Milosav Majstorović,
kapetan, dipl. inž.
Rajko Terzić,
kapetan, dipl. inž.

Jedan pristup projektovanju informacionih sistema na primeru segmenta IS Univerziteta Vojske Jugoslavije

U radu se izlaže jedan pristup projektovanju informacionih sistema. On se zasniva na funkcionalnoj dekompoziciji sistema, modeliranju podataka i implementaciji na relacionom sistemu za upravljanje bazom podataka. Pristup je ilustrovan na primeru jednog segmenta informacionog sistema (IS) Univerziteta Vojske Jugoslavije koji je u sferi razvoja.

U uvodu se obrazlaže ovaj pristup, daju neki osnovni pojmovi i definicije. Sledi osvrt na metodološke osnove razvoja IS. Opisuje se metoda strukturne sistem analize (SSA), modeli podataka, prevođenje MOV u relacioni model i specifikacija pravila integriteta baze podataka. U prilozima se daju primeri implementacije u ORACLE RDBMS V.6.

Uvod

Rezultat projektovanja informacionog sistema (IS) jeste, pre svega, određeni softverski proizvod. Zato su svi problemi softverskog inžinjerstva i problemi projektovanja IS.

Već dvadesetak godina se govori o »softverskoj krizi«. Neki podaci koji ilustruju veličinu ove krize poslednjih godina mogu se naći npr. u [7]. Razni aspekti softverske krize davno su uočeni i podstakli su razvoj mnogih metoda razvoja softvera. Međutim, metode razvijene sedamdesetih godina nisu značajnije doprinele rešavanju problema softverske krize. Jedini izuzetak predstavlja prototipski razvoj uz pomoć jezika četvrte generacije [7].

Imajući u vidu ove činjenice u ovom radu se izlaže jedan pristup projektovanja informacionih sistema koji se zasniva na funkcionalnoj dekompoziciji pomoću *strukturne sistem analize* (SSA), a zatim integraciji podmodela (»pogleda«), koristeći semantički bogat model podataka — *model objekti-veze* (MOV) [3, 8].

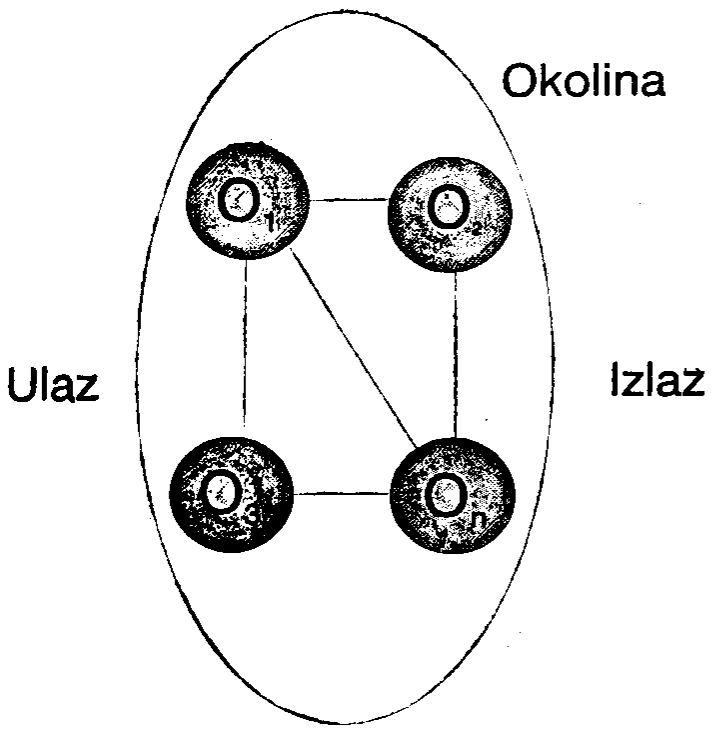
Kao implementacioni model koristi se model podataka II generacije — *relacioni model*. Ovaj model, pored ostalog, pruža mogućnost primene jezika četvrte generacije koji omogućuju generisanje brzog prototipa. Pošto nemamo na raspolaganju objektnu (odnosno objektno-orijentisanu) bazu podataka ili CASE alat koji bi omogućio primenu objektno-transformacionog pristupa [7, 8], ovaj pristup je sasvim zadovoljavajuće rešenje.

Metodologija razvoja IS zahteva da se precizno definiše šta se pod pojmom informacionih sistema podrazumeva, koje su njegove funkcije i kakav je položaj u sistemu u kome deluje. Metodologija razvoja IS treba da bude opšta, primenljiva na sisteme bilo koje vrste, odnosno na neki opšti sistem. Zbog toga se u sledećem delu daju neki osnovni pojmovi i definicije.

Osnovni pojmovi i definicije

(1) *Sistem* je skup objekata i njihovih veza. Objekti u sistemu opisuju se preko svojih svojstava koja se nazivaju atributima. Na sl. 1 prikazana je 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. Zato je neophodno 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.



Sl. 1 — Opšta definicija sistema

(2) *Informacija* je kapacitet povećanja znanja (I. Wilson).

(3) *Informacioni sistem* je sistem u kojem se veze između objekata i veze sistema sa okolinom ostvaruju razmenom informacija.

(4) *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.

(5) Da bi definisali osnovne koncepte baze podataka, navešćemo osnovne nedostatke *klasične obrade podataka* [4]:

- redundansa podataka;
- zavisnost programa od organizacije podataka;
- niska produktivnost u razvoju IS;
- pasivan odnos korisnika.

Rešavanje ovih problema klasične obrade dovelo je do razvoja sistema za upravljanje bazama podataka (SUBP).

(6) *Sistem za upravljanje bazama podataka* je složeni softverski sistem koji treba da omogući [4]:

- skladištenje podataka sa minimumom redundanse;
- korišćenje zajedničkih podataka od strane svih ovlašćenih korisnika;
- logičku i fizičku nezavisnost programa od podataka;
- jednostavno komuniciranje sa bazom podataka preko jezika bliskih korisniku.

U ovakvoj tehnologiji obrade, podaci su, umesto razbacani po nezavisnim datotekama, organizovani u jedinstvenu bazu podataka.

(7) *Baza podataka* je kolekcija međusobno povezanih podataka, uskladištenih sa minimumom redundanse, koje koriste, zajednički, svi procesi obrade u sistemu.

Tehnologija baza podataka omogućila je da se problem razvoja IS postavi na novi način.

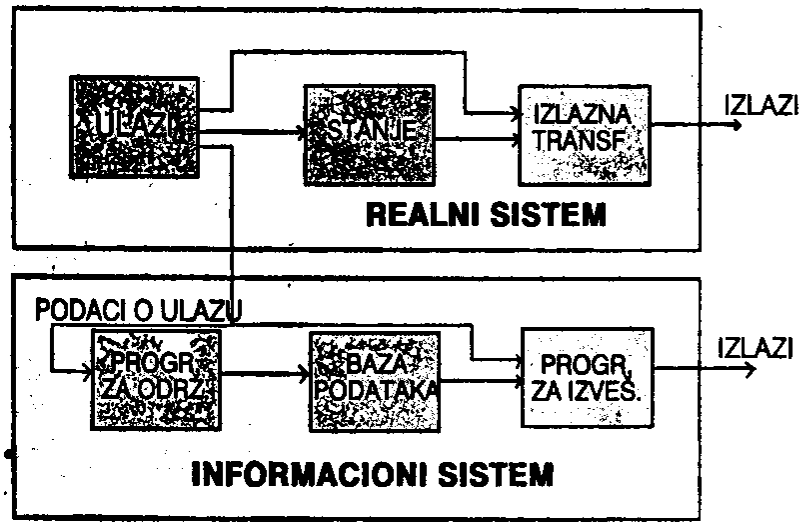
Metodološke osnove razvoja informacionih sistema

Sistem se definiše kao *skup objekata* i njihovih *međusobnih* veza. Objekti u sistemu mogu biti fizički objekti, koncepti, događaji i drugo. U modelu nekog sistema objekti se opisuju preko svojih *svojstava (atributa)*. Dejstvo okoline na sistem opisuje se preko *ulaza* u sistem, a dejstvo sistema na okolinu preko njegovih *izlaza*.

Slika (preuzeta iz [7]) pokazuje dinamičko ponašanje realnog sistema. U ovom slučaju stanje sistema opisuje fundamentalne karakteristike sistema. U jednom trenutku vremena ono predstavlja skup objekata sistema, skup njihovih međusobnih 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 [7].

Kao što se vidi sa slike 2., informacioni sistem predstavlja model real-

nog sistema u kojem deluje. Ovakav položaj informacionog sistema je bitno drugačiji od onog iz sedamdesetih godina koji se zasnivao na tzv. »upravljačkom« informacionom sistemu [3]. Slabost ovog pristupa proizilazi iz toga što su se metode razvoja IS zasnivale na tzv. »životnom ciklusu« IS koji predstavlja, detaljnije definisan, niz faza razvoja IS (planiranje razvoja, analiza i specifikacija zahteva, projektovanje, implementacija i održavanje). Osnovni problem u ovakvom pristupu je specifikacija zahteva korisnika [4].



Sl. 2 — Položaj informacionog sistema u odnosu na realni sistem

U informacionom sistemu, kao modelu realnog sistema (slika 2), osnovu čini baza podataka koja predstavlja fundamentalne, stabilne, sporo izmenljive karakteristike sistema, objekte u sistemu i njihove međusobne veze.

