

UNIVERZITET U NOVOM SADU  
FAKULTET TEHNIČKIH NAUKA

Dragan Mihajlović

INFORMACIONI SISTEMI I  
PROJEKTOVANJE BAZA PODATAKA

(Odabrana poglavlja za predmet **OSNOVE BAZA PODATAKA**)



<b>7. MODEL OBJEKTI - VEZE - OBELEŽJA</b>	<b>127</b>
<b>7.1. STRUKTURNΑ KOMPONENTA</b>	<b>129</b>
<b>7.2. INTEGRITETNA KOMPONENTA</b>	<b>142</b>
<b>7.3. OPERACIJSKA KOMPONENTA</b>	<b>147</b>
<b>7.3.1. OPERACIJE IZVEŠTAVANJA</b>	147
<b>7.3.2. OPERACIJE ODRŽAVANJA BAZE PODATAKA</b>	148
<b>7.3.2.1. OPERACIJE NAD KLASAMA OBJEKATA I KLASAMA PRESLIKAVANJA</b>	148
<b>7.3.2.2. OPERACIJE NAD BAZOM PODATAKA - STRUKTURNΑ PRAVILA INTEGRITETA</b>	150
<b>7.4. PROCES IZRADA MOV</b>	<b>157</b>
<b>7.4.1. ODREDJIVANJE TIPOVA OBJEKATA</b>	160
<b>7.4.2. ODREDJIVANJE TIPOVA VEZA</b>	162
<b>7.4.3. INTEGRISANJE PODMODELΑ</b>	166
<b>8. RELACIONI MODEL PODATAKA</b>	<b>177</b>
<b>8.1. OPERACIJSKA KOMPONENTA</b>	<b>186</b>
<b>8.1.1. RELACIONA ALGEBRA</b>	186
<b>8.2. NUL - VREDNOSTI</b>	<b>196</b>
<b>8.3. INTEGRITETNA KOMPONENTA</b>	<b>197</b>
<b>8.3.1. INTEGRITET OBJEKTA</b>	198
<b>8.3.2. REFERENCIJALNI INTEGRITET</b>	198
<b>8.3.3. IMPLEMENTACIJA INTEGRITETSKIH OGRANIČENJA</b>	199
<b>8.4. NORMALIZACIJA</b>	<b>203</b>
<b>8.4.1. METODE NORMALIZACIJE</b>	204
<b>8.4.2. DEKOMPONICIJA BEZ GUBITKA INFORMACIJA</b>	206
<b>8.4.3. VERTIKALNA NORMALIZACIJA DEKOMPONICIJOM</b>	208
<b>9. JEZICI ZA MANIPULISANJE PODACIMA U RELACIONOM MODELУ</b>	<b>219</b>
<b>5. KONCEPCIJA BAZE PODATAKA</b>	<b>103</b>
<b>5.1. JEZIK BAZE PODATAKA</b>	<b>112</b>
<b>5.1.1. JEZIK ZA OPIS PODATAKA</b>	114
<b>5.1.2. JEZIK ZA MANIPULISANJE PODACIMA</b>	115
<b>5.2. PRINCIPI RADA SISTEMA ZA UPRAVLJANJE BAZOM PODATAKA</b>	<b>115</b>
<b>6. MODELI PODATAKA</b>	<b>119</b>

<b>9.1. OBRADA UPITA</b>	<b>219</b>	<b>II.1. PARALELNO IZVRŠAVANJE TRANSAKCIJA</b>	<b>302</b>
<b>9.2. STANDARDNI UPITNI JEZIK - <i>SQL</i></b>	<b>221</b>	<b>II.2. REŠAVANJE PROBLEMA PARALELNOG IZVRŠAVANJA TRANSAKCIJA</b>	<b>304</b>
9.2.1. NAREDBE ZA DEFINISANJE PODATAKA	225		
9.2.1.1. KREIRANJE NOVE TABELE - <i>CREATE TABLE</i>	225		
9.2.1.2. KREIRANJE POGLEDA - <i>CREATE VIEW</i>	227		
9.2.1.3. PROMENA STRUKTURE TABELE NAKON ŠTO JE KREIRANA - <i>ALTER TABLE</i>	229		
9.2.1.4. BRISANJE TABELE IZ BAZE PODATAKA - <i>DROP TABLE</i>	230		
9.2.1.5. INDEKSI	231		
9.2.2. NAREDBE ZA MANIPULISANJE PODACIMA	232		
9.2.2.1. DODAVANJE NOVIH N-TORKI U TABELOU - <i>INSERT</i>	232		
9.2.2.2. PRETRAŽIVANJE RELACIONE BAZE PODATAKA - <i>SELECT</i>	235		
9.2.2.2.1. PRETRAŽIVANJE JEDNE TABELE UZ PRIKAZ NEIZMENJENOG SADRŽAJA	237		
9.2.2.2.2. PRETRAŽIVANJE JEDNE TABELE UZ OBLIKOVANJE DOBILJENIH PODATAKA	243		
9.2.2.2.3. ULAGANJE UPITA NAD JEDNOM TABELOM U UPIT NAD DRUGOM TABELOM	252		
9.2.2.2.4. POVEZIVANJE VIŠE TABELE	256		
9.2.2.2.5. PRETRAŽIVANJE TABELE KOJE U SEBI SADRŽE STRUKTURU TIPA TABLA	261		
9.2.2.3. IZMENA SADRŽAJA TABELE - <i>UPDATE</i>	263		
9.2.2.4. BRISANJE N-TORKI TABELE - <i>DELETE</i>	266		
9.2.3. REČNIK PODATAKA	267		
<b>9.3. JEZIK UPITA NA OSNOVU PRIMERA - <i>QBE</i></b>	<b>269</b>		
<b>9.4. TRANSFORMACIJA UPITA</b>	<b>272</b>		
 <b>10. PREVODJENJE MODELA OBJEKTI-VEZE OBELEŽJA U RELACIONI MODEL</b>	 <b>277</b>		
 <b>10.1 POSTUPAK PREVODJENJA</b>	 <b>278</b>		
 <b>11. OSNOVE OBRADE TRANSAKCIJA</b>	 <b>299</b>		

Tabela 4.3.

KRITERIJUM	PREPORUKA
Kohezija	Kohezija svakog modula treba da je što jača. Treba koristiti funkcionalno kohezivne module ili u najmanju ruku sekvensijalne ili komunikacione module.
Povezanost	Povezanost između modula treba da je što slabija.
Podela odluke	Deo u kome se donosi odluka ne treba razdvajati od dela gde se odluka izvršava.
Editovanje	Editovati na sukcesivnim nivoima, a najprije editovanja vršiti na najnižim nivoima.
Poruke o greškama	Poruke o greškama držati na jednom mestu.
Faktorisanje	Podela na što jednostavnije delove.
FAN-IN	Što više nadredjenih modula.
FAN-OUT	Ograničiti broj podredjenih modula na najviše sedam.
Internamemorija	Ne treba je koristiti bez većeg razloga.

**FAN-IN**

FAN-IN kod jednog modula označava broj njegovih neposredno nadredjenih modula.

Treba izbegavati situaciju gde se nadredjeni moduli nalaze na različitim nivoima.

Moduli sa FAN-IN moraju imati dobru koheziju: funkcionalnu ili što je lošije, sekvensijalnu ili komunikacionu. Svaki interfejs jednog modula (koji ima više nadredjenih modula) mora da ima isti broj i tipove parametara.

Sumarni pregled prethodno razmatranih dodatnih kriterija kohezije i povezanosti između modula dat je u tabeli 4.3.

Tabela 4.3.

**5. KONCEPCIJA BAZE PODATAKA**

Koncepcija baze podataka predstavlja korak u evoluciji rešavanja problema u strukturiranju, organizovanju, i korišćenju podataka pri automatskoj obradi podataka koje prethodni sistemi nisu mogli da reše na zadovoljavajući način.

Pojam *baze podataka* pojavio se krajem 60-tih godina. U oblasti automatske obrade podataka tada se govorilo o datotekama i skupovima podataka. I kao što je to čest slučaj da se novi pojmovi pomodno i nepravilno upotrebljavaju tako su i mnogi korisnici za svoje datoteke počeli upotrebljavati naziv baza podataka.

U vreme 60-tih godina javlja se potreba obrade na računaru velike količine podataka koji se susreću pre svega u finansijsko-ekonomskim problemima. Do tada računari su se najčešće koristili za rešavanje naučno-tehničkih problema koji se odlikuju velikim brojem numeričkih operacija nad relativno malim skupovima podataka.

Za potpuno razumevanje pojma baze podataka dalje se daje kratak pregled evolucije metoda organizacije podataka.

Do pojave računara treće generacije programска podrška realizovala je u osnovi operacije ulaza-izlaza na memoriske jedinice sa neznatnim pomoćnim sredstvima obrade podataka. Podaci su obično organizovani u sekvensijalnim datotekama na magnetnim trakama. Obrada podataka odvijala se u paketnom režimu bez mogućnosti pristupa podacima u realnom vremenu. Kada se menjala organizacija podataka ili memoriska jedinica što je bilo neizbežno zbog pojave novih efikasnijih tehnoloških rešenja morali su se menjati *aplikacioni programi* i isti ponovo prevoditi i testirati.

Pod aplikacionim programom podrazumevamo program napisan od strane programera na nekom programskom jeziku, namenjen za rešavanje nekog konkretnog problema na računaru.

Isti podaci retko su korišćeni u više aplikacija, a u jednoj aplikaciji koristi se obično više datoteka.

Pod *aplikacijom* se podrazumeva skup programa, datoteka i pravila za korišćenje namenjenih za određenu primenu računara u praćenju ponašanja jednog dela realnog sistema.

Pri ažuriranju datoteke formirana je nova datoteka sa čuvanjem po više ranijih verzija iste datoteke.

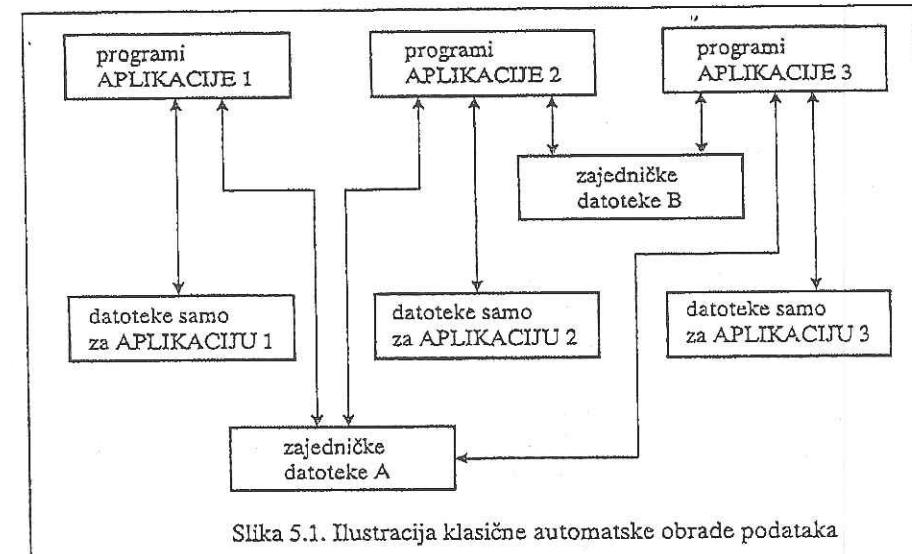
Redundansa (suvišnost, preopširnost, ponavljanje) podataka u jednom informacionom sistemu bila je vrlo velika zbog postojanja više datoteka sa istim podacima.

Kraj 60-tih godina karakteriše organizacija podataka nastala korišćenjem novih memorijskih jedinica i metoda pristupa. Želi se izbeći uticaj karakteristika memorijskih jedinica na aplikacione programe koji zadržavaju opis i logičke i fizičke strukture podataka datoteka koje koriste. Ako istu datoteku koristi više aplikacionih programa i ako se zbog potrebe jednog od njih izmeni logička ili fizička struktura te datoteke, moraju se menjati i svi drugi programi. S obzirom na to da u praksi ovakve situacije nastaju dosta često dolazi se u situaciju kada programeri veći deo svog vremena troše na održavanje (menjanje, prevodjenje, testiranje) postojećih programa.

Jedna datoteka bi se mogla koristiti u više aplikacija što bi moglo skratiti vreme obrada i smanjiti zauzeće memorijskog prostora, to se pak vrlo retko koristi iz više razloga: u novoj aplikaciji postoji potreba za podacima koji nisu prisutni u postojećoj datoteci ili se zahteva takav pristup podacima koji u postojećoj datoteci nije predviđen. Nastaje situacija u kojoj je jednostavnije za nove potrebe formirati posebnu datoteku nego u desetinama programa vršiti promene. Posledica navedenog pristupa je nastajanje novih redundantnih datoteka. Na taj način u jednom informacionom sistemu nastaje više nezavisnih aplikacija koje kreiraju i održavaju posebne datoteke sa svim potrebnim podacima. Takva automatska obrada podataka naziva se klasičnom. Susreće se i danas. Ima svojih prednosti i nedostataka u odnosu na druge pristupe. Na slici 5.1. šematski je ilustrovan odnos programa i datoteka više aplikacija u klasičnoj automatskoj obradi podataka. Kao što se vidi klasična automatska obrada podataka ne isključuje i korišćenje zajedničkih datoteka. Izmedju skupova datoteka koje se koriste samo za pojedine aplikacije postoji dosta istih podataka obično sa različitim formatom zapisa ili metodama pristupa.

Osnovni nedostaci klasične automatske obrade podataka su:

- nepovezanost aplikacija,
- redundantnost podataka,
- čvrsta povezanost programa i podataka.



Slika 5.1. Ilustracija klasične automatske obrade podataka

Osnovna prednost klasičnog pristupa leži u jednostavnosti projektovanja i realizacije.

Informacioni sistem sa više nepovezanih aplikacija i posebnim datotekama za svaku aplikaciju vremenom obično vrlo brzo dolazi u stanje neusaglašenosti podataka mada se za pojedinačne aplikacije može reći da funkcionišu dobro. Zbog vremenske neusaglašenosti pri ažuriranju podataka nastaju situacije u kojima ista obeležja istog objekta (entiteta) dobijaju različite vrednosti u različitim datotekama. Na primer, za istu osobu u više datoteka nalaze se različiti podaci o adresi.

Potrebe u obradi poslovnih podataka zahtevale su aplikativna programska rešenja nezavisna ne samo od jedinica za memorisanje podataka i povećanja veličine datoteka već i od dodavanja novih obeležja i podataka u već memorisanim podacima kao i novih uzajamnih veza. Integracijom aplikacija mogu se ublažiti neke loše osobine klasične automatske obrade podataka kao što je redundantnost i nekozistentnost podataka, ali bi suštinski problem čvrste

povezanosti programa i podataka ostao time bez rešenja. Prišlo se integriranju datoteka samostalnih aplikacija što je nazvano *bazom podataka*.

Osnovni ciljevi koncepcije baze podataka su:

- sve podatke jednog informacionog sistema treba integrisati u jednu fizičku strukturu podataka,
- svi aplikacioni programi pri obradi baze podataka koriste standardizovane softverske rutine nazvane *sistem za upravljanje bazom podataka*.

Integracija podataka više aplikacija ostvaruje se formiranjem nove logičke strukture nazvane šemom baze podataka (globalna logička organizacija) nad skupom obeležja tipova zapisa datoteka kao struktura nad skupom zapisa različitog tipa.

Šema predstavlja apstraktни model realnog sistema i njegove baze podataka. Nad šemom se gradi odgovarajuća fizička struktura podataka koja predstavlja samu bazu podataka. Svaki program korišćenjem šeme i SUBP koristi ili menja stanje baze podataka. S obzirom da jedinstvena šema baze podataka treba da zadovolji potrebe svih programa obično nastaje veoma kompleksna fizička struktura baze podataka. Drugu dimenziju kompleksnosti fizičke strukture baze podataka nameće potreba memorisanja veza (odnosa) između pojava zapisa različitih tipova. Zahvaljujući SUBP programerim su zaštićeni od problema vezanih za takve kompleksne strukture podataka.

U klasičnoj automatskoj obradi podataka programeri su vodili brigu o samo svojoj aplikaciji. Korišćenje baze podataka kao osnove svih programa i najznačajnijeg dela informacionog sistema ne može se u potpunosti poveriti programerima. Jedan dobar SUBP sam za sebe još nije dovoljan, već ga treba na odgovarajući način koristiti na celishodno izgradjenoj bazi podataka. Ne mogu se jednostavno prepustiti dobroj volji korisnika poslovi kao što su:

- stavaranje uslova za efikasnu obradu podataka,
- konzistentno sprovodjenje promena,
- redovna izrada sigurnosnih kopija baze podataka,
- regulisanje prava pristupa podacim i sl.

Navedene poslove obavlja *administrator baze podataka* (DBA - data base administrator).

Administrator baze podataka je osoba odgovorna za specificiranje, projektovanje, implementiranje, efikasan rad i održavanje baze podataka.

Identifikovanje posebne uloge administratora baze podataka sledi iz koncepta nezavisnosti podataka i iz shvatanja da baza podataka čini značajan i vredan zajednički resurs. Administrator baze podataka radi na sledećim poslovima:

- sa korisnicima pri ustanovljavanju zahteva za bazu podataka, u sklopu aktivnosti specifikacije sistema;
- koristi jezik za opis podataka za definisanje baze podataka, u sklopu aktivnosti projektovanja sistema;
- radi sa programerima čiji programi treba da pristupaju bazi podataka;
- odgovoran je za punjenje baze podataka podacima, u sklopu aktivnosti implementacije sistema;
- nadgleda rad baze podataka, korišćenjem raspoloživih uslužnih programa, da bi odredio kada podatke treba reorganizovati ili izvršiti ponovo projektovanje baze podataka.

Osnovni ciljevi koje želi da postigne administrator baze podataka su:

- integritet baze podataka,
- bezbednost i efikasnost.

*Integritet baze podataka* je stanje baze podataka u kojem su sve vrednosti podataka korektnе, u smislu da:

- a) odslikavaju stanje realnog sveta i to do zadata tačnosti i pravovremenosti,
- b) poštuju pravila uzajamne konzistentnosti.

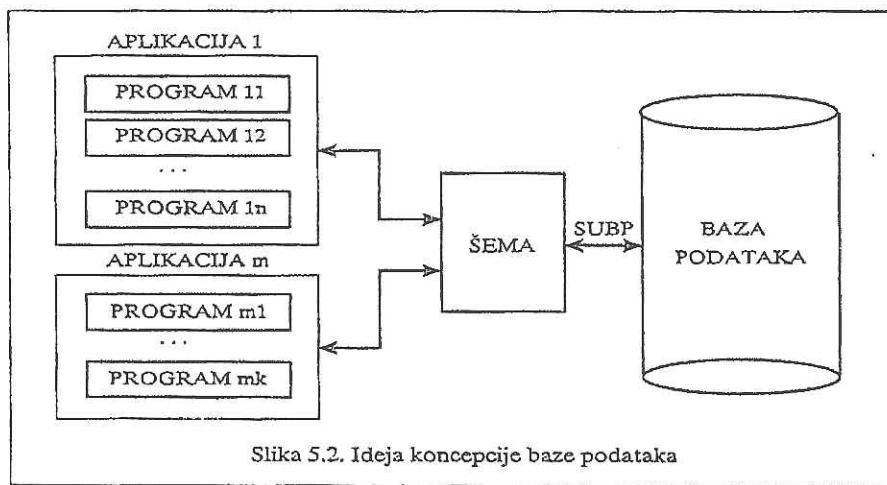
Održavanje integriteta baze podataka podrazumeva proveru integriteta i oporavak iz bilo kojeg nekorektnog stanja koje može biti otkriveno. Za održavanje integriteta baze podataka zadužen je administrator baze podataka, koji pri tome koristi pogodnosti SUBP.

*Bezbednost baze podataka* podrazumeva sprečavanje ili zaštitu od:

- a) neovlašćenog pristupa podacima,
  - b) namernog i neovlašćenog uništavanja ili menjanja istih.
- Bezbednost baze podataka potrebna je zbog zaštite kako od nemernih, tako i od namernih pokušaja pristupa poverljivim informacijama.

Administrator baze podataka posebno se stara da uravnoteži konfliktne zahteve krajnjih korisnika i programera koji proističu iz činjenice da više različitih aplikacija može imati zajedničku bazu podataka.

Zahvaljujući identifikovanim potrebama za postojanjem administratora baze podataka, programeri su bili oslobođeni od poslova projektovanja fizičke strukture podataka, generisanja i održavanja baze podataka.



Slika 5.2. Ideja koncepcije baze podataka

Posle određenog iskustva u korišćenju prvih SUBP i baza podataka (smatra se da je to period do 1975 godine) uočena je potreba dodatnog stepena nezavisnosti programa i podataka. Po pravilu logička struktura podataka je složena, a sa novim potrebama i korisnicima ona se menja. Zbog toga je važno omogućiti promene u šemi baze podataka bez promena u svim programima koji je koriste. Pokazala se dakle potreba za uvodjenjem sledeća dva nivoa nezavisnosti programa i podataka:

- logička nezavisnost,
- fizička nezavisnost.

*Logička nezavisnost* znači da izmene šeme ne smeju uticati na izmene programa (naravno, pod uslovom da se ne menjaju obeležja koja program baš koristi). Logička nezavisnost se postiže uvodjenjem pojma podšeme.

*Podšema* predstavlja pojam koji se odnosi na onaj deo logičke strukture obeležja baze podataka, koji je dovoljan za realizaciju jedne ili više aplikacija.

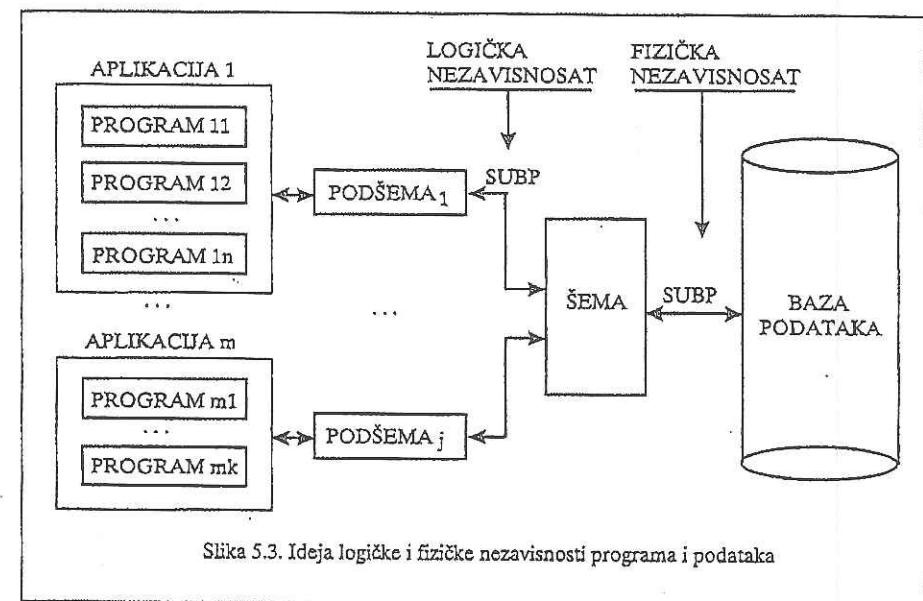
Šema i podšema predstavljaju modele podataka realnog sistema i jednog njegovog dela. U principu, ta dva modela se nalaze na istom nivou apstrakcije.

Mada u određenom smislu, podšema može predstavljati model podataka na višem nivou apstrakcije od šeme. Naime, podšema sadrži tipove zapisa koji se mogu formirati od obeležja različitih tipova zapisa šeme. Dok tipovi zapisa šeme predstavljaju modele klase entiteta realnog sistema, tipovi zapisa podšema mogu predstavljati model neke kombinacije klasa entiteta. U podšemama definisane odnose SUBP prilagodjava formi šeme baze podataka i obavlja potrebne konverzije. Na primer, realni sistem može sadržati klase entiteta BRODOVI, AVIONI i KAMIONI, a podšema može sadržati generalizirani tip PREVOZNA-SREDSTVA sa obeležjima, koja su zajednička za sve tri klase entiteta.

Pojam logičke nezavisnosti odnosi se na činjenicu da izmene šeme baze podataka ne smeju dovoditi do izmena svih podšema i svih programa.

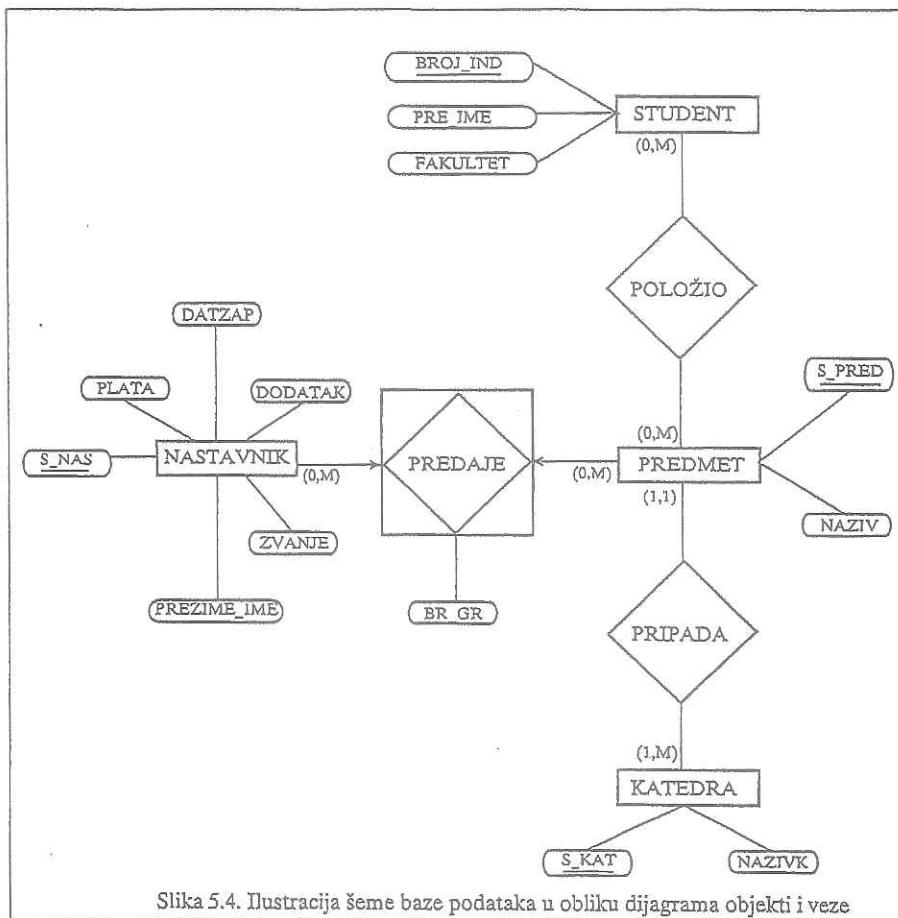
*Fizička nezavisnost* podataka znači da se fizički raspored i organizacija podataka mogu menjati, a da se pri tome ne mora menjati šema, podšema i programi.

Na slici 5.3 ilustrovana je ideja realizacije logičke i fizičke nezavisnosti programa i podataka.



Slika 5.3. Ideja logičke i fizičke nezavisnosti programa i podataka

U fazi projektovanja šemu i podšemu obično predstavljamo u grafičkom obliku pomoću dijagrama u kojima se koriste međusobno povezani blokovi, dok u fazi realizacije baze podataka vršimo prevodjenje takve grafičke predstave na opis šeme pomoću jezika za opis podataka. Postoji više načina za predstavljanja šeme baze podataka u grafičkom obliku. Danas često primenjivan način za predstavljanje šeme i podšema baze podataka u obliku dijagrama su *dijagrami objekti i veze* (DOV). Bez ulaženja u tačno značenje simbola koji se primenjuju pri crtanjtu dijagrama objekti i veze o čemu će biti više reči kasnije, radi ilustracije predstavljanja šeme, na slici 5.4 data je šema jedne hipotetičke baze podataka u obliku DOV.



Tip objekta NASTAVNIK sadrži obeležja: šifra nastavnika (S\_NAS), prezime i ime (PREZIME\_IME), zvanje (ZVANJE), plata (PLATA), datum zaposlenja (DATZAP), iznos posebnog dodatka (DODATAK).

Tip objekta PREDMET sadrži obeležja: šifra predmeta (S\_PRED) i naziv predmeta (NAZIV).

Tip objekta STUDENT sadrži obeležja: broj indeksa (BROJ\_IND), prezime i ime (PRE\_IME) i fakultet koji pohadja (FAKULTET).

Tip objekta KATEDRA sadrži obeležja: šifra katedre (S\_KAT) i naziv katedre (NAZIVK).

Na šemi sa slike 5.4. vide se i međusobne veze tipova objekata, imenovane sa: PREDAJE, POLOZIO i PRIPADA. Tip veze PREDAJE sadrži obeležje: broj grupe (BR\_GR) koje označava koliko grupa predavanja ima jedan nastavnik na jednom predmetu. Označene su i kardinalnosti povezanih tipova objekata.

Šema baze podataka ilustrovana na slici 5.4. može se predstaviti u relacionom modelu podataka sledećom šemom baze podataka:

NASTAVNIK (S\_NAS, PREZIME\_IME, ZVANJE, DATZAP, PLATA, DODATAK),  
 PREDMET (S\_PRED, NAZIV, S\_KAT),  
 STUDENT (BROJ\_IND, PRE\_IME, FAKULTET),  
 KATEDRA (S\_KAT, NAZIVK),  
 PREDAJE (S\_NAS, S\_PRED, BR\_GR),  
 POLOZIO (BROJ\_IND, S\_PRED).

Pitanja kako se određuje šema baze podataka u jednom IS u obliku DOV, kako se DOV prevode u relacionu šemu baze podataka i kako se relaciona šema predstavlja na jeziku za opis podataka biće u kasnijim razmatranjima detaljnije prikazana.

Za jednu bazu podataka definiše se jedna šema baze podataka. Nad jednom šemom može biti definisano više podšema. Jednu podšemu može koristiti više programa.

Šema baze podataka se naziva i globalnom šemom ili konceptualnom šemom. Podšema se naziva i eksternom šemom. Opis fizičke strukture baze podataka naziva se i fizičkom šemom ili internom šemom.

### 5.1. Jezik baze podataka

Da bi SUBP praktično realizovao svoje funkcije on sadrži jezike za definisanje i pristupanje bazi podataka. Jezik baze podataka sadrži jedan ili više jezika za opis podataka i jedan ili više jezika za manipulaciju podacima. Tela koja se bave standardizacijom jezika baze podataka obuhvataju:

- ANSI/SPARC (American National Standards Institute/Systems Planning and Requirements Committee - Američki nacionalni institut za standarde - Komitet za sistemsko planiranje i zahteve sistema)
- CODASYL (Conference on Data Systems Language - Konferencija o jezicima podataka),
- ISO (International Organization for Standards - Medjunarodna organizacija za standarde).

Kako je razmatranje jezika baze podataka povezano sa korisnicama baze podataka i SUBP, kratko razmotrimo osobine i potrebe korisnika baza podataka.

Razlikujemo sledeće dve glavne kategorije korisnika baza podataka:

- aplikativni programeri,
- korisnici koji ne programiraju.

Aplikativni programeri samostalno sastavljaju konkretne programe na osnovu zahteva korisnika ili za svoje potrebe. Zahtevi su unapred poznati, broj im je ograničen te se u celosti mogu unapred isprogramirati. Bazu podataka na ovaj način mogu koristiti samo iskusni programeri.

Većina korisnika, međutim, nisu programeri, uopšte ne znaju programirati ili raspolažu samo minimalnim znanjem programiranja. Njih takođe možemo podeliti u dve grupe. U prvu grupu spadaju korisnici čiji zahtevi imaju ad hoc prirodu. Oblik i sadržina njihovih zahteva se uvek formulišu u zavisnosti od situacije i prema tome ne mogu se unapred sastaviti odgovarajući programi.

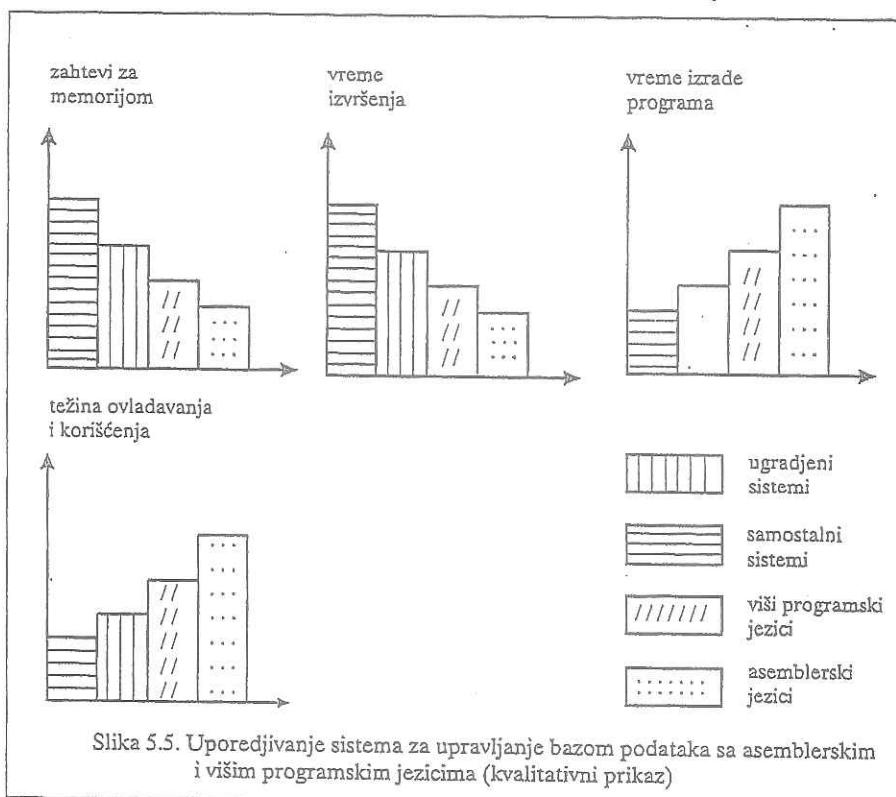
Za njih je potreban jedan jednostavan jezik za rukovanje podacima koji i početnik u računarskoj tehnici može lako naučiti, a bazira se prvenstveno na opisu jednostavnih logičkih veza i osnovnih matematičkih operacija. Uz pomoć ovog jezika korisnici sami definišu načine povezivanja podataka za svoje potrebe kao i radnje koje treba obaviti sa podacima.

U drugu grupu spadaju oni korisnici čiji se zahtevi za informacijama odnose na unapred odredjene, zaokružene teme i skup pitanja. Oni se zovu korisnici koji se služe parametrima. Njima nije potreban ni taj jednostavan jezik da bi svoja pitanja postavili. Zadavanjem aktuelnih parametara na nekoliko unapred isprogramiranih pitanja dobijaju traženu informaciju (npr. rezervacija avionskih karata, šalterski terminal u banci).

U zavisnosti od toga, čije zahteve od navedene dve kategorije korisnika se žele prvenstveno zadovoljiti razlikuju se sledeće varijante SUBP:

- a) *ugradjeni sistemi* (Host Language Systems) - koriste se kao proširenje nekog postojećeg programskega jezika koji se naziva *matičnim jezikom*. Samostalno se ne mogu koristiti. Pogodni su samo za opis šeme baze podataka i za definisanje ulazno-izlaznih operacija. Svi ostali operacioni zahtevi u vezi podataka se definišu na nekom programskom jeziku višeg nivoa (COBOL, PL/1, FORTRAN). Iz tog višeg programskega jezika SUBP se poziva naredbama tipa CALL, da bi aplikativnom programu predao ili od njega preuzeo željene podatke. Na jezicima COBOL i PL/1 u koje se sistemi najčešće ugradjuju, te naredbe spadaju u prošireni deo jezika koji je namenjen za rukovanje bazom podataka i koje programeri koriste na isti način kao i ostale naredbe jezika.
- b) *samostalni sistemi* (Self-contained Systems) - koriste sopstveni jezik nezavisno od bilo kojeg drugog programskega jezika. Rešavanje problema se definiše putem ovog jezika koji se može lako naučiti. Ti jezici su, svakako, znatno siromašniji od programskih jezika višeg nivoa, stvaraju programe koji troše znatno više mašinskog vremena, međutim njihova primena je znatno prikladnija pogotovu za korisnike koji nisu programeri. Korisnik manipuliše podacima samo na logičkom nivou. Zbog toga se opis celokupne baze podataka zajedno sa imenima podataka nalazi u sistemu i u programima ne treba definisati nikavu strukturu podataka.
- c) *kombinovani sistemi* - koriste obe prethodne načina.

Na slici 5.5 prikazana su kvalitativna poređenja samostalnih i ugradjenih SUBP sa programskim jezicima višeg i nižeg reda po različitim kriterijumima.



### 5.1.1. Jezik za opis podataka

Nezavisnost podataka praktično ostvarujemo pomoću jezika za opis podataka (JOP) (DDL - Data Definition Language).

Jezik za opis podataka predstavlja sredstvo za opis podataka na tri različita nivoa:

1. Na nivou korisnika u vidu neke podšeme - jezik za opis podataka podšeme (JOP - podšeme).
2. Na nivou logičke strukture podataka tj. opis kompletne šeme baze podataka - jezik za opis podataka šeme (JOP - šeme).
3. Na nivou fizičke strukture podataka - jezik za opis fizičke strukture podataka (JFS). Na tom nivou se opisuje način fizičkog pristupa podacima, memorisanje podataka i fizička veza između podataka (zone, stranice, pointeri, indeksi, rutine za hašing, itd.).

Jezik za opis podataka je neki poseban jezik ili je sličan nekom od viših programskih jezika, COBOL na primer. Program napisan na JOP se prevodi tako da ga mogu koristiti rutine za upravljanje podacima i aplikacioni programi. Razlike između JOP šeme i JOP podšeme su male.

### 5.1.2. Jezik za manipulisanje podacima

Jezik za manipulisanje podacima (JMP) (Data Manipulation Language - DML) omogućava korisniku izvršavanje operacija nad bazom podataka koja je definisana JOP. Te operacije omogućavaju upisivanje novih zapisu, učitavanje zapisu u radnu zonu programa, upisivanje izmenjenog zapisu u bazu podataka, potpuno ili delimično brisanje zapisu, pronalaženje zapisu (npr. na osnovu ključa ili relativne adrese) itd.

Jezik za manipulisanje podacima može da bude samostalan jezik (Self-contained System), ili se može ugraditi u neki programski jezik višeg nivoa (Host Language System).

### 5.2. Principi rada sistema za upravljanje bazom podataka

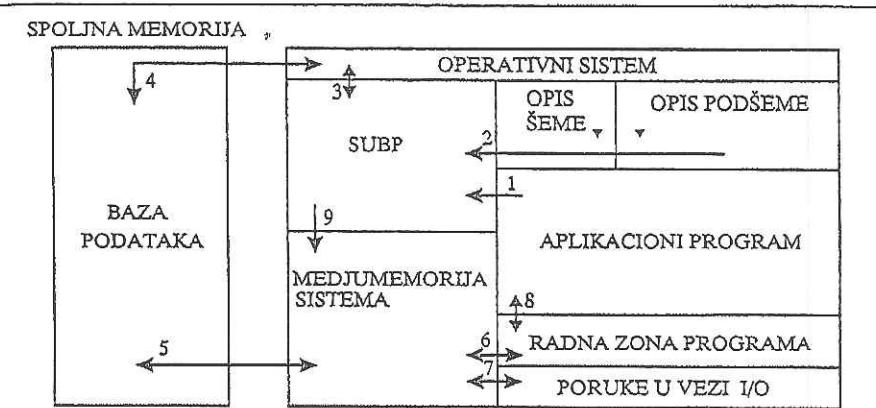
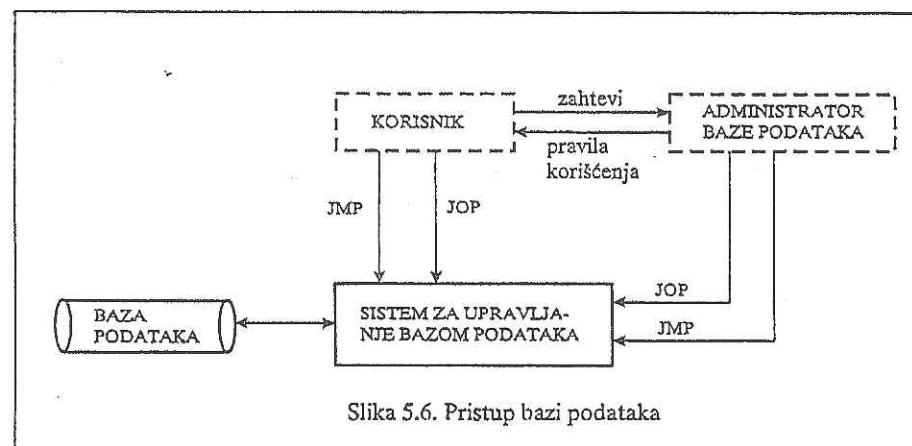
Sve ono što čini bazu podataka može se podeliti u sledeće dve grupe:

1. podaci,
2. softver.

Podaci su upisani na nosioce podataka na potpuno isti način kao i u klasičnoj organizaciji podataka. Softver za upravljanje bazom podataka je veoma složen, sadrži veći broj programa i rutina preko kojih obezbeđuje sledeće funkcionalne karakteristike koje baza podataka mora da ima:

- jezik za manipulaciju podacima,
- jezik za opis šeme,
- jezik za opis podšeme,
- jezik za opis fizičke strukture podataka,
- način definisanja i kontrole tajnosti podataka,
- način obezbeđenja sigurnosti podataka,
- pomoćne programe za:
  - praćenje osobina baze podataka,
  - reorganizaciju podataka i dr.
- upravljanje medjumemorijom potrebnom za blokove baze podataka,
- upravljanje greškama u eksploataciji i dr.

Do baze podataka korisnik može doći samo putem SUBP pri čemu se realizuje koncept zaštite podataka, kako je to ilustrovano na slici 5.6. Pravila za korišćenje baze podataka određuju administrator baze podataka i ugradjuje ih u SUBP. Većina SUBP je strogo nadgradnjena na operativni sistem računara. Fizičko pomeranje podataka između operativne i spoljne memorije na taj način vrše rutine operativnog sistema i stvarne metode pristupa podacima određuju mogućnosti operativnog sistema.



Slika 5.7. Pojednostavljeni model funkcionisanja sistema za upravljanje bazom podataka

Na slici 5.7 predstavljen je pojednostavljen model funkcionisanja sistema za upravljanje bazom podataka. Ako aplikacioni program traži ili želi da smesti jedan podatak u bazu podataka, tada se to ostvaruje naredbom u jeziku koji vrši manipulaciju podataka (strelica 1). Rutine za upravljanje podacima (RUP) kao deo SUBP prihvataju i analiziraju tu naredbu i dopunjaju je potrebnim aktuelnim vrednostima na osnovu opisa cele baze podataka (šeme) i podšeme koja se odnosi na taj program (strelica 2). Zatim se operativnom sistemu daju I/O uputstva (strelica 3), koji stupa u kontakt sa spoljnom memorijom (strelica 4) i prenosi podatke u medjumemoriju sistema (strelica 5). Odavde će SUBP preneti podatke u radne zone programa (strelica 6). Ako su podaci u drugačijem obliku od oblika koji se koristi u aplikacionom programu, tada će se izvršiti neophodna konverzija (npr. konvertovanje iz pakovanog u nepakovan oblik podataka). Može se desiti da će posredstvom I/O uputstva jednog korisnika SUBP tražene podatke moći staviti na raspolaganje samo posle niza fizičkih I/O uputstava, u slučaju da se polja sakupljaju iz različitih fizičkih zapisa. Na koji način će se to u jednom konkretnom slučaju obaviti, odrediće SUBP na osnovu opisa šeme i podšema. Rezultat I/O operacije SUBP signalizira aplikacionom programu (strelica 7) i u slučaju povoljnog signala program obraduje podatke, odnosno nastavlja se sa izvršavanjem (strelica 8). U interesu obezbeđenja dalje obrade SUBP vodi računa o sadržaju I/O bafera (strelica 9).

U cilju efikasnog korišćenja baze podataka potrebno je da, pored aplikativnog programa i rezidentnog dela operativnog sistema, u operativnoj memoriji budu i opis šeme i podšema baze podataka i značajan deo SUBP.

## ***6. MODELI PODATAKA***

Model podataka je skup međusobno povezanih podataka koji opisuju objekte, njihove veze i osobine, realnog sistema. Opis realnog sistema čiji se informacioni sistem projektuje pomoću termina modela podataka naziva se model baze podataka.

Osnovni zadatak istraživanja u području modeliranja podataka je naći koncepte pomoću kojih će se izgraditi model koji reprezentira naše znanje o sistemu. Koncepti su intelektualno sredstvo, a dobiveni model podataka je globalni apstraktni tip podataka koji reprezentuje ceo sistem.

U modelu podataka ne opisuje se potpuni skup znanja o sistemu, već se vrši odabir i opis relevantnih karakteristika sistema. Relevantne karakteristike sistema su one za koje pri analizi podataka utvrdimo da su važne sa aspekta upotrebe modela u dатој situaciji.

Model podataka sastoji se od sledeće tri komponente:

1. struktura,
2. integritetna,
3. operacijska.

Strukturalna komponenta modela sadrži skup primitivnih koncepata i skup pravila za izgradnju složenijih koncepata. Pojam koncepta se odnosi na apstraktну predstavu jedne klase delova realnog sveta. Taj deo realnog sveta može biti neki konkretni subjekat, objekat, dogadjaj, konkretna osobina objekta ili veza između takva dva dela. Primitivni koncept je takav koncept koji se ne može dalje dekomponovati na koncepte. U modelima podataka primitivni koncept mogu na primer predstavljati obeležje i domen obeležja.

Kako bi se sačinio model podataka realnog sistema i znanje unelo u model, koristi se koncepcija apstrakcije opisa podataka. Apstrakcija je misaoni postupak u čijem krajnjem rezultatu je predstavljanje samo opštih, zajedničkih i bitnih osobina pojedinih koncepata iz realnog sistema uz zanemarivanje

nebitnog. Može se izvoditi uz kontrolisano uključivanje detalja u model podataka sistema. Pri izradi modela podataka koriste se sledeće tri apstrakcije:

1. Klasifikacija i uzorkovanje,
2. Generalizacija i specijalizacija,
3. Agregacija i dekompozicija.

1. **Klasifikacija ili tipizacija** je apstrakcija u kojoj se skup sličnih objekata predstavlja jednom klasom objekata. Kriterijum sličnosti igra značajnu ulogu u definisanju klase objekata. Slični objekti su oni objekti koji imaju ista obeležja (svojstva), koji mogu da stupe u iste veze sa drugim objektima u sistemu i na koje se mogu primeniti iste operacije. U zavisnosti od usvojenog kriterijuma sličnosti dva objekta se mogu nalaziti u istoj ili u dve različite klase. Takođe isti objekat može pripadati različitim klasama.

Na primer, skup svih nastavnika, skup svih studenata, skup svih predmeta na jednom fakultetu čine realne klase objekata NASTAVNIK, STUDENT, PREDMET respektivno. Kada se za realnu klasu objekata utvrde obeležja bitna za realizaciju informacionog sistema dobija se model realne klase objekata koji se naziva **tipom objekta**.

Tip objekta formalno možemo predstaviti sa  $O(A_1, \dots, A_n)$

gde je:  $O$  - naziv klase objekata,  
 $A_i$  - naziv obeležja za klasu objekata, ( $1 \leq i \leq n$ ).

Na primer, tip objekta STUDENT(BROJ\_IND, PRE ime, FAKULTET) reprezentuje sve studente jednog univerziteta.

Postupak detaljizacije za ovu apstrakciju se naziva **uzorkovanjem**, i predstavlja prikazivanje jednog pojavljivanja (uzorka, primerka) datog tipa ili klase. I samo obeležje nekog objekta se može tretirati u najopštijem smislu, kao objekat. Na taj način se može definisati **tip obeležja** (na primer FAKULTET) i **pojavljivanja tipa obeležja** (na primer: Ekonomski, Pravni, Elektrotehnički). Skup svih pojava jednog tipa obeležja se naziva **domen obeležja**.

Kada svako obeležje tipa objekta uzme vrednost iz svog domena nastaje **pojava tipa objekta** koja predstavlja model jedne pojave iz realne klase objekata.

U modelima podataka često se ne pravi jasna razlika izmedju tipa i klase objekta. Ako posmatramo neki skup sličnih objekata tip objekta predstavlja **intenziju** tog skupa. Pod intenzijom podrazumevamo definisanje nekog skupa navodjenjem uslova koje njegovi elementi treba da zadovolje. Klasa objekata predstavlja istovremeno i **intenziju** i **ekstenziju** skupa objekata. Pod ekstenzijom se podrazumeva definisanje skupa navodjenjem svih njegovih članova. Skup pojava tipa objekta predstavlja ekstenziju.

2. **Generalizacija** je apstrakcija gde se skup sličnih tipova objekata tretira kao novi generički (izvedeni) tip objekta na višem nivou. Pod sličnim tipovima objekata mogu se tretirati tipovi objekata koji imaju jedan broj istih (zajedničkih) osobina, tipova veza sa drugim objektima i operacija.

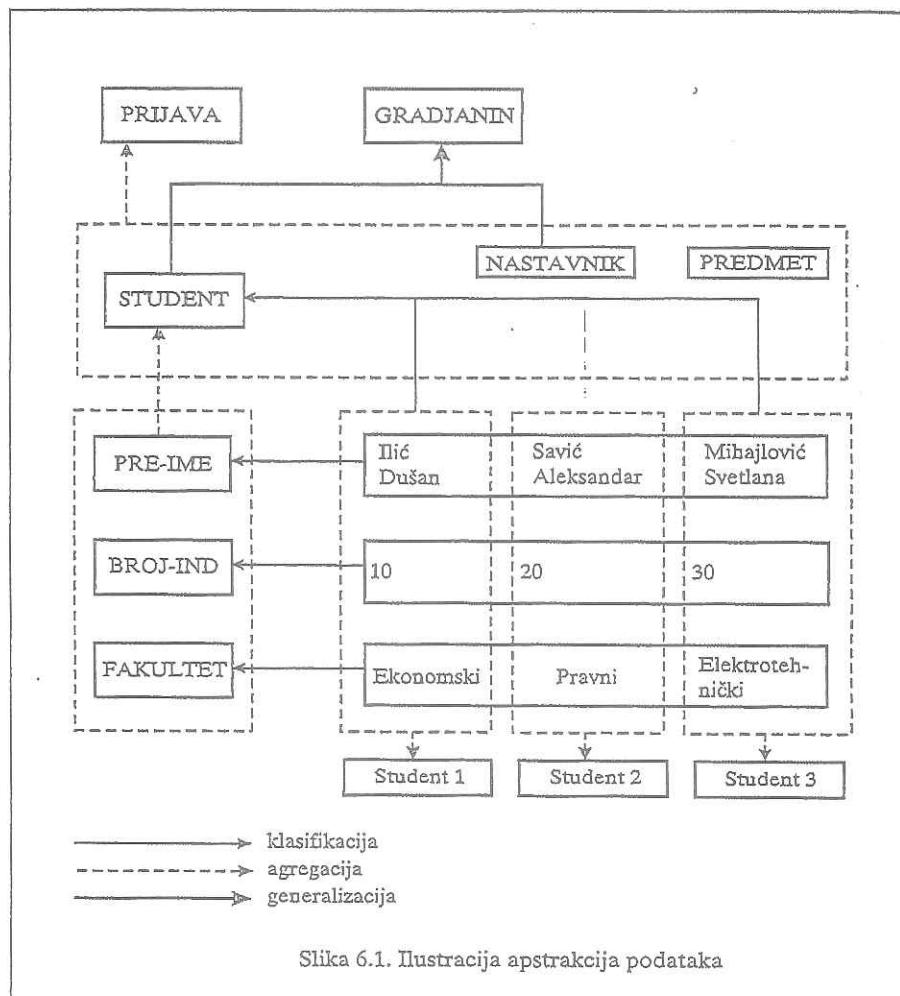
Na primer, tipovi objekata STUDENT, RADNIK, PENZIONER, mogu se generalizovati u tip objekta GRADJANIN.

Inverzan postupak generalizacije je **specijalizacija**.

3. **Agregacija** je apstrakcija gde se skup tipova objekata i njihovih veza (kao i obeležja objekata i veza) predstavlja novim agregiranim (izvedenim) tipom objekta na višem nivou.

Postupak inverzan agregaciji se naziva **dekompozicija**. Sam tip objekta može se posmatrati kao najniži nivo ove apstrakcije, kao agregacija njegovih tipova obeležja.

Na slici 6.1. na jednom primeru su ilustrovane sve uvedene apstrakcije podataka. Skupovi vrednosti {Ilić Dušan, Simić Aleksandar, Mihajlović Svetlana}, {60, 20, 30}, i {Ekonomski, Pravni, Elektrotehnički} klasifikuju se u tipove obeležja PRE ime, BROJ-IND i FAKULTET respektivno. Pojavljivanja tipova obeležja (vrednosti) <Ilić Dušan, 10, Ekonomski>, <Simić Aleksandar, 20, Pravni> i <Mihajlović Svetlana, 30, Elektrotehnički> agregiraju se u pojavljivanja objekata Student-1, Student-2, Student-3, respektivno, a ona se klasifikuju u tip objekta STUDENT, odnosno pripadaju klasi STUDENT. Tip objekta STUDENT istovremeno predstavlja i agregaciju tipova obeležja <PRE ime, BROJ-IND, FAKULTET>. Tipovi objekata STUDENT i NASTAVNIK su podtipovi tipa objekta GRADJANIN (generalizuju se u tip objekta GRADJANIN). Tipovi objekata <STUDENT, NASTAVNIK, PREDMET> se agregiraju u tip objekta PRIJAVA, odnosno pripadaju klasi objekata PRIJAVA.



Agregirani i generalizirani tipovi objekata se dalje mogu agregirati i generalizirati u nove tipove objekata. Povezivanjem tipova objekata nastaje struktura modela u koju je ugradjeno znanje o realnom sistemu. Potrebnu semantiku realnog sistema nije moguće u potpunosti ugraditi u model podataka. Zbog toga se deo semantike realnog sistema ugrađuje u model procesa čime se otežava realizacija IS posebno u fazi programiranja. Modeli podataka treba da podrže sve apstrakcije podataka na svim nivoima

apstrakcije i na taj način smanje količinu znanja o realnom sistemu koju treba ugraditi u model procesa.

*Integritetnu komponentu* modela podataka čini skup uslova integriteta ili kako se još naziva skup ograničenja. Uslovi integriteta ograničavaju sve pojave baze podataka nad jednom šemom na skup dozvoljenih (neprotivrečnih) pojava baze podataka isključujući one pojave koje ne zadovoljavaju neko od ograničenja.

Uslovi integriteta iskazuju se preko:

- dozvoljenih vrednosti podataka pojedinog obeležja - ograničenje na pridruživanje,
- dozvoljenih podataka u okviru jednog obeležja - ograničenje na vrednost,
- dozvoljenih veza medju tipovima objekata - ograničenje na veze,
- dozvoljenih podataka pojedinog obeležja - ograničenje na domenu.

*Ograničenja na pridruživanje* obeležja objektu ili vezi znači da dati objekat može imati jednu, nijednu ili više vrednosti tog obeležja.

*Ograničenje na vrednost* obeležja objekta ili veze, znači da vrednost obeležja mora biti prema datim pravilima (u datim granicama).

*Ograničenja na vezu* izmedju tipova objekata znači da jedna pojava jednog tipa objekta može biti povezan sa jednom, nijednom ili više pojava drugog tipa objekta, i obrnuto.

*Ograničenja na domenu* (tip vrednosti) znači da sva obeležja koja su definisana nad tom domenom mogu imati samo vrednosti iz te domene.

Skup pravila strukturne komponente služi za definisanje dozvoljenih objekata i njihovih veza u okviru modela podataka. Nedozvoljeni objekti ili veze isključuju se iz strukture podataka putem opštih i posebnih ograničenja.

*Operacijsku komponentu* modela podataka čini skup koncepta koji omogućuju interpretaciju dinamičkih karakteristika skupa podataka. Koncept strukture i ograničenja reprezentuju statičke osobine realnog sistema, a operatori omogućuju da se stanje podataka u bazi podataka menja u skladu sa promenom stanja u realnom sistemu.

S obzirom na količinu znanja o realnom svetu koja se može ugraditi u model podataka, modeli podataka se dele u generacije. Teorija modela podataka postojiće modele podataka klasificuju u sledeće tri generacije:

#### *Modeli podataka I generacije*

Svaki konvencionalni programski jezik je zaseban model podataka. Podaci se modeliraju preko koncepata kojima dati jezik raspolaže. Ti koncepti su na primer: integer, real, matrica, pointer, stek i dr. Apstrakcija klasifikacije koristi se pri definisanju prostih tipova (INTEGER, REAL, CHARACTER i sl.) i datoteka. Agregacije koje se koriste su zapisi (kao agregacija polja i drugih zapisa ili grupa i vektora) i vektori (array) kao agregacije istovrsnih elemenata. U nekim jezicima može se delimično primeniti apstrakcija generalizacije. Ograničenja nad vrednostima pojedinih tipova nije moguće eksplisitno zadati. Modeli podataka I generacije i na njima zasnovani programski jezici nisu dovoljno pogodni za modeliranje realnog sistema pa im je praktična primena ograničena.

#### *Modeli podataka II generacije*

Za prezentaciju podataka sadrže koncepte kao što su: stablo, set, relacija i dr. U osnovi koriste iste apstrakcije kao i modeli I generacije bez potpune podrške konceptu agregacije i generalizacije. Moguće je eksplisitno definisati specifične vrste ograničenja na vrednosti podataka. Može se definisati baza podataka kao skup međusobno povezanih podataka. I pored toga što su semantički bogatiji od modela I generacije ne mogu dati semantički zadovoljavajući opis složenih sistema u kojima se definišu složeni objekti koji se ne mogu lako predstaviti konceptom zapisa. Ovoj generaciji pripadaju sledeći modeli podataka:

- funkcionalni model podataka,
- hijerarhijski model podataka,
- mrežni model podataka,
- klasični relacioni model podataka,
- Warnier-ovi dijagrami.

Postoji niz komercijalnih SUBP zasnovanih na ovim modelima.

#### *Modeli podataka III generacije*

Sadrže koncepte generalizacije i agregacije. Pored atomske semantike omogućuju ugradnju molekularne semantike u model podataka. Podržavaju sve vrste apstrakcija, poseduju mogućnosti eksplisitnog definisanja ograničenja na vrednosti podataka i moćne operacije nad objektima visokog nivoa apstrakcije. Dobro modeliraju realni sistem. Korisniku su razumljivi. Ovoj generaciji pripadaju sledeći modeli podataka:

- model objekti-veze (MOV) (Chen 1976),
- binarni model podataka,
- prošireni relacioni model podataka (RM/T) (Codd 1979),
- SDM-IBM (IBM),
- model podataka semantičkih mreža (Roussopoulos i Mylopoulos),
- semantički model podataka (SDM) (Hammer i McLeod 1978),
- Petri-jeve mreže (Reti di Petri),
- model semantičkih asocijacija (SAM) (Stanley),
- D-graph model (Weber),
- Palmer (Palmer),
- diam II (Senko).

Modeli podataka III generacije su danas uglavnom bez razvijenih komercijalnih SUBP.

