Projektovanje baza podataka

Odgovori na pitanja (sređeni i dopunjeni)

profesor: Saša Malkov

2018/2019

Zorana Gajić, 400/2016

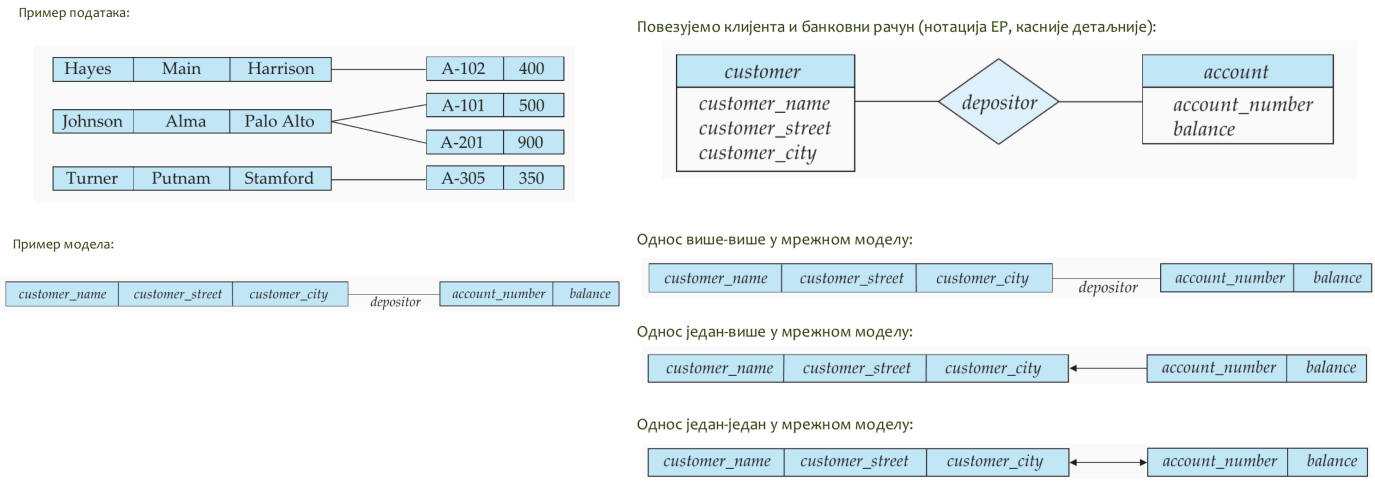
1. Navesti najvažnije modele podataka kroz istoriju računarstva do danas.

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

2. Objasniti osnovne koncepte mrežnog modela podataka

Mrežni model podataka je nalik na dijagramsko povezivanje podataka.

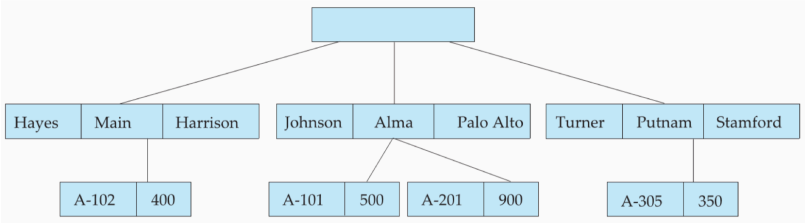
- Struktura podataka - nalik na slogove u programskim jezicima. Slog sadrži podatke jedne instance entiteta, sastoji se od polja.
- Strukture se povezuju «vezama» (link) - nalik na pokazivače.



