

8

PROJEKTOVANJE RELACIONE BAZE PODATAKA

Iz sadržaja dosadašnjih poglavlja uverili smo se u valjanost šeme relacione baze podataka BIBLIOTEKA (Prilog A) koja nam je do sada služila kao osnovni primer. Niti smo nailazili na smetnje pri formulisanju upita nad tom bazom podataka, niti je u njoj prisutna i jedna od anomalija koje smo naveli u prethodnom poglavlju. Sa druge strane, pojedine izmenjene šeme relacija iz te baze podataka poslužile su nam za ilustraciju tih anomalija. Jednom rečju, šema relacione baze podataka BIBLIOTEKA je savršena u svakom pogledu.

Pitanje koje se prirodno nameće je: *kako* je nastala šema relacione baze podataka BIBLIOTEKA? Da li na osnovu osećaja i slobodne procene ili na neki drugi način? Kako u praksi nastaju isto tako savršene šeme relacionih baza podataka, ali sa stotinama ili više relacija?

Odgovor na sva ta pitanja je kratak i jasan i glasi: savršenost šeme relacione baze podataka postiže se pažljivim postupkom projektovanja, pri čemu se striktno poštuju određena pravila. Sam postupak projektovanja svakog informacionog sistema je dvojakog karaktera i obuhvata:

- projektovanje podataka, odnosno šeme relacione baze podataka;
- projektovanje postupaka, odnosno procesa za održavanje i korišćenje podataka.

Pri tome je od najvećeg značaja prvi vid projektovanja, iz razumljivih razloga: nikakvo projektovanje postupaka ne može da otkloni nedostatke i anomalije loše šeme relacione baze podataka, dok se posledice lošeg projektovanja postupaka nad valjanom šemom mogu naknadno otkloniti.

U ovom poglavlju razmatraćemo oba vida projektovanja relacione baze podataka odnosno informacionog sistema zasnovanog na njoj. Predmet našeg interesovanja biće prvo metod projektovanja podataka poznat pod nazivom "model objekata i odnosa". Ovaj model čine: skup grafičkih simbola, skup pravila korektnosti i skup pravila prevođenja u model nižeg reda koji se direktno može prevesti u odgovarajuće strukture podataka. Nakon toga osvrnućemo se na projektovanje postupaka metodom funkcionalne dekompozicije i algoritamske specifikacije.

8.1 Projektovanje podataka

Kao što je već navedeno u poglavlju 3, model objekata i odnosa spada u grupu modela koji su semantički višeg reda, i predstavlja mešavinu modela sistema i modela podataka. Preciznije rečeno, model objekata i odnosa se nalazi "iznad" relacionog modela podataka, pa kao takav sadrži i pravila prevođenja u relacioni model. Važno je imati na umu da se ovim modelom predstavljaju *klase* objekata i odnosi između njih, a ne pojedinačne instance objekata.

Razmatranje modela objekata i odnosa koje sledi sprovedeno je u skladu sa podelom entiteta na specijalne slučajeve objekata i veza. Na kraju, izložen je postupak prevođenja tog modela u relacioni model i dat je primer projektovanja relacione baze podataka BIBLIOTEKA.

8.1.1 Nezavisne klase objekata

Definicija

Nezavisna klasa objekata u posmatranom sistemu je svaka klasa objekata čije instance egzistiraju nezavisno od instanci drugih klasa objekata u sistemu.

Navedimo dva primera za naš sistem BIBLIOTEKA:

- klasa CLAN je nezavisna, pošto članovi kao osobe egzistiraju potpuno nezavisno od instanci drugih klasa;
- klasa NASLOV je takođe nezavisna: na prvi pogled, egzistencija naslova uvek zavisi od autora, ali to nije tako - primer za to je naslov koga izdavač prvo osmisli a zatim nalazi autore koji će da ga napišu.

Grafički simbol za nezavisnu klasu objekata je pravougaonik iscertan jednostrukom linijom. Unutar tog pravougaonika u pregrađenom delu pri vrhu ispisujemo velikim slovima naziv klase, a u delu ispod toga ispisujemo jedan ispod drugog nazive klasifikacionih svojstava, pri čemu podvlačimo identifikator (klasifikaciono svojstvo koje svojom vrednošću identifikuje svaku instancu u klasi).

Pravilo prevođenja nezavisne klase objekata u odgovarajući koncept relacionog modela je jednostavno:

- od nezavisne klase objekata nastaje šema relacije istog naziva i sa atributima koji odgovaraju klasifikacionim svojstvima, pri čemu je atribut koji odgovara identifikatoru primarni ključ.

Primer

Za klasu CLAN u našem primeru biblioteke imamo sledeći simbol modela objekata i odnosa i odgovarajuću šemu relacionog modela:



8.1.2 Zavisne klase objekata

Definicija

Zavisna klasa objekata u posmatranom sistemu je svaka klasa objekata čije instance egzistiraju zavisno od konstantnog broja instanci drugih klasa objekata u sistemu.

Iz prethodne definicije zaključujemo da modelom objekata i odnosa ne može da se predstavi zavisnost od promenljivog broja instanci. Tako, u našem primeru biblioteke nismo u mogućnosti da klasu NASLOV predstavimo zavisnom od klase AUTOR, pošto pojedini naslovi imaju različiti broj autora.

Sistem BIBLIOTEKA naveden u poglavlju 1 je suviše jednostavan i kao takav ne može da posluži kao primer za zavisne klase objekata, ali ga u tu svrhu možemo dopuniti (što je i urađeno u prilogu A) praćenjem sledećih događaja:

- POZAJMICA: skup svojstava koji govori o tome da je određeni član pozajmio određenu knjigu određenog naslova i određeni broj dana;
- REZERVACIJA: skup svojstava koji govori o tome da je određeni član rezervisao određeni naslov određenog datuma.

Navedena dve klase su zavisne, i to iz sledećih razloga:

- ni jedna pozajmica ne može da egzistira nezavisno od člana koji ju je izvršio i knjige koja je pozajmljena;
- ni jedna rezervacija ne može da egzistira nezavisno od člana koji ju je izvršio i naslova koji je rezervisan.

Uz to, za obe klase važi da su zavisne od po dve klase objekata. To ne mora biti uvek slučaj: zavisnost može biti od jedne, dve ili više klasa.

Zavisnost klase objekata može biti dvojaka, prema stepenu zavisnosti:

- egzistencijalna: instance zavisne klase poseduju identifikaciono svojstvo, a zavisna je samo njihova egzistencija;
- identifikaciona: instance zavisne klase ne poseduju identifikaciono svojstvo, i u njihovoj identifikaciji učestvuju i identifikatori klasa od kojih zavise.

Identifikaciona zavisnost je u većini slučajeva nepoželjna, pošto nameće problem određivanja skupa identifikacionih svojstava a uz to može biti uzrok problema u obezbeđivanju dinamičkih uslova integriteta u bazi podataka. Takva zavisnost se uvek može izbeći time što se za zavisnu klasu objekata veštački uvodi identifikaciono svojstvo u vidu rednog broja (brojača nastanka) instance.

Zavisna klasa objekata se predstavlja pravougaonikom iscrtanim dvostrukom linijom. Unutar njega u pregrađenom delu ispisujemo velikim slovima naziv klase, a u delu ispod toga ispisujemo jedan ispod drugog nazive klasifikacionih svojstava. Ako postoji identifikator, podvlačimo ga, kao i kod nezavisne klase objekata. Ako postoji jedno ili više svojstava koji uz identifikatore uslovljavajućih klasa učestvuju u identifikaciji, podvlačimo ih isprekidano. Na kraju, usmerenim isprekidanim linijama spajamo takav pravougaonik sa pravougaonicima koji odgovaraju svim klasama od kojih postoji zavisnost. Uz te linije naznačujemo vrste zavisnosti: E ili ništa za egzistencijalnu, I za identifikacionu.

Zavisnost između klasa objekata je tipičan slučaj neposrednog osnosa klasa, a kako su klase skupovi instanci objekata taj odnos možemo posmatrati i kao odnos preslikavanja između dva skupa. Na osnovu toga, mogu se za svaku zavisnost od neke klase definisati dve vrste kardinalnosti preslikavanja:

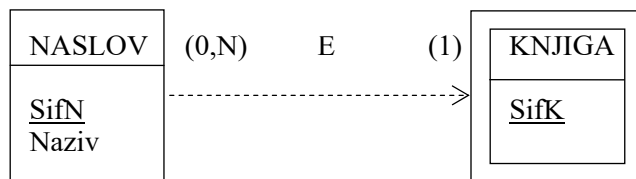
- kardinalnost uslovljavanja: par brojeva (m,n) koji se zapisuje uz klasu od koje je zavisnost i koji ima sledeće značenje: svaka instanca te klase mora da uslovljava bar m a može najviše n instanci zavisne klase;
- kardinalnost uslovljenosti: broj (m) koji se zapisuje uz zavisnu klasu i koji ima sledeće značenje: svaka instanca te klase mora biti zavisna od tačno m instanci klase od koje zavisi.

Pravilo prevođenja za zavisnu klasu objekata je znatno složenije nego za slučaj nezavisne klase:

- pre prevođenja zavisne klase objekata u odgovarajući koncept relacionog modela, mora biti sprovedeno takvo prevođenje za sve klase od kojih postoji zavisnost;
- od zavisne klase objekata nastaje šema relacije istog naziva, a attribute šeme pored klasifikacionih svojstava te klase čine i primarni ključevi šema relacija svih klasa od kojih postoji zavisnost; svaki primarni ključ uzima se onoliko puta kolika je kardinalnost odgovarajuće uslovljenosti, uz eventualnu promenu naziva;
- ako je zavisnost egzistencijalna (sve zavisnosti od drugih klasa su egzistencijalne), identifikator zavisne klase objekata postaje primarni ključ nastale šeme relacije; u suprotnom, za primarni ključ se uzima ona minimalna kombinacija atributa koja se po prirodi stvari može javiti samo jednom.