## 7. MODEL OBJEKTI - VEZE - OBELEŽJA

Model objekti - veze (MOV) potiče od naziva modela objekti - veze - obeležja (Entity-Relationship-Atribute, E-R-A, ER-model) (ostaje nejasno zašto je iz naziva izostavljeno *obeležje*). MOV je prvi put objavljen u Chen-ovom članku 1976 godine i jedan je od danas najčešće korišćenih modela podataka.

MOV je nastao kao sinteza dobrih osobina tri druga modela: mrežnog, relacionog i skupa entiteta. Postoje više verzija ovog modela podataka. U odnosu na originalni Chen-ov model uvedeno je više proširenja tako da ga danas susrećemo pod nazivom *prošireni model objekti - veze* (PMOV). Kod nas poznata verzija PMOV definisana je na Fakultetu organizacionih nauka (FON) u Beogradu. Razmatranja modela objekti - veze ovde će se oslanjati na PMOV definisan na FON. Ovo opredeljenje proizilazi pre svega zbog mogućnosti da se u modeliranju koristi CASE alat ARTIST koji je takodje razvijen na FON.

Model objekti-veze je najpopularniji i u praksi najčešće korišćen semantički model podataka koji se koristi kao grafički jezik za projektovanje konceptualne šeme baze podataka. *Konceptualna* šema predstavlja takav model realnog sistema i njegove baze podataka koji ne zavisi od konkretnog sistema za upravljanje bazom podataka. Konceptualna šema baze podataka prema MOV može se lako prevesti u šemu baze podataka na kojoj je SUBP zasnovan.

Konceptualna šema realizovana kao MOV po pravilu se predstavlja uz pomoć dijagrama koje nazivamo dijagrami objekti veze (DOV). Simboli za crtanje DOV prikazani su na slici 7.1.

Tip objekta se predstavlja pravougaonim simbolom sa upisanim nazivom.

Tip veze predstavlja se rombom sa upisanim nazivom ili rombom sa oznakom S.

Od romba koji predstavlja tip veze povlače se linije do svih onih simbola tipova objekata koje ta veza povezuje. Obeležja tipa objekta se prikazuju kao

ovali sa upisanim nazivom i povezuju linijama sa odgovarajućim tipom objekta. Da bi se istakla obeležja kojima se identificuju pojave tipa objekta ista su podvučena.

GRAFIČKI PRIKAZ (simbol)	OPIS
rectangle	osnovni objekt
rectangle with rounded corners	slab objekt
diamond	veza
diamond with letter S	veza nadtipa i podtipa
rectangle with diamond inside	mešoviti tip objekta
oval	obeležje
line	linija za povezivanje

Slika 7.1. Grafički simboli za crtanje DOV

Zbog bolje preglednosti konceptualne šeme kao dijagrama tipova objekata i veza, dijagrami se mogu crtati na različitim nivoima detaljnosti. Najniži nivo detaljnosti je kada DOV sadrži tipove objekata i tipove veza. Viši nivo detaljnosti nastaje ako se u DOV pored tipova objekata i tipova veza označe samo identifikaciona obeležja i informacije o minimalnom i maksimalnom broju pojava tipova objekata koje učestvuju i vezama. Na najvišem nivou detaljnosti DOV sadrži i sva obeležja tipova objekata i akcije u slučaju narušavanja pravila integriteta.

Do danas nije razvijen neki šire primenjiv SUBP zasnovan na MOV mada model poseduje sve neophodne komponente (strukturalna, integritetna, operacijska).

### 7.1. Strukturalna komponenta

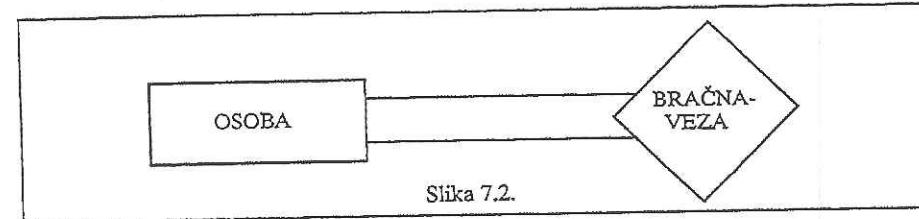
Primitivni koncepti u MOV su: objekt, veza (medju objektima) i obeležje objekta.

Koristeći apstrakciju klasifikacije, pojedinačni objekti se klasifikuju u tipove objekata. Objekti koji se identificuju nezavisno od ostalih objekata u sistemu nazivaju se *osnovni objekti* (kernel, jaki objekti).

Tip osnovnog objekta se predstavlja pravougaonikom sa upisanim nazivom.

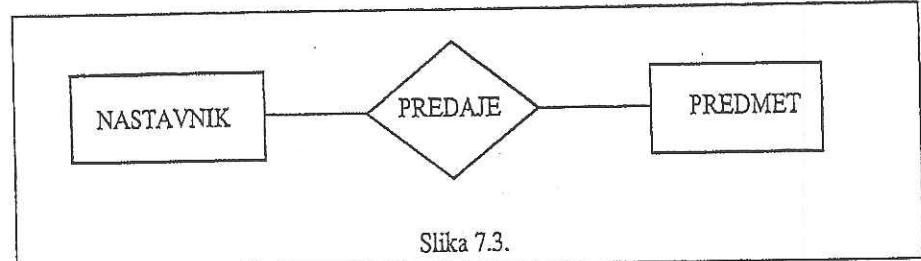
Veze u modelu opisuju način povezivanja objekata. Broj tipova objekata koji učestvuju u vezi definiše *red veze*. Mogu postojati *rekurzivne*, *binarne* i veze *višeg reda*.

*Rekurzivna veza* je veza izmedju dva objekta istog tipa (binarna veza nad jednom klasom objekata). Na slici 7.2. dat je ilustrativni primer rekurzivne veze koju možemo interpretirati na sledeći način: Osoba se nalazi u bračnoj vezi sa drugom osobom.



Slika 7.2.

*Binarnu vezu* čine povezani objekti različitog tipa. Na slici 7.3. dat je primer binarne veze koju možemo interpretirati na sledeći način: Nastavnik predaje predmet, a predmet je predavan od nastavnika.



Slika 7.3.

U MOV se preporučuje korišćenje rekurzivnih i binarnih veza. Veze višeg reda mogu se predstaviti pomoću binarnih veza ili se takva veza tretira kao agregacija.

Koristeći apstrakciju klasifikacije definišu se pojmovi tipa veze i pojavljivanje tipa veze.

Informacija o prirodi odnosa između objekata iz realnog sveta u modelu podataka iskazuje se *kardinalitetom tipa veze*.

Veza V između pojava tipova objekata  $O_1$  i  $O_2$  definiše dva tipa preslikavanja:

$V_1: O_1 \longrightarrow P(O_2)$  preslikavanje sa skupa pojavljivanja  $O_1$  u skup pojavljivanja  $O_2$  ( $O_1$  je domen, a  $O_2$  kodomen preslikavanja),

$V_2: O_2 \longrightarrow P(O_1)$  (inverzno) preslikavanje sa skupa pojavljivanja  $O_2$  u skup pojavljivanja  $O_1$  ( $O_2$  je domen, a  $O_1$  kodomen preslikavanja),

gde su:  $V_1$  i  $V_2$  - nazivi preslikavanja,

$P(O)$  - partitivni skup skupa  $O$ .

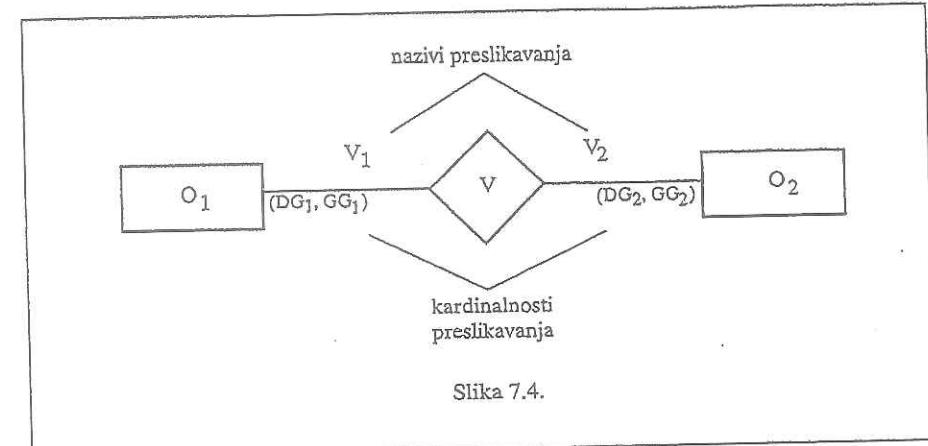
Za svako od ovih preslikavanja definiše se par (DG, GG), gde DG (donja granica) predstavlja najmanji, a GG (gornja granica) najveći broj elemenata partitivnog skupa u koji se preslikava jedan elemenat skupa originala. Par (DG, GG) predstavlja kardinalnost preslikavanja. DG može imati vrednost 0, 1 ili neki poznat ceo broj  $> 1$ . Za  $DG = 0$  kažemo da je veza *parcijalna* (opciona), dok za  $DG > 0$  kažemo da je *veza totalna* (obavezna). GG može imati vrednost 1, neki poznat ili nepoznat ceo broj  $> 1$  koji se označava sa M. Očigledno je da u jednom preslikovanju mora biti zadovoljeno  $DG \leq GG$ .

Označavanje kardinalnosti preslikavanja  $V_1: O_1 \longrightarrow P(O_2)$  na DOV je na liniji koja spaja tip objekta  $O_1$  sa tipom veze, a preslikavanja  $V_2: O_2 \longrightarrow P(O_1)$  je na liniji koja spaja tip objekta  $O_2$  sa tipom veze kako je to ilustrovano na slici 7.4.

U praksi i literaturi često se pri razmatranju kardinalnosti preslikavanja i njihovom označavanju navode samo gornje granice, dok se donja granica  $DG = 0$  zamenjuje rečima *veza je parcijalna*, a  $DG = 1$  *veza je totalna*. Tako se govori o sledećim vezama:

- 1:1 koja obuhvata:  $(0,1) : (0,1)$  (parcijalna sa obe strane),  
 $(0,1) : (1,1)$  (parcijalna sa jedne i totalna sa druge strane),  
 $(1,1) : (1,1)$  (totalna sa obe strane).
- 1:M koja obuhvata:  $(0,1) : (0,M)$  (parcijalna sa obe strane),  
 $(0,1) : (1,M)$  (parcijalna sa strane 1 i totalna sa strane M),  
 $(1,1) : (0,M)$  (parcijalna sa strane M i totalna sa strane 1),  
 $(1,1) : (1,M)$  (totalna sa obe strane).
- M:M koja obuhvata:  $(0,M) : (0,M)$  (parcijalna sa obe strane),  
 $(0,M) : (1,M)$  (parcijalna sa jedne i totalna sa druge strane),  
 $(1,M) : (1,M)$  (totalna sa obe strane).

Uočavamo da dva preslikavanja definišu jednu binarnu vezu, pa je koncept veze izvedeni pojam, što ne znači da je potpuno redundantan i da bi ga trebalo izostaviti. Nazivom veze se uspostavlja odnos između dva međusobno inverzna preslikavanja. Veza se može posmatrati i kao agregirani objekat jer predstavlja agregaciju objekata koji su u vezi.

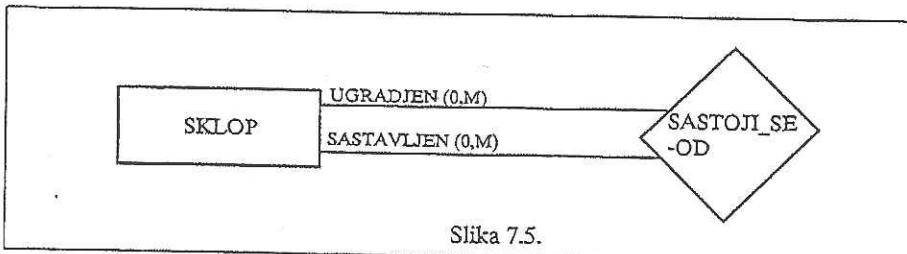


Slika 7.4.

Nazive preslikavanja bira projektant i oni moraju biti jedinstveni u modelu. Ako se ne možemo odlučiti za naziv preslikavanja isti se može formirati od naziva veze i objekta. Kažemo da je tada za naziv preslikavanja uzeta

podrazumevana (default) vrednost pri čemu se ista i ne mora naznačiti na dijagramu.

Kao što je napred bilo rečeno veze se mogu uspostaviti i izmedju pojavljivanja objekata istog tipa (nad istom klasom objekata). Na slici 7.5. ilustrovana je veza **SASTOJI\_SE\_OD** sa parom preslikavanja <**SASTAVLJEN**, **UGRADJEN**> uspostavljena nad klasom objekta **SKLOP**. Preslikavanje **SASTAVLJEN** je preslikavanje koje za jedan sklop daje skup njemu podredjenih sklopova, odnosno njegovih sastavnih delova, a preslikavanje **UGRADJEN** je preslikavanje koje za jedan sklop, daje skup njemu nadredjenih sklopova tj. skup sklopova u koje je on ugradjen. U vezi **SASTOJI\_SE\_OD** tip objekta **SKLOP** ima dvostruku ulogu, jednom se tretira kao nadredjen (sastavljen), a drugi put kao podredjen (ugradjen) sklop u odgovarajućoj strukturi (sastavnici).



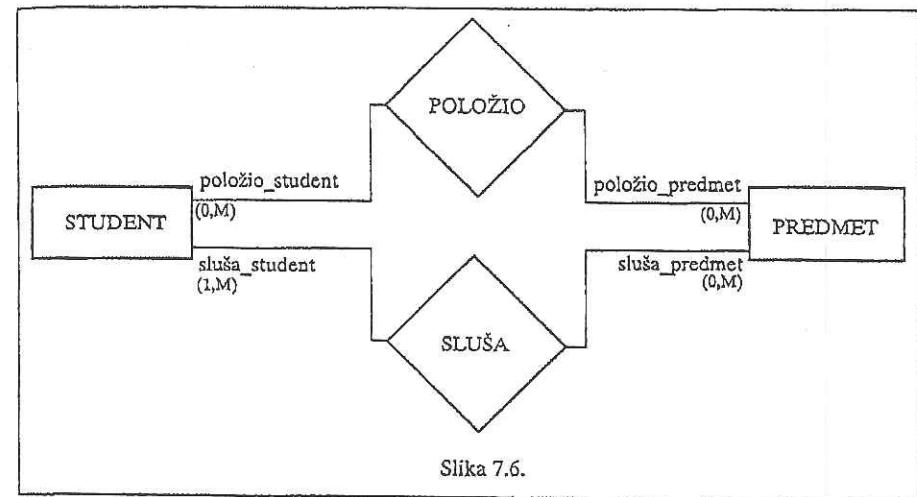
Slika 7.5.

Izmedju dva tipa objekata moguće je uspostaviti više različitih tipova veza kako je to ilustrovano na slici 7.6. Nazivi preslikavanja na slici 7.6. dati su po pravilu formiranja podrazumevanog naziva.

Preslikavanje **položio\_student** predstavlja vezu izmedju tipova objekata **STUDENT** i **PREDMET** sa kardinalnošću (0,M) što znači da jedan student može imati najmanje 0 i najviše M položenih predmeta.

Pored navedenih primitivnih koncepata u MOV postoje i sledeći specifični koncepti:

- tip slabog objekta,
- podtip,
- agregacija.



Slika 7.6.

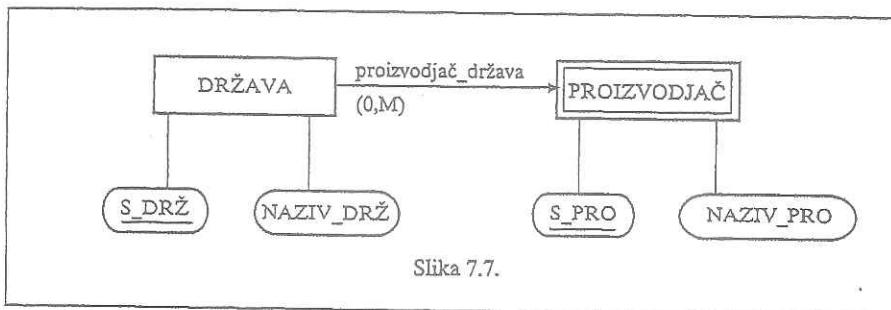
#### Tip slabog objekta

**Tip slabog objekta** je takav tip objekta koji nema vlastitih obeležja za identifikaciju, već koristi i obeležja nekog drugog nadredjenog tipa objekta koji može biti osnovni ili slabi objekat. Kažemo još i da tip slab objekat *identifikaciono zavisi* od nekog drugog tipa objekta.

Označavanje tipa slabog objekta se vrši upisivanjem njegovog naziva u dvostruki pravougaonik.

Na slici 7.7. ilustrovan je način predstavljanja tipa slabog objekta **PROIZVODJAČ** koji identifikaciono zavisi od osnovnog tipa objekta **DRŽAVA**. Jedna država ima više proizvodjača, a jedan proizvodjač nalazi se u jednoj državi. Tip objekta **PROIZVODJAČ** ima vlastito obeležje **S\_PRO** (šifra proizvodjača) koje ga jedinstveno identificuje u jednoj državi, ali nema vlastitih obeležja koja ga jedinstveno identificuju u više država. Jednoznačni identifikator tipa objekta **PROIZVODJAČ** predstavlja kombinacija obeležja **S\_DRŽ** (identifikator objekta **DRŽAVA**) i **S\_PRO**.

Veza izmedju nadredjenog i tipa slabog objekta predstavljena je usmerenom linijom koja polazi od grafičkog simbola nadredjenog tipa objekta i završava na simbolu tipa slabog objekta.



U vezi nadredjenog i tipa slabog objekta postoji samo preslikavanje od nadredjenog prema slabom objektu čiji naziv i kardinalnost treba označiti na dijagramu. Podrazumevani naziv ovog preslikavanja je naziv tipa slabog objekta i naziv tipa objekta od kojeg on zavisi.

Pojava tipa slabog objekta ne može postojati u bazi podataka bez postojanja pojave tipa njemu nadredjenog objekta koji ga identificuje što znači da identifikaciona zavisnost podrazumeva da je donja granica kardinalnosti preslikavanja od slabog objekta prema nadredjenom objektu 1. Jedna pojava tipa slabog objekta u vezi je sa samo jednom pojmom tipa njemu nadredjenog objekta što znači da identifikaciona zavisnost podrazumeva da je gornja granica kardinalnosti preslikavanja od slabog objekta prema nadredjenom objektu 1. Ako se u bazi podataka briše pojava tipa nekog objekta tada se u bazi podataka brišu i sva od njega identifikaciono zavisna pojavljivanja slabog objekta.

Da bi povećali preglednost i čitljivost DOV ne moramo uvek ispisivati nazive preslikavanja. U tim slučajevima smatramo da izostavljeni nazivi preslikavanja imaju podrazumevanu vrednost.

Tip slabog objekta može imati samo jedan tip objekta od kojeg je slab i može imati proizvoljno mnogo drugih tipova objekata koji su od njega slabi (zavisni) ili su sa njim u vezi.

### Podtip

Kao što je napred već rečeno generalizacija je apstrakcija u kojoj se skup sličnih tipova objekata tretira kao generički tip objekta (*nadtip*). Specijalizacija je inverzni postupak u kome se za neki tip objekta, definišu

njegovi podtipovi koji imaju neka njima specifična obeležja, veze i/ili operacije. Nadredjenom tipu objekta pripisuju se sva ona obeležja koja ima svaki od pripadnih podtipova objekata. Pojedinim podtipovima objekata pripisuju se ona obeležja koja su karakteristična za taj podtip objekta. Uvodjenje podtipa u MOV omogućava optimalan način pridruživanja obelžja objektima što će se jasnije videti iz primera koji slede.

Tip objekta  $O_2$  je podtip drugog tipa objekta  $O_1$ , ako je svaka pojava tipa objekta  $O_2$  ujedno i pojava tipa objekta  $O_1$ .

Generalizacija i specijalizacija predstavljaju specijalnu vezu koja se sastoji od dva preslikavanja:

1. generalizacije, odnosno preslikavanja  $PODTIP \longrightarrow NADTIP$  (trivijalno preslikavanje između podskupa i skupa za koje je uvek  $DG=1$  i  $GG=1$ ),
2. specijalizacije koja predstavlja preslikavanje  $NADTIP \longrightarrow PODTIP$ .

S obzirom na donju i gornju granicu kardinalnosti specijalizacije, definišu se sledeće vrste specijalizacije:

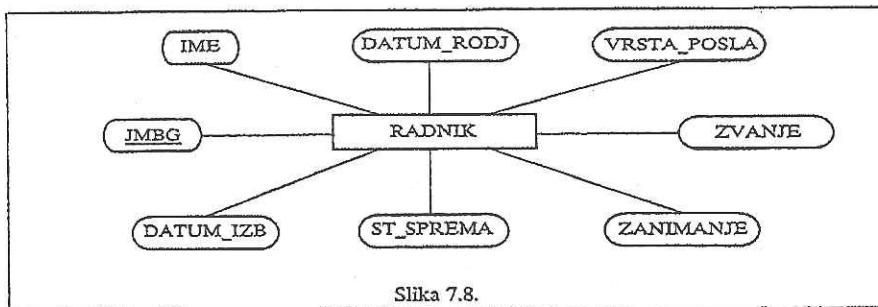
*Ekskluzivna specijalizacija* - svako pojavljivanje tipa specijalizuje se u jedan podtip i tada je  $DG = 1$  i  $GG = 1$ .

*Neekskluzivna specijalizacija* - jedno pojavljivanje tipa može se specijalizovati u više podtipova i tada je  $DG = 0$  i  $GG > 1$ .

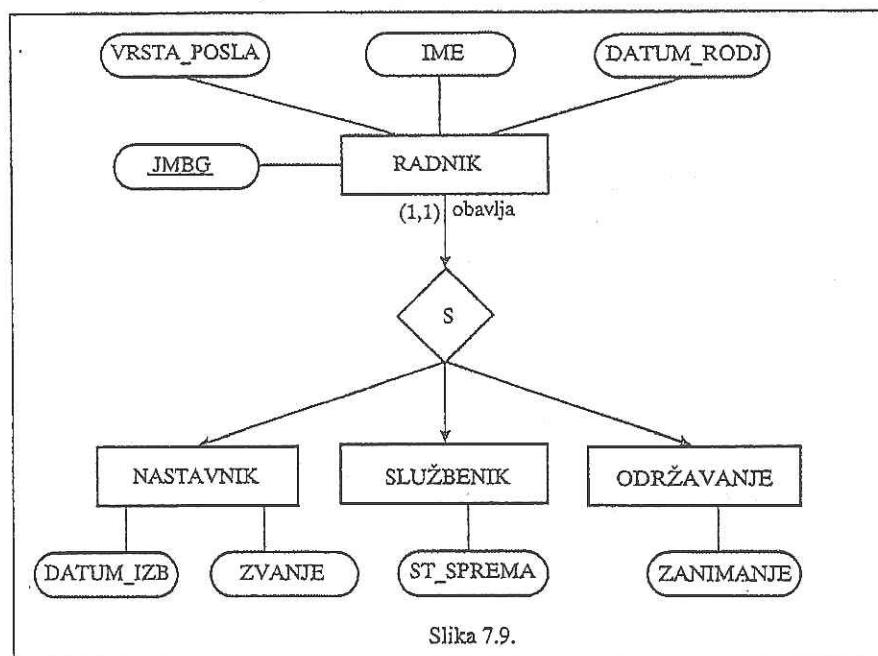
Neka smo na primer, u procesu izrade MOV definisali tip objekta RADNIK dat na slici 7.8. Uočavamo da su neka obeležja specifična samo za neke od radnika (tj. za neke od pojava tipa objekta RADNIK). Na primer, obeležje ZVANJE i DATUMIZB (datum izbora u zvanje) pripadaju samo onim radnicima koji za obeležje VRSTA\_POSLA imaju vrednost *nastavnik* dok za radnike koji nemaju tu vrednost obeležje VRSTA\_POSLA, ZVANJE i DATUMIZB predstavljaju neprimenjivo svojstvo. Radi primerenijeg opisivanja stanja u sistemu, tipu objekta RADNIK pridružujemo nekoliko podtipova na osnovu vrednosti obeležja VRSTA\_POSLA (tj. *nastavnik*, *službenik*, *održavanje*). Podtipovima sada pridružujemo samo njihova specifična obeležja, dok generički tip RADNIK zadržava obeležja koja su zajednička svim pojavama tog tipa objekta. Prikaz sa slike 7.8. zamenjujemo DOV na slici 7.9. Uočavamo novi grafički simbol za predstavljanje odnosa nadtipa i podtipa u koji umesto naziva veze stoji oznaka S. Linija koja spaja

nadtip i simbol veze ima strelicu prema vezi, na njoj se upisuje naziv preslikavanja od nadtipa prema podtipovima (naziv specijalizacije) i kardinalnost tog preslikavanja. Od simbola veze prema podtipovima postoje usmerene linije sa strelicom na podtipu. Na ovim linijama nema oznake kardinalnosti i naziva preslikavanja.

Kao što je bilo rečeno preslikavanje podtip-nadtip predstavlja trivijalno preslikavanje sa kardinalnošću (1, 1) i ne označava se na DOV.



Slika 7.8.

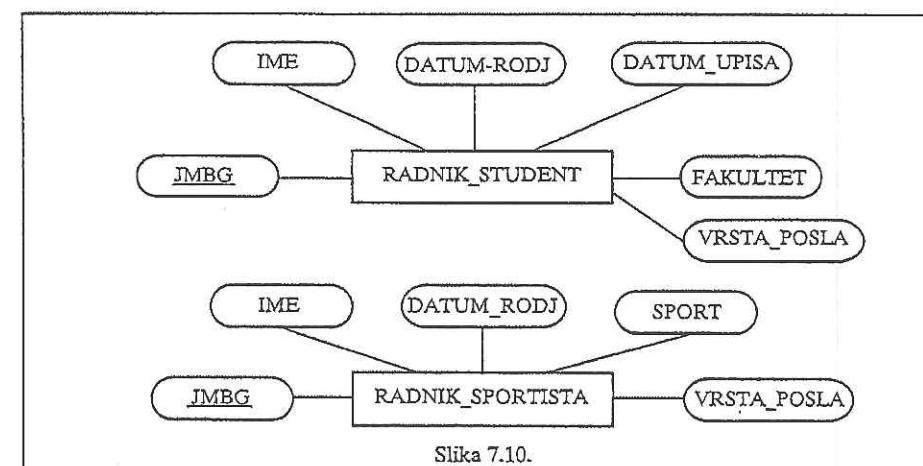


Slika 7.9.

U primeru na DOV sa slike 7.9. prepostavljamo da je podela radnika po vrsti posla *disjunktna* (ekskluzivna specijalizacija) što znači da jedan radnik obavlja u jednom trenutku vremena samo jednu vrstu posla (ili nastavnik ili službenik ili održavanje) te je zbog toga gornja granica kardinalnosti preslikavanja *obavlja* 1.

Pretpostavimo da su u toku daljeg rada na modelu podataka definisani tipovi objekata RADNIK\_STUDENT i RADNIK\_SPORTISTA kako je to ilustrovano na slici 7.10. Sa slike 7.10. možemo uočiti da su neka obeležja zajednička i za radnika studenta i za radnika sportista, kao i da su neka karakteristična samo za neke od radnika. Zajednička obeležja su: JMBG, IME i DATUM\_RODJA. Obeležje FAKULTET (fakultet koji neko pohadja) karakteristično je za radnika studenta, dok je obeležje SPORT (sport kojim se osoba bavi) karakteristično za radnika sportista.

Za razliku od prethodnog primera ovde prepostavljamo da jedan radnik može istovremeno biti i radnik student i radnik sportista odnosno skupovi radnik student i radnik sportista ne moraju biti medjusobno disjunktni (neekskluzivna specijalizacija). Prikaz sa slike 7.10. možemo sada zameniti DOV datim na slici 7.11.



Slika 7.10.

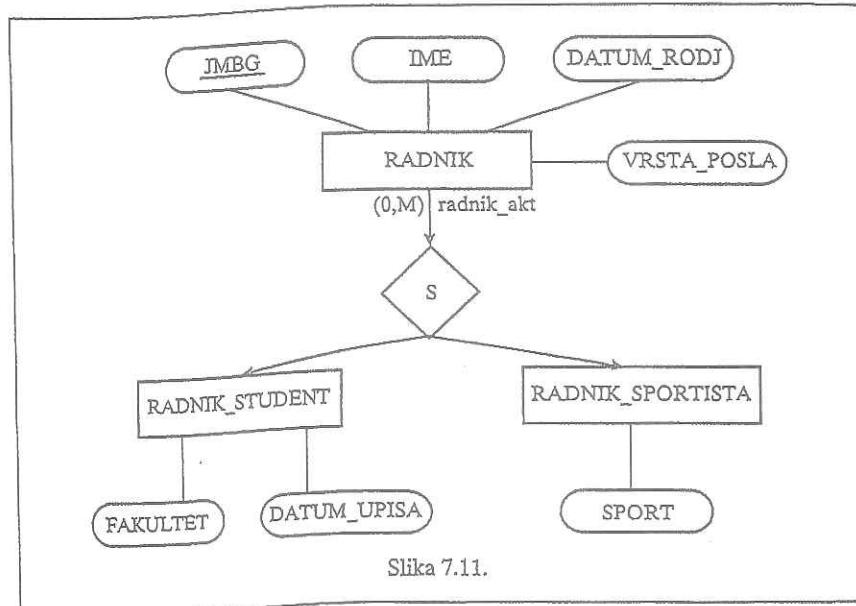
Posmatrajući sliku 7.9. i 7.11. uočavamo razliku u predstavljanju na DOV ekskluzivne i neekskluzivne specijalizacije. S obzirom da radnici studenti i radnici sportisti jesu radnici, model podataka dat DOV na slici 7.9. odnosi se i na njih. Da bi se u model uključilo i znanje predstavljeno DOV na slici 7.11.

potrebno je izvršiti integraciju ova dva podmodela pri čemu nastaje DOV na slici 7.12. Možemo uočiti da je neekskluzivna specijalizacija nastala uvođenjem podtipova objekata RADNIK\_STUDENT i RADNIK\_SPORTISTA nezavisna od postojeće ekskluzivne specijalizacije.

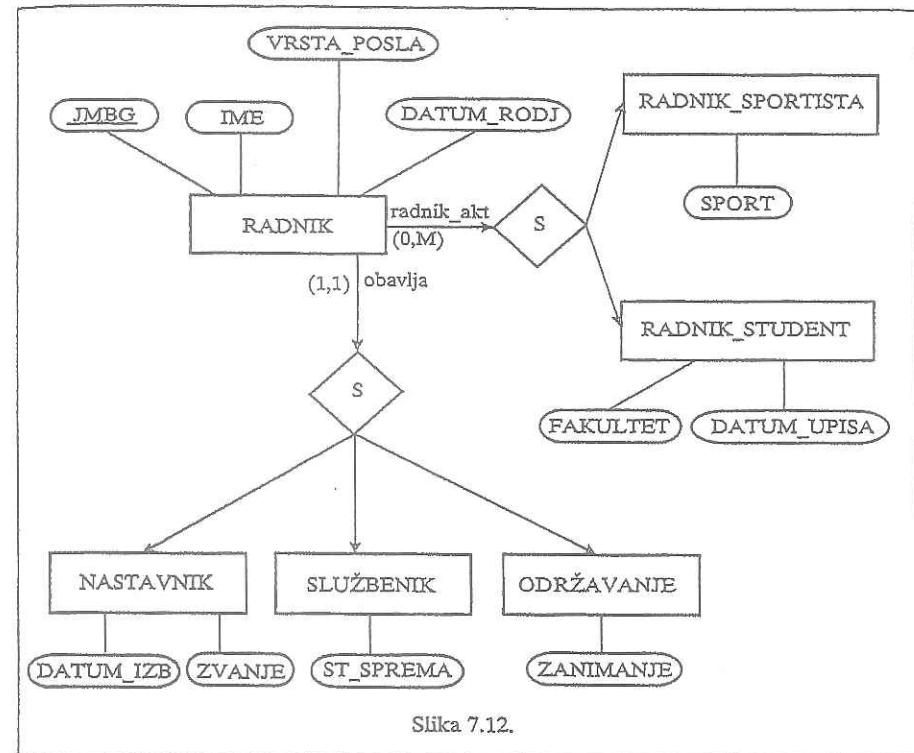
Jedan tip objekta može se po više različitih kriterijuma specijalizovati u različite skupove podtipova kao što je to ilustrovano za tip objekta radnik na slici 7.12.

U generalizacionoj zavisnosti objekata važi sledeće pravilo nasledjivanja osobina i pravilo nasledjivanja operacija:

- Podtipovi nasleđuju sva obeležja i veze svoga nadtipa.
- Podtipovi nasleđuju sve operacije svoga nadtipa.



Na primer, kako je predstavljeno na slici 7.12. tip objekta RADNIK\_STUDENT je podtip objekta RADNIK i nasleđuje obeležja svog nadtipa, JMBG, IME, DATUM\_RODJ, VRSTA\_POSLA. Obeležja FAKULTET i DATUM\_UPISA karakteristična su za tip objekta RADNIK\_STUDENT.



Definisanjem podtipova nekog tipa razrešava se problem obeležja koja nisu karakteristična za sve objekte u skupu objekta jednog tipa, na taj način što se definiše podtip kao skup pojavljivanja na koje je dato obeležje primenjivo. Podtip tada ima samo to obeležje (obeležja), a ostala obeležja karakteristična svima nasleđuju od nadtipa.

### Agregacija

U PMOV obeležje mogu imati osnovni, slabi i objekat podtip. Na DOV vezu je moguće uspostaviti samo izmedju dva tipa objekta. Kada je potrebno da tip veze ima obeležje ili da se uspostavi veza izmedju tipa veze i tipa objekta tada tip veze postaje *mešovit tip objekat-veza*. Mešovit tip objekta može imati obeležja i može se povezati sa drugim tipovima objekata.

Prevodjenje tipa veze u tip objekta izvodimo apstrakcijom agregacije u kojoj se veza izmedju dva ili više tipova objekata tretira kao objekat na višem nivou apstrakcije. Zbog toga što istovremeno predstavlja i tip objekat i tip veze, agregacija se naziva i mešovit tip objekat-veza. Objekti koji čine agregaciju se nazivaju komponentama agregacije. Postupak inverzan agregaciji se naziva *dekompozicija*. Kardinalnost preslikavanja KOMPONENTA → AGREGACIJA mora biti navedena, dok je za inverzno preslikavanje uvek DG = 1 i GG = 1.

Agregirani tip objekta se razlikuje od ostalih objekata u sistemu po tome što nema svoj sopstveni identifikator, već ga identifikuju samo objekti koje on agregira.

Primer agregacije dat je na slici 7.13. Saglasno tome da mešovit tip objekta istovremeno predstavlja i tip objekta i tip veze za njegovo predstavljanje na DOV koristi se kombinacija simbola za tip objekta i tip veze tj. pravougaonik u koji je ucrtan romb. Pojavljivanja objekta iz klase STUDENT i PREDMET agregiraju se u pojavljivanje objekta iz klase DIPLOMSKI. Agregacija DIPLOMSKI ima obeležje *datum\_dipl* i vezuje se sa klasom objekta NASTAVNIK, koji ocenjuju diplomski rad. Linija koja povezuje komponentu i aggregirani objekat ima strelicu na aggregiranom objektu, na njoj se upisuje kardinalnost i naziv preslikavanja komponenta aggregirani objekat. Podrazumevani naziv preslikavanja komponenta aggregirani objekat je naziv aggregiranog objekta i naziv tipa objekta komponente. Za DOV na slici 7.13. za naziv preslikavanja *iz*, podrazumevani naziv je *diplomski\_student*, a preslikavanja *za* je *diplomski\_predmet*.

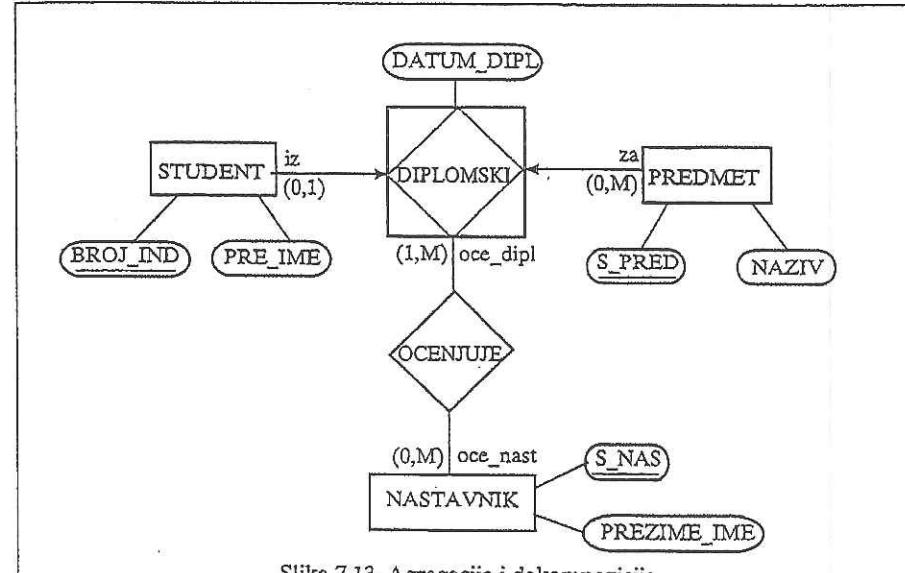
Agregirani objekat se u modelu tretira kao i bilo koji drugi objekat. To znači da on može da ima svoja obeležja i/ili da bude u vezi sa nekim drugim objektima (moguće aggregiranim, takodje), da ima svoje podtipove i slično.

Na kraju možemo rezimirati da PMOV ima sledeće koncepte:

1. Klase objekata:

- osnovni objekat (kernel),
- slabi objekat,
- mešovit objekat (agregacija),
- podtip.

2. Klasa veza.



Slika 7.13. Agregacija i dekompozicija

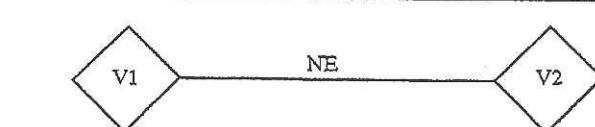
Pri gradnji DOV podtoče sledeća formalna ograničenja:

1. Tipovi osnovnih objekata ne smeju biti spojeni direktno bez tipa veze kao na slici 7.14. Izmedju dva osnovna tipa objekta može postojati samo tip veze ili aggregirani tip objekta.



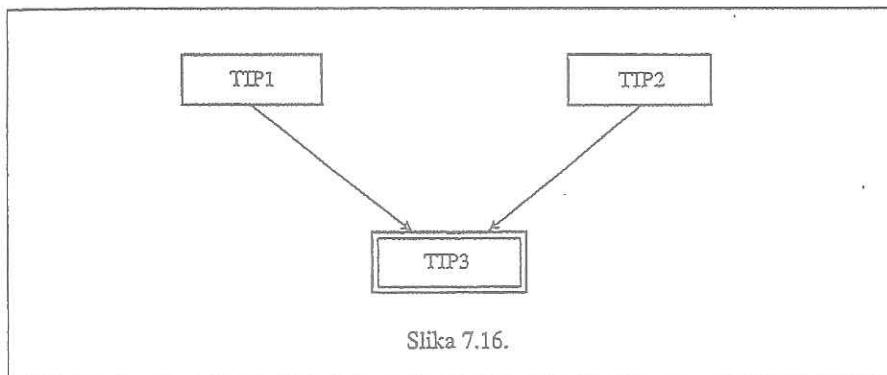
Slika 7.14.

2. Tipovi veze ne smeju biti direktno spojeni kao na slici 7.15.



Slika 7.15.

3. Tip slabog objekta može imati samo jedan nadredjeni objekat. Veze ilustrovane na slici 7.16. nisu dozvoljene.



## 7.2. Integritetna komponenta

Radi očuvanja integriteta podataka u modelu objekti veze uvedene su sledeće dve klase ograničenja:

1. Struktura ograničenja (inherentna) ograničenja,
2. Posebna (eksplicitna) ograničenja.

Ograničenja definisana strukturom MOV su:

- a) identifikacija objekta,
- b) ograničenje postojanja pojave jednog tipa objekta u zavisnosti od postojanja pojave drugog tipa objekta (egzistencijalna zavisnost),
- c) ograničenje mogućnosti identifikacije pojave jednog tipa objekta bez poznavanja identifikatora drugog tipa objekta,
- d) specijalni tipovi veze.

U prethodnom delu prilikom uvođenja i definisanja pojedinih koncepcata strukture MOV, definisana su navedena strukturalna ograničenja koja se iskazuju na DOV.

Posebna pravila integriteta ne mogu se predstaviti strukturom modela. Na primer, obeležje ZVANJE objekta NASTAVNIK može da ima vrednost iz skupa {docent, vanredni profesor, redovni profesor} što se ne može predstaviti strukturom modela.

Postoje sledeće vrste posebnih ograničenja:

### a) Ograničenje na dozvoljene vrednosti domena

Objekti u sistemu se opisuju pomoću svojih obeležja. Svako obeležje u jednom trenutku vremena ima neku vrednost. Obeležja uzimaju vrednosti iz skupa pojedinačnih vrednosti koje nazivamo *domen*. Svaki domen se tretira kao podtip *standardnih domena*. Pod standardnim domenom podrazumevamo tipove podataka koji postoje u standardnim programskim jezicima (ceo broj, niz karaktera i sl.). Dozvoljene vrednosti u domenu mogu se definisati eksplicitnim navođenjem svih dozvoljenih vrednosti ili definisanjem jednog ili više intervala iz kojih se vrednosti mogu, ali ne moraju pojaviti. Na primer domen Datum je skup datuma koji se nalaze u intervalu:

01.01.1900. ≤ datum ≤ današnji datum.

### b) Ograničenje na dozvoljene vrednosti obeležja

Obeležja objekata uzimaju vrednosti iz domena. Ograničenje na vrednost obeležja može se postaviti tako da obeležje može poprimiti samo uži skup vrednosti iz domena. Na primer, obeležje datum rođenja definisano je nad domenom datum, ali se vrednosti mogu ograničiti od 01.01.1900. do tekućeg datuma. Ograničenje vrednosti jednog obeležja može se postaviti u odnosu na vrednost nekog drugog obeležja. Na primer, obustave radniku ne mogu biti veće od jedne trećine zarade radnika.

### c) ograničenje na vrstu preslikavanja (kardinalnost) izmedju tipa objekta i vrednosti obeležja

Kardinalnosti preslikavanja izmedju tipa objekta i vrednosti obeležja mogu biti:

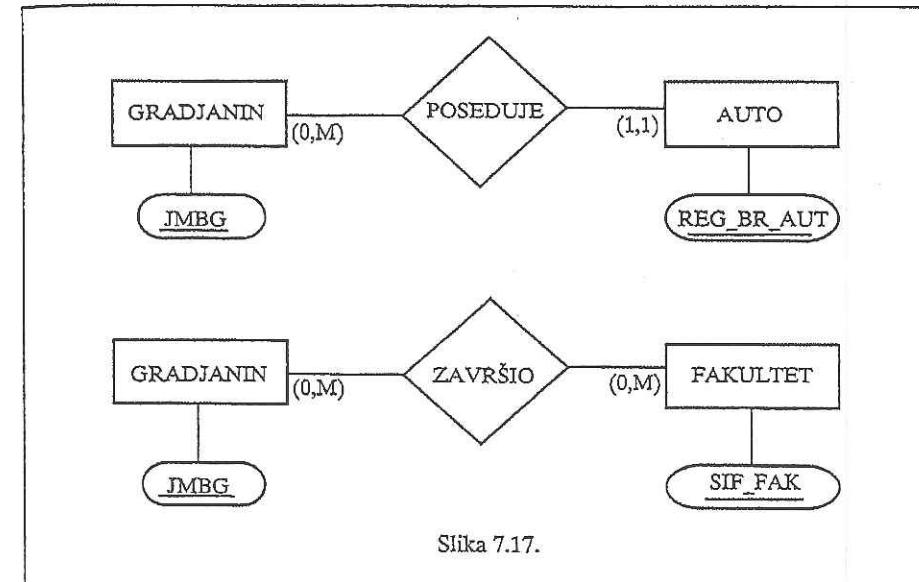
- (1) Tipa (1:1) - jedan tip objekta ima samo jednu vrednost nekog obeležja, a tu vrednost obeležja može imati samo jedna pojave tipa objekta. Na primer, jedan građanin za obeležje JMBG ima jednu vrednost, i ta vrednost

pripada samo jednom gradjaninu. Obeležje JMBG može da predstavlja identifikator tipa objekta GRADJANIN.

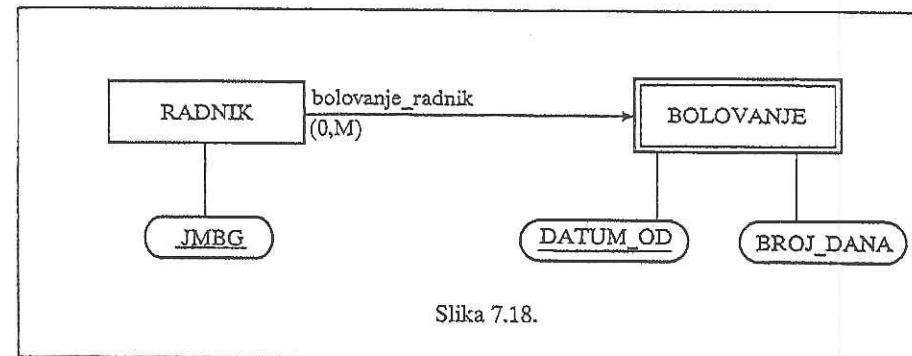
- (2) Tipa (1:M) - jedan tip objekta ima samo jednu vrednost nekog obeležja, i istu vrednost obeležja mogu imati više pojava tipa objekta. Na primer, jedan gradjanin za obeležje DATUM\_RODJ ima jedan datum, ali isti datum rođenja može imati više gradjana.
- (3) Tipa (M:1) - jedan tip objekta može imati više vrednosti nekog obeležja, i istu vrednost obeležja može imati samo jedna pojava tipa objekta. Na primer, jedan gradjanin za obeležje REG\_BROJ\_AUT (registarski broj automobila) može imati više vrednosti (vlasnik je više automobila), ali jedan registarski broj pripada samo jednom gradjaninu.
- (4) Tipa (M:M) - jedan tip objekta može imati više vrednosti nekog obeležja, i istu vrednost obeležja može imati više pojava tipa objekta. Na primer, jedan gradjanin za obeležje FAKULTET može imati više vrednosti (završio je više fakulteta) i isti fakultet imaju (završilo je) više gradjana.

Iz praktičnih razloga, zbog jednostavnosti prevodjenja MOV u model u kojem će se izvršiti realizacija baze podataka u MOV se ne koriste višeznačna obeležja koja su prethodno predstavljena slučajem (3) i (4). Situacija iz realnog sistema predstavljena višeznačnim obeležjima u modelu se obuhvata na jedan od sledeća dva načina:

1. Ako domen višeznačnog obeležja tipa objekta ima unapred zadat semantički značajan skup vrednosti, tada se to obeležje modelira kao klasa objekata tj. novim tipom objekta, a višeznačnost obeležja predstavlja novodefinisanom vezom posmatranog tipa objekta sa ovim novim tipom objekta. Na DOV na slici 7.17. ilustrovane su situacije iz primera (3) i (4) bez korišćenja višeznačnih obeležja.
2. Ako domen višeznačnog obeležja nema unapred zadat semantički značajan skup vrednosti, tada ga je pogodno predstaviti pomoću tipa slabog objekta. Na primer, jedan radnik za bolovanje predstavljeno obeležjima DATUM\_OD (datum početka bolovanja) i BROJ\_DANA (broj dana bolovanja) može imati više vrednosti pa se bolovanje predstavlja posebnim tipom slabog objekta koji ima osobinu identifikacione zavisnosti od nadređenog objekta RADNIK kako je to ilustrovano DOV na slici 7.18.



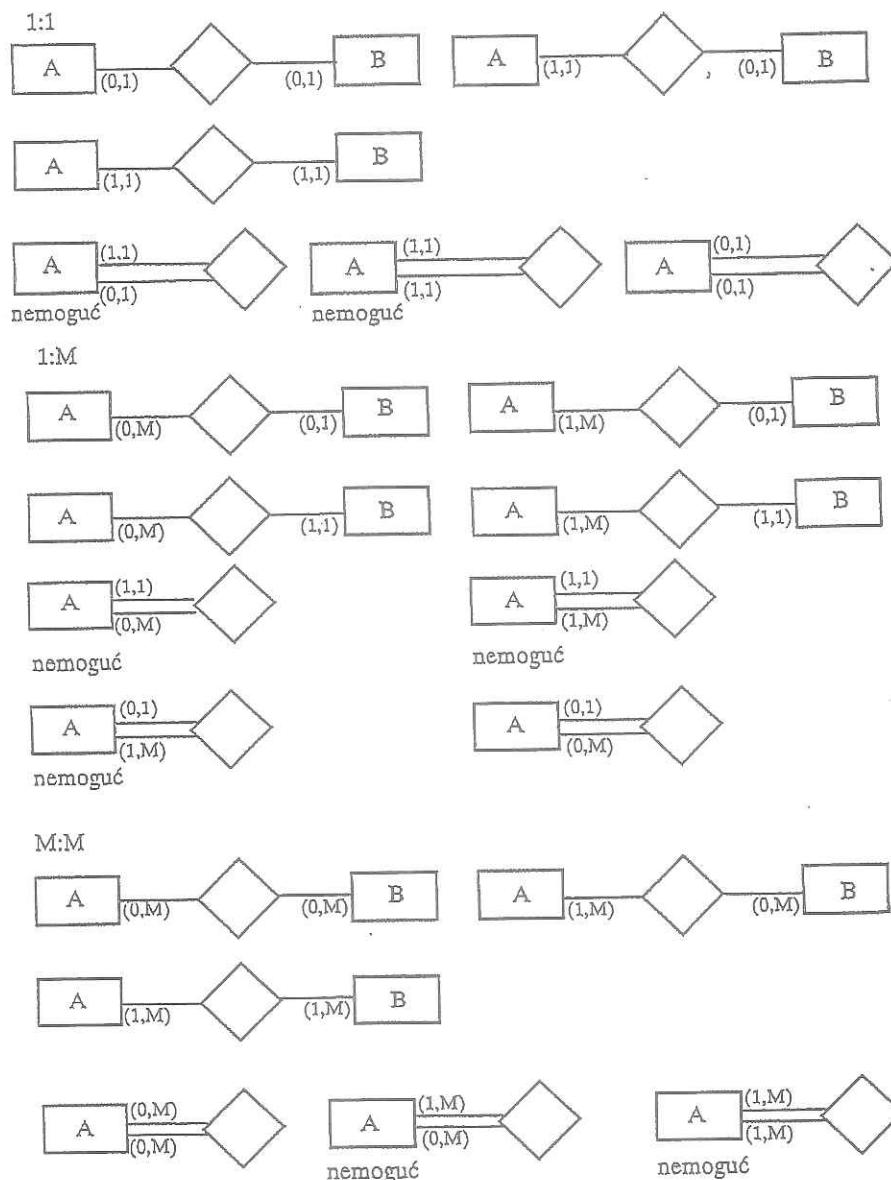
Slika 7.17.



Slika 7.18.

#### d) Ograničenje na kardinalnost preslikavanja izmedju tipova objekata

Kardinalnost preslikavanja izmedju tipova objekata eksplicitno se upisuje na DOV. Dalje se daje pregled veza u MOV sa oznakom u praksi nemogućih odnosa.



### 7.3. Operacijska komponenta

U vreme nastanka modela objekti - veze, JMP nije bio odgovarajuće definisan. Tokom osamdesetih godina predloženo je niz različitih JMP pri čemu većina ne sadrži potpun skup operatora nad strukturu MOV. Ovde će dalje biti izložen JMP prema PMOV definisanom na FON-u.

U MOV operacije se izvršavaju nad klasama objekata (skupu pojavljivanja određenog tipa objekta) i klasama preslikavanja (skup pojavljivanja određenog tipa veze) i nad celom bazom podataka. Promenljive operacije su **objektne promenljive**, tj. promenljive koje uzimaju vrednost iz date klase. Pogodno je operacije MOV klasifikovati na operacije izveštavanja (pretraživanja) i operacije održavanja (ažuriranja).

#### 7.3.1. Operacije izveštavanja

Operacije izveštavanja formiraju željene izveštaje kao izvedene (složene) objekte iz osnovnih (baznih) objekata. Radi ilustracije formiranja izvedenog objekta razmotrimo primer na slici 7.19. Na slici 7.19. tabelarno je prikazan izgled izведенog objekta koji za svaku šifru i naziv predmeta sadrži broj indeksa, prezime i ime studenta koji je položio taj predmet, ocenu i srednju ocenu.

Izvedeni objekat ilustrovan na slici 7.19. nastao je od osnovnih objekata PREDMET i STUDENT.

PREDMET S PREDNAZIV	STUDENT				
	BROJ IND	PRE	IME	OCENA	PROSEK
002 OSNOVNE RAČUNARSTVA	2010	SIMA	7	8,5	
	4030	JOVAN	10		
003 STRUKTURE I BP	-	-	-	-	
005 PROGRAMIRANJERS	2010	SIMA	6	8	
	2020	ANA	9		
	3040	ACA	9		

Slika 7.19.

Sintaksa jezika za izvodjenje složenih objekata se bazira na sintaksi upitnih jezika relacionog tipa i nećemo je ovde razmatrati. U okviru relacionog modela podataka biće razmatrana sintaksa upitnih jezika i tada će biti jasno kako se može specificirati izraz za dobijanje izveštaja na slici 7.19.

### 7.3.2. Operacije održavanja baze podataka

Operacije nad klasama objekata i klasama preslikavanja su operacije koje se izolovano izvode nad jednom klasom objekata ili jednom klasom preslikavanja. Ove operacije se izvode bez obzira na struktura i vrednosna ograničenja modela, samo se u operaciji nad klasom objekta proveravaju zadata ograničenja nad domenima odgovarajućih obeležja.

#### 7.3.2.1. Operacije nad klasama objekata i klasama preslikavanja

Za specifikaciju operacija koristi se sintaksa slična većini relacionih upitnih jezika. Ovde će sintaksa i semantika ovih operacija biti iskazana navodnjem naziva operacije, ulaznih parametara odnosno argumenata operacije, preduslova za obavljanje operacije (*pre:*) i efekata operacije iskazanih preko postuslova (*post:*).

Koriste se sledeće oznake:

- X - naziv klase objekata nad kojom se vrši operacija,
- x - jedno pojavljivanje klase X,
- Y - naziv klase objekata kodomena datog preslikavanja,
- y - jedno pojavljivanje kodomena datog preslikavanja,
- X.P - naziv preslikavanja,
- x.P - podskup preslikavanja P za dato pojavljivanje x,
- $Y.P^{-1}$  - inverzno preslikavanje preslikavanju P.

Operacije su:

1. **upiši\_X(x)**

- pre:* ne postoji x u X (ne postoji pojavljivanje x u klasi X)
- post:* postoji x u X

2. **briši\_X(x)**

- pre:* postoji x u X
- post:* ne postoji x u X

3. **promeni\_X( $x_{1s}, x_{1n}$ )**

- pre:* postoji  $x_1$  u X
- post:* obeležja datog pojavljivanja date klase ( $x_{1s}$ ) dobijaju nove vrednosti ( $x_{1n}$ )

4. **poveži\_X.P(x,y)**

- pre:* ne postoji  $\langle x,y \rangle$  u klasi preslikavanja X.P
- post:* postoji  $\langle x,y \rangle$  u klasi preslikavanja X.P

5. **razveži\_X.P(x,y)**

- pre:* postoji  $\langle x,y \rangle$  u klasi preslikavanja X.P
- post:* ne postoji  $\langle x,y \rangle$  u klasi preslikavanja X.P

6. **preveži\_X(y,y<sub>1</sub>,y<sub>2</sub>)**

- pre:* postoji  $\langle x,y_1 \rangle$  u klasi preslikavanja X.P i  
ne postoji  $\langle x,y_2 \rangle$  u klasi preslikavanja X.P
- post:* ne postoji  $\langle x,y_1 \rangle$  u klasi preslikavanja X.P i  
postoji  $\langle x,y_2 \rangle$  u klasi preslikavanja X.P

Pretpostavljamo da su u prethodne operacije ugradjene i standardne poruke koje se pojavljuju u slučaju ako neki uslov nije zadovoljen ili kada se iz nekog razloga operacija nije mogla korektno izvršiti.

I kao što je bilo rečeno da se operacije održavanja izvode nezavisno od uslova integriteta iste mogu da naruše integritet baze podataka. Zbog toga je neophodno uvesti operacije nad bazom podataka, koje su analogne prethodno definisanim, s tim što njihova semantika obuhvata specifikaciju akcija koje treba preuzeti ukoliko dodje do narušavanja strukturnih ograničenja.

### 7.3.2.2. Operacije nad bazom podataka - struktturna pravila integriteta

Operacije nad bazom podataka nisu ograničene samo na jednu klasu objekta ili preslikavanja već mogu da se "šire" po celoj bazi podataka, s tim da se zadovolje sva struktura i vrednosna ograničenja definisana modelom.

Osnovne operacije nad bazom podataka su: **UPIŠI**, **BRIŠI**, **PROMENI**, **POVEŽI**, **RAZVEŽI** i **PREVEŽI**. Navedene operacije su analogne prethodno definisanim operacijama nad klasma objekata i klasma preslikavanja, s tim da se za svaku od njih mora definisati i način zadovoljavanja svakog ograničenja koje bi jednom takvom operacijom, moglo da bude narušeno.

Način na koji se zadovoljava strukturalni integritet pri izvodjenju neke operacije nazivamo *struktturnim pravilom integriteta* (SPI). Vrste strukturalnih pravila integriteta su skraćeni načini opisivanja akcija (makro operacija) koje se preduzimaju kada neka operacija naruši neko strukturalno ograničenje.

Moguće su sledeće akcije:

- (1) **RESTRICTED** - Operacija se ne izvodi ili se njeni efekti poništavaju, ako je uslov integriteta narušen.
- (2) **NULLIFIES** - Narušeni integritet se zadovoljava kreiranjem *null objekta* koji zamenjuje objekat čije je nepostojanje uzrok narušavanja integriteta.
- (3) **DEFAULT** - Narušeni integritet se zadovoljava kreiranjem *default objekta* koji zamenjuje objekat čije je nepostojanje uzrok narušavanja integriteta.
- (4) **CASCADES** - Osnovna operacija se nastavlja operacijama zadovoljavanja integriteta nad objektima kod kojih je integritet narušen.

Za svaku operaciju održavanja nad bazom podataka i prema svakom konceptu iz okoline klase objekta ili klase preslikavanja za koju je operacija zadata, zadaje se kao ulazni parametar specifikacija strukturalnog pravila integriteta (SPI). Ako prema nekom konceptu iz okoline nije zadato SPI, smatra se da operacija ne može da naruši strukturalni integritet prema tom konceptu. Okolinu za operacije **UPIŠI** i **BRIŠI** čine sva preslikavanja posmatrane klase, a za operacije **POVEŽI**, **PREVEŽI** i **RAZVEŽI** klasa objekata kodomena preslikavanja. Operacija **PROMENI** ne može da naruši strukturalni integritet modela.

