1. Ograničenja u relacionom modelu.

DINAMIČKA PRAVILA INTEGRITETA

Pravila integriteta definišu dozvoljena stanja i dozvoljene prelaze sistema iz stanja u stanje. Pravilo integriteta u relacionom modelu se iskazuje definisanjem ograničenja na vrednosti atributa i akcijama koje se preduzimaju kada neka operacija ažuriranja naruši posmatrano ograničenje.

OGRANIČENJA I PRAVILA INTEGRITETA

Uobičajeno je da se u relacionom modelu definišu dve vrste pravila integriteta:

• Pravila integriteta modela, koja su posledica strukture relacinog modela, pa su zbog toga opšta i moraju da važe u svakom konkretnom relacionom modelu.

> *Definišu se dva opšta pravila integriteta relacionog modela:*
> *(1) Integritet entiteta (integritet ključa): Ni jedan atribut koji je primarni ključ ili deo primarnog ključa neke bazne relacije ne može da uzme nula vrednost.*
> *(2) Referencijalni integritet. Ako neka bazna relacija (recimo R2) poseduje spoljni ključ (recimo SK) koji ovu relaciju povezuje sa nekom drugom baznom relacijom (recimo R1), preko primarnog ključa (recimo PK), tada svaka vrednost SK mora biti bilo jednaka nekoj vrednosti PK, ili biti nula vrednost. Relacije R1 i R2 ne moraju biti*

• Poslovna pravila integriteta, odnosno specifična ograničenja za dati relacioni model. Naziv "poslovna" proističe iz činjenice da se preko ovih ograničenja iskazuju specifični odnosi vrednosti atributa koji važe u datom realnom (najčešće poslovnom) sistemu.

Uobičajeno je da se ova pravila integriteta podele na sledeće podtipove:

> ***Pravila integriteta za domene**, preko kojih se specifikuje koje vrednosti postoje u domenu;*
> ***Pravila integriteta za atribute**, preko kojih se definišu dozvoljene vrednosti nekog atributa nezavisno od vrednosti drugih atributa u bazi;*
> ***Pravila integriteta za relacije**, preko kojih je moguće vezati vrednost jednog, za vrednost drugog atributa u jednoj relaciji;*
> ***Pravila integriteta za bazu**, preko kojih je moguće povezati vrednosti atributa iz više relacija.*

SQL standard podrzava sledece vrste ogranicenja:

Ogranicenja domena
Ogranicenja tabela i kolona
Opsta ogranicenja

• Ogranicenje domena je CHECK ogranicenje. Primenjuje se na sve kolone definisane nad posmatranim domenom.
• Ogranicenje tabele definisano je za jednu baznu tabelu. Ogranicenje tabele je:
-UNIQUE ogranicenje,
-PRIMARY KEY ogranicenje,
-referencijalno (FOREIGN KEY) ogranicenje ili
-CHECK ogranicen

-UNIQUE ogranicenje obezbedjuje jedinstvenost vrednosti jedne ili vise kolona bazne tabele. UNIQUE ogranicenje je zadovoljeno ako i samo ako ne postoje dva reda bazne tabele sa istim definisanim (NOT NULL) vrednostima u kolonama nad kojima je ogranicenje specificirano.
-PRIMARY KEY ogranicenje definise primarni kljuc tabele. Ono je zadovoljeno ako i samo ako ne postoje dva reda bazne tabele sa istim vrednostima u kolonama nad kojima je ogranicenje specificirano i ne postoji ni jedna nedefinisana vrednost ni u jednoj od navedenih kolona.
-Referencijalno ogranicenje realizuje referencijalni integritet na taj nacin sto specificira jednu ili vise kolona bazne tabele kao referencirajuce kolone i njima odgovarajuce referencirane kolone u nekoj (ne neophodno razlicitoj) baznoj tabeli.

> *•Tabela sa referencirajucim kolonama naziva se referencirajuca tabela, a tabela sa referenciranim kolonama - referencirana tabela. Nad referenciranim kolonama referencirane tabele definisano je PRIMARY KEY ili UNIQUE ogranicenje. Referencijalno ogranicenje je zadovoljeno ako su, za svaki red referencirajuce tabele, vrednosti referencirajucih kolona jednake vrednostima odgovarajucih referenciranih kolona nekog reda referencirane tabele. Ako neka od referencirajucih kolona sadrzi null vrednost, zadovoljenje referencijalnog integriteta zavisi od tretmana null vrednosti (match opcija). Kao deo specifikacije referencijalnog ogranicenja mogu se navesti i akcije za zadovoljavanje ogranicenja, kojima se definisu promene koje treba sprovesti nad referencirajucom tabelom, a bez kojih bi promene nad referenciranom tabelom dovele do narusavanja referencijalnog ogranicenja.*

– CHECK ogranicenje definise uslov. Ogranicenje je naruseno ako je za bilo koji red tabele rezultat evaluacije uslova FALSE (lazno), a zadovoljeno ako je rezultat TRUE (istinito) ili UNKNOWN (nepoznato).

• Opste ogranicenje (assertion) je CHECK ogranicenje, kojim se definise uslov nad podacima vise tabela baze podataka. Ogranicenje je naruseno ukoliko je rezultat evaluacije uslova FALSE (lazno), a zadovoljeno ako je rezultat TRUE (istinito) ili UNKNOWN (nepoznato).

• SQL:1999 standard, kao i SQL-92, definise ogranicenje kolone kao sintaksnu skracenicu ogranicenja tabele . Medjutim, pored UNIQUE, PRIMARY KEY, FOREIGN KEY i CHECK ogranicenja, ogranicenje kolone moze biti i NOT NULL.

2. Objasniti definicije i dati primer za ulazne tačke, definisanje promenljivih i osnovni upitni blok OQL-a. Može se koristiti ODL šema rešenja zadatka 2b.
-"ULAZNE TAČKE" U BAZU PODATAKA MOGU DA BUDU BILO KOJI OPSEG (EXTENT) NEKE KLASE ILI BILO KOJI PERZISTENTNI OBJEKAT ČIJE JE IME DEFINISANO PREKO OPERACIJE bind OBJEKTA Database . NA PRIMER "ULAZNA TAČKA" odseci ukazuje na PERZISTENTNU KOLEKCIJU OBJEKATA.
-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 SLEDEĆE NAČINE:
Studenti x
x in studenti
studenti as x

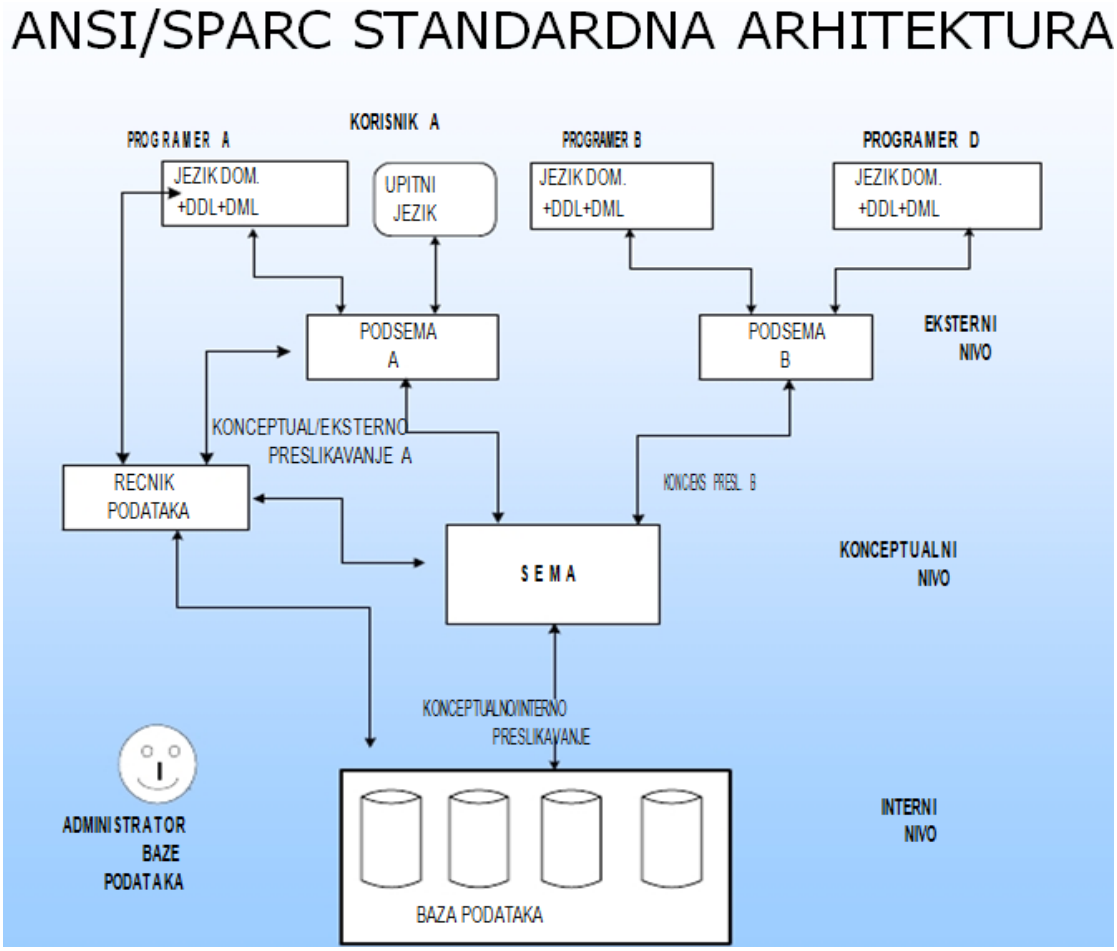
Osnovni uitni oblik blok select…from…where
npr: *select x.ime*
from studenti x
where x.pol = "M";
NIJE NEOPHODNO DA SE UPIT POSTAVLJA PREKO "select...from...where" KLAUZULE. U NAJPROSTIJEM SLUČAJU JE DOVOLJNO NAVESTI NAZIV PERZISTENTNOG OBJEKTA. SVAKI PERZISTENTNI OBJEKAT SAM PO SEBI SE U OQL-u TRETIRA KAO UPIT.

3. Prikazati i objasniti ANSI/SPARC arhitekturu sistema za upravljanje bazom podataka.
ANSI/SPARC arhitektura je troniovska arhitektura u kojoj pojedini nivoi imaju za cilj da, s jedne strane, ucine neyavisnom logicku od fizicke strukture baze podataka, a sa druge strane, aplikacione programe I od fizicke I od celokupne logicke strukture baze. Niovi u ovoj arhitekturi su:

- Interni (fizicki) nivo, koje definise nacin na koji su podaci fizicki organizovani na spoljnim memorijama;
  Na internom nivou, preko jezika koji se naziva interni jezik za definiciju podataka (Internal Data Definition Language-IDDL), sepcificira se fizicka organizacija podataka. Definisu se fizicki zapisi (blokovi), nacin na koji se pojedini podaci memorisu, metode pristupa, indeksi iindeksiranje itd.
- Konceptualni nivo (sema baze podataka), koji definise opstu logicku strukturu baze podataka, sve podatke u sistemu, njihove logicke odnose (veze) I koji treba da omoguci upravljanje podacima kao zajednickim resursom u celom sistemu;
  Na konceptualnom nivou opisuje se opsta logicka struktura baze podataka kao I "preslikavanje" odnosno prevodjenje ove strukture u interni nivo.
- Eksterni (korisnicki) nivo (podsema), na kome se definise logicka struktura podataka pogodna za specificne zahteve, odnosno programe;
  Na eksternom nivou se definisu podmodeli baze podataka pogodni za razvoj pojedinacnih aplikacija. Porgramer razvija svoju palikaciju na dlogickom struktuorom koja predstavlja njegov "pogled" na celokupnu bazu podataka, odnosno nacin na koji je on vidi.

  (Sema je na sledecoj strani)

Slika seme:



4. Protokoli zaključavanja.

Proveru serijabilnosti i preduzimanje odgovarajućih akcija planer izvršenja teško može da obavi u realnom vremenu. Zbog toga se serijabilnosti izvršavanja skupa transakcija najčešće ostvaruje "forsirano" primenjujući mehanizam ***zaključavanja (locking).***

Ekskluzivno zaključavanje (exclusive (XL) lock ili write lock). Ako neka transakcija postavi ekskluzivni lokot na objekat baze podataka, nijedna druga transakcija ne može na taj objekat da postavi bilo koji drugi lokot.

Deljivo zaključavanje (shared (SL) lock ili read lock). Ako neka transakcija postavi deljivi lokot na objekat baze podataka, neka druga transakcija takođe može da postavi deljivi lokot na isti objekat baze, ali nijedna druga ne može da postavi ekskluzivni lokot na taj objekat.

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, na primer) 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 transakcije 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 E lokot obavezno, a po pravilu i S lokot na kraju, sa COMMIT ili ROLLBACK naredbom.

I X i S lokot se postavljaju implicitno, zajedno sa operacijama ažuriranja, odnosno čitanja podataka baze.
Dvofazni protokol zaključavanja:

- Pre nego što operiše sa nekim objektom baze podataka, transakcija mora da postavi lokot na njega;
- Posle oslobađanja nekog lokota, transakcija ne može više postaviti lokot ni na jedan objekat baze.

Može se dokazati teorema da ako u nekom skupu sve transakcije poštuju dvofazni protokol zaključavanja, taj skup se uvek serijabilno izvršava.

5. Osnovne karakteristike i klasifikacija SQL:1999 trigera.
Trigeri su specifična vrsta ECA (Event-Condition_Action) pravila.
-Događaj je operacija ažuriranja baze podataka, uslov je proizvoljni SQL predikat, a akcija je sekvenca SQL naredbi.
-Sekvenca SQL naredbi se izvršava ako je detektovan događaj (pokrenuta odgovarajuća operacija ažuriranja baze) i ako je uslov zadovoljen (sračunava se u true).

-Osnovne karakteristike trigera
SQL:1999 standard definiše trigere kao objekte šeme baze podataka, koji su vezani za tačno jednu baznu tabelu i koji se pozivaju svaki put:
• kada se ubaci jedan ili više redova u tabelu za koju je definisan triger (INSERT naredba),
• kada se promeni jedan ili više redova u tabeli za koju je definisan triger (UPDATE naredba), ili
• kada se obriše jedan ili više redova iz tabele za koju je definisan triger (DELETE naredba).

