistem ili uopšte neće imati neispravnosti tokom rada ili će one biti prevazilažene u hodu tako da ne bude prekida operativnosti
  - Ili retko dolazi do neispravnosti ili je sistem u potpunosti sposoban da se dovoljno brzo oporavi da one ne ometaju rad

Sistem može biti visoko odgovoran a da nije visoko raspoloživ i obrnuto.

## 147. Objasniti kako se mere osnovni aspekti pouzdanosti sistema?

Postavlja se pitanje kolika je verovatnoća da će sistem primiti/odgovoriti na zahtev?

Mera odgovornosti:

- Sistem je potpuno odgovoran ako nikada ne trpi oštećenja tokom obrade zahteva
- Mera odgovornosti je verovatnoća da će sistem odgovoriti na zahteve koje je primio
- $R(t) = 1 - F(t)$  gde je  $R(t)$  odgovornost u intervalu vremena dužine  $t$ , a  $F(t)$  je verovatnoća da će doći do greške u intervalu dužine  $t$
- $R(t) = 1 - \int_0^t f(x)dx$  gde je  $f(t)$  je gustina verovatnoće uspeha
- Uobičajena mera je srednje vreme do neuspeha u oznaci  $MTTF = R^{-1}(0.5)$  tj vreme za koje je  $R(t) = 0.5$

Primer - Srednje vreme do neuspeha za neki sistem je godinu dana - to znači da u roku od godinu dana verovatnoća da se dogodi neka greška je 50%.

- Koristi se i mera srednje vreme između neuspeha

Mera raspoloživosti:

- Sistem je potpuno raspoloživ ako je uvek u stanju da primi zahtev
- Mera raspoloživosti  $A(t)$  je verovatnoća da će sistem primiti zahtev u nekom trenutku  $t$ , tj. da će sistem ili raditi ispravno u intervalu  $[0, t]$  ili će poslednji problem biti otklonjen u nekom trenutku  $x$ ,  $0 < x < t$
- Ova mera se naziva trenutna raspoloživost
- Uobičajena mera je i srednje vreme opravke u oznaci  $MTTR$  - očekivano trajanje popravljanje sistema nakon neuspeha
- U praksi se obično koristi granična raspoloživost:  $\lim_{t \rightarrow \infty} A(t) = \frac{MTTF}{MTTF + MTTR}$   
(koliko je u proseku sistem raspoloživ)
- Pored toga upotrebljava se i metod eksperimentalnog merenja - ako se u intervalu  $[0, t]$  registruju intervali  $u_i$  u kojima sistem nije raspoloživ, onda je raspoloživost:  $A(t) = 1 - \frac{\sum_i u_i}{t}$

## 148. Šta je replikacija podataka i koje su njene osnovne karakteristike?

Replikacija je svako udvajanje neke komponente računarskog sistema koje donosi neki nivo redundantnosti.

Dva osnovna motiva:

- Podizanje performansi
  - Kroz raspodelu opterećenja na replike
  - Posebno efikasno u slučaju čitanja
  - U slučaju pisanja moguć negativan uticaj
- Podizanje pouzdanosti
  - Implementacija tolerancije neispravnosti - kroz maskiranje neispravnosti i rekonfigurisanje sistema
  - Prepoznavanje neispravnosti - kroz dupliranje (najjednostavniji oblik replikacije)

## 149. Objasniti vrste izvora replikacije

Predmet replikacije mogu biti:

- Podaci
- Procesi
- Objekti
- Poruke

Nas zanima replikacija podataka (baza podataka, tabela, fragment tabele, globalni katalog baze podataka, sistem datoteka, pojedinačna datoteka)

## 150. Koji su osnovni ciljevi replikacije?

Osnovni ciljevi replikacije su: povećanje raspoloživosti i pouzdanosti sistema, poboljšanje performansi (skraćenje vremena odgovora na upit)

## **151. Koja su ograničenja replikacije?**

Osnovnu cenu replikacije predstavlja:

- Dodatna cena prostora zbog redundantnosti čuvanja podataka
- Povećana cena pisanja zbog menjanja podataka na više lokacija
- Povećan obim prenosa podataka
- Cena dodatne obrade usled implementacije replikacije

Ograničenje efikasnosti - ako je ograničena raspoloživost jedne kopije A, onda je i raspoloživost repliciranog sistema ograničena (zavisi od vrste upotrebe)

## **152. Koji su osnovni načini implementacije replikacije podataka?**

Osnovni princip - podaci se održavaju na više (distribuiranih) lokacija.

Osnovni načini replikacije:

- Čitaj jedan piši sve (ROWA)
- Konsenzus kvoruma (KK) (glasanje)
- Konsenzus kvoruma u uređenim mrežama

## **153. Šta je protokol ROWA? Kako se u osnovi implementira?**

Čitaj jedan piši sve (ROWA - Read One Write All)

Koncept:

- Čita se samo primarna kopija - odnosno ako hoćemo da čitamo podatke pristupamo samo jednom serveru
- Sve kopije se menjaju

Čitanje je potencijalno ubrzano ako su podaci na svim serverima isti, brže je onoliko koliko imamo servera. Pisanje je toliko puta sporije. Pitanje je koliko se isplati imati replika?

Šta se dešava ako uređaj otkaže? U tom slučaju na raspolaganju su drugi uređaji sa kojih može da se čita. Ali ne može da se piše na sve čvorove što je neophodno za ovaj protokol.

Protokoli ROWA:

- Tolerišu otkaze lokacija kada je u pitanju čitanje
- Ne tolerišu otkaze lokacija kada je upitanu pisanje
- Ne tolerišu otkaze komunikacija

## **154. Koje su osnovne varijante protokola ROWA?**

Varijante ROWA protokola:

- Osnovni ROWA protokol
- ROWA-A sa menjanjem raspoloživih kopija
- ROWA sa primarnom kopijom

- ROWA sa tokenima pravih kopija

## **155. Objasniti detaljno osnovni protokol ROWA**

Osnovni protokol ROWA:

- Prevodi svaku operaciju čitanja u jednu operaciju čitanja, na jednoj od kopija
- Prevodi svaku operaciju pisanja u N operacija pisanja, po jednu na svakoj kopiji
- Kontroler konkurentnosti na svakoj lokaciji sinhronizuje pristup kopijama - svaka promena mora da uspe na svim lokacijama ili ni na jednoj
- U slučaju otkazivanja neke lokacije - čitanje je i dalje moguće, pisanje nije moguće do oporavka neispravne lokacije

## **156. Objasniti detaljno protokol ROWA-A**

Read One Write All Available

- Razlika u odnosu na osnovni protokol - pisanje se izvodi ne na svim nego na svim raspoloživim kopijama
- U slučaju otkazivanja neke lokacije
  - Čitanje je i dalje moguće
  - Pisanje je i dalje moguće
  - Neispravne lokacije mogu da postanu raspoložive tek nakon oporavka uz puno prepisivanje svih izmena nastalih tokom neoperativnog perioda
- Sinhronizuje neuspehe I oporavke kontrolisanjem raspoloživosti kopija
- Operacije pisanja se šalju svim kopijama
  - ako neka lokacija nije operativna, od nje nema odziva u dopuštenim okvirima
  - ako jeste, onda je operacija pisanja obrađena ili odbačena
- Pre potvrđivanja transakcije koordinator započinje primenu dvofaznog protokola validacije
  - Validacija propuštenih pisanja - proverava da li su sve kopije koje su propustile neko pisanje još uvek neoperativne
    - Koordinator šalje poruku UNAVAIL svim kopijama od kojih još nije dobijen odgovor na zahtev za pisanje. Ako je neka postala aktivna i pokrenula pisanje, ona će primiti ovaj zahtev i odgovoriti signalom za prekid transakcije. Ako nije nijedna postala aktivna, nastavlja se sa sledećim korakom
  - Validacija pristupa - proverava se da li su sve kopije koje su čitale ili pisale još uvek raspoložive
    - Koordinator takvim kopijama šalje poruku AVAIL. Svaka koja je aktivna prima poruku i potvrđuje da je aktivna. Ako neka nije aktivna, prekida se transakcija.
- Problemi:
  - Svakoj neispravnoj lokaciji se šalju bar dva zahteva (1. Operacija pisanja, 2. Provera u toku validacije) i na svaki se čeka najviše dopušteno vreme - time se potencijalno gubi značajno vreme i smanjuje odzivnost sistema
  - Iako bezbedan, postupak dvofazne validacije nije efikasan - poželjno je prepoznati oporavak lokacije u što kraćem intervalu, ovaj postupak ignoriše oporavak u nekim slučajevima i odlaže ga sa kasnije, čime mu se potencijalno podiže cena

