Едиција "ТЕХНИЧКЕ НАУКЕ"

Оснивачи и издавачи Едиције : Факултет техничких наука у Новом Саду
МП "STYLOS", Нови Сад

Година оснивања : 1996.

Главни и одговорни уредник Едиције : Проф. др Душан Петровачки, декан

Дизајн : Јосип Згомба

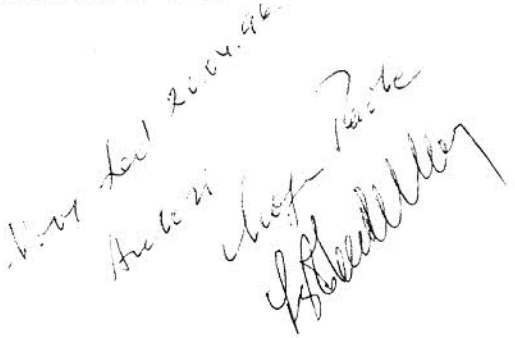
Информације о појединачним издањима :

1	Невенка Адић, Илија Ковачевић, Војислав Марић, Вера Унгар	"Математичка анализа II"
2	Данило Обрадовић	"Основи рачунарства"
3	Лепосава Шиђанин	"Машински материјали II"
4	Владимир Милошевић, Владо Делић	"Дигиталне телекомуникације - збирка задатака"
5	Ирена Чомић, Вера Унгар	" Теорија редова"
6	Павле Могин, Иван Луковић	"Принципи база података"
7	Раде Дорословачки	"Збирка решених испитних задатака из алгебре"
8	Бранко Перишић, Драган Иветић	"Основи рачунарства -методичка збирка задатака II (програмабилни аутомати)"
9	Илија Ковачевић, Небојша Ралевић	"Математичка анализа I - гранични процеси"
10	Илија Ковачевић, Небојша Ралевић, Лија Ваш	"Увод у математичку анализу"

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

Pavle Mogin Ivan Luković

PRINCIPI BAZA PODATAKA



1996

Edicija "TEHNIČKE NAUKE"

Naziv udžbenika : "Principi baza podataka"

Autori: Dr Pavle Mogin redovni profesor na FTN-u u Novom Sadu
Dr Ivan Luković, asistent na FTN-u u Novom Sadu

Recenzenti: Dr Branislav Lazarević, redovni profesor FON-a u Beogradu
Dr Dušan Malbaški, vanredni profesor na FTN-u u Novom Sadu
Dr Vojislav Mišić, docent na ETF-u u Beogradu

Osnivači i izdavači: Fakultet tehničkih nauka u Novom Sadu
MP "STYLOS", Novi Sad

Glavni i odgovorni urednik izdanja : Prof. dr Dušan Petrovački, dekan

Za MP "STYLOS" : Veselin Stefanović , urednik za izdavaštvo

Štampa : "S Print", Novi Sad, Dr Ilije Đuričića 3, Tel.: 021/58-366

Štampano u 500 primeraka

CIP-Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

681.3.06(075.8)

МОГИН, Павле
Principi baza podataka/Pavle Mogin, Ivan Luković.-
Novi Sad: Stylos: Fakultet tehničkih nauka, 1996
(Novi Sad: S Print).- 342 str.: graf. prikazi; 22 cm.

Tiraž 500. Библиографија: стр. 317-321.-
Регистар.

1.Луковић, Иван
а) Информациони системи - Базе података
0

Predgovor

U oblasti informacionih tehnologija, početak devedesetih godina je obeležen i pojavom takvih sistema za upravljanje bazama podataka, koji su, u velikoj meri, ispunili dugogodišnja očekivanja projektanata i korisnika. Ta očekivanja, definisana još tokom sedamdesetih godina, odnosila su se, pre svega, na:

- jednostavnost upotrebe,
- pouzdanost u radu,
- centralnu kontrolu ispravnosti podataka,
- lokacijsku distribuciju i
- prihvatljive performanse obrade baze podataka.

Pri tome, sve to je postalo dostupno na opremi relativno niske cene.

Isti vremenski period je kod nas obeležen:

- intenzivnom izgradnjom informacionih sistema, zasnovanih na bazama podataka i
- uvođenjem predmeta, posvećenih bazama podataka, na praktično svim visokoškolskim ustanovama, koje obrazuju informatičare.

Međutim, dok su, u svetu, ovi tokovi propraćeni obiljem stručne i naučne literature, kod nas se oseća nedostatak knjiga, koje bi pokrile kako potrebe dodiplomske i poslediplomske nastave, tako i prakse projektovanja i izgradnje baza podataka. Knjiga Principi baza podataka i njen prirodni nastavak Principi projektovanja baza podataka imaju ambiciju i cilj da, barem, ublaže nedostatak odgovarajuće literature na našem govornom području.

Materija, izložena u ovim knjigama, rezultat je višegodišnjeg teoretskog i praktičnog rada autora u oblasti baza podataka. Nastala je sređivanjem beležaka za predavanja, koja je bar jedan od autora držao, u okviru dodiplomske i poslediplomske nastave, na: Fakultetu tehničkih nauka, Prirodno - matematičkom fakultetu i Ekonomskom fakultetu Univerziteta u Novom Sadu i na Elektrotehničkom fakultetu Univerziteta u Beogradu. Takođe, u knjige su ugrađena iskustva i saznanja, stečena u praksi projektovanja i izgradnje baza podataka automatizovanih informacionih sistema.

Teorija i praksa sistema baza podataka zasnovana je na pojmu modela podataka. Knjiga Principi baza podataka ima za osnovni cilj da, relativno sveobuhvatno, prikaže teoretske postavke i praktične aspekte tri, danas najbitnija, modela podataka. To su: model

entiteta i poveznika, relacioni model i objektno - orijentisani model. Značaj modela entiteta i poveznika leži u činjenici da se on, najčešće, koristi u takozvanom konceptualnom projektovanju baza podataka. Na relacionom modelu su zasnovani, praktično svi, savremeni sistemi za upravljanje bazama podataka. Od objektno - orijentisanog modela se očekuje da dovede do širenja primene baza podataka na oblasti, kao što su: inženjersko projektovanje uz primenu računara i multimedija.

Pravo intelektualne svojine na delove knjige Principi baza podataka, raspoređeno je na sledeći način:

- Miro Govedarica je autor tačke 7.6,
- Ivan Luković je autor glave 6, priloga 1 i priloga 2,
- Pavle Mogin polaže autorsko pravo na sve ostale delove ove knjige.

Autori žele i ovim putem da izraze zahvalnost recenzentima: profesoru doktoru Branislavu Lažareviću, vanrednom profesoru doktoru Dušanu Malbaškom i docentu doktoru Vojislavu Mišiću na zapažanjima, primedbama i korisnim sugestijama, koje su svakako doprinele kvalitetu izložene materije.

Takođe, autori se zahvaljuju: Branimiru Jefiću, diplomiranom informatičaru, Nenadu Vujasinoviću, diplomiranom inženjeru, Marijani Lomić, diplomiranom informatičaru, Sič Petru, diplomiranom inženjeru i studentu Danijelu Balošu na pomoći u redigovanju i tehničkom uređivanju knjige.

Na kraju, autori mole čitaoce da obrate pažnju na poslednje strane knjige. Tamo su predstavljeni sponzori, koji su pomogli štampanje ove knjige.

Novi Sad, januar, 1996. godine

Autori

Sadržaj

7. Glava 217

Objektno - orijentisani model podataka

1. Glava

Modeli podataka

Model podataka je matematička apstrakcija, koja se koristi za projektovanje modela realnog sistema. Taj model treba da omogući izgradnju odgovarajućeg informacionog sistema i njegove baze podataka. U tom cilju, model realnog sistema treba da obezbedi informacije o:

- statičkim osobinama,
- ugrađenim ograničenjima i
- dinamičkim osobinama

realnog sistema. Statičke osobine su relativno nezavisne od vremena. Retko se menjaju. Nose informaciju o strukturi buduće baze podataka. Ograničenja govore o pravilima poslovanja i ponašanja u realnom sistemu. Takođe, ograničenja govore o dozvoljenim i nedozvoljenim vrednostima podataka i dozvoljenim i nedozvoljenim odnosima između podataka o komponentama stanja realnog sistema. Ta ograničenja treba da se preslikaju u uslove ***integriteta*** buduće baze podataka. Uslovi integriteta su mehanizmi, koji obezbeđuju da podaci o komponentama stanja realnog sistema, u bazi podataka, budu usaglašeni sa ograničenjima realnog sistema. Dinamičke osobine ukazuju na evolutivnu prirodu realnog sistema. Odslikavaju promene u realnom sistemu. Te promene se reflektuju putem izmene podataka o komponentama stanja realnog sistema. Da bi baza podataka predstavljala što verniju sliku realnog sistema, moraju se definisati operacije, koje će dovoditi sadržaj baze podataka u saglasnost sa aktuelnim podacima o komponentama stanja realnog sistema. Strukture nad skupom tih operacija predstavljaju programe. Ti programi menjaju bazu podataka saglasno promenama u realnom sistemu.

Primer 1.1. Podatke o komponentama stanja fakulteta, kao realnog sistema, predstavljaju:

- broj upisanih studenata,

- podaci o svakom studentu ponaosob,
- podaci o nastavnicima,
- podaci o predmetima,
- podaci o poveravanju izvođenja nastave iz pojedinih predmeta,
- ocene studenata iz predmeta i slično. □

Definicija 1.1. Model podataka M je matematička apstrakcija, izražena trojkom (S, I, O), gde je S strukturalna, I integritetna, a O operacijska komponenta modela. □

Ovo poglavlje ima dva cilja. Prvi je da ukaže na bitne opšte karakteristike svake od komponenata modela podataka i time obezbedi preduslove za praćenje poglavlja, posvećenih opisima konkretnih modela podataka. Drugi je da, putem kratkog pregleda razvoja, nabroji najznačajnije modele podataka.

1.1. *Strukturalna komponenta modela podatka*

Strukturalna komponenta modela podataka sadrži skup primitivnih koncepata i skup pravila za izgradnju složenijih koncepata. Složeniji koncepti se grade od primitivnih. *Koncept* je apstraktna predstava jedne klase delova realnog sveta. Taj deo realnog sveta može biti:

- skup sličnih subjekata ili neki konkretni subjekat,
- skup sličnih objekata ili neki konkretan objekat,
- skup sličnih događaja ili neki konkretan događaj,
- zajednička osobina svih elemenata jednog skupa ili konkretna vrednost osobine i
- veza između dva skupa ili dva konkretna elementa nekih skupova.

U daljem tekstu će se realni subjekti, objekti i događaji nazivati ***činiocima*** ili ***entitetima***.

Primitivni koncept je takav koncept koji se ne može dalje dekomponovati na koncepte.

Primer 1.2. U građevinarstvu primitivne koncepte predstavljaju: pesak, voda, cement, kreč. Od njih se grade složeniji koncepti, kao što su: betonske grede, temelji, zidovi, kuće, gradovi i slično.

U modelima podataka, primitivne koncepte mogu predstavljati:

- opis osobine skupa činilaca i
- skup konkretnih vrednosti te osobine.

Od njih se grade složeniji koncepti, kao što su: opis skupa činilaca, opisi njihovih veza i opis baze podataka. □

Koncepti i pravila za njihovu izgradnju se koriste za izradu dve vrste modela. Jedno su ***modeli skupova činilaca***. Izgradnja modela skupova činilaca se, najčešće, vrši navođenjem naziva skupa i bitnih zajedničkih osobina elemenata skupa. Drugo su ***modeli konkretnih činilaca***. Njihovi modeli se grade navođenjem konkretnih vrednosti zajedničkih osobina, upotrebljenih za izgradnju modela odgovarajućeg skupa činilaca, jer svaki činilac pripada nekom skupu činilaca.

Primer 1.3. Na fakultetu, kao realnom sistemu, činioce predstavljaju: skup studenata, skup predmeta, skup nastavnika, skup učionica, skup ispita i drugi. Jedan apstraktni model skupa studenata bi mogao biti

$$Student(BRI, IME, PRZ, BPI),$$

gde je *Student* naziv skupa, a *BRI* (broj indeksa), *IME* (ime studenta), *PRZ* (prezime studenta), *BPI* (broj položenih ispita) predstavljaju opise zajedničkih osobina svih studenata.

Model jednog studenta, kao konkretnog činioca fakulteta, bi mogao biti

$$(159, Aca, Car, 13)$$

Apstraktni model skupa predmeta bi mogao biti

$$Predmet(OZP, NAP),$$

gde *OZP* (oznaka predmeta) i *NAP* (naziv predmeta) predstavljaju opise zajedničkih osobina svih predmeta. □

Činioci su u realnom sistemu međusobno povezani na razne načine. Da bi se u model statičkih osobina realnog sistema ugradila informacija o tim ***vezama***, uvode se relacije u skup modela skupova činilaca, ili se relacije uvode između modela samih činilaca. I za opisivanje tih relacija se koristi notacija, slična notaciji za predstavljanje modela skupova činilaca i modela konkretnih činilaca. Tako se dolazi do dve vrste struktura. Jedno su strukture nad skupom modela skupova činilaca, a drugo su strukture nad skupom modela samih činilaca realnog sistema. Te strukture predstavljaju dva modela statičkih osobina realnog sistema definisana na dva nivoa apstrakcije. Struktura nad skupom modela skupova činilaca je na višem nivou apstrakcije od strukture nad skupom modela samih činilaca.

Primer 1.4. Kao model relacije između skupa studenata i skupa predmeta, može poslužiti sledeća imenovana dvojka

$$Sluša(Student, Predmet).$$

To je koncept na nivou apstrakcije modela skupova činilaca. Model veze između konkretnog studenta i predmeta bi bio

$$((159, Aca, Car, 13), (p_1, Baze\ podataka)).$$ □

U vezi sa strukturalnom komponentom modela podataka, često se koriste dva pojma. To su: ***intenzija*** i ***ekstenzija***. Pojam intenzije se koristi za definicioni opis nekog skupa. Intenzija definiše skup navođenjem uslova koje njegovi elementi treba da zadovolje. Pojam ekstenzije se odnosi na prikaz jedne od mogućnih pojava skupa nabrajanjem elemenata. Intenzija je generalizacija skupa ekstenzija. Definiše sve zajedničke osobine svojih ekstenzija.

Predstavljanje ekstenzije je samo detaljnija ilustracija modela statičke strukture realnog sistema. Ona dopunjava, putem primera ograničenog obima, rešenje definisano putem intenzionalnog opisa statičke strukture.

Primer 1.5. Model skupa činilaca i jedan skup modela konkretnih činilaca predstavljaju, redom, intenziju i ekstenziju. Model skupa činilaca opisuje osobine svakog skupa modela svojih elemenata (individualnih činilaca).

Konkretno, model skupa činilaca *Student*, kao intenzija, opisuje osobine ekstenzionalnog modela (*159, Aca, Car, 13*), koji reprezentuje realnog studenta Acu Cara. Baza podataka sadrži skupove ovakvih modela realnih činilaca, a kardinalni broj svakog skupa modela odgovara kardinalnom broju odgovarajućeg skupa realnih činilaca. Drugim rečima, baza podataka sadarži model svakog realnog činioca. □

1.2. Integritetna komponenta modela podataka

Svaki realni sistem poseduje skup pisanih ili običajnih *pravila ponašanja* ili *pravila poslovanja*. Ta pravila se izražavaju putem:

- ograničenja mogućnih vrednosti odredenih zajedničkih osobina nekog skupa činilaca,
- ograničenja veza izmedu dva ili više konkretnih činilaca i
- ograničenja odnosa izmedu realnog činioca i njemu pridružene vrednosti zajedničke osobine.

Primer 1.6. U nekom trgovinskom preduzeću, moguća ograničenja su:

- podatak o zalihi robe ne može imati negativnu vrednost (ograničenje vrednosti osobine),
- popust ne može biti manji od *0* i veći od *10* posto (ograničenje vrednosti osobine),
- otpremnica mora imati bar jednu stavku sa robom (ograničenje odnosa izmedu dva činioca),
- sva roba na jednoj otpremnici mora biti iz istog skladišta (ograničenje veze izmedu dva činioca, otpremnice i skladišta),
- svaka roba ima jedinstvenu identifikacionu oznaku, a jedna identifikaciona oznaka se pridružuje najviše jednoj robi (ograničenje odnosa izmedu realnog činioca i vrednosti zajedničke osobine).

Na fakultetu, pravila ponašanja mogu diktirati sledeća ograničenja:

- ocena je ceo broj ne manji od *5* i ne veći od *10* (ograničenje vrednosti osobine),
- ocena, manja od *6* se ne evidentira (ograničenje vrednosti osobine),
- jedan student može imati najviše jednu ocenu iz jednog predmeta (ograničenje veze izmedu tri činioca),
- jedan nastavnik može predavati više predmeta, a jedan predmet može izvoditi više nastavnika (ograničenje veze izmedu dva činioca),
- svaki student ima jedan broj indeksa, a jedan broj indeksa se dodeljuje najviše jednom studentu (ograničenje odnosa izmedu realnog činioca i vrednosti zajedničke osobine).

Fraza "jedan nastavnik može predavati više predmeta, a jedan predmet može izvoditi više nastavnika" predstavlja pravilo ponašanja u realnom sistemu, pa prema tome i ograničenje, ali ne tako strogo kakvo bi bilo ograničenje "jedan nastavnik može predavati najviše jedan predmet". □

Baza podataka informacionog sistema treba da bude usaglašena sa pravilima ponašanja u realnom sistemu. Zato se ograničenja, u postupku projektovanja baze podataka, izražavaju putem pogodne notacije, a zatim ugraduju u bazu podataka, prilikom njene realizacije. Ograničenja, definisana u postupku projektovanja baze podataka, nazivaju se i ***uslovima integriteta*** baze podataka. Uslovi integriteta ograničavaju broj dozvoljenih sadržaja baze podataka.

Primer 1.7. Jedna od mogućnih notacija za ograničenje vrednosti neke osobine A na interval vrednosti $[a_1, a_2]$ je $a_1 \leq A \leq a_2$.

Notacija $(X(0, 1) : Y(1, N))$ se može interpretirati na sledeći način:

- jedan realni činilac iz skupa X mora biti povezan sa najmanje jednim činiocem iz skupa Y i
- jedan realni činilac iz skupa Y može biti povezan sa najviše jednim činiocem iz skupa X.

Ova interpretacija relativno precizno opisuje uslove povezivanja činilaca iz dva skupa.

Ako su S, P i O osobine, tada se notacija $SP \rightarrow O$ može upotrebiti u cilju iskazivanja pravila da svakoj kombinaciji SP vrednosti odgovara najviše jedna O vrednost. □

Definicija 1.2. Za bazu podataka, čiji sadržaj je u saglasnosti sa svim definisanim uslovima inegriteta, kaže se da je ***konzistentna*** (neprotivrečna). □

Primer 1.8. Baza podataka, u koju su ugradena ograničenja iz primera 1.7, je konzistentna, ako:

- ne sadrži $a_0 < a_1$ ili $a_3 > a_2$ vrednost za osobinu A,
- ne sadrži podatke o činiocu x iz X, koji nije povezan sa podacima o bar jednom činiocu y iz Y,
- ne sadrži podatke o vezi činioca y iz Y sa više činilaca iz X i
- ne sadrži takvu kombinaciju SP vrednosti, koja je povezana sa više od jedne O vrednosti. □

Skup koncepata i skup pravila strukturalne komponente i skup uslova integriteta se koriste za definisanje modela statičke strukture realnog sistema. Na osnovu modela, definisanog putem opisa skupova činilaca realnog sistema, gradi se baza podataka informacionog sistema. Baza podataka se nalazi na medijumima eksternih memorijskih uredaja računarskog sistema. Sadrži dozvoljene podatke o stanju realnog sistema i podatke o dozvoljenim vezama izmedu činilaca realnog sistema. Nedozvoljene vrednosti osobina i nedozvoljene veze, isključuju se iz baze podataka putem ograničenja. Svakom stanju realnog sistema odgovara jedna ***pojava*** baze podataka. Pojava baze podataka je odredena trenutnim vrednostima podataka i veza, izmedu tih podataka.

1.3. *Operacijska komponenta modela podataka*

Dinamičke osobine realnog sistema opisuju se putem skupa operacija O. Operacija $o \in O$ izvršava se na pojavi baze podataka. Skup operacija O odgovara skupu naredbi

takozvanog jezika za manipulisanje podacima. Svaka operacija $o \in O$ predstavlja funkciju u skupu stanja $D = \{d_i | i = 1,..., n\}$ baze podataka, tj. o: $D \rightarrow D$. Pri tome, funkcija o ne mora biti definisana na celom skupu D, jer rezultat njene primene može dovesti do kolizije sa definisanim skupom ograničenja. Tada se operacija o ne izvršava, proglašava se nedefinisanom.

Ne menja svaka operacija iz O pojavu baze podataka, ali menja njeno stanje. Naime, pored podataka, baza podataka sadrži i niz kontrolnih mehanizama ili indikatora. Vrednosti ovih indikatora i pojava baze podataka čine ***stanje baze podataka***. Najkarakterističniji kontrolni mehanizam predstavlja takozvani ***indikator tekućeg sloga*** (dalje: ***indikator aktuelnosti***). Vrednost indikatora aktuelnosti često predstavlja adresa lokacije sloga, kojem se poslednje pristupilo. Taj slog se naziva tekućim slogom.

Primer 1.9. Posmatra se indeks-sekvencijalna datoteka i naredba tipa *ČITAJ NAREDNI SLOG*. Trenutno stanje datoteke pre izvršenja naredbe, određeno je podacima u datoteci (pojava) i sadržajem indikatora aktuelnosti. Indikator aktuelnosti sadrži adresu sloga kojem se poslednje pristupilo. Izvršenje operacije ne menja pojavu datoteke, ali menja sadržaj indikatora aktuelnosti, a time i stanje datoteke, jer sada adresa narednog sloga predstavlja sadržaj indikatora aktuelnosti. □

Kada se operacije izvršavaju na bazi podataka, po pravilu se odnose na njen relativno mali deo. Izbor relativno malog dela baze podataka naziva se ***selekcijom***. Na današnjem stupnju razvoja računarske tehnike, pojam selekcije podrazumeva prenos tog dela baze podataka sa eksterne u operativnu memoriju.

Operacija se najčešće sastoji od dva dela. U prvom se definiše aktivnost a u drugom selekcija. Selekcija bira deo baze podataka na kojem treba da se izvrši aktivnost. U osnovi postoji pet ***aktivnosti***. To su:

- definisanje vrednosti indikatora aktuelnosti, koje se naziva i definisanjem logičkog mesta u strukturi podataka,
- čitanje podataka,
- upis novih podataka,
- brisanje postojećih podataka i
- modifikacija postojećih podataka.

Selekcija dela baze podataka se može vršiti uz pomoć:

- logičkog mesta u strukturi podataka (vrednosti indikatora aktuelnosti),
- odnosa između podataka i
- vrednosti osobine.

Selekcija podataka putem vrednosti indikatora aktuelnosti zahteva postojanje naredbi u okviru jezika za manipulisanje podacima, koje će eksplicitno, ili implicitno uticati na vrednost tog indikatora.

Primer 1.10. Naredba tipa *POSTAVI INDIKATOR AKTUELNOSTI NA NAREDNI SLOG*, eksplicitno postavlja adresu lokacije narednog sloga u indikator aktuelnosti. Naredba tipa *ČITAJ NAREDNI SLOG*, prvo implicitno postavlja adresu lokacije narednog sloga u indikator aktuelnosti, a zatim prenosi sam slog u radnu zonu korisnika. □

Selekcija podataka se može vršiti i na osnovu njihove povezanosti sa drugim podacima. Karakterističan primer predstavlja slučaj kada su povezani podaci o dva činioca realnog sistema.

Primer 1.11. Neka su u bazi podataka podaci o svakom radniku povezani sa podacima o njegovom radnom mestu putem podataka o radnikovom rasporedivanju na to radno mesto. *Radnik*, *Radno_Mesto* i *Je_Raspoređen* predstavljaju modele odgovarajućih skupova činilaca. Naredba tipa *ČITAJ Radno_Mesto ZA Radnik PUTEM Je-Raspoređen* dovela bi do prenošenja podataka o radnom mestu onog radnika, čiji slog je tekući u skupu slogova o radnicima, u radnu zonu korisnika. Prethodno je, odgovarajućom naredbom, morala biti definisana potrebna vrednost indikatora aktuelnosti za skup slogova o radnicima.

Ako bi se želeli dobiti podaci o radnim mestima za sve radnike sa nekom karakterističnom osobinom, na primer, za sve čije zanimanje je "inženjer", u slučaju selekcije podataka putem njihovog odnosa u strukturi, bilo bi potrebno napisati čitav program. Pojednostavljeno govoreći, taj program treba iterativno da:

- pristupi slogu svakog radnika,
- da proveri da li zadovoljava traženu osobinu i,
- u slučaju potvrdnog odgovora, da, putem podataka o njegovom rasporedivanju, pronađe podatke o njegovom radnom mestu. □

Selekcija podataka na osnovu zadate vrednosti osobine, naziva se i asocijativnim adresiranjem. Naredbe jezika, koji koriste ovakav postupak za izbor podataka imaju deo koji se naziva kvalifikacionim izrazom, sa sledećom osnovnom strukturom: (*osobina*, *uslov*, *vrednost*). Vrednost (osobine) služi kao kriterijum za selekciju (izbor), a uslov predstavlja jedan od znakova $<$, $\leq$, $>$, $\geq$, $=$, $\neq$. Kvalifikacioni izrazi se mogu povezivati logičkim operacijama.