Sledi opis semantike svih osnovnih operacija održavanja nad bazom podataka.

Koriste se sledeće oznake:

- |                   |   |
|-------------------|---|
| X                 | - naziv klase objekata nad kojom se vrši operacija,                 |
| x                 | - jedno pojavljivanje klase X,                                      |
| Y                 | - naziv klase objekata kodomena datog preslikavanja,                |
| y                 | - jedno pojavljivanje kodomena datog preslikavanja,                 |
| X.P               | - naziv preslikavanja,  |
| x.P               | - podskup preslikavanja P za dato pojavljivanje x,                  |
| Y.P <sup>-1</sup> | - inverzno preslikavanje preslikavanju P,                           |
| y.P <sup>-1</sup> | - podskup inverznog preslikavanja P za dato pojavljivanje y,        |
| vspi              | - jedno od RESTRICTED (R), CASCADES (C), NULLIFIES (N), DEFAULT(D), |
| nula-y            | - nula objekat klase Y,   |
| default-y         | - default objekat klase Y.  |

(1) **UPIŠI\_X(x,y<sub>1</sub>,y<sub>2</sub>,...,y<sub>n</sub>)**

SPI: vspi X.P<sub>1</sub>, vspi X.P<sub>2</sub>,..., vspi X.P<sub>n</sub>

Argumenti za operaciju **UPIŠI** su jedno pojavljivanje x i po jedno pojavljivanje objekata iz klase kodomena, za sva obavezna preslikavanja.

```

if nisu zadati svi argumenti
    them zadaj sve argumente;
end if;
if x JE_U X
    them odbij_operaciju;
end if;
for each X.Pi iz SPI do
    if card (yi;Pi-1) = GG (Pi-1)
        them odbij_operaciju;
    end for;
    upisi_X(x);
    for each Pi iz SPI do
        if yi JE_U Yi
            then povezi_X.Pi (x,yi)
        else case vspi of
            R: odbij_operaciju;
            N: povezi_X.Pi (x, nula_y);
        end case;
    end for;
end if;

```

```

D: poveži_X.Pi(x, default_y);
C: UPIŠI_Yi(yi);
    poveži_X.Pi(x,yi);
end case;
end if;
end for;
end;

```

Iskaz **odbij\_operaciju** znači povratak u program koji je zahtevao operaciju, i koji će dati odgovarajuću poruka u zavisnosti od mesta gde se ovaj iskaz javlja.

**(2) BRIŠI\_X(x)**  
SPI: vspi X.P<sub>1</sub>, vspi X.P<sub>2</sub>,..., vspi X.P<sub>n</sub>

Argument za operaciju **BRIŠI** je jedno pojavljivanje objekta x, odnosno samo njegov identifikator.

```

if x NOT JE_U X
    then odbij_operaciju;
end if;
for each X.Pi do
if POSTOJI y JE_U x.Pi AND card(y.Pi-1) = DG(Pi-1)
    AND X.Pi JE_U SPI
    then case vspi of
        R: odbij_operaciju;
        N: preveži_Y.Pi-1(y, x, nula_x);
        D: preveži_Y.Pi-1(y, x, default_x);
        C: for each y iz x.Pi do
            razveži_y.Pi-1(y, x);
            BRIŠI_Y(y);
            end for;
        end case;
    end if;
end for;
briši_X(x);
end;

```

**(3) PROMENI\_X(x<sub>1s</sub>,x<sub>1n</sub>)**

Argument x<sub>1s</sub> je stara, a x<sub>1n</sub> je nova pojava objekta X koji se menja. Ova operacija ne narušava struktura ograničenja u MOV.

**(4) POVEŽI\_X.P(x, y)**  
SPI: vspi Y

```

if x NOT JE_U X
    then odbij_operaciju;
end if;
if <x, y> JE_U X.P
    then odbij_operaciju;
end if;
if card(x.P) = GG(X.P)
    then odbij_operaciju;
end if;
if card(y.P)-1 = GG(Y.P-1)
    then odbij_operaciju;
end if;
if y JE_U Y
    then poveži_X.P(x, y)
else case vspi of
    R: odbij_operaciju;
    N: poveži_X.Pi(x, nula_y);
    D: poveži_X.P(x, default_y);
    C: UPIŠI_Y(y)
        poveži_X.P(x, y);
    end case;
end if;
end;

```

**(5) RAZVEŽI\_X.P(x, y)**  
SPI: vspi Y

```

if <x, y> NOT JE_U X.P
    then odbij_operaciju;
end if;

```

```

if card(x.P) = DG(X.P)
  then odbij_operaciju;
end if;
if card(y.P)-1 > DG(Y.P-1)
  then razveži_X.P(x, y)
else case vspis of
  R: odbij_operaciju;
  N: preveži_Y.P-1(y, x, nula_x);
  D: preveži_Y.P-1(y, x, default_x);
  C: BRIŠI_Y(y);
    RAZVEŽI_X.P(x,y);
  end case;
end if;
end;

```

(6) PREVEŽI\_X.P(x,y<sub>1</sub>,y<sub>2</sub>)  
SPI: vspis Y (sa), vspis Y (na)

```

if <x, y1> NOT JE_U X.P
  then odbij_operaciju;
end if;
if <x, y2> JE_U X.P
  then odbij_operaciju;
end if;
if card(x.P) = DG(X.P)
  then odbij_operaciju;
end if;
if card(y2.P)-1 > GG(Y.P-1)
  then odbij_operaciju;
end if;
if card(y1.P)-1 > DG(Y.P-1)
  then if y2 JE_U Y
    then razveži_X.P(x, y1);
      poveži_X.P(x, y2);
    else razveži_X.P(x, y1);
      RAZVEŽI_X.P(x, y2);
    end if;
  end if;

```

(pod uslovom vspis (na))

```

then poveži_X.P(x,y2)
  RAZVEŽI_X.P(x, y1); (pod uslovom vspis (sa))
else POVEŽI_X.P(x, y2); (pod uslovom vspis (na))
  RAZVEŽI_X.P(x, y1); (pod uslovom vspis (sa))
end if;
end if;
end;

```

Detaljnijim pregledom operacija održavanja nad bazom podataka može se uočiti da je dovoljno zadati odgovarajuće akcije u slučaju narušavanja strukturnih pravila integriteta samo za operacije POVEŽI i RAZVEŽI. To su već opisane četiri standardne akcije: RESTRICTED (R), CASCADES (C), NULLIFIES (N), DEFAULT (D). Koja akcija će se odabratи zavisi od prirode odnosa posmatranog objekta i objekta koji je u vezi. Kada i gde u procesu projektovanja odrediti odgovarajuće akcije?. S obzirom da se ove akcije zadaju za sva preslikavanja to je pogodno mesto za označavanje na DOV. Takođe, pogodno vreme za određivanje odgovarajućih akcija je prilikom izrade DOV. Pravila integriteta označavamo početnim slovima akcije (R, C, N, D) na liniji povezivanja objekata i to za operaciju POVEŽI i RAZVEŽI.

Akcija za operaciju POVEŽI x i y odnosi se na slučaj kada nema pojave y.

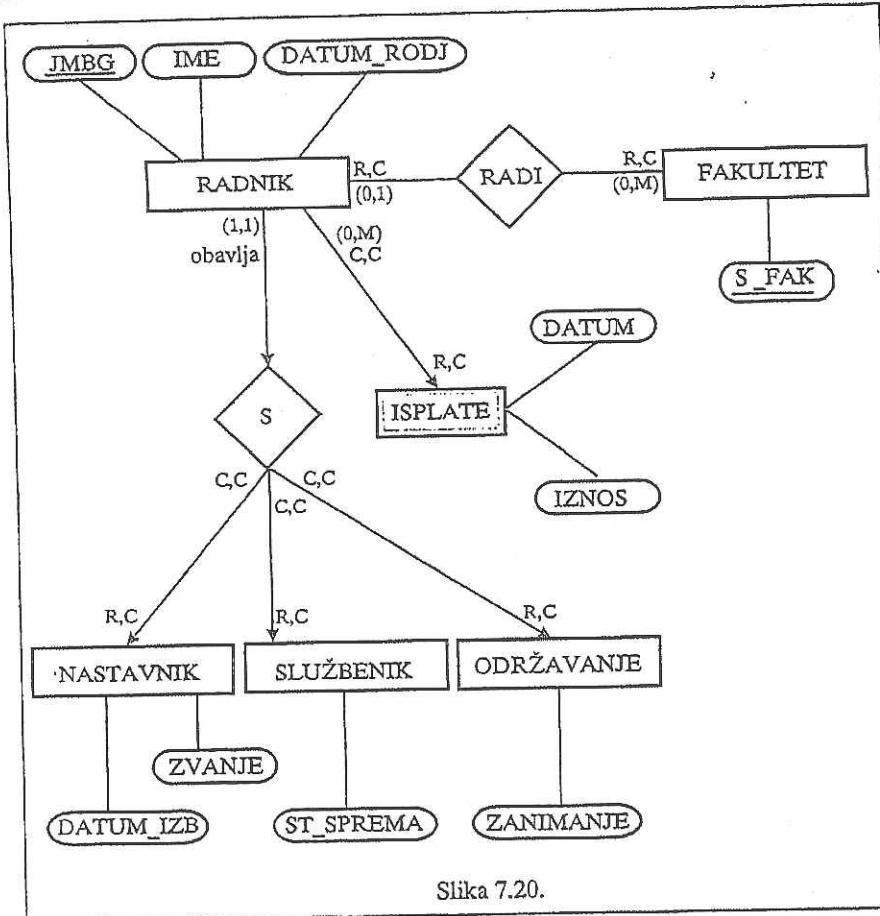
Akcija za operaciju RAZVEŽI odnosi se na slučaj kada bi time bila narušena donja granica preslikavanja y.P<sup>-1</sup>.

Na DOV na slici 7.20. ilustrovani su načini označavanja pravila integriteta.

Ilustracije radi objasnićemo akcije označene na liniji koja spaja tipove objekata RADNIK i ISPLATE na slici 7.20.

Oznaka C,C (prvo C odnosi se na operaciju POVEŽI, drugo C na operaciju RAZVEŽI) znači:

- ako operacija POVEŽI jednu pojavu tipa objekta RADNIK - x sa jednom pojmom tipa objekta ISPLATA - y ne može biti izvršena jer nema y, biće preduzeta akcija C (CASCADES) tj. biće ubačeno y čime bi bili ispunjeni uslovi za povezivanje što bi se i izvršilo,
- ako bi operacijom RAZVEŽI jednu pojavu tipa objekta RADNIK - x i jednu pojavu tipa objekta ISPLATE - y, bila narušena donja granica kardinalnosti preslikavanja ISPLATE - RADNIK tada treba brisati y čime bi bili ispunjeni uslovi za izvršavanje operacije RAZVEŽI.



Oznaka R,C (R se odnosi na operaciju POVEŽI, C na operaciju RAZVEŽI) znači:

- ako operacija POVEŽI jednu pojavu tipa objekta ISPLATE - x sa jednom pojavom tipa objekta RADNIK - y ne može biti izvršena jer nema y biće preduzeta akcija R (RESTRICTED) tj. operacija POVEŽI neće se izvršiti.
- ako bi operacijom RAZVEŽI jednu pojavu tipa objekta ISPLATE - x i jednu pojavu tipa objekta RADNIK - y, bila narušena donja granica kardinalnosti preslikavanja RADNIK-ISPLATE (u ovom primeru to nikada nije slučaj) tada treba brisati y čime bi bili ispunjeni uslovi za izvršavanje operacije RAZVEŽI.

#### 7.4. Proces izrade MOV

Izrada MOV za dati realni sistem obično se odvija u sledećim koracima:

- 1) Izrada MOV po delovima (podmodel - jedan DOV),
- 2) Integracija delova u jednu celinu koju nazivamo *globalnim modelom podataka*.
- 3) Uključivanje drugih semantičkih detalja u model, pre svega uslova integriteta.

Treba odmah istaći da je proces izrade MOV iterativan tj. da naknadna saznanja i zahtevi često dovode do izmena prethodnih rešenja.

Prilikom ručne izrade MOV preporučuje se korišćenje sledećih obrazaca:

- obrazac za dijagram strukture tipa objekta,
- obrazac za dijagram strukture tipa veze,
- obrazac za globalni DOV.

Obrazac za opis strukture tipa objekta daje detaljan opis svakog tipa objekta.

Uputstva za njegovo popunjavanje su sledeća:

- jedan tip objekta prikazati na jednoj stranici,
- prikazati samo jedan nivo u hijerarhiji podtipova,
- prikazati sva obeležja sa naznakom identifikacionih (podvlačenjem).

Na slici 7.21. dat je primer popunjeno obrasca za dijagram strukture tipa objekta.

Obrazac za opis strukture tipa veze prikazuje jedan tip veze i sve tipove objekata koji u vezi učestvuju. Mešovit tip objekat-veza se takođe prikazuje na obrascu za opis strukture tipa veze.

Na slici 7.22. dat je primer popunjeno obrasca za dijagram strukture veze.

Obrazac za globalni DOV koristi se za prikazivanje tipova objekata i njihovih međusobnih veza. Globalni DOV se pri tome mora podeliti na više stranica.

Na slici 7.23. dat je primer popunjeno obrasca za globalni DOV.

**DIAGRAM STRUKTURE TIPOA OBJEKTA**

Naziv sistema: IS fakulteta  
Analitičar: Petrović Saša

Strana: 10  
Datum: 200294

Opis:

```

graph TD
    NASTAVNIK[NASTAVNIK] --- S_NAS[S_NAS]
    NASTAVNIK --- PREZIME_IME[PREZIME_IME]
    NASTAVNIK --- ZVANJE[ZVANJE]
    NASTAVNIK --- PLATA[PLATA]
    NASTAVNIK --- DODATAK[DODATAK]
  
```

Broj pojavljivanja: Maximum 10000 Srednji 1000 Minimum \_\_\_\_\_  
Ovlašćenje za promene: Odeljenje opštih poslova  
Uslovi ubacivanja: Pri zapošljavanju  
izbacivanja: Ostaje trajno u bazi podataka

Slika 7.21.

**DIAGRAM STRUKTURE TIPOA VEZA**

Naziv sistema: IS fakultet  
Analitičar: Petrović Saša

Strana: 23  
Datum: 200294

Opis:

```

graph LR
    NASTAVNIK[NASTAVNIK] -- "radi na (0,1)" --> RADI{RADI}
    RADI -- "zapošljava (0,M)" --> FAKULTET[FAKULTET]
    FAKULTET --- S_FAK[S_FAK]
  
```

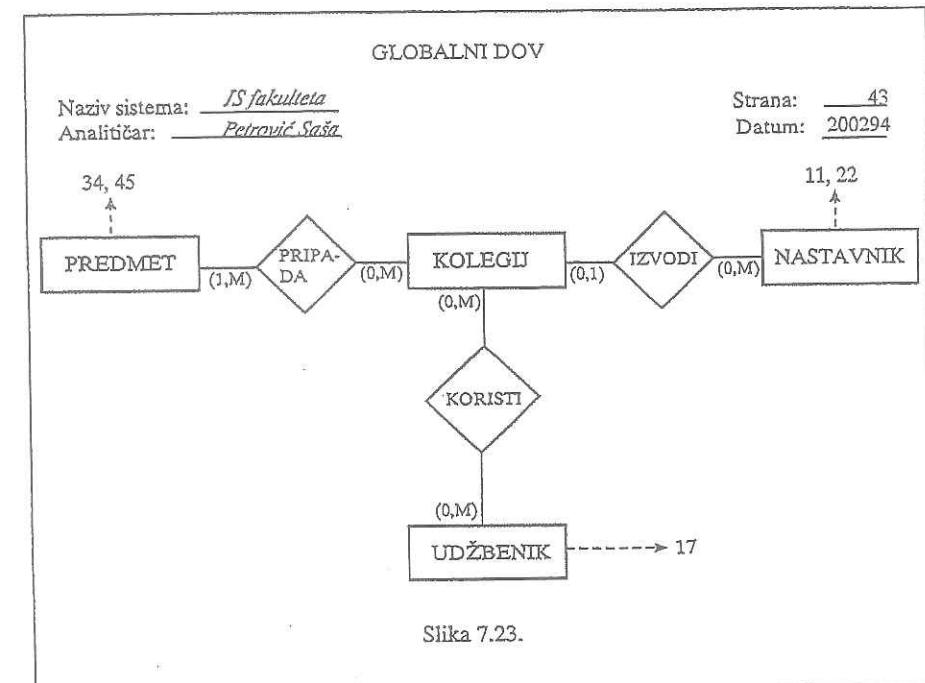
Broj pojavljivanja: Maximum 10000 Srednji 1000 Minimum \_\_\_\_\_  
Ovlašćenje za promene: Odeljenje opštih poslova  
Uslovi ubacivanja: Pri zapošljavanju  
izbacivanja: Ostaje trajno u bazi podataka

Slika 7.22.

Preporuke pri crtanju globalnog DOV su:

- na jednoj stranici treba predstaviti 3 do 7 tipova objekata,

- vezu ne treba razbijati, tj. svi tipovi objekata koji učestvuju u vezi treba da budu na jednoj stranici,
- isprekidane linije pokazuju da neki tip objekta učestvuje i u nekim drugim vezama, sa brojčanom oznakom stranice na kojoj su te veze opisane, odnosno gde je ponovo prikazan taj tip objekta.



Postoje više pristupa za izradu DOV. Opšta karakteristika svih pristupa je sledeći skup koraka:

1. Određivanje tipova objekata.
2. Određivanje obeležja tipova objekata.
3. Identifikacija ključnih obeležja tipova objekta.
4. Određivanje veza izmedju tipova objekata.

Treba napomenuti da se prethodni koraci ne moraju izvoditi prema navedenom redosledu, te da nema opšte saglasnosti u redosledu njihovog izvođenja. Jedan od pristupa je da se prethodni koraci izvode u redosledu: 1, 4, 2, 3.

Na bazi pretpostavke da ljudi svoje misli izražavaju koristeći koncepte **objekta, veze, obeležja i vrednosti** tj. sličnosti DOV i iskaza na prirodnom jeziku, analiza iskaza na prirodnom jeziku može poslužiti kao početna osnova za određivanje osnovnih koncepata u MOV. Postupak je posebno pogodan kada postoji neki formalizovani opis realnog sistema na prirodnom jeziku.

Veza MOV i opisa na prirodnom jeziku može se prikazati na sledeći način:

<u>opis na prirodnom jeziku</u>	<u>koncept MOV</u>
knjiga	model
poglavlje	podmodel
rečenica	objekti povezani vezama
imenica	tip objekta
glagol	tip veze
pridev	obeležje tipa objekta
prilog	obeležje tipa veze
glagolska imenica	mešovit tip objekat - veza.

Pri izradi podmodela posmatraju se obično zahtevi jednog korisnika. Prvo se utvrđuju tipovi objekata sistema. Zatim se određuju veze izmedju tih tipova objekata i na kraju obeležja tipova objekata sa posebnom pažnjom na određivanje identifikatora tipova objekata.

U drugom koraku vrši se integracija podmodela datih posebnim DOV u jedan jedinstveni globalni model podataka pri čemu se moraju ukloniti protivrečnosti i redundansa.

#### 7.4.1. Određivanje tipova objekata

Mada smo pojam tipa objekta ranije definisali dodajmo i to da isti predstavlja osnovni sadržaj pojave ili procesa o kojem treba imati informacije.

Da bi se lakše donela odluka o tome da li neki koncept realnog sistema predstavlja tip objekta, da li dva ili više koncepata predstavljaju isti tip objekta, kao i da bi se obuhvatili svi tipovi objekata korisno je poslužiti se sledećim preporukama:

- (1) **Sličnost obeležja.** Značajnija razlika odnosno sličnost u obeležjima govori o tome da se radi o različitim odnosno istim tipovima objekata.
- (2) **Način identifikovanja** Za svaki tip objekta mora da postoji jedno obeležje ili skup obeležja koja jedinstveno identifikuju pojavu tipa objekta.
- (3) **Učešće u tipu veze** Da bi se odredilo da li objekat jednog tipa predstavlja složen objekat sastavljen od nekih drugih tipova objekata treba ispitati veze u kojima ovi objekti učestvuju. Tako se mogu identifikovati podtipovi objekta.
- (4) **Na osnovu poznatih veza**: Svaka aktivnost je neka vrsta veze. Na primer, *predavati, položiti, rukovoditi* itd. Definišući ko vrši neku aktivnost mogu se identifikovati pojedini tipovi objekata. Na primer:

<u>aktivnost</u>	<u>izvršilac</u>
- predaje	- nastavnik
- predavan	- predmet
- polaže	- student
- položio	- predmet
- rukovodi	- rukovodilac
- rukovodjen	- katedra.

- (5) **Na osnovu obeležja**: Projektantu su obično poznati različiti dokumenti iz realnog sistema koji sadrže mnoštvo podataka. Analiziranjem takvih dokumenata mogu se definisati tipovi objekata. Ilustrujmo to sledećim primerom:

<u>podatak</u>	<u>pitanje</u>	<u>odgovor</u>
boja	boja čega?	automobila
broj indeksa	čiji broj indeksa	studenta
prezime ime	čije prezime ime?	studenta

- (6) **Fizički objekti**: Po pravilu svi fizički objekti u posmatranom realnom sistemu predstavljeni su nekim tipom objekta u modelu podataka. Napomenimo da obrnuto ne važi.
- (7) **Razrešavanje dileme** obeležje ili objekat: Ovo pitanje se često postavlja u modeliranju podataka.

Obeležje tipa objekta bolje je predstaviti kao poseban tip objekta nego kao obeležje ako je:

- samo obeležje ima neko posebno značenje u realnom sistemu,

- obeležje u osnovi identificuje drugi tip objekta (šifra predmeta ne treba da bude obeležje nastavnika koji taj predmet predaje, već treba formirati poseban tip objekta),
  - obeležje posmatranog tipa objekta je istovremeno i obeležje drugih tipova objekata,
  - obeležje tipa objekta je više značno tj. jednom pojavljivanju tipa objekta odgovaraju više vrednosti datog obeležja. Na primer za tip objekta student obeležje iznos kredita je više značno i treba ga predstaviti posebnim tipom objekta.
- (8) *Pristup od vrha na niže:* Definišu se neki opšti tipovi objekata, pa se zatim postepeno definišu podtipovi ovih objekata. Na primer, svaki poslovni sistem se sastoji od **OBJEKATA poslovanja**, **SUBJEKATA poslovanja**, **PARTNERA** u poslovanju, **OBAVEZA** u poslovanju i **TRANSAKCIJA** poslovanja. **OBJEKTI** mogu da budu **PROIZVOD** i **USLUGE**, **SUBJEKTI** organizaciona šema, **PARTNERI** su **KUPCI**, **DOBAVLJAČI**. **OBAVEZE** su **UGOVORI**, **PLANNOVI** itd.).
- (9) *Analiza opisa sistema na prirodnom jeziku:* Osnovne naznake ove analize su ranije iznete.

#### 7.4.2. Određivanje tipova veza

Svaka aktivnost, proces, odnosno zadatak u realnom sistemu predstavlja neki tip veze. Vezu definisemo tako da prvo utvrdimo koji se objekti iz posmatranog skupa objekata nalaze u vezi. Time je određen i red veze. Dalje za svaki od objekata koji učestvuje u vezi odredjujemo kardinalnost preslikavanja prema drugim objektima u istoj vezi. Na kraju, s obzirom da i veza može imati svojstva odredjujemo eventualna obeležja veze, kada vezu predstavljamo mešovitim tipom objekt-veza.

Jedan od načina određivanja koji objekti iz posmatranog skupa se nalaze u vezi je razmatranje svih parova objekata. Za svaki par objekata potrebno je izvršiti istraživanje koje daje odgovor na sledeća pitanja:

- da li se posmatrani objekti mogu koristiti u jednoj istoj operaciji nad bazom podataka?,
- da li se može zadati informacioni zahtev koji sadrži oba objekta?

Ako postoji pozitivan odgovor smatra se da objekti mogu biti u vezi, odnosno, ako ne postoji pozitivan odgovor da posmatrani objekti nisu u vezi. Zatim se određuje koje veze su najvažnije, a koje su redundantne.

Osnovni problemi koji se javljaju u procesu određivanja veza jesu:

- određivanje pravog reda veze,
- izbegavanje redundantnih veza.

Vezu smatramo redundantnom ako znanje o odnosima u sistemu koje je tom vezom iskazano, sledi iz preostalih veza u DOV. U tom slučaju takvu vezu nije neophodno eksplicitno predstavljati.

O problematici prisutnosti redundantne i načinu njenog uklanjanja detaljnije će biti reči pri razmatranju problematike integracije više posebnih DOV u jedan globalni model podataka gde i postoji najveća opasnost da neka od veza bude redundantna.

S obzirom da je određivanje pravog reda veze osnovni preduslov da se prikaže stvarno stanje realnog sistema, sledi detaljnije razmatranje problema utvrđivanja reda veze. Podjimo od sledećeg ilustrativnog primera: Neka smo u modelu podataka utvrdili objekte **AMBULANTA**, **PACIJENT** i **LEKAR** i neka vrede sledeći odnosi:

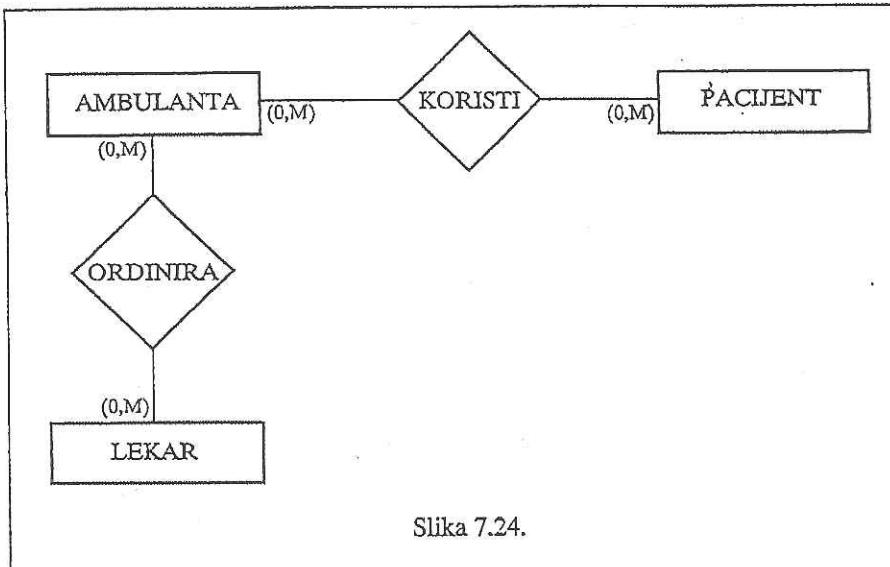
1. Jednu ambulantu može koristiti više pacijenata.
2. Jeden pacijent može koristiti usluge više ambulant.
3. U jednoj ambulanti ordinira više lekar.
4. Jeden lekar može ordinirati u više ambulant.

Prethodni odnosi i objekti mogu se iskazati DOV datim na slici 7.24. sa vezama **KORISTI** i **ORDINIRA**.

Možemo zapaziti da iskazi (1) - (4) ne preciziraju odnos **AMBULANTA - PACIJENT - LEKAR** odnosno DOV na slici 7.24. iskazuje nezavisne odnose objekata **AMBULANTA - PACIJENT** i **AMBULANTA - LEKAR**.

Iz iskaza (2) i (3) ne sledi zaključak da:

*svaki pacijent koji koristi usluge u jednoj ambulanti to radi kod svih lekara koji mogu ordinirati u toj ambulanti.*



Slika 7.24.

Napomenimo da ne sledi ni negacija prethodnog zaključka. To je zbog toga što iskazi (1) - (4) ne govore ništa o trojnoj vezi već samo o binarnim vezama. Na osnovu DOV na slici 7.24. koji istinito predstavlja odnose (1) - (4), nije moguće saznati *usluge kojih lekara u jednoj ambulanti koristi koji pacijent*.

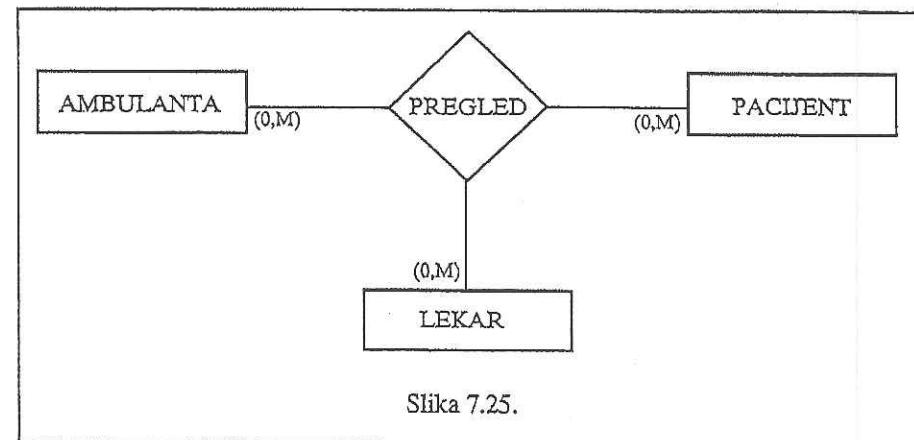
Situacija u kojoj svi pacijenti koji koriste usluge jedne ambulante ne moraju koristiti isti skup lekara koji ordiniraju u toj ambulanti može se opisati sledećim iskazima:

- 1". Usluge jednog lekara u jednoj ambulanti mogu koristiti više pacijenata.
- 2". Jeden pacijent u jednoj ambulanti može koristiti usluge više lekara.
- 3". Jeden lekar jednog pacijenta može pregledati u više ambulant.

Na slici 7.25. dat je DOV koji iskazuje odnose (1") - (3") sa trojnom vezom PREGLED koja omogućava da se izraze proizvoljni odnosi izmedju objekata AMBULANTA, PACIJENT i LEKAR, što dve binarne veze sa DOV na slici 7.24. nisu omogućavale.

Prema nekim autorima veze višeg reda u MOV nisu dozvoljene jer *zamagljuju* prirodu odnosa izmedju objekata, dovode do redundanse u bazi podataka i po pravilu identifikator takvih veza sastavljen je od više obeležja tj. od identifikatora svakog objekta u vezi, što nije u skladu sa preporukom o što

jednostavnijim identifikatorima. Obično se preporučuje da se prilikom modeliranja koriste veze proizvoljnog reda tj. prema potrebi i stanju, a da se po izradi modela izvrši zamena takvih veza binarnim vezama.

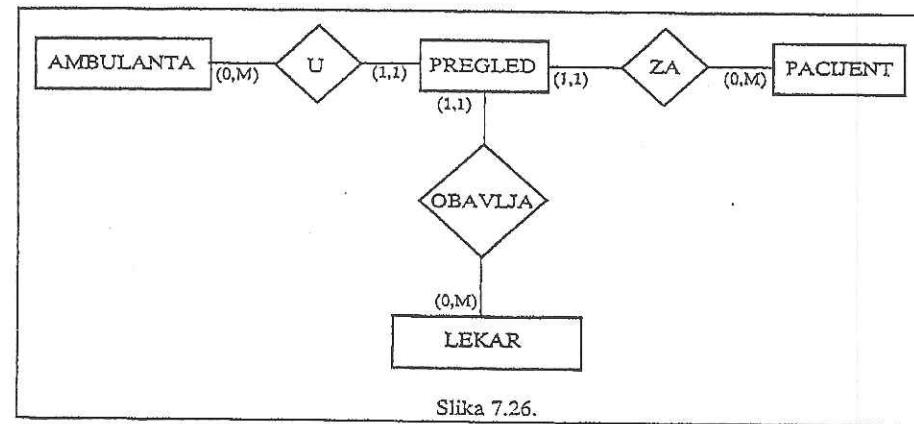


Slika 7.25.

Zamena (prevodjenje) *n-arne* veze sa *n binarnih* veza izvodi se na sledeći način:

- Vezu predstaviti kao tip objekta i dodeliti mu *novi identifikator* (tj. šifru);
- Svaki od n tipova objekata iz n-arne veze *binarno* se veže na novouvedeni tip objekta.

Trojna veza PREGLED sa DOV na slici 7.25. prevodi se u tri binarne veze kako je to dato DOV na slici 7.26.



Slika 7.26.

Prethodna analiza oblika povezanosti tipova objekata AMBULANTA, PACIJENT i LEKAR ukazuje na greške koje se mogu pojaviti prilikom definisanja reda veze. Greške slede iz činjenice da skup veza nižeg reda ne mora omogućavati logičko izvodjenje veza višeg reda. Kada u realnom sistemu postoje veze višeg reda treba ih eksplisitno predstaviti, a zatim ih zameniti sa binarnim vezama kako je prethodno opisano.

#### 7.4.3. Integriranje podmodela

Kada je završena izrada DOV po delovima tj. podmodelima pristupa se integriranju podmodela u jedan globalni (integralni) model podataka. U procesu integracije dolazi se do novih saznanja i vrlo često do potrebe izmene u pojedinim podmodelima.

Kao posledica prilagodjavanja javlja se i mogućnost da isti skup podmodela bude integriran u različite globalne modele. Ako su takvi globalni modeli dobijeni valjanim postupkom integracije smatramo ih medjusobno *ekvivalentnim*. Različitost pojedinih globalnih modela može se odraziti na efikasnost baze podataka kojim se dati model realizuje.

Globalni model podataka treba biti:

- *kompletan* (sadržavati sva znanja iz svih podmodela),
- *neredundantan* (ne sadržavati višestruke predstave istih znanja),
- *konzistentan* (ne sadržavati medjusobno protivrečna znanja).

Prethodni zahtevi na globalni model podataka mogu u nekim slučajevima biti neispunjivi. Naime, ako dva podmodela sadrže medjusobno protivrečne iskaze, onda globalni model ne može biti:

1. *konzistentan*, ako sadrži sve iskaze iz datih podmodela, ili pak
2. *kompletan*, ako izostavi neki od medjusobno protivrečnih iskaza iz tih podmodela.

U prethodnom slučaju uzrok nekonzistentnosti treba otkloniti na nivou pojedinačnih podmodela jer ako podmodeli iskazuju medjusobno protivrečne tvrdnje o sistemu, onda je bar jedna od tih tvrdnji neistinita.

Kao prvi korak u integraciji podmodela obično se izvodi *predintegracija podmodela* sa ciljem provere da li se u podmodelima javljaju:

- a) isti tipovi objekata,
- a) homonimi i sinonimi.

Dva tipa objekta su ista ako imaju iste *ekstenzije*, odnosno ako su skupovi njihovih pojava (instanci) identični. Zbog toga što je u praksi teško utvrditi ekstenzije tipova objekata to se identičnost tipova objekata utvrđuje na osnovu *intenzija* tipova objekata. Tip objekta u intenziji se predstavlja *skupom obeležja* i identifikatorom.

Dva tipa objekata identična su ako, i samo ako imaju identične intenzije.

S obzirom da u podmodelima mogu da postoje neispravnosti ili nepotpunosti, to se identičnost ili neidentičnost tipova objekata ne može sa sigurnošću utvrditi na osnovu njihovih intenzija. Pri integraciji podmodela treba porediti "slične" tipove objekata uz ponovnu analizu njihovih značenja i uloga u realnom sistemu.

Identični tipovi objekata iz različitih podmodela predstavljaju se jednim tipom objekta u globalnom modelu podataka.

*Sinonimima* nazivamo različita imena za iste tipove objekata. Na primer, ime RADNIK u jednom podmodelu i ime ZAPOSLEN u drugom podmodelu mogu označavati isti tip objekta. Pojava sinonima česta je i kod imenovanja obeležja. Na primer, ista svojstva istih tipova obeležja mogu u različitim podmodelima biti nazvana različitim imenima.

*Homonimima* nazivamo ona imena koja su medjusobno ista, ali označavaju različite tipove objekata. Pojava homonima javlja se pre svega između različitih podmodela. Na primer, naziv VOZILO može u jednom podmodelu označavati putničke automobile, dok u drugom označavati teretna vozila za prevoz robe. Prepoznavanje da se radi o homonimima oslanja se na različite uloge tih objekata u realnom sistemu iz čega sledi da i pripadni skupovi obeležja tih istoimenih tipova objekata treba da budu različiti.

Možemo uočiti da je otkrivanje sinonima i homonima samo drugi oblik odredjivanja jednakosti ili nejednakosti tipova objekata.

U procesu predintegracije, tipovi objekata treba da budu preimenovani tako da se svi sinonimi zamene jednim imenom, dok se svaki od homonima zamjenjuje jednim novim i različitim imenom, pri čemu jedan od homonima može zadržati prvočitno ime.

Posle izvršene predintegracije, odnosno usklajivanja imena pristupamo integraciji podmodela.

Integracija podmodela zasniva se na sledeće tri osnovne koncepcije:

- klasifikacija,
- generalizacija,
- agregacija.

Prethodni pojmovi predstavljaju i načine apstrakcije podataka razmatrane u poglavlju o modelima podataka i u kontekstu integracije podmodela imaju ista ranije opisana značenja. Pri integraciji podmodela navedene koncepcije se mogu proizvoljno kombinovati.

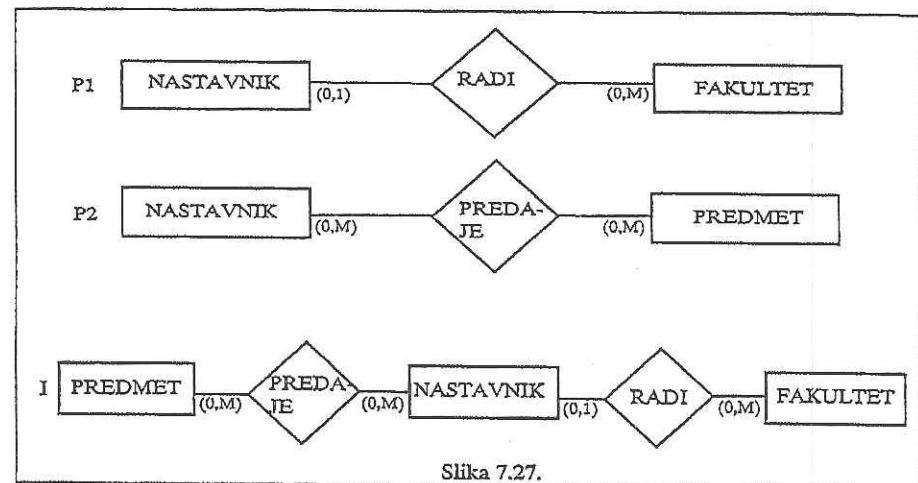
Nizom primera, u nastavku ilustrujemo principe integracije. Sa Pi su označeni podmodeli koji se integrišu, dok je rezultat integracije označen sa I. Zbog jednostavnosti ilustracija na DOV nisu predstavljana obeležja tipova objekata, mada ne možemo reći da su ta obeležja bez značaja za integraciju.

Prepostavimo da je u procesu predintegracije utvrđeno da su tipovi objekata NASTAVNIK iz podmodela P1 i P2 sa slike 7.27. identični. Napomenimo da isti nazivi tipova objekata u podmodelima nisu dovoljan uslov da se ti objekti proglaše identičnim. Rezultat integracije P1 i P2 dat je DOV označenim sa I na slici 7.27.

Na slici 7.28. prikazana je integracija podmodela koji sadrže objekte koji se nalaze u generalizacijskom odnosu sa neekskluzivnom specijalizacijom.

Vezu RUKOVODI preneli smo sa tipa objekta RADNIK iz podmodela P2, na tip objekta RUKOVODILAC iz podmodela P3. To znači da svaki od radnika koji rukovodi treba ujedno biti i pojava tipa objekta RUKOVODILAC. Tipovi objekata RUKOVODILAC i NASTAVNIK imaju generički tip objekta RADNIK uz pretpostavku da imaju isti identifikator (npr. JMBG) i sva obeležja tipa objekta RADNIK sa neekskluzivnom specijalizacijom. Jedan radnik može, ali ne mora biti istovremeno rukovodilac i/ili nastavnik. Do ovog

zaključka naravno možemo doći na osnovu semantike posmatranog realnog sistema.



Slika 7.27.

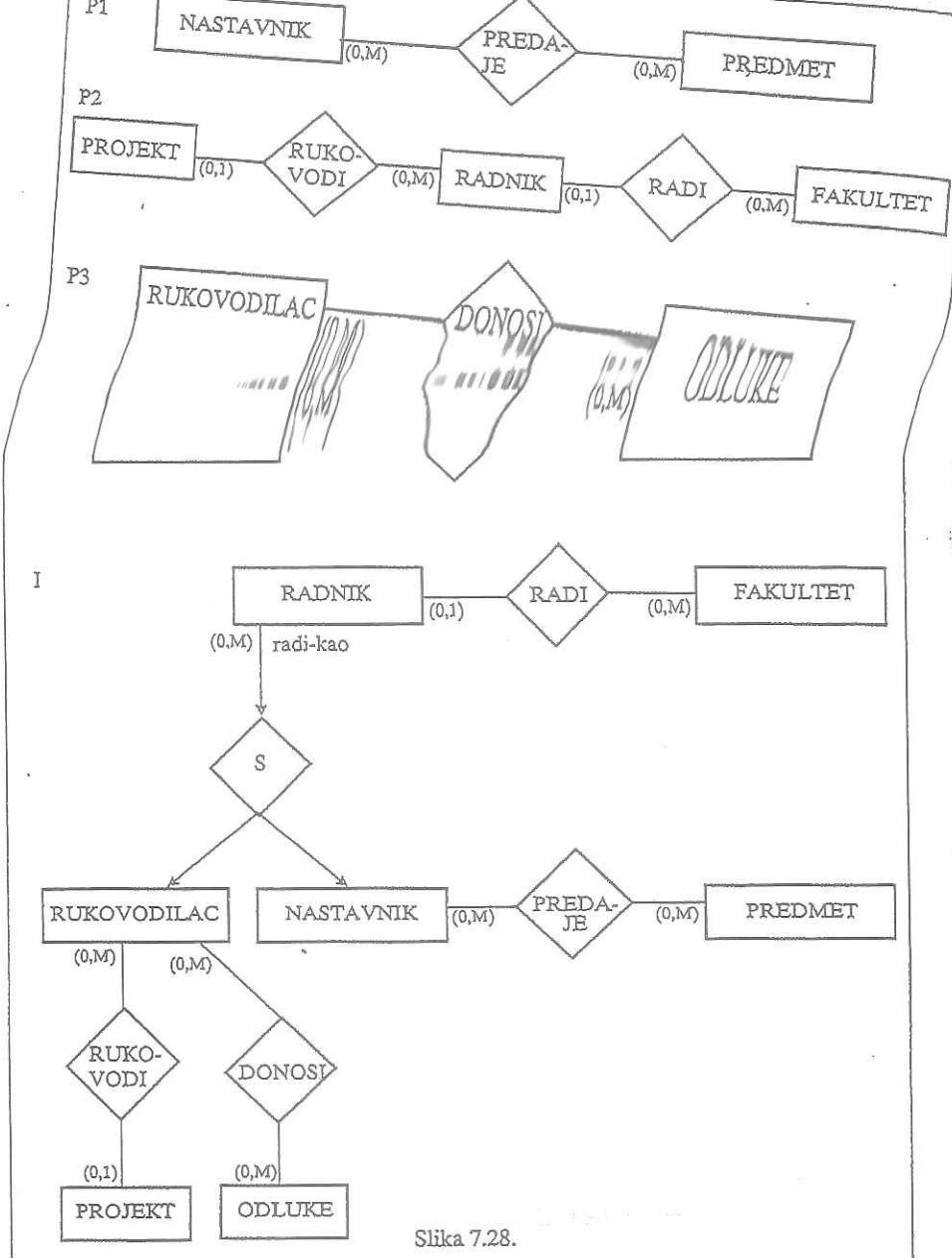
Na slici 7.29. prikazana je integracija podmodela koji sadrže tipove objekata NASTAVNIK i OP\_OSOBLJE (operativno osoblje: službenici, održavanje) koji se nalaze u generalizacijskom odnosu sa ekskluzivnom specijalizacijom. Pretpostavka je da tipovi objekata NASTAVNIK i OP\_OSOBLJE imaju isti identifikator (npr. JMBG), veliki broj zajedničkih obeležja (npr. IME, DATUM\_RODJ, DATUM\_ZAP, MESTO\_ST, ULICA\_ST, PLATA). Novouvedeni tip objekta RADNIK ima sva zajednička obeležja tipova objekata NASTAVNIK i OP\_OSOBLJE, dok tipovi objekata NASTAVNIK i OP\_OSOBLJE imaju samo njima svojstvena obeležja (npr. NASTAVNIK: ZVANJE, DATUM\_IZB; OP\_OSOBLJE: ZANIMANJE)

Možemo uočiti i razliku u odnosu na primer sa slike 7.28. gde rukovodilac može pripadati nastavnicima i obrnuto, dok ovde nastavnik ne može pripadati operativnom osoblju i obrnuto. Obeležje VRSTA\_POSLA koje pripada tipu objekta RADNIK je obeležje po kojem se vrši specijalizacija.

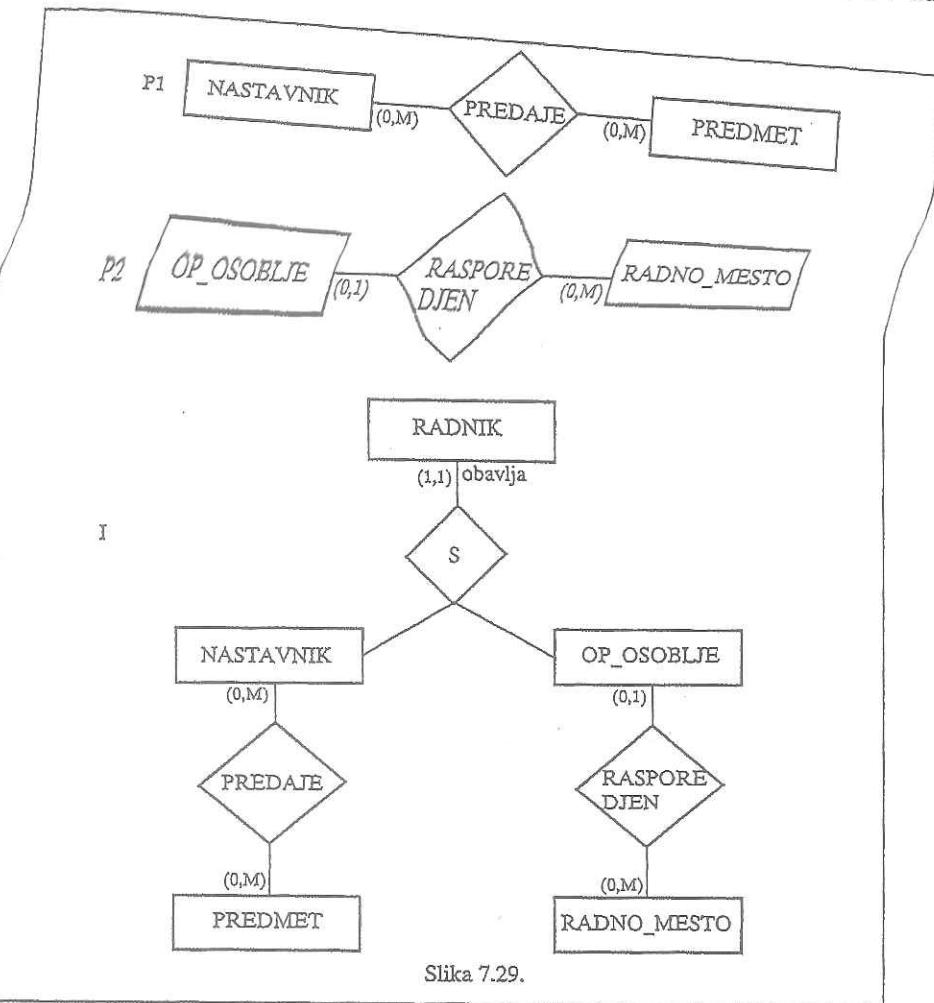
Mogućnosti i načini integracije veza zavise od njihovih osobina i to:

- reda veze,
- značenja/uloge veze,
- kardinalnosti preslikavanja objekata u vezi.

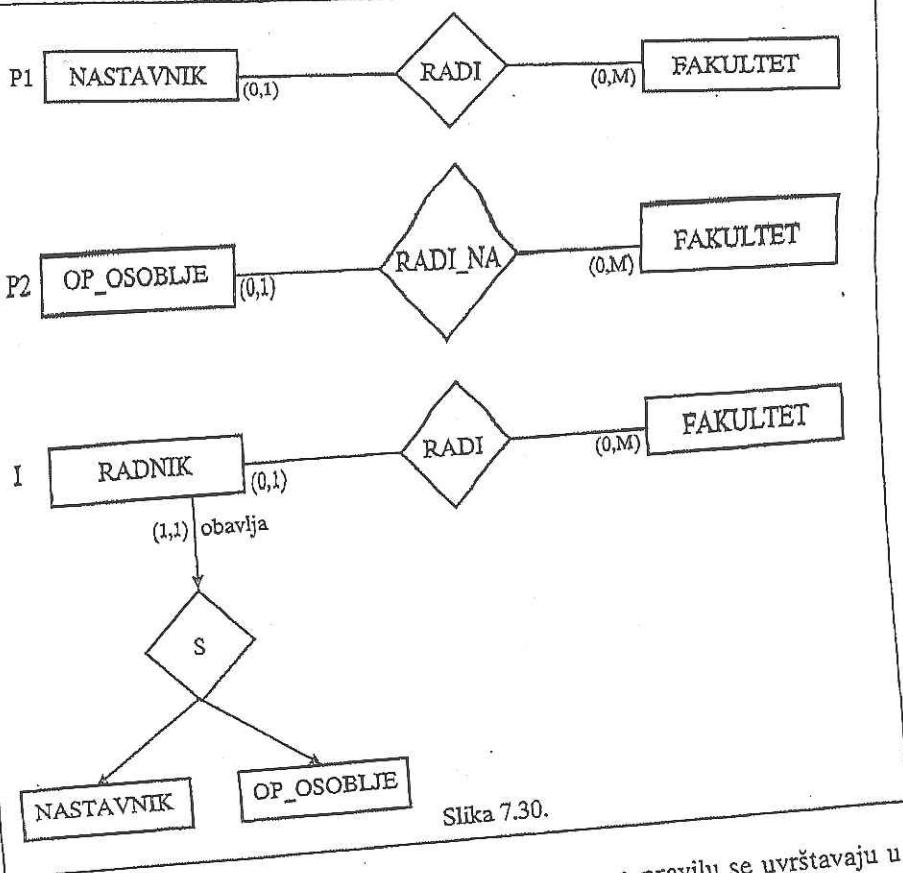
Uvođenje - veze - obeležja  
mogu biti predstavljene jednom vezom RADI, kako je to učinjeno u DOV na slici 7.30. označenim sa I.



Na slici 7.30. ilustrovana je integracija binarne veze RADI i RADI\_NA koje imaju isto značenje/ulogu u podmodelima P1 i P2. Uvodjenjem generativnog tipa objekta RADNIK (uz objašnjenja data u prethodnom primeru) te veze



Da su kardinalnosti povezanosti objekata (1:M) u vezama RADI i RADI\_NA sa slike 7.30. bile različite, ne bi bilo moguće izvršiti integraciju tih veza onako kako je to učinjeno na DOV I, bez gubitka značenja jedne od tih veza. Na slici 7.31. prikazano je na koji način možemo izvršiti integraciju podmodela u tom slučaju.



Slika 7.30.

Veze istog reda koje imaju različita značenja/uloge po pravilu se uvrštavaju u globalni model neizmenjene. Na slici 7.32. dat je primer integracije takve dve veze.

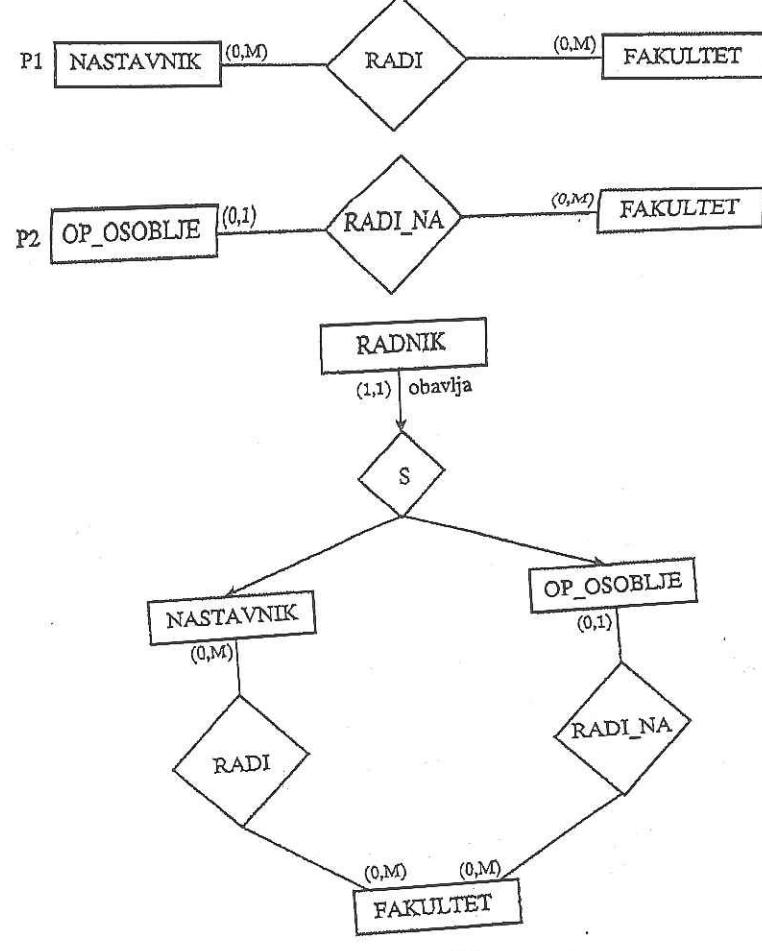
Veza koja tranzitivno sledi iz drugih veza sadržanih u modelu podataka je redundantna. Na slici 7.33. veza ZAPOSLEN ne postoji u integriranom modelu I jer tranzitivno sledi iz veza PRIPADA i U\_SASTAVU.

Redundantnost veze ZAPOSLEN možemo objasniti sledećim iskazima:

1. Radnik PRIPADA tačno jednoj katedri.
2. Katedra se nalazi U\_SASTAVU tačno jednog fakulteta.

Sledi:

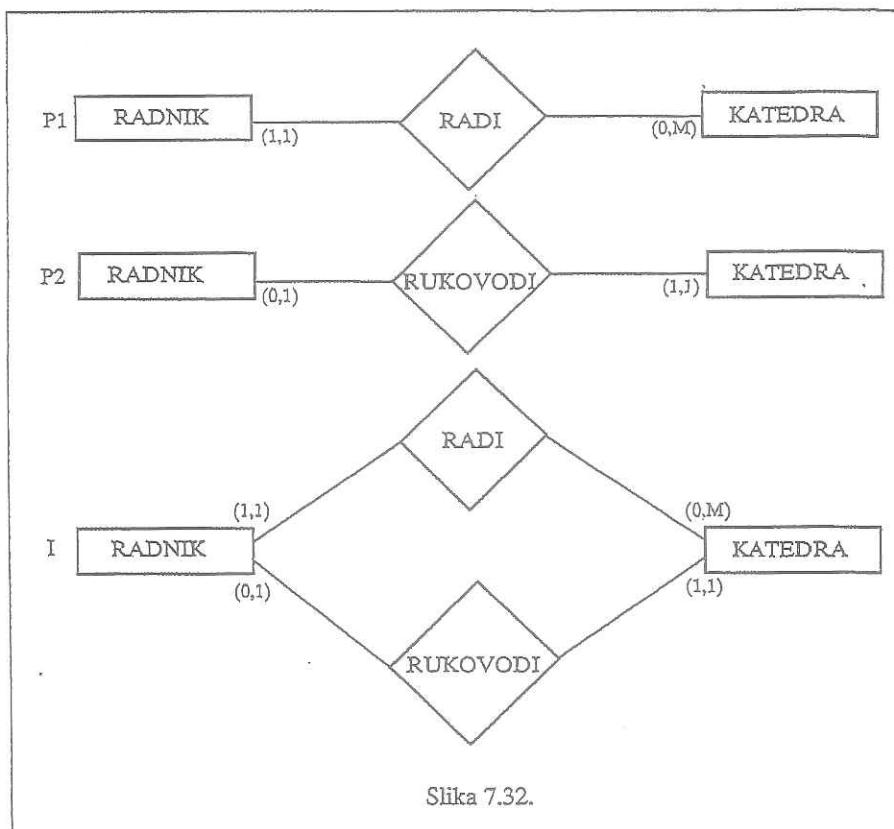
3. Ako je poznata pripadnost radnika katedri, i katedre fakultetu, onda je poznata i pripadnost radnika fakultetu. Zbog toga je veza ZAPOSLEN koja to eksplisitno iskazuje redundantna.



Slika 7.31.

U prethodnim primerima radi jednostavnosti ilustracija integracije podmodela posmatrano je do tri podmodela. U realnim situacijama susrećemo se sa potrebom integrisanja većeg broja podmodела. Postavlja se pitanje kako pristupiti integraciji podmodела u tim situacijama?

Jedno od rešenja je istovremeno integrisanje svih podmodела (u jednom koraku). Iako je u principu takav pristup moguć on ima niz nedostataka: zahteva dugotrajne provere, česta pojava grešaka i propusta, neefikasnost.

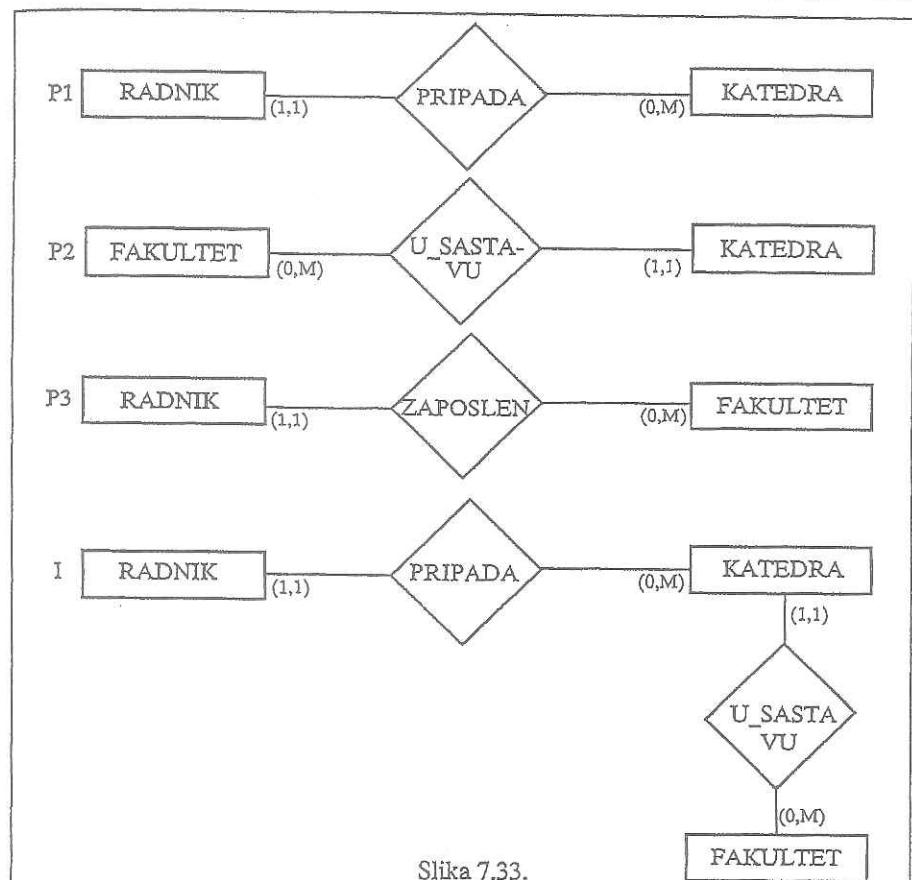


Slika 7.32.