Zahvaljujući ovoj fundamentalnoj postavci BP time se dobrim delom zaobilazi ključni problem u konvencionalnom pristupu razvoja IS, specifikacija zahteva za informacijama.

Dakle, postupak projektovanja se ne bazira na tim stalno promenljivim zahtevima, već na modeliranju fundamentalnih, stabilnih karakteristika sistema.

Ako je informacioni sistem model realnog sistema u kojem deluje, onda se projektovanje IS svodi na neku vrstu modeliranja realnog sistema, a za to su nam neophodna neka intelektualna sredstva (alati) i to [3]:

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

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

Imajući u vidu zadovoljavajući pristup razvoju informacionih sistema, istaknut u uvodu, faze projektovanja IS mogle bi biti:

- strukturna sistem analiza (SSA);
- formiranje modela realnog sistema koristeći semantički bogat model podataka model objekti-veze;
- prevođenje tako dobijenog modela u relacioni model koji će biti implementacioni model;
- realizacija u konkretnom softveru za upravljanje bazama podataka.

U nastavku ćemo prvo prikazati cilj i sredstva strukturne sistem analize, kroz ilustraciju na našem primeru.

Metoda strukturne sistem-analize

U projektovanju IS može se reći da precizna definicija zahteva korisnika, zahteva koje budući sistem treba da zadovolji, predstavlja bitan preduslov za uspešno projektovanje i implementaciju sistema. Osnovni cilj sistem-analize, kao prve faze projekta IS, upravo je specifikacija zahteva korisnika. Za ostvarenje ovog cilja, pored mnogo truda koji treba da ulože, sistem-analitičari treba da imaju na raspolaganju i odgovarajuća sredstva za opis sistema i specifikaciju zahteva, tehnike za pri-

menu tih sredstava na sistematizovan i dosledan način i metode za izbor strategije i efikasnu organizaciju postupka sistem-analize [7].

U nastavku dajemo definicije opštih pojmova iz ove problematike, a zatim ukratko prikazujemo sredstva strukturne sistem-analize.

Opšti pojmovi [1]:

Cilj *analize* je otkrivanje, kao i opisivanje funkcija sistema. Funkcija se opisuje u terminima: delova sistema, veza između delova sistema i veza delova sa okolinom.

Cilj *sistem-analize* (SA) isti je, s tim što predmet istraživanja predstavljaju kombinacije aktivnosti koje za neke ulaze podataka proizvode izlaze podataka koji imaju smisao (sadrže informacije).

Cilj *struktuirane (dobro struktuirane) sistem-analize* je opis funkcija u skladu strukturnosti, tj. cilj je da se sistem može opisati korišćenjem samo elementarnih operacija ili sekvencijom, selekcijom i iteracijom elementarnih operacija. Pored toga, *SA* opisuje logički tok podataka kroz informacioni sistem, kao i strukturu za svaki tok podataka.

Rezultat *SA* **je** FUNKCIONALNA SPECIFIKACIJA i nju koriste projektanti za proizvodnju detaljnih specifikacija.

Funkcija je proces koji koristi ulaz da bi proizveo izlaz. Elementarna funkcija nema podfunkcija i one su najniži nivo sistema koji se analizira.

Funkcionalna dekompozicija je tehnika koja se koristi za analizu funkcije. Najopštije rečeno, analiza funkcije sastoji se od dekompozicije tri dela funkcije:

1. dekompozicija ulaza u transakcije, podatke, zapise, grupe ili polja koje proces koristi;
2. dekompozicija izlaza u transakcije, zapise, grupe podataka ili elementarne podatke koje proces proizvodi;
3. dekompozicija procesa na operacije nad svakom komponentom ulaza, koje proizvode komponente izlaza.

Dakle, funkcionalna dekompozicija je analitička tehnika razvijanja funkcije na njene podfunkcije.

Standardna sistem-analiza (SSA) koristi ulaze da bi proizvela izlaze i transformiše ulaz u izlaz na razuman način. [1]

Ona omogućava:

- *da* se omeđi logički model i definiše specifikacija za *IS* korišćenjem dobro struktuiranih sredstava;
- da se celina IS sagledava sa aspekta toka podataka *intervjuisanjem korisnika,* jer oni sagledavaju celinu.

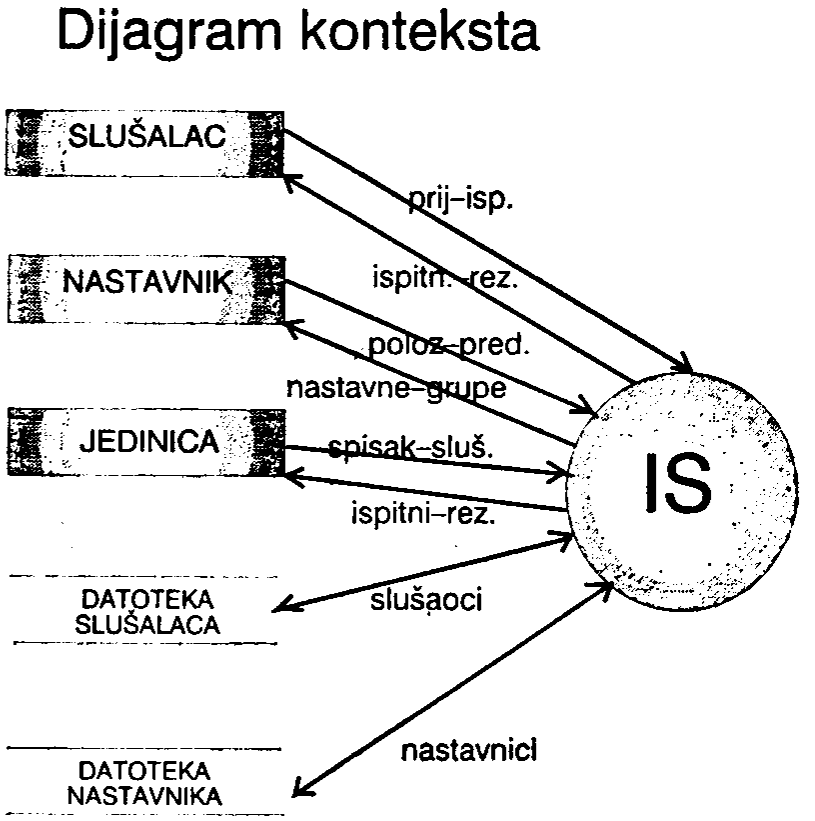
Sredstva strukturne sitem-analize

Strukturna sistem-analiza (SSA) nastala je kao odgovor na problem neadekvatne specifikacije zahteva korisnika pomoću klasičnih sredstava funkcionalne analize. Sredstva SSA predstavljaju osnovni alat sistem-analitičara, koji i sam mora da zadovolji određene zahteve. Prvo, to je precizno definisanje zahteva korisnika. Drugo, opis sistema i specifikacija zahteva predstavlja ulaz u sledeću fazu rada, projektovanja sistema. Stoga, sredstva sistem-analize treba da omoguće i formalizovanu specifikaciju zahteva, tako da ih je moguće automatizovati, sa krajnjim ciljem da se omogući automatsko generisanje aplikacije.

Sredstvo koje zadovoljava prvi od navedenih zahteva jeste dijagram toka podataka (DTP).

Dijagram toka podataka DTP prikazuje tokove podataka između procesa obrade, izvorišta i odredišta podataka, kao i internih skladišta podataka.

Sredstvo koje može zadovoljiti drugi zahtev je *rečnik podataka (RP),* pomoću kojeg se opisuje sadržaj i struktura svih tokova podataka i skladišta podataka. Formalna sintaksa za opis pomenutih struktura može se videti u [7], i ona u ovom radu nije prikazana. Postojanje standardnih sredstava SSA je potreban, ali ne i dovoljan uslov za uspešno provođenje sistem-analize. Za to su potrebne i određene tehnike i metode. One u ovom radu nisu posebno razmatrane, pošto se ovaj pristup projektovanju IS zasniva, pre svega, na modelu podataka. U nastavku se daju elementi DTP kroz analizu jednog segmenta IS za praćenje uspeha slušalaca Vojne akademije.



Sl. 3 — IS za praćenje uspeha slušalaca

Elementi DTP:

- proces,
- tok podataka,
- skladišta podataka,
- izvorišta i odredišta (interfejsi — spoljni objekti).