Klasifikacija trigera>
SQL:1999 trigeri mogu biti kategorizovani na više načina, u skladu sa različitim aspektima posmatranja:
• Trigeri mogu da se pozivaju neposredno pre (BEFORE) ili neposredno posle (AFTER) izvršavanja SQL naredbe (operacije ažuriranja) za koju je definisan triger. U skladu sa time mogu se razvrstati kao:
- BEFORE trigeri
- AFTER trigeri
• U zavisnosti od toga koji od tri tipa operacija ažuriranja baze podataka (INSERT, UPDATE, DELETE) predstavlja događaj koji izaziva pokretanje trigera, trigeri mogu biti:
- INSERT trigeri
- UPDATE trigeri
- DELETE trigeri

•Triger se može izvršiti jednom za operaciju ažuriranja koja ga pokreće ili jednom za svaki red koji je ažuriran (ubačen, izmenjen ili obrisan) operacijom koja pokreće triger. U skladu sa time razlikuju se:
- trigeri koji se pokreću na nivou naredbe (statement-level trigeri)
- trigeri koji se pokreću na nivou reda (row-level trigeri)

6. Objasniti koncept pogleda u relacionom modelu. Prikazati i objasniti SQL:1999 sintaksu naredbe za kreiranje pogleda.
Sta je pogled ?
– Pogled je "prozor" kroz koji se vide podaci baze podataka,
– Pogled je virtuelna tabela (sa njim se radi gotovo kao sa baznom tabelom, mada nema svoje podatke i ne zauzima nikakav memorijski prostor).
• Preciznije receno, pogled se koristi kao bilo koja druga tabela pri izvestavanju.
• Azuriranje baze podataka preko pogleda ima brojna ogranicenja.
– Ne moze se vrsiti azuriranje preko pogleda ukoliko je pogled definisan nad vise tabela. Takvo azuriranje bi znacilo da se jednom naredbom vrsI azuriranje vise tabela baze podataka, sto je suprotno definiciji INSERT, DELETE i UPDATE naredbi.
– Zatim se ne moze vrsiti azuriranje preko pogleda ukoliko se u definiciji pogleda iza SELECT klauzule nalaze funkcije i aritmeticki izrazi.
– Isto tako, azuriranje se ne moze vrsiti ukoliko u definiciju pogleda nisu ukljucene sve NOT NULL kolone tabele nad kojom je pogled definisan.
Zasto koristiti pogled ?
– Jednostavnost koriscenja - uproscava upite,
– Tajnost - mocan mehanizam kontrole pristupa podacima,
– Performanse - cuva se u kompajliranom obliku,
– Nezavisnost podataka - menjaju se definicije pogleda, a ne aplikacioni programi koji koriste podatke baze podataka preko pogleda.

Sintaksa:

```
CREATE VIEW naziv-pogleda[(nazivi atributa pogleda)] AS
      SELECT . . .
      FROM . . .} bilo koja ispravna select naredba
```

Primer: *Kreirati pogled sa atributima odeljenje, posao, ukupna i srednja primanja radnika odredjenog zanimanja u odredjenom odeljenju.*

```
CREATE VIEW FINANSIJE
   (ODELJENJE#, POSAO, UKUPNAPLATA, SREDNJAPLATA) AS
SELECT ODELJENJE#, POSAO, SUM (PLATA), AVG (PLATA)
FROM RADNIK
GROUP BY ODELJENJE#, POSAO;
```

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

NASLEĐIVANJE:

U ODMG se definišu dve posebne vrste nasleđivanja: nasleđivanje ponašanja i nasleđivanje stanja.
-Za nasleđivanje ponašanja se koristi veza nadtip-podtip (koja se ponekad naziva veza generalizacija-specijalizacija ili ISA veza. 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

Za nasleđivanje ponašanja se koristi veza nadtip-podtip. Činjenica da je Nastavnik podtip Radnika, a Asistent podtip Nastavnika označava se 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 izvršava na različit način u zavisnosti od tipa objekta sa kojim se izvršava, naziva se polimorfizam.

U ODMG modelu podržano je višestruko nasleđivanje ponašanja. Tada se, međutim, može desiti da dve (ili više) nasleđene operacije 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) u takvoj hijerarhiji nasleđivanja.

2.NASLEĐIVANJE STANJA

ODMG model uvodi EXTENDS vezu za definisanje nasleđivanja stanja. Na primer,

```
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 tip-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 (slično apstraktnoj klasi u nekim OO jezicima) služi samo za to da se iz njega naslede neke operacije.

Pošto se preko odnosa podtip/nadtip nasleđuje samo ponašanje, ako se želi u podtipu isti atribut koji postoji u interfejsu koji je nadtip, on se u podtipu mora "kopirati".

8. Spoljna unija. Objasniti operaciju i dati primer.
Spajanje tabela moze biti unutrasnje (INNER) ili spoljno (OUTER).
-Ukoliko se vrsta spajanja ne navede eksplicitno podrazumeva se unutrasnje spajanje.
-Ponovimo primere za ekvispajanje i prirodno spajanje sa eksplicitnim navodjenjem klauzule za unutrasnje (INNER) spajanje:

```
SELECT *
FROM RADNIK INNER JOIN ODELJENJE
ON RADNIK.ODELJENJE# = ODELJENJE.ODELJENJE#;

SELECT *
FROM RADNIK NATURAL INNER JOIN ODELJENJE;
```

Spoljno spajanje moze biti levo (LEFT), desno (RIGHT) i centralno (FULL).
– Levo spoljno spajanje omogucuje ukljucivanje u rezultujucu tabelu svih redova tabele sa leve strane JOIN klauzule tako sto se praznim redom prosiruje tabela sa desne strane.
– Desno spoljno spajanje omogucuje ukljucivanje u rezultujucu tabelu svih redova tabele sa desne strane JOIN klauzule tako sto se praznim redom prosiruje tabela sa leve strane.
– Centralno spoljno spajanje omogucuje ukljucivanje u rezultujucu tabelu svih redova i leve i desne tabele tako sto se obe tabele prosiruju praznim redom.
Smisli proste primere

Rezultat uključuje sve redove obe tabele i dodaje prazan red u obe tabele.

TabelaX

RB X	**Ime X**
1	X1
2	X2
3	X3

TabelaY

RB Y	**Ime Y**	**RB X**
10	Y1	1
20	Y2	
30	Y3	2

SELECT Ime X, Ime Y FROM TabelaX LEFT OUTER JOIN TabelaY ON TabelaX.RB X = TabelaY.RB X

Ime X	Ime Y
X1	Y1
X2	Y3
X3	

SELECT ImeX, ImeY FROM TabelaX RIGHT OUTER JOIN TabelaY ON TabelaX.RB X = TabelaY.RB X

Ime X	Ime Y
X1	Y1
	Y2
X2	Y3

SELECT ImeX, ImeY FROM TabelaX FULL OUTER JOIN TabelaY ON TabelaX.RB X = TabelaY.RB X

Ime X	Ime Y
X1	Y1
X2	Y3
X3	
	Y2

9. Vremensko označavanje transakcija.

Svaka transakcija na ulazu u sistem dobije identifikacioni broj u redosledu dolaska. Problem nastaje kad neka transakcija hoće da vidi rekord koji je neka mlađa transakcija (transakcija sa manjim identifikatorom) već ažurilala, ili kada neka transakcija hoće da ažurira neki rekord koji je mlađa transakcija već videla ili ažurirala.

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}
 if t >= UMAX **and** t >= RMAX

then { operacija se prihvata}
UMAX := t;
else { konflikt}
restart T;

10. Konstruisani tipovi objektno-relacionog modela. *(9 poena)*
Konstruisani tipovi

Referentni tipovi –prost tip

-Referenciranju može biti odredjen opseg (SCOPE), koji se specificira navodjenjem naziva relacije čije se n-torke referenciraju:
A REF(T) SCOPE R
Referentna kolona tabele čije se n-torke referenciraju odredjuje se dodavanjem sledeće klauzule u CREATE TABLE naredbu tabele:
REF IS <naziv atributa> <nacin generisanja>
-Naziv atributa je naziv dat koloni koja će služiti kao "identifikator objekta" za kolonu. Način generisanja je tipično:
• SYSTEM GENERATED, sa značenjem da je SUBP odgovoran za održavanje jedinstvenosti vrednosti kolone u svakoj n-torki
• DERIVED, sa značenjem da će SUBP koristiti vrednosti primarnog ključa relacije za izvodjenje jedinstvenih vrednosti kolone

Tip vrsta-slozen tip

Tip vrsta je niz polja koja čine parovi (<naziv podatka>, <tip podatka>)
novina u SQL:1999 je to da je sada moguće definisati promenljive i parametre koji su tipa vrsta, odnosno definisati kolonu u tabeli koja će imati kompleksnu strukturu

ROW (<naziv polja, tip polja> [{, <naziv polja, tip polja>} …])

Tip vrste u tabeli može:
-predstavljati vrstu tabele ili
-pripadati jednoj koloni tabele kao složeni tip podatka

Primer:

```
CREATE TABLE adresa (ulica CHAR(30),
broj INTEGER,
grad CHAR(20));

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

Kolekcija-slozen tip

Kolekcija je grupa koja se sastoji od nula ili više elemenata istog tipa.
Broj elemenata kolekcije se naziva kardinalnost kolekcije
Niz A je uredjena kolekcija u kojoj je svaki elemenat povezan sa tačno jednom rednom pozicijom (koja se naziva indeks)
Dva niza su uporediva ako i samo ako su tipovi njihovih elemenata uporedivi
-Formalna definicija niza je:
<tip elemenata niza> ARRAY [<maksimalna kardinalnost niza>]

Sledećom naredbom se ubacuje jedna n-torka u tabelu sekcija i zatim se prikazuje naziv sekcije i član sekcije na drugoj poziciji u nizu kojim su predstavljeni članovi.

```
-CREATE TABLE sekcija(naziv CHAR(15),
        clan CHAR(20) ARRAY[20]);
-INSERT INTO sekcija (naziv,clan)
        VALUES ('dramska', ARRAY ['Markovic', 'Popovic', 'Denic']);
-SELECT naziv, clan[2] AS ime FROM sekcija;
```

11. Katalog baze podataka; model objekti-veze za relacioni katalog.

-Katalog je imenovana kolekcija sema baze podataka u SQL okruzenju. SQL okruzenje sadrzi nula ili vise kataloga, a katalog sadrzi jednu ili vise sema. Svaki katalog sadrzi semu sa nazivom INFORMATION_SCHEMA, koja predstavlja recnik podataka. Nju cini skup pogleda, odnosno sistemskih tabela, koje sadrze sve bitne informacije o SQL okruzenju. Sadrzaj sistemskih tabela se automatski odrzava.

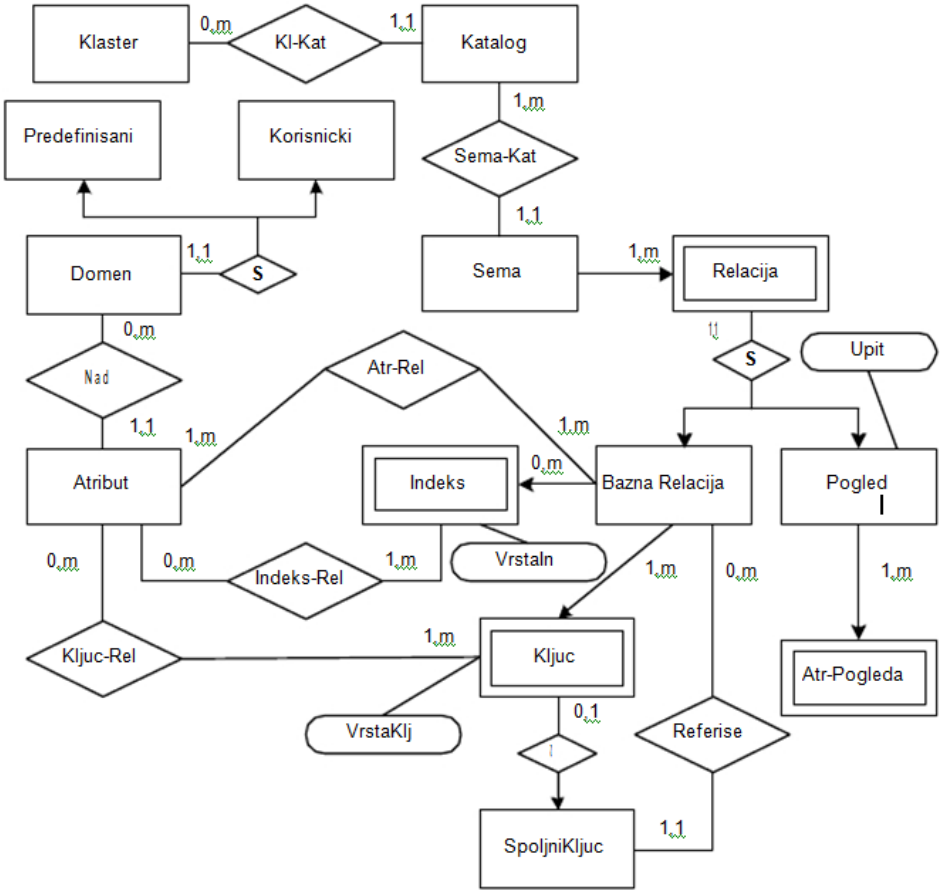
-Sema je kolekcija svih objekata koji dele isti prostor imenovanja. Svaki objekat (tabela, pogled, itd.) pripada tacno jednoj semi. Pod pripadnoscu se ne podrazumeva fizicka pripadnost, vec hijerarhijska veza u kojoj, na primer, sema sadrzi nula ili vise tabela, a svaka tabela logicki pripada tacno jednoj semi.

-U SQL standardu ne postoje naredbe za kreiranje i unistavanje kataloga. Nacin njihovog kreiranja i unistavanja je implementaciono-definisan (implementation-defined), odnosno prepusten je vlasnicima softverskih proizvoda koji implementiraju SQL okruzenje.

-Pun naziv objekata seme ima tri komponente, razdvojene tackama: naziv kataloga, naziv seme i naziv objekta. Ako je posmatrani objekat tabela, tada se njen pun naziv specificira na sledeci nacin: <naziv kataloga>.<naziv seme>.<naziv tabele>.

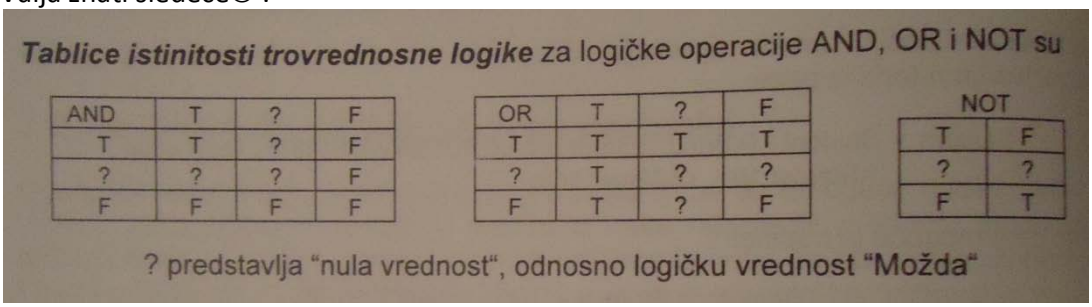
Objekti seme se mogu referencirati sa eksplicitnim ili implicitnim nazivom kataloga i seme:
FROM <naziv tabele> -- nekvalifikovan naziv
FROM <naziv seme>.<naziv tabele>-- delimicno kvalifikovan naziv
FROM <naziv kataloga>.<naziv seme>.<naziv tabele>-- potpuno kvalifikovan naziv

model objekti-veze za relacioni katalog:



12. Navesti sve dodatne operacije relacione algebre koje su uvedene zbog postojanja nula vrednosti u bazi podataka i OBAVEZNO dati primere.