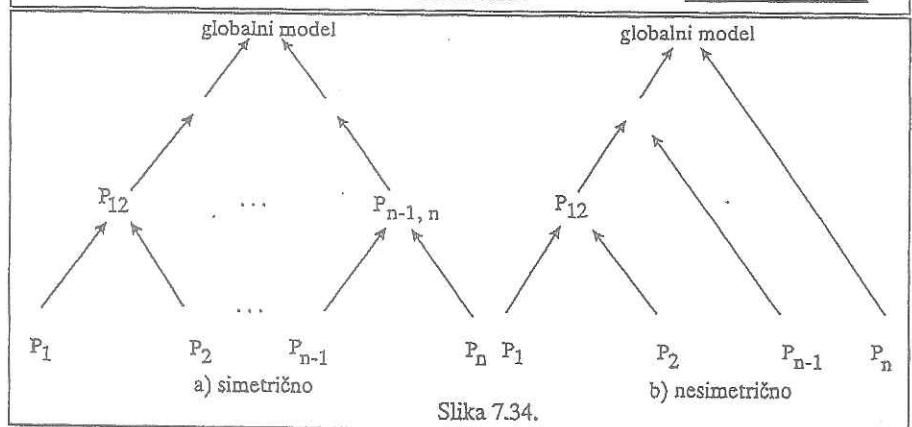
Kao drugo prihvatljivije rešenje je integrisanje u jednom koraku dva podmodela odnosno binarno integriranje koje može biti simetrično ili nesimetrično kako je to ilustrovano na slici 7.34.

Prema simetričnom integriranju prvo se integrišu dva po dva polazna podmodela. Zatim se integrišu dva po dva tako dobijena podmodela. Postupak se ponavlja do konačnog globalnog modela.

Prema nesimetričnom integriranju prvo se integrišu dva polazna podmodela. Zatim se dobijeni integrirani podmodel integriše sa jednim polaznim podmodelom. Prethodni korak se ponavlja dok se integracijom ne obuhvate svi polazni podmodeli čime se dobija i globalni model podataka.



Slika 7.33.



Slika 7.34.

## *8. RELACIONI MODEL PODATAKA*

Koncepcija relacionog modela podataka je prvi put objavljena u Codd-ovom članku iz 1970 godine. Osnovni razlozi za definisanje relacionog modela podataka bili su sledeći nedostaci uočeni u korišćenju tada poznatih modela podataka:

- nepostojanje jasne granice izmedju logičkih i fizičkih aspekata baza podataka,
- strukturna kompleksnost podataka,
- navigacioni jezici za manipulisanje podacima,

Ciljevi razvoja relacionog modela podataka odnosili su se pre svega na uklanjanje prethodnih nedostataka.

Nakon serije Codd-ovih radova započeo je razvoj relationalnih modela i relationalnih baza podataka. No, i u relationalnom modelu podataka ubrzo je uočen niz nedostataka, tako da je Codd 1979 objavio prošireni relacioni model podataka. Danas se kao dve vrste relationalnog modela podataka susreću: *klasični relacioni model* (RM) i *proširen relacioni model* (RM/T). Prema ranije učinjenoj klasifikaciji modela podataka može se videti da klasični relacioni model pripada II, a proširen relacioni model III generaciji. RM/T model u odnosu na RM model sadrži dodatne koncepte strukture, pravila integriteta i moćne operatore. No i pored toga do danas ovaj model ima manji praktični značaj. Neki koncepti ovog modela prihvaćeni su i preuzeti u drugim modelima podataka i implementirani u nekim SUBP. Iz navedenih razloga ovde će biti razmatran samo klasični relacioni model podataka.

Osnovni pojam relationalnog modela podataka je *relacija*. Relacija se može posmatrati sa sledeća dva aspekta: *značenje i sadržaj*. Značenje relacije naziva se *intenzijom*, i formalno se iskazuje *šemom relacije*. Sadržaj relacije naziva se *ekstenzijom*, a iskazuje se *tabelom podataka* ili *pojavom šeme relacije*.

Razmotrimo radi objašnjenja prvo matematičko shvatanje pojma relacije polazeći od sledeće dve definicije:

1. Dekartov proizvod. Neka su  $D_1, D_2, \dots, D_n$  proizvoljni konačni skupovi. Dekartov proizvod tih skupova u oznaci  $D_1 \times D_2 \times \dots \times D_n$  definišemo kao skup uredjenih n-torki oblika  $\langle d_1, d_2, \dots, d_n \rangle$ , gde je  $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$ . Na primer, neka je  $D_1 = \{a_1, a_2\}$  i  $D_2 = \{b_1, b_2, b_3\}$ . Dekartov proizvod  $D_1 \times D_2$  glasi  $D_1 \times D_2 = \{a_1b_1, a_1b_2, a_1b_3, a_2b_1, a_2b_2, a_2b_3\}$ .

2. Relacija  $r$  prema matematičkom shvatanju na skupovima  $D_1, D_2, \dots, D_n$  naziva se podskup dekartovog proizvoda  $D_1 \times D_2 \times \dots \times D_n$  odnosno:

$$r \subseteq D_1 \times D_2 \times \dots \times D_n.$$

U relacionom modelu podataka relacija odgovara dvodimenzionalnoj tabeli u kojoj svaki red sadrži jednu n-torku, a svaka kolona elemente jednog domena (odnosno vrednosti obeležja zadatog na toj domeni). Ako svakoj koloni pridružimo naziv obeležja, dobit ćemo zaglavje ili šemu relacije slika 8.1.

Zaglavje ili šema relacije	(BROJ_IND	IME	FAKULTET)
Telo relacije	00010	Jovan	Pravni
n-torka	00050	Ivana	Ekonomski
	00020	Svetlana	Elektrotehnički

kolona

Slika 8.1.

Svaka kolona relacije jednoznačno je određena nazivom obeležja u šemi relacije. Redosled kolona u relaciji postaje zahvaljujući tome nebitan. Takođe i redosled članova u n-torci postaje nebitan. Navedeno svojstvo relacije u relacionom modelu nije u skladu sa definicijom relacije u matematičkom smislu (preko Dekartovog proizvoda).

### Relaciona šema

Šema relacije, u oznaci  $N(R, F)$ , je u opštem slučaju određena svojim nazivom  $N$ , skupom naziva obeležja  $R = \{A_1, A_2, \dots, A_n\}$  i skupom ograničenja  $F = \{o_1, o_2, \dots, o_m\}$ .

Skup ograničenja može biti zadat skupom *funkcionalnih zavisnosti* koje opisuju odnose između elemenata domena skupova  $X$  i  $Y$  koji su podskupovi skupa obeležja  $R$  ( $X \subseteq R$  i  $Y \subseteq R$ ).

Iskaz oblika  $X \rightarrow Y$  naziva se funkcionalnom zavisnošću (čitamo  $X$  funkcionalno određuje  $Y$  ili  $Y$  funkcionalno zavisi od  $X$ ) ako niti jedan element skupa  $X$  ne može biti preslikan na više od jednog elementa skupa  $Y$ .

Funkcionalnu zavisnost (FZ) oblika  $X \rightarrow Y$  nazivamo *potpunom FZ* ako ne postoji skup  $X'$ ,  $X' \subset X$ , za koji vredi  $X' \rightarrow Y$ . Tada kažemo da  $Y$  *potpuno* zavisi od  $X$ . Funkcionalnu zavisnost koja nije potpuna nazivamo *parcijalnom*.

Smatramo da vredi  $X \rightarrow Y$ , ako u bilo kojoj relaciji  $r$  zadatoj na  $R$ , za bilo koje dve n-torce  $t_1$  i  $t_2$ , za koje vredi  $t_1[X] = t_2[X]$ , vredi  $t_1[Y] = t_2[Y]$ .

Sa  $t$  smo označili n-torku relacije. Vrednost na koju je preslikano obeležje  $A_i$  u n-torci označavamo sa  $t[A_i]$ , odnosno vrednost na koju je preslikan neki podskup obeležja  $X$  sa  $t[X]$ .

Da li u nekoj relaciji  $r$ , u određenom momentu vredi  $X \rightarrow Y$ , možemo jednostavno proveriti. S obzirom da relacija nije statički objekat već se vremenom menja, istinitost tvrdnje  $X \rightarrow Y$  u jednoj relaciji ne znači automatski i istinitost u svim relacijama, odnosno u relacionoj šemi. Na žalost, ne postoji postupak kojim bi smo, na osnovu stanja u pojednim relacijama, mogli ustanoviti da neka funkcionalna zavisnost vredi u relacionoj šemi. Jedini način da ustanovimo koje funkcionalne zavisnosti vrede u određenoj relationalnoj šemi jeste da ispitamo semantičke karakteristike obeležja od kojih se relaciona šema sastoji.

Razmotrimo na primer, tipove objekata nastavnik, predmet i student između kojih postoji sledeći odnosi:

- svaki nastavnik predaje samo jedan predmet (isti predmet može predavati više nastavnika),
- svaki student sluša određeni predmet kod samo jednog nastavnika.

Šema relacije, koja predstavlja model opisanih odnosa iz realnog sistema, može imati sledeći izgled:

$NPS(S\_NAS, S\_PRED, BROJ\_IND,$   
 $S\_NAS \rightarrow S\_PRED, S\_PRED \text{ BROJ\_IND} \rightarrow S\_NAS)$

sa sledećim značenjem naziva obeležja:

- S\_NAS - šifra nastavnika,
- S\_PRED - šifra predmeta,
- BROJ\_IND - broj indeksa studenta.

S obzirom da je svaka n-torka relacije jedinstvena to ujedno znači da postoji jedinstven identifikator istih. Taj identifikator naziva se **kandidatom ključa**. Jedna šema relacije može imati više kandidata ključeva (ekvivalentni ključevi) od kojih se jedan bira za **primarni ključ**.

Neka je R skup obeležja šeme relacije. Kandidatom za ključ naziva se skup obeležja X,  $X \subseteq R$  ako i samo ako zadovoljava sledeće uslove:

#### 1. Jedinstvenost

Ni u jednoj ekstenziji relacije R ne postoje dve n-torce koje imaju jednakе vrednosti svih obeležja iz X.

#### 2. Minimalnost

Ne postoji takav skup X',  $X' \subset X$ , koji bi ispunjavao uslov jedinstvenosti.

Uslov minimalnosti ima za cilj da broj obeležja u ključu svede na minimum. Naime, svaki skup obeležja koji u sebi sadrži minimalni ključ jednoznačno određuje n-torku relacije. Podskup obeležja u relacionoj šemi R, koji zadovoljava uslov jednoznačnosti, a ne zadovoljava uslov minimalnosti, nazivamo **super ključ** relacije zadat na R.

Obeležja koja ulaze u sastav bilo kojeg kandidata ključa nazivaju se **ključna obeležja**. Obeležja koja ne ulaze niti u jedan od kandidata ključa nazivaju se **neklijučna obeležja**.

U relaciji uvek postoji ključ. U krajnjem slučaju sva obeležja relacione šeme na kojoj je relacija zadata čine ključ. Obeležja koja pripadaju primarnom ključu u oznaci šeme relacije ćemo podvlačiti.

Ako je R skup obeležja šeme relacije, a X jedan ključ te šeme relacije, tada važi funkcionalna zavisnost  $X \rightarrow R$ . Funkcionalna zavisnost  $X \rightarrow R$  se naziva zavisnost ključa. Svaki ključ ukazuje na jednu funkcionalnu zavisnost, ali obrnuto, da svaka funkcionalna zavisnost ukazuje na jedan ključ ne mora vredeti. U praksi se često za označavanje šeme relacije koristi oblik N(R) pri čemu se obeležja koja pripadaju primarnom ključu podvlače. Jasno je da se u

takvoj oznaci šeme relacije u odnosu na N(R,F) mogu izgubiti neke informacije o realnom svetu.

Ilustrujmo sada primerom aspekt značenja i sadržaja relacije. Neka relacija ISPIT ima sledeće značenje: Studenti polažu ispite određenog datuma sa uspehom iskazanim ocenom ispita.

Značenje relacije ISPIT predstavlja se u relacionom modelu podataka sledećom šemom relacije:

ISPIT (BROJ\_IND, S\_PRED, DATUM\_ISPITA, OCENA).

Očigledno je da šema relacije ne predstavlja podatke, već njihovu formu, a *interpretacijom* i značenje. Napomenimo da značenje relacije ne proizilazi iz same šeme, jer se šema u principu može interpretirati na bezbroj načina. Stoga je bolje reći da je šema formalna predstava značenja.

Pojava šeme relacije ili ekstenzija relacije je *skup*, čiji je sadržaj iskazan tabelom. S obzirom na činjenicu da se svaki element u skupu javlja samo jednom, tabela ne sme sadržavati dva identična reda. Nadalje, obzirom na činjenicu da elementi skupa u opštem slučaju nisu uređeni, redosled javljanja redova u tabeli je nevažan. To znači, ako se ekstenzije dve relacije razlikuju samo po redosledu javljanja redova u tabeli, tada te dve relacije imaju identične ekstenzije. Takodje, ako se ekstenzije dve relacije razlikuju samo po redosledu javljanja kolona u tabeli, tada te dve relacije imaju identične ekstenzije.

Pojava šeme relacije ISPIT odredjena je skupom zapisa o pojedinačnim ispitimima. Primer pojave šeme relacije ISPIT dat je na slici 8.2.

ispit

BROJ_IND	S_PRED	DATUM_ISPITA	OCENA
I10	P1	020293	8
I20	P1	020293	6
I20	P2	050293	10
I30	P1	200693	6
I40	P2	220693	9

Slika 8.2.

Za šemu relacije, odnosno njenu značenje možemo smatrati da se ne menja tokom vremena. S druge strane, sadržaj tabele menja se tokom vremena. Na primer, svaki novo-položeni ispit menja sadržaj tabele date na slici 8.2. tako da se u nju upisuju novi redovi.

U relacionom modelu podataka domeni obeležja su skupovi *atomarnih vrednosti*. Atomarnim vrednostima nazivamo one vrednosti koje nije moguće rastaviti, a da time ne bude uništeno njihovo značenje. Na primer, ako se broj indeksa studenta rastavi na skup znakova tada broj indeksa postaje izgubljen.

Razmotrimo sada slučaj da u tabelu na slici 8.2. treba da upišemo informaciju o tome da je student sa brojem indeksa I50 položio ispite iz predmeta P1, P2 i P3 dana 010793 svaki sa ocenom 6.

Kao rešenje postavljenog zadatka razmotrimo dobijenu tabelu na slici 8.3. u kojoj su sve informacije zapisane u jednom redu. Uočavamo da obeležje S\_PRED ima vrednosti {P1, P2, P3} što prema iskazanom ograničenju o domenima obeležja nije dopušteno. Međutim, informacije o položenim ispitima studenta sa brojem indeksa I50 mogu se upisati kako je to ilustrovano tabelom na slici 8.4.

Za relaciju čije sve domene sadrže samo atomarne vrednosti kažemo da je *normalizovana*, odnosno da se nalazi u *prvoj normalnoj formi*. Iz rečenog sledi da se svaka relacija iz relacionog modela podataka nalazi u prvoj normalnoj formi. Drugim rečima, prva normalna forma relacije je *kriterij prihvatljivosti oblika* tabele u relacionom modelu podataka. O postupku normalizacije šeme relacije i normalnim formama biće još reči kasnije.

ispit

BROJ IND	S_PRED	DATUM ISPITA	OCENA
I10	P1	020293	8
I20	P1	020293	6
I20	P2	050293	10
I30	P1	200693	6
I40	P2	220693	9
I50	P1	010793	6
I50	P2	010793	6
I50	P3	010793	6

Slika 8.3.

ispit

BROJ IND	S_PRED	DATUM ISPITA	OCENA
I10	P1	020293	8
I20	P1	020293	6
I20	P2	050293	10
I30	P1	200693	6
I40	P2	220693	9
I50	P1	010793	6
I50	P2	010793	6
I50	P3	010793	6

Slika 8.4.

Obeležja koja su sastavljena od niza vrednosti možemo smatrati atomarnim obeležjima ukoliko predstavljaju nerazdvojivu celinu. Na primer, obeležje DATUM sastoji se od DAN, MESEC, GODINA. Obeležje DATUM može se zameniti sa tri obeležja DAN, MESEC i GODINA, ali to nije nužno učiniti zato što svaki pojedinačni datum predstavlja nerazdvojivu celinu. Takodje se može uočiti da datum nije moguće izraziti sa više nezavisnih redova, kako je to učinjeno sa skupom {P1, P2, P3}.

*Šema relacione baze podataka* može se predstaviti kao  $S(N, I)$  gde je:

- S - naziv baze podataka,
- N - skup šeme relacija,
- I - skup uslova integriteta baze podataka.

Šeme relacija opisuju se na jedan od ranije opisanih načina. Skup uslova integriteta baze podataka odnosi se na ograničenja na bazu podataka kao celinu odnosno njima se iskazuju ograničenja koja na primer postoje između šeme relacija. Takav je na primer, referencijski integritet.

Jedna pojava šeme baze podataka nad šemom S predstavlja skup pojava šema relacija koje čine šemu baze podataka uz uvažavanje uslova integriteta koji se definišu za bazu podataka.

Na primer, neka imamo sledeći skup šeme relacija:

NASTAVNIK (S\_NAS, PREZIME\_IME, ZVANJE, S\_DIR, DATZAP,

PLATA, DODATAK),

PREDMET (S\_PRED, NAZIV),

PREDAJE (S\_NAS, S\_PRED, BR\_GR).

Ako se prethodni skup šema relacija dopuni opisom uslova integriteta koji se odnose na ograničenja između pojave šema relacija dobijamo šemu baze podataka.

Prethodni skup šema relacija opisuje objekte NASTAVNIK, PREDMET, i njihov međusobni odnos iskazan šemom relacije PREDAJE. Ograničenja u šemama relacija iskazana su primarnim ključem koji čine podvučena obeležja.

Obeležja objekta NASTAVNIK su:

- |             |                                     |
|-------------|-------------------------------------|
| S_NAS       | - šifra nastavnika,                 |
| PREZIME_IME | - prezime i ime nastavnika,         |
| ZVANJE      | - zvanje nastavnika,                |
| S_DIR       | - šifra prepostavljenog,            |
| DATZAP      | - datum prvog zaposlenja,           |
| PLATA       | - plata nastavnika,                 |
| DODATAK     | - novčani iznos dodatka nastavnika. |

Obeležja objekta PREDMET su:

- |        |                   |
|--------|-------------------|
| S_PRED | - šifra predmeta, |
| NAZIV  | - naziv predmeta. |

Obeležja objekta PREDAJE su:

- |        |   |
|--------|---|
| S_NAS  | - šifra nastavnika,   |
| S_PRED | - šifra predmeta,   |
| BR_GR  | - broj grupa predavanja jednog nastavnika na jednom predmetu. |

Primećujemo da se ključevi šema relacija NASTAVNIK i PREDMET javljaju kao obeležja šeme relacije PREDAJE. Takva obeležja nazivamo *spoljnim ključevima* (strani ključ) šeme relacije PREDAJE. Očigledno, spoljni ključ omogućava da se iz date n-torke jedne relacije pristupi tačno jednoj n-torci druge relacije. Na primer, iz date n-torke relacije PREDAJE, spoljnim ključem S\_NAS moguće je jednoznačan pristup do one n-torke u relaciji NASTAVNIK u kojoj su sadržani podaci o određenom nastavniku. Spoljni ključevi mogu ujedno biti elementi ključa. Kao što se iz primera vidi ključ šeme relacije PREDAJE sastoji se od spoljnih ključeva S\_NAS i S\_PRED.

Primer pojave šema relacija u nekom trenutku vremena može da izgleda:

### nastavnik

S_NAS	PREZIME_IME	ZVANJE	S_DIR	DATZAP	PLATA
002	ILIĆ PETAR	DOCENT	001	01-JAN-70	11500
001	SIMIĆ SIMA	PROFESOR		02-JUN-67	13500
010	NADA IVIĆ	ASISTENT	002	03-AUG-91	8200

### predmet

S_PRED	NAZIV
001	INFORMACIONI SISTEMI
002	OSNOVI RAČUNARSTVA
003	STRUKTURE I BP

### predaje

S_NAS	S_PRED	BR_GR
001	002	2
002	002	1
001	003	1

Šema relacije ne sme sadržavati dva jednakaka obeležja. Različite šeme relacija mogu sadržavati ista obeležja. Prilikom izvodjenja nekih operacija nad relacijama potrebno je da ime svakog obeležja bude jednoznačno identifikovano. S obzirom da je ime relacione šeme u modelu podataka jedinstveno, a ime obeležja unutar šeme relacije isto tako, time je i svako obeležje iz modela podataka jednoznačno identifikovano. Na primer, obeležje S\_PRED relacije PREDMET jednoznačno se identificuje sa

### PREDMET.S\_PRED.

Kažemo da smo obeležje S\_PRED *kvalifikovali*. Kvalifikaciju obeležja treba uvek vršiti ako u više šeme relacija postoje isti nazivi obeležja.

## 8.1. Operacijska komponenta

Operacijska komponenta omogućava manipulisanje podacima u bazi podataka, što se pre svega odnosi na postavljanje upita i ažuriranje baze podataka.

Smisao operacija nad relacionom bazom podataka sastoji se u promeni pojave šeme baze podataka (promena pojava postojećih šema relacija) ili u formiraju novih relacija, ukoliko se rezultat upita na bazu podataka posmatra kao formiranje nove relacije koja zadovoljava postavljene uslove.

Za relationalni model podataka postoji više poznatih i opšteprihvaćenih relationalnih jezika. Svi postojeći jezici za relacione baze podataka razvijeni su na osnovu jednog od sledećih operatora:

- relacione algebre,
- relationalnog računa.

Relaciona algebra je model proceduralnog jezika i sastoji se od skupa operatora za rad sa relacijama, odnosno skupa odgovarajućih operacija definisanih tim operatorima. Kombinovanjem operacija moguće je izvršiti pretraživanje, odnosno ažuriranje baze podataka.

Relacioni račun zasnovan je na računu predikata prvog reda i spada u neproceduralne jezike. Pomoću relationalnog računa definiše se traženi rezultat, a SUBP se prepusta da dodje do tog rezultata.

Relaciona algebra i relationalni račun su medjusobno ekvivalentni, što znači da se bilo koji izraz relationalnog računa može transformisati u semantički ekvivalentan niz operacija relacione algebre i obratno.

### 8.1.1. Relaciona algebra

Naše razmatranje ograničićemo na osam osnovnih operacija relacione algebre koje možemo podeliti u sledeće dve grupe:

Tradicionalne operacije nad skupovima u relationalnoj algebri:

- unija,
- presek,
- razlika,
- Dekartov proizvod.

Posebne operacije relacione algebre:

- selekcija,
- projekcija,
- spoj,
- deljenje.

Binarne operacije unija, presek i razlika mogu se izvoditi samo na relacijama koje su medjusobno uporedive. Nema smisla izvoditi operaciju unije nad dve medjusobno neuporedive relacije kao što su na primer, relacije *student* (u kojoj je svaka n-torka opis osobina jednog studenta) i relacije *predmet* (u kojoj je svakom n-torkom opisan jedan predmet). Relacije koje dozvoljavaju da se nad njima izvode operacije unija, presek i razlika su uporedive i nazivamo ih *unijski kompatibilne relacije*.

Dve relacije su unijski kompatibilne:

- ako imaju isti broj obeležja (isti stepen),
- ako izmedju dva skupa jednostavnih domena na kojima su zadate relacije, postoji preslikavanje 1:1 kojim se svaka domena jedne relacije preslikava na unijski kompatibilnu domenu druge.

Jednostavna domena je skup čiji su elementi istovrsni. Na primer, celi brojevi ili nizovi znakova predstavljaju jednostavne domene. Dve domene čiji su elementi celi brojevi, unijski su kompatibilne. Domene od kojih jedna sadrži skup kućnih brojeva (celi brojevi), a druga nazine ulica (nizovi znakova) nisu unijski kompatibilne.

#### Operacija UNIJA

Neka su  $r(R)$  i  $p(P)$  dve unijski kompatibilne relacije.

Unija relacija  $r(R)$  i  $p(P)$  u oznaci  $r \cup p$  je skup svih n-torki t sadržanih u relaciji  $r$ , relaciji  $p$  ili obe, tj.:

$$r \cup p = \{t \mid t \in r \vee t \in p\}.$$

Prilikom izvodjenja operacije unija potrebno je izvršiti usklajivanje redosleda kolona i naziva unijski kompatibilnih obeležja relacija  $r$  i  $p$ .

Operacija unija ima osobine komutativnosti i asocijativnosti.

**Primer:** Operaciju unija na dve unijski kompatibilne relacije  $r(R)$  i  $p(P)$  ilustrujemo sledećim primerima:

a)	$r(A B C)$
	a b c
	d e f
	c b f

$p(A B C)$
a b c
a b c
a b f

Vredi  $R = P$  (iste relacione šeme) i redosled obeležja uskladjen.

$r \cup p(A B C)$
a b c
d e f
c b f
a b f

b)	$r(A B C)$
	a b c
	d e f
	c b f

$p(C A B)$
c a b
f a b

Vredi  $R = P$  ali redosled obeležja nije uskladjen. Nakon usklajivanja redosleda obeležja u relaciji  $p$  sa redosledom obeležja u relaciji  $r$  ista izgleda:

$p(A B C)$
a b c
a b f

Rezultat operacije  $r \cup p$  koji se nakon toga dobija jednak je rezultatu iz prethodnog slučaja.

c)	$r(A B C)$
	a b c
	d e f
	c b f

$p(D E F)$
a b c
a b f

Vredi  $R \neq P$  (različite relacione šeme) i  $\text{dom}(A) = \text{dom}(D)$ ,  $\text{dom}(B) = \text{dom}(E)$  i  $\text{dom}(C) = \text{dom}(F)$ . Pre izvodjenja operacije unija potrebno je uskladiti nazive obeležja u relaciji  $p$  sa nazivima u relaciji  $r$ . Posle usklajivanja relacija  $p$  glasi:

$p(A B C)$
a b c
a b f

Rezultat operacije unija ove dve relacije jednak je rezultatu dobijenom u prethodna dva slučaja.

### Operacija RAZLIKA

Razlika izmedju dve unijski kompatibilne relacije  $r(R)$  i  $p(P)$  u oznaci  $r - p$  je skup svih n-torki sadržanih u relaciji  $r$ , koje istovremeno nisu sadržane u relaciji  $p$ , tj.:

$$r - p = \{t \mid t \in r \wedge t \notin p\}.$$

Operacija razlika nema osobine komutativnosti i asocijativnosti.

**Primer:** Neka su  $r(R)$  i  $p(P)$  dve unijski kompatibilne relacije i neka vredi  $R = P$ :

$r(A B C)$	$p(A B C)$
a b c	a b c
d e f	a b f
c b f	
( $r - p$ )(A B C)	( $p - r$ )(A B C)
d e f	a b f
c b f	

Kao i kod operacije unija i kod operacije razlika potrebno je pre izvodjenja operacije izvršiti usklajivanje redosleda i naziva unijski kompatibilnih obeležja.

### Operacija DEKARTOV PROIZVOD

Dekartov proizvod dve relacije  $r(R)$  i  $p(P)$  u oznaci  $r \times p$  predstavlja skup svih n-torki koje su nastale kao rezultat spajanja svake pojedine n-torke  $t_r$ ,

sadržane u relaciji  $r$  sa svakom pojedinom n-torkom  $t_p$  iz relacije  $p$ , tj.:

$$r \times p = \{(t_r, t_p) \mid t_r \in r \wedge t_p \in p\}.$$

Relaciona šema na kojoj je zadata relacija Dekartovog proizvoda ima za obeležja uniju skupova obeležja relacionih šema R i P. Ako je presek skupova obeležja relationalnih šema R i P prazan skup, tada Dekartov proizvod relacija  $r(R)$  i  $p(P)$  predstavlja relaciju, u suprotnom ne. Relaciona šema na kojoj bi bila zadata relacija Dekartovog proizvoda relacija  $r(R)$  i  $p(P)$  za koju presek skupova obeležja relationalnih šema R i P nije prazan, sadžavala bi ista obeležja što nije dozvoljeno.

Dekartov proizvod relacija se razlikuje od Dekartovog proizvoda skupova po tome što redosled vrednosti obeležja u n-torkama Dekartovog proizvoda relacije može da se menja, za razliku od članova u uredjenoj n-torci. Posledica ove činjenice je i to da je Dekartov proizvod relacija komutativna operacija za razliku od Dekartovog proizvoda skupova koji to nije.

*Primer:* Neka su  $r(R)$  i  $p(P)$  relacije na relacionim šemama R i P.

a) Neka vredi  $R \cap P = \emptyset$ . Dekartov proizvod  $r \times p$  je relacija.

$$\begin{array}{ll} r(A B) & p(C D) \\ \begin{matrix} a & b \\ c & d \\ a & c \end{matrix} & \begin{matrix} b & c \\ d & e \\ c & b \\ c & d \\ a & c \\ a & c \end{matrix} \\ (r \times p)(A B C D) & \begin{matrix} a & b & b & c \\ a & b & d & e \\ c & d & b & c \\ c & d & d & e \\ a & c & b & c \\ a & c & d & e \end{matrix} \end{array}$$

b) Neka vredi  $R \cap P \neq \emptyset$ . Dekartov proizvod  $r \times p$  nije relacija.

$$\begin{array}{ll} r(A B) & p(B D) \\ \begin{matrix} a & b \\ c & d \\ a & c \end{matrix} & \begin{matrix} b & c \\ d & e \\ c & b \\ c & d \\ a & c \\ a & c \end{matrix} \\ (r \times p)(A B B D) & \begin{matrix} a & b & b & c \\ a & b & d & e \\ c & d & b & c \\ c & d & d & e \\ a & c & b & c \\ a & c & d & e \end{matrix} \end{array}$$

### Operacija PROJEKCIJA

Projekcija je unarna operacija kojom se iz date relacije izdvajaju pojedine kolone. Neka su X i R skupovi obeležja i neka vredi  $X \subseteq R$ . Neka je  $r(R)$  relacija zadata na skupu obeležja R. Projekciju relacije  $r$  na skup obeležja X predstavlja relacija koju označavamo sa  $\pi_X(r)$  i definišemo kao

$$\pi_X(r) = \{t[X] \mid t \in r\}.$$

*Primer:* Neka je  $r(R)$  relacija na relacionoj šemi R koja sadrži obeležja A, B i C. Projekcija relacije  $r$  na obeležjima A i B biće:

$$\begin{array}{lll} r(A B C) & (A B) & \pi_{AB}(r) = r'(A B) \\ \begin{matrix} a & b & c \\ d & e & f \\ a & b & f \end{matrix} & \begin{matrix} a & b \\ d & e \\ a & b \end{matrix} & \begin{matrix} a & b \\ d & e \\ a & b \end{matrix} \end{array}$$

### Operacija SELEKCIJA

Selekcija ili kako se ponekad naziva restrikcija unarna je operacija kojom se iz relacije izdvaja određeni podskup n-torki. U operaciji selekcija mora biti definisan uslov F, ili kriterij na osnovu kojeg se izvodi izdvajanje n-torki.

Selekcija na relaciji  $r$ , prema uslovu F, je relacija koja sadrži sve n-torke sadržane u  $r$  koje zadovoljavaju uslov F, tj.:

$$\sigma_F(r) = \{t \mid t \in r \wedge t \text{ zadovoljava } F\}.$$

Kriterij za selekciju F može sadržavati: konstante, nazive obeležja relacije nad kojom se selekcija izvodi, aritmetičke operatore poredjenja ( $=, \neq, <, \leq, >, \geq$ ) i logičke operatore ( $\vee, \wedge, \neg$ ).

*Primer:* Neka je  $r(R)$  relacija sledećeg oblika:

$$\begin{array}{ll} r(A B C) & \\ \begin{matrix} a & 1 & 2 \\ d & 3 & 4 \\ a & 5 & 5 \end{matrix} & \end{array}$$

$\sigma_{A=a}(r) (A \ B \ C)$	$\sigma_{B=c}(r) (A \ B \ C)$	$\sigma_{B < C}(r) (A \ B \ C)$
a 1 2	a 5 5	a 1 2
a 5 5		d 3 4

Od osam operatora relacione algebre do sada smo definisali pet koji spadaju u grupu jednostavnih ili osnovnih operacija. Operacije presek, spoj i deljenje realizuju se kombinacijom jednostavnih operacija i spadaju u grupu složenih operacija. Značaj ovih složenih operacija, a posebno operacija spoj, za rad sa bazom podataka daleko je veći od značaja nekih jednostavnih operacija. Na primer, Codd kao jedan od uslova da bi se neki SUBP mogao minimalno smatrati relacionim postavlja zahtev da pored operacija projekcija i selekcija, podržava i operaciju spoj.

### Operacija PRESEK

Operacija presek je binarna operacija koja, kao i unija i razlika, zahteva uniju kompatibilnost relacija nad kojima se izvodi.

Presek dve uniju kompatibilne relacije  $r(R)$  i  $p(P)$  je relacija koja sadrži sve one n-torce koje su istovremeno elementi i relacije  $r(R)$  i relacije  $p(P)$ , tj.:

$$r \cap p = \{t \mid t \in r \wedge t \in p\}.$$

Operacija presek može biti izražena i pomoću operacije razlika:

$$r \cap p = r - (r - p).$$

*Primer:* Neka su  $r(R)$  i  $p(P)$  relacije i neka vredi  $R = P$ .

$r(A \ B \ C)$	$p(A \ B \ C)$	$r \cap p = r'(A \ B \ C)$
a b c	a b c	a b c
d e f	a b f	e f g
e f g	e f g	
a b g		

### Operacija SPOJ

Operacija spoj (join) je složena binarna operacija za koju možemo reći da se izvodi u sledeća tri koraka:

1. Korak - formiranje dekartovog proizvoda relacija;

2. Korak - iz Dekartovog proizvoda izdvajaju se n-torce koje zadovoljavaju postavljene uslove;
3. Korak - iz tabele dobijene u drugom koraku izdvajaju se odredjene kolone. Ovaj korak izvodi se samo u slučaju prirodnog spoja.

*Primer:* Neka su  $r(R)$  i  $p(P)$  relacije i neka obeležje B pripada šemama relacija R, a obeležje C šemama relacija P. Operacija spoj relacija  $r \cup p$  na osnovu uslova  $B < C$  ima sledeći tok:

$r(A \ B)$	$p(C \ D)$
a 2	4 d
e 5	1 f
f 2	

1. Korak:  $(r \times p) = r' (A \ B \ C \ D)$

a 2 4 d
a 2 1 f
e 5 4 d
e 5 1 f
f 2 4 d
f 2 1 f

2. Korak:  $\sigma_{B < C}(r') = r'' (A \ B \ C \ D)$

a 2 4 d
f 2 4 d

*Theta spoj* ( $\Theta$  - spoj) predstavlja najopštiji oblik operacije spoj. Neka su  $r(R)$  i  $p(P)$  relacije i neka vredi  $A_i \in R$  i  $B_j \in P$ . Theta spoj relacija  $r \cup p$  na osnovu izdvajanja  $A_i \Theta B_j$  ( $\Theta \in \{=, \neq, <, \leq, >, \geq\}$ ) u označi  $r[A_i \Theta B_j]p$  je skup svih n-torki koji zadovoljavaju sledeće uslove:

- podskup je Dekartovog proizvoda relacija  $r \cup p$ , i
- svaki element tog podskupa zadovoljava uslov izdvajanja  $[A_i \Theta B_j]$ , odnosno:

$$r[A_i \Theta B_j]p = \{(t_r, t_p) \mid t_r \in r \wedge t_p \in p \wedge t_r[A_i] \Theta t_p[B_j]\}.$$

Operacija  $\Theta$  - spoj nad relacijama  $r(R)$  i  $p(P)$  je relacija ukoliko je i Dekartov proizvod nad istim relacijama relacija (setimo se uslova  $R \cap P = \emptyset$ ). U tom slučaju operaciju  $\Theta$  - spoj možemo formalno definisati na osnovu operacija Dekartov proizvod i selekcija:

$$r[A_i \Theta B_j]p = \sigma_{A_i \Theta B_j}(r \cup p).$$

Postoje sledeća dva posebna slučaja  $\Theta$  - spoja.

Prvi poseban slučaj je kada uslov izdvajanja ne postoji i kada operacija  $\Theta$  - spoj prelazi u Dekartov proizvod.

Dруги poseban slučaj je kod kojeg je  $\Theta$  operator znak jednakosti ( $=$ ), koji nazivamo spoj sa izjednačavanjem (equi-join). Rezultat operacije spoj sa izjednačavanjem je tabela sa dve potpuno identične kolone. Obeležja čije vrednosti su sadržane u tim kolonama mogu, ali ne moraju biti različita.

**Primer:** Neka su  $r(R)$  i  $p(P)$  relacije i neka je  $B \in R$  i  $C \in P$ . Neka je uslov izdvajanja  $B = C$ .

$r(A\ B)$	$p(C\ D)$	$r[B = C]p = q(A\ B\ C\ D)$
a 6	1 d	a 6 6 f
b 1	6 f	b 1 1 d
e 3		e 1 1 d
e 1		

U prethodnom rezultatu  $\Theta$  - spoja uočavamo dve iste kolone koje pripadaju obeležjima  $B$  i  $C$ , koje možemo bar po nazivu smatrati različitim pa prema tome i rezultat u ovom slučaju smatrano relacijom. To ne mora biti, i vrlo često i nije slučaj.

**Prirodni spoj.** Prirodni spoj (natural join) u tesnoj je vezi sa  $\Theta$  - spojem. Kao rezultat  $\Theta$  - spoja sa izjednačavanjem dobija se tabela sa najmanje dve kolone identičnih vrednosti obeležja, a često će i nazivi obeležja biti identični. Da bi smo tabelu, koja sadrži po dve kopije jednog ili više obeležja transformisali u relaciju, moramo odstraniti kopije obeležja iz zaglavljaja tabele zajedno sa odgovarajućim kolonama u tabeli. Ova dodatna operacija izvodi se u okviru operacije prirodni spoj.

Prirodnim spojem dve relacije spajaju se medjusobno n-torce tih relacija na osnovu vrednosti obeležja koja se nalaze u obe šeme relacija.

Neka su  $r(R)$  i  $p(P)$  relacije i neka je  $R \cup P = T$ . Prirodni spoj relacija  $r$  i  $p$  označavamo sa  $r \nabla p$ . Rezultat operacije prirodni spoj je relacija  $q(T)$ . Za svaku n-torku  $t_q$  u relaciji  $q$  postoje n-torce  $t_r \in r$  i  $t_p \in p$  za koje vredi

$t_r = t_q [R]$  i  $t_p = t_q [P]$  odnosno:

$$r \nabla p = \{t_q \mid t_q [R] = t_r \wedge t_r \in r \wedge t_q [P] = t_p \wedge t_p \in p\}.$$

Operaciju prirodni spoj možemo izraziti pomoću jednostavnih operacija Dekartov proizvod, selekcija i projekcija. Neka su  $r(R)$  i  $p(P)$  relacije, neka vredi  $R \cap P = X$  i neka je  $R.X \in R$  i  $P.X \in P$ . Vredi:

$$r \nabla p = \pi_{RP} \sigma_{R.X=P.X} (r \times p).$$

**Primer:** Neka su  $r(R)$  i  $p(P)$  relacije i neka vredi  $R = \{A, B, C\}$ ,  $P = \{B, C, D\}$  i  $T = \{A, B, C, D\}$ .

$r(A\ B\ C)$	$p(B\ C\ D)$	$r \nabla p = q(A\ B\ C\ D)$
a 2 4	2 7 d	a 2 4 k
b 1 5	1 5 h	b 1 5 h
c 2 7	2 4 k	c 2 7 d

Postoje dva posebna i za nas interesantna slučaja operacije prirodnog spoja.

Prvi slučaj javlja se kada za relacije  $r(R)$  i  $p(P)$  čiji prirodan spoj želimo odrediti vredi  $R \cap P = \emptyset$ , tj. relacije nemaju zajedničkog obeležja. U tom slučaju prirodan spoj jednak je Dekartovom proizvodu tih relacija.

Dруги poseban slučaj javlja se kada za relacije  $r(R)$  i  $p(P)$  vredi  $R = P$ , tj. kada su obe relacije zadate na istoj relacionoj šemi. Rezultat prirodnog spoja u tom slučaju jednak je preseku relacija  $r \cap p$ .

### Operacija DELJENJE

Neka su  $r(R)$  i  $p(P)$  relacije i neka vredi  $R = \{X, Y\}$  i  $Y$  je unijski kompatibilan sa  $P$ . Deljenje relacije  $r$  sa relacijom  $p$  označavamo sa  $r/p$ . Rezultat deljenja je relacija  $q(X)$  koja je najveći podskup od  $\pi_X(r)$  za koji vredi da je  $q \nabla p$  sadržan u  $r$ .

Dруги način za definisanje operacije deljenja je preko karakteristika n-torki od kojih se relacija  $q$  sastoji, odnosno:

$$r/p = \{t \mid \forall t_p \in p, \exists t_r \in r \text{ za koju vredi } t_r [X] = t \text{ i } t_r [Y] = t_p\}.$$

Deljenje kao složena operacija može se izraziti pomoću jednostavnih operacija:

$$r/p = \pi_X(r) - \pi_X((\pi_X(r) \nabla p) - r).$$

*Primer:* Neka su  $r(R)$  i  $p(P)$  relacije i neka vredi  $R = [X, Y]$  i  $Y = P$ .

$r(X Y)$	$p(Y)$	$(r/p)(X)$
$x_1 y_1$	$y_1$	$x_1$
$x_1 y_2$	$y_2$	$x_5$
$x_3 y_1$		
$x_4 y_2$		
$x_4 y_3$		
$x_5 y_1$		
$x_5 y_2$		

Rešavanje po fazama imalo bi sledeći tok:

$$\pi_X(r) = r(X) \quad r \nabla p = q(X Y) \quad q \cdot r = s(X Y)$$

$x_1$	$x_1 y_1$	$x_3 y_2$
$x_3$	$x_1 y_2$	$x_4 y_1$
$x_4$	$x_3 y_1$	
$x_5$	$x_3 y_2$	
	$x_4 y_1$	
	$x_4 y_2$	
	$x_5 y_1$	
	$x_5 y_2$	

$$\pi_X(s) = t(X) \quad r \cdot t = r/p = u(X)$$

$x_3$	$x_1$
$x_4$	$x_5$

Operacija deljenja nije niti komutativna niti asocijativna operacija.

## 8.2. Nul - vrednosti

Model baze podataka polazi od prepostavke da su nam sve relevantne informacije poznate i da iste mogu biti unesene u bazu podataka. U praksi ova prepostavka često nije ispunjena. Problemom nedostajućih informacija u relacionom modelu podataka bavili su se mnogi autori. Za sada postoji više

predloga rešenja, ali celovito rešenje informacija koje nedostaju, ne postoji. Zbog toga ovo područje predstavlja jedno od značajnih područja istraživanja u relationalnom modelu podataka.

Problem nedostajućih informacija se pojavljuje npr. kada u trenutku unosa podataka u bazu podataka nisu poznati datum rođenja, adresa boravka osobe i sl. Uz određeni trud te informacije bi se eventualno mogle pribaviti. Postoje međutim slučajevi kada bez obzira na potreban trud nije moguće doći do određenih informacija. Na primer, nije moguće utvrditi ime bračnog druga osobe koja nije oženjena.

Na mesto na koje treba upisati stvarnu vrednost obeležja koje nam nije poznato u bazu podataka upisuje se **nul - vrednost**.

Nul - vrednost predstavlja oznaku da stvarna vrednost nije poznata. Kao oznaku za nul - vrednosti, koristićemo znak "?".

Uvodjenje nul - vrednosti komplikuje definisanje operacijske komponente relacionog modela podataka, te ih treba izbegavati. Napomenimo na kraju da se nul - vrednosti, kao i operacije sa takvim vrednostima detaljno razmatraju u RM/T modelu.

## 8.3. Integritetna komponenta

Pod integritetom baze podataka kao što je već bilo rečeno podrazumevamo ispravnost i istinitost podataka sadržanih u bazi podataka. Uslovima ili pravilima integriteta se definišu ograničenja sadržaja baze podataka na neka dozvoljena stanja. Uslove integriteta koji se javljaju u jednoj relacionoj bazi podataka možemo podeliti u sledeće dve grupe:

1. Strukturna (inherentna) pravila integriteta,
2. Posebna (eksplicitna) pravila integriteta.

Strukturna pravila integriteta vrede u svim relacionim bazama podataka. U relacionom modelu podataka pojam ključa predstavlja takvo ograničenje. Jedna vrednost ključa može se samo jednom pojaviti u jednoj relaciji i sva ostala obeležja funkcionalno zavise od ključa.

Posebna pravila integriteta definišu uslove nad jednom bazom podataka, a mogu biti statička i dinamička. Statičkim pravilima integriteta definisani su uslovi koji moraju važiti pre i posle izvršenja bilo koje operacije nad bazom podataka. Dinamičkim pravilima integriteta definisane su *procedure* na relacionom modelu, koje garantuju ostvarenje uslova integriteta.

Relacioni model podataka poseduje sledeća statička pravila integriteta:

- integritet objekta,
- referencijalni integritet.

### 8.3.1. Integritet objekta

Integritet objekta vezan je za pojam primarnog ključa šeme relacije koji smo ranije definisali. S obzirom na osobine primarnog ključa sledi i definicija integriteta objekta koja glasi:

*Vrednost primarnog ključa kao celine, i niti jedne njegove komponente ne sme biti jednak nul-vrednosti.*

Primarni ključ omogućava jednostavno i efikasno adresiranje n-torki. Nul-vrednost primarnog ključa kao celine ili neke njegove komponente ne bi mogla ostvariti prethodnu ulogu.

### 8.3.2. Referencijalni integritet

Referencijalni integritet predstavlja poseban slučaj opštijeg uslova integriteta, koji se naziva *zavisnost sadržavanja*.

Zavisnost sadržavanja se naziva izraz oblika:

$$R[Y] \subseteq P[X]$$

gde su R i P dve šeme relacije,  $Y \subseteq R$  i  $X \subseteq P$  skupovi unijiski kompatibilnih obeležja.

Zavisnost sadržavanja  $R[Y] \subseteq P[X]$  je zadovoljena, ako važi:

$$(\forall t_r \in r(R)) (\exists t_p \in p(P)) (t_r[Y] = t_p[X]).$$

Zavisnost sadržavanja definiše egzistencijalno ograničenje u smislu da se u relaciji  $r$  ne može upisati n-torka  $t_r$  (pozivajuća n-torka), ako u relaciji  $p$  ne postoji bar jedna n-torka  $t_p$  (ciljna n-torka) takva da važi  $t_r[Y] = t_p[X]$ . Pri brisanju n-torke  $t_p$  iz  $p$ , ako važi  $(\exists t_p \in p(P)) (t_p[X] = x)$ , moraju se brisati i sve torke  $t_r$  iz  $r(R)$  za koje važi  $t_r[Y] = x$ .

U odnosu na zavisnost sadržavanja referencijalnim integritetom se zahteva da skup obeležja X mora predstavljati ključ šeme relacije P. To znači ako u relaciji  $r(R)$  postoji skup obeležja Y koji odgovara primarnom ključu relacije  $p(P)$ , svaka vrednost obeležja Y u  $r$  mora biti:

- ili jednaka vrednosti primarnog ključa u nekoj od n-torki relacije  $p$ .
- ili jednaka nul-vrednosti.

Relacije  $r$  i  $p$  ne moraju biti različite relacije.

Skup obeležja Y koji ispunjava prethodne uslove naziva se kao što je već bilo rečeno spoljni ključ u relaciji R. Spoljni ključ ima ulogu da u relacionom modelu podataka poveže n-torke u semantičku strukturu.

Spoljni ključ relacije  $r$  ne može poprimiti nul-vrednost ukoliko je isti podskup skupa obeležja koja predstavljaju primarni ključ relacije, jer bi to bilo u suprotnosti sa ranije definisanim integritetom objekta.

Dakle, spoljni ključ, može poprimiti nul-vrednost samo ukoliko nije podskup skupa obeležja koja predstavljaju primarni ključ. Takve situacije se objektivno javljaju i moraju se dozvoliti. Na primer, u relaciji *nastavnik* spoljni ključ koji se odnosi na neposredno nadredjenog rukovodioca primiče nul-vrednost za radnika koji je na čelu organizacije.

### 8.3.3. Implementacija integritetskih ograničenja

Uslovi integriteta definišu dozvoljena stanja baze podataka i deo su relacionog modela podataka. Za nas je posebno interesantno i korisno razmatranje

načina implementacije uslova ograničenja što ne spada u deo relationalnog modela.

Najjednostavnije rešenje realizacije uslova integriteta je da SUBP odbije izvodjenje bilo koje operacije nad bazom podataka čiji rezultat bi doveo bazu podataka u nedozvoljeno stanje. U nekim slučajevima, takva operacija bi se mogla izvesti uz izvodjenje nekih dodatnih operacija kojima bi se kompenzirali zahtevi operacije tako da ukupan rezultat bude dozvoljeno stanje baze podataka. Odluku o eventualnom izvršenju zahtevane operacije treba preko SUBP prepustiti korisniku. Za pojedine operacije korisnik bi mogao izabrati:

- pod kojim uslovima treba odbiti njihovo izvodjenje,
- pod kojim uslovima izvodjenje operacije može biti dozvoljeno,
- koje kompenzacijalne operacije treba izvesti.

Implementacija uslova integriteta objekta svodi se na zabranu izvodjenja operacija koje mogu imati za posledicu da čitav ključ ili neki deo ključa neke n-torke u relaciji poprini nul-vrednost. Jedan od načina kao što ćemo kasnije videti je definisanje da li obeležje može ili ne može poprimiti nul-vrednost.

Što se tiče referencijskog integriteta postoje nešto veće mogućnosti izbora.

U slučaju da se zahteva brisanje neke ciljne n-torke moguća su sledeća rešenja:

1. Ciljna n-torka ne može se brisati ako postoji odgovarajuća pozivajuća n-torka (RESTRICTED).
2. Kao deo operacije brisanja ciljne n-torke treba izvršiti brisanje svih pozivajućih n-torki u bazi podataka, u kojima je vrednost spoljnog ključa jednaka vrednosti primarnog ključa ciljne n-torke (CASCADES).
3. Kao deo operacije brisanja ciljne n-torke vrednosti spoljnih ključeva u pozivajućim n-torkama ažuriraju se na nul-vrednost (NULLIFIES) pod uslovom da je to dozvoljeno.
4. Kao deo operacije brisanja ciljne n-torke vrednosti spoljnih ključeva u pozivajućim n-torkama ažuriraju se na prepostavljenu vrednost (DEFAULT).

Slične mogućnosti postoje i u slučaju kada se zahteva operacija ažuriranja ključa ciljne n-torke:

1. Operacija ažuriranja može biti izvedena samo u slučaju da u bazi podataka ne postoje odgovarajuće pozivajuće n-torce (RESTRICTED).
2. U okviru operacije ažuriranja vrednosti primarnog ključa ciljne n-torke, na novu vrednost primarnog ključa ciljne n-torke ažuriraju se vrednosti spoljnih ključeva u pozivajućim n-torkama (CASCADES).
3. Kao deo operacije ažuriranja vrednosti primarnog ključa ciljne n-torke na novu vrednost, ažuriraju se vrednosti spoljnih ključeva u pozivajućim n-torkama na nul-vrednost, ukoliko je to dozvoljeno (NULLIFIES).
4. Kao deo operacije ažuriranja vrednosti primarnog ključa ciljne n-torke na novu vrednost, ažuriraju se vrednosti spoljnih ključeva u pozivajućim n-torkama na prepostavljenu vrednost (DEFAULT).

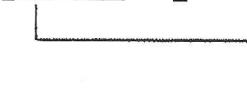
Izbor neke od izloženih mogućnosti zavisi pre svega od prirode semantičkog odnosa između pozivajuće i ciljne n-torke, odnosno od prirode odnosa koji je prikazan pomoću spoljnog ključa.

Sa nekoliko sledećih primera ilustrovaćemo neke od mogućih odnosa.

- a) Postojanje ciljne n-torke uslov je za postojanje pozivajuće n-torke. U tom slučaju brisanje ciljne n-torke ima za posledicu brisanje svih pozivajućih n-torki, a promena vrednosti primarnog ključa ciljne n-torke ima za posledicu odgovarajuće promene vrednosti spoljnih ključeva u pozivajućim n-torkama.

Kao primer takvog odnosa, možemo uzeti odnos između n-torki u relaciji *rukovodilac* i n-torki u relaciji *radnik*. Svaki rukovodilac mora biti radnik u radnoj organizaciji. U trenutku kada radnik napušta radnu organizaciju i kada treba brisati n-torku iz relacije *radnik* prestaje mogućnost da isti bude rukovodilac te iz relacije *rukovodilac* treba brisati odgovarajuću n-torku. U slučaju ažuriranja primarnog ključa n-torke u relaciji *radnik*, treba ažurirati i vrednost spoljnog ključa u relaciji *rukovodilac*.

RUKOVODILAC (SIF\_RUKOV, SIF\_ORG, ...)

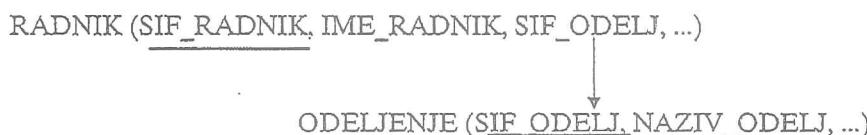


RADNIK (SIF\_RADNIK, IME\_RADNIK, ...).

Obeležja SIF\_RADNIK i SIF\_RUKOV imaju iste domene.

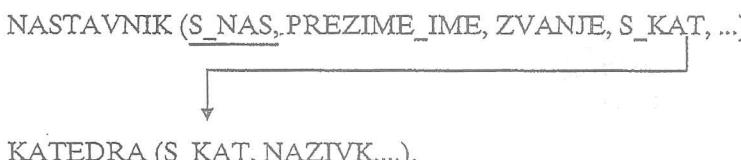
- b) Za svaku pozivajuću n-torku mora postojati pridružena ciljna n-torka. Za razliku od prethodnog slučaja, ciljna n-torka se može menjati. Brisanju ciljne n-torke mora u ovom slučaju prethoditi ažuriranje vrednosti spoljnog ključa u pozivajućim n-torkama.

Kao primer takvog odnosa su n-torke relacije *radnik* i n-torke relacije *odeljenje*. Svaki radnik mora biti rasporedjen u neko odeljenje organizacije. U trenutku kada se ukida neko odeljenje radnici iz tog odeljenja moraju biti rasporedjeni u druga odeljenja. U skladu sa tim brisanju n-torke u relaciji *odeljenje* mora prethoditi ažuriranje vrednosti spoljnih ključeva u n-torkama svih radnika tog odeljenja.



- c) Pozivajuća n-torka ne zavisi od ciljne. Ako se ciljna n-torka briše, vrednost spoljnih ključeva u pozivajućim n-torkama se mora ažurirati, ali može biti i nul-vrednost.

U ovoj vrsti odnosa stoje n-torke relacije *nastavnik* i n-torke relacije *katedra*. Razlog za brisanje n-torke katedre može biti ukidanje katedre. Nastavnik koji je pripadao jednoj katedri može dobiti drugu katedru, ali može ostati i bez katedre. U skladu sa iznetim, brisanju n-torke u relaciji *katedra* mora prethoditi ažuriranje vrednosti spoljnog ključa n-torke u relaciji *nastavnik*, pri čemu nova vrednost može biti i nul-vrednost. Uočavamo i slučaj kada brisanju n-torke relacije *katedra* za koju nije određen nastavnik ne uzrokuje nikakva ažuriranja vrednosti spoljnog ključa u relaciji *nastavnik*.



#### 8.4. Normalizacija

Osnovni cilj relacionog modela podataka je da odgovarajuća baza podataka:

1. Ne sadrži redundansu,
2. Da se može jednostavno koristiti i menjati.

*Normalne forme* daju formalne kriterije prema kojima se utvrđuje da li model podataka ispunjava prethodne zahteve.

*Normalizacija* je proces provere uslova normalnih formi i po potrebi sredstvene relacije na oblik koji zadovoljava iste.

Procesom normalizacije želi se razviti dobar model podataka tako da se iz nekog početno zadatog modela otklone slabosti (redundansa i problemi u održavanju).

Pre nego što detaljno opišemo normalne forme i proces normalizacije ukratko ćemo se upoznati sa redundansom i značenjem pojma jednostavno korišćenje i menjanje baze podataka.

Pod redundansom podrazumevamo višestruko memorisanje iste informacije u bazi podataka. Cilj koji se teži dostići pri projektovanju baze podataka je eliminisanje redundanse zbog niza negativnih posledica koje ona donosi. Višestruko memorisanje istog podatka dovodi do povećanog korišćenja memorijskog prostora i otežanog održavanja podataka. Memorisanjem više kopija istih podataka možemo u nekim slučajevima smanjiti vreme obrade podataka. Potpuno eliminisanje redundanse podataka u bazi podataka je skoro nemoguće ostvariti. Realni cilj pri projektovanju baze podataka je kontrolisana redundansa podataka.

Jednostavno korišćenje i menjanje podataka podrazumeva pre svega sprečavanje *anomalija održavanja podataka*. Pod anomalijama održavanja podataka podrazumevamo:

- anomaliju dodavanja,
- anomaliju brisanja,
- anomaliju promene.

Svaka od ovih anomalija manifestuje se na specifičan način, a njihov zajednički uzrok je povezivanje opisa svojstava različitih objekata u jedan zapis u bazi podataka.

*Anomalija dodavanja* (unošenja) javlja se u onim slučajevima kada su informacije o svojstvima jednog objekta memorisane u bazi podataka kao deo informacije o svojstvima nekog drugog objekta. Na primer u okviru opisa nastavnika memorisane su informacije o predmetu koji predaje ili katedre na kojoj radi. Informacije o predmetu, odnosno katedri nije moguće uneti u bazu podataka sve dok ne postoji bar jedan nastavnik koji taj predmet predaje odnosno dok ne postoji najmanje jedan nastavnik koji na toj katedri radi.

*Anomalija brisanja* je inverzija anomalije dodavanja. Neka su u okviru opisa svojstava nastavnik memorisane informacije o predmetu koji predaje. Svakim brisanjem opisa nastavnika briše se i jedna kopija podataka o predmetu koji predaje. Kada se brišu podaci o poslednjem nastavniku koji predaje neki predmet, biće brisana i poslednja kopija podataka o predmetu. S obzirom na to da pri brisanju podataka o nastavniku ne mislimo na druge posledice, na ovaj način moguće je ostati bez podataka o predmetu koji su potrebni i važni.

