

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

Dragan Mihajlović

INFORMACIONI SISTEMI I
PROJEKTOVANJE BAZA PODATAKA

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

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

9.1. OBRADA UPITA	219	II.1. PARALELNO IZVRŠAVANJE TRANSAKCIJA	302
9.2. STANDARDNI UPITNI JEZIK - SQL	221	II.2. REŠAVANJE PROBLEMA PARALELNOG IZVRŠAVANJA	
9.2.1. NAREDBE ZA DEFINISANJE PODATAKA	225	TRANSAKCIJA	304
9.2.1.1. KREIRANJE NOVE TABELE - CREATE TABLE	225	II.3. DIREKTNO ZAKLJUČAVANJE	311
9.2.1.2. KREIRANJE POGLEDA - CREATE VIEW	227	II.4. MEDJUSOBNO BLOKIRANJE TRANSAKCIJA	312
9.2.1.3. PROMENA STRUKTURE TABELE NAKON ŠTO JE KREIRANA - ALTER TABLE	229	II.5. RESTAURACIJA KONZISTENTNOG STANJA BAZE	
9.2.1.4. BRISANJE TABELE IZ BAZE PODATAKA - DROP TABLE	230	PODATAKA	313
9.2.1.5. INDEKSI	231		
9.2.2. NAREDBE ZA MANIPULISANJE PODACIMA	232		
9.2.2.1. DODAVANJE NOVIH N-TORKI U TABELU - INSERT	232		
9.2.2.2. PRETRAŽIVANJE RELACIONE BAZE PODATAKA - SELECT	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 - UPDATE	263		
9.2.2.4. BRISANJE N-TORKI TABELE - DELETE	266		
9.2.3. REČNIK PODATAKA	267		
9.3. JEZIK UPITA NA OSNOVU PRIMERA - QBE	269		
9.4. TRANSFORMACIJA UPITA	272		
10. PREVODJENJE MODELA OBJEKTI-VEZE OBELEŽJA U RELACIONI MODEL	277		
10.1 POSTUPAK PREVODJENJA	278		
11. OSNOVE OBRADE TRANSAKCIJA	299		

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.

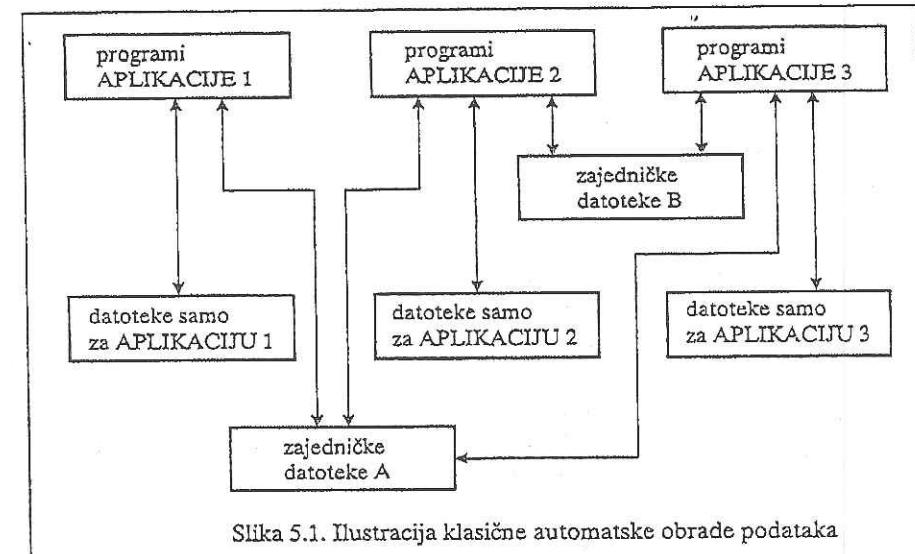
Redundansa (suvišnost, preopširnost, ponavljanje) podataka u jednom informacionom sistemu bila je vrlo velika zbog postojanja više datoteka sa istim podacima.

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 razlike 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.

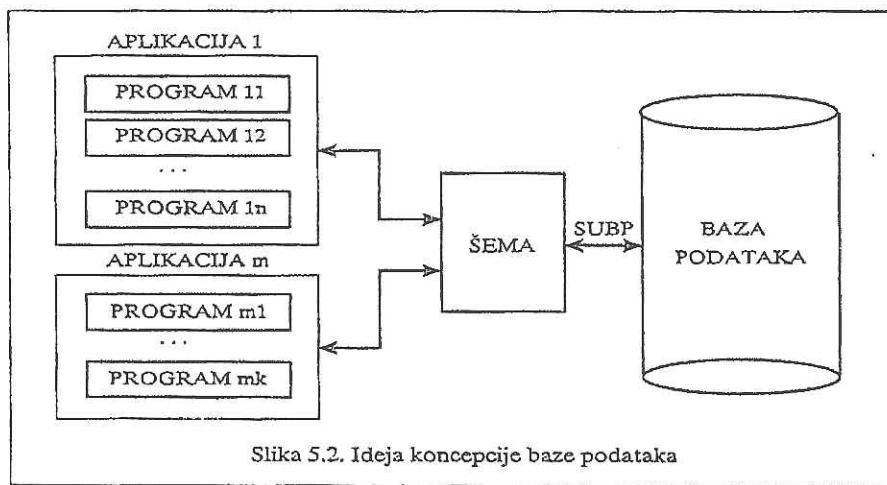
Održavanje integriteta baze podataka podrazumeva proveru integriteta i oporavak iz bilo kojeg nekorektnog stanja koje može biti otkriveno. Za održavanje integriteta baze podataka zadužen je administrator baze podataka, koji pri tome koristi pogodnosti SUBP.

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.

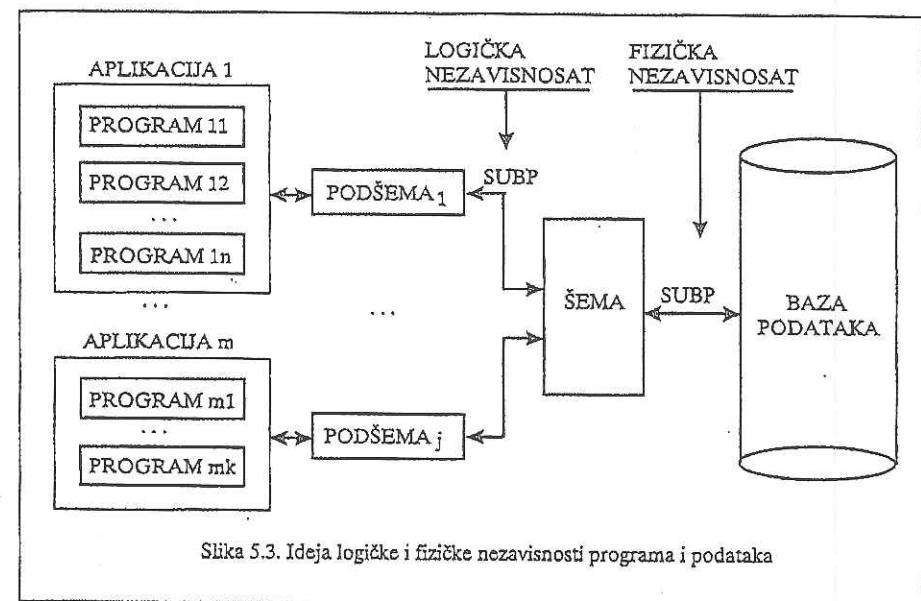
Šema i podšema predstavljaju modele podataka realnog sistema i jednog njegovog dela. U principu, ta dva modela se nalaze na istom nivou apstrakcije.

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.

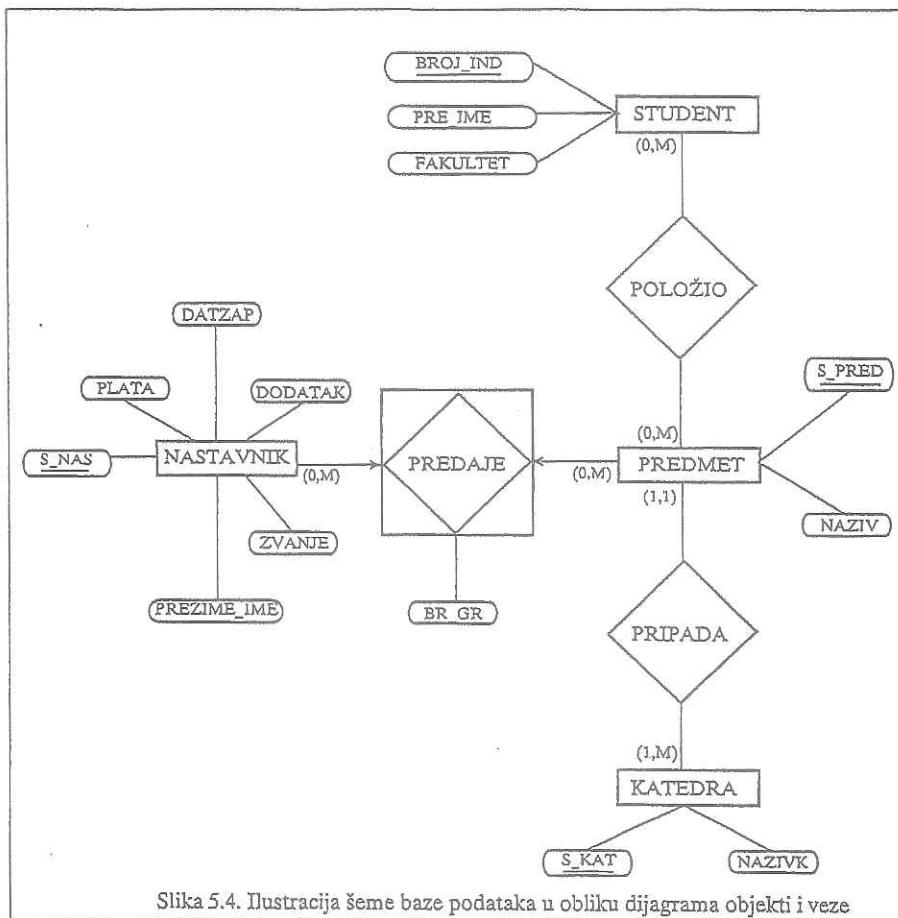
Fizička nezavisnost podataka znači da se fizički raspored i organizacija podataka mogu menjati, a da se pri tome ne mora menjati šema, podšema i programi.