Valja znati sledece☺ :



Dodatne operacije. Zbog postojanja "nula vrednosti" u bazi podataka, neophodno je da se definiše logička funkcija "IS NULL" čiji argument može da bude neki skalarni izraz, a koja se sračunava u T ako je vrednost tog argumenta "nula vrednost", a inače uzima vrednost F.

Možda Selekcija (MAYBE_SELECT- ?σ). Selektuju se one n-torke relacije za koje se predikat selekcije, na osnovu trovrednosnih tablica istinitosti, sračunava u "nula vrednost".

$R_8 := ?\sigma_A (R_3)$

R_8

A	B
?	b
?	?

Tabela R3 i osnovne skupovne operacije sa "nula":

Operacije ***unije, razlike, preseka i Dekartovog*** proizvoda ostaju neizmenjene, tim što se nula vrednost tretira kao i bilo koja druga vrednost. Na primer,

Unija: $R_3 := R_1 \cup R_2$

R_1

A	B
a	b
a	?
?	b
?	?

R_2

A	B
x	y
?	b
?	?

R_3

A	B
a	b
a	?
?	b
?	?
x	y

Diferencija: $R_4 := R_1 - R_2$

R_4

A	B
a	b
a	?

MožđaSpajanje (MAYBE_JOIN). U rezultatu spajanja se pojavljuju one n-torke za koje se predikat spajanja sračunava u "nula vrednost", na osnovu trovrednosnih tablica istinitosti.

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

Rb

A	B
a1	b1
a2	b2
?	b3

Rc

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

Ra

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

SpoljnoSpajanje (OUTER_JOIN). Neka su date relacije $R_1(A,B)$ i $R_2(C,D)$. Pretpostavimo da se vrši operacija ekvispajanja ovih relacija, R_1 [A = C] R_2. Ako je $\pi_A(R_1) \neq \pi_C(R_2)$ (projekcije relacija po atributima spajanja su različite), tada će se u rezultatu spajanja "izgubiti" neke n-torke relacija R_1 i/ili R_2. Ako se takvo gubljenje informacija ne želi, ***SpoljnoSpajanje*** ih može sačuvati. Postoje tri vrste ***SpoljnogSpajanja:***

- ***Levo spoljno spajanje*** ($\Lambda\Sigma\Sigma$) u kome se navode sve n_torke relacije R_1, a za one koje se "ne spajaju" sa n_torkama relacije R_2, atributi koji potiču iz relacije R_2 dobijaju "nula vrednost". Na primer, za relacije R_1 i R_2 rezultat ***Levog spoljnog spajanje*** je relacija R_9 :

$R_9 := R_1[\Lambda\Sigma\Sigma R_1.A = R_2.C]\ R_2$

R_1

A	B
1	2
2	1
4	?

R_2

C	D
3	4
2	2

R_9

A	B	C	D
1	2	?	?
2	1	2	2
4	?	?	?

- ***Desno spoljno spajanje*** ($\Delta\Sigma\Sigma$) u kome se navode sve n_torke relacije R_2, a za one koje se ne spajaju sa n_torkama relacije R_1, atributi koji potiču iz relacije R_1 dobijaju nula vrednost.

$R_{10} := R_1\ [R_1.A = \Delta\Sigma\Sigma\ R_2.C]\ R_2$

R_{10}

A	B	C	D
?	?	3	4
2	1	2	2

Centralno spoljno spajanje (ΤΣΣ) u kome se pojavljuju sve n-torke obe relacije, a one koje se "ne spajaju" dopunjavaju se sa nula vrednostima za atribute druge relacije.

$$R_{11} := R1\ [T\Sigma\Sigma R1.A = T\Sigma\Sigma\ R2.C]\ R_2$$

R_{11}

A	B	C	D
1	2	?	?
2	1	2	2
4	?	?	?
?	?	3	4

SpoljnaUnija (OUTER_UNION). Kao što je rečeno, operacija unije se može izvesti samo nad relacijama koje zadovoljavaju kriterijume kompatibilnosti (da su istog stepena i da su im odgovarajući atributi definisani nad istim domenima). 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, pa da je, zatim, izvršena operacija unije.

Primer: $R_3 := R_1$ OUTER_UNION R_2

R_1

A	B	C
a1	b1	c1
a2	b1	c2

R_2

A	D
a1	4
a1	7
a2	5

R_3

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

13. Operacija deljenja relacione algebre: dati definiciju, primer i postupak izvođenja na osnovu drugih operacija relacione algebre.

8. ***Deljenje***. Deljenje je operacija pogodna za upite u kojima se javlja reč "svi" ("sve", "sva"). Formalno se definiše na sledeći način:

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

$$A[Y \div Z]B = R(X)$$

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).

Primer (Slika 3.5.): Iz relacija Predmet i Prijava prikaži brojeve indeksa studenata koji su položili sve predmete.

Predmet

ŠifPred
P1
P2
P3

Prijava

Brind	ŠifPred
152/97	P1
152/97	P2
021/94	P1
003/94	P3
152/97	P3

Prijava [Prijava.ŠifPred ÷ Predmet.ŠifPred] Predmet

Brind
152/97

Slika 3.5. Primer za operaciju deljenja

Operacija deljenja nije primitivna operacija relacione algebre, već se može izvesti pomoću drugih operacija na sledeći način:

$$A(X, Y)\ [Y \div Z]B(Z) = \pi_X A - \pi_X ((\pi_X A \times B) - A)$$

Objašnjenje:

- $\pi_X A$ daje sve n-torke koje mogu da učestvuju u rezultatu,
- $(\pi_X A \times B)$ daje relaciju u kojoj se za svaku vrednost z iz B pojavljuju parovi <x,z> sa svim vrednostima x,
- $((\pi_X A \times B) - A)$ ne sadrži ni u jednom paru <x,z> one vrednosti x za koje u relaciji A, kao vrednosti y, postoje sve vrednosti z.

Rezultat je, znači, relacija koja sadrži one n-torke x za koje postoje u paru <x,y>, kao vrednosti y, sve vrednosti z.

14. Objasniti pojmove „živog“ i „mrtvog“ lokota i opisati načine razrešavanja.
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.
Pored pojma “mrtvog lokota”, ponekad se definiše i pojam “živog lokota”. To 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” jednostavno se rešava uvođenjem nekog redosleda zaključavanja objekta (na primer FIFO).
-Za razrešavanje problema mrtvih lokota koriste se tri tehnike:
1. Prekidanje transakcije posle isteka intervala vremena predviđenog za čekanje na lokot za neki elemenat. Ovo pravilo koristi parametar “timeout” koji menadžer lokota dodeljuje transakciji pri pokušaju zaključavanja nekog objekta. Kada ovo vreme istekne transakcija se poništava i ponovo startuje, očekujući da tada neće proizvesti “mrtvi lokot”.
2. Prevencija lokota. Mogu se definisati različiti protokoli koji će sprečiti da do “mrtvog čvora” dođe. Jedan od takvih protokola se zasniva na uređenju elemenata baze podataka. Na primer, blokovi se mogu urediti po njihovim adresama na spoljnoj memoriji (A1 < A2 < A3 < ...< An). Ako se zahteva da svaka transakcija zaključa elemente u njihovom definisanom redosledu, očigledno je da do “mrtvog čvora” ne može da dođe. Na primer ako T1 zahteva A1 pa A2, transakcija T2 neće moći da traži lokote u redosledu A2 pa A1, koji bi doveo do “mrtvog čvora”, jer ovakvo zaključavanje nije po definisanom protokolu. Drugi protokol prevencije zasniva se na dodeljivanju, za ovaj protokol specifične, vremenske oznake transakciji. Ako je transakcija koja zahteva lokot starija (manja vremenska oznaka) od transakcije koja drži lokot na elementu baze, dozvoljava joj se čekanje na dobijanje lokota, u protivnom se prekida. Moguće je primeniti i obrnuti uslov prekidanja.
3. Detekcija “mrtvog čvora. Dozvoljava da “mrtvog lokota” dođe, pa se tada neka od transakcija koja ga je izazvala "ubije", njeni efekti na bazu podataka poništе, a ona sama, eventualno, ponovo startuje, nadajući se da tada neće izazvati mrtvi lokot. Za otkrivanje “mrtvih lokota” u sistemu koristi se tzv. graf čekanja (Wait-For graf) čiji su čvorovi transakcije T u izvršenju, a grana Ti - Tj postoji ako transakcija Ti zahteva neki objekat koji je transakcija Tj zaključala. Na Slici 11.15 prikazan je jedan graf čekanja. Transakcija T1 čeka na objekat koji je zaključala transakcija T2, ova čeka na objekat koji je zaključala T3, a T3 čeka na objekat koji je zaključala T1. Ispitivanje mrtvih lokota se može vršiti bilo svaki put kada neka transakcija prelazi u stanje "čekaj", bilo periodično, ili se može smatrati da je mrtvi lokot nastao ako transakcija ništa nije uradila u predefinisanom periodu vremena.
Očigledno je da se nalaženje “mrtvog lokota” svodi na nalaženje ciklusa u grafu čekanja. Kriterijumi za izbor transakcije koja će se poništiti ("žrtve") mogu biti različiti, na primer, transakcija koja je poslednja startovala ili transakcija koja zahteva najmanje lokota. "Ubijanje" transakcije i poništavanje njenih efekata oslobađa i sve lokote koje je ona postavila.

15. Fizičko projektovanje relacionih baza podataka.

FIZICKO PROJEKTOVANJE BAZA PODATAKA SE VRSI NAKON POTPUNOG LOGICKOG PROJEKTOVANJA, NA OSNOVU JASNE LOGICKE STRUKTURE BAZE PODATAKA.

• FIZICKO PROJEKTOVANJE BAZA PODATAKA VEOMA JE ZAVISNO OD KONKRETNOG SUBP.

• NIJE UOBICAJENO DA SE VRSE NEKI DETALJNI "FIZICKI PRORACUNI", RADIJE SE PRIMENJUJU EKSPERTSKA ZNANJA I KASNIJE PODESAVANJE FIZICKE STRUKTURE.

FIZICKO PROJEKTOVANJE RELACIONIH BAZA PODATAKA - KORACI:

1. PRILAGODJAVANJE LOGICKE STRUKTURE KONKRETNOM SKUPU APLIKACIJA - DENORMALIZACIJA
2. DISTRIBUCIJA BAZE PODATAKA - RAZLICITE "KLIJENT- SERVER" ARHITEKTURE
3. "KLASTEROVANJE" - PODACI KOJI SE ZAJEDNO KORISTE TREBA DA BUDU FIZICKI BLISKI
4. ODREDJIVANJE METODA PRISTUPA (INDEKSIRANJE I EVENTUALNO "HESING")

16. Oporavak u distribuiranim bazama podataka.

Slike sa fotoaparata

- Dvonivoski mehanizam oporavka baze podataka. Na prvom nivou su mehanizmi oporavka lokalnih SUBP, a na drugom tzv. Koordinator koji koordinira mehanizme oporavka sa prvog nivoa. Implementira se dvofazni protokol potvrđivanja sa sledeće dve faze:
  1. kada svi SUBP na prvom nivou pošalju signal koordinatoru da je deo transakcije završen, koordinator odgovara porukom „pripremi se". Upisuju se neophodni podaci na log i šalje „ok" ili „not ok" u zavisnosti od uspešnosti.
  2. ako su svi SUBP poslali poruke „ok" vrši se potvrđivanje (COMMIT) svih delova transakcije. U protivnom se poništavaju promene (ROLL BACK).

17. Objasniti sličnosti i razlike između slededih koncepata:

a) Relaciona algebra; Relacioni račun
- relaciona algebra je proceduralni, a relacioni račun je neproceduralni jezik. Obe se koriste za iskazivanje operacija relacionog modela.

b) Vrednosna ograničenja; Strukturna ograničenja u Modelu objekti-veze
- 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.

c) Operacija spajanja; Operacija unije u Relacionom modelu
- Unija sadrži sve n-torke koje se pojavljuju bilo u jednoj od tabela. Spajanje podrazumeva n-torke obe relacije da zadovoljavaju uslov zadat nad njihovim atributima.

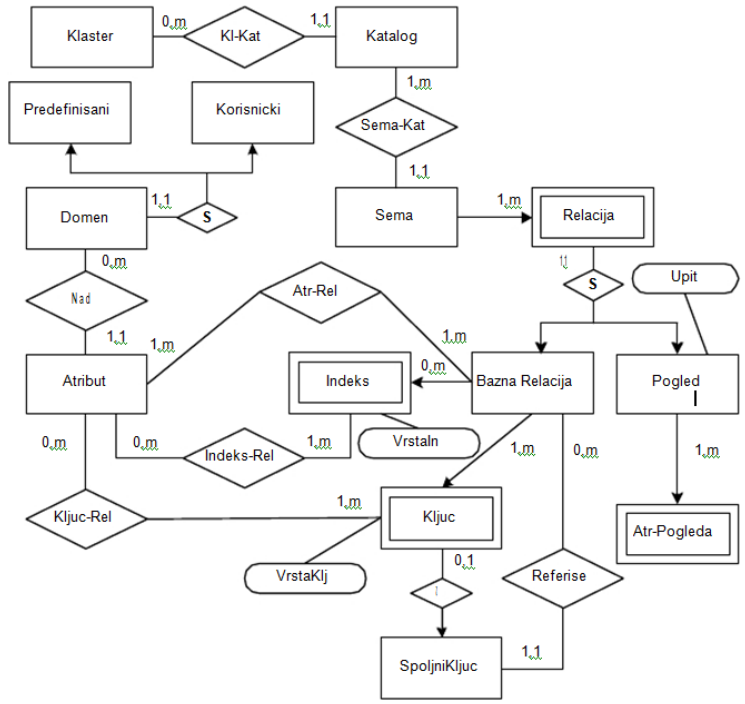
d) Distinkt tip; Struktuirani tip u Objektno-relacionom modelu
- distinkt tipovi su jednostavni, perzistentni, imenovani, korisnički definisani i konačni, tj. ne mogu da imaju podtipove i nije podržano nasleđivanje. Struktuirani tip može da ima jedan ili više atributa, podržava nasleđivanje i ima metode i ne mora da ima sopstvene instance. Nije konačan.

e) Deljivi lokot; Ekskluzivni lokot
- ako jedna transakcija postavi ekskluzivni lokot ni jedna druga ne može da uradi isto sa tim atributom. Kod deljivih lokota više transakcija može da ih postavi na isti objekat, ali ni jedna ne može da stavi ekskluzivni ako postoji već neki deljivi lokot.

(10 poena)

18. Šta je Transakcija? Objasniti skup osobina koje transakcija mora da poseduje.
Transakcija je jedno izvršenje neke "logičke jedinice posla", jedno izvršenje neke logičke celine jednog programa, ili jedno izvršenje celog programa.
Transakcija mora da poseduje sledeći skup osobina (ACID osobine):
1. Atomnost (Atomicity). Transakcija, kao "logička jedinica posla", mora predstavljati atomski skup aktivnosti.
2. Konzistentnost (Consistency). Izvršenje transakcije treba da prevede bazu podataka iz jednog u drugo konzistentno stanje.
3. Izolacija (Isolation). Transakcija ne treba svoje promene baze podataka da učini vidljivim drugim transakcijama pre nego što se ona okonča, odnosno promene potvrde u bazi podataka.
4. Trajnost (Durability). Kada su, u bazi podataka, potvrđene promene koje je izvršila neka transakcija, ove promene se više ne mogu izgubiti.

