FAKULTET ORGANIZACIONIH NAUKA
Laboratorija za informacione sisteme

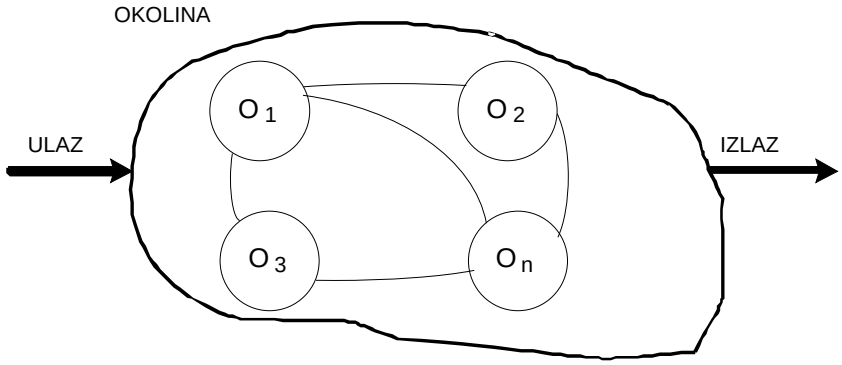
PROŠIRENI MODEL OBJEKTI-VEZE

Beograd, oktobar 2000.

1. UVOD - PODACI, INFORMACIJE, BAZE PODATAKA I INFORMACIONI SISTEMI

Metodologija razvoja informacionih sistema zahteva da se precizno definiše šta se pod pojmom informacionog sistema podrazumeva, koje su njegove funkcije i kakav je njegov položaj u sistemu u kome deluje. Metodologija razvoja informacionih sistema treba da bude opšta, primenljiva na sisteme bilo koje vrste, odnosno na neki "opšti sistem". Zbog toga je potrebno poći od sledećih opštih pojmova i definicija:

(1) **Sistem je skup objekata i njihovih veza** (medjusobno povezanih objekata). Objekti u sistemu se opisuju preko svojih svojstava koja se nazivaju atributima. Slika šematski predstavlja i detaljnije obrazlaže ovu definiciju.



Slika 1. Opšta definicija sistema

Granice sistema definišu skup objekata koji će se u tom sistemu posmatrati. Objekti nekog sistema su, naravno, povezani sa objektima van njegovih granica, a ovi sa nekim drugim "daljim" i tako dalje. Neophodno je zato odrediti granice sistema koje izoluju objekte od interesa od "okoline" sistema. Dejstvo okoline na sistem naziva se **"ulaz"**, a dejstvo sistema na okolinu "izlaz" sistema.

Sistem na Slici 1 može predstavljati mrežu puteva ili ulica, sistem za prenos električne energije, cirkulaciju dokumenata unutar neke organizacije, kretanje materijala koji se obradjuje itd. Objekti u sistemu mogu biti veoma različiti, a veze izmedju objekata u sistemu i dejstvo okoline na sistem se ostvaruje na tri načina: razmenom materije, energije ili informacija. ("Klasična predstava o svemiru koji se sastoji od materije i energije, mora da ustupi mesto predstavi o svetu sastavljenom od tri komponente: energije, materije i informacija, jer bez informacija organizovani sistemi nisu mogući. Jedino moguće materijalističko tumačenje održavanja organizovanosti je neprekidno izvlačenje iz spoljnjeg sveta (okoline) i iz samog sistema mase informacija o pojavama i procesima koji se tu odvijaju" - A. Lerner).

(2) U svakodnevnom govoru reč informacija ima smisao obaveštenja, objašnjenje prenošenja znanja. Za ovaj pojam obično se daju sledeće definicije:

- **"Informacija je kapacitet povećanja znanja" - I.Wilson**
- **"Informacija je nešto što ukida ili smanjuje neodredjenost" - N.Winer.**

Sa tačke gledišta upravljanja i donošenja odluka, informacija se može tretirati kao svaka vrsta znanja ili poruka koja može da se upotrebi za poboljšanje upravljanja u nekom sistemu.

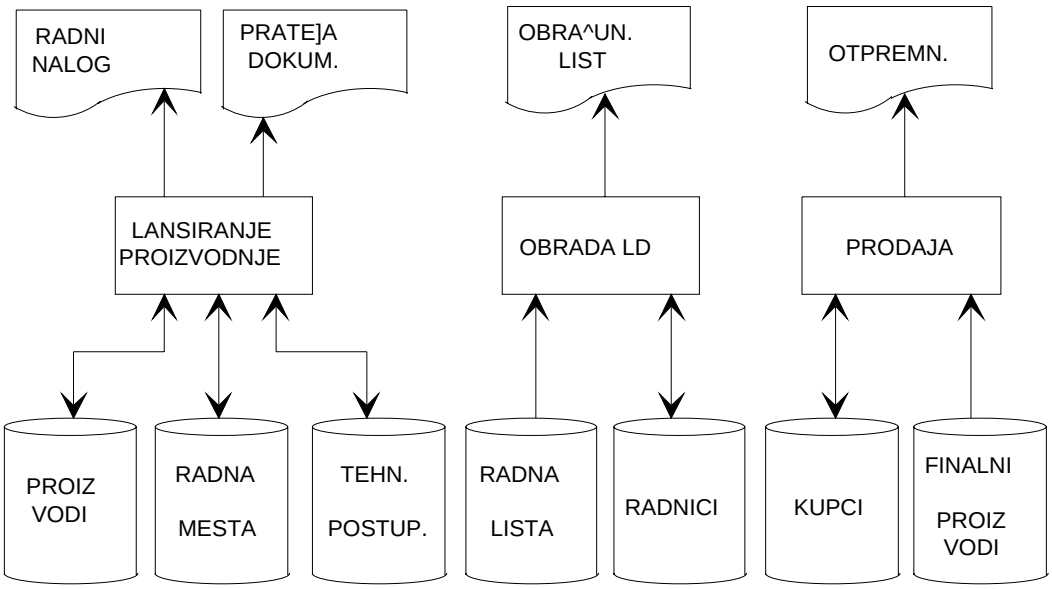
(3) Ako se povežu definicije pojmova sistema i informacije, može se izvesti sledeća opšta definicija informacionog sistema:

Informacioni sistem je sistem u kome se veze izmedju objekata i veze sistema sa okolinom ostvaruju razmenom informacija.

Informacioni sistemi (IS) su sastavni deo upravljanja ("održanja željene organizovanosti") nekog sistema i sa te tačke gledišta može im se pridodati atribut "**upravljački**" i definisati **upravljački informacioni sistema kao sistem koji prenosi, čuva i obradjuje podatke u informacije potrebne za upravljanje.**

(4) U svakodnevnom govoru reči podatak i informacija se koriste kao sinonimi. Medjutim, za precizno razgraničenje koncepata o kojima govorimo, neophodno je i ova dva pojma precizno definisati i razgraničiti. **Podatak je kodirana predstava o nekoj činjenici iz realnog sveta, on je nosilac informacije i služi za tehničko uobličavanje informacija, kako bi se one mogle sačuvati ili preneti. Informacija je protumačeni podatak o pojavi koju podatak prikazuje**. Krajnje tumačenje nekom podatku daje sam primalac (čovek), uz pomoć različitih postupaka obrade podataka. **Osnovna funkcija informacinog sistema je čuvanje i prenos podataka o činjenicama iz sistema i njegove okoline i njihova obrada u informacije koje zahteva korisnik**.

(5) Da bi smo definisali koncept baze podataka, analizirajmo nedostatke klasične obrade podataka, manuelne ili automatizovane preko skupa programa i skupa nepovezanih datoteka. Na Slici 2 prikazan je "sistemski dijagram" jednog skupa "aplikacija" u nekom informacionom sistemu. Svaka aplikacija je razvijana posebno, koristi svoje "privatne" datoteke sa fizičkom strukturom podataka pogodnom za tu aplikaciju.



Slika 2. Klasična obrada podataka. Nezavisne "privatne" datoteke za svaku aplikaciju

Osnovni nedostaci ovakve obrade podataka su:

- **Redundansa podataka**, odnosno višestruko pamćenje istih podataka je neminovno. Na primer, isti podaci o proizvodima se, u nekoj proizvodnoj radnoj organizaciji pamte i nekoliko desetina puta, ili isti podaci o gradjanima u nekom komunalnom informacionom sistemu i stotinu puta. Višestruko skladištenje istih podataka dovodi do nepremostivih problema pri njhovom ažuriranju. Kada se neki podatak promeni, to se mora učiniti na svim mestima na kojima se on čuva. To, ne samo da značajno povećava troškove obrade podataka, nego je organizaciono praktično neizvodljivo, pa se, u raznim izveštajima, pojavljuju različite verzije istog podataka.

- **Zavisnost programa od organizacije podataka**. Programi su zavisni i od logičke i od fizičke strukture podataka. Fizička struktura podataka definiše način memorisanja podataka na spoljnim memorijama, a logička struktura je struktura podataka koja je predstavljena programeru. U klasičnim datotekama razlika fizičke i logičke strukture podataka je mala. Zavisnost programa od logičke strukture se ogleda u tome što

program zavisi, na primer, od naziva i redosleda polja u zapisu, što ubacivanje novog polja u zapis ili bilo kakvo drugo restruktuiranje zapisa, koje ne menja sadržaj podataka koje program koristi, ipak zahteva i izmenu samog programa. Fizička zavisnost se ogleda u tome što program zavisi od izabrane fizičke organizacije datoteke i izabrane metode pristupa, načina sortiranja i slično. U osnovi i logička i fizička organizacija podataka su prilagodjene konkretnom programu. Novi zahtevi za informacijama, iz podataka koji već postoje u datotekama, mogli bi zahtevati izmenu fizičke i logičke strukture podataka, a to bi zahtevalo izmene u ranije razvijenim programima. Umesto toga, obično se za novi program stvara nova datoteka, a to dalje povećava redundansu podataka.

- **Niska produktivnost u razvoju IS**. Struktuiranje podataka u nezavisan skup datoteka je jedan od uzroka veoma niske produktivnosti u razvoju IS. Na primer, čak i kada postoje svi podaci koji se u nekoj aplikaciji zahtevaju, ali se oni nalaze u različitim datotekama, na raznim medijumima, sa različitim fizičkim organizacijama, zadovoljenje i nekog jednostavnog informacionog zahteva iziskuje značajne programerske napore. Nad strukturom podataka predstavljenom velikim brojem nezavisnih datoteka veoma je teško razviti softverske alate za brz i visoko produktivan razvoj aplikacija i za neposrednu komunikaciju korisnika sa sistemom.

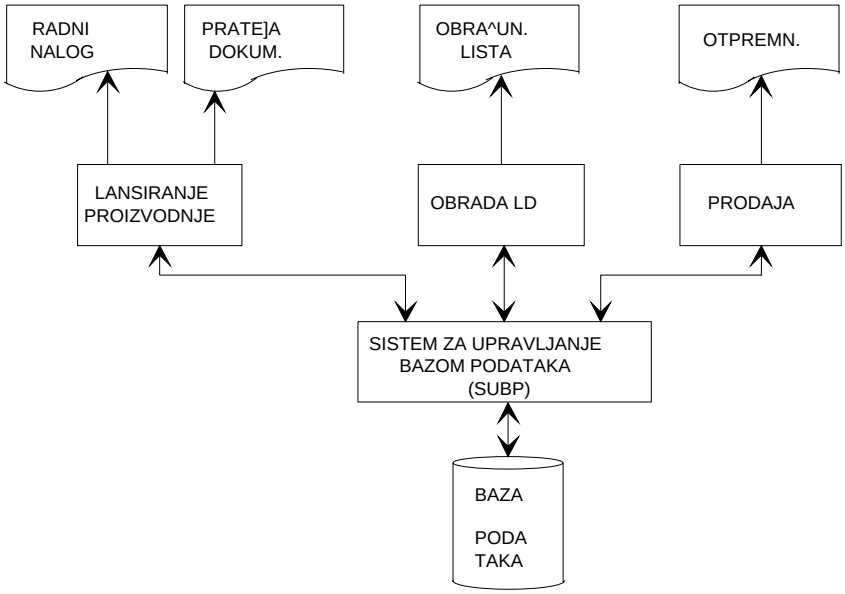
- **Pasivan odnos korisnika**. Ovakav način čuvanja i obrade podataka, ne pruža mogućnost korisniku da sam zadovoljava svoje jednostavnije programske zahteve. Svoje zahteve korisnik može da realizuje samo posredstvom sistem analitičara, projektanta i programera, pa se samim tim, zbog njihove niske produktivnosti, zahtevi korisnika sporo i nepotpuno zadovoljavaju.

Rešavanje ovih osnovnih problema klasične poslovne obrade podataka zahteva bitne tehnološke izmene u ovoj oblasti. Od sredine šezdesetih godina do danas, oblast sistema za upravljanje bazom podataka se izuzetno brzo razvija. Sistem za upravljanje bazom podataka (Slika 3) je jedan složeni softverski sistem koji treba da omogući:

- Skladištenje podataka sa minimumom redundanse.
- Korišćenje zajedničkih podataka od strane svih ovlašćenih korisnika.
- Logičku i fizičku nezavisnost programa od podataka. Bez obzira što se podaci fizički pamte, po pravilu, samo jednom, u jedinstvenoj fizičkoj organizaciji, svaki korisnik dobija svoju sopstvenu logičku sliku (logičku strukturu) podataka kakva njemu najviše odgovara.
- Jednostavno komuniciranje sa bazom podataka preko jezika bliskih korisniku, kako bi se neprofesionalni korisnici neposredno uključili u razvoj informacionog sistema, a profesionalnim programerima značajno povećala produktivnost.

U ovakvoj tehnologiji obrade, podaci su, umesto razbacani po nezavisnim datotekama, organizovani u jedinstvenu bazu podataka. **Baza podataka (BP) je kolekcija medjusobno povezanih podataka, uskladištenih sa minimumom redundanse, koje koriste, zajednički, svi procesi obrade u sistemu.**

Podaci su jedinstveni resurs u nekom sistemu, i njima se mora upravljati na jedinstven način, onako kako se upravlja i drugim vitalnim resursima poslovnih sistema. Svi nedostaci klasične obrade podataka su posledica toga što ova značajna činjenica nije uočavana ili poštovana. Tehnologija baza podataka omogućuje da se sa podacima postupa kao sa jedinstvenim vitalnim resursom nekog sistema.

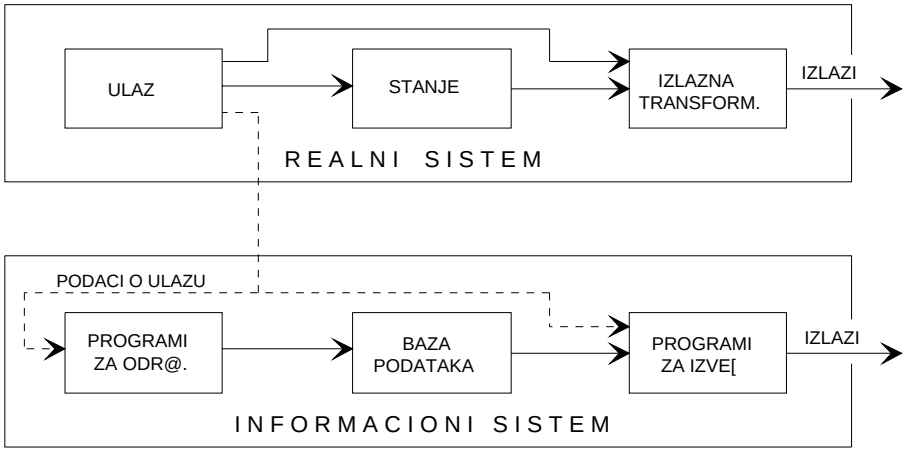


Slika 3. Sistem za upravljanje bazom podataka

2. METODOLOŠKE OSNOVE RAZVOJA INFORMACIONIH SISTEMA

Sistem se, kao što je rečeno, najopštije definiše kao **skup objekata (entiteta) i njihovih medjusobnih veza**. Objekti u sistemu mogu da budu neki fizički objekti, koncepti, dogadjaji i drugo. Objekti se u modelu nekog sistema opisuju preko svojih svojstava (atributa). Dejstvo okoline na sistem opisuje se preko ulaza u sistem, a dejstvo sistema na okolinu preko njegovih izlaza.

Dinamičko ponašanje realnog sistema standardno se šematski predstavlja kao na gornjem delu Slike 4. Ulazi u sistem menjaju stanja sistema. Stanje sistema se definiše kao skup informacija o prošlosti i sadašnjosti sistema koji je potreban da bi se, pod dejstvom budućih poznatih ulaza, mogli odrediti budući izlazi. U stanju sistema skoncentrisana je celokupna istorija realnog sistema. **Očigledno je da stanje sistema opisuje fundamentalne karakateristike sistema, u jednom trenutku vremena ono predstavlja skup objekata sistema, skup njihovih medjusobnih veza i skup vrednosti atributa objekata u tom trenutku vremena**. Izlazna transformacija definiše neki način merenja ili posmatranja dinamičkog ponašanja realnog sistema i daje, na osnovu trenutnog stanja i trenutnih ulaza sistema, njegove izlaze.



Slika 4. Položaj informacionog u odnosu na realni sistem

Ilustrujmo uvedene koncepte posmatrajući neko skladište proizvoda kao sistem. Objekti u ovom sistemu su sami proizvodi, njihovi atributi su na primer, naziv, karakteristike i količina u skladištu. Veza izmedju objekata je, na primer, sastav proizvoda, koji pokazuje od kojih se drugih proizvoda, posmatrani proizvod sastoji. Svako prihvatanje ili izuzimanje proizvoda iz skladišta predstavlja ulaz u sistem (jer menja njegova stanja). Osnovna komponenta stanja u sistemu je količina svakog proizvoda u skladištu. Izlaz iz sistema može da bude količina zaliha svakog proizvoda (samo stanje sistema), zatim ukupna vrednost zaliha, ukupna vrednost zaliha proizvoda odredjene vrste i slično. Izlazna transformacija daje, ovde postupak sračunavanja izlaza na osnovu trenutnog stanja i trenutnih ulaza sistema.

Kao što je to na Slici 4 prikazano, informacioni sistem predstavlja model realnog sistema u kome deluje. Podaci o ulazu (trebovanja, prijemnice, otpremnice i slično, za navedeni primer), preko programa za održavanje, koji modeliraju dejstvo ulaza na sistem, deluju na bazu podataka (kartica zaliha materijala, za navedeni primer), koja modelira stanje sistema. Programi za izveštavanje predstavljaju model izlazne transformacije (postupak sračunavanja ukupne vrednosti zaiha, na primer) i daju zahtevane izlaze.

Osnovu informacionog sistema čini baza podataka, koja se sada može definisati i kao **kolekcija medjusobno povezanih podataka koja modelira (prikazuje) objekte, veze objekata i atribute objekata posmatranog realnog sistema**. Ona zbog toga predstavlja **fundamentalne, stabilne, sporo izmenljive karakteristike sistema**, objekte u sistemu i njihove medjusobne veze. Zato se projekat IS mora bazirati na bazi podataka. Ako je baza podataka dobar model stanja realnog sistema, ako programi za održavanje dobro modeliraju dejstvo ulaza na stanje realnog sistema, onda će se bilo koja informacija potrebna za upravljanje (izlazi), čak i one unapred nepredvidjene, moći dobiti iz IS.

Ako je informacioni sistem model realnog sistema u kome deluje, onda se postupak projektovanja IS svodi na neku vrstu modeliranja realnog sistema, a za to su nam neophodna neka intelektualna sredstva (alati) i to:

(1) **Model podataka** kao intelektualno sredstvo za prikazivanje objekata sistema, njihovih atributa i njihovih medjusobnih veza (statičkih karakteristika sistema) preko logičke strukture baze podataka.

(2) **Model procesa** kao intelektualno sredstvo za opisivanje dinamike sistema, dejstva ulaza na stanje sistema i izlazne transformacije, preko programa nad definisanim modelom podataka.

Ovde ćemo se baviti samo modelima podataka, dok su modeli procesa (Strukturna sistem anliza) prikazani ranije, a ovde se koriste u meri u kojoj je to neophodno za izučavanje metodološkog pristupa modeliranju podataka.

3. MODELI PODATAKA

Model podataka je intelektualno sredstvo za opis statičkih karkatristika sisema, opis karakteristika sistema u nekom stacionarnom stanju. Stacionarno stanje nekog sistema karakteriše se skupom zavisnosti koje postoje izmedju objekata sistema. Ove zavisnosti se, u modelu podataka, mogu predstaviti bilo strukturom podataka, bilo skupom ograničenja na vrednosti podataka. Pored toga, neophodno je definisati i skup operacija modela podataka, da bi se preko njih, u modelima procesa, mogla da opiše i dinamika realnog sistema. Zbog toga svaki model podataka poseduje tri osnovne komponente:

(1) Strukturu modela, odnosno skup koncepata za opis objekata sistema njihovih atributa i njihovih mejdusobnih veza.
(2) Ograničenja - semantička ograničenja na vrednosti podataka koja u svakom stacionarnom stanju moraju biti zadovoljena. Ova ograničenja se obično nazivaju pravilima integriteta modela podataka.
(3) Operacije nad konceptima strukture, pod definisanim ograničenjima, preko kojih je moguće opisati dinamiku sistema u modelima procesa.

Na primer, ako se jezik COBOL tretira kao model podataka, tada:

- Osnovni koncepti strukture modela su rekord kao agregacija polja i grupa polja (uz moguće višestruko ponavljanje polja ili grupa - OCCURS iskaz) i datoteka kao kolekcija rekorda.
- Ograničenja koja je moguće ovde zadati su samo tipovi polja (PICTURE).
- Operacije su, na primer, promena vrednosti polja u rekordu (MOVE A TO POLJE_A OF REKORD_A) ili ubacivanje novog pojavljivanja rekorda u datoteku (WRITE REKORD_A).

3.1. Apstrakcije podataka

Kao što je u materijalu Analiza metodoloških pristupa razvoju IS rečeno, osnovni problem u razvoju IS je savladavanje složenosti u opisivanju (modeliranju) IS. Opšti metodološki pristup opisivanju složenih sistema je apstrakcija. Apstrakcija je **kontrolisano uključivanje detalja, "sakrivanje" detalja, odnosno "izvlačenje" opštih karakteristika** u opisivanju nekog sistema. Postupak inverzan apstrakciji nazivamo detaljisanje. Koristeći se različitim nivoima apstrakcije, neki složeni sistem se može istovremeno i jasno i detaljno opisati: na višim nivoima jasno, na nižim detaljno, postepenim i kontrolisanim uključivanjem detalja.

I u modeliranju podataka i u modeliranju procesa koristi se princip apstrakcije. U modeliranju podataka definišu se različite apstrakcije podataka, a u modeliranju procesa tzv. proceduralne apstrakcije.

Apstrakcije podataka će biti detaljno diskutovane kasnije, a ovde se, u svrhu podele modela podataka samo ukratko definišu.

(1) **Klasifikacija (tipizacija) i uzorkovanje**. Klasifikacija ili tipizacija je apstrakcija u kojoj se skup sličnih objekata pretstavlja jednom klasom objekata, odnosno svaki objekat iz posmatranog skupa odgovarajućim tipom objekta. Na primer, skup {Ana, Zoran, Goran, Mira} se pretstavljaja klasom STUDENT, odnosno svaki objekat iz toga skupa tipom objekta Student. Slični objekti su oni objekti koji imaju iste atribute (svojstva), koji mogu da stupe u iste veze sa drugim objektima u sitemu i na koje se mogu primeniti iste operacije. Postupak detaljisanja za ovu apstrakciju se naziva uzorkovanje, i pretstavlja prikazivanje jednog **pojavljivanja (uzorka, primerka)** datog tipa ili klase. I sam atribut nekog objekta može se tretirati, u najopštijem smislu, kao objekat. Na taj način se očigledno može definisati tip atributa (na primer BOJA) i pojavljivanje atributa (na primer Plavo, Crveno, Zeleno). Skup svih mogućih pojavljivanja jednog tipa atributa se naziva domen atributa.

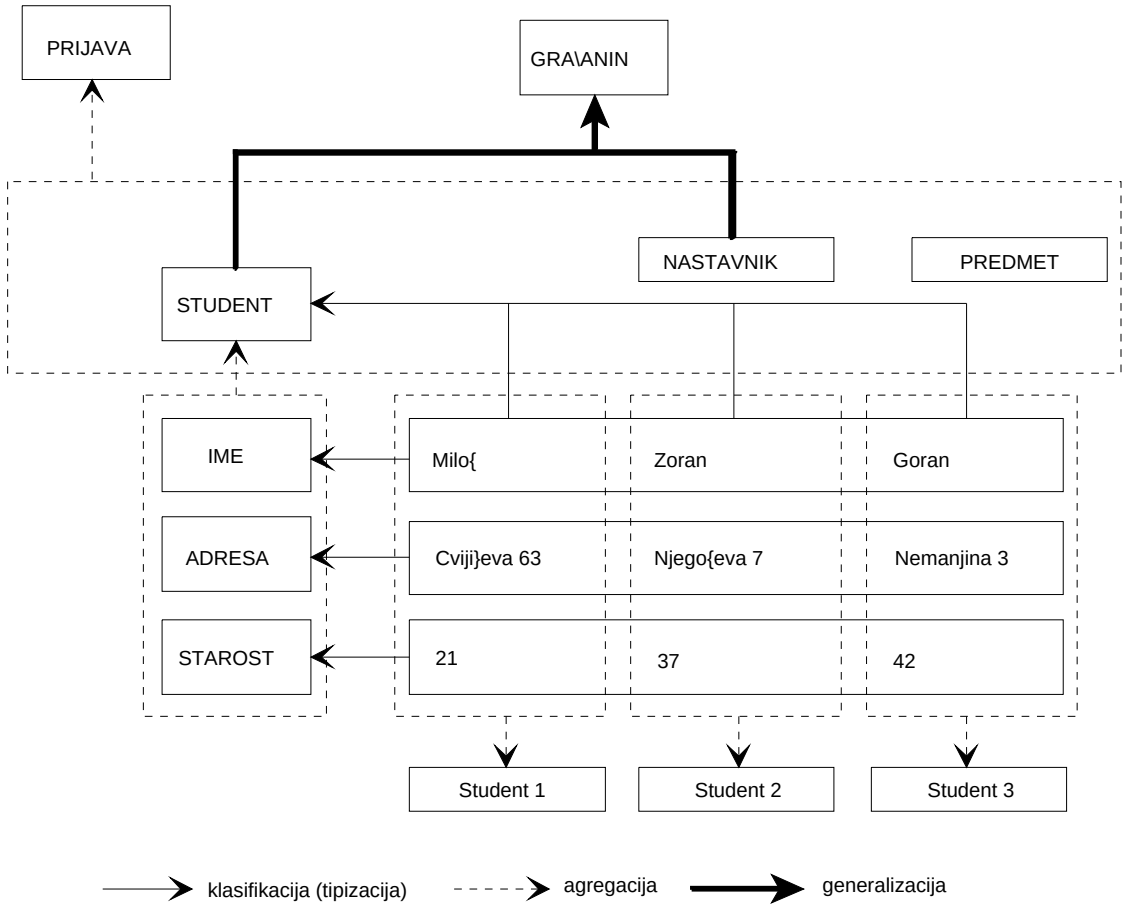
(2) **Generalizacija i specijalizacij**a. Generalizacija je apstrakcija u kojoj se skup sličnih tipova objekata pretstavlja opštijim generičkim tipom (nadtipom). Pod sličnim tipovima objekata ovde se mogu tretirati tipovi objekata koji imaju jedan broj istih (zajedničkih) atributa, tipova veza sa drugim objektima i operacija. Na primer, skup tipova objekata {Student, Radnik, Dete, Penzioner} mogu se generalizovati u tip objekta Gradjanin. Inverzni postupak generalizaciji je specijalizacija. Tip objekta Gradjanin se specijalizuje u podtipove Student, Radnik, Dete i Penzioner.

