Baze podataka – ispitna pitanja 2011. – 2014.

1. Navesti sve Specijalne relacione operacije relacione algebre. OBAVEZNO dati primere za svaku vrstu operacije.

Projekcija – operacija projekcije je unarna operacija koja iz neke relacije selektuje skup navedenih atributa, odnosno vadi vertikalni podskup iz odgovarajuće tabele. Formalno se definiše na sledeći način :

Neka je R(A1, A2, … , An) relacija, a X podskup njenih atributa. Označimo sa Y komplement {A1, A2, …, An} - X. Rezultat operacije projekcije relacije R po atributima X je

$$\Pi_X(R)=\{x \mid \exists\, y, \langle x,y\rangle \in R\}$$

$Pr_1 := \pi_{Ime, Starost}(Građanin)$

Građanin

MLB	Ime	Starost	MestoRod
16309723331981	Ana	19	Beograd
13975673331981	Mirko	21	Valjevo
11145276418976	Zoran	20	Beograd
23243723331981	Ana	19	Niš
22222223331981	Mirko	21	Beograd
11145276418976	Zoran	20	Novi Sad
23456786418976	Miloš	22	Beograd

Pr_1

Ime	Starost
Ana	19
Mirko	21
Zoran	20
Miloš	22

Selekcija – unarna operacija koja iz date relacije selektuje n-torke koje zadovoljavaju zadati uslov, vadi horizontalni podskup tabele. Formalno se definiše na sledeći način:

Data je relacija R(A1, A2, … , An) i predikat Θ definisan nad njenim atributima. Rezultat operacije selekcije:

$$\sigma\Theta(R) = \{x \mid x \in R \text{ AND } \Theta(x)\}$$

je skup n-torki x relacije R koje zadovoljavaju uslov Θ.

$Pr_2 := \sigma_{Starost > 20 \text{ AND } MestoRod = \text{"Beograd"}}(Građanin)$

Pr_2

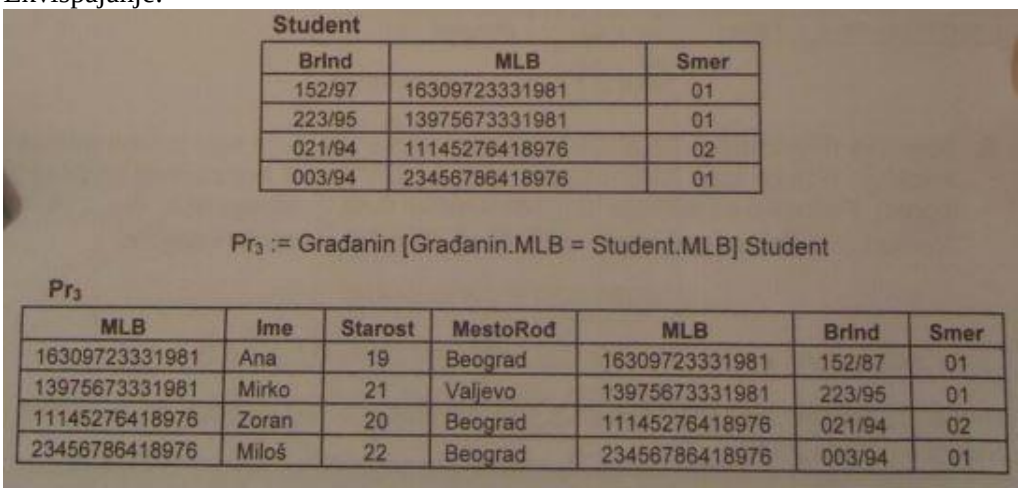
MLB	Ime	Starost	MestoRod
22222223331981	Mirko	21	Beograd
23456786418976	Miloš	22	Beograd

Spajanje (JOIN) – binarna operacija koja spaja dve relacije na taj način da se u rezultatu pojavljuju oni parovi n-torki jedne i druge relacije koje zadovoljavaju uslov zadat nad njihovim atributima. Dva tipa: ekvispajanje i prirodno spajanje.

Date su relacije R1(A1, A2, … , An) i R2(B1, B2, … , Bm) i predikat Θ definisan nad njihovim atributima. Obeležimo sa X i Y skupove atributa relacija R1 i R2, respektivno. Rezultat operacije spajanja ovih relacija je

$$R1[x\Theta]R2 = \{\langle x,y\rangle \mid x \in R1 \text{ AND } y \in R2 \text{ AND } \Theta(x,y)\}$$

Ekvispajanje:



Student

BrInd	MLB	Smer
152/97	16309723331981	01
223/95	13975673331981	01
021/94	11145276418976	02
003/94	23456786418976	01

Pr_3 := Građanin [Građanin.MLB = Student.MLB] Student

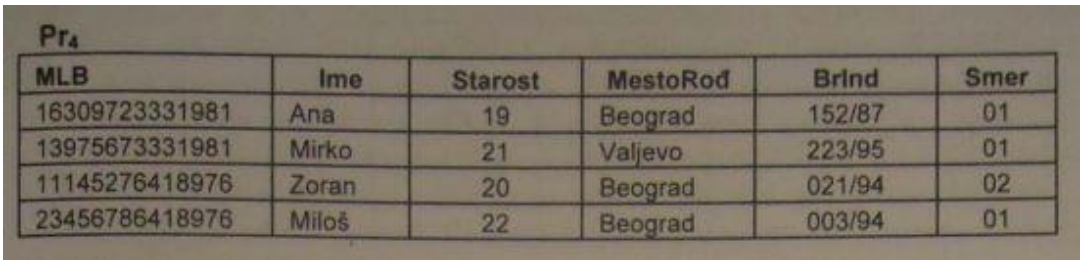
Pr_3

MLB	Ime	Starost	MestoRod	MLB	BrInd	Smer
16309723331981	Ana	19	Beograd	16309723331981	152/87	01
13975673331981	Mirko	21	Valjevo	13975673331981	223/95	01
11145276418976	Zoran	20	Beograd	11145276418976	021/94	02
23456786418976	Miloš	22	Beograd	23456786418976	003/94	01

Očigledno je da se u rezultatu ekvispajanja uvek pojavljuju dve iste kolone(MLB). Ako se jedna od te dve kolone izbaci, takvo spajanje se naziva prirodno spajanje. Prirodno spajanje podrazumeva i da su atributi po kojima se relacije spajaju istoimeni.