## **157. Koje su prednosti protokola ROWA-A u odnosu na osnovni protokol ROWA? Ograničenja?**

Prednost protokola ROWA-A je što je u slučaju otkaza pisanje i dalje moguće, što nije slučaj kod osnovnog protokola.

Ograničenja - pitanje iznad.

## **158. Objasniti detaljno protokol ROWA sa primarnom kopijom**

Pogledati pitanje 161.

- Ako primarna kopija postane neoperativna, izabrana rezervna kopija postaje nova primarna
  - Da bi ovo bilo moguće potrebno je da bude moguće nedvosmisleno razlikovati neoperativnost primarne kopije od problema u komunikaciji
  - U suprotnom se može dogoditi da postoji više primarnih kopija, što nije dobro
- Ako rezervna kopija postane neoperativna
  - Ona ne može postati primarna sve dok ne bude u potpunosti oporavljena
  - Može preuzeti neke uloge rezervne i nakon delimičnog oporavka

## **159. Ponovljeno pitanje 157**

## **160. Ponovljeno pitanje 158**

## **161. Koje su prednosti protokola ROWA sa primarnom kopijom u odnosu na osnovni protokol ROWA?**

Razlika u odnosu na osnovni protokol:

- Jedna kopija se proglašava za primarnu
- Ostale kopije su rezervne
- Čitanje se izvodi samo za primarne kopije - tj svi zahtevi čitanja idu na tu jednu primarnu kopiju
- Pisanje se izvodi na primarnoj i svim raspoloživim rezervnim kopijama

Ovaj protokol ne podiže performanse čitanja, unapređuje samo pouzdanost sistema.

Najčešće upotrebljiv jer je jednostavan.

## **162. Objasniti detaljno protokol ROWA sa tokenima pravih kopija**

- Razlika u odnosu na osnovni protokol:
  - Svaki podatak, u bilo kom trenutku vremena, ima ili jedan ekskluzivan token ili skup deljivih tokena
  - Operacija pisanja mora da dobije ekskluzivan token
  - Operacija čitanja mora da dobije bar deljni token
- Konceptualno slično distribuiranom zaključavanju
- Ako postoji deljni token na podatku to znači da je podatak AŽURAN
- Uzimanje tokena obuhvata, po potrebi, i ažuriranje podataka

- Pri pisanju - kada kopija mora da izvrši operaciju pisanja, ona locira i uzima ekskluzivan token za konkretni podatak. Ako on ne postoji, ona locira i proglašava za neispravne sve deljive tokene za konkretni podatak i pravi novi ekskluzivan umesto njih.
- Pri čitanju - kada kopija mora da izvrši operaciju čitanja, kopija locira deljivi token (locira neku drugu kopiju koja ima deljivi token, odnosno na kojoj su podaci ažurni), kopira njegovu vrednost i pravi i čuva novi deljivi token. Ako ne postoji deljivi token, ona locira ekskluzivan token i konvertuje ga u deljivi i zatim postupa kao u prethodnom slučaju.
- Token ima objedinjenu semantiku nalik na lokalni katanac na distribuiranom podatku
- Predstavlja mehanizam za prepoznavanje konflikata između više pisanja ili između pisanja i čitanja podataka
- U slučaju problema - samo lokacije koje imaju token na nekom podatku mogu koristiti taj podatak. Ako su svi tokeni na podatku locirani na neoperativnoj lokaciji, taj podatak nije raspoloživ.
- Nakon pisanja promena bi trebalo da se distribuira, što se može ostvariti različitim mehanizmima (na primer, asinhroni proces obilazi sve ekskluzivne tokene, konverte je ih u deljive i kopira izmenjene podatke na određen broj drugih lokacija, uz pravljenje deljivih tokena)
- Prednosti i mane:
  - Pisanje nije usporeno jer je lokalno, ne pišemo na svim kopijama
  - Čitanje se usporava, jer moramo da tražimo podatke, prepisujemo

### **163. Koji od protokola ROWA se najčešće upotrebljava u praksi? Zašto?**

Iako je ROWA-A algoritam relativno jednostavan i ima nekih slabosti, ipak je jedan od najčešće upotrebljavanih, naravno, samo u slučajevima gde je jedini cilj podizanje nivoa pouzdanosti.

### **164. Šta je konsenzus kvoruma? Po čemu se razlikuje od protokola ROWA?**

Konsenzus kvoruma - grupa algoritama, pokušavaju da se reše problemi osnovnog protokola ROWA (ne može da se piše u slučaju otkaza)

Za razliku od protokola ROWA:

- Dopušta pisanje na podskupu (kvorum pisanja) operativnih lokacija
- Čitanja se izvode sa podskupa (kvorum čitanja) lokacija koji mora da ima neprazan presek sa kvorom pisanja

Pravilo preseka kvoruma obezbeđuje da će svako čitanje moći da se odvija sa najsvežije pisanim vrednostima. Za lokaciju koja učestvuje u operaciji se kaže da je glasala za operaciju.

Jednostavnije rečeno - svi čvorovi se dele u dva skupa: iz jednog vršimo čitanje iz drugog pisanje. Bitno je da uvek postoji presek između ova dva skupa. Kad čitamo, traži se podatak sa svih čvorova u konsenzusu i traži se najsvežiji.

Pri oporavku čvorovi ne moraju automatski da se ažuriraju, već mogu postepeno. Ako se često dešavaju neispravnosti ili sistem ima mnogo čvorova onda je ovaj algoritam bolji od ROWA, ali su performanse niže. Performanse pisanja su više jer ne pišemo na sve čvorove, ali pri čitanju moramo da čitamo sa svih čvorova i da proveravamo najsvežiji.

## **165. Opisati opšte karakteristike konsenzusa kvoruma**

Ova tehnika maskira neispravnosti, bez dodatnih zahteva pri oporavku neispravnih lokacija, efikasnije nego u slučaju validacije u dva koraka.