(3) **Agregacija i dekompozicija**. Agregacija je apstrakcija u kojoj se skup tipova objekata i njihovih veza tretira kao jedinstveni agregirani tip objekta. Na primer, tipovi objekata Student, Predmet i Nastavnik se agregiraju u objekat Prijava čiji su atributi DATUM_POL i OCENA. Postupak inverzan agregaciji se naziva dekompozicija. Sam objekat u sistemu može se tretirati kao najniži nivo ove apstrakcije, kao agregacija njegovih atributa.

Na slici 5. na jednom primeru su ilustrovane sve uvedene apstrakcije podataka. Skupovi vrednosti {Miloš, Zoran, Goran}, {Cvijićeva 63, Njegoševa 7, Nemanjina 3} i {21, 37, 42} klasifikuju se u tipove atributa IME, ADRESA i STAROST, respektivno. Pojavljivanja (vrednosti) atributa <Miloš, Cvijićeva 63, 21> , <Zoran, Njegoševa 63, 37> i <Goran, Nemanjina 3, 42> agregiraju se u pojavljivanja objekta 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 atributa <IME, ADRESA, STAROST> . 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.

Apstrakcije podataka omogućuju da se i jasno i detaljno opišu složeni objekti nekog sistema i da se njihovim medjusobnim povezivanjem formira struktura modela u koju je u najvećoj mogućoj meri ugradjena semantika realnog sistema (znanje o realnom sistemu). Preostala semantika realnog sistema implementira se preko modela procesa. Očigledno je da modeli podataka treba da podrže sve apstrakcije podataka na svim nivoima apstrakcije, jer će na taj način smanjiti količinu znanja o realnom sistemu koju treba implementirati preko modela procesa i time značajno olakšati razvoj IS u svim njegovim fazama, a posebno u fazi implementacije (programiranja).



Slika 5. Klasifikacija, genrealizacija i agregacija

3.2. Generacije modela podataka

Na osnovu definicije, očigledno je da svaki model podataka treba da zadovolji dva bitna kriterijuma:

- Da poseduje koncepte pogodne za modeliranje realnih sistema.
- Da se njegovi koncepti, struktura, ograničenja i operacije mogu jednostanvo implementirati na računaru.

Na osnovu toga kako zadovoljavaju ova dva kriterijuma modeli podataka se mogu klasifikovati u sledeće "generacije":

Prvu generaciju čine konvencionalni programski jezici (jezici treće generacije), koji se takodje mogu tretirati kao modeli podataka. Apstrakcija podataka u njima su tipovi podataka kojima raspolažu.

Apstrakcija klasifikacije se koristi samo pri definisanju prostih tipova (INTEGER, REAL,

CHARACTER i slično) i datoteka. Jedine agregacije koje se koriste su rekordi (kao agregacije polja i drugih rekorda ili grupa i vektora) i vektori (array) kao agregacije istovrsnih elemenata. Samo u nekim jezicima se apstrakcija generalizacije može delimično primeniti preko koncepta "promenljivog" (CASE) rekorda. Agregirani tipovi u njima, prvenstveno rekordi, se ne mogu eksplicitno povezivati, pa baza podataka koja odgovara ovakvim modelima pretstavlja skup nepovezanih datoteka. Operacije nad objektima višeg nivoa apstrakcije su relativno siromašne i moraju se realizovati preko procedura koje se formiraju preko operacija nad njihovim komponentama. Ograničenja nad vrednostima pojedinih tipova nije moguće eksplicitno zadati. Zbog svega ovoga oni nisu dovoljno pogodni za modeliranje realnog sistema (zato je programiranje u njima teško), ali se model realnog sistema opisan pomoću njih može direktno implementirati na računaru.

Drugu generaciju čine tri klasična modela baze podataka, hijerarhijski, mrežni i relacioni model. Oni u osnovi koriste iste apstrakcije podataka kao i modeli prve generacije. Pored toga u njima je moguće eksplicitno definisati specifične način povezivanja rekorda, odnosno bazu podataka kao skup medjusobno povezanih podataka, znatno moćnije (makro) operacije, a ponekad i eksplicitno definisati specifične vrste ograničenja na vrednosti podataka. Medjutim i ovde nije u potpunosti podržan koncept agregacije, a pogotovo generalizacije. Iako semantički bogatiji od modela prve generacije, modeli druge generacije nisu dovoljno pogodni za opisivanje složenih sistema, sistema u kojima se prirodno definišu složeni objekti koji se ne mogu lako pretstaviti konceptom rekorda. Postoje komercijalno raspoloživi softveri, SUBP, za njihovu direktnu implementaciju na računaru.

Treću generaciju čine **tzv. semantički bogati modeli podataka i objektni modeli podataka** (Model entiteti-veze, Semantic Data Model(SDM), Prošireni relacioni model, Semantičke mreže i različiti objektno orjentisani modeli podataka). Oni u mnogo većoj meri podržavaju sve vrste apstrakcija, poseduju mogućnost eksplicitnog iskazivanja ograničenja na vrednosti podataka i moćne operacije nad objektima visokog nivoa apstrakcije. Medjutim, za njih još uvek ne postoje komercijalni softveri preko kojih bi se direktno mogli implementirati na računaru.

3.3. Objektno-orjentisani transformacioni pristup razvoju IS

Imajući u vidu karakteristike generacije modela podatka, zadovoljavajući metodološki pristup projektovanju informacionih sistema bi mogao da bude:

(1) Koristeći neki model Treće generacije, formirati semantički bogat model realnog sistema koji se analizira.
(2) Prevesti dobijeni model u neki od modela Prve ili Druge generacije, u zavisnoti od softvera kojim se raspolaže i drugih činioca i na taj način implementirati bazu podataka na računaru. Postupak prevodjenja sa semantički bogatijeg, na semantički siromašniji model može se uvek formalizovati, pa samim tim i automatizovati.

Ovakav metodološki pristup je ekvivalentan pristupu razvoju softvera preko apstraktnih tipova podataka. Prvi korak, formiranje semantički bogatog modela podataka ralnog sistema, pretstavlja specifikaciju IS preko apstraktnih tipova podataka. U drugom koraku se IS implementira preko jezičkih (virtuelnih) tipova podataka nekog od modela podataka druge ili prve generacije. S obzirom da apstraktni tipovi podataka modeliraju objekte sistema bilo koje složenosti i da je transformacija modela treće u model druge ili prve generacije u najvećoj meri formalna i automatizovana, pristup koji se predlaže ima sve karakteristike objektno-orjentisanog transformacionog pristupa razvoju informacionih sistema.

U metodologiji razvoja IS koja se ovde predlaže, kao model Treće generacije koristi se Prošireni model objekti-veze, najpopularniji semantički model podataka. Pored već standardnih proširenja u odnosu na orginalnu Chen-ovu verziju (uvodjenje generalizacije i agregacije), preciznijih definicija nekih koncepata strukture modela, ovde se daje i jezik za definisanje ograničenja bilo koje vrste i definišu operacije modela objekti veze, čime se omogućuje da se IS u potpunosti specificira pomoću modela objekti veze.

4. MODEL OBJEKTI-VEZE

Model objekti-veze je najpopularniji i u praksi najviše korišćeni semantički model podatka (model podataka treće generacije). Postoji više različitih verzija ovog modela. Ovde se izlaže jedna specifična verzija ovog modela, Prošireni model objekti-veze (PMOV) u kome se definišu i jezik za specifikaciju ograničenja i operacije modela, čime se dobija alat za formalnu specifikaciju IS.

Kao što je ranije rečeno, model podataka predstavlja intelektualno sredstvo pomoću koga se prikazuje kako su podaci o nekom ralnom sistemu medjusobno povezani. Model podataka obezbedjuje interpretaciju podataka o posmatranom realnom sistemu. Interpretacija podataka se u nekom modelu podataka ostvaruje kroz tri njegove osnovne komponente:

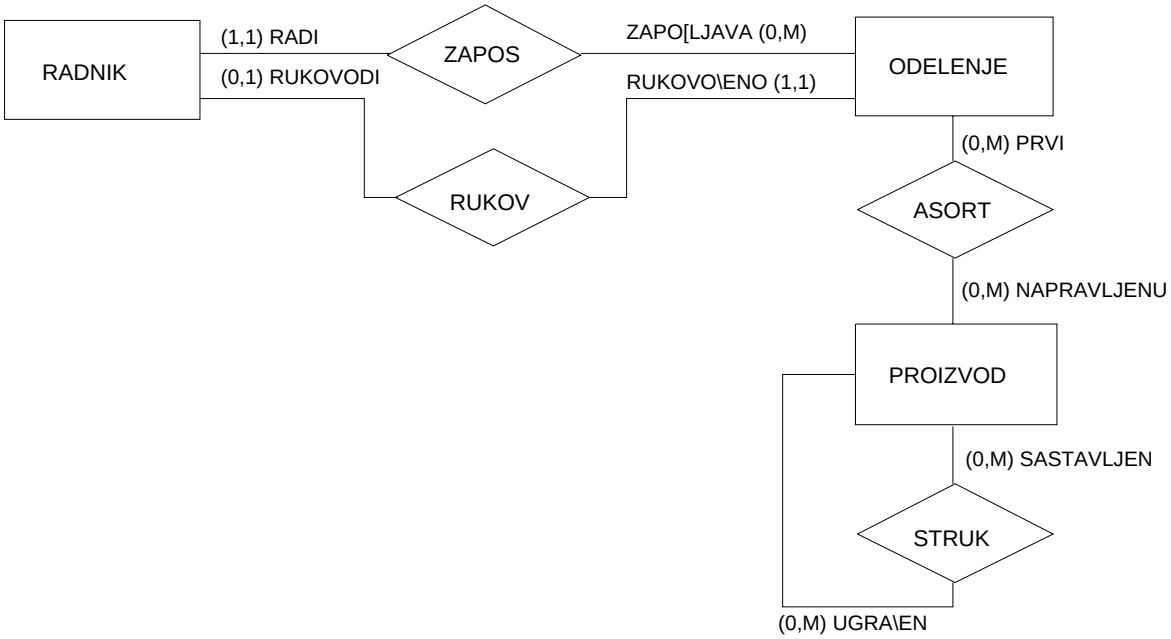
(1) **Strukturu podataka**, preko koje se predstavljaju statičke karkateristike sistema;

(2) **Ograničenja** - logička ograničenja na vrednosti podatka koja u svakom trenutku posmatranja (stacionarnom stanju) treba da budu zadovoljena. Ova dodatna ograničenja na podatke koja nisu obuhvaćena samom strukturom nazivaju se i vrednosnim pravilima integriteta modela podataka.

(3) **Operacije** nad konceptima strukture modela, preko kojih je moguće opisati dinamiku sistema, odnosno dati interpretaciju podataka kroz obradu podataka u modelu podataka.

4.1. Struktura Proširenog modela objekti-veze

Struktura PMOV predstavlja se **dijagramima objekti-veze** (DOV). Simbole na ovim dijagramima uvodićemo postepeno sa uvodjenjem pojedinih koncepata modela. Na slikama 6 i 7 predstavljeni su osnovni koncepti modela.



Slika 6. Osnovni koncepti modela objekti-veze

4.1.1. Objekti i veze

U modelu PMOV sistem se opisuje kao skup objekata i njihovih veza. Objekat u modelu može da predstavlja neki fizički objekat realnog sistema (konkretan proizvod, konkretnog radnika (Jovan Jovanović), vremenski trenutak ili period i slično), ili neki koncept (klasa daktilografije, smer studija i slično).

Koristeći apstrakciju klasifikacije, pojedinačni objekati u sistemu se klasifikuju u tipove objekata. Tip objekta je opšti predstavnik neke klase, a svaki pojedinačni objekat predstavlja jedno pojavljivanje (primerak) datog tipa. Klasa objekata je skup pojavljivanja objekata datog tipa. Na primer, tip objekata je Radnik, pojavljivanja toga tipa su Jovan Jovanović, Petar Petrović i drugi, a klasa objekata je RADNIK i predstavlja skup pojavljivanja tipa Radnik. U modelima podataka često se ne pravi jasna razlika izmedju tipa i klase objekata. Ako posmatramo neki skup sličnih objekata tip objekta pretstavlja intenziju tog skupa (definiciju skupa preko iskazivanja zajedničkih karakteristika elemenata skupa), a klasa objekata predstavlja istovremeno i intenziju i ekstenziju toga skupa. (Ekstenzija skupa je definisanje skupa navodjenjem svih njegovih članova). Ako bi tipove i klase objekata pretstavljali preko tipova podataka u nekom jeziku, definisali bi dva tipa podatka, objekat i klasu i za navedeni primer to iskazali na sledeći način:

tip objekta - Radnik: **objekat**
klasa objekta -RADNIK: **sef_of** Radnik

Na dijagramima objekti veze (DOV) klase objekata se prikazuju pravougaonicima. Svaka klasa definiše istovremeno i tip objekta. U daljem tekstu, nazivi klasa objekata će se pisati velikim, a nazivi tipova objekata malim slovima.

Veze u modelu opisuju način povezivanja (uzajamna dejstva) objekata. Apstrakcija klasifikacije može se primeniti i na veze i definisati pojmovi tipa, klase i pojavljivanja veze. Tako je, na primer, par <Jovan Jovanović, Računski centar> jedno pojavljivanje tipa veze Zapos u modelu na slici 6. Klase veza se na DOV predstavljaju sa rombovima. U PMOV se pretpostavlja korišćenje samo binarnih veza, veza izmedju dva tipa objekata. Svaki tip veze izmedju dva tipa objekata e1 i e2 definiše dva tipa preslikavanja, preslikavanje sa skupa pojavljivanja (klase) E1 u skup pojavljivanje E2 (E1 je domen, a E2 kodomen preslikavanja) i (inverzno) preslikavanje sa skupa E2 u skup E1. Na primer, u vezi ZAPOS, RADI odgovara preslikavanju RADNIK -----> ODELENJE, ("radnik radi u odelenju"), a ZAPOŠLJAVA odgovara preslikavanju ODELENJE -----> RADNIK. ("odeljenje zapošljava radnike"). Preslikavanja definišu "uloge" objekata u vezi. U tipu veze Zapos, Radnik ima "ulogu" Radi, a Odelenje ima "ulogu" Zapošljava.

Veze se mogu uspostavljati i izmedju pojavljivanja objekata istog tipa (nad istom klasom objekata). Na primer veza STRUK sa parom preslikavanja <SASTAVLJEN, UGRADJEN> nad klasom objekata PROIZVOD. Preslikavanje SASTAVLJEN je preslikavanje koje za jedan nadredjeni proizvod, daje skup njemu podredjenih proizvoda, njegovih sastavnih delova, a preslikavanje UGRADJEN je preslikavanje koje za jedan podredjeni proizvod (sastavni deo), daje skup nadredjenih u koje je on ugradjen. Drugim rečima, u vezi STRUK, tip objekta Proizvod ima dvostruku ulogu, jednom se tretira kao nadredjen (Sastavljen), a drugi put kao podredjen (Ugradjen) proizvod u odgovarajućoj strukturi (sastavnici).

SASTAVLJEN: PROIZVOD (nadredjen) ------> PROIZVOD (podredjen);
UGRADJEN: PROIZVOD (podredjen) -------> PROIZVOD (nadredjen).

Isto tako moguće je uspostaviti više različitih tipova veza izmedju istih tipova objekata (veze ZAPOS i RUKOV izmedju klasa objekata RADNIK i ODELJENJE).

Očigledno je da dva preslikavanja definišu jednu binarnu vezu, pa je koncept veze izvedeni pojam, što ne znači i da je potpuno redudantan, odnosno da bi ga trebalo izostaviti. Nazivom veze se uspostavlja odnos izmedju dva medjusobno inverzna preslikavanja. Veza pretstavlja agregaciju dva

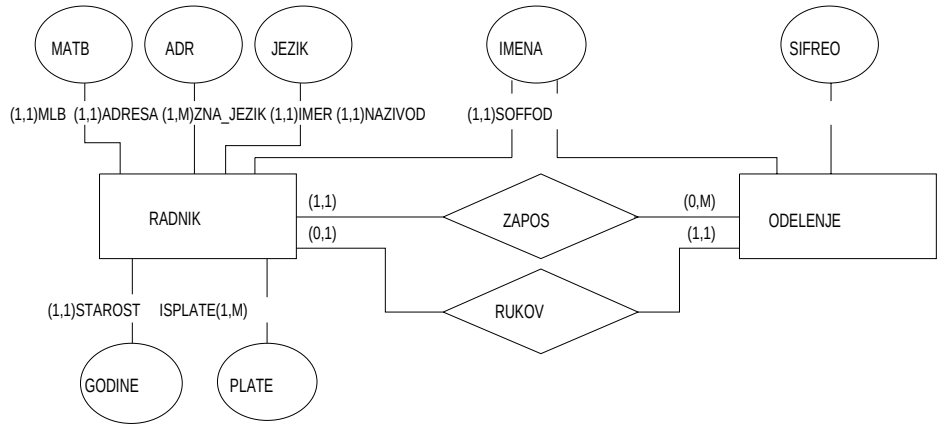
objekta, i može se, kao što će to biti kasnije pokazano, tretirati kao poseban agregirani objekat. Koncept veze je "prirodniji", pa se lakše prihvata. Takodje, iz prkatičnih razloga, da bi se broj naziva na DOV smanjio zadržava se koncept veze i usvaja sledeća konvencija:

- Nazivi veza se uvek zadaju.
- Nazivi preslikavanja se obavezno zadaju u rekurzivnim vezama (binarnim vezama nad jednom klasom objekata).
- Nazivi preslikavanja u ostalim vezama su opcioni i ako nisu zadti dobijaju po "difoltu", bilo ime veze, bilo ime objekta kodomena preslikavanja.

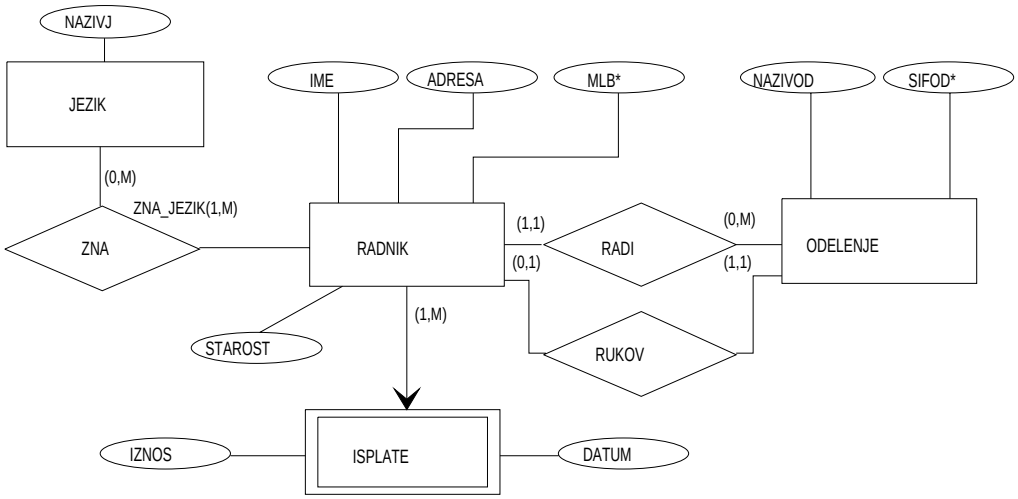
Jedna od bitnih karakteristika veza izmedju objekata je **kardinalnost preslikavanja** koja je čine. Kardinalnost preslikavanja E1 --> E2 definiše se parom (DG,GG), gde DG (donja granica) daje najmanji mogući, a GG (gornja granica) najveći mogući broj pojavljivanja tipa objekata E2, za jedno pojavljivanje tipa objekata E1. DG može imati vrednost 0, 1 ili neki poznat ceo broj > 1. GG može imati vrednost 1, neki poznat ceo broj > 1, ili nepoznat ceo broj > 1 koji se označava sa M. Očigledno je da u jednom preslivanju mora biti zadovoljeno DG GG. Naprimer, u preslikavnju RADI jednom pojavljivanju objekta RADNIK odgovara najmanje jedno i najviše jedno pojavljivanje objekta ODELENJE (DG =1, GG = 1) - jedan radnik radi u jednom odelenju i svaki radnik radi u jednom odelenju. Ili, u preslikavanju PROIZVODI jednom pojavljivanju objekta ODELENJE odgovara nula, jedno ili više pojavljivanja objekta PROIZVOD (DG = 0, GG = M). Bez obzira da li se naziv preslikavanja izostavlja ili ne, kardinalnost preslikavanja se uvek mora definisati (sem u slučajevima trivijalnih preslikavanja, o kojima će kasnije biti reči).

4.1.2. Atribut i domen

Objekti u sistemu se opisuju preko svojih svojstava, odnosno atributa. Na primer (Slika 7a) atributi objekata RADNIK su MATIČNI LIČNI BROJ (MLB), IME, STAROST. Svaki atribut u jednom trenutku vremena ima neku vrednost. Atributi uzimaju vrednost iz skupa mogućih vrednosti. Ovi skupovi se nazivaju domenima.



a) Definicija atributa



b) Konvencija za predstavljanje atributa

Slika 7. Definicija konvencije za predstavljanje atributa

Domeni mogu biti:

- "predefinisani", odnosno standardni programsko-jezički domeni, kao što su INTEGER, CHARACTER, REAL, LOGICAL i DATE.

- "semantički", kada se definišu posebno, preko svoga imena, predefinisanog domena i, eventualno, ograničnenja na mogući skup vrednosti predefinisanog domena. Na primer, semantčki domen SEMESTRI se definiše kao

  SEMSESTRI DEFINED AS INTEGER (2) BETWEEN 1, 10

Ključna reč DEFINED_AS vezuje imenovani i predefinisani domen, a iskaz BETWEEN 1,10 ograničava vrednosti semestara na ovaj opseg prirodnih brojeva. Opšta sintaksa za iskazivanje ograničenja na predefinisani skup vrednosti za semantički domen je:

definicija_semantičkog__domena :
naziv_domena 'DEFINED AS' predefinisani_domen [ograničenje]

(Kao što je uobičajeno, uglasta zagrada znači opicioni deo iskaza.)

O definiciji predefinisanih domena i drugih ograničenja, osim ovog navedenog u primeru, govoriće se kasnije.

Činjenica da atribut uzima vrednost iz nekog domena označava se na sledeći način:

naziv polja : domen [ograničenje]

Na primer:

BI: CARACTER(7)
SEMESTAR: SEMESTRI
OCENA: INT(2) IN (5,6,7,8,9,10)
SEMESTVŠ: SEMESTRI IN (1,2,3,4) *Semestar više škole*

(Tekst koji se daje između zvezdica je objašnjenje sintakse.)

Osnovni razlog za uvođenje semantičkih domena je jasno iskazivanje semantičke sličnost dva atributa. Naime dva atributa su semantički slična samo ako su definisana nad istim domenom. Drugim rečima, semantički domeni uspostavljaju razliku između pojedinih istovrsnih predefinisanih domena iste veličine koji nemaju semantičku sličnost. Na primer, ako bi se i atribut SEMESTAR definisalo direktno nad predefinisanim domenom integer, kao

SEMESTAR: INTEGER (2) BETWEEN 1,10

tada bi atributi OCENA i SEMESTAR bili semantiki slični, mogli bi se povezivati operatorima definisanim nad predefinisanim domenom INTEGER, na primer, moglo bi se pisati

IF SEMESTAR > OCENA THEN ....

što očigledno nema smisla. Međutim, ako je atribut SEMESTAR definisan nad semantičkim domenom SEMESTRI, a atribut OCENA nad predefinisanim domenom INTEGER, prethodni izraz bi bio i sintaksno nekorektan. Različiti atributi se mogu povezati nekim operatorom samo ako su definisani nad istim domenom i ako je operator definisan u tom domenu.

Predefinisani domeni su standardni progrmsko-jezički domeni, koji se ovde definišu na sledeći način:

INTEGER(dužina)
CHARACTER(dužina)
REAL(dužina celokupnog broja, dužina iza zareza)
LOGICAL
DATE

Pored ograničenja na vrednosti atributa, odnosno vrednosti domena koja su data u primerima definišu se i druga. Ograničenja mogu biti prosta i složena. Lista dozvoljenih prostih ograničenja je:

(a) □ konstanta, gde je □ bilo koji operator poređenja koji se na datom domenu može definisati (na primer, <, □, >, □, = □ za brojne domene), a konstanta je neka definisana vrednost iz datog domena. Na primer,

STAROST: INI(2) < 65

(b) BETWEEN konstanta, konstanta, gde su konstante vrednosti iz datog domena. Na primer,

SEMESTAR: INTEGER (2) BETWEEN 1,10

(c) IN (lista vrednosti), gde se lista formira od konstanti iz odgovarajućeg domena. Na primer,

OCENA INT(2) IN (5,6,7,8,9,10)

(d) NOT NULL, kada dato polje ne može da dobije "nulla vrednost", odnosno mora uvek da ima vrednost. Na primer,

BROJ_INDEKSA: CHARACTER (7) NOT NULL

Predpostavlja se da svaki domen uključuje u sebe i "nula vrednost", specijalnu vrednost koja se dodeljuje nekom atributu objekta kada se njegova vrednost ne poznaje (ili, preciznije, još uvek ne poznaje). Na primer, ako ne znamo starost nekog radnika vrednost atributa STAROST za tog radnika će biti nula vrednost. Da bismo iskazali da neki atribut ne može da dobije nula vrednost (mora da ima vrednost za svako pojavljivanje objekta u klasi) uvodi se gornji specifičan iskaz.

Složena ograničenja se formiraju od prostih ili drugih složenih ograničenja vezujići ih logičkim operatorima AND, OR i NOT. Na primer,

STAROST: INI(2) < 65 AND NOT NULL

Bilo prosto, bilo složena ograničenje se može imenovati, odnosno posebno definisti kao Bullova (logička funkcija) i samo ime navesti kao ograničenje. Na primer, pretpostavimo da atribut ŠIFRA_PR (šifra_proizvoda) ima kontrolu "po modulu 11" Tada bi se mogla definisati funkcija MODUO_11 koja dobija vrednost TRUE ako je pomenuta kontrola zadvovoljena, a FALSE ako nije. Tada se ograničenje defiše kao

ŠIFRA_PR INT(13) MODUO_11

Složeni (komponovani) domeni. U nekim slučajevima i sam domen ima svoju strukturu i pretstavlja se kao agregacija drugih prostih ili komponovanih domena. Na Slici 7 prikazan je komponovani domen PLATE koji pretstavlja agregaciju domena <DATUM, IZNOS>. U čestim slučajevima kada atribut pretstavlja neku klasifikacionu ("govoreću") šifru, odgovarajući domen ima tačno definisanu strukturu. Da bi se prikazala ova struktura i nad pojedinim delovima domena definisala ograničenja uvedena je funkcija SUBSTRING koja ima za cilj da izvuče deo domena i prikaže neke karakteristike toga dela. Funkcija SUBSTRING se definiše kao

SUBSTRING(položaj prvog člana, položaj poslednjeg člana)

Na primer,

MLBR CHARACTER(13) SUBSTRING(1,2) < 31 AND SUBSTRING(3,4) < 12 AND
SUBSTRING(5,7) BETWEEN 000, 999 AND 000.

(Prva dva karaktera predstavljaju datum u mesecu, druga dva mesec u godini itd.)

Konverzija domena. S obzirom da domeni u PMOV imaju karakter semantičkih domena, poredjenje vrednosti iz različitih domena nije moguće i tretira se kao greška. Medjutim, ponekad se domeni razlikuju samo po tome što su njihove vrednosti date u drugim jedinicama iste "semantičke veličine". Na primer jedan u metrima, a drugi u kilometrima, a oba prestavljaju semantički istu veličinu, dužinu, ili jedna ili druga forma pretstavljanje datuma. Da bi se omogućilo poredjenje različitih domena (kada se to želi) neophodno je definisati pravila (procedure) konverzije jednog u drugi domen.