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 pohađa (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.

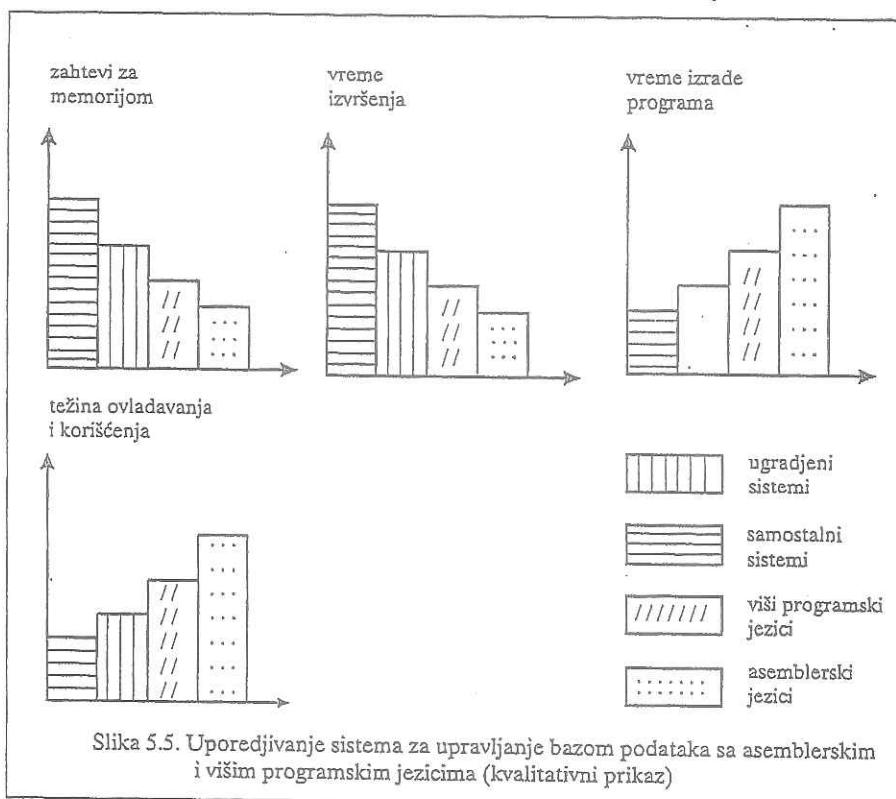
Za njih je potreban jedan jednostavan jezik za rukovanje podacima koji i početnik u računarskoj tehnici može lako naučiti, a bazira se prvenstveno na opisu jednostavnih logičkih veza i osnovnih matematičkih operacija. Uz pomoć ovog jezika korisnici sami definišu načine povezivanja podataka za svoje potrebe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

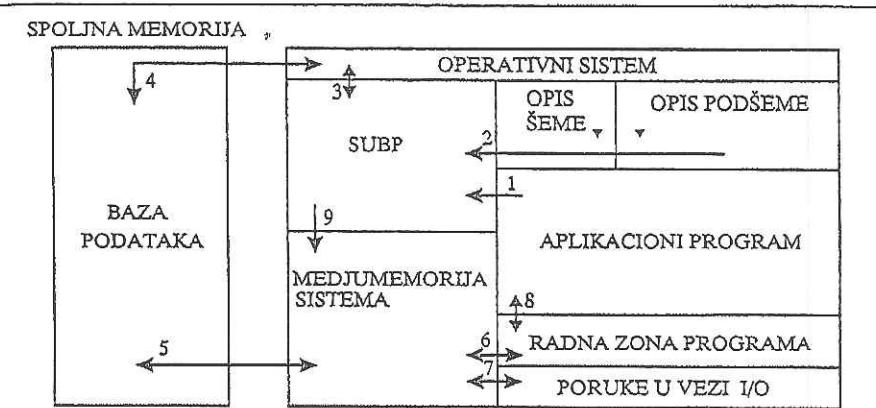
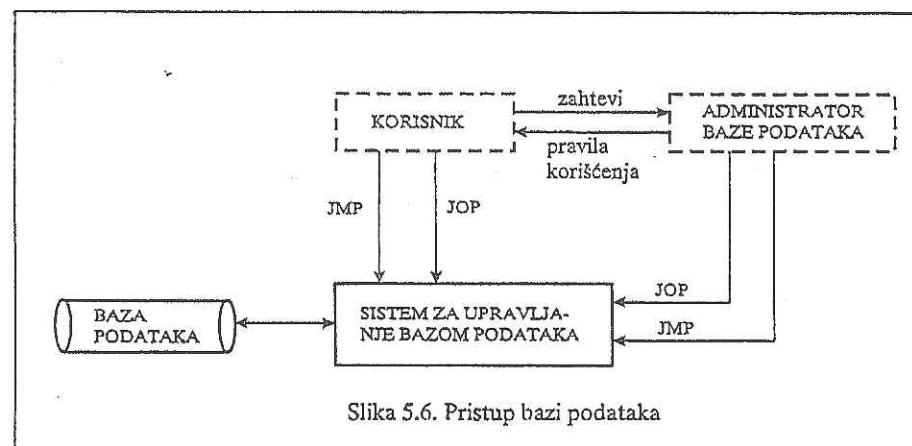
Sve ono što čini bazu podataka može se podeliti u sledeće dve grupe:

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.

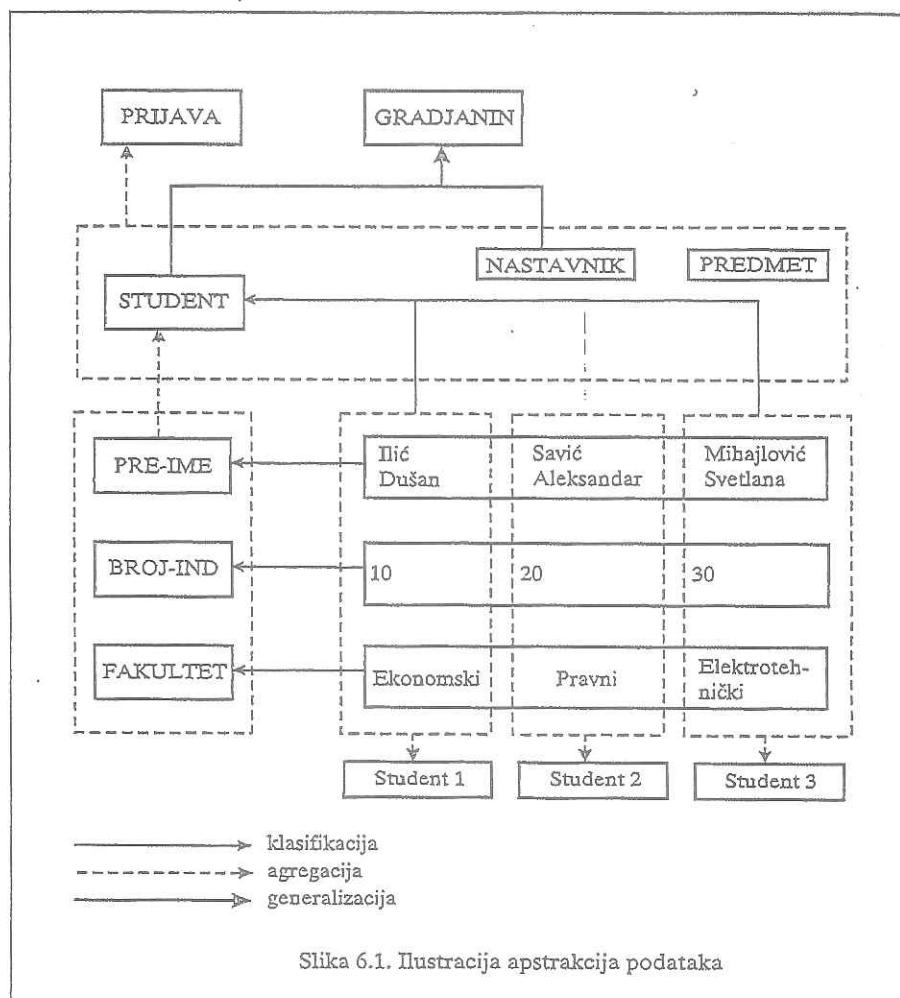
Na primer, tipovi objekata STUDENT, RADNIK, PENZIONER, mogu se generalizovati u tip objekta GRADJANIN.

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

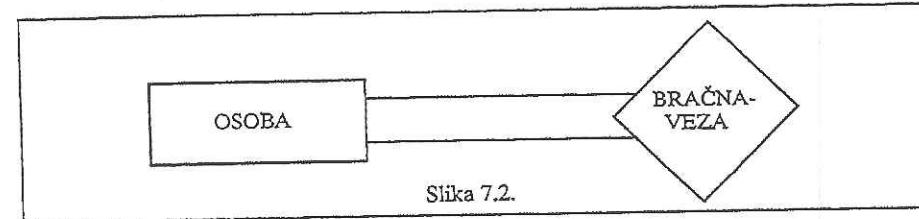
Primitivni koncepti u MOV su: objekt, veza (medju objektima) i obeležje objekta.

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.