Primer:

U proširenom sistemu biblioteke, koji je poslužio kao osnova za bazu podataka BIBLIOTEKA, klasa KNJIGA je zavisna od klase NASLOV, s obzirom na to da ne postoji "prazna" knjiga i da svaka knjiga sadrži tačno jedan naslov. Ako sa SifK označimo identifikator (inventarni broj knjige), imamo:



⇓

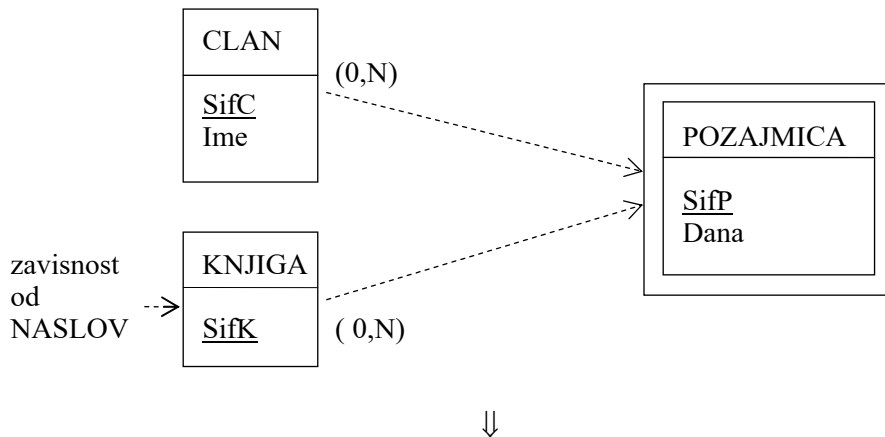
NASLOV (SifN, Naziv)

KNJIGA (SifK, SifN)

Uobičajeno je da se egzistencijalna zavisnost i kardinalnost uslovljenosti (1) posebno ne naznačuju, pa ćemo se toga pridržavati u narednim primerima.

Primer:

Za već opisanu situaciju zavisne klase POZAJMICA imamo:



CLAN (SifC, Ime)

POZAJMICA (SifP, SifC, SifK, Dana)

KNJIGA (SifK, SifN)

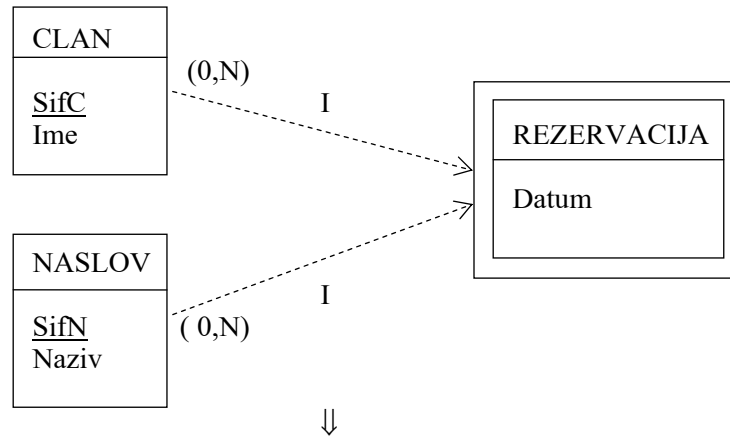
Ovde su neophodne određene napomene:

- šema relacije KNJIGA sadrži atribut SifN na osnovu zavisnosti od klase NASLOV, što nije u celini prikazano;
- ni jedna kombinacija atributa SifC, SifK i Dana nema osobinu unikatnog pojavljivanja vrednosti, pa nije mogla da posluži kao primarni ključ (moguće je, mada malo verovatno, da isti član pozajmi istu knjigu u trajanju od istog broja dana, pogotovo ako je to jedini primerak određenog naslova);
- prethodna situacija je prevaziđena tako što smo za klasu POZAJMICA uveli identifikator SifP sa značenjem rednog broja pozajmice.

Prethodna dva primera odnosila su se na egzistencijalne zavisnosti.

Primer:

Za zavisnu klasu objekata REZERVACIJA koju smo već opisali imamo:



CLAN (SifC, Ime)

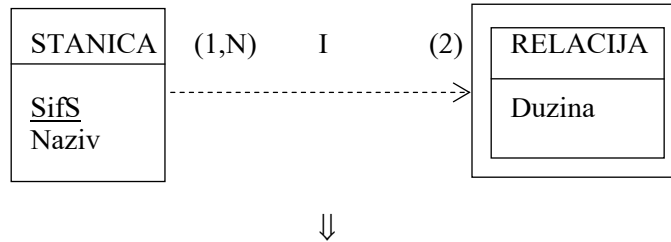
NASLOV (SifN, Naziv)

REZERVACIJA (SifC, SifN, Dana)

Ovde imamo primer identifikacione zavisnosti, a unikatnost kombinacije atributa SifC i SifN je jasna: niko ne rezerviše isti naslov dva puta.

Primer:

Posmatrajmo deo sistema železnice, u kome postoje stanice, a između po dve stanice relacije sa svojstvom dužine:



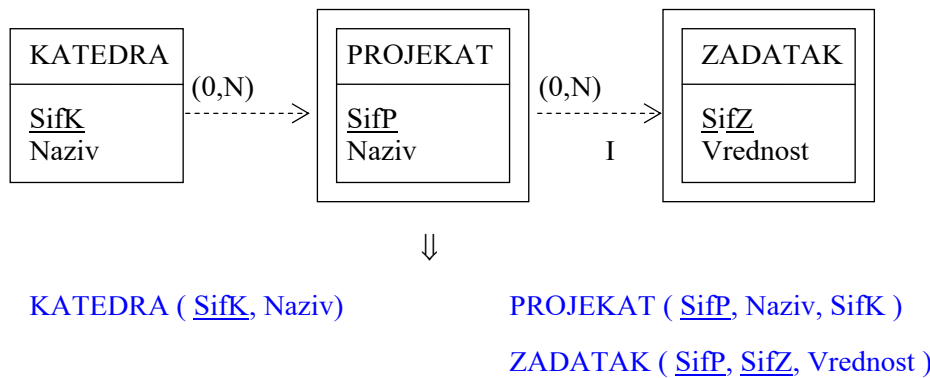
STANICA (SifS, Naziv)

RELACIJA (Duzina, SifSPoc, SifSZad)

Ovde imamo identifikacionu zavisnost sa kardinalnošću uslovljenosti 2, pa se zato primarni ključ šeme STANICA javlja dva puta u šemi RELACIJA. Pri tome smo bili u obavezi da promenimo naziv za bar jedno pojavljivanje ta dva atributa, ali smo promenili oba radi preglednosti, da bi naglasili šta je početna a šta zadnja stanica.

Primer:

Zavisnost klase objekata može biti i od druge zavisne klase objekata. Neka nam kao primer posluži neki fakultet na kome postoje katedre sa šifrom i nazivom kao svojstvima. Unutar katedri rade se projekti, od kojih svaki kao svojstva ima na nivou fakulteta unikatnu šifru i naziv. Konačno, unutar svakog projekta radi se jedan ili više zadataka, od kojih svaki ima kao svojstva šifru koja je unikatna za taj projekat i vrednost. Za opisanu situaciju imamo:



U složenijim situacijama mogu se javiti čitavi "lanci zavisnosti" klasa objekata, pa je jasno zašto je uzmeravanje isprekidane linije zavisnosti neophodno.

8.1.3 Specijalizacija i generalizacija

Definicija:

Za neku klasu objekata (podklasu) kažemo da predstavlja specijalizaciju ako predstavlja specijalan slučaj neke druge klase objekata (nadklase) po bar jednom od sledeća dva kriterijuma:

- ima specifična klasifikaciona svojstva;
- ima specifične odnose sa drugim klasama objekata.

Pored značenja specijalnog slučaja neke nadklase objekata, specijalizacija kao pojam ima i značenje postupka pri kome uočavamo da neka klasa objekata, nadklasa, ima po bilo kom od navedenih kriterijuma specijalne slučajeve. Pri tome, nadklasa će imati sva klasifikaciona svojstva koja su zajednička za sve specijalne slučajeve, dok će podklase imati, ako je to osnov specijalizacije, samo specifična svojstva.

Specijalizacija se u okviru modela objekata i odnosa predstavlja tako što se i za nadklasu i za podklase koristi dosadašnji način označavanja pomoću pravougaonika odgovarajućeg tipa i sadržaja, s tim što se od nadklase povlači puna linija koja preko trougla vodi do jedine podklase ili se grana do svih podklasa, ako ih ima više. Za nadklasu se navodi identifikator i sva svojstva koja su zajednička za sve podklase. Za svaku podklasu navode se samo njoj specifična svojstva, ako ih ima.

I specijalizacija predstavlja slučaj neposrednog odnosa klasa, odnosno preslikavanja instanci nadklase na instance podklasa, pa se i ovde definiše odgovarajuća kardinalnost:

- kardinalnost specijalizacije: par brojeva (m,n) koji se zapisuje uz nadklasu koja specijalizuje i koji ima sledeće značenje: svaka instanca nadklase specijalizira na po jednu instancu u najmanje m i najviše n podklasa.

Jasno je da n ne može biti veće od broja podklasa specijalizacije. Prema kardinalnosti specijalizacije mogu se izvesti dve podele. Prva je po m i obuhvata dva slučaja:

- parcijalna (neobavezna) specijalizacija: slučaj kada je $m=0$, odnosno kada u nadklasi mogu da postoje instance koje ne učestvuju u specijalizaciji;
- totalna (obavezna) specijalizacija: slučaj kada je $m>0$ (a najčešće 1), odnosno kada svaka instanca u nadklasi mora da specijalizuje u instance bar m podklasa.

Druga podela je na osnovu n i obuhvata takođe dva slučaja:

- ekskluzivna (isključujuća) specijalizacija: slučaj kada je $n=1$, odnosno kada svaka instanca nadklase može da specijalizuje u instancu najviše jedne podklase;
- inkluzivna (uključujuća) specijalizacija: slučaj kada je $n>1$, odnosno kada svaka instanca nadklase može da specijalizuje u instance najviše n podklasa.

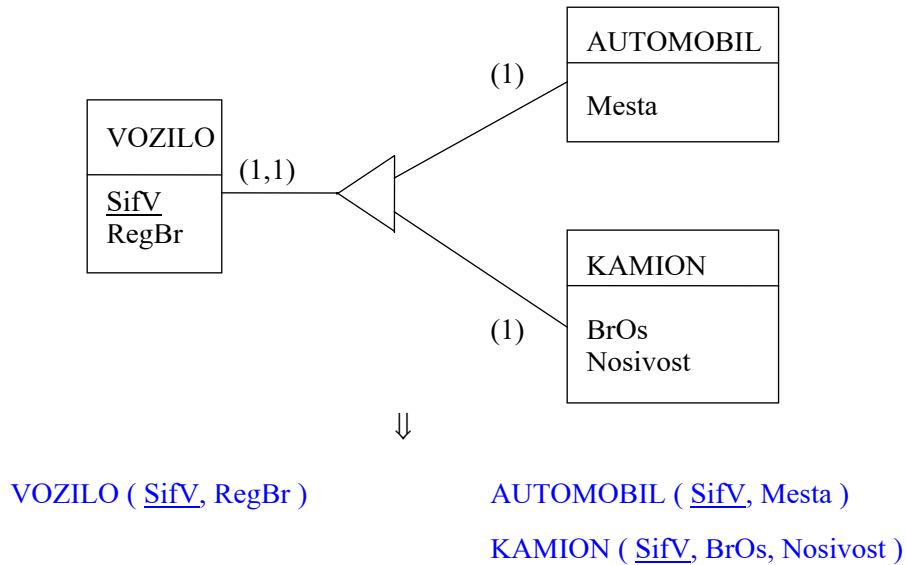
Pravilo prevođenja specijalizacije u odgovarajuće koncepte relacionog modela je jednostavno:

- od nadklase objekata koja specijalizuje nastaje šema relacija istog naziva i sa atributima koji odgovaraju klasifikacionim svojstvima, pri čemu je atribut koji odgovara identifikatoru primarni ključ;
- od svake podklase objekata nastaje šema relacije istog naziva, a attribute te šeme čine klasifikaciona svojstva podklase i primarni ključ šeme koja odgovara nadklasi, koji postaje i primarni ključ šeme podklase.