Formalno se atribut objekta može definisati kao preslikavanje iz skupa objekata datog tipa (klase objekata) u skup vrednosti (domen). Na primer,

```
                    (1,1)
MLB: RADNIK   ------------->  MATB
                    (1,1)
IMER:RADNIK   --------------> IMENA
```

Ako je kardinalnost atributa (DG = 1, GG = 1) onda se takav atribut naziva jednoznačni atribut objekata. Ako i inverzno preslikavanje jednoznačnog atributa (preslikavanje DOMEN ----> OBJEKAT) takodje ima kardinalnost (DG = 1, GG= 1) tada se takav atribut naziva identifikator objekta, jer jedno pojavljivanje takvog atributa jedinstveno odredjuje jedno pojavljivanje objekta u skupu pojavljivanja objekata datog tipa. Na primer, atribut MLB je identifikator objekata RADNIK.

Ako je gornja granica kardinalnost atributa GG = M, onda se takav atribut naziva višeznačni atribut objekta. Na primer,

```
                                  (1,M)
ZNA-JEZIK: RADNIK ------------> JEZIK
```

Iz praktičnih razloga, zbog jednostavnije transformacije PMOV u implementacioni model, jednostavnijeg definisanja njegovih ograničenja i operacija u PMOV se ne koriste višeznačni atributi. Semantika realnog sistema predstavljena višeznačnim atributom, obuhvata se na jedan od sledeća dva načina:

(1) Ako domen višeznačnog atributa ima unapred zadat, semantički značajan skup vrednosti, tad se on modelira kao klasa objekata, a višeznačni atribut se pretstavlja kao preslikavnje u novodefinisanoj vezi posmatranog objekta sa ovim novim objektom (Slika 7b ilustruje ovakvu transformaciju višeznačnog atributa ZNA-JEZIK).

(2) Ako domen višeznačnog atributa nema unapred zadat semantički značajan skup vrednosti, tada ga je pogodno pretstaviti preko novog koncepta identifikaciono zavisnog slabog objekta. Pojavljivanja identifikaciono zavisnog slabog objekta nemaju sama za sebe nikakvo značenje, ne predstavljaju (identifikuju) ni jedan objekat od interesa u posmatranom realnom sistemu, već dobijaju značenje tek kada se povežu sa nekim drugim (nadredjenim) objektom. Na Slici 7b dat je primer transformacije višeznačnog atributa ISPLATE koji je bio definisan nad složenim domenom PLATE sa značenjem komponenti <datum, iznos> . Sam za sebe objekat sa atributima DATUM I IZNOS nema semantički značaj (neko pojavljivanje ovog objekta ne nosi nikakvu informaciju-čija plata?), pa se zbog toga pretstavlja slabim objektom ISPLATE. **Slabi objekat u sistemu ne može da postoji (egzistencijalno je zavisan) i njegova pojavljivanja ne mogu da se identifikuju (identifikaciono je zavisan) od njemu nadredjenog objekta**. Identifikaciona i egzistencijalna zavisnost znače da neki slab objekat ne može postojati u bazi podataka ako konkretno pojavljivanje objekata koje ga identifikuje takodje nije u bazi. Drugim rečima, ako se neko konkretno pojavljivanje "nadredjenog" objekta izbaci iz baze, automatski se izbacuju i sva od njega identifikaciono zavisna pojavljivanja slabog objekata. Na primer, ako se u modelu na slici 7b izbaci neko pojavljivanje objekta RADNIK, automatski se izbacuju i sva odgovarajuća pojavljivanja objekta ISPLATE. Postoji i drugačija egzistencijalna zavisnost objekata koja označena binarnom vezom sa barem jednim "totalnim" preslikavanjem (preslikavanjem sa kardinalnošću DG = 1). Na primer, preslikavanje RADI u vezi ZAPOS izmedju RADNIKA i ODELENJA, definiše egzistencijalnu zavisnost RADNIKA od ODELENJA u smislu da svaki radnik mora da radi u jednom odelenju. Medjutim, izbacivanje nekog odelenja iz baze podataka ne mora da znači da će se svi radnici toga odelenja automatski izbaciti iz baze, oni mogu biti premešteni u drugo odeljenje. (Tako nešto očigledno nije imalo smisla uraditi sa isplatama za jednog radnika).

Iz definicije slabog objekta očigledno je kardinalnost preslikavanja SLABI -----> NADREDJENI uvek DG = 1 i GG = 1, pa se ne navodi, dok se kardinalnost inverznog preslikavanja NADREDJENI -----> SLABI mora specificirati.

Gore izvedena diskusija i druge specifične konvencije koje se koriste u PMOV ovde se ukratko rekapituliraju:

(1) U PMOV se ne koriste višeznačni atributi, već se odgovarajuća svojstva objekata pretstavljaju bilo kao preslikavanja posmatranog objekta prema novodefinisanom objektu koji pretstavlja domen višeznačnog atributa ili kao "slabi" objekti.

(2) Svi atributi moraju da budu primenljiva svojstva na sve objekte u odgovarajućoj klasi. Zbog toga je donja granica preslikavanja KLASA_OBEKATA ----> DOMEN uvek DG = 1. Kako se ne koriste višeznačni atributi, to je za ovo preslikavanje i GG = 1, pa se kardinalnosti atributa ne moraju predstavljati na DOV.

(3) Atributi identifikatori objekata mogu posebno označiti (na primer sa zvezdicom, ili podvlačenjem).

(4) Posebno je interesantan problem semantičkih domena, odnosno potrebe da dva ili više atributa uzimaju vrednosti iz istog semantičkog domena. Ovim se, kao što je rečeno, iskazuje "semantička sličnost", odnosno neka vrsta odnosa između tih atributa. Po pravilu, u MOV takav odnos treba eksplicitno prikazati, preko veze objekata kojima ti atributi pripadaju. (Napomenimo da se postavlja i zahtev da atributi jednog objekta, osim činjenice da kao agegacija čine taj objekat i činjenice da ih identifikator funkcionalno određuje, ne mogu imati nikakvu drugu međusobnu vezu. Svaka druga veza ukazivala bi na složeniju strukturu objekta, na činjenicu da on poseduje neke semantički značajne komponente, pa bi te komponente i njihove međusobne veze trebalo eksplicitno prikazati. Analogno relacionoj terminologiji, ovaj zahtev se postavlja da bi sam objekat bio "potpuno normalizovan") Zbog toga se uvodi konvencija da se na MOV ne prikazije mogućnost definisanja više atributa nad istim domenom.

Imajući u vidu ove konvencije, pogodno je na dijagramu objekti-veze (DOV), da bi se on pojednostavio, prikazivati samo atribute, odnosno izostaviti prikazivanje domena i kardinalnosti preslikavanja, a definiciju domena i vezu atributa i domena prikazati posebno.

Na Slici 7b prikazan je model sa slike 7a uz korišćenje navedenih konvencija. Sintaksa za definisanje domena data je ranije, a praktično je njednostavnije prikazati odnos atributa i domena prko sledeće tabele:

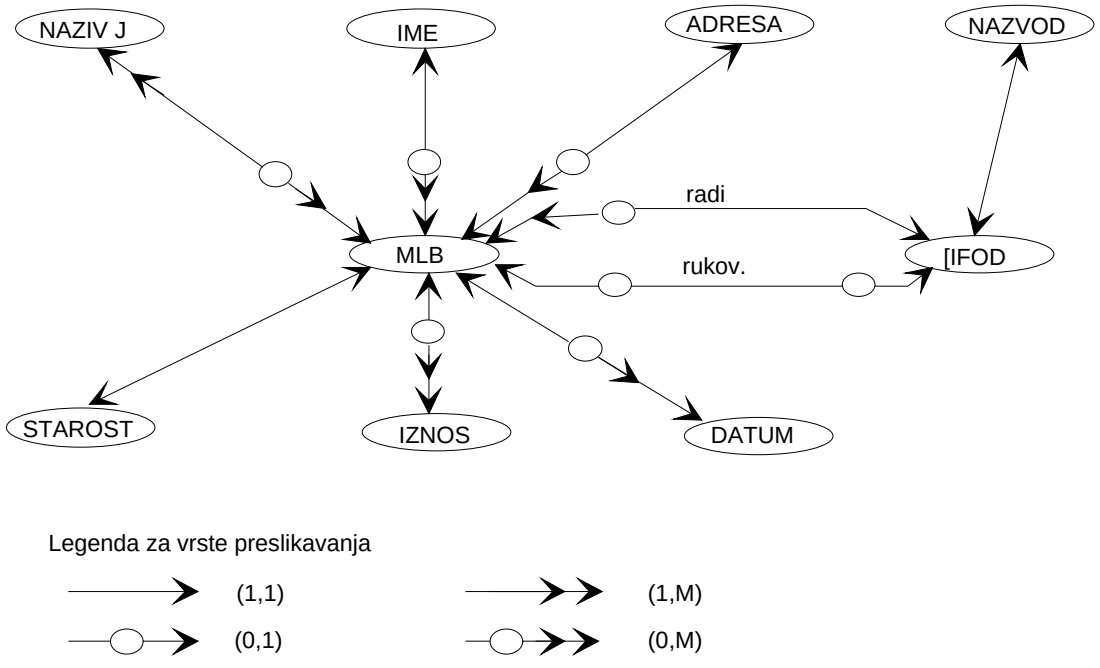
ATRIBUT	DOMEN	OGRANIČENJE
MLB	CHAR(13)	NOTNULL AND SUBSTRING (1,2) BETWEEN 1,31 AND SUBSTRING (3,4) BETWEEN 1,12 ...
NAZIVJ	CHAR(15)	IN (SRPSKI, RUSKI, ENGLESKI, NEMAČKI)
ADRESA	CHAR(20)	
DATUM	DATE	
STAROST	INT(2)	BETWEEN 15,65

4.1.3. Atribut i veza - druge vrste modela podataka

Očigledno je, na osnovu definicije koncepata atributa i veze, da ne postoji formalna (a možda i suštinska) razlika izmedju ova dva koncepta. Veza definiše dva preslikavanja, direktno i inverzno, izmedju dve klase objekata, a aribut preslikavanje izmedju klase objekata i odgovarajućeg domena. Ako bi se svi domeni tretirali kao klase objekata tada bi atribut pretstavljao jedno preslikavanje u vezi te klase sa odgovarajućim objektom. Na primer, ako se domen IMENA tretira kao klasa objekata (čija su pojavljivanja nizovi karaktera dužine 20) tada bi atribut IME predstavljao preslikavanje RADNIK -----> IMENA u vezi objekata RADNIK i objekata IMENA. Isto tako, moglo bi se postupiti i obrnuto i preslikavanja u datim vezama tretirati kao atribute čiji su domeni odgovarajuće klase objekata. Na primer, ako bi se klasa objekata ODELJENJE tretirala kao domen, (složeni domen koji bi predstavljao skup parova < SIFOD, NAZIVOD > tad bi se preslikavanja RADI i RUKOVODI u odgovarajućim vezama, mogla tretirati kao atributi objekata RADNIK. To znači da su u modelima podataka dovoljna samo dva koncepta, bilo samo koncepti objekta i veze (preslikavanja), bilo samo koncept objekta i atributa (preslikavanja).

Prva vrsta modela koji koriste samo koncepte objekata i preslikavanja se nazivaju funkcionalni modeli podataka. Na Slici 8 prikazana je jedna vrsta funkcionalnih modela, za primer sa slike 7a (data je i legenda za grafičko prikazivanje kardinalnosti preslikavanja u vezama pojedinih objekata.) Klase objekata su ovde pojedinačni domeni, odnosno atributi.

U drugu vrstu modela koji koriste samo koncepte objekata i atributa spadaju pojedini semantički modeli podataka (SDM) i objektno_orjentisani modeli. Na Slici 9, sa jednom specifičnom očiglendom sintaksom, ilustrovan je princip modeliranja sistema ovakvom vrstom modela. (Svaki konkretan model ove vrste ima specifičnu sintaksu i u sebe uključuje i druge složenije semantičke koncepte, agregaciju i generalizaciju, koji će ovde biti kasnije uvedeni). Osnovna karakteristika ovih modela je ta što se pojam veze izmedju objekata eksplicitno ne koristi, već se implementira na taj način što se kao domen atributa jednog objekta tretira klasa drugih objekata. Iskazom INVERZNO u specifikaciji nekog atributa, ukazuje se na inverzno preslikavnje (atribut) koji sa posmatranim atributom čini jednu binarnu vezu.



Slika 8. Primer funkcionalnog modela podataka

Objekat RADNIK

Atributi:

MLB	MATB, (1,1), ident;
IMER	IMENA, (1,1);
ZNA_JEZIK	JEZIK, (1,M);
ADRESA	ADR, (1,1);
STAROST	GODINE, (1,1);
ISPLATE	PLATE, (1,M);
RADI	ODELENJE, (1,M), inverzno ZAPOŠLJAVA;
RUKOVODI	ODELENJE, (0,1), inverzno RUKOVODJENO;

Objekat ODELENJE

Atributi:

SIFOD	SIFREO, (1,1), ident;
NAZIVOD	IMENA, (1,1);
ZAPOŠLJAVA	RADNIK (0,M), inverzno RADI;
RUKOVODJENO	RADNIK (1,1), inverzno RUKOVODI

Slika 9. Primer objektno-orjentisanog modela podataka

U PMOV, iz semantičkih i praktičnih razloga, ako se model objekti-veze implementira preko nekog "klasičnog", rekord orjentisanog jezika ili sistema za upravljanje bazom podataka (COBOL, relacioni, mrežni ili hijerarhijski model), pogodno je razlikovati koncept atributa od koncepta veze.

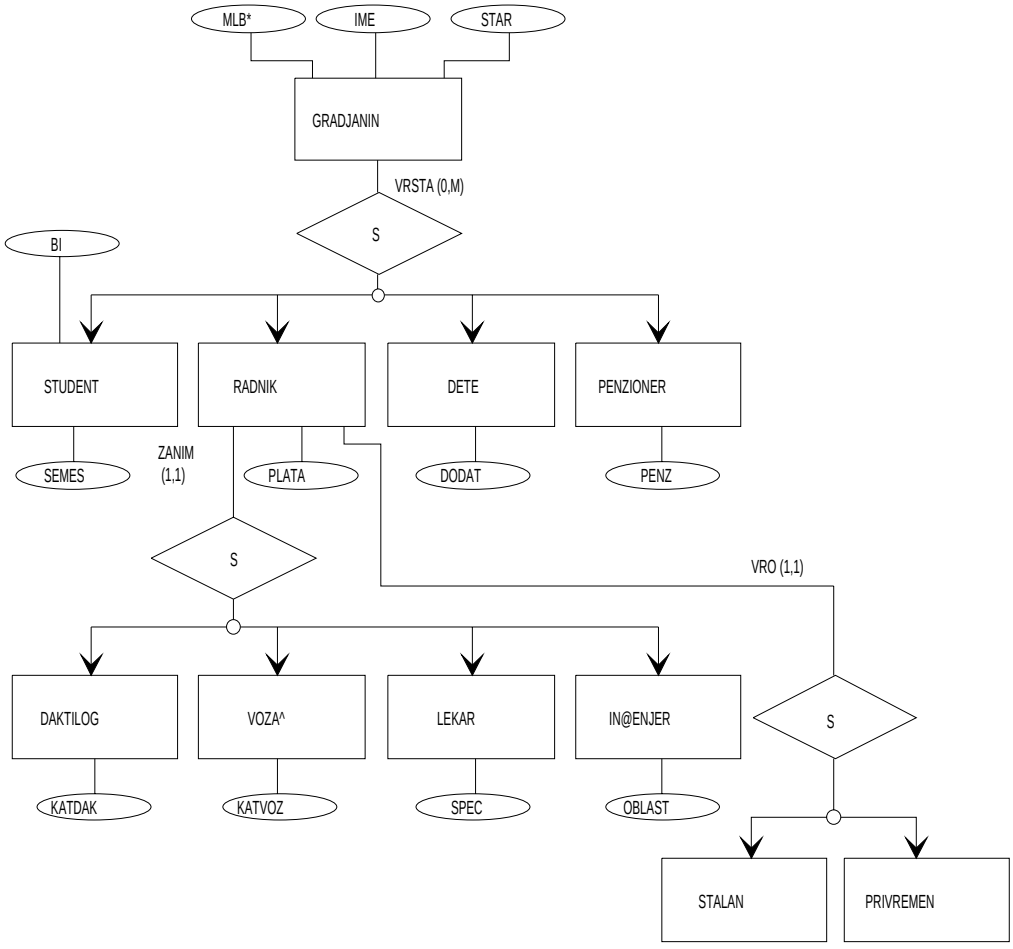
Sa semantičke tačke gledišta prihvata se da je prirodnije razdvojiti koncepte atributa i veza. Tako, na primer, prirodnije je tretirati IME kao atribut (svojstvo) objekata RADNIK, a RADI kao preslikavanje RADNIK ------> ODELENJE veze izmedju objekata RADNIK i ODELENJE. Kriterijum "prirodnije" je veoma relativan, svakom nešto drugo može biti prirodnije. Primena tog slabo definisanog kriterijuma "prirodnije" dovodi do dilema u modeliranju - kada neki skup objekata predstaviti kao domen, odnosno njegov odnos sa nekim drugim tipom objekata kao atribut, a kada ga predstaviti kao tip objekata, odnosno njegov odnos sa nekim drugim tipom objekata kao vezu. Uputstva za razrešenje ove dileme su sledeća:

- Ako ne postoji potreba za posebnim identifikovanjem svakog pojavljivanja objekta u skupu i ako su ta pojavljivanja vrednosti koje su "ugradjeni" tipovi podataka (bazni domeni) nekog SUBP-a ili nekog jezika (celobrojne vrednosti, karakteri, nizovi karaktera i slično), tada skup takvih objekata treba tretirati kao domen.
- Ako ne postoji potreba da se neki skup objekata opisuje atributima, tada ga treba tretirati kao domen. Drugačije rečeno, ako neka osobina objekata uzima vrednosti iz skupa prostih vrednosti, treba je predstaviti kao atribut, a dati skup kao domen, ili, ako neka osobina objekta uzima vrednosti iz skupa n-torki (parova, trojki), tu osobinu treba predstaviti kao vezu, a dati skup vrednosti kao tip objekata. Ova preporuka ne sprečava korišćenje složenog domena, ako je to povoljnije.
- Ako se, u toku modeliranja, ukaže potreba da se neki već definisani atribut poveže sa nekim objektom, odnosno da se definiše preslikavanje DOMEN ---> OBJEKAT, tada odgovarajući domen treba tretirati kao tip objekta, a atribut kao preslikavanje u vezi.

Sa praktične tačke gledišta, u mnogim primenama, za implementaciju modela, pogodno je jedan objekat i skup njegovih atributa tretirati kao rekord, pa je to još jedan razlog da se razdvoje koncepti atributa i veze. Time se imlicitno uvodi i koncept agregacije, jer se objekat može definisati kao agregacija njegovih atributa. O apstrakciji agregacije diskutovaće se docnije.

4.1.4. Generalizacija i specijalizacija

Generalizacija je, kao što je rečeno, apstrakcija u kojoj se skup sličnih tipova objekata tretira kao generički tip objekta (nadtip). "Slični" tipovi objekata su oni tipovi koji imaju neke zajedničke atribute, veze i/ili operacije. Specijalizacija je inverzni postupak u kome se za neki tip objekta, definišu njegovi podtipovi, koji imaju neke njima specifične atribute, veze i/ili operacije. Na primer, (Slika 10) skup tipova Radnik, Student, Penzioner, Dete može se predstaviti generičkim tipom Gradjanin. Isto tako se tip Radnik može specijalizovati u podtipove Vozač, Inženjer, Lekar i druge. Klasa nadtipa sadrži sva pojavljivanja "sličnih" objekata, sa onim atributima i preslikavanjima koji su im zajednički, a klase podtipova su podskupovi pojavljivanja nadtipa sa dodatnim, njima specifičnim preslikavanjima.



Slika 10. Generalizacija (specijalizacija)

Postupci generalizacije i specijalizacije se u PMOV pretstavljaju specijalnom "S" (Subtype ili IS-A (is a)) **vezom koja se sastoji od dva preslikavanja**: **(1) generalizacije, odnosno preslikavanja PODTIP ----> NADTIP** (trivijalno preslikavanje izmedju podskupa i skupa za koje je uvek DG = 1 i GG

= 1, pa se kardinalnost ovoga preslikavanja i ne pretstavlja na DOV) i (2) **specijalizacije, koja predstavlja preslikavanje NADTIP ----> NEPOVEZANA UNIJA PODTIPOVA**. "Nepovezana unija" podtipova predstavlja skup koji sadrži sva pojavljivanja svakog podtipa kao različite elemente u skupu, čak i kada su neka pojavljivanja dva (ili više) podtipova jednaka. Nepovezane linije skupova A,B,C ..., se formalno definiše kao skup E (unija) parova koji se sastoje od naziva skupa i elemenata skupa odnosno {<A, a_1>, <A, a_2>,... <A, a_n>, <B, b_1>, <B, b_2>,... <B, b_n>, <C, c_1>, <C, c_2>,... <C, c_n>}. Nepovezana unija tipova objekata predstavlja se kružićem na dijagramima objekti veze. Donja i gornja granica kardinalnosti specijalizacije definišu vrste specijalizacije odnosno generalizacije:

- Kada se jedno pojavljivanje tipa može specijalizovati samo u jedan podtip (ekskluzivna specijalizacija) tada je GG = 1. Na primer, jedno pojavljivanje tipa RADNIK može imati samo jedno zanimanje (u jednom trenutku vremena). Kada se jedno pojavljivanje tipa može specijalizovati u više podtipova (neekskluzivna specijalizacija) tada je GG > 1. Na primer, neki gradjanin može istovremeno da bude i dete i student ili i student i radnik.

- Kada se svako pojavljivanje tipa mora specijalizovati u neki podtip (unija podtipova je jednaka nadtipu) tada je u specijalizaciji DG = 1. Ako je u specijalizaciji DG = 0, tada ne mora svako pojavljivanje tipa biti specijalizovano u neki podtip (unija podtipova je podskup nadtipa).

Jedan tip objekta može se po više različitih kriterijuma (kategorija) specijalizovati u različite skupove podtipova. Na primer, tip Radnik se po kategoriji ZANIM(anje) specijalizuje u podtipove Daktilograf, Vozač, Lekar, Inženjer, a po kategoriji V (vrsta) R (radnog) O (odnosa) u podtipove Stalan i Privremen. Naziv preslikavanja NADTIP ----> NEPOVEZANA UNIJA PODTIPOVA iskazuje kategoriju (kriterijum) specijalizacije.

U generalizacionoj hijerarhiji objekata važi pravilo nasledjivanja osobina i pravilo nasledjivanja operacija:

- Podtipovi nasledjuju sve atribute i veze svoga nadtipa.

- Podtipovi nasledjuju sve operacije svoga nadtipa.

Na primer objekat Student je podtip objekta Gradjanin i pored svojih atributa BI i SEMES, on nasledjuje i atribute svoga "nadtipa", MLB, IME, STAR. Atributi objekta Lekar su njegovi atributi i atributi svih njegovih nadtipova po generalizacionoj hijerarhiji, tj. SPEC, PLATA, MLB, IME i STAR. Objekat Lekar nasledjuje i operacije svojih nadtipova u sledećem smislu: Lekar, pored toga što "zna" da obavi svoje operacije (prihvati bolesnika, na primer), može da obavi i sve operacije koje obavlja Radnik (da se zaposli, da bude premešten ili otpušten) i sve operacije koje obavlja Gradjanin (da se preseli, na primer).

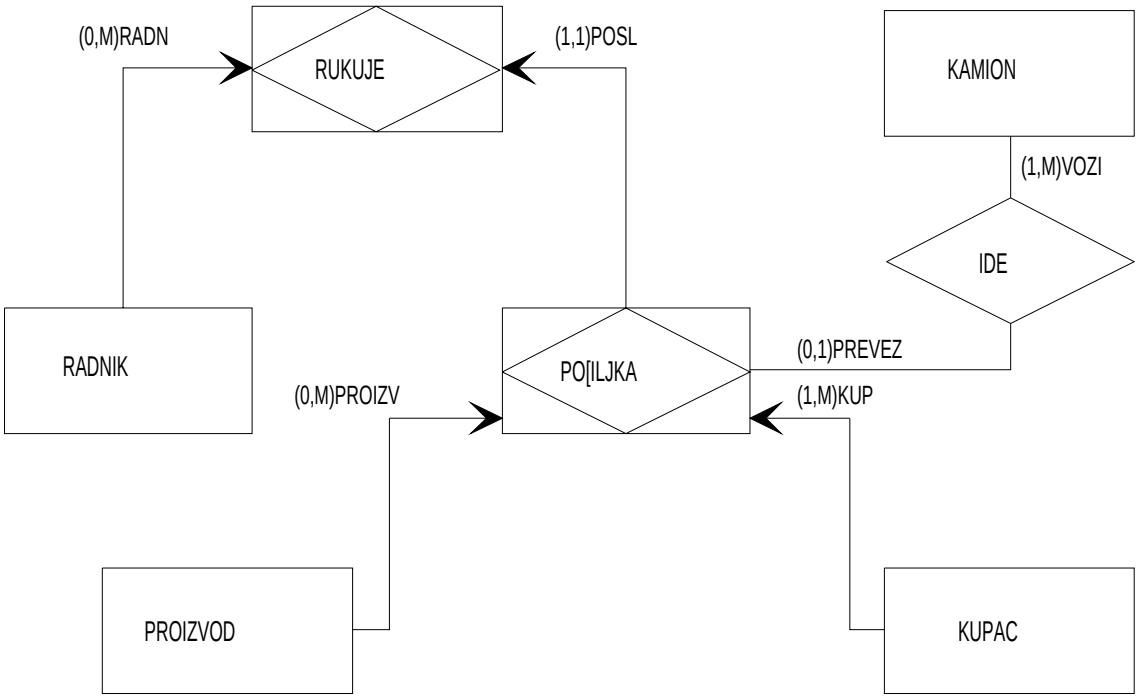
Očigledno je da se definisanjem podtipova nekog tipa razrešava problem atributa koji nisu primenljive osobine svih objekata u skupu objekata jednog tipa, na taj način što se definiše podtip kao skup pojavljivanja na koje je dato svojstvo primenljivo. Podtip tada ima samo to svojstvo (ta svojstva) kao atribut, a ostale atribute svojstvene svima, "nasledjuje" od nadtipa.

4.1.5. Agregacija i dekompozicija

Agregacija je apstrakcija u kojoj se veza izmedju dva ili više tipova objekata tretira kao objekat na višem nivou apstrakcije. Zbog toga što istovremeno pretstavlja i objekat i vezu agregacija se često naziva i **mešoviti tip objekta-veza**. Objekti koji čine agregaciju se nazivaju komponentama agregacije. Postupak inverzan aregaciji se naziva dekompozicija. Kardinalnost preslikavanja **KOMPONENTA ---> AGREGACIJA** mora biti specificirana, dok je za inverzno preslikavnaje uvek DG = 1 i GG = 1, što znači da je agregacija egzistencijalno zavisna od svojih komponenata. Agregirani objekat se razlikuje od ostalih

objekata u sistemu po tome što nema svoj sopstveni identifikator, već ga identifikuju objekti koje on agregira.

Postupak agregacije je već implicitno uveden u model objekti-veze uvodjenjem koncepta objekta i atributa. Jedan objekat se može predstaviti kao agregacija njegovih atributa. Primeri agregacija dati su na slici 11. Pojavljivanja objekata iz klasa PROIZVOD i KUPAC agregiraju se u pojavljivanje objekata iz klase POŠILJKA. Agregacija POŠILJKA se, s jedne strane vezuje sa klasom objekata KAMION, a sa druge, zajedno sa klasom RADNIK, formira agregiranu klasu objekata RUKUJE.