Primer 1.12. Neka je *Radnik*(*MBR*, *IME*, *PRZ*, *GRD*) model skupa radnika. Složeni kvalifikacioni izraz "(*IME* = *Ivo*) $\wedge$ (*GRD* = *1940*)" doveo bi do prenošenja podataka o svim radnicima koji se zovu *Ivo* i rođeni su *1940* godine u radnu zonu korisnika. □

Jezici za manipulisanje podacima se razlikuju s obzirom na činjenicu da li jedna selekcija vrši izbor podataka o najviše jednom činiocu ili vrši izbor podataka skupa činilaca sa nekom zajedničkom osobinom. Ako selekcija vrši izbor podataka o najviše jednom činiocu praćenjem nekog logičkog puta u strukturi podataka, jezici se nazivaju ***navigacionim***. Za selekciju u navigacionim jezicima je karakteristično korišćenje indikatora aktuelnosti i veza podataka u strukturi. Ovi jezici su proceduralni, u smislu da se putem njih mora saopštiti sistemu za upravljanje bazom podataka ne samo šta se želi dobiti, već i kako to treba postići. Proceduralni su, jer se za selekciju podataka mora napisati program, koji, u opštem slučaju, sadrži sve tri osnovne programske strukture: ***sekvencu***, ***selekciju*** i ***iteraciju***, kako je to opisano u poslednjem delu primera 1.11. Pri tome, treba naglasiti da selekcija podataka iz baze podataka i selekcija, kao osnovna programska struktura, predstavljaju različite pojmove sa istim nazivom.

Ako se putem jedne operacije vrši, u opštem slučaju, izbor podataka o skupu činilaca sa istom osobinom, jezik se naziva ***specifikacionim*** ili ***deklarativnim***. Deklarativni jezici koriste vrednosti osobina za selekciju. Nazivaju se i neproceduralnim, jer se

putem naredbi tog jezika izražava samo šta (koji podaci) se žele dobiti, ali ne i kako to treba postići. Kod deklarativnih jezika, selekcija se specificira: navođenjem naziva modela skupova činilaca i kvalifikacionog izraza (kao u primeru 1.12).

1.4. Kratak pregled razvoja modela podataka

Tokom poslednjih tridesetak godina, razvijen je veći broj različitih modela podataka. Sudbina tih modela podataka je bila različita. Neki od njih su predstavljali samo interesantan pokušaj ili usputnu stanicu u razvoju drugih modela, a drugi su ostavili trajan trag kako u teoriji tako i u praksi baza podataka. U modele podataka, koji spadaju u ovu drugu kategoriju, spadaju:

- mrežni model podataka,
- hijerarhijski model podataka,
- relacioni model podataka,
- model entiteta i poveznika,
- funkcionalni model podataka,
- model semantičkih hijerarhija,
- semantički model podataka,
- objektno - orijentisani model podataka i
- logički model podataka.

Mrežni i hijerarhijski model podataka su se pojavili u drugoj polovini šezdesetih godina. Već početkom sedamdesetih ušli su u dnevnu upotrebu ***sistemi za upravljanje bazama podataka*** (SUBP), zasnovani na ovim modelima podataka. Međutim, ti ***sistemi baza*** podataka (SUBP plus sama baza podataka) predstavljali su određeno razočaranje za korisnike. Nisu doveli do očekivanog porasta produktivnosti ni programera ni krajnjih korisnika. Razlozi za to razočarenje ležali su u:

- nedovoljnom razdvajanju logičkih od fizičkih aspekata baze podataka,
- kompleksnosti struktura podataka i
- korišćenju proceduralnog i navigacionog jezika.

Osnovni cilj definisanja i razvoja relacionog modela podataka, bio je eliminisanje baš tih nedostataka. Od njegovog predstavljanja 1970. godine [C1], za relacioni model je stalno rastao interes prvo naučnika, a kasnije i korisnika. Taj interes je bio posledica niza poželjnih osobina, u koje, između ostalih, spadaju: formalno - matematička osnova, homogenost, dobro definisani kriterijumi za ocenu kvaliteta projekta baze podataka i, verovatno iznad svega, deklarativni jezik za korišćenje baze podataka. Taj jezik je zasnovan na matematičkoj logici, ali veoma dobro prilagođen potrebama komforne i lake upotrebe. Međutim, relacioni, kao i svi drugi matematički modeli podataka, ne poklanja dovoljno pažnje semantici (smislu, značenju) podataka. Strukture podataka relacionog modela su semantički nedovoljno bogate, te je krajnjem korisniku teško da ih poveže sa konkretnim realnim sistemom.

Nedovoljna snaga relacionog modela za izražavanje semantike, dovela je krajem sedamdesetih godina do povećanog interesa za semantičke modele podataka. Taj interes je

porastao pojavom modela entiteta i poveznika [Che], modela semantičkih hijerarhija [SS] i semantičkog modela podataka [HM]. I sam rodonačelnik relacionog modela podataka, Codd, je u svom radu [C2] predložio uvođenje novih koncepata za povećanje semantičke izražajnosti relacionog modela. Interes za semantičke modela podataka se nastavio i u osamdesetim godinama. Širenje horizonata korišćenja baza podataka vodi ka sve intenzivnijoj primeni semantički bogatijih modela podataka, naročito u oblastima inženjerskog projektovanja, multimedija i baza znanja.

Osamdesete godine su bile obeležene intenzivnim širenjem SUBP zasnovanih na relacionom modelu i istraživanjima u oblasti objektno - orijentisanih i logičkih modela podataka. Bez pretenzija na strogost definicije, objektno - orijentisani modeli se razvijaju na osnovama mrežnog modela, semantičkih modela i objektno - orijentisanih programskih jezika. Logički modeli se mogu posmatrati kao dalja nadgradnja relacionog modela podataka konceptima programskog jezika Prolog. Ta nadgradnja znači, pre svega, uvođenje dedukcije u baze podataka.

Prva polovina devedesetih godina je obeležena: dominacijom relacionih SUBP, pojavom SUBP, zasnovanih na objektno - orijentisanom modelu podataka, ali i uključivanjem pojedinih objektno - orijentisanih koncepata u relacione SUBP, ili čak kombinovanjem objektno - orijentisanih programskih jezika sa relacionim SUBP. Realno je pretpostaviti da će i do kraja ovog veka, relacione tehnologije i objektno - orijentisani pristup igrati, ako ne dominantnu, onda svakako važnu ulogu u izgradnji sistema baza podataka.

2. Glava

Model entiteta i poveznika

Postoji više verzija modela podataka zasnovanih na entitetima i poveznicima. Najpoznatiju verziju predstavlja Chenov model entiteta i poveznika [Che]. Kako sam Chen kaže, model je nastao kao sinteza dobrih strana tri druga modela: mrežnog, relacionog i modela skupova entiteta [SAAF]. Za geometrijsku reprezentaciju koncepata modela koriste se grafovi i tabele. Intenzionalni deo modela se reprezentuje grafovima, a ekstenzionalni deo modela se reprezentuje tabelama. Mada je Chen u svom osnovnom radu opisao sve tri komponente modela entiteta i poveznika, sam model je prevashodno namenjen za projektovanje statičkog modela realnog sistema, zbog širokih mogućnosti za izgradnju semantički bogatog modela realnog sistema i zbog pogodne dijagramske tehnike za geometrijsko reprezentovanje tog modela. Za model podataka se kaže da je semantički (smisaono) bogat, ako se putem njegovih koncepata može izgraditi veran model praktično svakog realnog sistema. Model entiteta i poveznika je poslužio i kao osnova za definisanje semantičkog modela podataka, koji se intenzivno koristi u veštačkoj inteligenciji i za definisanje strukturalne komponente objektno - orijentisanog modela podataka.

Model entiteta i poveznika se, često, naziva i ***ER modelom podataka***, a sreću se i drugi nazivi. Jedan od njih je model podataka ***objekti - veze***.

Cilj ovog poglavlja je ovladavanje metodama, tehnikama i konvencijama za transformisanje slike realnog sistema, kakva egzistira u projektantovom intelektu, u intenzionalni i ekstenzionalni opis statičke strukture tog realnog sistema.

2.1. Strukturalna komponenta ER modela podataka

Osnovna ideja, na kojoj je zasnovan model entiteta i poveznika, je da se realni svet (ili njegov deo, realni sistem) može opisati pomoću dva osnovna koncepta. To su entitet i poveznik. ***Entitet***[*)] je "nešto" što se može jednoznačno identifikovati. Entitet se takođe opisuje kao "jedinica posmatranja". Predstavlja termin, koji se može odnositi na svaki realni subjekat, objekat, događaj, pojavu ili neki apstraktni pojam. *Poveznik* predstavlja vezu između dva ili više entiteta. Poveznik konstituišu povezani entiteti i opis njihove veze.

Primer 2.1. Student *Milan Brkić* sa brojem indeksa *89034* je jedan entitet. Predmet *Baze podataka* je drugi entitet. *Polaganje ispita* iz predmeta Baze podataka na dan *09.10. 1994.* godine je takođe jedan entitet. Međutim, jedan mrav se može proglasiti entitetom, samo pod uslovom da smo u stanju da ga jednoznačno identifikujemo među drugim mravima, što je malo verovatno. Entitet bi mogla biti neka kolonija mrava.

Jedan poveznik je "*student Milan Brkić sluša predmet Baze podataka*", drugi je "*student Milan Brkić je položio predmet Baze podataka*". To su različiti poveznici između dva ista entiteta. U oba slučaja, entiteti su Milan Brkić i Baze podataka. Opisi njihove veze su: "sluša" i "položio". Jedan poveznik između tri entiteta je "*student Milan Brkić je položio predmet Baze podataka na ispitu od 09.10.1994. godine*" (treći entitet je ispit). □

2.1.1. Entitet i skup (klasa) entiteta

Entiteti, koji egzistiraju kao predstave realnih subjekata, objekata, događaja u ljudskom intelektu mogu se klasifikovati u skupove sličnih entiteta. Formalno, neka je e enititet, tada je skup entiteta $E = \{e|\mathcal{P}(e)\}$, gde je $\mathcal{P}(e)$ predikat čija istinitosna vrednost ukazuje da li e pripada skupu E. Ako e poseduje osobinu $\mathcal{P}(e)$, tada e pripada E. Isti entitet e može pripadati različitim skupovima entiteta.

Primer 2.2. Ako je $\mathcal{P}(e)$ = "e je student", onda skupu E pripadaju samo studenti, a ne i ostali ljudi. Međutim, ako je $E' = \{e|e$ je ljudsko biće$\}$, tom skupu pripadaju svi ljudi, ali ne i ostali sisari. Pri tome, studenti pripadaju i skupu E i skupu E'. □

2.1.2. Intenzija ER modela podataka

Na nivou intenzije, koncepte ER modela podataka čine: tip entiteta i tip poveznika. Obeležje predstavlja osnovni gradivni elemenat za konstituisanje ovih, složenijih koncepata.

---

*) Latinski, ens, entis znači biće, bitnost.

2.1.2.1. Obeležje

Skupovi sličnih entiteta se nazivaju i ***klasama*** entiteta. Svi entiteti jedne klase poseduju bar jednu zajedničku osobinu, na osnovu koje su i svrstani u istu klasu. U opštem slučaju, broj zajedničkih osobina entiteta jedne klase je veći od jedan. Ove osobine nazivaju se ***obeležjima*** (atributima). Obeležja se označavaju velikim slovima latinske azbuke, skraćenim nazivom (mnemonikom) ili punim nazivom. Velika slova latinske azbuke se koriste kada semantika obeležja nije važna.

Primer 2.3. Ako je *MATIČNI_BROJ_RADNIKA* pun naziv obeležja, odgovarajući mnemonik bi mogao biti *MBR*. □

Obeležje, koje se dalje ne može dekomponovati, ili koje se u posmatranom slučaju dalje ne dekomponuje na komponente, koje takođe predstavljaju obeležja, naziva se ***elementarnim obeležjem.*** Skup, niz, ili logički proizvod elementarnih obeležja predstavlja ***složeno obeležje***. Tom nizu obeležja se može pridružiti neko ime.

Primer 2.4. Obeležja *NAZIV_PROIZVODA*, *BOJA_AUTOMOBILA*, *IME_STANOVNIKA* predstavljaju elementarna obeležja različitih klasa entiteta. Složena obeležja predstavljaju, na primer, *ADRESA* = {*MESTO*, *ULICA*, *BROJ*}, {*IME*, *PRZ*, *MESTO*}, ili *DATUM_UPLATE* = {*DAN*, *MESEC*, *GODINA*}. □

Složena obeležja se, često, označavaju slovima sa kraja abecede, na primer X ili Y ili Z, a elementarna slovima sa početka abecede, na primer A, B ili C. Saglasno rečenom, složeno obeležje je $X = (A_1, A_2, \ldots, A_k)$, odnosno $X = \{A_1, A_2, \ldots, A_k\}$, gde su A_i, $1 \le i \le k$, elementarna obeležja. Pri tome, složeno obeležje X je jedanput predstavljeno kao niz, a drugi put kao skup.

2.1.2.2. Domen

Svakom obeležju odgovara jedan skup svih mogućnih vrednosti koje to obeležje, u konkretnim slučajevima, može imati. Taj skup vrednosti se naziva ***domenom obeležja***. Domen obeležja A se obeležava sa $dom(A)$. Takođe, domen može posedovati i svoje, posebno ime. Isti domen se može pridružiti većem broju različitih obeležja.

Primer 2.5. Za obeležje *BOJA_AUTOMOBILA* skup vrednosti je

$$dom(BOJA\_AUTOMOBILA) = \{bela, žuta, crna, plava, \ldots\}.$$

Obeležjima *IME_STUDENTA* i *IME_NASTAVNIKA* se može pridružiti isti domen sa nazivom *IME*. Taj domen sadrži, kao svoje elemente, sva moguća lična imena. □

U strukturama podataka, pojam domena se ne koristi u uobičajenom matematičkom smislu, kao skup originala funkcije. Domen može predstavljati skup originala funkcije, ali i skup slika. Pojam domena se koristi u smislu skupa iz kojeg semantički definisani objekti, kao što su tip entiteta i obeležje, uzimaju vrednosti.

2.1.2.3. *Tip entiteta*

Sa tačke gledišta zadataka informacionog sistema, nisu sva obeležja klase entiteta jednako važna. Od obeležja, bitnih za realizaciju zadataka informacionog sistema, gradi se ***model*** realne klase entiteta. Model klase entiteta naziva se ***tipom entiteta***.

Definicija 2.1. Izraz oblika $N(A_1,\ldots,A_n)$ predstavlja model skupa entiteta $E=\{e|P(e)\}$ i naziva se tipom entiteta, ako i samo ako N predstavlja ime skupa $\{e|P(e)\}$, a $A_1,\ldots,A_n$ obeležja entiteta skupa $\{e|P(e)\}$. □

Kao i svaki model, tip entiteta predstavlja samo približnu sliku klase entiteta realnog sistema. Neki put se za reprezentaciju klase entiteta, umesto oznake za tip entiteta $N(A_1,\ldots,A_n)$ koristi samo naziv N. Pošto niz $(A_1,\ldots,A_n)$ predstavlja složeno obeležje, tip entiteta $N(A_1,\ldots,A_n)$ predstavlja imenovano složeno obeležje. Naziv tipa entiteta predstavlja semantičku komponenetu tog apstraktnog opisa klase realnih entiteta. On daje smisao nizu obeležja, koji iza njega sledi.

Klasa entiteta poseduje konačno mnogo osobina zajedničkih svim realnim entitetima. Neka je $\{A_1,\ldots,A_m\}$ skup osobina klase entiteta E. Tada je skup obeležja $\{A_1,\ldots,A_n\}$ odabranih za izgradnju tipa entiteta kao modela klase E, pravi ili nepravi podskup skupa obeležja $\{A_1,\ldots,A_m\}$.

Primer 2.6. Tip entiteta *Student*(*BROJ_INDEKSA*, *IME*, *PREZIME*, *NAZIV_FAKULTETA*) reprezentuje sve studente jednog univerziteta. □

2.1.2.4. *Skup poveznika, uloga entiteta i tip poveznika*

Skup poveznika R predstavlja relaciju, u matematičkom smislu, između $n(\geq 2)$ skupova entiteta,

$$R=\{(e_1,\ldots,e_n)|\ e_i\in E_i,\ i=1,\ldots,n\}.$$

Pri tome, skupovi E_i ne moraju biti različiti. Svaka n - torka $(e_1,\ldots,e_n)$ u R predstavlja jedan poveznik. Svaki entitet e_i u n - torki ima svoju ulogu. Ako se uloge u_i entiteta e_i u torki eksplicitno navedu, redosled navođenja entiteta u torki postaje nevažan. Eksplicitno navođenje uloga posebno dobija na važnosti kada su u pitanju skupovi poveznika između entiteta, koji pripadaju istim skupovima entiteta. Treba, takođe, naglasiti i da između dva ista skupa entiteta može egzistirati više različitih skupova poveznika. U takvim slučajevima, entiteti bar jednog od povezanih skupova imaju različite uloge u svakom od posmatranih skupova poveznika. Ako poveznik povezuje entitete jednog skupa, naziva se ***rekurzivnim***.

Primer 2.7. Posmatraju se skup entiteta *Radnik* i skup entiteta *Radno_Mesto*. Neka je Aca elemenat u skupu *Radnik*, a *programer* u skupu *Radno_Mesto*. Poveznik (*Aca*, *programer*) ima semantiku "(radnik) je raspoređen na (radno mesto)".

Ako se u skup *Radnik* uvede relacija poretka sa semantikom "je rukovodilac", tada uređeni par (*Iva*, *Aca*) nosi informaciju da je *Iva Acin* rukovodilac. U paru (*Iva*, *Aca*)

oba entiteta pripadaju istom skupu, ali u povezniku imaju potpuno različite uloge. Te uloge u posmatranom povezniku nisu eksplicitno navedene. Isti poveznik, sa eksplicitno navedenim ulogama entiteta, bi bio (*Iva* (rukovodilac), *Aca* (potčinjeni)).

Posmatra se skup entiteta *Radnik* i skup entiteta *Projekat*, pri čemu *Lido* i *Faktura* pripadaju skupu *Projekat*. Parovi (*Aca* (nosilac), *Lido*) i (*Aca* (programer), *Faktura*) predstavljaju različite poveznike između ta dva skupa. U tim poveznicima su uloge entiteta *Aca* eksplicitno navedene. □

Definicija 2.2. Izraz oblika $N(E_1,\ldots, E_n; B_1,\ldots, B_k)$ predstavlja model skupa poveznika R i naziva se ***tipom poveznika***, ako i samo ako N predstavlja naziv skupa poveznika R, $E_1,\ldots, E_n$ su povezani skupovi entiteta, a $B_1,\ldots, B_k$ obeležja poveznika skupa $R = \{(e_1,\ldots, e_n) | e_i \in E_i, i = 1,\ldots, n\}$. □

Primer 2.8. Tip poveznika *Raspoređen* (*Radnik*, *Radno_Mesto*, *DATUM_RASP*) predstavlja model skupa poveznika oblika (*radnik*, *radno_mesto*) proširenih navođenjem njihove zajedničke osobine - datuma raspoređivanja (radnika na radno mesto).

Tip poveznika *Rukovodi* (*Radnik*, *Radnik*) je model rukovodne hijerarhije, čiji je jedan reprezent par (*Iva*, *Aca*) iz primera 2.7.

Tipovi poveznika *Rukovodi* (*Radnik*, *Projekat*) i *Radi* (*Radnik*, *Projekat*) predstavljaju modele odnosa između radnika i projekata. Informacija o različitim ulogama entiteta iz skupa *Radnik*, ugrađena je u naziv tipa poveznika. □

2.1.3. *Dijagrami tipova entiteta i tipova poveznika*

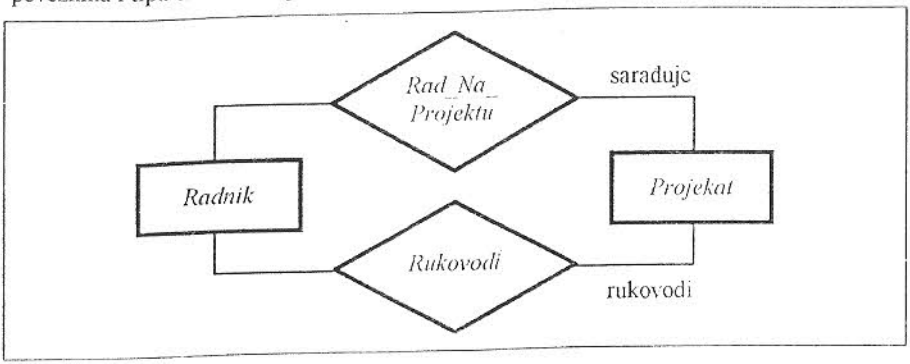
Model statičke strukture realnog sistema, realizovan putem ER modela podataka, po pravilu se predstavlja uz pomoć takozvanih ER dijagrama. Tip entiteta se crta kao pravougaonik sa upisanim nazivom (tipa entiteta). Tip poveznika se crta kao romb sa upisanim nazivom (tipa poveznika). Ako tip poveznika R povezuje tipove entiteta $E_1,\ldots, E_k$ tada se crta po jedan neusmereni poteg od romba R do geometrijske slike svakog od E_i.

Skupovi vrednosti (domeni) se prikazuju kao ovali sa upisanim nazivom domena i povezuju orijentisanim potegom sa odgovarajućim tipom entiteta ili tipom poveznika. Orijentacija potega je ka ovalu skupa vrednosti. Naziv obeležja se upisuje na poteg domena, ako se nazivi obeležja i domena razlikuju.

ER dijagrami se mogu crtati na dva nivoa detaljnosti. To su: nivo detaljnosti naziva i nivo detaljnosti obeležja. Nivo detaljnosti naziva daje preglednije dijagrame, a nivo detaljnosti obeležja daje dijagrame sa većom količinom informacije. Da bi se dobili dijagrami i sa dobrom preglednošću i sa većom količinom informacija, dva nivoa detaljnosti crtanja dijagrama se kombinuju. Na jednom dijagramu se predstavljaju tipovi entiteta i tipovi poveznika na nivou detaljnosti naziva, a na nizu drugih dijagrama predstavlja se svaki tip entiteta i tip poveznika sa pripadajućim obeležjima i skupovima vrednosti.

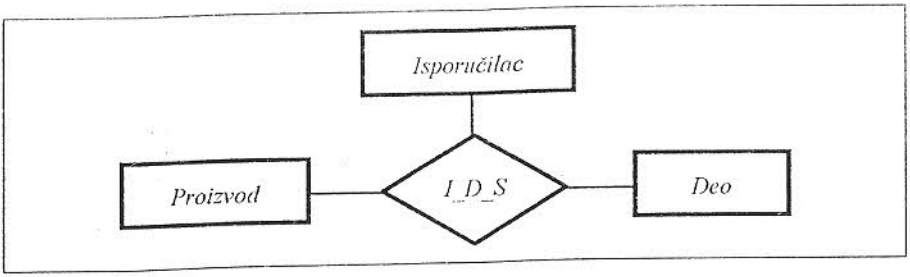
Primer 2.9. Na slici 2.1 prikazana su dva tipa entiteta *Radnik* i *Projekat*. Između ta dva tipa entiteta postoje dva tipa poveznika *Rad_Na_Projektu* i *Rukovodi*. Entiteti tipa *Radnik* igraju dve uloge u poveznicima dva tipa. To su uloga saradnika na projektu i uloga

rukovodioca projekta. Te uloge su ispisane uz potege između odgovarajućeg tipa poveznika i tipa entiteta *Projekat*.

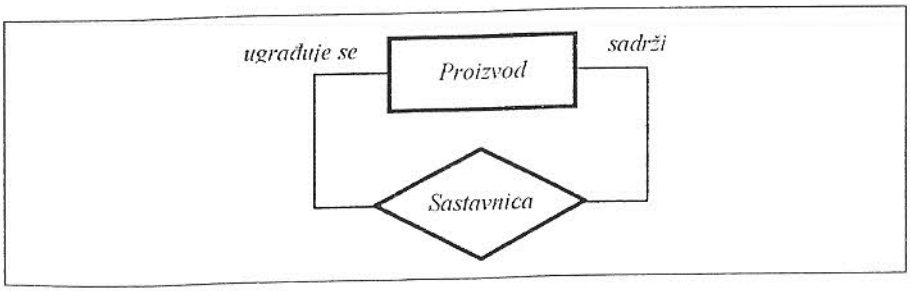


Slika 2.1.

Na slici 2.2 su prikazana tri tipa entiteta *Isporučilac*, *Deo* i *Proizvod* između kojih je definisan jedan tip poveznika *I_D_S*, sa semantikom isporučilac (*I*) isporučuje deo (*D*) za ugradnju u proizvod (*P*).



Slika 2.2.

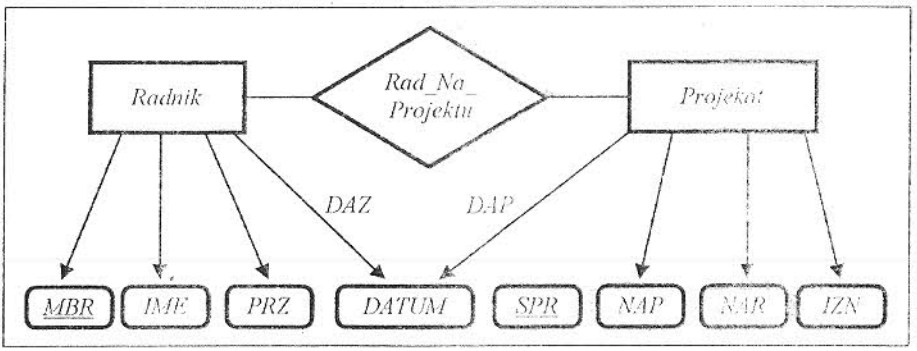


Slika 2.3.

Na slici 2.3 prikazan je tip entiteta *Proizvod* i tip poveznika *Sastavnica*. Svaka pojava tipa poveznika *Sastavnica* sadrži dva entiteta iz skupa *Proizvod*. Jedan ima ulogu

proizvoda, a drugi komponente, koja se u taj proizvod ugrađuje. Ove uloge su na slici 2.3 označene, redom, kao "sadrži" i "ugrađuje se". Drugim rečima, na slici 2.3 je, putem koncepata modela entiteta i poveznika, predstavljen model sastavnice proizvoda. □

Primer 2.10. Na slici 2.4 su prikazani tipovi entiteta *Radnik*, *Projekat* i tip poveznika *Rad_Na_Projektu* sa odgovarajućim domenima (skupovima vrednosti). Obeležja *DAZ* (datum zaposlenja radnika) i *DAP* (datum početka rada na projektu) dele isti skup vrednosti - domen sa imenom *DATUM*. □



Slika 2.4.