Kvorumi (tj. oni skupovi) mogu biti:

- Statički - određeni glasanjem pri podizanju sistema
- Dinamički - ako lokacije mogu da se rekonfigurišu

## **166. Koje su vrste konsenzusa kvoruma? U čemu je osnovna razlika među vrstama?**

Vrste konsenzusa kvoruma:

- KK sa uniformnom većinom - svaki čvor vredi jedan glas, ako hoćemo da pišemo moramo da pišemo na više od pola čvorova, ako hoćemo da čitamo moramo da čitamo sa više od pola čvorova
- KK sa težinskom većinom
- KK sa težinskom većinom za direktorijume
- Uopšteni KK za apstraktne tipove podataka
- Hibrid ROWA / KK

## **167. Objasniti detaljno konsenzus kvoruma sa uniformnom većinom**

- Najstariji od metoda kvoruma
- Operacija čitanja ili pisanja uspeva ako većina lokacija odobri izvršavanje
- Ne moraju sve lokacije koje su glasale i da izvrše operaciju na svojim kopijama podataka
  - Čitanje je dovoljno da se izvrši na jednoj kopiji - iako se priprema i provera konkurentnosti odvijaju na svim kopijama koje su glasale
  - Pisanje se mora izvršiti na većini kopija
- Počiva na prepoznavanju eventualnih konflikata od strane upravljača konkurentnošću na pojedinačnim lokacijama
- Postiže se otpornost na neispravnosti - na lokacijama i u komunikaciji
- Cena - relativno visoka i pri čitanju i pri pisanju, bar polovina + 1 lokacija mora učestvovati u svakoj operaciji kroz glasanje

## **168. Objasniti detaljno konsenzus kvoruma sa težinskom većinom**

Predstavlja uopštenje algoritma KK sa uniformnom većinom:

- Svaka kopija  $d_s$  podatka  $d$  dobija ne-negativnu težinu tako da suma svih težina na jednom podatku bude  $u$  (Na primer, imamo na raspolaganju 50 računara koji čine sistem, neki su jači, neki su slabiji. Ideja je da oni koji mogu da izdrže veće performanse dobijaju veću težinu. Onda se piše na onoliko čvorova tako da ukupna težina bude neka zadata vrednost. Češće pišemo na one čvorove koji su efikasniji i često čitamo sa onih čvorova koji su efikasniji.)
- Sam podatak  $d$  dobija prag čitanja  $r$  i prag pisanja  $w$ , tako da važi:
  - $r + w > u$  (ekvivalentno nepraznom preseku kvoruma)
  - $w > \frac{u}{2}$  (ekvivalentno većini, neophodan ako se najsvežija kopija ustanavljava na osnovu broja verzije)

- Kvorum čitanja (ili pisanja) podatka  $d$  je svaki skup kopija čija je težina bar  $r$  (ili  $w$ )
- Za ustanovljavanje aktuelnosti kopija se obično upotrebljava mehanizam verzija podataka. Svaka kopija podatka je označena brojem verzije, koji se inicijalno postavlja na 0.
- Načelno se ne zahteva složen algoritam oporavka:
  - Kopija koja je propustila neko ažuriranje jednostavno neće imati poslednju verziju podatka, pa ona neće biti korišćena bar do prvog narednog ažuriranja
  - Ipak u slučaju otkaza se može dogoditi da većina nema ažurnu vrednost, pa da čitanje bude neispravno
- Algoritam je fleksibilan
  - Menjanjem parametara  $r$  i  $w$  i dodeljivanjem težina pojedinačnim kopijama pojedinačnih podataka može se upravljati pravljenjem kvoruma i prilagođavati različitim pouzdanostima komponenti
  - U slučaju ekstremnih vrednosti parametara  $r$  i  $w$  algoritam se svodi na ROWA (sve težine iste,  $r = 1, w = N$ )
  - U slučaju ujednačenih vrednosti parametara  $r$  i  $w$  algoritam se svodi na KK sa uniformnom većinom (ako su sve težine = 1)
- Izbor težina i pragova može biti složen posao, alternativa je automatsko određivanje težina
  - Jeden način je postupkom kaljenja

## **169. Objasniti razlike između konsenzusa kvoruma sa uniformnom većinom i konsenzusa kvoruma sa težinskom većinom**

## **170. Koje su specifičnosti konsenzusa kvoruma za direktorijume i apstraktne tipove podataka?**

Uopštenje za direktorijume

- Direktorijum je preslikavanje prostora ključeva u prostor vrednosti
- Primjenjuje se i na nerelacione baze zasnovane na katalozima

Postoje sledeće operacije na direktorijumima:

- *Insert (Key k, Value v)* - dodeljivanje vrednosti  $v$  ključu  $k$ , samo ako ključ  $k$  ne postoji u direktorijumu
- *Update (Key k, Value v)* - dodeljivanje nove vrednosti  $v$  ključu  $k$ , samo ako ključ  $k$  postoji u direktorijumu
- *Delete (Key k)* - brisanje ključa  $k$  iz direktorijuma, samo ako ključ  $k$  postoji u direktorijumu
- *ValueLookup (Key k)* - čitanje vrednosti dodeljene ključu  $k$  (ili *false* ukoliko ne postoji)

Menja se granularnost operacija tako da se ne izvršavaju na velikim direktorijumima nego na njihovim manjim delovima.

Ideja je da svi ključevi koji na primer počinju sa 1 predstavljaju jedan skup podataka, oni koji počinju sa 2 predstavljaju drugi. Za te razlike skupove mogu se napraviti različiti kvorumi.

## **171. Objasniti detaljno hibridni protokol ROWA / konsenzus kvoruma. Objasniti motivaciju za njegovu primenu**

Osnovna slabost metoda KK je neopravdano visoka cena u slučaju retkih otkazivanja komunikacija. Hibridni pristup redukuje cenu tako što koristi ROWA sistem u normalnom režimu (dok se ne dogodi otkaz), a kad se dogodi otkaz onda se prelazi u režim kvoruma. Vraćanje iz režima kvoruma nije jednostavno, mora da izvršimo kompletno ažuriranje podataka.

Postojanje otkaza se prepozna kroz propuštena pisanja:

- Ako transakcija prepozna da je bilo izostajanja pisanja neke kopije podatka koju je već čitala, ona mora da bude prekinuta
- Ako transakcija prepozna da je bilo izostajanja pisanja neke kopije pre čitanja sa te kopije, onda mora da pređe u režim KK i da čita sa kopija koje nisu propustile pisanje

Ako transakcija  $T$  uputi zahtev za pisanje i dođe do izostajanja pisanja na nekoj kopiji:

- Samo transakcija  $T$  to lako uočava i menja režim
- Da bi druge transakcije to doznale potrebni su dodatni mehanizmi - jedan način je vođenje liste propuštenih pisanja uz svaku kopiju podatka

## 172. Šta je konsenzus kvoruma u uređenim mrežama? Motivacija?

Ono što je problem sa konsenzusom kvoruma je što se zahteva čitanje sa velikog broja čvorova i pisanje na veliki broj čvorova. Da li taj broj čvorova može da se redukuje broj lokacija (čvorova) koje odlučuju u glasanju? Može sa uvođenjem logičke strukture u skup lokacija, tj. čvorovi više neće biti ravnopravni, već uređeni na odgovarajući način.

## 173. Koji su osnovni algoritmi koji se upotrebljavaju u konsenzusu kvoruma u uređenim mrežama?

Vrste:

- Algoritam  $\sqrt{n}$
- Protokol matrice GRID
- Asimptotska visoka raspoloživost
- Drvo kvoruma
- KK sa hijerarhijskom težinskom većinom
- KK sa višedimenzionalnom težinskom većinom

## 174. Objasniti algoritam $\sqrt{n}$ . Koje su mu osnovne karakteristike? Zašto se tako zove?

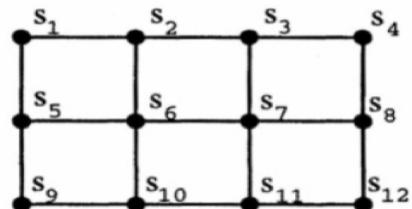
Algoritam se zove  $\sqrt{n}$  jer se u osnovi svodi na to da veličina kvoruma može biti reda  $\sqrt{n}$ .