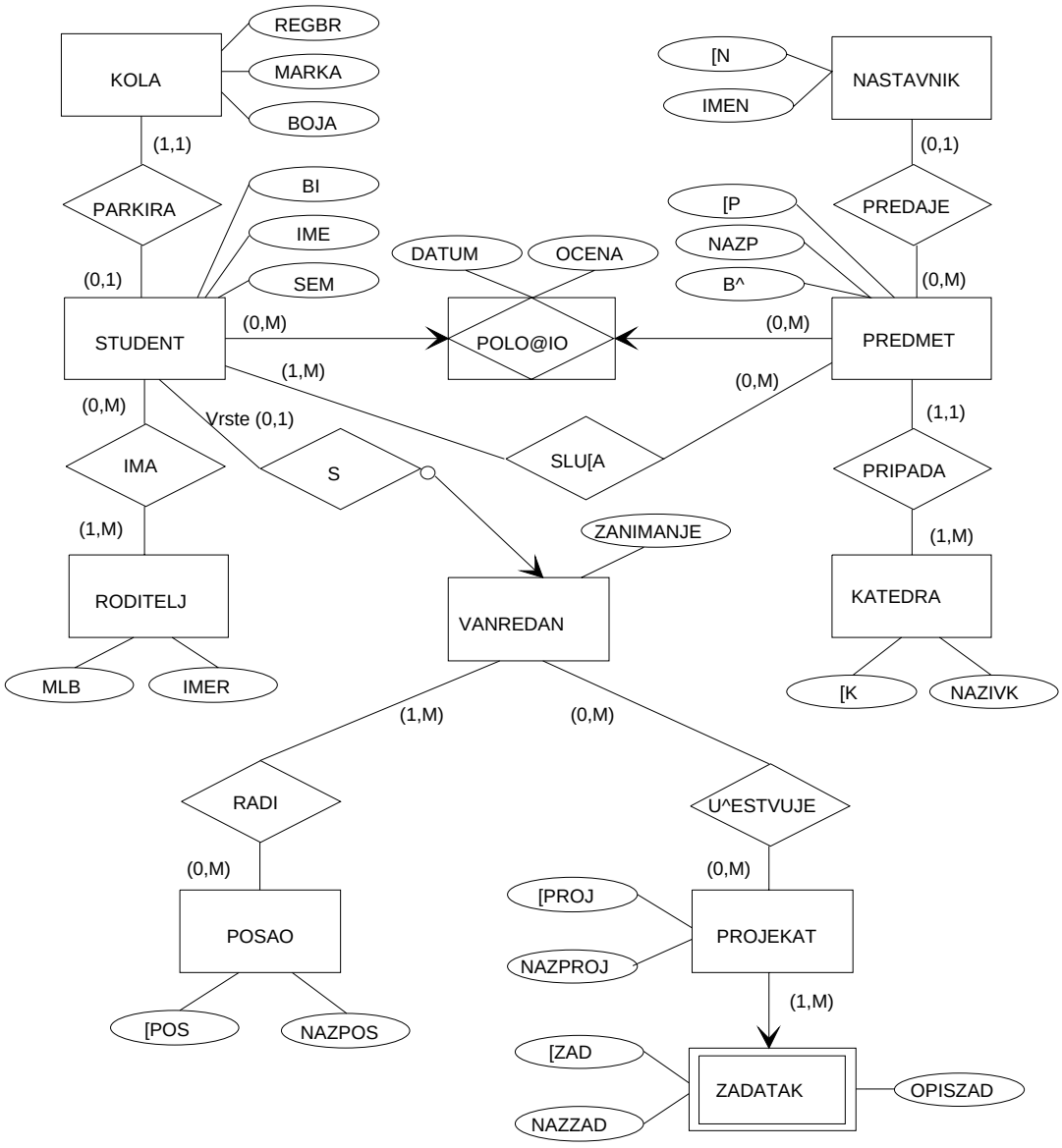
Slika 11. Agregacija i dekompozicija

Agregirani objekat se u modelu tretira kao i bilo koji drugi objekat. To znači da on može da ima svoje atribute i/ili da bude u vezi sa nekim drugim objektima (moguće agreiranim, takodje), da ima svoje podtipove i slično. Mada svaka veza agregira dva objekata, samo one binarne veze kojima bi imalo smisla dodati neke atribute ili koje bi imalo smisla povezivati sa drugim objektima treba tretirati kao agregirane objekte (agregacije). Dobra praksa modeliranja može da bude i to da se svaka binarna veza u kojoj oba preslikavanja imaju GG = M, tretira kao agregacija.

U PMOV koriste se samo binarne veze objekata. Ako postoji potreba da se direktno predstavi veza tri ili više objekata tada se takva veza tretira kao agregacija. Na primer, činjenicu da dobavljači dobavljaju neke proizvode za specifične projekte može se predstaviti kao agregacija objekata Dobavljač,

Proizvod i Projekat. Ako se višestruke veze predstavljaju kao agregirani objekti, tada model uključuje samo binarne veze što omogućuje jasno definisanje kardinalnosti direktnog i inverznog preslikavanja koje ih čine.

Slika 12 daje jedan kompletan model koji će poslužiti za ilustraciju ostalih delova teksta.



Slika 12. DOV za fakultetski IS

4.2. Ograničenja u PMOV

U prethodnom delu, pri uvodjenju i definisanju pojedinih koncepata strukture PMOV, definisana su i neka ograničenja - kardinalnost preslikavanja i ograničenja na vrednosti domena. Medjutim, postoje i mnogo složenija semantička ograničenja (uzajamni odnosi objekata i njihovih atributa) koje je nemoguće ili nepraktično prikazati strukturom PMOV, pa je neophodno definisati jezik za njihovu specifikaciju. Pre nego što uvedemo jezik za specifikaciju ograničenja, pogodno je klasifikovati ograničenja u sledeće klase:

(1) Strukturna ograničenja (ograničenja na preslikavanja)
(2) Vrednosna ograničenja (ograničenja na vrednosti atributa)

Na kraju ovog dela definisane su i neke funkcije koje omogućuju da se u nekom konkretnom ograničenju povežu strukturni i vrendosni iskazi.

4.2.1. Strukturna ograničenja

Sama struktura PMOV, odnosno dijagram objekti veze iskazuje ova ograničenja. Na svakom DOV zadate su kardinalnosti preslikavanja za sve veze. Neka od njih se ne upisuju, već se podrazumevaju, kako je to diskutovano. Isto tako, s obzirom na to da se koriste samo jednoznačni atributi, da svaki atribut mora biti "primenljivo svojstvo" za sva pojavljivanja posmatranog objekta i da su identifikatori objekata označeni, implicitno su definisane kardinalnosti direktnog (atributa) i inverznog preslikavanja OBJEKAT ----> DOMEN.

Pored ovih ograničenja, u redjim slučajevima je potrebno definisati i ograničenje tipa "redosled" za preslikavanja čija je gornja granica kardinalnosti GG > 1, koje definiše zahtevani redosled povezivanja pojavljivanja objekta kodomena preslikavanja sa datim pojavljivanjem objekta domena preslikavanja. Ograničenje ove vrste može se iskazati sintaksom:

def_redosleda_presl:: naziv_preslikanja REDOSLED vrsta_redosleda
vrsta_redosleda:: PRVI | POSLEDNJI | ISPRED | IZA |
SORTIRANO_PO [OPADAJUĆEM] naziv_atributa_kodomena

Značenje pojedinih vrsta redosleda je sledeće:

- Pri ubacivanju novog pojavljivanja objekta kodomena ovaj se postavlja kao PRVI, ili POSLEDNJI u nizu, ISPRED ili IZA datog ("tekućeg") pojavljivanja kodomena ili na zahtevano sortno mesto. Ako nije specificiran sortni redosled OPADAJUĆI, podrazumeva se rastući.

4.2.2. Vrednosna ograničenja

Mada se u osnovi zadaju na isti način, iz praktičnih razloga je pogodno ovu klasu ograničenja podeliti na dve podklase: Ograničenja na domene i medjuzavisnosti atributa. Ograničenja na domene se jednostavno iskazuju i daju se uz definiciju domena kako je to ranije pokazano. Za iskazivanje medjuzavisnosti atributa koriste se formule "objektnog računa", predikatskog računa prvog reda u kome su promenljive objekti u sistemu.

Osnovni pojmovi predikatskog računa su:

(1) Afirmativna rečenica, koja ima smisla i koja je istinita ili neistinita naziva se sud.

(2) Afirmativna rečenica koja ima smisla i koja sadrži jedan ili više promenljivih parametara i koja postaje sud uvek kada parametri iz rečenice dobiju konkretnu vrednost naziva se predikat. Broj parametara u predikatu se naziva dužina predikata. Na primer, $x^2 + y^2 <= 1$ je primer predikata dužine 2.

(3) Promenljive u objektnom računu su objekti odnosno, može se da promenljiva uzima vrednost iz neke klase objekta, što se označava na sledeći način:

* x : O, gde je x pojavljivanje, a O klasa nekog objekta, odnosno, promenljiva x uzima vrednosti iz skupa pojavljivanja objekta O.

* x : O.P, x je objektna promenljiva koja uzima vrednosti iz kodomena preslikavanja P objekta O.

(4) **Predikatski ili kvantifikatorski** račun je matematička teorija čiji su objekti formule koje predstavljaju predikate. Simboli koji se koriste da označe neki sud nazivaju se atomskim formulama ili atomima. Atomi u objektnom računu su:

x.A □ y.B gde su x i y promenljive (objekti), A i B su atributi tipova objekata O1 i O2 iz čijih pojavljivanja, respektivno, promenljive x i y uzimaju vrednosti (x : O1, y : O2), a □ je operacija poredjenja (na primer <, >, =, IZMEDJU,...) definisana nad domenom atributa A i B (A i B moraju biti definisani nad istim domenom).

x.A □ c gde su x, A i □ kao i u prethodnom stavu, a c je konstanta koja ima isti domen kao i A.

Dobro definisane formule (ddf) se formiraju od atoma preko sledećih pravila:

```
ddf:: =  atom |
         (ddf) |
         NOT ddf |
         ddf AND ddf |
         ddf OR ddf |
         EXISTS naziv_promenljive (ddf) |
         FOREACH naziv_promenljive (ddf) |
         IF ddf THEN ddf |
```

POSTOJI () je egzistencijalni kvantifikator, a FOREACH () je univerzalni kvantifikator. Jedna promenljiva u nekoj formuli se može pojaviti više puta. Promenljive mogu biti "slobodne" i "vezane". Vezana promenljiva u formuli je neka vrsta "prividne" promenljive u zoni dejstva kvantifikatora. Drugim rečima vezana promenliva u je (EXISTS u), (FOREACH u) ili (EXISTS u)A ili (FOREACH u)A, gde je A formula u kojoj se pojavljuje u. Sve promenljive koje u nekoj formuli nisu vezane, slobodne su u toj formuli. Na primer, u formuli

EXISTS x (x > 3)

x je vezana promenljiva, pa je gornja formula ekvivalentna sa

EXISTS y (y > 3)

U formuli

EXISTS x (x > 3) AND x < 0

prvo pojavljivanje promenljive x je vezano, a drugo slobodno, pa je ova formula ekvivalentna sa

EXISTS y (y > 3) AND x < 0

Svako ograničenje je neka korisnički definisana formula u kojoj promenljive uzimaju vrednosti iz definisanih klasa objekata.

Ograničenje se definiše na sledeći način:

naziv_ograničenja (lista_promeljivih) : = ddf;

Pre iskazivanja ograničenja treba definisati promenljive, na primer:

```
x  : STUDENT
    OGR1 (x) := x.SEM BETWEEN 1 i 9;
```

Ako se usvoji konvekcije da promenljive dobije naziv klase iz koje uzima vrednost tada se ona ne mora definisati

```
OGR1 (STUDENT)  : = STUDENT.SEM BETWEEN 1 i 9;
```

Primenjivaće se prvi način (kada se želi da se skrati ime promenljive u iskazima) inače ova druga konvekcije.

Navedimo nekoliko primera ograničenja:

(1) Student petog ili višeg semestra mora da ima više od 10 položenih ispita.

```
OGR2(STUDENT,POLOŽIO)  : = FOREACH  STUDENT (IF  STUDENT.SEM  >= 5
                                        THEN CARD (STUDENT.POLOŽIO)>10);
```

(Ovde je uvedena funkcija CADR (S) koja daje broj elemenata skupa S)

(2) Student trećeg semestra je položio Matematiku.

```
OGR3(STUDENT,POLOŽIO,PREDMET) : =FOREACH STUDENT (IF STUDENT.SEM = 3 THEN
EXISTS STUDENT. POLOŽIO (STUDENT.POLOŽIO.PREDMET. NAZP ='MATEM'));
```

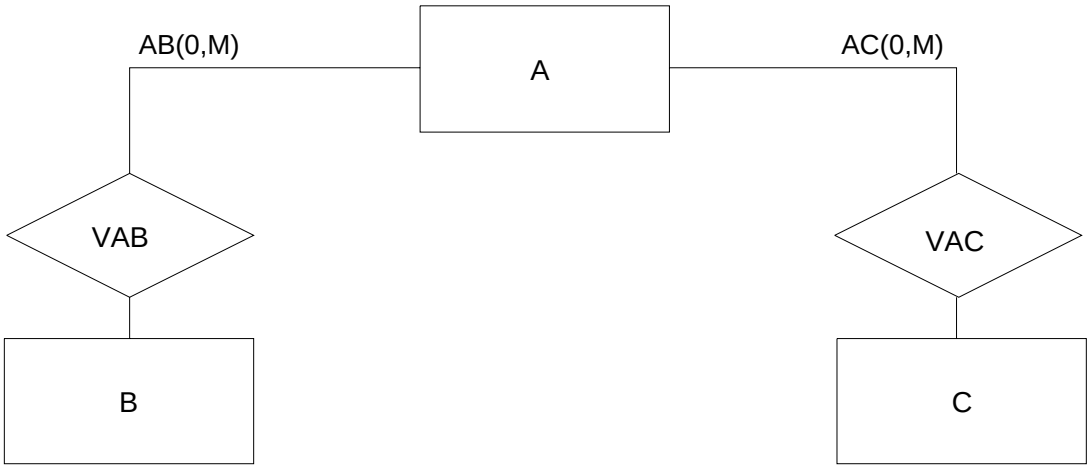
(3) Predpostavimo da u modelu na Slici 12 objekat STUDENT ima i izvedeni atribut PROSOC (prosečna ocena). Tada očigledno važi sledeće ograničenje:

```
OGR4(STUDENT,POLOŽIO)  : = FOREACH  STUDENT (STUDENT.PROSOC =
                SUMA (STUDENT. POLOŽIO, OCENA) / CARD (STUDENT. POLOŽIO);
```

Uvedena funkcija CARD (X) omogućuje da se i strukturna ograničenja na kardinalnost preslikavanja predstave kao vrednosna ograničenja, odnosno da se kombinuju strukturna i vrednosna ograničenja u jednom iskazu. Ilustrovaćemo to na primerima koji, još jednom, pokazuju kako se ista semantika realnog sistema može prikazivati na različite načine.

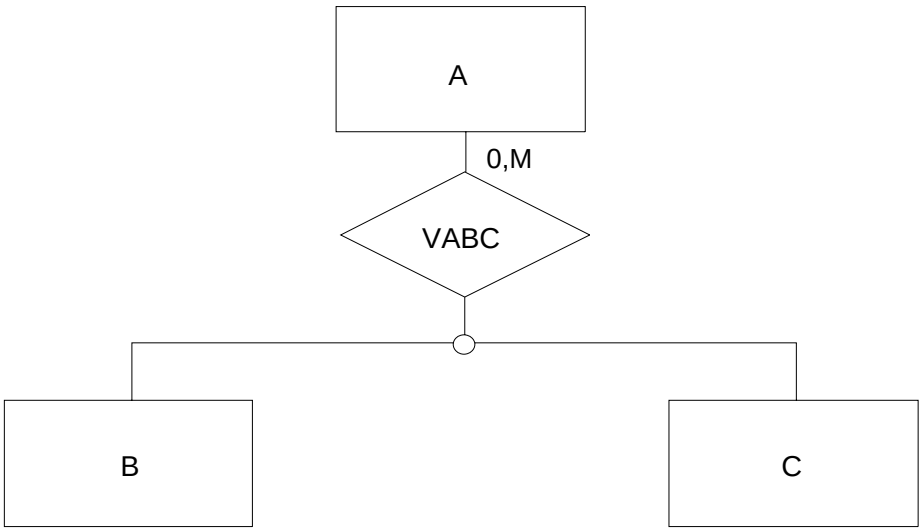
Medjusobno ekskluzivna preslikavanja. Ako je nekom pojavljivanju objekta A u preslikavanju AB, pridruženo jedno ili više pojavljivanja objekta B tada tom pojavljivanju A nije pridruženo ni jedno pojavljivanje objekta C, u preslikavanju AC. (Neka operacija u tehnološkom postupku se obavlja bilo ručno bilo na datoj mašini, ili avion prevozi bilo putnike bilo teret). Na Slici 13a i 13b dati su različiti načini prikazivanja ovakvog odnosa. Na Slici 13a, definisane su posebne veze VAB i VAC, a uslov

ekskluzivnosti preslikavanja A ---> B i A ----> C iskazuje se dodatnim ogrnaičenjem OGRE (A,B,C). Na Slici 13b uvedena je nova grafička oznaka za prikazivanje eksluzivnih preslikavanja - "razgranata veza" gde prazan kružić znači uslov ekskluzivnosti. Zbog jednostavnosti se preporučuje korišćenje grafičke (strukturne) pretstave.



OGRI(A,B,C):=FOREACH A((IF CARD A.AB > 0 THEN CARD(A.AC)=0) OR
IF CARD(A.AC)>0 THEN CARD(A.AB)=0))

(a) Kombinacija strukture i ograničenja

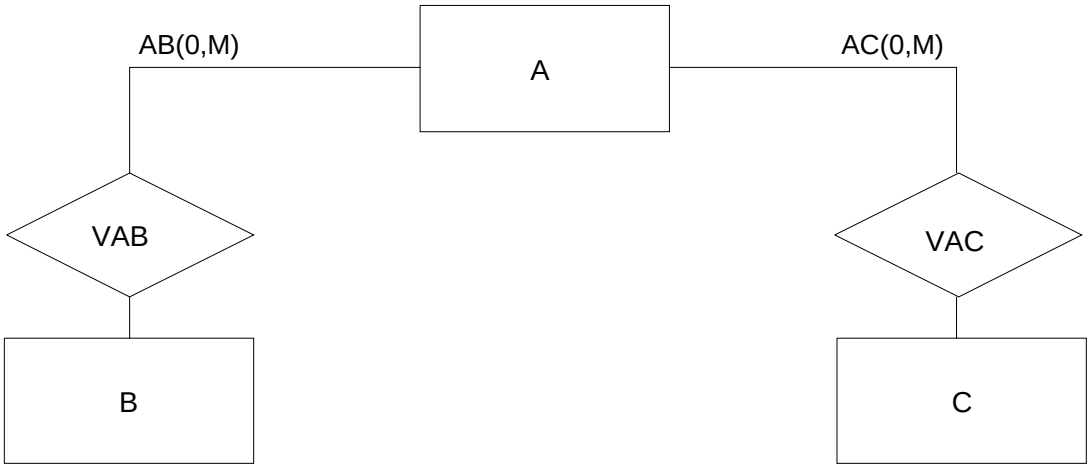


(b) Predstavljanje preko strukture

Slika 13. Medjusobno eksluzivna preslikavanja

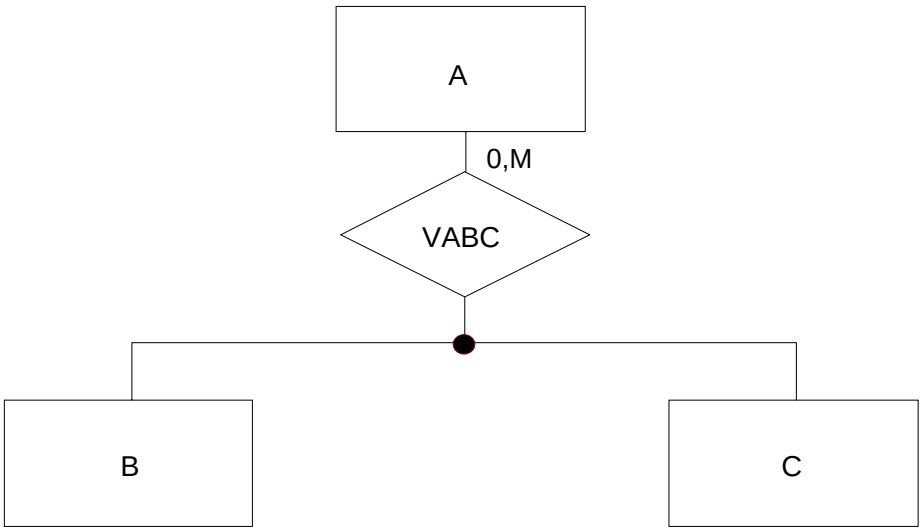
Medjusobno inkluzivna preslikavanja. Ako je nekom pojavljivanju objekta A u preslikavanju AB pridruženo barem jedno pojavljivanje objekta B, tada tom pojavljivanju objekta A, u preslikavanju AC, mora biti pridruženo barem jedno pojavljivanje objekta C. Na Slici 14a korišćene su standardne oznake za strukturu PMOV uz eksplicitno iskazivanje uslova inkluzivnosti, a na slici 14b je data nova

grafička oznaka, oznaka za inkluzivnu vezu - "razgranata veza" gde pun kružić označava uslov inkluzivnosti. I ovde bi se moglo preporučiti korišćenje grafičke oznake.



OGRI(A,B,C):= FOREACH A((IF CARD A.AB>0 THEN CARD A.AC>0) OR
(IF CARD A.AC>0 THEN CARD A.AB>0)

(a) Kombinacija strukture i ograničenja



(b) Predstavljanje preko strukture

Slika 14. Medjusobno inkluzivna preslikavanja

Medjutim, grafičke oznake mogu da "pokriju" samo na gornji način iskazane uslove inkluzivnosti i ekskluzivnosti, a ne i neke druge, slične. Na primer, uslov inkluzivnosti da ako je nekom pojavljivanju objekta A pridružen odredjeni broj pojavljivanja objekata B u preslikavanju AB, mora mu biti pridružen isti toliki broj pojavljivanja objekta C u preslikavanju AC se ne može iskazati grafički. Zbog toga je neophodno uvesti jezik za eksplicitno definisanje ograničenja.

4.3. Operacije u PMOV

Pogodno je operacije u PMOV podeliti na operacije održavanja (ažuriranja) baze podataka i operacije pretraživanja (izveštavanja). Primeri i jednih i drugih operacija će se dati za model na Slici 16.

4.3.1. Operacije održavanja

Iz same strukture PMOV očigledno je da se operacije održavanja baze podataka svode na operacije dodavanja novog objekta u klasu, izbacivanja objekta iz klase, izmenu vrednosti nekog atributa, zatim povezivanja, razvezivanja i prevezivanja dva objekta preko zadatog preslikavanja, odnosno veze. Sintaksa i detaljnije objašnjenje ovih operacija slede:

(1) UBACIVANJE
insert naziv_objekta(dvotačka_lista parova atribut-vrednost)
dvotačka_lista parova id_susednog objekta-vrenost)

Na primer

insert RADNIK (MLB:307,IME:Ana, STAROST:25, ŠIFOD:03)

U listi parova atribut-vrednost neophodno je zadati sve atribute koji ne mogu imati nula vrednost. Podrazumeva se da je među njima i atribut identifikator objekta, jer on, očigledno ne može da ima nula vrednost. Lista parova id_susednog objekta-vrednost daje indentifikatore svih susednih objekata prema kojima dati objekat ima"obavezno" preslikavanje (preslikavanje sa donjom granicom većom od nule).

Podrazumeva se da operacija ubaci prouzrokuje operacije poveži prema svim okolnim objektima prema kojima posmatrani objekat ima obavezno preslikavanje.

(2) IZBACIVANJE
delete naziv_objekta (naziv_id_objekta:vrednost)

Na primer
delete RADNIK (MLB:307)

Podrazumeva se da operacija izbaci prouzrokuje operacije razveži za sva preslikavanja u kojima učestvuje posmatrani objekat.

(3) AŽURIRANJE

update naziv_objekta (naziv_id_objekta:vrednost,
dvotačka_lista parova aribut_vrednost)
Na primer
update RADNIK(MLB:307,IME:Mira)

Atributi pojavljivanja objekta zadatog identifikatorom dobijaju zadate vrednosti. Identifikator objekta se ne može ažurirati.

(4) POVEZIVANJE
connect naziv_preslikavanja (_id_domena:vrednost,
id_kodomena:vrednost)

Operacija connect se ne može direktno primeniti na preslikavanja čije su kardinalnosti DG = 1 i GG = 1, jer se to povezivanje ostvaruje automatski sa ubacivanjem objekta domena preslikavanja. Na primer,

connect RUKOVODI(MLB:307,ŠIFOD:03)

Podrazumeva se da operacija poveži za jedno preslikavanje prouzrokuje i operaciju poveži za njeno inverzno preslikavanje.

(5) RAZVEZIVANJE
disconnect naziv_preslikavanja (_id_domena:vrednost,
id_kodomena:vrednost)

Operacija disconnect se ne može direktno primeniti na preslikavanja čije su kardinalnosti DG = 1 i GG = 1 inače bi uvek narušila uslov integriteta. Na primer,

disconnect RUKOVODI(MLB:307,ŠIFOD:03)

Podrazumeva se da operacija razveži za jedno preslikavanje prouzrokuje i operaciju razveži za njeno inverzno preslikavanje.

(6) PREVEZIVANJE
reconnect naziv_preslikavanja
(id_domena:vrednost, id_kodomena_sa:vrednost, id_kodomena-na:vrednost)

Operacija reconnect se sastoji od uzastopne prinene operacije disconnect sa jednog kodomene i connect na drugi, u jednoj atomskoj transakciji. Ona se primenjuje na ona preslikavanja na koja se operacije disconnect i connect ne mogu direktno primeniti. Na primer,

reconnect RADI(MLB:037, ŠIFOD:03, ŠIFOD:07)

4.3.2. Operacije pretraživanja

U PMOV je moguće definisati dve vrste operacija za pretraživanje. Prvi skup operacija je "navigacione" prirode, slično operacijama u mrežnom i hijerarhijskom modelu, i one omogućuju kretanje kroz model i pristup pojedinačnim pojavljivanjima objekata "tekuće" klase, klase kojoj se u tom trenutku pristupilo. Drugu grupu operacija čine tzv "specifikacione" operacije, odnosno odgovarajući upitni jezik, kojima se iskazuje struktura i uslovi koje rezultat operacije treba da zadovolji.

Da bi se definisale operacije nad klasama objekata i omogućilo jednostavno povezivanje klasa objekata preko preslikavanja definišu se promenljive u jeziku na sledeći način:

naziv_promenljive ":" naziv_klase_objekata,

što zanači da promenljiva uzima vrednosti iz klase (skupa) objekata. Na primer x:RADNIK, promenljiva x uzima za svoje vrednosti pojavljivanja klase objekata RADNIK ili

naziv_promenljive_1 ":" naziv_promenljive_2 "." naziv_preslikavanja,

što znači da prva promenljiva uzima vrednost iz kodomena navedenog preslikavanja za vrednost domena preslikavanja datog vrednošću druge promenljive. Na primer, ako se definiše:

x: ODELENJE

y: x.ZAPOŠLJAVA,

y uzima vrednosti iz podskupa klase RADNIK koji čine pojavljivanja koja su su preko preslikavanja ZAPOŠLJAVA vezani za dato odelenje preko tekuće vrednosti promenljive x.

