Projektovanje baza podataka

- odgovori na ispitna pitanja -

Mina Milošević
mi17081@alas.matf.bg.ac.rs

2020/2021

Sadržaj

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 je nalik na slogove u programskim jezicima. Slog sadrži podatke jedne instance entiteta i sastoji se od polja. Strukture se povezuju "vezama" (link) - nalik na pokazivače u programskim jezicima, slično povezivanju elemenata dijagrama strelicama.

3. Objasniti osnovne koncepte hijerarhijskog modela podataka.

Skup “slogova” povezanih “vezama” tako da grade “hijerarhiju”. U osnovi slično mrežnom modelu, ali se zahteva hijerarhija. Kod mrežnog sloja je hijrarhija bila implicirana smerom veza, a ovde je eksplicitna 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 nivoe apstrakcije kod savremenih baza podataka.

Kod savremenih baza podataka razlikujemo 4 osnovna nivoa apstrakcije podataka:

- *Spoljašnji nivo* - nivo korisnika (šta korisnik vidi)
- *Konceptualni (logički) nivo* - sve što čini logički model podataka
- *Nivo fizičkih podataka* - fizička organizacija podataka u softverskom sistemu
- *Nivo fizičkih uređaja* - fizička organizacija podataka na fizičkim uređajima

5. Objasniti uglove posmatranja arhitekture baze podataka.

Iz ugla komponenti sistema - komponente sistema se definišu zajedno sa njihovim međusobnim odnosima. 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 sistema - 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 nedovoljnom uvidu u način ostvarivanja funkcija i ciljeva.
Iz ugla podataka sistema - prepoznaju se različite vrste podataka. Arhitektura se određuje tako da definiše funkcionalne jedinice koje koriste podatke na različite načine. Kako su podaci centralna tema SUBP-a, ovaj pristup se često preporučuje kao 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.

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 predložen 1975. godine, ali nikada nije formalno usvojen. Načelno počiva na podacima ali je ustvari objedinjen pogled na arhitekturu baza podataka. Prepoznaju se 3 nivoa (pogleda) podataka:

- *Spoljašnja shema* - najviši nivo apstrakcije. Odvojeno i od implementacije i od ograničenja modela podataka. Predstavlja pogled na bazu podataka iz ugla korisnika. Često se naziva i nivo pogleda ili nivo korisnika.
- *Konceptualna shema* - nivo umerene apstrakcije (odvojen od fizičke implementacije). Opisuje zaista implementiran skup podataka i odnosa među podacima. Često se naziva i logički nivo ili nivo administratora.
- *Interna shema* - najniži nivo apstrakcije. Opisuje elemente fizičke implementacije. Često se naziva i nivo sistema.

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:

- 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
- Eksterni nivo čine pogledi koji pružaju denormalizovanu sliku dela domena

8. Objasniti primer arhitekture klijent-server.

Osnovna ideja je razdvojiti funkcionalnosti servera i klijenta. Server pruža usluge, a klijent koristi te usluge. Server je zadužen za upravljanje podacima - obrada upita, optimizacija, izvođenje transakcija. Klijent (klijentski deo SUBP) je zadužen za - 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.
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). Ova arhitektura je prirodno proširenje ANSI/SPARC modela.

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.

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

Strukturni deo – način modeliranja podataka. Osnovna ideja je da se *sve* modelira relacijama. *Sve* znači i entiteti i odnosi.

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

Manipulativni deo – način rukovanja modeliranim podacima. Ključno mesto u manipulativnom delu modela ima pojam upita. Upit je definicija nove relacije na osnovu već poznatih relacija baze podataka. Pored upita, važno mesto zauzima ažuriranje baze podataka.

13. Šta je „integritetni deo relacionog modela“. Objasniti ukratko.

Integritetni deo – način obezbeđivanja valjanosti podataka. Ovaj model čine koncepti i mehanizmi koji omogućavaju da se automatizuje proveravanje zadovoljenosti određenih uslova. Stanje baze je konzistentno (tj. ispravno) ako sadržaj baze podataka ispunjava sve uslove integriteta. Promena sadržaja baze podataka je dopušteno akko prevodi bazu iz jednog u drugo ispravno stanje. Baza podataka se opisuje relacijama. Uslovi integriteta u relacionom modelu se opisuju predikatima nad relacijom ili bazom podataka. Formalnost modela olakšava formulisanje uslova integriteta.

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

Neka imamo crvenu, plavu, belu i zelenu kutiju, i u njima lopte, kocke i pločice. Binarna relacija *kutijaSadrži(K, P)* je zadovoljena ako kutija K sadrži P. Njen domen je:

Dom(kutijaSadrži) = {crvena, plava, bela, zelena} x {lopte, kocke, pločice}

Neka naredni model relacije opisuje šta se nalazi u kojoj kutiji:

kutijaSadrži = {(plava, lopte), (plava, kocke), (zelena, pločice), (crvena, kocke)}

Iskaz *kutijaSadrži*(plava, kocke) je tačan. Iskaz *kutijaSadrži*(zelena, lopte) nije tačan. Iskaz *kutijaSadrži*(žuta, kocke) nije definisan zato što argumenti nisu u domenu relacije.

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

Entitetima nazivamo sve raličite postojeće elemente („objekte“) sistema koji posmatramo, koje modeliramo bazom podataka. Neka je R neki skup entiteta. Kažemo da se skup entiteta karakteriše konačnim skupom atributa $A(E) = \{A_1,\ldots,A_n\}$, u oznaci $E(A_1,\ldots,A_n)$, akko:

- svaki atribut A_i predstavlja funkciju koja slika entitete u odgovarajući domen atributa 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.
Skup svih atributa određuje funkciju: $\alpha(e) = (A_1(e),\ldots,A_n(e))$. Slika skupa entiteta funkcijom α je skup: $R = \alpha(E)$. Ako je α injektivna funkcija, tada kažemo da je slika $R = \alpha(E)$ relacija R sa atributima $A_1,\ldots,A_n$, domenom relacije $Dom(R) = D_1x...xD_n$ i nazivima atributa $Kol(R) = (t_1...t_n)$.

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

Relaciona baza podataka je skup relacija. Opis relacije čine domen relacije i nazivi atributa. Relaciona shema je skup opisa relacija koje čine bazu podataka.

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

Funkcija je injektivna akko za sve različite originale daje različite slike tj. ako važi: $\alpha(e) = \alpha(u)$ akko $e = u$. Ako je α injektivna funkcija, tada kažemo da je slika $R = \alpha(E)$ relacija R sa atributima $A_1,\ldots,A_n$, domenom relacije $Dom(R) = D_1x...xD_n$ i nazivima atributa $Kol(R) = (t_1...t_n)$ i tada skup entiteta E sa atributima $A_1,\ldots,A_n$ modeliramo relacijom R.

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

Odnosi se modeliraju na isti način kao i entiteti – relacijama. Ako su dva entiteta $e \in E$ i $f \in F$ u nekom odnosu, onda to možemo opisati relacijom $\rho(e,f)$, čiji je domen $Dom(\rho) = E \times F$. Ako su entiteti $e \in E$ i $f \in F$ modelirani kao $e = (x_1,\ldots,x_n)$, $f = (y_1,\ldots,y_m)$, onda njihov odnos može da se modelira kao: $\rho(e,f) = (x_1,\ldots,x_n, y_1,\ldots,y_m)$. Ako su K_E i K_F neki ključevi relacija E i F, onda skup atributa $K_R = K_E \cup K_F$ predstavlja ključ relacije ρ.

19. Šta čini manipulativni deo relacionog modela?

Ključno mesto u manipulativnom delu modela ima pojam upita. Upit je definicija nove relacije na osnovu već poznatih relacija baze podataka. Pored upita, važno mesto zauzima ažuriranje baze podataka. Ažuriranje baze podataka je zamenjivanje vrednosti promenljive baze podataka novom vrednošću baze podataka.

20. Objasniti ukratko relacioni račun.

Relacioni račun je primena predikatskog računa na relacije. Kod je definisao dve varijante računa – relacioni račun n-torki i relacioni račun domena. Ekvivalentan je relacionoj algebri. Jedna od razlika u odnosu na relacionu algebru je korišćenje kvantifikatora *forall* i *exists*.

21. Objasniti ukratko relacionu algebru.

Relaciona algebra je proširenje skupovne algebre, koju čine uobičajene skupovne operacije na relacijama. Osnovne dodatne operacije su: projekcija (izdvajanje podskupa atributa), restrikcija (izdvajanje podskupa redova) i prirodno spajanje (uparivanje svih torki jedne sa svim torkama druge relacije). Kombinovanjem osnovnih operacija dobijaju se složeniji upiti kao što su prirodno spajanje (proizvod, pa restrikcija po uslovu jednakosti atributa sa istim imenom, pa projekcija na različite atribute) i slobodno spajanje (spajanje sa restrikcijom po slobodnom uslovu).

22. Šta čini integritetni deo relacionog modela?

Ovaj model čine koncepti i mehanizmi koji omogućavaju da se automatizuje proveravanje zadovoljenosti određenih uslova. Stanje baze je konzistentno (tj. ispravno) ako sadržaj baze podataka ispunjava sve uslove integriteta. Promena sadržaja baze podataka je dopušteno akko prevodi bazu iz jednog u drugo ispravno stanje. Baza podataka se opisuje relacijama. Uslovi integriteta u relacionom modelu se opisuju predikatima nad relacijom ili bazom podataka. Formalnost modela olakšava formulisanje uslova integriteta.

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

Postoje – opšti uslovi integriteta (implicitni) i specifični uslovi integriteta (eksplicitni).
Opšti uslovi moraju da važe za svaku relaciju i svaku bazu podataka. Podrazumevaju se na nivou modela ili implementacije SUBP-a. Ne definišu se eksplicitno za svaku relaciju ili bazu podataka. Najvažnije vrste logičkih pravila integriteta – integritet domena, (primarnog) ključa i stranog ključa. Specifični uslovi se odnose na pojedinu relaciju, atribut ili bazu podataka. Određuju se pri određivanju strukture baze podataka. Eksplicitno se navode odgovarajući logički uslovi.

24. Objasniti integritet domena u relacionom modelu.

Svaka relacija mora da ima tačno određen domen. Svaka vrednost nekog atributa u bazi podataka mora da pripada odgovarjućem domenu atributa. Domen je neki od podržanih tipova podataka, a može da obuhvata i dužinu podataka, opcionu deklaraciju jedinstvenosti, opcionu deklaraciju podrazumevane vrednosti.

25. Objasniti integritet ključa u relacionom modelu.

Integritet ključa se odnosi na jednu relaciju. Određuje se uslovom ključa – određuje minimalan podskup atributa relacije koji predstavlja jedinstveni identifikator torke relacije. Podskup X atributa $A_{i1},\ldots,A_{ink}$ relacije R je ključ (ili ključ kandidat) relacije R akko su ispunjeni sledeći uslovi:

- X funkcionalno određuje sve atribute relacije R
- nijedan pravi podskup od X nema prethodno svojstvo

Jedna relacija može da ima više ključeva. Jedan od ključeva se proglašava za primarni ključ i upotrebljava za jedinstveno identifikovanje torki relacije. Podskup X skupa atributa relacije R predstavlja natključ ako zadovoljava samo prvi od uslova ključa. Svaka relacija mora da ima primarni ključ. Torke relacije se po pravilu referišu pomoću primarnog ključa.