19. Objasniti pojam „Rečnik podataka“. Rečnik baze podataka u SQL okruženju. MOV za relacioni rečnik.
Baza podataka, pored podataka, sadrži i metapodatke, odnosno tzv “Recnik podataka” (Data Dictionary, Data Directory, Catalog). Rečnik baze podataka opisuje posmatranu bazu podataka (strukturu baze, pravila očuvanja integriteta podataka, prava korišćenja i slično).
Rečnik podataka je "baza podataka o bazi podataka", pa se taj deo baze podataka naziva i metabaza podataka.
SQL okruženje definiše i sledeće:
• Šema je kolekcija tabela, pogleda, domena, tvrdnji (assertions)
• Katalog je kolekcija šema. U katalogu postoji i informaciona šema (INFORMATION_SCHEME), katalog u užem smislu reči, u kojoj se daju informacije o svim šemama u katalogu.
• Klaster je kolekcija kataloga.
MOV za relacioni:



20. Objasniti sledede koncepte:
a) Determinanta; Kandidat za ključ
Determinanta realacije R je bilo koji atribut, prost ili slozen, od koga neki drugi atribut u relaciji potpuno funkcionalno zavisi.
Kolekcija atributa K takvih da zadovoljavaju uslove **jedinstvenosti** i **neredundandnosti** nazivaju se **kandidati za ključ**.
b) Potpuna funkcionalna zavisnost; Tranzitivna zavisnost
Potpuna funkcionalna zavisnost se definiše: **Atribut Y relacije R je potpuno funkcionalno zavisan od atributa X relacije R ako je funkcionalno zavisan od atributa X, a nije funkcionalno zavisan ni od jednog pravog podskupa atributa X.**
Atribut C **je tranzitivno funkcionalno zavisan** od atributa A ako je funkcionalno zavisan od A i ako je funkcionalno zavisan od nekog atributa B koji je sam funkcionalno zavisan od A.
A→ B
B→C
A→C
B-/->A
C-/->A
c) Uslovi koje treba tabela da zadovolji da bi bila relacija
Tabela mora da zadovolji sledeće uslove da bi bila relacija

1. Ne postojie duplikati vrsta
2. Redosled vrsta nije značajan
3. Redosled kolona nije značajan
4. Sve vrednosti atributa u relacijama su atomske – odnosno nije dozvoljeno da vrednosti nekih atributa u relaciji budu relacije.

d) Integritet entiteta; Referencijalni integritet
Integritet entiteta (integritet ključa) Ograničenje u ovom pravilu integriteta se iskazuje na sledeći način. **Ni jedan atribut koji je primarni ključ ili deo primarnog ključa neke bazne relacije ne može da uzme nula vrednost.**
Referencijalni integritet. Ako neka bazna relacija (recimo R2) poseduje spoljni ključ (recimo SK) koji ovu relaciju povezuje sa nekom drugom baznom relacijom (recimo R1), preko primarnog ključa (recimo PK) tada se svaka vrednost SK mora biti bilo jednaka nekoj vrednosti PK ili biti nula vrednost. Relacija R1 i R2 ne moraju biti različite.
e) Objasniti dvofazni protokol zaključavanja(10 poena)
Dvofazni protokol zaključavanja
• Pre nego što operiše sa nekim objektom baze podataka, transakcija mora da postavi lokot na njega;
• Posle oslobađanja nekog lokota, transakcija ne može više postaviti lokot ni na jedan objekat baze.
-Može se dokazati teorema da ako u nekom skupu sve transakcije poštuju dvofazni protokol zaključavanja, taj skup se uvek serijabilno izvršava.

21. Vrste ograničenja u relacionom modelu. Pokazati na primeru definisanje ograničenja.
Kada se proveravaju ograničenja?

Ogranicenja domena
Ogranicenja tabela i kolona
Opsta ogranicenja

• Ogranicenje domena je CHECK ogranicenje. Primenjuje se na sve kolone definisane nad posmatranim domenom.
• Ogranicenje tabele definisano je za jednu baznu tabelu. Ogranicenje tabele je:
-UNIQUE ogranicenje,
-PRIMARY KEY ogranicenje,
-referencijalno (FOREIGN KEY) ogranicenje ili
-CHECK ogranicen
-UNIQUE ogranicenje obezbedjuje jedinstvenost vrednosti jedne ili vise kolona bazne tabele. UNIQUE ogranicenje je zadovoljeno ako i samo ako ne postoje dva reda bazne tabele sa istim definisanim (NOT NULL) vrednostima u kolonama nad kojima je ogranicenje specificirano.

```
CREATE TABLE RADNIK
   (...
     MLB CHAR (13) UNIQUE,
     ...
     ...);
```

-PRIMARY KEY ogranicenje definise primarni kljuc tabele. Ono je zadovoljeno ako i samo ako ne postoje dva reda bazne tabele sa istim vrednostima u kolonama nad kojima je ogranicenje specificirano i ne postoji ni jedna nedefinisana vrednost ni u jednoj od navedenih kolona.

```
CREATE TABLE ODELJENJE
   (ODELJENJE# INTEGER PRIMARY KEY,
     ...
     ...);
```

-Referencijalno ogranicenje realizuje referencijalni integritet na taj nacin sto specificira jednu ili vise kolona bazne tabele kao referencirajuce kolone i njima odgovarajuce referencirane kolone u nekoj (ne neophodno razlicitoj) baznoj tabeli.

```
CREATE TABLE RADNIK
   (...
     ...
     ODELJENJE# INTEGER NOT NULL
           REFERENCES ODELJENJE (ODELJENJE#),
     ... );
```

•Tabela sa referencirajucim kolonama naziva se referencirajuca tabela, a tabela sa referenciranim kolonama - referencirana tabela. Nad referenciranim kolonama referencirane tabele definisano je PRIMARY KEY ili UNIQUE ogranicenje. Referencijalno ogranicenje je zadovoljeno ako su, za svaki red referencirajuce tabele, vrednosti referencirajucih kolona jednake vrednostima odgovarajucih referenciranih kolona nekog reda referencirane tabele. Ako neka od referencirajucih kolona sadrzi null vrednost, zadovoljenje referencijalnog integriteta zavisi od tretmana null vrednosti (match opcija). Kao deo specifikacije referencijalnog ogranicenja mogu se navesti i akcije za zadovoljavanje ogranicenja, kojima se definisu promene koje treba sprovesti nad referencirajucom tabelom, a bez kojih bi promene nad referenciranom tabelom dovele do narusavanja referencijalnog ogranicenja.

– CHECK ogranicenje definise uslov. Ogranicenje je naruseno ako je za bilo koji red tabele rezultat evaluacije uslova FALSE (lazno), a zadovoljeno ako je rezultat TRUE (istinito) ili UNKNOWN (nepoznato).

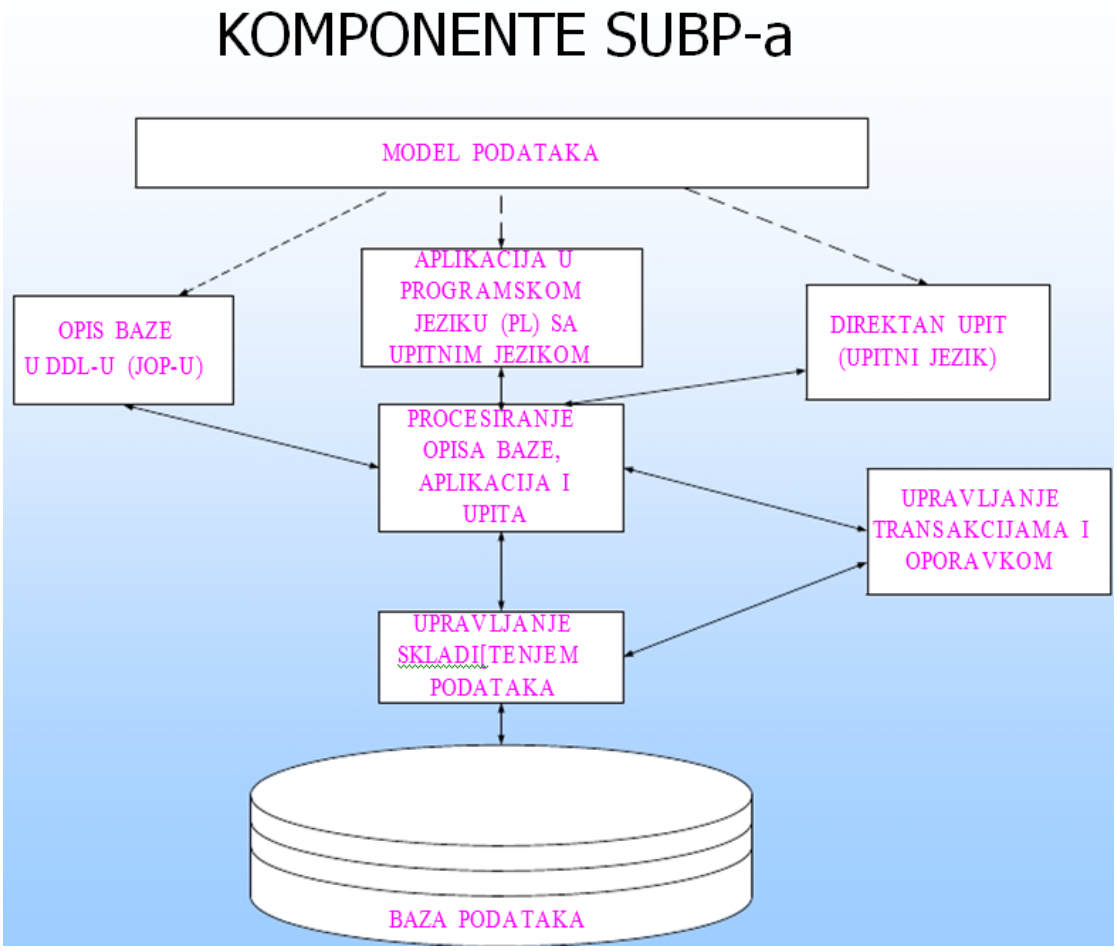
```
CREATE TABLE RADNIK

   (...

     IME VARCHAR (20) NOT NULL CHECK(IME = UPPER (IME)),

     ...

     ...);
```

• Opste ogranicenje (assertion) je CHECK ogranicenje, kojim se definise uslov nad podacima vise tabela baze podataka. Ogranicenje je naruseno ukoliko je rezultat evaluacije uslova FALSE (lazno), a zadovoljeno ako je rezultat TRUE (istinito) ili UNKNOWN (nepoznato).

• SQL:1999 standard, kao i SQL-92, definise ogranicenje kolone kao sintaksnu skracenicu ogranicenja tabele . Medjutim, pored UNIQUE, PRIMARY KEY, FOREIGN KEY i CHECK ogranicenja, ogranicenje kolone moze biti i NOT NULL.

Provera ogranicenja:
• Ogranicenja se, po default-u, proveravaju na kraju izvrsavanja svake SQL naredbe.
• SQL:1999 standard dozvoljava da se provera ogranicenja odlozi i izvrsi na kraju transakcije.
• Za svako ogranicenje je moguce specificirati da li je ili ne dozvoljeno odlaganje provere do kraja transakcije, sto se zapisuje navodjenjem opcije[NOT] DEFERRABLE.
• Ako je izabrana opcija DEFERRABLE, tada je neophodno navesti da li je na pocetku transakcije provera ogranicenja odlozena (INITIALLY DEFERRED) ili ne (INITIALLY IMMEDIATE).
• Provera ogranicenja, koje je specificirano kao INITIALLY IMMEDIATE DEFERRABLE, moze se odloziti do kraja transakcije koriscenjem naredbe SET CONSTRAINTS.

22. Prikazati šemu komponenti Sistema za upravljanje bazama podataka i opisati ulogu svake komponente.



1. Baza podataka :
Njena primarna uloga je čuvanje podataka. Međutim, na njoj se nalazi i Rečnik podataka, tj. metapodaci (podaci o samoj bazi) kao što su : struktura baze, prava korišćenja i pravila očuvanja integriteta (koja služe da se njima definiše tačnost, tj dozvoljene vrednosti podataka, i konzistentnost, tj.dozvoljeni odnosi podataka). Pored toga jedan deo baze obuhvata I baza indeksa,gde indeks u najopštijem smislu predstavlja strukturu podataka koja omogućava brz pristup indeksiranim podacima.
Velike baze pored sekundarne memorije (kao što je disk) mogu zahtevati I tercijalnu memoriju reda terabajta (kao što je npr. Sistem kompakt diskova sa robotom za pristup konkretnom disku).

2. Sistem za upravljanje skladištenjem podataka :
Sastoji se od Upravljanja datotekama i Upravljanja 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 vraća taj blok na disk.

3. 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.

Jezici baze mogu biti neproceduralni ili mogu imati I proceduralne delove u sebi. Neproceduralni jezici omogućavaju specifikovanje uslova koje rezultat treba da zadovolji ali ne omogućavaju specifikovanje procedure preko koje se dolazi do rezultata pa je njihova osnovna namena specifikovanje upita zbog čega se nazivaju još I upitni jezici. Međutim, mogu se koristiti I za kreiranje I modifikovanje šeme baze podataka, zbog čega je neophodna transformacija neproceduralnog iskaza u niz akcija koje treba da obavi Sistem za upravljanje skladištenjem podataka. Ova transformacija vrši se preko Procesora upita I njen najteži deo je optimizacija upita, tj.nalaženje najpogodnijeg niza akcija. Najpoznatiji upitni jezik je SQL kod relacionih baza I QQL kod objektnih.

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.
Aplikacije mogu biti definisane I preko generatora aplikacija, koji generiše aplikacije na osnovu sličnoti između structure definisanog korisničkog interfejsa I structure podataka u bazi. On daje okvir aplikacije sa specifikovanim procedurama, koje predstavljaju odziv sistema na određene akcije korisnika. Kod ovh procedura treba da razvije programer preko Jezika IV generacije (kombinacije proceduralnih I neroceduralniih jezika).

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

4. Upravljanje transakcijama I oporavkom:

Omogućava da baza podataka ostane u konzistentnom stanju pri konkurentnoj obradi podataka (kada dva ili više korisnika istovremeno pristupaju istim podacima) a kopiranjem podataka na arhivske memorije I registrovanjem promena koje su se desile između dva kopiranja omogućava efikasan oporavak baze nakon otkaza sistema.