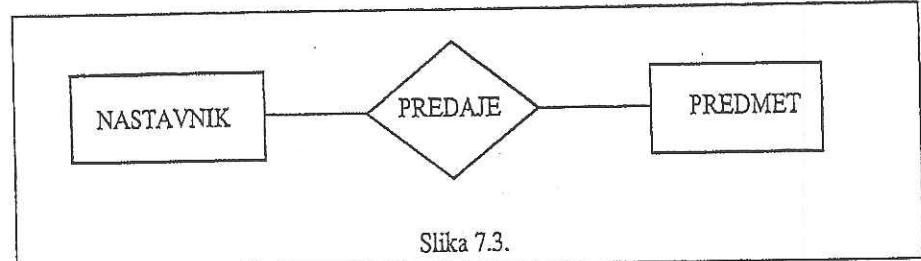
Veze u modelu opisuju način povezivanja objekata. Broj tipova objekata koji učestvuju u vezi definiše *red veze*. Mogu postojati *rekurzivne*, *binarne* i veze *višeg reda*.

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 .

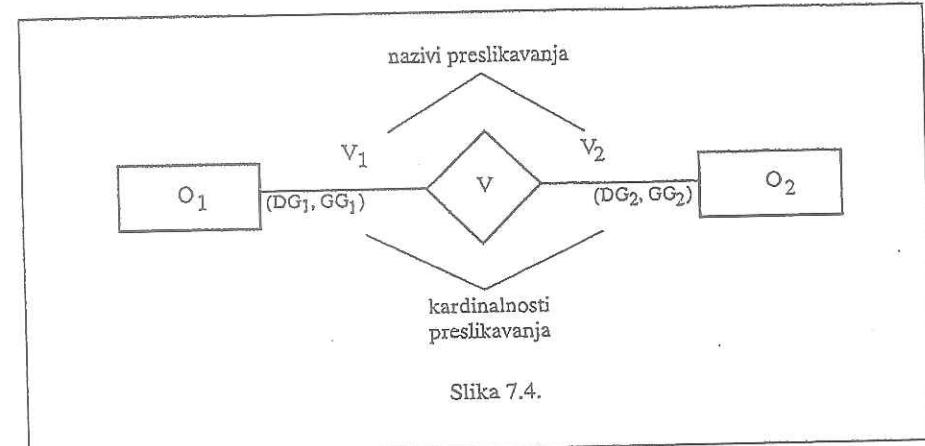
Za svako od ovih preslikavanja definiše se par (DG, GG), gde DG (donja granica) predstavlja najmanji, a GG (gornja granica) najveći broj elemenata partitivnog skupa u koji se preslikava jedan elemenat skupa originala. Par (DG, GG) predstavlja kardinalnost preslikavanja. DG može imati vrednost 0, 1 ili 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.

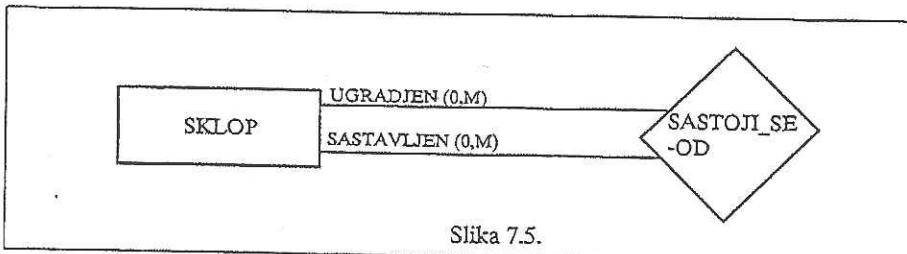


Slika 7.4.

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).



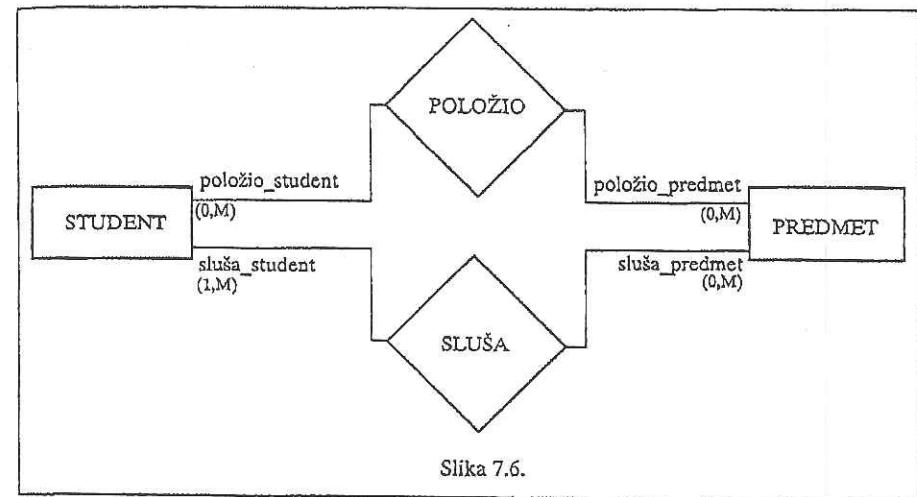
Slika 7.5.

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.

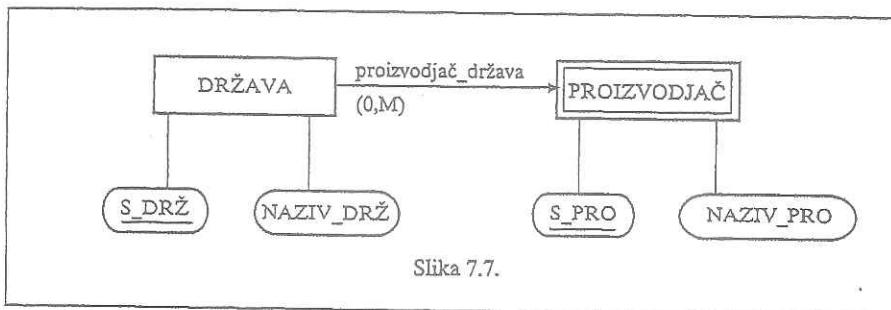
Tip slabog objekta

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:

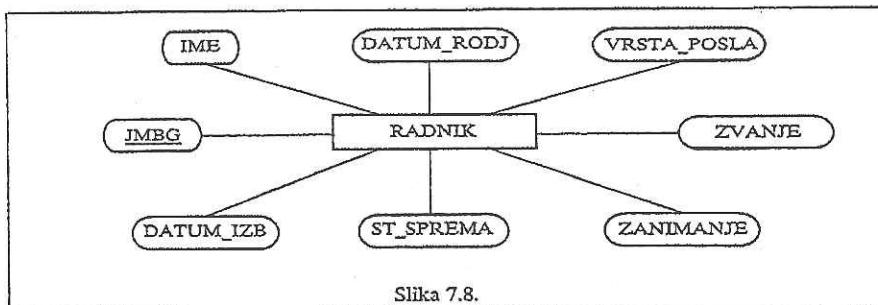
Ekskluzivna specijalizacija - svako pojavljivanje tipa specijalizuje se u jedan podtip i tada je $DG = 1$ i $GG = 1$.

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

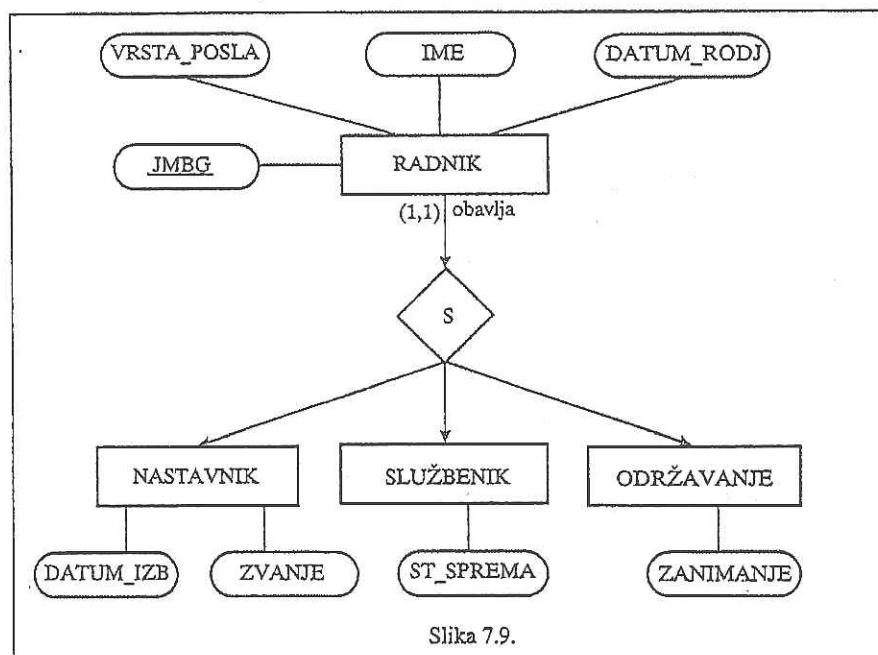
Neka smo na primer, u procesu izrade MOV definisali tip objekta RADNIK dat na slici 7.8. Uočavamo da su neka obeležja specifična samo za neke od radnika (tj. za neke od pojava tipa objekta RADNIK). Na primer, obeležje ZVANJE i DATUMIZB (datum izbora u zvanje) pripadaju samo onim radnicima koji za obeležje 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 pojавama 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.

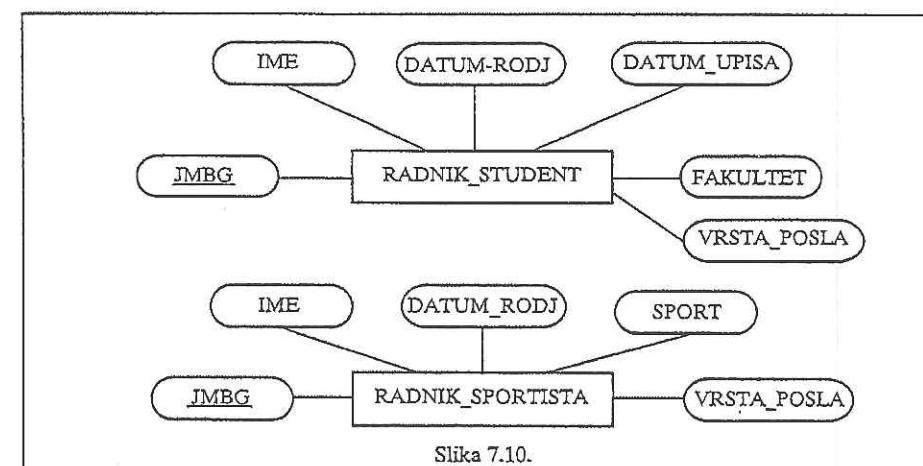


Slika 7.9.