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. Pored primarnog ključa, mogu da se eksplicitno deklarišu i jedinstveni ključevi, koji su po svemu isti kao primarni ključevi, ali nije uobičajeno da se koriste pri referisanju. Osnovna namena je implementacija dodatnih uslova jedinstvenosti torki.

27. Objasniti referencijalni integritet u relacionom modelu.

Referencijalni integritet predstavlja uslove o međusobnim odnosima koje moraju da zadovoljavaju torke dveju relacija:

- 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
- ne sme se dodati torka sa neispravnom referencom (takva da ne postoji torka na koju se odnosi)

U slučaju nedoslednih implementacija, koje podržavaju nedefinisane vrednosti, dodaje se još jedan uslov:

- referenca koja sadrži nedefinisane vrednosti je ispravna akko je u potpunosti nedefinisana tj. svi njeni atributi su nedefinisani

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 sledeći uslovi:

- 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 roditeljskom relacijom. Poštovanje integriteta stranog ključa pri menjanju sadržaja baze podataka se određuje pravilom brisanja i pravilom ažuriranja.

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

Pravila brisanja – ako se pokuša brisanje torke bazne relacije B, za koju postoji zavisna torka relacije R onda možemo da imamo: aktivnu zabranu brisanja (RESTRICT), pasivnu zabranu brisanja (NO ACTION), kaskadno brisanje (CASCADE), postavljanje nedefinisanih vrednosti (SET NULL) ili postavljanje podrazumevanih vrednosti (SET DEFAULT).
Pravila ažuriranja – ako se pokuša menjanje primarnog ključa torke relacije B, za koju postoji zavisna torka relacije R onda možemo da imamo: RESTRICT, NO ACTION, CASCADE, SET NULL ili SET DEFAULT.

30. Objasniti opšte uslove integriteta relacionog modela.

Pored opisanih uslova mogu da se odrede i dodatni specifični uslovi integriteta – na atributu, na torki, na relaciji, na bazi podataka.
Na atributu – mogu da se propišu i dodatni uslovi integriteta atributa:

- lokalnog karaktera, odnosi se na vrednost jednog atributa jedne torke
- može da se koristi za dodatno sužavanje domena
- može da se koristi za proveru ispravnosti složenih tipova podataka

Na torki – dodatni uslovi integriteta torke:

- 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 relaciji – 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 bazi podataka – dodatni uslovi integriteta između više relacija:

- globalnog karaktera, odnosi se na vrednosti torki u različitim relacijama
- koristi se za proveru složenijih uslova integriteta

31. Objasniti aktivno održavanje integriteta u relacionim bazama podataka.

Aktivno održavanje integriteta podrazumeva da se integritet održava tako što se na određene promene reaguje pokretanjem eksplicitno definisanog programskog koda. Mehanizam aktivnog održavanja integriteta su okidači.

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

Okidači podrazumevaju da se na određene promene reaguje pokretanjem odgovarajućeg programskog koda. Svaki okidač je definisan relacijom na kojoj se pamte promene, vrstom promena na koje se reaguje, programskim kodom koji se izvršava, trenutkom izvršavanja, granularnošću izvršavanja i podacima koji se referišu u programskom kodu. Okidači se izvršavaju pre ili posle naredbe za dodavanje, menjanje ili brisanje torki. Na jednoj relaciji može da bude više okidača. Nekada se mora reagovati pre ili posle, a nekada je svejedno.

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

Savremeni SUBP nudi okidače na 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. Okidači na pogledima omogućavaju skrivanje veoma složenih operacija kojima se spoljašnja shema razdvaja od konceptualne, ili konceptualna od interne.

34. Objasniti motivaciju za pravljenje modela entiteta i odnosa.

Predložio ga je 1976. godine Piter Čen. Predlaže se kao naredni korak i dalje unapređenje, posle mrežnog, hijerarhijskog i relacionog modela.
„Rad predstavlja model entiteta i odnosa, koji je napredniji od navedena tri modela. Model entiteta i odnosa usvaja prirodniji pristup da se stvarni svet sastoji od entiteta i odnosa.“
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.

35. Objasniti osnove koncepte i pretpostavke modela entiteta i odnosa.

Model entiteta i odnosa prepoznaje dva različita *osnovna koncepta* – entitete i odnose. Teži da u samom modelu očuva sve bitne semantičke informacije o domenu koji se modelira.
Prepoznajemo 4 nivoa pogleda na podatke:

1. Informacije koje se tiču entiteta i odnosa, koje postoje u našem umu (odgovara konceptualnom modelu tj. apstraktnim semantičkim informacijama)
2. Strukturu informacija tj. način organizovanja informacija u kome se entiteti i odnosi predstavljaju podacima (odgovara logičkom modelu, to su strukture podataka koje čuvamo u bazi podataka)
3. Strukture podataka nezavisne od pristupnog puta tj. koje ne zahtevaju sheme pretraživanja, sheme indeksiranja i sl. (odgovara višem fizičkom modelu sa određenim nivoom apstrakcije)
4. Strukture podataka koje zavise od pristupnog puta (niži fizički model, praktično bez apstrakcije, implementacija niskog nivoa)

Mrežni model se bavi prvenstveno nivoom 4. Relacioni model se bavi prvenstveno nivoima 2 i 3. Model entiteta i odnosa se bavi prvenstveno nivoima 1 i 2.

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 identifikuje. Odnos je neko međusobno pridruživanje entiteta. Entiteti se klasifikuju u različite skupove entiteta. Svakom skupu entiteta odgovara predikat koji proverava da li mu neki entitet pripada. Skupovi entiteta ne moraju da budu disjunktni. Skup odnosa je matematička relacija između N entiteta koji pripadaju nekim skupovima entiteta, a čiji elemenati su 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 skupove vrednosti. Atribut može da se definiše kao funkcija koja preslikava skup entiteta ili skup odnosa u skup vrednosti ili Dekartov proizvod skupova 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.

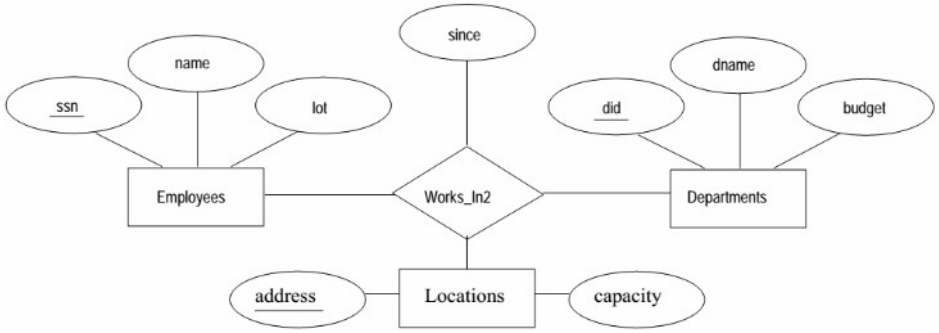
Entitet je stvar koja može da se jednoznačno identifikuje. Odnos je neko međusobno pridruživanje entiteta. Entiteti se klasifikuju u različite skupove entiteta. Svakom skupu entiteta odgovara predikat koji proverava da li mu neki entitet pripada. Skupovi entiteta ne moraju da budu disjunktni. Skup odnosa je matematička relacija između N entiteta koji pripadaju nekim skupovima entiteta, a čiji elemenati su odnosi.

38. Šta su slabi i jaki entiteti?

Jaki entiteti – identifikuju se sami za sebe. Uglavnom nezavisni od drugih entiteta u bazi. Njihovo postojanje nije uslovljeno postojanjem drugih entiteta.
Slabi entiteti – identifikuju 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 pravougaonicima. Atribut se predstavlja elipsom. Primenjivost atributa na skup entiteta se predstavlja linijom. Podvlačenjem se ističu atributi koji predstavljaju ili čine primarni ključ. Odnosi se predstavljaju rombovima, i oni mogu imati opisne atribute. Odnosi mogu da uključuju više entiteta. Odnosi mogu da budu i između više instanci istog skupa entiteta.



40. Šta su i kako se na ER dijagramima 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 (označava se strelicom od entiteta prema odnosu).
Uslov učešća – ako svaki entitet skupa učestvuje u bar jednom odnosu, onda je to „puno učešće u odnosu“, a inače je „parcijalno učešće“. Puno učešće u odnosu se označava debljom linijom.

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. Označava se po brojem na svakoj od linija koje vode od entiteta prema odnosu. Broj označava u koliko takvih odnosa može (mora) da učestvuje jedan entitet. Umesto jednog broja, može da stoji opseg vrednosti (A,B) ili A..B.
Kardinalnost se delimično poklapa sa navedenim uslovima: „bar 1“ == uslov učešća; „najviše 1“ == uslov ključa; „tačno 1“ == uslov učešća i ključa.

42. Kako se na ER dijagramima označavaju slabi entiteti?

Elementi slabog skupa entiteta se identifikuju samo u sklopu odnosa sa nekim drugim entitetom. Taj odnos mora biti „jedan-više“ i naziva se identifikujući odnos. Slab entitet mora imati puno učešće u identifikujućem odnosu. I identifikujući odnos i učešće slabog entiteta u njemu se označavaju debljim linijama.

43. Kako se na ER dijagramima označavaju hijerarhije?

Hijerarhije klasa su tzv. „ISA“ hijerarhije. Opštije od nasleđivanja u OOP. Uslov preklapanja – da li isti entitet može da pripada dvema potklasama ili ne. Uslov prekrivanja – da li svaki entitet natklasi pripada nekoj od potklasa ili ne.

44. Šta je i kako se na ER dijagramima označava „agregacija“?

Agregacije su odnosi koji učestvuju u drugim odnosima. Takve odnose uokvirujemo isprekidanom linijom. To nije isto što i ternatne relacije. Alternativa je da se takav odnos okruži pravougaonikom.

45. Objasniti osnovne slabosti ER modela.

Postoje određene nedoslednosti. Počiva na pretpostavci da postoje dva 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 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”.
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 na sveobuhvatni model podataka. Danas se ER najčešće i ne koristi kao model, već kao dijagramska tehnika relacionog modela.

46. Objasniti dilemu „entitet ili atribut“?

Ako imamo složen atribut, možemo da ga modeliramo kao jedan složen atribut, više prostih atributa ili jedan entitet.
Argumetni 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; 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; ako važi slično i kada je to atribut u različitim skupovima entiteta.
Atribut mora da preraste u entitet – ako nekom entitetu odgovara istovremeno više vrednosti tog atributa; ako postoji međuzavisnost vrednosti atributa.

47. 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.
U većini slučajeva je jednako ispravno da nešto bude bilo odnos bilo entitet.