*Anomalija menjanja* (ažuriranja) javlja se u slučaju kada promenu podataka o jednom objektu treba izvršiti na više od jedne kopije podataka. Razmotrimo ponovo prethodni primer gde su podaci o predmetu i katedri memorisani u okviru opisa nastavnika. U bazi podataka u jednom trenutku postoji toliko opisa katedre koliko nastavnika radi na toj katedri. Ako treba promeniti podatke o opisu katedre (na primer naziv katedre) tada tu promenu treba izvršiti na onoliko mesta koliko nastavnika radi na toj katedri. Ako se promena ne izvrši na svim kopijama nastaje situacija u kojoj o istom svojstvu jednog objekta imamo više različitih tvrdnji od kojih bar jedna nije istinita. Ovakvo stanje smatramo *nekonzistentnom bazom podataka*.

#### 8.4.1. Metode normalizacije

U najopštijem smislu normalizacija je postupak kojim se proizvoljna, nenormalizovana relacija transformiše u skup manjih normalizovanih relacija. Normalizacija se izvodi na osnovu zavisnosti koje iskazuju zakonitosti koje vrede u svetu čiji model podataka gradimo.

Bitna osobina koja se očekuje od normalizacije je *reverzibilnost* tj. da ne sme doći do gubitka informacija sadržanih u polaznoj relaciji. Polazeći od skupa normalizovanih relacija, mora biti moguća rekonstrukcija polazne nenormalizovane relacije.

Postoje sledeće dve tehnike normalizacije:

1. Vertikalna normalizacija,
2. Horizontalna normalizacija.

*Vertikalna normalizacija* je postupak kojim se proizvoljna nenormalizovana šema relacije transformiše u skup manjih i normalizovanih šema relacija. Iz relacione šeme se izdvajaju obeležja koja stoje u nedozvoljenim odnosima sa ostalim obeležjima u šemi. Od izdvajenih obeležja formira se nova šema relacije. Transformacija relacije zadate na relacionoj šemi neposredna je posledica normalizacije relacione šeme.

Vertikalna normalizacija zasniva se na operacijama *projekcija i prirodni spoj*. Pomoću operacije projekcija relaciju vertikalno razbijamo na dve ili više manjih relacija. Pri tome dolazi do cepanja svake pojedine n-torce u relaciji. Operaciju prirodni spoj koristimo da bi dokazali reverzibilnost, tj. da bi rekonstruisali polaznu, nenormalizovanu relaciju.

*Horizontalnom normalizacijom* relacija se rastavlja na podskupove n-torki - *fragmente relacije* koji zadovoljavaju odredjene uslove. Horizontalna normalizacija zasniva se na operacijama *selekcija i unija*. Sama tehnika je još uvek u razvoju, a značajnu ulogu mogla bi odigrati kod distribuiranih baza podataka. Kod distribuiranih baza podataka relacija ne mora u potpunosti biti memorisana na jednoj lokaciji. Fragmenti relacije memorišu se na pojedinim lokacijama, što bi se moglo koristiti za samu normalizaciju.

U daljim razmatranjima normalizacije ograničićemo se samo na vertikalnu normalizaciju.

Postoje sledeće dve varijante vertikalne normalizacije:

- normalizacija dekompozicijom,
- normalizacija sintezom.

*Normalizacija dekompozicijom* započinje od proizvoljne nenormalizovane relacione šeme i izvodi se u koracima. Svakim korakom normalizacije

relaciona šema prevodi se u višu normalnu formu, tako da se polazni skup oboležja deli u dva skupa i od svakog formira posebna relaciona šema. Svaki korak normalizacije mora biti reverzibilan.

*Normalizacija sintezom* polazi od skupa oboležja i od skupa zavisnosti zadatih na tom skupu oboležja. Postupak se ne izvodi u koracima već se direktno formiraju relateone šeme koje ispunjavaju uslove zahtevane normalne forme.

#### 8.4.2. Dekompozicija bez gubitka informacija

Istakli smo ranije da je bitna osobina normalizacije reverzibilnost odnosno dekompozicijom ne sme doći do gubitka informacija. Dekompozicija je bez dekompozicije ako se polazna relacija može generisati spajanjem relacija koje su generisane dekompozicijom.

Napomenimo da bi bolji naziv za *dekompoziciju bez gubitka informacija* mogao biti *dekompozicija bez gubitka pouzdanosti* (ili kraće *pouzdana dekompozicija*). Naime, kod dekompozicije sa gubitkom informacija, podaci se u pravilu ne gube, već ih po spajanju ima više nego što ih je bilo u polaznoj relaciji.

Razmotrimo sada malo detaljnije pod kojim uslovima će dekompozicija biti bez gubitka informacija.

Dekompozicija relateone šeme  $R(A_1, A_2, \dots, A_n)$  je zamena relateone šeme  $R$  sa skupom relateonih šema  $\{R_1, R_2, \dots, R_k\}$  za koje vredi  $R_i \subseteq R$  ( $1 \leq i \leq k$ ) i skupom oboležja relacija  $R_i \subseteq R$  ( $1 \leq i \leq k$ ) jednaka je skupu  $R_1 R_2 \cup \dots \cup R_k = R$  (unija oboležja relacija  $R_i \subseteq R$  ( $1 \leq i \leq k$ ) jednaka je skupu oboležja relacije  $R$ ). Relateone šeme  $R_i$  ne moraju imati medjusobno disjunktne skupove oboležja.

Neka je  $R$  relateona šema i neka je skup  $\{R_1, R_2, \dots, R_k\}$  dekompozicija od  $R$ . Neka je  $r$  relacija zadata na  $R$ , i neka su  $r_i$  projekcije te relacije zadate na  $R_i$ . Neka je  $r$  relacija zadata na  $R$ , i neka su  $r_i$  projekcije te relacije zadate na  $R_i$  ( $1 \leq i \leq k$ ). Smatramo da je dekompozicija relateone šeme  $R$  bez gubitka informacija, ako za bilo koju relaciju  $r$  zadatu na  $R$  vredi da je rezultat prirodnog spajanja projekcija  $r_i$  zadatih na  $R_i$ .

Reverzibilnost dekompozicije relateone šeme dokazuje se reverzibilnošću dekompozicije relacija zadatih na toj relateonoj šemi. Ranije smo rekli da su

osnov vertikalne normalizacije operacije projekcija i prirodni spoj. Operacija projekcija koristi se da bi se relacija  $r(R)$  rastavila na dve ili više manjih i pravilnih relacija  $r_i(R_i)$ . Operaciju prirodni spoj koristimo da bi polazeći od projekcija neke relacije rekonstruisali samu relaciju.

Problemi koji se javljaju imaju svoj uzrok u činjenici da operacije projekcija i prirodni spoj nisu medjusobno inverzne. Samo pod određenim uslovima biće moguće prirodnim spajanjem projekcija neke relacije rekonstruisati samu relaciju. Na jednom primeru razmotriti ćemo o čemu je reč.

**Primer:** Neka je  $p$  relacija zadata na relateonoj šemi  $P(X, Y, Z)$ . Neka su  $r(R)$  i  $s(S)$  projekcije relacije  $p$  zadate na relateonim šemama  $R$  i  $S$  za koje vredi  $R \cup S = P$ .

$$p(X \ Y \ Z)$$

$$\begin{array}{lll} x_1 & y_1 & z_1 \\ x_2 & y_1 & z_2 \\ x_3 & y_2 & z_2 \end{array}$$

$$a) \pi_{XY}(p) = r(X \ Y) \quad \pi_{XZ}(p) = s(X \ Z) \quad r \nabla s = p(X \ Y \ Z)$$

$$\begin{array}{ll} x_1 & y_1 \\ x_2 & y_1 \\ x_3 & y_2 \end{array} \quad \begin{array}{ll} x_1 & z_1 \\ x_2 & z_2 \\ x_3 & z_2 \end{array} \quad \begin{array}{lll} x_1 & y_1 & z_1 \\ x_2 & y_1 & z_2 \\ x_3 & y_2 & z_2 \end{array}$$

$$b) \pi_{XY}(p) = r(X \ Y) \quad \pi_{YZ}(p) = s(Y \ Z) \quad (r \nabla s)(X \ Y \ Z)$$

$$\begin{array}{ll} x_1 & y_1 \\ x_2 & y_1 \\ x_3 & y_2 \end{array} \quad \begin{array}{ll} y_1 & z_1 \\ y_1 & z_2 \\ y_2 & z_2 \end{array} \quad \begin{array}{lll} x_1 & y_1 & z_1 \\ x_1 & y_1 & z_2 \\ x_2 & y_1 & z_1 \\ x_2 & y_1 & z_2 \\ x_3 & y_2 & z_1 \\ x_3 & y_2 & z_2 \end{array}$$

$$c) \pi_{XY}(p) = r(X \ Y) \quad \pi_Z(p) = s(Z) \quad (r \nabla s)(X \ Y \ Z)$$

$$\begin{array}{ll} x_1 & y_1 \\ x_2 & y_1 \\ x_3 & y_2 \end{array} \quad \begin{array}{l} z_1 \\ z_2 \end{array} \quad \begin{array}{lll} x_1 & y_1 & z_1 \\ x_1 & y_1 & z_2 \\ x_2 & y_1 & z_1 \\ x_2 & y_1 & z_2 \\ x_3 & y_2 & z_1 \\ x_3 & y_2 & z_2 \end{array}$$

Samo u slučaju a) dekompozicija relacije  $P$  je reverzibilna. U slučaju c) vredi  $R \cap S = \emptyset$ , pa je rezultat prirodnog spoja jednak Dekartovom proizvodu projekcija. Očigledno presek obeležja projekcija ne sme biti prazan skup. Taj uslov je nužan, ali kao što se vidi iz slučaja b) nije dovoljan.

Uslov koji mora biti ispunjen da bi dekompozicija bila reverzibilna glasi:

*Dekompozicija relacione šeme P na relacione šeme R i S je reverzibilna ako su zajednička obeležja u relacionim šemama kandidat za ključ u bar jednoj od ove dve šeme.*

Drugim rečima, dekompozicija relacione šeme P na relacione šeme R i S je bez gubitka informacija ako je presek skupova obeležja šema relacija R i S kandidat za ključ u barem jednoj od tih šema relacija. Napomenimo da prethodno iskazan uslov o dekompoziciji bez gubitka informacija, pored primera kojim je ilustrovan ima i strog matematički dokaz.

#### 8.4.3. Vertikalna normalizacija dekompozicijom

U kontekstu vertikalne normalizacije definisano je šest *normalnih formi* (NF) šema relacija:

- prva normalna forma (1NF),
- druga normalna forma (2NF),
- treća normalna forma (3NF),
- Boyce/Coddova normalna forma (BCNF),
- četvrta normalna forma (4NF),
- peta normalna forma (5NF).

Zadatak postupka normalizacije je da relacionu šemu prvo transformiše u 1NF, zatim u 2NF, 3NF i tako redom. Što je redni broj normalne forme veći to su i uslovi koji se postavljaju strožiji.

Polazeći od pojmove funkcionalne zavisnosti i dekompozicije bez gubitka informacija, definisane su prva, druga, treća i Boyce/Coddova normalna forma i postupak normalizacije kojim se te forme postižu. S obzirom da se u praksi često zadovoljava sa 3NF to će ovde razmatranja biti ograničena na prve tri normalne forme. Za razmatranja 4NF i 5NF potrebno je uvodjenje pojma višeznačne funkcionalne zavisnosti.

#### Prva normalna forma

*Šema relacije je u prvoj normalnoj formi ako i samo ako je domena svakog od njenih obeležja skup atomarnih vrednosti.*

S obzirom da je u kontekstu relacionog modela sama relacija definisana kao neprazan podskup Dekartovog proizvoda atomarnih domena, sledi da je svaka šema relacije u 1NF.

Do sada izneti primeri ukazuju da 1NF šeme relacije nije dovoljan uslov za dobar model podataka (ne otklanja se redundansa i anomalije održavanja).

#### Druga normalna forma

Druga normalna forma nema većeg praktičnog značaja. Ovde je navodimo kao neophodnog prethodnika 3NF.

*Relaciona šema R nalazi se u 2NF ako je svako neključno obeležje od R potpuno zavisno od kandidata ključa.*

Iz ove definicije jasno je da relacija koja se nalazi u 2NF mora biti i u 1NF. Sva neključna obeležja relacije u 2NF moraju biti funkcionalno zavisna od ključa relacije (uslov za 1NF), i ta funkcionalna zavisnost mora biti potpuna (dodatni uslov).

Specijalni slučaj ispunjenosti uslova 2NF je ako su u relaciji R sva obeležja ključna ili ako se svi kandidati za ključ sastoje samo od po jednog obeležja.

Neka relaciona šema  $R(A_1, A_2, \dots, A_n)$  nije u 2NF. Postoji takva dekompozicija relacione šeme R u skup relationalnih šema koje su u 2NF.

Dekompozicija relacije R izvodi se na osnovu ranije definisanih uslova za dekompoziciju bez gubitka informacija.

Za relacionu šemu  $R(A_1, A_2, \dots, A_n)$  koja nije u 2NF postoje podskupovi X i Y skupa obeležja  $\{A_1, A_2, \dots, A_n\}$  takvi da:

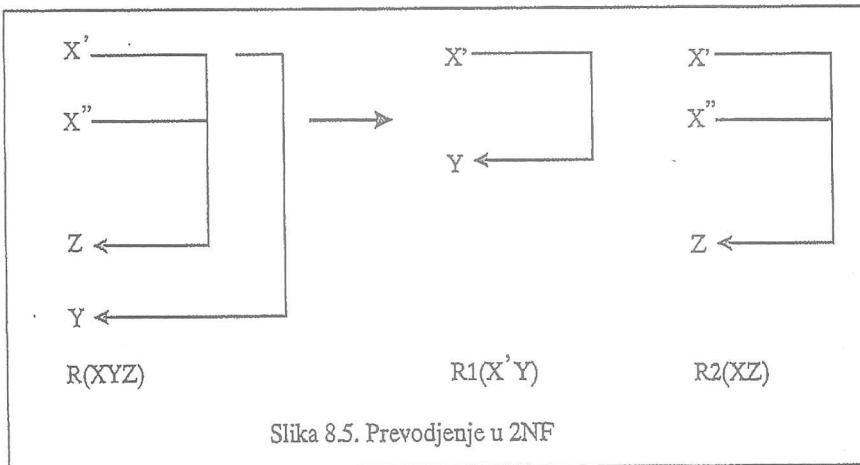
- Y nisu ključna obeležja,
- X je kandidat za ključ i
- $X \rightarrow Y$  je parcijalna FZ.

$X$  se mož predstaviti kao  $X = X' X''$  (kao unija skupova obeležja  $X'$  i  $X''$ ) gde je  $X' \rightarrow Y$  potpuna FZ. Sledi da je  $X' \subset X$ .

Neka je  $Z$  skup svih obeležja šeme relacije  $R$  koji nisu ni u  $X$  ni u  $Y$ .

Šemu relacije  $R(A_1, A_2, \dots, A_n) = R(XYZ)$  dekomponujemo (zamenjujemo) na šeme relacija  $R1(X'Y)$  i  $R2(XZ)$ .

Prevodjenje šeme relacije  $R(XYZ)$  u 2NF možemo ilustrovati grafički slikom 8.5.



Relacione šeme  $R1(X'Y)$  i  $R2(XZ)$  ispunjavaju ranije definisana dva uslova dekompozicije bez gubitka informacija:

1. Unija obeležja šema relacija  $R1(X'Y)$  i  $R2(XZ)$  jednaka je skupu obeležja polazne relacije  $R$ .
2. Šeme relacija  $R1(X'Y)$  i  $R2(XZ)$  sadrže zajedničko obeležje (ili obeležja)  $X'$  koje je kandidat za ključ u šemi relacije  $R1(X'Y)$ .

Šema relacije  $R1(X'Y)$  je u 2NF, a ako šema relacije  $R2(XZ)$  nije u 2NF, vršimo njenu dekompoziciju na isti način kako smo to uradili za  $R(XYZ)$ . Postupak je konačan jer svakom dekompozicijom dobijamo šeme relacija sa manjim brojem obeležja (šema relacije  $R1(X'Y)$  ne sadrži obeležja  $X''$  i  $Z$ , dok šema relacije  $R2(XZ)$  ne sadrži obeležje  $Y$  iz polazne relacije). U najgorem slučaju postupak će biti okončan time što će sva obeležja postati ključna ili će

se svi kandidati ključa sastojati od po samo jednog obeležja što smo ranije naveli kao specijalni slučaj ispunjenosti 2NF.

*Primer:* Neka imamo sledeću šemu relacije:

**NASTAVNIK\_PREDMET** (S\_NAS, PREZIME\_IME,  
S\_PRED, NAZIV, BR\_GR).

Jedini kandidat za ključ i ujedno primarni ključ šeme relacije **NASTAVNIK\_PREDMET** čine obeležja S\_NAS i S\_PRED (podvučena obeležja u šemi relacije). S obzirom na ranije definisanu ulogu ključa vrede sledeće FZ:

$S\_NAS\ S\_PRED \rightarrow PREZIME\_IME$   
 $S\_NAS\ S\_PRED \rightarrow NAZIV$   
 $S\_NAS\ S\_PRED \rightarrow BR\_GR$ .

Neključna obeležja su: PREZIME\_IME, NAZIV, BR\_GR.

U navedenom skupu FZ uočavamo (na osnovu dodatnih znanja iz realnog sveta) da su prve dve parcijalne jer vrede i sledeće FZ:

$S\_NAS \rightarrow PREZIME\_IME$   
 $S\_PRED \rightarrow NAZIV$ .

Postupak normalizacije razmotrićemo u odnosu na prvu parcijalnu FZ mada smo razmatranje normalizacije mogli započeti i sa drugom.

Zadatu relationalnu šemu dekomponujemo na osnovu ranije iznetih pravila na sledeće dve šeme relacije:

**NASTAVNIK** (S\_NAS, PREZIME\_IME),  
**N\_P** (S\_NAS, S\_PRED, NAZIV, BR\_GR)

Relacione šeme **NASTAVNIK** i **N\_P** sadrže zajedničko obeležje S\_NAS koje je kao što se vidi kandidat za ključ u relaciji nastavnik (tačnije primarni ključ) čime je ispunjen ranije definisan uslov da je dekompozicija reverzibilna odnosno da je bez gubitka informacija.

*Dragan Mihajlović*

Šema relacije NASTAVNIK je u 2NF jer se jedini kandidat za ključ sastoji od samo jednog obeležja, pa bilo kakva parcijalna FZ nije moguća.

U šemi relacije N\_P na osnovu osobina ključa vrede sledeće FZ:

$$\begin{aligned} S\_NAS \ S\_PRED &\rightarrow NAZIV \\ S\_NAS \ S\_PRED &\rightarrow BR\_GR. \end{aligned}$$

Takodje vredi i već ranije utvrđena parcijalna FZ:

$$S\_PRED \rightarrow NAZIV.$$

Relaciona šema N\_P nije u 2NF te pristupamo njenoj dekompoziciji na sledeće dve šeme relacije:

$$\begin{aligned} PREDMET (S\_PRED, NAZIV), \\ PREDAJE (S\_NAS, S\_PRED, BR\_GR). \end{aligned}$$

Relacija NASTAVNIK nalazi se u 2NF. Relacija PREDAJE nalazi se takodje u 2NF. Funkcionalna zavisnost S\_NAS S\_PRED → BR\_GR je potpuna.

Prevodjenje polazne relacije na relacije koje zadovoljavaju 2NF je ovim završeno.

### *Treća normalna forma*

Pretpostavimo sledeću situaciju: Jedan nastavnik zaposlen je samo na jednom fakultetu i svaki fakultet nalazi se samo u jednom mestu. Svaki fakultet može imati više zaposlenih nastavnika i u svakom mestu može biti više fakulteta.

Relaciona šema koja modelira prethodni opis glasi:

$$NFM (S\_NAS, PREZIME\_IME, FAKULTET, MESTO).$$

Vrede sledeće FZ:

$$\begin{aligned} S\_NAS &\rightarrow PREZIME\_IME, \\ S\_NAS &\rightarrow FAKULTET, \\ S\_NAS &\rightarrow MESTO, \\ FAKULTET &\rightarrow MESTO, \end{aligned}$$

Prethodna relaciona šema nalazi se u 2NF mada kao što ćemo dalje videti relacije nad njom pokazuju sve anomalije održavanja.

Anomalija dodavanja ogleda se u tome što podatke o mestu u kojem se nalazi neki fakultet nije moguće upisati sve dok taj fakultet nema bar jednog zaposlenog nastavnika. Obeležje S\_NAS je primarni ključ relacije i ne može imati nul-vrednost.

Anomalija brisanja sastoji se u tome što posle brisanja poslednjeg nastavnika nekog fakulteta gube se informacije o fakultetu kao i o lokaciji tog fakulteta, što je verovatno sasvim nepoželjni efekat.

Informacije o mestu u kojem se nalazi neki fakultet memorisane su u svakoj n-torci nastavnika koji radi na tom fakultetu. Ako se promeni lokacija fakulteta treba menjati sve n-torce nastavnika koji rade na tom fakultetu (promena podataka o MESTU). Nezavisne promene polja MESTO za pojedine n-torce mogu dovesti do remećenja FZ FAKULTET → MESTO. Prema tome relacija NFM poseduje i anomaliju menjanja podataka.

Razlog za postojanje opisanih anomalija u relaciji NFM je već navedena FZ NASTAVNIK → MESTO koju nazivamo i *tranzitivna zavisnost* jer proizilazi iz zavisnosti S\_NAS → FAKULTET, FAKULTET → MESTO.

Dekompozicijom relacione šeme NFM(S\_NAS, PREZIME\_IME, FAKULTET, MESTO) na relacione šeme NF(S\_NAS, PREZIME\_IME, FAKULTET) i FM(FAKULTET, MESTO) dobili smo dve relacione šeme koje ispunjavaju uslove dekompozicije bez gubitka informacija. Relacione šeme NF i FM nalaze se u 2NF, ali kao što ćemo kasnije moći da proverimo i u 3NF i ne poseduju prethodno navedene anomalije održavanja.

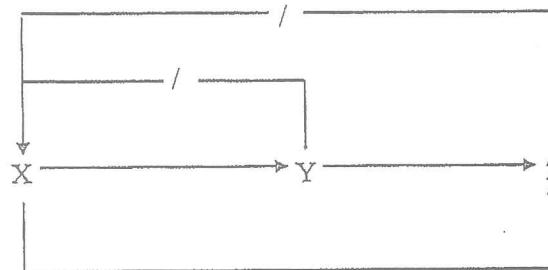
Pošto se definicija 3NF zasniva na *tranzitivnoj zavisnosti* daćemo prvo definiciju tranzitivne zavisnosti.

Neka su X, Y i Z skupovi obeležja. Kažemo da Z tranzitivno zavisi od X ako i samo ako su ispunjeni sledeći uslovi:

$$X \rightarrow Y, Y \rightarrow X, Y \rightarrow Z$$

gde FZ  $Y \rightarrow Z$  nije parcijalna.

Prethodnog sledi  $X \rightarrow Z$ ,  $Z \rightarrow X$ , dok  $Z \rightarrow Y$  nije niti uslov niti je zabranjeno. Ove zavisnosti se mogu ilustrovati grafički sledećim dijagramom:



Parcijalna FZ je poseban slučaj tranzitivne zavisnosti. Neka je  $R(X, A_1, A_2, \dots, A_n)$  relaciona šema. Neka je  $X'$  podskup obeležja u  $R$  za koji vredi  $X' \subset X$ , i neka vredi  $X \rightarrow A_1, \dots, A_n$  i  $X' \rightarrow A_m$ . Zbog refleksivnosti (jedan od aksioma za FZ: svaki skup obeležja funkcionalno određuje svaki svoj podskup) vredi  $X \rightarrow X'$ , a iz zavisnosti  $X \rightarrow X'$  i  $X' \rightarrow A_m$  proizilazi da je zavisnost  $X \rightarrow A_m$  tranzitivna funkcionalna zavisnost.

*Relaciona šema R nalazi se u 3NF ako je u 1NF i ako ni jedno neključno obeležje u R nije tranzitivno zavisno od ključa od R.*

3NF podrazumeva ispunjenost 1NF i 2NF. Prethodna definicija 3NF eksplicitno zahteva da se relacija nalazi samo u 1NF. S obzirom da je parcijalna funkcionalna zavisnost poseban slučaj tranzitivne zavisnosti, iz uslova da neključna obeležja relacione šeme u 3NF ne smeju biti tranzitivno zavisna od ključa, proizilazi da se relaciona šema u 3NF mora nalaziti i u 2NF.

Neka relaciona šema  $R(A_1, A_2, \dots, A_n)$  nije u 3NF. Postoji takva dekompozicija relacione šeme R u skup relacionih šema koje su u 3NF.

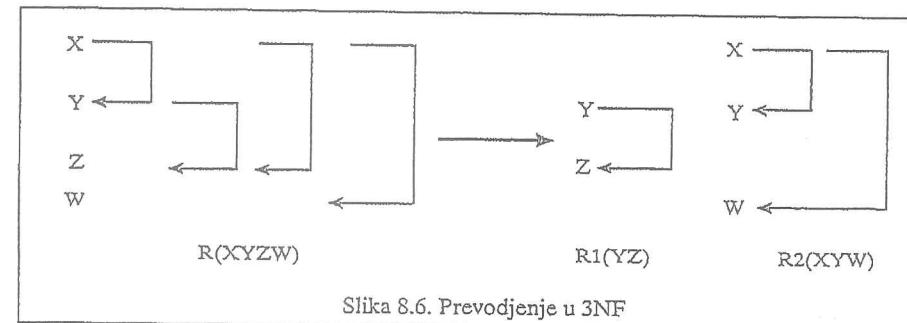
Za relacionu šemu  $R(A_1, A_2, \dots, A_n)$  koja nije u 3NF postoje podskupovi X, Y i Z skupa obeležja  $\{A_1, A_2, \dots, A_n\}$  takvi da:

- Y i Z nisu ključna obeležja,
- X je kandidat za ključ i vredi  $X \rightarrow Y$  i  $Y \rightarrow X$ ,
- $Y \rightarrow Z$  je potpuna FZ.

Neka je W skup svih obeležja šeme relacije R koji nisu ni u X ni u Y ni u Z.

Šemu relacije  $R(A_1, A_2, \dots, A_n) = R(XYZW)$  dekomponujemo (zamenjujemo) na šeme relacija  $R1(YZ)$  i  $R2(XYW)$ .

Prevodjenje šeme relacije  $R(XYZW)$  u 3NF možemo ilustrovati grafički slikom 8.6.



Relacione šeme  $R1(YZ)$  i  $R2(XYW)$  ispunjavaju uslove dekompozicije bez gubitka informacija.

Ukoliko relaciona šema  $R1(YZ)$  ili  $R2(XYW)$  ili obe ne ispunjavaju uslove 3NF postupak normalizacije se na njima nastavlja.

*Primer:* Šemu relacije  $NFM(S\_NAS, PREZIME\_IME, FAKULTET, MESTO)$  u kojoj vrede sledeće funkcionalne zavisnosti:

$$\begin{aligned} S\_NAS &\rightarrow FAKULTET, \\ FAKULTET &\rightarrow MESTO, \\ MESTO &\rightarrow FAKULTET, \\ S\_NAS &\rightarrow MESTO \text{ tranzitivna FZ.} \end{aligned}$$

prevesti na oblik koji zadovoljava uslove 3NF.

Šema relacije  $NFM(S\_NAS, PREZIME\_IME, FAKULTET, MESTO)$  ispunjava uslove 1NF i 2NF, ali ne ispunjava uslove 3NF.

Na osnovu datog načina za dekompoziciju šeme relacije koja nije u 3NF, šemu  $NFM(S\_NAS, PREZIME\_IME, FAKULTET, MESTO)$  dekomponujemo na relacione šeme  $NF(S\_NAS, PREZIME\_IME, FAKULTET)$  i  $FM(FAKULTET, MESTO)$ . Šeme relacija  $NF(S\_NAS, PREZIME\_IME,$

FAKULTET) i FM(FAKULTET, MESTO) ispunjavaju uslove 3NF čime je postupak svodjenja na 3NF završen.

*Primer:* Prevesti na oblik koji zadovoljava uslove 3NF šemu relacije

$$R(JMBG\_DETETA, IME\_DETETA, JMBG\_MAJKE, IME\_MAJKE, JMBG\_OCA, IME\_OCA, MESTO\_BORAVKA\_OCA, POSTA\_BROJ).$$

u kojoj pored zavisnosti od ključa JMBG\_DETETA vrede i sledeće funkcionalne zavisnosti:

$$\begin{aligned} JMBG\_OCA &\rightarrow IME\_OCA \\ JMBG\_MAJKE &\rightarrow IME\_MAJKE \end{aligned}$$

Obeležja IME\_OCA, MESTO\_BORAVKA\_OCA, POSTA\_BROJ, i IME\_MAJKE su tranzitivno zavisna od ključa te prema tome relacija R ne ispunjava uslov 3NF.

Postoje sledeće dve tranzitivne zavisnosti:

$$\begin{aligned} JMBG\_DETETA &\rightarrow JMBG\_MAJKE \rightarrow IME\_MAJKE, \\ JMBG\_DETETA &\rightarrow JMBG\_OCA \rightarrow IME\_OCA \\ &\quad MESTO\_BORAVKA\_OCA \\ &\quad POSTA\_BROJ. \end{aligned}$$

Ako dekompoziciju relacione šeme R izvršimo uvažavajući drugu tranzitivnu zavisnost dobijamo sledeće šeme relacija:

$$\begin{aligned} R1(JMBG\_OCA, IME\_OCA, MESTO\_BORAVKA\_OCA, POSTA\_BROJ), \\ R2(JMBG\_DETETA, IME\_DETETA, JMBG\_OCA, JMBG\_MAJKE, IME\_MAJKE). \end{aligned}$$

Relaciona šema R1 nije u 3NF jer poseduje sledeću tranzitivnu zavisnost:  
 $JMBG\_OCA \rightarrow POSTA\_BROJ \rightarrow MESTO\_BORAVKA\_OCA$ .

Relacionu šemu R1 dekomponujemo na sledeće dve:

$$\begin{aligned} R11(POSTA\_BROJ, MESTO\_BORAVKA\_OCA), \\ R12(JMBG\_OCA, IME\_OCA, POSTA\_BROJ), \end{aligned}$$

## Glava 8. Relacioni model podataka

Relacione šeme R11 i R12 ispunjavaju uslove 3NF.

Relaciona šema R2 nije u 3NF jer poseduje sledeću tranzitivnu zavisnost:

$$JMBG\_DETETA \rightarrow JMBG\_MAJKE \rightarrow IME\_MAJKE.$$

Relacionu šemu R2 dekomponujemo na sledeće dve:

$$R21(JMBG\_MAJKE, IME\_MAJKE).$$

$$R22(JMBG\_DETETA, IME\_DETETA, JMBG\_OCA, JMBG\_MAJKE),$$

Relacione šeme R21 i R22 ispunjavaju uslove 3NF.

U ovom primeru polazeći od relacione šeme R i prve tranzitivne zavisnosti ( $JMBG\_DETETA \rightarrow JMBG\_MAJKE \rightarrow IME\_MAJKE$ ) postupkom normalizacije došli bi do istog rezultata.

### Rezime postupka normalizacije

Relacija čije sve domene sadrže samo atomarne vrednosti kažemo da je normalizovana, odnosno da se nalazi u 1NF. Prevodjenje relacione šeme u 2NF znači eliminisanje neključnih obeležja koja su funkcionalno zavisna od dela primarnog ključa. Konačno, prevodenjem relacione šeme u 3NF iz šeme su izbačena sva neključna obeležja koja su bila tranzitivno zavisna od ključa.

Neključna obeležja relacione šeme koja je u 3NF ispunjavaju sledeće uslove:

- funkcionalno su zavisna od primarnog ključa šeme,
- nisu funkcionalno zavisna od podskupa obeležja šeme koji nije kandidat za ključ odnosno nisu tranzitivno zavisni od primarnog ključa.

Neka je R relaciona šema, i neka su X i Y podskupovi obeležja u R s tim da je X ključ od R, a Y bilo koji podskup obeležja. Neka je A bilo koje neključno obeležje. S obzirom na to da je X ključ u R mora vredeti:

$$X \rightarrow Y$$

$$X \rightarrow A.$$

R se nalazi u 3NF ako vredi:

- a)  $Y \rightarrow A$ , ili  
 b)  $Y \rightarrow X$ .

U slučaju da vredi a) niti jedno neključno obeležje nije funkcionalno zavisno o bilo kom podskupu obeležja  $Y$ , koji nije ključ relacione šeme, i šema se nalazi u 3NF.

Ako vredi b), podskup obeležja  $Y$  je kandidat za ključ relacione šeme, i šema se nalazi u 3NF.

Ako  $Y$  nije kandidat za ključ od  $R$  i u  $R$  vredi  $Y \rightarrow A$ ,  $R$  nije u 3NF. Pri tome se javljaju sledeće dve mogućnosti:

- 1)  $Y \subset X$ , ili
- 2)  $Y \cap X = \emptyset$ .

U slučaju 1) obeležje  $A$  je parcijalno funkcionalno zavisno od ključa relacione šeme  $R$ . U slučaju 2) obeležje  $A$  je tranzitivno zavisno od ključa relacione šeme  $R$ .

Transformacija relacione šeme u 3NF ima veliki značaj za praksu, relativno je jednostavna i uvek izvodiva uz postizanje većine ciljeva normalizacije.

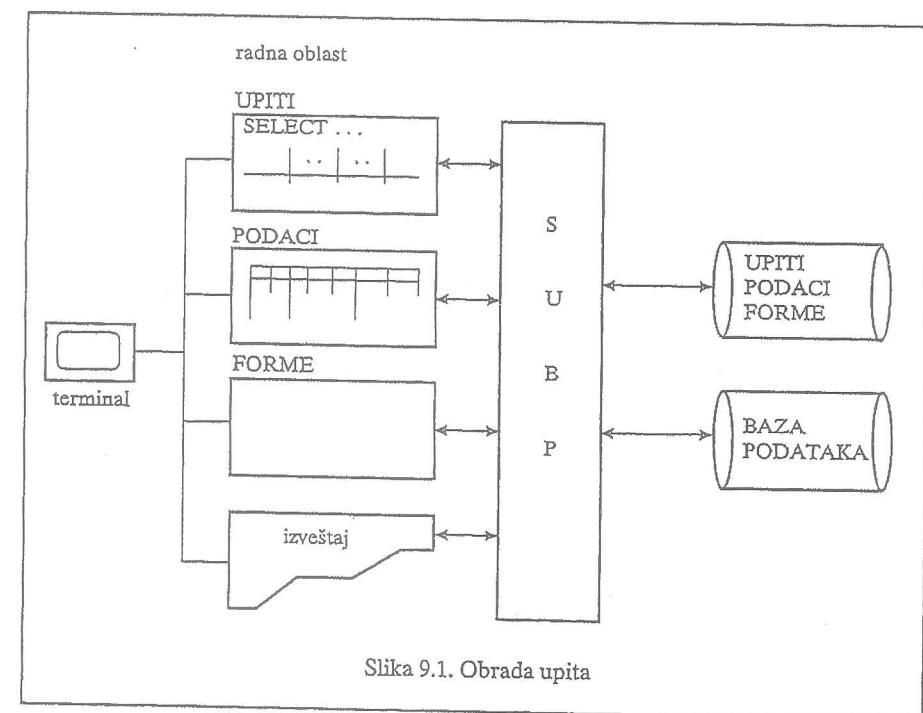
Kao nedostatak postupka vertikalne normalizacije dekompozicijom možemo navesti to da broj dobijenih šema relacija koje zadovoljavaju uslove 3NF ne mora biti najmanje moguć za datu nenormalizovanu šemu relacije. Za dobijeno rešenje ne znamo da li je optimalno ili postoji rešenje sa manjim brojem relacionih šema u 3NF. Ukupan broj relacionih šema u 3NF zavisi o prepostavljenom vlasniku nenormalizovane šeme relacije i izboru ključa. Kada dobijeni broj šema relacija nije optimalan to se može negativno odraziti na troškove održavanja i korišćenja baze podataka.

Kao drugi problem pomenimo da kod vertikalne normalizacije dekompozicijom može doći do gubitka zavisnosti što se može negativno odraziti na održavanje baze podataka.

## 9. JEZICI ZA MANIPULISANJE PODACIMA U RELACIONOM MODELU

### 9.1. Obrada upita

Kako bi korisnici mogli brzo i efikasno da koriste bazu podataka u SUBP se obično nalazi jedan skup programa namenjen obradi upita. Tipičan interaktivni način rada za obradu upita ilustrovan je na slici 9.1, a sastoji se iz sledećih koraka:



1. Koristeći neki od jezika za definisanje upita korisnik definiše upit. Takav upit memoriše se u privremenoj radnoj oblasti nazvanoj UPITI.
2. Zahtevanje izvršavanja upita iz oblasti UPITI. Rezultati upita memorišu se u drugoj privremenoj radnoj oblasti nazvanoj PODACI.
3. SUBP formira takozvanu *sistemsku formu* koju memoriše u privremenoj radnoj oblasti FORME i koristi za prikaz podataka iz oblasti PODACI.
4. Posle pregledanja rezultata iz prethodnog koraka korisnik može na ekran pozvati formu i izvršiti njenu redakciju.
5. Korisnik zahteva prikaz podataka iz oblasti PODACI prema formi definisanoj u oblasti FORME pri čemu nije neophodno ponovo izvršavanje upita.
6. Koraci 4 i 5 ponavljaju se sve dotele dok se ne dobije zadovoljavajući rezultat.
7. Zahtev za štampanjem rezultata.

UPITI, PODACI i FORME iz privremenih radnih memorijskih zona odgovarajućim naredbama mogu biti i trajno memorisani i ponovo naknadno korišćeni.

Upitni jezici za relacioni model podataka dele se u sledeće dve klase:

1. *Algebarski jezici* (zasnovani na relacionoj algebri) kod kojih se upiti izražavaju primenom posebnih operatora nad relacijama.

Primeri algebarskih upitnih jezika su:

ISBL,  
ASTRID.

2. *Jezici predikatskog računa* kod kojih se upitima opisuje skup n-torki rezultata navodeći predikate koje n-torke moraju da zadovolje.

Primeri jezika predikatskog računa:

QUEL (koristi se u SUBP INGRES),  
SQL (koristi se u više SUBP, npr. ORACLE, DB2),  
QBE (koristi se u SUBP DB2).

## 9.2. Standardni upitni jezik - SQL

SQL (Structured Query Language) je standardni relacioni upitni jezik. Njegov tvorac je Chamberlin, a nastao je 1974 godine u IBM-ovoj istraživačkoj laboratoriji (IBM Research Laboratory) u San Jose, Kalifornija na istom mestu gde je E.F. Codd 1970 godine definisao osnovne koncepte relacionog modela podataka.

SQL nije samo upitni jezik već predstavlja kompletan jezik podataka koji sadrži jezike i za: definiciju podataka, ažuriranje, kontrolu, konzistentnost, konkurentni rad i jezik za održavanje rečnika podataka. Ugradjen je u većinu komercijalno raspoloživih SUBP za personalne računare do velikih računara (npr. DB2, SQL/DS, ORACLE, INGRES, PROGRES). Na žalost, i pored toga što SQL u svom nazivu sadrži reč standardni postoje odredjene razlike u njegovoj realizaciji. No ipak znajući SQL moguće je na gotovo identičan način raditi sa bilo kojim od SUBP. Proizvodjači relacionih SUBP sa drugim upitnim jezicima prinudjeni su zbog svog opstanka na tržištu da u svojim SUBP omoguće i korišćenje SQL. Zbog navedene činjenice o velikim mogućnostima korišćenja u daljem tekstu ovog poglavlja biće opisana struktura i semantika većeg broja naredbi SQL sa napomenom da to ni u kom slučaju ne može u potpunosti da zameni odgovarajuću dokumentaciju konkretnog SUBP koja se odnosi na SQL.

SQL je korisnički orijentisan jezik. Uči se lako i brzo, a prethodno iskustvo u automatskoj obradi podataka nije neophodno. Osnovne karakteristike SQL su:

### 1. Jednostavnost i jednoobraznost pri korišćenju

Tabela se kreira jednom izvršnom naredbom i odmah po kreiranju spremna je za korišćenje (upisivanje, promena, ukidanje i pretraživanje podataka). Podaci se prikazuju u obliku tabele. Termin *tabela* u SQL-u predstavlja sinonimom za pojam *relacija* pa će se tako i ovde koristiti.

### 2. Mogućnos interaktivnog i klasičnog programiranja

Koristeći SQL dobijaju se odgovori na trenutno postavljene zahteve ili se SQL blokovi ugradjuju u neki viši programski jezik (FORTRAN, COBOL, PL/I, C, ...).

### 3. Neproceduralnost (tj. proceduralnost u minimalnom stepenu)

Ni za jedan jezik se ne može reći da je potpuno neproceduralan, već da je proceduralan u većoj ili manjoj meri. SQL-je u velikoj meri neproceduralan jer se njime definiše ŠTA se želi dobiti a ne KAKO se do rezultata dolazi: koji podaci se žele, koje tabele se referenciraju i koji uslovi treba da budu ispunjeni, bez specifikacije procedure za dobijanje podataka. Pravi smisao mere neproceduralnosti SQL čitalac će shvatiti posle upoznavanja samog jezika.

Za opis naredbi SQL koristimo sintaksnu notaciju sa zagradama sa sledećim elementima:

**Ekskluzivna disjunkcija** - Od više objekata unutar zagrada, medjusobno razdvojenih uspravnom crtom (| - uobičajeni znak eksluzivne disjunkcije) može se odabrat samo jedan.

Na primer, iskaz alfa ili beta ili gama definišemo na sledeći način:

{alfa | beta | gama}.

**Opcioni elementi** - su elementi koji se po želji mogu izostaviti tj. njihov broj pojavljivanja je 0 ili 1 puta. Označavaju se navodnjem u uglatim zagradama.

Na primer, definicija:

[alfa | beta | gama]

ukazuje da se na tom mestu može koristiti prazno, a ako se koristi nešto drugo onda to može i mora biti alfa ili beta ili gama.

**Ključne reči** - su terminalni simboli jezika i imaju ulogu reči separatora u naredbama. U definicijama naredbi napisane su velikim slovima, a pri upotrebi trebaju biti navedene u istom obliku pri čemu pisanje velikim ili malim slovima nema značaja. Ukoliko su navedene u uglatim zagradama mogu se izostaviti. Objekti navedeni velikim slovima sa podvlačenjem označavaju prepostavljene vrednosti tj. i bez navodjenja odredjeno je navedeno značenje. Reči separatore predstavljaju i '(', ')', ';', ',' i '|'. Uočavamo da znakovi '[', ']', '{', '}' i '[' ne predstavljaju separator te da se nikada ne navode u naredbama.

Na primer, definicija:

```
CREATE TABLE naziv_tabele
  (naziv_obl, tip_podataka [NULL | NOT NULL]
  [,naziv_obl, tip_podataka [NULL | NOT NULL]]...)
  [drugi_parametri];
```

može biti zamenjena naredbom

```
CREATE TABLE NASTAVNIK
  (S_NAS          NUMBER(3)      NOT NULL,
   PREZIME_IME    CHAR(10)       NOT NULL,
   ZVANJE         CHAR(6),
   S_DIR          NUMBER(3),
   DATZAP        DATE,
   PLATA          NUMBER(8,2)     NOT NULL,
   DODATAK        NUMBER(8,2));
```

u kojoj su ključne reči: CREATE, TABLE, NULL, NOT NULL, (, ) i ;.

Od konstrukcije [NULL | NOT NULL] može biti navedeno NULL ili NOT NULL, pri čemu ako se ništa ne navede (to je moguće zbog uglatih zagrada) smatra se kao da je navedeno NULL (prepostavljena vrednost označena podvlačenjem).

Minimalno u datom primeru naredba mora glasiti (naredba sa obaveznim elementima):

```
CREATE TABLE naziv_tabele
  (naziv_obl tip_podataka);
```

**Promenljive reči** - Objekti navedeni malim slovima trebaju biti zamenjeni konkretnim značenjima odabranim od strane korisnika. Skup objekata koji mogu biti imenovani od strane korisnika i pravila za izbor imena su specifičnosti SUBP. Kao što je ranije navedeno preporuka pri izboru imena je da ona podesećaju (asociraju) na značenja iz realnog sveta, čime se olakšava komunikacija i pisanje aplikacije.

Na primer, u definiciji iz prethodnog primera promenljiva naziv-tabele zamenjena je sa NASTAVNIK, obli sa S\_NAS, tip\_podataka sa NUMBER(13) itd.

*Ponavljanje prethodnog objekta* nula, jednom ili više puta označava se tako da se iza navedenog objekta napišu tri tačke (...).

Na primer, deo definicije iz prethodnog primera

[,naziv\_obl, tip\_podataka [NULL | NOT NULL]]...

zamenjen je sa:

PREZIME_IME	CHAR(10)	NOT NULL,
ZVANJE	CHAR(6),	
S_DIR	NUMBER(3),	
DATZAP	DATE,	
PLATA	NUMBER(8,2)	NOT NULL,
DODATAK	NUMBER(8,2)	

U SQL sledeći objekti mogu biti imenovani: tabele, pogledi, sinonimi, kolone, indeksi i korisničke promenljive.

Pravila za izbor imena su:

- dužina imena može biti od 1 do 30 simbola,
- prvi simbol u imenu mora biti slovo,
- ime promenljive sadrži A-Z, a-z, 0-9, \_, \$ i #.
- ime promenljive ne sme biti rezervisana (službena) reč u SUBP.

Primeri ispravnih imena:

STUDENT, PREDMET, ul.\_plata.

Primeri neispravnih imena:

25maj, "zvezada", ULAZ-IZLAZ.

Uobičajeno je da ime promenljive sadrži više od jednog simbola. Pogodno je imena birati tako da ona podsećaju na značenje iz realnog sveta, čime se olakšava pisanje aplikacije.

## 9.2.1. Naredbe za definisanje podataka

### 9.2.1.1. Kreiranje nove tabele - CREATE TABLE

Naredba CREATE TABLE se koristi za kreiranje nove tabele (bazna tabela, bazna relacija). Potrebno je navesti naziv tabele i za svako obeležje (kolona tabele) naziv, tip, dužinu i da li su NULL vrednosti dozvoljene.

Opšti oblik naredbe je:

```
CREATE TABLE naziv_tabele
  (naziv_obl, tip_podataka [NULL | NOT NULL]
   [,naziv_obl, tip_podataka [NULL | NOT NULL]]...)
   [drugi_parametri];
```

gde je: naziv\_obl - naziv i-tog obeležja tabele,  
tip\_podataka - tip i dužina podatka obeležja koje prethodi,  
drugi\_parametri - opis memorijskog prostora, klastera ili upita na osnovu kojeg može biti definisana tabela i preneti podaci.

Pri kreiranju tabele ili izmeni definicije tabele mora se definisati tip podatka svakog obeležja. Osnovni tipovi podataka u SQL su:

**karakter** CHAR(*dužina*) - alfanumerički podaci (velika ili mala slova, decimalne cifre 0-9 i specijalni znaci kao npr. +, -, \$ itd.). Broj znakova određen je parametrom *dužina*.

**datum** DATE - za datum postoji više različitih formi pri čemu je prepostavljena (standardna) forma DD-MON-YY.

**numeric** NUMBER (*dužina*) - za numeričke podatke dužine *dužina*.  
NUMBER (*dužina.dec*) - za numeričke podatke dužine *dužina* sa *dec* cifara posle decimalne tačke.

**Primer:** Kreirati tabele sledećih šema relacija:

NASTAVNIK (S\_NAS, PREZIME\_IME, ZVANJE, S\_DIR,  
DATZAP, PLATA, DODATAK),  
PREDMET (S\_PRED, NAZIV),  
PREDAJE (S\_NAS, S\_PRED, BR\_GR).

Rešenje:

```

CREATE TABLE NASTAVNIK
  (S_NAS          NUMBER(3)      NOT NULL,
   PREZIME_IME    CHAR(10)       NOT NULL,
   ZVANJE         CHAR(6),
   S_DIR          NUMBER(3),
   DATZAP        DATE,
   PLATA          NUMBER(8,2)     NOT NULL,
   DODATAK        NUMBER(8,2));
CREATE TABLE PREDMET
  (S_PRED          NUMBER(3)      NOT NULL,
   NAZIV           CHAR(30)       NOT NULL);
CREATE TABLE PREDAJE
  (S_NAS          NUMBER(3)      NOT NULL,
   S_PRED          NUMBER(3)      NOT NULL,
   BR_GR          NUMBER(2)      NOT NULL);

```

Zadata relaciona šema nije optimalna sa stanovišta ažuriranja i izveštavanja jer tabela NASTAVNIK može sadržavati NULL vrednosti nekih obeležja što kao što smo rekli i kao što ćemo videti komplikuje operacijsku komponentu. Bolje rešenje za prethodni primer je da na primer, obeležje DODATAK ima opciju NOT NULL.

S obzirom da će navedena relaciona šema poslužiti za kasniju ilustraciju i drugih naredbi, korišćenjem obeležja DODATAK u tabeli NASTAVNIK sa NULL vrednostima pokazaće nam do kakvih sve problema pri korišćenju dovodi postojanje takvog obeležja.

Za obeležja koja nemaju klauzule NULL unapred definisana vrednost je NULL što znači da pri dodavanju n-torce u tabelu ista ne moraju dobiti vrednost tj. imaju neodredjenu vrednost. Neodredjena vrednost koristi se za predstavljanje nepoznate vrednosti ili neprimenljivog svojstva neke n-torce. Obeležja definisana kao NOT NULL prilikom dodavanja n-torki u tabelu moraju dobiti vrednost.

Može se uočiti veliki nedostatak naredbe CREATE TABLE koji se odnosi na nemogućnost definisanja primarnog ključa tabele. Klauzula NOT NULL ne obezbeđuje jedinstvenost vrednosti nekog obeležja ili grupe obeležja pa ne može zameniti ulogu ključa. U SQL-u uloga ključa se može nadomestiti

formiranjem jedinstvenog indeksa nad kombinacijom obeležja koje čine ključ. Nad jednom tabelom može se definisati više ključeva.

### 9.2.1.2. Kreiranje pogleda - CREATE VIEW

Pogled je jedan "prozor" kroz koji se vide podaci baze podataka. Pogled nema svoje posebne podatke pa je zato virtualna tabela sa kojom se radi gotovo kao i sa baznom tabelom. Pogled se koristi kao bilo koja druga tabela pri izveštavanju dok pri ažuriranju baze podataka preko pogleda postoje niz ograničenja.

Razlozi za korišćenje pogleda su:

- pojednostavljuje korišćenje baze podataka,
- tajnost - realizuje kontrolu pristupa podacima,
- performanse - čuva se u kompajliranom obliku,
- nezavisnost podataka - menjaju se definicije pogleda, a ne aplikacioni programi koji koriste podatke baze podataka preko pogleda.

Opšti oblik naredbe za kreiranje pogleda je:

```

CREATE VIEW naziv_pogleda
  [(naziv_obl, [ , naziv_obl ]...)]
  AS podupit
  [WITH CHECK OPTION];

```

gde je: naziv\_obl - naziv i-tog obeležja (kolone) pogleda,  
 podupit - ispravna SELECT naredba bez operatora UNION, bez  
 klauzule ORDER BY.

*Primer:* Kreirati pogled

NASTAVNIK\_PL (PREZIME\_IME, PLATA)

"kroz koji se vide" podaci o nastavnicima koji imaju zvanje  
 DOCENT.

Rešenje:

```
CREATE VIEW NASTAVNIK_PL (PREZIME_IME, PLATA)
AS SELECT PREZIME_IME, PLATA
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

U pogledu NASTAVNIK\_PL dva obeležja PREZIME\_IME i PLATA odgovaraju obeležjima PREZIME\_IME i PLATA bazne tabele NASTAVNIK. Ukoliko imena obeležja pogleda u naredbi CREATE VIEW nisu navedena, tada pogled dobija ista obeležja iz podupita navedenog iza SELECT. U našem primeru to bi bila obeležja: PREZIME\_IME i PLATA. Sva imena obeležja pogleda trebaju biti specificirana u sledećim slučajevima:

- a) kada se neko obeležje pogleda dobija primenom standardnih funkcija, aritmetičkog izraza ili konstante, pa prema tome nema obeležja bazne tabele od kojeg bi moglo biti nasledjeno.
- b) kada bi dva ili više obeležja imala isto ime.

*Primer:* Kreirati pogled R\_P (S\_PRED, UK\_GRUPA) u kojem je obeležje S\_PRED, šifra predmeta, a UK\_GRUPA predstavlja ukupan broj grupa predavanja.

Rešenje:

```
CREATE VIEW R_P (S_PRED, UK_GRUPA)
AS SELECT S_PRED SUM(BR_GR)
FROM PREDAJE
GROUP BY S_PRED;
```

U navedenom primeru nema takvog obeležja koje bi se moglo naslediti za drugo obeležje pogleda R\_P imenovano UK\_GRUPA, te su zbog toga imena obeležja pogleda morala biti navedena. Može se uočiti da pogled R\_P ne predstavlja prost podskup n-torki tabele PREDAJE već jednu izvedenu statističku tabelu.

S obzirom da se pogled može definisati pomoću bilo kog dozvoljenog podupita i da se upiti mogu definisati nad pogledima, to se i pogled može definisati nad pogledom.

Klauzula WITH CHECK OPTION (sa proverom) ukazuje da se operacije UPDATE i INSERT nad tim pogledom trebaju izvršavati sa proverom uslova definisanog u SELECT klauzuli.

Brisanje definicije pogleda ima sledeći opšti oblik:

```
DROP VIEW naziv_pogleda;
```

Izvršavanjem naredbe DROP VIEW iz baze podataka se briše definicija navedenog pogleda kao i svih pogleda koji su na osnovu navedenog definisani.

U jeziku SQL nema naredbe ALTER VIEW. Izvršavanje naredbi ALTER TABLE kojom se dodaju obeležja bazne tabele nemaju uticaja na do tada definisane poglede nad tom tabelom osim u slučaju da su kolone pogleda dobijene iz svih kolona podupita (SELECT \* FROM...) kada pogled više neće funkcionisati korektno.

### 9.2.1.3. Promena strukture tabele nakon što je kreirana - ALTER TABLE

Naredba ALTER TABLE omogućava da:

- 1) Dodamo s desna nova obeležja postojećoj tabeli;
- 2) Promenimo opis obeležja koje već postoji u tabeli.

Dodavanje s desna novih obeležja (kolona) postojećoj tabeli izvodi se naredbom sledećeg opštег oblika:

```
ALTER TABLE naziv_tabele
ADD (naziv_obi; tip_podatka [NULL | NOT NULL]
[,naziv_obii+1; tip_podatka [NULL | NOT NULL]]...);
```

*Primer:* U tabelu PREDMET dodati obeležje SEMESTAR.

Rešenje:

```
ALTER TABLE PREDMET
ADD (SEMESTAR NUMBER (2));
```

Sve postojeće n-torce tabelle PREDMET proširuju se sa četvrtim poljem koje ima neodredjenu vrednost (NULL). U naredbi ALTER TABLE specifikacija NOT NULL dozvoljena je samo ako tabela nema nijedne n-torce (prazna tabela). Ako se u tabelu koja sadrži n-torce treba dodati obeležje koje treba biti opisano kao NOT NULL moraju se izvršiti sledeće operacije:

1. Dodati novo obeležje u postojeću tabelu sa klauzulom NULL.
2. Za sve n-torce tabele upisati konkretne (definisane) vrednosti novog obeležja.
3. Izvršiti izmenu definicije sada postojećeg obeležja navodjenjem klauzule NOT NULL.

Izmena definicije postojećeg obeležja tabelle (tip, dužina, NULL, NOT NULL) može se izvršiti naredbom sledećeg opštег oblika:

```
ALTER TABLE naziv_tabele
    MODIFY (naziv_obi, tip_podataka [NULL | NOT NULL]
            [,naziv_obi, tip_podataka [NULL | NOT NULL]]...);
```

*Primer:* U tabeli NASTAVNIK povećati broj znakova obeležja PREZIME\_IME na 35 znakova.

Rešenje:

```
ALTER TABLE NASTAVNIK
    MODIFY (PREZIME_IME CHAR (35));
```

#### 9.2.1.4. Brisanje tabelle iz baze podataka - **DROP TABLE**

Brisanje definicije tabelle, zajedno sa podacima koje sadrži može se uraditi naredbom DROP TABLE sledećeg oblika:

```
DROP TABLE naziv_tabele;
```

Brišanjem definicije tabelle automatski se brišu definicije indeksa definisanih nad tom tabelom, dok se definicije pogleda ne brišu ali postaju neupotrebljive. Definicije pogleda možemo takodje brisati ili ih redefinisati.

*Primer:* Brisati sledeće definicije tabela i same tabele iz baze podataka: NASTAVNIK, PREDMET i PREDAJE.

Rešenje:

```
DROP TABLE NASTAVNIK;
DROP TABLE PREDMET;
DROP TABLE PREDAJE;
```

#### 9.2.1.5. Indeksi

Indeksi se koriste iz sledećih razloga:

- Obezbedjivanje bržeg pristupa po kolonama koje se indeksiraju bez čitanja cele tabelle,
- Obezbedjivanje jedinstvenosti vrednosti kolona koje čine indeks (kolone koje čine indeks mogu imati ulogu primarnog ključa).

Opšti oblik naredbe za kreiranje indeksa je:

```
CREATE [UNIQUE] INDEX naziv_indeksa
    ON naziv_tabele (naziv_obi, [ASC | DESC]
                      [,naziv_obi, [ASC | DESC]]...)
    [drugi_parametri];
```

Po jednoj tabeli može se kreirati više indeksa. Svaki indeks povećava vreme potrebno za održavanje tabelle. Svaki indeks mora imati jedinstveno ime. Klauzule ASC | DESC određuju rastući ili opadajući redosled uredjenja indeksa. Klauzula UNIQUE određuje da tabela ne može sadržavati dve n-torce sa istim sadržajem obeležja koja čine indeks. Ovim mehanizmom se može obezrediti uloga primarnog ključa. Ako klauzula UNIQUE nije navedena mogu postojati dve ili više n-torki sa istim vrednostima indeksa.

Posle izdavanja naredbe za kreiranje indeksa on se automatski održava. Zahtev za kreiranja UNIQUE indeksa nad tabelom koja sadrži podatke koji ne ispunjavaju uslov jedinstvenosti neće biti izvršen.

Indeks se briše naredbom:

```
DROP INDEX naziv_indeksa;
```

*Primer:* Kreirati indeks nad tabelom NASTAVNIK po obeležju PREZIME\_IME.

Rešenje: CREATE INDEX ABC\_IME  
ON NASTAVNIK (PREZIME\_IME);

Indeks ABC\_IME omogućava pristup podacima tabele NASTAVNIK po abecedi obeležja PREZIME\_IME. Mogu postojati više nastavnika sa istim prezimenom i imenom.

*Primer:* Kreirati indeks tabele NASTAVNIK koji može imati ulogu primarnog ključa.

Rešenje: CREATE UNIQUE INDEX NASTAVNIK\_KEY  
ON NASTAVNIK (S\_NAS);

## 9.2.2. Naredbe za manipulisanje podacima

### 9.2.2.1. Dodavanje novih n-torki u tabelu - *INSERT*

Opšti oblik INSERT naredbe je:

```
INSERT INTO naziv_tabele [(naziv_obl, [, naziv_obl2]...)]  
{VALUES (vrednost_obl, [,vrednost_obl2]...) | podupit};
```

gde se podupitom selektuju n-torki koje će biti upisane u tabelu.

Razmotrimo sledeće tipove insert naredbi:

1. Unos vrednosti za *sva* obeležja jedne n-torke,
2. Unos vrednosti *nekih* obeležja jedne n-torke,
3. Prepisivanje podataka iz jedne tabele u drugu.

1. Naredba INSERT za unos vrednosti *svih* obeležja jedne n-torke tabele ima sledeći opšti oblik:

```
INSERT INTO naziv_tabele  
VALUES (vrednost_obl, [,vrednost_obl2]...);
```

Za svako obeležje tabele definisano naredbom CREATE ili ALTER mora biti upisana vrednost, pri čemu je NULL dozvoljena opcija za svako obeležje koje nije NOT NULL. U listi vrednosti obeležja, vrednost\_obi odgovara i-tom obeležju u listi obeležja tabele.

*Primer:* Potrebno je dodati podatke u tabelu NASTAVNIK za PETRA PETROVIĆA sa šifrom 002, sa zvanjem DOCENTA koji radi od 1 februara 1982, prima platu od 11500 dinara, nema dodatka i kome je rukovodilac nastavnik sa šifrom 001.

Rešenje:  
INSERT INTO NASTAVNIK VALUES (002,  
'PETROVIĆ PETAR', 'DOCENT', 001,'01-FEB-82', 11500,NULL);

*Primer:* Dodati u tabelu NASTAVNIK i sledeće podatke.