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.

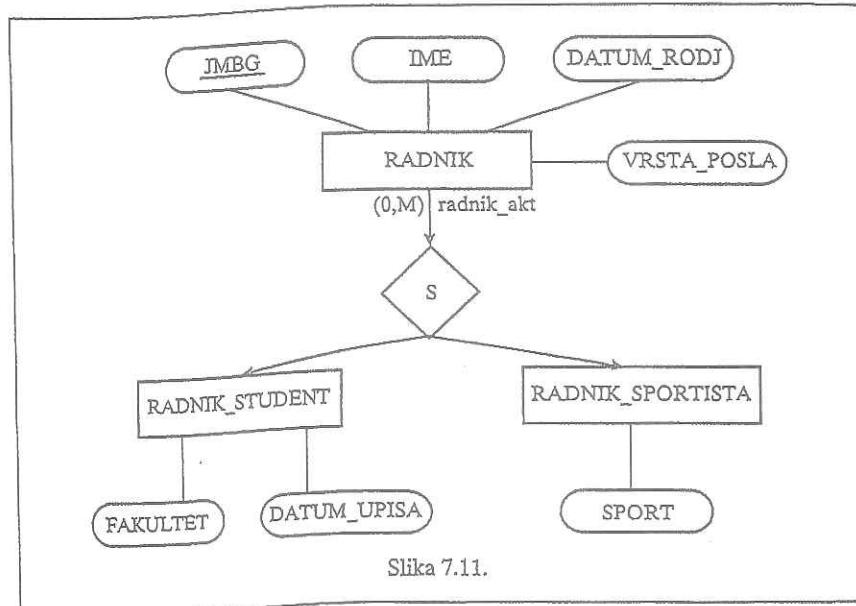
Posmatrajući sliku 7.9. i 7.11. uočavamo razliku u predstavljanju na DOV ekskluzivne i neekskluzivne specijalizacije. S obzirom da radnici studenti i radnici sportisti jesu radnici, model podataka dat DOV na slici 7.9. odnosi se i na njih. Da bi se u model uključilo i znanje predstavljeno DOV na slici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

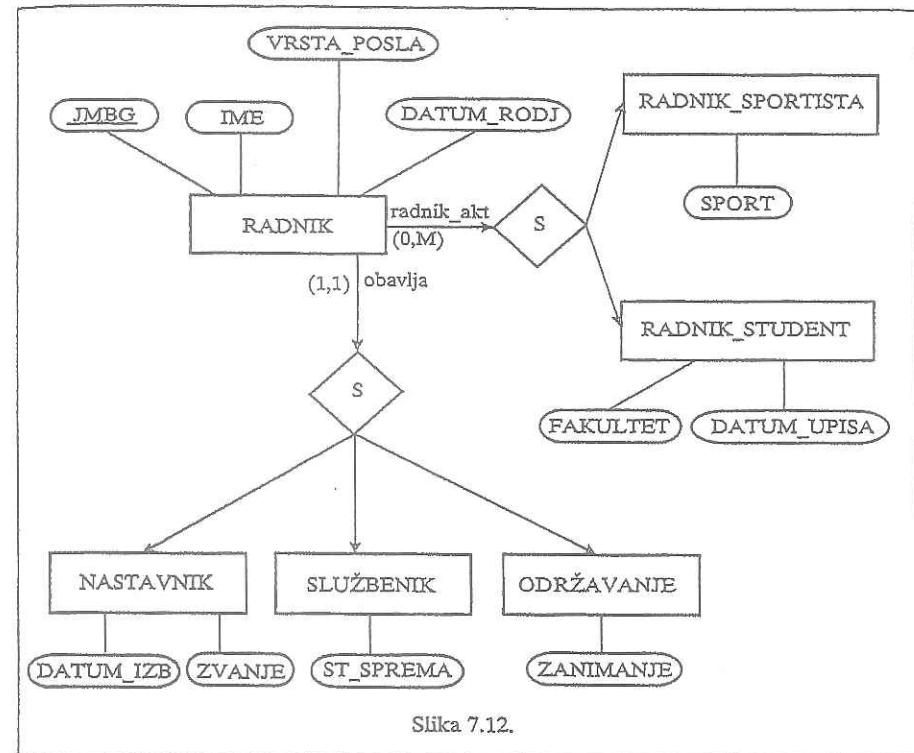
Jedan tip objekta može se po više različitih kriterijuma specijalizovati u različite skupove podtipova kao što je to ilustrovano za tip objekta radnik na slici 7.12.

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*.

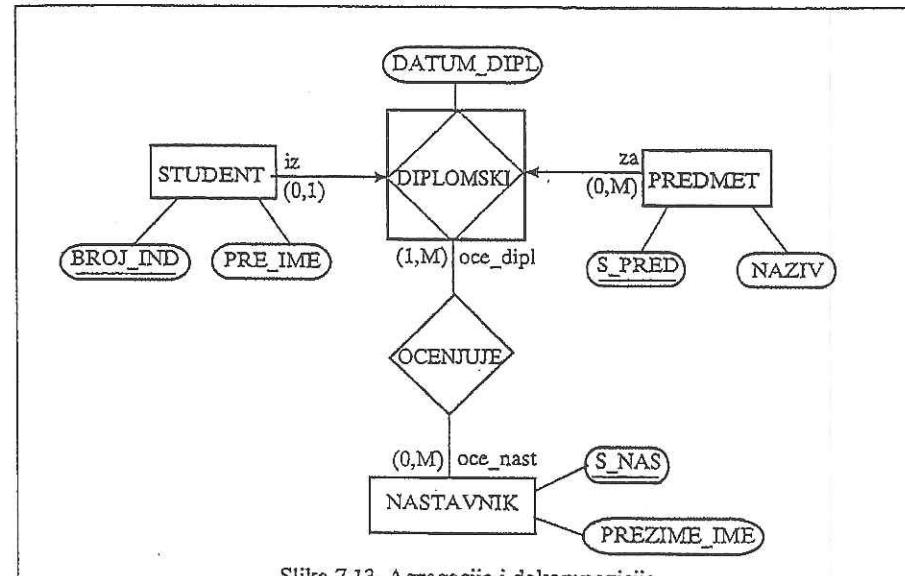
Agregirani objekat se u modelu tretira kao i bilo koji drugi objekat. To znači da on može da ima svoja obeležja i/ili da bude u vezi sa nekim drugim objektima (moguće aggregiranim, takodje), da ima svoje podtipove i slično.

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

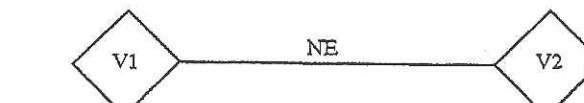
Pri gradnji DOV podtoče sledeća formalna ograničenja:

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.



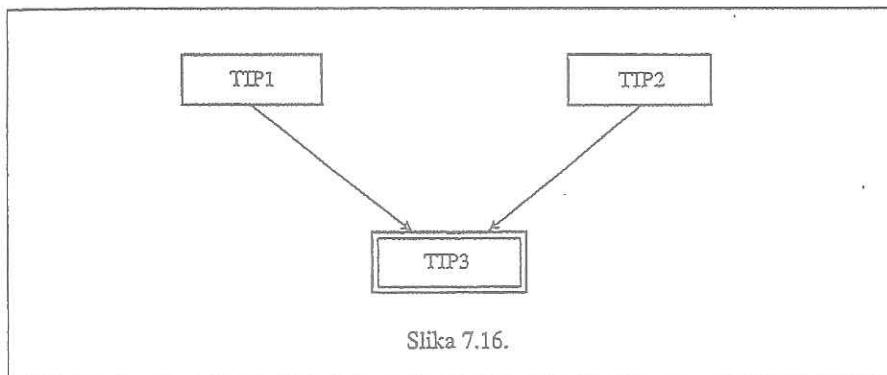
Slika 7.14.

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:

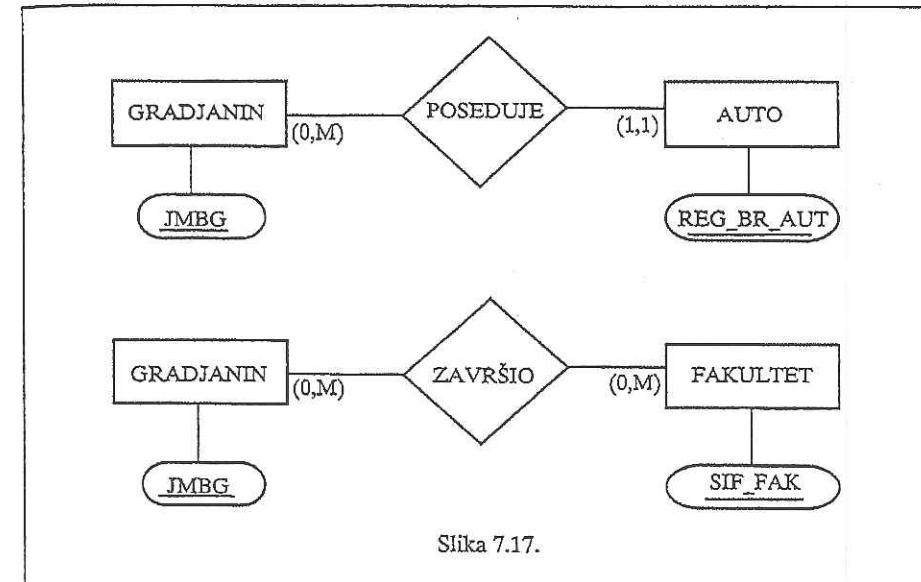
- (1) Tipa (1:1) - jedan tip objekta ima samo jednu vrednost nekog obeležja, a tu vrednost obeležja može imati samo jedna pojava tipa objekta. Na primer, jedan gradjanin za obeležje JMBG ima jednu vrednost, i ta vrednost