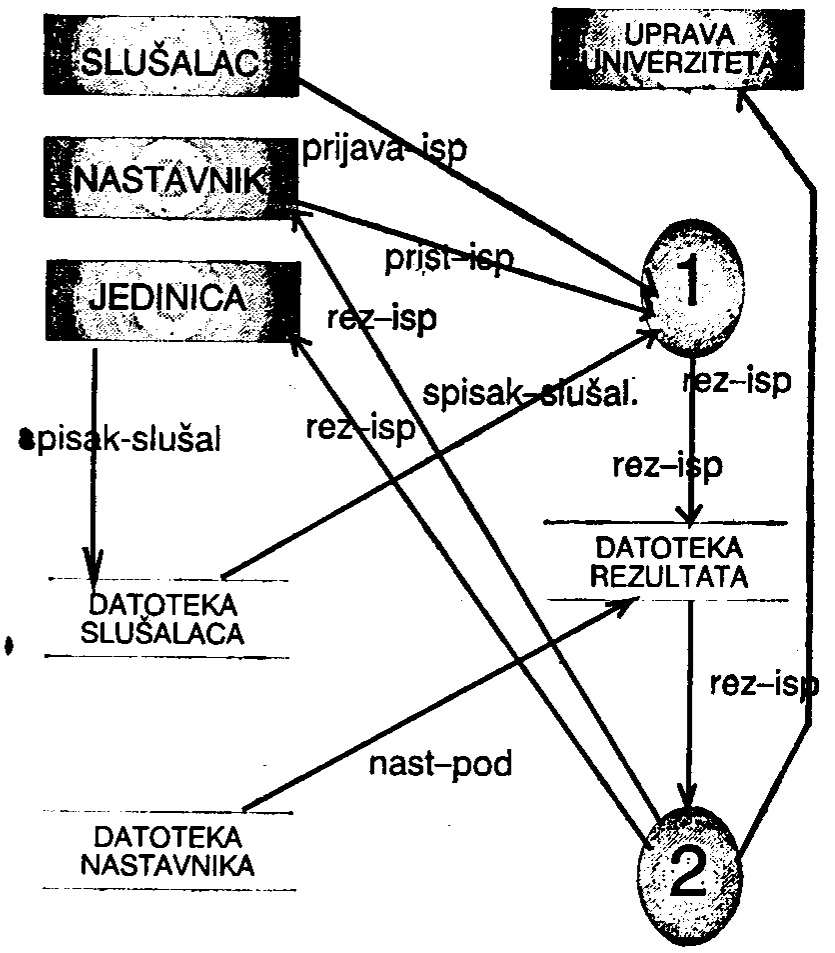
Većina sistema je složena i zahteva hijerarhijski opis, tj. podelu na podsisteme na više nivoa apstrakcija. Dakle, kao rezultat SSA dobijamo hijerarhijski organizovan skup DTP-a.

Dijagram konteksta je najviši nivo apstrakcije, prikazuje neto ulaze i izlaze i služi da ograniči domen posmatranja, tj. postavi granicu sistema (sl. 3).

Srednji nivo prikazuje podelu sistema na delove koji imaju smisla i lako su razumljivi, ali su još uvek prekrupni za detaljnu analizu (sl. 4).

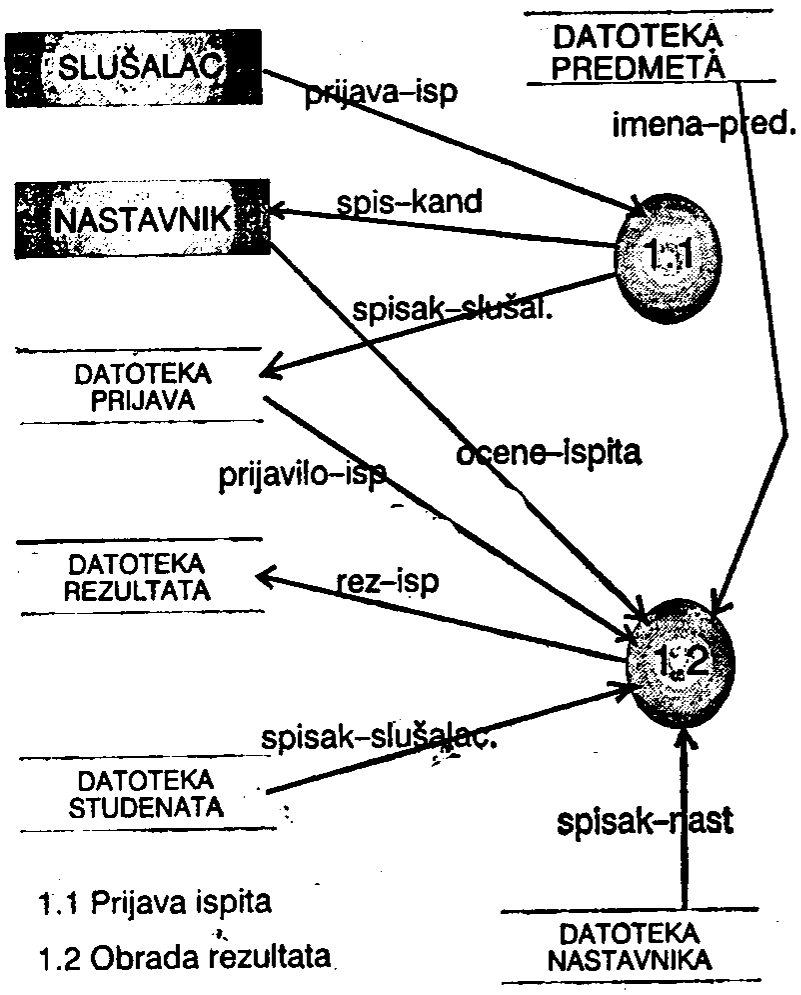
Primitivne funkcije su na najnižem nivou opisa funkcije. One ne moraju da se dalje dekomponuju, jer su potpuno razumljive (sl. 5).

Druga primitivna funkcija (sl. 6).

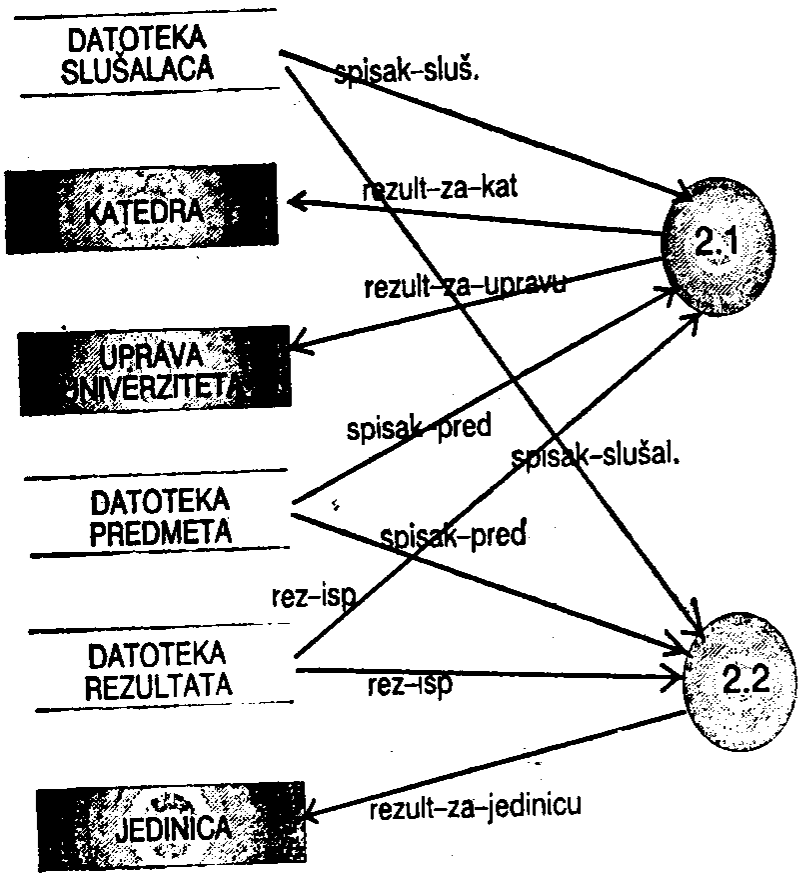


1. Prikupljanje podataka i obrada
2. Izrada različitih vrsta izveštaja

Sl. 4 — Srednji nivo



Sl. 5 — Primitivne funkcije



Sl. 6 — Primitivne funkcije

Sada, radi ilustracije, dajemo primer opisa strukture toka podataka (rezultati-ispita) i skladišta podataka (datoteka-slušalaca) prema sintaksi iz [7].

REZULTAT-ISPITA: < šif-predmeta, naziv-predmeta, ime-nastavnika, datum-ispita, {broj-indeksa, ime-slušaoca, ocena}>

DATOTEKA-SLUŠALACA: {broj--indeksa, ime-slušaoca, smer, {šifra--predmeta, datum-ispita, ocena-ispeta}}

Rečnik podataka se dalje koristi za definisanje modela objekti veze. Naravno, za izradu MOV koriste se i dodatna znanja o realnom sistemu, što će se videti u sledećem poglavlju.

Modeli podataka

Model podataka je intelektualno sredstvo za opis statičkih karakteristika sistema, opis karakteristika sistema u nekom stacionarnom stanju. Stacionarno stanje nekog sistema karakteriše se skupom zavisnosti koje postoje između 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 opisati i dinamika realnog sistema. Interpretacija podataka se u nekom modelu podataka ostvaruje kroz tri njegove osnovne komponente [7]:

1. *strukturu modela,* odnosno skup koncepata za opis objekata sistema njihovih atributa i njihovih međusobnih 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.

Apstrakcija podataka

Glavni problem u opisivanju realnih sistema je njihova složenost. Opšti metodološki pristup za rešavanje ovog problema je *apstrakcija*, odnosno kontrolisano uključivanje detalja. Postupak inverzan apstrakciji je detaljisanje.

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* [7]. Radi razmatranja modela podataka ovde se apstrakcije podataka ukratko definišu.