2.1.4. *Ekstenzija ER modela podataka*

Ekstenzionalne koncepte ER modela podataka čine: pojava tipa entiteta, skup pojava tipa entiteta, pojava tipa poveznika i skup pojava tipa poveznika. Podatak predstavlja primitivni koncept, putem kojeg se komponuju nabrojani složeniji koncepti.

2.1.4.1. *Podatak*

Entiteti i poveznici se opisuju putem skupova parova (obeležje, vrednost). Obeležje predstavlja semantičku (smisaonu) komponentu te dvojke.

Definicija 2.3. Par (obeležje, vrednost) predstavlja ***podatak*** o nekom entitetu ili povezniku, ako obeležje predstavlja osobinu posmatranog entiteta ili poveznika, a vrednost pripada domenu tog obeležja. □

Primer 2.11. Ako se posmatra obeležje *MATIČNI_BROJ_RADNIKA* i jedna njegova vrednost *110451*, tada *110451* predstavlja podatak o nekom radniku jedino ako se unapred zna da *110451* predstavlja vrednost obeležja *MATIČNI_BROJ_RADNIKA*. U suprotnom *110451* ne predstavlja podatak, jer mu je smisao neodređen.

Broj 7 ne predstavlja podatak, jer mu je smisao neodređen. Može se interpretirati na više načina: kao ocena studenta, kao starost deteta i slično. □

Formalno, neka je $E = \{e_i | i = 1,\ldots, m\}$ klasa entiteta, a A jedno od obeležja te klase. Obeležje A predstavlja funkciju $A: E \rightarrow dom(A)$. Drugim rečima, obeležje A pridružuje svakom entitetu $e_i \in E$ jednu vrednost iz $dom(A)$, tako da $A(e_i)$ predstavlja podatak o e_i s obzirom na A.

Vrednost elementarnog obeležja predstavlja elementarni podatak, a vrednost složenog obeležja predstavlja složeni podatak. Ako je $X = \{A_1,\ldots, A_k\}$, tada je $dom(X) \subseteq dom(A_1) \times \ldots \times dom(A_k)$, tako da je $(a_1,\ldots, a_k)$ jedna vrednost obeležja X, gde je, za svako i, $a_i \in dom(A_i)$.

Obeležje čije se vrednosti dobijaju primenom nekog algoritma na vrednosti drugih obeležja naziva se ***izvedenim obeležjem***, a njegove vrednosti ***izvedenim podacima***.

Primer 2.12. Obeležje *SREDNJA_OCENA* je izvedeno. Njegove vrednosti se dobijaju sabiranjem vrednosti obeležja *OCENA* za sve položene predmete i deljenjem tog zbira vrednošću obeležja *BROJ_POLOŽENIH_ISPITA*. □

2.1.4.2. *Pojava i skup pojava tipa entiteta*

Svaka klasa entiteta predstavlja skup sličnih entiteta. Svakom entitetu posmatrane klase odgovaraju konkretne vrednosti obeležja klase entiteta. U okviru informacionog sistema, svaki entitet se predstavlja modelom koji sadrži konkretne vrednosti onih obeležja, uz pomoć kojih je opisana klasa entiteta. ***Uređenje podataka*** u modelu entiteta diktirano je ***uređenjem skupa obeležja*** u tipu entiteta. Obeležjima u tipu entiteta odgovaraju vrednosti tih obeležja (podaci) u modelu entiteta istim redom.

Model jednog entiteta naziva se ***pojavom*** odgovarajućeg tipa entiteta. Tip entiteta i pojava entiteta predstavljaju apstraktne opise konkretnih realnih entiteta. Tip entiteta predstavlja model entiteta na višem nivou apstrakcije nego pojava tipa entiteta.

Formalno, za dati tip entiteta $N(A_1,\ldots, A_n)$ skup funkcija

$$\{A_i : E \rightarrow dom(A_i) | i = 1,\ldots, n\}$$

jednoznačno određuje funkciju

$$(A_1,\ldots, A_n) : E \rightarrow dom(A_1) \times \ldots \times dom(A_n),$$

gde je E klasa entiteta, a

$$dom(A_1) \times \ldots \times dom(A_n) = \{(a_1,\ldots, a_n) | a_i \in dom(A_i),\ i = 1,\ldots, n\}.$$

Funkcija $(A_1,\ldots, A_n)$ je definisana sa $(A_1,\ldots, A_n)(e) = (A_1(e),\ldots, A_n(e))$, gde je $e \in E$ entitet. Drugim rečima, $(a_1,\ldots, a_n)$ predstavlja pojavu tipa entiteta $N(A_1,\ldots, A_n)$, ako je zadovoljena kvantifikatorska formula

$$\mathcal{F}((a_1,\ldots, a_n)) = (\exists e \in E)(\forall i \in \{1,\ldots, n\})(a_i = A_i(e)),$$

a skup P pojava tipa entiteta je

$$P = \{(a_1,\ldots, a_n) \mid \mathcal{F}((a_1,\ldots, a_n))\}.$$

Uređenje podataka u n - torki $(a_1,\ldots, a_n)$ je bitno, jer nosi informaciju o tome da je $a_i \in dom(A_i)$.

Primer 2.13. Za tip entiteta *Student* iz primera 2.6, moguće pojave predstavljaju sledeće četvorke: (*159, Ivo, Ban, Fakultet tehničkih nauka*) i (*313, Eva, Tot, Ekonomski fakultet*). □

2.1.4.3. Ključ tipa entiteta

Definicija pojma entiteta kao "jedinice posmatranja" ukazuje na potrebu da se entiteti posmatrane klase mogu razlikovati. To zahteva da i modeli dva entiteta budu različiti. Neka su e_1 i e_2 entiteti klase E, a $N(A_1,\ldots, A_n)$ tip entiteta, tada mora važiti $(A_1,\ldots, A_n)(e_1) \neq (A_1,\ldots, A_n)(e_2)$. Znači, mora postojati neprazan skup obeležja $X \subseteq \{A_1,\ldots, A_n\}$ takav da je $X(e_1) \neq X(e_2)$.

Definicija 2.4. Neka je $P = \{p_i \mid i = 1,\ldots, k\}$ skup pojava tipa entiteta $N(A_1,\ldots, A_n)$, a $p[X]$ restrikcija pojave p na obeležje X. Obeležje X predstavlja *ključ* tipa entiteta $N(A_1,\ldots, A_n)$ ako, za svaki skup P pojava tipa entiteta N, važe sledeća dva uslova:

1° $(\forall p_i, p_j \in P)(p_i \neq p_j \Rightarrow p_i[X] \neq p_j[X])$ i
2° $(\forall X' \subset X)(\neg\, \mathbf{1°})$. □

Ako X zadovoljava samo uslov **1°**, naziva se ***superključem*** tipa entiteta N. Uslov **1°** ukazuje na takozvanu jedinstvenost vrednosti ključa. U skupu pojava tipa entiteta ne postoje dve pojave sa istom vrednošću ključa. Uslov **2°** je uslov takozvane minimalnosti ključa. Ključ ne poseduje suvišna obeležja.

Svaki tip entiteta poseduje bar jedan ključ. Neka je $X = \{A_1,\ldots, A_n\}$. Tada uslov **1°** trivijalno važi. Ako važi i uslov **2°**, X predstavlja ključ tipa entiteta $N(A_1,\ldots, A_n)$. Inače, mora postojati neprazan podskup skupa $\{A_1,\ldots, A_n\}$ za koji važi **1°** i **2°**. Ovo razmatranje, ujedno, sugeriše i jedan od mogućih postupaka za određivanje ključa tipa entiteta.

Jedan tip entiteta može posedovati više ključeva. Nazivaju se ***ekvivalentnim***. Jedan od ekvivalentnih ključeva se bira za ***primarni***. U okviru notacije za tip entiteta sva obeležja jednog ključa podvlače se jednom, kontinualnom linijom.

Primer 2.14. Tip entiteta *Student* (*BROJ_INDEKSA, IME, PREZIME, ADRESA, MATIČNI_BROJ_STANOVNIKA*) poseduje dva ključa. To su *BROJ_INDEKSA* i *MATIČNI_BROJ_STANOVNIKA*. Ako je reč o tipu entiteta, definisanom u projektu fakultetskog informacionog sistema, obeležje *BROJ_INDEKSA* bi, najverovatnije, bilo izabrano za primarni ključ. □

2.1.4.4. Pojava tipa poveznika

Jedan tip poveznika predstavlja apstraktni model većeg broja različitih skupova torki sličnih osobina definisanih nad istim domenima. Jedna torka predstavlja jednu ***poja-***

vu tipa poveznika. Ako je tip poveznika binaran, tada njegove pojave predstavljaju uređene trojke. Prvu komponentu trojke predstavlja jedna pojava prvog tipa entiteta, drugu komponentu trojke predstavlja jedna pojava drugog tipa entiteta. Treću komponentu predstavlja torka $(b_1,..., b_k) \in dom(B_1) \times ... \times dom(B_k)$. Pošto svaku pojavu tipa entiteta reprezentuje odgovarajuća vrednost ključa, dovoljno je da pojava tipa poveznika sadrži vrednosti primarnih ključeva povezanih tipova entiteta umesto kompletnih pojava tih tipova entiteta.

Saglasno rečenom, tip poveznika između tipova entiteta $E_1,..., E_n$ se može predstaviti i kao imenovani niz obeležja, u oznaci $R(A_1,..., A_m)$, pri čemu važi $K_1,..., K_n \subseteq \{A_1,..., A_m\}$, gde K_i, $i = 1,..., n$, predstavljaju primarne ključeve povezanih tipova entiteta. Pri tome važi

$$\{B_1,..., B_k\} = \{A_1,..., A_m\} \setminus \bigcup_{i=1}^{n} K_i$$

gde je $\{B_i | i = 1,..., k\}$ skup obeležja skupa poveznika R. Ovaj skup obeležja može biti prazan.

Opisani način predstavljanja tipa poveznika zahteva da se definiše i pojam ključa tipa poveznika. Ključ K tipa poveznika R predstavlja pravi ili nepravi podskup unije primarnih ključeva povezanih tipova entiteta, u oznaci

$$K \subseteq \bigcup_{i=1}^{n} K_i.$$

Primer 2.15. Ključ tipa poveznika *Rad_Na_Projektu* sa slike 2.4 je $K = \{MBR, SPR\}$, a ključ tipa poveznika *Rasporeden* sa slike 2.5 je $K = \{MBR\}$. Ključ tipa poveznika *Rasporeden* je pravi podskup unije ključeva povezanih tipova entiteta. □

Primer 2.16. Za tip poveznika

Ispit (*BROJ_INDEKSA, IDBROJ_PREDMETA*, *OCENA*),

ključ predstavlja složeno obeležje {*BROJ_INDEKSA, IDBROJ_PREDMETA*}.

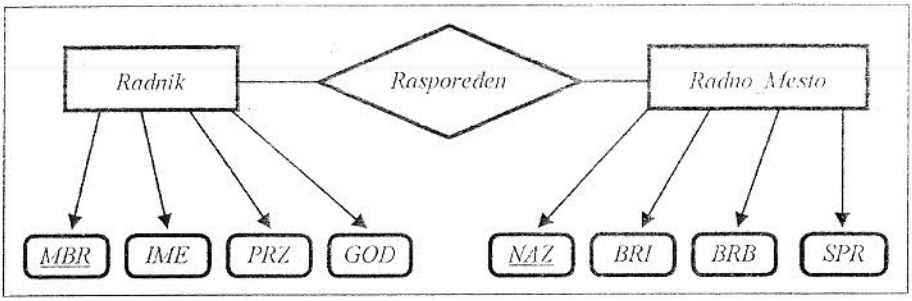
Tip poveznika *Poveravanje* (*IDBROJ_NASTAVNIKA*, *IDBROJ_PREDMETA*) opisuje odnos između nastavnika i predmeta u realnom sistemu, gde se nastavniku poverava izvođenje predavanja iz predmeta. Ako se za ključ tipa entiteta *Poveravanje* odredi obeležje *IDBROJ_NASTAVNIKA*, tada tip entiteta opisuje odnos, prema kojem se svakom nastavniku poverava samo jedan predmet. Ako se za ključ odredi *IDBROJ_PREDMETA*, tip entiteta opisuje realnu situaciju u kojoj nastavu iz svakog predmeta izvodi samo jedan nastavnik.

Ako se pretpostavi da se istom nastavniku može poveriti izvođenje nastave iz više predmeta, a isti predmet može poveriti većem broju nastavnika, tada sva obeležja tipa entiteta *Poveravanje* (*IDBROJ_NASTAVNIKA*, *IDBROJ_PREDMETA*) predstavljaju ključ. Može se zaključiti da semantika modela skupa pojava tipa poveznika značajno zavisi od odabranog ključa. □

2.1.4.5. Predstavljanje ekstenzije ER modela

Ekstenzija modela entiteta i poveznika se predstavlja putem tabela, slično kao u relacionom modelu. Svakom tipu entiteta ili tipu poveznika odgovara jedna tabela sa svim njegovim obeležjima u zaglavlju tabele i podacima o entitetima u vrstama tabele. Vrste predstavljaju modele entiteta ili poveznika posmatranog skupa. Ove tabele se, redom, nazivaju relacijama entiteta i relacijama poveznika.

Primer 2.17. Na slici 2.5 je prikazan dijagram entiteta i poveznika sa tipovima entiteta *Radnik* i *Radno_Mesto* i tipom poveznika *Raspoređen*. Obeležje *MBR* predstavlja ključ tipa poveznika *Raspoređen*. Na slici 2.6 su prikazane odgovarajuće relacije entiteta i relacija poveznika. Ključevi tipova entiteta i poveznika su podvučeni. Treba zapaziti da, saglasno tabelama na slici 2.6, postoje neraspoređeni radnici i radna mesta na koja nije raspoređen nijedan radnik.



Slika 2.5.

Radnik

MBR	*IME*	*PRZ*	*GOD*
88	*Eva*	*Pap*	*1950*
50	*Ivo*	*Ban*	*1950*
01	*Ana*	*Ras*	*1961*
81	*Ana*	*Tot*	*1963*

Radno_Mesto

NAZ	*BRI*	*BRB*	*SPR*
Programer	*5*	*530*	*VSS*
Daktilograf	*3*	*700*	*SSS*
Projektant	*3*	*700*	*VSS*
Kurir	*1*	*300*	*NSS*

Raspoređen

MBR	*NAZ*
88	*Daktilograf*
01	*Programer*
81	*Programer*

Slika 2.6.

Između obeležja i tipa entiteta nema jasne razlike, jer se imenovano složeno obeležje koristi za predstavljanje tipa entiteta. Na prvi pogled teško je povući jasnu granicu i između tipa entiteta i tipa poveznika. Međutim, za povlačenje ove granice postoji pouzdan kriterijum. To je pitanje da li postojanje pojave jednog tipa entiteta zavisi od postojanja pojave drugog. Ako torke relacije mogu samostalno da egzistiraju, relacija predstavlja ekstenziju tipa entiteta, inače je reč o tipu poveznika.

Primer 2.18. Tvrđenje da je granica između pojma obeležja, tipa entiteta i tipa poveznika često nejasna, može se ilustrovati na sledeći način. Sa jedne tačke gledišta, *Projekat* i *Projektant* predstavljaju tipove entiteta, a *Rad_Na_Projektu* tip poveznika. Pri tome, tip entiteta *Projekat* može sadržati obeležje *NARUČILAC*. Sa druge tačke gledišta, *Naručilac* može predstavljati tip entiteta, a *Projekat* tip poveznika (u smislu da projekat povezuje naručioca i projektanta). Sa treće tačke gledišta, *Rad_Na_Projektu* može predstavljati tip entiteta sa ključem *IDBAK* (čije vrednosti jednoznačno identifikuju aktivnosti na bilo kom projektu), obeležjem *PROJEKTANT* i obeležjem *DATUM_POČETKA*.

Ako se iz relacije entiteta *Radnik* na slici 2.6 ukloni torka (*88, Eva, Pap, 1950*), torka (*88, Daktilograf*) u relaciji poveznika *Raspoređen* gubi svaki smisao, jer u relaciji *Radnik* ne postoji pojava sa vrednošću ključa *MBR = 88*. Međutim, ako se iz relacije *Raspoređen* ukloni torka (*01, Programer*), to ne znači da treba bilo šta menjati u relacijama *Radnik* i *Radno_Mesto*. Jednostavno, radnik sa *MBR = 01* nije više raspoređen na radno mesto programera. □

2.2. *Integritetna komponenta ER modela podataka*

S obzirom da je model entiteta i poveznika prevashodno namenjen za predstavljanje konceptualne šeme, poseduje relativno dobre mogućnosti za definisanje ograničenja. U modelu se ta ograničenja izražavaju, uglavnom, eksplicitno. U ta ograničenja spadaju:

- integritet domena,
- kardinalnost tipa poveznika sa svojim brojnim varijantama i semantičkim interpretacijama i
- slabi tip entiteta.

2.2.1. *Integritet domena*

Inegritet domena predstavlja ograničenje, svojstveno praktično svim modelima podataka. U opštem slučaju, integritet ili ograničenje domena je trojka (tip podatka, dužina podatka, uslov). Tip i dužina podatka je standardno programsko - jezičko ograničenje i predstavlja obavezni deo ograničenja domena. Tip podatka definiše vrstu znakova, putem kojih se izražava vrednost obeležja. Dužina podatka se izražava putem maksimalnog broja znakova, koji mogu biti upotrebljeni za izražavanje vrednosti obeležja. Prema tome, tip i dužina podatka ograničavaju vrednosti obeležja na tip i broj znakova.

Primer 2.19. Jedno standardno ograničenje domena može predstavljati sledeći skup tipova podataka sa dužinama:

- INTEGER (broj cifara),
- REAL (broj cifara ispred zareza, broj cifara iza zareza),
- CHARACTER (broj znakova),
- LOGICAL,

- DATE.

Tip podataka "logical" ukazuje da je reč o logičkoj promenljivoj, koja uzima vrednosti iz skupa {T, ⊥}.

Pridruživanje *dom*(*OCE*) standardnog ograničenja *INTEGER* (*2*), ukazuje da važi *dom*(*OCE*) = {*0*, *1*,..., *99*}.

Pridruživanje *dom*(*IME*) standardnog ograničenja *CHARACTER* (*10*), ukazuje da *dom*(*IME*) može pripadati bilo koji niz od *10* znakova. □

Standardna ograničenja domena, često, nisu dovoljno precizna. Zato se uvodi i uslov, kao treća, opciona, komponenta ograničenja domena. Uslov može biti regularni izraz ili funkcija. Postoje prosti i složeni regularni izrazi.

Prosti regularni izrazi se definišu s obzirom na neke konstante. Te konstante moraju zadovoljiti standardno ograničenje domena. Neka je k konstanta iz domena pridruženog obeležju A, a θ operator poređenja iz skupa $\{<, >, \leq, \geq, \neq, =\}$. Tada proste regularne izraze predstavljaju (θk) i $(k_1, k_2, \ldots, k_n)$. Izraz (θk) ukazuje da sve vrednosti pridružene obeležju A moraju biti u relaciji θ sa konstantom k. Izraz $(k_1, k_2, \ldots, k_n)$ ukazuje da obeležje A sme uzimati vrednosti iz skupa $\{k_1, k_2, \ldots, k_n\}$.

Složeni regularni izrazi se komponuju od drugih regularnih izraza (prostih ili složenih) njihovim povezivanjem putem logičkih operatora $\wedge$, $\vee$ i $\neg$.

Primer 2.20. Pridruživanje *dom*(*OCE*) sledeće trojke sa prostim regularnim izrazom (*INTEGER*, (*2*), *<11*), ukazuje da obeležje *OCE* može uzeti vrednost iz skupa {*0*, *1*,..., *10*}. Pridruživanje *dom*(*OCE*) trojke sa prostim regularnim izrazom (*INTEGER*, (*2*), (*6*, *7*,..., *10*)) ograničava vrednosti tog obeležja na uobičajeni opseg pozitivnih ocena. Do istog rezultata se dolazi i pridruživanjem *dom*(*OCE*) sledeće trojke sa složenim regularnim izrazom (*INTEGER*, (*2*), *>5* ∧ *<11*). □

Za izražavanje ograničenja nad domenima nekih obeležja nisu dovoljni ni regularni izrazi. Tada se koriste funkcije, čije argumente predstavljaju neka obeležja, ili se koriste algoritmi.

Primer 2.21. Posmatraju se obeležja *KOL* (količina artikla), *JED_CEN* (jedinična cena artikla) i *IZNOS*. Ograničenje da *dom*(*IZNOS*) sme sadržati samo vrednosti dobijene množenjem elemenata *dom*(*KOL*) sa elementima iz *dom*(*JED_CEN*), može se izraziti putem funkcije *iznos* = *kol* * *jed_cen*.

Posmatraju se obeležja *REGIJA* i *PTT_BROJ*. Ograničenje da u regiji "Vojvodina" svi poštanski brojevi počinju cifrom *2*, može se izraziti putem sledećeg algoritma:

AKO JE *REGIJA* = "*Vojvodina*" ***TADA***
PTT_BROJ = "*2**"
INAČE
KRAJ AKO.

Pri tome, znak * predstavlja zamenu za niz bilo kojih znakova. □

Nula vrednost predstavlja specijalno ograničenje domena. Putem tog ograničenja se specificira da li obeležje može imati nedefinisanu vrednost. Sama nula vrednost ima, najčešće jedno od sledeća dva značenja:

- postojeća, ali nepoznata vrednost ili
- neprimereno svojstvo.

Bitno je naglasiti da nula vrednost ne predstavlja broj *0* već specijalnu vrednost, čije značenje se, najčešće, interpretira na jedan od navedena dva načina.

Primer 2.22. Posmatra se obeležje *K_TEL* (broj kućnog telefona). Za neke ljude to obeležje predstavlja neprimereno svojstvo, jer telefon kod kuće nemaju, te se za evidentiranje njihovog broja telefona koristi nula vrednost sa odgovarajućom interpretacijom značenja. Drugi ljudi imaju telefon, ali je njegov broj, u trenutku evidentiranja, nepoznat. Tada se ponovo koristi nula vrednost, ali sada u drugom značenju. □

2.2.2. *Kardinalnost tipa poveznika*

Često se od modela realnog sistema i modela baze podataka informacionog sistema zahteva da pruži informaciju ne samo o vezama između klasa entiteta već i o prirodi odnosa između entiteta povezanih klasa.

Primer 2.23. U slučaju modela dela realnog sistema prikazanog na slikama 2.5 i 2.6, može se zahtevati da već apstraktni model na slici 2.5 ukaže da:

- jedan radnik može biti raspoređen na najviše jedno radno mesto,
- na jedno radno mesto može biti raspoređeno više radnika i
- na jedno radno mesto ne mora biti raspoređen nijedan radnik. □

U modelu realnog sistema, informaciju o prirodi odnosa između entiteta povezanih klasa daje takozvani ***kardinalitet*** tipa poveznika R odnosno kardinalitet odgovarajuće relacije R.

Posmatra se binarna relacija R između skupova pojava dva tipa entiteta E_1 i E_2. Ova relacija se može predstaviti putem dva preslikavanja $R_1 : E_1 \rightarrow \mathcal{P}(E_2)$ i $R_2 : E_2 \rightarrow \mathcal{P}(E_1)$, gde je $\mathcal{P}(E)$ partitivni skup skupa E. Preslikavanjima R_1 i R_2 se može dati sledeća semantička interpretacija. R_1 je uloga entiteta iz skupa E_1, a R_2 je uloga entiteta iz skupa E_2 u njihovoj vezi, opisanoj relacijom R. Za svako od ovih preslikavanja se definiše minimalni i maksimalni kardinalitet. Pojam kardinaliteta preslikavanja se odnosi na brojnost (kardinalitet) elementa partitivnog skupa u koji se preslikava jedan elemenat skupa originala. Minimalni i maksimalni kardinalitet jednog preslikavanja, recimo R_1, određuje se identifikacijom minimalne i maksimalne brojnosti podskupa skupa E_2 u koji se može preslikati neki (bilo koji) elemenat skupa E_1. Kardinalitet preslikavanja R_1 se označava sa $R_1(E_2(a_2, b_2))$, gde je a_2 minimalni, a b_2 maksimalni kardinalitet. Kardinalitet relacije R, odnosno tipa poveznika R, se označava sa $R(E_1(a_1, b_1) : E_2(a_2, b_2))$.

Parametrima a i b se najčešće dodeljuju sledeće karakteristične vrednosti:

- parametru a se dodeljuje vrednost *0*, ako se bar jedan elemenat skupa originala preslikava u prazan skup, inače mu se dodeljuje vrednost *1*,

- parametru b se dodeljuje vrednost 1, ako kardinalitet slike svakog originala nije veći od 1, inače mu se dodeljuje vrednost N ili M, gde je $1 < N$, $M \leq |E|$.

Ako je $b_2 = 1$, preslikavanje $R_1 : E_1 \rightarrow \mathcal{P}(E_2)$ se transformiše u preslikavanje $R : E_1 \rightarrow E_2$.

Ako je $a_2 = 1$, odgovarajuće preslikavanje se naziva totalnim, jer se svaki original preslikava u neprazan podskup. Drugim rečima, u realnom sistemu svaki entitet prve klase entiteta povezan je sa barem jednim entitetom druge klase. Slučaj $a_2 = 1$ se naziva ***egzistencijalnim ograničenjem***, jer se može tumačiti i na sledeći način: da bi e_1 pripadao E_1, mora biti povezan sa barem jednim e_2 iz E_2.

Preslikavanje R_1 je parcijalno, ako je $a_2 = 0$. Drugim rečima, ako u realnom sistemu može postojati bar jedan entitet klase E_1, koji nije povezan ni sa jednim entitetom klase E_2, tada je $a_2 = 0$.