```
INSERT INTO NASTAVNIK VALUES (003, 'PEIĆ PETAR',  
'DOCENT',005,'01-MAR-83',11500,NULL);  
INSERT INTO NASTAVNIK VALUES (004,'SIMIĆ SIMA',  
'DOCENT',005,'01-APR-84',11500,NULL);  
INSERT INTO NASTAVNIK VALUES (005,'ILIĆ JOVAN',  
'DOCENT',005,'01-MAY-85',11500,NULL);  
INSERT INTO NASTAVNIK VALUES (006,'SAVIĆ ILIJA',  
'V PROF',001,'01-JUN-75',12500,1000);  
INSERT INTO NASTAVNIK VALUES (007,'TOT ANA',  
'V PROF',001,'01-AUG-75',12500,1000);  
INSERT INTO NASTAVNIK VALUES (008,'SAVIĆ MILAN',  
'R PROF',001,'01-SEP-65',13500,2000);  
INSERT INTO NASTAVNIK VALUES (001,'RADOVIĆ NIKOLA',  
'R PROF',NULL,'01-OCT-60',14500,4000);
```

*Primer:* Dodati u tabelu PREDMET sledeće podatke.

S_PRED	NAZIV	SEMESTAR
001	INFORMACIONI SISTEMI	08
002	STRUKTURE I BP	03
003	OSNOVE RAČUNARSTVA	01
004	TEHNIKE PROGRAMIRANJA	08
005	PROGRAMIRANJE RS	02

Rešenje:

```
INSERT INTO PREDMET
    VALUES (001, 'INFORMACIONI SISTEMI', 08);
INSERT INTO PREDMET
    VALUES (002, 'STRUKTURE I BP', 03);
INSERT INTO PREDMET
    VALUES (003, 'OSNOVE RAČUNARSTVA', 01);
INSERT INTO PREDMET
    VALUES (004, 'TEHNIKE PROGRAMIRANJA', 08);
INSERT INTO PREDMET
    VALUES (005, 'PROGRAMIRANJE RS', 02);
```

*Primer:* Dodati u tabelu PREDAJE podatke koji odgovaraju sledećem stanju:

PREZIME_IME	NAZIV	BR_GR
RADOVIĆ NIKOLA	Osnove računarstva	2
RADOVIĆ NILOLA	Programiranje RS	1
PETROVIĆ PETAR	Informacioni sistemi	1
PEIĆ PETAR	Programiranje RS	1
ILIĆ JOVAN	Strukture i BP	1

Rešenje:

```
INSERT INTO PREDAJE VALUES (001,003,2);
INSERT INTO PREDAJE VALUES (001,005,1);
INSERT INTO PREDAJE VALUES (002,001,1);
INSERT INTO PREDAJE VALUES (003,005,1);
INSERT INTO PREDAJE VALUES (005,002,1);
```

Ukoliko želimo da unesemo vrednosti za samo neka obeležja jedne n-torke tabele koristi se naredba INSERT sledećeg opštег oblika:

```
INSERT INTO naziv_tabele (naziv_obl1, [naziv_obl2]...)
    VALUES (vrednost_obl1, [vrednost_obl2]...);
```

U listi vrednosti obeležja, vrednost\_obl<sub>i</sub> odgovara i-tom obeležju iz liste naziva obeležja. Uočimo da se sva NOT NULL obeležja moraju nalaziti u listi

naziva obeležja, što znači da se u listi vrednosti moraju naći vrednosti svih NOT NULL obeležja. Redosled obeležja s leva na desno ne mora odgovarati redosledu obeležja navedenih u naredbi CREATE (ili ALTER).

*Primer:* Potrebno je dodati u tabelu NASTAVNIK podatke za PETRIĆ JANKA sa šifrom 009, sa zvanjem redovni profesor koji radi od 1 novembra 1961 godine, ima platu od 13500 dinara, dodatak i njegov rukovodilac još nisu određeni.

Rešenje:

```
INSERT INTO NASTAVNIK
    (S_NAS, PREZIME_IME, DATZAP, ZVANJE, PLATA)
    VALUES (009, 'PETRIĆ JANKO', '01-NOV-61', 'R PROF', 13500);
```

Prepisivanje podataka iz jedne tabele u drugu izvodi se naredbom sledećeg opštег oblika:

```
INSERT INTO naziv_tabele [(naziv_obl1, naziv_obl2...)]
    podupit;
```

gde podupit definiše n-torke koje se selektiraju i dodaju u tabelu (naredba SELECT);

### 9.2.2.2. Pretraživanje relacione baze podataka - SELECT

Opšti oblik naredbe SELECT u SQL je:

```
SELECT [ALL | DISTINCT] { * | {naziv_tabele.*} | element_selekcijske } [alias]
    [{,naziv_tabele.*} | element_selekcijske ] [alias] ... }
    FROM naziv_tabele [alias] [, naziv_tabele [alias]] ...
    [WHERE uslov]
    [CONNECT BY uslov [START WITH uslov]]
    [GROUP BY naziv_obl1, [naziv_obl2]... [HAVING uslov]]
    [{UNION | INTERSECT | MINUS} SELECT ...]
    [ORDER BY { naziv_obl1, pozicija} [ASC|DESC]
        [, {naziv_obl2} pozicija] [ASC|DESC]]...]
    [FOR UPDATE OF kolona, [,kolona2] ... [NOWAIT]];
```

gde je:	naziv_tabele	- naziv tabele ili naziv pogleda,
	element_selekcijske	- obeležje, funkcija, izraz,
	alias	- privremeno ime (sinonim, drugo ime),
	naziv_obeležja	- obeležje tabele,
	uslov	- validan uslov,
	pozicija	- relativna pozicija kolone u SELECT listi,
	kolona	- kolona tabele iz FROM klauzule.

Prepostavljena vrednost ALL će kao rezultat dati sve n-torce koje su rezultat pretraživanja. Klauzulom DISTINCT iz prikaza će biti isključene identične n-torce.

Svaki *element-selekcijske* je ime jedne kolone u prikazu rezultata, dok *tabela.\** postaje skup kolona (po jedna kolona za svaku kolonu iz *tabele*) u redosledu koji je definisan pri kreiranju te tabele.

Karakter \* koji se može pojaviti samostalno je ekvivalentan sa *tabela.\** za svaku tabelu koja se pojavljuje iza FROM u redosledu navodjenja.

FROM *naziv\_tabele* definiše tabelu ili pogled iz koje se pretražuju podaci. Ako je navedeno više od jednog naziva tabele mora se izvršiti operacija spoj.

WHERE definiše *uslov* za pretraživanje n-torki i/ili spoj tabela.

CONNECT BY se koristi za prikaz rezultata kada su n-torce tabele strukturirane u stablo (kada jedna n-torka referencira drugu n-torku u istoj tabeli).

GROUP BY i HAVING se koristi za prikaz sumarnih rezultata grupe n-torki koje imaju istu vrednost u jednom ili više obeležja.

Operatori {UNION | INTERSECT | MINUS} SELECT ... se koristi za kombinovanje rezultata dve SELECT naredbe u jednu tabelu.

ORDER BY definiše redosled n-torki rezultata.

FOR UPDATE OF se koristi za zaključavanje selektiranih n-torki tabele. Naredbu SELECT ... FOR UPDATE OF normalno sledi jedna ili više naredbi UPDATE, INSERT ili DELETE. Sve do naredbe COMMIT ili ROLLBACK ostali korisnici nemaju pristupa zaključanim n-torkama tabele.

Obavezni elementi SELECT naredbe su:

```
SELECT {*} | {naziv_tabele.* | element_selekcijske}
      FROM naziv_tabele;
```

Rezultat operacije SELECT je nova tabela koja se na neki način formira iz zadatih tabela baze podataka, ali koja nema svog imena. S obzirom na to moguće je primeniti drugu operaciju SELECT na rezultat jedne operacije SELECT. To opet znači da operacije SELECT mogu biti uložene (ugradjene) jedna u drugu.

### 9.2.2.2.1. Pretraživanje jedne tabele uz prikaz neizmenjenog sadržaja

Prikaz kompletног sadržaja tabele - SELECT \*

Primer: Prikazati kompletan sadržaj tabele: PREDMET, NASTAVNIK, i PREDAJE.

Rešenje:

```
SELECT *
      FROM PREDMET;
```

PREDMET.NAZIV	SEMESTAR
1 INFORMACIONI SISTEMI	8
2 STRUKTURE I BP	3
3 OSNOVE RACUNARSTVA	1
4 TEHNIKE PROGRAMIRANJA	8
5 PROGRAMIRANJERS	2

```
SELECT *
      FROM NASTAVNIK;
```

S_NAS	PREZIME_IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
2	PETROVIĆ PETAR	DOCENT	6	01-FEB-82	11500	
3	PEIĆ PETAR	DOCENT	6	01-MAR-83	11500	
4	SIMIĆ SIMA	DOCENT	6	01-APR-84	11500	
5	I利Ć JOVAN	DOCENT	6	01-MAJ-85	11500	
6	SAVIĆ ILIJA	V PROF	1	01-JUN-75	12500	1000
7	TOT ANA	V PROF	1	01-AUG-75	12500	1000
8	SAVIĆ MILAN	R PROF	1	01-SEP-65	13500	2000
1	RADOVIĆ NIKOLA	R PROF		01-NOV-61	14500	4000
9	PETRIĆ JANKO	R PROF		01-OCT-60	13500	

SELECT \* FROM PREDJAE;

S_NAS	S_PRED	BR_GR
1	3	2
1	5	1
2	1	1
3	5	1
5	2	1

#### Selekcija kolona

Primer: Prikazati S\_NAS i PREZIME\_IME n-torki iz tabele NASTAVNIK.

Rešenje: SELECT S\_NAS, PREZIME\_IME FROM NASTAVNIK;

S_NAS	PREZIME_IME
2	PETROVIĆ PETAR
3	PEIĆ PETAR
4	SIMIĆ SIMA
5	I利Ć JOVAN
6	SAVIĆ ILIJA
7	TOT ANA
8	SAVIĆ MILAN
9	PETRIĆ JANKO
1	RADOVIĆ NIKOLA

Kolone rezultata pojavljuju se u redosledu definisanom u SELECT naredbi. Redosled n-torki rezultata određuje SUBP.

#### Selekcija n-torki - klauzula WHERE

U prethodnom primeru smo videli kako vršimo selekciju kolona tabele. Za selekciju n-torki tabele potrebno je uključiti WHERE klauzulu.

Korišćena u SELECT naredbi WHERE klauzula omogućava:

1. Selekciju odredjenih n-torki tabele,
2. Selekciju n-torki koje zadovoljavaju višestruke uslove,
3. Selekciju n-torki koje zadovoljavaju bar jedan od više uslova,
4. Selekciju n-torki koje ne zadovoljavaju odredjene uslove,
5. Selekciju n-torki korišćenjem AND i OR logičkih operatora,
6. Selekciju n-torki unutar definisanog raspona,
7. Selekciju n-torki koje zadovoljavaju vrednosti u listi vrednosti,
8. Selekciju n-torki koje sadrže određenu kombinaciju karaktera.

Za izražavanje potrebnih uslova selekcije n-torki u WHERE klauzuli mogu se koristiti sledeći operatori:

- |          |   |
|----------|---|
| =        | jednako,  |
| !=       | nije jednako,   |
| >>= < <= | veće, veće ili jednako, manje, manje ili jednako,                         |
| NOT IN   | nije jednako nijednoj vrednosti što je ekvivalentno sa ' <b>!= ALL</b> ', |
| IN       | jednako sa nekom vrednošću što je ekvivalentno sa ' <b>= ANY</b> ',       |
| ANY      | poredjenje vrednosti sa svakom dobijenom vrednošću rezultata,             |
| ALL      | poredjenje vrednosti sa svim dobijenim vrednostima rezultata,             |

[NOT] BETWEEN x AND y [Not] veće ili jednako od x i manje ili jednako od y.

EXISTS istinito ako se kao rezultat dobije bar jedna n-torka,  
[NOT] LIKE slaganje ili ne slaganje sekvence karaktera koja sledi.  
'%' slaganje bilo koje sekvene karaktera,  
'\_' slaganje jednog karaktera,

IS [NOT] NULL je [nije] nul vrednost,  
NOT negacija logičkog rezultata,  
AND logičko I,  
OR logičko ILI,

*Primer:* Prikazati sve predmete koji se predaju u 8 semestru:

Rešenje:

```
SELECT *
  FROM PREDMET
 WHERE SEMESTAR = 8;
```

S_PRED_NAZIV	SEMESTAR
1 INFORMACIONI SISTEMI	8
4 TEHNIKE PROGRAMIRANJA	8

#### Izražavanje uslova pomoću raspona podataka - BETWEEN

Operator BETWEEN omogućava selekciju n-torki sa sadržajem koji pripada definisanom rasponu podataka.

*Primer:* Prikazati sve nastavnike koji imaju platu izmedju 11000 i 12500.

Rešenje:

```
SELECT PREZIME_IME, PLATA
  FROM NASTAVNIK
 WHERE PLATA BETWEEN 11000 AND 12500;
```

PREZIME_IME	PLATA
PETROVIĆ PETAR	11500
PEIĆ PETAR	11500
SIMIĆ SIMA	11500
ILIĆ JOVAN	11500
SAVIĆ ILIJA	12500
TOT ANA	12500

Prethodni upit realizuje i sledeća naredba:

```
SELECT PREZIME_IME, PLATA
  FROM NASTAVNIK
 WHERE PLATA >= 11000 AND PLATA <= 12500;
```

#### Pretraživanje za vrednosti u listi - IN

Operator IN omogućava izbor n-torki koje sadrže vrednost koja je jednaka jednoj od vrednosti navedenih u listi.

*Primer:* Prikazati sve predmete čija je šifra 2 ili 3.

Rešenje:

```
SELECT *
  FROM PREDMET
 WHERE S_PRED IN (2, 3);
```

S_PRED_NAZIV	SEMESTAR
2 STRUKTURE I BP	3
3 OSNOVE RACUNARSTVA	1

Prethodni upit realizuje i sledeća naredba:

```
SELECT *
  FROM PREDMET
 WHERE S_PRED = 2 OR S_PRED = 3;
```

#### Operator LIKE

Operator LIKE omogućava izbor n-torki koje imaju parcijalno definisan sadržaj određenog obeležja (odredjenu kombinaciju karaktera).

*Primer:* Prikazati prezime i ime svih nastavnika koji kao treće slovo u prezimenu imaju "I".

Rešenje:

```
SELECT PREZIME_IME
  FROM NASTAVNIK
 WHERE PREZIME_IME LIKE '_ _ I %';
```

PREZIME_IME
PEIĆ PETAR
ILIĆ JOVAN

Karakter \_ označava jedan karakter (bilo koji), dok % označava bilo koji niz karaktera.

Operatorima BETWEEN, IN i LIKE može prethoditi reč NOT (negacija). Isti mogu biti povezani sa AND i OR operatorima u izgradnji kompleksnih izraza za pretraživanje.

#### *Definisanje redosleda n-torki rezultata pretraživanja - ORDER BY*

U prethodnim primerima SELECT naredbe redosled n-torki u rezultatu određen je od strane SUBP. Korišćenjem klauzule ORDER BY na kraju SELECT naredbe možemo kontrolisati redosled n-torki rezultata.

*Primer:* Prikazati platu, prezime i ime i zvanje nastavnika prema opadajućem iznosu plate.

Rešenje:

```
SELECT PLATA, PREZIME_IME, ZVANJE
      FROM NASTAVNIK
     ORDER BY PLATA DESC;
```

PLATA	S_NAS	PREZIME_IME	ZVANJE
14500	1	RADOVIĆ NIKOLA	R PROF
13500	9	PETRIĆ JANKO	R PROF
13500	8	SAVIC MILAN	R PROF
12500	6	SAVIC ILIJA	V PROF
12500	7	TOT ANA	V PROF
11500	2	PETROVIĆ PETAR	DOCENT
11500	3	PEIĆ PETAR	DOCENT
11500	4	SIMIĆ SIMA	DOCENT
11500	5	ILIĆ JOVAN	DOCENT

#### *Eliminisanje jednakih n-torki u rezultatu - klauzula DISTINCT*

Prepostavimo da želimo da prikažemo listu svih zvanja iz tabele NASTAVNIK. Odgovarajuća naredba glasi:

```
SELECT ZVANJE
      FROM NASTAVNIK;
```

#### ZVANJE

```
DOCENT
DOCENT
DOCENT
DOCENT
V PROF
V PROF
R PROF
R PROF
R PROF
```

Vidimo u rezultatu više n-torki sa istim sadržajem jer postoji više nastavnika sa istim zvanjem. Ako želimo eliminisati iste n-torke u rezultatu koristimo klauzulu DISTINCT koja znači: *prikaži samo različite n-torke*.

Za prethodni primer:

```
SELECT DISTINCT ZVANJE
      FROM NASTAVNIK;
```

#### ZVANJE

```
DOCENT
V PROF
R PROF
```

#### *9.2.2.2.2. Pretraživanje jedne tabele uz oblikovanje dobijenih podataka*

##### *Izrazi i funkcije*

Pored prikazivanja vrednosti memorisanih u tabelama (proste vrednosti), SQL ima više funkcija koje mogu biti korišćene za dobijanje sumarnih informacija, kao i mogućnost primene aritmetičkih funkcija i funkcija nad nizovima karaktera (stringovi).

Funkcije za dobijanje sumarnih informacija nad numeričkim kolonama su:

AVG ([ALL | DISTINCT] *obeležje*) - izračunava srednju vrednost,  
 SUM ([ALL | DISTINCT] *obeležje*) - izračunava ukupnu vrednost,  
 MIN ([ALL | DISTINCT] *izraz*) - nalazi minimalnu vrednost od *izraz*,  
 MAX ([ALL | DISTINCT] *izraz*) - nalazi maksimalnu vrednost od *izraz*.

Funkcija COUNT definisana je nad kolonama bilo kog tipa. Ona ima sledeći opšti oblik:

COUNT ({\*} | [ALL | DISTINCT] naziv\_obeležja).

Iz ovog opšteg oblika mogu nastati sledeći oblici:

COUNT (\*) - nalazi broj n-torki,  
 COUNT (*izraz*) - nalazi broj n-torki sa NOT NULL vrednostima od *izraz*,  
 COUNT (DISTINCT *izraz*) - nalazi broj različitih n-torki sa NOT NULL vrednostima *izraz*,  
 COUNT (ALL *izraz*) - nalazi broj svih n-torki sa NOT NULL vrednostima od *izraz*,

*Primer:* Naći minimalnu, srednju i maksimalnu platu svih nastavnika, kao i ukupan broj nastavnika.

Rešenje:

```
SELECT MIN(PLATA), AVG(PLATA), MAX(PLATA), COUNT(*)
FROM NASTAVNIK;
```

MIN(PLATA)	AVG(PLATA)	MAX(PLATA)	COUNT(*)
11500	12500	14500	9

*Primer:* Napisati odgovarajuću SQL naredbu za prikaz ukupne plate i dodatka za sve DOCENTE.

Rešenje:

```
SELECT SUM(PLATA), SUM(DODATAK)
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

SUM(PLATA) SUM(DODATAK)

46000

*Sumiranje informacija - klauzula GROUP BY*

Upotreba klauzule GROUP BY omogućava dobijanje informacija za svaku različitu vrednost obeležja po kojem se vrši grupisanje.

Razmotrimo potrebu dobijanja informacija o srednjoj plati i broju nastavnika svakog pojedinog zvanja. Na osnovu do sada rečenog ovaj upit bi se mogao realizovati izvršavanjem onoliko SQL SELECT naredbi koliko ima različitih zvanja. Pri tome se SELECT i FROM klauzule ne bi menjale, dok bi se uslov u WHERE klauzuli menjao u zavisnosti od zvanja.

Jedna od takvih SQL naredbi čiji je rezultat jedna n-torka, glasi:

```
SELECT ZVANJE, AVG(PLATA), COUNT(*)
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

ZVANJE	AVG(PLATA)	COUNT(*)
DOCENT	11500	4

Informaciju o svim različitim zvanjima dobili bi upitom (jedan od ranijih primera):

```
SELECT DISTINCT ZVANJE
FROM NASTAVNIK;
```

ZVANJE
DOCENT
V PROF
R PROF

Kao rezultat ponavljanja SELECT naredbe za sva zvanja dobili bi ukupno tri n-torke.

Identičan rezultat možemo dobiti samo jednom SQL naredbom (umesto četiri) sledećeg oblika:

```
SELECT ZVANJE, AVG(PLATA), COUNT(*)
  FROM NASTAVNIK
 GROUP BY ZVANJE;
```

ZVANJE	Avg(PLATA)	COUNT(*)
DOCENT	11500	4
V PROF	12500	2
R PROF	13500	3

Dejstvo GROUP BY klauzule je identično ponavljanju SELECT naredbe sa različitim WHERE uslovom.

U SELECT listi za prikaz kolona rezultata pored funkcija može doći samo obeležje koje se nalazi u GROUP BY klauzuli.

Grupisanje n-torki može se vršiti po više obeležja koja se navode u GROUP BY klauzuli.

#### *Uslovi pretraživanja za grupu - klauzula HAVING*

Pošto su grupe već formirane sa GROUP BY klauzulom korišćenjem HAVING klauzule definiše se kriterij za selekciju grupa.

*Primer:* Prikazati u kom zvanju ima više od dva nastavnika.

Rešenje:

```
SELECT ZVANJE, COUNT(*)
  FROM NASTAVNIK
 GROUP BY ZVANJE HAVING COUNT(*) > 2;
```

ZVANJE	COUNT(*)
DOCENT	4
R PROF	3

WHERE i GROUP BY klauzula se mogu koristiti u istoj SELECT naredbi pri čemu WHERE klauzula uvek ide pre GROUP BY klauzule. Sa WHERE klauzulom se prvo definiše uslov selekcije n-torki, zatim se selektovane n-torce grupišu GROUP BY klauzulom, pa se eventualno izvrši selekcija formiranih grupa sa HAVING klauzulom.

SQL naredbe mogu sadržavati aritmetičke izraze sastavljene od imena kolona i konstanti povezanih aritmetičkim operatorima (+, \*, -, /). Korišćenjem zagrade u izrazima može se definisati redosled sračunavanja izraza.

*Primer:* Koliko je srednje godišnje primanje redovnih profesora.

Rešenje:

```
SELECT ZVANJE, AVG(PLATA + DODATAKA) * 12
  FROM NASTAVNIK
 WHERE ZVANJE = 'R PROF'
 GROUP BY ZVANJE;
```

ZVANJE	AVG(PLATA+DODATAK) * 12
R PROF	186000

#### *Aritmetičke funkcije*

SQL podržava sledeće aritmetičke funkcije:

ABS( <i>broj</i> )	- apsolutna vrednost od <i>broj</i> ,
MOD( <i>broj<sub>1</sub></i> , <i>broj<sub>2</sub></i> )	- izračunava <i>broj<sub>1</sub></i> moduo <i>broj<sub>2</sub></i> ,
POWER( <i>broj</i> , <i>e</i> )	- diže <i>broj</i> na <i>e</i> -ti stepen,
ROUND( <i>broj</i> [,d])	- zaokružuje <i>broj</i> na d decimalna,
SIGN( <i>broj</i> )	- daje +1 ako je <i>broj</i> > 0, 0 ako je <i>broj</i> = 0, -1 ako je <i>broj</i> < 0,
SQRT( <i>broj</i> )	- izračunava kvadratni koren od <i>broj</i> ,
TRUNC( <i>broj</i> [,d])	- u <i>broj</i> odbacuje ostatak posle d-tog decimalnog mesta,

*Primer:* Koji nastavnici zaradjuju više od 70 dinara po satu. Zaradu po satu izračunati tako da se ukupni iznos plate i dodatka podeli sa 176 (broj sati u mesecu). U prikazu zaradu po satu zaokruži na ceo broj.

Rešenje:

```
SELECT S_NAS, PREZIME_IME, ROUND((PLATA+DODATAK)/176)
      "ZARADA PO SATU"
  FROM NASTAVNIK
 WHERE (PLATA + DODATAK) / 176 > 70;
```

S_NAS	PREZIME_IME	ZARADA PO SATU
6	SAVIC ILIJA	77
7	TOT ANA	77
8	SAVIC MILAN	88
1	RADOVIĆ NIKOLA	105

Mogli smo do ovog primera zapaziti da se u rezultatu dobijenom SELECT naredbom kolone imenuju kao i navedeni *element selekcije*. Kada se u prikazu rezultata želi drugačiji naziv kolone od naziva elementa selekcije koristi se privremeno ime (sinonim) za prikaz elementa selekcije tako da se navede iza *elementa selekcije* između dva znaka ". Ukoliko se privremeno ime sastoji od jedne reči znakovi " se ne moraju navoditi. U prethodnom primeru može se uočiti da je to učinjeno za treću kolonu rezultata.

Pažljivom proverom tabele NASTAVNIK možemo se zapitati zašto u prethodnom rezultatu nema nastavnika PETRIĆ JANKA koji ima platu 13500, dodatak je NULL vrednost i za kojeg je zarada po satu  $13500/176 > 70$ . Razlog je u tome što se obeležja sa NULL vrednostima ne koriste pri izračunavanju izraza i funkcija. Da bi se izračunavanje ipak omogućilo možemo koristiti NVL funkciju koja privremeno menja NULL vrednost sa vrednošću za koju se odlučimo, tj. sa vrednošću koja odgovara željenoj operaciji odnosno rezultatu.

Funkcija NVL(*obeležje, broj*) izvršava se na sledeći način:

Ako je vrednost za *obeležje* NULL vrednost ista se privremeno zamenjuje sa *broj*.

Ako je vrednost za *obeležje* definisana tada se uzima ta vrednost.

Razmotrimo sada još jednom prethodni primer sa sledećom naredbom za pretraživanje:

```
SELECT S_NAS, PREZIME_IME, ROUND((PLATA +
          NVL(DODATAK,0)) / 176) "ZARADA PO SATU"
    FROM NASTAVNIK
   WHERE (PLATA + NVL(DODATAK, 0)) / 176 > 70;
```

S_NAS	PREZIME_IME	ZARADA PO SATU
6	SAVIC ILIJA	77
7	TOT ANA	77
8	SAVIC MILAN	88
1	RADOVIĆ NIKOLA	105
9	PETRIC JANKO	77

Uočavamo na prethodnom primeru da za posledicu korišćenja NULL vrednosti uvek treba voditi računa o tome šta predstavlja rezultat aritmetičkog izraza ili funkcije, odnosno koje n-torce su uključene u rezultat, a koje nisu. Zbog toga je i napomenuto pri razmatranju operacijske komponente relacionog modela podataka, da kad god je to moguće treba eliminisati kreiranje baze sa NULL vrednostima.

#### Funkcije nad nizom karaktera

U SQL je definisan veći broj funkcija nad podacima tipa karakter.

Slede neke od funkcija:

- niz*, || *niz* - spaja nizove karaktera,
- INSTR (*niz, podniz [,pos]*) - traži *podniz* u *niz* od pozicije *pos*. Ako je nadjen vraća njegovu poziciju, inače je 0.
- LENGTH (*niz*) - nalazi dužinu od *niz*,
- LOWER (*niz*) - konvertuje sva velika slova od *niz* u mala,
- LPAD (*niz<sub>1</sub>, broj [,niz<sub>2</sub>]*) - popunjava levu stranu od *niz<sub>1</sub>* sa karakterom *niz<sub>2</sub>* u dužini *broj*, ako je *niz<sub>2</sub>* izostavljen sa blanko,
- RPAD (*niz<sub>1</sub>, broj [,niz<sub>2</sub>]*) - popunjava desnu stranu od *niz<sub>1</sub>* sa karakterom *niz<sub>2</sub>* u dužini *broj*, ako je *niz<sub>2</sub>* izostavljen sa blanko,
- SUBSTR (*niz, pos [,duz]*) - daje podniz od *niz* počinjući od pozicije *pos*, dužine *duz*.
- TO\_NUM (*niz*) - konvertuje *niz* karaktera (numeričkih) u broj,
- TO\_CHAR (*broj*) - konvertuje *broj* u niz karaktera,

`TO_CHAR (datum[format])` - konvertuje `datum` u karakter vrednost oblik `format`,

`TO_DATE (niz, format)` - konvertuje karakter vrednost datuma u datum sa formatom `format`.

`UPPER (niz)` - konvertuje sva mala slova od `niz` u velika.

Sve ove funkcije navode se iza SELECT naredbe.

**Primer:** Prikazati prezime ime nastavnika iza kojeg treba neposredno prikazati zvanje. Redosled n-torki rezultata treba da je u rastućem redosledu obeležja PREZIME\_IME.

Rešenje:

```
SELECT PREZIME_IME || ',' || ZVANJE NASTAVNIK
      FROM NASTAVNIK
     ORDER BY PREZIME_IME;
```

NASTAVNIK
ILIĆ JOVAN, DOCENT
PEIĆ PETAR, DOCENT
PETRIĆ JANKO, R PROF
PETROVIĆ PETAR, DOCENT
RADOVIĆ NIKOLA, R PROF
SAVIC ILLIJA, V PROF
SAVIĆ MILAN, R PROF
SIMIĆ SIMA, DOCENT
TOT ANA, V PROF

Prethodno pretraživanje spaja tri niza (prvi niz je PREZIME\_IME, drugi je zarez i jedno prazno mesto i treći niz je vrednost obeležja ZVANJE) koji se u prikazu rezultata imenuju sa NASTAVNIK.

#### Prikazivanje i rad sa datumima

Jedan od tipova podataka za definisanje obeležja relacione baze podataka je datum. Standardni oblik datuma izgleda na primer: 15-JAN-93. Ovaj oblik se piše 'DD-MON-YY'. Ako drugačije nije naglašeno prikaz i unos datuma je u ovom obliku.

Ranije pomenutom funkcijom `TO_CHAR` može se odabratи drugi oblik za prikaz datuma njegovim pretvaranjem u karakter vrednost.

Oblici koji se mogu koristiti za datum su:

OBLIK	PRIMER	OBJAŠNJENJE
(bez formata)	15-JAN-93	standardni format
MM/DD/YY	15/01/93	svi brojevi od dve brojke
DD/MM/YYYY	15.01.1993	"." kao separator, puna godina
Month DD, YYYY	January 15, 1993	mesec napisan slovima,
DY DD MON YY	WED 15 JAN 92	dan i mesec skraćeni i napisani velikim slovima
Day Mon DD	Wednesday Jan 15	dan napisan slovima, nema godine

**Primer:** Prikazati prezime ime, i datum zaposlenja nastavnika sa zvanjem vanrednog profesora. Datum zaposlenja prikazati u standardnom obliku i oblicima sa punom godinom.

Rešenje:

```
SELECT PREZIME_IME, DATZAP,
       TO_CHAR(DATZAP, 'DD.MM.YYYY') 'DD MM.YYYY',
       TO_CHAR(DATZAP, "Mesec" Month DD, YYYY)
                           'Month DD, YYYY'
    FROM NASTAVNIK
   WHERE ZVANJE = 'V PROF';
```

PREZIME_IME	DATZAP	DDMM.YYYY	Month DD, YYYY
SAVIC ILLIJA	01-JUN-75	01.06.1975	Mesec June 01, 1975
TOT ANA	01-AUG-75	01.08.1975	Mesec August 01, 1975

Sa podacima tipa datum mogu se izvoditi sledeće aritmetičke operacije:

`datum + broj` - na `datum` se dodaje `broj` dana,

`datum - broj` - od `datum` se oduzima `broj` dana,

`datum1 - datum2` - oduzima `datum2` od `datum1`, i kao rezultat daje razliku iskazanu brojem dana.

**Primer:** Prikazati prezime ime i broj godina radnog staža svakog nastavnika.

Rešenje:

```
SELECT PREZIME_IME, (SYSDATE - DATZAP) / 365
      'GODINA STAŽA'
   FROM NASTAVNIK
  ORDER BY PREZIME_IME;
```

NASTAVNIK	GODINA STAŽA
ILIĆ JOVAN	8.39402
PEIĆ PETAR	10.5639
PETRIĆ JANKO	31.9064
PETROVIĆ PETAR	11.6406
RADOVIĆ NIKOLA	32.9913
SAVIĆ ILLJA	18.3173
SAVIĆ MILAN	28.0707
SIMIĆ SIMA	9.47622
TOT ANA	18.1502

U prethodnom primeru SYSDATE je sistemski tekući datum i vreme, a rezultati odgovaraju za 19.09.1993. Godine staža su izračunate za 365 dana u godini čime u razmatranje nisu uključene prestupne godine. Korišćenjem na primer, izraza MONTHS\_BETWEEN (SYSDATE, DATZAP) / 12 dobili bi broj godina radnog staža nezavisno od prestupnih godina. Funkcija MONTHS\_BETWEEN (SYSDATE, *datum*) daje broj meseci između datum i tekućeg datuma.

#### 9.2.2.2.3. Ulaganje upita nad jednom tabelom u upit nad drugom tabelom

Jedan od načina povezivanja tabela u bazi podataka je ulaganjem upita nad jednom tabelom u upit nad drugom, odnosno dinamičkom zamenom rezultata jednog upita u WHERE klauzuli drugog upita.

*Primer:* Prikazati prezime ime i zvanje svih nastavnika koji imaju isto zvanje kao i RADOVIĆ NIKOLA.

Korak 1. Utvrdimo koje zvanje ima RADOVIĆ NIKOLA.

```
SELECT ZVANJE
      FROM NASTAVNIK
     WHERE PREZIME_IME = 'RADOVIĆ NIKOLA';
```

ZVANJE
R PROF

Korak 2. Prkažimo prezime ime i zvanje svih nastavnika koji imaju zvanje R PROF.

```
SELECT PREZIME_IME, ZVANJE
      FROM NASTAVNIK
     WHERE ZVANJE = 'R PROF';
```

PREZIME_IME	ZVANJE
RADOVIĆ NIKOLA	R PROF
PETRIĆ JANKO	R PROF
SAVIĆ MILAN	R PROF

Povezivanje tabela dinamičkom zamenom rezultata jednog upita u WHERE klauzuli drugog sastoji se u tome da se prvi upit nadje u WHERE klauzuli drugog.

U prethodnom primeru to izgleda:

```
SELECT PREZIME_IME, ZVANJE
      FROM NASTAVNIK
     WHERE ZVANJE = (SELECT ZVANJE
                      FROM NASTAVNIK
                     WHERE PREZIME_IME = 'RADOVIĆ NIKOLA');
```

Dobijamo isti rezultat kao i posle koraka 2.

Upit iz koraka 1 ili prvi upit navodi se u zagradama drugog upita i naziva se *unutrašnji upit* (uloženi upit). Drugi upit naziva se *spoljašnji upit*.

Prilikom izvršavanja upita sa uloženim upitom prvo se izvršava unutrašnji upit a zatim spoljašnji upit.

*Primer:* Prikazati prezime ime, zvanje i platu nastavnika sa zvanjem DOCENT koji ne predaju ni jedan predmet.

Rešenje:

```
SELECT PREZIME_IME, ZVANJE, PLATA
  FROM NASTAVNIK
 WHERE ZVANJE = 'DOCENT' AND
       S_NAS NOT IN (SELECT DISTINCT S_NAS
                      FROM PREDAJE);
```

PREZIME_IME	ZVANJE	PLATA
SIMIC SIMA	DOCENT	11500

*Primer:* Koji nastavnik u svakom zvanju ima najveću platu ?

Rešenje:

```
SELECT PREZIME_IME, ZVANJE, PLATA
  FROM NASTAVNIK
 WHERE (ZVANJE, PLATA) IN (SELECT ZVANJE, MAX(PLATA)
                           FROM NASTAVNIK
                           GROUP BY ZVANJE)
 ORDER BY PLATA DESC;
```

PREZIME_IME	ZVANJE	PLATA
RADOVIĆ NIKOLA	R. PROF.	14500
SAVIC ILIJA	V. PROF.	12500
TOTANA	V. PROF.	12500
PETROVIĆ PETAR	DOCENT	11500
SIMIC SIMA	DOCENT	11500
ILIĆ JOVAN	DOCENT	11500

*Primer:* Prikazati prezime ime nastavnika koji imaju najmanje jednog nastavnika koji im je podredjen.

Rešenje:

```
SELECT PREZIME_IME
  FROM NASTAVNIK A
 WHERE EXISTS (SELECT *
                  FROM NSATAVNICI B
                 WHERE A.S_NAS = B.S_DIR)
 ORDER BY PREZIME_IME;
```

NASTAVNIK
RADOVIĆ NIKOLA SAVIC ILIJA

Prethodni primer pokazuje da spoljašnji i unutrašnji upit mogu biti povezani vrednostima više obeležja koja se u tom slučaju navode u zagradama.

Spoljašnji i unutrašnji upit mogu biti sastavljeni od dva ili više SELECT blokova medjusobno povezanih operatorima UNION, INTERSECT ili MINUS:

- UNION (unija) prikazuje sve rezultujuće n-torke prvog SELECT bloka i one rezultujuće n-torke drugog SELECT bloka koje nisu medju rezultujućim n-torkama prvog SELECT bloka.

- INTERSECT (presek) prikazuje sve rezultujuće n-torke prvog SELECT bloka koje su istovremeno i rezultujuće n-torke drugog SELECT bloka.

- MINUS prikazuje sve rezultujuće n-torke prvog SELECT bloka koje nisu istovremeno i rezultujuće n-torke drugog SELECT bloka.

Da bi navedene operacije mogle biti primenjene rezultati upita SELECT blokova koji učestvuju u operaciji moraju imati isti broj rezultujućih kolona i iste moraju odgovarati po tipu.

*Primer:* Prikazati prezime ime, zvanje i plate nastavnika sa zvanjem docenta i redovnog profesora.

Rešenje:

```
SELECT PREZIME_IME, ZVANJE, PLATA
FROM NSTAVNIK
WHERE ZVANJE = 'DOCENT'
UNION
SELECT PREZIME_IME, ZVANJE, PLATA
FROM NSTAVNIK
WHERE ZVANJE = 'R PROF';
```

PREZIME_IME	ZVANJE	PLATA
PETROVIC PETAR	DOCENT	11500
PEIĆ PETAR	DOCENT	11500
SIMIC SIMA	DOCENT	11500
ILIC JOVAN	DOCENT	11500
SAVIC MILAN	R PROF	13500
RADOVIC NIKOLA	R PROF	14500
PETRIC JANKO	R PROF	13500

Prethodno pretraživanje se može postići i sa:

```
SELECT PREZIME_IME, ZVANJE, PLATA
FROM NSTAVNIK
WHERE ZVANJE = 'DOCENT' OR ZVANJE = 'R PROF';
```

Kao što i prethodni primer pokazuje primena operacija UNION, INTERSECT i MINUS nemaju puno smisla nad jednom tabelom, jer se isti rezultati mogu dobiti korišćenjem logičkih operatora AND, OR i NOT.

#### 9.2.2.2.4. Povezivanje više tabела

U svim prethodnim primerima SELECT naredbe rezultat pretraživanja formiran je iz jedne tabele. Kako doći do rezultata pretraživanja koji nije smešten u jednoj tabeli? Povezivanje dve ili više tabele omogućava kombinovanje podataka sa prikazom rezultata u jednoj tabeli.

Prepostavimo da želimo da znamo prezime i ime nastavnika koji predaje predmet sa šifrom 1. Posmatrajući naše tabele NASTAVNIK, PREDMET i PREDAJE vidimo da do željenog rezultata možemo doći u sledeća dva koraka:

Korak 1. Pretraživanjem tabele PREDAJE dobit ćemo šifru nastavnika koji predaje predmet sa šifrom 1.

```
SELECT S_PRED, S_NAS
FROM PREDAJE
WHERE S_PRED = 1;
```

S_PRED	S_NAS
1	2

Korak 2. Sada pretraživanjem tabele NASTAVNIK za šifru nastavnika 2 dobijemo prezime i ime nastavnika.

```
SELECT S_NAS, PREZIME_IME
FROM NASTAVNIK
WHERE S_NAS = 2;
```

S_NAS	PREZIME_IME
2	PETROVIC PETAR

Kao rezultat prethodna dva upita dobili smo odgovor da PETROVIĆ PETAR predaje predmet sa šifrom 1.

Do istog rezultata možemo doći koristeći umesto dva upita samo jedan. Uopšteno rečeno, da bi se izvršilo pretraživanje u kojem rezultat zavisi od sadržaja više tabele moraju se u FROM klauzuli navesti tabele koje treba pretražiti, u WHERE klauzuli obeležja i uslov za spajanje tabela. Pored uslova za spajanje u WHERE klauzuli mogu biti navedeni i uslovi selekcije n-torki. Uslov spajanja (odredjen operatorom izmedju unijiski kompatibilnih obeležja) može biti znak jednakosti, ali i bilo koji drugi smisleni odnos izmedju posmatranih obeležja (>, >=, <, <=, !=).

Ukoliko se izostavi uslov spajanja u WHERE klauzuli izvršava se Dekartov proizvod tabela iza FROM. Za dve tabele svaka n-torka prve spaja se sa svakom n-torkom druge.

Jedinstveni upit iz prethodnog primera glasi:

```
SELECT PREDAJE.S_PRED, S_NAS, PREZIME_IME
  FROM PREDAJE, NASTAVNIK
 WHERE NASTAVNIK.S_NAS = PREDAJE.S_NAS AND
       PREDAJE.S_PRED = 1;
```

S_PRED	S_NAS	PREZIME_IME
1	2	PETROVIC PETAR

Uslovi u WHERE klauzuli definišu uslove koji trebaju biti ispunjeni da bi se izvršilo spajanje n-torki tabela PREDAJE i NASTAVNIK.

Ukoliko se u prethodnom primeru u WHERE klauzuli izostavi uslov spajanja tabela kao rezultat bi dobili Dekartov proizvod (CARTESIAN JOIN) tabela PREDAJE i NASTAVNIK (ukupno 45 n-torki).

Nazivi tabela u WHERE klauzuli moraju se pisati samo ukoliko nazivi obeležja nisu jedinstveni u posmatranom skupu tabela. U prethodnom primeru u WHERE klauzuli umesto PREDAJE.S\_PRED mogli smo pisati samo S\_PRED jer je to jedinstveno obeležje u navedenom skupu tabela. Primećujemo i obavezu kvalifikacije obeležja iza SELECT.

*Primer:* Prikazati za svaku šifru nastavnika, odgovarajuće prezime i ime, šifru predmeta koji predaje i odgovarajući naziv predmeta.

Rešenje:

```
SELECT NASTAVNIK.S_NAS, PREZIME_IME, PREDMET.S_PRED, NAZIV
  FROM NASTAVNIK, PREDMET, PREDAJE
 WHERE NASTAVNIK.S_NAS = PREDAJE.S_NAS AND
       PREDAJE.S_PRED = PREDMET.S_PRED;
```

S_NAS	PREZIME_IME	S_PRED	NAZIV
2	PETROVIC PETAR	1	INFORMACIONI SISTEMI
5	ILIC JOVAN	2	STRUKTURE I BP
1	RADOVIC NIKOLA	3	OSNOVE RACUNARSTVA
1	RADOVIC NIKOLA	5	PROGRAMIRANJE RS
3	PEIC PETAR	5	PROGRAMIRANJE RS

*Primer:* Za svakog nastavnika prikazati prezime ime, platu kao i prezime ime i platu njegovog direktora. Redosled n-torki rezultata treba da bude po abecedi prezimena i imena nastavnika.

Rešenje:

```
SELECT A.PREZIME_IME, A.PLATA, B.PREZIME_IME
      "PREZIME_IME_DIR", B.PLATA PLATA_DIR
    FROM NASTAVNIK A, NASTAVNIK B
   WHERE A.S_DIR = B.S_NAS
 ORDER BY A.PREZIME_IME;
```

PREZIME_IME	PLATA	PREZIME_IME_DIR	PLATA_DIR
ILIC JOVAN	11500	SAVIC ILIJA	12500
PEIC PETAR	11500	SAVIC ILIJA	12500
PETROVIC PETAR	11500	SAVIC ILIJA	12500
SAVIC ILIJA	12500	RADOVIC NIKOILA	14500
SAVIC MILAN	13500	RADOVIC NIKOILA	14500
SIMIC SIMA	11500	SAVIC ILIJA	12500
TOTANA	12500	RADOVIC NIKOILA	14500

U ovom primeru rezultat je dobijen iz spoja tabele NASTAVNIK same sa sobom (SELF JOIN). Oznake A i B su upotrebljene kao dva privremena imena za tabelu NASTAVNIK. Spoj tabela A i B je izvršen za n-torce koje zadovoljavaju uslov jednakosti obeležja S\_DIR iz prve sa S\_NAS iz druge. Projekcija je izvršena po obeležjima navedenim iza SELECT. Klauzulom ORDER BY definisan je rastući redosled n-torki rezultata po obeležju PREZIM\_IME iz tabele A.

Uočavamo da se u rezultatu ne dobijaju podaci o nastavnicima koji za obeležje S\_DIR imaju NULL vrednost. Ako se žele prikazati n-torce koji za obeležje koje učestvuje u WHERE uslovu spajanja imaju NULL vrednost potrebno je izvršiti *spoljno spajanje* (OUTER JOIN). Spoljno spajanje dve tabele realizuje se tako da se označenoj tabeli (tabela koja ne sadrži NULL vrednosti) za potrebe spajanja doda još jedna n-torka koja sadrži null-vrednosti svih obeležja. Takva dodata n-torka spaja se sa onim n-torcama za koje se ne pronadje odgovarajuća stvarna n-torka. Spoljno spajanje se označava navodnjem u WHERE klauzuli oznake "(+)" iza *tabela.obeležje* kojoj treba dodati n-torku.

SQL izraz za prethodni primer glasi:

```
SELECT A.PREZIME_IME, A.PLATA, B.PREZIME_IME
      "PREZIME_IME_DIR", B.PLATA PLATA_DIR
   FROM NASTAVNIK A, NASTAVNIK B
  WHERE A.S_DIR = B.S_NAS (+)
 ORDER BY A.PREZIME_IME;
```

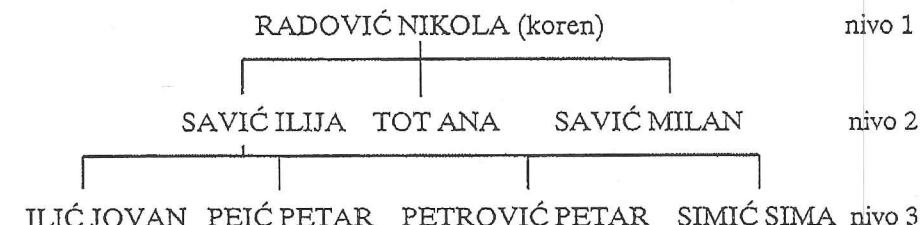
PREZIME_IME	PLATA	PREZIME_IME_DIR	PLATA_DIR
ILIĆ JOVAN	11500	SAVIĆ ILIJA	12500
PEIĆ PETAR	11500	SAVIĆ ILIJA	12500
PETRIĆ JANKO	13500		
PETROVIĆ PETAR	11500	SAVIĆ ILIJA	14500
RADOVIĆ NIKOLA	14500		
SAVIĆ ILIJA	12500	RADOVIĆ NIKOLA	14500
SAVIĆ MILAN	13500	RADOVIĆ NIKOLA	14500
SIMIĆ SIMA	11500	SAVIĆ ILIJA	12500
TOT ANA	12500	RADOVIĆ NIKOLA	14500

Treba uočiti da je u WHERE, uslov A.S\_DIR = B.S\_NAS identičan sa B.S\_NAS = A.S\_DIR, dok uslov A.S\_DIR = B.S\_NAS (+) nije identičan sa B.S\_NAS = A.S\_DIR (+).

Možemo se ovde još jednom podsetiti na neproceduralnost SQL tj. izražavanja ŠTA želimo dobiti, a ne i KAKO. Zamislimo šta bi sve morali da uključimo u programski kod ukoliko bi prethodni zahtev trebalo realizovati nekim od viših programskih jezika (npr. COBOL, PASCAL, FORTRAN).

### 9.2.2.2.5. Pretraživanje tabela koje u sebi sadrže strukturu tipa stabla

Kažemo da tabela sadrži strukturu tipa stabla kada sadrži n-torku koja referencira na jednu ili više drugih n-torki iste tabele. Primer takve tabele je tabela NASTAVNIK u kojoj se nalaze i obeležja S\_NAS i S\_DIR. Vrednost obeležje S\_DIR u jednoj n-torci referencira na nadredjenog nastavnika za kojeg takodje postoji odgovarajuća n-torka. Koristeći obeležja S\_NAS i S\_DIR odnosi u tabeli NASTAVNIK mogu se prikazati na sledeći način:



Za prikaz informacija strukture tipa stabla koriste se klauzule:

CONNECT BY - definije strukturu stabla,  
START WITH - definije koren stabla.

*Primer:* Prikazati šifru, prezime ime, zvanje i šifru pretpostavljenog nastavnika u obliku stabla.

Rešenje:

```
SELECT S_NAS, PREZIME_IME, ZVANJE, S_DIR
      FROM NASTAVNIK
     CONNECT BY PRIOR S_NAS = S_DIR
     START WITH PREZIME_IME = 'RADOVIĆ NIKOLA';
```

S_NAS	PREZIME_IME	ZVANJE	S_DIR
1	RADOVIĆ NIKOLA	R PROF	
6	SAVIC ILIJA	V PROF	1
2	PETROVIĆ PETAR	DOCENT	6
3	PEIĆ PETAR	DOCENT	6
4	SIMIĆ SIMA	DOCENT	6
5	ILIĆ JOVAN	DOCENT	6
7	TOT ANA	V PROF	1
8	SAVIC MILAN	R PROF	1

Prethodni rezultat se formira na sledeći način: Prvo se selektira n-torka iz tabele NASTAVNIK po uslovu u START WITH klauzuli. To je n-torka o nastavniku sa prezimenom imenom RADOVIĆ NIKOLA i šifrom 1. Zatim se izvršava uslov u CONNECT BY klauzuli koji kaže da prethodna šifra nastavnika (1) postaje šifra prepostavljenog nastavnika za selekciju sledeće n-torce. Pretraživanjem tabele NASTAVNIK za uslov S\_DIR = 1 dobija se n-torka o nastavniku SAVIĆ ILIJI sa šifrom 6. Zatim ta šifra postaje kriterij selekcije po koloni S\_DIR.

Čvorovi u stablu mogu biti označeni brojevima nivoa u zavisnosti od udaljenosti od korena, korišćenjem pseudokolone LEVEL.

```
SELECT LEVEL, S_NAS, PREZIME_IME, ZVANJE, S_DIR
  FROM NASTAVNIK
 CONNECT BY PRIOR S_NAS = S_DIR
 START WITH PREZIME_IME = 'RADOVIĆ NIKOLA';
```

LEVEL	S_NAS	PREZIME_IME	ZVANJE	S_DIR
1	1	RADOVIĆ NIKOLA	R PROF	
2	6	SAVIC ILIJA	V PROF	1
3	2	PETROVIĆ PETAR	DOCENT	6
3	3	PEIĆ PETAR	DOCENT	6
3	4	SIMIĆ SIMA	DOCENT	6
3	5	ILIĆ JOVAN	DOCENT	6
2	7	TOT ANA	V PROF	1
2	8	SAVIC MILAN	R PROF	1

Da bi se dobila organizaciona šema kao uredjena lista, potrebno je upotrebiti funkciju LPAD.

```
SELECT LPAD(' ',2 * LEVEL) || PREZIME_IME "PREZIME_IME",
        LEVEL, S_NAS, ZVANJE, S_DIR
  FROM NASTAVNIK
 CONNECT BY PRIOR S_NAS = S_DIR
 START WITH PREZIME_IME = 'RADOVIĆ NIKOLA';
```

PREZIME_IME	LEVEL	S_NAS	ZVANJE	S_DIR
RADOVIĆ NIKOLA	1	1	R PROF	
SAVIC ILIJA	2	6	V PROF	1
PETROVIĆ PETAR	3	2	DOCENT	6
PEIĆ PETAR	3	3	DOCENT	6
SIMIĆ SIMA	3	4	DOCENT	6
ILIĆ JOVAN	3	5	DOCENT	6
TOT ANA	2	7	V PROF	1
SAVIC MILAN	2	8	R PROF	1

### 9.2.2.3. Izmena sadržaja tabele - UPDATE

Opšti oblik naredbe je:

a) UPDATE naziv\_tabele [alias]  
 SET naziv\_obi<sub>1</sub> = izraz<sub>1</sub>, [naziv\_obi<sub>2</sub> = izraz<sub>2</sub>]...  
 [WHERE kriterijum za izmenu n-torki];

b) UPDATE naziv\_tabele [alias]  
 SET (naziv\_obi<sub>1</sub>, [naziv\_obi<sub>2</sub>]) = (podupit)  
 [WHERE kriterijum za izmenu n-torki];

gde je: izraz<sub>i</sub> - izraz koji određuje vrednost koja se pridružuje obi<sub>i</sub>.  
 podupit - selekcija vrednosti (SELECT naredba)

Ovom naredbom menja se sadržaj navedenih obeležja onih n-torki koje zadovoljavaju kriterijum za izmenu n-torki u WHERE klauzuli. Ako nije navedena klauzula WHERE izmena sadržaja vrši se u svim n-torkama tabele.

Klauzula SET određuje obeležja koja podležu promeni i vrednosti koje će im biti dodeljene.

U svakoj n-torki koja zadovoljava WHERE klauzulu, obeležja levo od znaka jednakosti dobijaju vrednosti odredjene izrazom sa desne strane znaka jednakosti.

Ako je u SET klauzuli naveden podupit on mora vratiti tačno jednu n-torku tako da se vrednosti odredjene u SELECT klauzuli podupita pridružuju obeležjima navedenim u listi koja je u zagradama. Uočavamo da broj vrednosti podupita mora odgovarati broju obeležja te da je pridruživanje vrednosti po redosledu navodjenja.

*Primer:* Za nastavnike sa zvanjem VANREDNI PROFESOR upisati dodatak od 1200 dinara:

Rešenje:

```
UPDATE NASTAVNIK
SET DODATAK = 1200
WHERE ZVANJE = 'V PROF';
```

*Primer:* Za nastavnike sa zvanjem DOCENT povećati platu za 5 % i ukinuti dodatak:

Rešenje:

```
UPDATE NASTAVNIK
SET PLATA = 1.05 * PLATA, DODATAK = 0
WHERE ZVANJE = 'DOCENT';
```

Provera rezultata prethodne naredbe može se jednostavno izvršiti na sledeći način:

```
SELECT *
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

S_NAS	PREZIME_IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
2	PETROVIĆ PETAR	DOCENT	6	01-FEB-82	12075	0
3	PEIĆ PETAR	DOCENT	6	01-MAR-83	12075	0
4	SIMIĆ SIMA	DOCENT	6	01-APR-84	12075	0
5	LJUČ JOVAN	DOCENT	6	01-MAY-85	12075	0

*Primer:* Svim nastavnicima sa zvanjem DOCENTA upisati dodatak u iznosu od 10 % dodatka nastavnika RADOVIĆ NIKOLE:

Rešenje:

```
UPDATE NASTAVNIK
SET DODATAK = (SELECT 0.1 * DODATAK
                 FROM NASTAVNIK
                WHERE PREZIME_IME = 'RADOVIĆ NIKOLA')
              WHERE ZVANJE = 'DOCENT';
```

*Primer:* Promeniti šifru predmeta "informacioni sistemi" na 6:

Rešenje:

```
UPDATE PREDMET
SET S_PRED = 006
WHERE S_PRED = 001;
```

Moramo izvršiti i odgovarajuću promenu u tabeli PREDAJE:

```
UPDATE PREDAJE
SET S_PRED = 006
WHERE S_PRED = 001;
```

Jednom naredbom UPDATE nije moguće izvršiti promenu u više od jednoj tabeli. Zbog toga se u prethodnom primeru susrećemo sa problemom integriteta podataka. Posle izvršavanja prve naredbe UPDATE iz prethodnog primera baza podataka u tabeli PREDAJE sadrži n-torce u kojima se navodi nepostojeća šifra predmeta. Promena redosleda UPDATE naredbi ne rešava problem. Zbog toga je važno obezbediti izvršavanje obe naredbe, a ne samo jedne. Problemom očuvanja integriteta baze podataka kada se zahteva više promena nećemo se ovde baviti.

#### 9.2.2.4. Brisanje n-torki table - DELETE

Opšti oblik naredbe je:

```
DELETE FROM naziv_tabele
[WHERE kriterijum brisanja n-torki];
```

Naredbom DELETE brišu se sve n-torce tabele koje ispunjavaju kriterijum za brisanje n-torki. Ako nije zadata klauzula WHERE brišu se sve n-torce tabele.

*Primer:* Brisati podatke o nastavniku sa šifrom 008 (brisanje jedne n-torce):

Rešenje:

```
DELETE FROM NASTAVNIK
WHERE S_NAS = 008;
```

*Primer:* Brisati podatke o angažovanju svih nastavnika na predmetu sa šifrom 005 (brisanje više n-torki):

Rešenje:

```
DELETE FROM PREDAJE
WHERE S_PRED = 005;
```

*Primer:* Brisati sve n-torce tabele PREDMET:

Rešenje:

```
DELETE FROM PREDMET;
```

Ova operacija ne ukida definiciju tabele već samo n-torce pa po svom dejstvu nije ekvivalentna naredbi DROP.

*Primer:* Brisati sve n-torce u tabeli PREDAJE za koje je semestar u kojem se predmet predaje osmi:

Rešenje:

```
DELETE FROM PREDAJE
WHERE S_PRED IN (SELECT DISTINCT PREDMET.S_PRED
                  FROM PREDMET, PREDAJE
                 WHERE PREDMET.S_PRED = PREDAJE.S_PRED
                   AND PREDMET.SEMESTAR = 8);
```

#### 9.2.3. Rečnik podataka

Rečnik podataka sistema za upravljanje relacionom bazom podataka je skup tabela i pogleda koji sadrže informacije o bazi podataka. Na primer u DB2 rečnik podataka ima 20 - 25 tabela.

Rečnik podataka opisuje tabele, poglede, indekse, grupe, korisnike, privilegije i druge informacije o trenutnom stanju baze podataka. SUBP automatski kad god se neki objekat kreira ili menja, ažurira rečnik podataka.