Kako je objekat agregacija njegovih atributa, promenljivoj x se takođe mogu pridodati atributi preko iskaza:

naziv_promrnljive "." naziv atributa
Na primer x.NAZOD ili y.IME ili y.MLB i slično.

Kao i ranije, pri definiciji ograničenja, može se umesto definicije promanljive usvojiti konvencija da su nazivi promenljivih jednaki nazivima klasa iz koji uzimaju vrednosti.

4.3.2.1. Navigacione operacije

Očigledno je da je osnovna operacija pretraživanja izvlačenje jednog pojavljivanja objekta iz zadate klase. Ako je klasa prazna, odgovarajuća operacija vraća kod "EOK".

Definišu se sledeće navigacione operacije pretraživanja u modelu objekti veze:

(1) GET ANY naziv_ promenljive {daje bilo koje pojavljivanje klase nad kojom je promenljiva definaisana ili EOK ako je klasa prazna}. Na primer:

x:ODELENJE, y:x.ZAPOŠLJAVA

GET ANY x - daje bilo koje pojavljivanje objekta odelenje,
GET ANY y - daje bilo koje pojavljivanje objekta radnik iz podskupa radnika vezanih za tekuće pojavljivanje objekta odelenje.

(2) GET ANY naziv_promenljive WHERE uslov { Uslov se može iskazati preko relacionih operatora (=, <, > i slično) koji se mogu definisati nad domenima atributa klase nad kojom je definisana promenljiva i logičkih operatora AND OR i NOT. Rezultat operacije je jedno, bilo koje, pojavljivanje objekta koje zadovoljava uslov.} Na primer, sekvenca naredbi

GET ANY x WHERE x.ŠIFOD = 011;
GET ANY y WHERE y.ZANIMANJE = 'lekar'AND y.STAROST > 40;

daje odelenje sa šifrom 011 i sve lekare starije od 40 godina iz tog odelenja ili EOK ako takvih radnika nema.

Operacija GET ANY daje samo jedno pojavljivanje date klase, odnosno jedno pojavljivanje koje zadovoljava dati uslov. Da bi se mogli izvući i druga (ili sva) pojavljivanja uvode se operacije

(3) GET DUPLICATE naziv_promenljive i
(4) GET DUPLICATE naziv_promenljive WHERE uslov

koje daju jedno od preostalih pojavljivanja date klase, posle primene operacija GET ANY ili GET DUPLICATE. Na primer, ako bi želeli da prikažemo sve lekare, starije od 40 godina, iz odelenja O11, odgovarajuća procedura bi imala sledeći pseudokod:

```
BEGIN
GET ANY x WHERE x.ŠIFOD = 011;
IF EOK THEN
```

```
            BEGIN
            PRINT 'ne postoji to odelenje';
            GO TO kraj;
           END;
          ELSE
            BEGIN
            PRINT  x;
            GET   ANY   y   WHERE y.ZANIMANJE = 'lekar'
                        AND  y.STAROST > 40;
            IF EOK THEN
                 BEGIN
                 PRINT 'ne postoji takav radnik';
                 GO TO kraj;
                 END;
                ELSE
                BEGIN
                 PRINT y;
                 DO WHILE NOT EOK
                 GET DUPLICATE y WHERE y.ZANIMANJE = 'lekar'
                           AND y.STAROST >40;
                 PRINT y;
                 END DO;
                 END;
            END;
      kraj       STOP;
       END.
```

Promenljiva koja je zadata nad specijalizacijom kao preslikavanjem ima neke specifične karakteristike. Naime, domen preslikavanja specijalizacije je nepovezana unija podtipova posmatranog tipa. Nepovezana unija skupova definiše se kao unija parova koji se sastoje od naziva skupa i elementa skupa koji se "uniraju". Imajući to u vidu promenljiva definisana nad ovom preslikavanjem ima jedan atribut sa imenom "type" i promenljivu strukturu zavisno od konkretne vrednosti atributa "type". Ako je

```
      x: naziv_nadipa
      y: x.naziv_specijalizacije
tada promenljiva y ima strukturu:
      < type, CASE type OF
            naziv_prvog_podtipa :lista_atributa_prvog_ podtipa;
            naziv_drugog_podtipa :lista_atributa_drugo_ podtipa;
.
.
           naziv_n-tog_ podtipa : lista_atributa_n-tog_ podtipa>
```

Nadalje slede dva primera prikazivanja objekata u ekskluzivnoj i neekskluzivnoj specijalizaciji.

(i) Neobavezna i neekskluzivna specijalizacija na Slici 10.

```
   Prikazati sve podatke o građaninu sa MLB = 321
   x: GRAĐANIN
   y: x.VRSTA
   BEGIN
   GET ANY x WHERE x.MLB = 321;
   IF EOK THEN BEGIN
            PRINT 'ne postoji takav građanin';
```

```
        STOP;
        ELSE BEGIN
        PRINT x.MLB, x.STAR
        GET ANY y
        IF EOK THEN STOP;
            ELSE  PERFORM UNTIL NOT EOK
                CASE y.type OF
                RADNIK : PRINT y.PLATA;
                STUDENT: PRINT y.BI, y.SEMES;
                PENZIONER: PRINT y.PENZ;
                DETE: PRINT y.DODAT
                GET DUPLICATE y;
                END PERFORM
          END;
END.
```

(Zbog toga što jedan građanin može da se specijalizuje u više podipova neophodna je iteracija u gornjem programu).

(ii) Obavezna i eksluzina specijalizacija, primer sa Slike 16. Prikazati sve podatke o priizvodu sa šifrom 231 P

```
x: PROIZVOD
y: x.VRSTA
BEGIN GET ANY x WHERE x.ŠPRO = 231;
IF EOK THEN BEGIN
        PRINT 'ne postoji takav proizvod';
        STOP;
        END;
     ELSE BEGIN
        PRINT x.ŠPRO, x.NAZPRO
        GET ANY y
        CASE y.type OF
        MAŠINA: PRINT y.AMORT;
        MATREIJAL: PRINT y.POREKLO;
        END;
END;
```

Pošto je specijalizacija obavezna nije neophodno testirati da li podtip postoji. Pošto je specijalizacija eksluzivna postoji samo jedan podtip pa nije neophodna iteracija, već samo jedan pristup podtipu.

4.3.3. Operacije izveštavanja

Operacije izveštavanja formiraju izvedene (složene) objekte, objekte koji se mogu izvesti iz baznih objekata predstavljenih strukturom PMOV. Da bismo jasnije objasnili operacije formiranja izvedenih objekata, prikazaćemo prvo jedan primer takvog objekta. Na Slici 15 tabelarno je prikazana struktura izvedenog objekta koji se dobija tako što se uz svaku šifru i naziv odelenja navode matični brojevi, imena i starost radnika starijih od 40 godina i srednja starost radnika po odelenju.

ODELENJE RADNICI

SIFOD	NAZIVOD	MLB	IME	STAROST	SREDNJA_STAROST
10	Razvoj	1303945	Ana	43	49,3
		1402935	Zoran	53	
		1511936	Pera	52	
22	Nabavka	1610937	Mira	51	56
		1709227	Aca	61	
23	ERC	-	-	-	-

Slika 15. Tabelarni prikaz izvedenog objekta Stari_radnici_odelenja

Analizirajući izvedeni objekat pretstavljen tabelom na Slici 15, može se zaključiti da je jedno pojavljivanje ovoga tipa objekta agregacija:

- Jednog pojavljivanja nekih atributa objekta Odelenje (SIFOD i NAZIVOD);

- Jednog pojavljivanja preslikavanja ZAPOŠLJAVA: ODELENJE ---> RADNIK gde se iz kodomena RADNIK uzimaju samo neki atributi (MLB, IMER i STAROST) i selektuju samo ona pojavljivanja koja zadovoljavaju uslov RADNIK.STAROST > 40. Jedno pojavljivanje preslikavanja ZAPOŠLJAVA predstavlja podklasu objekata RADNIK (skup pojavljivanja tipa objekta Radnik).

- Izvedenog atributa SREDNJA_STAROST koji se dobija kao vrednost neke funkcije nad atributima posmatranog modela. (U ovom slučaju kao srednja vrednost atributa STAROST za radnike iz jednog odelenja, starije od 40 godina).

Ovaj izvedeni objekat se može dobiti sledećim izrazom:

```
(i)   x : ODELENJE, y : x. ZAPOŠLJAVA
      SELEKTUJ x.SIFOD, x.NAZIVOD,
                  (SELEKTUJ y.MLB,y.IMER,y.STAROST
                        GDE_JE y.STAROST > 40),
                  (SELEKTUJ AVG (x.ZAPOŠLJAVA,STAROST)
                        GDE_JE y.STAROST > 40);
```

U gornjem izrazu definisane su dve objektne promenljive: (1) x koja uzima vrednost iz klase ODELENJE i (2) y koja uzima vrednost iz kodomena preslikavanja ZAPOŠLJAVA za datu vrednost x. Iskaz SELEKTUJ definiše atribute od kojih se formira složeni objekat. Iskaz GDE_JE definiše uslov koji pojavljivanja objekata date klase treba, u rezultatu, da zadovolje. Podrazumeva se da su sve promenljive u gronjem izrazu univerzalno kvantifikovane, odnosno da se navedeni uslovi ispituju za svaku vrednost objektne promenljive i da se one koje zadovoljavaju uslove pojavljuju u rezultatu. U gornjem primeru se takodje podrazumeva da je definisana funkcija AVG koja sračunava srednju vrednost date kolekcije vrednosti.

Sintaksa izvodjenja objekata je:

```
izvodjenje_objekta  ::= lista_promenljivih izvodjenje;
lista_promenlivih  ::= def_prom  | def_prom, lista_promenljivih
def_prom  ::= naziv_promenljive : naziv_klase_objekata
izvodjenje  ::=SELEKTUJ lista_komponenti [GDE_JE ddf];
lista_komponenti  ::= komponenta |  komponenta, lista_komponenti
komponenta  ::= naziv_promenljive. naziv_atributa |
```

(izvodjenje)

Navedimo još nekoliko primera izvedenih objekata:

(ii) Prikaži imena studenata trećeg semestra, imena njihovih roditelja i marke kola koja voze.

```
x : STUDENT, y : x.KOLA, z : x.RODITELJ
SELEKTUJ x.IME, (SELEKTUJ z.IMER), (SELEKTUJ y.MARKA)
GDE_JE x. SEM = 3;
```

(iii) Prikaži imena vanrednih studenata petog semesta, njihova zanimanja i nazive projekata na kojima rade.

```
x : VANREDAN, y : x.PROJEKAT
SELEKTUJ x.IME, x. ZANIMANJE, (SELEKTUJ y.NAZPROJ)
GDE_JE x.SEM = 5;
```

(Napomena: Podtip nasledjuje sve atribute nadtipa)

(iv) Izlistaj imena studenata, nazive i ocene iz svih predmeta koje su položili za studenta trećeg semestra koji su položili matematiku sa ocenom većom od 8.

```
x : STUDENT, y : x.POLOŽIO, z :  y.PREDMET
SELEKTUJ x.IME, (SELEKTUJ y.OCENA, (SELEKTUJ z.NAZP))
GDE_JE x.SEM = 3 AND POSTOJI y (y.OCENA >8 AND z.NAZP=' Matem');
```

(v) Izlistaj imena studenata trećeg semestra i za one studente koji su položili matematiku sa ocenom većom od 8 nazive predmeta i ocene za sve predmete koje su položili.

```
x : STUDENT, y : x.POLOŽIO, z : y.PREDMET
SELEKTUJ x.IME, (SELEKTUJ y.OCENA, (SELEKTUJ z.NAZP)
GDE_JE y.OCENA  >8 AND z.NAZP = 'Matem' )
GDE_JE x.SEM = 3;
```

Zapazimo razliku izmedju rezultata izraza (iv) i (v). U rezultatu izraza (iv) pojaviće se samo studenti trećeg semestra koji su položili matematiku sa ocenom većom od 8, dok će se u rezultatu izraza (v) pojaviti svi studenti trećeg semestra, za one studente koji su položili matematiku sa ocenom većom od 8, pojaviće se lista svih predmeta koje su položili, dok će za sve druge atributi OCENA i NAZP imati nula vrednosti. To ukazuje na sledeće opšte karakteristike rezultata pri formiranju složenog objekta: **Ako uslov u "ugnježdenoj" SELEKTUJ naredbi nije zadovoljen, odgovarajući atributi dobijaju nula vrednost. Ako uslov u osnovnoj (spoljnoj) SELEKTUJ naredbi nisu zadovoljeni, odgovarajuće pojavljivanje objekta se ne prikazuje, odnosno ako nisu nikad zadovoljeni dobija se poruka "nijedno pojavljivanje nije formirano (nadjeno)"**.

Moguće je, kao i ranije, izjednačiti imena promenljivih sa imenima klase objekata i time preskočiti definiciju promenljivih. Izraz (i), u tom slučaju bi bio:

```
SELEKTUJ ODELENJE. SIFOD, ODELENJE. NAZIVOD,
  (SELEKTUJ ODELJENJE. ZAPOŠLJAVA. MLB, ODELENJE. ZAPOŠLJAVA. IMER,
       ODELENJE. ZAPOŠLJAVA. STAROST
                    GDE_JE ODELENJE. ZAPOŠLJAVA. STAROST > 40),
                    (SELEKTUJ  AVG (ODELENJE.ZAPOŠLJAVA, STAROST)
                     GDE_JE ODELENJE. ZAPOŠLJAVA. STAROST >40);
```

Da bi se izbeglo stalno ponavljanje naziva klasa može se uvesti u izraz SELEKTUJ i deo IZ koji govori o kojoj se klasi, odnosno promenljivoj radi. Za gornji primer to bi izgledalo:

```
SELEKTUJ SIFOD, NAZIVOD,
          (SELEKTUJ MLB, IMER, STAROST
         IZ ODELENJE. ZAPOŠLJAVA
                   GDE_JE STAROST >  40),
                    (SELEKTUJ AVG (STAROST)
                   IZ ODELENJE. ZAPOŠLJAVA
                   GDE_JE STAROST >  40)
 IZ ODELENJE;
```

Zbog toga što je poslednji način iskazivanja izvedenog objekta najsličniji standardnom relacionom upitnom jeziku SQL, on se preporučuje za korišćenje i nadalje će se, uglavnom, koristiti. Time se omogućuje da se i neke druge konvencije i mogućnosti koje postoje u SQL-u uvedu i ovde bez dodatnog objašnjenja. (Na primer * za oznaku da se preuzimaju svi atributi neke klase objekata.)

4.3.4. Definisanje pogleda

Pogled je izvedeni objekat koji se nadalje tretira kao i svaki drugi objekat u PMOV. Nad njim se mogu primenjivati sve operacije kao i nad ostalim objektima. Ako rezultat izvodjenja (i) želimo da "sačuvamo" kao pogled sa nazivom STARI_RADNICI_ODELENJA primenićemo sledeću definiciju:

```
    DEFINIŠI POGLED STARI_RADNICI_ODELENJA:
                   {<ŠIFRAOD, NAZODEL,  {<IMBROJ, IME, GODINE>}, PROSSTAR>}
KAO SELEKTUJ SIFOD, NAZIVOD,
                   (SELEKTUJ MLB, IMER, STAROST
                    IZ ODELENJE. ZAPOŠLJAVA
                              GDE_JE STAROST > 40),
                               (SELEKTUJ AVG (STAROST)
                              IZ ODELENJE. ZAPOŠLJAVA
                                        GDE_JE STAROST >  40)
IZ ODELENJE;
```

Pogled se definiše preko svoga imena, svoje strukture i izvodjenja. Izvodjenje se daje na isti način kao i ranije. Struktura pogleda se opisuje pomoću:

(1) {a} - konstruktora skupa pojavljivanja tipa a, odnosno konstruktora klase,

(2) <a, b, c,... > - konstruktora agregacije tipova a, b, c itd.

(3) [a , b , c].. - konstruktora ekskluzivne specijalizacije u podtipove a ili b ili c itd.

(4) /a,b,c/ ... - konstruktora ikluzivne specijalizacije u podtipove a i/ili b i/ili c.

Na primer, struktura koja bi trebalo da prikaže matične brojeve i imena gradjana i brojeve indeksa i semestre, ako su oni studenti, odnosno plate, ako su oni radnici (Slika 10), bi bila:

{<MLB, IME, [<BI, SEM>, PLATA]>}

Iz pogleda se mogu izvoditi drugi objekti preko ugnježdenih SELEKTUJ naredbi, kao na primer:

```
SELEKTUJ NAZODEL, (SELEKTUJ IME, GODINE
```

```
                                    GDE_JE GODINE < 50)
IZ STARI_RADNICI_ODELENJA
GDE_JE NAZODEL = 'Razvoj' OR NAZODEL = 'Prodaja' ;
```

4.4. Dinamička pravila integriteta

U prethodnim poglavljima definisane su različite vrste ograničenja, odnosno pravila integriteta. Ova pravila su statička, odnosno daju uslove koje podaci u bazi podataka treba da zadovolje u stacionarnom stanju, odnosno po okončanju bilo koje transakcije. Međutim, za potpunu specifikaciju baze podataka nekog IS, pored ograničenja, od interesa je i akcija koju treba preduzeti kada neka operacija ažuriranja BP naruši definisano ograničenje. Ovakva specifikacija daje se preko tzv "dinamičkih pravila integriteta". Jedno dinamičko pravilo integriteta čini trojka

<OPERACIJA, OGRANIČENJE, AKCIJA>

preko koje se, za svaku OPERACIJU održavanja BP i svako OGRANIČENJE koje ona može da naruši, definiše AKCIJA koju treba preduzeti ako je ograničenjne narušeno.

Kako smo izvršili podelu ograničenja na ograničenja na domene atributa, složena vrednosna ograničenjnja i strukturna ograničenja, na isti način možemo podeliti i dinamička pravila integriteta.

4.4.1. Dinamička pravila integriteta za atribute

Ograničenja na vrednosti atributa mogu narušiti operacije održavanja insert i update. Očigledno je da će se, bez obzira na to koja je od ovih operacija narušila ograničenje, preduzeti ista akcija. Zbog toga u definisanju dinamičkog pravila integriteta ove vrste nije neophodno navoditi operaciju. Ranije je pokazano da je najpraktičnije ograničenje na atribute prikazati preko tabele sa kolonama ATRIBUT, DOMEN, OGRANIČENJE. Za iskazivanje dinamičkog pravila integriteta za atribute dovoljno je ovu tabelu proširiti sa kolonom AKCIJA i formirati na primer sledeću tabelu:

ATRIBUT	DOMEN	OGRANIČENJE	AKCIJA
MLB	CHAR(13)	NOTNULL AND SUBSTRING(1,2) BETWEEN 1,31 AND SUBSTRING (3,4) BETWEEN 1,12	PORUKA: Nekorektan MLB
NAZIVJ	CHAR(15)	IN (SRPSKI, RUSKI, ENGLESKI, NEMAČKI)	PORUKA: Jezik ne postoji
ADRESA	CHAR(20)		
DATUM	DATE		
STAROST	INT(2)	BETWEEN 15,65	PORUKA: Starost van opsega
.....			.....

Kako je u ovoj vrsti dinamičkog pravila integriteta akcija, gotovo uvek, poruka sa odgovarajućim tekstom, ovu vrstu pravila obično nije neophodno ni navoditi, već je dovoljno zadati samo ograničenja na atribute, podrazumevajući da su ona ovako specificirana.

4.4.2. Složena vrednosna dinamička pravila integriteta

Kao što je ranije rečeno složena vrednosna ograničenja definišu međuzavisnosti vrednosti različitih atributa i specifikuju se pomoću iskaza "ojektnog računa". Za svako vrednosno ograničenje i listu operacija koje ga mogu narušiti definiše se akcija koju treba preduzeti kada je oganičenje narušeno. Sitaksa iskaza vrednosnog dinamičkog pravila integriteta je:

zapeta_lista operacija održavanja ":" vrednosno ograničenje
OTHERWISE akcija

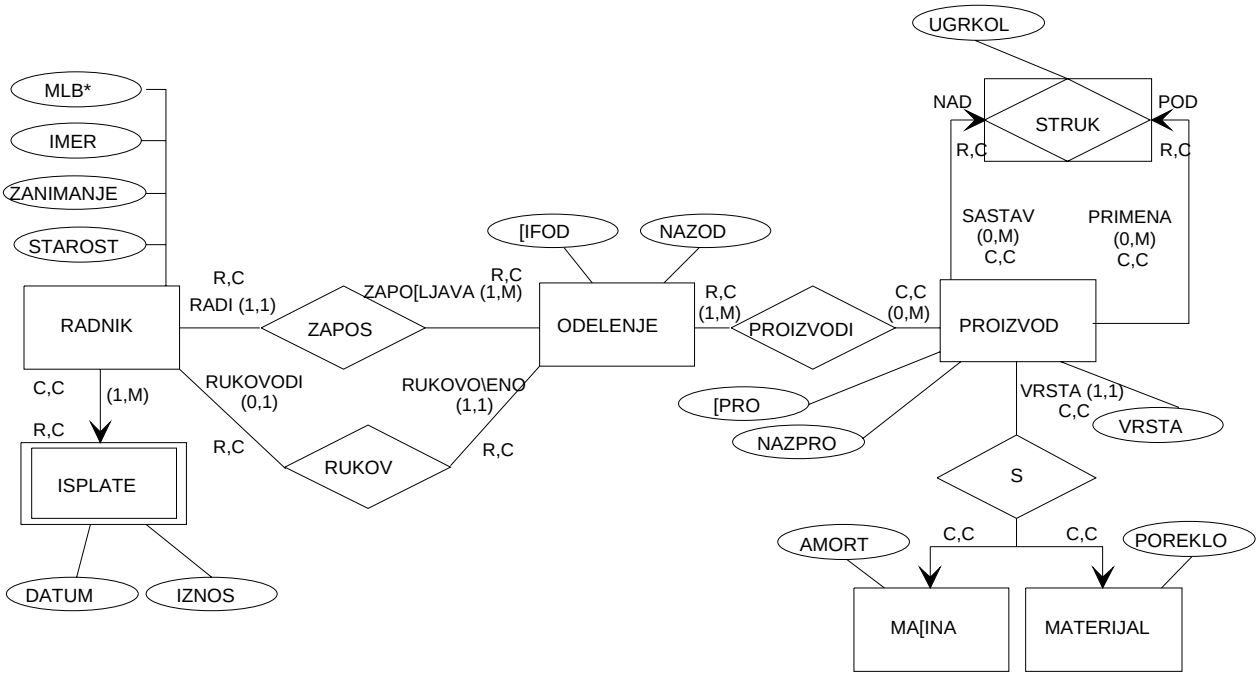
gde zapeta_lista operacija održavanja predstavlaj listu operacija održavanja razmaknutih zapetama za koje narušavanje datog ograničenja rezultuje u istu akciju.

Pretpostavimo da je za model na slici 16 definisano ograničenje: radnici koji rade u odelenju "Razvoj" ne mogu biti stariji od 45 godina. Formalni iskaz ovog ograničenja je

OGR1(RADNIK,RADI) := FOREACH RADNIK (IF EXISTS
RADNIK(RADNIK.RAD.NAZOD = 'Razvoj') THEN RADNIK.STAROST <= 45);

Operacije koje mogu da naruše ovo ograničenje su insert RADNIK, update RADNIK, update ODELENJE, connect ZAPOŠLJAVA, pa se dinamičko pravilo integriteta definiše kao

insert RADNIK, update RADNIK, update ODELENJE, connect ZAPOŠLJAVA :
OGR1 OTHERWISE Prikaži 'Radnik u Razvoju striji od 45 godina,
Odbi operaciju



Slika 16. Primer DOV sa upisanim strukturnim dinamičkim pravilima integriteta

4.4.3. Strukurna dinamička pravila integriteta.

Strukturna ograničenja u PMOV definisana su samom strukturom modela, odnosno načinom međusobnog povezivanja različitih klasa objekata. Operacije održavanja baze podataka mogu da naruše strukturna ograničenja narušavajući kardinalnost preslikavanja preko kojih su ostvarene veze između klasa objekata. Pri definiciji semantike pojedinih operacija održavanja baze podataka predpostavljeno je sledeće:

- Operacija insert poziva automatsko izvršavanje operacije connect za sva obavezna preslikavanja (preslikavanja sa DG > 0) koja poseduje posmatrana klasa;

- Operacija delete poziva automatsko izvršavanje operacije disconnect za sva preslikavanja date klase;

- Operacija recconect se svodi na uzastopnu primenu operacija disconnect i connect, u jednoj atomskoj transakciji;

- Operacije connect i disconnect na jednom preslikavanju podrazumevaju da se odgovarajuća operacija obavi i na njemu inverznom preslikavanju;

- Pretpostavlja se da se operacija update neće primenjivati na identifikator objekta;

Imajući ovo u vidu. očigledno je da strukturna ograničenja mogu narušiti samo operacije connect i disconnect, bilo da se neposredno primenjuju, bilo da su automatski idukovane od drugih operacija, pa je samo za ove operacije treba definisati dinamička strukturna pravila integriteta.

Analizirajmo načine na koji ove operacije mogu da naruše strukturno pravilo integriteta i akcije koje se u tim sličajevima mogu preduzeti.

Operacija

connect naziv_preslikavanja(id_domena:vrednost,
id_kodomena:vrednost)

na primer

connect RUKOVODI (MLB:307, ŠIFOD:03)

narušava integritet ako u klasi objeketa koja pretstavlja kodomen preslikavanja ne postoji navedeno pojavljivanje. Pretpostavlja se da pojavljivanje klase domena mora uvek da postoji da bi se ova opercija uopšte izvršila.

Akcije koje se u ovom slučaju mogu da preduzmu su:

(i) Operacija se odbija jer ne postoji objekat kodomen. Ovu opciju ćemo nazvati RESTRICTED;

(ii) Umesto sa navedenim objektom kodomena, ako to pojavljivanje ne postoji, povezivanje se izvršava sa specijalnim pojavljivanje kodomena, tzv "nula objektom", čije je značenje "nepoznati objekat". Pretpostavlja se da je u ovom slučaju definisano pojavlivanje "nula objekta" u klasi kodomena. Ovu opciju nazvaćemo NULLIFIES.

(iii) Umesto sa navedenim objektom kodomena, ako to pojavljivanje ne postoji, povezivanje se izvršava sa "default" objektom kodomena. Pretpostavlja se da je default objekat kodomena definisan. Ovu opciju zadovoljavanja integriteta nazvaćemo DEFAULT.

(iv) Ako objekat kodomena sa kojim dati objekat domena treba da se poveže ne postoji moguće ga je prvo kreirati operacijom insert, pa zatim izvršiti odgovarajuće povezivanje. Ovu opciju zadovoljavanja

integriteta nazvaćemo CASCADES. Očigledno je da ova opcija može da okine (trigger) čitavu kaskadu opercija u bazi podataka. Ubacivanje nepostojećeg pojavljivanja u klasu kodomena povlači automatski operacije connect za sva njegova obavezna preslikavanja, a ove preko svojih mogućih opcija zadovoljenja integiteta CASCADES ubacivanje objekata u odgovarajuće klase kodomena i tako dalje.