- *Klasifikacija (tipizacija) i uzorkovanje.* Klasifikacija ili tipizacija je apstrakcija u kojoj se skup sličnih objekata predstavlja jednom klasom objekata, odnosno svaki objekat iz posmatranog skupa odgovarajućim *tipom* objekata [7]. Slični objekti su oni koji imaju iste atribute, koji mogu da stupe u iste veze sa drugim objektima u sistemu i na koje se mogu primeniti iste operacije. Postupak detaljisanja za ovu operaciju naziva se *uzorkovanje.*

- *Generalizacija i specijalizacija. Generalizacija* je apstrakcija u kojoj se skup sličnih tipova objekata predstavlja opštijim generičkim tipom (nadtipom). *Specijalizacija* je inverzni postupak u kome se za neki tip navode njegova moguća pojavljivanja (tip se specijalizuje u podtipove) [7].

- *Agregacija i dekompozicija. Agregacija* je apstrakcija u kojoj se skup tipova objekata i njihovih veza tretira kao jedinstven agregirani tip objekta. Postupak inverzan agregaciji naziva se *dekompozicija.*

Navedeni tipovi apstrakcije biće ilustrovani na primerima kroz prikaz modela objekti-veze.

Očigledno je da modeli podataka treba (poželjno je) da podrže sve apstrakcije podataka, kao i da se mogu lako realizovati, na implementacionom nivou. Prema tome, svaki model podataka treba da zadovolji dva bitna kriterijuma:

1. da poseduje koncepte pogodne za modeliranje realnih sistema,
2. da se njegovi koncepti, struktura, ograničenja i operacije mogu jednostavno implementirati na računaru.

Na osnovu toga kako zadovoljavaju ova dva kriterijuma izvršena je klasifikacija modela podataka u sledećem poglavlju.

Generacije modela podataka

Prvu generaciju čine klasični programski jezici (jezici treće generacije) u kojima se apstrakcije podataka realizuju preko tipova podataka kojima raspolažu. Ovde je problem što su koncepti ovih programskih jezika veoma daleko od objekata realnog sistema, pa treba dosta softverskog napora i umeća da bi se informacioni sistem implementirao.

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 načine povezivanja rekorda, odnosno bazu podataka kao skup međusobno povezanih podataka, znatno moćnije (makro) operacije, a ponekad i eksplicitno definisati specifične vrste ograničenja na vrednosti podataka [7]. Nedostaci klasičnih modela podataka su [9]:

- klasični modeli podataka su, u osnovi, rekord-orijentisani (bliži su konceptima računara, nego korisniku);
- mali broj modelirajućih koncepata;
- nedostatak koncepata za mode liranje apstrakcije (generalizacije i agregacije);

- otežana specifikacija ograničenja.

Međutim, postoje komercijalno raspoloživi softveri, SUBP, za njihovu direktnu implementaciju na računaru. Zbog toga se najpopularniji predstavnik ove generacije — relacioni model koristi i kao implementacioni model u ovom radu.

Treću generaciju čine takozvani semantički bogati modeli podataka i objektni modeli podataka (model objekti-veze, Semantic Data Model (SDM), prošireni relacioni model, semantičke mreže, i mnogi drugi), kao i različiti objektno orijentisani modeli podataka. Osnovne karakteristike koncepata za modeliranje u semantičkim modelima podataka su:

- koncepti su korisnički orijentisani,
- dopuštaju projektantu da predstavi objekte od interesa i veze između njih, slično pogledu korisnika na njih;
- podržavaju sve mehanizme apstrakcije (klasifikacije, generalizacije i agregacije);
- dozvoljava notaciju »izvedenih podataka«;
- svojstvo nasleđivanja unutar generalizacione hijerarhije.

Objektno orijentisani modeli [10] razvijeni su sa ciljem da prošire standardni domen primene SUBP, i tzv. »inžinjerijske aplikacije« (CAD/CAM, CASE, i slično). Oni uvode dinamiku u korisnički definisane objekte kroz standardne i korisnički orijentisane operacije (metode). Nivo apstrakcije, potreban za specifikaciju složenih objekata koji se u ovim aplikacijama javljaju, ostvaruje se mehanizmima, preuzetim iz objektno-orijentisanih jezika, koji omogućuju izvođenje operacija sa složenim objektima kao celinom.

Pošto se u ovom radu razmatraju, pre svega, informacioni sistemi u poslovnom okruženju čija je polazna tačka integralni semantički bogat model podataka, čijom se implementacijom dobija opšte upotrebljiva baza podataka, mi ćemo konceptualno modeliranje upravo i zasnivati na semantičkim modelima podataka.

Može se reći da semantički modeli podataka smanjuju semantički jaz između koncepata realnog sistema koga IS modeluje i koncepata implementacionog sistema na koje se koncepti realnog sistema moraju svesti prilikom implementacije. U tom smislu u nastavku se opisuje najpopularniji i u praksi najviše korišćeni predstavnik ove generacije *model objekti-veze.*

Model objekti-veze

Model objekti-veze (izvorno entity-relationship model — ERM) koji je prvi put objavljen u Chenovom članku [11] jedan je od prvih i najpopularnijih semantičkih modela podataka. Ovaj model se kraće naziva MOV ili ERM. Upotreba ERM dovela je do tzv. ER pristupa u projektovanju informacionih sistema i softverskom inženjerstvu. Početne motivacije za razvoj ER pristupa su:

(1) potreba za jedinstvenim modelom podataka;
(2) potreba za metodologijom logičkog projektovanja baze podataka (BP);
(3) potreba za prevođenjem podataka između različitih DBMS-a [12].

Glavni razlozi za dalju široku primenu i popularnost ER pristupa su:

(1) koncepti ERM (objekti, veze, atributi) jesu jednostavni, laki za razumevanje, ali prirodni i snažni koncepti za modeliranje i percepciju realnog sveta;
(2) on pokazuje da tri konvencionalna (hijerarhijski, mrežni i relacioni) modela podataka mo-

gu biti izvedeni iz semantički bogatijeg — ER modela;

(3) koncepti ER modela su formalizovani i predstavljaju osnovu za dalje istraživanje [12].

Učesnici 5. Internacionalne konferencije o ER pristupu, (Dijon, Novembar 1986), jesu istakli i sledeće prednosti ER pristupa:

- konstrukti ER pristupa pogodniji su za konceptualno modeliranje od drugih modela podataka. Tako, relacioni model usmerava pažnju ljudi sa realnog problema na stvari koje nisu važne (kao što je normalizacija);
- ER pristup, takođe, eksplicitno podržava strukturu posmatranih objekata ne samo u terminima objekata već, takođe, atributa i veza (koji su često odsutni kod drugih modela podataka);
- ER pristup omogućuje i strukturu atributa, što je korisno, jer često atribut na prvi pogled nije atomski;
- ER pristup dopušta top-down pristup sistem analizi. On omogućuje prvo uočavanje najvažnijih objekata, koji se tada obogaćuju sa atributima i kasnije normalizovanim strukturama podataka;
- ER pristup je vrlo korisna tehnika za planiranje razvoja informacionih sistema. On pomaže odgovoru na pitanje koju fazu (segment) prvo implementirati;
- jedna značajna prednost ER pristupa u odnosu na relacioni model je izražajna snaga ER dijagrama;
- ER pristup čini, takođe analizu događaja u sistemu lakšom. Analitičari npr. mogu videti kako poslovni događaji utiču na poslovne objekte u terminima operacija kreiranja, ažuriranja i brisanja;
- popularnost ER pristupa kod softverskih projektanata je u porastu, i on je geografski univerzalan. Zbog toga je relativno sigurna komercijalna investicija.

Sve to je, svakako, uticalo na izuzetnu popularnost i interesovanje za MOV (ERM) i ER pristup uopšte, kako u akademskim, tako i u poslovnim krugovima. To se, naravno, odrazilo na obim i intenzitet istraživanja na ovom polju. Posledica toga je razvoj velikog broja CASE alata koji podržavaju razvoj IS uz konceptualno modeliranje podataka upotrebom MOV, kao i DBMS zasnovanih na MOV. Treba reći da postoji više različitih verzija MOV, počev od originalne verzije [11] do verzija sa veoma bogatim različitim konceptima (npr. [13]). U ovom radu koristi se verzija iz [3]. U nastavku se MOV opisuje kroz prikaz strukture, ograničenja i operacija.