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

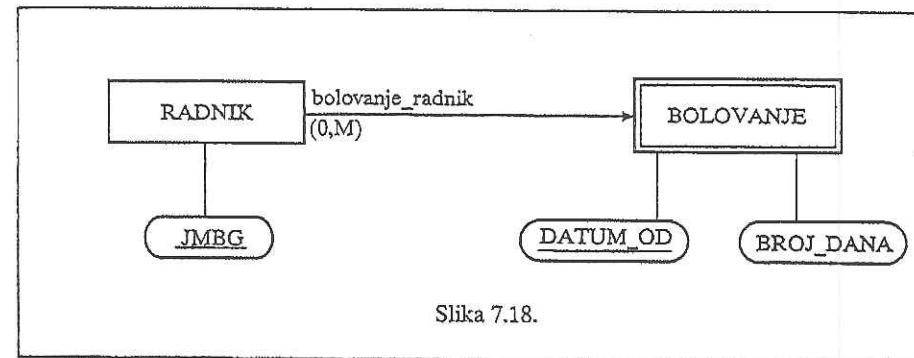
- (2) Tipa (1:M) - jedan tip objekta ima samo jednu vrednost nekog obeležja, i istu vrednost obeležja mogu imati više pojava tipa objekta. Na primer, jedan gradjanin za obeležje DATUM_RODJ ima jedan datum, ali isti datum rođenja može imati više gradjana.
- (3) Tipa (M:1) - jedan tip objekta može imati 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.



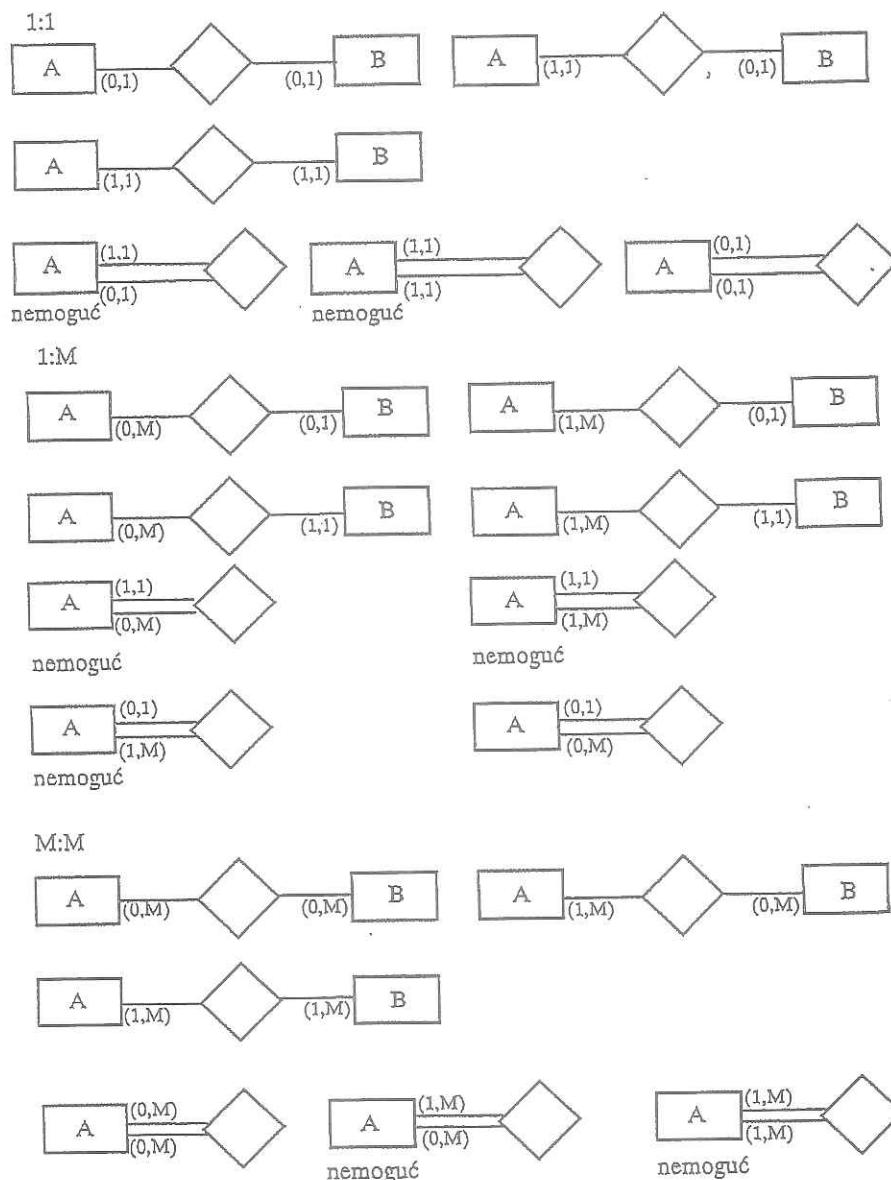
Slika 7.17.



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,y₁,y₂)**

- 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 ⁻¹ | - inverzno preslikavanje preslikavanju P, |
| y.P ⁻¹ | - 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,y₁,y₂,...,y_n)**

SPI: vspi X.P₁, vspi X.P₂,..., vspi X.P_n

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 poveži_X.Pi (x,yi)
        else case vspi of
            R: odbij_operaciju;
            N: poveži_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.P₁, vspi X.P₂,..., vspi X.P_n

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_{1s},x_{1n})

Argument x_{1s} je stara, a x_{1n} 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,y₁,y₂)
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;
else 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 odabrat 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.

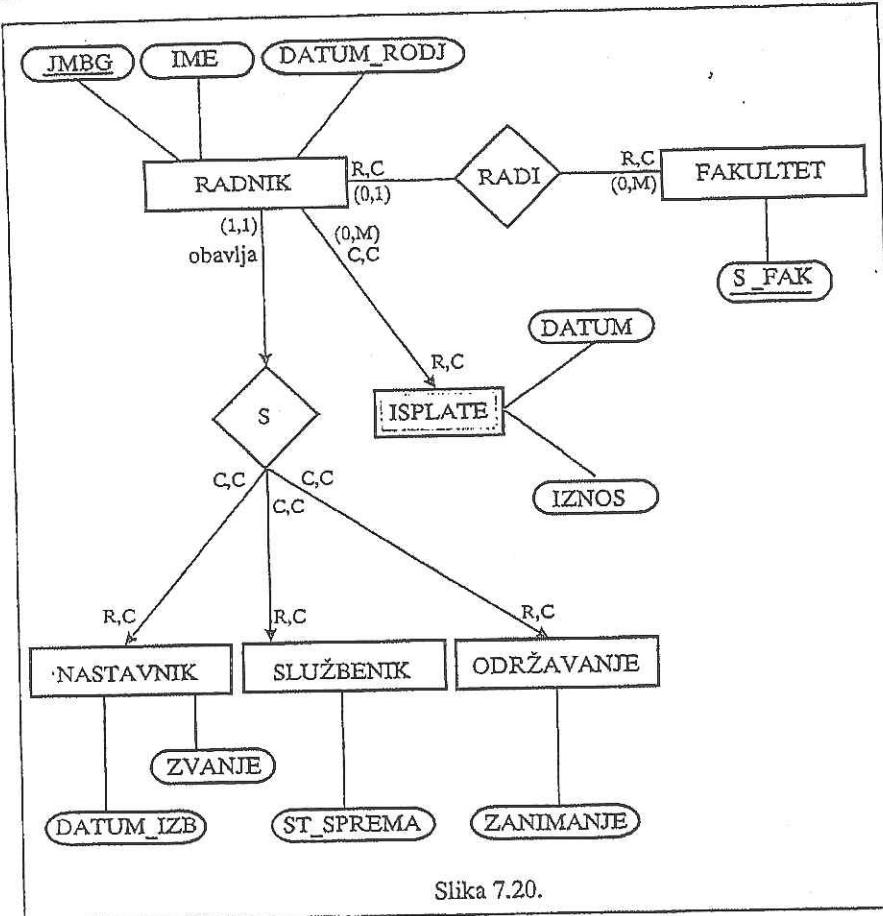
Akcija za operaciju RAZVEŽI odnosi se na slučaj kada bi time bila narušena donja granica preslikavanja y.P⁻¹.