Jasno je da pre formiranja šema relacija podklasa moramo doći do šeme nadklase. U retkim slučajevima, moguća je i specijalizacija zavisnih klasa objekata. Isto tako, mogu postojati lanci specijalizacije sa dva i više nivoa.

Primer:

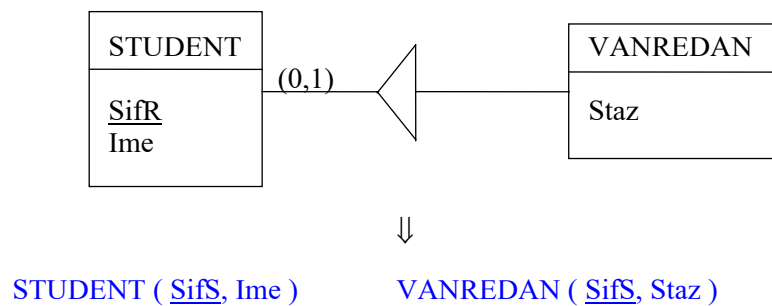
Posmatrajmo klasu vozila sa svojstvima šifre i registarskog broja, koja kao specijalne slučajeve sadrži automobile sa dodatnim svojstvom broja mesta i kamione sa dodatnim svojstvima broja osovina i nosivosti. Za opisanu situaciju imamo sledeće:



Ovo je primer totalne i ekskluzivne specijalizacije, što je i naznačeno kardinalnošću (1,1), iz razumljivih razloga: svako vozilo mora biti ili automobil ili kamion, ali nikako oboje istovremeno. Kardinalnost preslikavanja sa strane podklasa je uvek (1) i podrazumeva se, pa je više nećemo navoditi.

Primer:

Ovaj primer se odnosi na okolnost da su neki od studenata vanredni i da kao zaposleni imaju svojstvo radnog staza, a specijalizacija je parcijalna i vrši se na samo jednu podklasu:



Generalizacija je postupak koji je obrnut u odnosu na specijalizaciju i daje isti krajnji rezultat, odnosno istu nadklasu i podklase.

Definicija:

Za neku klasu objekata (nadklasu) kažemo da predstavlja generalizaciju ako predstavlja opšti slučaj jedne ili više drugih klasa objekata (podklase) po zajedničkim klasifikacionim svojstvima.

Pored toga što predstavlja opšti slučaj nekih podklasa objekata, generalizacija kao pojam ima i značenje postupka pri kome uočavamo da neke klase objekata, podklase, imaju po navedenom kriterijumu nešto zajedničko što se može pridružiti jednoj nadklasi. Pri tome, nadklasa će imati sva klasifikaciona svojstva koja su zajednička za sve podklase, a podklase će imati samo specifična svojstva.

U poređenju sa specijalizacijom, **pravilo prevođenja generalizacije** u odgovarajuće komponente relacionog modela ima na samom početku dva dodatna koraka:

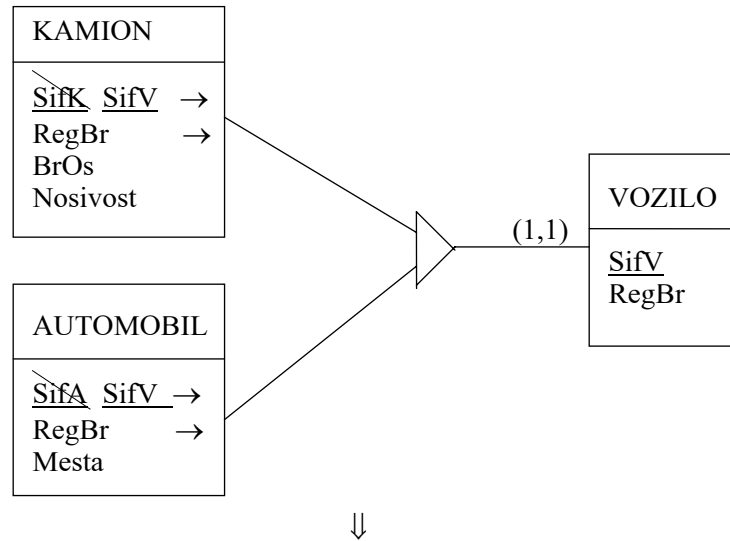
- u svim podklasama vrši se usaglašavanje klasifikacionih svojstava po nazivu, ako su zatečeni nazivi različiti;
- sva klasifikaciona svojstva koja su zajednička za sve podklase postaju zajednička svojstva nadklase, a podklase zadržavaju samo specifična svojstva;
- od svake podklase objekata nastaje šema relacije istog naziva, a attribute te šeme čine klasifikaciona svojstva podklase i primarni ključ šeme koja odgovara nadklasi, koji postaje i primarni ključ šeme podklase;
- od nadklase objekata koja specijalizuje nastaje šema relacija istog naziva i sa atributima koji odgovaraju klasifikacionim svojstvima, pri čemu je atribut koji odgovara identifikatoru primarni ključ.

S obzirom da je krajnji rezultat u modelu nekog sistema isti, sve ostalo što je navedeno za specijalizaciju važi i za generalizaciju.

Kategoriju, najsloženiji koncept modela entiteta i odnosa, nećemo razmatrati.

Primer:

Za generalizaciju će nam poslužiti raniji primer vozila. Neka su u našem modelu entiteta i odnosa prvo nastale klase objekata AUTOMOBIL i KAMION, i neka smo tek tada uočili da za njih postoji generalizacija u vidu nadklase VOZILO:



AUTOMOBIL (SifV, Mesta)

VOZILO (SifV, RegBr)

KAMION (SifV, BrOs, Nosivost)

Redosled nastanka modela je s leva na desno. Precrtavanjem smo označili promenu naziva klasifikacionog svojstva, a strelicom prebacivanje u nadklasau.

8.1.4 Predstavljanje klasa veza

Veze (ili asocijacije) predstavljaju posredan odnos između klasa objekata.

Definicija

Klasa veza u posmatranom sistemu je svaka klasa čija instanca predstavlja odnos konstantnog broja instanci (najmanje dve) iz jedne ili više klasa objekata, pri čemu taj odnos može imati i određena svojstva.

Treba naglasiti da neka instanca u klasi veze nastaje kada se između odgovarajućih instanci objekata uspostavi odnos koji ta veza predstavlja, a nestaje kada se taj odnos raskine.

Bitna razlika ovog odnosa klasa objekata i odnosa zavisnosti i specijalizacije je u tome što taj odnos nije neposredan nego se ostvaruje posredstvom dodatne klase - klase veze.

Navedimo dva primera za prošireni sistem biblioteke:

- klasa veze PRIPADA između klasa objekata NASLOV i OBLAST, koja nema svojstva;
- klasa veze DRZI između klasa objekata CLAN i KNJIGA, koja ima svojstvo Datum (datum od kada član drži knjigu kod sebe).

Klasa veze se u modelu objekata i odnosa predstavlja rombom, unutar koga se velikim slovima upisuje naziv. Od ivica romba se povlače linije do svake klase objekata koji stupaju u vezu. Ako veza ima svojstva, umesto romba se koristi simbol dobijen kombinacijom romba i pravougaonika i u raspoloživi prostor se upisuju nazivi svojstava veze.

Treba naglasiti da veze nisu ograničene samo na nezavisne klase objekata. Moguće su veze između nezavisnih i zavisnih klasa objekata, kao i samo između zavisnih klasa objekata. Uz to, objekti koji su u vezi mogu biti i učesnici u specijalizaciji.

I u slučaju klase veze može se govoriti o preslikavanju, doduše posrednom, između klasa objekata, pa se za svaku klasu objekata koja učestvuje u vezi može definisati odgovarajuća kardinalnost preslikavanja:

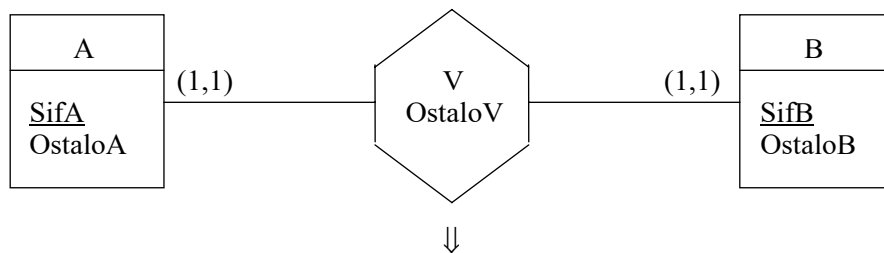
- kardinalnost veze: par brojeva (m,n) koji se zapisuje uz klasu od koja je u vezi i koji ima sledeće značenje: svaka instanca te klase mora da bude učesnik u bar m a može u najviše n instanci veze.

Prema kardinalnosti veze može se sprovesti sledeća podela:

- parcijalna (neobavezna) veza: slučaj kada je $m=0$, odnosno kada u odgovarajućoj klasi objekata mogu da postoje instance koje ne učestvuju u vezi;
- totalna (obavezna) veza: slučaj kada je $m>0$ (a najčešće 1), odnosno kada svaka instanca u odgovarajućoj klasi objekata mora da učestvuje u bar m veza.