48. Objasniti dilemu „složeni odnos ili više binarnih odnosa“?

Većina složenih odnosa može da se podeli na više binarnih odnosa. Složene odnose ima smisla podeliti na više jednostavnijih isključivo ako svaki od dobijenih odnosa ima svoj semantički smisao u posmatranom domenu. Čak i tada, ako je semantička očiglednija iz složenog odnosa, onda bi taj složeni odnos trebalo da ostane.

49. Objasniti dilemu „agregacija ili ternarni odnos“?

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 obično bolje da se koriste složeni odnosi.

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

Svaki entitet se prevodi u relaciju. Atributi entiteta se prevode u atribute relacije. Ključni atributi se prevode u primarni ključ.
Većina odnosa se prevodi u relacije. Atributi odnosa se prevode u atribute dobijene relacije. Dodaju se i ključni atributi uključenih entiteta. Primarni ključ dobijene relacije čine primarni ključevi relacije koje odgovaraju uključenim entitetima. Primarnom ključu se dodaju i svi ključni atributi odnosa. Strani ključevi su primarni ključevi učesnika. Ako je odnos *:0..1 ili *:1, onda ne mora da se pravi dodatna relacija za odnos, atributi odnosa se dodaju relaciji koja odgovara entitetu sa kardinalnošću 0..1. Slično može i u slučaju 0..1:0..1.

51. Kako se hijerarhije ER modela prevode u relacioni model?

Hijerarhije se mogu rešiti na tri osnovna načina:

- cela hijerarhija u jednu relaciju – obuhvata sve atribute koji postoje u hijerarhiji; za svaki entitet se koristi samo deo atributa, ostali imaju nedefinisane vrednosti; relativno efikasna upotreba; neefikasno zauzeće prostora.
- svaki entitet u posebnu relaciju – svaka relacija obuhvata samo one atribute koji su za nju specifični; zahtevaju se česta spajanja pri čitanju i potencijalno je neefikasan.
- svaki entitet-list u posebnu relaciju, ali tako da uključi sve nasleđene atribute – svaka relacija obuhvata atribute koji su za nju specifični i sve atribute svih baznih klasa; atributi ključa bazne relacije se koriste kao strani ključ; za entitet koji nije list se ili pravi relacija koja ima samo primarni ključ ili se čitava relacija izostavlja.; upotreba pojedinalnih entiteta-listova je efikasna; pretraživanje baznog skupa entiteta je neefikasno.

Moguće je i kombinovanje metoda, za različite delove hijerarhije.

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

Slab entitet i odgovarajući odnos se obično modeliraju jednom relacijom. Dodaje se i ključni atribut vezanog jakog entiteta, kao deo primarnog ključa.

53. Šta su „dijagrami tabela (relacija)“?

Oznake i termini su obično tabele/kolone/ključevi. Nema dovoljno semantičkih informacija. Kardinalnosti se predstavljaju drugačije nego na ER-dijagramima, po uzoru na UML. Prilagođeni su logičkom i fizičkom nivou relacionog modela. Semantika odnosa je predstavljena u meri u kojoj to dopušta relacioni model.

54. 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.

55. Kako se označavaju ključevi u dijagramima tabela?

Ako kolona pripada primarnom ključu – simbol +, oznaka PK ili crtež ključa.
Ako kolona pripada stranom ključu – simbol #, oznaka FK, crtež ključa sa strelicom i obično iskošen naziv.

56. Kako se označavaju različite vrste odnosa i kardinalnosti odnosa u dijagramima tabela?

Obično se predstavljaju isprekidanim linijama (puna linija predstavlja odnos slabog entiteta sa matičnim). Na kraju linije se predstavlja kardinalnost – u duhu UML-a, broj entiteta uz koji stoji oznaka, koji mogu da budu u odnosu sa jednim entitetom koji je na drugoj strani. Grafičke oznake: 0 – kružić; 1 – upravna linija; * - linija se grana u tri kratke linije prema tabeli; 0-1, 0-*, 1-* - kombinacije prethodnih slučajeva.

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

Na *konceptualnom* nivou sadrži samo osnovnu strukturu podataka. Dijagram ne sadrži dodatne kolone surogat ključeva, dodatne kolone stranih ključeva, oznake ključeva niti tipove kolona. Ovakav pojednostavljen dijagram može da se koristi i na *logičkom* nivou, mada je obično bolje da bude u osnovnom obliku.
Na *fizičkom* nivou se dodaju svi ostali podaci neophodni za preslikavanje u relacioni model – kolone koje su surogat ključevi, kolone vezanih tabela koji čine strane ključeve, oznake ključeva i vrsta ključeva, tipovi kolona i odnosi više-više se zamenjuju tabelama koje ih modeliraju.

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

Pogledi – obično pravougaonici označeni dodatnim simbolom i vezani linijom sa tabelama na osnovu kojis se definišu. Ako dopuštaju menjanje podataka, može i dodatno označavanje.
Sekvence – mogu biti vezane za više tabela istovremeno.
Okidači – uglavnom su vezani za jednu tabelu.
Uslovi integriteta – dodaju se na različite načine, obično u vidu komentara.

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

UML dijagram klasa opisuje strukturu podataka klasa. 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, ima manje nedostatke koji mogu da se prevaziđu. Modeli klasa i relacija se konceptualno razlikuju – relacije su skupovi podataka, a klase su tipovi podataka. Može da se prevaziđe u praksi:

- ako je potrebno modelirati više skupova istog tipa, uvodimo više naslednika klase koja određuje tip
- svaka „klasa“ može da se označi kao «type» ili «persistent»
- problem se ignoriše i smatra da je klasa skup a ne tip

Za projektovanje modela baze podataka se koristi tzv. konceptualni dijagaram klasa. Za razliku od uobičajenog dijagrama klasa - u prvom planu su atributi i odnosi, ponašanje se skoro potpuno zanemaruje, i enkapsulacija je u drugom planu.

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

UML sadrži standardizovane koncepte koji omogućavaju uvođenje novih načina označavanja:

1. *stereotipovi*
2. *označene vrednosti*
3. *proširenja*

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

Stereotip predstavlja vrstu šablona - apstrakciju opštijeg slučaja nečega, neki predefinisani skup osobina, neko predefinisano ponašanje. Koriste se za navođenje dodatnih deklaracija ili napomena. Mogu da se navode i uz klase i uz atribute.
Podrazumevano označavanje je navođenjem naziva između dvostrukih izlomljenih zagrada: «abstract» , «entity», «view», «persistent», «table», «generated», «auto», «key»… Obično se navodi iznad, ispred ili ispod naziva klase. Za neke uobičajene stereotipove su uvedene i grafičke oznake (stoje kao dopuna u gornjem desnom uglu uobičajenog simbola za tu vrstu objekta).

62. Objasniti osnovne odnose u dijagramima klasa podataka.

Osnovni odnosi:

1. *asocijacija* – odnos između dve klase. Označava da bar jedna od klasa zna za neke objekte druge klase i na neki način upravlja njima ili komunicira sa njima. Može da bude funkcionalna („uradi nešto za mene“) ili strukturalna („budi nešto za mene“ - važnija)
2. *agregacija* – posebna vrsta asocijacije. Implicira da se objekat jedne klase „sastoji“ od objekata druge klase. Predstavlja slabiji oblik strukturalne asocijacije.
3. *kompozicija* – predstavlja jači oblik strukturalne asocijacije – uvek je binarni 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.
4. *nasleđivanje* – odnos koji je suštinski za OO modeliranje, pa time i za model klasa podataka. Osnovna klasa predstavlja opštiji slučaj izvedenih klasa (generalizacija). Izvedene klase predstavljaju posebne slučajeve bazne klase (specijalizacija).
5. *navigacija* – model klasa podataka načelno podrazumeva da se za referisanje drugih objekata koriste neki vidovi jedinstvenih identifikatora tzv. OID. Većina OOBP podrazumeva da identifikator objekta nema nikakvo semantičko značenje i njegova vrednost često čak ne može ni da se vidi ili promeni. Takvi ključevi identifikuju promenljive, a ne podatke.

63. Kako se dijagrami klasa podataka koriste na različitim nivoima modeliranja?

Razlike u načinu predstavljanja modela su slične kao kod dijagrama tabela. Konceptualni dijagram ne mora da sadrži – dodatne atribute surogat ključeva, dodatne atribute stranih ključeva, oznake ključeva, precizne tipove atributa. Fizički dijagram sadrži i – atribute surogat ključeva, atribute vezanih tabela koji čine strane ključeve, oznake ključeva i vrsta ključeva, tačne tipove atributa.
Iste odnose često možemo da modeliramo ispravno na različite načine. Izbor često zavisi od nivoa modela:

- na konceprualnom nivou cilj je da očuvamo semantiku odnosa iz domena problema
- na logičkom nivou cilj je da ostvarimo stabilnu i jednoznačnu strukturu
- na fizičkom nivou cilj je da omogućimo i dobre performanse

64. Kako se u dijagramima klasa označavaju ključevi?

Simbolima ili tekstom.

65. Kako se u dijagramima klasa označavaju uslovi integriteta?

U okviru klasa u odeljku ponašanja, ili kao povezane klase ili komentari.

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

1. *Konceptualno projektovanje* – odgovara spoljašnjoj shemi i delu konceptualne sheme. Najpre se pravi po apstraktan pojedinačni konceptualni model za svaku od spoljašnjih shema, a zatim se od njih pravi jedinstven objedinjeni konceptualni model.
2. *Logičko projektovanje* – uglavnom odgovara konceptualnoj shemi. Prilagođavanje konceptualnog modela konkretnom modelu podataka. Pravi se konkretan logički model.
3. *Fizičko projektovanje* – odnosi se na delove konceptualne sheme i fizičke sheme. Pravi se fizički (implementacioni) model.
4. *Projektovanje bezbednosti* – uglavnom ortogonalno u odnosu na ostale korake. Prepliće se sa njima, ali se u najvećoj meri odvija u toku i nakon fizičkog projektovanja.

67. Objasniti korak Konceptualno projektovanje pri projektovanju baza podataka.

Sastoji se iz sledećih podkoraka:

1) Analiza zahteva – razumevanje podataka (strukture podataka, obima podataka, odnosa među podacima) i razumevanje aplikacija (potrebe za podacima, učestalost upita i transakcija, performanse)
2) Konceptualno projektovanje pojedinačnih spoljašnjih shema – pravljenje modela podataka visokog nivoa za svaku spoljašnju shemu posebno i opisivanje struktura podataka, odnosa među strukturama podataka i uslova integriteta
3) Pravljenje objedinjenog konceptualnog modela – pravljenje objedinjenog apstraktnog modela podataka, ujednačavanje rečnika i opisa domena i integrisanje pogleda u jedinstveni model
4) Grupisanje entiteta – pravljenje više preglednijih dijagrama modela, grupisanje jače povezanih entiteta u celine i uočavanje centralnih entiteta za odgovarajuće grupe

68. Objasniti korak Logičko projektovanje pri projektovanju baza podataka.

1) Prevođenje konceptualnog u logički model – svi opisi se prevode na jezik logičkog modela, inicijalno pravljenje modela i osnovno prilagođavanje konceptualnog modela konkretnom implementacionom modelu podataka
2) Prečišćavanje sheme – dosledno prilagođavanje modela jeziku i pravilima konkretnog modela podataka, prepoznaju se potencijalni problemi i rešavaju se, u slučaju relacionog modela obično je osnovni cilj eliminacija redundantnosti.