23. Osnovna struktura XML dokumenta (dati primer). Koje uslove mora da zadovolji dobro oformljen XML dokument? Sta je validan XML dokument?Primer XML dokumenta:

```
<?xml version="1.0" encoding="UTF-8"?>
<BioskopskiRepertoar xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Film FilmId="ID_1">
        <NazivFilma>Spider-Man 1</NazivFilma>
        <ImeRezisera>Tamo Neki Baja</ImeRezisera>
        <Trajanje>120 min</Trajanje>
    </Film>
    <Bioskop>
        <NazivBioskopa>TuckWood</NazivBioskopa>
        <Sala>
            <NazivSale>Rita Hayworth</NazivSale>
            <Projekcija FilmId="ID_1">
                <DatumProjekcije>24.09.2004.</DatumProjekcije>
                <BrojPosetilaca>12</BrojPosetilaca>
            </Projekcija>
            <Projekcija FilmId="ID_1">
                <DatumProjekcije>25.09.2008</DatumProjekcije>
                <BrojPosetilaca>42</BrojPosetilaca>
            </Projekcija>
        </Sala>
    </Bioskop>
</BioskopskiRepertoar>
```

Dobro oformljen XML dokument mora da zadovolji sledeće uslove:

1. da postoji XML deklaracija dokumenta, <?xml version="1.0" encoding="UTF-8"?>
2. da poseduje samo jedan element dokumenta (koreni element) u kome su ugnježdeni svi ostali elementi
3. svi elementi i atributi u dokumentu moraju da budu sintaksno ispravni, svi moraju imati oznaku za početak i kraj, i svaka vrednost atributa mora biti unutar znaka navoda

Kada se definiše **tip** XML dokumenta, može se uvesti i pojam **validnog XML dokumenta.** Validni XML dokumenat mora da poštuje strukturu definisanu u opisu tipa dokumenta. Tip dokumenta se definise na dva nacina: XML 1.0 DTD specifikacijom ili XML Schemom

24. Navesti sve Konevncionalne skupovne operacije i Specijlane relacione operacije realcione algebre. OBAVEZNO dati primere za svaku vrstu operacije.

Konevncionalne skupovne operacije:

BrInd	MLB	Ime	Starost	ŠifSmer
152/97	16309723331981	Ana	19	01
223/95	13975673331981	Mirko	21	01
021/94	11145276418976	Zoran	20	02

S_2

BrInd	MLB	Ime	Starost	ŠifSmer
223/95	13975673331981	Mirko	21	01
021/94	11145276418976	Zoran	20	02
003/94	23456786418976	Miloš	22	01

1. ***Unija.*** Date su relacije R_1 i R_2 koje zadovoljavaju navedeni uslov kompatibilnosti. Rezultat operacije unije

$$R_3 := R_1 \cup R_2$$

je relacija R3 koja sadrži sve n-torke koje se pojavljuju bilo u R_1 bilo u R_2.

Primer: $S_3 := S_1 \cup S_2$

S_3

BrInd	MLB	Ime	Starost	ŠifSmer
152/97	16309723331981	Ana	19	01
223/95	13975673331981	Mirko	21	01
021/94	11145276418976	Zoran	20	02
003/94	23456786418976	Miloš	22	01

2. ***Diferencija.*** Date su relacije R_1 i R_2 koje zadovoljavaju navedeni uslov kompatibilnosti. Rezultat operacije diferencije

$$R_3 := R_1 - R_2$$

su n-torke relacije R_1 koje nisu istovremeno i n-torke relacije R_2.

Primer: $S_4 := S_1 - S_2$

S_4

BrInd	MLB	Ime	Starost	ŠifSmer
152/97	16309723331981	Ana	19	01

3. ***Presek.*** Date su relacije R_1 i R_2 koje zadovoljavaju navedeni uslov kompatibilnosti. Rezultat operacije preseka

$$R_3 := R_1 \cap R_2$$

je relacija R_3 koja sadrži n-torke koje se pojavljuju u obe relacije R_1 i R_2.

Primer: $S_5 := S_1 \cap S_2$

S_5

BrInd	MLB	Ime	Starost	ŠifSmer
223/95	13975673331981	Mirko	21	01
021/94	11145276418976	Zoran	20	02

4. ***Dekartov proizvod***. Dekartov proizvod se može primeniti na bilo koje dve relacije. Rezultat ove operacije

$$R_3 := R_1 \times R_2$$

je relacija R_3 čije su n-torke svi "parovi" koje čine jedna n-torka relacije R_1 i jedna n-torka relacije R_2.

Dekartov proizvod ćemo prikazati na primeru datom na slici 3.2. Dekartov proizvod se često naziva i "nekontrolisano spajanje" jer, očigledno, može da posluži za spajanje tabela. "Nekontrolisano" je zbog toga što se u rezultatu javljaju sve kombinacije n-torki polaznih relacija, za razliku od operacije spajanja (o kojoj će se kasnije govoriti) koja u rezultatu daje samo kombinacije n-torki koje zadovoljavaju neki uslov.

R_1

C	D	E
1	2	3
2	4	6

R_2

A	B
a1	b1
a2	b2
a3	b3

$R_3 = R_1 \times R_2$

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

Slika 3.2. Dekartov proizvod

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

3.2.1.2. Specijalne relacione operacije

Pored skupovnih operacija u relacionom modelu se definišu i sledeće specifične operacije: projekcija, selekcija, spajanje i deljenje.

5. ***Projekcija.*** Operacija projekcije je unarna operacija koja iz neke relacije selektuje skup navedenih atributa, odnosno "vadi vertikalni podskup" iz odgovarajuće tabele. Formalno se može definisati na sledeći način:

   Neka je $R(A_1, A_2, ..., A_n)$ relacija, a X podskup njenih atributa. Označimo sa Y komplement $\{A_1, A_2, ..., A_n\}$ - X. Rezultat operacije projekcije relacije R po atributima X je

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

Primer: Posmatrajmo tabelu Građanin i njenu projekciju Pr_1. Kada se iz neke relacije selektuje podskup atributa preko operacije projekcije u rezultatu bi se mogli pojaviti duplikati n-torki. Operacija projekcije podrazumeva da se ovi duplikati eliminišu, da bi se zadovoljila definicija relacije, kako je to u navedenom primeru i urađeno. Na Slici 3.3. data je relacija Građanin i rezultat operacije projekcije

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

Građanin

MLB	Ime	Starost	MestoRođ
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

Slika 3.3. Operacija projekcije

6. Selekcija (Restrikcija). Selekcija je takođe 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 $R(A_1, A_2, ..., A_n)$ 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 predikat (uslov) Θ

Primer: Iz relacije Građanin (Slika 3.3.) prikaži građane koji su stariji od 20 godina i rođeni su u Beogradu.

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

Pr_2

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

Originalno se ova operacija nazivala ***restrikcija***, čime se želelo da kaže da uslov ograničava koje n-torke mogu da se jave u rezultatu. Sada se češće koristi naziv ***selekcija.***

7. ***Spajanje (Join***). Spajanje je binarna operacija koja spaja dve relacije na taj način da se u rezultatu pojavljuju oni parovi n-torki jedne i druge relacije koji zadovoljavaju uslov zadat nad njihovim atributima. Formalno se definiše na sledeći način:

   Date su relacije $R_1(A_1, A_2, ..., A_n)$ i $R_2(B_1,B_2, ..., B_m)$ i predikat Θ definisan nad njihovim atributima. Obeležimo sa X i Y skupove atributa relacija R_1 i R_2, respektivno. Rezultat operacije spajanja ovih relacija (tzv. ***teta spajanje***) je

$$R_1[{}_x\Theta]R_2 = \{<x,y> \mid x \in R_1 \text{ AND } y \in R_2 \text{ AND } \Theta(x,y)\}.$$

Oznaka ${}_x\Theta$ za operaciju spajanja ukazuje na činjenicu, očiglednu iz definicije teta spajanja, da ova operacija nije primitivna operacija relacione algebre, već se može izvesti uzastopnom primenom operacije Dekartovog proizvoda (x) nad relacijama koje se spajaju i selekcije po predikatu Θ nad tako dobijenom relacijom.

Ako je predikat Θ definisan sa $A_k = B_j$, s tim da su i atributi A_k i B_j definisani nad istim domenima, tada se takvo spajanje naziva ***ekvispajanje.*** Primer ekvispajanja ćemo prikazati na relacijama Građanin (Slika 3.3.) i Student koju ćemo, delimično modifikovanu, ponovo dati.

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	MestoRođ	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, u gornjem primeru 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. Oznaka za operaciju prirodnog spajanja je *

Pr_4 := Građanin * Student

Pr_4

MLB	Ime	Starost	MestoRođ	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

Na Slici 3.4. prikazana su još dva primera spajanja relacija pod složenijim uslovima.

R_1

C	D	E
1	2	3
2	4	6

R_2

A	B
1	1
2	2
3	3

$R_3 := R_1 [R_1.C > R_2.A] R_2$

C	D	E	A	B
2	4	6	1	1

$R_4 := R_1 [R_1.C < R_2.A \text{ AND } R_1.D > R_2.B] R_2$

C	D	E	A	B
2	4	6	3	3

Slika 3.4. Primeri spajanja relacija

8. ***Deljenje***. Deljenje je operacija pogodna za upite u kojima se javlja reč "svi" ("sve", "sva"). Formalno se definiše na sledeći način:

   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

   $$A[Y \div Z]B = R(X)$$

   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).

Primer (Slika 3.5.): Iz relacija Predmet i Prijava prikaži brojeve indeksa studenata koji su položili sve predmete.

Predmet

ŠifPred
P1
P2
P3

Prijava

Brind	ŠifPred
152/97	P1
152/97	P2
021/94	P1
003/94	P3
152/97	P3

Prijava [Prijava.ŠifPred ÷ Predmet.ŠifPred] Predmet

Brind
152/97

Slika 3.5. Primer za operaciju deljenja

Operacija deljenja nije primitivna operacija relacione algebre, već se može izvesti pomoću drugih operacija na sledeći način:

$$A(X, Y)\,[Y \div Z]B(Z) = \pi_X A - \pi_X ((\pi_X A \times B) - A)$$

Objašnjenje:

- $\pi_X A$ daje sve n-torke koje mogu da učestvuju u rezultatu,
- $(\pi_X A \times B)$ daje relaciju u kojoj se za svaku vrednost z iz B pojavljuju parovi <x,z> sa svim vrednostima x,
- $((\pi_X A \times B) - A)$ ne sadrži ni u jednom paru <x,z> one vrednosti x za koje u relaciji A, kao vrednosti y, postoje sve vrednosti z.

Rezultat je, znači, relacija koja sadrži one n-torke x za koje postoje u paru <x,y>, kao vrednosti y, sve vrednosti z.

25. Navesti operacije sa XML dokumentima. Prikazati onsovni oblik ovih operacija. (redolsed slika predstavlja redosled stranica)

8.5. Operacije sa XML dokumentima

I ovde, kao i u većini modela, mogu se definisati ***navigacione*** i ***specifikacione*** operacije. Pod navigacionim operacijama se podrazumevaju operacije koje omogućavaju "kretanje" kroz XML dokument, odnosno adresiranje pojedinih njegovih komponenti. Specifikacione operacije su u stvari upiti preko kojih se definiše deo XML dokumenta koji se želi u rezultatu i uslov koji takav dokument treba da zadovolji. Skup operacija koje nazivamo »navigacione« operacije definisane su XPath specifikacijom. Upitni jezik XQuery daje konstrukcije za iskaz specifikacionih operacija. Pored toga i ovde se može govoriti o povezivanju XML-a sa konvencionalnim i objektnim programskim jezicima, kao i o specifičnim jezicima koji transformišu jedan XML dokument u drugi. Ovde će se nešto detaljnije govoriti o XPath i XQuery jezicima, a o ostalima će biti date samo osnovne napomene.

8.5.1. XPath

XPath je jezik koji omogućava adresiranje delova ili navigaciju do delova XML dokumenta. Osim toga on omogućava i osnovnu manipulaciju stringovima, brojevima i logičkim podacima. XML dokument se ovde posmatra kao ***stablo čvorova***. Termin čvor se koristi za bilo koji deo XML dokumenta (elementi, atributi,

234

njihove vrednosti, komentari i instrukcije obrade). Pored toga definiše se čvor koji se naziva ***koren dokumenta.*** Koren dokumenta je jedan fiktivni, bezimeni čvor, čije je dete osnovni (koreni) element XML dokumenta.

Čvor se adresira preko tzv. ***izraza putanje***. Izraz putanje je niz od jednog ili više ***koraka*** razdvojenih sa „/". Izraz putanje počinje bilo sa „/" ili sa „//". Ako putanja počinje sa „/" polazni čvor je neimenovani koren XML dokumenta. Ako izraz putanje počinje sa „//" polazni čvor je neki drugi čvor dokumenta dobijen na osnovu imena, bez obzira gde se on u stablu čvorova nalazi. Adresiranje polaznog čvora na osnovu imena podrazumeva određeni implicitni apsolutni izraz putanje čiji je prvi čvor neimenovani koren dokumenta.

U izrazu putanje korak se definiše sa: „naziv_ose :: test_cvora [predikat] "

- ***osa (axis)*** pravac posmatranog koraka putanje u stablu;
- ***test čvora (node test).*** Korak u putanji dovodi do skupa čvorova različitog tipa i naziva. Pomoću testa čvora selektuju se čvorovi određenog tipa ili datog naziva;
- ***predikat.*** Dodatno se selektuju oni čvorovi iz skupa čvorova koji zadovoljavaju dati predikat.