Pravilo prevođenja veze u odgovarajući koncept relacionog modela sadrži niz varijanti, zavisno od kardinalnosti veze. Radi jasnoće, ovde je izloženo pravilo za slučaj veze sa dva učesnika, uz ilustraciju svakog slučaja:

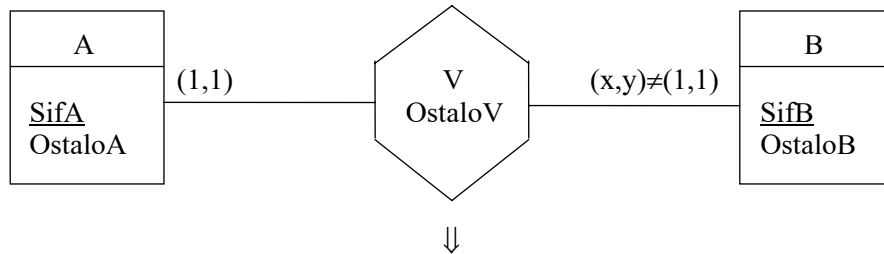
- **pre prevođenja klase veze u odgovarajući koncept relacionog modela, mora biti sprovedeno takvo prevođenje za sve klase objekata koje učestvuju u vezi;**
- **ako je kardinalnost veze za obe klase objekata (1,1), ne nastaje posebna šema relacije veze; u pitanju je greška u projektovanju koja je dovela do razdvajanja jedne klase objekata na dve; u tom slučaju, šema relacije jedne od klasa objekata se ukida, a šema relacije druge klase objekata dopunjuje se atributima ukinute šeme i atributima koji odgovaraju svojstvima klase veze, ako postoje; primarni ključ šeme relacije veze je bilo koji od primarnih ključeva šema objekata:**



A (SifA, OstaloA, SifB, OstaloB, OstaloV)
 ili
 A (SifA, OstaloA, SifB, OstaloB, OstaloV)

/ jedno od SifA ili SifB se
 može izostaviti ako nije
 svojstvo sa značenjem /

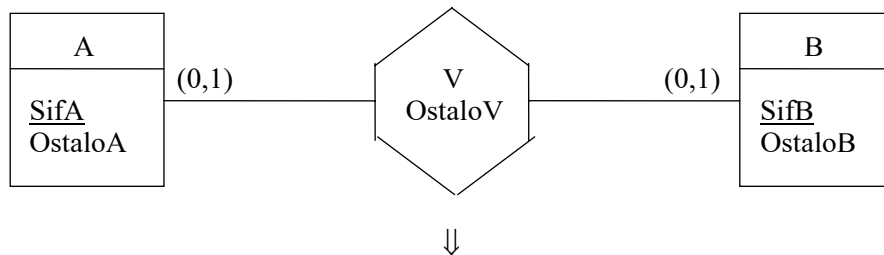
- ako je kardinalnost veze za jednu klasu objekata (1,1) a za drugu bilo šta osim (1,1), ne nastaje posebna šema relacije veze; u tom slučaju, šema relacija koja odgovara klasi objekata sa (1,1) strane dopunjuje se primarnim knjučem šeme relacije druge klase objekata i atributima koji odgovaraju svojstvima klase veze, ako postoje:



$A (\underline{\text{SifA}}, \text{OstaloA}, \text{SifB}, \text{OstaloV})$

$B (\underline{\text{SifB}}, \text{OstaloB})$

- ako je kardinalnost veze za obe klase objekata (0,1), nastaje posebna šema relacije veze; u tom slučaju, šemu relacije veze čine primarni ključevi šema relacija klase objekata i atributi koji odgovaraju svojstvima klase veze, ako postoje; primarni ključ šeme relacije veze je bilo koji od primarnih ključeva šema relacija objekata; izbor je po principu “čvrstine” (“po prirodi stvari” je jače) pa onda dužine:



$A (\underline{\text{SifA}}, \text{OstaloA})$

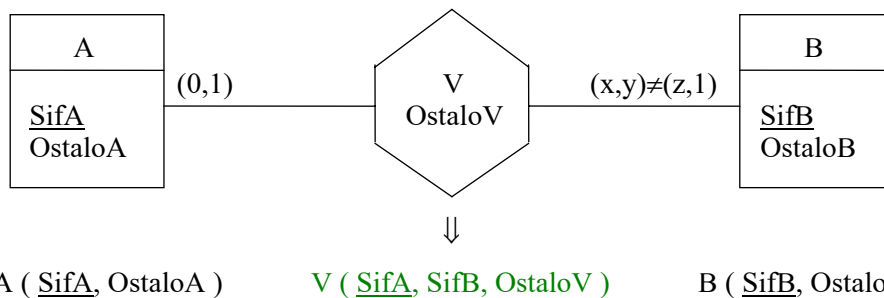
$V (\underline{\text{SifA}}, \underline{\text{SifB}}, \text{OstaloV})$

$B (\underline{\text{SifB}}, \text{OstaloB})$

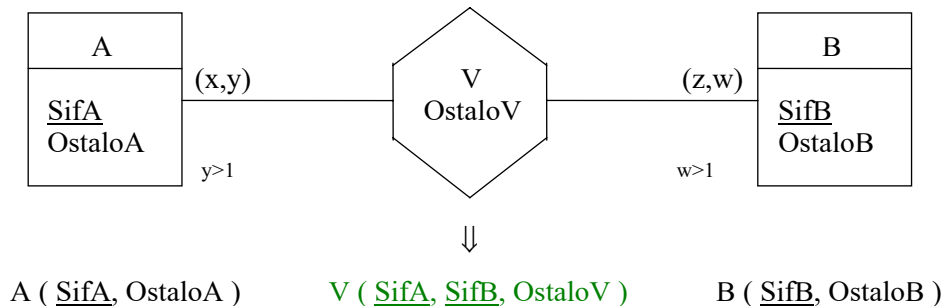
ili

$V (\text{SifA}, \underline{\text{SifB}}, \text{OstaloV})$

- ako je kardinalnost veze za jednu klasu objekata (0,1) a za drugu bilo šta osim (1,1) i (0,1), nastaje posebna šema relacije veze; u tom slučaju, šemu relacije veze čine primarni ključevi šema relacija objekata i atributi koji odgovaraju svojstvima klase veze, ako postoje; primarni ključ šeme relacije veze je primarni ključ šeme relacije objekta sa (0,1) strane:

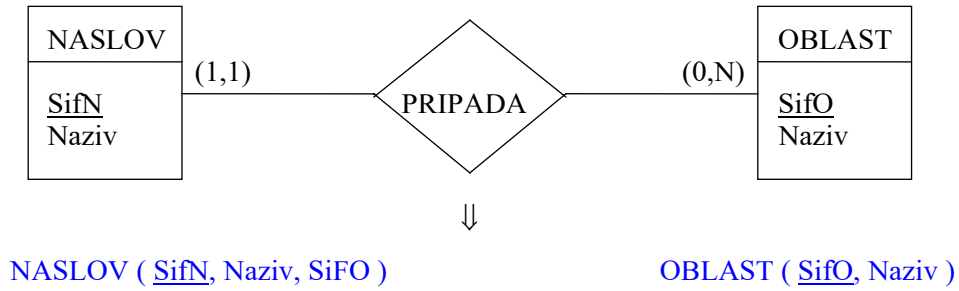


- u svim ostalim slučajevima, nastaje posebna šema relacije veze, koju čine primarni ključevi šema relacija klasa objekata i atributi koji odgovaraju svojstvima klase veze, ako postoje; primarni ključ šeme relacije veze čine dodati primarni ključevi zajedno:



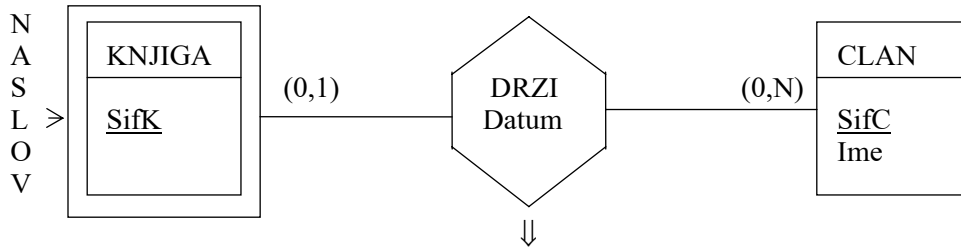
Primer:

U proširenom sistemu biblioteke, na koji se odnosi baza podataka BIBLIOTEKA (prilog A), imamo klasu veze PRIPADA između klasa objekata NASLOV i OBLAST. Veza je bez svojstava, pa je prikazujemo romбом.



Primer:

U istom sistemu imamo između klasa objekata CLAN i KNJIGA vezu drži koja ima svojstvo Datum:

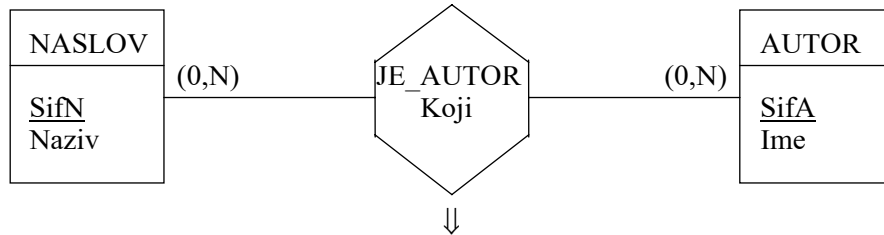


KNJIGA (SifK, SifN) DRZI (SifK, SifC, Datum) CLAN (SifC, Ime)

Šema relacije KNJIGA je proširena atributom na osnovu zavisnosti od klase objekata NASLOV koja je samo naznačena. Kardinalnost (0,1) je jasna sama po sebi: jedna knjiga može biti u nekom trenutku biti izdata samo jednom članu.

Primer:

U sistemu biblioteke klase objekata NASLOV i AUTOR su povezane preko klase veze JE_AUTOR koja ima dodatno svojstvo Koji, pa imamo:



$NASLOV (\underline{SifN}, Naziv) \quad JE_AUTOR (\underline{SifN}, \underline{SifA}, Koji) \quad AUTOR (\underline{SifA}, Ime)$

Kardinalnosti veza (0,N) sa strane obe klase objekata smo odabrali kako bi bili u mogućnosti da evidentiramo naslove i autore a da pri tome odmah ne moramo da evidentiramo i autorstva.

8.1.5 Predstavljanje agregacije

Definicija:

Agregacija je klasa veze koja se ponaša kao klasa objekata na taj način što može da učestvuje u vezama.

U modelu objekata i odnosa agregacija se predstavlja simbolom veze uokvirenim pravougaonikom.