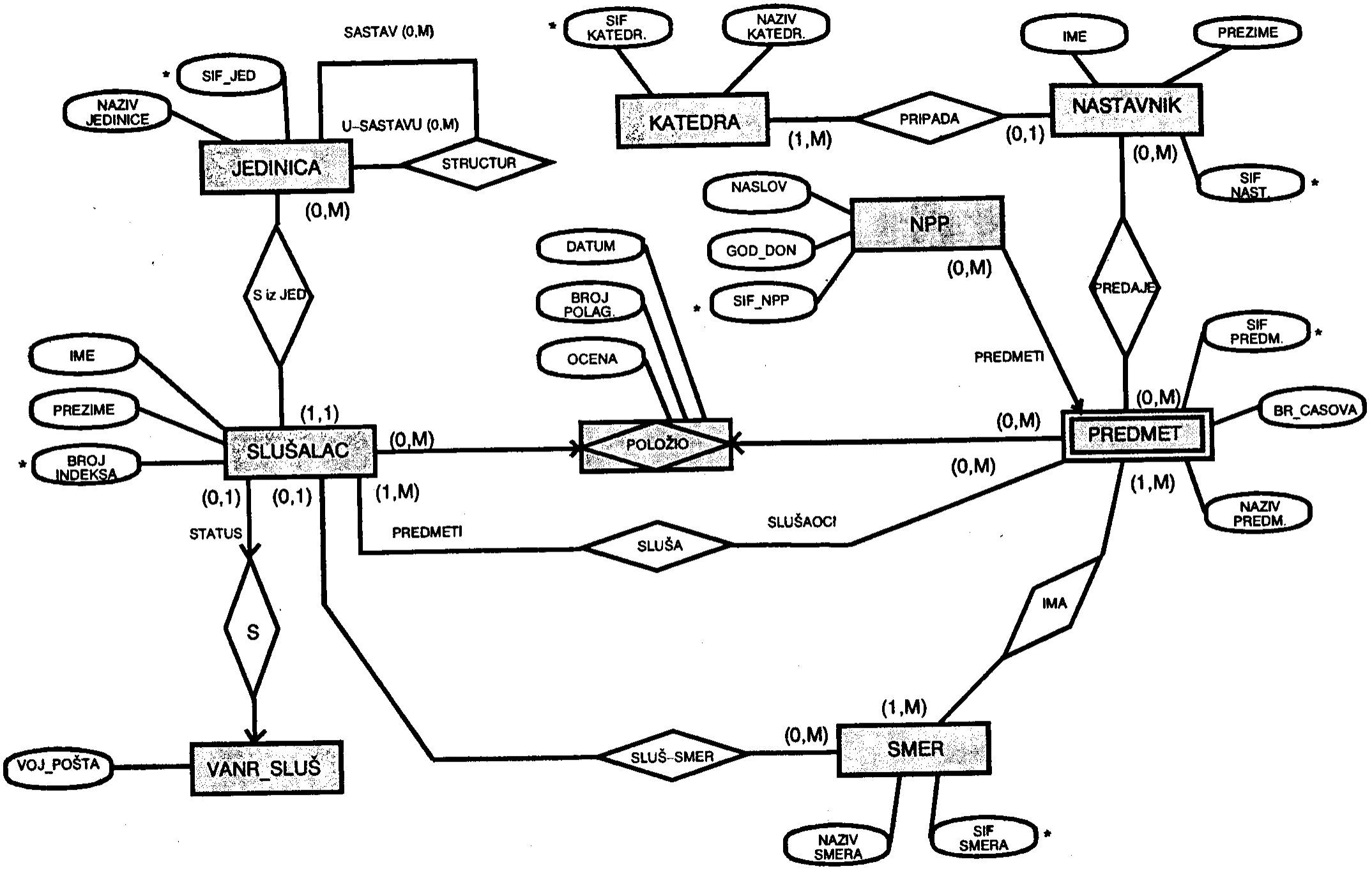
Struktura modela objekti-veze

Struktura modela objekti-veze predstavlja se *dijagramima objekti-veze (DOV).*

Na sl. 7 se vidi DOV za konkretni primer koji ćemo koristiti u daljem tekstu. U modelu se vide osnovni koncepti ovog modela podataka.

U MOV sistem se opisuje kao skup objekata i njihovih veza. Pojedinačni objekti (entiteti) u sistemu se klasifikuju (apstrakcije, klasifikacije-tipizacije) u tipove objekata. Tip objekta je opšti predstavnik neke klase, a svaki pojedinačni objekat predstavlja jedno pojavljivanje (primerak) datog tipa. Tipovi objekata u našem primeru (sl. 7) su npr. jedinica, slušalac, nastavnik, i dr.

Veze u modelu opisuju način povezivanja objekata. Svaki tip veze između dva objekta definiše dva preslikavanja, preslikavanje sa skupa pojavljivanja prvog objekta u skup pojavljivanja drugog objekta i (inverzno) preslikavanje sa skupa drugog objekta u skup prvog



Sl. 7 — Model objekti — veze

objekta. Svakom preslikavanju u vezi mogu se dodeliti imena.

Npr. veza SLUŠA ima preslikavanje

PREDMETI: SLUŠALAC→PREDMET (za svakog slušaoca daje predmete koje sluša),

SLUŠALAC: PREDMET.→SLUŠALAC (za svaki predmet daje slušaoce koji ga slušaju).

Ovde se može uočiti da postoji mogućnost uspostavljanja više *tipova veza* između istih tipova objekata (veze SLUŠA i POLOŽIO između objekata SLUŠALAC I PREDMET).

Jedna od bitnih karakteristika veza između objekata je *kardinalnost preslikavanja* koja ga čine. Na primer, kardinalnost preslikavanja definiše se parom (DG, GG), gde je DG (donja granica) daje najmanji mogući, a GG (gornja granica) najveći mogući broj pojavljivanja tipa objekta PREDMET za jedno pojavljivanje tipa objekta SLUŠALAC.

Ovde mora biti zadovoljen uslov $DG \leqslant GG$.

Svaka binearna veza definisana je sa dva preslikavanja. Nazivi ovih preslikavanja znatno opterećuju sam DOV, a koriste se tek kasnije pri pretvaranju modela objekti-veze u relacioni model i pri razvoju aplikacija. Zbog toga se u DOV unose obično samo nazivi veza i kardinalnost preslikavanja.

Atribut i domen. Objekti u sistemu opisuju se preko svojih ATRIBUTA (svojstava). Na primer, atributi objekta SLUŠALAC su IME, PREZIME, BR_IND. Atributi uzimaju vrednost iz skupa mogućih vrednosti. Ovi skupovi se nazivaju demenima.

Npr. atribut *BR_IND* uzima vrednosti iz skupa mogućih brojeva indeksa. Uz određene pretpostavke imena atributa i domena mogu se izjednačiti, što je i učinjeno na sl. 7.

Atribut se formalno može predstaviti kao preslikavanje iz skupa objekata datog tipa u skup vrednosti (domen).

U zavisnosti od kardinalnosti preslikavanja, postoje različite vrste atributa:

a) jednoznačni atribut objekta (DG =1 i GG=1);

b) identifikator objekta (i kod inverznog preslikavanja DG=1 i GG=1);

c) višeznačni atribut (DG=1 i GG =M).

Uslov za postojanje atributa je DG>O. Ovde se može uočiti razlika između koncepta veze i koncepta atributa.

Veza definiše dva preslikavanja, direktno i inverzno, između dva tipa objekta, a *atribut* preslikavanja između tipa objekta i odgovarajućeg domena.

Pored navedenih koncepata postoje drugi koncepti MOV koji služe za semantički bogatiji opis realnog sistema:

— *slabi objekat* je onaj koji zavisi od nekog drugog objekta i u sistemu ne može da se identifikuje nezavisno od tog objekta *(identifikaciona zavisnost)* [3].

Primer slabog tipa objekta je predmet (sl. 7). Na primer, ne radi se o istom pojavljivanju »matematike« za redovno i vanredno školovanje (pošto se radi o dva različita nastavna plana i programa). Ovo ima za posledicu da je tip objekta Predmet identifikaciono zavistan od tipa objekta NPP;

— *generalizacija i specijalizacija (podtipovi).* Ovaj vid apstrakcije je već definisan a u MOV ima eksplicitnu podršku. Na primer, specijalizacijom tipa objekta slušalac dobili smo podtip vanredni slušalac (VANR-SLUŠ). To znači da je svaki vanredni slušalac i slušalac (veza S), i da nasleđuje sve atribute svoga nadtipa (slušalac), kao i da može imati dodatne (vojna pošta). Naravno, u skladu sa ovim tipom apstrakcije podtip nasleđuje i sve veze i preslikavanja svog nadtipa.

Definisanjem podtipova može se razrešiti problem tzv. opcionih atributa (DG=0 kao i u pojedinim slučajevima višeznačnih atributa (GG>1) [17];

— *agregirani ili mešoviti tip objekat-veza.* Ovaj tip podržava koncept agregacije u MOV. Ovaj koncept je implicitno uveden u MOV preko koncepta atributa (objekat je agregacija njegovih atributa). U ovoj verziji MOV problem višestrukih veza i atributa veza razrešava se upotrebom ovog koncepta (ova verzija modela dozvoljava samo binearne veze i nema koncept atributa veze). U našem primeru agregacija položio (sl. 7) nastala je kao posledica postojanja atributa odnosa (veze) između slušaoca i predmeta.

Ograničenja u modelu objekti-veze

Samom strukturom MOV definišu se ograničenja — kardinalnost preslikavanja. Međutim, postoje i mnogo složenija semantička ograničenja koje je nemoguće ili nepraktično prikazati strukturom MOV. Zbog toga se može definisati jezik za njihovu specifikaciju [7, 14, 15]. Ova ograničenja mogu se podeliti na sledeće vrste:

(1) ograničenja na domene atributa;

(2) vrednosna ograničenja;

(3) strukturna ograničenja.

Međutim, specifikacija ovih ograničenja nije dovoljna za specifikaciju IS, neophodno je uz svako ograničenje specificirati i akciju koju treba preduzeti kada je neka operacija nad bazom podataka narušila ovo ograničenje. Definicija ograničenja, zajedno sa akcijom koju treba preduzeti kada je ono narušeno, naziva se pravilom integriteta [14]. U ovom radu specifikacija pravila integriteta daje se u implementacionom (relacionom) modelu, tako da nije učinjeno proširenje MOV u ovom smislu. Prava dobit od specifikacije ovih ograničenja u MOV bila bi ako bi se MOV tretirao kao izvršna specifikacija. Naravno, za tako nešto neophodan nam je odgovarajući CASE alat koji bi podržavao ovakav pristup. [15].