69. Objasniti korake Fizičko projektovanje i Projektovanje bezbednosti pri projektovanju baza podataka.

Fizičko projektovanje:

1) Fizičko projektovanje implementacije – optimizovanje logičkog modela specifičnostima primenjenog SUBP i prema očekivanom modelu upotrebe i određivanje tačne interne sheme baze podataka, kao i određivanje mehanizama preslikavanja fizičke u konceprualnu shemu
2) Fizičko projektovanje spoljašnjih shema – projektovanje implementacije spoljašnjih shema, aplikativnih interfejsa za pristupanje podacima i prilagođavanje interne sheme specifičnostima primenjenih spoljašnjih alata

Projektovanje bezbednosti – prepoznavanje vrste i uloge svakog korisnika; prepoznavanje vrste aplikacije; definisanje korisničkih grupa i odg. minimalnih skupova privilegija za sve različite uloge i vrste korisnika i aplikacija; prepoznavanje delova baze podataka koji su posebno osetljivi i u kojima je potrebno dodatno redukovati pristup; definisanje i implementiranje odg. bezbednosnih mehanizama.

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

Pravljenje konceptualnog modela počiva na sledećim koracima (kao 67. pitanje):

1. *Analiza zahteva* – razumevanje podataka i razumevanje aplikacija
2. *Konceptualno modeliranje podataka i odnosa* – pravljenje modela podataka visokog nivoa za svaku spoljašnju shemu posebno i opisivanje struktura podataka, odnosa među strukturama podataka i uslova integriteta
3. *Integrisanje pogleda* – pravljenje objedinjenog apstraktnog modela podataka, ujednačavanje rečnika i opisa domena i integrisanje pogleda u jedinstveni model
4. *Grupisanje entiteta* – pravljenje više preglednijih dijagrama modela, grupisanje jače povezanih entiteta u celine i uočavanje centralnih entiteta za odgovarajuće grupe

71. Objasniti korak Analiza zahteva pri pravljenju konceptualnog modela BP.

Analiza zahteva je veoma važna i zahtevna. Tesno je povezana sa procesom analize zahteva u kontekstu projektovanja informacionog sistema. Da bi se razumele potrebe za podacima često moraju da se detaljno razumeju svi procesi koji se odvijaju u sistemu.

72. Navesti i objasniti ciljeve koraka Analiza zahteva pri pravljenju konceptualnog modela BP.

Osnovni ciljevi:

- prevođenje funkcionalnih zahteva u kontekst trajnih podataka
- razumevanje i opisivanje podataka i njihove uloge
- definisanje nefunkcionalnih zahteva
- određivanje i oblikovanje dodatnih uslova implementacije
- izrada iscrpne dokumentacije za sve navedeno
- dekompozicija sistema na skup slabije povezanih delova/segmenata

73. Objasniti ukratko korak Konceptualno modeliranje podataka pri pravljenju konceptualnog modela BP.

Glavni koraci su klasifikacija skupova podataka (prepoznavanje entiteta/klasa i atributa) i prepoznavanje odnosa (generalizacija/serijalizacija i hijerarhije, asocijacije i složeniji odnosi).

74. Koje poslove obuhvata konceptualno modeliranje podataka?

Cilj je pravljenje modela podataka visokog nivoa za svaku spoljašnju shemu posebno i opisivanje struktura podataka, odnosa među strukturama podataka i uslova integriteta. Sve se opisuje iz ugla korisnika.

75. Šta obuhvata klasifikacija skupova podataka?

Svaka imenica je kandidat za entitet ili atribut. Klasifikacija skupova podataka je prepoznavanje entiteta/klasa i atributa.

76. Kako se ustanovljava da li bi nešto trebalo da bude atribut ili entitet?

Neke smernice:

- entiteti bi trebalo da sadrže opisne informacije
- ako nešto ima višestruku ili složenu vrednost onda je verovatno entitet
- atributi bi trebalo da pripadaju onim entitetiima koje najneposrednije opisuju

77. Koji elementi odnosa moraju da se ustanove pri modeliranju odnosa?

Za svaki odnos je potrebno prepoznati – učesnike, kardinalnost, dodatne atribute koji opisuju odnos i jasan naziv i semantiku odnosa u domenu.

78. Objasniti problem redundantnih odnosa. Kako se uočavaju redundantni odnosi?

Redundantni odnosi imaju za rezultat teško otklonjivu redundantnost u modelu i implementaciji i ometaju konzistentnost i normalizaciju. Potrebno je eliminisati redundantne odnose ili bar naglasiti da su redundantni.

79. Kako se postupa sa odnosima sa više od dva učesnika?

Uvek je potrebno da se dodatno razmisli da li su ovi odnosi neophodni. Vrlo često je bolje da se odnos višeg reda podeli na više jednostavnijih odnosa. Uvek pokušati sa više binarnih odnosa; ako se ispostavi da je nemoguće, tek onda koristiti odnose višeg reda.

80. 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. Osnovna motivacija – analiza i modeliranje manjih delova domena se izvode jednostavnije i efikasnije; manji modeli se lakše oblikuju i razumeju. Svaki pojedinačan model se naziva pogledom ili lokalnom shemom. Opšta (globalna) shema se dobija integrisanjem lokalnih shema (pogleda).

81. 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. Cilj je pravljenje objedinjenog apstraktnog modela podataka, ujednačavanje rečnika i opisa domena i integrisanje pogleda u jedinstveni model. Osnovni koraci:

- poređenje shema i prepoznavanje konflikata
- preuređivanje shema i razrešavanje konflikata
- spajanje i restukturiranje shema

82. Navesti i objasniti vrste konflikata koji mogu da nastanu pri integrisanju pogleda.

1. *Konflikti imena* - neujednačeno imenovanje entiteta i odnosa. Dva 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.
2. *Strukturni konflikti* – različiti strukturni elementi se upotrebljavaju za modeliranje istih koncepata. Da bi pogledi mogli da se spoje, ogovarajući koncepti moraju da imaju identične strukture
3. *Konflikti ključeva* – istom entitetu su u različitim pogledima dodeljeni različiti ključevi. Neophodno je da se ključevi ujednače. U suprotnom, preti opasnost od redundantnih ključeva.
4. *Konflikti zavisnosti* - različito prepoznate funkcionalne zavisnosti

83. Objasniti detaljno konflikte imena pri integrisanju pogleda.

*N*eujednačeno imenovanje entiteta i odnosa. Dva 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.

84. Kako se razrešavaju konflikti pri integrisanju pogleda?

Može da zahteva dodatnu ili ponovljenu analizu zahteva. Može da zahteva ozbiljno modifikovanje pogleda. Dobro je da učestvuju projektanti konfliktnih pogleda

85. Kojim principima se rukovodi pri spajanju i restrukturiranju lokalnih shema? Objasniti ih.

- *Potpunosti* – svi koncepti iz lokalnih shema moraju da se očuvaju i u potpunosti prenesu u globalnu shemu. Nakon razrešavanja konflikata najpre spajamo sve odgovarajuće koncepte i ostvarujemo potpunost.
- *Minimalnosti* – svi redundantni koncepti moraju da se uklone iz globalne sheme. Prepoznajemo redundantne koncepte i pokušavamo da ih razrešimo (uvođenjem generalizacije ili uklanjanjem reduntantnih odnosa).
- *Razumljivosti* – globalna shema mora da bude razumljiva svim korisnicima. Po potrebi dodatno uređujemo model da bi bio razumljiviji.

86. 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. Obrišu se atributi i eventualno još neki detalji predstavljanja entiteta i entitet se zaokruži dvostrukom linijom. Grupa entiteta se zamenjuje simbolom grupe sa podebljanim okvirom. Važno je jer nam olakšava sagledavanje sheme.

87. Navesti i objasniti principe grupisanja entiteta pri pravljenju konceptualnog modela BP.

- *Grupisanje prema dominantnosti* – dominantni entiteti se uočavaju na osnovnu odnosa – učestvuju u većem broju odnosa i posredno povezuju veće delove dijagrama. Jedan dominantan entitet se grupiše sa svim pripadajućim nedominatnim entitetima.
- *Grupisanje prema apstraktnosti* – vrši se ako postoje hijerarhije. Hijerarhija se predstavlja jednim baznim entitetom.
- *Grupisanje prema uslovima* – ako postoje neki složeni uslovi koji moraju da važe u nekim odnosima, grupišu se odgovarajući entiteti. Cilj je sklanjanje složenih uslova iz velikog dijagrama, radi povećanja čitljivosti.
- *Grupisanje prema odnosima* – odnosi višeg reda i odgovarajući entiteti mogu da se grupišu. Ovakva grupa predstavlja odnos kao jednu celinu.

88. Objasniti postupak grupisanja entiteta pri pravljenju konceptualnog modela BP?

- Prepoznaju se elementi koji se grupišu u okviru funkcionalnih oblasti
- Grupišu se entiteti - svaka grupa mora da u potpunosti pripada jednoj funkcionalnoj oblasti
- Napraviti grupe višeg reda - rekurzivnom primenom postupka
- Proveriti ispravnost dijagrama - svi interfejsi moraju da budu konzistentni

89. Šta je logički model baze podataka?

Logički model se spušta bliže izabranom implementacionom modelu. Uzima u obzir model podataka koji će se upotrebljavati u implementaciji. Preduslov za započinjanje logičkog modeliranja je odabiranje vrste baze podataka tj. modela podataka. Ulaz za postupak logičkog modeliranja je konceptualni model, a izlaz je detaljan logički model.

90. U čemu je osnovna razlika između konceptualnog i logičkog modeliranja?

Konceptualan model je fokusiran na semantiku i odnose (uglavnom ne zavisi od vrste baza podataka koja će biti upotrebljavana u implementaciji).
Logički model se spušta bliže izabranom implementacionom modelu (uzima u obzir model podataka koji će se upotrebljavati u implementaciji).

91. Šta je očekivani rezultat logičkog modeliranja?

Izlaz je detaljan logički model – logička shema svih trajnih objekata (relacija), specifikacija svih uslova i ograničenja, poznati su sci ključevi i odgovarajući surogat-atributi, kao i način implementacije svih odnosa.

92. Zašto nije dobro preskočiti logički model i praviti fizički model na osnovu konceptualnog?

Takav pristup nosi rizik da se izgube neke od poželjnih karakteristika baze podataka – kompletnost, integritet, fleksibilnost, efikasnost, upotrebljivost.

93. Kako teče postupak pravljenja logičkog modela?

Logički model se pravi iterativno. Prva iteracija se pravi na osnovu konceptualnog modela. Svaka naredna se pravi menjanjem prethodne (flexing).

94. 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?