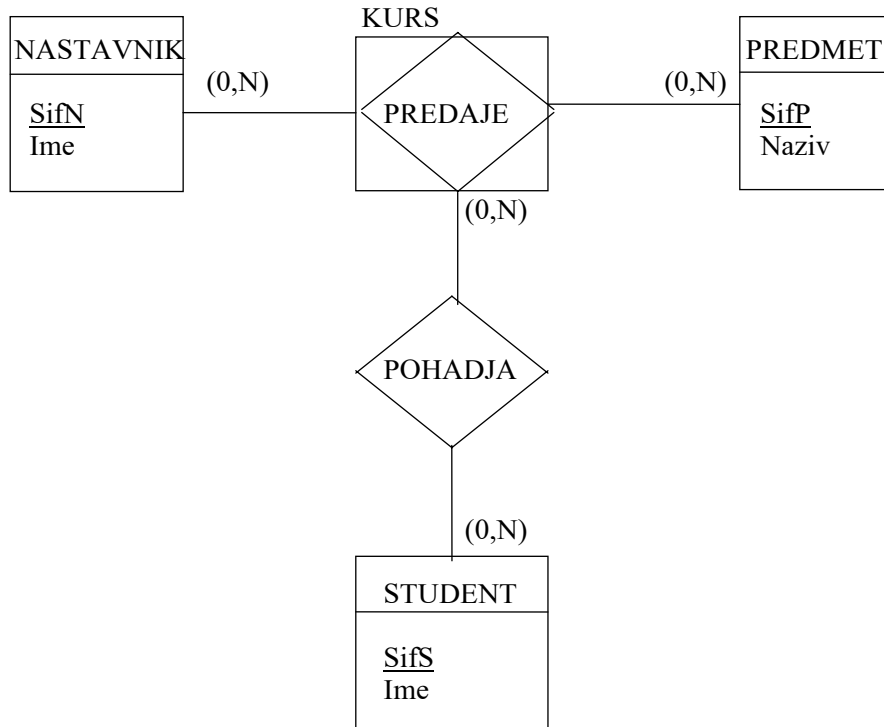
Prevođenje agregacije u odgovarajući koncept relacionog modela je jednostavno:

- agregacija se posmatra kao klasa objekata i na osnovu toga tretira na ranije opisane načine.

Pojasnimo ovaj pojam pogodno odabranim primerom koji se odnosi na deo sistema koga čini neki fakultet.

Primer:

Neka između klasa objekata NASTAVNIK i PREDMET postoji klasa veze PREDAJE. Studenti na fakultetu slušaju predavanja iz određenih predmeta, ali kod određenih nastavnika. To možemo da predstavimo tako što klasu veze PREDAJE tretiramo kao agregaciju koju možemo nazvati KURS, a zatim između te agregacije i klase objekata STUDENT uspostavimo klasu veze POHADJA:



Prvo vršimo prevođenje u relacioni model za klase objekata. Dobijamo:

NASTAVNIK (SifN, Ime) STUDENT (SifS, Ime) PREDMET (SifP, Naziv)

Nakon toga, formiramo šemu relacije za klasu veze PREDAJE:

PREDAJE (SifN, SifP)

Ova šema relacije se u uslovima agregacije ponaša kao šema relacije za klasu objekata koju smo radi jasnoće označili sa KURS, pa nam preostaje da po pravilu za veze formiramo šemu relacije za klasu veza POHADJA:

POHADJA (SifN, SifP, SifS)

Uočimo da se pomoćna oznaka KURS nigde ne pojavljuje u relacionom modelu. Uz to, primetimo da u ovom specifičnom slučaju dobijeni primarni ključ nije minimalan zato što “po prirodi stvari” jedan student samo jednom pohađa jedan predmet, odnosno u POHADJA važi funkcijska zavisnosti SIFS,SIFP→SIFN. Drugim rečima. u ovakvim situacijama uvek treba proveriti minimalnost primernog ključa. U našem konkretnom slučaju dobijamo:

POHADJA (SifN, SifP, SifS)

8.1.6 Prevođenje u relacioni model

Na osnovu svega do sada izloženog možemo formulisati celoviti postupak prevođenja modela objekata i odnosa u relacioni model. Redosled je sledeći:

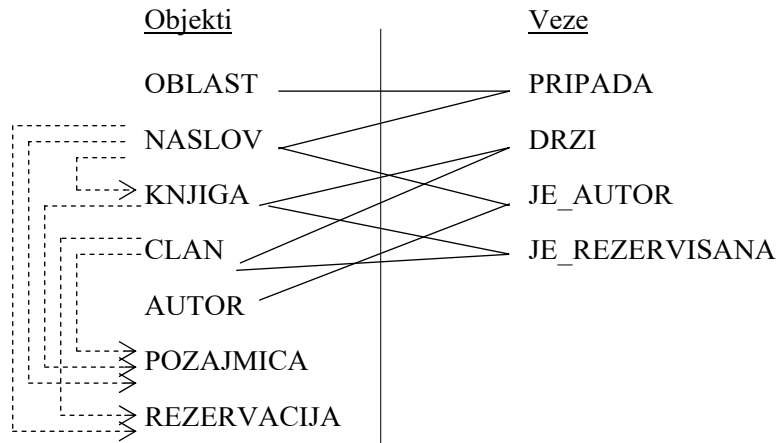
- prvo se na osnovu pravila izloženog u odeljku 8.1.1 vrši prevođenje za sve nezavisne klase objekata, a po pravilu iz odeljka 8.1.3 vrši se prevođenje i za sve njihove specijalizacije;
- posle toga se na osnovu pravila izloženih u odeljku 8.1.2 vrši prevođenje za sve zavisne klase objekata, a po pravilu iz odeljka 8.1.3 i za njihove specijalizacije;
- zatim se prema pravilu navedenom u odeljku 8.1.4 i 8.1.5 vrši prevođenje za sve veze koje su istovremeno i agregacije;
- na kraju, prema pravilima za veze iz odeljka 8.1.4 vrši se prevođenje za sve preostale veze.

Navedeni postupak ćemo ilustrovati sa jednim primerom.

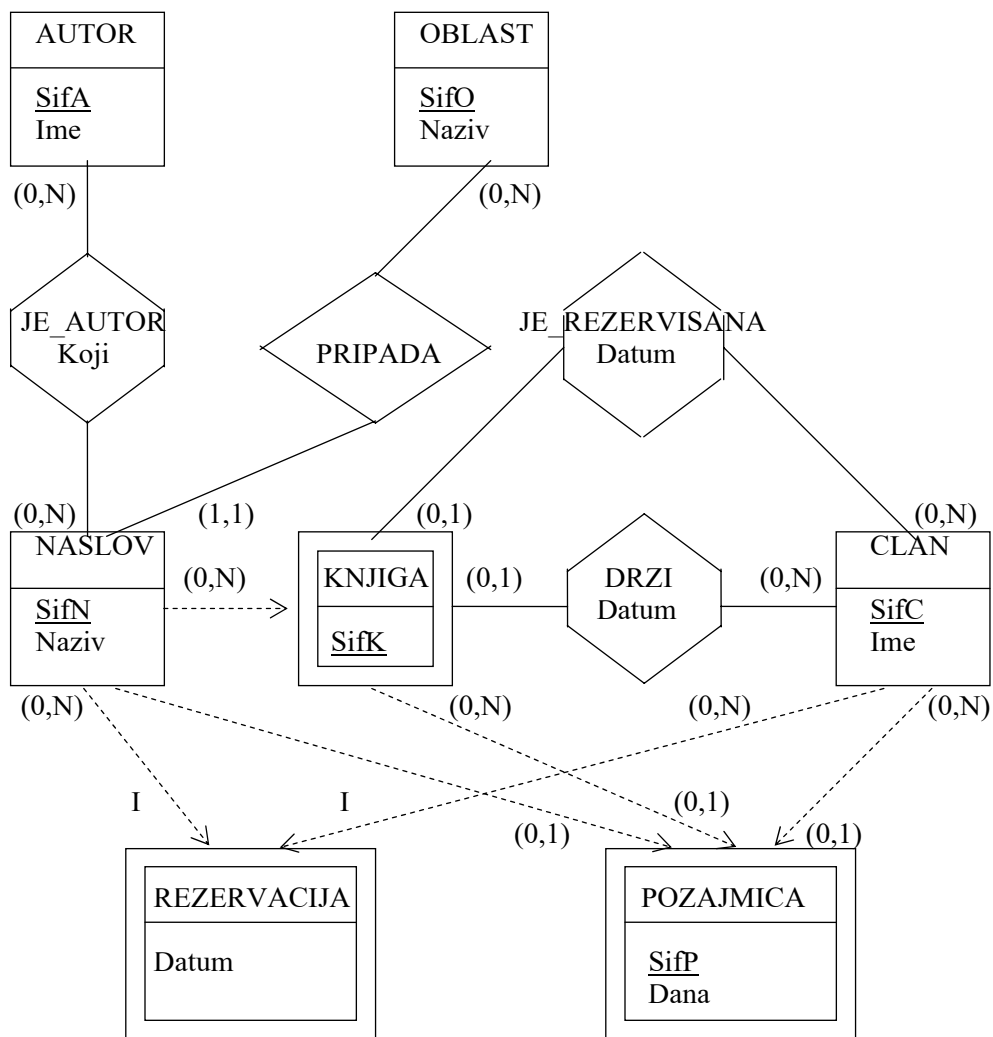
Primer:

Neka je posmatrani sistem naša biblioteka. Pokazaćemo kako je dobijena šema relacione baze podataka BIBLIOTEKA data u prilogu A.

Postupcima abstrakcije opisanim u poglavlju 3 dolazimo do šematskog prikaza sledećih klasa objekata i veza od interesa u posmatranom sistemu:



Ovaj šematski prikaz je koristan i preporučuje se kao prvi korak u razvoju modela podataka, pošto iz njega sagledavamo koje klase su najmanje opterećene u smislu zavisnosti i učešća u vezama. Svakoј takvoj klasi pripada periferno mesto u modelu objekata i odnosa. U našem slučaju to su AUTOR i OBLAST, pa dobijamo model:



Na osnovu ovog modela dobija se relacioni model kao i u prilogu A, na kome je prikazana dopuna šeme objekta na odnosu pravila za (1,1) vezu:

AUTOR (<u>SifA</u> , Ime)	DRZI (<u>SifK</u> , SifC, Datum)
CLAN (<u>SifC</u> , Ime)	JE_AUTOR (<u>SifA</u> , <u>SifN</u> , Koji)
OBLAST (<u>SifO</u> , Naziv)	JE_REZERVISANA (<u>SifK</u> , SifC, Datum)
NASLOV (<u>SifN</u> , Naziv / , SifO)	< dopunjeno zbog veze PRIPADA
KNJIGA (<u>SifK</u> , SifN)	
REZERVACIJA (<u>SifN</u> , <u>SifC</u> , Datum)	
POZAJMICA (<u>SifP</u> , SifC, SifK, SifN, Dana)	

U vezi prethodnog modela i izvedene relacije {eme neophodne su određene napomene o odnosu zavisnosti između klasa objekata. Konkretno, radi se o klasi POZAJMICA gde smo u odnosu na ranija razmatranja uveli i zavisnost od klase NASLOV tako da imamo sledeću situaciju zavisnosti:



Na prvi pogled, zavisnost klase POZAJMICA od klase NASLOV se čini suvišnom pošto se podatak o tome koji naslov je pozajmljen može izvesti posredstvom zavisnosti od klase KNJIGA. Postoje dva razloga protiv takvog zaključka:

- konceptualno posmatrano, član je tražio i pozajmio naslov a sa time obavezno i knjigu kao njegovog fizičkog nosioca;
- ako ne bi evidentirali koji naslov je pozajmljen, u slučaju brisanja knjige iz evidencije (usled gubitka ili oštećenja) izgubili bi podatke o pozajmicama naslova sa te knjige.