Pojam kardinaliteta tipa poveznika se može objasniti i na sledeći način. Neka $e_1 \in E_1$, a $e_2 \in E_2$ i neka graf relacije R sadrži bar jedan uređeni par (e_1, e_2). Ako može postojati $e_1 \in E_1$ takvo da se ne javlja nijedanput kao prva komponenta para (e_1, e_2), tada je $a_2 = 0$, inače $a_2 = 1$. Ako za svako $e_1 \in E_1$ važi da se javlja najviše jedanput kao prva komponenta para (e_1, e_2), tada je $b_2 = 1$, inače je $b_2 = N$. U slučaju ovakve interpretracije pojma kardinaliteta tipa poveznika, odgovarajuća notacija bi bila $R(E_1(a_2, b_2) : E_2(a_1, b_1))$.

Kardinalitet $a_2 = 0$ ukazuje da može, ali ne mora postojati entitet klase E_1, koji nije povezan sa bar jednim entitetom klase E_2. Kardinalitet $a_2 = 1$ ukazuje da svaki entitet klase E_1 mora biti povezan sa bar jednim entitetom klase E_2. Kardinalitet $b_2 = 1$ ukazuje da svaki entitet klase E_1 može biti povezan sa najviše jednim entitetom klase E_2, ali ne mora ni sa jednim. Kardinalitet $b_2 = N$ ukazuje da može, ali ne mora, postojati bar jedan entitet klase E_1 koji je povezan sa više od jednog entiteta klase E_2.

Minimalnom kardinalitetu $a = 1$ se može dati i semantička interpretacija da on ukazuje na ***prethođenje*** i ***sledenje***. Naime, ako je $R_1(E_2(1, b_2))$, to ukazuje da prvo mora postojati neki entitet e_2 u E_2, da bi entitet e_1, koji je sa njim u vezi, mogao biti uključen u skup E_1. Znači, ako je $a_2 = 1$, postojanje entiteta e_2 iz skupa E_2 mora prethoditi nastanku entiteta e_1 iz E_1.

S obzirom na maksimalne vrednosti kardinaliteta, tipovi poveznika se mogu podeliti u tri grupe. To su grupa $M:N$, grupa $1:N$ i grupa $1:1$.

2.2.3. *Predstavljanje kardinaliteta tipa poveznika u ER dijagramima*

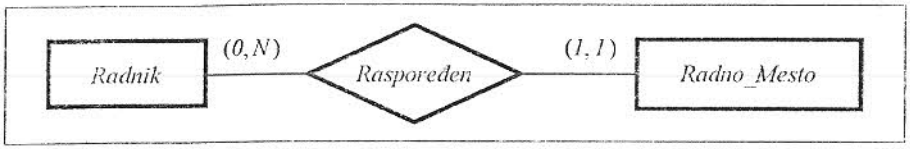
U dijagramima entiteta i poveznika kardinalitet tipa poveznika se predstavlja navođenjem ili para (a_1, b_1) i (a_2, b_2) ili samo maksimalnih vrednosti kardinaliteta b_1 i b_2 uz grafičku predstavu odgovarajućeg tipa entiteta.

Postoje dva postupka navođenja kardinaliteta u ER dijagramima. Kod jednog, par (a_1, b_1) se navodi na potegu uz tip entiteta E_1, a par (a_2, b_2) se navodi na potegu uz tip entiteta E_2. Smisao ovog postupka navođenja kardinaliteta je da se minimalna i maksimalna brojnost podskupova, recimo, skupa entiteta E_2, u koje se može preslikati jedan elemenat

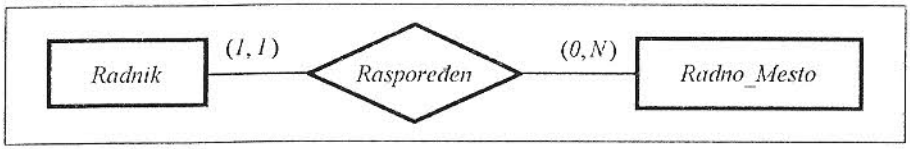
skupa entiteta E_1, upisuje baš uz tip entiteta E_2. Kod drugog postupka predstavljanja kardinaliteta, par (a_1, b_1) se navodi na potegu uz tip entiteta E_2, a par (a_2, b_2) se navodi na potegu uz tip entiteta E_1. Semantika ovog postupka predstavljanja je da se jedna pojava tipa entiteta E_1 javlja, kao prva komponenta u skupu pojava tipa poveznika minimalno a_2 i maksimalno b_2 puta, a da se jedna pojava tipa entiteta E_2 javlja, kao druga komponenta u skupu pojava tipa poveznika minimalno a_1 i maksimalno b_1 puta.

Primer 2.24. Na slici 2.7 prikazan je ER dijagram sa kardinalitetima tipova poveznika dobijenih upisivanjem dvojki (a_i, b_i) uz tip entiteta N_i, dok je na slici 2.8 prikazan ER dijagram sa kardinalitetima dobijenim upisivanjem dvojki (a_i, b_i) uz tip entiteta E_j, gde je $i \neq j$. U oba slučaja, opisan je isti realni sistem, u kojem:

- radnik mora biti raspoređen na tačno jedno radno mesto i
- na jedno radno mesto može biti raspoređeno više radnika, ali mogu postojati radna mesta na koja nije niko raspoređen. □



Slika 2.7.

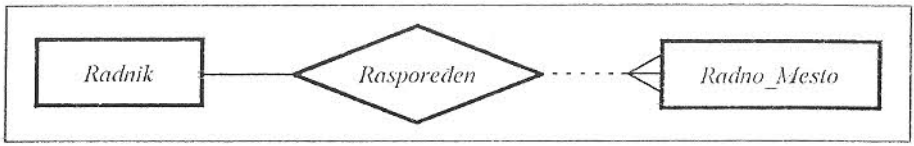


Slika 2.8.

Teško je reći da bilo koji od opisanih postupaka predstavljanja kardinaliteta tipa poveznika u ER dijagramima ima prednost nad drugim. Primena jednog, a ne drugog zavisi od ukusa i navika projektanta. Na jugoslovenskom informatičkom prostoru je u široj upotrebi drugi postupak (par (a_i, b_i) se upisuje uz tip entiteta N_j) predstavljanja kardinaliteta tipa poveznika i to je jedini razlog što će se, u daljem tekstu, koristiti taj postupak.

Neki put se u ER dijagramima parovi (a, b) predstavljaju putem specijalnih geometrijskih simbola. Minimalni kardinalitet $a = 0$ se predstavlja putem isprekidanog potega između tipa entiteta i tipa poveznika, a $a = 1$ se predstavlja punom linijom. Maksimalni kardinalitet $b = N$ se predstavlja račvanjem dela potega uz geometrijsku predstavu tipa entiteta, a $b = 1$ potegom, koji nema račvu na svom kraju.

Primer 2.25. Na slici 2.9 je prikazan ER dijagram istog dela realnog sveta kao na slici 2.8, ali navođenjem kardinaliteta tipova poveznika putem specijalnih geometrijskih simbola. □



Slika 2.9.

2.3. Karakteristične strukture ER modela podataka

U narednom tekstu je analizirana semantika jednog broja karakterističnih struktura ER modela podataka. S obzirom na mali broj osnovnih koncepata (tip entiteta i tip poveznika) i na krajnju jednostavnost njihove veze, osnovna pažnja u narednoj analizi je poklonjena semantici kardinaliteta tipa poveznika. U analizi se polazi od karakteristične strukture, a zatim se interpretira kakav treba da bude realni sistem, pa da posmatrana ER struktura bude veran model nekog njegovog dela. Pošto je opisano preslikavanje jedan na jedan, to znači da važi i obratno. Na osnovu poznavanja karakteristika dela realnog sistema, može se izabrati odgovarajuća ER struktura, kao njegova verna slika.

2.3.1. Strukture sa kardinalitetima grupe $M : N$

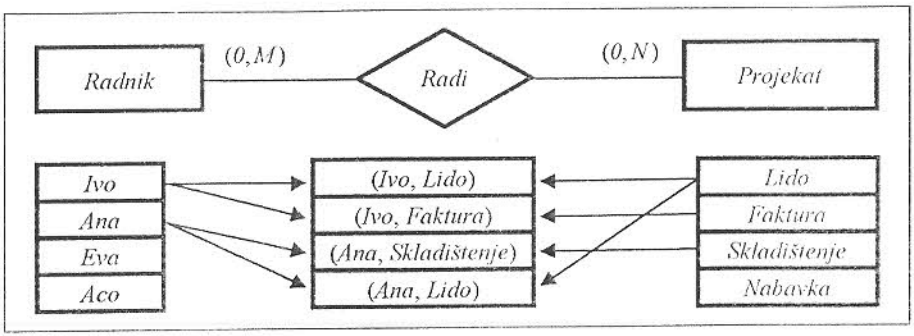
Najopštiji slučaj kardinaliteta tipa poveznika opisuje se sa $R(E_1(a_2, M) : E_2(a_1, N))$, što se javlja kada, u realnom sistemu, jedan entitet klase E_1 može biti u vezi sa $M(>1)$ entiteta klase E_2 i kada isto važi i obratno. Tada se kaže da između dva tipa entiteta (E_1 i E_2) u modelu važi odnos više - prema - više, što se označava i sa $M:N$. Vrednosti minimalnih kardinaliteta definišu sledeće tri slučaja ovog odnosa. To su:

- Slučaj kod kojeg je $a_1 = 0$ i $a_2 = 0$, kada minimalni kardinaliteti ukazuju da u obe klase mogu postojati entiteti, koji nisu povezani sa bilo kojim entitetom druge klase. Međutim, i slučaj kada su svi entiteti posmatranih klasa međusobno povezani, nije zabranjen.
- Slučaj kod kojeg je $a_1 = 1$ i $a_2 = 0$ ili $a_1 = 0$ i $a_2 = 1$, kada minimalni kardinaliteti ukazuju da svi entiteti jedne od povezanih klasa (recimo klase E_1) moraju biti povezani sa bar jednim entitetom druge klase (klase E_2). To dalje znači da je egzistencija entiteta u jednoj klasi (E_1) uslovljena njihovom povezanošću sa entitetima u drugoj klasi(E_2).
- Slučaj kod kojeg je $a_1 = 1$ i $a_2 = 1$, kada minimalni kardinaliteti ukazuju da svaki entitet jedne klase mora biti povezan sa bar jednim entitetom druge klase. Egzistencija entiteta u obe klase uslovljena je njihovom povezanošću sa bar jednim entitetom druge klase, što predstavlja izuzetno strogo ograničenje.

Primer 2.26. U svojstvu primera odnosa $R(E_1(a_2, M) : E_2(a_1, N))$ mogu se posmatrati tipovi entiteta *Radnik*, *Projekat* i tip poveznika *Radi*. Maksimalni kardinaliteti tipa

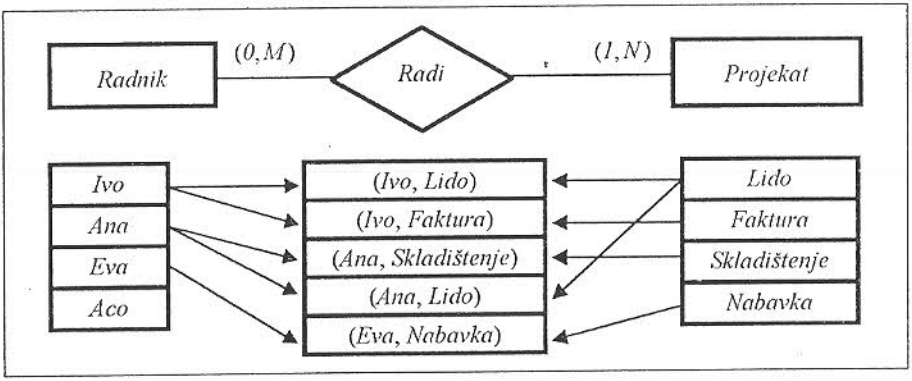
poveznika ukazuju da jedan radnik može raditi na više projekata i da na svakom projektu može biti angažovano više radnika.

Ako je $a_1 = 0$ i $a_2 = 0$, to znači da ne mora svaki radnik raditi na bar jednom projektu i da mogu postojati projekti na kojima još, ili više, ne radi ni jedan radnik. Na slici 2.10 je prikazana ekstenzija koja zadovoljava ova ograničenja.



Slika 2.10.

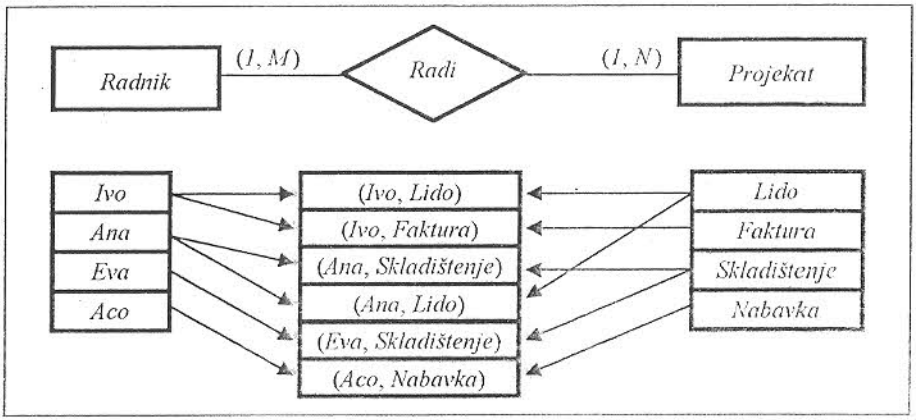
Ako je $a_1 = 1$ i $a_2 = 0$, reč je o realnom sistemu u kojem ne mora svaki radnik raditi na bar jednom projektu, ali na svakom projektu mora biti angažovan bar jedan radnik. To znači da se novi projekat ne može evidentirati, ako nije određen bar jedan radnik da na njemu radi. Takođe, onog trenutka kada prestane angažman poslednjeg radnika na projektu, gasi se i projekat. Na slici 2.11 prikazana je ekstenzija koja zadovoljava ova ograničenja. Ekstenzija na slici 2.11 zadovoljava i ograničenja $a_1 = 0$ i $a_2 = 0$.



Slika 2.11.

Ako je $a_1 = 1$ i $a_2 = 1$, reč je o realnom sistemu u kojem svaki radnik mora raditi na bar jednom projektu i na svakom projektu mora biti angažovan bar jedan radnik. Ek-

stenzija na slici 2.12 zadovoljava ovo ograničenje ali i ograničenja $a_1 = 0$, $a_2 = 0$ i $a_1 = 1$, $a_2 = 0$. □



Slika 2.12.

2.3.2. *Strukture sa kardinalitetima grupe N : 1*

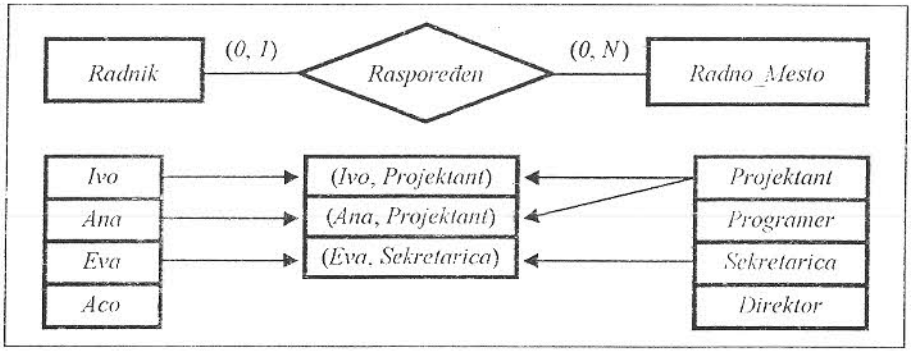
Slučaj $b_1 = N$ i $b_2 = 1$ opisuje situaciju u realnom sitemu kada jedan entitet prve klase može biti povezan sa jednim entitetom druge klase i kada jedan entitet druge klase može biti povezan sa više entiteta prve klase. Ova situacija se naziva odnosom više - prema - jedan i označava sa $N:1$. U zavisnosti od vrednosti parametara a_1 i a_2, razlikuju se četiri slučaja ovog odnosa:

- U prvom slučaju, kardinalitet tipa poveznika $R(E_1(0, 1):E_2(0, N))$ opisuje odnos između dve realne klase entiteta u kojem svaki entitet klase E_1 može biti u vezi sa najviše jednim entitetom klase E_2, dok svaki entitet klase E_2 može biti povezan sa više entiteta klase E_1, ali ne mora svaki entitet klase E_2 biti povezan sa bar jednim entitetom klase E_1.
- U odnosu na prethodni slučaj, tip poveznika sa kardinalitetom $R(E_1(1, 1):E_2(0, N))$ dovodi do određene izmene u opisu realnog sistema. Izmena se odnosi na činjenicu da sada svaki entitet prve klase mora biti povezan sa jednim i samo jednim entitetom druge klase.
- Treći slučaj kardinaliteta grupe $N:1$ se dobija kada je $a_1 = 1$ i $a_2 = 0$. Sada su entiteti druge klase egzistencijalno zavisni od entiteta prve klase, tako da je svaki entitet prve klase povezan sa najviše jednim entitetom druge klase, dok svaki entitet druge klase mora biti povezan sa bar jednim entitetom prve klase.
- Četvrtu varijantu kardinaliteta grupe $N:1$ predstavlja slučaj kada su entiteti obe povezane klase međusobno egzistencijalno uslovljeni. Notacija za odgovarajući kardinalitet tipa poveznika je $R(E_1(1, 1):E_2(1, N))$.

Primer 2.27. Kao ilustracija odnosa $R(E_1(0, 1):E_2(0, N))$ može poslužiti odnos između tipova entiteta *Radnik* i *Radno_Mesto* opisan tipom poveznika *Raspoređen*. Kardinalitet tipa poveznika

$$Raspoređen(Radnik(0, 1):Radno\_Mesto(0, N))$$

opisuje realnu situaciju u kojoj svaki radnik može biti raspoređen na jedno radno mesto, ali ne mora svaki radnik biti raspoređen. Naravno, na jedno radno mesto može biti raspoređeno više radnika, a mogu postojati i radna mesta bez raspoređenih radnika, slika 2.13.

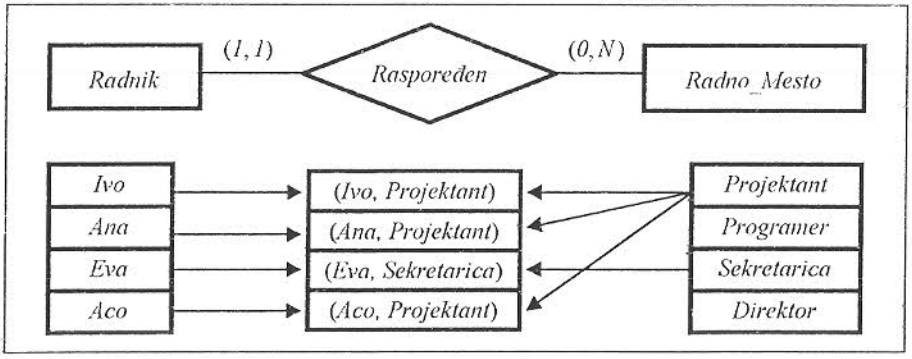


Slika 2.13.

Kardinalitet tipa poveznika

$$Raspoređen(Radnik(1, 1):Radno\_Mesto(0, N))$$

ukazuje da svaki radnik mora biti raspoređen na jedno i samo jedno radno mesto i da ne može postojati neraspoređen radnik, slika 2.14.

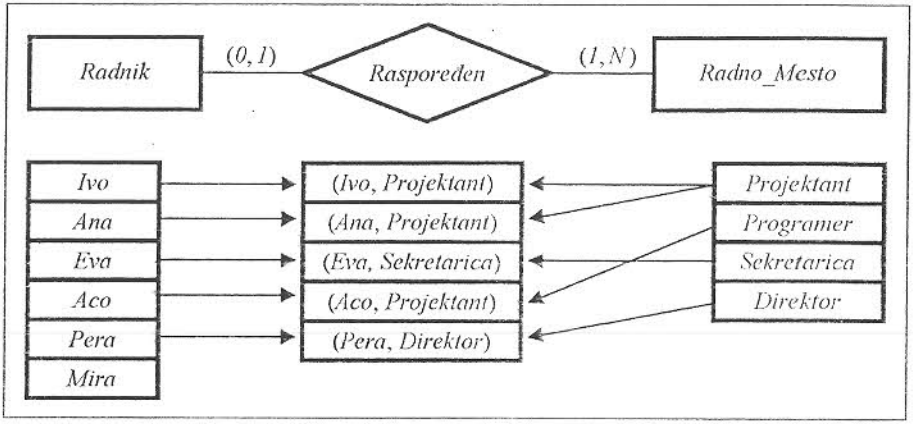


Slika 2.14.

Kardinalitet tipa poveznika

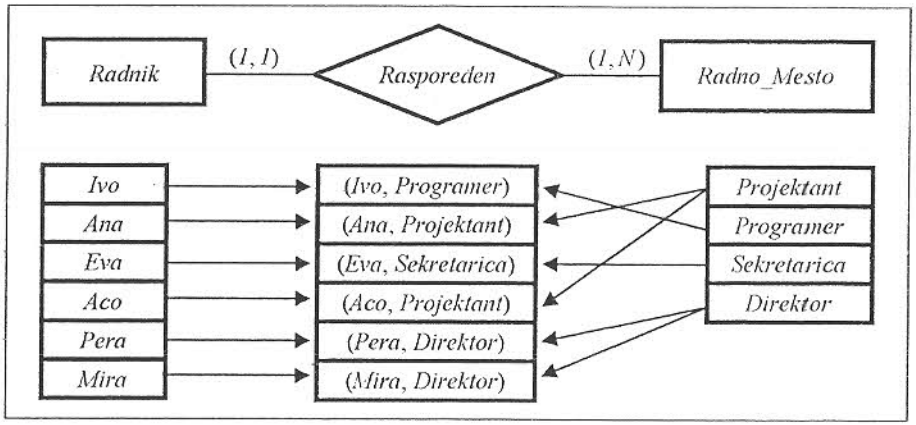
$$Raspoređen(Radnik(0, 1) : Radno\_Mesto(1, N))$$

ukazuje da mogu postojati neraspoređeni radnici, ali da ne može postojati radno mesto, na koje nije raspoređen bar jedan radnik, slika 2.15.



Slika 2.15.

U slučaju tipova entiteta *Radnik*, *Radno_Mesto* i tipa poveznika *Raspoređen*, kardinalitet $R(E_1(1, 1) : E_2(1, N))$ ukazuje da svaki radnik mora biti raspoređen na jedno i samo jedno radno mesto i da na svako radno mesto mora biti raspoređen bar jedan radnik, slika 2.16. □



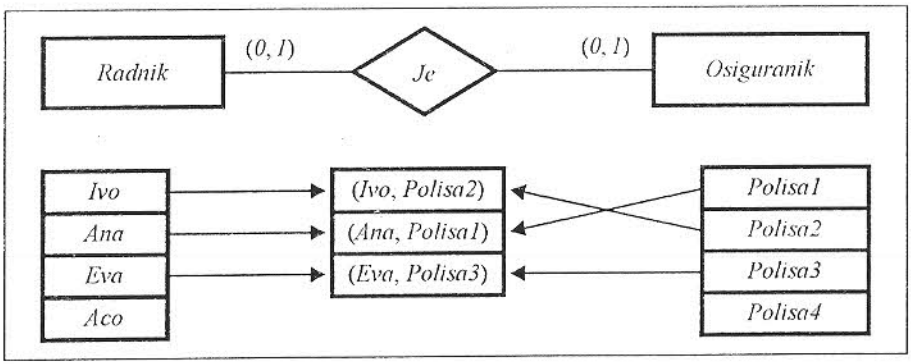
Slika 2.16.

2.3.3. *Strukture sa kardinalitetima grupe 1 : 1*

Slučaj $b_1 = 1$ i $b_2 = 1$ opisuje situaciju u realnom sistemu kada jedan entitet prve klase može biti povezan sa jednim entitetom druge klase, a jedan entitet druge klase sa jednim entitetom prve klase. Ova situacija se naziva jedan - prema - jedan i označava sa $1 : 1$. U zavisnosti od vrednosti a_1 i a_2, razlikuju se sledeća tri slučaja:

- Tip poveznika sa kardinalitetom $R(E_1(0, 1) : E_2(0, 1))$ opisuje odnos između klasa entiteta E_1 i E_2, gde svaki entitet jedne klase može biti povezan sa najviše jednim entitetom druge klase.
- U odnosu na prethodni slučaj, tip poveznika sa kardinalitetom $R(E_1(1, 1) : E_2(0, 1))$ uvodi ograničenje da svaki entitet klase E_1 mora biti povezan sa jednim i samo jednim entitetom klase E_2.
- Na kraju, tip poveznika sa kardinalitetom $R(E_1(1, 1) : E_2(1, 1))$ opisuje situaciju kada je svaki entitet jedne klase povezan sa jednim i samo jednim entitetom druge klase, što ukazuje i na njihovu uzajamnu egzistencijalnu povezanost.

Primer 2.28. U svojstvu primera odnosa $R(E_1(0, 1) : E_2(0, 1))$ mogu se posmatrati tipovi entiteta *Radnik*, *Osiguranik* i tip poveznika *Je* u nekom osiguravajućem društvu. Kardinalitet tipa poveznika *Je* (*Radnik* (0, 1) : *Osiguranik* (0, 1)), opisuje realni sitem gde radnik može, a ne mora biti osiguranik preduzeća u kojem radi i gde osiguranik može biti radnik posmatranog osiguravajućeg društva, ali ne mora svaki osiguranik biti radnik, slika 2.17.