Prirodno spajanje:

Pr4:= Gradjanin * Student



Pr_4

MLB	Ime	Starost	MestoRod	BrInd	Smer
16309723331981	Ana	19	Beograd	152/87	01
13975673331981	Mirko	21	Valjevo	223/95	01
11145276418976	Zoran	20	Beograd	021/94	02
23456786418976	Miloš	22	Beograd	003/94	01

Deljenje – operacija pogodna za upite u kojima se javlja reč „svi“(„sve“, „sva“).

Neka su A(X,Y) i B(Z) relacije gde su X, Y i Z skupovi atributa takvi da su Y i Z jednakobrojni, a odgovarajući domeni su im jednaki. Rezultat operacije deljenja:

A[Y ÷ Z]B=R(X)

gde n-torka x uzima vrednosti iz A.X, a par <x,y> postoji u A za sve vrednosti y koje se pojavljuju u B(Z).

Predmet

SifPred
P1
P2
P3

Prijava

BrInd	SifPred
152/97	P1
152/97	P2
021/94	P1
003/94	P3
152/97	P3

Prijava [Prijava.SifPred ÷ Predmet.SifPred] Predmet

BrInd
152/97

2. Objasniti šta je pogled. Koje su osnovne prednosti u korišćenju pogleda? Dati uslove koje treba da ispuni pogled da bi mogao da posluži za ažuriranje baze.

Pogled je virtuelna tabela, "prozor" kroz koji se vide podaci baze podataka , koristi se kao bilo koja druga tabela pri izveštavanju, nema svoje podatke i ne zauzima nikakav memorijski prostor.

Prednosti: **jednostavnost korišćenja**, **tajnost**, **performanse**, **nezavisnost podataka**

Uslovi: - Pogled mora biti definisan nad jednom tabelom

- U definiciju pogleda moraju biti uključene sve NOT NULL kolone te tabele
- Kolone pogleda moraju sadržati samo prost , neizmenjen sadržaj kolona tabele nad kojom je pogled definisan

3. Nasleđivanje u objektnim bazama podataka. Navesti primere i objasniti ih.

U ODMG se definišu dve posebne vrste nasleđivanja: nasleđivanje ponašanja i nasleđivanje stanja. Za nasleđivanje ponašanja se koristi veza nadtip-podtip (koja se po nekad naziva veza generalizacija- specijalizacija). Nadtip u nasleđivanju ponašanja mora da bude interfejs. Za nasleđivanje stanja koristi se specifična veza EXTENDS (Proširenje). U ovoj vezi između klasa, podređena klasa nasleđuje celokupno stanje i ponašanje klase koju proširuje.

1. Nasleđivanje ponašanja. Činjenica je da je Nastavnik podtip Radnik-a, a Asistent podtip Nastavnika, što se označava na sledeći način:

```
intrerface Radnik {...specifikacija tipa Radnik..}
interface Nastavnik: Radnik {...specifikacija tipa Nastavnik..}
interface Asistent: Nastavnik {...specifikacija tipa Asistent..}
class StalniRadnik: Radnik {...specifikacija tipa StalniRadnik..}
class PrivremeniRadnik:Radnik {...specifikacija tipa PrivremeniRadnik..}
```

U podtipu se može i redefinisati ponašanje specifikovano u nadtipu. Na primer, ako je u tipu Radnik specifikovana operacija ObračunZarade, ista operacija se može različito implementirati za podtipove StalniRadnik i PrivremeniRadnik. Osobina da se ista operacija izvrpava na različit način u zavisnosti od tipa objekta nad kojim se izvršava, naziva se polimorfizam.

U ODMG modelu podržano je višestruko nasleđivanje ponašanja. Tada se, međutim, može desiti da dve ili više nasleđenih operacija imaju isti naziv, a različit broj i/ili tip argumenata. Da bi se ovaj problem izbegao, nije dozvoljeno „preopterećenje" (overloading) imena operacija (davanje istih imena operacijama različitih tipova).

2. Nasleđivanje stanja

```
class Lice {
attribute string ime;
attribute Date datumRođenja;
```

};

class ZaposlenoLice extends Lice: Radnik { attribute Date datumZapošljavanja; attribute Carency plata;

relationship Rukovodilac šef inverse Rukovodilac ::podređeni;

};

class Rukovodilac extends ZaposlenoLice {

relationship set <ZaposlenoLice> podređeni inverse ZaposlenoLice šef;

};

Odnos nadtip-podtip (generalizacija-specijalizacija) u ODMG modelu se primenjuje samo na nasleđivanje ponašanja. Zbog toga interfejs i klasa mogu biti podtipovi samo interfejsa. Interfejsi i klase ne mogu biti podtipovi klasa. Pošto se interfejs ne može instancirati, on služi samo sa to da se iz njega naslede neke operacije.

4. Navesti sve dodatne operacije relacione algebre koje su uvedene zbog postojanja nula vrednosti u BP i OBAVEZNO dati primere.

Tablica istinitosti trovrednoste logike

AND	T	?	F
T	T	?	F
?	?	?	F
F	F	F	F

OR	T	?	F
T	T	T	T
?	T	?	?
F	T	?	F

NOT	
T	F
?	?
F	T

? – predstavlja nula vrednost, odnosno logičku vrednost možda

MožđaSelekcija (MAYBE_SELECT) – selektuju se one n-torke relacije za koje se predikat selekcije, na osnovu trovrednosnih tablica istinitosti sračunava u nula vrednost.

R1

A	B
a	b
a	?
?	b
?	?

R2

A	B
?	b
?	?

R2 := ?sigmaA (R3)

Rezultat operacije je (selektovane su samo oni redovi koji u A koloni imaju nula vrednost)

Možda Spajanje (MAYBE_JOIN) – u rezultatu spajanja se pojavljuju one n-torke za koje se predikat spajanja sračunava u nula vrednost na osnovu trovrednosnih tablica istinitosti.