U ovom konkretnom slučaju radi se o gubitku podataka usled nestanka posrednika u lancu zavisnosti, ali u posebnim okolnostima posledice mogu biti još gore, što najbolje ilustruje primer video-kluba gde za razliku od biblioteke kaseta kao fizički nosilac filma može presnimavanjem promeniti sadržaj. Bez zavisnosti pozajmice od filma imali bi sledeću situaciju:

FILM (SIFF, NAZIV) KASETA (SIFK) SADRZI (SIFK, SIFF) CLAN (SIFC, IME)
 POZAJMICA (SIFP, SIFC, SIFK, DANA)

U slučaju presnimavanja kasete, odnosno promene atributa SIFF u relaciji SADRZAJ za tu kasetu, kao efekat bi imali ne gubitak podataka nego pogrešne podatke: sve pozajmice filma koji se ranije nalazio na kaseti bile bi pripisane novom filmu.

Zaključak koji sledi iz ovih primera je jasan: pre nego što se neka zavisnost između klasa objekata proglasi za suvišnu i odbaci treba pažljivo analizirati alternativni lanac zavisnosti odnosno posrednike.

U vezi šeme relacije POZAJMICA postoji još jedna okolnost koja je vredna pažnje. U nameri da minimiziramo gubitak podataka u slučaju brisanja iz evidencije člana, knjige ili naslova, za strane ključeve SIFC, SIFK i SIFN usvojili smo za dinamičku specifikaciju referencijalnog integriteta **SET NULL** za operaciju **DELETE** (poglavlje 6, naredbe kreiranja tabela). To nameće izvesno prilagođavanje pojma zavisnosti između klasa objekata:

- u suštini, može se govoriti o dva aspekta odnosa zavisnosti; prvi je zavisnost nastanka koji podrazumeva da instanca slabog objekta ne može da nastane nezavisno od svojih uslovitelja; drugi je zavisnost postojanja koji podrazumeva da instanca slabog objekta ne može da postoji nezavisno od svojih uslovitelja;
- kardinalnost uslovljenosti je u slučaju da je u pitanju samo zavisnost nastanka par brojeva (0,N).

8.1.7 Model objekata i odnosa i zavisnosti između atributa

Osnovna namena modela objekata i odnosa je da posle pažljive analize nekog sistema dodamo do dobro ustrojene šeme relacione baze podataka za taj sistem. Striktno govoreći, ovaj model se sastoji samo iz strukturne komponente, pošto se integritetska komponenta podrazumeva, s obzirom da strani ključevi u šemama relacija nastaju isključivo primenom opisanih pravila prevođenja.

Treba naglasiti da projektovanje relacione baze podataka pomoću modela objekata i odnosa ne isključuje analizu mogućih zavisnosti između atributa i dodatnu transformaciju dobijene šeme relacione baze podataka postupcima normalizacije, o čemu je bilo reći u poglavlju 7. Naime, među odabranim svojstvima klasa mogu postojati neželjene zavisnosti, pa to treba ispitati u završnoj fazi projektovanja relacione baze podataka. Konkretno, radi se o sledećem:

- ako se izbegava identifikaciona zavisnost sve šeme relacije objekata nastale iz modela imaju proste identifikatore pa je time automatski ispoštovana druga normalna forma; takva garancija ne postoji za treću normalnu formu i stoga za svaku šemu relacije objekata treba ispitati da li između neključnih atributa postoje tranzitivne zavisnosti;
- za svaku šemu relacije veze “više prema više” koja ima složeni primarni ključ i neključne attribute treba ispitati da li između neključnih atributa ima tranzitivnih zavisnosti i da li ima neključnih atributa koji zavise od dela primarnog ključa;
- za svaku šemu relacije veze koja je sa jedne strane “jedan prema jedan” ili “jedan prema više” koja ima prost primarni ključ i neključne attribute treba ispitati da li između neključnih atributa ima tranzitivnih zavisnosti.

U slučaju da nastupi bilo koja od navedenih neželjenih situacija sprovodi se postupak normalizacije kako je opisano u poglavlju 7.

8.1.8 Kompromisi u šemi relacione baze podataka

Kompromisi u šemi relacione baze podataka se bez obzira na vrstu sprovode iz jednog jedinog razloga – poboljšanja performansi rada sa bazom podataka. Pri tome je osnovni cilj da se kod upita izbegnu zahtevne operacije spajanja tabela kao i razna obimna svođenja podataka. Postoje dve vrste kompromisne izmene šeme relacione baze podataka:

- redukcija šeme relacione baze podataka;
- redudansa šeme relacione baze podataka.

Kako i sam naziv govori, suština redukcije šeme relacione baze podataka jeste u tome da se smanji broj šema relacija. Postoje tri situacije kada je to moguće:

- tretman veze kardinalnosti (0,1) po pravilu za kardinalnost (1,1): posebna šema relacije veze se ukida a primarni ključ šeme relacije drugog učesnika u vezi i eventualna svojstva veze se dodaju šemi relacije učesnika sa (0,1) strane; u toj relaciji vrednosti dodatih atributa će biti NULL za one instance koje nisu u vezi;
- implozija odnosa specijalizacije/generalizacije: šeme relacija specijalnih objekata se ukidaju a svi atributi iz njih se dodaju šemi relacije generalnog objekta; svaka instanca u toj šemi će imati vrednosti NULL za one attribute koji nisu njeni; radi lakšeg raspoznavanja koja instanca je koji specijalni slučaj, jedinstvenoj šemi relacije se može dodati selektorski atribut u slučaju ekskluzivne specijalizacije, odnosno odgovarajući broj indikatorskih atributa u slučaju inkluzivne specijalizacije;
- implozija odnosa zavisnosti kardinalnosti uslovljavanja (0,1): šema relacije slabog objekta se ukida a svi neključni atributi iz nje se dodaju šemi relacije objekta-uslovitelja; ovo je moguće samo kod slabih objekata koji zavise od jedne klase uslovitelja; svaka instanca objekta-uslovitelja koja nije uslovila instancu slabog objekta imaće vrednost NULL za one attribute koji nisu njegovi.

U našem primeru biblioteke nema mogućnosti primene poslednja dva pravila, ali se zato može primeniti prvo. Time umesto šema relacija

KNJIGA (SIFK, SIFN)

DRZI (SIFK, SIFC, DATUM) JE_REZERVISANA (SIFK, SIFC, DATUM)

ukidanjem šema relacija veza DRZI i JE_REZERVISANA dobijamo

KNJIGA (SIFK, SIFN, SIFCDRZ, DATUMDRZ, SIFCREZ, DATUMREZ)

Ovim se značajno pojednostavljuje postupak obrade zahteva člana za naslovom. Umesto da se sprovodi upit nad relacijama KNJIGA, DRZI, JE_REZERVISANA, sada se sve odvija nad dopunjenom relacijom KNJIGA.

Redudansa šeme relacione baze podataka se može sprovesti u dva vida:

- replikativna redudansa: suština je u tome da se izbegne spajanje tabela; atribut iz jedne šeme relacije se dodaje-replicira u drugoj šemi relacije; pri tome replika mora biti sinhronizovana sa originalom – u slučaju izmene originala treba izmeniti i sve njene replike; na izvestan način, replicirani podatak se ponaša kao strani ključ za koji je za operaciju **UPDATE** specificirano **CASCADE**;
- svodna redudansa: suština je u tome da se izbegne svođenje podataka (brojanje, sabiranje itd.) u tabelama; šemi relacije se dodaje atribut koji predstavlja svodni podatak; pri tome, svaki put kada se ažuriraju tabele iz kojih je izveden svodni podatak mora se ažurirati i svodni podatak.

Kao primer za replikativnu redudansu za naš slučaj biblioteke uzmimo dopunjenu šemu relacije o pozajmicama koja kao takva omogućava da se podaci o ukupnom broju i trajanju pozajmica po oblastima dobiju bez spajanja sa tabelom NASLOV:

POZAJMICA (SIFP, SIFC, SIFK, SIFN, DANA, SIFO)

Svodnu redudansu možemo ilustrovati primerom dopunjene šeme relacije NASLOV koja obezbeđuje brzo dobijanje podatka o ukupnom broju i trajanju pozajmica naslova:

NASLOV (SIFN, NAZIV, SIFO, **BROJPOZ**, **DANAPOZ**)

Cena koju smo platili za tu brzinu jeste složeniji postupak održavanja podataka: svaki put kada se vraća pozajmljena knjiga treba dodatno ažurirati podatke BROJPOZ i DANAPOZ u relaciji NASLOV. Savremeni SQL dozvoljava mogućnost da se nad tabelom definiše tzv. “okidač” (TRIGGER) koji reaguje na izmenu u tabeli tako što izvršava specificirani niz naredbi. To u mnogome olakšava održavanje svodnih redundantnih podataka,

Poslednji primer koji ilustruje gotovo spektakularne efekte svodne redudanse odnosi se na jednu banku koja posluje sa štedišama tako što im omigučava da imaju otvorene račune određenih vrsta (tekući, žiro, oročeni itd.) za koje mogu vršiti uplate i isplate. To se bez redudanse može predstaviti sledećim skupom šema relacija:

VRSTA_RACUNA (SIFV, NAZIV) RACUN (SIFR, STEDISA, SIFV)

PROMET (SIFR, REDBR, DATUM, UPLATA, ISPLATA)

Pošto se evidentiraju sve uplate i isplate, iz ovih šema se spajanjem i sumiranjem za svaki račun svakog štediša mogu dobiti ukupna uplata, ukupna isplata i stanje (saldo). Isto to se može dobiti i za svaku vrstu računa, a kada se sumira za sve vrste računa i za celu banku. Problem je samo u tome što tipična banka ima desetak vrsta računa i u okviru njih stotinak hiljada računa štediša od kojih svaki mesečno ima na desetine stavki prometa. Drugim rečima, da bi se došlo do podataka koliko banka ima ukupno na svim vrstama računa treba prethodno sumirati iznose svih stavki prometa čiji se broj kreće na milione. To u uslovima brzog donošenja odluka u savremenom poslovanju nije prihvatljivo. Rešenje navedenog problema je u uvođenju redudantnog skupa šema relacija:

VRSTA_RACUNA (SIFV, NAZIV, UPLATA, ISPLATA)

RACUN (SIFR, STEDISA, SIFV, UPLATA, ISPLATA)

PROMET (SIFR, REDBR, DATUM, UPLATA, ISPLATA)

Ovim smo obezbedili najbrže moguće odgovore na upite koliko banka ima sredstava ukupno ili po vrstama računa, ali smo zato opteretili obradu svake uplate i isplate ažuriranjem svodnih podataka za račune i vrste računa što se u radu praktično ne oseća s obzirom da je unos podataka o uplatama i isplatama manuelan.