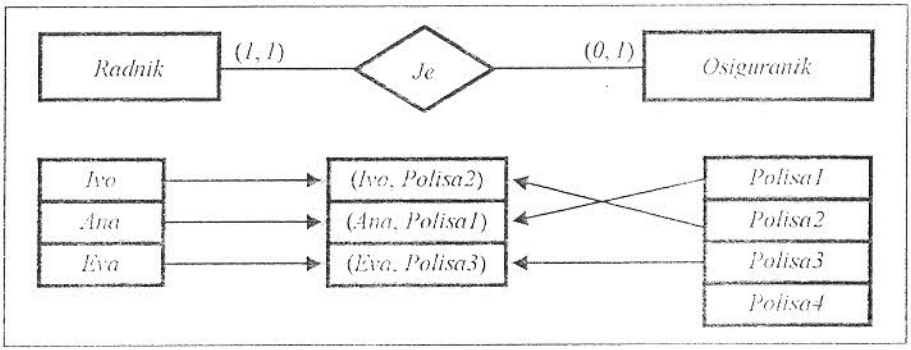


Slika 2.17.

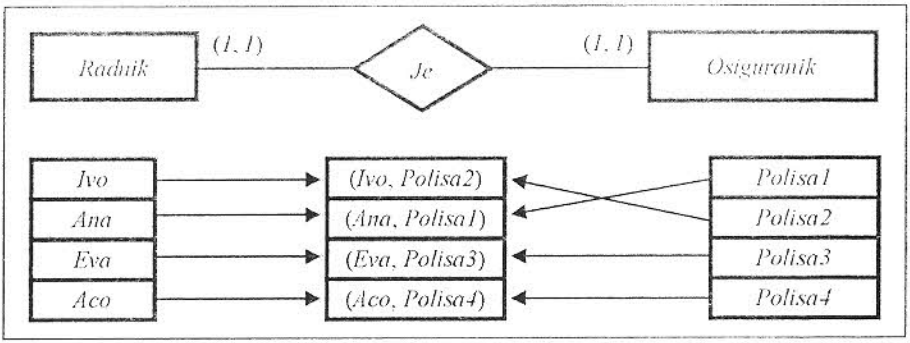
U slučaju primera sa tipovima entiteta *Radnik*, *Osiguranik* i tipom poveznika *Je*, kardinalitet *Je* (*Radnik* (1, 1) : *Osiguranik* (0, 1)) bi značio da, u posmatranom osiguravajućem društvu važi pravilo poslovanja prema kojem svaki radnik mora biti osiguranik, ali da osiguranici mogu biti i ljudi van posmatranog kolektiva, slika 2.18.

Posmatrajući ponovo deo modela osiguravajućeg društva sa tipovima entiteta *Radnik* i *Osiguranik*, kardinalitet tipa poveznika *Je* bi bio *Je* (*Radnik* (1, 1) : *Osiguranik* (1, 1)), što bi, u ovom slučaju opisivalo, verovatno apsurdnu situaciju, prema kojoj

osiguravajuće društvo osigurava samo svoje radnike. Međutim, treba napomenuti da se u praksi, istina retko, mogu sresti i slučajevi kada je primena tipa poveznika sa kardinalitetom $R(E_1(1, 1):E_2(1, 1))$ neophodna za veran opis realnog sistema, slika 2.19. □



Slika 2.18.



Slika 2.19.

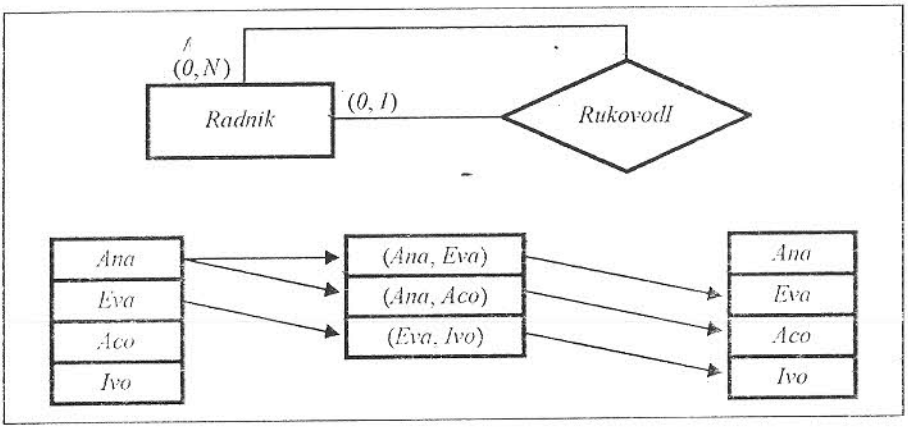
2.3.4. *Rekurzivne veze*

Rekurzivni tip poveznika predstavlja model relacije u jednom skupu, relacije koja povezuje entitete jedne klase. U ER strukturi, isti tip entiteta reprezentuje i prvu i drugu klasu realnih entiteta, jer jedna od tih klasa, po pravilu, predstavlja podskup druge. Pri tome, entiteti posmatrane klase igraju različite uloge u svom odnosu. Bez obzira na prividnu neobičnost ove strukture, ona služi za modeliranje veoma čestih situacija u realnim sistemima. Kardinaliteti ovog tipa poveznika može uzimati sve moguće vrenosti.

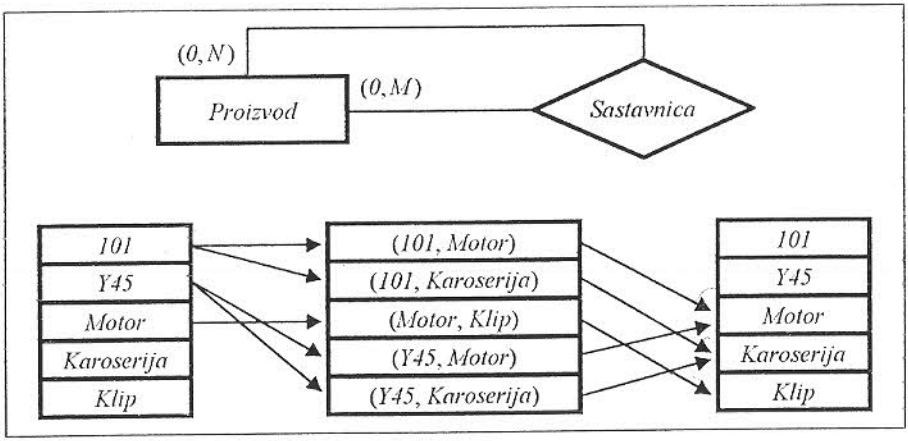
Primer 2.29. Posmatra se tip entiteta *Radnik* i rekurzivni tip poveznika *Rukovodi* sa kardinalitetima

(*Radnik* (podređeni)(*0, 1*) : *Radnik* (rukovodi)(*1, N*)).

prema slici 2.20. Semantika ove strukture je: jedan rukovodni radnik rukovodi sa najmanje jednim podređenim radnikom i svaki radnik ima najviše jednog direktnog rukovodioca. Fraza "najviše jedan" ukazuje da mogu postojati i radnici koji nemaju pretpostavljenih, što, u najmanju ruku, važi za direktora, kako je to na slici 2.20 i prikazano.



Slika 2.20.



Slika 2.21.

Putem rekurzivnog tipa entiteta se gradi i model sastavnice. Na slici 2.21 je prikazan tip entiteta *Proizvod* i tip poveznika *Sastavnica*. Pošto i sklopovi i delovi predstavljaju proizvode, pojave tipa entiteta *Proizvod* sadrže podatke o: proizvodima, sklopovima i

delovima. Tip poveznika *Sastavnica* povezuje svaki proizvod sa svojim direktnim komponentama, kako to slika 2.21 i prikazuje. □

2.3.5. *Tip poveznika reda većeg od dva*

Kao što je već rečeno, tip poveznika može da poveže i više od dva tipa entiteta. Kardinalitet tipa poveznika reda $n(>2)$ se određuje putem postupka, koji predstavlja uopštenje postupka za određivanje kardinaliteta tipa poveznika reda $n = 2$. Ponovo, kao i kod tipova poveznika reda dva, postoje dva mogućna pristupa. Prema prvom, postupak se ponavlja n puta, za svaki od povezanih tipova entiteta jedanput. U postupku, posmatra se jedan poveznik $(e_1, e_2, \ldots, e_{n-1})$ reda $n - 1$, gde je $e_i \in E_i$, i skup entiteta E_n, čiji entiteti nisu uključeni u poveznik reda $n - 1$. Zatim se određuje kardinalitet preslikavanja

$$R_{1,\ldots,n-1}(E_n(a_n, b_n)).$$

Pri tome se identifikuju mogući kardinaliteti podskupova skupa E_n, povezanih sa posmatranim poveznikom $(e_1, e_2, \ldots, e_{n-1})$.

Prema drugom pristupu, postupak se ponovo ponavlja n puta, za svaki od povezanih tipova entiteta jedanput. Pri tome, analizira se skup poveznika $P = \{(e_1, e_2, \ldots, e_n) \mid e_i \in E_i\}$ posmatranog tipa i, za proizvoljan entitet $e_i \in E_i$, utvđuje se u koliko n - torki skupa P se e_i može minimalno i maksimalno javiti kao komponenta. Na taj način se određuje broj mogućih pojavljivanja proizvoljnog entiteta $e_i \in E_i$ u poveznicima skupa P.

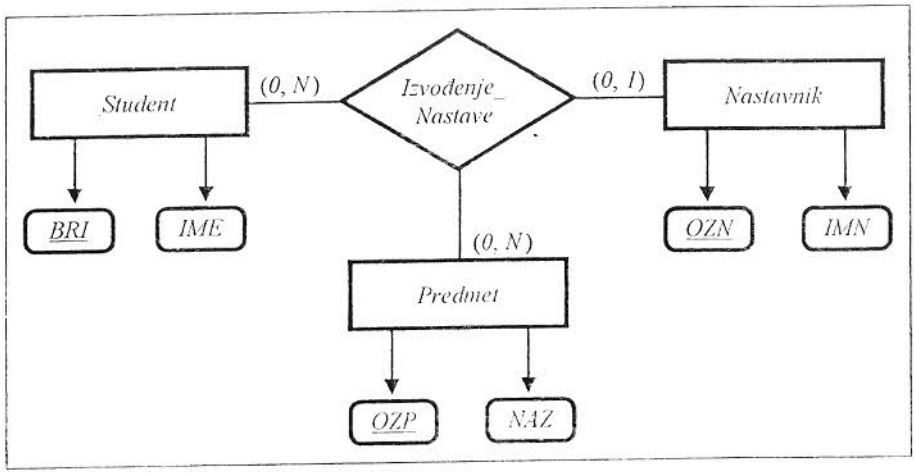
Primer 2.30. Posmatraju se tipovi entiteta *Student*, *Nastavnik* i *Predmet*. Između entiteta odgovarajućih klasa u realnom sistemu važe sledeći odnosi:

1. student s sluša predmet p kod najviše jednog nastavnika n,
2. ako nastavnik n izvodi predmet p, izvodi ga za više studenata, mada ne mora ni za jednog,
3. student s, kod nastavnika n može slušati više predmeta, ali ne mora ni jedan,
4. postoje nastavnici koji ne predaju ni jedan predmet bilo kom studentu,
5. postoje studenti koji ne slušaju ni jedan predmet kod bilo kog nastavnika,
6. ne postoje predmeti, koje ne predaje ni jedan nastavnik bilo kom studentu i
7. jedan nastavnik može predavati više predmeta za više studenata.

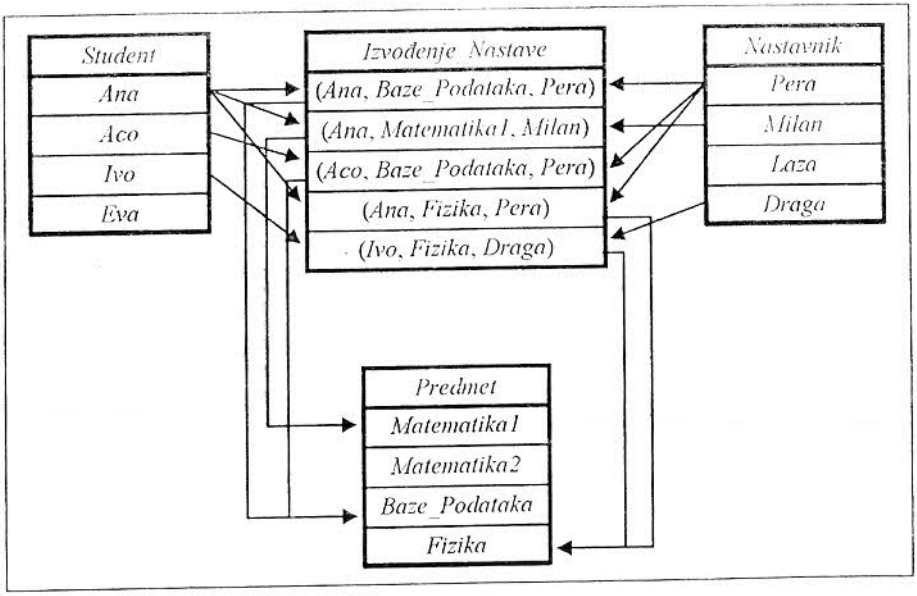
Prema prvom pristupu, za definisanje kardinaliteta tipa poveznika *Izvođenje_Nastave* reda 3, koji povezuje tipove entiteta *Student*, *Predmet* i *Nastavnik*, bitni su odnosi 1., 2. i 3. Navedeni odnosi ukazuju da tip poveznika *Izvođenje_Nastave* poseduje, redom, sledeće kardinalitete:

$R_{sp}(Nastavnik(0, 1))$, odnosno $((s, p) : n = (0, 1)$,
$R_{np}(Student(0, N))$, odnosno $((n, p) : s = (0, 1)$,
$R_{sn}(Predmet(0, N))$, odnosno $((s, n) : p = (0, N)$.

Na slici 2.22 je prikazan ER dijagram opisanog dela realnog sistema, a na slici 2.23 je data odgovarajuća ekstenzija.



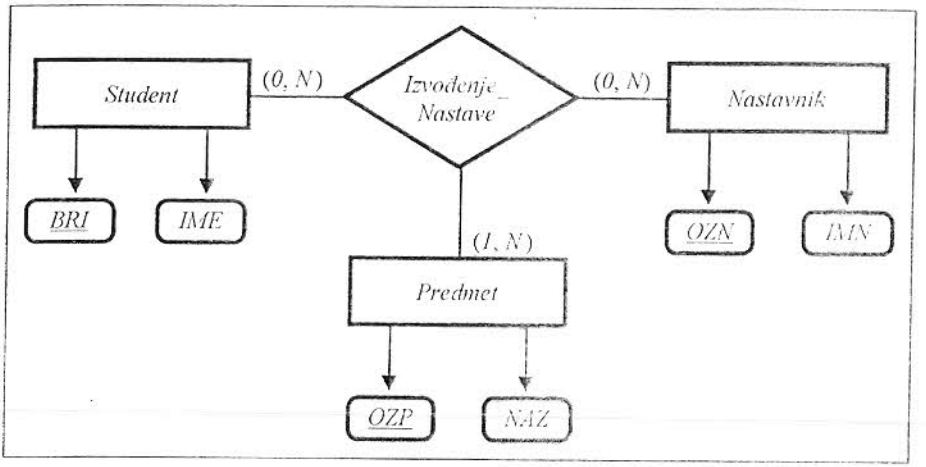
Slika 2.22.



Slika 2.23.

Za definisanje kardinaliteta tipa poveznika *Izvođenje_Nastave*, prema drugom pristupu, odnos 1. je nebitan, a bitni su svi ostali navedeni odnosi. Saglasno tim odnosima, nacrtan je ER dijagram, na slici 2.24. Ilustracije radi, prema odnosu 7., jedan nastavnik se

može javiti kao komponenta više poveznika, a prema odnosu 4., postoje nastavnici, koji se ne javljaju ni jedanput kao komponenta poveznika. □



Slika 2.24.

Uspoređujući ER dijagrame sa slika 2.22 i 2.24, zapažaju se različiti kardinaliteti tipa poveznika *Izvođenje_Nastave*, mada je ekstenzija sa slike 2.23, u oba slučaja, ista. Te razlike potiču od činjenice da:

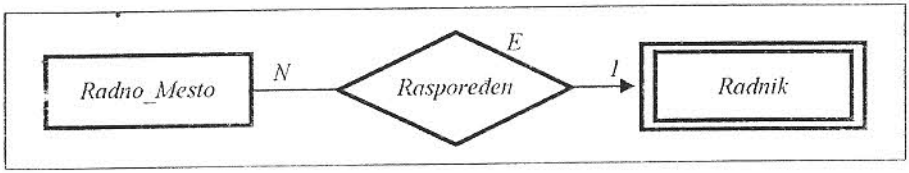
- prvi pristup identifikuje koliko je entiteta skupa E_n povezano sa jednim poveznikom $(e_1, e_2, \ldots, e_{n-1})$, a
- drugi pristup identifikuje koliko puta se neki entitet skupa E_n javlja kao komponenta u skupu poveznika P.

2.3.6. *Slabi tip entiteta*

Kardinalitet tipa poveznika predstavlja jedan od primera eksplicitnih ograničenja, koja se zadaju u modelu entiteta i poveznika. Često se, u ER dijagramima, zadaju samo maksimalne vrednosti kardinaliteta, tada se koristi poseban postupak zadavanja egzistencijalne zavisnosti, odnosno ograničenje tipa $R(E_1(a_2, b_2) : E_2(1, b_1))$, koje ukazuje da u modelu, na nivou ekstenzije, za svako e_2 iz E_2 postoji bar jedno e_1 iz E_1 takvo da su e_1 i e_2 međusobno povezani relacijom R. Taj poseban postupak zadavanja egzistencijalne zavisnosti rešava se uvođenjem pojma slabog tipa entiteta i posebnog načina označavanja takvih tipova entiteta. Dijagramski način označavanja slabog tipa entiteta se vrši upisivanjem njegovog naziva u dvostruki pravougaonik uz navođenje velikog slova E za egzistencijalnu zavisnost. Takođe, poteg koji povezuje geometrijsku sliku za slabi tip entiteta sa geometrijskom slikom za odgovarajući tip poveznika se orijentiše ka slabom tipu entiteta. I tip poveznika, koji povezuje neki slabi tip entiteta, naziva se slabim tipom poveznika. Egzis-

tencijalno nezavisni tip entiteta i tip poveznika koji povezuje samo egzistencijalno nezavisne tipove entiteta, nazivaju se regularnim.

Primer 2.31. Na slici 2.25 su nacrtani tipovi entiteta *Radno_Mesto*, *Radnik* i tip poveznika *Raspoređen*. Tip entiteta *Radnik* je slab. Činjenica da je tip entiteta *Radnik* označen kao slabi tip entiteta, interpretira se na sledeći način: ne može postojati radnik (u posmatranoj organizaciji), a da nije raspoređen na neko radno mesto. □



Slika 2.25.

Ključ tipa entiteta, takođe, predstavlja jedno od ograničenja, ali se to ograničenje ne mora definisati za svaki tip entiteta u modelu entiteta i poveznika. Tip entiteta, koji ne poseduje ključ, takođe se naziva slabim, a putem takvog tipa entiteta se definiše ograničenje pod nazivom ***identifikaciona zavisnost***. Entiteti takvog skupa se ne mogu identifikovati putem vrednosti nekog svog obeležja, već putem svoje povezanosti sa entitetima nekih drugih skupova. Treba napomenuti da identifikaciona zavisnost automatski povlači za sobom i egzistencijalnu zavisnost, ali da egzistencijalno zavisni tip entiteta ne mora biti i identifikaciono zavistan.

Identifikaciono zavisni tip entiteta se u dijagramima predstavlja veoma slično kao egzistencijalno zavisni tip entiteta. Jedina je razlika da se umesto oznake E navodi oznaka ID.

Ekstenzija slabog tipa entiteta se naziva slabom relacijom entiteta. Tabelarni prikaz ekstenzije slabog tipa entiteta sadrži i obeležje I, čije vrednosti, zajedno sa vrednostima nekog podskupa V obeležja slabog tipa entiteta, jednoznačno određuju pojave slabog tipa entiteta. Pošto slabi tip entiteta nema ključ, vrednosti skupa obeležja $I \cup V$ u ekstenziji igraju ulogu vrednosti ključa.

Neka je W slabi tip entiteta sa skupom obeležja $\{A_1,\ldots,A_w\}$, a R jedini regularni tip entiteta sa kojim je W povezan, tako da između njih postoji samo jedan slabi tip poveznika. Ako je K ključ tipa entiteta R, tada za obeležje I, čije vrednosti identifikuju pojave slabog tipa entiteta W, važi

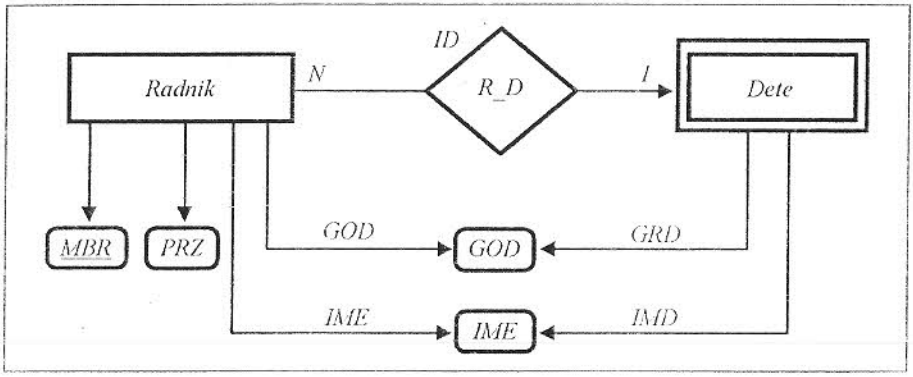
$$K \subset I \text{ i } I \setminus K \subseteq \{A_1,\ldots,A_w\}.$$

Neka su $R_1,\ldots,R_k$ regularni tipovi entiteta sa ključevima $K_1,\ldots,K_k$, respektivno i neka je W sa njima povezan bilo direktno ili posredstvom drugih slabih tipova entiteta. Neka je I_i unija identifikacionih obeležja slabih tipova entiteta između R_i i W, za $i = 1,\ldots,k$. (I_i može biti i prazan skup). Tada važi

$$\bigcup_{i=1}^{k} K_i I_i \subset I \text{ i } I \setminus \bigcup_{i=1}^{k} K_i I_i \subseteq \{A_1,\ldots,A_v\}.$$

Slabi tip poveznika nema odgovarajuću relaciju.

Primer 2.32. Na slici 2.26 su nacrtana dva tipa entiteta *Radnik*, *Dete* i tip poveznika *R_D*. Tip entiteta *Dete* je identifikaciono zavistan. Oznaka za identifikacionu zavisnost je upisana u geometrijsku sliku tipa poveznika *R_D*.



Slika 2.26.

Na slici 2.27. prikazana je regularna relacija entiteta *Radnik* i slaba relacija entiteta *Dete*. Torke slabe relacije sadrže vrednosti obeležja *MBR* kao svoje komponente, tako da je $I = \{MBR, IMD\}$. Pojave slabog, identifikaciono zavisnog tipa entiteta *Dete* određene su matičnim brojem radnika svog roditelja i svojim imenom. □

Radnik

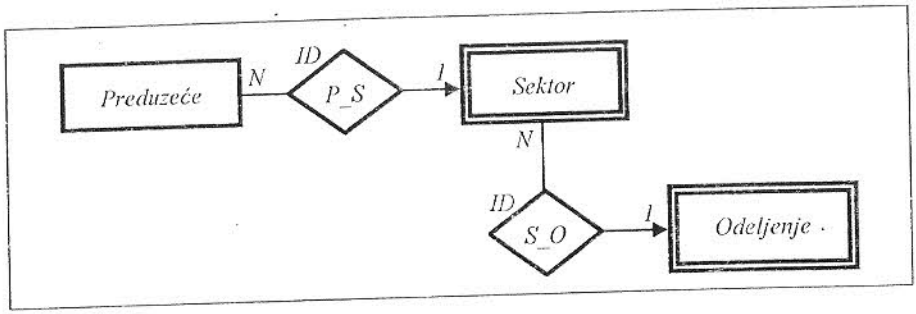
MBR	*IME*	*PRZ*	*GOD*
88	*Eva*	*Pap*	*1950*
50	*Ivo*	*Ban*	*1950*
01	*Ana*	*Ras*	*1961*
81	*Ana*	*Tot*	*1963*

Dete

MBR	*IMD*	*GRD*
88	*Goran*	*1975*
88	*Ivana*	*1970*
01	*Olga*	*1972*
50	*Ana*	*1972*

Slika 2.27.

Primer 2.33. Na slici 2.28 je prikazan dijagram entiteta i poveznika, koji opisuje strukturu dela radne organizacije. Reč je o hijerarhiji slabih tipova entiteta sa regularnim tipom entiteta u korenu stabla. Konkretna odeljenja se identifikuju putem ključa tipa entiteta *Preduzeće*, rednog broja sektora u preduzeću i rednog broja odeljenja u sektoru. Konkretan sektor se identifikuje putem ključa tipa entiteta *Preduzeće* i rednog broja sektora u preduzeću. □



Slika 2.28.

2.4. Operacijska komponenta

P.Chen je u svom radu [Che] predložio primenu jezika, sličnog jeziku SQL relacionog modela podataka, za ažuriranje i postavljanje upita u bazu podataka zasnovanu na modelu entiteta i poveznika. Međutim, model entiteta i poveznika je stekao široku popularnost u praksi kao osnova za razvoj metoda logičkog projektovanja baze podataka, ali ne i kao model na kojem bi bio zasnovan neki šire korišćen SUBP. Iz tih razloga operacijskoj komponenti modela entiteta i poveznika, u ovom tekstu, neće biti poklanjana dalja pažnja.