Rb

A	B
a1	b1
a2	b2
?	b3

Rc

C	D	E
c1	?	e1
c2	d2	e2
c3	?	e3

Ra := Rb[?A =?D]Rc

A	B	C	D	E
a1	b1	c1	?	e1
a1	b1	c3	?	e3
a2	b2	c1	?	e1
a2	b2	c3	?	e3
?	b3	c1	?	e1
?	b3	c2	d2	e2
?	b3	c3	?	e3

SpoljnaUnija (OUTER_UNION) – dodavanjem novih atributa i postavljanjem njihovih vrednosti na nula vrednost mogu se dve nekompatibilne relacije učiniti kompatibilnim. Spoljna unija podrazumeva da su na ovaj način dve nekompatibilne relacije učinjene kompatibilnim i da je zatim izvršena operacija unije.

R1

A	B	C
a1	b1	c1
a2	b1	c2

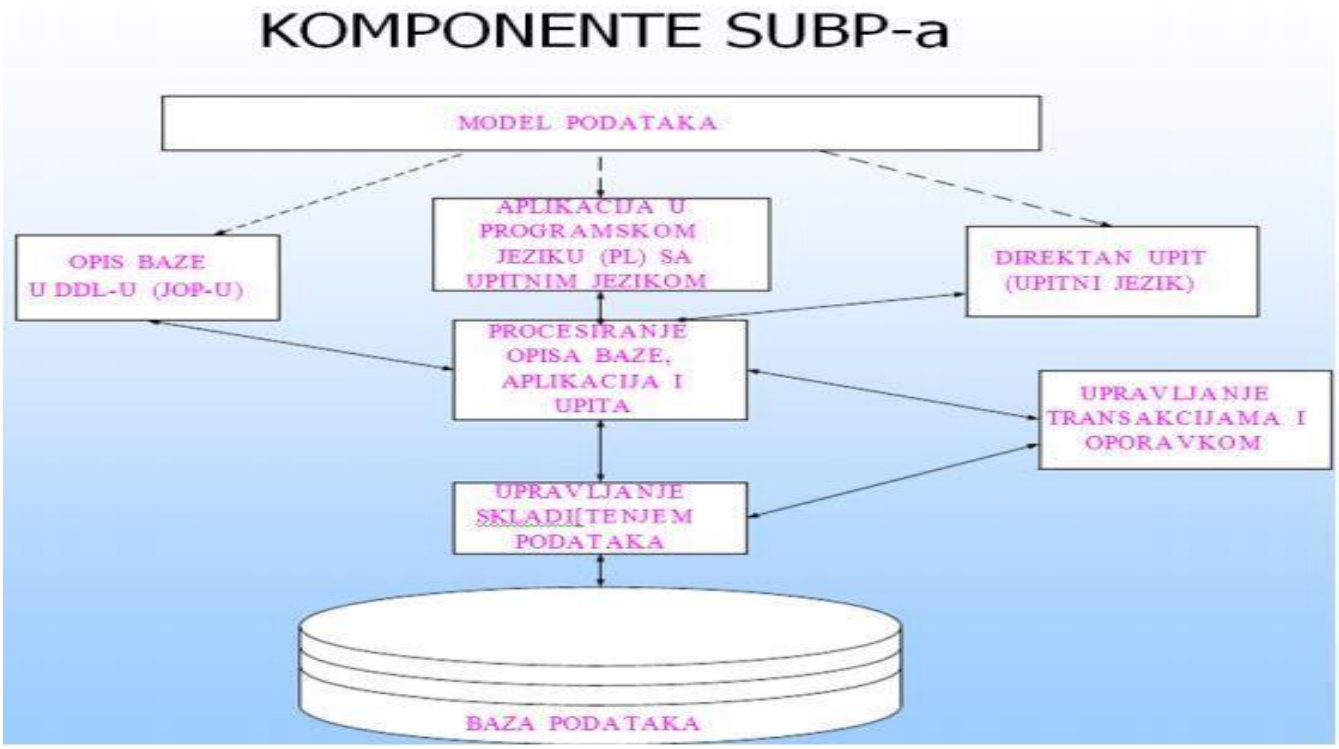
R2

A	D
a1	4
a1	7
a2	5

R3 := R1 OUTER_UNION R2

A	B	C	D
a1	b1	c1	?
a2	b1	c2	?
a1	?	?	4
a1	?	?	7
a2	?	?	5

5. Prikazati šemu komponenti SUPB i opisati ulogu komponenti



Baza podataka – njena primarna uloga je čuvanje podataka. Na njoj se nalazi i rečnik podataka, tj. metapodaci (podaci o samoj bazi) kao što su: struktura baze, prava korišćenja i pravila očuvanja integriteta. Jedan deo baze obuhvata i baza indeksa (indeks – struktura podataka koja omogućava brz pristup indeksiranim podacima).

SUBP – upravljanje datotekama i upravljanje baferima. Upravljanje datotekama vodi računa o lokaciji datoteka na bazi podataka i o pristupu blokovima podataka na zahtev upravljanja baferima. Blokovi podataka su kontinualni segmenti memorije veličine od 4000 do 16000 bajta. Svaki disk je obično podeljen na nekoliko blokova podataka. Upravljanje baferima nakon pristupa određenom bloku prihvata taj blok sa diska, dodeljuje mu stranicu centralne memorije i zadržava ga tu u skladu sa određenim algoritmom. Nakon toga oslobađa tu stranicu i braća blok na taj disk.

Ulazi u SUBP – upiti, aplikacije i upravljanje šemom baze podataka. Ulazi u SUBP realizovani su preko nekog jezika baze podataka, koji je zasnovan na nekom modelu podataka. Jezici baze podataka su: jezik za opis podataka (Data Definition Language - DDL) koji se koristi za realizaciju održavanja šeme baze podataka i jezik za manipulaciju podacima (Data Manipulation Language - DML) preko koga se vrši pretraživanje baze preko upita i modifikovanje baze podataka.

Upiti – omogućavaju pretraživanje baze podataka preko zahteva za podacima iz baze, sa uslovima koje ti podaci moraju da ispune. Preko upita je moguće i menjati sadržaj baze.

Aplikacije – programi pomoću kojih se realizuje prethodno definisani zahtev za podacima iz baze, izgrađuju se korišćenjem standardnih programskih jezika u koje se ugrađuje jezik baze podataka. Jezik baze podataka omogućava pristup, pretraživanje i menjanje podataka u bazi, a standardni programski jezik realizaciju složenijih algoritama.

Održavanje šeme baze podataka – podrazumeva kreiranje i modifikaciju šeme baze podataka (koja obuhvata opis strukture, prava korišćenja i pravila očuvanja integriteta baze).