- Uvodi se pretpostavka da neće biti otkazivanja komunikacije, ali može da se desi otkaz čvora i u tom slučaju sa tim čvorom niko nije povezan
- Svakoj lokaciji se dodeljuje jedan kvorum koji ima neprazan presek sa svim kvorumima koji su dodeljeni drugim lokacijama
  - Svaka dva kvoruma imaju neprazan presek
- Transakcija mora da obezbedi konsenzus svih lokacija u kvorumu dodeljenom njegovoj matričnoj lokaciji
- Kada element kvoruma  $s_i$  zatraži međusobno isključivanje, protokol traži konsenzus od kvoruma lokacija  $Q_i$  tako da važi:
  - $Q_i \cap Q_j \neq \emptyset$  - za svaka dva kvoruma mora postojati bar jedna zajednička lokacija, koja služi kao arbitar kada se neki članovi dvaju kvoruma ne slažu
  - $s_i \in Q_j$  - element kvoruma  $Q_i$
  - $|Q_i| = K$  - svaka lokacija šalje i prima isti broj poruka

- $|\{Q_i : s_i \in Q_i\}| = M$  - svaka lokacija je arbitar za isti broj lokacija, tj. učestvuje u istom broju kvoruma
- Prethodni uslovi impliciraju:  $n = K(K - 1) + 1$ 
  - Sa  $K$  razmena poruka se može dogovoriti  $n$  lokacija
- Protokol postiže uzajamno isključivanje sa  $c\sqrt{n}$  poruka gde je  $3 \leq c \leq 5$

## 175. Objasniti protokol GRID. Koje su mu osnovne karakteristike?

Idea je da sve čvorove vežemo u jednu mrežu, odnosno matricu. I onda se kvorumi definišu geometrijski u odnosu na tu matricu.



Skupovi lokacija se uređuju u obliku matrice sa  $N$  kolona i  $M$  vrsta, tako da  $M * N \leq n$ .

- Jedan kvorum čitanja obuhvata lokacije sa po tačno jednom lokacijom iz svake kolone
- Kvorum pisanja se sastoji od svih lokacija jednog kvoruma čitanja i jedne kolone
- Cilj je ravnomerno raspoređivanje opterećenja

Tj. kad nešto pišemo, pišemo u celoj jednoj koloni i jos na neku drugu, a kad nešto čitamo, čitamo iz svake kolone nešto.

- Skup lokacija  $G$  je  $C$ -pokrivanje ako svaka kolona ima neprazan presek sa  $G$
- Operacija čitanja bira proizvolju permutaciju  $\pi_r$  koja se sastoji od  $M$  lokacija u proizvoljnem redu  $r$ 
  - $\pi_r$  određuje redosled komuniciranja radi postavljanja katanaca za čitanje na  $C$ -pokrivaču - obezbeđuje ravnomerno opterećenje
  - Ako uspe, garantuje se da jedna od kopija ima najsvežiju vrednost
  - Ako ne uspe, pokušava se slično na narednoj vrsti dok se ne dobije katanac
  - Ako ne uspe ni na jednoj vrsti, operacija se prekida i katanci se oslobođaju
- Operacija pisanja
  - Najpre zaključava  $C$ -pokrivač koristeći protokol čitanja
  - Zatim zaključava sve lokacije proizvoljne kolone  $c$  u redosledu određenom permutacijom  $\pi_c$
  - Da bi neka sva pisanja radila konkurentno, svako od njih mora da zaključa po  $C$ -pokrivač i jednu kolonu, pa se mora prepoznati konflikt

## 176. Navesti i ukratko objasniti najvažnije ciljeve pri pravljenju distribuiranih sistema

Pri pravljenju distribuiranih sistema (ne samo baza podataka) kao najvažniji ciljevi se ističu tri važna svojstva - CAP

- Konzistentnost (consistency)
- Raspoloživost (availability)
- Prihvatanje razdvojenosti (partition tolerance)

## **177. Objasniti konzistentnost kao jedan od osnovnih ciljeva pri pravljenju distribuiranih sistema**

- Konzistentan sistem funkcioniše kao celina ili ne funkcioniše uopšte.
- Sva čitanja, na svim čvorovima, moraju da daju isti rezultat.
- Rezultat operacije (čitanja) nikada ne zavisi od čvora na kome se izvršava.
- Razlikuje se od istog termina u kontekstu osobina transakcija ACID

## **178. Objasniti raspoloživost kao jedan od osnovnih ciljeva pri pravljenju distribuiranih sistema**

- Sistem je uvek raspoloživ - raspoloživost se praktično definiše kao odzivnost sistema u nekim garantovanim granicama
- Sistem obično nije raspoloživ upravo kada je potreban
  - U vreme kada je raspoloživost najpotrebnija, onda je i najteže ostvariva, zato što je tada sistem najviše opterećen
  - Ako je sistem raspoloživ kada nije potreban, to nema značaja

## **179. Objasniti prihvatanje razdvojenosti kao jedan od osnovnih ciljeva pri pravljenju distribuiranih sistema**

- Nijedan skup problema, osim potpunog otkazivanja, ne sme da proizvede neispravan odziv sistema
  - Tj. sistem mora da prihvata delimične otkaze komunikacije i da nastavlja ispravno funkcionisanje
- Povremeni prekidi komunikacije među čvorovima su neizbežni
- Razdvojenost (partition) je stanje komunikacione mreže u kome su delovi sistema (obično usled kvara) podeljeni na particije (dva ili više razdvojenih skupova čvorova) između kojih ne postoji komunikacija

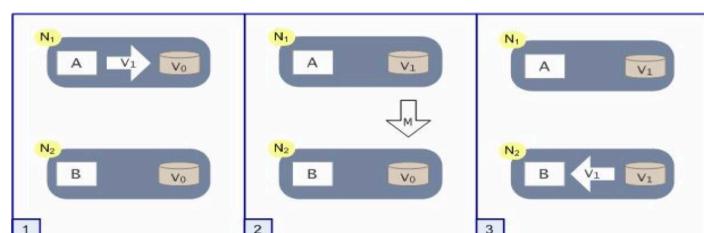
## **180. Navesti teoremu CAP i ukratko je objasniti**

Eric Brewer (2000)

- «Nije moguće definisati sistem koji zadovoljava sve CAP uslove»
- Moguće je definisati sistem koji zadovoljava izabrat dva od ovih uslova

Prepostavke:

- Neka imamo dva čvora i na njima repliciran podatak  $V$ 
  1. Neka na čvoru 1 transakcija  $A$  ažurira  $V$
  2. Promena se porukom  $M$  propagira na čvor 2
  3. Neka na čvoru 2 nešto kasnije transakcija  $B$  čita  $V$



Prepostavimo da poruka  $M$  nije stigla na odredište?

- Zbog razdvojenosti delova sistema

U osnovi imamo tri moguća ponašanja sistema:

- Transakcija  $A$  se poništava - znači da sistem ne prihvata razdvojenost svojih delova
- Transakcija  $A$  se uspešno nastavlja (i zatim završava) iako poruka nije stigla na odredište - znači da sistem nije konzistentan
- Transakcija  $A$  čeka na uspešno izvršavanje slanje poruke  $M$  - znači da sistem nije raspoloživ (nepoznato vreme odziva)

### **181. Koji vidovi kompromisa se prave radi prevazilaženja ograničenja koja proističu iz teoreme CAP?**

Pri projektovanju distribuiranog sistema neophodno je napraviti neki kompromis:

- Odbacivanje tolerancije razdvojenosti
- Odbacivanje raspoloživosti
- Odbacivanje konzistentnosti
- Ublažavanje uslova (tj. odbacivanje više strogo definisanih uslova)
- Zasnivanje sistema na drugačijem skupu uslova
- Projektovanje zaobilaznih puteva

### **182. Objasniti odbacivanje prihvatanja razdvojenosti kao posledicu teoreme CAP**

Odbacivanje tolerancije razdvojenosti:

- Pristajemo da sistem ne radi u slučaju razdvojenosti
- Jedan način prevazilaženja je da sve bude na jednoj mašini
- Alternativa je da se razdvojenost (a time i otkaz sistema) svede na najmanju moguću meru pomoću višestrukog umrežavanja
- Obe alternative su veoma skupe

Imajući u vidu da se distribuirana rešenja koriste samo onda kada su neophodna zbog obima podataka i poslova, ovaj vid kompromisa se retko pravi

### **183. Objasniti odbacivanje raspoloživosti kao posledicu teoreme CAP**

Odbacivanje raspoloživosti:

- U slučaju razdvojenosti ne garantuje se vreme odziva
- Problem se redukuje pažljivim projektovanjem sistema, tj. uspostavljanjem što niže sprege među čvorovima

Sistem koji nije raspoloživ je praktično neupotrebljiv:

- Zbog toga se ovaj pristup koristi relativno retko
  - Ako je konzistentnost primarna
  - Ako je tolerancija razdvojenosti nezaobilazna
  - Teži se niskoj spregnutosti među čvorovima

## **184. Objasniti odbacivanje konzistentnosti kao posledicu teoreme CAP**

Odbacivanje konzistentnosti:

- Dopuštamo da isti upit daje različite rezultate na različitim čvorovima
- Ako su razlike prihvatljivije nego niska raspoloživost
- Ako je učestalost pojavljivanja svedena na prihvatljivu meru
- tj. ako je cena nekonzistentnosti prihvatljiva

Konzistentnost je jedan od osnovnih uslova za uspešan rad baza podataka. Ipak postoje slučajevi kada nije primarna: na primer, veb pretraživači (ukoliko tražimo neku infomaciju sa laptopa i računara, nije nam mnogo bitno da li daju iste rezultate za isti upit)

## **185. Na čemu počivaju alternativni skupovi uslova za projektovanje distribuiranih sistema, koji se uvode radi prevazilaženja posledica teoreme CAP?**

Pojmovi iz teoreme počivaju na uobičajenim konceptima rada sa bazama podataka, odnosno postoji zavisnost sa osobinama transakcija - ACID

Može se definisati I drugačiji skup uslova, manje oštar, u kom slučaju se sve odgovarajuće osobine mogu uskladiti - BASE umesto ACID.

## **186. Navesti I objasniti izmenjen skup uslova integriteta baze podataka - BASE**

- **Basically Available**
  - Ne garantuje se raspoloživost odgovora, već samo sistema
  - Ako ne može da se dobije odgovor, bar će se dobiti obaveštenje o tome
  - I pored toga, visoka raspoloživost na štetu preostalih uslova
- **Soft-state**
  - Stanje sistema može da se menja čak i kada nije u toku nijedna transakcija, na primer radi ostvarivanja konzistentnosti
- **Eventually consistent**
  - Žrtvuju se garantovana stalna konzistentnost i izolovanost transakcija zarad raspoloživosti
  - Sistem će u nekom trenutku postati konzistentan
  - Ali će raditi i davati odgovore i do tada

Suština koncepta je u odloženom usklađivanju sadržaja na različitim čvorovima

- Konzistentnost se ostvaruje, ali ne obavezno u okviru iste transakcije
- Sadržaj čvorova se menja i van odvijanja transakcija, a u cilju njihovog usklađivanja
- Ako sistem nije u potpunosti raspoloživ, čvor može da pokuša da pruži manje pouzdan ili samo delimičan odgovor, uz odgovarajuću informaciju o tome

## **187. Objasniti specifičnosti projektovanja baze podataka u odnosu na uslove BASE**

U osnovi se svode na dve novi aktivnosti:

- Funkcionalna dekompozicija podataka u fragmente - ako prepostavljamo da se već projektuje distribuirana baza podataka, ovde samo malo preciznije određujemo uslove za definisanje fragmenata
- Implementacija transakcija prema *BASE* uslovima - operacije se ne menjaju, ali se menja vreme njihovog izvršavanja, a time i način implementiranja
- Određivanje uslova replikacije

Dekompozicija - skup podataka se deli na fragmente, koji predstavljaju najmanje moguće funkcionalne celine, unutar kojih moraju da važe *ACID* uslovi, a među kojima je dovoljno da važe *BASE* uslovi.

Podela transakcija na sinhroni I asinhroni deo:

- Svaka transakcija se locira u jednom matičnom fragmentu
- Promene podataka u matičnom fragmentu se implementiraju sinhrono, odnosno na uobičajen način u okviru transakcije
- Promene podataka u drugim fragmentima se u okviru transakcije samo evidentiraju
- Evidentirane potrebne izmene se u drugim fragmentima sprovode asinhrono

Određivanje uslova replikacije:

- Pretpostavlja se da među replikama važe *BASE* uslovi
- Sinhronizacija replika se odvija asinhrono, van transakcija

Primer:

Transakcija ažuriranja cene proizvoda u režimu ACID:  
BEGIN TRANSACTION  
UPDATE Product SET Price = :new\_price  
WHERE ProductID = :product\_id  
UPDATE CartItem SET Price = :new\_price  
WHERE ProductID = :product\_id  
AND CartID in (SELECT CartID FROM Cart WHERE Status = 'OPEN')  
END TRANSACTION

Ista transakcija u režimu BASE:

BEGIN TRANSACTION  
UPDATE Product SET Price = :new\_price  
WHERE ProductID = :product\_id  
QUEUE ADD  
UPDATE CartItem SET Price = :new\_price  
WHERE ProductID = :product\_id

```
        AND CartID in (SELECT CartID FROM Cart WHERE Status = 'OPEN')
END TRANSACTION
```

## 188. Objasniti pojam konflikata pri projektovanju baze podataka na osnovu uslova **BASE** i načine njihovog razrešavanja

Postoje različite vrste konflikata pri implementaciji sistema sa BASE skupom uslova.

Osnovni oblik - ako se dve replike istog podatka nezavisno menjaju, pitanje je koja je verzija ispravna i kako uspešno izvesti usklađivanje

Složeniji oblik - ako se koriste i redundantni podaci a ne samo replike

Načini razrešavanja:

- Zabrana menjanja podataka u slučaju particonisanja - umanjena raspoloživost
- Definisanje hijerarhija među redundantnim kopijama konkretnih vrsta podataka
  - Definisanje primarnih i sekundarnih fragmenata za sve podatke
    - Samo na primarnim se izvode transakcije u odnosu na te podatke
    - Na sekundarnim se samo propagiraju odložene izmene
  - Ne umanjuje upotrebljivost, ali komplikuje implementaciju
- Višestruke verzije podataka (MVCC)
  - Alternativa zaključavanju
  - Za svaku kopiju podatka se vodi verzija
  - Povezano sa konceptima optimističkih transakcija i optimističke replikacije
  - Konflikti mogu automatski da se prepoznaju, ali ne i da se otkloni

## 189. Šta su nerelacione baze podataka?

Relacionim bazama podataka su prethodile baze podataka zasnovane na drugim modelima podataka: hijerarhijske, mrežne... Ali se danas termin nerelacione baze podataka ne odnosi na njih.

U savremenom razvoju softvera termin nerelacione baze podataka se odnosi na sisteme za upravljanje kolekcijama podataka:

- Nemaju strogu statičku strukturu
- Nemaju iscrpnu proveru uslova integriteta
- Ne koriste upitni jezik SQL
- (nekada se podrazumeva i distribuiranost)

Nema jedinstvene definicije, ali može se reći NRBP je BP koja ne počiva na relacionom modelu podataka, lako se distribuirira i horizontalno je skalabilna.