8.2 Projektovanje postupaka

Kao što je već naglašeno, projektovanje postupaka podrazumeva dva koraka. Prvo što se sprovodi jeste funkcionalna dekompozicija posmatranog sistema. Kada se taj postupak obavi do određenog nivoa detalja stižu se uslovi za sprovođenje narednog koraka, a to je algoritamska specifikacija.

8.2.1 Funkcionalna dekompozicija

Sušтина postupka funkcionalne dekompozicije jeste da se funkcionalnost posmatranog sistema dekomponuje na sastavne podfunkcije i da se to sukcesivno ponavlja na dobijene podfunkcije sve do dostizanja željenog nivoa detalja kada se može pristupiti algoritamskoj specifikaciji. U savremenoj informatici postoje četiri oblika funkcionalne dekompozicije od kojih će ovde biti navedena samo prva dva:

- funkcionalna dekompozicija prve vrste: jedino što je vidljivo jeste hijerarhijski odnos funkcionalnosti, odnosno to koje funkcije su sastavljene iz kog skupa podfunkcija; ne postoji nikakva naznaka podataka preko kojih funkcije komuniciraju, redosleda izvršavanja funkcija, uslovljavanja njihovog izvršavanja ili ponavljanja;
- funkcionalna dekompozicija druge vrste: pored hijerarhijskog odnosa funkcionalnosti, postoji i naznaka podataka preko kojih funkcije komuniciraju, redosleda izvršavanja funkcija, uslovljavanja njihovog izvršavanja i ponavljanja izvršavanja.

Sprovedimo prvo funkcionalnu dekompoziciju prve vrste za sistem BIBLIOTEKA kako je opisan u odeljku A3 priloga A. Pri tome će naglasak biti na održavanju podataka, s obzirom da se korišćenje podataka svodi na pogodno formulisane upite. Nizom transformacija dobijamo redom:

BIBLIOTEKA

Održavanje maticnih podataka
Obrada prometa

BIBLIOTEKA

Održavanje maticnih podataka
Održavanje podataka o oblastima
Održavanje podataka o naslovima
Održavanje podataka o autorima
Održavanje podataka o knjigama
Održavanje podataka o članovima
Obrada prometa
Obrada traženja naslova
Obrada vraćanja knjige
Obrada gubitka knjige

BIBLIOTEKA

- Odrzavanje maticnih podataka
 - Odrzavanje podataka o oblastima
 - Dodavanje nove oblasti
 - Izmena postojece oblasti
 - Brisanje postojece oblasti
 - Odrzavanje podataka o naslovima
 - Dodavanje novog naslova
 - Izmena postojeceg naslova
 - Brisanje postojeceg naslova
 - Odrzavanje podataka o autorima
 - Dodavanje novog autora
 - Izmena postojeceg autora
 - Brisanje postojeceg autora
 - Odrzavanje podataka o knjigama
 - Dodavanje nove knjige
 - Izmena postojece knjige
 - Brisanje postojece knjige
 - Odrzavanje podataka o clanovima
 - Dodavanje novog clana
 - Izmena postojeceg clana
 - Brisanje postojeceg clana
- Obrada prometa
 - Obrada trazanja naslova
 - Ocitavanje neophodnih podataka
 - Provera da li clan ima kod sebe trazeni naslov
 - Provera da li ima slobodne knjige
 - Evidentiranje izdavanja knjige
 - Evidentiranje rezervacije naslova
 - Obrada vracanja knjige
 - Ocitavanje neophodnih podataka
 - Evidentiranje vracanja
 - Evidentiranje pozajmice
 - Provera ima li rezervacije
 - Opsluzivanje rezervacije
 - Obrada gubitka knjige
 - Ocitavanje neophodnih podataka
 - Evidentiranje gubitka

Ovde treba naglasiti da se situacije izmene matičnih podataka odnose na unos naknadnih izmena podataka ili ispravku pogrešno unetih podataka.

Funkcionalnu dekompoziciju druge vrste navešćemo samo za poslednju od prethodnih dekompozicija. Pri tome je naglasak stavljen na obradu prometa:

BIBLIOTEKA

Odrzavanje maticnih podataka

Odrzavanje podataka o oblastima

Dodavanje nove oblasti (I:OBLAST)
 Izmena postojece oblasti (U:OBLAST)
 Brisanje postojece oblasti (D:OBLAST)

Odrzavanje podataka o naslovima

Dodavanje novog naslova (I:NASLOV)
 Izmena postojeceg naslova (U:NASLOV)
 Brisanje postojeceg naslova (D:NASLOV)

Odrzavanje podataka o autorima

Dodavanje novog autora (I:AUTOR)
 Izmena postojeceg autora (U:AUTOR)
 Brisanje postojeceg autora (D:AUTOR)

Odrzavanje podataka o knjigama

Dodavanje nove knjige (I:KNJIGA)
 Izmena postojece knjige (U:KNJIGA)
 Brisanje postojece knjige (D:KNJIGA)

Odrzavanje podataka o clanovima

Dodavanje novog clana (I:CLAN)
 Izmena postojeceg clana (U:CLAN)
 Brisanje postojeceg clana (D:CLAN)

Obrada prometa

Obrada trazanja naslova

1 Ocitanje neophodnih podataka
 (< SifC, < SifN)
 2 Provera da li clan ima kod sebe trazeni naslov
 (> SifC, > SifN, < Ishod, S:DRZI,KNJIGA)
 ?3 Provera da li ima slobodne knjige / Ishod
 (> SifN, < Ishod, < SifK, S:KNJIGA,DRZI,JE_REZERVISANA)
 ?4e Evidentiranje izdavanja knjige / Ishod
 (> SifK, > SifC, I:DRZI)
 ?4e Evidentiranje rezervacije naslova / Ishod
 (> SifC, > SifN, I:REZERVACIJA)

Obrada vraćanja knjige

1 Ocitanje neophodnih podataka
 (< SifK, < SifN, < SifC, < Datum, S:DRZI,KNJIGA)
 2 Evidentiranje vraćanja
 (> SifK, D:DRZI)
 2 Evidentiranje pozajmice
 (> SifC, > SifK, > SifN, > Datum, I:POZAJMICA)
 3 Provera ima li rezervacije
 (> SifN, < Ishod, < SifC, S:REZERVACIJA)
 ?4 Opsluživanje rezervacije / Ishod
 (> SifK, > SifC, I:JE_REZERVISANA)

Obrada gubitka knjige

Ocitanje neophodnih podataka
 (< SifK)
 Evidentiranje gubitka
 (> SifK, D:DRZI,KNJIGA)

Ovde je u pogledu korišćene notacije neophodan niz napomena:

- uslovljenost neke funkcije se označava znakom `?` i navođenjem uslovljavajućeg podatka iza naziva funkcije i odvojeno znakom `/` (na primer, to je funkcija `Opsluzivanje rezervacije`);
- redosled izvršavanja funkcija se označava rednim brojevima; za funkcije bez redosleda ili sa istim redosledom sekvenca njihovog izvršavanja je nebitna (to važi za funkcije `Evidentiranje vratanja` i `Evidentiranje pozajmice`);
- uslovljene funkcije koje se uzajamno isključuju označavaju se istim brojem redosleda i sa `e` iza toga (to je slučaj kod funkcija `Evidentiranje izdavanja knjige` i `Evidentiranje rezervacije naslova`);
- u zagradama iza ili ispod naziva funkcije navode se kvalifikatorima `>`, `<` ili `<>` ulazni, izlazni i ulazno-izlazni podaci;
- unutar istih zagrada navode se operacije funkcije nad entitetima modela podataka odnosno relacijama u bazi podataka, i to: `S`-očitavanje, `I`-ubacivanje, `U`-izmena i `D`-brisanje.

8.2.2 Algoritamska specifikacija

Sušтина algoritamske specifikacije je u tome da se funkcionalna dekompozicija druge vrste za posmatrani sistem dopuni dodatnim detaljima do te mere da je njena implementacija svedena na jednoznačni rutinski postupak.

Kao primer algoritamske specifikacije navešćemo onu za funkciju **Obradatrazenjanaslova**. U vezi takvog izbora treba naglasiti da algoritamska specifikacija nije vezana isključivo za najniže sastavne funkcije nego da se može primeniti i na funkcije višeg nivoa.

Kao osnova za sastavljanje navedene algoritamske specifikacije poslužiće nam opis rada biblioteke koji je izložen u odeljku A.3 priloga A. Prema onome što je tamo navedeno, kada član traži neki naslov prvo se proverava da li član već ima kod sebe knjigu sa traženim naslovom. Ako ima, ne izdaje mu se druga knjiga, a ako nema proverava se da li je za njega već odvojena kao rezervisana knjiga sa tim naslovom. Ako jeste, ta knjiga mu se izdaje, a samo ako nije traži se da li ima slobodne knjige sa tim naslovom. Ako takva knjiga postoji ona se izdaje članu, a ako ne postoji član se izjašnjava da li želi da rezerviše traženi naslov. Ako želi, to se i evidentira.