6. Operacije u modelu objekti - veze

-**Ubacivanje** (Insert) novog pojavljivanja objekta u klasu,
-**Izbacivanje** (Delete) pojavljivanja objekta iz klase,
-**Ažuriranje** (Update) odnosno izmena vrednosti nekog atributa datog pojavljivanja objekta neke klase,
-**Povezivanje** (Connect) pojavljivanja O1 klase A sa pojavljivanjem O2 klase B,
-**Razvezivanje** (Disconnect) pojavljivanja O1 klase A od pojavljivanja O2 klase B,
-**Prevezivanje** (Reconnect) pojavljivanja O1 klase A os pojavljivanja O2 klase B.

7. Spoljno spajanje. Objasniti svaku vrstu SS i OBAVEZNO dati primer

Spoljno spajanje se koristi da bi se u rezultat spajanja uključili i oni redovi koji ne zadovoljavaju uslov spajanja. Može biti levo, desno i centralno.

SifraOdeljenja	NazivOdeljenja
1	Uprava
2	Racunarski centar
3	Prizvodnja

SifraRadnika	ImePrezime	SifraOdelj
1000	PP	1
1001	MM	1
1002	ZZ	2
1003	LL	2
1004	RR	

1) **Levo spoljno spajanje** – omogućava uključivanje u rezultujuću tabelu svih redova tabele sa leve strane JOIN klauzule tako što se praznim redom proširuje tabela sa desne strane.

   odeljenje LEFT OUTER JOIN radnik USING (**SifraOdeljenja**)

SifraOdeljenja	NazivOdeljenja	SifraRadnika	ImePrezime	SifraOdelj
1	Uprava	1000	PP	1
1	Uprava	1001	MM	1
2	Racunarski centar	1002	ZZ	2
2	Racunarski centar	1003	LL	2
3	Proizvodnja	?	?	?

2) **Desno spoljno spajanje** – omogućava uključivanje u rezultujuću tabelu svih redova tabele sa desne strane JOIN klauzule tako što se praznim redom proširuje tabela sa leve strane.

   odeljenje RIGHT OUTER JOIN radnik USING (SifraOdeljenja)

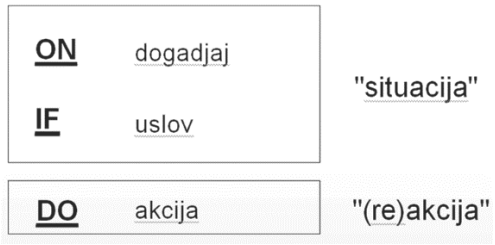
SifraOdeljenja	NazivOdeljenja	SifraRadnika	ImePrezime	SifraOdelj
1	Uprava	1000	PP	1
1	Uprava	1001	MM	1
2	Racunarski centar	1002	ZZ	2
2	Racunarski centar	1003	LL	2
?	?	1004	RR	?

3) **Centralno spajanje –** omogućuje uključivanje u rezultujuću tabelu svih redova i leve i desne tabele tako što se obe tabele proširuju praznim redom.

SifraOdeljenja	NazivOdeljenja	SifraRadnika	ImePrezime	SifraOdelj
1	Uprava	1000	PP	1
1	Uprava	1001	MM	1
2	Racunarski centar	1002	ZZ	2
2	Racunarski centar	1003	LL	2
3	Proizvodnja	?	?	?
?	?	1004	RR	?

8. Semantika i obrada ECA pravila

Svako pravilo se odnosi na određeni događaj,vrši proveru uslova po nastanku događaja i izvršava definisanu akciju ako je uslov bio zadovoljen. Pravila u aktivnim bazama podataka su poznata pod imenom ECA (Event-Condition-Action) produkciona pravila i specifikuju se na sledeći način:



Da bi se pravilo moglo formirati prethodno moraju biti definisani događaji. Generalno govoreći, moguće je razlikovati primitivne i složene događaje.

Primitivni događaji su:
• Ažuriranje podataka
• Prikaz podataka
• Vreme
• Aplikativno definisan događaj
Akcije se pokreću neposredno nakon (posle) ili neposredno pred (i ranije) detektovanja događaja.

Složeni događaji se formiraju kombinovanjem primitivnih i prethodno definisanih složenih događaja.

Korisni operatori za kombinovanje događaja mogu biti:
• Logički operatori. Događaji mogu biti kombinovani korišćenjem logičkih operatora AND, OR, NOT (I, ILI, NE, itd.).
• Sekvenca. Pravilo može biti pokrenuto kada se dva ili više događaja pojave u određenom, definisanom redosledu.
• Vremenska kompozicija. Pravilo može biti pokrenuto kombinovanjem vremenskih i događaja koji nisu vremenski, na prajmer "5 sekundi nakon događaja D1" ili "Svaki sat nakon prvog pojavljivanja događaja D2".

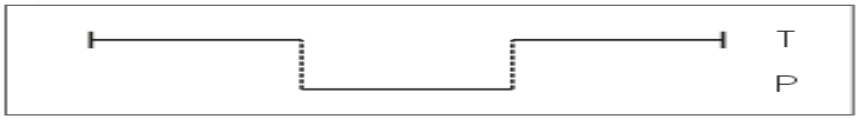
Uslov može biti predikat nad stanjem baze podataka, upit nad bazom podataka ili izvršavanje određene aplikativne procedure. U poslednja dva slučaja uslov je zadovoljen ako upit, odnosno aplikativna procedura vraća neke podatke.

Akcija je bilo koji program. Takav program može da uključuje i operacije nad bazom podataka i implicitno ili eksplicitno generiše nove događaje.

Postoji nekoliko različitih načina povezivanja transakcije, odnosno događaja koji pokreće pravilo i izvršavanja ECA-pravila:

Trenutan način

Prouzrokuje prekid izvršavanja transakcije odmah po identfikovanju događaja za koji postoji definisano ECA-pravilo, nakon čega se to pravilo proverava .

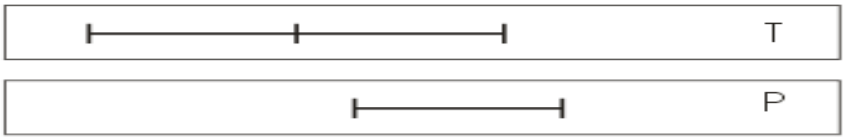


Odložen način

Izrvšavanje pravila se odlaže do završetka transakcije koja je proizvela događaj.