Druge česte karakteristike:

- Bez statičke sheme
- Laka replikacija
- Jednostavan API
- Eventualna konzistentnost počiva na skupu uslova **BASE** a ne **ACID**
- Prepostavlja izuzetno velike količine podataka

## **190. Objasniti motivaciju za razvijanje i korišćenje nerelacionih baza podataka**

Zbog slabosti RBP koje su navedene u sledećem pitanju razmatraju se efikasna nerelaciona rešenja:

- Ako je potrebno da se ostvari
  - Jednostavnije i efikasnije distribuiranje podataka
  - Lako dodavanje čvorova
  - Visoka raspoloživost
  - Slobodna (ili bar fleksibilnija) struktura podataka
- Ako je prihvatljivo da se žrtvuje
  - Određen nivo konzistentnosti
  - Automatska provera integriteta

## **191. Koje osnovne slabosti RBP pokušavaju da se prevaziđu nerelacionim bazama podataka?**

Relacione baze imaju mnoge kvalitete ali ti kvaliteti imaju cenu:

- Relativno visoka cena čitanja - zbog redundantnosti i stroge strukture podataka, zbog čestog spajanja podataka
- Otežano distribuiranje - pre svega zbog ultimativne konzistentnosti podataka, opterećenje raste eksponencijalno, nije jednostavno dodavati i sklanjati čvorove
- Skupa promena strukture - zbog povezanosti strukture sa upotrebom i optimizacijama

## **192. Navesti osnovne vrste nerelacionih baza podataka i tipične probleme koji se njima rešavaju**

Osnovne vrste nerelacionih baza podataka:

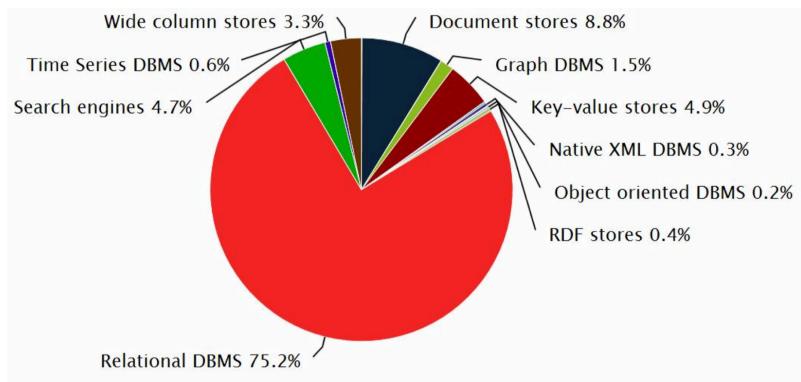
- Parovi ključeva i vrednosti
- Skladišta širokih kolona
- Skladišta dokumenata
- Grafovske baze podataka
- Objektne baze podataka
- Tabelarne baze podataka
- Skladište torki
- Viševrednosne
- Multimodelne
- XML baze podataka
- Skladišta sadržaja (dokumenata, resursa...)
- Sistemi za pretraživanja
- Baze vremenskih serija

Tipični problemi i alternativna rešenja:

- Složene strukture podataka u specifičnim domenima
  - Objektne baze podataka
- Visok nivo distribuiranja
  - Različite nerelacione baze podataka
- Slobodna ili fleksibilna struktura podataka
  - Različite nerelacione baze podataka
- Neophodne izuzetno visoke performanse
  - Memorijске baze podataka
- Ogomorne količine podataka niske složenosti
  - Različite vrste matričnih baza podataka

### 193. Navesti najčešće modele podataka nerelacionih baza podataka

- Baze parova ključeva i vrednosti
- Baze sa proširivim sloganima
- Baze dokumenata
- Grafovske baze podataka



### 194. Baze parova ključeva i vrednosti - karakteristike, doprinosi, slabosti i primjeri

Imamo samo ključ i vrednost.

Struktura:

- Svaka kolekcija podataka je jedna heš tabela
- Podacima se pristupa isključivo na osnovu poznatog ključa ili sekvencialno
- Vrednosti su u načelu jednostavne ili vrlo nisko strukturirane - ako vrednosti imaju visoku složenost, obično se BP klasificiše kao baza sa proširivim sloganima ili baza dokumenata

Doprinosi:

- Podržavaju veoma velike skupove podataka
- Veoma su brze
- Obično podržavaju automatsko repliciranje i horizontalno particionisanje kolekcija

Slabosti:

- Podrazumeva se visok nivo redundantnosti jer ne postoje primarni ključevi, strani ključevi, povezanost

- Složene strukture se implementiraju velikim brojem kolekcija
- Ako su podaci gusto povezani, efikasnost drastično opada
- U osnovi nemaju mehanizme za očuvanje integriteta - često ne obezbeđuju čak ni atomičnost transakcija
- Ne mogu da se pretražuju po podacima - svaki indeks je nova kolekcija podataka
- Uslov traženja je isključivo fiksna vrednost ključa ili raspon vrednosti ključa

Primeri: Redis, Memcached, Hayelcast, Riak, Oracle NoSQL

## **195. Baze sa proširivim slogovima - karakteristike, doprinosi, slabosti i primeri**

Struktura:

- U osnovi slično kao baze parova ključeva i vrednosti
- Vrednost predstavlja kolekciju velikog broja parova imena i vrednosti
  - Obično broj parova bar načelno nije ograničen
  - Sve skupa izgleda kao baza parova ključeva i vrednosti sa dve dimenzije

Doprinosi:

- Podržavaju veoma velike skupove podataka
- Veoma su brze - osim u nekim slučajevima vrednosti sa veoma složenom strukturom
- Većina podržava automatsko repliciranje i horizontalno particonisanje kolekcija

Slabosti:

- Sve kao za baze parova ključeva i vrednosti

Primeri: Cassandra, BigTable, Druid, Accumulo, HDFS (Hadoop Distributed File System), HBase

## **196. Baze dokumenata - karakteristike, doprinosi, slabosti i primeri**

Struktura:

- U osnovi liči na bazu parova ključeva i vrednosti
- Svaki dokument predstavlja vrednost kome se dodeljuje ključ
- Dokumenti se zapisuju u strukturiranim (XML, YAML, JSON, BSON...) ili nestrukturiranim oblicima (npr. PDF)
- Svaki dokument ima metapodatke - obično nestrukturirane ili sa vrlo fleksibilnom strukturom
- Telo dokumenta (ako je strukturiran) i metapodaci se automatski interno strukturiraju

Doprinosi:

- Obično veoma jednostavan i efikasan za rad
- Obično podržavaju bar poluautomatsko repliciranje i horizontalno particonisanje kolekcija - često samo replikacija tipa glavni-podređeni

Slabosti:

- Relativno ograničen domen primene
- Mnoge implementacije ne omogućavaju ad-hoc upite i menjanja podataka
- Neke implementacije ne trpe veliku učestalost menjanja podataka

Primeri: MongoDB, CouchDB, MarkLogic, RavenDB, Amazon DynamoDB, Google Cloud Datastore

## 197. Grafovske baze podataka - karakteristike, doprinosi, slabosti i primeri

Struktura:

- Bazu čine čvorovi i veze među njima
- Nema čvrste sheme, podaci imaju veoma slobodnu strukturu
- Akcenat je na odnosima između podataka, a ne na strukturi podataka

Doprinosi:

- Nasuprot drugim nerelacionim BP, veoma su efikasne kada su u pitanju uobičajene operacije sa grafovima
- Neke podržavaju transakcije i *ACID* režim uslova
- Obično samo replikacija tipa glavni-podređeni

Slabosti:

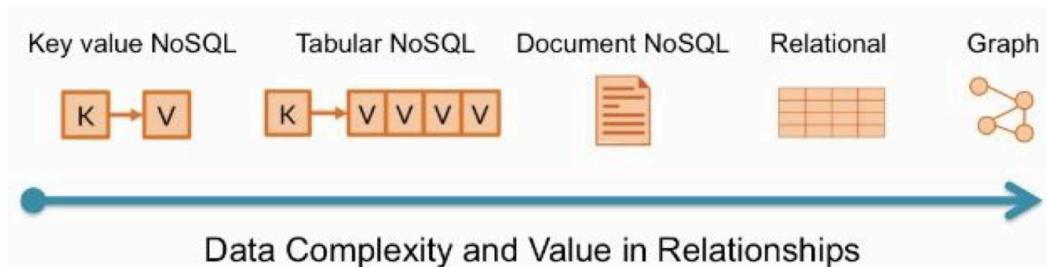
- Relativno ograničen domen primene
- Nisu pogodne van svog domena

Primeri: Neo4J, OrientDB, ArangoDB, Allegro, Virtuoso, MS Azure Cosmos DB, Titan, GraphDB

## 198. Osnovne slabosti nerelacionih baza podataka

Slabosti NRBPs:

- Nemaju strogu statičku strukturu
- Nemaju iscrpnju proveru uslova integriteta
- Ne koriste upitni jezik SQL
- Složenost strukture podataka



## 199. Nerelaciona baza podataka Apache Cassandra - osnovne karakteristike

- Nerelacioni SUBP
- Primarno namenjen za DBP
- Dinamična shema
- Inicijalno razvijen od strane Facebook-a, projekat otvorenog koda, jedan od glavnih projekata fondacije Apache
- Među najrasprostranjenijim NRSUBP-ovima
- Upitni jezik CQL3 - Cassandra Query Language
- Počiva na proširenom modelu kataloga - tj. baza parova ključeva i vrednosti. Osnovni pojmovi su:
  - Kolona
  - Familijska kolona
  - Superkolona
  - Familijska superkolona
  - Ključ
  - Prostor ključeva

Ovi pojmovi su drugačiji nego kod RBP!

**Kolona** - jedna vrednost, praćena vremenom poslednje izmene. Pojam kolone je blizak pojmu attribute kod RBP.

- Trojka (ime, vrednost, vreme izmene)

```
{  
    name: « prezime »  
    value: « Petrović »  
    timestamp: 12321432  
}
```

- Pojednostavljen zapis

```
    prezime: « Petrović »
```

**Superkolona** - složena vrednost, bez vremena poslednje izmene. Sadrži jednu ili više kolona. Ne postoji ekvivalent kod RBP, nešto kao složeni atribut.

```
{ name: « licniPodaci »  
    value: {  
        ime: { name: « ime », value: « Goran », timestamp: 1232454 } ,  
        prezime: { name: « prezime », value: « Petrović », timestamp: 2343242 } ,  
        grad: { name: « grad », value: « Smederevo », timestamp: 12313424 }  
    }  
}
```

**Familijska kolona** - struktura koja može da sadrži veći broj redova (konceptualno neograničeno). Preslikava ključ u red:

- Red = skup kolona
- Red je sličan superkoloni

Pojam familije kolona je blizak pojmu tabele kod RBP. Konceptualno predstavlja katalog redova.

Autori : {

```
ivoAndric: {  
    // pojednostavljen zapis, bez naziva i TS  
    ime: «Ivo» ,  
    prezime: «Andrić» ,  
    ...  
},  
goranPetrovic: {  
    ime: «Goran» ,  
    prezime: «Petrović» ,  
    ...  
}, ...  
}
```

**Prostor ključeva** - struktura koja sadrži više familija kolona ili superkolona. Konceptualno odgovara pojmu baze podataka ili sheme kod RBP.

Cassandra ima dva nivoa ugnezdenosti:

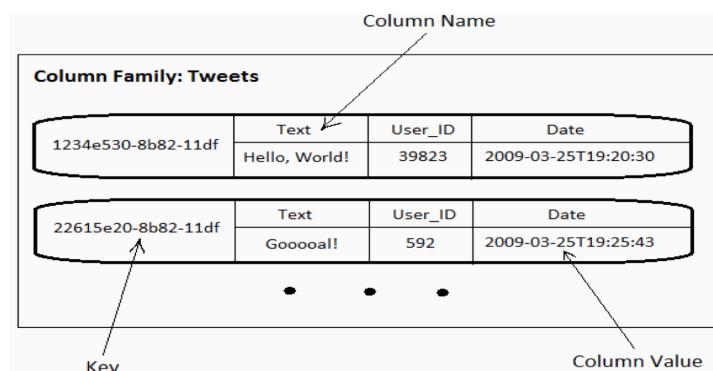
- Prvi nivo je obavezan
  - Čini ga par (ime kolone, kolona)
  - Jeden red (slog) se sastoji od proizvoljnog broja parova
  - Parovi su uređeni samo po imenu kolone
  - Red ima oblik kataloga
  - Red mora da ima bar jednu kolonu
- Drugi nivo je opcion
  - Umesto da drugi element para bude kolona, to može biti kolekcija parova - superkolona

Imena kolona predstavljaju istovremeno: ključeve i vrednosti

Svaki red ili superkolona može da sadrži proizvoljno mnogo kolona - nazivi kolona mogu da imaju ulogu vrednosti, vrednosti kolona mogu biti prazne.

Primer: tvitovi

Svaka poruka je poseban red, ključ reda je kodirano vreme pisanja poruke, kolone odgovaraju atributima



**Particionisanje** - svaka familija kolona je horizontalno partitionisana. Partitionisanje se vrši seckanjem (heširanje) po ključu, na odgovarajući broj particija.

### **Strategije replikacije**

- SimpleStrategy - podrazumevana strategija, parametrom replikacije se određuje na koliko čvorova želimo da se ponovi svaka replika
- NetworkTopologyStrategy - naprednija strategija, parametrom replikacije se određuje na koliko čvorova u svakom centru podataka želimo da se ponovi svaka replika

**Tipovi podataka:** tekstualni (ascii, inet, text...) , numerički (bigint, decimal, float, double, int... ) , logički (boolean), kolekcije (list, map, set, tuple), univerzalni jedinstveni identifikatori...

Ali ne može list<list<...>>

## **200. Šta je plan izvršavanja upita? Na osnovu čega se pravi?**

U savremenim SUBP upiti se izvršavaju u nekoliko koraka:

- Pravi se jedan ili više planova izvršavanja upita sa procenjenim troškovima izvršavanja
- Bira se najefikasniji plan
- Izvršava se izabrani plan

Da bi se neki upit izvršio nad bazom podataka, najpre se pravi plan izvršavanja upita koji obuhvata:

- Redosled koraka
- Operacije koje se izvršavaju u pojedinim koracima
- Strukture podataka koje se upotrebljavaju
- Način svakog pojedinačnog pristupanja podacima
- Procenjenu cenu svakog od koraka i celog posla

## **201. Šta je optimizacija upita? Automatska i manuelna optimizacija**

Optimizacija upita je manuelno ili automatsko odabiranje najpovoljnijeg plana izvršavanja upita radi postizanja boljih performansi. Većina savremenih SUBP raspolaže solidnim optimizatorima upita.

Manuelna optimizacija upita je ranije bila mnogo potrebnija nego danas - projektant ili administrator može na različite načine da sugerise SUBP-u koji plan izvršavanja da primeni. Danas je uloga manuelne optimizacije upita važnija u fazi pravljenja fizičkog modela neko u eksploataciji - ne optimizuje se upit, nego se analiziraju planovi izvršavanja radi sagledavanja i prevazilaženja eventualnih slabosti predloženog modela.