Imajući u vidu ove opcije, sintaksa za iskazivanje dinamičkog strukturnog pravila integriteta je

connect naziv_preslikavanja opcija
opcija :: RESTRICTED | NULLIFIES |
DEFAULT | CASCADES

Na primer,

connect RUKOVODI RESTRICTED {Ako vrednost argumenata kodomena ne postoji u klasi ODELENJE operacija se odbija}

connect RADI CASCADES { Operacija connect RADI ne može se direktno izvršavati jer ovo preslikavanje ima kardinalnosti DG = 1 i GG = 1, odnosno svaki radnik u bazi je već vezan za neko odelenje i može biti vezan samo za jedno odelenje. Međutim, ova operacija se automatski poziva pri izvršenju operacije

insert RADNIK (MLB:mmm;IMER:iiiii, ZANIMANJE:z, STAROST:35, ŠIFOD:xx)

Ako odelenje sa ŠIFOD = xx ne postoji u klasi ODELENJE ono se kreira i izvrši povezivanje preslikavanja RADI}

Pri izvršavanju operacije connect podrazumeva se da se ne narušava gornja granica karinalnosti odgovarajućeg preslikavanja, odnosno operacija connect uvek se odbija ako narušava gornju granicu kardinalnosti preslikavanja.

Operacija disconnect je još interesantnija.

disconnect naziv_preslikavanja(id_domena:vrednost,
id_kodomena:vrednost)

na primer

disconnect RUKOVODI (MLB:307, ŠIFOD:03)

Operacija disconnect se može direktno izvršavati samo kod preslikavanja kod kojih je donja granica kardinalnosti različita od gornje. Ako se ne može izvršavati direktno ova operacija može biti automatski pozvana od operacije delete iz klase domena, disconnect inverznog preslikavanja ili operacije recconect. Ova operacija raskida vezu objekta domena sa nekim kodomenom i na taj način, ako je posmatrano preslikavanje obavezno, narušava odgovarajuće strukturno ograničenje. Očigledno je da je operacija disconnect kritična za ona preslikavanja čija je donja granica kardinalnosti DG = 1 (ili preciznije DG > 0). U tom slučaju pojavljivanje u klasi domena ostaje bez svog obaveznog preslikavanja, pa se može primeniti, ponovo jedna od gore navedenih opcija:

(i) RESTRICTED - odbija se opreacija,

(ii) NULLIFIES - razvezuje se pojavljivanje objekta domena od navedenog pojavljivanja objekta kodomena i vezuje za "nula objekat" kodomena,

(iii) DEFAULT - razvezuje se pojavljivanje objekta domena od navedenog pojavljivanja objekta kodomena i vezuje za "default" objekat kodomena,

(iv) CASCADES - izbacuje se pojavljivanje objekta domena koji posle operacije disconnect ostaje da "visi".

Sintaksa za iskazivanje dinamičkog strukturnog pravila integriteta za operaciju disconnect je ista kao i za connect

```
disconnect naziv_preslikavanja opcija
opcija :: RESTRICTED | NULLIFIES |
        DEFAULT | CASCADES
```

Analizirajmo najčešće opcije RESTRICTED i CASCADES za nekoliko primera. Na Slici 17a pretstavljen je jedan model i data sledeća strukturna pravila integriteta:

disconnect AB RESTRICTED; disconnect BA CASCADES

Ni jedna od ovih operacija ne može se izvršavati direktno (zbog DG = GG = 1). One se automatski pozivaju pri izvršenju operacija delete A i delete B, respektivno. Prikazaćemo uopšteno algoritme izvođenja ovih operacija, pri zadatim dinamičkim pravilima integriteta.
Izbacivanje A:

delete A (KA:ka1)

disconnect AB (KA:ka1,KB:kb1) {Pozivaju se automatski operacije razvezivanja svih preslikavanja izbačenog objekta. Bez obzira na uslov RESTRICTED operacija se ne odbija, jer je pojavljivanje domena ka1 već izbačeno}
disconnect BA (KB:kb1,KA:ka1) {Automatski se poziva razvezivanje inverznog preslikavanja}
delete B (KB:kb1) { Zbog uslova CASCADES izbacuje se pojavljivanje klase B koja posle prethodne opercije ostaje da visi}

Izbacivanje B:

delete B (KB:kb1)
disconnect BA (KB:kb1,KA:ka1)
disconnect AB (KA:ka1,KB:kb1)
ROLLBACK {Zbog uslova RESTRICTED na preslikavanju AB, ne dozvoljava se operacija disconnect AB, odnosno poništavaju se efekti svih operacija sa naredbom ROLLBACK, pa se ne može izvršiti ni delete B}.

Na sličan nači postupa se i u situacijama gde je kardinalnost preslikavanja takva da je GG > DG, a DG = 1 (odnosno >0) (Slika 17b)

Pretpostavima da su u prvom slučaju zadati sledeći uslovi dinamičkog integriteta

disconnect AB RESTRICTED; disconnect BA CASCADES

Izbacivanje A

delete A (KA:ka1)
disconnect AB (KA:ka1,KB:kb1)
disconnect BA (KA:kb1,KA:ka1)
IF pojavljivanje sa ključem ka1 je poslednje u preslikavanju BA za
pojavljivanje B sa ključem kb1

THEN delete B (KB:kb1).

Razvezivanje BA

disconnect BA (KB:kb1,KA:ka1)
IF pojavljivanje sa ključem ka1 je poslednje u preslikavanju BA za
pojavljivanje B sa ključem kb1
THEN delete B (KB:kb1)
disconnect AB (KA:ka1,KB:kb1)
ROLLBACK.

Izbacivanje B

delete B(KB:kb1)
FOREACH pojavljivanje klase A povezano sa kb1 DO
disconnect BA (KB:kb1,KA:ka1)
disconnect AB (KA:ka1,KB:kb1)
END DO
ROLLBACK.

Sa ovako definisanim pravilima integriteta, zbog uslova RESTRICTED na preslikavanju AB, ne može se direktno izvršiti nikada ni operacija delete B ni disconnect BA. Ove dve operacije se mogu izvršiti samo u kaskadi opercija pri delete A.

Očigledno je da je drugačija situacija ako se uslovi dinamičkog integriteta zadaju kao

disconnect AB CASCADES; disconnect BA RESTRICTED

Izbacivanje A

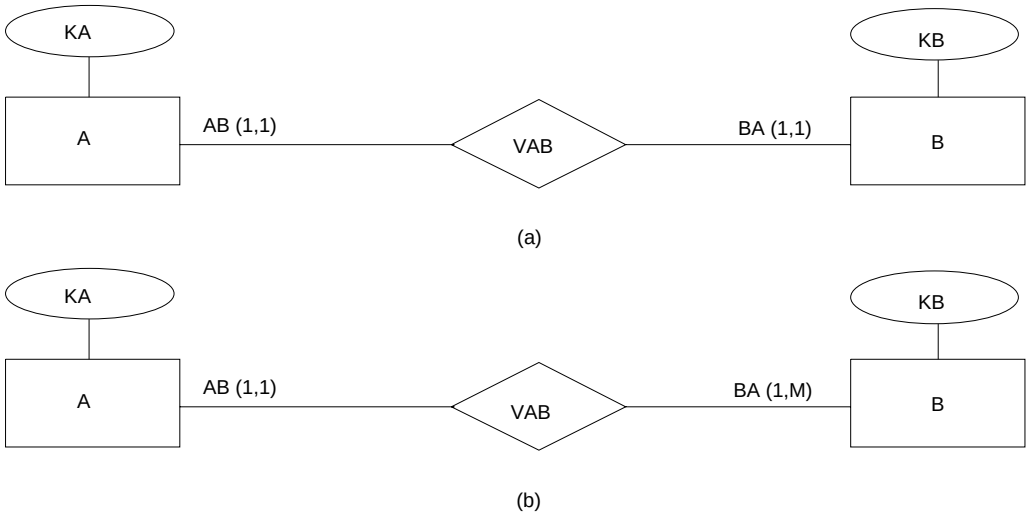
delete A (KA:ka1)
disconnect AB (KA:ka1,KB:kb1)
disconnect BA (KA:kb1,KA:ka1)
IF pojavljivanje sa ključem ka1 je poslednje u preslikavanju BA za
pojavljivanje B sa ključem kb1
THEN ROLLBACK.

Razvezivanje BA

disconnect BA (KB:kb1,KA:ka1)
IF pojavljivanje sa ključem ka1 je poslednje u preslikavanju BA za
pojavljivanje B sa ključem kb1
THEN ROLLBACK
ELSE disconnect AB (KA:ka1,KB:kb1)
delete A(KA:ka1)

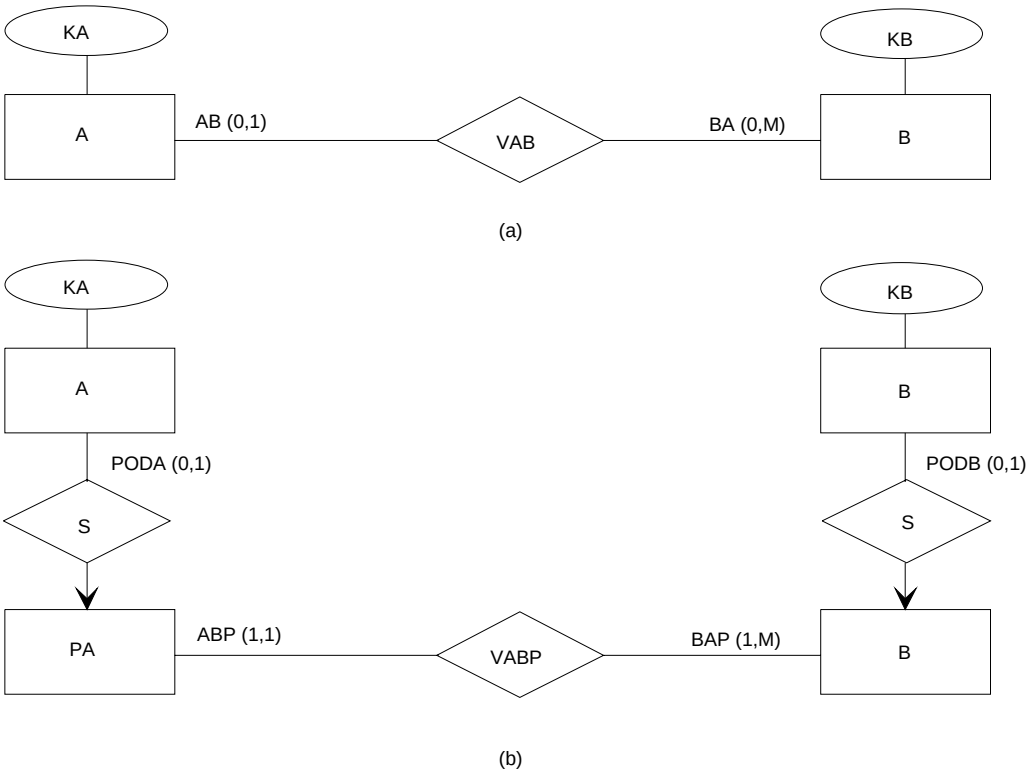
Izbacivanje B

delete B(KB:kb1)
FOREACH pojavljivanje klase A povezano sa kb1 DO
disconnect BA (KB:kb1,KA:ka1)
END DO
delete A(KA:ka1)



Slika 17. Primer za iskazivanje pravila dinamičkog strukturnog integriteta.

Do sada smo analizirali kako operacija disconnect narušava integritet i koje se akcije u tom slučaju mogu izvršiti. Posmatrajmo model na Slici 18a. Operacije razvezivanja preslikavanja AB i BA ne narušavaju strukturna ograničenja jer su za oba preslikavanja DG = 0. Međutim, isti model se može pretstaviti i slikom 18b, gde su kao podipovi izdvojena ona pojavljivanja klasa A i B koja su međusobno povezana. Zbog toga preslikavanja ABP i BAP imaju DG = 1, pa njihova razvezivanja mogu da dovedu do narušavanja integriteta. Definisanje dinamičkih pravila integriteta na ovim preslikavanjima proširuje semantičke mogućnosti navedenih opcija. Naime, na ovaj način ne samo da se iskazuje šta treba uraditi sa pojavlivanjem neke klase koje je moralo da bude povezano, a razvezano je, već i šta treba da se uradi sa pojavljivanjem koje nije moralo da bude povezano, bilo je povezano, a razvezano je.



Slika 18. Primer za iskazivanje pravila dinamičkog strukturnog integriteta za preslikavanja sa DG = 0.

Očigledno je da se operacija disconnect AB sa modela (a) svodi na operaciju delete PA koja dalje okida operacije disconnect ABP, disconnect BAP i eventualno delete PB u zavisnosti od definisanog pravila dinamičkog integriteta, a operacija disconnect BA na kaskadu operacija koju prouzrokuje delete PB.

Na primer ako se specifikuje

disconnect ABP CASCADES; disconnect BAP RESTRICTED

tada imamo sledeće načine odvijanja različitih operacija

Razvezivanje AB (disconnect AB(KA:ka1,KB:kb1)

delete PA (KA:ka1)
diconnect ABP (KA:ka1,KB:kb1)
disconnect BAP(KB:kb1,KA:ka1)
IF pojavljivanje sa ključem ka1 je poslednje u preslikavanju BAP za
pojavljivanje PB sa ključem kb1
THEN ROLLBACK

U odnosu na model na Slici 18a tumačenje gornjih pravila dinamičkog integriteta je:

- ako je neko pojavljivanje klase A jedino pojavljivanje u preslikavanju BA tada se ne može izvršiti operacija disconnect AB, bez obzira što ne narušava integritet.

Razvezivanje BA (disconnect BA(KB:kb1,KA:ka1)

delete PB (KB:kb1)
disconnect BAP(KB:kb1,KA:ka1)
IF pojavljivanje sa ključem ka1 je poslednje u preslikavanju BAP za
pojavljivanje PB sa ključem kb1
THEN ROLLBACK
ELSE delete PA(KA:ka1)

Očigledno je da se za iskazivanja dinamičkih pravila integriteta kod preslikavanja kod kojih operacija disconnect ne narušava integritet ne mora izvršiti transformacija sa Slike 18a u Sliku 18b, već je moguće na Slici 18a direktno zadati ova pravila. Slika 18b poslužila je samo da se objasni semantika opcija kada operacija generalno ne narušava strukturni ntegritet.

Iz prethodne diskusije je očigledno da se dinamička strukturna pravila integriteta zadaju za svko preslikavanje na modelu i to posebno za operaciju connect, a posebno za operaciju disconnect. Ako se uvedu skraćenice za opcije R - RESTRICTED, N - NULLIFIES, D - DEFAULT i C - CASCADES i ako se pretpostavi da prvo slovo u paru zanači opciju za operaciju connect, a drugo za disconnect, tada se opcije zadovoljavanja dinamičkih pravila strukturnog integriteta mogu zadati na Dijagramima objekti veze, kako je to na Slici 16 i prikazano.

5. METODOLOŠKI ASPEKTI MODELIRANJA

Model objekti-veze predstavlja samo "intelektualni alat" za definisanje modela podataka realnog sistema, odnosno za opisivanje skupa povezanih objekata realnog sistema i dogadjaja u njemu preko jednog složenog apstraktnog tipa podatka (strukture podataka, ograničenja i operacija). Dok sam "intelektualni alat" može biti manje ili više formalan, postupak modeliranja realnog sistema je uvek

"veština" ("umetnost"), zavisi od sposobnosti, znanja i iskustva analitičara i ne mogu se dati strogo formalna pravila modeliranja koja bi vodila do jedinstvenog modela složenog realnog sistema, bez obzira na to ko modeliranje vrši. Mogu se dati samo opšte metodološke preporuke, opšti metodološki pristupi, kao pomoć u ovom složenom poslu.

U osnovi postoje tri osnovna pristupa zasnovana na konceptu apstrakcije: (1) integracija podmodela , (2) konkretizacija opštih (generičkih) modela i (3) "inverzno inženjerstvo".

5.1. Integracija podmodela

Integracija podmodela ("pogleda") je pristup "od dna na gore" (bottom-up). Predpostavlja se da je realni sistem već dekomponovan na podsisteme (na primer funkcionalnom dekompozicijom, pomoću Strukturne sistema analize). Za svaki podsistem (funkciju) formira se jedan podmodel objekti-veze, pa se zatim ovi podmodeli integrišu u jedan, integralni model objekti-veze.

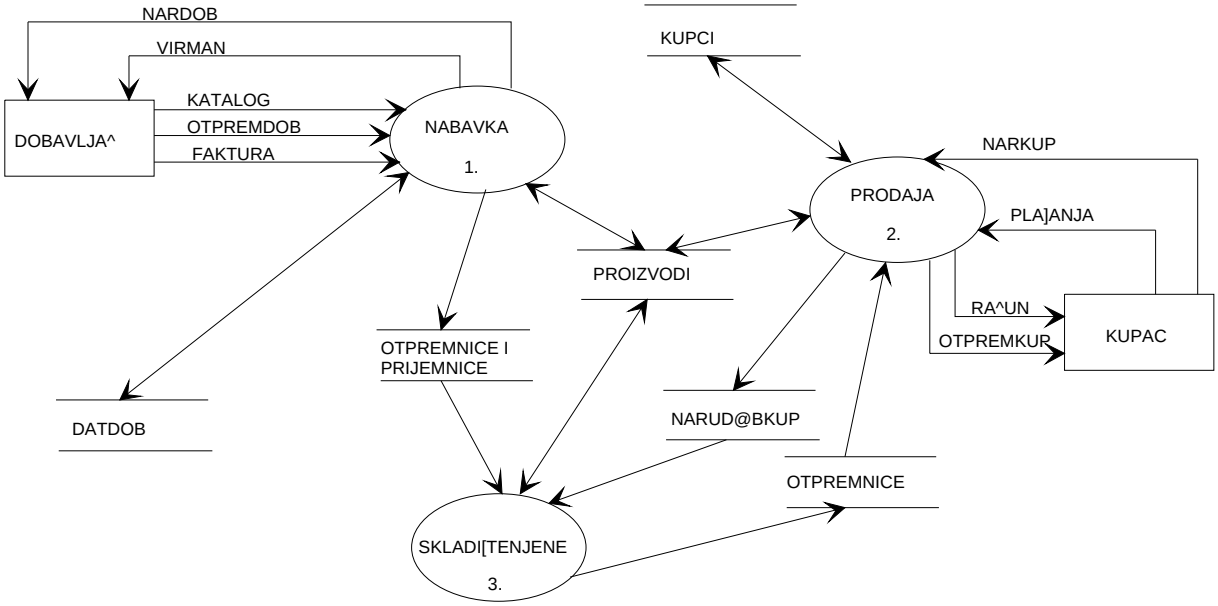
Tokovi i skladište podataka pretstavljaju složene (izvedene) objekte. Njihovom dekompozicijom ("analizom objekata") može se doći do baznih objekata i njihovih medjusobnih veza. Naime, model objekti veze za jedan skup dijagrama tokova podataka je minimalna realizacija strukture podataka iz koje se bilo koji tok ili skladište podataka može generisati.

Na slikama 19 - 25 prikazan je ovaj metodološki postupak za nalaženje modela objekti-veze za jedno trgovačko preduzeće koje nabavlja neke proizvode od dobavljača, skladišti ih, a zatim prodaje kupcima. Na slici 19 prikazan je dijagram prvog nivoa za ovaj sistem. Detaljni dijagrami za svaki od osnovnih procesa (NABAVKA, PRODAJA, SLADIŠTENJE) dati su na slikama 20, 21 i 22. Podmodel objekti veze za funkciju prodaje prikazan je na slici 23, a za funkciju nabavke na slici 24. Podmodel za funkciju skladištenja nije posebno prikazan zato što u sebi sadrži osnovne objekte sa prethodna dva, a jedino novo što donosi je atribut KOLSKLAD objekata MODEL. Na slici 25 prikazan je model objekti veze za ceo sistem, dobijen integracijom podmodela.

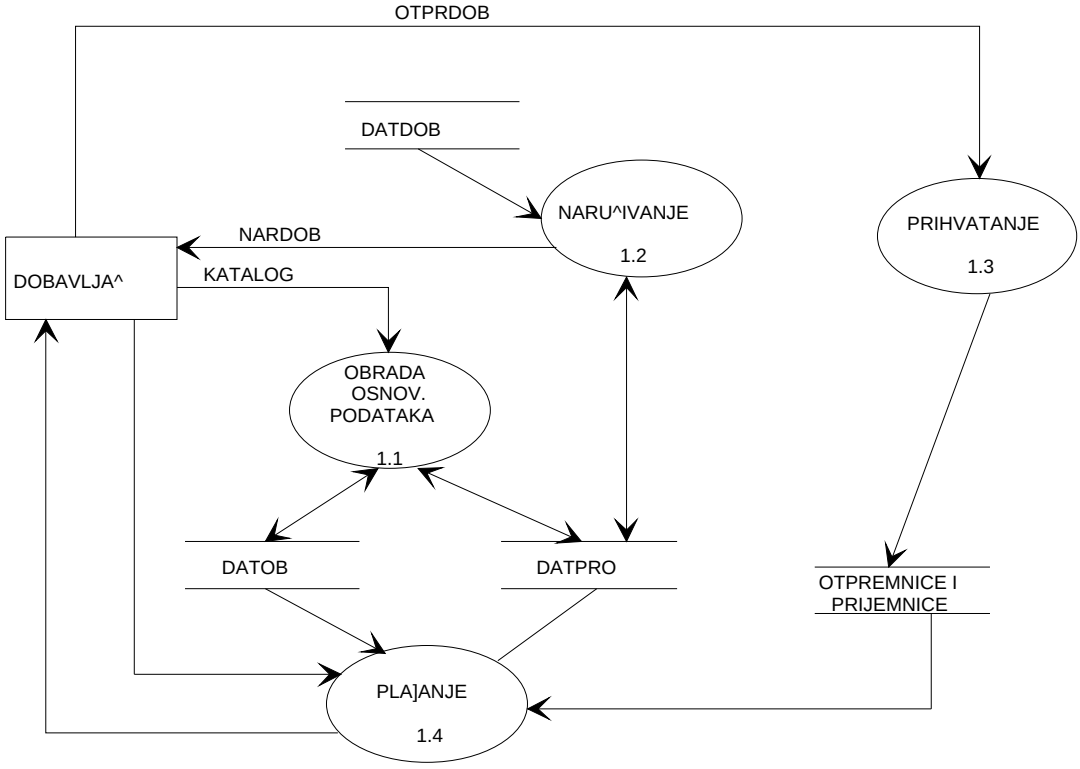
Pri postupku integracije podmodela moraju se razrešiti neke, moguće, njihove medjusobne nekonzistentnosti:

- Imenovanje koncepata, različita imena za isti koncept ili isto ime za različite koncepte. Jedini način razrešenja ovoga problema je vodjenje jedne vrste rečnika podataka sa definicijama (opisom) svih naziva u modelima.
- Semantičke nekonzistentnosti izmedju podmodela. Neophodno je da isti objekat na različitim podmodelima bude iste vrste (slab, podtip, agregacija, jezgro (objekat koji nije ni slab, ni podtip, ni agregacija)), da ista preslikavanja na različitim podmodelima imaju iste kardinalnosti. Takodje treba otkriti i sve semantički nekonzistentne "petlje", petlje u kojima, na primer, preko različitih veza neki objekat može da postane podtip sam sebi ili slab od samog sebe i slično.
- Redudantne veze. Potrebno je otkriti da li veze izmedju istih tipova objekata, po različitim putanjama imaju istu semantiku pa neke od ovih veza treba prekinuti.

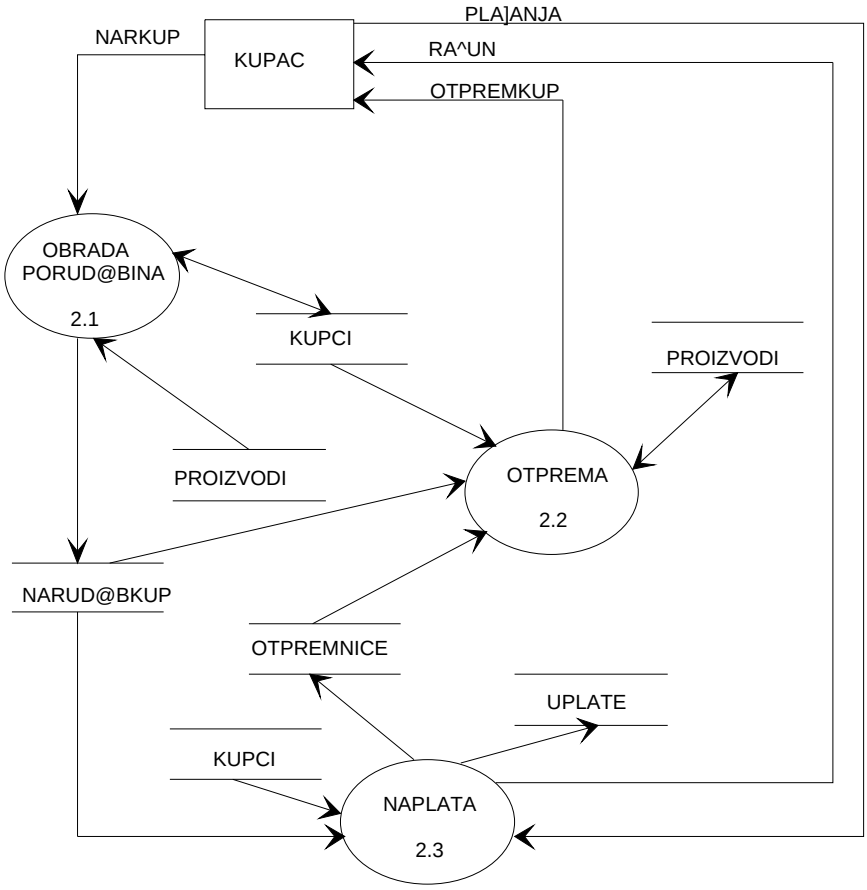
Sve je ovo relativno teško otkriti pri integraciji većeg broja podmodela. Zato se preporučuje postepena integracija i korišćenje CASE alata u kojima se neke nekonzistentnosti automatski otkrivaju.