Operacije u modelu objekti-veze

Operacije za rukovanje podacima na nivou MOV nisu bile odgovarajuće definisane u vremenu nastanka samog modela. Prvi radovi se, uglavnom, koncentrišu na prikaz upotrebe sredstava za opisivanje sistema upotrebom MOV. Kasnije je predloženo niz različitih jezika za rukovanje (manipulaciju) podacima u okviru MOV. Mada postoji niz radova, nije usaglašeno šta je osnova za manipulaciju, odnosno koji tipovi mogu biti operandi. Većina radova ne daje potpun skup operatora nad strukturom MOV.

Operacije kojima se generišu tipovi objekata (sa svojim atributima) i tipovi veza su statičke prirode — njima se definiše struktura modela. Ove operacije se, najčešće, zadaju preko nekog interfejsa (editora), mada može biti i tekstualni specifikacioni jezik za opis strukture MOV.

Dinamičkim operacijama vrši se »oživljavanje« modela. Njima se tipovi objekata i veza tretiraju kao ekstenzije — skupovi objekata i veza. Dakle, ove operacije MOV se definišu nad objektima i vezama kao osnovnim konceptima ovog modela podataka. To su stanrardne operacije:

— *ubacivanje (kreiranje) objekta.* Ovom operacijom se kreira objekat koji pripada određenom tipu objekta. Argument za ovu operaciju je skup imenovanih vrednosti za odgovarajuće ključeve (jedan za regularni, a dva ili više za slabi ili agregirani objekat) i skup imenovanih vrednosti odgovarajućih atributa;

— *izbacivanje (uništavanje) objekta.* Ovom operacijom izbacuje se objekat, određeni tip objekta, iz baze. Ne-

ophodan argument za ovu operaciju je jedinstveni identifikator (ključ) objekta;

— *ažuriranje objekta.* Ovom operacijom ažuriraju se vrednosti atributa objekta. Argument za ovu operaciju je identifikator objekta i skup imenovanih vrednosti koje predstavljaju vrednosti ažuriranih atributa;

— *selekcija objekata.* Ovom operacijom izdvajaju se objekti koji zadovoljavaju neki uslov. Dakle, ova operacija kao argument prihvata funkciju čija je funkcionalnost *objekat* - -> *boolean* i na osnovu nje izdvaja objekte koji se evaluacijom ove funkcije preslikavaju u istinitosnu vrednost;

Projekcija objekta. Ova operacija je posebno pogodna za generisanje izveštaja. Možemo reći da ona izdvaja podskup atributa od tipa objekta koji je u ulozi operanda. Ako uzmemo da se interno formira nov tip objekta sa ovim podskupom atributa, onda se ovom operacijom vrši svojevrsna generalizacija tipa objekta. Kao argument ova operacija prihvata listu atributa i specifikaciju objekta nad kojim se izvršava.

— *povezivanje dva objekta (kreiranje veze).* Ova operacija kao argument prihvata specifikacije objekata koje treba povezati i naziv uloge jednog od specificiranih objekata u toj vezi;

— *razvezivanje dva objekta (izbacivanje veze).* Ova operacija prihvata iste argumente kao i prethodna i vrši razvezivanje specificiranih objekata.

Sama struktura MOV i ograničenja definisana samom strukturom (DOV) ili posebnim jezikom za specifikaciju ograničenja definiše semantiku ovih operacija u smislu da su posle njihovog izvršenja zadovoljeni svi uslovi integriteta. Kako se u ovom radu MOV implementira prevođenjem u relacioni model, detaljnija specifikacija skupa ovih operacija neće biti ni data.

Međutim, predvođenje mora rezultovati ne samo u strukturi relacionog modela, već treba da dâ i skup ograničenja i procedura zadovoljavanja integriteta odgovarajućih operacija MOV [4]. Zbog toga, u sledećem delu se i daje primer specifikacije održavanja integriteta na nivou implementacionog (relacionog) modela.

Dakle, može se uočiti da model objekti-veze predstavlja samo *intelektualni alat* za definisanje modela podataka realnog sistema, odnosno za opisivanje skupa povezanih objekata realnog sistema i događaja u njemu preko jednog složenog apstraktnog tipa podataka (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 vrši modeliranje. Mogu se dati samo opšte metodološke preporuke, opšti metodološki pristupi, kao pomoć u ovom složenom poslu.

Relacioni model

Ranije je istaknuto da se u početnim fazama razvoja (I) nastoji obuhvatiti što više semantike realnog sistema, pa se koriste semantički bogati modeli podataka (u ovom radu MOV). Za takve modele ne postoje komercijalno raspoloživi softveri, koji bi ih učinili implementacionim (izvršnim). Semantički bogati modeli se zato prevode u semantički manje bogate modele (u ovom radu relacioni) koji su podržani komercijalnim softverom. Ovde je, kao implementacioni model, izabran relacioni model kao najpopularniji model druge generacije, koji najlakše podržava prototipski razvoj korišćenjem jezika četvrte generacije i CASE alata. Dalje se prikazuje postupak prevođenja MOV u relacioni model i specifikacija statičkih i dinamičkih integriteta, kao i primeri implementacije u ORACLE bazi podataka.

Prevođenje modela objekti-veze u relacioni model

Prevođenjem modela objekti-veze u relacioni model treba da se definiše:

- *relaciona šema*, odnosno skup relacija preko koga se implementira struktura modela objekti-veze, sa definisanim ključem za svaku relaciju;
- *uslovi integriteta* za svaki spoljni ključ dobijene relacione šeme. Preko ovih uslova integriteta implementiraju se oni koncepti strukture modela objekti-veze (veze i njihova kardinalnost, odnosi tipova i podtipova, odnosi agregacije i objekata koje ih čine, kao i odnosi jakih i slabih objekata) koje nije moguće implementirati strukturom relacionog modela (relacionom šemom).
- *način ostvarivanja uslova integriteta*, odnosno skup akcija nad bazom podataka, koje treba preduzeti pre ili posle svake operacije održavanja baze podataka (dodavanja, izbacivanja svakog objekta ili veze i izmene vrednosti identifikatora objekta).

Pravila za konstruisanje relacione šeme: [3].

Objekti. Posmatra se redom svaki objekat u modelu i sve njegove neposredne veze i postupa na sledeći način:

(1) svaki objekat postaje relacija,

(2) atributi ove relacije:

- svi atributi objekta;
- identifikator svih objekata koji su u vezi sa posmatranim objektom, prema kojima preslikavanje ima kardinalnost (1.1.);
- identifikator nadtipa, ako je posmatrani objekat podtip;
- identifikator nadređenog objekta ako je posmatrani objekat slabi objekat;
- identifikator objekta koji čine agregaciju ako je posmatrani objekat agregacija;

(3) ključ relacije je:

- identifikator objekta za objekte koji nisu podtipovi, agregacije ili slabi objekti;
- identifikator nadtipa, ako je posmatrani objekat podtip;
- složeni ključ koji se sastoji od svih identifikatora agregirajućih objekata, osim jednog od onih koji ulaze u agregaciju sa preslikavanjem GG = 1, za agregirani objekat;
- složeni ključ koji se sastoji od identifikatora nadređenog objekta i atributa slabog objekta koji, jednostavno, identifikuje jedno preslikavanje slabog objekta u okviru jednog pojavljivanja nadređenog, za slabe objekte.

Veze. Gornjim pravilima za prevođenje objekta u relacije već su u relacionom modelu implementirane sve veze koje imaju bar jedno preslikavanje sa kardinalnošću (1.1), sve veze agregirajućih prema agregiranim objektima, veze nadtipova i podtipova i veze nadređenih i podređenih objekata. Nisu implementirane veze u kojima nijedno preslikavanje nema kardinalnost (1.1).

- Veze u kojima oba preslikavanja imaju GG>1 (=M) postaju relacije. Atributi ove relacije su identifikatori objekata koji su u vezi i oba čine složeni ključ relacije.
- Veze u kojima jedno preslikavanje ima kardinalnost (0,1), a drugo GG>1>(=M) postaju relacije. Atributi ovakvih relacija su identifikatori objekata koji su u vezi, a ključ je identifikator objekta čije preslikavanje ima kardinalnost GG>1 (=M).
- veze u kojima oba preslikavanja imaju kardinalnost (0,1) postaju relacije. Atributi ovakvih

relacija su identifikatori objekata koji su u vezi i oba su kandidati za ključ.

Pridržavajući se navedenih pravila, prevođenje MOV-a bi izgledalo:

(1) Objekti KATEDRA, NASTAVNIK, SMER, JEDINICA, NPP nisu ni slabi, ni agregacije, ni podtipovi. Oni nemaju ni u jednoj vezi preslikavanja sa kardinalnošću (1,1) prema drugim objektima. Zbog toga odgovarajuće relacije imaju samo atribute tih objekata, a ključevi su im njihovi identifikatori.

KATEDRA (ŠIF KATEDR,NAZIV_KATEDR),

NASTAVNIK (ŠIF NAST, IME, PREZIME),

SMER (ŠIF SMER,NAZIV_SMER),

JEDINICA(SIF_JED, NAZIV JED).

NPP (SIF NPP, GOD_DON, NASLOV)

(2) Objekat SLUŠALAC nije ni slab, ni agregacija, ni podtip, ali ima preslikavanje S iz JED sa kardinalnošću (1,1), pa se identifikator objekta JEDINICA(SIF-JED,NAZIV JED) prenosi kao atribut u njemu ogovarajuću relaciju.