95. Kako se konceptualni model „prevodi“ u logički?

Prvi korak pri pravljenju logičkog modela je prevođenje konceptualnog modela u logički model. Ako se oba modela izražavaju na ER-u ili UML-u, onda je prevođenje često transformisanje ili dopunjavanje i nadograđivanje.
Logički model predstavljamo jezikom relacionog modela – entiteti se predstavljaju kao relacije; složeni odnosi se predstavljaju kao relacije; jednostavni odnosi se predstavljaju stranim ključevima.
Postupak prevođenja počinje prevođenjem entiteta u relacije (usput mogu da se prevedu i neki odnosi, 1-*) - prepoznaje se primarni ključ i prevode se svi neključni atributi.
Zatim se nastavlja prevođenjem odnosa (najpre se prevode odnosi koji proizvode relacije, pa preostali odnosi, i na kraju se doteruje prevod odnosa) – mora da se uzme u obzir priroda relacionog modela. Različite vrste odnosa – binarni 1-*, binarni *-*, složeni odnosi, generalizacija.

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

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. Čest slučaj kod agregacije. Relacija koja je *deo* – dodaje se strani ključ u odnosu na *celinu*. Strani ključ *može* da bude 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. Čest slučaj kod kompozicije. Relacija koja je *deo* - dodaje se strani ključ u odnosu na *celinu.* Strani ključ *ne sme* da bude nedefinisan.
0..1 – 1. Uobičajeno za odnose: nadodređeni – podređeni, celina – opcioni deo. Relacija koja je *opcioni deo* – dodaje se strani ključ u odnosu na *celinu*. Strani ključ *ne sme* da bude nedefinisan i vrednost ključa *mora* da bude jedinstvena.
0..1 – 0..1. Uobičajeno za dvosmerne opcione odnose. Jednoj od relacija se dodaje strani ključ u odnosu na drugu. Strani ključ *može* da bude nedefinisan i vrednost ključa *mora* da bude jedinstvena. Alternativa, pravi se nova *vezna* relacija koja sadrži samo strane ključeve u odnosu na obe vezane relacije; ključevi *ne smeju* da budu nedefinisani, svi atributi čine primarni ključ i vrednost svakog od stranih ključeva mora da bude jedinstvena.

97. 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. Relacija koja je *deo* – dodaje se strani ključ u odnosu na *celinu.* Strani ključ *može* da bude nedefinisan. Dodaje se uslov baze podataka – za svaki entitet A mora da postoji bar jedan odgovarajući entitet B. Alternativa, među svim delovima jedne celine se prepoznaje jedan izabran; u relaciju A se dodaje strani ključ u odnosu na B i time se celina povezuje sa izabranim delom; olakšano je proveravanje postojanja „bar jednog“, ali je uvedena redundantnost.
1 – 1..N. Uobičajeno za odnose kompozicije sa obaveznim delovima. Relacija koja je *deo* – dodaje se strani ključ u odnosu na *celinu.* Strani ključ *ne sme* da bude nedefinisan. Dodaje se uslov baze podataka – za svaki entitet A mora da postoji bar jedan odgovarajući entitet B.
1-1. Uobičajeno za odnose obostranog ekskluzivnog pridruživanja. Jednoj od relacija se dodaje strani ključ u odnosu na drugu. Strani ključ *ne sme* da bude nedefinisan. Vrednost ključa *mora* da bude jedinstvena. Dodaje se uslov baze podataka – za svaki entitet B mora da postoji bar jedan odgovarajući entitet A. Alternativa, u svakoj od relacija se dodaje strani ključ u odnosu na drugu; atributi ključeva *ne smeju* da budu nedefinisani i dodaje se uslov kojim se proverava uzajamnost veze.

98. 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 vezane relacije; ključevi *ne smeju* da budu nedefinisani; svi atributi čine primarni ključ.
0..N – 1..N. Uobičajeno za odnose: asocijacije 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 baze podataka – za svaki entitet A mora da postoji bar jedan odgovarajući entitet B. Alternativa, u A se dodaje strani ključ prema izabranom 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 baze podataka – za svaki entitet A mora da postoji bar jedan odgovarajući entitet B, i obrnuto. Alternativa, i u A i u B se dodaju strani ključevi prema izabranim delovima/celinama.

99. Kako se u logičkom modelu modeliraju binarni ciklični odnosi?

1-1. Bilo da su opcioni ili ne, predstavljaju se dodatnim atributima stranog ključa. Ako je opcioni, sme da bude NULL.
1-*. Kod ovog odnosa strani ključ se uvodi na strani *N*.
****-****. Predstavlja se novom relacijom, kao i obični binarni odnosi *-*.

100. 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.
1-1-*. Nova relacija sa stranim ključevima. Zavisnosti se uređuju dopuštanjem NULL i jedinstvenim ključevima.
1-*-*. Nova relacija sa stranim ključevima. Zavisnosti se uređuju izborom primarnog ključa.
****-*-****. Nova relacija sa stranim ključevima.

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

Generalizacija – cela hijerarhija u jednu relaciju; svaki entitet u posebnu relaciju; svaki entitet-list u posebnu relaciju, ali tako da uključi sve nasleđene atribute.
Agregacija – svodi se na obične odnose, obično 1-*.
Odnosi sa više od 3 učesnika – slično kao odnosi sa 3 učesnika; mora više da se vodi računa o funkcionalnim zavisnostima.
Slabi entiteti – obično ne zahteva dodatno postupanje; prevođenje odnosa sa jakim entitetima rešava problem.

102. 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.
Pri prečišćavanju sheme: analiziramo dobijeni logički model; za sve pojedinačne elemente proveravamo usklađenost sa osnovnim konceptima odgovarajućeg modela podataka; po potrebi uvodimo izmene u model; vodimo računa o (funkcionalnim) zahtevima.
Osnovni cilj je dosledno i potpuno usklađivanje modela sa modeliranim domenom, poznatim zahtevima i teorijskim modelom baze podataka.

103. Objasniti probleme koji nastaju usled redundantnih podataka.

- *Redundantno čuvanje podataka* – neki podaci se ponovljeno čuvaju, pa više kopija istog podatka nepotrebno opterećuje prostorne kapacitete
- *Anomalije ažuriranja* – ako se jedna kopija ponovljenog podatka ažurira, baza podataka će biti nekonzistentna ako se ne ažuriraju i ostale kopije
- *Anomalije dodavanja* – zapisivanje nekih podataka može da dovodi do nekonzistentnosti ako se ne zapišu još neki dodatni podaci
- *Anomalije brisanja* – brisanje nekih podataka može da dovodi do nekonzistentnosti ako se ne obrišu još neki podaci

104. Uloga nedefinisanih vrednosti u rešavanju problema redundantnosti.

Nedefinisane vrednosti mogu da pomognu u rešavanju nekih anomalija dodavanja i brisanja. Ako moramo da dodamo podatak a ne znamo druge potrebne podatke, onda možemo da navedemo nedefinisane vrednosti. Ako hoćemo da obrišemo neke redundantne podatke, a da ne brišemo cele redove, onda možemo da umesto brisanja upišemo nedefinisane vrednosti.

105. Objasniti postupak dekompozicije kao alat za otklanjanje redundantnosti.

Redundantnost se pojavljuje na mestima na kojima postoje ne sasvim prirodne veze između atributa. Uobičajeno i ispravno sredstvo za rešavanje redundantnosti je dekompozicija. Ideja je da se jedna relacija sa mnogo atributa zameni većim brojem relacija sa po manje atributa.

106. Objasniti funkcionalne zavisnosti?

Funkcionalna zavisnost je odnos između dva skupa atributa na skupu torki jedne relacije. 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 $X \rightarrow Y$ akko za svaki par torki t_1 i t_2 iz *r* važi:

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

Ako govorimo o relacijama, onda imamo tranziciju pojmova: umesto da uočavamo zavisnosti na konkretnim podacima, mi ćemo da propisujemo da je sadržaj relacije ispravan akko ga čini skup redova za koji važe date funkcionalne zavisnosti. FZ na nivou relacije predstavlja pravilo integriteta.

Aksiome izvođenja FZ – za sve skupove atributa X, Y, Z neke relacije važe:

1. *refleksivnost*: $X \rightarrow X$ tj. $Y \subseteq X \Rightarrow X \rightarrow Y$
2. *proširivost*: $X \rightarrow Y \Rightarrow (\forall Z)\ XZ \rightarrow Y$ tj. $X \rightarrow Y \Rightarrow (\forall Z)\ XZ \rightarrow YZ$
3. *tranzitivnost*: $X \rightarrow Y \wedge Y \rightarrow Z \Rightarrow X \rightarrow Z$

Najvažnije izvedene osobine FZ:

1. *aditivnost (unija)*: $X \rightarrow Y \wedge X \rightarrow Z \Rightarrow X \rightarrow YZ$
2. *projektivnost (dekompozicija)*: $X \rightarrow YZ \Rightarrow X \rightarrow Y \wedge X \rightarrow Z$
3. *pseudotranzitivnost*: $X \rightarrow Y \wedge YZ \rightarrow W \Rightarrow XZ \rightarrow W$

107. Šta su normalne forme? Objasniti suštinu i navesti najvažnije normalne forme.

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

1NF. Relacija je u prvoj normalnoj formi ako svaki atribut može da ima samo atomične vrednosti.

2NF. Relacija je u drugoj normalnoj formi ako je u prvoj normalnoj formi i ako nijedan neključan atribut nije funkcionalno zavisan od nekog pravog podskupa atributa nekog kandidata za ključ.

3NF. Ako je R relacija i X neki podskup njenih atributa i A neki atribut relacije R, onda je R u trećoj normalnoj formi akko za svaku FZ $X \rightarrow A$ na relaciji R važi jedno od: $A \in X$, X je natključ ili A je deo nekog ključa relacije.

NFEK. Relacija je u normalnoj formi elementarnog ključa ako je u 3NF i akko za svaku elementarnu FZ $X \rightarrow Y$ na relaciji R važi jedno od: X je ključ ili Y je elementarni ključ ili deo elementarnog ključa.

BCNF. Ako je R relacija i X neki podskup njenih atributa i A neki atribut relacije R, onda je R u Bojs-Kodovoj normalnoj formi akko za svaku FZ $X \rightarrow A$ na relaciji R važi jedno od: $A \in X$ ili X je natključ.

108. Šta je „normalizacija“? Kada se primenjuje? Zašto? Kako?

Normalizacija logičkog modela baze podataka je svođenje 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 rekurzivno

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.

109. Šta je fizički model baze podataka?

Fizički model je najniži model baze podataka. Opisuje konkretnu implementaciju. Uzima u obzir mnoge tehničke aspekte implementacije. Često se naziva i "modeliranje podataka".

110. Šta čini fizički model baze podataka?

Elementi fizičkog modela:

- *strukture podataka* – na fizičkom nivou se govori o tabelama i kolonama
- *interna (fizička) organizacija podataka* – prostori za tabele, kontejneri, stranice, baferi…
- *pomoćne komponente* - indeksi

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

Da bismo mogli da procenimo opterećenje i performanse, moramo da imamo: model obrade podataka, nestrukturne zahteve, matricu entiteta i procesa, zahtevane performanse, prepoznate ciljne implementacije SUBP, prepoznata eventualna ograničenja prostora, prepoznate eventualne razvojne probleme.