Razdvojen način

Pravilo postaje sasvim nova transakcija koja se izvršava nezavisno od polazne.



9. Navesti, opisati i dati primer za složene konstruisane tupove u objektno- relacionom modelu

- **Referentni tip** – tabele definisane direktno nad struktuiranim tipovim mogu da imaju referentnu kolonu, koja služi kao identifikator n-torki. Predstavlja preferencu na neki objekat, tabelu.

  Primer: REF IS<naziv atributa><nacin generisanja>

- **Tip vrsta** – je niz polja koji čine parovi (<naziv podataka>,<tip podataka>). U SQL: 1999 je moguće definisati kolonu u tabeli koja će imati kompleksnu strukturu. Tip vrsta se specificira sledećom sintaksom: ROW(ulica CHAR(30), broj INTEGER, grad CHAR (20))

  Primer: CREATE TABLE student(br_indeksa CHAR(6), ime CHAR(15), prezime CHAR(15), adresa ROW(ulica CHAR(30), broj INTEGER, grad CHAR(20)));

- **Kolekcija** – Predstavlja grupu koja se sastoji od nula ili više elemenata istog tipa. Broj elemenata kolekcije se naziva kardinalnost kolekcije. Tip kolekcije je određen vrstom kolekcije i tipom elemenata kolekcije. SQL:1999 standard obezbeđuje samo jednu vrstu kolekcije – niz, dok je standardizacija drugih vrsta ostavljena za naredne generacije jezika. Formalna definicija niza:

<tip elemenata niza>ARRAY[<maksimalna kardinalnost niza>]

Primer: CREATE TABLE sekcija (naziv CHAR (15),
clan CHAR (20), ARRAY[20]);
INSERT INTO sekcija (naziv,clan)
VALUES (‘dramska, ARRAY [‘Markovic’,’Popovic’]);
SELECT naziv, clan[2] AS ime FROM sekcija;

10. Objasniti i dati primer za sledeće OQL koncepte:

- Definisanje (iteratorske) promenljive,
- OQL putanja,
- Struktura kao rezultat izvršavanja SQL upita

Kad god se kolekcija referencira u OQL-u, neophodno je nad njom definisati promenljivu (tzv. „iteratorska promenljiva“) koja uzima vrednost iz te kolekcije. Ova promenljiva se definiše na isti način kao u SQL-u, u „FROM“ delu upita. Uz „SELECT“ deo se opisuje rezultat, a „WHERE“ definiše uslov za selekciju elemenata kolekcije. Promenljivu x je moguće definisati na sledečće načine:

Studenti x
x in studenti studenti as x

Za konstrukciju upita koristi se i tzv. izraz putanje (Path Expression). Ovaj izraz omogućava da se, polazeći od neke ulazne tačke u bazu, dospe do željenog objekta ili atributa. Izraz putanje je istovremeno i upit. Izraz putanje se definiše korišćenjem tzv. dot notacije.

Primer: odsekZaIS.sef; - vraća objekat tipa Nastavnik,
odsekZaIS.imenast; - vraća literal tipa string,
odsekZaIS.rade; - vraća objekat tipa set<Nastavnik>
odsekZaIs.rade.imenast; - nije dozvoljeno, jer deo odseka ZaIS.rade ove putanje vraća kolekciju

Upit u OQL može da vrati kao rezultat i kompleksnu strukturu koja se definiše u samom upitu. Na primer:

```
select struct (OsnovniPodaci: struct(BrojInd: x.bi, ImeStudenta: x.ime),
Ocene: (select struct (NazivPredmeta: y.pred.nazivpr, Ocena: y.ocena)
        from x.polozio y),
SrednjaOcena: x.sred_ocena)
from studenti x;
```

Prethodni upit daje kao rezultat strukturu koja predstavlja nenormalizovanu tabelu sledećeg oblika:

OsnovniPodaci		Ocene		SrednjaOcena
BrojInd	ImeStudenta	Predmet	Ocena	

10. a) Navesti i objasniti osnovne elemente grafa prethođenja transakcija, dati definiciju kada Ti predhodi Tj
b) Proveriti da li postoji konflikt serijabilnost izvršenja S1 skupa transakcija a posle i S2.
Dati obrazloženje.

S1: r2(A), r2(B), w2(A), r1(B), w1(B), r3(A), w3(A), w3(B) S2:
r2(A), r2(B), r1(B), w1(B), r3(A), w3(A), w3(B), w2(A)

Graf prevođenja transakcija se sastoji od čvorova koji predstavljaju transakcije i usmerenih grana koje prikazuju prethođenje transakcija. Kaže se da transakcija Ti predhodi transakciji Tj(Ti<STj) u izvršenju S ako postoji operacija Oi transakcije Ti i operacija Oj transakcije Tj tako da je:

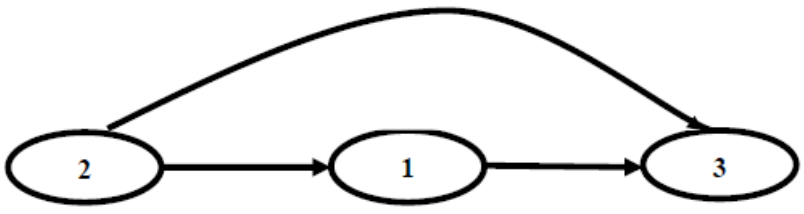
- Oi prethodi Oj u S
- i Oi i Oj se odnose na isti element baze podataka
- barem jedna od operacija Oi i Oj je operacija upisivanja

Ako postoji ciklus u grafu tada izvršenje nije moguće učiniti serijabilnim (nije konflikt -

serijabilno)

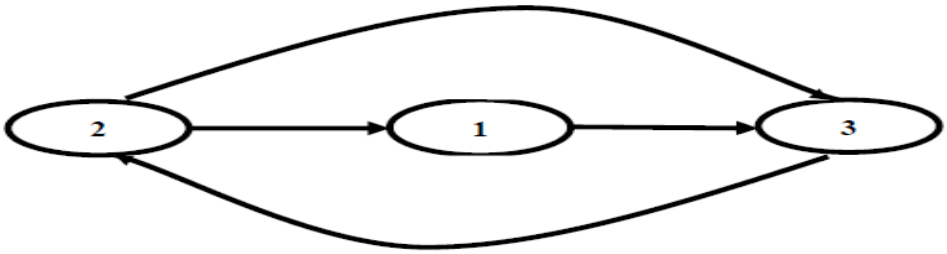
b) **S1**: T2 predhodi T3 (operacije 3 i 6), T2 predhodi T1 (operacije 2 i 5), T1 predhodi T3 (operacije 5 i

8)



Pošto na grafu nema ciklusa izvršenje **jeste konflikt serijabilno**.