2.5. Proširenja modela entiteta i poveznika

Tokom kraja sedamdesetih godina, razni autori, među kojima se ističu Smith i Smith [SS], Hammer i Mc Load [HM], uveli su određeni broj novih koncepata i novih postupaka za definisanje složenijih koncepata u ER model podataka. Osnovni cilj uvođenja tih inovacija je bio obezbeđenje veće moći izražavanja semantike pri izgradnji modela realnog sistema. U nove koncepte spadaju: potklasa, superklasa, gerund i kategorija, a u postupke: specijalizacija, generalizacija, kategorizacija, agregacija i asocijacija. Svi ovi postupci nazivaju se apstrakcijama. Za navedene pojmove tesno je vezan i važan mehanizam nasleđivanja obeležja (osobina).

2.5.1. Potklasa i superklasa

Tip entiteta se u ER modelu koristi za predstavljanje skupa sličnih realnih entiteta. U skupu realnih entiteta se često mogu identifikovati pravi podskupovi entiteta sa specifičnim osobinama ili ulogama, te se nameće potreba njihovog eksplicitnog predstav-

ljanja i u modelu realnog sistema. Predstavljanje podskupova entiteta sa specifičnim osobinama i ulogama se postiže putem koncepta superklasa/potklasa, na sledeći način:

- zajednička obeležja svih entiteta se grupišu u superklasu, a
- specifična obeležja, shodno različitim ulogama entiteta, grupišu se u odgovarajuće potklase.

Pri tome, *specifično* obeležje je ono, koje samo za pravi potskup posmatranog skupa entiteta predstavlja karakterističnu osobinu, a za ostale entitete je neprimereno svojstvo.

Superklasa i njene potklase povezuje relacija, koja se često naziva *IS_A* hijerarhijom. Naziv potiče od engleskih reči "is a", u smislu entitet potklase je (is a) i entitet superklase. Kriterijum za definisanje potklasa predstavlja klasifikaciono obeležje, koje pripada superklasi. Svaka potklasa odgovara jednom elementu domena klasifikacionog obeležja, ali ne mora svakom elementu domena klasifikacionog obeležja odgovarati jedna potklasa.

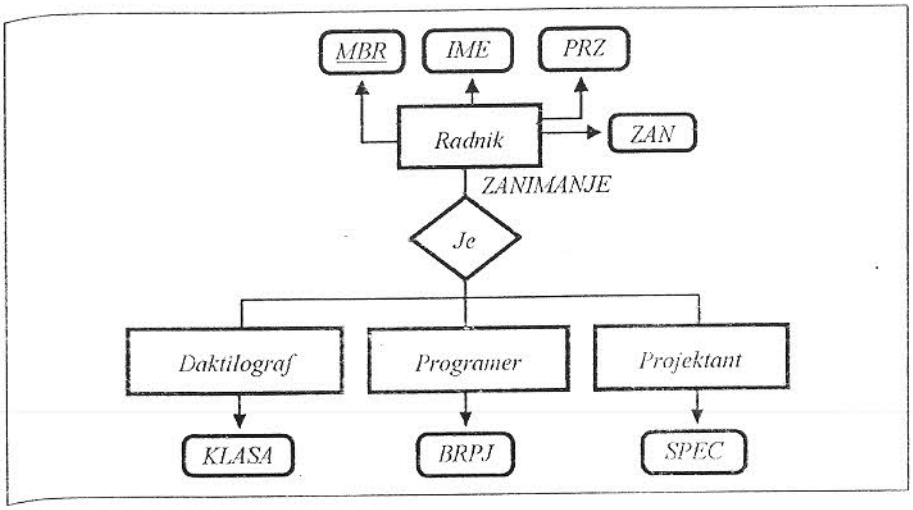
2.5.1.1. Nasleđivanje osobina

Superklasa prestavlja intenzionalni model svih entiteta jedne klase, a potklasa ukazuje na posebnu ulogu entiteta određenog podskupa posmatrane klase. Pojava potklase sadrži samo vrednost primarnog ključa superklase i vrednosti specifičnih obeležja. Saglasno tome, na nivou ekstenzije, svakoj pojavi tipa entiteta potklase odgovara tačno jedna pojava tipa entiteta superklase. Tako se dolazi do pojma mehanizma nasleđivanja osobina odnosno obeležja. Potklasa nasleđuje sva obeležja svoje superklase i ne može se posmatrati izdvojeno, bez svoje superklase. Mehanizam tog nasleđivanja je posredan. Realizuje se putem iste vrednosti ključa pojave superklase i pojave potklase. Tek nakon povezivanja neke pojave potklase sa odgovarajućom pojavom superklase, dobija se ekstenzionalni model konkretnog entiteta iz posmatranog podskupa realnih entiteta sa nekom specifičnom ulogom.

Primer 2.34. Na slici 2.29 je prikazana jedna od mogućih geometrijskih reprezentacija *IS_A* hijerarhije sa superklasom *Radnik* i poktlasama *Daktilograf*, *Projektant* i *Programer*. Potklasama odgovaraju, redom, sledeća specifična obeležja: *KLASA* (daktilografska klasa), *SPEC* (projektantska specijalnost) i *BRPJ* (broj programskih jezika). Obeležje *ZAN* (zanimanje) sa domenom $dom(ZAN) = \{daktilograf, pravnik, programer, projektant, \ldots\}$ je upotrebljeno za klasifikaciju, što je i navedeno uz poteg od superklase ka potklasama. Činjenica da nije definisana potklasa *Pravnik*, ne ukazuje da ne postoje radnici tog zanimanja, već da ta potklasa nije bitna sa tačke gledišta zadataka automatizovanog informacionog sistema. Semantika dijagrama sa slike je da su daktilografi, projektanti i programeri radnici sa specifičnim osobinama, bitnim za realizaciju zadataka automatizovanog informacionog sistema. Na nivou ekstenzije, svakom daktilografu, projektantu ili programeru odgovaraju dva modela. Jedan, kao pojava potklase i jedan kao pojava superklase. □

IS_A hijerarhije se koriste samo za predstavljanje situacije kada pravi podskupovi skupa entiteta igraju posebne uloge u realnom sistemu. Ako svi entiteti posmatranog skupa imaju više uloga, takva situacija se može rešiti uvođenjem ekvivalentnih ključeva u

odgovarajući tip entiteta. Svaki od ekvivalentnih ključeva nosi informaciju o jednoj od uloga svih entiteta posmatranog skupa.



Slika 2.29.

Primer 2.35. Svi zaposleni u preduzeću imaju bar dve uloge. To su: uloga radnika i uloga socijalnog osiguranika. Da bi se ukazalo na te dve uloge, u tip entiteta *Radnik* se uvode dva ključa. To su: *MBR* (matični broj radnika) i *IDSO* (identifikacioni broj socijalnog osiguranika). □

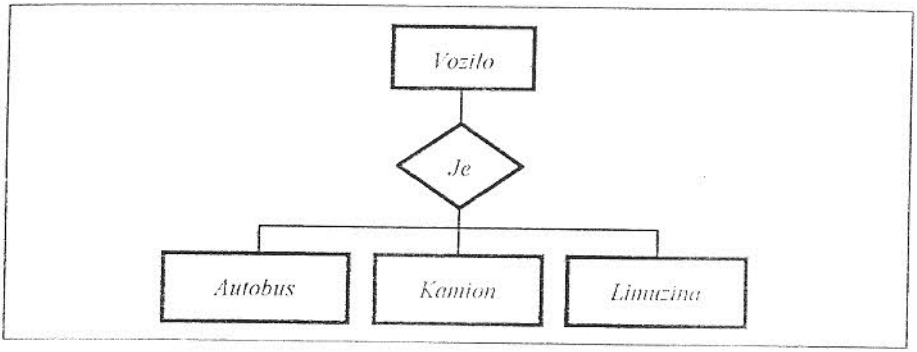
2.5.1.2. *Specijalizacija*

IS_A hijerarhije se definišu primenom ili specijalizacije ili generalizacije. Kod specijalizacije se kreće od tipa entiteta, buduće superklase, iz kojeg se, saglasno klasifikacionom obeležju, izdvajaju potklase sa specifičnim obeležjima. Isti polazni tip entiteta se može podvrgnuti specijalizacijama na osnovu vrednosti više klasifikacionih obeležja. Takođe, svaka potklasa može predstavljati superklasu za neke nove potklase.

2.5.1.3. *Generalizacija*

Generalizacija predstavlja proces suprotan specijalizaciji. Generalizacijom se od različitih tipova entiteta, zanemarivanjem razlika i identifikacijom zajedničkih osobina, gradi zajednička superklasa. Polazni tipovi entiteta postaju potklase dobijene superklase. Potklase zadržavaju samo specifična obeležja. Specijalizacija i generalizacija, očigledno, predstavljaju međusobno inverzne procese, koji daju isti rezultat.

Primer 2.36. Na slici 2.30 su nacrtani tipovi entiteta *Autobus*, *Kamion* i *Limuzina*. Njihovom generalizacijom se dobija superklasa *Vozilo*. □



Slika 2.30.

2.5.1.4. Karakteristike ekstenzije IS_A hijerarhija

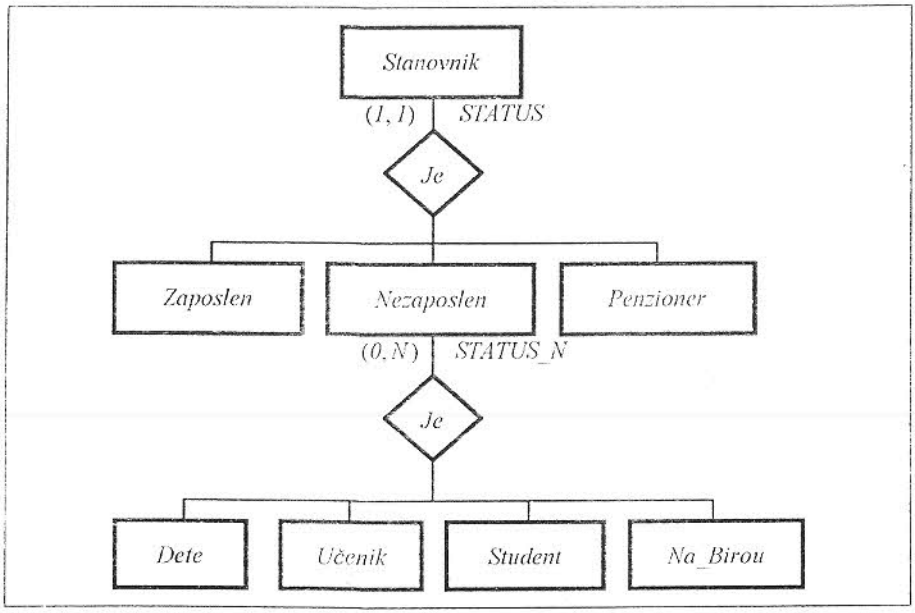
U ekstenziji, svaka pojava potklase nasleđuje vrednost ključa od odgovarajuće (svoje) pojave u superklasi, jer svakoj pojavi potklase odgovara tačno jedna pojava u superklasi.

Kardinalnost odnosa između skupa potklasa i superklase i disjunktnost ekstenzije predstavljaju bitne karakteristike *IS_A* hijerarhija. Kardinalnost preslikavanja sa skupa pojava potklasa na skup pojava superklase je uvek (*1*, *1*), jer svakoj pojavi bilo koje potklase odgovara jedna i samo jedana pojava superklase. Zbog toga se taj kardinalitet na ER dijagramima i ne navodi. Preslikavanje sa skupa pojava superklase na skup pojava potklasa može biti ili ***totalno*** ili ***parcijalno***. Ako svakoj pojavi superklase odgovara bar jedna pojava neke od potklasa, *IS_A* hijerarhija se naziva totalnom, a minimalni kardinalitet tog preslikavanja je $a = 1$. Inače, ako bar jednoj pojavi superklase ne odgovara nijedna pojava bilo koje potklase, *IS_A* hijerarhija se naziva parcijalnom, a minimalni kardinalitet tog preslikavanja je $a = 0$.

Disjunktnost ekstenzije govori o broju pojava različitih potklasa, koje odgovaraju jednoj pojavi superklase. Ako je svakoj pojavi superklase pridružena pojava iz najviše jedne potklase, *IS_A hijerarhija je **disjunktna***, a maksimalni kardinalitet preslikavanja sa superklase u potklase je $b = 1$. Inače, ako bar jednoj pojavi superklase odgovaraju pojave iz više od jedne potklase, *IS_A* hijerarhija je ***presečna***, a maksimalni kardinalitet preslikavanja sa skupa pojava superklase na skup pojava potklasa je $b = N$.

Primer 2.37. Na slici 2.31 je prikazana trinivovska *IS_A* hijerarhija sa superklasom *Stanovnik*. Potklase ove superklase su: *Zaposlen*, *Nezaposlen* i *Penzioner*. Ove potklase su totalne i disjunktne. Potklasa *Nezaposlen* predstavlja superklasu za potklase: *Dete*, *Učenik*, *Student* i *Na_Birou*. Potklasa *Na_Birou* sadrži sve nezaposlene, koji su prijavljeni na biro za zapošljavanje. Ove potklase su parcijalne i presečne. Parcijalne su, jer

među nezaposlenim postoje i drugi podskupovi stanovnika, kao što su, na primer, domaćice. Potklase su presečne, jer studenti mogu biti prijavljeni na birou za zapošljavanje. □



Slika 2.31.

2.5.1.5. *Cilj i efekti uvođenja IS_A hijerarhija*

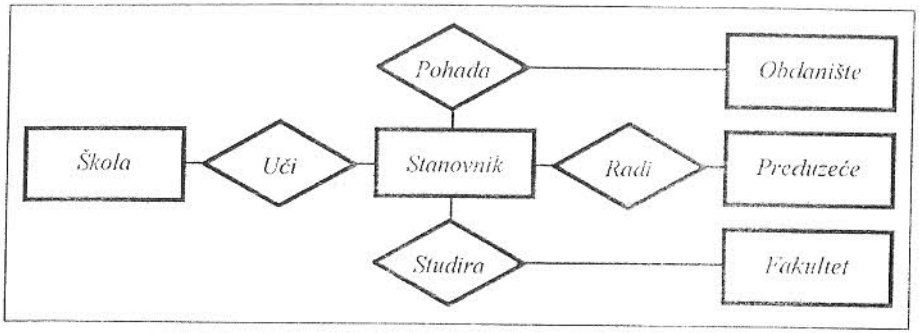
Osnovni cilj uvođenja *IS_A* hijerarhija u skup koncepata ER modela podataka je izgradnja semantički bogate i verne slike statičke strukture realnog sistema. Pri tome se, na nivou intenzije, postižu sledeći efekti:

- eksplicitno ukazivanje na činjenicu da je postojanje različitih uloga pojedinih entiteta u realnom sistemu bitno za zadovoljavanje zahteva korisnika informacionog sistema,
- prirodnije predstavljanje veza između entiteta sa određenim ulogama i entiteta nekog drugog skupa.

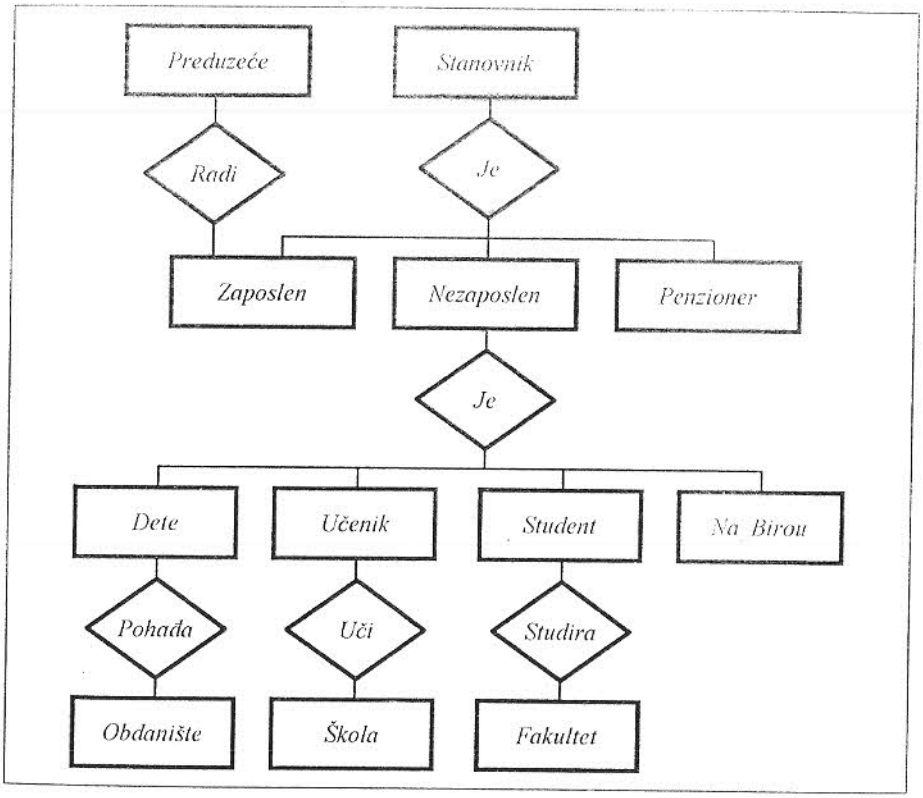
Primer 2.38. ER dijagrami na slikama 2.32 i 2.33 ilustruju tvrdnje da se uvođenjem *IS_A* hijerarhije omogućava prirodnije predstavljanje veza između entiteta sa određenom ulogom i entiteta nekog drugog skupa. □

Na ekstenzionalnom nivou se, takođe, postižu određeni efekti. To su:

- izbegavanje nula vrednosti za specifična obeležja, u slučaju primene specijalizacije i
- izbegavanje ponavljanja istih podataka, u slučaju primene generalizacije na polazne tipove entiteta sa presečnim ekstenzijama.



Slika 2.32.



Slika 2.33.

Primer 2.39. Na slici 2.34 je prikazana jedna ekstenzija tipa entiteta *Radnik*, a na slici 2.35 je prikazana odgovarajuća ekstenzija *IS_A* hijerarhije sa slike 2.29. Znaci ω u kolonama obeležja *KLASA*, *BRPJ* i *SPEC* na slici 2.34 predstavljaju nula vrednosti. □

Radnik

MBR	*IME*	*PRZ*	*ZAN*	*KLASA*	*BRPJ*	*SPEC*
154	*Ivo*	*Ban*	*Programer*	ω	*3*	ω
113	*Ana*	*Tot*	*Programer*	ω	*2*	ω
168	*Eva*	*Ras*	*Projektant*	ω	ω	*BP*
103	*Ana*	*Kon*	*Daktilograf*	*A*	ω	ω
100	*Aca*	*Car*	*Pravnik*	ω	ω	ω

Slika 2.34.

Radnik

MBR	*IME*	*PRZ*	*ZAN*
159	*Ivo*	*Ban*	*Programer*
113	*Ana*	*Tot*	*Programer*
168	*Eva*	*Ras*	*Projektant*
103	*Ana*	*Kon*	*Daktilograf*
100	*Aca*	*Car*	*Pravnik*

Programer

MBR	*BRPJ*
159	*3*
113	*2*

Projektant

MBR	*SPEC*
168	*BP*

Daktilograf

MBR	*KLASA*
103	*A*

Slika 2.35.

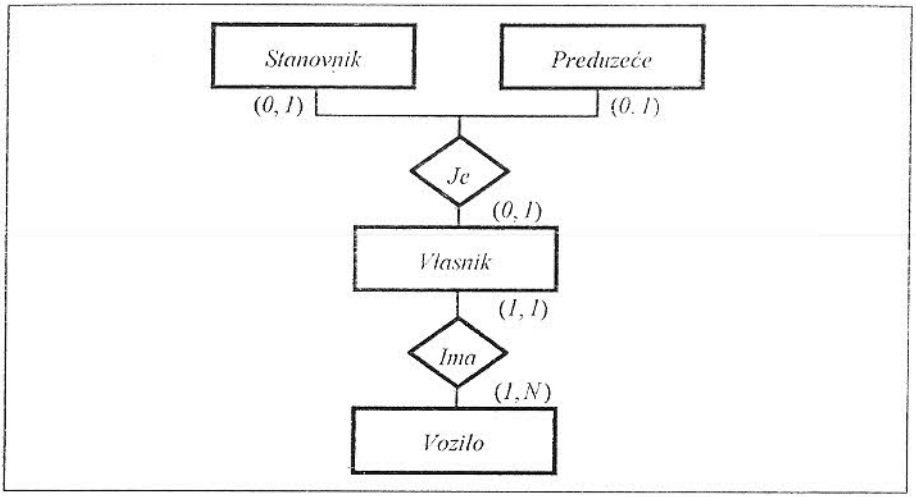
2.5.2. *Kategorija i kategorizacija*

U svim do sada posmatranim primerima sve potklase jedne superklase su pripadale jednoj kategoriji. Neki put se javlja potreba izgradnje modela u kojem potklasa objedinjuje pojave potpuno različitih tipova entiteta. U tom slučaju se potklasa naziva kategorijom. Pojmovi kategorije i kategorizacije se uvode putem sledećeg primera.

Primer 2.40. Posmatraju se tipovi entiteta *Stanovnik* i *Preduzeće*. Vlasnik vozila može biti bilo stanovnik bilo preduzeće. Potrebno je kreirati klasu koja će uključiti entitete oba tipa (stanovnik i preduzeće), pod uslovom da poseduju vozilo. U tu svrhu se definiše kategorija, pod nazivom *Vlasnik*. Skup pojava tipa entiteta *Vlasnik* je podskup unije

skupova entiteta *Stanovnik* i *Preduzeće*, te saglasno tome predstavlja potklasu. Rezultat opisane kategorizacije prikazan je na slici 2.36.

U slučaju kategorizacije, mehanizam nasleđivanja obeležja je selektivan. Na slici 2.36, svaka pojava tipa entiteta *Vlasnik* nasleđuje obeležja ili od *Stanovnika* ili od *Preduzeća*, ali svakako ne od oba. Pošto superklase jedne kategorije poseduju različite ključeve, za jednoznačnu identifikaciju članova kategorije uvodi se novo obeležje - ključ kategorije. Ovaj ključ se naziva i surogatom. Pored ključa - surogata, kategorija može posedovati i druga specifična obeležja. □



Slika 2.36.

2.5.3. *Gerund*

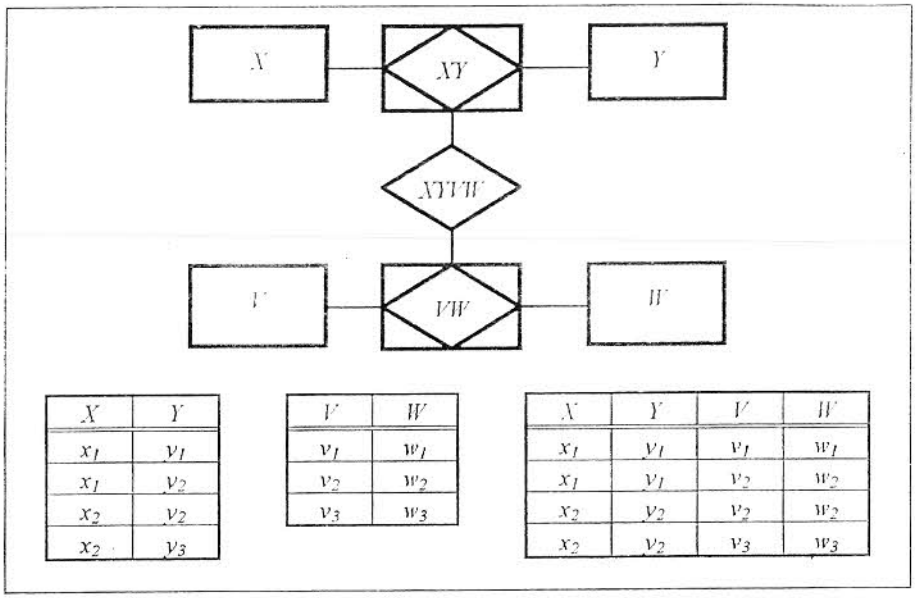
Gerundij, ili kratko gerund, je glagolska imenica. U ER modelu podataka se ***gerundom*** naziva tip entiteta, dobijen transformacijom tipa poveznika. Osnovni razlog za uvođenje gerunda u model podataka je da bi se povećalo bogatstvo semantike za izgradnju modela realnog sistema. Uvođenjem gerunda se prevazilazi i problem direktnog povezivanja dva tipa poveznika. U ER dijagramima se gerund predstavlja rombom upisanim u pravougaonik.

Gerund se koristi za modeliranje situacija, kod kojih su (ne nužno sve) pojave jednog tipa poveznika povezane sa pojavama nekog drugog tipa poveznika. Tada se povezani tipovi poveznika transformišu u gerunde. Do slične situacije dolazi kada je potrebno povezati neki tip poveznika R sa nekim tipom entiteta. Tada se ponovo tip poveznika R pretvara u gerund. Na slici 2.37 je prikazan principijelni ER dijagram prime-

ne gerunda sa mogućim primerom ekstenzije za relacije poveznika. Semantika ovog ER dijagrama je sledeća:

- entiteti klase X su u vezi sa entitetima klase Y, (to su (x, y) parovi),
- entiteti klase V su u vezi sa entitetima klase W, (to su (v, w) parovi),
- neki (eventualno svi) (x, y) parovi su povezani sa nekim (v, w) parovima i ta veza može egzistirati samo izmedu postojećih (x, y) i (v, w) parova.

Saglasno rečenom i slici 2.37, četvorka (x_1, y_2, v_1, w_2) ne može pripadati ekstenziji tipa poveznika $XYVW$.