Tabela DTAB opisuje tabele koje čine rečnik podataka sa sledeća dva obeljžja: TNAME - naziv\_tabele, REMARKS - opis tabele. Korišćenjem SQL naredbe SELECT može se prikazati sadržaj tabele DTAB.

```
SELECT *
  FROM DTAB;
```

Deo rezultata dobijen prethodnim upitom glasi:

IME TABELE	OPIS TABELE
CATALOG	Tabele i pogledi kojima korisnik ima pravo pristupa
COLUMNS	Obeležja tabele sa pravom dostupa
DTAB	Opis tabele i pogleda u rečniku podataka SUBP
INDEXES	Indeksi koje je kreirao korisnik
SPACES	Definicija prostora za kreiranje tabela i pogleda
SYSCOLUMNS	Opis obeležja u tabelama
SYSUSERLIST	Lista korisnika SUBP
SYSVIEWS	Lista pogleda kojima korisnik ima pravo pristupa

*Primer:* Prikazati opis korisničke tabele NASTAVNIK.

Opis obeležja u tabelama nalazi se u tabeli SYSCOLUMNS koja ima više obeležja i koje treba poznavati da bi realizovali prethodni zahtev.

```
SELECT CNAME "OBL", COLTYPE TIP, WIDTH "DUŽINA",NULLS
      FROM SYSCOLUMNS
     WHERE TNAME = 'NASTAVNIK';
```

OBL	TIP	DUŽINA	NULLS
S_MAS	NUMBER	3	NOT NULL
PREZIME_IME	CHAR	15	NOT NULL
ZVANJE	CHAR	6	NULL
S_DIR	NUMBER	3	NULL
DATZAP	DATE	7	NULL
PLATA	NUMBER	8	NOT NULL
DODATAK	NUMBER	8	NULL

Korišćenjem naredbi SQL za pretraživanje rečnika podataka, korisnik može doći do informacija o bazi podataka, a zatim preći na njeno pretraživanje. U tradicionalnim sistemima rečnik podataka i baza podataka su bili različito organizovani te je i pristup bio preko različitih interfejsa.

I dok se operacija SELECT iz SQL može obavljati nad tabelama i pogledima rečnika podataka, operacije UPDATE, DELETE i INSERT nisu dozvoljene. Ovakvo ograničenje je sasvim razumljivo s obzirom da SUBP koristi rečnik podataka za održavanje baze podataka. Promene samo u rečniku podataka mogle bi da izazovu nemogućnost korišćenja baze podataka. Naredbe SQL automatski vrše održavanje rečnika podataka. Na primer, posle naredba CREATE TABLE ... u:

- a) SYSTABLES biće upisan nova n-torka,
- b) SYSCOLUMNS za svako obeležje tabele biće upisana jedna n-torka,
- c) Neka druga dejstva koja nisu predmet razmatranja.

Rečnik podataka sadrži informacije i o samim tabelama rečnika podataka. Takve podatke generiše SUBP prilikom konfigurisanja sistema.

### 9.3. Jezik upita na osnovu primera - QBE

Jezik postavljanja upita na osnovu primera (Query-By-Example-QBE) je jezik u kome korisnik postavlja upit na relationalnu bazu podataka odgovarajućim popunjavanjem definisane tabele. SQL je svakako korisnicima prihvatljiviji od SQL i to pre svega korisnicima koji nemaju iskustva u programiranju. U SQL se prepostavlja odredjeno iskustvo u programiranju i on predstavlja kako jezik u klasičnom smislu tako i jezik vrlo visokog nivoa. Nasuprot SQL-u QBE se upiti definišu jednostavnom popunom tabele. Takav pristup je veoma lak za učenje i shvatanje i to pre svega za manje iskusne korisnike. Na primer, u sistemu za upravljanje relacionom bazom podataka DB2 preko dela za upravljanje upitima korisnik može izabratи da li će upit definisati u SQL ili QBE.

U QBE postoje sledeće operacije:

QBE	SQL	objašnjenje
P.	SELECT	štampa
U.	UPDATE	izmena
D.	DELETE	brisanje
I.	INSERT	dodavanje

Druge operacije nisu moguće u QBE i moraju se izvršavati korišćenjem SQL.

Neke osnovne načine korišćenja QBE ilustrovati ćemo na nekoliko primera. Pretpostavljamo da su tabele definisane prema istoj relacionoj šemi koju smo koristili za ilustraciju SQL naredbi.

*Primer:* Prikazati šifru, prezime ime, i platu svih nastavnika sa zvanjem docenta.

NASTAVNIK	S_NAS	PREZIME_IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
	P.	P.	DOCENT			P.	

Korisniku se nudi prazna tabela koju on popunjava saglasno potrebama upita. Korisnik može izvršiti redakciju tabele za postavljanje upita saglasno potrebama. U prethodnom primeru se redakcijom mogu isključiti kolone S\_DIR, DATZAP i DODATAK.

Ako želimo prikaz svih kolona iz tabele NASTAVNIK specifikacija P. može se zadati i na sledeći način:

NASTAVNIK	S_NAS	PREZIME	IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
P.								

Podsećanja radi prethodni upit ekvivalentan je u SQL sledećem upitu:

```
SELECT *
  FROM NASTAVNIK;
```

Uslovi koji su povezani operatorom AND zadaju se u jednom redu tabele dok se uslovi povezani operatorom OR zadaju u više redova ili primenom *lemenata veze i bloka uslova*.

*Primer:* Prikazati prezime ime, zvanje i platu nastavnika koji imaju zvanje vanrednog ili redovnog profesora i platu veću od 12500. Rezultujuće n-torce prikazati po abecedi zvanja i prezimena imena nastavnika.

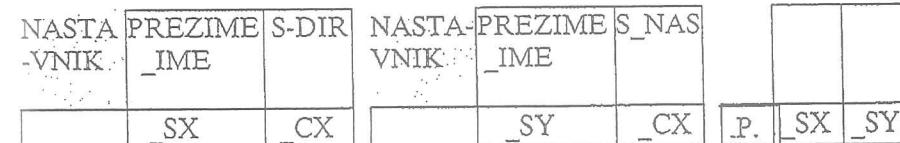
Rešenje:

NASTAVNIK	PREZIME	IME	ZVANJE	PLATA
	P.AO		P. "V PROF"	P. > 12500
	P.AO		P. "R PROF"	P. > 12500

U prethodnom primeru AO označava rastući redosled (DO - označava opadajući redosled). Uslovi navedeni u istom redu povezani su AND operatorom, dok uslovi navedeni u različitim redovima povezani su OR operatorom.

*Primer:* Prikazati prezime ime nastavnika i prezime ime njemu prepostavljenog nastavnika.

Rešenje:



Oznake \_SX, \_CX i \_SY predstavljaju elemente veze. Treća tabela može biti bez naziva i bez naziva kolona, a definije oblik rezultata. To su parovi prezime ime nastavnika (SX)/prezime ime njemu prepostavljenog nastavnika (SY) nastali putem elementa veze \_CX n-torki tabele NASTAVNIK.

*Primer:* Nastavniku sa šifrom 2 upisati zvanje vanrednog profesora i platu povećati 10%.

Rešenje:

NASTAVNIK	S_NAS	ZVANJE	PLATA	PLATA
P.	002	U. "V PROF"	WT	U. WT + 0,1 * WT

Može se uočiti da se oznaka U. (izmena sadržaja) odnosi na zvanje i platu.

*Primer:* U tabeli nastavnik brisati podatke o PETROVIĆ PETRU.

Rešenje:

NASTAVNIK	PREZIME	IME
D.	"PETROVIĆ PETAR"	

Oznaka D. odnosi se na kompletну n-torku i zbog toga je upisana u koloni imena tabele.

Na kraju napomenimo da postoje upiti koji se mogu realizovati u SQL ali ne mogu u QBE. Na primer QBE u DB2 nema odgovarajućih naredbi za sledeće SQL mogućnosti:

- NOT EXISTS,
- standardne funkcije (SUM, AVG itd.),
- GROUP BY i HAVING.

Korišćenjem nekih drugih rutina DB2 može ipak realizovati neke od prethodnih mogućnosti SQL, mada je i SQL deo DB2.

## 9.4. Transformacija upita

Transformacija upita iz jednog oblika u drugi posebno je potrebna iz razloga optimizacije upita radi efikasnog rada i korišćenja računarskog sistema. Optimizacija zahteva za obradom podataka susreće se kod *prevodioca sa optimizacijom* koji menjaju redosled operacija i otklanjaju suvišna izračunavanja. I dok se pri takvoj optimizaciji dobitak iskazuje u mikrosekundama, u slučaju složenih informacionih zahteva na bazi podataka optimizacija može smanjiti broj pristupa spoljnoj memoriji za red veličine, što predstavlja značajnu razliku. Na primer, sasvim je moguće slučaj da je za izvršavanje jednog upita na jedan način potrebno nekoliko časova, a na drugi način nekoliko minuta. Obično prethodna situacija nastaje u slučaju kada se po prvom načunu čitaju skoro svi zapisi baze podataka i za svaki od njih pokušavaju pronaći drugi odgovarajući zapisi da bi se na kraju izabralo samo nekoliko zapisa. Sasvim sigurno brži način sastoji se u pokušaju da se na samom početku izabere manji skup relevantnih zapisa koji se dalje obraduju. Prvi od načina zasniva se na izvršavanju upita na način kako je zapisan, pri čemu po pravilu korisnik koji takav zahtev definiše ne zna da je takav zahtev neefikasan. Dobar sistem treba dozvoliti korisniku definisanje upita kako je to njemu pogodno bez obaveza razmišljanja o efikasnosti pri čemu će sam sistem preuzeti ulogu optimizacije.

Drugi važan razlog transformacije upita su sve više i sve češće korišćene distribuirane baze podataka. Korisnik koji poznaje jedan upitni jezik želi pristupiti udaljenoj bazi podataka u kojoj se koristi drugi upitni jezik. Umesto da takvog korisnika upućujemo na izučavanje tog drugog upitnog jezika, može se realizovati automatsko prevodenje korisnikovog informacionog zahteva na njemu poznatom upitnom jeziku na informacioni zahtev definisan prema upitnom jeziku koji dozvoljava data baza podataka.

Postojanje nekoliko upitnih jezika predstavlja prednost i u slučaju kada korisnik radi sa jednom bazom podataka. U tom slučaju korisnik može odabrati upitni jezik koji njemu odgovara bez obaveze izučavanja novog upitnog jezika.

Najbolje rešenje je realizacija jednog upitnog jezika na najefikasniji način pri čemu bi se informacioni zahtevi na drugim upitnim jezicima prevodili na taj jezik, umesto nezavisne realizacije svakog posebnog upitnog jezika. U većini

danas korišćenih sistema za upravljanje relacionim bazama podataka taj opšti upitni jezik zasniva se na relationalnoj algebri.

Načini formulisanja upita na relationalnoj algebri i relacionom računu su sasvim različiti. U praksi se potreba prevodenja iz relacione algebre na relacioni račun javlja vrlo retko. Većina relationalnih SUBP ima mogućnost izvršavanja naredbi kao što je spajanje pa je problem pre u prevodenju izraza na relacionom računu u izraze relacione algebre koji su pogodni za realizaciju. Detaljna razmatranja prevodenja izraza relacione algebre u izraze relationalnog računa i obrnuto nećemo razmatrati.

Značajni efekti u optimizaciji upita na relationalnoj algebri često se mogu postići jednostavnom променом načina sračunavanja izraza (promena redosleda izvodjenja operacija). Na primer, izvodjenjem operacije selekcije pre operacije spajanja često možemo doći brže do rezultata. To proizilazi iz toga što operacije tipa spajanja i Dekartovog proizvoda generišu veliki broj torki. Ako pre izvodjenja takvih operacija možemo izvršiti selekciju tada možemo značajno smanjiti broj generisanih torki. Ovo je jedan od primera primene opštег principa: uvek treba koristiti postojeće informacije datog zadatka da bi se smanjila količina informacija na izlazu generatora, umesto da se prvo formira izlaz svih informacija iz kojeg će zatim biti odstranjene nepotrebne informacije.

Prethodni princip zasniva se na sledećem pravilu:

Neka su  $r(R)$  i  $p(P)$  dve tabele i neka uslov selekcije F ne sadrži obeležja tabele  $p$ . Tada vredi pravilo o ekvivalentnosti izraza:

$$\sigma_F(r \Delta p) = \sigma_F(r) \Delta p.$$

Prethodni izraz možemo iskazati rečima na sledeći način: selekcija prirodnog spoja tabela  $r$  i  $p$  prema uslovu F jednaka je prirodnom spoju selekcije nad tabelom  $r$  prema uslovu F i tabele  $p$ .

Dokaz prethodnog pravila zasniva se na komutativnosti logičke i operacije.

Razmotrimo sada jedan upit koji treba izraziti sa operatorima relacione algebre nad ranije definisanim relacionim šemama NASTAVNIK, PREDMET i PREDAJE.

**Primer:** Korišćenjem operatora relacione algebre prikazati šifru predmeta, šifru nastavnika i prezime ime nastavnika koji predaje predmet sa šifrom 1.

Rešenje:

a) Upit izražen operacijama u relacionoj algebri glasi:

$$\pi_{S\_PRED \ S\_NAS \ PREZIME\_IME} (\sigma_{S\_PRED=1} (NASTAVNIK \ \nabla \ PREDAJE)).$$

Rezultat:

(S_PRED S_NAS PREZIME_IME)		
1	2	PETROVIC PETAR

Do željenog rezultata dolazimo izvodjenjem sledećih operacija:

(1) Prirodni spoj tabela NASTAVNIK i PREDAJE.

(1.1) Dekartov proizvod tabela NASTAVNIK i PREDAJE  
(rezultat 45 torki)

(1.2) Selekcija nad tabelom dobijenom u (1.1) prema uslovu  
NASTAVNIK.S\_NAS = REDAJE.S\_NAS (rezultat 5 torki)

(1.3) Projekcija po obeležjima koja se nalaze u tabeli NASTAVNIK i  
obeležjima koja se nalaze u tabeli PREDAJE a ne nalaze se u  
tabeli NASTAVNIK (eliminisanje istih kolona) (rezultat 5 torki).

(2) Selekcija torki iz rezultata dobijenog u (1) prema uslovu S\_PRED = 1  
(rezultat 1 torka).

(3) Projekcija po obeležjima S\_PRED, S\_NAS i PREZIME\_IME nad  
tabelom dobijenom u (2) (rezultat 1 torka).

b) Korišćenjem prethodno razmatranog pravila isti rezultat možemo dobiti  
ako zahtev iskažemo sledećim izrazom u relacionoj algebri:

$$\pi_{S\_PRED \ S\_NAS \ PREZIME\_IME} ((\sigma_{S\_PRED=1} (PREDAJE)) \ \nabla \ NASTAVNIK).$$

Rezultat:

$$(S\_PRED \ S\_NAS \ PREZIME\_IME)$$

1	2	PETROVIC PETAR
---	---	----------------

Do prethodnog rezultata možemo doći izvodjenjem sledećih operacija:

(1) Selekcija nad tabelom PREDAJE prema uslovu S\_PRED = 1  
(rezultat 1 torka odnosno tabela sa oznakom TEMP).

(2) Prirodni spoj tabele NASTAVNIK i tabele dobijene kao rezultat u  
koraku (1).

(2.1) Dekartov proizvod tabele NASTAVNIK i rezultata iz koraka (1)  
(rezultat 9 torki)

(2.2) Selekcija nad tabelom dobijenom u (2.1) prema uslovu  
NASTAVNIK.S\_NAS = TEMP.S\_NAS (rezultat 1 torka)

(2.3) Projekcija po obeležjima koja se nalaze u tabeli NASTAVNIK  
i obeležjima koja se nalaze u tabeli TEMP a ne nalaze se u  
tabeli NASTAVNIK (eliminisanje istih kolona) (rezultat 1 torka).

(3) Projekcija po obeležjima S\_PRED, S\_NAS i PREZIME\_IME nad  
tabelom dobijenom u (2) (rezultat 1 torka).

Iz prethodnog primera možemo uočiti da se prema rešenju a) (prvo  
izvršavanje spoja) kao prvi medjurezultat dobija tabela od 45 torki, dok se  
prema rešenju b) (prvo izvršavanje selekcije) kao medjurezultat dobija tabela  
sa 9 torki. Sasvim je očigledno da se do rezultata brže dolazi sračunavanjem  
izraza pod b).

Prethodno pravilo na kojem se zasniva transformacija jednog izraza relacione  
algebre u drugi možemo uopštiti ako uslov selekcije F predstavimo kao  
konjukciju sledeća četiri uslova:

- $F_r$  koji se odnosi samo na tabelu  $r$ ,
- $F_p$  koji se odnosi samo na tabelu  $p$ ,
- $F_q$  koji se odnosi na obeležja (kolone) zajednička za tabele  $r$  i  $p$ ,
- $F_s$  koji sadrži sve što nije moguće uključiti u  $F_r$ ,  $F_p$  ili  $F_q$ .

Svaki od uslova  $F_r$ ,  $F_p$ ,  $F_q$  ili  $F_s$  može biti izostavljen što znači da se selekcija ne  
izvršava.

Sada imamo:

$$\sigma_{F_r \wedge F_p \wedge F_q \wedge F_s} (r \nabla p) = \sigma_{F_s} (\sigma_{F_r \wedge F_q} (r) \nabla \sigma_{F_p \wedge F_q} (p)).$$

Prethodna razmatranja optimizacije izraza relacione algebre imala su za cilj da ilustruju moguće posledice različito iskazanog istog upita. Kao što je bilo napomenuto od korisnika se ne može očekivati potpuno poznavanje svih mogućih principa optimizacije već se isti mogu ugraditi u SUBP i automatski izvoditi.

## 10. PREVODJENJE MODELA OBJEKTI-VEZE OBELEŽJA U RELACIONI MODEL

Pri prevodenju jednog modela podataka u drugi ne znači da svakom konceptu strukture jednog modela mora odgovarati koncept strukture drugog modela, svakoj operaciji jednog modela da mora odgovarati operacija drugog modela i svakom eksplicitnom ograničenju jednog modela da mora odgovarati eksplicitno ograničenje drugog modela.

Kod prevodenja jednog semantički bogatog modela (modela III generacije) u semantički manje bogat model (model II generacije) dolaze do izražaja prethodno navedena neslaganja. Po pravilu se deo semantike iskazan strukturon semantički bogatog modela mora predstaviti eksplicitnim ograničenjima semantički manje bogatog modela. Iz toga neposredno sledi da apstraktnoj operaciji semantički bogatog modela odgovara procedura semantički manje bogatog modela.

S obzirom da za sada ne postoji komercijalno raspoloživ SUBP zasnovan na MOV to je u praktičnoj realizaciji potrebno model podataka izradjen prema MOV prevesti u neki od modela za koji postoji SUBP.

Kako za relacioni model podataka postoji više komercijalnih softvera prevodenje iz MOV najčešće se vrši u relacioni model podataka.

Model objekti - veze - obeležja je semantički bogat model (model III generacije), dok je relacioni model semantički manje bogat (model II generacije), pa sve do sada rečeno za prevodenje semantički bogatog u semantički manje bogat model u potpunosti važi i za njih.

Prevodenje MOV u relacioni model izvodi se na sledeći način:

1. Deo strukture MOV, odnosno DOV se predstavlja relacionom šemom.
2. Ograničenja, operacije i deo strukture MOV se predstavljaju operacijama za očuvanje integriteta definisanim nad relacionim modelom, a implementiraju korišćenjem sredstava odabranog relacionog SUBP.

Ovde se razmatraju odnose samo na prevodjenje DOV u relacioni model. Kao rezultat prevodjenja DOV dobija se relaciona šema baze podataka sa odgovarajućim pravilima integriteta. Na osnovu skupa pravila koja se u prevodjenju primenjuju, postupak prevodjenja DOV u relacioni model može biti formalizovan i automatizovan. Postoje više postupaka za prevodjenje DOV u relacioni model podataka, a ovde se izlaže jedan praktičan postupak koji se može i automatizovati.

## 10.1 Postupak prevodjenja

### 1. Pravila za objekte:

1.1. Svaki objekat iz DOV postaje šema relacije. Ime tipa objekta postaje ime šeme relacije. Obeležja objekta su obeležja šeme relacije. Za osnovne objekte identifikator objekta postaje primarni ključ šeme relacije.

1.2. Svaki slab objekat takođe postaje šema relacije. Ime slabog objekta postaje ime šeme relacije. Obeležja objekta su obeležja šeme relacije. Identifikator nadredjenog objekta postaje obeležje šeme relacije koja odgovara slabom objektu. Identifikator slabog objekta čini identifikator nadredjenog objekta i obeležja slabog objekta koja jedinstveno identificuju pojavljivanje slabog objekta u okviru pojavljivanja njemu nadredjenog objekta.

Na primer, za DOV dat na slici 7.7. prevodjenjem u relacioni model primenom do sad definisanih pravila dobijamo sledeće šeme relacija:

DRŽAVA(S\_DRŽ, NAZIV\_DRŽ) (prema pravilu 1.1)  
PROIZVODJAČ(S\_DRŽ, S\_PRO, NAZIV\_PRO) (prema pravilu 1.2)

1.3. Objekat nadtip (generalizovani tip objekta) postaje šema relacije. Ime nadtipa postaje ime šeme relacije. Obeležja nadtipa su obeležja šeme relacije. Identifikator nadtipa postaje ključ šeme relacije.

1.4. Objekat podtip takođe postaje šema relacije. Ime podtipa postaje ime šeme relacije. Obeležja podtipa su obeležja šeme relacije. Identifikator nadtipa postaje obeležje šeme relacije podtipa i predstavlja ključ šeme relacije.

DOV dat na slici 7.9. prevodi se u relacioni model sa sledećim šemama relacija:

RADNIK(JMBG, IME, DATUM_RODJ,	VRSTA_POSLA),	(prema pravilu 1.3)
NASTAVNIK(JMBG, ZVANJE, DATUM_IZB),		(prema pravilu 1.4)
SLUŽBENIK(JMBG, ST_SPREMA),		(prema pravilu 1.4)
ODRŽAVANJE(JMBG, ŽANIMANJE),		(prema pravilu 1.4)

DOV dat na slici 7.11. prevodimo u relacioni model sa sledećim šemama relacija:

RADNIK(JMBG, IME, DATUM_RODJ,	VRSTA_POSLA),	(prema pravilu 1.3)
RADNIK_STUDENT(JMBG, FAKULTET,	DATUM_UPISA),	(prema pravilu 1.4)
RADNIK_SPORTISTA(JMBG, SPORT).		(prema pravilu 1.4)

### 2. Pravila za binarne veze:

#### 2.1. Veza sa kardinalnošću 1:1

Veze tipa 1:1 po pravilu nemaju obeležja. Sva obeležja koja bi eventualno mogla biti pripisana samoj vezi, zapravo su obeležja jednog od objekata koji čine tu vezu. Dakle mogu biti pripisana tom objektu, a time postati obeležja šeme relacije kojom se taj tip objekta predstavlja.

2.1.1. Vezu sa kardinalnošću (1,1) : (1,1), i oba objekta koji u njoj učestvuju prevodimo u jednu šemu relacije čija su obeležja sva obeležja jednog i drugog objekta. Kandidati za ključ u ovoj šemi relacije su identifikatori jednog i drugog objekta koji su u vezi.

2.1.2. Vezu sa kardinalnošću (0,1) : (1,1), i objekte u vezi prevodimo u dve šeme relacije. Za svaki objekat u vezi po jedna šema relacije (prema već definisanom pravilu 1.1), s tim što se identifikator jednog od objekta koji su u vezi ubaci za obeležje druge šeme relacije. Dakle veza se predstavlja *spojnjim ključem*.

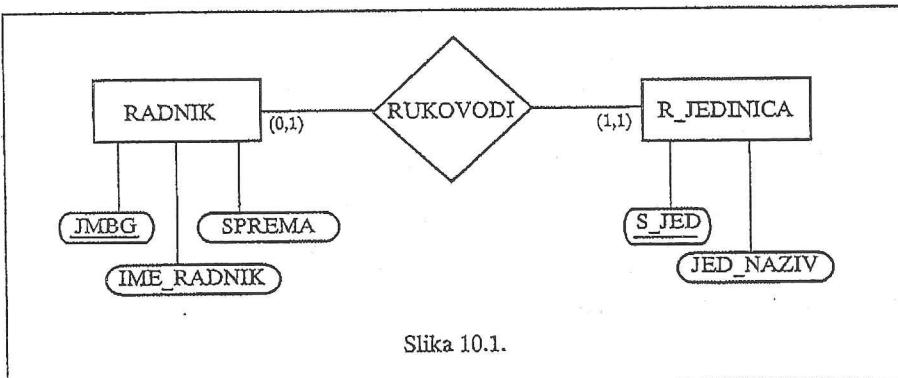
Šemu relacije u koju ćemo uvrstiti spoljni ključ biramo tako da relacija sadrži što manje nul - vrednosti i da njeno korišćenje bude što efikasnije.

DOV dat na slici 10.1. prevodimo u relacioni model sa sledećim šemama relacija:

R\_JEDINICA(S\_IED, JED\_NAZIV) (prema pravilu 1.1)  
RADNIK(JMBG,IME\_RADNIK,SPREMA,S\_JED) (prema pravilu 1.1 i 2.1.2)

ili

R\_JEDINICA(S\_JED, JED\_NAZIV, JMBG) (prema pravilu 1.1 i 2.1.2)  
RADNIK(JMBG, IME\_RADNIK, SPREMA) (prema pravilu 1.1)



Slika 10.1.

Šeme relacija pod a) uzrokovale bi da svaka n - torka za svakog radnika koji nije rukovodilac ima nul - vrednost obeležja S\_JED (tj. spoljni ključ).

Prema šemama relacija pod b), s obzirom da radna jedinica obavezno ima rukovodioca, predstavljanje veze RUKOVODI spoljnjim ključem u šemi relacije R\_JEDINICA ne dovodi do pojave nul - vrednosti.

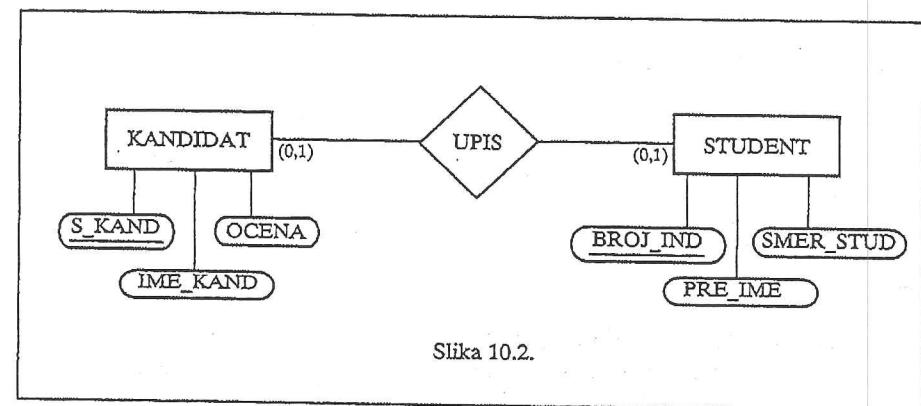
Prema tome, pravilo za prevodjenje veze sa kardinalnošću (0,1) : (1,1) je njeno predstavljanje spoljnjim ključem u šemi relacije objekta sa strane (1,1).

**2.1.3.** Za vezu sa kardinalnošću (0,1) : (0,1), kreiraju se tri šeme relacija. Po jedna za svaki objekat (prema već definisanom pravilu 1.1) i jedna za vezu. Obeležja u šemi relacije koja odgovaraju vezi su i identifikatori objekata koji su u vezi i oba su kandidati za ključ.

DOV dat na slici 10.2. prevodimo u relacioni model sa sledećim šemama relacija:

KANDIDAT(S\_KAND, IME\_KAND, OCENA), (prema pravilu 1.1)  
STUDENT(BROJ\_IND, PRE\_IME, SMER\_STUD) (prema pravilu 1.1)  
UPIS(BBROJ\_IND, S\_KAND) (prema pravilu 2.1.3)

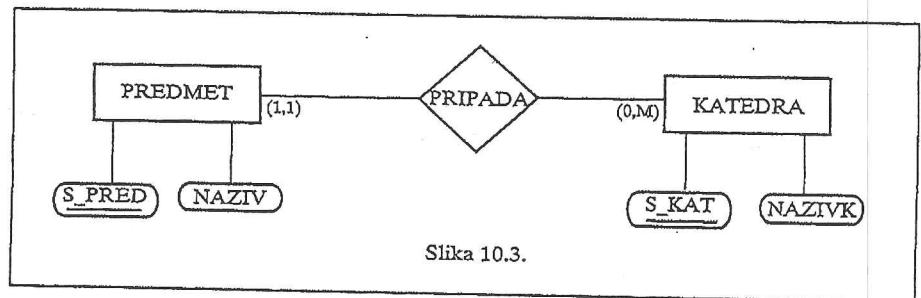
Alternativno šema relacije UPIS može za primarni ključ imati S\_KAND umesto BROJ\_IND.



Slika 10.2.

**2.2.** Veze sa kardinalnošću (1,1) : (0,M) i (1,1) : (1,M), ne postaju posebne šeme relacija. Identifikator objekta sa strane za koju je GG = M postaje obeležje šeme relacije koja odgovara objektu sa strane za koju je GG = 1. DOV dat na slici 10.3. prevodimo u relacioni model sa sledećim šemama relacija:

PREDMET(S\_PRED, NAZIV, S\_KAT) (prema pravilu 1.1 i 2.2)  
KATEDRA(S\_KAT, NAZIVK) (prema pravilu 1.1).

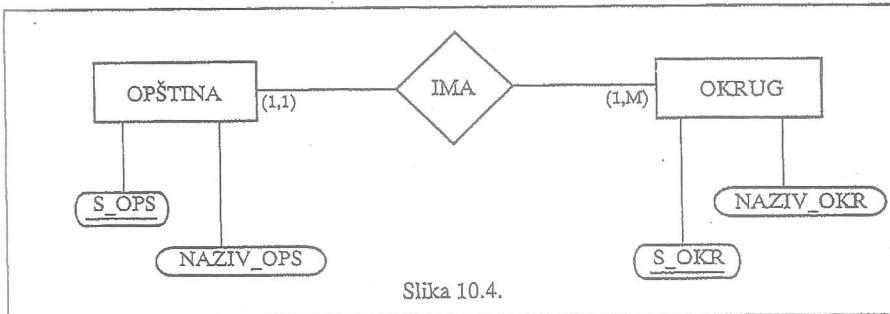


Slika 10.3.

DOV dat na slici 10.4. prevodimo u relacioni model sa sledećim šemama relacija:

OPSTINA(S\_OPS, NAZIV\_OPS, S\_OKR) (prema pravilu 1.1 i 2.2)  
OKRUG(S\_OKR, NAZIV\_OKR) (prema pravilu 1.1).

2.3. Veza izmedju nadredjenog i slabog objekta kao i veza izmedju nadtipa i podtipa ne postaju posebne šeme relacija. One su već ostvarene pravilima 1.2, 1.3 i 1.4.



Slika 10.4.

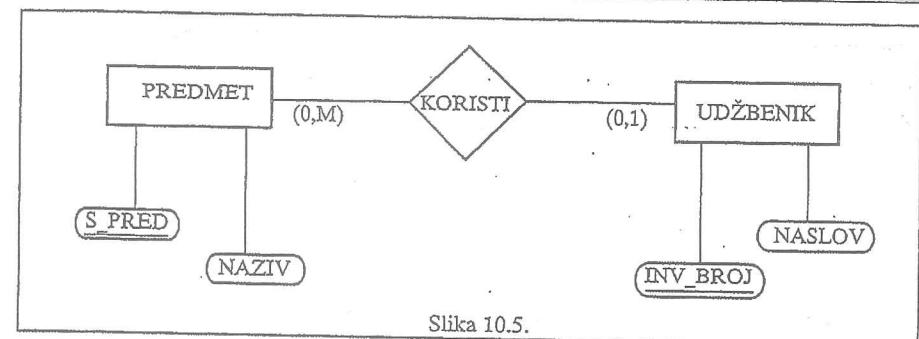
2.4. Veze sa kardinalnošću (0,1) : (0,M) i (0,1) : (1,M), postaju posebna šema relacija. Obeležja ove šeme relacije su identifikatori objekata koji su u vezi, a ključ šeme relacije je identifikator objekta za koji je GG = 1.

DOV dat na slici 10.5. prevodimo u relacioni model sa sledećim šemama relacija:

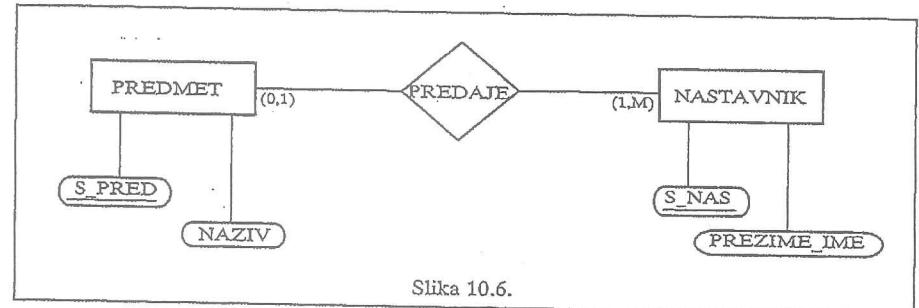
PREDMET(S\_PRED, NAZIV) (prema pravilu 1.1)  
UDŽBENIK(INV\_BROJ, NASLOV)  
KORISTI(INV\_BROJ, S\_PRED) (prema pravilu 2.4)

DOV dat na slici 10.6. prevodimo u relacioni model sa sledećim šemama relacija:

PREDMET(S\_PRED, NAZIV) (prema pravilu 1.1)  
NASTAVNIK(S\_NAS, PREZIME\_IME)  
PREDAJE(S\_PRED, S\_NAS) (prema pravilu 2.4)

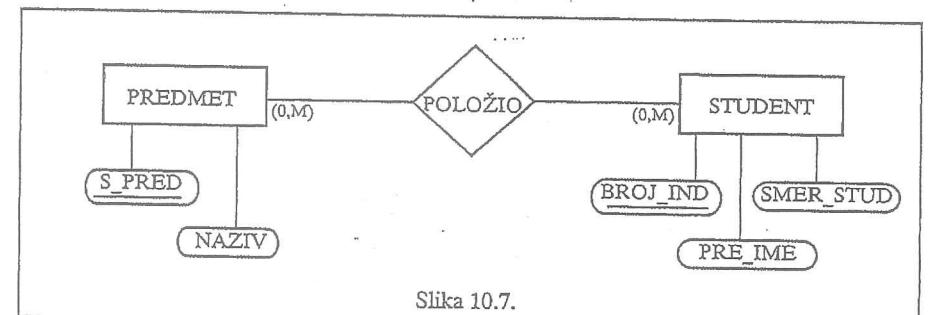


Slika 10.5.



Slika 10.6.

2.5. Veze sa kardinalnošću preslikavanja (0,M) : (0,M), (1,M) : (0,M), i (1,M):(1,M) postaju posebne šeme relacija. Obeležja ove šeme relacije su identifikatori objekata koji su u vezi, a ključ šeme relacije je složeni ključ koji se sastoji od identifikatora objekata koji su u vezi.



Slika 10.7.

DOV dat na slici 10.7. prevodimo u relacioni model sa sledećim šemama relacija:

PREDMET(S\_PRED, NAZIV) (prema pravilu 1.1)  
 STUDENT(BROJ\_IND, PRE\_IME, SMER\_STUD) (prema pravilu 1.1)  
 POLOŽIO(BROJ\_IND, S\_PRED) (prema pravilu 2.5).

2.6. Agregirani objekat (mešovit tip objekat-veza) se posmatra na isti način kao i odgovarajuća veza. Ukoliko veza poseduje obeležja ista postaju obeležja šeme relacije veze kada se veza prevodi u posebnu šemu relacije ili se uključuju u onu šemu relacije u koju se upisuje spoljni ključ.

DOV dat na slici 7.13. prevodimo u relacioni model sa sledećim šemama relacija:

STUDENT(BROJ\_IND, PRE\_IME) (prema pravilu 1.1)  
 PREDMET(S\_PRED, NAZIV) (prema pravilu 1.1)  
 NASTAVNIK(S\_NAS, PREZIME\_IME) (prema pravilu 1.1)  
 OCENJUJE(BROJ\_IND, S\_PRED, S\_NAS) (prema pravilu 2.5)  
 DIPLOMSKI(BROJ\_IND, S\_PRED, DATUM\_DIPL) (prema pravilu 2.6).

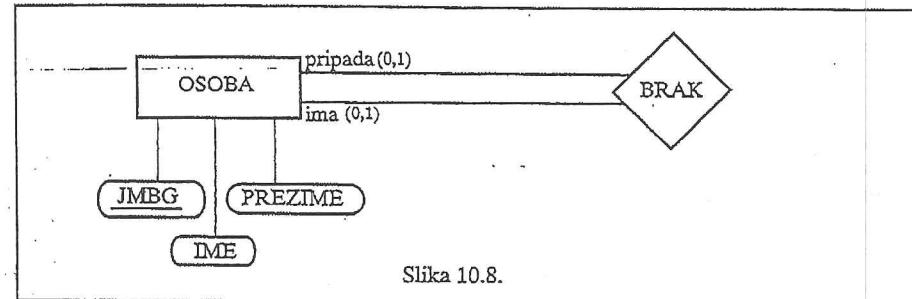
### 3. Pravila za unarne veze

Prevodjenje unarnih veza (unarnom nazivamo vezu izmedju dva objekata istog tipa) u relacioni model podataka zavisi od kardinalnosti tipa veze i izvodi se kao i za druge tipove ranije opisanih binarnih veza. Napomenimo da kod unarne veze tipa (1:1) parcijalnost samo na jednoj strani veze, odnosno totalnost samo na jednoj strani veze, ne bi imala smisla. Naime, time bi se istom tipu objekta istovremeno dopušтало и poricalo opciono učestvovanje u vezi.

Pri prevodjenju unarnih veza s obzirom da bi spoljni ključ u šemi relacije imao isto ime kao i primarni ključ, vršimo njegovo preimenovanje.

Model podataka dat DOV na slici 10.8. predstavlja situaciju shvatanja braka u našim uslovima. Naime, jedna osoba nije u vezi ni sa jednom drugom osobom (nije u braku), ili je u braku sa samo jednom osobom. Prevodjenjem u relacioni model dobijamo sledeće šeme relacija:

OSOBA(JMBG, IME, PREZIME) (prema pravilu 1.1)  
 BRAK(JMBG, JMBG\_BRAČNI\_DRUG) (prema pravilu 2.1.3)



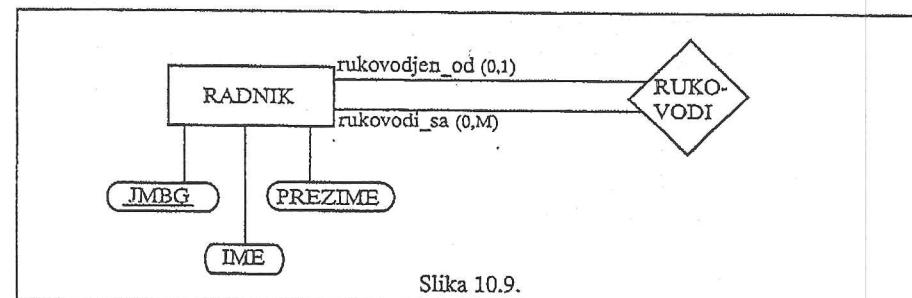
Slika 10.8.

U šemi relacije BRAK za ključ relacije mogli smo odabrat JMBG\_BRAČNI\_DRUG umesto JMBG.

Model podataka dat DOV na slici 10.9. predstavlja situaciju u kojoj jedan radnik može da rukovodi sa više radnika i može imati jednog nadredjenog rukovodioca. Svaki radnik ne mora imati nadredjenog rukovodioca i svaki radnik ne mora biti rukovodioc.

Prevodjenjem u relationalni model dobijamo sledeće šeme relacija:

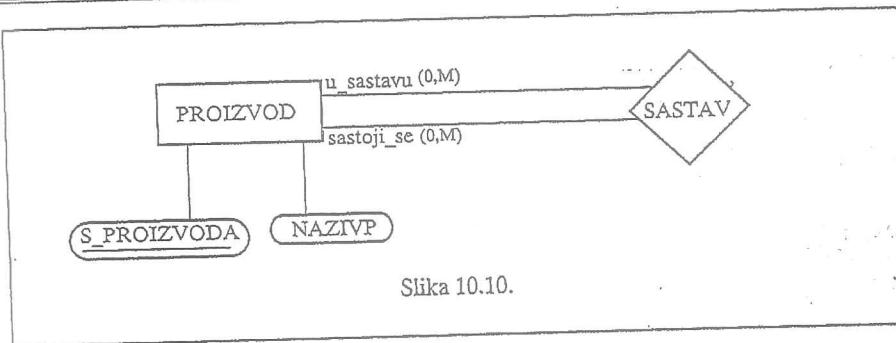
RADNIK(JMBG, IME, PREZIME) (prema pravilu 1.1)  
 RUKOVODI(JMBG, JMBG\_RUKOVODIOC) (prema pravilu 2.4).



Slika 10.9.

Model podataka dat DOV na slici 10.10. predstavlja model sastavnice u kojem jedan proizvod može sam da predstavlja *sastavljeni objekat* ili se sastoji od jednog ili više objekata koji *ulaze u sastav*.

Prevodjenjem u relationalni model dobijamo sledeće šeme relacija:  
 PROIZVOD(S\_PROIZVODA, NAZIVP) (prema pravilu 1.1)  
 SASTAV(S\_PROIZVODA, S\_PROIZVODA\_S) (prema pravilu 2.5).



Slika 10.10.

**Rezime postupka prevodjenja:**

Pravila se ne moraju primenjivati redom kako su zadata, već se prevodjenje vrši primenjujući sva odgovarajuća pravila odjednom na pojedine objekte i veze u sistemu, po redosledu koji je obično očigledan iz samog modela. Za svaki objekat se, pored vrste kojoj pripada (pravila 1.1, 1.2, 1.3 i 1.4), utvrđuje i da li će prilikom prevodjenja veze doći do prostiranja ključa ili će se formirati posebna šema relacije.

Kada se na ovaj način postupi sa svim objektima, prevode se preostale veze po pravilima 2.4, 2.5 i 2.6.

Pod pretpostavkom da je DOV korektno urađen njegovim prevodjenjem u relacioni model uvek se dobijaju šeme relacija koje su u trećoj normalnoj formi u kojima ne postoji nula - vrednosti kao rezultat "neprimenjivog svojstva".

Kao što je napred bilo rečeno MOV je semantički bogatiji od relacionog modela podataka. Na primer, sa DOV slike 10.6. se vidi da svaki NASTAVNIK predaje bar jedan PREDMET. Iz odgovarajućih šema relacija u relacionom modelu to se ne može zaključiti. Zbog toga se pri prevodjenju MOV u relacioni model, pored šema relacija koje se dobijaju primenom MOV u relationalni model, pored šema relacija koje se dobijaju primenom navedenih pravila, mora definisati i skup posebnih pravila integriteta, kako bi se zadržala semantika MOV.

Posebna pravila integriteta mogu biti statička i dinamička. Statičkim pravilima integriteta određeni su uslovi koji moraju važiti pre i posle izvršenja bilo koje operacije nad bazom podataka. Dinamičkim pravilima integriteta definisane

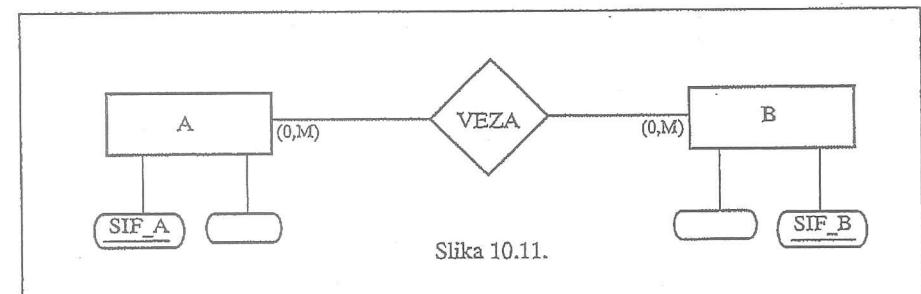
su procedure u relacionom modelu, koje odgovaraju apstraktnim operacijama MOV-a i koje garantuju ostvarenje uslova integriteta.

Do statičkih i dinamičkih pravila integriteta dolazimo posmatranjem jedne veze i objekata koji u njoj učestvuju. Na primer, na slici 10.11. dat je DOV u kome su objekti A i B povezani vezom (0,M):(0,M) parcijalnom sa obe strane. Šeme relacija koje prema pravilima prevodjenja odgovaraju prikazanom DOV su:

A (SIE\_A, ...)  
B (SIE\_B, ...)  
VEZA (SIE\_A, SIE\_B).

Neka je  $a$  vrednost obeležja SIF\_A, a  $b$  obeležja SIF\_B u šemi relacije VEZA. Pravilo integriteta za dati tip veze dva objekta može se iskazati zahtevom da vrednost  $a$  mora postojati kao vrednost ključa u relaciji A i vrednost  $b$  mora postojati kao vrednost ključa u relaciji B. Ovaj integritet odgovara referencijalnom integritetu relacionog modela i može se zapisati na sledeći način:

1. VEZA [SIF\_A]  $\subseteq$  A [SIF\_A],
2. VEZA [SIF\_B]  $\subseteq$  B [SIF\_B].



Slika 10.11.

Kako oba iskaza 1. i 2. moraju biti ispunjena, podrazumevamo da je između njih logički AND operator.

Pored navedenog načina specifikacije statičkog integriteta isti se može iskazati i pomoću naredbi SQL.

SQL naredbe za iskaze 1. i 2. glase:

```
SELECT DISTINCT SIF_A
  FROM VEZA
 IN
SELECT DISTINCT SIF_A
  FROM A;
```

U opštem slučaju, operaciji u MOV ne odgovara operacija u relacionom modelu već procedura. Procedura se sastoji od niza akcija koje su ili izvršne ili proveravaju odredjene uslove. Za operaciju u MOV u proceduri se pojavljuju kao akcije odgovarajuće operacije relacionog modela. Na primer, operaciji MOV UPIŠI objekat... odgovara akcija iskazana operacijom INSERT INTO relacija....

Za predstavljanje procedure tj. specifikaciju dinamičkih uslova integriteta koristićemo SQL.

Za DOV na slici 10.11. dinamički integritet može se specificirati na sledeći način:

UPIŠLA(SIF\_A:a)

```
INSERT INTO A(SIF_A,...)
VALUES (a,...);
```

BRIŠLA(SIF\_A:a)

1. DELETE A  
WHERE SIF\_A = a;
2. DELETE VEZA  
WHERE SIF\_A = a;

UPIŠLB(SIF\_B:b)

```
INSERT INTO B(SIF_B,...)
VALUES (b,...);
```

BRIŠLB(SIF\_B:b)

1. DELETE B  
WHERE SIF\_B = b;
2. DELETE VEZA  
WHERE SIF\_B = b;

UPIŠI VEZA(SIF\_A:a,SIF\_B:b)

1. INSERT INTO A(SIF\_A)
 

```
SELECT a
      FROM TEST
      WHERE a NOT IN (SELECT DISTINCT SIF_A FROM A);
```
2. INSERT INTO B(SIF\_B)
 

```
SELECT b
      FROM TEST
      WHERE b NOT IN (SELECT DISTINCT SIF_B FROM B);
```
3. INSERT INTO VEZA(SIF\_A, SIF\_B)
 

```
VALUES (a, b);
```

U operacijama 1. i 2. TEST je pomoćna relacija čija je namena da zadovolji formalnu strukturu SELECT naredbe. Pomoćna tabela TEST ima samo jedno obeležje dužine jednog karaktera i sa samo jednom unetom n-torkom.

Operacijom 1. proverava se da li u relaciji A postoji n-torka sa SIF\_A = a, i ukoliko ne postoji, u relaciju A upisuje se takva n-torka. Operacija 2. radi slično sa relacijom B.

Operacija 3. u relaciju VEZA upisuje zahtevanu n-torku.

BRIŠI VEZA(SIF\_A:a,SIF\_B:b)

```
DELETE VEZA
WHERE SIF_A = a AND SIF_B = b;
```

Sledeća statička pravila integriteta koja moraju važiti u relacionom modelu, a koja proizilaze iz DOV su:

**Pravilo I1 - integritet objekta.** Vrednost primarnog ključa kao celine, i niti jedne njegove komponente ne sme biti jednaka nul - vrednosti.

**Pravilo I2 - integritet slabog objekta.** Neka je SIF\_A obeležje u šemci relacije slabog objekta koja je dobijena primenom pravila prevodjenja 1.2 (prevodjenje identifikaciono zavisnog objekta) i koje odgovara ključu nadredjenog objekta. Tada obeležje SIF\_A ne može dobiti neku vrednost (npr. a), ako ta vrednost a ne postoji kao vrednost ključa u relaciji nadredjenog objekta.

za S\_DRZ ako se ta ista vrednost u relaciji podtipa ne može da postoji neka ista vrednost ne javlja kao vrednost ključa u relaciji podtipa.

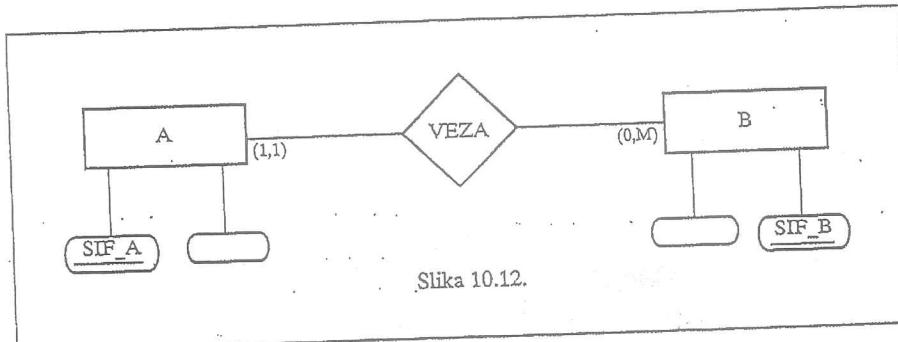
Na primer, za DOV dat na slici 7.9. statički uslov integriteta glasi:

NESTAVNIK [JMBG]  $\subseteq$  RADNIK [JMBG],  
SLUŽBENIK [JMBG]  $\subseteq$  RADNIK [JMBG],  
ODRŽAVANJE [JMBG]  $\subseteq$  RADNIK [JMBG].

Pravilo I4 - integritet veze 1:M.

I4.1 - veze sa kardinalnošću (1,1) : (0,M) slika 10.12.

Ne može postojati neka vrednost za SIF\_B (npr. b) u relaciji A, ako se ta ista vrednost b ne javlja kao vrednost ključa u relaciji objekta B.



Na primer, za DOV dat na slici 10.12. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

PREDMET [S\_KAT]  $\subseteq$  KATEDRA [S\_KAT].

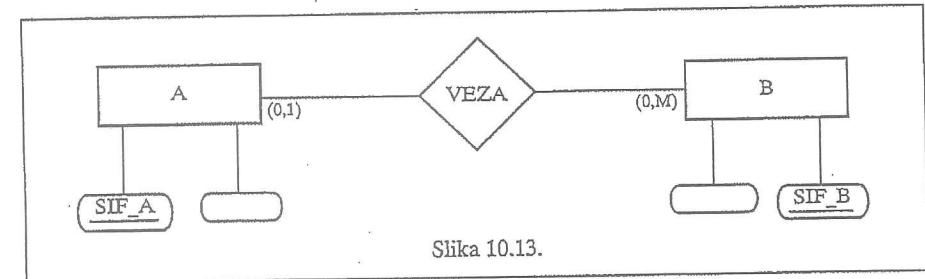
I4.2 - veza sa kardinalnošću (0,1) : (0,M) slika 10.13.

Neka vrednost obeležja SIF\_A (npr. a) i neka vrednost obeležja SIF\_B (npr. b) u relaciji koja odgovara vezi tih objekata ne mogu da postoje ako ne postoji ista vrednost a obeležja SIF\_A u relaciji objekta A, odnosno ista vrednost b obeležja SIF\_B u relaciji objekta B.

Na primer, za DOV dat na slici 10.13. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

A (SIF_A, ...)	(prema pravilu 1.1)
B (SIF_B, ...)	(prema pravilu 1.1)
VEZA (SIF_A, SIF_B)	(prema pravilu 2.4)

VEZA [SIF\_A]  $\subseteq$  A [SIF\_A],  
VEZA [SIF\_B]  $\subseteq$  B [SIF\_B].



Na primer, za DOV dat na slici 10.5. statički uslovi integriteta su:

KORISTI [INV\_BROJ]  $\subseteq$  UDŽBENIK [INV\_BROJ],  
KORISTI [S\_PRED]  $\subseteq$  PREDMET [S\_PRED].

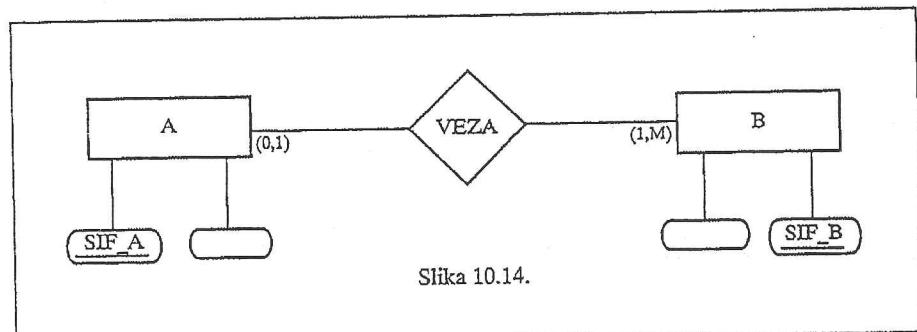
neka vrednost obeležja SIF\_B (npr. b), ako se ta ista vrednost b obeležja SIF\_B ne javlja i u veznoj relaciji. Radi se o slučaju *uzajamnih integriteta* koji zahteva istovremeno ažuriranje relacije objekta B i vezne relacije.

Pored ovoga, neka vrednost obeležja SIF\_A (npr. a) u relaciji koja odgovara vezi ne može da postoji, ako u relaciji A ne postoji ista vrednost a obeležja SIF\_A.

Na primer, za DOV dat na slici 10.14. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

A (SIF_A, ...)	(prema pravilu 1.1)
B (SIF_B, ...)	(prema pravilu 1.1)
VEZA (SIF_A, SIF_B)	(prema pravilu 2.4)

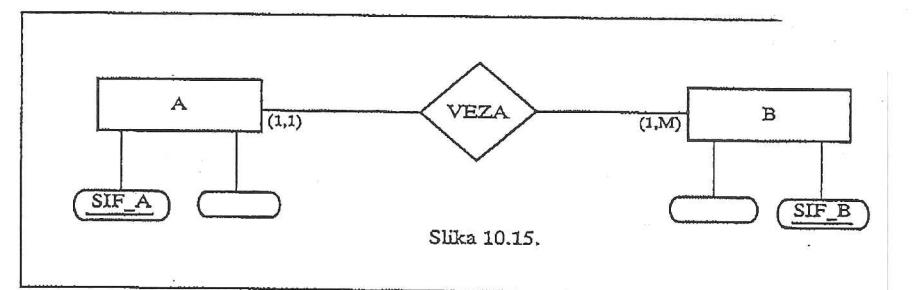
VEZA [SIF\_B] = B [SIF\_B],  
VEZA [SIF\_A] ⊆ A [SIF\_A].



Na primer, za DOV dat na slici 10.6. statički uslovi integriteta su:

PREDAJE [S\_NAS] = NASTAVNIK [S\_NAS],  
PREDAJE [S\_PRED] ⊆ PREDMET [S\_PRED].

Na primer, za DOV dat na slici 10.15. prevodjenje u relacioni model sa statičkim uslovom integriteta glasi:



Slika 10.15.

Na primer, za DOV dat na slici 10.15. prevodjenje u relacioni model sa statičkim uslovom integriteta glasi:

A (SIF_A, SIF_B ...)	(prema pravilu 1.1 i 2.2)
B (SIF_B, ...)	(prema pravilu 1.1)

A [SIF\_B] = B [SIF\_B].

Na primer, za DOV dat na slici 10.4. statički uslov integriteta glasi:

OKRUG [S\_OKR] = OPSTINA [S\_OKR].

**Pravilo 15 - integritet veze za koju oba preslikavanja za gornju granicu imaju vrednost GG = M .**

Neka je SIF\_A ključ jedne, a SIF\_B ključ druge šeme relacije objekata koji su u vezi. Ključ odgovarajuće vezne šeme relacije je složen ključ SIF\_A,SIF\_B.

**I5.1 - veza sa kardinalnošću (0,M) : (0,M)** slika 10.11.

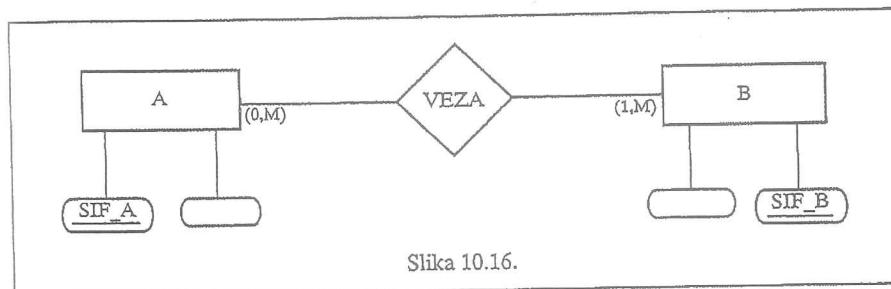
Neka je a vrednost obeležja SIF\_A a, b obeležja SIF\_B u veznoj relaciji. Tada ista ta vrednost a mora postojati kao vrednost ključa u relaciji A i ista ta vrednost b mora postojati kao vrednost ključa u relaciji B.

Na primer, za DOV dat na slici 10.7. statički uslovi integriteta su:

$$\begin{aligned} \text{POLOŽIO [S\_PRED]} &\subseteq \text{PREDMET [S\_PRED]}, \\ \text{POLOŽIO [BROJ\_IND]} &\subseteq \text{STUDENT [BROJ\_IND]}. \end{aligned}$$

**I5.2 - veza sa kardinalnošću (0,M) : (1,M)** slika 10.16.

U veznoj relaciji, ne može postojati vrednost obeležja SIF\_A (npr. a) ako ta ista vrednost a ne postoji kao vrednost ključa SIF\_A u relaciji objekta A. Isto tako, ne može, u veznoj relaciji, postojati vrednost obeležja SIF\_B (npr. b) ako ta ista vrednost ne postoji u relaciji objekta B, kao ključ, i istovremeno, ne može u relaciji objekta B postojati neka vrednost obeležja SIF\_B (npr. b) ako ta ista vrednost b ne postoji kao deo ključa u veznoj relaciji. Očigledno radi se o uzajamnom integritetu, odnosno biće potrebno istovremeno ažuriranje vezne relacije i relacije objekta B.