S2: T2 predhodi T3 (operacije 1 i 6), T3 predhodi T2 (operacije 6 i 8), T2 predhodi T1 (operacije 2 i
4), T1 prethodi T3 (operacije 4 i 7)



Pošto na grafu ima ciklusa izvršenje **nije konflikt serijabilno.**

*Objašnjenje:

Primer: S2: r3(X), r3(Y), r1(Y), w1(Y), r2(X), w2(X), w2(Y), w3(X)
Prvo odvojim na jednu stranu sve sa X-om a na drugu sve sa Y. Dakle:

1) r3(X) r2(X) w2(X) w3(X) *2) r3(Y) r1(Y) w1(Y) w2(Y)*

Sada pravim kombinaciju svakog sa svakim, s tim sto se r i r ne mogu kombinovati:

1) r3(X) w2(X) 3→2
r3(X) w3(X) /
r2(X) w2(X) /
r2(X) w3(X) 2→3
w2(X) w3(X) 2→3

2) r3(Y) w1(Y) 3→1
r3(Y) w2(Y) 3→2
r1(Y) w1(Y) /
r3(Y) w2(Y) 3→2
w1(Y) w2(Y) 1→2

Kada su indeksi 1-1, 2-2, 3-3 piše se ova /, tj. to se ne gleda. Ovde se već vidi da ovo nije konflikt-serijabilno jer ce postojati ciklus (postoji i 3→2 i 2→3).

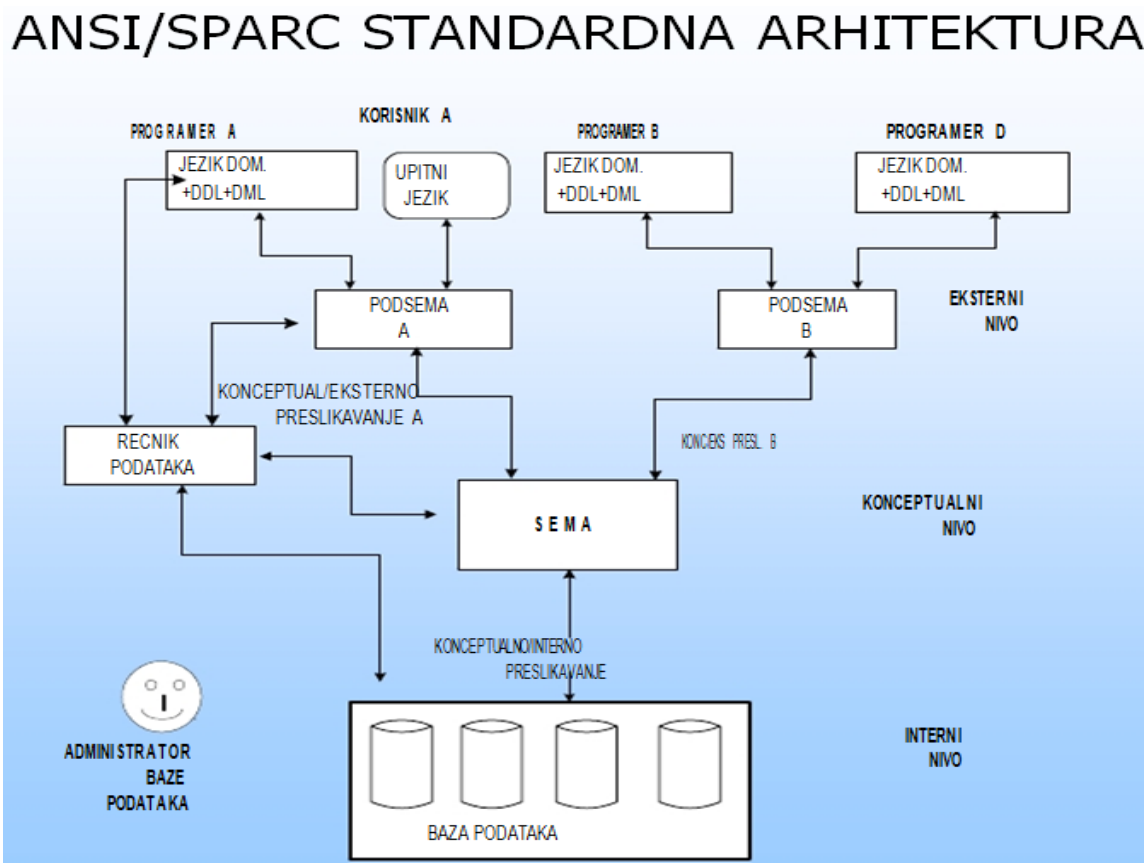
Pošto 3 prethodi 2, 3 prethodi 1 i 1 prethodi 2 cvorovi ce pri crtanju biti poredjani 3 - 1 - 2. I onda samo strelicama povezati čvorove, od 3 ka 2, od 2 ka 3, od 3 ka 1 i od 1 ka 2.

Da ciklus nije postojao, onda bi bilo konflikt-serijabilno!

11. Nacrtati sliku i objasniti ANCI|SPARC tronivoosku arhitekturu

ANSI/SPARC je tronivoska arhitektura u kojoj pojedini nivoi imaju za cilj da, s jedne strane, učine nezavisnom logičku od fizičke strukture baze podataka, a sa druge, aplikacione programe i od fizičke i od logičke strukture baze. Nivoi u ovoj arhitekturi su:

- **Interni (fizički) nivo**, koji definiše način na koji su podaci fizički organizovani na spoljnim memorijama,
- **Konceptualni nivo (šema baze podataka)**, koji definiše opštu logičku strukturu baze podataka i treba da omogući upravljanje podacima kao zajedničkim resursom u celom sistemu
- **Eksterni (korisnički) nivo**, na kome se definiše logička struktura podataka pogodna za programe.