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 ubaćeno 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]
  
```

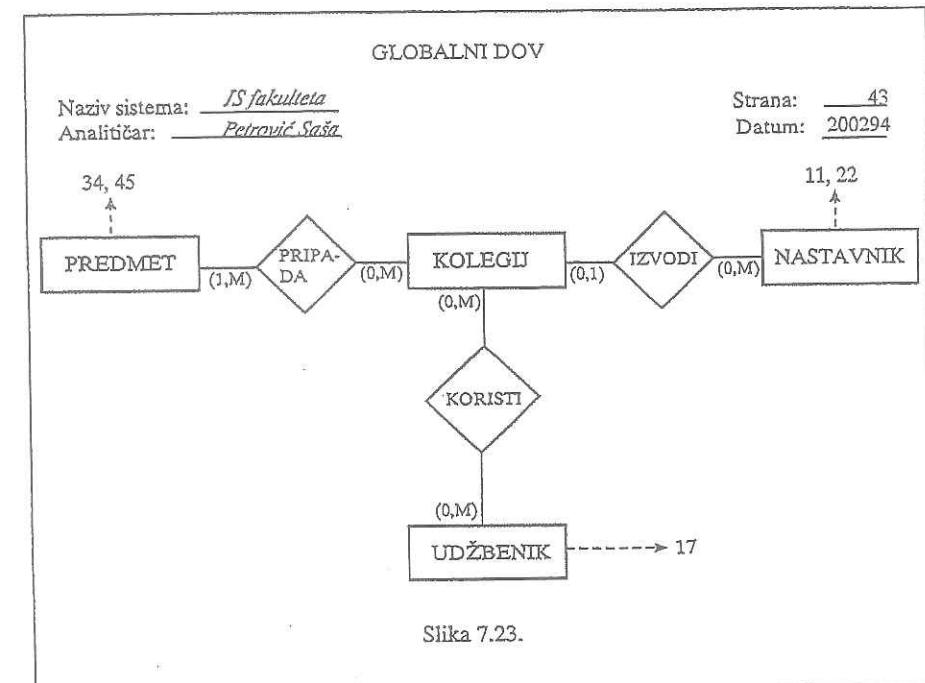
Broj pojavljivanja: Maximum 10000 Srednji 1000 Minimum _____
 Ovlašćenje za promene: Odeljenje opštih poslova
 Uslovi ubacivanja: Pri zapošljavanju
 izbacivanja: Ostaje trajno u bazi podataka

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, PLANOVI 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:

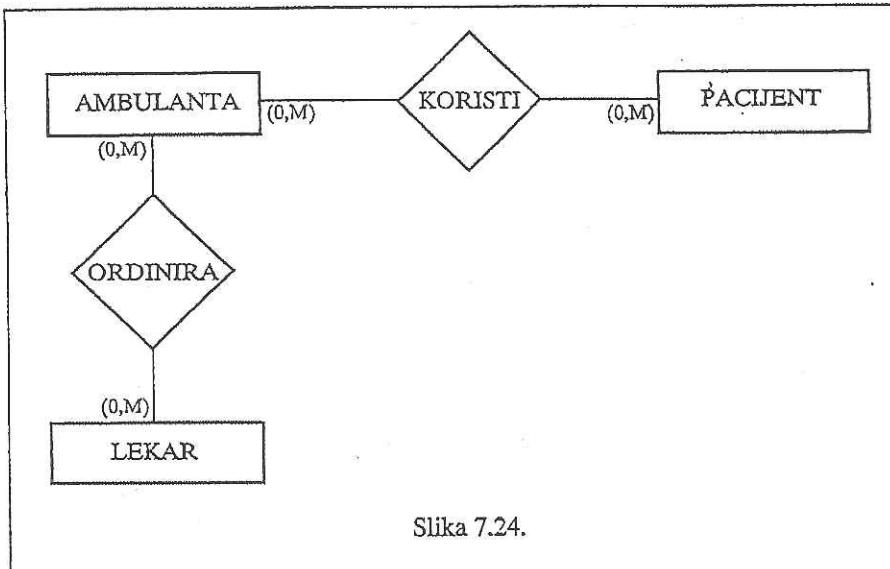
1. Jednu ambulantu može koristiti više pacijenata.
2. Jeden pacijent može koristiti usluge više ambulant.
3. U jednoj ambulanti ordinira više lekara.
4. Jeden lekar može ordinirati u više ambulant.

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*.

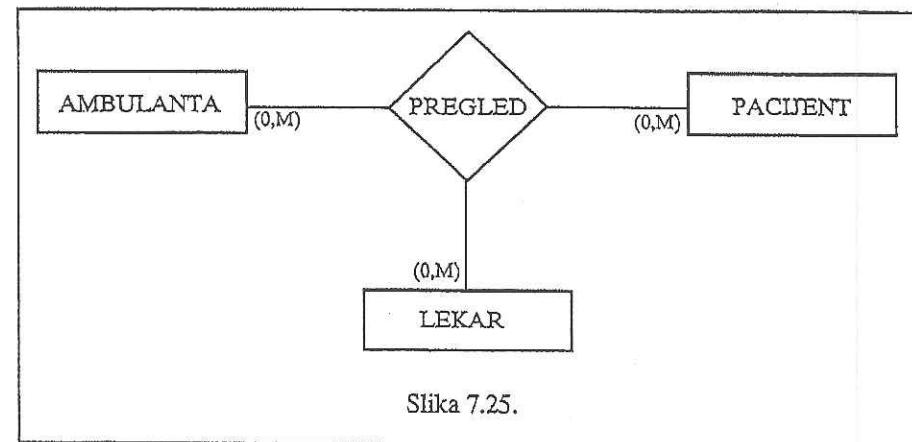
Situacija u kojoj svi pacijenti koji koriste usluge jedne ambulante ne moraju koristiti isti skup lekara koji ordiniraju u toj ambulanti može se opisati sledećim iskazima:

- 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 redundantse 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.

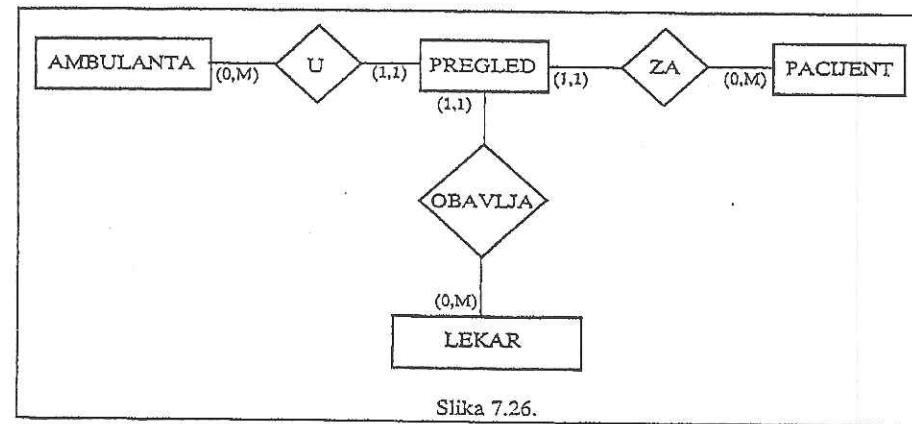


Slika 7.25.

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.

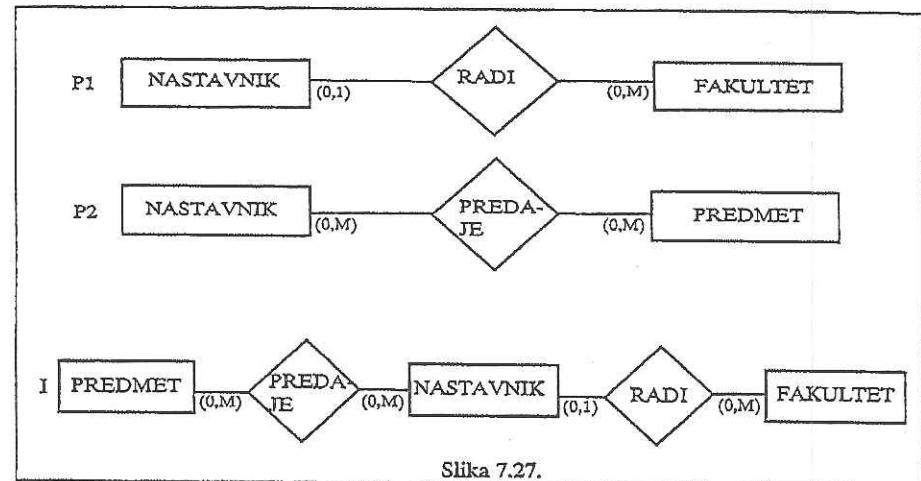
Nizom primera, u nastavku ilustrujemo principe integracije. Sa Pi su označeni podmodeli koji se integrišu, dok je rezultat integracije označen sa I. Zbog jednostavnosti ilustracija na DOV nisu predstavljana obeležja tipova objekata, mada ne možemo reći da su ta obeležja bez značaja za integraciju.

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.

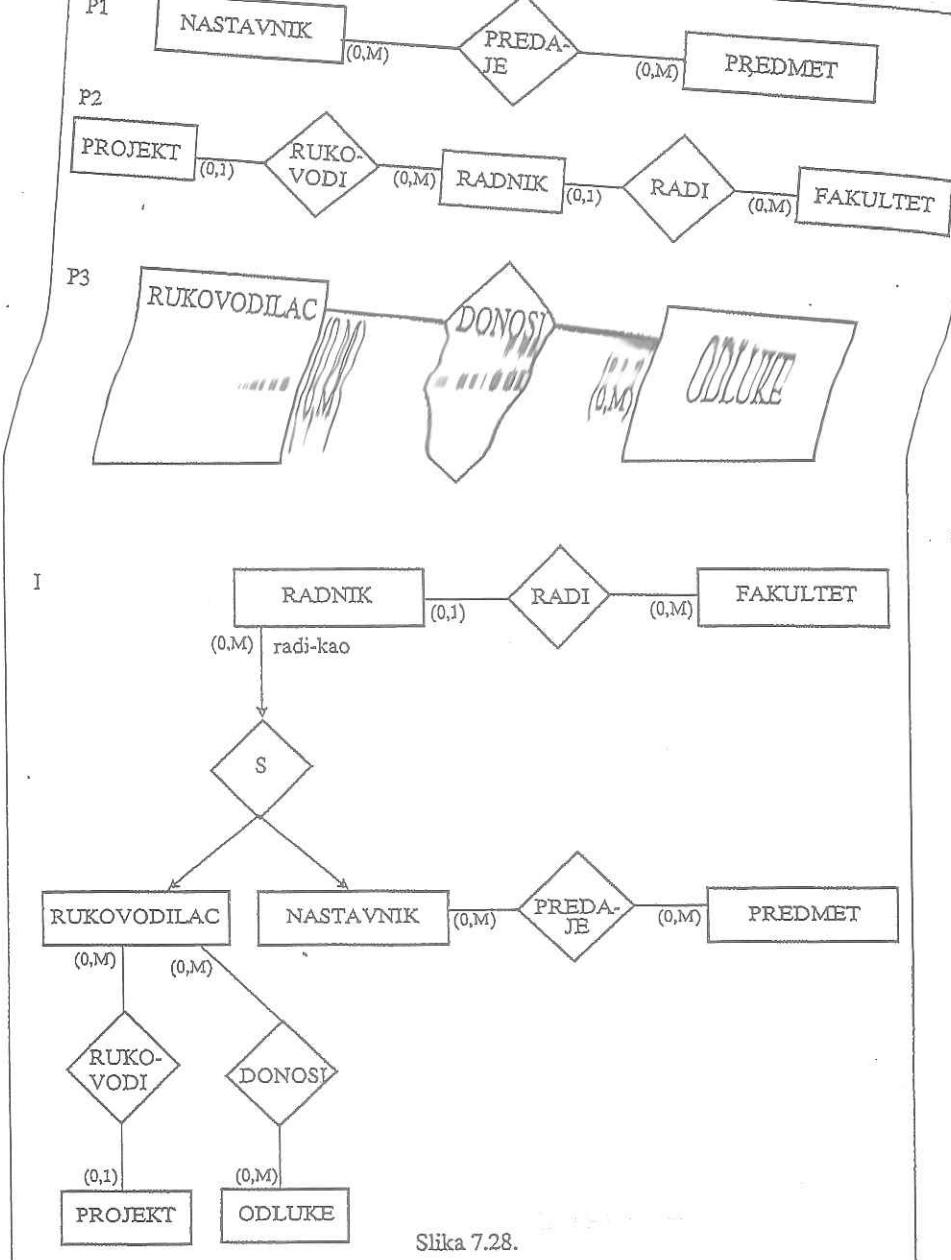