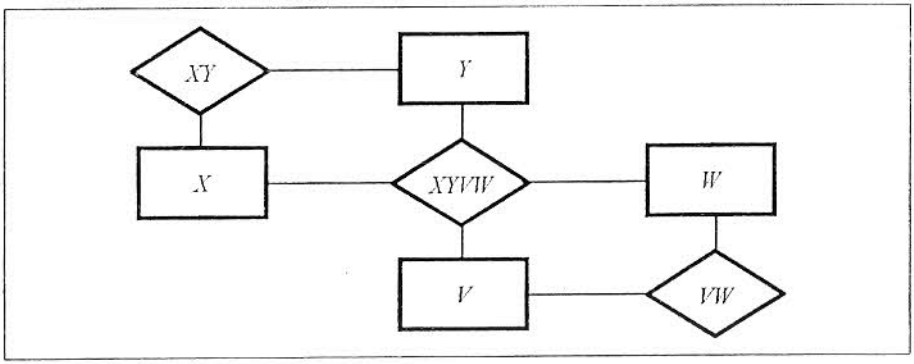


X	Y
x_1	y_1
x_1	y_2
x_2	y_2
x_2	y_3

V	W
v_1	w_1
v_2	w_2
v_3	w_3

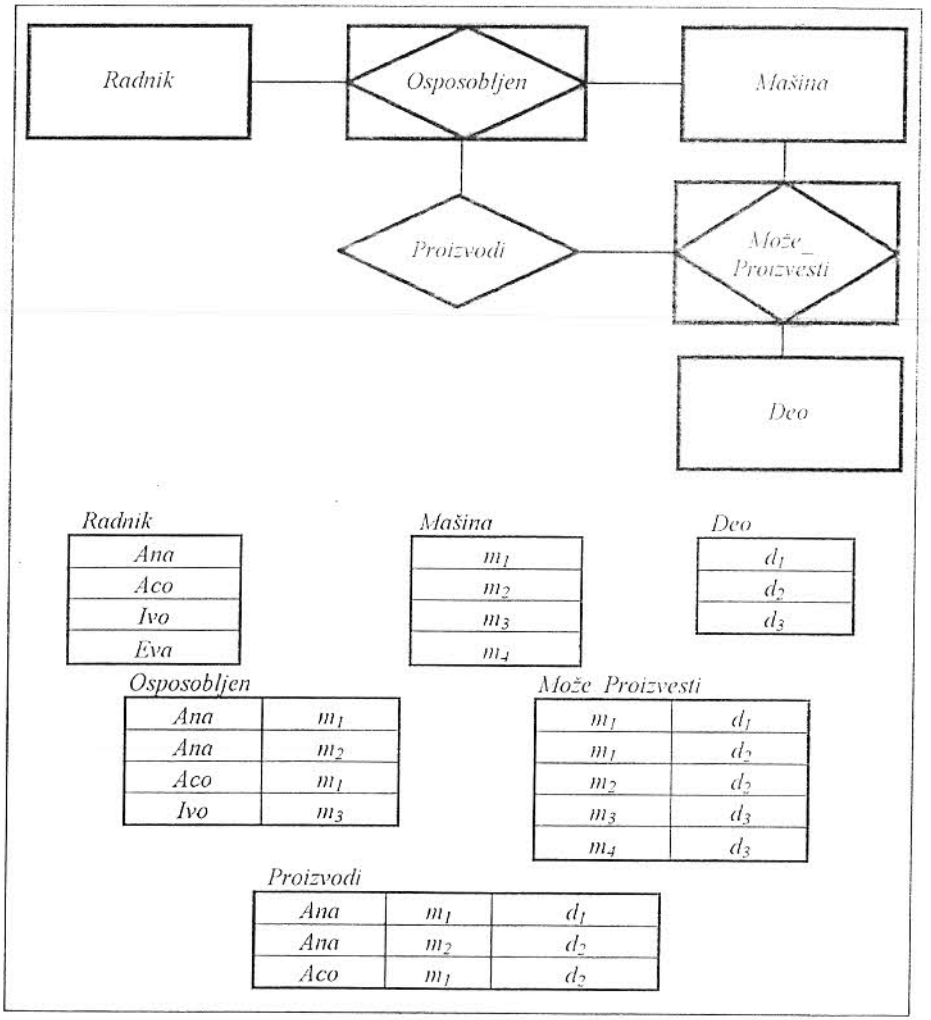
X	Y	V	W
x_1	y_1	v_1	w_1
x_1	y_1	v_2	w_2
x_2	y_2	v_2	w_2
x_2	y_2	v_3	w_3

Slika 2.37.



Slika 2.38.

Alternativni ER dijagram sa odgovarajućom ekstenzijom, prikazan na slici 2.38, ne poseduje istu semantiku kao dijagram sa gerundima. Naime, sa dijagrama na slici 2.38 se ne može zaključiti da je preduslov za postojanje svake (x, y, v, w) četvorke, postojanje po jedne (x, y) i (v, w) dvojke sa istim x, y, v i w vrednostima. Šta više, dijagram na slici 2.38 sugeriše da su tipovi poveznika XY, VW i $XYVW$ međusobno potpuno nezavisni, te da može postojati četvorka (x, y, v, w), a da ne postoji odgovarajuća (x, y) ili (v, w) dvojka.



Radnik

Ana
Aco
Ivo
Eva

Mašina

m_1
m_2
m_3
m_4

Deo

d_1
d_2
d_3

Osposobljen

Ana	m_1
Ana	m_2
Aco	m_1
Ivo	m_3

Može_Proizvesti

m_1	d_1
m_1	d_2
m_2	d_2
m_3	d_3
m_4	d_3

Proizvodi

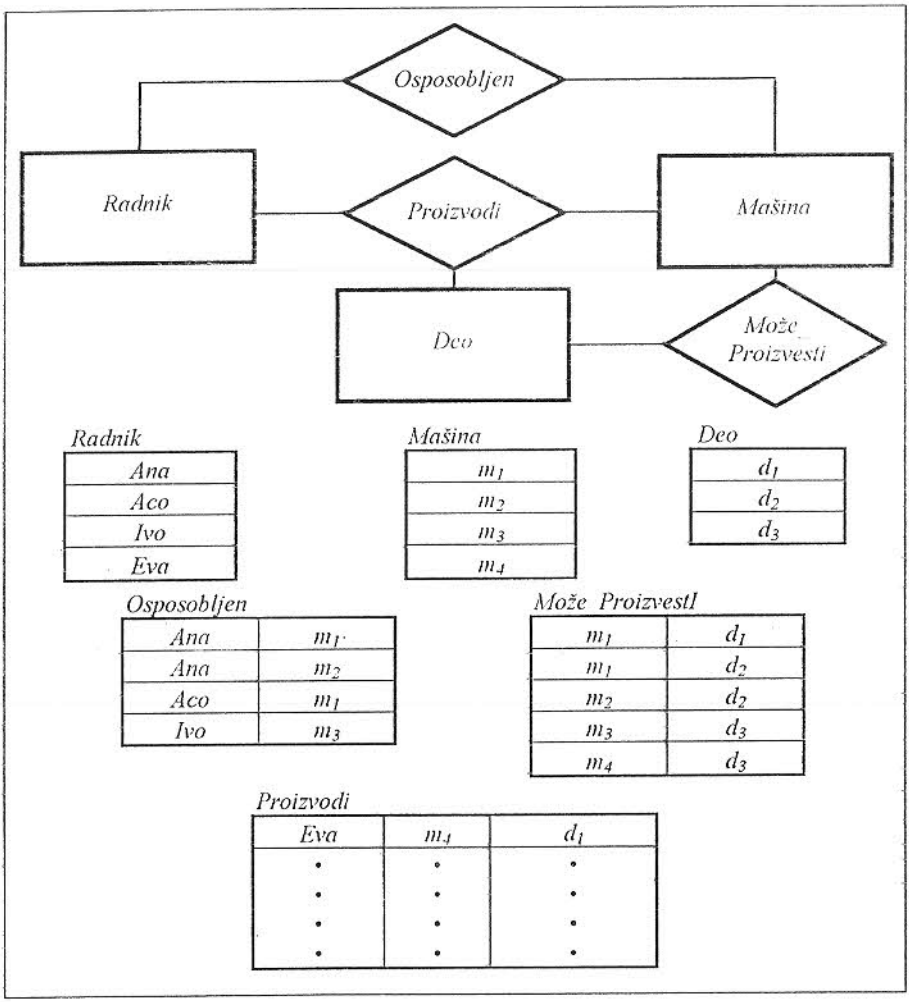
Ana	m_1	d_1
Ana	m_2	d_2
Aco	m_1	d_2

Slika 2.39.

Primer 2.41. Posmatraju se klase entiteta: *Radnik* , *Mašina* i *Deo*. Između entiteta ovih klasa važe sledeći odnosi:

- radnik *r* je osposobljen za rad na mašini *m*,
- na mašini *m* se može proizvesti deo *d*,
- radnik *r*, na nekim od mašina *m*, za koje je osposobljen, izrađuje delove *d*, koji se na tim mašinama mogu proizvesti.

Odgovarajući ER dijagram, prikazan je, sa mogućom ekstenzijom na slici 2.39.



Slika 2.40.

Na slici 2.40 prikazan je ER dijagram sa mogućom ekstenzijom, koji opisuje situaciju:

- radnik r je osposobljen za rad na mašini m,
- na mašini m se može proizvesti deo d,
- radnik r proizvodi deo d na mašini m.

Treba zapaziti da, saglasno strukturi ER dijagrama na slici 2.40, ekstenzija može sadržati podatke da na mašini m radi radnik r, koji nije osposobljen za rad na toj mašini i da čak proizvodi deo koji se na toj mašini ne može proizvesti. □

2.5.4. *Heuristička uputstva za projektovanje modela realnog sistema putem ER modela podataka*

Model entiteta i poveznika je prevashodno namenjen za izgradnju približne slike entiteta i poveznika, koji egzistiraju u ljudskom intelektu, putem koncepata na nivou apstrakcije naziva skupova sličnih entiteta ili poveznika, uz eventualno navođenje njihovih zajedničkih osobina.

Osnovni preduslov za projektovanje modela statičke strukture realnog sistema predstavlja precizno poznavanje tog realnog sistema. Precizno poznavanje realnog sistema znači obezbeđenje informacija o parametrima, kao što su: njegova struktura, funkcije, ciljevi poslovanja, pravila poslovanja i odnosi između entiteta. Do tih saznanja se dolazi snimanjem realnog sistema. Rezultati tog snimanja se, često, izražavaju putem neformalnog, tekstualnog opisa parametara realnog sistema. Zadatak projektovanja modela statičke strukture realnog sistema se tada svodi na izradu ER dijagrama, prevođenjem navoda iz tog tekstualnog opisa u koncepte ER modela podataka. U daljem tekstu se navode neka heuristička uputstva za prevođenje tekstualnog opisa u ER dijagrame.

- ***Imenice*** ukazuju na potrebu uvođenja tipova entiteta.
- ***Glagolski oblici*** ukazuju na potrebu uvođenja tipova poveznika ili gerunda.
- Fraze oblika "***bar jedan***", "***najmanje jedan***", "***više***" i slične, ukazuju na kardinalitete tipova poveznika ili gerunda.
- Postojanje različitih ***uloga*** entiteta jednog skupa u vezama sa entitetima drugih skupova, ukazuje na potrebu uvođenja više tipova poveznika između odgovarajućih tipova entiteta.
- ***Veze između entiteta jednog skupa*** ukazuju na potrebu uvođenja rekurzivnog tipa poveznika. Kod rekurzivnih veza je preporučljivo da se uloge entiteta eksplicitno navedu.
- Vremensko ***prethođenje*** entiteta jednog skupa u odnosu na entitete nekog drugog skupa, ukazuje na egzistencijalnu zavisnost entiteta drugog skupa od entiteta prvog skupa i potrebu uvođenja minimalnog kardinaliteta $a_1 = 1$.
- Potreba takvog ***selektivnog*** povezivanja entiteta tri ili više skupova, kod kojeg u vezi mogu učestvovati samo entiteti, koji su već u nekakvoj drugoj vezi sa entitetima jed-

nog (ili više) drugih skupova, ukazuje na neophodnost korišćenja gerunda, kao modela tih drugih veza.

- Postojanje entiteta, jednog skupa, sa ***specifičnim osobinama*** ili sa ***specifičnim vezama*** sa entitetima drugih skupova, ukazuje na potrebu uvođenja *IS_A* hijerarhije.
- Postojanje entiteta sa ***istim ulogama***, u okviru dva različita skupa entiteta, ukazuje na potrebu uvođenja kategorije.
- Svako ***obeležje*** može pripadati samo jednom tipu entiteta ili samo jednom tipu poveznika (tek u ekstenziji: pojave tipa poveznika nasleđuju ključeve povezanih pojava tipova entiteta, pojave slabog tipa entiteta nasleđuju ključ pojave regularnog tipa entiteta, pojave potklase nasleđuju ključ i osobine pojave superklase).
- Tip entiteta ili tip poveznika sadrži samo ona obeležja skupa entiteta ili skupa poveznika, koja su bitna za realizaciju *ciljeva* postavljenih pred automatizovani informacioni sistem.

3. Glava

Koncepcija baze podataka i sistem za upravljanje bazom podataka

Da bi se stekla precizna predstava o bazama podataka, nije dovoljno samo definisati pojam baze podataka. Potrebno je prvo baze podataka sagledati u kontekstu njihovog istorijskog razvoja. Naime, vrednost svakog sistema, pa i baze podataka kao sistema se najbolje shvata ne na osnovu poznavanja njega samog, već na osnovu činjenice da taj sistem predstavlja korak u evoluciji rešavanja onih problema, koje prethodni sistemi nisu mogli da reše.

Cilj ovog teksta je da kroz prikaz razvoja postupaka za organizovanje i upravljanje podacima i putem analize prednosti i nedostataka svakog od razvojnih koraka, ukaže na ideje i ciljeve baze podataka, kao pristupa organizovanju podataka i na zadatke sistema za upravljanje bazama podataka (SUBP). U tekstu se koristi pojam aplikacije u smislu skupa čije elemente čine programi, strukture podataka i pravila za njihovo korišćenje. Pri tome se podrazumeva da je aplikacija namenjena za automatizovano praćenje ponašanja jednog dela (na primer, jedne funkcije, kao što je prodaja) nekog realnog sistema.

3.1. O pojmu fizičke strukture podataka

U daljem tekstu ovog poglavlja se često koristi pojam fizičke strukture podataka, te se on ovde, ukratko, uvodi. ***Fizička struktura podataka*** se dobija kada se ekstenzionalni model realnog sistema ili samo jedne klase entiteta smesti na medijum memorijskog

uređaja. Fizička struktura podataka zavisi od intenzionalnog modela, od tipova entiteta, tipova poveznika i njihovih obeležja, ali i od niza drugih parametara. U te parametre spadaju:

- definisane dužine vrednosti obeležja,
- postupci dodele adresa lokacija memorijskog prostora pojavama tipova entiteta i poveznika,
- postupci za memorisanje informacija o vezama između pojava tipova entiteta i poveznika,
- pomoćne strukture podataka, kao što su indeksi,
- veličina fizičkih organizacionih jedinica podataka (blok, stranica),
- raspodela pojava tipova entiteta i poveznika po fizičkim organizacionim jedinicama podataka,
- karakteristike eksternog memorijskog uređaja i drugo.

Postupci za dodelu adresa i memorisanje veza, kao i, eventualno, postojanje pomoćnih struktura podataka određuju ***organizaciju podataka***. Ovi pojmovi su detaljno obrađeni u literaturi, na primer [Mo].

3.2. *Klasična organizacija datoteka*

Automatska obrada podataka (AOP) kod koje su pojedine aplikacije jednog informacionog sistema međusobno nezavisne i kod koje se za svaku aplikaciju kreiraju i održavaju posebne datoteke sa svim potrebnim podacima, naziva se klasičnom. Takođe se klasičnim naziva i odgovarajući pristup organizovanju podataka. Do druge polovine šezdesetih godina, to je predstavljao i jedini poznat pristup organizovanju podataka informacionog sistema. Sreće se i danas. Ima svojih prednosti nad drugim pristupima, ali i ozbiljnih nedostataka. Osnovna prednost klasičnog pristupa leži u jednostavnosti projektovanja i realizacije.

Informacioni sistem sa nepovezanim aplikacijama i posebnim datotekama za svaku aplikaciju, vremenom dolazi u kontradikciju sa samim sobom, zbog neusaglašenosti podataka o istoj komponenti stanja realnog sistema (objekta upravljanja) u različitim datotekama. Razlog za ovu neusaglašenost predstavlja takozvana nekonzistentnost podataka, koja se ogleda u činjenici da ista obeležja za iste entitete, imaju različite vrednosti u datotekama različitih aplikacija. Pri tome, svaka aplikacija, izolovano gledano, funkcioniše dobro. Objašnjenje ovog fenomena leži u nezavisnom razvoju pojedinih aplikacija i u vremenski neusaglašenom ažuriranju različitih datoteka istim podacima. Uzrok ove pojave, svakako predstavlja ***redundantnost*** ili preopširnost podataka.

Ozbiljan nedostatak klasične organizacije AOP predstavlja i ***čvrsta povezanost programa i podataka***. Svaki program sadrži opis i logičke i fizičke strukture podataka datoteke koju koristi. Takođe, procedure u programu zavise od fizičke strukture podataka.

U jednoj aplikaciji , istu datoteku može koristiti više programa. Ako istu datoteku koristi više programa i ako se zbog potreba jednog od njih izmeni logička ili fizička struktura te datoteke, moraju se menjati i svi drugi programi. Ova činjenica dovodi, vremenom, do situacije kada se čak 80% programerskih resursa angažuje na održavanju (menjanju,

kompiliranju i testiranju) postojećih programa. Posebno se često menja logička struktura datoteke, kao posledica: izmena pravila i uslova poslovanja u realnom sistemu, ili novih zahteva korisnika. Te promene se reflektuju u potrebi dodavanja novih obeležja ili ograničenja u logički opis zajedničke datoteteke i do izmene svih programa, koji je koriste, bez obzira da li su izmene strukture datoteke bitne za svaki od programa. Izmene fizičke strukture datoteke diktira potreba da se obezbede uslovi za efiklasnije izvršavanje jednog od programa. Ta promena ili povlači izmene opisa datoteke i delova procedure, koji se bave manipulisanjem podacima, u drugim programima, ili dovodi do potrebe da se vrši reorganizacija datoteke za potrebe izvršavanja drugih programa.

Primer 3.1. Kao ilustracija zavisnosti programa i podataka može poslužiti i situacija, tipična za klasičnu organizaciju podataka. Tu su karakteristične dve vrste datoteka: takozvane matične i transakcione datoteke. Matične datoteke sadrže podatke o entitetima jedne klase, a transakcione datoteke sadrže podatke o događajima vezanim za entitete različitih klasa. Na početku obrade, svaki slog transakcione datoteke proširuje se podacima iz svih potrebnih matičnih datoteka. Na taj način se dobija datoteka, čiji tip entiteta sadrži relativno veliki broj obeležja. Ova datoteka služi za davanje odgovora na veći broj informacionih zahteva, što znači da je koristi veći broj programa. Izmena logičke strukture jedne od matičnih datoteka, dovodi do izmene strukture proširene transakcione datoteke i do potrebe izmene svih programa, koji ovu datoteku koriste. □

3.3. *Ideja baze podataka*

Negativna iskustva klasične organizacije podataka i pokušaja da se do integrisanog informacionog sistema dođe povezivanjem aplikacija, dovela su do definisanja novog pristupa za realizaciju integrisanih informacionih sistema. Novi pristup je zasnovan na ideji da treba integrisati podatke, a ne aplikacije. Rezultat integrisanja datoteka različitih aplikacija nazvan je bazom podataka. Baza podataka nije predstavljala novu tehniku memorisanja podataka na medijumima eksternih memorijskih uređaja, niti novu tehniku dodela adresa lokacija slogovima, već nov pristup upravljanju podacima.

Tri osnovna nedostatka klasične organizacije podataka predstavljaju:

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

Osnovne postavke koncepcije baze podataka su da se:

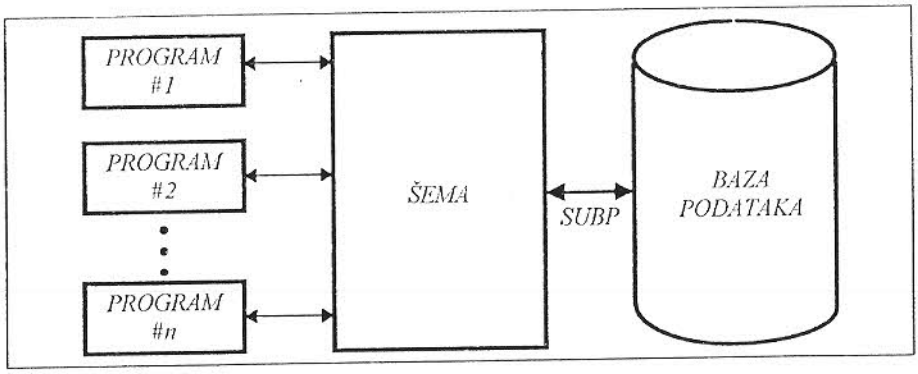
- svi podaci jednog informacionog sistema integrišu u jednu fizičku strukturu - bazu podataka,
- svi programi, pri obradi baze podataka, koriste standardizovanim softverskim rutinama, takozvanim softverom za upravljanje bazom podataka (SUBP), koji preslikava fizičku strukturu podataka na, programima poznatu, *šemu baze podataka*.

3.3.1. Fizička nezavisnost

Integracija podataka informacionog sistema u jednu fizičku strukturu podataka, može se shvatiti i kao integracija podataka ranije nezavisno razvijenih datoteka za različite aplikacije. Pri tome, nad skupom obeležja tipova entiteta datoteka, formira se takozvana šema kao struktura nad skupom različitih tipova entiteta. Šema predstavlja apstraktni model realnog sistema i njegove baze podataka. Idealni, ali i praktično nikad dostignuti cilj izgradnje šeme je da se svako obeležje nađe u samo jednom tipu entiteta, kako bi se izbegla redundansa podataka.

Nad šemom se gradi odgovarajuća *fizička* struktura podataka, koja predstavlja samu bazu podataka. Svaki program poznaje samo šemu i na osnovu nje, putem SUBP, koristi ili menja stanje baze podataka. Pošto svi programi koriste istu šemu, fizička struktura baze podataka je veoma kompleksna. Ovu kompleksnost nameće potreba različitih načina pristupa istim podacima za potrebe različitih programa.

Primer 3.2. Za jedan program pojavama određenog tipa entiteta treba pristupati saglasno rastućim vrednostima primarnog ključa, za potrebe drugog treba pristupiti grupama pojava istog tipa entiteta saglasno vrednostima nekog sekundarnog ključa, a za potrebe trećeg programa istim pojavama treba pristupati direktno - transformacijom vrednosti ključa u adresu. Da bi se to postiglo, potrebno je kombinovati nekoliko vrsta organizacije podataka (u datom primeru, redom, spregnutu, indeksnu i rasutu). □



Slika 3.1.

Drugu dimenziju kompleksnosti fizičke strukture nameće potreba memorisanja veza (odnosa) između pojava različitih tipova entiteta. SUBP ima zadatak da izoluje programe (i programere) od kompleksnosti fizičke strukture podataka. Međutim, treba naglasiti da to SUBP ne radi samostalno, već po prethodno specificiranim uputstvima. Ova uputstva radi takozvani *administrator* baze podataka, jedan ili grupa visokostručnih ljudi, koji projektuju fizičku strukturu baze podataka i brinu o zaštiti baze podataka od neovlašćenog korišćenja i od uništenja. Administrator saopštava SUBP, putem specijanih programa, uputstva vezana za izgradnju strukture, za korišćenje i ažuriranje baze podataka, što SUBP, na zahtev korisničkih programa, sprovodi. Kompleksnost projektovanja fizičke

strukture baze podataka, njenog generisanja i održavanja preneta je sa programera na administratora baze podataka, ali je olakšano njeno korišćenje i ažuriranje, što je ostalo u delokrugu zadataka programera. Slika 3.1 daje grafički prikaz ove, osnovne, ideje koncepcije baze podataka.

Opisani pristup dovodi do niza poboljšanja u organizovanju i rukovanju podacima. To su:

- smanjenje zavisnosti šeme i programa od fizičke strukture podataka,
- smanjenje redundantnosti podataka,
- poboljšanje konzistentnosti podataka i
- zajedničko korišćenje podataka od strane svih aplikacija.

3.3.2. *Logička nezavisnost*

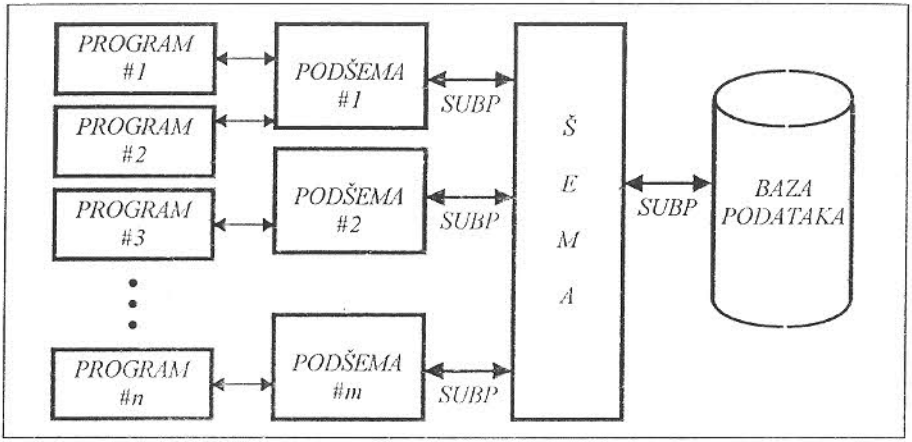
Ubrzo po uvođenju prvih baza podataka u eksploataciju, postalo je očigledno da je potrebno obezbediti jedan dalji nivo nezavisnosti programa i podataka. Naime, šema se, u mnogim slučajevima, pokazala veoma kompleksnom. Takođe, svaka promena u šemi zahtevala je izmene u svim programima. Pokazala se potreba za uvođenjem dva nivoa nezavisnosti programa i podataka; nivoa logičke i nivoa fizičke nezavisnosti.

Logička nezavisnost znači da izmene šeme ne smeju uticati na izmene programa (naravno, pod uslovom da se ne menjaju obeležja koja program baš koristi). Logička nezavisnost se postiže uvođenjem pojma ***podšeme***. Podšema predstavlja pojam koji se odnosi na onaj deo šeme baze podataka, koji je dovoljan za realizaciju zadataka jednog ili grupe programa.