12. Poslovna pravila integriteta relacionog modela. Dati za svaki podtip ovog ograničenja primer

- **Pravila integriteta za domene**, preko kojih se specifikuje koje vrednosti postoje u domenu.
Postoje dve vrste domena – predefinisani i semantički. Svodi se na definisanje semantičkog domena.

Primer: CREATE DOMAIN kol INTEGER
FORALL kol (kol > 5000 AND kol < 50000 AND MOD (kol, 50) = 0);

-**Pravila integriteta za atribute**, preko kojih se definišu dozvoljene vrednosti nekog atributa nezavisno od vrednosti drugih atributa u bazi. Mogu se definisati preko sledeće četvorke <nazivatributa><domen><ograničenje><akcija>

Primer:

Atribut	Domen	Ogranicenje
ocena	Integer	ocena>5 AND ocena <= 10

CRETE DOMAIN ocene INT (2)
FORALL ocenA (ocene > 5 AND ocene <= 10);

- **Pravila integriteta za relacije**, preko kojih je moguće vezti vrednost jednog za vrednost drugog atributa u jednoj relaciji.

Primer: CREATE INTEGRITY RULE starost_smer
FORALL Student (IF Student.SSmer = 01 THEN Student.Strost < 28);

- **Pravila integracije za bazu**, preko kojih je moguće povezati vrednosti atributa iz više relacija. Moguće je iskazati bilo kakvo složeno ograničenje na vrednosti atributa u bazi podataka, ograničenje koje povezuje vrednosti atributa iz više relacija.

Primer: CREATE INTEGRITY RULE ocene_smer
FORALL Student (IF Student.SifSmer = 01 THEN
EXIST Prijava (Prijava.Ocena > 7 AND Student.BrInd = Prijava.BrInd
AND Prijava.SifPred = Predmet.SifPred AND Predmet.NazivPred = 'Matematika'));

13. Navesti sve konvencionalne skupovne operacije Dati primere.

R1

BrInd	Ime	Starost	SifSmer
152/97	Ana	19	01
223/95	Mirko	21	01
021/94	Zoran	20	02

R2

BrInd	Ime	Starost	SifSmer
223/95	Mirko	21	01
021/94	Zoran	20	02
003/94	Milos	22	01

Unija – Rezultat operacije unije R3 := R1 unija R2 je relacija R3 koja sadrži sve n-torke koje se pojavljuju bilo u R1 bilo u **R2**

BrInd	Ime	Starost	SifSmer
152/97	Ana	19	01
223/95	Mirko	21	01
021/94	Zoran	20	02
003/94	Milos	22	01

Diferencija – Rezultat operacije diferencije R3 := R1 - R2 su n-torke relacije R1 koje nisu istovremeno i n-torke relacije **R2.**

BrInd	Ime	Starost	SifSmer
152/97	Ana	19	01

Presek – Rezultat operacije preseka R3 := R1 presek R2 je relacija R3 koja sadrži n-torke koje se pojavljuju u obe relacije **R1 i R2.**

BrInd	Ime	Starost	SifSmer
223/95	Mirko	21	01
021/94	Zoran	20	02

Dekartov proizvod – Rezultat ove operacije R3 := R1 x R2 je relacija R3 čije su n-torke svi „parovi" koje čine jedna n-torka relacije R1 i jedna n-torka relacije R2.

R1

C	D	E
1	2	3
2	4	6

R2

A	B
a1	b1
a2	b2
a3	b3

R3

C	D	E	A	B
1	2	3	a1	b1
1	2	3	a2	b2
1	2	3	a3	b3
2	4	6	a1	b1
2	4	6	a2	b2
2	4	6	a3	b3

Operacije unije, preseka i Dekartovog proizvoda su kumulativne i asocijativne, što ne važi i za operaciju razlike.

14. Ukratko objasniti 4 osnovne komponente svakog modela podataka.

Svaki model podataka treba da čine sledeće komponente:
1) **Struktura modela**, odnosno skup koncepata za opis objekata sistema, njihovih atributaa i njihovih međusobnih veza,
2) **Ograničenja na vrednosti podataka u modelu**, koja u svakom trenutku posmatranja moraju biti zadovoljena,
3) **Operacije nad konceptima strukture**, preko kojih je moguće prikazati i menjati vrednosti podataka u bazi,
4) **Dinamička pravila integriteta**, kojima se definiše osnovno dinamičko ponašanje modela.

Može se predstaviti trojkom <operacija,ograničenje,akcija>

15. Vremensko ozačavanje transakcija.

Svaka transakcija na ulazu u sistem dobije identifikacioni broj u redosledu dolaska. Problem nastaje kad neka transakcija hoće da čita neki slog koji je neka mlađa transakcija (transakcija sa manjim identifikatorom) već ažurilala, ili kada neka transakcija hoće da ažurira neki slog koji je mlađa transakcija već videla ili ažurirala.Konflikti se razrešavaju ponovnim startovanjem transakcije.

Svakom objektu baze podataka pridružuju se dve oznake:
- **RMAX**, Najveći identifikator transakcije koja je uspešno izvršila čitanje posmatranog objekta i
- **UMAX**, Najveći identifikator transakcije koja je uspešno izvršila ažuriranje posmatranog objekta.

Algoritam vremenskog označavanja:
read(R)
if (t >= UMAX)
then //operacija se prihvata
RMAX := MAX(t, RMAX);
else //konflikt
restart T;
write(R) //zajedno sa COMMIT
ажурирање:
if(t >= UMAX **and** t >= RMAX)
then //operacija se prihvata UMAX :=
t;
else { konflikt}
restart T;

U ovoj tehnici se objekti se ne zaključavaju i ne dolazi do pojavljivanja ni mrtvih ni živih lokota. Međutim, obično dolazi do velikog broja restartovanja transakcija. Takođe, naredbe se moraju baferovati do commit-a.

16. Protokoli zaključavanja.

Ovaj protokol omogućava da transakcija „zaključa“ objekta baze podataka kome je pristupila, da bi onemogućila da druge transakcije nekorektno operišu sa istim objektom. Postoji ekskluzivno i deljivo zaključavanje. Ako neka transakcija postavi ekskluzivni lokot na objekat baze podataka, ni jedna druga transakcija ne može na taj objekat da postavi bilo koji drugi lokot. Ako neka transakcija postavi deljivi lokot na objekat baze podataka, neka druga transakcija takođe može da postavi deljivi lokot na isti objekat baze, ali ni jedna druga transakcija ne može da postavi ekskluzivni lokot na taj objekat.