SLUŠALAC (BR_IND, IME, PREZIME,ŠIF_JED)

(3) Objekat POLOŽIO je agregacija objekata SLUŠALAC i PREDMET. Nije u vezi ni sa jednim drugim objektom:

POLOŽIO(BR IND,ŠIF PRED,ŠIF NPP,DATUM,OCENA,BR_POL)

(4) Objekat VANR_SLUŠ je specijalizacija od objekta SLUŠALAC pa, prema tome, on nasleđuje sve atribute svog nadtipa, a zadržava i svoje:

VANR_SLUŠ(BROJ_IND,IME,PREZIME, ŠIF JED, VOJ POŠ)

(5) Objekat PREDMET je identifikaciono zavisan od objekta NPP, pa, prema tome, ne može da se identifikuje nezavisno od jakog objekta NPP.

PREDMET(ŠIF NPP,ŠIF _PREDM, BR_ČASOVA, NAZIV_PREDMETA)

(6) Veza *struktura* je veza između istih tipova entiteta, pa se oba preslikavanja imenuju i ulaze kao imena atributa relacije STRUKTURA (čiji je domen ključ tipa entiteta JEDINICA — šif jed).

STRUKTURA(SASTAV,U_SASTAVU)

Pri prevođenju veza u relacioni model usvaja se sledeća konvencija za imenovanje relacija:

(7) Za veze čija oba preslikavanja imaju GG>1 (=M), dobijamo

SLUŠA(BR IND,ŠIF PRED)

PREDAJE(ŠIF NAST,ŠIF PRED)

IMA(ŠIF SMER,ŠIF PRED)

(8) Za vezu između objekta NASTAVNIK i KATEDRA čije jedno preslikavanje PRIPADA ima kardinalnost (0,1), a drugo (1,M), dobija se:

PITOMAC IZ SMERA(ŠIF SMERA, BROJ_IND)

PRIPADA(ŠIF_KATEDR, ŠIF_NAST)

Treba reći da primenom navedenih pravila, pod pretpostavkom da je model objekti-veze korektno napravljen, uvek dolazi do skupa relacija koje su u trećoj normalnoj formi i u kojima ne postoji *NULL* vrednost kao rezultat neprimenljivog svojstva.

Definisanje integriteta relacionog modela

Pravila integriteta omogućuju da se realni sistem preciznije opiše pomoću modela podataka (povećaju *seman-*

tičko bogatstvo modela podataka). Pod pojmom *integritet* podrazumevaćemo mehanizme za implementaciju *pravila integriteta*, odnosno mehanizme koji ograničavaju mogući skup stanja baze podataka na samo dozvoljene skupove.

Razlikuju se dve vrste pravila integriteta:

- inherentna pravila,
- eksplicitna pravila.

Inherentna pravila integriteta ugrađena su u samu strukturu modela podataka i nije ih neophodno eksplicitno iskazivati (kardinalnost i totalnost u MOV-u i pojam ključa u relacionom modelu).

Eksplicitna pravila integriteta iskazuju se statičkim i dinamičkim pravilima.

Statičkim pravilima iskazuju se uslovi koji moraju važiti *pre* i *posle* izvršenja bilo koje operacije nad bazom podataka.

Dinamičkim pravilima integriteta definisane su procedure u relacionom modelu, koje odgovaraju apstraktnim operacijama MOV-a i koje garantuju ostvarenje uslova integriteta.

Statička pravila integriteta

Statička pravila integriteta treba da budu definisana za svaki spoljni ključ dobijenog skupa relacija. Pravila su sledeća [3]:

1. Ako je u relaciji R spoljni ključ A rezultat implementacije preslikavanja sa kardinalnošću (1,1) odgovarajućeg objekta R prema nekom objektu V čije je A identifikator, tada:

a) ako je u inverznom preslikavanju vrednost DG = O, važi $P(A)R \subseteq \subseteq P(A)V$;

b) ako je u inverznom preslikavanju DG =1, važi P(A)R=P(A)V.

U ovom slučaju:

$P(ŠIF\_JED)SLUŠALAC \subseteq \subseteq P(ŠIF\_JED)JEDINICA$

2. Ako je u relaciji R, koja predstavlja slabi objekat, spoljni ključ A identifikator nadređenog objekta V, tada je:

$P(A) \subseteq P(A)V$

U ovom slučaju:

$P(ŠIF\_NPP)PREDMET \subseteq \subseteq P(ŠIF\_NPP)NPP$

3. Ako je relacija R, koja predstavlja agregaciju ili vezu, spoljni ključ A identifikator objekta V koji učestvuje u agregaciji ili vezi i ako:

a) njegovo preslikavanje ima kardinalnost DG = 0 važi:

$P(A)R \subseteq P(A)V$

b) njegovo preslikavanje ima kardinalnost DG = 1, važi:

P(A) = P(A) V

U ovom slučaju:

$P(BR\_IND)\ POLOŽIO \subseteq \subseteq P(BR\ IND)SLUŠALAC$
$P(ŠIF\_\ PRED)POLOŽIO \subseteq \subseteq P(ŠIF\_PRED)PREDMET$
P(BR_IND)SLUŠA = P(BR_IND) SLUŠALAC
$P(ŠIF\_PRED,ŠIF\_NPP)SLUŠA \subseteq \subseteq (ŠIF\_PRED,ŠIF\_NPP)PREDMET$
$P(ŠIF\_NAST)PREDAJE \subseteq \supseteq P(ŠIF\_\ NAST)\ NASTAVNIK$
$P(ŠIF\_PRED)PREDAJE \subseteq \subseteq P(ŠIF\_PRED)PREDMET$
P(ŠIF_SMER)IMA=P(ŠIF_SMER) SMER
P(ŠIF_PRED)IMA = P(ŠIF_PRED) PREDMET
$P(BROJ\_IND)PIT\_IZ\ SMERA \subseteq \subseteq P(BROJ\_IND)SLUŠALAC$
$P(ŠIF\_SMER)PIT\ \ IZ\_SMERA \subseteq \subseteq (ŠIF\_SMER)SMER$
$P(ŠIF\_KATEDR)PRIPADA = = P(ŠIF\_KATEDR)KATEDRA$
$P(ŠIF\_\ NAST)PRIPADA \subseteq \subseteq P(ŠIF\ \ NAST)NASTAVNIK$

Struktura:

P(SASTAV)STRUKTURA⊆
⊆P(ŠIF JED)JEDINICA
P(U SASTAVU)STRUKTURA ⊆
⊆P(ŠIF JED)JEDINICA

Specifikacija dinamičkih uslova integriteta

Specifikacija dinamičkih uslova integriteta sastoji se u određivanju dozvoljenih operacija za pojedine objekte i veze MOV-a i određivanju načina zadovoljavanja integriteta za svaku operaciju koja se potencijalno narušava.

Mogući načini zadovoljavanja integriteta su:

RESTRICTED — integritet baze podataka se zadovoljava poništavanjem operacije koja ga narušava;

CASCADE — integritet se zadovoljava prenošenjem operacija na objekte kod kojih je došlo do narušavanja integriteta;

DEFAULT — integritet se zadovoljava povezivanjem sa default objektom, koji zamenjuje objekat čije je nepostojanje uzrok narušavanja integriteta;

NUULIFIES — integritet se zadovoljava povezivanjem sa null objektom, koji zamenjuje objekat čije je nepostojanje uzrok narušavanja integriteta.

PRIMER

SLUŠALAC

UBACI SLUŠALAC
RESTRICTED JEDINICA
IZBACI_SLUŠALAC
CASCADE SLUŠA, SLUŠ_SMER

PREDMET

UBACI_PREDMET
RESTRICTED NPP
IZBACI_PREDMET
RESTRICTED SLUŠA,IMA,NPP
CASCADES POLOŽIO

Kao što se iz primera vidi, procedure se definišu za operacije UBACI I IZBACI dva entiteta SLUŠALAC i PREDMET i veze koje ih prate.

Važi opšte pravilo da se definišu procedure za operacije UBACI i IZBACI onih koncepata MOV-a koji postaju relacije pri prevođenju u relacioni model.

Ukoliko se sve akcije uspešno završe, baza podataka je *provedena* u novo dozvoljeno stanje. Ukoliko se bar jedna akcija ne završi uspešno, baza podataka se *vraća* u stanje pre otpočinjanja prve akcije unutar procedure.

Prenos dejstva osnovne operacije na koncepte u vezi može se vršiti na sledeće načine:

a) nastavkom transakcije prenošenjem osnovne operacije na koncepte u vezi;

b) prekidom transakcije kod koncepata u vezi uz poništavanje prethodnih dejstava i završetkom transakcije kod koncepata u vezi postavljanjem vrednosti određenih atributa na *DEFAULT* ili *NULL* vrednosti.

Implementacija dinamičkih uslova integriteta u ORACLE **bazi podataka objekta slušalac**

Implementacija specificiranih dinamičkih uslova integriteta, pri korišćenju ORACLE SQL*Forms modula, postiže se primenom SQL naredbi koje se vezuju ili za vrednosti polja interaktivnih aplikacija ili za operacije nad bazom podataka.

SQL naredbe koje se specificiraju za polja omogućuju kontrolu vrednosti do nivoa karaktera i prikaz informacija iz povezanih tabela. Njima se realizuju gotovo svi RESTRICTED uslovi za operaciju *UBACI*.