3. Objasniti osnovne koncepte hijerarhijskog modela podataka

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

- Slično mrežnom modelu, ali se zahteva hijerarhija. Kod mrežnog sloja je 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 nivoa apstrakcije:

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

5. Objasniti uglove posmatranja arhitekture baze podataka

Referentni model i arhitektura se mogu posmatrati iz 3 ugla:

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

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

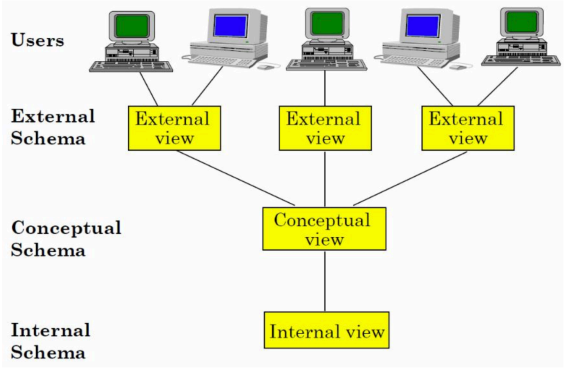
6. Objasniti standardizovanu arhitekturu ANSI/SPARC

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

Prepoznaju se 3 nivoa (pogleda) podataka:

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

- Interni nivo - kako su podaci implementirani na računaru. Obuhvata elemente fizičke implementacije.



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

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

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

8. Objasniti primer arhitekture klijent-server

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

Server je zadužen za upravljanje podacima:

- obrada upita
- optimizacija
- izvođenje transakcija.

Klijent (klijentski deo SUBP) je zadužen za:

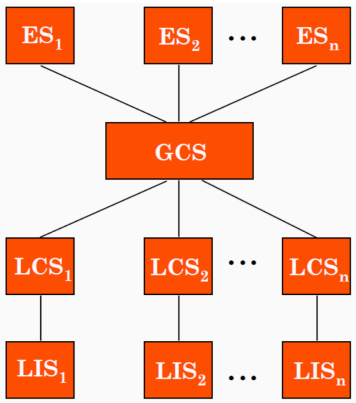
- ostvarivanje komunikacije između aplikacije i servera
- upravljanje podacima koji su keširani na strani klijenta - podaci, katanci, provera konzistentnosti transakcija.

9. Objasniti koncept distribuiranih arhitektura na primeru arhitekture ravnopravnih čvorova

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

Primer: arhitektura ravnopravnih čvorova, federativne baze podataka.

Arhitektura ravnopravnih čvorova:



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

Globalna konceptualna shema je zapravo unija svih lokalnih konceptualnih shema.

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

10. Objasniti osnovne koncepte relacionog modela podataka

Objedinjeno modeliranje:

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

Integritet se obezbeđuje ključevima.

Relacioni model čine:

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

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

Objašnjenje matematičke formulacije:

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

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

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

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

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

Imamo funkciju f_1, možemo da imamo i f_2 naprimer $prezime$, f_3 i tako dalje. Što znači da imamo skup funkcija. Svaka od tih funkcija ima svoj domen. Da bismo dobro opisali jedan skup entiteta potrebno nam je nekoliko takvih funkcija, zato ćemo napraviti jednu složenu funkciju $F(e) := (f_1(e), f_2(e), \ldots, f_k(e)) = r$. Ovo je opis jednog svojstva, modeliramo jedno svojstvo ime, $visinu$, … . Domen složene funkcije je $F : E \mapsto D_1 \times D_2 \times \ldots \times D_k$.

Za svaki entitet e, dobićemo neku n-torku.

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

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

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

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

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

Zašto se ovo naziva relacijom?

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

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

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

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

Relacija predstavlja odnos, a model je skup!

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

Strukturni deo relacionog modela - način modeliranja podataka.

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

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

Pitanje 19.

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

Pitanje 22.

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

Neka su nam date 4 kutije: crvena, plava, bela i zelena. U tim kutijama se nalaze lopte, kocke i pločice.
Binarna relacija *kutijaSadrži(K, P)* je zadovoljena ako kutija K sadrži predmet P.

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

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

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

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

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

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

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

kutijaSadrži	
BOJA	**SADRŽAJ**
plava	lopte
plava	kocke
zelena	pločice
crvena	kocke

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

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

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

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

  $A_i : E \mapsto D_i$
- Svaki atribut A_i ima jedinstven naziv t_i

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

Primer:

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

Skup svih atributa određuje funkciju:

$f(x) = (ime(x), prezime(x), osnovna\_plata(x))$

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

Skup svih atributa određuje funkciju:

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

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

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

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

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

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

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

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

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

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

Odnosno, relaciona baza podataka je skup relacija.