## **202. Koji faktori utiču na rezultat optimizacije upita?**

Pri pronalaženju najboljeg plana izvršavanja uzimaju se u obzir raspoloživi podaci o bazi podataka:

- Struktura tabela i ključeva
- Struktura upita
- Postojeći indeksi

- Statistički podaci o sadržaju tabela i atributa
- Podaci o brzini fizičkih uređaja
- Veličina bafera stranica

Da bi procena troškova izvršavanja bila što tačnija, moraju da postoje isrcpne i ažurne statistike o bazi podataka. Statistike moraju da se prave na realnom sadržaju baze podataka. Detaljnost i preciznost može da se bira.

Statistike o tabelama mogu da se računaju:

- Za sve redove ili samo za izabrane redove - redovi se mogu birati na različite načine, uz određivanje stepenom zastupljenosti i načinom izabiranja uzorka
- Za svaku od kolona ili samo za izabrane kolone - kolone primarnog ključa, stranih ključeva, manuelno izabrane
- Samo opseg vrednosti ili detaljna raspodela - distribucija vrednosti (frekvencija, kvantili...)
- Automatski ili manuelno

### **203. Postupak optimizacije upita**

### **204. Na kojim nivoima se izvodi optimizacija baze podataka**

- Optimizacija na nivou interne organizacije podataka
  - Ne utiče se na logički model podataka, ne menja se skup tabela i kolona
  - Ostvaruje se kroz upravljanje internom organizacijom podataka, pomoćnim komponentama i resursima
- Optimizacija na nivou upita
  - Vrši se pisanje upita na način koji omogućava njihovo efikasnije izvršavanje
  - Ne menja se logički model
- Optimizacija na nivou strukture podataka
  - Fizička struktura podataka se menja u odnosu na logički model

### **205. Objasniti optimizaciju baze podataka na nivou strukture podataka**

Ako imamo uzorak radnog opterećenja i odgovarajuće planove izvršavanja, onda možemo da sprovedemo optimizaciju strukture baze podataka u cilju povećanja performansi datog radnog opterećenja.

- Optimizacija fizičkog dizajna menjanjem fizičke strukture baze podataka

Uzorak radnog opterećenja je skup tipičnih upita (i promena baze podataka) za koje se zna (ili se procenjuje) da će činiti najčešći oblik pristupanja bazi podataka.

Dobar uzorak se sastoji od :

- Skupa upita i promena baze podataka
- Za svaki upit i promenu postoji i procenjena učestalost izvršavanja i posebno vršno opterećenje formulisanih ciljnih performansi

Na osnovu uzorka:

- Za upite

- Prepoznaje se koje relacije (tabele, indeksi) koriste
- Koji atributi se izdvajaju
- Koji atributi učestvuju u uslovima spajanja i restrikcije
- Za promene baze, pored toga:
  - Vrsta promene
  - Atributi koji se menjaju (za ažuriranje)

Cilj strukturne optimizacije:

- Prepoznavanje indeksa koje moramo da napravimo
- Prepoznavanje potrebnih promena u fizičkoj shemi
  - Alternativna normalizacija
  - Denormalizacija
  - Uvođenje pogleda da bi se sakrile načinjene promene - ako se promeni fizička struktura baze podataka, onda na konceptualnom nivou mogu da se uvedu novi pogledi koji skrivaju načinjene izmene

## **206. Šta je alternativna normalizacija? Kada se i zašto koristi?**

Pri normalizovanju ne postoji samo jedno moguće rešenje:

- Normalizacija do 3NF ili BCNF
- Normalizacija do 3NF različitim putevima

Primer: Neka relacija P opisuje projekte

$P(\text{BrProjekta}, \text{Grad}, \text{Naziv})$

Imamo sledeće FZ:

fd1:  $\text{BrProjekta} \rightarrow \text{Grad}$

fd2:  $\text{BrProjekta} \rightarrow \text{Naziv}$

Ako u jednom gradu postoji najviše jedan projekat i svi projekti imaju različite nazive, onda možemo da smatramo da postoje i

fd3:  $\text{Grad} \rightarrow \text{Naziv}$

fd4:  $\text{Naziv} \rightarrow \text{Grad}$

fd5:  $\text{Naziv} \rightarrow \text{BrProjekta}$

Odatle sledi da imamo alternativne normalizacije:

- Normalizacija 1

$P(\text{BrProjekta}, \text{Grad})$

$\text{PN}(\text{Grad}, \text{Naziv})$

- Normalizacija 2

$P(\text{BrProjekta}, \text{Naziv})$

$\text{PG}(\text{Naziv}, \text{Grad})$

## **207. Uloga denormalizacije u optimizaciji baze podataka**

Optimizacija strukture baze podataka često predstavlja vid denormalizacije.

Neke tehnike:

- Podela tabele (dekompozicija)
- Spajanje tabele (kompozicija)
- Dupliranje podataka

## **208. Dekompozicija tabela u optimizaciji baze podataka**

Ako imamo tabelu sa mnogo redova ili kolona, koja se često menja, onda promene mogu da budu neefikasne:

- Indeksi postaju duboki i često se čita sa diska
- Upiti koji ne koriste indekse su tim pre neefikasni

Jedan način da se rad ubrza je da se tabela podeli na dve ili više tabela.

Postoje dve vrste podele:

- Horizontalna
  - Redovi tabele se podele u dve tabele sa istom strukturom - obično jedna sadrži tzv. nove ili aktivne podatke, a druga stare ili arhivske podatke
  - Prednost je što ubrzavaju neke operacije (pre svega dodavanje i menjanje novih podataka)
  - Mana je što se dodatno komplikuju neke operacije (pre svega analitički upiti nad svim podacima)
  - Particionisanje tabela je vid horizontalne podele
- Vertikalna
  - Ako se većina kolona ne koristi u svim upitim, već samo relativno retko, onda se česte operacije mogu ubrzati vertikalnom podelom
  - Prave se dve ili više tabela:
    - Svaka sadrži kolone primarnog ključa
    - Sve ostale kolone se grupišu prema upotrebi
  - Prednost je što se ubrzavaju neke česte operacije
  - Mana je što se dodatno komplikuju neke operacije (pre svega analitički upiti nad svim podacima)

## **209. Spajanje tabela u optimizaciji baze podataka**

Tabele koje se često (ili čak svaki put) spajaju u upitim, može da bude korisno spojiti u jednu tabelu.

- Prednost je što je spajanje veoma skupa operacija i ovakva promena može da ima značajne efekte
- Mane:
  - Dobijena tabela obično narušava normalne forme i sadrži neke redundantnosti
  - To dodatno otežava održavanje podataka i staranje o integritetu
  - Može da se značajno poveća broj redova ili ukupno zauzeće prostora

## **210. Optimizacija hijerarhija entiteta**

Hijerarhije entiteta smo pominjali u modelu entiteta i odnosa. Hijerarhije postoje i u objektnom modelu i dijagramu klasa podataka.

Pri pravljenju logičkog modela hijerarhija može da se prevede u relacije (tabele) na bar tri osnovna načina:

- Jedna integralna tabela
- Za svaki tip zasebna tabela, koja ima samo kolone ključa i kolone novih atributa
- Za svaki neapstraktan tip zasebna tabela, koja ima sve potrebne kolone
- Različite kombinacije

Čest je problem sa hijerarhijama:

- Pri pravljenju logičkog modela se razmatraju kriterijumi logičkih veza među podacima
- Pri pravljenju fizičkog modela zbog efikasnosti može da bude bolje da se primeni drugo rešenje

Stabilno rešenje za hijerhije:

- Da se sve operacije čitanja odvijaju kroz poglede koji zaokružuju sve potrebne podatke o elementima hijerahije
- Da se zabrane direktne operacije menjanja podataka u tabelama koje pripadaju hijerarhiji
- Da se sva pisanja izvode kroz okidače na pogledima ili predefinisane procedure