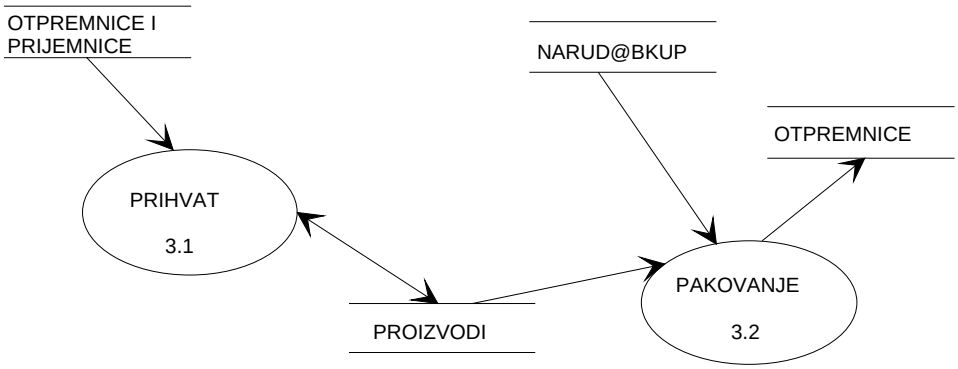
Slika 19. Dijagram prvog nivoa



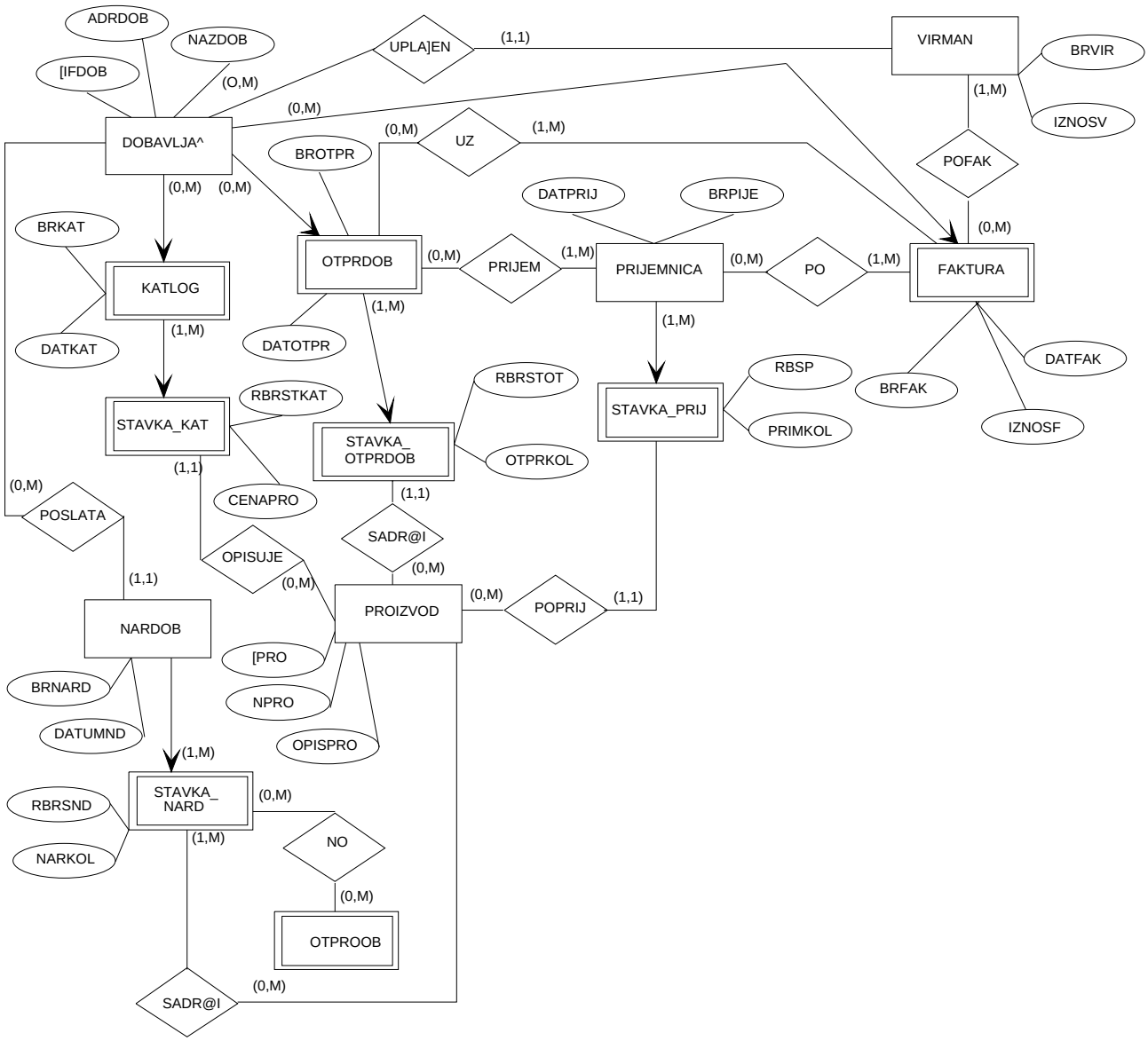
Slika 20. DTP 1: Funkcija NABAVKE



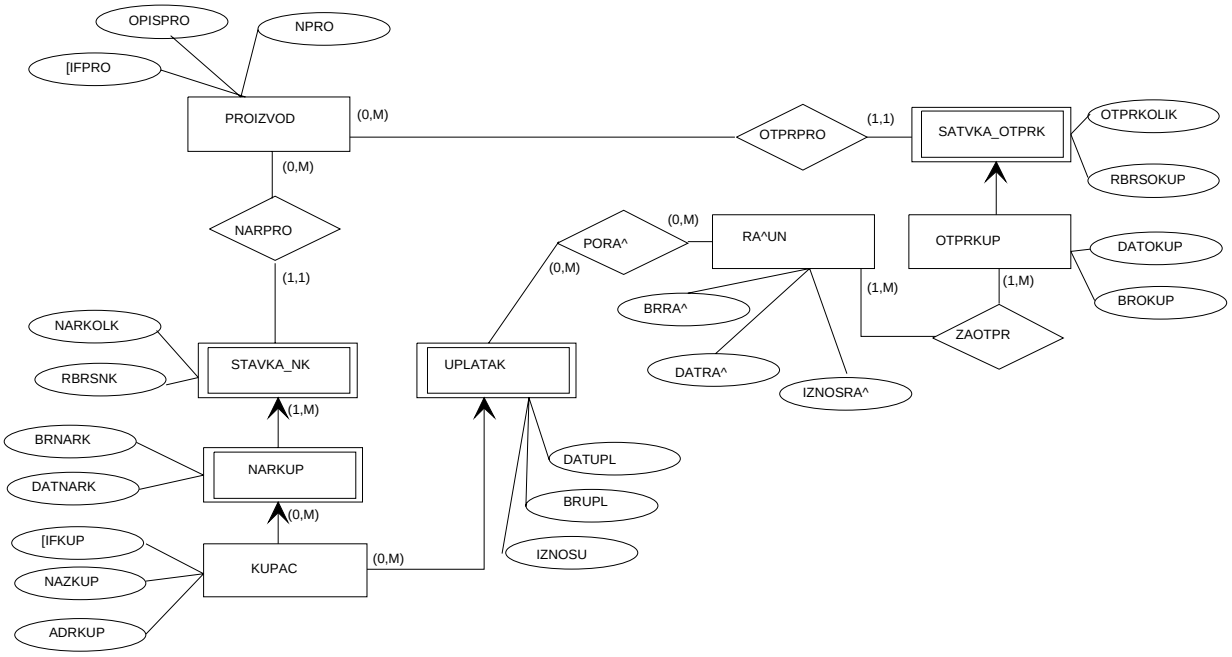
Slika 21. DTP 2: Funkcija PRODAJE



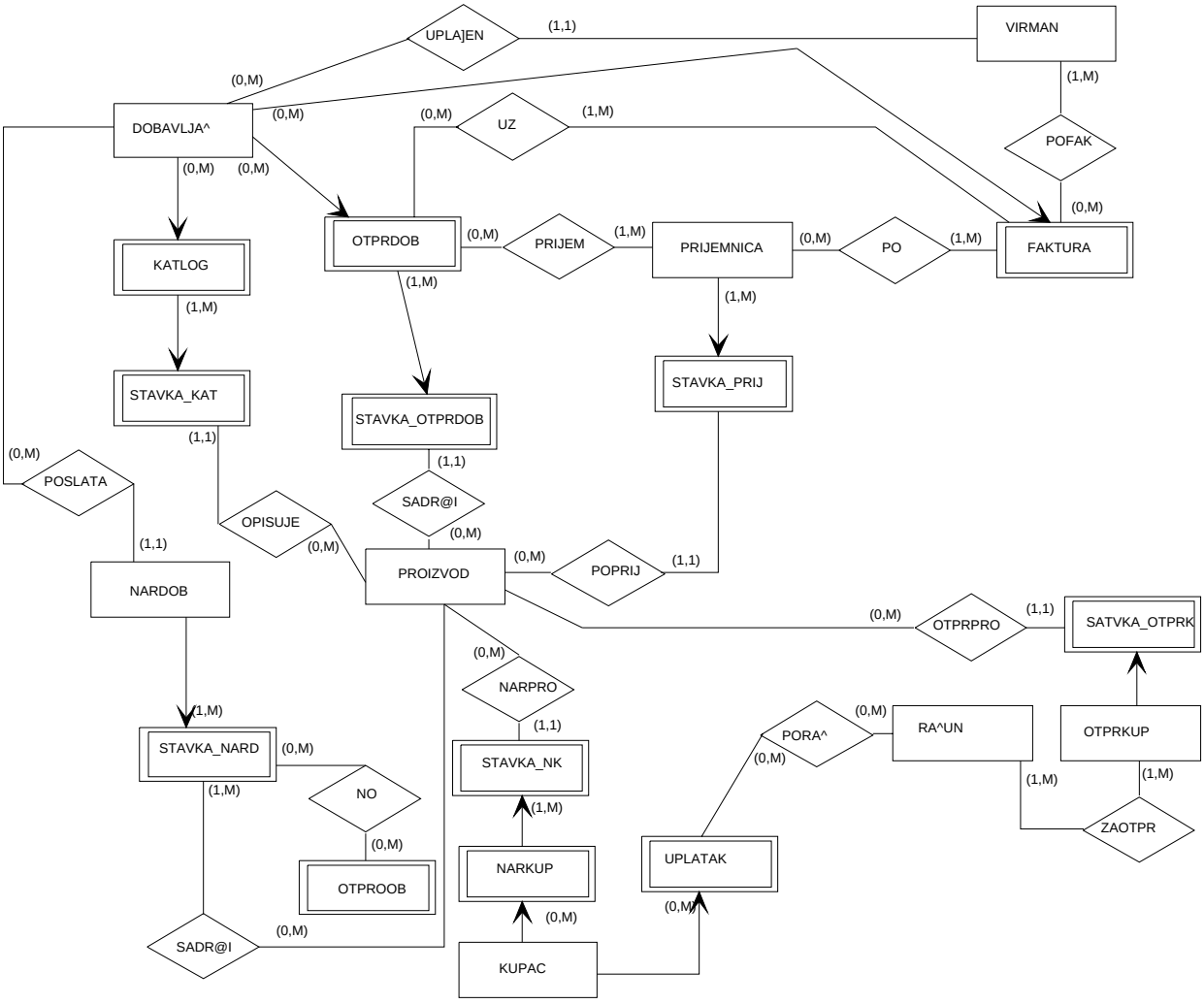
Slika 22. DTP 3: Funkcija SKLADIŠTENJE



Slika 23. DOV za funkciju NABAVKE



Slika 24. DOV za funkciju PRODAJE



Slika 25. Integrisani model objekti-veze

5.2. Generički modeli podataka

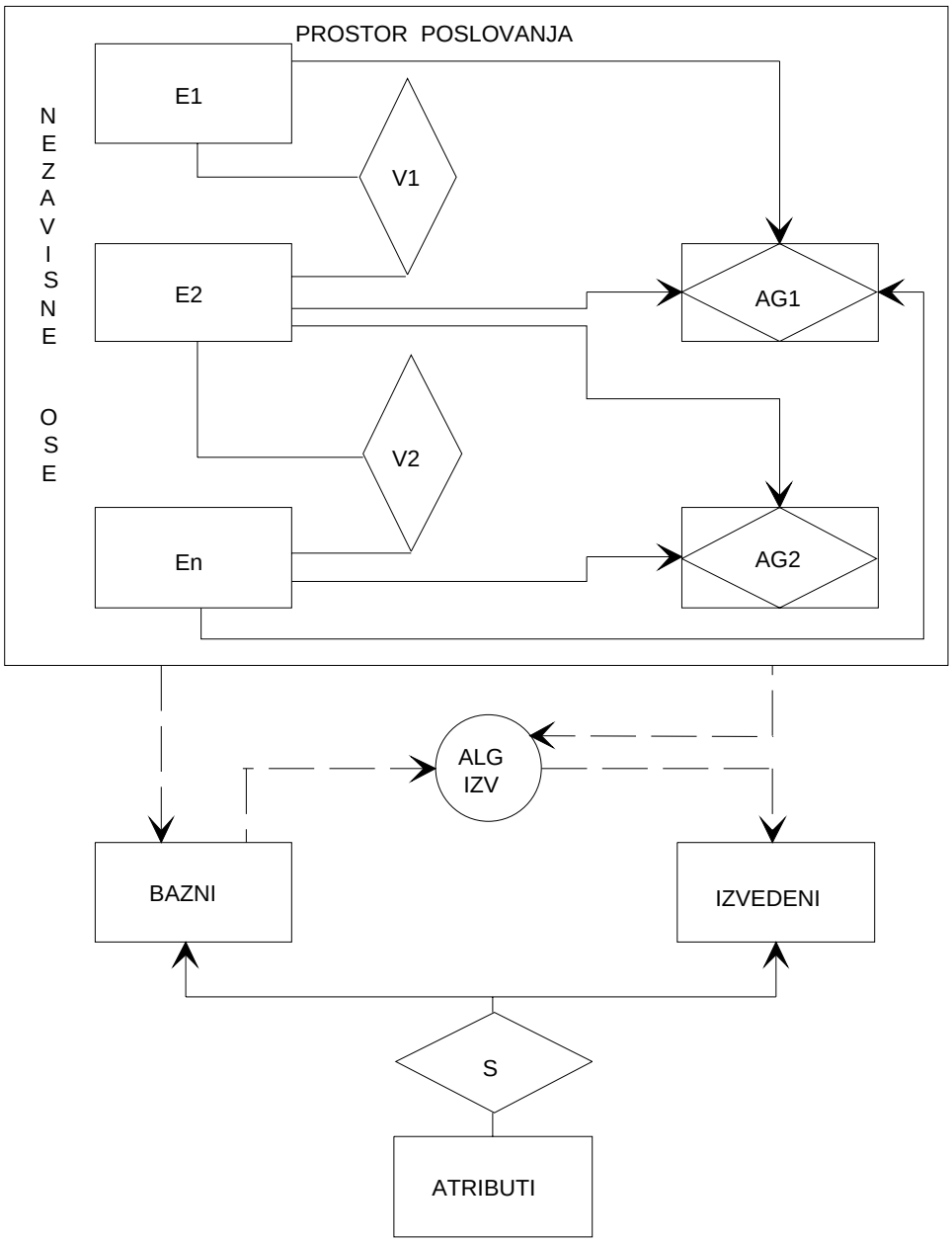
Ovaj pristup koristi činjenicu da su objekti ne samo "stabilni" u jednom konkretnom sistemu (stabilni u smislu da se bitnije ne menjaju čak i kada se funkcije sistema i zahtevi za informacijama u sistemu menjaju), već i u čitavoj klasi srodnih sistema. Ovde su od interesa prvenstveno poslovni sistemi, pa se prikaz ovoga pristupa odnosi na ovu klasu sistema. Čitav skup objekata visokog nivoa apstrakcije (generičkih objekata) je isti u različitim poslovnim sistemima, pa oni treba da posluže kao osnova za razvoj modela podataka.

Sa druge strane (Slika 26), u skupu objekata nekog poslovnog sistema postoji skup "nezavisnih" objekata, čije postojanje nije uslovljeno postojanjem nekog drugog objekata u sistemu. Tipovi takvih objekata se mogu tretirati kao nezavisno promenljive koordinantne ose, a njihova pojavljivanja kao tačke na tim osama, jednog apstraktnog n-dimenzionog prostora koga nazivamo "apstraktni prostor poslovanja". Pri tome se smatra da je pojavljivanje nekog objekta predstavljeno njegovim ključem (ili, još bolje surogatom ključa, identifikatorom koga dodeljuje sistem kad god se neko novo pojavljivanje kreira), atributi ovih objekata su funkcije definisane u odgovarajućim tačkama posmatrane koordinatne ose. Očigledno je da pojavljivanja agregacija, koje se predstavljaju Dekartovim proizvodom tipova (skupova) objekata odredjuju tačke u "prostoru poslovanja" (van koordinatnih osa) u kojima su definisane neke druge funkcije - atributi agregacija.

Bazni atributi, atributi koji se ne mogu izvesti iz drugih atributa u sistemu predstavljaju samo jedan skup informacija preko kojih se prati ponašanje poslovnog sistema. Iz njih se različitim obradama podataka mogu izvesti i mnoge druge. Osim izvodjenja koja se mogu vršiti nad skupovima ili podskupovima pojavljivanja osnovnih objekata i agregacija (prosečna plata radnika i druge agregatne funkcije, na primer), veze izmedju objekata u modelu definišu ostale tačke u "poslovnom prostoru" u kojima se ovakva izvodjenja mogu vršiti, a sami algoritmi obrade, funkcije izvodjenja.

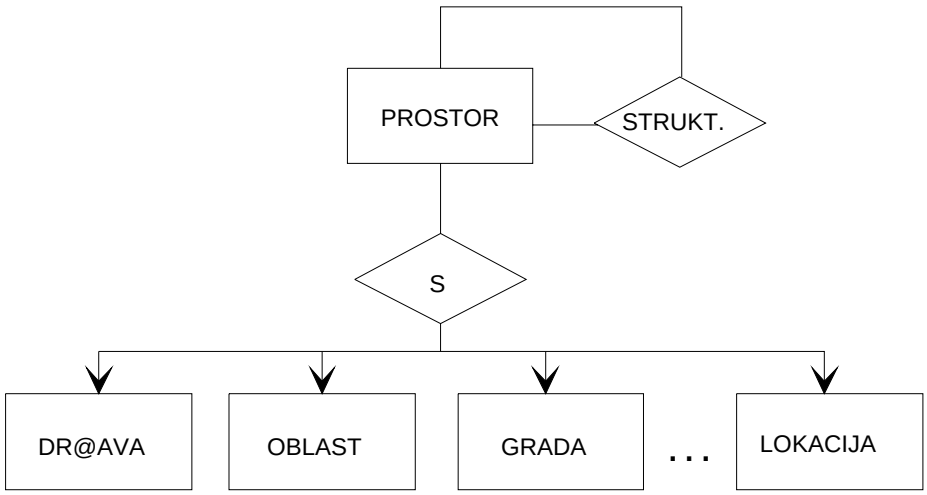
Imajući u vidu apstraktni model i neke generičke objekte poslovnih sistema prikazane na slikama 27-32, razvoj modela podataka odvija se kroz:

- Izbor generičkih objekata.
- Rasčlana generičkih objekata.
    - Specijalizacija, generisanje hijararhije podtipova.
    - Definisanje strukture objekata.
    - Definisanje baznih atributa objekata.
- Definisanje osnovnih agregacija i njihova rasčlana.
- Definisanje veza izmedju osnovnih objekata i agregacija.
- Analiza funkcija sistema (poslovnih procesa) sa ciljem da se otkriju dodatni objekti, agregacije, atributi i veze izvodjenja, odnosno da se izvrši dalja rasčlana sistema.

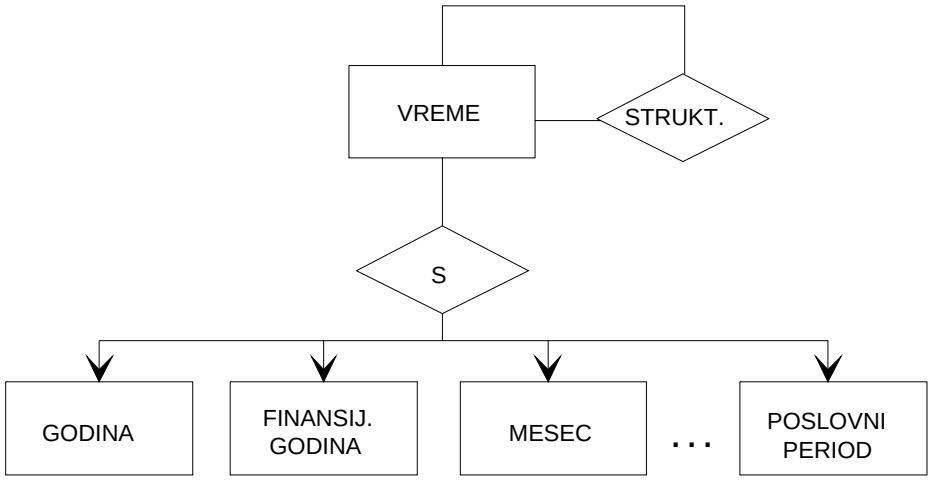


Slika 26. Generički model podataka

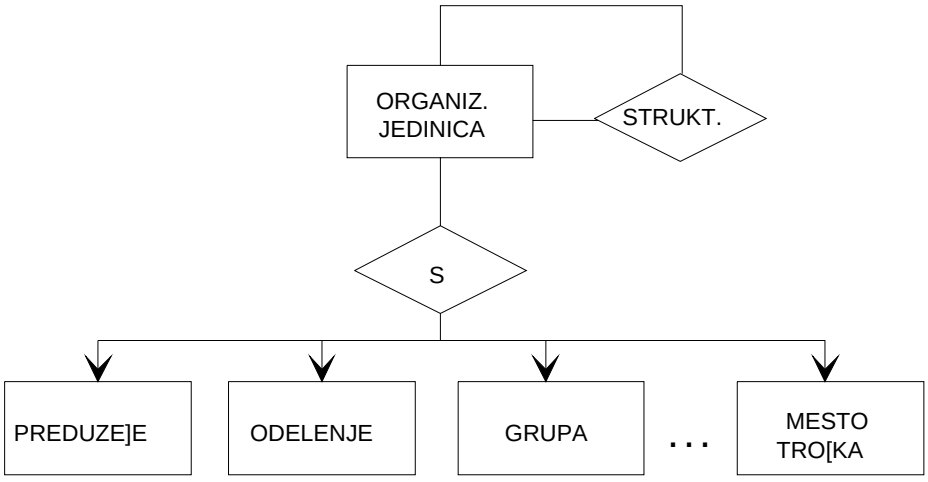
Ovakav pristup omogućuje da se na specifičan, struktuiran način izvede i analiza poslovnih procesa, kao i odnosa poslovnih procesa i objekata (klasa podataka), na osnovu čega se odredjuje opšta arhitektura informacionog istema (po metodi BSP, na primer). Naime, agregacije objekata relativno visokog nivoa generalizacije, predstavljaju složene objekte koji su glavni "objekti posmatranja" u nekom sistemu. Na primer, na Slici 33 predstavljena je jedna takva agregacija, NABAVKA. Analizom standardnih faza "životnog ciklusa" ovakvog složenog "objekta posmatranja" mogu se utvrditi i logički povezati odgovarajuće funkcije sistema i u njima otkriti nove, detaljnije komponente modela podataka.



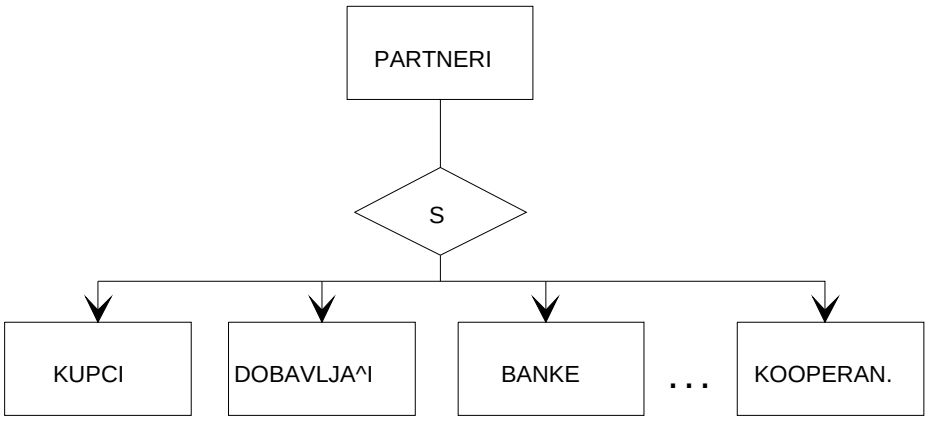
Slika 27. Geografska koordinata



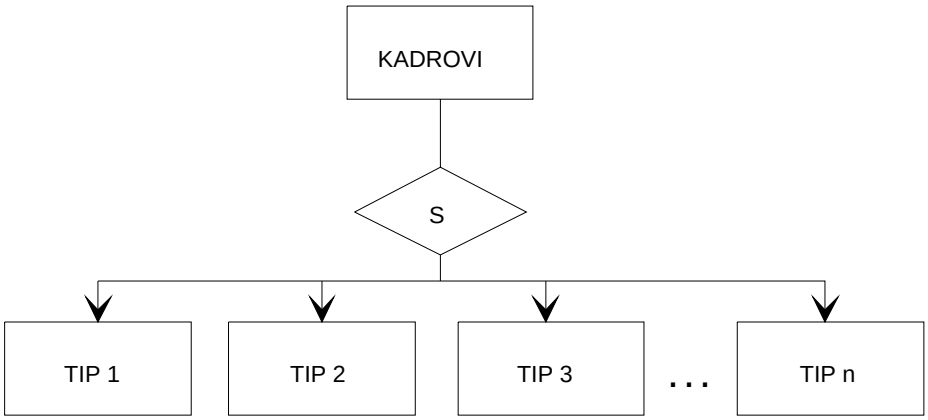
Slika 28. Vremenska Koordinata



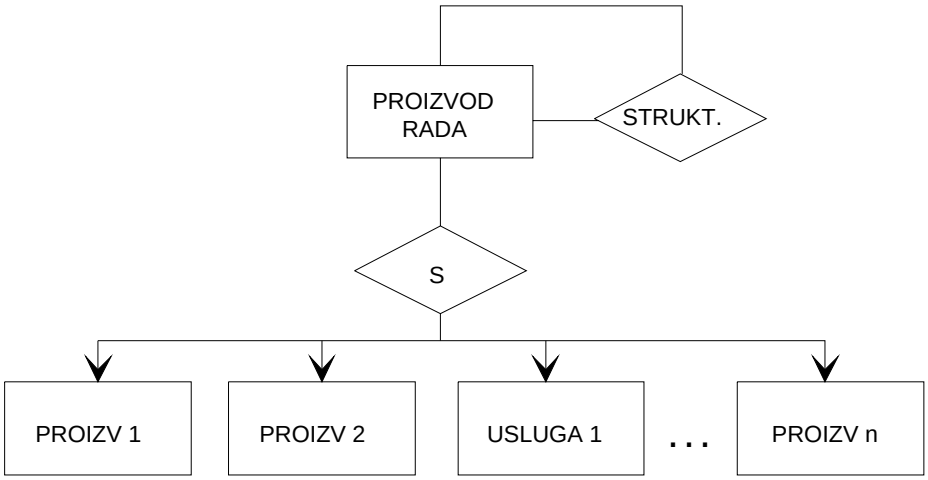
Slika 29. Koordinata "organizacije"



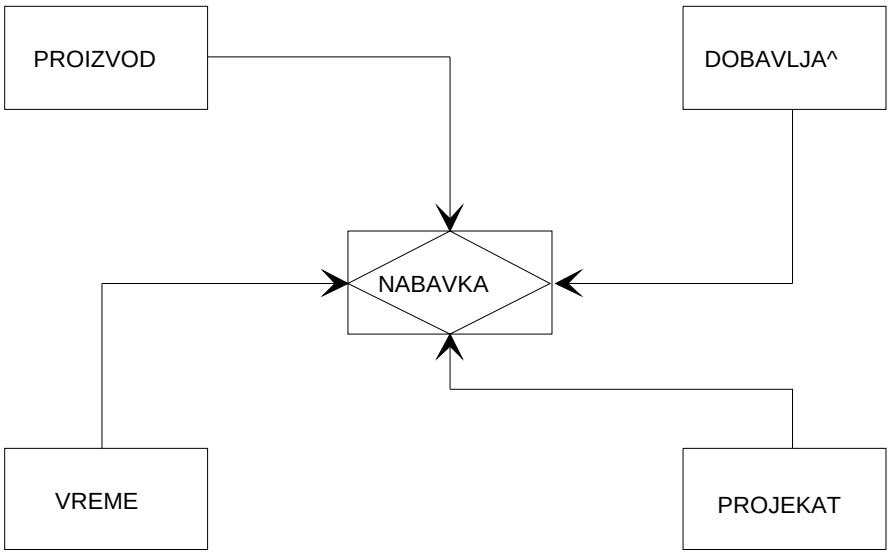
Slika 30. Koordinata "okoline"



Slika 31. Koordinata "kadrova"



Slika 32. Koordinata "proizvoda rada"



Slika 33. "Objekt posmatranja" NABAVKA

Za dati primer, funkcije bi okvirno bile:

FAZE "ŽIVOTNOG CIKLUSA"	POSLOVNI PROCESI
"Priprema radjanja"	Planiranje proizvodnje Istraživanje tržišta
"Radjanje"	Priprema nabavke (upiti za ponude), Ugovaranje
"Razvoj"	Naručivanje, Prihvatanje Plaćanje
"Nestanak"	Skladištenje

5.3. "Inverzno inženjerstvo"

Termin "inverzno inženjerstvo" (reverse engineering) se koristi za postupak kojim se od gotovog softverskog proizvoda formira njegova specifikacija . Na primer, u ovom slučaju od gotovog COBOL programa, ili skupa COBOL programa, model objekti-veze. Naime, dugogodišnjim razvojem nekih IS izradjeni su mnogi programi sa mnogim datotekama koji su u upotrebi, a za koje, osim izvornog koda , ne postoji druga dokumentacija, jer nisu ni razvijeni nekom standardnom metodologijom . Da bi sistem mogao dalje da se razvija korišćenjem standardne metodologije, neophodno je, prvo, napraviti standardnu specifikaciju dela sistema koji već radi, pa na osnovu nje nastaviti razvoj.