Osa se u koraku označava sa „naziv_ose ::". Ose su:

- **self** - daje sam kontekstni čvor;
- **child** - daje decu kontekstnog čvora;
- **descendant** - daje sve "potomke kontekstnog čvora, decu, decu dece itd.;
- **descendant-or-self** - daje tekući čvor i svu njegovu decu, decu dece itd.;
- **following–sibling** - daje sve sledeće („desne") blizance kontekstnog čvora. Ako je kontekstni čvor čvor-atribut ili čvor-prostor imena, ova osa je prazna;
- **following** - daje sve sledeće („desne") blizance kontekstnog čvora i njihove potomke isključujući čvor-atribute i čvor-prostora imena;
- **attribute** - daje sve čvor-atribute konteksnog čvora
- **namespace** - daje sve čvorove decu kontekstnog čvora koji su iz istog prostora imena kao i kontekstni čvor
- **parent** - daje roditelja kontekstnog čvora, ako on postoji;
- **ancestor** - daje roditelja, roditeljeve roditelje itd., sve do korena stabla;
- **ancestor-or-self** - daje tekući čvor, njegovog roditelja, roditeljeve roditelje itd., sve do korena stabla;
- **preceding** - daje sve prethodne („leve") blizance kontekstnog čvora i njihove potomke isključujući čvor-atribute i čvor-prostora imena;
- **preceding–sibling** - daje sve prethodne(„leve") blizance konteksnog čvora. Ako je konteksni čvor čvor-atribut ili čvor-prostor imena, ova osa je prazna;

Dajemo nekoliko primera izraza putanja,

1. `/child::*` *{daje svu decu korena dokumenta (na osnovu definicije dobro-oformljenog dokumenta kao rezultat ovog koraka može se dobiti samo jedan koreni (glavni) element i nula ili više komentara ili instrukcija obrade)}*

235

2. `/child::Otpremnica` *{daje čvor koji predstavlja glavni element dokumenta čiji je naziv Otpremnica }*
3. `/child::Otpremnica/child::Stavka` *{daje sve čvorove dokumenta koji imaju naziv Stavka i koji su deca glavnog elementa sa nazivom Otpremnica }*
4. `/descendant::Stavka/parent::node()/atribute::*` *{daje sve atribute roditelja za čvorove u dokumentu sa nazivom Stavka bez obzira gde se oni nalaze}*
5. `/child::Otpremnica/child::Datum/following-sibling::*` *{daje sve čvorove bilo kojeg imena ili tipa koji prethode čvoru (leva braća) sa nazivom Datum korenog elementa Otpremnica}*
6. `/descendant::node()/attribute::valuta` *{daje sve čvorove potomke korena dokumenta koji imaju atribut sa nazivom valuta }*
7. `/child::Otpremnica/child::Stavka[position()=3]` *{daje treće pojavljivanje čvora sa nazivom Stavka u okviru Otpremnice}*
8. `/descedent::Datum["2003-04-04"]/parent::node()` *{daje roditelja za svaki čvor u dokumentu koji se naziva Datum i čija je vrednost jednaka "2003-04-04" }*
9. `/child::Otpremnica/child::Stavka[position(2)=4][Vrednost>=1000]` *{daje četvrtu stavku koja je dete čvora Otpremnica ukoliko je njena vrednost veća od 1000 }*
10. `/descedent::Vrednost[atribute::valuta]/ancestor::node()` *{za sve elemente Vrednost koji imaju atribut valuta daje njihove roditelje i roditelja roditelje sve do korena dokumenta}*

Test čvora se zadaje ili preko naziva čvora ili preko oznake tipa čvora koji može biti: comment, text, processing-instruction ili node. Zvezdica se koristi za test čvora koji daje sve čvorove bez obzira na njihovo ime ili tip.

Da bi se omogućilo lakše pisanje izraza putanje, definišu se i odgovarajuće skraćenice. Pošto je osa child najčešće korišćena osa, onda se ona definiše kao difolt tj. može se izostaviti iz opisa koraka. Osa self može se zameniti sa „.“, a osa parent sa „..“. Kao skraćenica za atribut osu koristi se „@“. Tako na primer izrazi definisani u prethodnim primerima se mogu skraćeno zapisati na sledeći način.

```
1. /*
2. /Otpremnica
3. /Otpremnica/Stavka
4. /descedant::Stavka/../@*
5. /Otpremnica/Datum/following-sibling::*
6. /descendant::node()/@valuta
7. /Otpremnica/Stavka[3]
8. /descedent::Datum[text()='2003-04-04']/..
9. /Otpremnica/Stavka[4][Vrednost>=1000]
10. /descedent::Vrednost[@valuta]/ancestor::node()
```

Za korak definisan kao ***descedant-or-self::node()*** definiše se skraćenica “//” kojim se daju svi čvorovi stabla. Ova osa kao što znamo daje tekući čvor , svu njegovu decu i decu dece i često se naziva rekurzivni operator.

- `//Proizvod` *{daje sve čvorove sa nazivom Proizvod}*

- `//SifraProizvoda/text()` *(daje sve čvorove sa nazivom SifraProizvoda i za njih decu koja su tipa tekst tj. vraća sve vrednosti sifre proizvoda koje postoje u dokumentu)*

Definisan je i skup funkcija koje se mogu koristiti u XPath izrazima. Ove funkcije omogućavaju rad sa stringovima i brojevima i dobijanje i formatiranje podataka iz originalnog dokumenta. Svaka od funkcija data je preko svog imena, argumenata i povratne vrednosti. XPath definiše četiri tipa funkcija:

- Funkcije čvorova
- Tekstualne funkcije
- Logičke funkcije
- Numeričke funkcije

Funkcije čvorova. U zavisnosti od redosleda pojavljivanja čvorova u skupu definišu se funkcije koje daju redni broj pojavljivanja čvora u dokument-redosledu. To su funkcije ***last()*** i ***position()***. Funkcija ***position()*** vraća redni broj pojavljivanja čvora u nekom skupu, dok funkcija ***last()*** vraća redni broj poslednjeg pojavljivanja. Funkcija ***id(objekat)*** vraća čvor koji se može identifikovati preko vrednosti argumenta objekat. Ukoliko želimo dobiti pun naziv tekućeg čvora koristićemo funkciju ***name()***, a za dobijanje prostora imena odnosno lokalnog imena čvora koriste se funkcije ***local-name()*** odnosno ***namespace-uri()***. Ukoliko je argument ovih funkcija skup čvorova vraća se podatak samo prvog čvora (prvi čvor po redosledu pojavljivanja u dokumentu). Nad grupom čvorova može se koristiti agregirajuća funkcija ***count***() koja daje ukupan broj čvorova u datom skupu.

Tekstualne funkcije omogućavaju manipulaciju sa stringom. String je bilo koji određeni niz karaktera. Funkcija ***substring(string, p [,d])*** izdvaja podniz iz datog niza (string) počev od pozicije *p* u dužini *d*. Definisane su i funkcije ***substring-before(string, str2)*** i ***substring-after(string, str2)*** koje vraćaju podniz iz datog niza *string* pre, odnosno posle pojavljivanja niza karaktera *str2* u datom nizu. Funkcija ***translate(string, str1,str2)*** omogućava zamenu karaktera u datom nizu string i to zamenjujući sve karaktere koji se pojavljuju u prvom nizu sa karakterima sa odgovarajućih pozicija u nizu str2. Ukoliko želimo iz stringa izbaciti višak blanko karaktera tada se koristi funkcija ***normalize-space(string)***. Ova funkcija zamenjuje sekvencu blanko karaktera sa jednim blanko znakom. Spajanje podnizova karaktera omogućeno je preko funkcije ***concat(str1, str2, [str3...])***.

Na primer,

```
substring('abcd',2)                          vraća 'bcd'
substring-after('aBcd','c')                  vraća 'd'
substring-before('aBcd','B')                 vraća 'a'
translate('Ab--C--','ABC-', 'abc')           vraća 'abc'
normalize-space('a     D')                   vraća 'a D'
```

237

Logičke funkcije. Definišu se i logičke promenljive koje se mogu koristiti u izrazima poređenja i to ***false()*** i ***true()***, a za negaciju se koristi funkcija ***not(boolean)***.

Numeričke funkcije omogućavaju manipulaciju sa brojevima i to su ***round(number)***, ***floor(number)*** i ***ceiling(number)*** koje omogućavaju zaokruživanje realnog broja, odnosno daje veći ili manji ceo broj za dati realni broj. Funkcija ***sum()*** vraća sumu svih čvorova čija je string-vrednost broj.

8.5.2. XQuery - XML upitni jezik

XQuery predstavlja XML upitni jezik. Upiti se formiraju na osnovu nekoliko vrsta izraza, kombinovanih u složenije izraze korišćenjem odgovarajućih operatora. XQuery pretpostavlja da su za dokumente koji se koriste u upitu definisane XML šeme. XQuery koristi sve osnovne tipove podataka definisane XML šemom. Ako šema nije definisana XQuery pretpostavlja difolt šemu u kojoj svi elementi imaju tip **anyType**, a svi atributi **anySimpleType.** Pored toga neophodno je navesti prostor imena iz koga elementi i atributi uzimaju imena. Na primer,

```
import schema "http://www.fon.bg.ac.yu/2003/XMLprimer"
          at "http://www.fon.bg.ac.yu/labis/XML/Otpremnica.xsd"
declare namespace fon="http://www.fon.bg.ac.yu/2003/XMLprimer"
```

a) Upiti nad jednim XML dokumentom kojima se prikazuje prost neizmenjen sadržaj

XQuery definiše funkciju za identifikovanje XML dokumenta, ***document(URI)*** koja identifikuje XML dokument preko URI adrese. XPath obezbeđuje da se iz dokumenta izvuče neki njegov deo koji zadovoljava definisani uslov. Slede primeri upita nad jednim XML dokumentom preko XPath izraza.

Primer: Prikazati sve podatke o svim otpremljenim proizvodima u XML dokumentu otpremnice.xml

```
document("otpremnice.xml")//Proizvod/*
```

Primer: Prikazati sve OtpremljeneKolicine veće od 5 za Otpremnicu sa brojem 10.

```
document("otpremnice.xml")
    //Otpremnica[Broj='10']/Stavka/OtpremljenaKolicina[text() gt 5]
```

Kao što se iz primera vidi upit nad jednim dokumentom se postavlja preko XPath izraza. Operator "veće od" se u predikatu daje kao *gt*.

b) Upiti nad jednim XML dokumentom kojima se prikazuje izmenjeni sadržaj

XPath izrazi omogućavaju samo selektovanje postojećih čvorova XML dokumenta pa se stoga definišu nova vrste upita preko ***ForLetWhereReturn*** (***FLWR***) izraza, koji se čita kao engleska reč "flower".

Opšti iskaz ovog izraza je sledeći:

```
FOR    $iteratorPromenljiva IN xpath_putanja
LET    $promenljiva := xpath_putanja
WHERE  kvalifikacioni_izraz
RETURN xml_konstrukcija
```

U FOR i LET klauzuli definišu se različute vrste promenljivih nad putanjama XML dokumenata.

Promenljive se označavaju sa prefiksom "$". *LET* dodeljuje promenljivoj vrednosti izraza putanje dok *FOR* klauzula definiše iterator promeljivu koja uzima vrednosti iz skupa čvorova definisanog preko putanje. Na primer u izrazu

```
FOR $x IN document("Otpremnica.xml")//Stavka
```

promenljiva **x** uzima kao svoju vrednost redom svaku stavku dokumenta *Otpremnica* , dok u izrazu

```
LET $x := document("Otpremnica.xml")//Stavka
```

promenljiva **x** uzima kao svoju vrednost ceo skup stavki dokumenta *Otpremnica*.

Jedan FLWR izraz može sadržati više FOR i LET klauzula.

WHERE klauzula definiše uslove za selekciju rezultata čija se provera pokreće za svaku vrednost iteratorPromenljive definisane FOR klauzulom.

RETURN definiše XML strukturu rezultata upita. Može se formirati nova struktura stabla kombinujući vrednosti promenljivih i konstrukcije novih čvorova koristeći XML sintaksu.

Za upit "Prikaži sve proizvode koji se nalaze na otpremnici" pogodno je sa LET klauzulom definisati promenljivu koja kao vrednost uzima skup svih proizvoda. U tom slučaju upit samo prikazuje ovaj skup. U primeru koji sledi rezultat će biti prikazan kao novi XML dokument.

```
LET    $x := document("Otpremnica.xml")//Proizvod
RETURN <sviProizvodi>
         $x
       </sviProizvodi>
```

Sledeći upit daje sve proizvode sa Otpremnice čiji naziv počinje sa slovom "P". Pošto je neophodno proveriti svaku stavku definiše se iteratorska promenljiva sa klauzulom FOR.

```
FOR  $a IN document("Otpremnica.xml")//Proizvod
WHERE substring($a/nazivArtikla,1,1)="P"
RETURN <artikli>
          $a/*
       </artikli>
```

U uslovu kao i u rezultujućoj xml konstrukciji mogu se koristiti i funkcije definisane u XPath-u za manipulaciju sa atomskim vrednostima i čvorovima.

Korišćenjem agregatnih funkcija moguće je obraditi XML dokument i prikazati izvedene podatke. Na primer upit koji daje prosečne isporučene količine proizvoda sa sifrom "P22".

```
LET  $x := document("Otpremnica.xml")//Stavka
                  [Proizvod/SifraArtikla="P22"]
RETURN <ukupno>
          avg($x/OtpremljenaKolicina)
       </ukupno>
```

c) Upiti nad više dokumenta

Za identifikaciju kolekcije XML dokumenata nad kojom se postavlja upit koristi se funkcija collection(«URI») koja definiše skup XML dokumenta identifikovan preko iste URI adrese. Upit nad ovako definisanom skupom dokumenta definiše se na isti način kao i nad jednim XML dokumentom. Na primer ako bismo želeli da postavimo upit nad skupom svih Otpremnica sve otpremnice bi stavili na istu URI adresu:

```
http://fon.bg.ac.yu/otpremnice
```

Upit koji bi vratio sve Otpremnice na kojima postoji proizvod sa šifrom "P33" mogao bi se napisati na sledeći način.

```
FOR  $x IN document("http://fon.bg.ac.yu/otpremnice")//Otpremnica
WHERE $x//SifraArtikla="P33"
RETURN <otpremnice>
          $x
       </otpremnice>
```

Ako dokumenti po kojima se želi da postavi upit nisu na istoj URI adresi neophodno je postaviti upit sa operacijom ***spajanja***. Operacije spajanja više XML dokumenata definiše se preko deklaracije iterator promenljivih. Naime u jednoj FOR klauzuli se može definisati više promenljivih koje uzimaju vrednosti iz različitih izra

Na primer ako su na dve URI adrese data dva XML dokumenta: dobavljaci.xml koji sadrži sve podatke o svim dobavljačima i otpremnice.xml koji sadrže podatke o svim otpremnicama tada možemo definisati sledeći upit. Upit «Prikaži sve podatke o dobavljačima i broj i datum otpremnica koje su dobavljači isporučili u 2003 godini» može se napisati na sledeći način

```
FOR  $d IN document("dobavljaci.xml")//Dobavljac,
     $o IN document("otpremnice.xml")//Otpremnica
          [Dobavljac/SifraDobavljaca = $d/SifraDobavljaca]
WHERE substring($o/Datum,1,4) = "2003"
RETURN <dobavljac>
          $d/*,
          <isporuka> $o/Datum,  $o/Broj </isporuka>
       </dobavljac>
```

Kao što se iz upita vidi spajanje dokumenata je izvršeno već pri deklarisanju promenljivih. Izraz za putanju nad kojom je definisana promenljiva «o» koristi predikat koji zahteva da se spoje dobavljači sa njihovim otpremnicama. Isti upit se može iskazati i spajanjem u WHERE uslovu na sledeći način.

```
FOR  $d IN document("dobavljaci.xml")//Dobavljac,
     $o IN document("otpremnice.xml")//Otpremnica
WHERE $o/Dobavljac/SifraDobavljaca = $d/SifraDobavljaca
  AND substring($o/Datum,1,4) = "2003"
RETURN <dobavljac>
          $d/*,
          <isporuka> $o/Datum,  $o/Broj </isporuka>
       </dobavljac>
```

U FLWR izrazima može se koristiti funkcija ***distinct-values(****KolekcijaVrednosti)* koja ima istu ulogu kao i klazula DISTINCT u SQL-u. U svim prethodnim primerima u rezultatu su bili mogući i duplikati. Funkcija distinct-values se može koristiti i pri definisanju promenljivih . Time se postiže da promenljiva iz kolekcije vrednosti uzima datu vrednost samo jednom. Na taj način se mogu realizovati upiti kojima bi u SQL-u odgovarala kaluzula GROUP BY..HAVING. Na primer «Prikaži ukupne otpremljene količine proizvoda koji se pojavljuju više od četiri puta u skupu svih otpremnica»

```
LET $otp := document("otpremnice.xml")
FOR $sifp IN distinct-values($otp//SifraArtikla)
LET $i := $otp//Stavka[Proizvod/SifraArtikla=$sifp]
WHERE count($i) gt 4
RETURN  <proizvod>
          $sifp  ,
          <ukupno> sum($i/OtpremljenaKolicina) </ukupno>
        </proizvod>
```

XQuery je funkcionalni jezik. On omogućava definisanje bilo kojeg izraza kao funkcije čiji su argumenti sekvence i koji vraća sekvence. Sekvenca je uređena kolekcija čvorova i atomskih vrednosti. Tako se i sam FLWR izraz može posmatrati kao funkcija koja vraća sekvencu. Imajući to u vidu jedan FLWR izraz se može koristiti kao argument u drugim funkcijama.