Specificirani dinamički uslovi integriteta, koji se mogu implementirati SQL* SELECT naredbama na nivou polja (svi CASCADES i složeniji RES-

TRICTED uslovi) pri korišćenju SQL* Forms programskih modula, implementiraju se preko mehanizma *trigera*.

Postoji tri vrste operacija za ažuriranje baze podataka: INSERT, UPDATE i DELETE.

Trigeri se izvršavaju izdavanjem naredbe COMMIT (prenos dejstva na bazu) neposredno pre ili posle operacije na koju se odnose. Sa operacijama na koje se odnose čine atomsku transakciju (ili će se izvršiti i operacija i pridruženi triger ili se neće izvršiti ništa). Kako postoje 3 operacije za ažuriranje baze podataka, a trigeri se izvršavaju neposredno pre ili posle njih, postoje sledeći tipovi trigera:

- PRE_INSERT;
- POST_INSERT;
- PRE_UPDATE;
- POST_UPDATE;
- PRE_DELETE;
- POST_DELETE.

Za svaku SQL naredbu, bez obzira na to da li je definisana za polje ili unutar trigera, definiše se i poruka koja se javlja ukoliko izvršavanje SQL naredbe nije uspešno (obaveza ili upozorenje).

Ukoliko poruka ima značenje upozorenja, proces ažuriranja biće nastavljen, a ukoliko ima snagu obaveze — biće prekinut.

Za objekat SLUŠALAC, iz prikazanog modela, dinamički uslovi su implementirani SQL naredbama na sledeći način: za relaciju SLUŠALAC, koja odgovara objektu SLUŠALAC, definisan je blok interaktivne aplikacije SLUŠALAC. Za polje koje odgovara koloni ŠIF_JED definisana je SQL SELECT naredba:

```
SELECT ŠIF_JED
INTO SLUŠALAC.ŠIF_JED
FROM JEDINICA
WHERE ŠIF_JED# =:SLUŠA-
LAC.ŠIF  JED#
```

poruka: *Pogrešna šifra jedinice. Koriguj je ili izlistaj dozvoljene.

značenje: OBAVEZA

Njom je realizovan RESTRICTED uslov za operaciju UBACI_SLUŠAOCA.

CASCADES uslovi za operaciju IZBACI_SLUŠAOC realizovani su PRE_DELETE trigerom za blok SLUŠAOC sledećeg izgleda:

```
DELETE SLUŠA
WHERE ŠIF_PREDM# =:
:SLUŠALAC.ŠIF_PREDM#
```

poruka: *Izbrisani su redovi tabele SLUŠA.

značenje: OBAVEZA,

```
DELETE SLUŠA_SMER
WHERE ŠIF_SMER# =:
:SLUŠALAC.ŠIF_SMER#
```

poruka: *Izbrisani su redovi tabele SLUŠA_SMER.

značenje: OBAVEZA,

PRILOZI:

Prilog 1

Creiranje tabela u ORACLE V.6:

```
create table slusalac
    (broj_indeksa char(10) not null,
    prezime char(20) not null,
    ime char(15) not null,
    sif_jed char(5) not null,
    primary key (broj_indeksa))
    tablespace uspeh;

create table slusa
    (broj_indeksa char(10) not null,
    sif_predmeta char(5) not null,
    primary key (broj_indeksa))
    tablespace uspeh;
```

```
create table jedinica
    (sifra_jed char(5) not null,
    naziv_jedinice char(25) not null,
    primary key (sifra_jed))
    tablespace uspeh;

create table slusa_smer
    (bro._indeksa char(10) not null,
    sif_smera char(5) not null,
    primary key (broj_indeksa))
    tablespace uspeh;
```

Prilog 2

Primer realizacije PRE-INSERT trigera u ORACLE V.6:

```
DEFINE TRIGGER
  NAME = PRE-INSERT
  TRIGGER_TYPE = V3
  SHOW_KEY = ON
  DESCRIPTION =
  TEXT = <<<
  if :slusa.sif predmeta iz null then
    message('Slusalac mora slusati
    bar jedan predmet!');
    raise form_trigger_failure;
  end if;
  ENDDEFINE TRIGGER
```

Prilog 3

Primer realizacije PRE-DELETE trigera u ORACLE V.6:

```
DEFINE TRIGGER
  NAME = PRE-DELETE
  TRIGGER_TYPE = V3
  SHOW_KEY = ON
  DESCRIPTION =
  TEXT = <<<
  lock table slusa in share update
  mode;
  delete from slusa where BROJ_
  INDEKSA = :slusalac.BROJ_IN-
  DEKSA;
  lock table slusa_smer in share
  update mode;
  delete from slusa_smer where
  BROJ_INDEKSA = :slusalac.
  BROJ_INDEKSA;
ENDDEFINE TRIGGER
```

Prilog 4

Definisane procedure koje koriste trigeri

```
DEFINE PROCEDURE
  NAME = check_package_failure
  DEFINITION = <<<
  procedure check_package_ failure is
  begin
   if not form_success then
    raise form_trigger _failure;
   end if;
  end;

ENDDEFINE PROCEDURE

DEFINE PROCEDURE
 NAME = clear_slusalac_details
 DEFINITION = <<<
 procedure clear_slusalac details is
 begin
  if (name_in('slusalac.BROJ_INDE-
  KSA') is not null) then
  go_block('slusa');
  check_package_failure;
  if :system.block_status = 'CHAN-
  GED' then
   clear_block(ASK_COMMIT);
   if :system.block_status = 'CHA-
   NGED' then
    go_block('slusalac');
    raise FORM_TRIGGER_FAI-
    LURE;
   end if;
  end if;
  clear_block;
  go_block('slusalac');
 end if;
end;

ENDDEFINE PROCEDURE
DEFINE PROCEDURE
 NAME = query_slusalac_details
```

```
DEFINITION = <<<
procedure query_slusalac_details is
begin
 if (name in('slusalac.BROJ_INDE-
 KSA') is not null) then
  go block('slusa');
  check_package failure;
  execute query;
  go_block('slusalac');
 end if;
end;
ENDDEFINE PROCEDURE
```

Zaključak

U radu je opisan jedan pristup projektovanju informacionih sistema, zasnovan na prototipskom razvoju upotrebom standardnih alata i metoda. U fazi analize sistema koristi se metoda strukturne sistem-analize, a u fazi modeliranja podataka jedan od najpopularnijih semantičkih modela podataka — MOV.

Kao implementacioni model izabran je relacioni model. Pokazano je da je u odsutnosti CASE alata, koji direktno podržavaju objektivno-transvormacioni pristup [14], ovo sasvim zadovoljavajuće rešenje.

Pristup je ilustrovan na primeru jednog segmenta informacionog sistema Univerziteta Vojske Jugoslavije (praćenje uspeha slušalaca). U daljem radu na ovom sistemu svakako treba koristiti CASE alate koji bi podržavale i automatizovale pojedine faze u ovom pristupu. To se, pre svega, odnosi na potpuniju operacionalizaciju modeliranja podataka kroz MOV kao izvršne specifikacije.

Literatura:

[1] B. Lazarević, V. Jovanović, M. Vučković: »Projekovtanje informacionih sistema I deo«, Naučna knjiga, Beograd, 1988.

[2] Z. Marjanović: »Korišćenje relacionih podataka«, Beograd: Sfairos, 1989.

[3] B. Lazarević: »Model objekti — veze«, Materijal za interne kurseve, Beograd, 1989.

[4] B. Lazarević: »Baze podataka«, Radni materijal za interne kurseve, Beograd, mart 1989.

[5] V. Jovanović: »Modeliranje podataka«, priručnik za slušaoce, FON-Beograd, 1989.

[6] S. Alagić: »Relacione baze podataka«, Svjetlost, Sarajevo, 1984.

[7] B. Lazarević: »Prošireni model objekti-veze«, Interni materijal, FON, 1990.

[8] B. Lazarević, S. Nešković: »Tendencije u razvoju pristupa projektovanju informacionih sistema sa posebnim osvrtom na objektno orjentisani prototipski razvoj«, Implementaciono--upravljački sistemi '89, Dubrovnik 17.—22. 04. 1989.

[9] W. D. Potter, R. P. Truebload: »Traditional, Semantic, and Hyper-Semantic Approaches to Data Modeling«, IEEE Computer, June 1988.

[10] W. Kim: »Introduction to Object-Oriented Databases«, Cambridge: MIT Press, 1990.

[11] P. P. Chem, »The Entity — Relationship Model — Towards a Unifed View of Data«, ACM TODS, Vol. 1, No. 1, 1976.

[12] C. G. Davis, S. Jajodia, P. A-B. Ng. i R. T. Yeh (Editors), »Entity — Relationship Approach to Software Engineering«, North--Holand, Amsterdam, the Netherlands, 1983.

[13] C. Parent, S. Spaccopietre: »Complex Objects Modeling: An Entity — Relationship Approuch«, Nested Relations and Complex Objects in Databases, Springer-Verlag, 1989.

[14] B. Lazarević, S. Nešković: »Objektivno orjentisana transformaciona metoda razvoja informacionih sistema«, SIMOPIS, Kupari, 1990.

[15] S. Ilić: »Specifikacija CASE alata za podršku projektovanju informacionih sistema korišćenjem proširenog modela objekti-veze«, SIMOPIS, Kupari, 1990.