Na osnovu svega prethodno navedenog za funkciju **Obradatrazenjanaslova** sledi i formalna algoritamska specifikacija. Radi preglednosti, kao komentari su navedene sve podfunkcije. Prvo je izložena specifikacija po notaciji koja je nešto izmenjena u odnosu na onu izloženu u prilogu B i poglavlju 4, a zatim sledi i još detaljnija algoritamska specifikacija koja koristi i elemente programskog SQL jezika izložene u Prilogu C.

```

ObradaTrazenjaNaslova
: Ocitanje neophodnih podataka i inicijalizacija
: : OUTPUT ( "Unesite sifre clana i naslova" )
: : INPUT ( vSifC, vSifN )
: : vIshod = 'NEMA_KNJIGE'
: Provera da li vec ima naslov kod sebe
: : DO FOR EACH ( vDrzi IN DRZI )
: : | WHERE ( vDrzi.SIFC == vSifC )
: : | WHILE ( vIshod == 'NEMA_KNJIGE' )
: : | DO FOR FIRST ( vKnjiga IN KNJIGA )
: : | | WHERE ( vKnjiga.SIFK == vDrzi.SIFK AND
: : | | vKnjiga.SIFN == vSifN )
: : | | vIshod = 'VEC_IMA'
: : | | OUTPUT ( "Clan ima knjigu sa trazanim naslovom" )
: Provera ima li slobodne knjige
: : IF ( vIshod == 'NEMA_KNJIGE' )
: : | Provera da li je za clana rezervisana knjiga
: : | : DO FOR EACH ( vJeRezerv IN JE_REZERVISANA )
: : | : | WHERE ( vJeRezerv.SIFC == vSifC )
: : | : | WHILE ( vIshod == 'NEMA_KNJIGE' )
: : | : | DO FOR FIRST ( vKnjiga IN KNJIGA )
: : | : | | WHERE ( vKnjiga.SIFK == vJeRezerv.SIFK AND
: : | : | | vKnjiga.SIFN == vSifN )
: : | : | | vIshod = 'IMA_KNJIGE'
: : | : | | vSifK = vKnjiga.SIFK
: : | : Provera da li ima knjige koja nije kod clana
: : | : IF ( vIshod == 'NEMA_KNJIGE' )
: : | : | DO FOR EACH ( vKnjiga IN KNJIGA )
: : | : | | WHERE ( vKnjiga.SIFN == vSifN )
: : | : | | WHILE ( vIshod == 'NEMA_KNJIGE' )
: : | : | | vSlobodna = 'DA'
: : | : | | DO FOR FIRST ( vDrzi IN DRZI )
: : | : | | | WHERE ( vDrzi.SIFK == vKnjiga.SIFK )
: : | : | | | vSlobodna = 'NE'
: : | : | | DO FOR FIRST ( vJeRezerv IN JE_REZERVISANA )
: : | : | | | WHERE ( vJeRezerv.SIFK == vKnjiga.SIFK )
: : | : | | | WHILE ( vSlobodna = 'DA' )
: : | : | | | vSlobodna = 'NE'
: : | : | | IF ( vSlobodna == 'DA' )
: : | : | | | vIshod = 'IMA_KNJIGE'
: : | : | | | vSifK = vKnjiga.SIFK
: : IF ( vIshod == 'IMA_KNJIGE' )
: : | Evidentiranje izdavanja knjige
: : | : OUTPUT ( 'Izdati clanu knjigu', vSifK )
: : | : INSERT DRZI ( vSifK, vSifC, Datum() )
: +- ( vIshod == 'NEMA_KNJIGE' )
: | Evidentiranje rezervacije naslova
: | : OUTPUT ( "Unesite da li clan zeli rezervaciju" )
: | : INPUT ( vZeliRezervaciju )
: | : IF ( vZeliRezervaciju == 'DA' )
: | : | INSERT REZERVACIJA ( vSifN, vSifC, DatumVreme() )

```

U vezi korišćene notacije može se kao prvo konstatovati da je u pitanju specifična forma relacionog računa n-torki, s obzirom da u petljama oblika

```
DO FOR EACH ( Varijabla IN Relacija )
```

promenljiva **Varijabla** varira nad n-torkama relacije **Relacija**. Što se tiče izmena u odnosu na notaciju iz priloga B i poglavlja 4, one su bazirane na osnovu saznanja da se često javljaju algoritamske konstrukcije tipa

```
DO FOR EACH ( Var IN SkuP )      DO FOR EACH ( Var IN Skup )
| IF ( Uslov )                    | IF ( Uslov )
| | Postupak                      | | Postupak
                                <|--EXIT
```

odnosno petlja nad elementima skupa koja izvršava **Postupak** za sve elemente koji zadovoljavaju **Uslov** i petlja nad elementima skupa koja izvršava **Postupak** samo za prvi element koji zadovolji **Uslov** a zatim prekida dalji rad. Za navedene dve situacije usvojene su respektivno algoritamske konstrukcije

```
DO FOR EACH ( Var IN SkuP )      DO FOR FIRST ( Var IN Skup )
| WHERE ( Uslov )                 | WHERE ( Uslov )
| Postupak                       | Postupak
```

čime je povećana preglednost specifikacije i smanjena dubina ugnežđjavanja. Ovde uslov u **WHERE** klauzuli predstavlja uslov izbora elementa. Ako se želi zadavanje uslova “dokle god” kojim se iteriranje prekida kada taj uslov postane **FALSE**, to se postiže dodatnom klauzulom **WHILE**. Tako se dobijaju konstrukcije

```
DO FOR EACH ( Var IN SkuP )      DO FOR FIRST ( Var IN Skup )
| WHERE ( UslovIzbora )           | WHERE ( UslovIzbora )
| WHILE ( UslovRada )             | WHILE ( UslovRada )
| Postupak                       | Postupak
```

Klauzule **WHERE** i **WHILE** su opcione. Inače, za sve varijante iteriranja nad relacijama sa i bez ugnježdjavanja mogu se formulisati pravila za prevođenje u odgovarajuće SQL konstrukcije (upite i kursore) što obezbeđuje rutinski i jednoznačni postupak implementacije.

Algoritamska specifikacija zasnovana na elementima programskog SQL jezika je još detaljnija od prethodne i jednostavnija je za implementaciju:

ObradaTrazenjaNaslava

```

:  Ocitanje neophodnih podataka i inicijalizacija
:  :  OUTPUT ( "Unesite sifre clana i naslova" )
:  :  INPUT  ( vSifC, vSifN )
:  :  vIshod = 'NEMA_KNJIGE'
:  Provera da li vec ima naslov kod sebe
:  :  SELECT SIFK
:  :  INTO :vSifK
:  :  FROM DRZI D
:  :  WHERE SIFC = :vSifC
:  :  AND EXISTS ( SELECT SIFK
:  :  :  FROM KNJIGA
:  :  :  WHERE SIFK = D.SIFK
:  :  :  AND SIFN = :vSifN ) ;
:  :  IF ( SQLCODE != NOT_FOUND )
:  :  |  vIshod = 'VEC_IMA'
:  :  |  OUTPUT ( "Clan ima knjigu sa traženim naslovom" )
:  Provera ima li slobodne knjige
:  :  Provera da li je za clana rezervisana knjiga
:  :  :  IF ( vIshod == 'NEMA_KNJIGE' )
:  :  :  |  SELECT SIFK
:  :  :  :  INTO :vSifK
:  :  :  :  FROM JE_REZERVISANA JR
:  :  :  :  WHERE SIFC = :vSifC
:  :  :  :  AND EXISTS ( SELECT SIFK
:  :  :  :  :  FROM KNJIGA
:  :  :  :  :  WHERE SIFK = JR.SIFK
:  :  :  :  :  AND SIFN = :vSifN ) ;
:  :  :  |  IF ( SQLCODE != NOT_FOUND )
:  :  :  |  |  vIshod = 'IMA_KNJIGE'
:  Provera da li ima knjige koja nije kod clana
:  :  IF ( vIshod == 'NEMA_KNJIGE' )
:  :  |  SELECT SIFK
:  :  :  INTO :vSifK
:  :  :  FROM KNJIGA
:  :  :  WHERE SIFN = :vSifN
:  :  :  AND SIFK NOT IN ( SELECT SIFK
:  :  :  :  FROM DRZI
:  :  :  :  UNION
:  :  :  :  SELECT SIFK
:  :  :  :  FROM JE_REZERVISANA ) ;
:  :  |  IF ( SQLCODE != NOT_FOUND )
:  :  |  |  vIshod = 'IMA_KNJIGE'

```

nastavlja se

nastavak

```

: IF ( vIshod == 'IMA_KNJIGE' )
: | Evidentiranje izdavanja knjige
: | : OUTPUT ( 'Izdati clanu knjigu", vSifK )
: | : INSERT INTO DRZI
: | : VALUES ( :vSifK, :vSifC, Datum() ) ;
: +- ( vIshod == 'NEMA_KNJIGE' )
: | Evidentiranje rezervacije naslova
: | : OUTPUT ( "Unesite da li clan zeli rezervaciju" )
: | : INPUT ( vZeliRezervaciju )
: | : IF ( vZeliRezervaciju == 'DA' )
: | : | INSERT INTO REZERVACIJA
: | : | VALUES ( :vSifN, :vSifC, DatumVreme() )

```

Ovakva algoritamska specifikacija je korak do implementacije: ako se formalizuje, moguće ju je automatski prevesti na izabrani programski jezik uz minimalne dodatne intervencije programera.

8.3 Zaključne napomene

U prethodnim odeljcima razmotrena su oba vida projektovanja relacione baze podataka: projektovanje podataka i projektovanje postupaka. Ostalo je otvoreno pitanje njihovog redosleda, odnosno da li se prvo u celini obavlja projektovanje podataka pa u celini projektovanje postupaka, ili je obrnuto.

Odgovor glasi: ni jedno ni drugo. U pitanju je objedinjeni postupak projektovanja postupaka i podataka kod koga projektovanje postupaka počinje prvo i uvek u izvesnoj meri prednjači u odnosu na projektovanje podataka, pri čemu se vrši postepena razrada “korak po korak”. To je razumljivo samo po sebi: da bi projektant mogao da sagleda koji su podaci potrebni u bazi podataka neophodno je da ima bar neku predstavu o tome kako budući sistem treba da funkcioniše. Sa svakim korakom razrade pojaviće se nove funkcije nižeg reda koje će zahtevati uvođenje novih podataka. Pri tome, pogodno sredstvo za objedinjavanje projektovanja postupaka i podataka predstavlja matrica “postupci – operacije” koju smo u pojednostavljenoj formi tabele već izložili u poglavlju 4.

Uz prethodno, treba naglasiti i to da je vrlo važno da se na samom početku projektovanja sagleda šta budući korisnici očekuju od baze podataka. To se sve može formulisati u vidu odgovarajućih upita iz kojih proizilazi i potreba za odgovarajućim podacima u bazi podataka.

Takođe, sastavni deo projektovanja relacione baze podataka jeste formulacija dinamičkih uslova referencijalnog integriteta. To je za primer BIBLIOTEKA već urađeno u poglavljima 4 i 6 i stoga neće biti ponavljano ovde.