Na primer u okviru RETURN klauzule moguće je definisati novi FLWR izraz. Upit "Formirati pregled izdatih proizvoda za svakog dobavljača" mogao bi se definisati na sledeći način:

```
FOR  $dob IN document("dobavljaci.xml")//Dobavljac
RETURN
  <dobavljac>
    $dob/NazivDobavljaca,
    FOR $i IN document("otpremnice.xml")//Proizvod
        [../../Dobavljac/SifraDobavljaca=$dob/SifraDobavljaca]
    RETURN $i
  </dobavljac>
```

XQuery definiše i operacije nad sekvencama i to operacije unije(*union*), preseka(*itersect*) i razlike(*except*) koje vraćaju sekvence eleminišući duplikate. XQuery omogućava korišćenje kako uslovnih izraza ***if-then-else***, tako i kvantifikujućih izraza (***some***, ***every***). Isto tako moguće je definisati i funkcije. Na primer funkcija koja kao argument dobija skup Stavki a treba da vrati ukupno količine po različitim proizvodima definiše se na sledeći način

```
define function dajProizvode($stavke as element Stavka*)
          as element Proizvod*
{
 FOR $x in distinct-values($stavke/Proizvod/SifraArtikla)
 LET $p := $stavke/Proizvod[SifraArtikla=$x]
 RETURN
     <Proizvod>
       $p,
       <ukupnaKolicina>
          sum($p/../OtpremljenaKolicina)
       </ukupnaKolicina>
     </Proizvod>
}
```

Domen ulaznih parametara funkcije su tipovi podataka definisani XML šemom ili tipovi čvorova (node, attribute itd.). Kao povratna vrednost definiše se sekvenca koja može sadržavati elemente istog tipa. Broj pojavljivanja vrednosti argumenta u parametrima i povratnoj vrednosti funkcije može da sadrži osim tipa komponente i oznaku za broj pojavljivanja elementa u sekvenci i to (+ za 1..M, ? za 0..1 i * za 0..M).

Ostale mogucnosti manipulisanja sa XML dokumentima su operacije transformacije –deklarativne XSLT (eXtebsible Stylesheet Language Transfprmation), Proveduralna obrada XML dokumenta sa DOM (Document Object Model) i SAX (Simpple API fro XML). XML se koristi I kao mehanizam za ostvarivanje integracije u distribuiranim sistemima (npr AJAX).

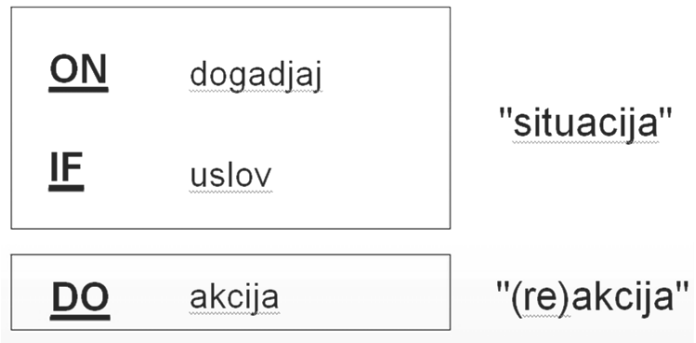
26. Realcioni racun n-torki. Prikazati osnovni oblik operacije
-Relacioni račun n-torki je Predikatski račun prvog reda u kome promenljive uzimaju vrednosti n-torki relacija date baze podataka.
-U RELACIONOM RAČUNU N-TORKI:
Promenljive su n-torke relacija;
Atomske formule se definišu nad atributima n-torki;
Pravila izvođenja su standardna pravila Predikatskog računa prvog reda.

Činjenica da u relacionom računu n-torki promenljiva x uzima kao svoju vrednost n-torku relacije R označava se sa x : R.
Atomi u relacionom računu n-torki su:
• x.A U y.B (x.A je vrednost atributa A relacije R1 za n- torku x koja iz ove relacije uzima vrednost. Na isti način y.B je vrednost atributa B neke druge relacije R2 iz koje promenljiva y uzima n-torke kao svoje vrednosti (x: R1, y: R2). Pretpostavlja se da su atributi A i B definisani nad istim domenom, a U je operacija poređenja definisana nad tim domenom.)
• x.A U c gde su x, A i U kao i u prethodnom stavu, a c je konstanta koja ima isti domen kao i atribut A.

Primeri Prikaži brojeve indeksa i imena studenata koji su stariji od 20 godina.
x: STUDENT
x.BI, x.IME WHERE x.STAROST > 20;

Prikaži imena studenata koji studiraju "Informacioni sistemi". smer
x: STUDENT, y: SMER
x. IME WHERE ∃y (y.ŠSMER = x. ŠSMER AND y.NAZIVS = 'InfSist');

27. Semantika i obrada ECA pravila.
Svako pravilo definiše situaciju i akcije koje se preduzimaju sa nastankom situacije. 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 (AFTER) ili neposredno pre (BEFORE) detektovanja događaja. Složeni događaji formiraju se kombinovanjem primitivnih i prethodno definisanih složenih događaja.

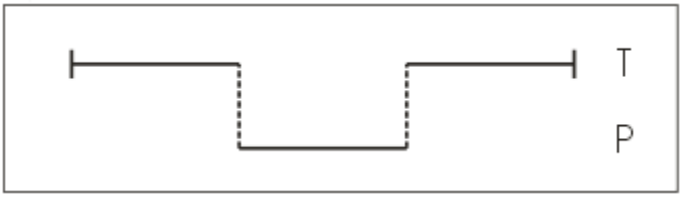
Korisni operatori za kombinovanje događaja mogu biti:
• Logički operatori. Događaji mogu biti kombinovani korišćenjem
logičkih operatora AND, OR, NOT, itd.
• Sekvenca. Pravilo može biti pokrenuto kada se dva ili više događaja pojave u određenom, definisanom redosledu.
• Vremenska kompozicija. Pravilo može biti pokrenuto kombinovanjem vremenskih i događaja koji nisu vremenski, na primer "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 operacije nad bazom podataka i implicitno ili eksplicitno generiše nove događaje.

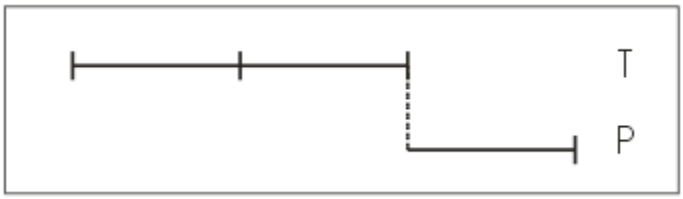
Model znanja treba da uključi i implicitne događaje (pravilo se izvršava uvek (ALWAYS) ili prvi put (FIRST) kada uslov bude zadovoljen), (selektivnu) aktivaciju/deaktivaciju pravila, preciziranje konteksta pravila kroz grupisanje, odnosno bolju organizaciju velikih baza pravila itd.

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



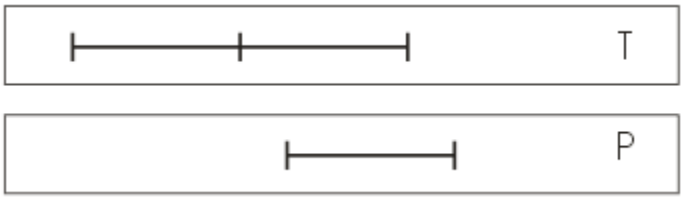
Prouzrokuje **prekid izvršavanja transakcije odmah po identfikovanju događaja** za koji postoji definisano ECA-pravilo, nakon čega se to pravilo proverava.

b) Odložen način:



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

c) Razdvojen način:



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

28. Date su tabele *R*(*P,A*) i *S*(*F,B*). Uz pretpostavku da postoji referencijalno ograničenje: *S.F* je spoljni ključ koji se referencira na primarni *R.P*, napisati triger na nivou naredbe kojim se implementira dinamičko pravilo integriteta "*on update default*" i „*on delete nullifies*".

Odgovor na 4. pitanje:

```
Create trigger on_update_default
After update on R
referencing old table as oldr
for each statement
begin atomic update
s set f = default
where f in (select p from oldr) end;

Create trigger on_delete_nullifies
After delete on R
referencing old table as oldr
for each statement
begin atomic update
s set f = null
where f in (select p from oldr) end;
```

28.a) Navesti i objasniti osnovne elemente grafa prethođenja transakcija. Dati definiciju kada Ti prethodi Tj.

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

1. **Oi prethodi Oj u S**
2. **I Oi i Oj se odnose na isti element baze podataka**
3. **Barem jedna od operacija Oi i Oj je upisivanje**

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

b) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija, a posle i S2. Dati obrazloženje.
S1: r3(X), r3(Y), w3(X), r1(Y), w1(Y), r2(X), w2(X), w2(Y)
S2: r3(X), r3(Y), r1(Y), w1(Y), r2(X), w2(X), w2(Y), w3(X)
(ZA OVE ZADATKE IMA OBJASNJENJE U SKRIPTI ZA BAZU2010 KAKO SE RESAVAJU ALI SAM ISKOPIRAO SVE ZADATKE CISTO ZAVEZBU ☺)

29.b) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija, a posle i S2. Dati obrazloženje.
S1: r3(X), r3(Y), w3(X), r1(Y), w1(Y), r2(X), w2(X), w2(Y)
S2: r3(X), r3(Y), r1(Y), w1(Y), r2(X), w2(X), w2(Y), w3(X)

30.b) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija, a posle i S2. Dati obrazloženje.
S1: r3(X), r3(Y), w3(X), r1(Y), w1(Y), r2(X), w2(X), w2(Y)
S2: r3(X), r3(Y), r1(Y), w1(Y), r2(X), w2(X), w2(Y), w3(X)
(6 poena)
31.a) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija. Dati obrazloženje.
S1: r1(X), r2(X), r2(Y), w2(X), r3(Y), w3(Y), r3(Z), w1(Z).

b) Poređati operacije tako da novodobijeno izvršenje bude konflikt-serijabilno, poštujudi pravila o zameni mesta operacija. Nacrtati graf prethođenja za novo izvršenje.

32. Napisati SQL naredbe kojima se kreira objektno-relaciona tabela *Prodaja* sa atributima knjižara, knjiga i cena, kao i potrebni tipovi. Svi atributi su definisani preko navedenih objektnih tipova.
CREATE TABLE Prodaja (knjizara Knjizara CHAR(30), kjniga CHAR(40) , cena DOUBLE);
- Knjižaru definisati kao *tip vrsta* sačinjenu od polja naziv i lokacija.
CREATE TYPE lokacija (ulica CHAR(30), broj INTEGER, grad CHAR(20));
CREATE TYPE Knjizara (naziv CHAR(30), lokacija ROW(ulica CHAR(30), broj INTEGER, grad CHAR(20)));
- Knjigu definisani kao *struktuirani tip* sa poljima naslov, autor i godinaIzdanja.
CREATE TYPE Knjiga AS (naslov CHAR(30), Autor CHAR(40), godinaIzdanja DATE)
NOT INSTANTIABLE NOT FINAL;
- Cenu definisati preko imenovanog *distinct tipa* sa mogudnocu da implicitne konverzije u INTEGER predefinisani tip.
CREATE TYPE Cena AS DOUBLE FINAL
CAST(DISTINCT AS SOURCE) WITH INTEGER;

33. Date su tabele *R*(*P,A*) i *S*(*F,B*). Uz pretpostavku da postoji referencijalno ograničenje: *S.F* je spoljni ključ koji se referencira na primarni *R.P*, napisati triger na nivou naredbe kojim se implementira dinamičko pravilo integriteta "*on delete cascade*".

34.b) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija, a posle i S2. Dati obrazloženje.
S1: r3(X), r3(Y), w3(X), r1(Y), w1(Y), r2(X), w2(X), w2(Y)
S2: r3(X), r3(Y), r1(Y), w1(Y), r2(X), w2(X), w2(Y), w3(X)

35. Date su tabele Kartica(Proizvod, Datum, TipPromene, Kolicina) i Stanje(*Proizvod*, KolStanje). Atribut TipPromene uzima vrednosti iz skupa ('Ulaz', 'Izlaz').
a) napisati triger kojim se pri brisanju promena u kartici azurira ukupno stanje u tabeli *Stanje*.
b) napisati triger koji se zabranjuje da atribut KolStanje ima negativnu vrednost.

odgovor:
uzas je napisao:

```
CREATE OR REPLACE TRIGGER ZABRANA_BRISANJA_I_AZURIRANJA
BEFORE DELETE OR UPDATE OF Proizvod ON Kartica
FOR EACH ROW
DECLARE
broj INTEGER;
greska EXCEPTION;
CURSOR pronadji (PR NUMBER) IS
        SELECT Proizvod FROM Stanje WHERE Proizvod = PR;
BEGIN
OPEN pronadji (:old.Proizvod);
FETCH pronadji INTO broj;
IF pronadji%FOUND THEN
RAISE greska;
END IF;
CLOSE pronadji;
END;
```

36.b) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija, a posle i S2. Dati obrazloženje.
S1: r1(X), r1(Y), w1(X), r3(Y), w3(Y), r2(X), w2(X), w2(Y)
S2: r1(X), r1(Y), r3(Y), w3(Y), r2(X), w2(X), w2(Y), w1(X)

37. Date su tabele Kartica(Proizvod, Datum, TipPromene, Kolicina) i Stanje(*Proizvod*, KolStanje). Atribut TipPromene uzima vrednosti iz skupa ('Ulaz', 'Izlaz').
a) napisati triger kojim se pri unosu promene u karticu ažurira ukupno stanje u tabeli *Stanje*.
b) napisati triger koji se zabranjuju operacije brisanja i ažuriranja u tabeli *Kartica*.

38.b) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija, a posle i S2. Dati obrazloženje.
S1: r2(X), r2(Y), w2(X), r1(Y), w1(Y), r3(X), w3(X), w3(Y)
S2: r2(X), r2(Y), r1(Y), w1(Y), r3(X), w3(X), w3(Y), w2(X)

39. b) Proveriti da li postoji konflikt-serijabilnost izvršenja S1 skupa transakcija, a posle i S2. Dati obrazloženje.
S1: r2(X), r2(Y), w2(X), r1(Y), w1(Y), r3(X), w3(X), w3(Y)
S2: r2(X), r2(Y), r1(Y), w1(Y), r3(X), w3(X), w3(Y), w2(X)

41.