Očigledno da se model objekti-veze može delimično rekonstruisati iz DATA DIVISION COBOL programa. Svaki opis rekorda predstavlja jedan osnovni ili složeni tip objekta. Dekompozicijom složenih rekorda (na primer rekorda sa OCCURS) može se definisati jedan podmodel podataka. Veze izmedju objekta se ostvaruju preko istoimenih polja (ili grupa polja) u različitim rekordima. Medjutim, često se semantički isto polje u različitim rekordima različito naziva. Zbog toga je neophodno izvršiti i analizu PROCEDURE DIVISON i utvrditi gde se polja iz različitih rekorda medjusobno porede, kako ulaze u pojedina sračunavanja i slično, da bi se i na ovaj način utvrdile veze izmedju objekata.

Složenost ovakvog "inverznog inženjerstva" zavisi od strukture izvornog programa. U dobro struktruiranim programima, ovaj postupak može da bude relativno lak zadatak.

Inverzno inženjerstvo je često i poseban modul pojedinih CASE alata. Na ulazu ovog modula je izvorni program u nekom jeziku, a njihov izlaz je model objekti veze ili dijagram toka podataka za strukturunu sistem analizu.

Napomenimo, da se na ovaj način ne može dobiti potpun model objekti-veze. Naime, implementacija nekog softvera je semantički siromašnija od njegove specifikacije, jer predstavlja model podataka niže generacije, pa se potpuna semantika sistema iz nje ne može rekonstruisati.

5.4. Modeliranje podataka I faze razvoja IS

Modeli podataka se primenjuju u svim fazama razvoja IS:

(1) U fazi planiranja razvoja IS (idejni projekat) definiše se globalni model podataka koristeći odgovarajuće generičke modele i grubo date podmodele podataka za pojedine osnovne funkcije sistema. Ovaj model sadrži samo najznačajnije objekte i veze i samo osnovne atribute objekata, koji su neophodni da bi definicija objekata i veza bila jasnija.

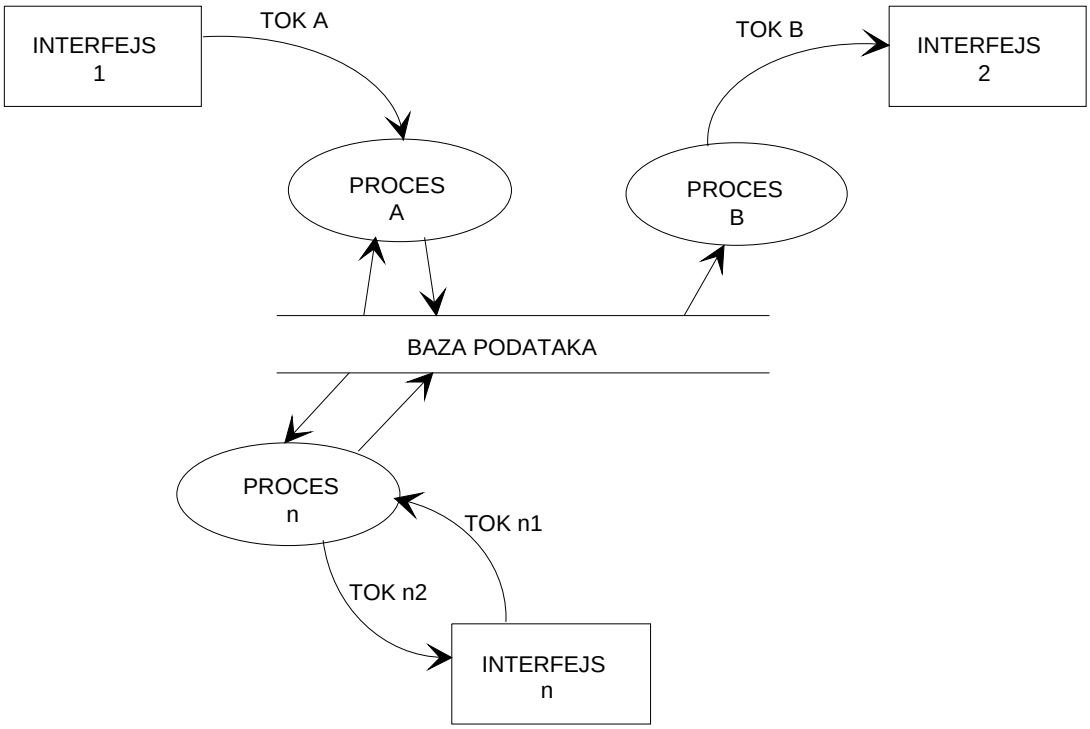
(2) U fazi detaljne specifikacije zahteva razradjuju se detaljni podmodeli za sve obuhvaćene podsisteme i funkcije u njima i oni se integrišu u jedinstveni model podataka sistema. Ovde se definišu svi atributi objekata i sva pravila integriteta.

(3) U fazi projektovanja i implementacije sistema model podataka je osnova za projektovanje baze podataka i aplikacija. Iz modela objekti-veze se formalno i automatizovano mogu definisati logička struktura baze podataka i programi za održavanje baze podataka.

(4) U fazi operativnog rada sistema model podataka, implementiran kao rečnik baze podataka, je osnova za održavanje sistema (izmene koje se zahtevaju).

6. SPECIFIKACIJA INFORMACIONOG SISTEMA POMOĆU PROŠIRENOG MODELA OBJEKTI-VEZE

Specifikaciju nekog IS radimo pomoću Strukturne sistemske analize i Proširenog modela objekti-veze. Pomoću PMOV specifikovali smo jedinstvenu bazu podataka IS. Preko SSA smo procese u sistemu hijerarhijski dekomponovali do primitivnih procesa. Imajući u vidu da procesi na DTP međusobno komuniciraju samo preko skladišta, a da PMOV definiše jedinstvenu bazu podataka na kojoj se zasniva budući IS, njegov fizički DTP se uopšteno može prikazati dijagramom na Slici 34. Na Slici 34 prikazani su samo primitivni procesi i oni pretstavljaju aplikacije budućeg sistema. Dijagrami dekompozicije koji višu hijerarhijsku strukturu procesa predstavljaju sad samo "menije", odnosno način izbora pojedinih aplikacija.



Slika 34. Fizički DTP IS zasnovanog na integrisanoj bazi podataka

Imajući u vidu opštu arhitekturu budućeg fizičkog sistema prikazanu na Slici 34, može se zaključiti da potpuna specifikacija budućeg IS obuhvata:

(i) Specifikaciju baze podataka
(ii) Specifikaciju aplikacija
(iii) Nefunkcionalne specifikacije

6.1. Specifikacija baze podataka

Kao što je ranije diskutovano, specifikacija baze podataka obuhvata:

(1) Dijagram proširenog modela objekti veze sa definisanim pravilima dinamičkog strukturnog integriteta (na primer dijagram na Slici 16)

(2) Definiciju svih atributa i domena sa ograničenjima i eventualno akcijama koje se pozivaju pri narušavanju integriteta. Na primer za model na Slici 7, deo tabele za specifikaciju atributa izgleda

ATRIBUT	DOMEN	OGRANIČENJE	AKCIJA
MLB	CHAR(13)	NOTNULL AND SUBSTRING(1,2) BETWEEN 1,31 AND SUBSTRING (3,4) BETWEEN 1,12	PPORUKA: Nekorektan MLB
NAZIVJ	CHAR(15)	IN (SRPSKI, RUSKI, ENGLESKI, NEMAČKI)	PORUKA: Jezik ne postoji
ADRESA	CHAR(20)		
DATUM	DATE		
STAROST	INT(2)	BETWEEN 15,65	PORUKA: Starost

..... van opsega .....

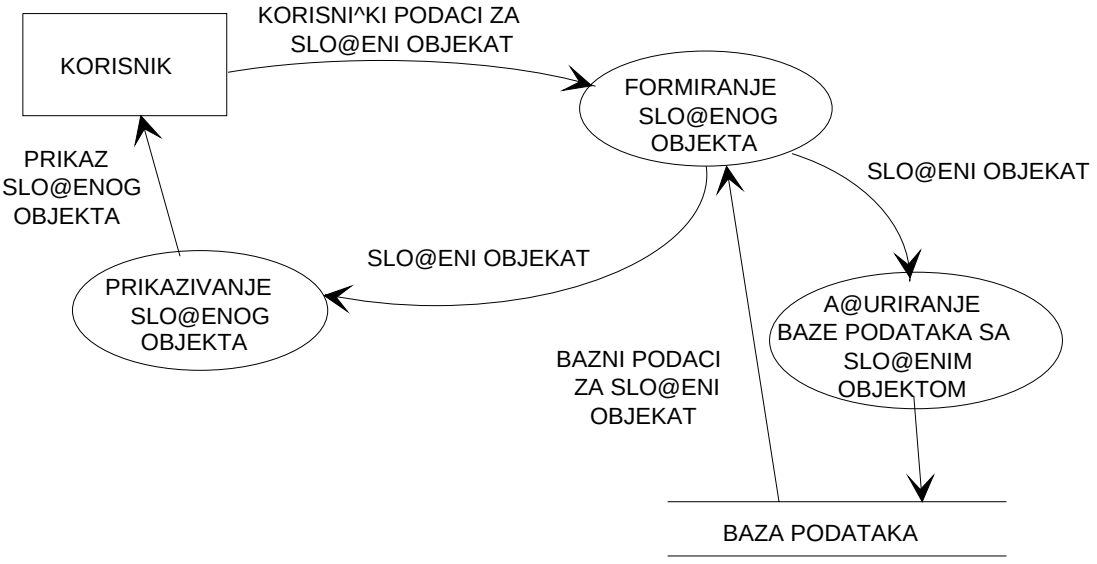
(3) Specifikaciju svih vrednosnih ograničenja i odgovarajućih dinamičkih pravila integriteta. Na primer,

```
OGR1(RADNIK,RADI) := FOREACH RADNIK ( IF EXISTS
    RADNIK(RADNIK.RADI.NAZOD = 'Razvoj') THEN RADNIK.STAROST <= 45);

insert RADNIK, update RADNIK, update ODELENJE, connect ZAPOŠLJAVA :
OGR1 OTHERWISE Prikaži 'Radnik u Razvoju striji od 45 godina,'
        Odbi operaciju
```

6.2. Specifikacija aplikacija

Da bi se specifikovala aplikacija neophodno je definisati strukturu ulaznih, odnosno izlaznih tokova i način na koji ulazni tok ažurira bazu podataka, odnosno način na koji se iz baze dobija izlazni tok. Struktura ulaznih i izlaznih tokova opisana je u Rečniku podataka SSA. Ovi tokovi pretstavljaju složene objekte koji se mogu dobiti (izvesti) iz strukture baze podataka (baznih objekata) jer je baza podataka (PMOV) projektovana na osnovu ovih njih.

Opšta arhitektura jedne aplikacije prikazana je na Slici 35. U svakoj aplikaciji se formira neki složeni objekat, a zatim se on bilo koristi da se preko njega izvrši ažuriranje baze podataka i/ili se prikazuje korisniku. Formiranje celog ili dela složenog objekta može izvršiti korisnik operacijama "editovanja" ili se to može učiniti preko funkcija (procedura) na osnovu podataka iz baze podataka. Svaka aplikacija ne mora da ima sve procese i sve tokove podataka sa slike 35



Slika 35. Opšta arhitektura aplikacije

Očigledno je da specifikacija aplikacija obuhvata:

- Prikazivanje strukture njenih složenih objekata;

- Prikazivanja načina formiranja složenog objekta, odnosno njegovih komponenti;
- Definisanje operacija ažuriranja koje se preko složenog objekta izvršavaju;
- Specifikacija spoljneg izgleda složenog objekta.

Prikazivanje načina formiranja složenog objekta daje se u okviru opisa njegove strukture i definisanjem posebnih operacija izvođenja. Zato specifikacija aplikacija sadrži deo za specifikaciju strukture složenog objekta , deo za opis operacija (izvođenja i ažuriranja) i deo za opis izgleda složenog objekta.

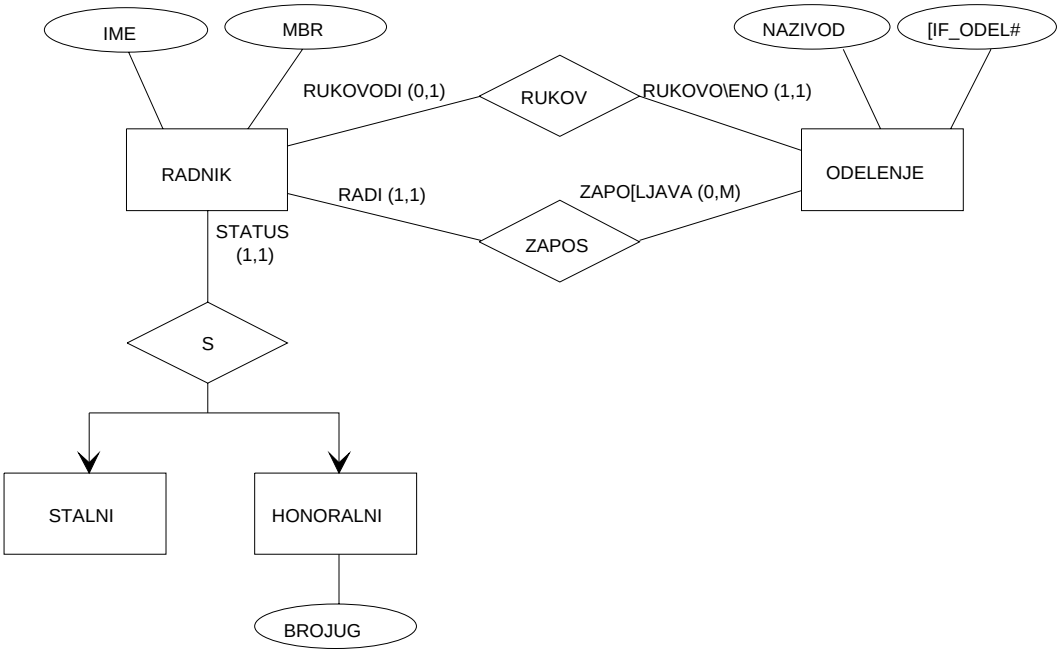
6.2.1. Specifikacija složenog objekta

Ovde će prvo biti data i objašnjena sintakse za opis strukture složenog objekta, a zatim će se navesti nekoliko pimera.

Sintaksa za opis složenih ulazno-izlaznih objekata je:

```
opis_objekta :: DEFINE ime-objekta [SUBTYPE_OF] ime_objekta
            kompl_struk

kompl_struk :: '{' struktura',' [CODOPUP]'}'
          | '<' lista_komponenti '>'
          | '['lista_komponenti ']'
          | '/' _lista_komponenti '/'

* zagrade određuju kao i ranije, vrstu kompleksne strukture *

lista_komponenti :: komponenta
                | komponenta ',' lista_komponenti

komponeneta :: naziv_ komp ':' struktura_d_v
                | naziv_komp ':' struktura_d_v ',' način_dobijanja

struktura_d_v :: prosta
       | kompl_struk
       | kompl_struk FROM naziv_preslikavanja
       | [COMPLEX] OBJECT naziv_objekta
       | [COMPLEX] OBJECT naziv_objekta FROM naziv_preslikavanja
       | ENTITY naziv_objekta
       | ENTITY naziv_objekta FROM naziv_preslikavanja

prosta :: naziv_ atributa_objekta
        | naziv_semantičkog domena
        | standardni_domen

način_dobijanja :: procedura | procedura ',' NE
procedura :: tip_procedure naziv_procedure
tip_procedure :: PROC | PROCR |  PROCU
```

Sintaksu za definisanje složenog objekta objasnićemo na primeru složenog objekta Radnici_odelenja koji se izvodi iz modela objekti-veze prikazanog Slici 36. Složeni objekat Radnici_odelenja daje za svako odelenje u firmi njegovu šifru i naziv matični broj i ime njegovog rukovodioca, ukupan broj radnika i popis svih radnika sa njihovim matičnim brojevima, uz naznaku da li je posmatrani radnik stalni ili honorarni kao i imenima i brojevima ugovora za honorarne radnike.



Slika 36. Primer PMOV za specifikaciju aplikacija

Struktura složenog objekta se, po pravilu, izvodi iz strukture baze podataka, polazeći od jednog objekta. Složeni objekat je podtip (opciona ključna reč SUBTYPE_OF) ovog "polaznog objekta"

DEFINE Radnici_odelenja SUBTYPE_OF Odelenje

Posle definicije naziva složenog objekta i njegovog "polaznog" baznog objekta daje se njegova struktura (kompl_struk u sintaksi). Komponenta strukture može da bude imenovana i neimenovana. Ime strukture od njenog opisa se razdvaja sa ":". Struktura složenog objekta može da bude agregacija njegovih komponenti (konstruktor agregacije je < >), skup pojavljivanja tipa komponente (konstruktor skupa pojavljivanja tipa komponente je { }), ekskluzivna specijalizacija (konstruktor ekskluzivne specijalizacije je **[ ]**), ili neekskluzivna specijalizacija (konstruktor / /).

Komponenta strukture se definiše preko njenog imena (naziv_komp), opisa same strukture (struktura_d_v) i, ponekad, načina dobijanja vrednosti komponente (način_dobijanja).

Komponenta strukture može da bude prosta ili složena (odnosno ponovo agregacija, skup, eksluzivna ili neeksluzivna specijalizacija).Sa konceptom"struktura_d_v" definiše se bilo domen komponente, bilo veza komponente sa objektom baze podataka. Domen se zadaje samo za proste komponente kada može da bude bilo predefinisani ili semantički. I za proste i za složene komponente može se zadati njihova veza sa objektima baze podataka.

Veza između komponenti složenog objekta i baze podataka ostvaruje se na dva načina:

- preko prostih komponenti čiji su "domeni" (strukture, po imenovanju u datoj sintaksi) atributi baze podataka (na primer, Šifra : šif_odel#),

- navođenjem da se data struktura izvodi preko nekog preslikavanja u modelu objkekti-veze baze podataka - ključna reč FROM. Na primer za objekat odelenje, popis svih radnika se daje preko strukture

Rade { <Matični_broj: mbr, Ime: ime>} FROM zapošljava

Koncept "način_dobijanja" definiše način na koji se dobija vrednost komponente. Vrednost komponente se može dobiti direktnim unosom (editovanje) i u tom slučaju se "način_dobijanja" vrednosti ne zadaje. Drugim rečima, za sve komponente se podrazumeva, ako nije drugačije naglašeno, sa ključnom reči "NE" (not edit), da se mogu editovati. Vrednost komponente se može i sračunavati (izvoditi) preko neke procedure. Razlikuju se, preko ključnih reči sledeće vrste procedura:

PROCR - procedura se izvršava prilikom generisanja složenog objekta iz baze podataka;
PROCU - procedura se izvršava ako se ažurira baza podataka sa složenim objektom;
PROC - procedura se izvršava u oba gore spomenuta slučaja.

Ako se komponenta dobijena nekom procedurom ne može posle toga editovati dodaje se ključna reč NE. U opisu složenog objekta navodi se samo naziv procedure, a "telo procedure" se opisuje posebno, bilo kao upit, bilo kao pseudokod u delu za specifikaciju operacija.

Pošto komponenta složenog objekta može da bude i skup, takva komponenta se može tretirati kao višeznačni atribut složenog objekta. Na primer Rade je višeznačni složeni atribut objekta Radnici_odelenja. Ako se takav objekat koristi za ažiriranje baze podataka, tada je neophodno navesti koje se pojavljivanje višeznačnog atributa ubacuje u bazu, koje iz nje izbacuje, a koje služi za promenu već postojeće vrednosti višeznačnog atributa. Na primer, koji se novi radnici ubacuju u odelenje, koji iz njega izbacuju i za koje se radnike menjaju podaci. Zato se uz komponentu koja predstavlja skup dodaje i promenljiva sa nazivom CODOPUP (kod operacije ažuriranja) koja može, za svaki elemenat skupa, da prihvati jedan od kodova operacije ažuriranja:

- A - dodavanje elementa
- D - izbacivanje elemeta
- U - promena vrednosti elementa

Na osnovu dosadašnjih objašnjenja, moguće je definisati sledeći deo strukture složenog objekta Radnici_odelenja:

DEFINE Radnici_odelenja SUBTYPE_OF ODELENJE
<Šifra: ŠIF_ODEL#,
Naziv: NAZIVOD, PROCU Upit_1,
Rukovodilac: < MBR> FROM RUKOVOĐENO
Rade :{ <Matični_broj: MBR, Ime: IME>, CODOPUP} FROM ZAPOŠLJAVA >

Složeni objekat Radnici_odelenja je podtip baznog objekta ODELENJE.
On pretstavlja agregaciju svojih komponenti (atributa):

- Šifra koji je vezan sa atributom ŠIF_ODEL# baznog objekta Odelenje;

- Naziv koji se dobija kao rezultat Upita_1 i koji će docnije, u popisu operacija biti specificiran preko upita

```
Upit_1:
    SELECT NAZIVOD
    FROM  ODELENJE
    WHERE ŠIFOD = Radnici_odelenja.Naziv;
```

Ovaj upit kao rezultat daje atribut nazivod baznog objekta Odelenje za vrednost atributa šifod zadatu u atributu Naziv, složenog objekta Radnici_odelenja;

- Rukovodilac koja je "vezana" sa atributom mbr baznog objekta Radnik preko preslikavanja Rukovodi;

- Višeznačnog složenog atributa (skupa pojavljivanja agregacije) Rade koja se sastoji od atributa Matični_broj i Ime koji su vezani sa atributima MBR i IME, respektivno, baznog objekta Radnik preko preslikavanja Zapošljava.

Primer dobijanja vrednosti komponenata preko pseudokoda je:

```
DEFINE Radnici_odelenja SUBTYPE_OF ODELENJE
<Šifra: ŠIF_ODEL#,
 Naziv: NAZIVOD, PROCU Upit_1,
 Rukovodilac: < MBR> FROM RUKOVOĐENO
 Rade :{ <Matični_broj: MBR, Ime: IME>, CODOPUP} FROM ZAPOŠLJAVA,
 Broj_radnika: INT(3), PROCR Ukupan_br_radnika >


PROCEDURA Ukupan_br_radnika
x: ODELENJE
y: x.ZAPOŠLJAVA
BEGIN
GET ANY x WHERE x.ŠIF_ODEL = Radnici_odelenja.Šifra;
Ukupan_br_radnika = 0;
GET ANY y;
IF EOK THEN RETURN
     ELSE BEGIN
     PERFORM UNTIL EOK
     Ukupan_br_radnika := Ukupan_br_radnika + 1;
     GET DUPLICATE y
     END PERFORM
ENDIF
END
```

Objašnjenje ostalih delova prikazane sintakse:

- Opcija da se struktura komponente specificira kao ENTITY ime_objekta koristi se da se prikaže samo postojanje nekog objekta, bez prikazivanja bilo kog njegovog atributa. Na primer, komponenta

Vrsta_radnog_odnosa se može definisati sa

```
Vrsta_radnog_odnosa: [Stalni: ENTITY Stalni,
          Honorarac: br_ugovora FROM Hnorarni ] FROM Status
```

Komponenta Vrsta_radnog_odnosa je ekskluzivna unija dve druge komponente, Stalni i Honorarac. Ako je posmatrano pojavljivanje radnika stalni radnik, komponenta stalni imaće neku oznaku, a ako je honorarani prikazaće mu se i broj ugovora.

- Komponenta, odnosno struktura nekog složenog objekta može da bude i ceo objekat, bazni ili složeni izvedeni (kada se koristi opciona ključna reč COMPLEX). U ovom slučaju se smatra da je komponenta jedno pojavljivanje objekta sa svim njegovim atributima (odnosno strukturom). Na primer kada bi se želelo da se u komponenti Rade složenog objekta Radnici_odelenja preuzmu svi atributi radnika sa njihovim imenima tada bi se ova komponenta mogla definisati kao:

Rade: { OBJECT RADNIK } FROM ZAPOŠLJAVA

Ova definicija bi bila ekvivalentna (skraćeni oblik) definicije

Rade { <mbr: MBR, ime:IME, pol:POL>} FROM ZAPOŠLJAVA

- Prosta struktura (domen atributa složenog objekta) može, pored atributa iz baze podataka, da bude i neki, ranije definisani, u okviru baze podataka ili dijagrama tokova podataka, semantički domen, ili neki standardni domen (INT, REAL, SRING i drugi).

6.2.2 Specifikacija operacija sa složenim objektom

Operacije sa složeni objektom mogu da budu:
- Operacije žuriranja baze sardržajem složenog objekta. U tom slučaju je dovoljno da se navede samo dali se složeni objekat ubacuje u bazu podatka (operacija INSERT naziv_složenog_objekta), iz nje izbacuje (operacija DELETE naziv_složenog_objekta) ili se sa složenim objektom ažurira baza (operacija UPDATE naziv_složenog objekta).
- Opreacije izvođenja komponenata složenog objekata koje se prikazuju preko odgovarajućih fukcija, kako je to već ranije pokazano. Operacijama izvođenja, očigledno, može se izvesti i ceo objekat, a ne samo neke njegove komponente.

Specifikacija jedne aplikacije u kojoj bi se najpre za datu šifru odelenja prikazo njegov naziv i ukupan broj radnika u njemu, a zatim uneli izmenjeni podaci o radnicima tog odelenja i sa njima ažurirala baza podataka bila bi:

```
STRUCTURES

DEFINE Radnici_odelenja SUBTYPE_OF Odelenje
<Šifra: ŠIF_ODEL#,
 Naziv: NAZIVOD, PROCR Upit_1,
 Rukovodilac: < MBR> FROM RUKOVOĐENO
 Rade :{ <Matični_broj: MBR, Ime: IME,
 Vrsta_radnog_odnosa: [Stalni Entity STALNI,
        Br_ug_honorarnog:FROM HONORARNI ] FROM STATUS>, CODOPUP}
 FROM ZAPOŠLJAVA,
Broj_radnika: INT(3) PROCR Ukupan_br_radnika >

OPERATIONS

UPDATE Radnici_odelenja;

Upit_1:
    SELECT NAZIVOD
    FROM  ODELENJE
    WHERE ŠIFOD = Radnici_odelenja.Naziv;

PROCEDURA Ukupan_br_radnika
x: ODELENJE
y: x.ZAPOŠLJAVA
BEGIN
GET ANY x WHERE x.ŠIF_ODEL = Radnici_odelenja.Šifra;
Ukupan_br_radnika = 0;
```

```
GET ANY y;
IF EOK THEN RETURN
     ELSE BEGIN
     PERFORM UNTIL EOK
     Ukupan_br_radnika := Ukupan_br_radnika + 1;
     GET DUPLICATE y
     END PERFORM
ENDIF
END
```

6.2.3. Specifikacija spoljnjeg izgleda objekta

Specifikacija spoljnjeg izgleda objekta u velikoj meri zavisi od softvera kojim se raspolaže. Zato se ovde o tome neće govoriti, već docnije kada se detaljnije bude govorilo o implementaciji.