Na slici 7.29. prikazana je integracija podmodela koji sadrže tipove objekata NASTAVNIK i OP_OSOBLJE (operativno osoblje: službenici, održavanje) koji se nalaze u generalizacijskom odnosu sa ekskluzivnom specijalizacijom. Pretpostavka je da tipovi objekata NASTAVNIK i OP_OSOBLJE imaju isti identifikator (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.

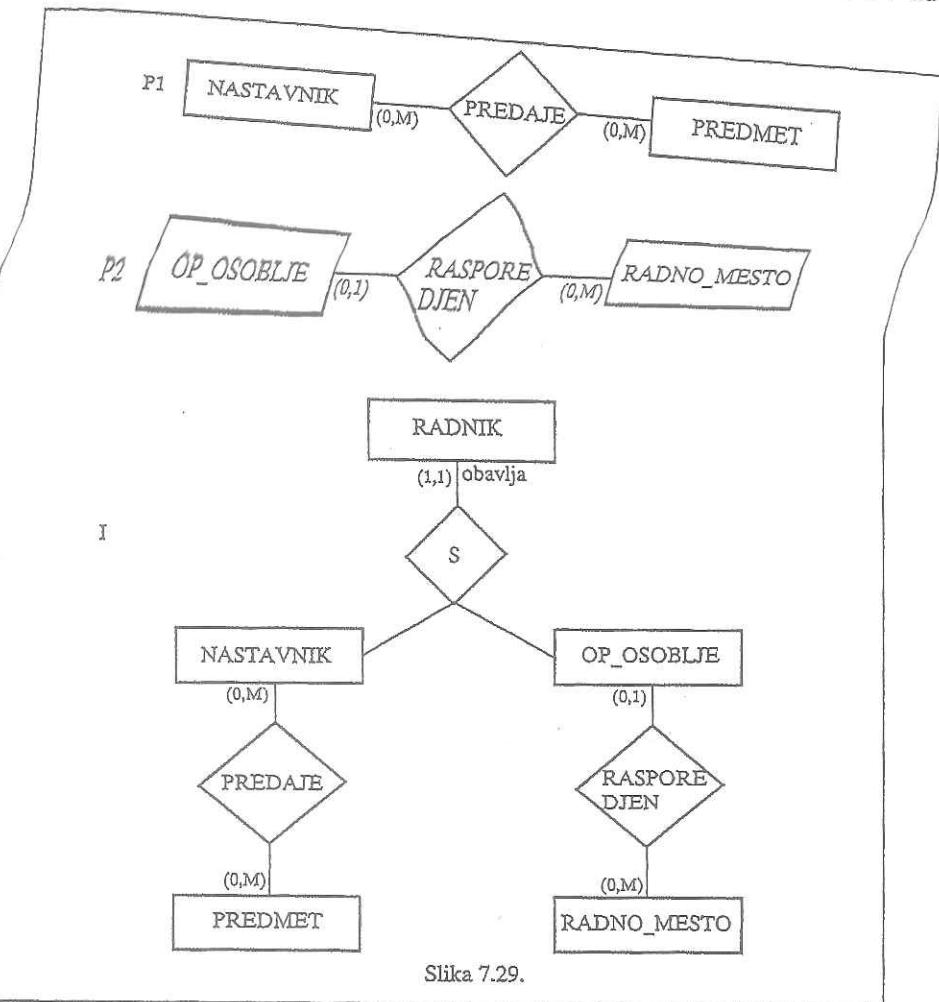
Mogućnosti i načini integracije veza zavise od njihovih osobina i to:

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

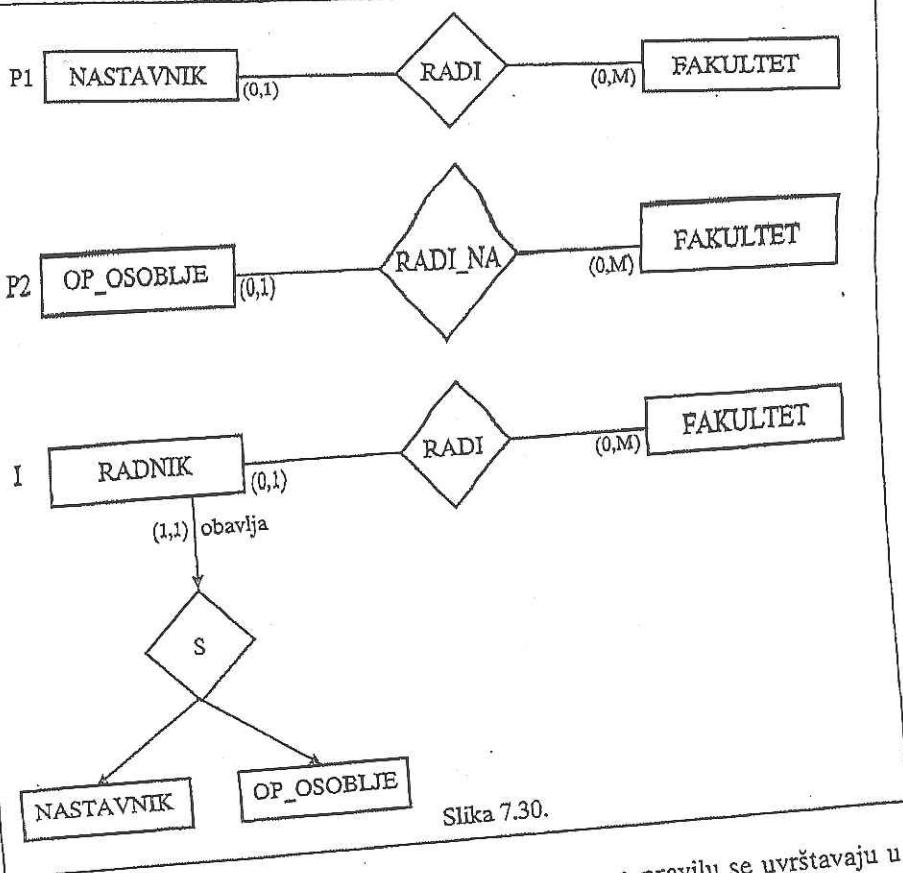
mogu biti predstavljene jednom vezom RADI, kako je to učinjeno u DOV na slici 7.30. označenim sa I.



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.

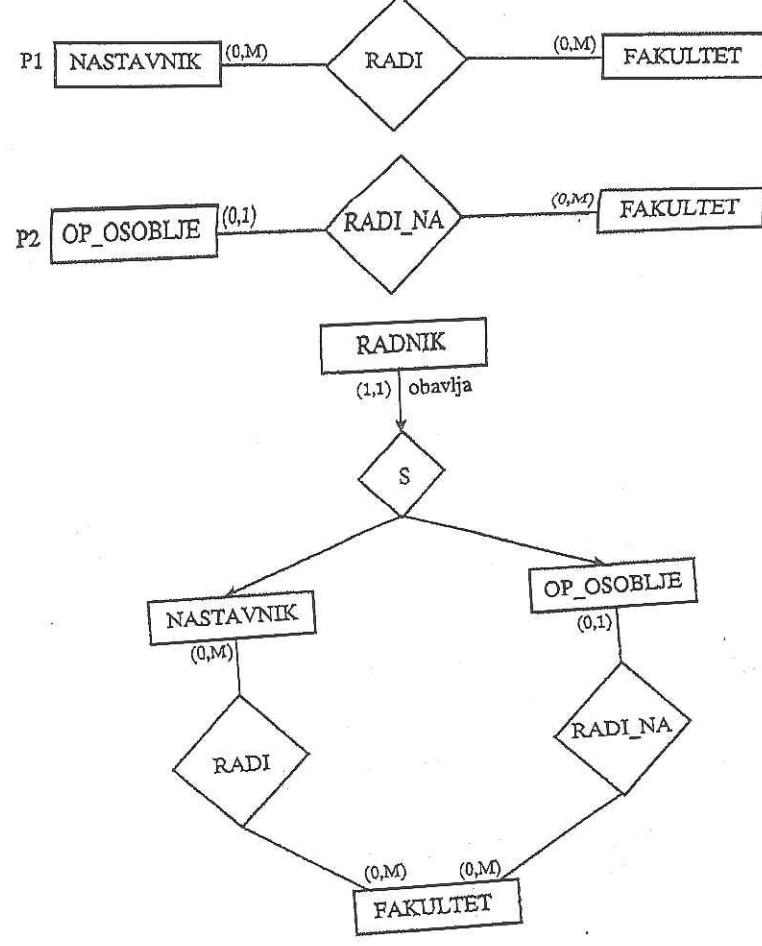
Veza koja tranzitivno sledi iz drugih veza sadržanih u modelu podataka je redundantna. Na slici 7.33. veza ZAPOSLEN ne postoji u integriranom modelu I jer tranzitivno sledi iz veza PRIPADA i U_SASTAVU.

Redundantnost veze ZAPOSLEN možemo objasniti sledećim iskazima:

1. Radnik PRIPADA tačno jednoj katedri.
2. Katedra se nalazi U_SASTAVU tačno jednog fakulteta.

Sledi:

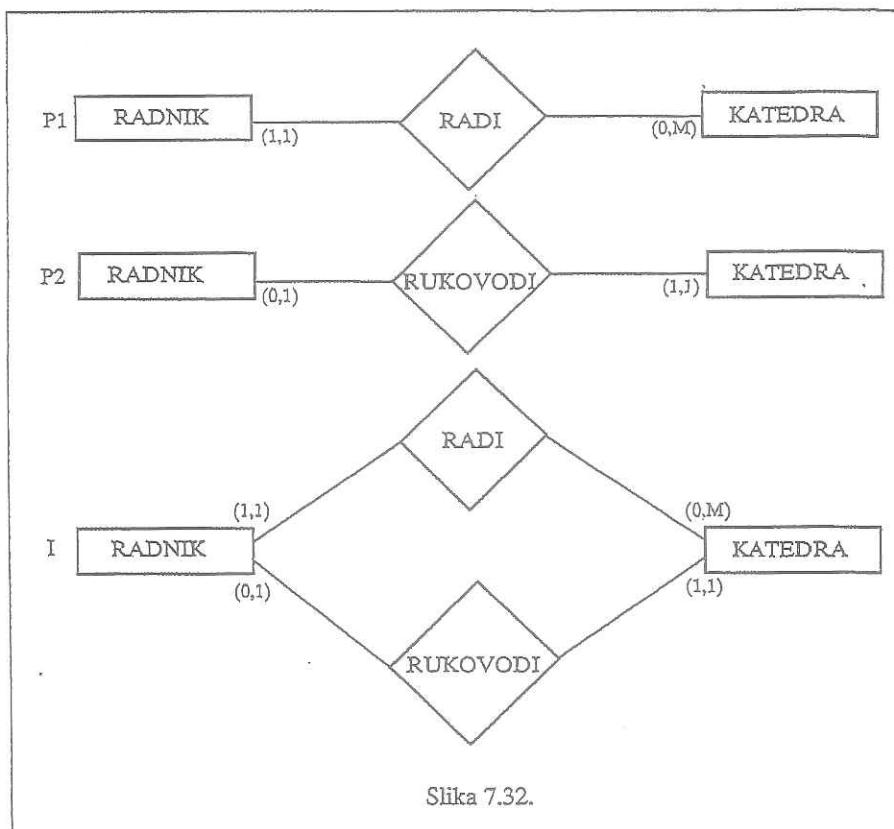
3. Ako je poznata pripadnost radnika katedri, i katedre fakultetu, onda je poznata i pripadnost radnika fakultetu. Zbog toga je veza ZAPOSLEN koja to eksplicitno iskazuje redundantna.



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.

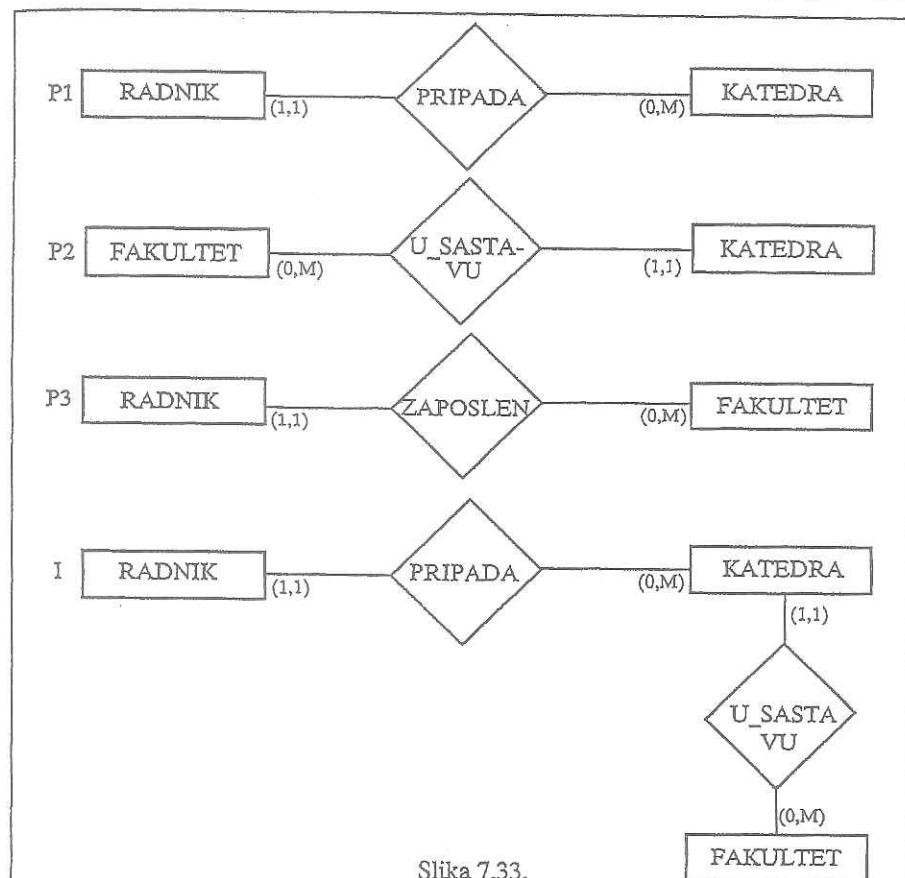


Slika 7.32.

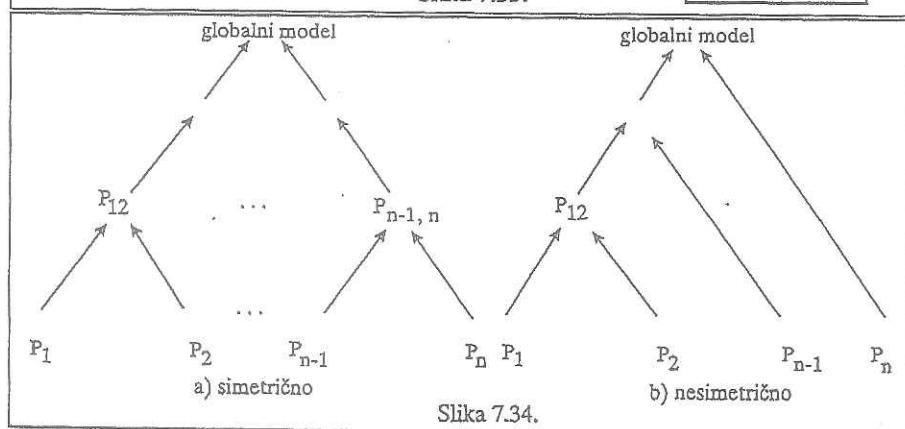
Kao drugo prihvatljivije rešenje je integrisanje u jednom koraku dva podmodela odnosno binarno integriranje koje može biti simetrično ili nesimetrično kako je to ilustrovano na slici 7.34.