5. Dat je DTD koji prikazuje učešće studenata na projektima.

```
<!DOCTYPE SP [
<!ELEMENT SP (Projekat*)>
<!ELEMENT Projekat (Naslov, Student+)>
<!ATTLIST Projekat ProjNum ID>
<!ELEMENT Naslov (#PCDATA)>
<!ELEMENT Student>
<!ATTLIST Student BrInd ID Ime CDATA> ]>
```

a) Napraviti UML dijagram klasa ili model objekti-veze kojim se najbolje prikazuje dati DTD. *(4 poena)*
b) Napisati primer XML dokumenta koji je validan u skladu sa datim DTD-om. *(4 poena)*
c) Napisati XPath izraz kojim se prikazuju svi projekti u kojima učestvuje bar jedan student sa imenom "Marko". *(3 poena)*

Za DTD pogledati na sajtu http://www.w3schools.com/dtd/ sta koji tag znaci mada nije tesko:
DOCTYPE je tip
ELEMENT je element
ATTLIST su atributi

42. Objasniti osnove relacionog upitnog jezika QBE (Query by Example).
QBE (Query By Example) je upitni jezik koji predstavlja implementaciju relacionog računa domena, preko specifične "dvodimenzione" sintakse, koja je vrlo bliska korisniku, jer se preko nje, direktno, u skeletu tabele predstavljene na ekranu, zadaje "primer odgovora" koji korisnik želi. Otuda i ime Query By Example (upit na osnovu primera).

U rubrici RELACIJA unosi se ime željene relacije, nakon čega se u rubrikama za ATRIBUTE pojavljuju nazivi atributa te relacije.
U rubrici za OPERACIJE ispod rubrike za relaciju upisuju se operacije koje se odnose na n-torku: **P. (PRINT - prikazivanje), I. (INSERT - ubacivanje), D. (DELETE - izbacivanje) I U. (UPDATE - ažuriranje)**. Ako se neka od operacija odnosi samo na pojedinačne atribute, kod operacije unosi se u koloni odgovarajućeg atributa, gde se unose I uslovi pretraživanja, promenljive I konstante.
Uslov unet u koloni nekog atributa odnosi se samo na taj atribut. Ako su uslovi različitih atributa upisani u istom redu onda su povezani logičkim operatorom AND. Da bi bili povezani logičkim operatorom OR upisuju se u različitim redovima.
U QBE je razvijen koncept za postavljanje upita nad više tabela – **koncept promenljive**. Promenljiva u QBE predstavlja bilo koji niz karaktera koji počinje donjom crtom. Ako se ista promenljiva navede u različitim tabelama, te tabele se spajaju izjednačavanjem vrednosti atributa u kojem je promenljiva definisana (vrši se ekvispajanje).

43. Za dati XML dokument:
<?xml version="1.0"?>
<portfolio xmlns:dt="urn:schemas-microsoft-com:datatypes" xml:space="preserve">

```
  <stock exchange="nyse">
        <name>zacx corp</name>
        <symbol>ZCXM</symbol>
        <price dt:dt="number">28.875</price>
        <share dt:dt="number">1000</share>
  </stock>
  <stock exchange="nasdaq">
        <name>zaffymat inc</name>
        <symbol>ZFFX</symbol>
        <price dt:dt="number">92.250</price>
        <share dt:dt="number">1500</share>
  </stock>
  <stock exchange="nasdaq">
        <name>zysmergy inc</name>
        <symbol>ZYSZ</symbol>
        <price dt:dt="number">20.313</price>
        <share dt:dt="number">2000</share>
  </stock>
</portfolio>
```

a) Napisati XML schemu posmatrajudi dati dokument kao templejt uz sledede pretpostavke: svi atributi su obavezni; redosled oznaka je bitan; atribut „exchange“ moze imati samo vrednosti nyse, nasdaq, i amx.

Odgovor:

```
<xs:attribute name="exchange" use="required">
  <xs:simpleType>
   <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="nyse"/>
        <xs:enumeration value="nasdaq"/>
        <xs:enumeration value="amx"/>
   </xs:restriction> </xs:simpleType>
</xs:attribute>
```

b) Uz pretpostavku da se podaci o validnim XML dokumentima skladiste u relacionoj bazi podataka, definisati relacije kojima se omogucava njihovo skladistenje sa najmanje redudansi.

c) Napisati XQuery izraz kojim se prikazuju sva portfolija koja sadrže deonice (stocks) kompanije „zysmergy inc.“.

44. Poslovna pravila integriteta relacionog modela. Dati za svaki podtip ovih ograničenja primer.

Poslovna pravila integriteta, odnosno specifična ograničenja za dati relacioni model. Naziv "poslovna" proističe iz činjenice da se preko ovih ograničenja iskazuju specifični odnosi vrednosti atributa koji važe u datom realnom (najčešće poslovnom) sistemu.

POSLOVNA PRAVILA INTEGRITETA

Uobičajeno je da se ova pravila integriteta podele na sledeće podtipove:

-Pravila integriteta za domene, preko kojih se specifikuje koje vrednosti postoje u domenu;

CREATE DOMAIN Boja CHAR(6) IN ('Crvena', 'Plava', 'Bela','Zelena');

-Pravila integriteta za atribute, preko kojih se definišu dozvoljene vrednosti nekog atributa nezavisno od vrednosti drugih atributa u bazi;

Tabela student

ATRIBUT	DOMEN	OGRANICENJA
BRIND	*CHAR(7)*	*NOTNULL*
IME	*CHAR(30)*	*--*
STAROST	*INTEGER*	*FOR ALL STRAOST (STAROST >19 AND STRAOST <50)*

-Pravila integriteta za relacije, preko kojih je moguće vezati vrednost jednog, za vrednost drugog atributa u jednoj relaciji;
CREATE INTEGRITY RULE Starost_smer
∀Student (IF Student.ŠSmer = 01 THEN Student.Starost < 28);
-Pravila integriteta za bazu, preko kojih je moguće povezati vrednosti atributa iz više relacija.
CREATE INTEGRITY RULE Ocene_smer
∀Student (IF Student.ŠifSmer = 01 THEN
∃Prijava (Prijava.Ocena >7 AND Student.BrInd = Prijava.BrInd
AND Prijava. ŠifPred = Predmet. ŠifPred AND Predmet. Nazivpred = 'Matematika'.));

45. 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)

46. Date su tabele *R*(*P,A*) i *S*(*F,B*). Uz pretpostavku da postoji referencijalno ograničenje: *S.F* je spoljni ključ koji se referencira na primarni *R.P*, napisati triger na nivou naredbe kojim se implementira dinamičko pravilo integriteta "*on update default*" i „*on delete nullifies*".

47. 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)

48. Objasniti šta je pogled. Koje su osnovne prednosti u koricenju pogleda. Dati uslove koje treba da ispuni pogled da bi mogao da posluži za ažuriranje baze.
Sta je pogled ?
– Pogled je "prozor" kroz koji se vide podaci baze podataka,
– Pogled je virtuelna tabela (sa njim se radi gotovo kao sa baznom tabelom, mada nema svoje podatke i ne zauzima nikakav memorijski prostor).

• Preciznije receno, pogled se koristi kao bilo koja druga tabela pri izvestavanju.
Zasto koristiti pogled ?
– Jednostavnost koriscenja - uproscava upite,
– Tajnost - mocan mehanizam kontrole pristupa podacima,
– Performanse - cuva se u kompajliranom obliku,

– Nezavisnost podataka - menjaju se definicije pogleda, a ne aplikacioni programi koji koriste podatke baze podataka preko pogleda.

• Azuriranje baze podataka preko pogleda ima brojna ogranicenja.
  – Ne moze se vrsiti azuriranje preko pogleda ukoliko je pogled definisan nad vise tabela. Takvo azuriranje bi znacilo da se jednom naredbom vrsI azuriranje vise tabela baze podataka, sto je suprotno definiciji INSERT, DELETE i UPDATE naredbi.
  – Zatim se ne moze vrsiti azuriranje preko pogleda ukoliko se u definiciji pogleda iza SELECT klauzule nalaze funkcije i aritmeticki izrazi.
  – Isto tako, azuriranje se ne moze vrsiti ukoliko u definiciju pogleda nisu ukljucene sve NOT NULL kolone tabele nad kojom je pogled definisan.

Dakle, da bi mogli vrsiti azuriranje preko pogleda:
– pogled mora biti definisan nad jednom tabelom
– u definiciju pogleda moraju biti ukljucene sve NOT NULL kolone te tabele i
– kolone pogleda moraju sadrzati samo prost, neizmenjen sadrzaj kolona tabele nad kojom je pogled definisan.

49. Ukratko objasniti vrste korisničkih i konstruisanih tipova. Kreirati objektno-relacioni model kojim se omogudava skladištenje kolekcije studenata sa jednoznačnim atributima BrojIndeksa, ImePrezime, višeznačnim atributom PoloženiIspiti i referentnom kolonom koja ukazuje na NastavniPlan. NastavniPlan ima atribute ŠifraPlana i DatumOd.
Korisnički definisani tipovi
  Distinct tip - Distinkt tip je jednostavan, perzistentni, imenovani korisnički definisani tip, čijim uvodjenjem je podržano strogo tipiziranje.
  Struktuirani tip - Omogućuje definisanje perzistentnih, imenovanih tipova, koji mogu imati jedan ili više atributa. Atributi mogu biti bilo kog tipa, uključujući druge struktuirane tipove, nizove...
  Metode – predstavljaju ponasanje definisano od strane korisnika. Ima naziv, ulazne parametre,tip povratne vrednosti
Konstruisani tipovi
  Referentni tipovi – tip koji predstavlja referencu na neki objekat, tabelu
  Tip vrsta - niz polja koja čine parovi (<naziv podatka>, <tip podatka>)
  Kolekcija - Kolekcija je grupa koja se sastoji od nula ili više elemenata istog tipa.Broj elemenata kolekcije se naziva kardinalnost kolekcije

Odgovor:
CREATE TABLE Studenti (BrojIndeksa CHAR (10), ImePrezime CHAR (30), PolozeniIspiti CHAR (30) ARRAY [44],REF IS NastavniPlan DERIVED);
CREATE TYPE NastavniPlan AS (SifraPlana CHAR (20), DatumOd DATE) NOT FINAL;

51. Ukratko opisati 4 osnovne komponente svakog modela podataka.
1.Struktura modela odnosno skup kocepata za opsi objekata sistema, njihovihatributa I njihovih medjusobnih veza.
2. Ograničenja na vrednostima 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 modelu;
4.Dinamicka pravila integriteta kojima se definise onsovno dinamicko ponasanje modela. Za svaku osnovnu operaciju azuriranja i svako narusavanje strukture ili ogranicenja koja ona moze da proizvede, definise se akcija koju tada treba peduzeti. Dinamicko pravilo predstavlja trojka <Operacija, Ogranicenje, Akcija>

52. Date su tabele T1(A,B) i T2(C,D). Uz pretpostavku da postoji referencijalno ograničenje: T2.C je spoljni ključ koji se referencira na primarni T1.A, napisati triger kojim se implementira dinamičko pravilo integriteta "on insert default" na nivou naredbe i „on delete cascades" na nivou reda.
Odgovor:
CREATE TRIGGER on_delete_cascade
AFTER DELETE ON T1
REFERENCING OLD AS StariRed
FOR EACH ROW (jer je na nivou reda)
BEGIN ATOMIC DELETE from T2
WHERE C = StariRed.A
END;

53. Operacije u modelu objekti-veze
OPERACIJE
Očigledno je da se u MOV mogu definisati sledeće operacije održavanja baze podataka, analogne operacijama u Mrežnom modelu:
-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 i
-Prevezivanje (Reconnect) pojavljivanja O1 klase A od pojavljivanja O2 klase B.

54.a) Koje su tri karakteristike modela podataka?
b) kako se mogu klasifikovati modeli podataka po kriterijumu nacina opisa dinamike sistema?

a) -Model podataka je intelektualni alat za definisanje modela sistema, za prikazivanje objekata sistema, njihovih atributa i njihovih dozvoljenih vrednosti, medjusobnih veza objekata i dinamike sistema.
-Model podataka je specifičan teorijski okvir pomoću koga se specifikuje, projektuje i implementira neka konkretna baza podataka ili informacioni sistem, uopšte.
-Model podataka je osnova za razvoj Sistema za upravljanje bazom podataka (SUBP)
b) POSTOJI VIŠE KRITERIJUMA ZA KLASIFIKACIJU MODELA PODATAKA:
1.NAČIN OPISIVANJE DINAMIKE SISTEMA
1.KONVENCIONALNI (HIJERARHIJSKI, MREZNI,RELACIONI, MODEL OBJEKTI-VEZE)
-BAZA PODATAKA JE POTPUNO STATIČKI KONCEPT SA IZUZETKOM JEDNOSTAVNIH DINAMIČKIH PRAVILA INTEGRITETA. SVA DINAMIKA JE U APLIKACIJAMA
2.OBJEKTNI

-DINAMIKA SE OBUHVATA NA ISTI NAČIN I U BAZI PODATAKA I U APLIKACIJAMA

3.AKTIVNE BAZE PODATAKA

- PREKO KONCEPTA PRAVILA KOJA SE ISKAZUJU KAO KOMBINACIJA <USLOV, AKCIJA>, ZNATNO VEĆA "KOLIČINA" DINAMIKE SISTEMA SE NALAZI U BAZI PODATAKA

(ZA SVAKI SLUCAJ EVO I OSTALIH KLASIFIKACIJA)

2.NAČIN OSTVARIVANJA OSNOVNIH CILJEVA SUBP-a

OSNOVNI CILJEVI BP:

(1) Neredundatno pamćenje podataka

(2)Višestruko paralelno (konkurentno) korišćenje podataka

(3) Ostvarivanje nezavisnosti programa i logičke i fizičke strukture baze podataka)

KONVENCIONALNE BP – preko sema

OBJEKTNE BP – preko objekata

3.DA LI SE MODEL KORISTI SAMO ZA PROJEKTOVANJE BP, SAMO KAO OSNOVA ZA NEKI SUBP ILI I JEDNO I DRUGO

-ZA PROJEKTOVANJE: MODEL OBJEKTI VEZE, OBJEKTNI MODEL, RELACIONI MODEL

-KAO OSNOVA SUBP-a (IMPLEMENTACIJA): HIJERARHIJSKI, MREŽNI, RELACIONI, OBJEKTNI

-Najčešća kombinacija za razvoj softvera danas: Objektni pristup i jezici za razvoj aplikacija i relacioni SUBP.

4.NAČIN KAKO PRETSTAVLJAJU OBJEKTE I VEZE

-VREDNOSNO ORJENTISANI: vrednosti atributa se koriste i za identifikaciju objekata i za pretstavljanje veza: Relacioni model

-OBJEKTNO ORJENTISANI: Objekti se identifikuju prilikom kreiranja, veze se uspostavljaju preko "pokazivača": Hijerarhijski, Mrežni,Objektni,