Dvofazni protokol zaključavanja:

1. Pre nego što operiše sa nekim objektom baze, transakcija mora da postavi lokot na njega
2. Posle oslobađanja nekog lokota, transakcija ne može da postavi lokot ni na jedan objekat baze

Ako sve transakcije poštuju dvofazni protokol zaključavanja, taj skup se serijabilno izvršava.

Imajući u vidu ekskluzivni i deljivi lokot, protokol zaključavanja koji bi mogao da reši prikazane probleme može se iskazati na sledeći način:

1. Transakcija koja želi da pročita neki objekat baze podataka (n-torku, npr.) mora prvo da postavi deljivi lokot na taj objekat
2. Transakcija koja želi da ažurira neki objekat baze podataka mora prvo da postavi ekskluzivni lokot na taj objekat. Ako je ta transakcija ranije postavila S lokot, ona treba da taj lokot transformiše u X lokot
3. Ako transakcija nije uspela da postavi lokot na željeni objekat baze podataka, zbog toga što neka druga transakcija već drži nekompatibilan lokot nad posmatranim objektom, tada ona prelazi u stanje čekanja
4. Transakcija oslobađa X lokot obavezno, a po pravilu i S lokot na kraju sa COMMIT ili ROLLBACK naredbom.

17. Uporediti sledeće termine:

1.) Kandidat za ključ; determinanta

Kandidat za ključ je determinanta koja u potpunosti funkcionalno određuje sve neključne atribute posmatrane relacije.

Determinanta relacije R je bilo koji atribut, prost ili složen, od koga neki drugi atribut u relaciji potpuno funkcionalno zavisi.

2.) *Nepotpuna funkcionalna zavisnost; tranzitivna zavisnost*

Atribut Y relacije R je potpuno funkcionalno zavisan od atributa X relacije R ako je funkcionalno zavistan od atributa X, a nije funkcionalno zavisan ni od jednog pravog podskupa atributa X.
Atribut C je tranzitivno savisan od atribuat A ako je funkcionalno zavisan od A i ako je funkcionalno zavisan od nekog atributa V koji je i sam funkcionalno zavisan od A

3.) *relacija; tabela*

Uslovi koje tabela treba da ispuni da bi bila relacija:

- Ne postoje duplikati vrsta tabela,
- Redosled vrsta nije značajan,
- Redosled kolona nije značajan,
- Sve vrednosti atributa u relaciji su atomske

4.) integritet entiteta; referencijalni integritet

Integritet entiteta – ni jedan atribut koji je primarni ključ ili je deo primarnog ključa neke bazne relacije ne može da uzme „nula vrednost“.

Referencijalni integritet – ako neka bazna relacija poseduje spoljni ključ koji ovu relaciju povezuje sa nekom drugom, preko primarnog ključa, tada vrednost sekundarnog ključa mora da bude bilo jednaka vrednosti primarnog ključa ili nula vrednosti.

5.) relaciona algebra; relacioni račun

Relaciona algebra je proceduralni, a **relacioni račun** je neproceduralni jezik. Oba se koriste za iskazivanje operacija relacionog modela.

6.) vrednosna ograničenja; strukturna ograničenja u MOV-u

Strukturna ograničenja su jezički iskaz grafičke predstave MOV-a i odnose se na kardinalnosti preslikavanja. **Vrednosna ograničenja** definišu dozvoljene vrednosti atributa i dozvoljene promene ovih vrednosti.

7.) operacija spajanja; operacija unije u Relacionom modelu

Unija sadrži sve n-torke koje se pojavljuju u bar jednoj od tabela. **Spajanje** podrazumeva da n-torke obe relacije zadovoljavaju uslov zadat nad njihovim atributima.

8.) distinkt tip; struktuirani tip u Objektno-relacionom modelu

Distinktni tipovi su jednostavni, perzistentni, imenovani, korisnički definisani i konačni, tj. ne mogu da imaju podtipove i nije podržano nasleđivanje. **Struktuirani tip** može da ima jedan ili više atributa, da podržava nasleđivanje i ima metode i ne mora da ima sopstvene instance. Nije konačan.

9.) nasleđivanje ponašaja; nasleđivanje stanja u Objektnim SUBP

Odgovor se nalazi u 3. pitanju

18. Objasniti pojmove „živog“ i „mrtvog lokota“. Navesti i objasnite tehnike za razrešavanje „mrtvog lokota“

U toku izvršenja skupa transakcija, moguće je da dve ili više transakcija formiraju “mrtvi lokot“, odnosno da lokoti koje su one postavile na objekte baze podataka sve njih dovode u stanje čekanja.

„Živi lokot“ je situacija u kojoj je neka transakcija stalno u stanju čekanja na neki objekat baze podataka zbog toga što druge transakcije uvek pre nje postave lokot na taj objekat. Problem živog lokota se jednostavno rešava uvođenjem nekog redosleda zaključavanja objekata (FIFO). Za rešavanje problema mrtvih lokota koriste se 3 tehnike:

1) Prekidanje transakcije posle isteka intervala vremena. Koristi parameta timeout koji menadžer lokota dodeljuje transakciji pri pokušaju zaključavanja nekog objekta. Kad ovo vreme istekne transakcija se poništava i ponovo startuje.
2) Prevencija lokota. Jedan od protokola za prevenciju mrtvog čvora se zasniva na uređenju elemenata baze podataka. Blokovi se mogu urediti po njihovim adresama na spoljnoj memoriji. Ako se zahteva da svaka transakcija zaključa elemente u njihovom redosledu koji je definisan na spoljnoj memoriji, do mrtvog čvora neće doći.
3) Detekcija mrtvog čvora – dozvoljava se da do mrtvog čvora dođe, pa se tada neka od transakcija koja ge je izazvala „ubije“, njeni efekti na bazu poništе, a ona eventualno ponovo startuje, nadajući se da tada neće izazvati mrtvi lokot. Za otkrivanje mrtvih lokota u sistemu se koristi tzv. graf čekanja (Wait-For graf).

19. Prikazati arhitekturu objektnih SUBP i dati opis svih komponenti

⇨ ANCI|SPARC tronivovska arhitektura (11. pitanje)