Opis relacije čine domen relacije i nazivi atributa.

Relaciona shema je skup opisa relacija koje čine bazu podataka.

Čitavu bazu podataka opisujemo jednom relacionom shemom.

Primer:

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

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

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

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

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

Shema

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

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

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

D - skup svih datuma

$Kol(ORG\_JEDINICA)$ = ('org_jed_id', 'nad_org_jed_id', 'naziv')

$Dom(ORG\_JEDINICA) = N(3) \times N(3) \times C(60)$

I slično za ostale relacije.

Primer sadržaja

ORG_JEDINICA		
ORG_ JED_ID	NAD_ORG _JED_ID	NAZIV
40	-	Softver
30	40	Projektovanje
11	10	Dokumentacija
10	40	Planiranje
50	40	Analiza
60	40	Dizajn i modeliranje
70	40	Kodiranje

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

(Iz pitanja 15)

Skup svih atributa određuje funkciju:

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

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

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

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

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

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

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

Osnovna ideja:

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

Ako postoji neki podskup atributa $A_{i_1}, \ldots, A_{i_{nk}}$ takav da postoji preslikavanje k koje za svaki entitet $e \in E$ jednoznačno preslikava izabrani podskup atributa $(x_{i1}, \ldots, x_{ink})$ entiteta e u kompletan model entiteta $e(x_1, \ldots, x_n)$ onda u modelu relacije taj skup atributa može da se koristi umesto punog skupa atributa $A_1, \ldots, A_n$.

19. Šta čini manipulativni deo relacionog modela?

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

Naprimer, «Koliko je studenata polozilo PBP?»

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

Najpoznatiji formalni upitni jezici u relacionom modelu su:

- Relaciona algebra
- Relacioni račun

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

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

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

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

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

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

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

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

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

20. Objasniti ukratko relacioni račun

Primena predikatskog računa na relacije.

Ekvivalentan relacionoj algebri.

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

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

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

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

$$\begin{aligned}\{(x.ime, x.prezime) \mid x \in Radnik \\ \wedge (\exists y \in Org\_Jedinica) \\ (y.naziv =' Planiranje' \wedge x.org\_jed\_id = y.org\_jed\_id) \\ \wedge (\exists z \in Kat\_Staza) \\ (z.naziv =' Srednja' \wedge z.od\_godine \leq staz(x) \leq z.do\_godine)\end{aligned}$$

21. Objasniti ukratko relacionu algebru

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

Osnovne dodatne operacije su:

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

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

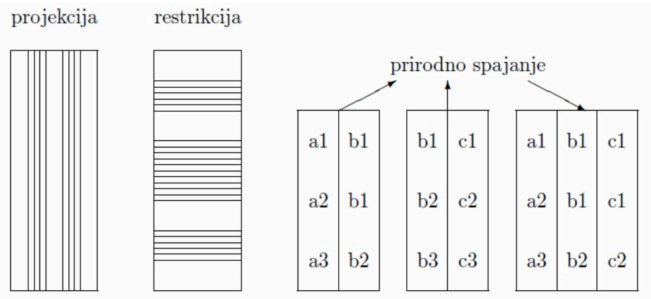
Kombinovanjem osnovnih operacija dobijaju se složeniji upiti, kao:

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

$$\{(x, y) | x \in Radnik \wedge y \in Org\_Jedinica \wedge x.org\_jed\_id = y.org\_jed\_id\}$$

- Slobodno spajanje - spajanje sa restrikcijom po slobodnom uslovu

$$\{(x, y) | x \in Radnik \wedge y \in Kat\_Staza \wedge y.od\_godine \leq staz(x) \leq y.do\_godine\}$$



22. Šta čini integritetni deo relacionog modela?

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

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

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

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

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

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

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

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

Formalnost modela olakšava formulisanje uslova integriteta.