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

Model obrade podataka obuhvata:

- *okolnosti dodavanja novih redova* – koliko redova u proseku; koliko redova pri najvećem opterećenju; 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 PK istovremeno koriste
- *okolnosti brisanja redova* - oliko 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; koje druge tabele se često koriste zajedno sa posmatranom

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

- *Trajanje podataka* – koliko dugo se podaci zadržavaju u tabeli pri brisanju ili arhiviranju?
- *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 privremeno, koliko često i dugo podaci mogu/smeju da budu nedostupni korisnicima?
- *Ažurnost podataka* – koliko ažurni moraju da budu podaci koji se koriste?
- *Bezbednosni zahtevi*

114. 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. Fizički model se ne razlikuje u odnosu na logički.
- *Optimizacija na nivou upita* – vrši se pisanje upita na način koji omogućava njihovo efikasnije izvršavanje. Fizički model se ne razlikuje u odnosu na logički.

- *Optimizacija na nivou strukture podataka* – fizička struktura podataka se menja u odnosu na logički model.

115. Koji su osnovni elementi fizičke organizacije podataka (na primeru SUBP DB2)?

Osnovni elementi su: prostori za tabele, stranice, baferi stranica, particionisane tabele i kompresija podataka.

116. Šta je prostor za tabele? Čemu služi?

Prostor za tabele je osnovni skladišni prostor. Svaka baza podataka mora da ima jedan ili više prostora za tabele. Prostor za tabele pripada jednoj bazi podataka. Jedan prostor za tabele može da sadrži više tabela. U particionisanim bazama podataka jedna tabela može da bude i u više prostora za tabele.
Prostor za tabele predstavlja logički nivo fizičke organizacije. Određuju se osobine i način upotrebe prostora za tabele. Ne određuje se neposredno gde se podaci nalaze.
Na nivou prostora za tabele definišu se: veličina stranice, veličina jedinice čitanja (u rasponu 2-256 stranica), da li se podaci čuvaju komprimovano ili ne, da li i kako može da se automatski povećava, način i uslovi baferisanja stranica.
Fizička lokacija skladištenja se određuje kao skup kontejnera – može da bude direktorijum, particija diska, disk, prealociran fajl ali i skladišna grupa.

117. Šta je stranica baze podataka? Čemu služi?

Stranica je osnovni element fizičkog zapisa tabele. Sve operacije se fizički odvijaju nad stranicama. Uvek se čita sa diska ili zapisuje na disk cela stranica. Svaka tabela i svaki indeks se sastoji od stranica, tj. svaka tabela se zapisuje u skupu stranica jednog prostora za tabele.
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; manje se traži po disku; manja je dubina indeksa; redovi tabele mogu da budu veći; uobičajeno za skladišta podataka i analitičke baze podataka.
Ako su stranice male: veća je iskorišćenost prostora na disku; manje je nepotrebnog čitanja i pisanja; uobičajeno za transakcione baze podataka.
Stranice se sastoje od zaglavlja i zapisa.

118. Šta je bafer za stranice? Čemu služi?

Bafer za stranice je memorijski prostor predviđen za čuvanje kopija jednog broja stranica radi omogućavanja bržeg pristupa podacima. Određuje se na nivou prostora za tabele. Što je bafer za stranice veći, to je broj pristupa disku manji.

119. Šta su katanci? Kako se i kada koriste? Šta je eskalacija katanaca?

Mehanizam katanaca je uobičajeno sredstvo ostvarivanja izolovanosti transakcija. Svaki katanac ima: objekat koji se zaključava, trajanje i vrstu katanca. Objekat može biti: atribut, red tabele, stranica tabele, grupa stranica, cela tabela, prostor za tabele, indeks. Veličina objekta određuje granularnost katanca.
Katanac može da zaključava jedan ili više redova (ili jednu ili više stranica). Veliki broj katanaca može značajno da uspori rad zbog mnogo više provera pri radu; zato je broj katanaca ograničen.
Kada se prevaziđe dopušten broj katanaca, dolazi do *eskalacije katanaca*. Tada se više manjih katanaca zamenjuje jednim većim ili se više katanaca na stranicama zamenjuje katancem na tabeli.

120. 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. Indeks je fizička struktura podataka koja se zapisuje u izabranom prostoru za tabele. Definicija indeksa obuhvata: vrstu indeksa, kolone koje čine uslov uređivanja tj. ključ pristupanja i ostala svojstva (da li je indeks jedinstven ili ne; da li je indeks jednosmeran ili dvosmeran; da li je uređujući ili ne; da li je particionisan ili nije).

121. Navesti poznate vrste indeksa i ukratko objasniti?

- *Jedinstveni indeksi* – ne dozvoljavaju ponavljanje više redova sa istim ključem. Koristi se I kao sredstvo za implementiranje integriteta ključa. Tri vrste su: zabranjene nedefinisane vrednosti, dozvoljene nedefinisane vrednosti (jednako) i dozvoljene nedefinisane vrednosti (različite)
- *Grupišući indeksi* – obezbeđuju da su redovi u tabeli poređani u odgovarajućem poretku. Najviše jedan ovakav indeks na tabeli. Ubrzavaju izdvajanje “podsekvence” redova u datom poretku. Usporavaju održavanje, ne menja se samo indeks nego i fizički zapis redova.
- *Particionisani indeksi* – particionisana tabela može da ima: neparticionisani indeks (ceo indeks u jednoj particiji) ili particionisani indeks (indeks se nalazi u više prostora za tabele)
- *Dvosmerni indeksi* – omogućavaju pretraživanje u oba smera. Za neke indekse to ne predstavlja veliku promenu u implementaciji.
- *Indeksi sa strukturom B-stabla* – podaci se čuvaju u balansiranom drvetu. Svaki red tabele je referisan iz lista jednake dubine. Čvorovi i listovi sadrže po više podataka.
- *Bit-mapirani indeksi* – indeks je organizovan kao niz vrednosti ključa. Uz svaku vrednost ključa sledi niz bitova, za svaki red tabele po jedan.
- *Heš indeksi* – indeks se implementira kao heš tabela sa datim ključevima. Računaju se heš vrednosti na osnovu ključnih atributa, a onda se pomoću dobijene vrednosti neposredno pristupa podacima.

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

Grupišući indeksi obezbeđuju da su redovi u tabeli poređani u odgovarajućem poretku. Najviše jedan ovakav indeks na tabeli. Ubrzavaju izdvajanje “podsekvence” redova u datom poretku. Usporavaju održavanje, ne menja se samo indeks nego i fizički zapis redova.
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 i ne ubrzava pristupanje pojedinačnim redovima.

123. Šta su indeksi sa strukturom B-stabla? Implementacija? Karakteristike? Prednosti i slabosti?

Podaci se čuvaju u balansiranom drvetu. Svaki red tabele je referisan iz lista jednake dubine. Čvorovi i listovi sadrže po više podataka. Jedinica organizacije indeksa je stranica. Veličina čvorova odgovara stranici prostora za tabele.
Prednosti – jednostavni i efikasni algoritmi za održavanje. Slabosti – nije posebno efikasan ako je mnogo redova a malo različitih vrednosti ključa.

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

*I*ndeks 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 imamo 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 i slabo efikasan; održavanje indeksa je često značajno skuplje nego u slučaju B-stabla.

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

Indeks se implementira kao heš tabela sa datim ključevima. Računaju se heš vrednosti na osnovu ključnih atributa, a onda se pomoću dobijene vrednosti neposredno pristupa podacima. Alternativa je da se pravi B-stablo a da se kao ključ koriste heš vrednosti.
Prednosti – alternativa klasičnim indeksima B-stabla je efikasnija za pristupanje pojedinačnim redovima sa tačno zadatim ključem.
Slabosti – nisu dobri za sekvencijalno pristupanje većem broju redova ili ako operator poređenja nije jednakost.

126. Kada pravimo indekse, zašto i koliko? Da li uvek moramo da imamo indekse?

U slučaju malih tabela, može da bude brže da se uvek pretraže svi redovi nego da se koriste indeksi.
Indeksi omogućavaju brži pristup konkretnim redovima tabele. Ako čitanje zahteva samo kolone sadržane u indeksu, onda tabeli ne mora ni da se pristupa.
Indeksi moraju da se održavaju. Zauzimaju prostor. Mogu da povećaju broj zaključavanja ili granularnost katanaca. Podižu cenu reorganizovanja ili prenošenja tabela.
Idealan broj i vrsta indeksa zavise od vrste, namene i strukture tabele i baze podataka, kao i načina upotrebe. Za transakcione tabele, 3-5 indeksa. Za analitičke tabele, bez ograničenja.

127. Šta su distribuirane baze podataka i distribuirani SUBP?

Distribuirana baza podataka – skup više logički međuzavisnih baza podataka koje su distribuirane na računarskoj mreži.
Distribuirani SUBP – softverski sistem koji omogućava upravljanje DBP tako da je distribuiranost transparentna za korisnika. SUBP je distribuiran akko je lociran na više različitih čvorova u mreži.

128. Koji su osnovni doprinosi distribuiranih baza podataka?

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

129. Objasniti šta znači transparentno upravljanje distribuiranim i repliciranim podacima?

Transparentnost podrazumeva razdvajanje semantike visokog nivoa od problema koji nastaju pri implementaciji na niskom nivou.

130. Koji su aspekti transparentnosti upravljanja distribuiranim i repliciranim podacima?

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

131. Objasniti nezavisnost podataka u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima.

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. Aplikacija ne sme da trpi nikakve posledice u slučaju promene fizičke strukture podataka, makar ona uključivala i promene u načinu distribuiranja podataka.

132. Objasniti mrežnu transparentnost u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima.

U centralizovanim bazama podataka jedini resurs o kome se stara transparentno 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.

133. Objasniti transparentnost replikacije u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima.

Replikacija je čuvanje istih podataka na više lokacija. Preduzima se radi performansi, raspoloživosti i pouzdanosti. Korisnici ne bi trebalo da budu svesni činjenice da se podaci repliciraju.

134. Objasniti transparentnost fragmentacije u kontekstu transparentnosti upravljanja distribuiranim i repliciranim podacima.

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.

135. 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
- *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

136. 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
- Globalni upiti se paralelizuju – podaci su distribuirani, pa se globalni upiti dele na manje, koji se odnose na lokalne podatke

137. Koji su osnovni otežavajući faktori pri implementaciji DBP? Objasniti.

- *Složenost* – distribuirani sistemi su sve samo ne jednostavni. DSUBP mora da reši sve probleme koje rešava, a centralizovan SUBP i još mnoge druge
- *Cena* – distribuirani sistemi zahtevaju dodatni hardver. Softverski aspekti sistema su mnogo složeniji i skuplji 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

138. Navesti najvažnije probleme i teme istraživanja u oblasti DSUBP.

- 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

139. 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) i homogenost strukture (jer struktura različitih baza podataka nije usaglašena). I tada je potrebno rešavati probleme kao što su: distribuirano izvršavanje upita, upravljanje metapodacima i upravljanje konkurentnošću.