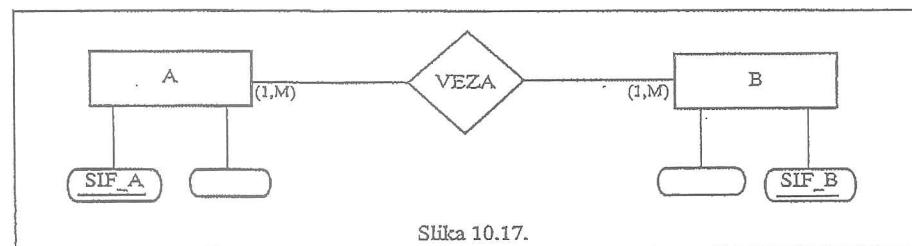
Na primer, za DOV dat na slici 10.16. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

$$\begin{aligned} A (\text{SIF\_A}, \dots) && (\text{prema pravilu 1.1}) \\ B (\text{SIF\_B}, \dots) && (\text{prema pravilu 1.1}) \\ \text{VEZA} (\text{SIF\_A}, \text{SIF\_B}) && (\text{prema pravilu 2.5}) \end{aligned}$$

$$\begin{aligned} \text{VEZA} [\text{SIF\_A}] &\subseteq A [\text{SIF\_A}], \\ \text{VEZA} [\text{SIF\_B}] &\subseteq B [\text{SIF\_B}]. \end{aligned}$$

**I5.3 - veza sa kardinalnošću (1,M) : (1,M)** slika 10.17.

Neka vrednost a obeležja SIF\_A, u veznoj relaciji, ne može da postoji ako ta ista vrednost ne postoji kao vrednost ključa u relaciji odgovarajućeg objekta, i istovremeno, ne može postojati vrednost a ključa SIF\_A u relaciji objekta A, ako se ta ista vrednost ne pojavljuje kao vrednost dela ključa SIF\_A u veznoj relaciji. Isto tako, ne može da postoji, u veznoj relaciji neka vrednost b obeležja SIF\_B ako ta ista vrednost ne postoji kao ključ u relaciji objekta B, i istovremeno, ne može postojati vrednost b ključa SIF\_B u relaciji objekta B, ako ta ista vrednost b nije vrednost dela ključa SIF\_B u veznoj relaciji. Ovde se radi o *dvostrukom uzajamnom integritetu*, odnosno potrebno je istovremeno ažuriranje sve tri relacije.



Slika 10.17.

Na primer, za DOV dat na slici 10.17. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

$$\begin{aligned} A (\text{SIF\_A}, \dots) && (\text{prema pravilu 1.1}) \\ B (\text{SIF\_B}, \dots) && (\text{prema pravilu 1.1}) \\ \text{VEZA} (\text{SIF\_A}, \text{SIF\_B}) && (\text{prema pravilu 2.5}) \end{aligned}$$

$$\begin{aligned} \text{VEZA} [\text{SIF\_A}] &= A [\text{SIF\_A}], \\ \text{VEZA} [\text{SIF\_B}] &= B [\text{SIF\_B}]. \end{aligned}$$

**Pravilo 16 - integritet 1:1 veze.**

**I6.1 - veza sa kardinalnošću (1,1) : (1,1).**

U jedinstvenoj šemi relacije koja se dobija prevodenjem i koja predstavlja objekte u vezi i samu vezu ni SIF\_A ni SIF\_B, kao kandidati za ključ, ne smeju

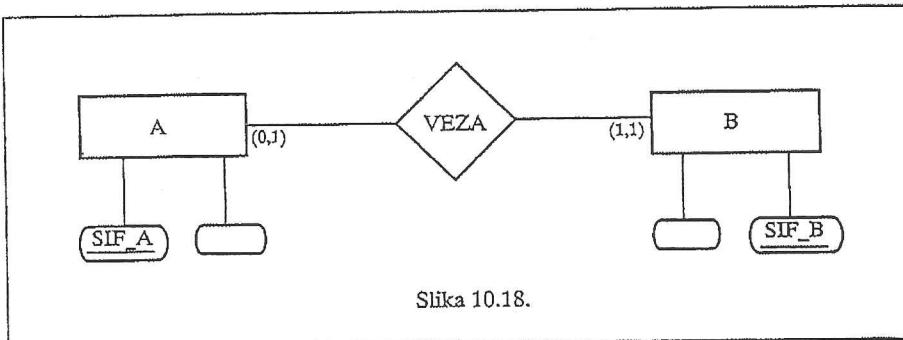
da imaju nul-vrednost. Radi se o proširenju pravila integriteta I1 i na kandidate za ključ.

#### I6.2 - veza sa kardinalnošću (0,1) : (1,1) slika 10.18.

U relaciji objekta B, ne može postojati vrednost obeležja SIF\_A (npr. a) ako ta ista vrednost a ne postoji kao vrednost ključa u relaciji objekta A.

Na primer, za DOV dat na slici 10.18. prevodjenje u relacioni model sa statičkim uslovom integriteta glasi:

$$\begin{array}{ll} A (\text{SIF\_A}, \dots) & \text{(prema pravilu 1.1)} \\ B (\text{SIF\_B}, \text{SIF\_A} \dots) & \text{(prema pravilu 1.1 i 2.1.2)} \\ \\ B [\text{SIF\_A}] \subseteq A [\text{SIF\_A}]. & \end{array}$$

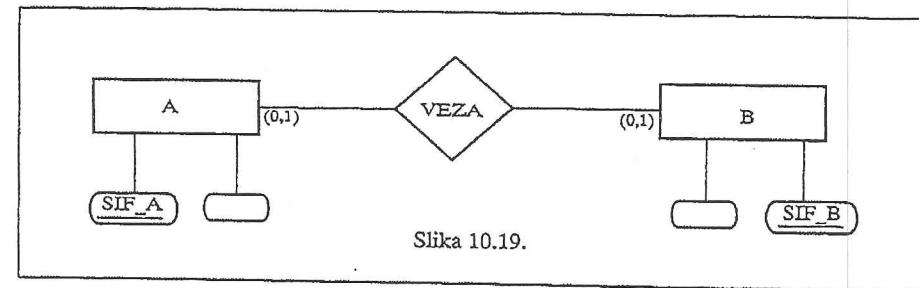


Na primer, za DOV dat na slici 10.1. statički uslov integriteta glasi:

$$\text{R\_JEDINICA} [\text{JMBG}] \subseteq \text{RADNIK} [\text{JMBG}].$$

#### I6.3 - veza sa kardinalnošću (0,1) : (0,1) slika 10.19.

Neka vrednost obeležja SIF\_A (npr. a) ne može postojati u veznoj relaciji ako ta ista vrednost a ne postoji kao vrednost ključa u relaciji objekta A. Isto tako vrednost obeležja SIF\_B (npr. b), ne može da postoji u veznoj relaciji ako ta ista vrednost b ne postoji kao vrednost ključa u relaciji objekta B.



Na primer, za DOV dat na slici 10.19. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

$$\begin{array}{ll} A (\text{SIF\_A}, \dots) & \text{(prema pravilu 1.1)} \\ B (\text{SIF\_B}, \dots) & \text{(prema pravilu 1.1)} \\ \text{VEZA} (\text{SIF\_A}, \text{SIF\_B}) & \text{(prema pravilu 2.1.3)} \end{array}$$

$$\begin{array}{l} \text{VEZA} [\text{SIF\_A}] \subseteq A [\text{SIF\_A}], \\ \text{VEZA} [\text{SIF\_B}] \subseteq A [\text{SIF\_B}]. \end{array}$$

Na primer, za DOV dat na slici 10.2. statički uslovi integriteta su:

$$\begin{array}{l} \text{UPIS} [\text{S\_KAND}] \subseteq \text{KANDIDAT} [\text{S\_KAND}], \\ \text{UPIS} [\text{BROJ\_IND}] \subseteq \text{STUDENT} [\text{BROJ\_IND}]. \end{array}$$

Kada su odredjeni uslovi integriteta za svaku vezu i objekte koji je čine vrši se objedinjavanje uslova integriteta za one objekte koji učestvuju u više veza.

## II. OSNOVE OBRADE TRANSAKCIJA

*Transakcija* je vremenski uredjeni niz nedeljivih radnji nad bazom podataka koje u celini ne remete uslove integriteta. Transakcija predstavlja *logičku jedinicu* rada nad bazom podataka. Sa aspekta definisanih uslova integriteta transakcija transformiše jedno konzistentno stanje baze podataka u drugo takodje konzistentno stanje baze podataka. Kao što ćemo videti iz primera koji će uslediti između ova dva konzistentna stanja dozvoljeno je da se baza podataka nadje i u nekonzistentnom stanju. Bitno je da transakcija bude tako označena da iz bilo kojih razloga (npr. otkaz hardvera ili softvera, višekorisnički rad) ne ostavi trajno bazu podataka u nekonzistentnom stanju.

Razmotrimo primer transakcije na sledećem pojednostavljenom modelu podataka:

Šeme relacija:

PREDMET (S\_PRED, NAZIV),  
PREDAJE (S\_NAS, S\_PRED, BR\_GR),

Uslov integriteta:

PREDAJE [S\_PRED]  $\subseteq$  PREDMET [S\_PRED].

Zahtev: Promeniti šifru predmeta "INFORMACIONI SISTEMI" koja glasi 001 na šifru 006.

Upotreboom notacije jezika SQL možemo definisati sledeći niz naredbi koje treba da realizuju prethodni zahtev:

```
UPDATE PREDMET  
    SET S_PRED = 006  
    WHERE S_PRED = 001;  
UPDATE PREDAJE  
    SET S_PRED = 006  
    WHERE S_PRED = 001;
```

Smisao prethodnog primera je u tome što zahtev promene šifre predmeta "INFORMACIONI SISTEMI" sa tekuće vrednosti 001 na novu 006 od strane korisnika se posmatra kao jedinstvena operacija koja zahteva dve operacije UPDATE nad bazom podataka. Izmedju ove dve operacije UPDATE baza podataka može se naći i u nekonzistentnom stanju. Posle izvršavanja prve naredbe UPDATE uslovi integriteta ne važe. Posle izvršavanja druge naredbe UPDATE uslovi integriteta ponovo važe. Sa tačke gledišta uslova integriteta transakcija se ponaša kao jedinična radnja što ne važi za pojedinačne radnje od kojih se transakcija sastoji. Primetimo da ako se u prethodnoj transakciji redosled UPDATE operacija promeni baza podataka se takođe privremeno dovodi u nekonzistentno stanje.

Jasno je iz prethodnog primera da ne smemo dozvoliti da se od dve operacije UPDATE transakcije izvrši samo jedna, a druga ne. U tom slučaju baza podataka ostala bi trajno u nekonzistentnom stanju. Mi bi smo želeli biti sigurni da će obe operacije UPDATE biti izvršene bez obzira na mogućnosti pojave greške i to u najnepovoljnijem trenutku za transakciju. Na primer, greška u hardveru ili softveru računarskog sistema može se dogoditi izmedju dve operacije UPDATE. Računarski sistemi koji podržavaju *transakcionu obradu* moraju da garantuju da, ukoliko se transakcijom vršilo ažuriranje baze podataka i iz bilo kojih razloga transakcija se nije normalno završila, poništite sva ažuriranja delimično izvršenih transakcija. Na taj način transakcija se ili u potpunosti izvršava ili se u potpunosti poništavaju njena dejstva. Transakciona obrada odvija se pod kontrolom *administratora transakcija* kao dela programske podrške čije naredbe COMMIT i ROLLBACK u potpunosti određuju način njegovog funkcionisanja.

Naredba COMMIT signalizira administratoru transakcija o uspešnom završetku transakcije. Baza podataka ponovo se nalazi u konzistentnom stanju i sva dejstva transakcije (privremena ažuriranja) mogu biti trajno preneta na bazu podataka. Nasuprot, naredba ROLLBACK signalizira administratoru transakcija o neuspešnom završetku transakcije. U tom slučaju potrebno je sva dejstva nad bazom podataka takve transakcije poništiti. Naredbe COMMIT i ROLLBACK se zadaju direktno ili automatski od strane SUBP.

Da bi se mehanizam trajnog prenošenja ažuriranja na bazu podataka ili poništavanja ažuriranja mogao uspešno i korektno realizovati potrebno je čuvati informacije o bazi podataka pre i posle ažuriranja. Da bi se znao vremenski trenutak od kojeg počinje čuvanje informacija i kada prestaje, nad bazom podataka postavljaju se *tačke sinhronizacije*.

*Tačka sinhronizacije* predstavlja graničnu tačku izmedju dve serijske transakcije. To je tačka u kojoj se baza podataka nalazi u konzistentnom stanju.

Naredbe COMMIT i ROLLBACK kao što je bilo napomenuto ne moraju biti direktno zadate.

Naredba COMMIT jezika SQL ima sledeći opšti oblik:

COMMIT [WORK];

Naredba COMMIT signalizira uspešno izvršavanje transakcije i postavlja tačku sinhronizacije. Sva ažuriranja baze podataka izvršena datim programom od prethodne tačke sinhronizacije postaju trajna.

Neobavezni element WORK nema posebnog značaja u naredbi.

Naredba ROLLBACK jezika SQL ima sledeći opšti oblik:

ROLLBACK [WORK];

Naredba ROLLBACK signalizira neuspešno izvršavanje transakcije i postavlja tačku sinhronizacije. Sva ažuriranja baze podataka od prethodne tačke sinhronizacije, od strane datog programa se poništavaju.

Neobavezni element WORK nema posebnog značaja u naredbi.

Iz navedenih razmatranja sledi da transakcije ne mogu biti uložene jedna u drugu obzirom da naredbe COMMIT ili ROLLBACK završavaju jednu i iniciraju drugu transakciju. Kao posledica sledi da se jedan program sastoji od jedne ili više serijskih transakcija.

Administrator transakcija mora garantovati da će posle uspešnog izvršavanja naredbe COMMIT od strane transakcije sva ažuriranja nad bazom podataka postati trajna, mada je moguće da se u sledećem trenutku dogodi otkaz računarskog sistema. Na primer, otkaz sistema nastaje u trenutku posle izdavanja naredbe COMMIT ali pre nego što ažuriranja budu fizički zapisana u bazu podataka.

Ažuriranja koja se u trenutku otkaza nalaze u baferima operativne memorije na taj način mogla bi biti izgubljena. Ako se to dogodi procedura restarta računarskog sistema treba na osnovu žurnala (podataka o izvršenim promenama na bazi podataka) ipak da izvrši trajno prenošenje ažuriranja na bazu podataka. Sledi da podaci o žurnaluu trebaju biti trajno memorisani pre izdavanja naredbe COMMIT.

### II.1. Paralelno izvršavanje transakcija

Sistem za upravljanje bazom podataka treba da omogući paralelni rad većem broju korisnika nad istom bazom podataka što znači paralelno izvršavanje više transakcija. Da bi to bilo moguće potrebno je obezbediti adekvatno upravljanje paralelnim izvršavanjem transakcija kako transakcije ne bi nepoželjno mešale svoja dejstva jedna na drugu.

U osnovi postoje tri slučaja kada je transakcija samostalno korektna, ali zbog mešanja transakcija može nastati pogrešan rezultat ukoliko ne postoji odgovarajući mehanizam upravljanja. Radi se o situacijama u kojima dolazi do mešanja operacija iz dve korektne transakcije pri čemu je rezultat nekorekstan.

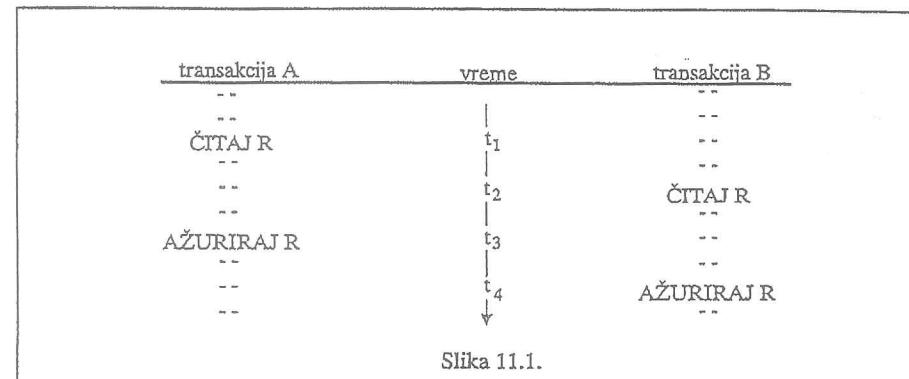
Na prethodno ukazane probleme odnose se:

1. Izgubljena ažuriranja.
2. Zavisnosti od privremenih ažuriranja.
3. Narušavanje serijabilnosti.

#### Problemi izgubljenog ažuriranja

Razmotrimo situaciju priказанu na slici 11.1. koju možemo interpretirati na sledeći način:

Transakcija A čita zapis R u trenutku  $t_1$ . Transakcija B čita isti zapis R u trenutku  $t_2$ . Transakcija A ažurira zapis R u trenutku  $t_3$  na osnovu vrednosti pročitanih u trenutku  $t_1$ . Transakcija B ažurira zapis R u trenutku  $t_4$  na osnovu vrednosti pročitanih u trenutku  $t_2$  koje su iste sa vrednostima pročitanim u trenutku  $t_1$  od strane transakcije A. Ažuriranje od strane transakcije A biva poništeno u trenutku  $t_4$  jer transakcija B ne uzima u obzir ažuriranja od strane transakcije A.

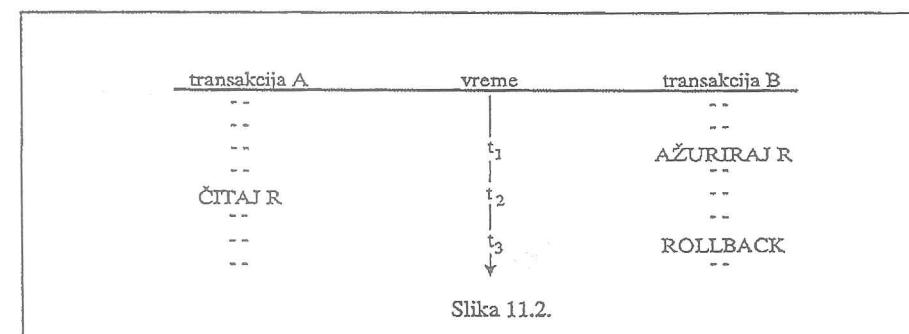


Slika 11.1.

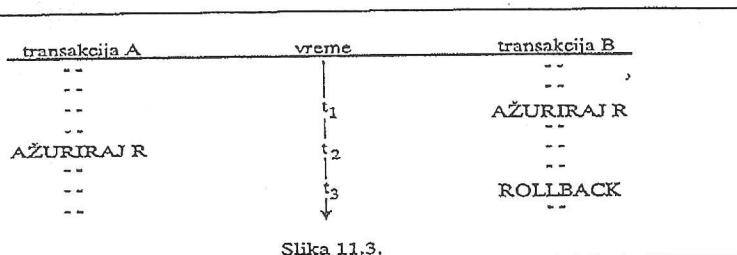
#### Problem zavisnosti od privremenih ažuriranja

Problem povezan sa zavisnošću od privremenih ažuriranja nastaje u slučaju kada se transakciji A dozvoljava čitanje ili što je još lošije ažuriranje zapisa koji je privremeno ažuriran od transakcije B pri čemu taj zapis još nije trajno ažuriran od transakcije B. S obzirom na činjenicu da zapis nije trajno ažuriran od transakcije B, postoji mogućnost da transakcija B poništa svoja privremena ažuriranja. Nastaje situacija u kojoj je transakcija A izvršila neke operacije na bazi podataka koji više ne postoje odnosno kao da nikada nisu ni postojali u bazi podataka.

Opisanu situaciju ilustruju slike 11.2 i 11.3. Transakcija A sa slike 11.2 "vidi" u trenutku  $t_2$  privremeno ažurirane podatke zapisa R. Transakcija B u trenutku  $t_3$  poništava svoja ažuriranja zapisa R. Očigledno je da je transakcija A izvršena na osnovu podataka u trenutku  $t_2$  koji mogu biti različiti od onih u trenutku  $t_1$ , što znači da i rezultat transakcije A može biti pogrešan.



Slika 11.2.



Slika 11.3.

Napomenimo da poništavanje transakcije B ne mora biti uzrokovano nekom greškom u B. Uzrok poništavanja transakcije B može biti na primer, otkaz sistema u trenutku kada se transakcija A u potpunosti završila, a transakcija B nije. Kao posledica otkaza sistema izdaće se naredba za poništavanje samo onih transakcija koje su bile u toku u trenutku otkaza odnosno poništavanje samo transakcije B.

Situacija sa slike 11.3. je još lošija. Transakcija A ne samo da je zavisna od privremeno ažuriranih podataka transakcije B već se njena ažuriranja zapisa R u trenutku t<sub>2</sub> poništavaju od strane transakcije B u trenutku t<sub>3</sub> koja zapisu R vraća stanje pre trenutka t<sub>1</sub>.

#### Problem narušavanja serijabilnosti

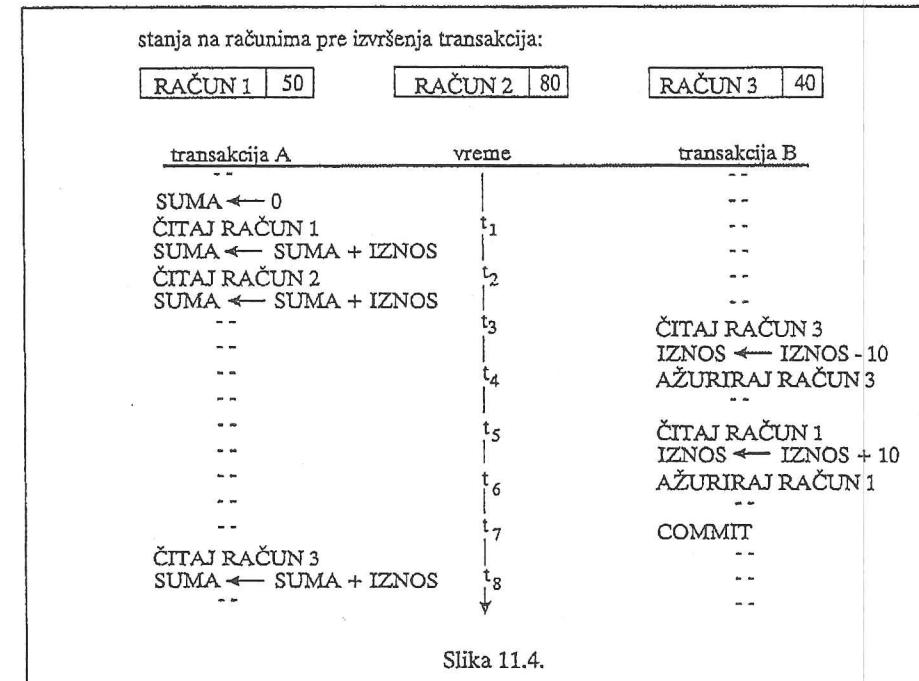
Razmotrimo situaciju na slici 11.4. na kojoj su prikazane dve transakcije A i B koje se izvršavaju nad zapisima tekućih računa gradjana. Transakcija A sumira stanja na računima. Transakcija B prenosi iznos 10 sa računa 3 na račun 1. Očigledno da je dobijeni rezultat 160 od transakcije A nekorektan. Ako bi transakcija A takav rezultat upisala u bazu podataka ista bi trajno ostala u protivrečnom stanju.

Uočimo razliku datog primera od problema zavisnosti transakcije A od privremenih ažuriranja iz ranijeg primera. Ovde transakcija B svoja ažuriranja trajno prenosi na bazu podataka pre nego što transakcija A čita zapis računa 3.

## 11.2. Rešavanje problema paralelnog izvršavanja transakcija

Uobičajeni način rešavanja problema paralelnog izvršavanja transakcija su *protokoli zaključavanja*. Osnovna ideja zaključavanja sastoji se u tome da

transakcija koja zahteva garancije da neki objekat (obično zapis baze podataka) za koji je ona zainteresovana ne može biti ažuriran u nekom vremenu. Takav objekat transakcija zaključava i sprečava njegovo ažuriranje od strane drugih transakcija. Na taj način transakcija raspolaže *stabilnim* podacima o objektu koliko ona to zahteva.



Slika 11.4.

Pre detaljnijih razmatranja principa delovanja zaključavanja radi pojednostavljenja polazimo od sledećih prepostavki:

1. Kao objekt podvrgnut protokolu zaključavanja može se pojaviti zapis baze podataka odnosno n-torka bazne tabele.
2. Biće razmatrana dva oblika protokola zaključavanja:
  - ekskluzivna zaključavanja (E) - omogućava transakciji koja je n-torku zaključala, čitanje i promenu vrednosti iste,
  - deljiva zaključavanja (D) - omogućava transakciji koja je n-torku zaključala, samo njeno čitanje.
3. Posmatraju se samo operacije na nivou zapisa (ČITAJ, AŽURIRAJ itd.). Operacije zaključavanja na nivou skupova mogu se posmatrati kao niz operacija na nivou zapisa.

Razmotrimo sada detaljnije protokol zaključavanja.

1. Ako transakcija A izvrši ekskluzivno zaključavanje (tip E) zapisa R, tada će zahtev za zaključavanje, bilo koje vrste, iz transakcije B preći u stanje čekanja. Transakcija B biće u stanju čekanja sve dok zapis ne bude otključan (deblokiran) od strane transakcije A.
2. Ako transakcija A izvrši deljivo zaključavanja (tip D) zapisa R tada:
  - a) Zahtev od transakcije B na zaključavanje tipa E zapisa R preći će u stanje čekanja i ostati u tom stanju sve dok transakcija A ne izvrši otključavanje zapisa R;
  - b) Zahtev od transakcije B na zaključavanje tipa D zapisa R biće izvršeno tj. i transakcija A i B mogu istovremeno imati zaključan zapis R.

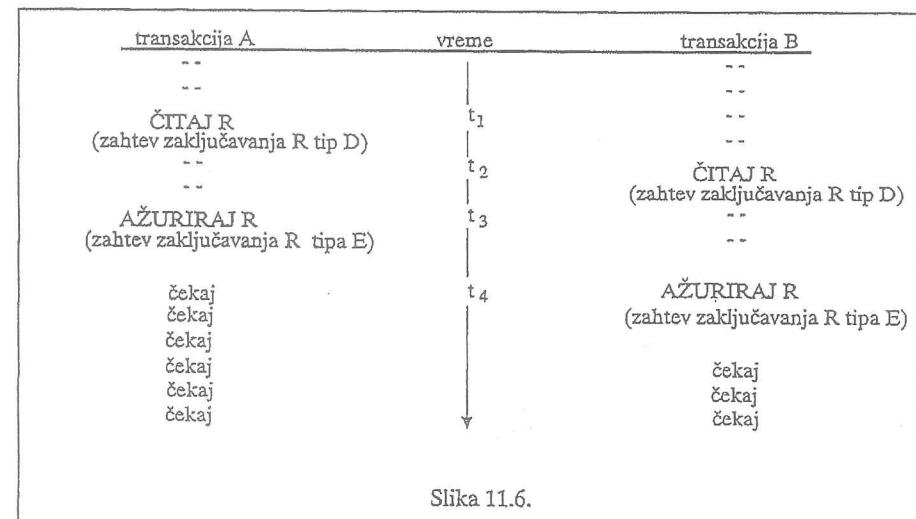
Prethodni zahtevi zaključavanja zapisa R od strane transakcija A i B mogu se rezimirati matricom uskladjenosti tipova zaključavanja koja je data na slici 11.5. i koju možemo interpretirati na sledeći način. Posmatrajmo neki zapis R. Pretpostavimo da u sadašnjem trenutku transakcija A ima zaključavanje zapisa R označeno u zagлавju tabele (crtica znači nezaključan zapis). Neka transakcija B zahteva zaključavanje zapisa R prema tipu označenom u prvom stupcu tabele. Oznaka N na preseku vrste i kolone matrice ukazuje na nesaglasnost zahteva transakcije A i B odnosno zahtev transakcije B neće biti izvršen i B prelazi u stanje čekanja. Oznaka Y ukazuje na saglasnost zahteva transakcije A i B što znači da će zahtev transakcije B biti izvršen.

transakcija A			
transakcija B	E	D	-
E	N	N	Y
D	N	Y	Y
-	Y	Y	Y

Slika 11.5.

3. Zahtevi transakcije za zaključavanje zapisa obično nisu direktni već se iniciraju automatski sa zahtevanjem izvršavanja određenih operacija: Da bi transakcija uspešno izvršila operaciju čitanja zapisa (ČITAJ R) ona prethodno zahteva zaključavanje tipa D na zapisu R. Ako je zaključavanje uspešno tada je i čitanje uspešno. Ako se zaključavanje ne može izvršiti tada ni čitanje nije moguće, a transakcija prelazi u stanje čekanja, odnosno u stanje pokušavanja da se zaključavanje-čitanje uspešno izvrši. Kada transakcija zahteva izvršavanje operacije ažuriranja (AŽURIRAJ R) ona prvo zahteva zaključavanje zapisa tipa E. Slično kao i kod operacije čitanja ako je zaključavanje uspešno tada je i ažuriranje uspešno. Ako se zaključavanje ne može izvršiti tada ni ažuriranje nije moguće, a transakcija prelazi u stanje čekanja, odnosno u stanje pokušavanja da se zaključavanje-ažuriranje uspešno izvrši. Ako je transakcija već izvršila zaključavanje zapisa (kao što to i proizilazi u slučaju sekvence operacija ČITAJ ... AŽURIRAJ) tada će operacija AŽURIRAJ pojačati zaključavanje sa tipa D na tip E.
4. Zaključavanja tipa E zadržavaju se do sledeće tačke sinhronizacije dok se zaključavanja tipa D takođe obično zadržavaju do sledeće tačke sinhronizacije, ali mogu biti ukinuta i ranije.

Uzimajući u obzir zaključavanje zapisa sada možemo ponovo razmotriti načine rešavanja problema paralelnog izvršavanja transakcija.



Slika 11.6.

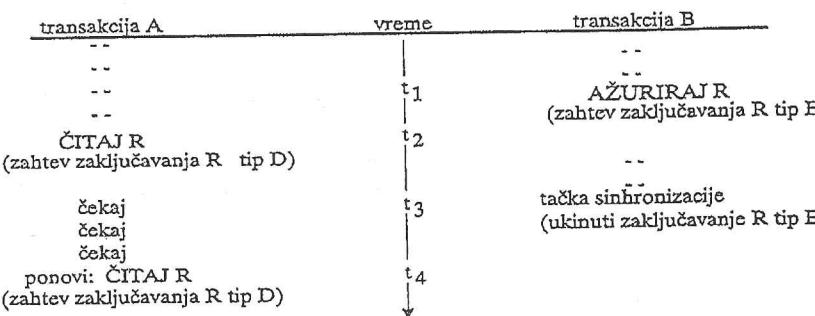
### Problemi izgubljenog ažuriranja

Slika 11.6. predstavlja modifikovanu varijantu sa slike 11.1 uzimajući u obzir paralelno izvršavanje transakcija A i B sa mehanizmom zaključavanja zapisa.

Uočavamo da zahtev transakcije A u trenutku  $t_3$  za ažuriranje zapisa R prelazi u stanje čekanja zbog toga što se ne može izvršiti zaključavanje tipa E na zapisu R koji je već zaključan od transakcije B prema tipu D. Zbog toga transakcija A prelazi u stanje čekanja. Analogno u trenutku  $t_4$  transakcija B prelazi u stanje čekanja. Nijedna od transakcija ne može biti nastavljena i ne može doći do gubljenja ažuriranja koja smo ranije imali. Očigledno da smo rešavanjem jednog problema stvorili drugi problem koji nazivamo **medjusobno blokiranje transakcija**. Rešavanje problema medjusobnog blokiranja transakcija biće razmatrano kasnije.

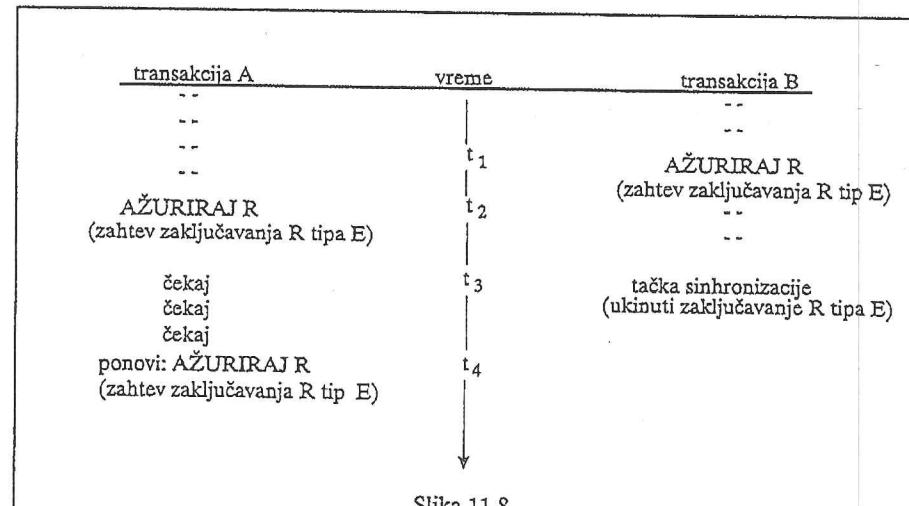
### Problem zavisnosti od privremenih ažuriranja

Uzimajući u obzir mehanizam zaključavanja pri paralelnom izvršavanju transakcija situacija sa slike 11.2. i 11.3. menja se u situaciju ilustrovanu na slikama 11.7. i 11.8. Operacije ČITAJ R odnosno AŽURIRAJ R od strane transakcije A u trenutku  $t_2$  neće biti izvršene zato što su njihovi zahtevi na zaključavanje u konfliktu sa zaključavanjima koja su već postavljena od strane transakcije B.



Slika 11.7.

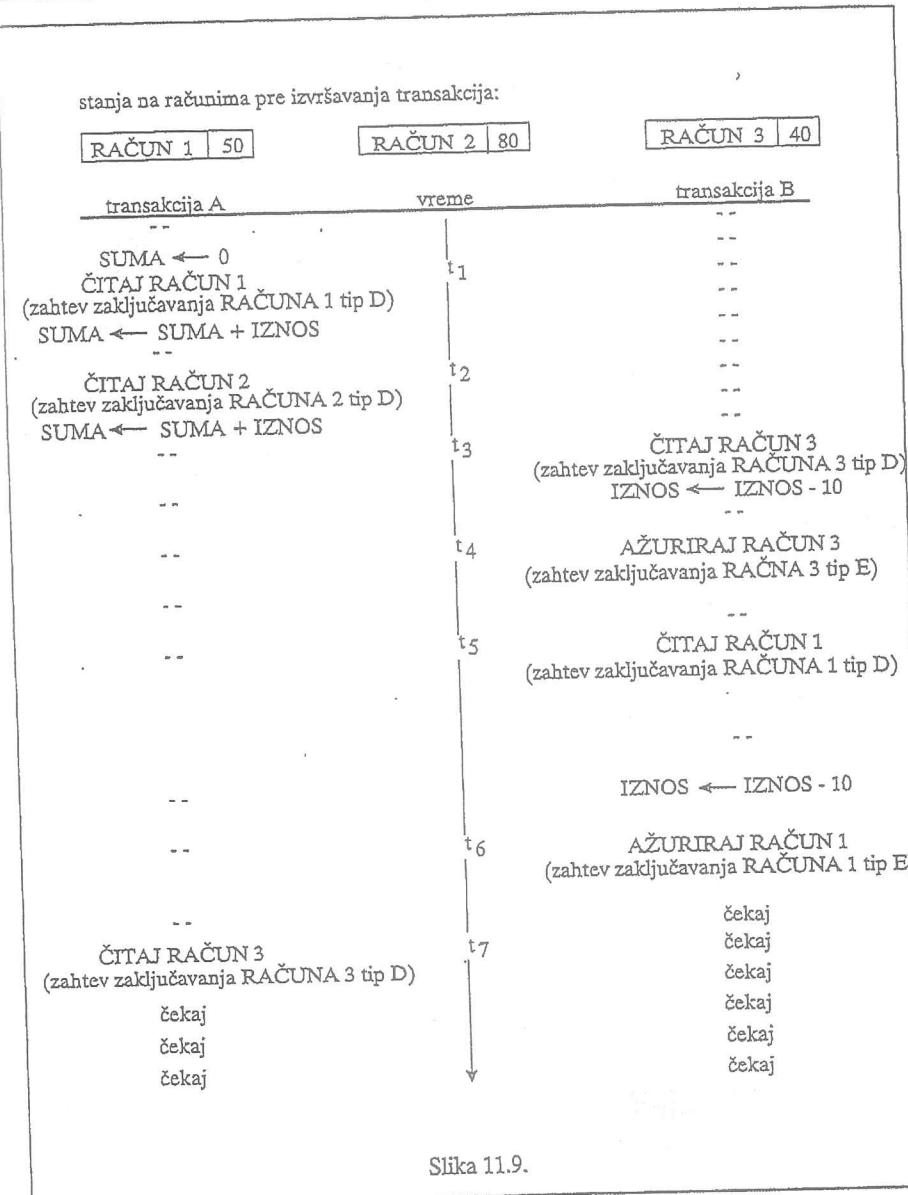
Zbog toga transakcija A prelazi u stanje čekanja sve dok transakcija B ne dostigne tačku sinhronizacije izvršavajući operaciju COMMIT ili ROLLBACK (trenutak  $t_3$ ). U trenutku  $t_3$  transakcija B deblokira zapis R i transakcija A može nastaviti izvršavanje. Transakcija A u trenutku  $t_3$  raspolaže zapisom R pre izvršavanja transakcije B ako je B završila sa ROLLBACK ili zapisom R koji je rezultat uspešnog završavanja transakcije B. U oba slučaja transakcija A više ne zavisi od privremenih ažuriranja druge transakcije.



Slika 11.8.

### Problem narušavanja serijabilnosti

Na slici 11.9. data je modifikovana varijanta slike 11.4. uzimanjem u razmatranje mehanizma zaključavanja pri paralelnom izvršavanju transakcija. Možemo uočiti da zahtev transakcije B u trenutku  $t_6$  za operaciju AŽURIRAJ neće biti izvršen, jer se ne može izvršiti zaključavanje tipa E zapisa RAČUN 1 jer je taj zapis zaključan po tipu D od strane transakcije A. Od trenutka  $t_6$  transakcija B prelazi u stanje čekanja. U trenutku  $t_7$  ne izvršava se zahtev ČITAJ RAČUN 3 transakcije A jer se ne može izvršiti zaključavanje tog zapisa po tipu D zbog toga što je RAČUN 3 zaključan od transakcije B po tipu E. Transakcija A u trenutku  $t_7$  prelazi u stanje čekanja. Rešavajući problem narušavanja serijabilnosti došli smo do medjusobnog blokiranja transakcija.



Slika 11.9

### *11.3. Direktno zaključavanje*

Prethodno su razmatrane mogućnosti načina automatskog zaključavanja zapisa odnosno n-torki prilikom izvršavanja određenih operacija. SUBP poseđuju i načine direktnog zaključavanja podataka mada ih u većini situacija nije neophodno koristiti. S obzirom da načini direktnog zaključavanja zavise od konkretnog SUBP ovde ćemo dalje razmotriti samo direktno zaključavanje korišćenjem naredbe **LOCK TABLE** jezika SQL.

Opšti oblik naredbe SQL za zaključavanje svih n-torki tabele glasi:

**LOCK TABLE naziv\_tabele [, naziv\_tabele]...**  
**IN {SHARE | SHARE UPDATE | EXCLUSIVE } MODE [NOWAIT].**

gde je naziv tabele ime bazne tabele i ne može biti ime pogleda.

Tabelu zaključavamo na jedan od sledećih načina:

SHARE - kada tabelu koristimo za pretraživanje i kada drugim korisnicima (transakcijama) dozvoljavamo korišćenje iste tabele za pretraživanje, ali ne i za ažuriranje.

SHARE UPDATE - kada možemo ažurirati tabelu i kada drugi korisnik može zaključati n-torku korišćenjem naredbe SELECT ... FOR UPDATE. Dozvoljeno je ažuriranje n-torke koja je zaključana od drugog korisnika pod uslovom da još nije ažurirana od drugog korisnika.

**EXCLUSIVE** - kada ažuriramo više n-torki tabele i sprečavamo druge korisnike da zaključaju ili ažuriraju bilo koju n-torku tabele:

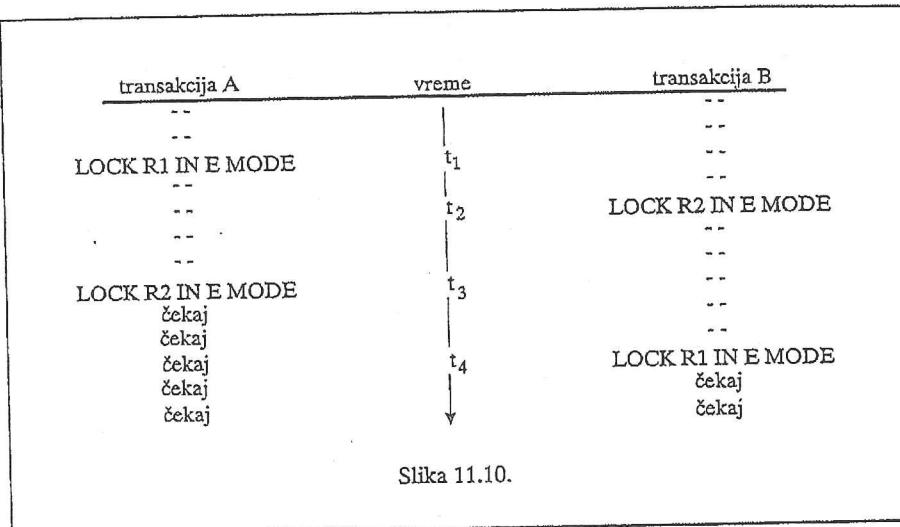
Ukoliko zahtev **LOCK TABLE** ne može biti izvršen zato što je na primer, drugi korisnik zaključao tabelu, **LOCK TABLE** prelazi u stanje čekanja do trenutka kada će moći biti izvršen. Ukoliko se navede opcija **NOWAIT** i zahtev **LOCK TABLE** ne može biti izvršen kontrola se vraća programu tako da dok čekate možete zahtevati izvršavanje nekih drugih naredbi.

Kada je jednom zaključana tabela ostaje zaključana sve do tačke sinhronizacije (COMMIT ili ROLLBACK).

## 11.4. Medjusobno blokiranje transakcija

Prethodno smo razmatrali kako se medjusobno blokiranje transakcija može koristiti za rešavanje nekih problema paralelnog izvršavanja transakcija i izbegavanja nepoželjnog mešanja podataka. Međutim videli smo na žalost da medjusobno blokiranje transakcija dovodi do blokiranog stanja.

I pored toga što smo ranije ilustrovali medjusobno blokiranje transakcija, razmotrimo stanje na slici 11.10. koje ilustruje opštiji slučaj razmatranog problema. Primetimo da operacije zaključavanja mogu biti bilo koje operacije koje iniciraju zaključavanja.



Blokirano stanje je takva situacija u kojoj se dve ili više transakcija istovremeno nalaze u stanju čekanja i svaka od njih čeka da ona duga transakcija skine blokiranje kako bi se omogućio njen dalji nastavak. Na slici 11.10. ilustrovano je medjusobno blokiranje dve transakcije A i B. Moguće je takođe medjusobno blokiranje tri, četiri i više transakcija. U praksi blokirana stanja skoro nikada nisu uzrokovana od strane više od dve transakcije.

Kada nastane blokirano stanje potrebno je isto prepoznati i ukloniti. Da bi se rešilo blokirano stanje jedna od transakcija koja učestvuje u blokiranju odabira se kao *žrtva* koja će biti našilno prekinuta uz oslobođanje svih

zaključavanja koje je ona postavila i poništavanjem svih efekata na bazi podataka koje je ona izvršila. Na taj način dozvoljava se drugoj blokiranoj transakciji nastavak izvršavanja dok se poništena transakcija aktivira od početka.

Ukažimo bez detaljnih razmatranja da je jedan od načina identifikovanja blokiranih stanja analiza koda koji pokazuje ishod izvršavanja date SQL naredbe, a koji se dobija posle izvršavanja svake naredbe.

Za smanjenje verovatnoće pojave blokiranih stanja kao preporuka može se dati da sve transakcije zaključavaju po mogućnosti n-torce u istom redosledu. Primljeno na situaciju ilustrovano na slici 11.10. to bi zahtevalo na primer, izmenu redosleda zaključavanja u transakciji B tako da se prvo zahteva zaključavanje R1, a posle toga zaključavanje R2.

## 11.5. Restauracija konzistentnog stanja baze podataka

Prethodno su razmatrani pojednostavljeni modeli transakcija bez nekih praktičnih aspekata koje ćemo sada nešto detaljnije analizirati. Realne transakcije se mogu opisati na sledeći način:

Transakcija je niz pojedinačnih radnji sa osobinama:

### a) Pojedinačne radnje

- (i) Radnja se ili uspešno obavi i tada ostavlja odgovarajući efekat na bazu podataka ili nema nikakvog efekta na bazu podataka.
- (ii) Radnje su nedeljive, što znači ako se dve radnje obavljaju nad istim objektom, ukupan efekat odgovara situaciji u kojoj se prvo jedna radnja izvrši u celosti, pa zatim druga.

### b) Kompozicija radnji

- (i) Radnje jedne transakcije se izvršavaju u redosledu koji je određen navodnjem tih radnji u transakciji.
- (ii) Transakcija se ili u celini uspešno izvrši i tada ostavlja na bazu podataka efekat sekvenčnog izvršenja svojih radnji ili ne ostavlja nikakav efekat na bazu podataka.

- (iii) Ako se transakcija uspešno završi, njeni efekti na bazu podataka su trajni. Jedini način da se ponište efekti uspešno završene transakcije je nova uspešna transakcija čiji efekti poništavaju efekte prve transakcije.
- (iv) Pre završetka transakcija može samu sebe prekinuti ili može biti prekinuta od SUBP. U oba slučaja transakcija ne ostavlja nikakvog efekta na bazu podataka.
- (v) U okviru transakcije je moguće definisati tačku sinhronizacije tj. pamćenja stanja izvršenja transakcije. Pre uspešnog završetka transakcije moguće je zahtevati poništavanje svih njenih efekata na bazu podataka do neke prethodno zapamćene tačke sinhronizacije.

Sledi nekoliko modela obrade transakcija koji imaju cilj da zadovolje prethodno navedene uslove:

#### *Privatni memorijski prostor*

Efekat pojedinih akcija od kojih je transakcija T sastavljena može se opisati na sledeći način:

##### 1. Počni transakciju T.

Rezerviše se privatni memorijski prostor transakcije T čija je funkcija da prihvata vrednosti koje T čita iz baze podataka ili ih upisuje u bazu podataka.

##### 2. Pročitaj vrednost objekta X.

Ispituje se da li privatni memorijski prostor transakcije T sadrži objekat X. Ako sadrži, tada se vrednost tog objekta X stavlja na raspolaganje transakciji T. U protivnom se iz baze podataka pročita vrednost objekta X. Ta se vrednost stavlja na raspolaganje transakciji T i istovremeno upisuje u privatni memorijski prostor transakcije T.

##### 3. Upiši novu vrednost objekta X.

Pretražuje se privatni memorijski prostor transakcije T kao i kod čitanja. Ako se u njemu nalazi objekat X, ažurira se vrednost objekta X. U suprotnom se u privatnom memorijskom prostoru transakcije T kreira primerak objekta X sa datom vrednošću. Možemo uočiti da efekat ove radnje nije menjanje vrednosti objekta X u bazi podataka.

#### 4. Kraj transakcije T.

Vrednosti svih objekata iz privatnog memorijskog prostora transakcije T se upisuju u bazu podataka. Posebnim *protokolom nedeljivog kompletiranja transakcije* se garantuje da se ili sva ažuriranja iz privatnog memorijskog prostora prenesu u bazu podataka ili nijedno. Oslobodi se privatni memorijski prostor transakcije T.

*Protokol nedeljivog kompletiranja transakcije* obezbeđuje da se obrade sva ažuriranja jedne transakcije ili nijedno. Sastoјi se iz sledeće dve faze:

*Prva faza* - za svaku radnju ažuriranja objekta X obavlja se radnja prethodnog upisivanja nove vrednosti objekta X u memoriju za koju se pretpostavlja da je pouzdana.

*Druga faza* - za svaku radnju ažuriranja objekta X obavlja se upisivanje nove vrednosti objekta X u bazu podataka.

Definisani protokol ima sledeće osobine:

- Ako dodje do pada sistema u toku prve faze, to neće izazvati nikakve probleme, jer se u toku te faze nikakve promene ne vrše na bazu podataka.
- Ako dodje do pada sistema u toku druge faze, baza podataka može biti u nekonistentnom stanju, jer su u njoj parcijalno zapisana ažuriranja jedne transakcije. U postupku uspostavljanja konzistentnog stanja baze podataka ovaj se problem rešava na osnovu podataka o izvršenim ažuriranjima koji su upisani u pouzdanu memoriju.

Modelom privatnog memorijskog prostora realizuju se sve osobine transakcije osim osobine (v). Poništavanje svih efekata transakcije (iv) postiže se oslobadjanjem privatnog memorijskog prostora.

#### *Dnevnik ažuriranja*

Po modelu privatnog memorijskog prostora zahteva se upisivanje nove vrednosti objekta u sigurnu memoriju koju možemo nazvati i *log transakcije*. Ako se u log transakcije, umesto nove vrednosti, upisuje stara vrednost objekta koji se ažurira, tada privatni memorijski prostor transakcije nije

neophodan. Svaka radnja ažuriranja vrednosti objekta prvo upisuje prethodnu vrednost objekta (stara vrednost) u log transakcije, a zatim direktno u bazu podataka upisuje novu vrednost. Suprotno modelu privatnog memorijskog prostora, ne odlaze se ažuriranje baze podataka do naredbe za kompletiranje transakcije. Poništavanje efekata neke transakcije pre njenog uspešnog kompletiranja postiže se čitanjem unatrag loga transakcije i restauriranjem prethodnih vrednosti objekta u bazi podataka. Za poništavanje efekata transakcije pored prethodnih vrednosti objekta za svaku radnju iz skupa radnji pomoću kojih se transakcija realizuje treba definisati i njoj odgovarajuću radnju poništavanja. Ovom metodom može se realizovati i osobina b.(v) tako što se svaka tačka sinhronizacije unosi u log transakcije čime se obezbeđuje mogućnost poništavanja ažuriranja transakcije do nekog prethodno zapamćenog stanja izvršenja.

Uspešan završetak transakcije se registruje upisivanjem tačke sinhronizacije u log transakcije. Ako dodje do pada sistema pre tog trenutka, svi efekti transakcije će biti automatski poništeni na osnovu prethodnih vrednosti i drugih relevantnih podataka iz loga transakcije prilikom ponovnog starta rada sa bazom podataka. Logovi pojedinačnih transakcija se kombinuju u jedinstven sistemski log koji nazivamo *dnevnik ažuriranja*, u kojem su pokazivačima povezani zapisi koji predstavljaju log jedne transakcije.

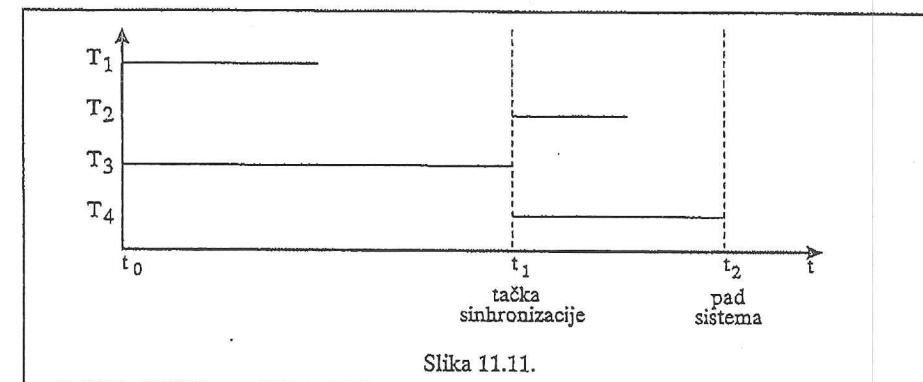
Protokol restauracije konzistentnog stanja baze podataka na osnovu dnevnika ažuriranja zavisi od načina postavljanja tačke sinhronizacije.

Najjednostavniji, ali istovremeno i najmanje prihvatljiv za korisnike, način postavljanja tačke sinhronizacije je odlaganje svih novih transakcija dok se transakcije koje su u toku ne završe. U trenutku kada više ni jedna transakcija nije u toku, u dnevnik ažuriranja se upisuje zapis tačke sinhronizacije i mogu da započnu nove transakcije. Restauriranje konzistentnog stanja baze podataka obavlja se prema sledećem protokolu:

Neka je data tačka sinhronizacije koja odgovara vremenskom trenutku  $t_1$  i dnevnik svih promena do trenutka  $t_2$ . Ako dodje do pada sistema u trenutku  $t_2$ , poništavaju se sve promene transakcija koje su počele posle poslednje tačke sinhronizacije tj.  $t_1$ . Zatim se ponavljaju radnje svih transakcija koje su počele posle poslednje tačke sinhronizacije, a uspešno su završene pre pada sistema.

Na slici 11.11. ilustrovano je jedno stanje odvijanja transakcija. Trenutak  $t_0$  predstavlja tačku sinhronizacije i tada su započele transakcije  $T_1$  i  $T_3$ . Dok se

ove dve transakcije ne završe ni jedna druga transakcija ne može započeti. U trenutku  $t_1$  transakcija  $T_1$  i  $T_3$  su završene, u dnevnik ažuriranja se upisuje tačka sinhronizacije i aktiviraju se transakcije  $T_2$  i  $T_4$ . U trenutku  $t_2$  nastaje pad sistema. Konzistentno stanje uspostavlja se poništavanjem svih efekata transakcija  $T_2$  i  $T_4$  i ponavljanjem svih radnji transakcije  $T_2$ .



Slika 11.11.

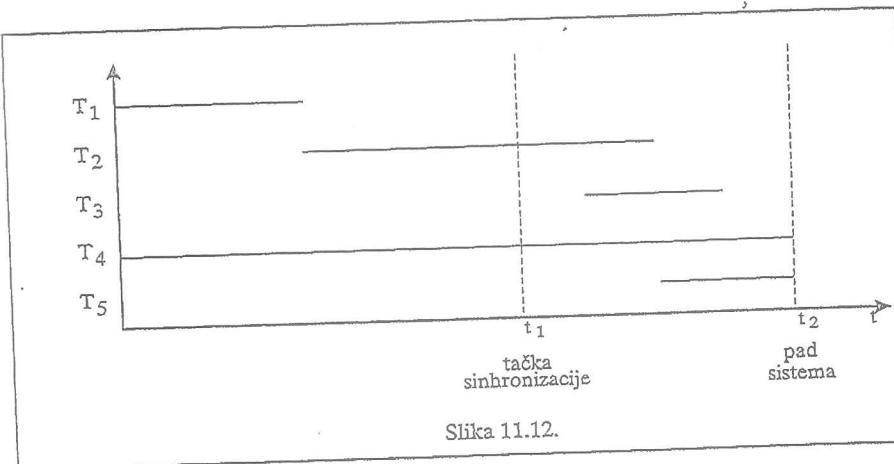
Nedostatak opisanog načina postavljanja tačaka sinhronizacije i uspostavljanja konzistentnog stanja su dugi periodi čekanja na završetak transakcija koje su u toku da bi startovali nove transakcije.

Raspoloživost sistema se može znatno poboljšati ako se tačka sinhronizacije može upisati u dnevnik ažuriranja u trenutku kada se ne obavlja nijedna radnja transakcija koje su u toku. U tački sinhronizacije uspostavlja se korespondencija između stanja baze podataka i dnevnika ažuriranja, tj. sve promene upisane u dnevnik ažuriranja od prethodne tačke sinhronizacije unose se u bazu podataka. Zatim se u dnevnik ažuriranja upisuje zapis tačke sinhronizacije koji sadrži spisak svih transakcija koje su u toku i za svaku od njih pokazivač na njen poslednji zapis u dnevniku ažuriranja.

Uspostavljanje konzistentnog stanja baze podataka nakon pada sistema se obavlja po sledećem protokolu:

Neka je data tačka sinhronizacije koja odgovara vremenskom trenutku  $t_1$  i dnevnik promena do trenutka  $t_2$ . Ako dodje do pada sistema u trenutku  $t_2$ , poništavaju se sve promene transakcija koje su imale neko dejstvo na bazu podataka od trenutka  $t_1$  do trenutka  $t_2$ . Zatim se ponavljaju sve radnje od trenutka  $t_1$ , onih transakcija koje su se uspešno završile u periodu od trenutka  $t_1$  do trenutka  $t_2$ .

Na slici 11.12. ilustrovano je jedno stanje odvijanja transakcija.



Efekti transakcija  $T_2, T_3, T_4$  i  $T_5$  koje su bile u toku u trenutku pada sistema u trenutku  $t_2$ , pri restauraciji konzistentnog stanja će biti poništeni. Sve radnje transakcije  $T_2$  će biti ponovljene od tačke sinhronizacije  $t_1$ . Transakcija  $T_3$  će biti ponovljena u celini. Poništavaju se sva ažuriranja koja je obavila transakcija  $T_4$ .

Dalje poboljšanje poslednjeg modela uspostavljanja konzistentnog stanja može se ostvariti uvodjenjem dve verzije baze podataka koje se mogu koristiti istovremeno. Jedna verzija se zove *tekuća*, a druga *sigurnosna*. Svojim radnjama transakcije uvek utiču samo na tekuću verziju i ne menjaju sigurnosnu verziju. Postoje sledeće dve karakteristične operacije sa tekućom i sigurnosnom verzijom baze podataka.

(1) Tekuća verzija postaje sigurnosna. Time se ažuriranja baze podataka koja su obavljena od poslednjeg izvršenja ove operacije i faktički upisana u bazu podataka.

(2) Sigurnosna verzija postaje tekuća. Time se poništavaju sva ažuriranja baze podataka koja su se odigrala od trenutka poslednjeg izvršenja operacije (1).

Operacija (1) se obavlja u vreme zapisivanja tačke sinhronizacije, a operacija (2) u postupku restauracije konzistentnog stanja baze podataka. Nakon operacije (2) se koristeći dnevnik ažuriranja, iz baze uklone efekti transakcija

koje se nisu uspešno završile, a restauriraju efekti uspešno završenih transakcija.

Uspostavljanje konzistentnog stanja baze podataka nakon pada sistema se obavlja po sledećem protokolu:

Neka je data tačka sinhronizacije koja odgovara vremenskom trenutku  $t_1$ . Pretpostavljamo da je pre njenog registrovanja izvršena operacija zamene sigurnosne verzije baze podataka njenom trenutnom verzijom. Neka je dat dnevnik svih ažuriranja do trenutka  $t_2$  kada nastaje pad sistema.

Obavlja se operacija zamene tekuće verzije baze podataka sigurnosnom verzijom tj. tekućom verzijom baze podataka postaje sigurnosna verzija.

Poništavaju se sve promene nastale do trenutka  $t_1$ , onih transakcija koje nisu uspešno završene do trenutka  $t_2$ .

Ponavljaju se sve radnje izvršene od trenutka  $t_1$  do trenutka  $t_2$  transakcija koje su se uspešno završile do trenutka  $t_2$ .

Na slici 11.12. ilustrovano je jedno stanje odvijanja transakcija koje smo već razmatrali. Prema modelu tekuće i sigurnosne verzije baze podataka, konzistentno stanje na datom primeru uspostavlja se na sledeći način:

Operacijom zamene tekuće verzije sigurnosnom, sva ažuriranja transakcija  $T_2, T_3, T_4$  i  $T_5$  posle tačke sinhronizacije  $t_1$  poništena su. Zatim se poništavaju sva ažuriranja koja je obavila transakcija  $T_4$ . Ponavljaju se sve radnje transakcija  $T_2$  i  $T_3$  koje su uspešno završene pre pada sistema.