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 šema relacija. Takav je na primer, referencijalni 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 šema 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 (Θ - 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 Θ - 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 Θ - 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 Θ - spoja.

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

Dруги poseban slučaj je kod kojeg je Θ 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 Θ - 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 Θ - spojem. Kao rezultat Θ - 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-torke (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, ...)

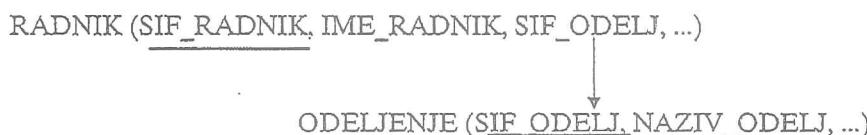


RADNIK (SIF_RADNIK, IME_RADNIK, ...).

Obeležja SIF_RADNIK i SIF_RUKOV imaju iste domene.

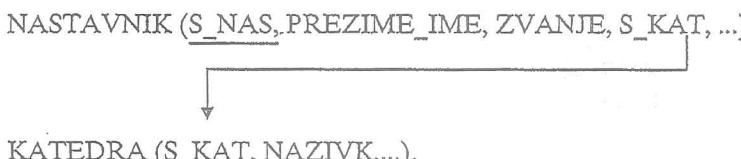
- b) Za svaku pozivajuću n-torku mora postojati pridružena ciljna n-torka. Za razliku od prethodnog slučaja, ciljna n-torka se može menjati. Brisanju ciljne n-torke mora u ovom slučaju prethoditi ažuriranje vrednosti spoljnog ključa u pozivajućim n-torkama.

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 $R \cup S = P$.

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

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

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

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

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

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

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

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

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:

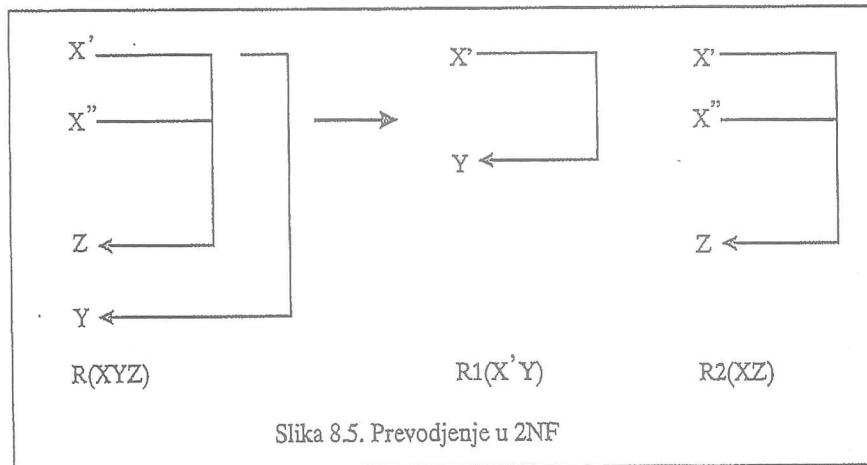
- Y nisu ključna obeležja,
- X je kandidat za ključ i
- $X \rightarrow Y$ je parcijalna FZ.

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.