140. Objasniti „izolovanje neispravnosti“ i „toleranciju neispravnosti“.

1. *Izolovanje naispravnosti* – obezbeđivanje da neispravnost jedne komponente ne utiče na funkcionalnost ostalih komponenti. Posledica: funkcija koju je obezbeđivala neispravna komponenta nije raspoloživa dok se problemi ne otklone i komponenta ne postane operativna.
2. *Tolerancija neispravnosti* – u slučaju neispravnosti neke od komponenti, druge komponente preuzimaju njene funkcije. Posledice: povećana fleksibilnost i pouzdanost sistema; povećana složenost i cena sistema.

141. Objasniti osnovne aspekte pouzdanosti sistema.

Pozdanost 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 uklanjaju
- *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

142. Objasniti kako se mere osnovni aspekti pouzdanosti sistema?

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: $R(t) = 1 – F(t)$,

R(t) je odgovornost u intervalu vremena dužine t, a F(t) je verovatnoća da će doći do greške u intervalu dužine t. $R(t)=1-\int_0^t f(x)\,dx$, f(t) je gustina verovatnoće neuspeha. Uobičajena mera je „srednje vreme do neuspeha“: MTTF = $R^{-1}(0.5)$, tj. vreme za koje je R(t)=0.5.
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*, gde je $0 < x < t$. Ova mera se naziva i „trenutna raspoloživost“. Uobičajena mera je i „srednje vreme opravke“ (MTTR) – očekivano trajanje popravljanja sistema nakon neuspeha. U praksi se često upotrebljava tzv. granična raspoloživost: $\lim_{t\to inf} A(t)=\frac{MTTF}{MTTF+MTTR}$. 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: $A(t)=1-\frac{\sum_i u_i}{t}$.

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

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

144. Objasniti vrste izvora replikacije.

Predmeti replikacije mogu biti: podaci, procesi, objekti, poruke. Nas prvenstveno zanima replikacija podataka (baza podataka, tabela, fragment tabele, globalni katalog baze podataka, sistem datoteka, pojedinačna datoteka).

145. Objasniti vrste replikacije?

Sinhrona i asinhrona.

146. Koji su osnovni ciljevi replikacije?

Dva osnovna motiva su podizanje performansi (ostvaruje se kroz raspodelu opterećenja na replike) i pouzdanosti (implementacija tolerancije neispravnosti i prepoznavanje neispravnosti).

147. Koja su ograničenja replikacije?

Osnovnu cenu replikacije predstavljaju:

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

Ograničenje efikasnosti - ako je ograničena raspoloživost jedne kopije A, onda je ograničena i raspoloživost repliciranog sistema.

148. Koji su osnovni načini implementacije replikacije podataka?

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

149. Šta je protokol ROWA? Kako se u osnovi implementira?

ROWA – “čitaj jedan, piši sve”. Samo primarna kopija se čita, a sve kopije se menjaju. Tolerišu otkaze lokacija kada je u pitanju čitanje. Ne tolerišu otkaze lokacija kada je u pitanju pisanje. Ne toloerišu otkaze komunikacija.

150. Šta je ROWA? Koje su osnovne varijante ovog protokola?

ROWA – “čitaj jedan, piši sve”. Varijante:

- osnovni ROWA protokol
- ROWA-A, sa menjanjem raspoloživih kopija
- ROWA sa primarnom kopijom
- ROWA sa tokenima pravih kopija

151. Objasniti detaljno 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, ali pisanje nije moguće do oporavka neispravne lokacije.

152. 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, a neispravne lokacije mogu da postanu raspoložive tek nakon oporavka uz puno prepisivanje svih izmena nastalih tokom neoperativnog perioda.
„Algoritam raspoloživih kopija“ – 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 se da li su sve kopije koje su propustile neko pisanje još uvek neoperativne
- „validacija pristupa“ – proverava da li su sve kopije koje su čitale ili pisale još raspoložive

153. 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.
Problemi: svakoj neispravnoj lokaciji se šalju bar dva zahteva 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.

154. Objasniti detaljno protokol ROWA sa primarnom kopijom.

Razlika u odnosu na osnovni protokol: jedna kopija se proglašava za „primarnu“; ostale kopije su „rezervne“; čitanje se izvodi samo sa primarne kopije; pisanje se izvodi na primarnoj i svim raspoloživim rezervnim kopijama. Ovaj protokol ne podiže performanse čitanja i unapređuje samo pouzdanost sistema.
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.

Ako rezervna kopija postane neoperativna, ona ne može postati primarna sve dok ne bude u potpunosti oporavljena i može preuzeti neke uloge rezervne i nakon delimičnog oporavka.

155. Koje su prednosti protokola ROWA sa primarnom kopijom u odnosu na osnovni protokol ROWA?

Ovaj protokol ne podiže performanse čitanja, unapređuje samo pouzdanost sistema. Najčešće upotrebljiv jer je jednostavan.

156. 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 deljivi token. „Uzimanje“ tokena obuhvata, po potrebi, i ažuriranje podatka.
Pri pisanju: kada kopija mora da izvrši operaciju pisanja, ona locira i uzima ekskluzivan token za konkretan podatak. Ako on ne postoji, ona locira i proglašava za neispravne sve deljive tokene za konkretan podatak i pravi novi ekskluzivan umesto njih.
Pri čitanju: kada kopija mora da izvrši operaciju čitanja, kopija locira deljivi token, 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.
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. To se postiže različitim mehanizmima.

157. 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.

158. Šta je konsenzus kvoruma? Po čemu se razlikuje od protokola ROWA?

Razlika: dopušta pisanje na podskupu operativnih lokacija; čitanja se izvode sa podskupa lokacija koji mora da ima neprazan presek sa kvorumom 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.

159. 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 mogu biti: statički (određeni glasanjem pri podizanju sistema) ili dinamički (ako lokacije mogu da se rekonfigurišu).

160. Koje su vrste konsenzusa kvoruma? U čemu je osnovna razlika među vrstama?

- KK sa uniformnom većinom
- KK sa težinskom većinom
- KK sa težinskom većinom za direktorijume
- Uopšteni KK za apstraktne tipove podataka
- Hibrid ROWA/KK

161. Objasniti detaljno konsenzus kvoruma sa uniformnom većinom.

Operacija čitanja ili pisanja uspeva akko 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; pisanje mora da se izvrši na većini kopija. Počiva na prepoznavanju eventualnih konflikata od strane upravljača konkurentnošću na pojedinačnim lokacijama.
Postiže se otpornost na neispravnosti: na lokacijama ili u komunikaciji.
Cena je relativno visoka i pri čitanju i pri pisanju. Bar polovina+1 lokacija mora učestvovati u svakoj operaciji kroz glasanje.

162. Objasniti detaljno konsenzus kvoruma sa težinskom većinom.

Predstavlja uopštenje algoritma KK sa uniformnom većinom:

- svaka kopija d_s podatka *d* dobija nenegativnu težinu tako da suma svih težina na jednom podatku bude *u*
- sam podatak *d* dobija prag čitanja *r* i prag pisanja *w*, tako da važi: $r+w>u$ i $w>u/2$
- kvorum čitanja (ili pisanja) podataka *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.
Svaka operacija pisanja *w[d]* se prevodu u skup pojedinačnih operacija pisanja $w[d_s]$ na svim kopijama u nekom kvorumu pisanja: čitanje svih kopija u kvorumu, čitanje njihovih verzija, povećavanje maksimalnog broja verzije, pisanje svih kopija sa novim brojem verzije.
Svaka operacija čitanja r*[d]* se prevodu u skup pojedinačnih operacija čitanja $r[d_s]$ na svim kopijama u nekom kvorumu čitanja: čitanje svih kopija, čitanje njihovih verzija, kopija sa najvećim brojem verzije predstavlja rezultat.
Zbog većinskog glasanja, u svakom kvorumu čitanja sigurno postoji tekuća kopija podatka.
Načelno se ne zahteva složen algoritam oporavka. Algoritam je fleksibilan.

163. Objasniti razlike između konsenzusa kvoruma sa uniformnom većinom i konsenzusa kvoruma sa težinskom većinom.

U slučaju ujednačenih vrednosti parametara *r* i *w* algoritam se svodi na KK sa uniformnom većinom.

164. Koje su specifičnosti konsenzusa kvoruma za direktorijume i apstraktne tipove podataka?

Direktorijum je preslikavanje prostora ključeva u prostor vrednosti. Primenjuje se i na nerelacione baze zasnovane na katalozima. Menja se granularnost operacija tako da se ne izvršavaju na „velikim“ direktorijumima nego na njihovim manjim delovima.
Apstraktni tipovi podataka – sličan problem kao za direktorijume.

165. 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 kroz: primenu metoda ROWA tokom pouzdanih operativnih perioda; primenu metoda KK u uslovima postojanja otkaza lokacija ili komunikacija; transakcije mogu da započnu rad u normalnom režimu i da tokom rada pređu u režim otkaza.
Postojanje otkaza se prepoznaje 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.

166. Šta je konsenzus kvoruma u uređenim mrežama? Motivacija?

Savremeni algoritmi replikacije pored tolerisanja neispravnosti stavljaju u prvi plan i optimizaciju troškova. Pokušava se postizanje smanjivanja broja lokacija koje odlučuju u glasanju uvođenjem logičke strukture u skup lokacija.

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

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

168. 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. Svakoj lokaciji se dodeljuje jedan kvorum koji ima neprazan presek sa svim kvorumima koji su dodeljeni drugim lokacijama. Transakcija mora da obezbedi konsenzus svih lokacija u kvorumu dodeljenom njegovoj matičnoj lokaciji. Kada lokacija s_i zatraži međusobno isključivanje, protokol traži konsenzus od kvoruma lokacija Q_i tako da važi:

- $Q_i \cap Q_j \neq \varnothing$ - za svaka dva kvoruma mora da postoji bar jedna zajednička lokacija, koja služi kao arbitar kada se neki članovi dvaju kvoruma ne slažu
- $s_i \in Q_j$ – lokacija koja postavlja zahtev to čini u odnosu na sebe, bez dodatnih poruka
- $|Q_i| = K$ – 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: $n = K(K\text{-}1) + 1$, sa K razmena poruka se može dogovoriti n lokacija.
Protokol postiže uzajamno isključivanje sa $c\sqrt{n}$ poruka ($3 \leq c \leq 5$).

169. Objasniti protokol GRID. Koje su mu osnovne karakteristike?

Skupovi lokacija se uređuju u obliku matrice sa N kolona i M vrsta, tako da je $M*N \leq n$. 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.
Skup lokacija G je C-pokrivanje ako svaka kolona ima neprazan presek sa G.
Operacija čitanja bira proizvolju permutaciju π_r koja se sastoji od M lokacija u proizvoljnom redu r. π_r određuje redosled komuniciranja radi postavljanja katanaca za čitanje na C-pokrivaču. 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 oslobađaju.
Operacija pisanja: najpre zaključava C-pokrivač koristeći protokol čitanja; zatim zaključava sve lokacije proizvoljne kolone c u redosledu određenom permutacijom π_c; da bi neka dva pisanja radila konkurentno, svako od njih mora da zaključa po C-pokrivač i jednu kolonu, pa se mora prepoznati konflikt.