23. Navesti osnovne vrste uslova integriteta u relacionoj bazi podataka

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

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

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

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

24. Objasniti integritet domena u relacionom modelu

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

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

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

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

25. Objasniti integritet ključa u relacionom modelu

Odnosi se na jednu relaciju.

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

Podskup X atributa $A_{i1}, \ldots, A_{ink}$ 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.

Ili prostije rečeno, kandidat ključ je skup atributa koji jednoznacno identifikuje redove u tabeli.

Jedna relacija može imati više ključeva.

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

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

- *X* funkcionalno određuje sve atribute relacije *R*

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

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

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

26. Objasniti integritet jedinstvenosti u relacionom modelu

Naziva se i «integritet entiteta».

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

U doslednim relacionim modelima:

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

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

27. Objasniti referencijalni integritet u relacionom modelu

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

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

Pravila referencijalnog integriteta:

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

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

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

(Ovde ne bismo smeli dodati studenta sa nepostojećim studentskim programom.)

U praksi se ostvarjue putem integriteta stranog ključa.

Ova pravila se menjaju ukoliko imamo nedefinisane ili nedostajuće vrednosti:

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

28. Objasniti integritet stranog ključa u relacionom modelu

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

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

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

Bazna relacija *B* se naziva i roditeljskom relacijom.

Ova pravila se menjaju ukoliko imamo nedefinisane ili nedostajuće vrednosti:

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

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

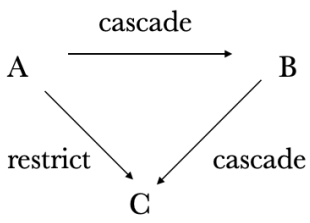
Poštovanje integriteta stranog ključa pri menjanju sadržaja baze podataka se određuju pravilima ažuriranja koja mogu da budu:

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

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

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

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



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

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

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

30. Objasniti opšte uslove integriteta relacionog modela

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

- Na atributu - Pri definisanju domena atributa mogu da se propišu i dodatni uslovi integriteta atributa.

  - To je uslov na atributu koji mora uvek da važi, obično vrlo lokalnog karaktera (odnosi se na vrednost jednog atributa jedne torke).

  - Uslov može da zavisi samo od vrednosti atributa.

  - Može da se koristi za dodatno sužavanje domena (naprimer, atribut mora da ima jednu od nekoliko vrednosti).

  - Može da se koristi za proveru ispravnosti složenih tipova podataka (naprimer, datum se predstavlja tekstom, ali se onda proverava da li tekstualni zapis predstavlja ispravan zapis datuma).

  Primer:

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

- Na torki - Pri definisanju relacije mogu da se propišu i dodatni uslovi integriteta relacije.

  - Lokalnog karaktera, odnosi se na vrednost jedne torke

  - Uslov može da zavisi od vrednosti svih atributa torke

  - Koristi se za proveru ispravnosti složenijiih saglasnosti atributa u okviru jedne torke (naprimer, ako se navodi neki interval, može da se proverava da li je početna vrednost intervala manja od krajnje vrednosti intervala).

  Primer:

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

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

  Primer:

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

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

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

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

31. Objasniti aktivno održavanje integriteta u relacionim bazama podataka

Osnovno sredstvo aktivnog održavanja integriteta su okidači.

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

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

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

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

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

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

Rekli smo da možemo da reagujemo pre ili posle naredbe (dodavanje, menjanje, brisanje torki).

Nije svejedno da li sve izvršava pre ili posle u slučaju brisanja i dodavanja! Ako izvršavamo posle brisanja, ne možemo da pristupamo starim vrednostima, jer su obrisane! A ako izvršavamo pre dodavanja, ne možemo da pristupamo novim vrednostima, još nisu ubačene.

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

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

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

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

Osnovno sredstvo razdvajanja nivoa shema u relacionom modelu su pogledi.

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

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

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

Savremeni SUBP nude okidače nad pogledima:

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

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

```
create trigger V1_update instead of update on V1
        referencing new as n old as o
        for each row mode db2sql
        update T1 set (c1, c2) = (n.c1, n.c2)
        where c1 = o.c1 and c2 = o.c2
```