170. Navesti i ukratko objasniti najvažnije ciljeve pri pravljenju distribuiranih sistema.

- Konzistentnost (C) – konzistentan sistem funkcioniše kao celina ili ne funkcioniše uopšte
- Raspoloživost (A) – odzivnost sistema u nekim garantovanim granicama
- Prihvatanje razdvojenosti (P) – nijedan skup problema, osim potpunog otkazivanja, ne sme da proizvede neispravan odziv sistema

171. Objasniti konzistentnost kao jedan od osnovnih ciljeva pri pravljenju distribuiranih sistema.

Konzistentan sistem funkcioniše kao celina ili ne funkcioniše uopšte. Sva čitanja moraju da daju isti rezultat. Rezultat operacije nikada ne zavisi od čvora na kome se izvršava.

172. Objasniti raspoloživost kao jedan od osnovnih ciljeva pri pravljenju distribuiranih sistema.

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

173. 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. Sistem mora da prihvata delimične otkaze komunikacije i da nastavlja ispravno funkcionisanje. Povremeni prekidi komunikacije među čvorovima su neizbežni. Razdvojenost je stanje komunikacione mreže u kome su delovi sistema podeljeni na particije između kojih ne postoji komunikacija. Očekuje se da sistem funkcioniše i daje ispravne rezultate čak i u uslovima razdvojenosti.

174. Navesti teoremu CAP i ukratko je objasniti.

“Nije moguće definisati sistem koji zadovoljava sve CAP uslove” (Eric Brewer, 2000).
Moguće je definisati sistem koji zadovoljava izabrana dva od ovih uslova. Ideja dokaza:
Pretpostavke – neka imamo dva čvora i na njima repliciran podatak V:

- neka na čvoru 1 transakcija A ažurira V
- ažuriranje se porukom M propagira na čvor 2
- neka na čvoru 2 nešto kasnije transakcija B čita V

Pretpostavimo da poruka M nije stigla na odredište zbog razdvojenosti delova sistema. U osnovi imamo tri moguća ponašanja sistema:

1. transakcija A se poništava – sistem ne prihvata razdvojenost svojih delova
2. transakcija A se uspešno nastavlja i zatim završava – sistem nije konzistentan (čitanje na čvorovima 1 i 2 će davati različite rezultate)
3. transakcija A čeka na uspešno slanje poruke M – sistem nije raspoloživ (ne može da garantuje vreme u kome će se transakcija završiti)

175. Koji vidovi kompromisa se prave radi prevazilaženja ograničenja koja proističu iz teoreme CAP?

Odbacivanje tolerancije razdvojenosti; odbacivanje raspoloživosti; odbacivanje konzistentanosti; ublažavanje uslova; zasnivanje sistema na drugačijem skupu uslova; projektovanje zaobilaznih puteva.

176. Objasniti “odbacivanje prihvatanja razdvojenosti” kao posledicu teoreme CAP.

Pristajemo da sistem ne radi u slučaju razdvojenosti. Jedan način prevazilaženja je da sve bude na jednoj mašini, ali ako već moramo da pravimo DBP, onda to verovatno nije prihvatljivo. Alternativa je da se razdvojenost svede na najmanju moguću meru pomoću višestrukog umrežavanja. Oba načina su veoma skupa. Ovaj vid kompromisa se retko pravi.

177. Objasniti “odbacivanje raspoloživosti” kao posledicu teoreme CAP.

U slučaju razdvojenosti ne garantuje se vreme odziva. Posledice problema se umanjuju 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.

178. Objasniti “odbacivanje konzistentnosti” kao posledicu teoreme CAP.

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.

179. 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 tj. postoji zavisnost sa osobinama transakcija – ACID. Ali, može da se definiše i drugačiji skup uslova, manje oštar, u kom slučaju sve odgovarajuće osobine mogu da se usklade – BASE.

180. 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.
- *Eventually consistent* - žrtvuju se garantovana stalna konzistentnost i izolovanost transakcija zarad raspoloživosti. Sistem će u nekom trenutku postati konzistentan ali će i pre tog trenutka raditi i davati odgovore.

Suština koncepta je u odloženom usklađivanju sadržaja na različitim čvorovima.

181. Objasniti specifičnosti projektovanja baze podataka u odnosu na uslove BASE.

U osnovi se svode na dve nove aktivnosti:

- funkcionalna dekompozicija podataka u fragmente
- implementacija transakcija prema BASE uslovima
- određivanje uslova replikacije

Dekompozicija – skup podataka se deli na fragmente: koji predstavljaju najmanje moguće funkcionalne celine, unutar kojih moraju da važe ACID uslovi, 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.

182. Objasniti pojam „konflikata“ pri projektovanju baze podataka na osnovu uslova BASE i načine njihovog razrešavanja.

Konflikti su osnovni vid problema kod svih vidova implementacije BASE skupa uslova. Postoje različite vrste konflikata. Osnovni oblik: ako se dve replike istog podatka nezavisno menjaju, pitanje je koja verzija je 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 particionisanja
- definisanje hijerarhija među redundantnim kopijama konkretnih vrsta podataka
- višestruke verzije podataka

183. Šta su „nerelacione baze podataka“?

U savremenom razvoju softvera termin nerelacione baze podataka se odnosi na sisteme za upravljanje kolekcijama podataka koje: nemaju strogu statičku strukturu podataka; nemaju iscrpnu proveru uslova integriteta; ne koriste upitni jezik SQL.
Nerelaciona baza podataka je baza podataka koja ne počiva na relacionom modelu podataka, lako se distribuira i horizontalno je skalabilna.

184. Objasniti motivaciju za razvijanje i korišćenje nerelacionih baza podataka?

Zbog slabosti relacionih baza podataka 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.

185. Koje osnovne slabosti RBP pokušavaju da se prevaziđu nerelacionim bazama podataka?

- Relativno visoka cena čitanja – zbog neredundantnosti i stroge strukture podataka
- Otežano distribuiranje – zbog „ultimativne“ konzistentnosti podataka
- Skupa promena strukture – zbog povezanosti strukture sa upotrebom i optimizacijom

186. Navesti osnovne vrste nerelacionih baza podataka i tipične probleme koji se njima rešavaju.

- 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 (memorijske baze podataka)
- Ogromne količine podataka niske složenosti (različite vrste matričnih baza podataka)

187. Navesti najčešće modele podataka nerelacionih baza podataka.

Baze parova ključeva i vrednosti, baze sa proširivim slogovima, baze dokumenata, grafovske baze podataka.

188. Baze parova ključeva i vrednosti – karakteristike, doprinosi, slabosti i primeri?

Karakteristike. Svaka kolekcija podataka je heš tabela. Podacima se pristupa isključivo na osnovu poznatog ključa. Vrednosti su u načelu jednostavne ili vrlo nisko strukturirane.
Doprinosi. Podržavaju veoma velike skupove podataka. Veoma su brze. Obično podržavaju automatsko repliciranje i horizontalno particionisanje kolekcija.
Slabosti. Podrazumeva se visok nivo redundantnosti. Složene strukture se implementiraju velikim brojem kolekcija. Ako su podaci gusto povezani, efikasnost drastično opada. U osnovi nemaju mehanizme za očuvanje integriteta. Ne mogu da se pretražuju po podacima. Uslov traženja je isključivo fiksna vrednost ključa ili raspon vrednosti ključa.
Primeri. Redis, Memcached, Hayelcast, Riak, Oracle NoSQL...

189. Baze sa proširivim slogovima – karakteristike, doprinosi, slabosti i primeri?

Karakteristike. U osnovi slično kao baze parova ključeva i vrednosti. Vrednost predstavlja kolekciju velikog broja parova imena i vrednosti.
Doprinosi. Podržavaju veoma velike skupove podataka. Veoma su brze. Većina podržava automatsko repliciranje i horizontalno particionisanje kolekcija.
Slabosti. Kao i kod baza parova ključeva i vrednosti.
Primeri. Cassandra, BigTable, Druid, Accumulo, HDFS (Hadoop Distributed File System), HBase

190. Baze dokumenata – karakteristike, doprinosi, slabosti i primeri?

Karakteristike. U osnovi liči na bazu parova ključeva i vrednosti. Svaki dokument predstavlja vrednost kome se dodeljuje ključ. Dokumenti se zapisuju u strukturiranim ili nestrukturiranim oblicima. Svaki dokument ima metapodatke. Telo dokumenta i metapodaci se automatski interno strukturiraju.
Doprinosi. Obično veoma jednostavan i efikasan rad. Obično podržavaju bar poluautomatsko repliciranje i horizontalno particionisanje kolekcija.
Slabosti. Relativno ograničen domen primene. Mnoge implementacije ne omogućavaju ad-hok upte i menjanja podataka. Neke implementacije ne trpe velike učestalost menjanja podataka.
Primeri. MongoDB, CouchDB, MarkLogic, RavenDB, Amazon DynamoDB, Google Cloud Datastore

191. Grafovske baze podataka – karakteristike, doprinosi, slabosti i primeri?

Karakteristike. 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. Veoma su efikasne u pitanju uobičajene operacije sa grafovima. Neke podržavaju transakcije i ACID režim uslova. Obično samo replikacija tip glavni-podređeni.
Slabosti. Relativno ograničen domen primene. Nisu pogodne van svog domena.
Primeri. Neo4J, OrientDB, ArangoDB, Allegro, Virtuoso, MS Azure Cosmos DB, Titan, GraphDB

192. Osnovne slabosti nerelacionih baza podataka?

Nemaju strogu statičku strukturu; nemaju iscrpnu proveru uslova integriteta; ne koriste upitni jezik SQL; složenost strukture podataka.

193. Nerelaciona baza podataka Apache Cassandra – osnovne karakteristike.

Apache Cassandra – nerelacioni SUBP. Primarno namenjen za distribuirane baze podataka. Dinamička shema. Počiva na proširenom modelu kataloga. Postepeno je izgrađen ozbiljan upitni jezik CQL. Počiva na modelu kataloga. Osnovni pojmovi su:

- *kolona* – jedna vrednost, praćena vremenom poslednje izmene (slično atributu kod RBP). Trojka *(name, value, timestamp)*
- *familija kolona* – struktura koja može da sadrži veći broj redova. Preslikava ključ u red. Red je skup kolona (sličan superkoloni). Blisko pojmu tabele kod RBP. Konceptualno predstavlja katalog redova.
- *superkolona* – složena vrednost, bez vremena poslednje izmene. Sadrži jednu ili više kolona
- *familija superkolona* -
- *ključ* -
- *prostor ključeva* – struktura koja sadrži više familija kolona ili superkolona. Odgovara pojmu baze podataka ili sheme kod RBP

Cassandra ima dva nivoa ugneždenosti – prvi nivo je obavezan, dok je drugi nivo opcion. Imena kolona predstavljaju istovremeno ključeve i vrednosti. Svaki red (ili superkolona) može da sadrži proizvoljno mnogo kolona. Svaka familija kolona je horizontalno particionisana. Particionisanje se vrši seckanjem po ključu, na odgovarajući broj particija.
Tipovi podataka: tekstualni, numerički, logički, kolekcije, univerzalni jedinstveni identifikator...