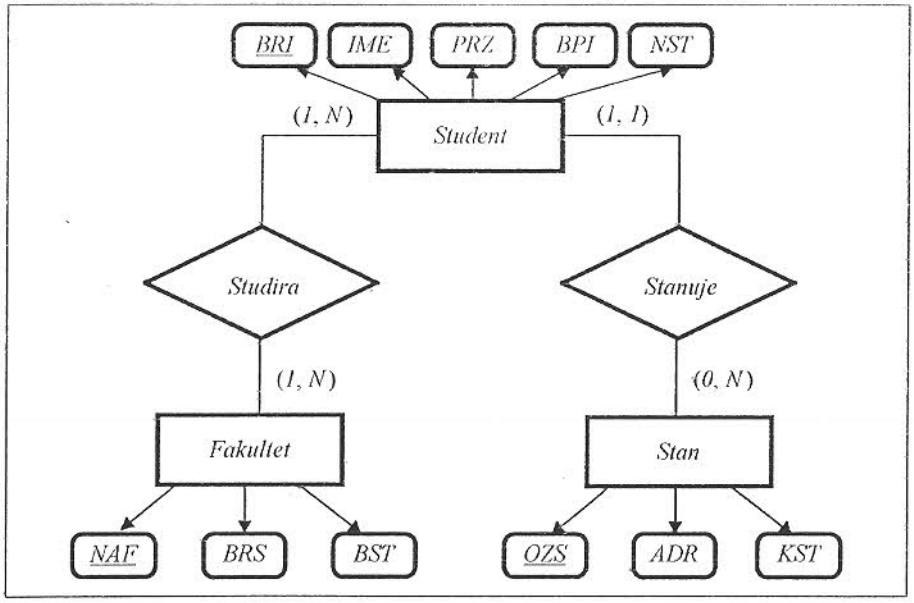
Šema i podšema predstavljaju, redom, modele realnog sistema i jednog njegovog dela. U principu, ta dva modela se nalaze na istom nivou apstrakcije. Mada u određenom smislu, podšema može predstavljati model na višem nivou apstrakcije od šeme.

Naime, podšema sadrži tipove entiteta koji se mogu konstruisati od obeležja različitih tipova entiteta šeme. Dok tipovi entiteta šeme predstavljaju modele klasa entiteta realnog sistema, tipovi entiteta podšeme mogu predstavljati model neke generalizacije klasa entiteta realnog sistema, ili model neke kombinacije klasa entiteta.

Prema ovoj zamisli, svaki program koristi bazu podataka putem svoje podšeme. SUBP prevodi zahtev programa, definisan s obzirom na tipove entiteta podšeme, u analogan zahtev, koji bi bio definisan s obzirom na tipove entiteta u šemi. Dalje, SUBP prenosi iz baze podataka u operativnu memoriju pojave tipova entiteta šeme i konstruiše od njih pojave tipova entiteta podšeme. Tek te podatke SUBP prezentira programu. Treba zapaziti da se pojam ***fizičke*** nezavisnosti odnosi na činjenicu da izmene u fizičkoj strukturi baze podataka ne smeju dovoditi do izmena šeme, podšeme i programa. Pojam logičke nezavisnosti odnosi se na činjenicu da izmene šeme ne smeju dovoditi do izmena podšema i programa. Slika 3.2 ilustruje ideju realizacije logičke i fizičke nezavisnosti programa i podataka.



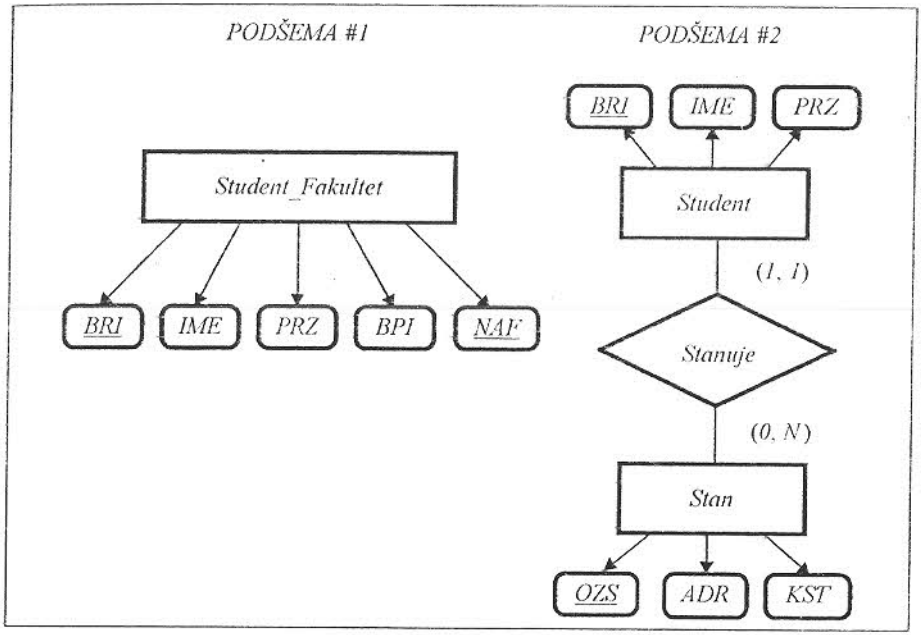
Slika 3.2.



Slika 3.3.

Primer 3.3. Na slici 3.3 je nacrtana šema jedne male, hipotetične univerzitetske baze podataka. Tip entiteta *Student* sadrži obeležja: broj indeksa (*BRI*), ime (*IME*), prezime (*PRZ*), broj položenih ispita (*BPI*) i način stanovanja (*NST*). Tip entiteta *Fakultet* sadrži obeležja: naziv fakulteta (*NAF*), broj semestara (*BRS*) i ukupni broj studenata

(*BST*). Tip entiteta *Stan* poseduje obeležja: oznaka stana (*OZS*) adresa (*ADR*) i kvalitet stanovanja (*KST*). Na slici 3.4 prikazane su dve mogućne podšeme šeme sa slike 3.3. Treba zapaziti da izmedu tipova entiteta šeme i podšema postoji odredena razlika. Tip entiteta *Student_Fakultet* pošeme *#1*, konstruisan je od obeležja tipova entiteta *Student* i *Fakultet* iz šeme. Ključ tipa entiteta *Student_Fakultet* je {*BRI*, *NAF*}. Tip entiteta *Student* u podšemi *#2* sadrži pravi podskup skupa obeležja tipa entiteta *Student* iz šeme. □

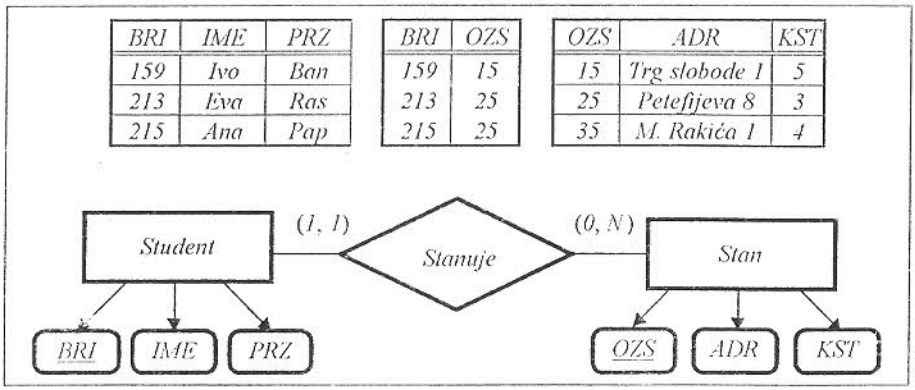


Slika 3.4.

Šema baze podataka se naziva i ***globalnom*** šemom, u smislu da predstavlja model statičke strukture kompletnog realnog sistema. Podšema se naziva i ***eksternom*** šemom. Opis fizičke strukture baze podataka naziva se ***fizičkom*** šemom ili ***internom*** šemom.

Šema i podšema baze podataka predstavljaju pojmove za čiji nivo apstrakcije je karakterističan pojam obeležja. Na nižem nivou apstrakcije, za koji je karakterističan pojam podataka, javljaju se, redom, pojmovi globalnog pogleda i pogleda. ***Globalni pogled*** predstavlja logičku strukturu podataka baze podataka. ***Pogled*** predstavlja takvu sliku baze podataka, kako je programer vidi na osnovu podšeme. Globalni pogled predstavlja sliku baze podataka kako je zamišlja projektant baze podataka.

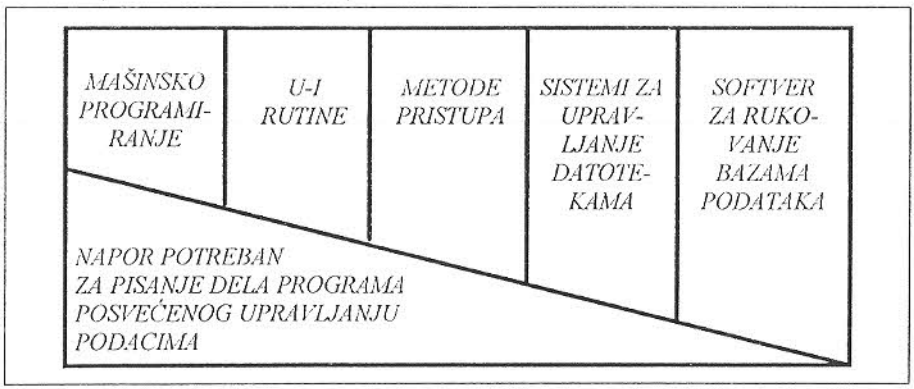
Primer 3.4. Slika 3.5 ilustruje odnos izmedu pojmova podšeme i pogleda. Pogled sa slike 3.5 bi odgovarao videnju baze podataka, recimo, nekog referenta za studentski standard. Pogled, koji bi odgovarao podšemi *#1* sa slike 3.4 odgovarao bi videnju baze podataka, recimo, nekog referenta za studentska pitanja. □



Slika 3.5.

3.4. *Uloga baze podataka u razvoju i korišćenju informacionog sistema*

Karakteristike baze podataka, opisane u prethodnom tekstu, predstavljaju osnovne ciljeve kojima treba težiti pri njenoj izgradnji. U kojoj meri će ti ciljevi biti postignuti zavisi kako od znanja projektanta, tako i od kvaliteta SUBP. Razvoj postupaka za organizovanje i upravljanje podacima imao je dva važna efekta. To su: povećanje produktivnosti programera i stvaranje preduslova za izgradnju integralnih informacionih sistema. Prvi efekat ilustrovan je slikom 3.6, a drugi slikom 3.7.



Slika 3.6.

Slika 3.6 sugeriše da je svaka nova etapa u razvoju: programskih jezika i postupaka upravljanja podacima donosila i smanjenje napora potrebnog za pisanje dela programa posvećenog upravljanju podacima. U fazi mašinskog programiranja, bilo je potrebno da se u program ugrade:

- i postupci za izgradnju fizičke strukture podataka i
- postupci za realizaciju razmene podataka između eksternog memorijskog uređaja i operativne memorije.

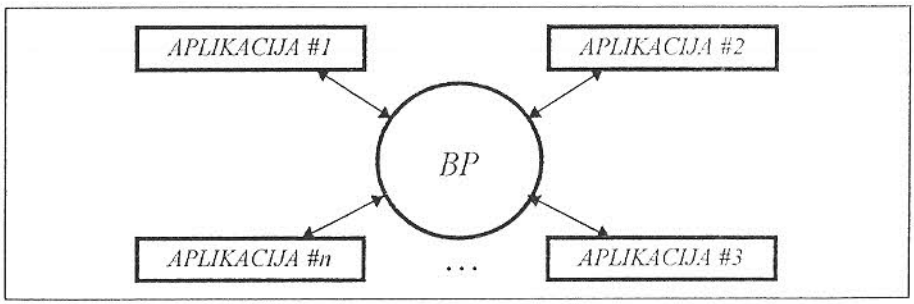
Pojava prvih operativnih sistema dovela je do upotrebe asemblerskih programskih jezika, a smanjenje napora za pisanje dela program posvećenog upravljanju podacima je predstavljala činjenica da su ti prvi operativni sistemi sadržali takozvane ulazno - izlazne (U/I) rutine. U/I rutine su gotovi programi za razmenu podataka između eksternog memorijskog uređaja i operativne memorije, koje je trebalo samo uključiti u korisnički program.

Dalje smanjenje napora donosi pojava viših programskih jezika i metoda pristupa. ***Metode pristupa*** su delovi operativnog sistema, koji brinu o fizičkoj strukturi podataka datoteke. Omogućavaju da se u programu samo deklariše vrsta organizacije datoteke i da se u proceduralnom delu programa razmena podataka između datoteke i programa inicira putem makro poziva tipa READ i WRITE.

Sistemi za upravljanje datotekama donose gotove programe za često primenjivane postupke rada sa datotekama, kao što su: sortiranje, reorganizovanje strukture, prepisivanje sa jednog medijuma na drugi i slično. Ti programi se pozivaju i specijalizuju putem naredbi komandnog jezika operativnog sistema.

Baze podataka sa: mehanizmima za logičku i fizičku nezavisnost i specifikacionim jezicima za manipulisanje podacima dovode do daljeg smanjenja napora potrebnog za pisanje dela programa posvećenog upravljanju podacima.

Slika 3.7 ilustruje ideju izgradnje integrisanih informacionih sistema. Prema današnjem pristupu projektovanju informacionih sistema, baza podataka predstavlja instrument njegove integracije, te joj pripada centralno mesto u strukturi informacionog sistema. Na nju se vezuju aplikacije (podsistemi informacionog sistema) koji izvršavaju specifične zadatke putem svojih programa.



Slika 3.7.

3.5. Sistem za upravljanje bazom podataka

Sistem za upravljanje bazom podataka (SUBP) je programski proizvod, koji omogućava efikasno: formiranje, korišćenje i menjanje baze podataka. Da bi mogao da izvršava ove svoje zadatke, SUBP:

- je zasnovan na nekom od modela podataka,
- podržava programske jezike za:
  - definisanje strukture i uslova integriteta baze podataka,
  - selekciju i izmene (upis, brisanje i modifikacija) sadržaja baze podataka,
- poseduje mehanizme za:
  - upravljanje transakcijama,
  - zaštitu od neovlašćenog pristupa podacima od strane korisnika,
  - zaštitu baze podataka od uništenja,
  - obezbeđenje efikasnog korišćenja baze podataka i
  - upravljanje distribuiranim delovima baze podataka.

Pošto je modelima podataka posvećeno prvo poglavlje knjige, u daljem tekstu će biti, ukratko, opisane ostale funkcije SUBP.

3.5.1. Programski jezici i SUBP

Na sadašnjem stupnju razvoja SUBP, deklarisanje strukture baze podataka se, po pravilu, vrši putem posebnog *jezika za definisanje podataka*. Postupci za kompleksna izračunavanja se pišu u nekom drugom programskom jeziku, takozvanom *jeziku domaćinu*, a postupci selekcije i izmene sadržaja baze podataka se realizuju putem naredbi jednog trećeg, takozvanog *jezika za manipulisanje podacima*. Naredbe manipulacionog jezika se ugrađuju u programe pisane u jeziku domaćinu. Razdvajanje deklaracije strukture baze podataka od postupaka: za selekciju, menjanje sadržaja i izračunavanja u posebne programe, posledica je činjenice da su podaci u bazi podataka trajno memorisani, te je njihovu intenzionalnu strukturu dovoljno opisati samo jednom, da bi se baza podataka mogla višekratno koristiti i menjati.

Kao što je već rečeno, šema baze podataka (intenzionalni opis baze podataka) se deklariše putem jezika za opis podataka. Taj jezik predstavlja deo SUBP. Jezik za opis podataka predstavlja notaciju za opis elemenata šeme baze podataka, putem koncepata nekog modela podataka.

Primer 3.5. Korišćenjem naredbi relacionog jezika podataka SQL i ne insistirajući na preciznoj sintaksi, opis tipa entiteta *Student* iz šeme sa slike 3.3, bi imao sledeći oblik

CREATE TABLE *Student*
(*BRI* : *integer* **NOT NULL**,
IME : *char*(*10*), *PRZ* : *char*(*10*), *BPI* : *integer*, *NST* : *char*(*15*));

CREATE UNIQUE INDEX ON *Student* (*BRI*).

Prva linija ovog programskog koda ukazuje da SUBP treba da formira (praznu) tabelu sa nazivom *Student*, u koju će se upisivati pojave odgovarajućeg tipa entiteta. Druga i treća linija opisuju obeležja tipa entiteta sa naznakom da ih fizički treba realizovati u obliku nizova karaktera određene dužine ili celih brojeva fiksne dužine. Četvrta linija ukazuje da SUBP treba da formira i održava stablo pristupa (indeks) na osnovu vrednosti obeležja *BRI* , čime je fizička organizacija skupa pojava tipa entiteta određena kao indeksna. Fraze NOT NULL i UNIQUE INDEX služe za deklarisanje ključa, a time jednog od uslova integriteta baze podataka.

Ostali tipovi entiteta i poveznika šeme iz primera 3.3 bi bili opisani na sledeći način:

CREATE TABLE *Fakultet*
(*NAF* : *char*(*15*) **NOT NULL**, *BRS* : *integer*, *BST* : *integer*);
CREATE UNIQUE INDEX ON *Fakultet* (*NAF*)

CREATE TABLE *Stan*
(*OZS*: *integer* **NOT NULL**, *ADR* : *char*(*25*), *KST* : *integer*);
CREATE UNIQUE INDEX ON *Stan* (*OZS*)

CREATE TABLE *Stu_Fa*
(*BRI* : *integer* **NOT NULL**, *NAF* : *char*(*15*) **NOT NULL**);
CREATE UNIQUE INDEX ON *Stu_Fa* (*BRI*, *NAF*)

CREATE TABLE *Stu_Sta*
(*BRI* : *integer* **NOT NULL**, *OZS* : *char*(*25*) **NOT NULL**);
CREATE UNIQUE INDEX ON *Stu_Sta* (*BRI*). □

Jezik za opis podataka se koristi za opis šeme baze podataka, pri njenom projektovanju ili pri izmeni projekta. Opisivanje podšema i pogleda se vrši ili u istom ili u nekom sličnom jeziku podataka.

Primer 3.6. Podšema iz primer 3.3 bi, u jeziku SQL bila opisana na sledeći način

CREATE VIEW *Student_Fakultet* **AS**
SELECT *BRI*, *IME*, *PRZ*, *BPI*, *NAF*
FROM *Student S*, *Stu_Fa F*
WHERE *S.BRI* = *F.BRI*.

Prva naredba ovog programa nalaže SUBP da formira pogled pod nazivom *Student_Fakultet*.

Druga linija ukazuje na obeležja, čije vrednosti pogled treba da sadrži, treća linija ukazuje kojim tipovima entiteta ta obeležja pripadaju, a četvrta da se pojave tipa entiteta *Student_Fakultet* dobijaju povezivanjem pojava tipa entiteta *Student* i tipa poveznika *Stu_Fa* sa istim *BRI* vrednostima. □

Za definisanje operacija na bazi podataka koristi se poseban jezik, koji se naziva ili jezikom za manipulisanje podacima ili upitnim jezikom. Taj jezik se koristi za

izražavanje naredbi za selekciju dela baze podataka i za menjanje sadržaja baze podataka. Termin ***upitni jezik*** se često koristi kao sinonim za termin jezik za manipulisanje podacima. Strogo govoreći, pojam upita se odnosi samo na one naredbe, koje vrše selekciju dela baze podataka, međutim, upitni jezici su predviđeni za interaktivni rad sa bazom podataka i, po pravilu, sadrže kako naredbe za selekciju, tako i naredbe za ažuriranje (menjanje sadržaje) baze podataka.

Primer 3.7. SQL, jezik podataka relacionog modela poseduje kako naredbe za definisanje podataka, tako i naredbe za selekciju i manipulisanje podacima. Izbor svih studenata fakulteta tehničkih nauka, koji se zovu Ana, realizovao bi se putem programa:

SELECT * **FROM** *Student_Fakultet*
WHERE *NAF* = '*fakultet tehničkih nauka*' **AND** *IME* = '*Ana*',

gde * ukazuje da se traže vrednosti svih obeležja, definisanih pogledom *Student_Fakultet*.

Upis podataka o novom studentu u bazu podataka bi se izvršio putem programa

INSERT INTO *Student*
VALUES (*159*, '*Iva*', '*Car*', *0*, '*studentski dom*'). □

Korišćenje i ažuriranje baze podataka se može vršiti bilo u interaktivnom bilo u paketnom režimu obrade. Kada je reč o interaktivnom režimu, ponovo postoje dve mogućnosti. Jedna je da se program piše "ad hoc", korišćenjem naredbi upitnog jezika i da se te naredbe odmah i interpretiraju, a druga je da postoji unapred napisan i preveden program, koji se po potrebi poziva i izvršava. Programi ovog drugog tipa se nazivaju (interaktivnim) ***aplikativnim programima***. Razlog za postojanje aplikativnih programa ne leži samo u činjenici da korisnici, često, nisu u stanju da napišu program u upitnom jeziku, već i u činjenici da upitni jezici, najčešće, ne poseduju naredbe za kompleksnija izračunavanja. Isto tako, često je potrebno da program sprovede neki dijalog sa korisnikom ili da rezultate selekcije prikaže na štampaču. Zbog toga se aplikativni programi pišu u jeziku domaćinu.

Jezik domaćin je neki od proceduralnih programskih jezika treće generacije ili jezik četvrte generacije. Naredbe jezika domaćina se koriste za izračunavanja, donošenje odluka, dijalog sa korisnikom, za sve osim za stvarnu selekciju i menjanje sadržaja baze podataka. Naredbe manipulacionog jezika se ugrađuju u program, pisan u jeziku domaćinu. Kooperativni rad između aplikativnog programa i naredbi manipulacionog jezika odvija se, u principu, na sledeći način. Aplikativni program poseduje, u operativnoj memoriji, zonu sa lokalnim podacima. Sa tim podacima aplikativni program manipuliše na uobičajeni način. Kada se, pri izvšavanju programa, naiđe na zahtev za selekciju podataka, podaci se, iz baze, prenose u zonu sa lokalnim podacima. Rezultati izračunavanja se, po potrebi, iz zone sa lokalnim podacima, upisuju u bazu podataka.

3.5.2. *Upravljanje transakcijama*

U opštem slučaju, bazu podataka konkurentno*) koristi više različitih programa. Sam program može biti aplikativni program, pisan u nekom jeziku domaćinu ili jednostavan interaktivni upit realizovan korišćenjem upitnog jezika. Različiti korisnici mogu inicirati i više različitih izvršenja istog programa. Jedno izvršavanje svakog programa naziva se *transakcijom*.

Primer 3.8. Svako izvršavanje svakog od programa iz primera 3.7 predstavlja jednu transakciju. □

Konkurentno korišćenje baze podataka krije u sebi određene opasnosti. Naime, ako jedna transakcija učita neki podatak u svoj radni prostor, a druga ga, odmah nakon toga, u bazi podataka izmeni, prva transakcija će koristiti netačan podatak. Očigledno, neophodno je sprečiti istovremeno korišćenje istog podatka od strane dve transakcije. Opisani problem se rešava uvođenjem pojma *zaključavanja* dela baze podataka. Zaključavanjem dela baze podataka, transakcija sprečava druge transakcije da pristupe tom delu baze podataka bilo u cilju čitanja, bilo u cilju njegovog menjanja. Deo baze podataka, koji se podvrgava zaključavanju, može biti:

- skup pojava nekog tipa entiteta ili tipa poveznika,
- određeni broj pojava nekog tipa entiteta ili tipa poveznika,
- jedna pojava nekog tipa entiteta ili tipa poveznika,
- ili čak samo jedan deo neke pojave.

Različiti SUBP zaključavaju delove baze podataka različite veličine. U daljem tekstu će se najmanji deo baze podataka, koji se može zaključati, nazivati *objektom zaključavanja*.

A u bazi podataka	3	3	3	3	4	4
T_1	*ČITAJ A*		*POSTAVI* $A \rightarrow A + 1$		*PIŠI A*	
T_2		*ČITAJ A*		*POSTAVI* $A \rightarrow A + 1$		*PIŠI A*
A u radnom području T_1	3	3	4	4	4	4
A u radnom području T_2		3	3	4	4	4

Slika 3.8.

Primer 3.9. Neka je A obeležje sa celobrojnim vrednostima, a T_1 i T_2 transakcije, koje treba da učitaju A iz baze podataka u svoj radni prostor, povećaju mu vrednost za 1 i upišu ga nazad u bazu podataka. Na slici 3.8 je prikazan jedan od mogućih redosleda iz-

*) Konkurentno korišćenje znači prividno istovremeno korišćenje, pri čemu je prividna istovremenost posledica multiprogramskog rada računara.

vršavanja instrukcija transakcija T_1 i T_2, kao i vrednosti obeležja A u bazi podataka i u radnim prostorima transakcija T_1 i T_2. Mada su obe transakcije dodale po 1, vrednost obeležje A je u bazi podataka povećana samo za jedan.

Rešenje ovog problema je da transakcija T_1, pre čitanja sadržaja obeležja A iz baze podataka, izvrši njegovo zaključavanje i time spreči transakciju T_2 da ga i ona prenese u svoje radno područje. Transakcija T_2 će moći da pročita sadržaj obeležja A tek kada ga transakcija T_1 oslobodi. □

Primer 3.9 sugeriše da program, koji bazu podataka koristi konkurentno sa drugim programima, pored naredbi za čitanje i pisanje u bazu podataka, treba da sadrži i naredbe za zaključavanje onih objekata, kojima želi da pristupi, bilo u cilju čitanja bilo u cilju njihovog menjanja i da te objekte prvo treba da zaključa, pa tek onda da im pristupa, a da otključavanje objekata mora uslediti tek nakon upisa u bazu podataka svih izmenjenih ili novih objekata. Pri tome, nijedna transakcija ne može da zaključa neki objekat, koji je već zaključala neka druga transakcija. Ako je objekat već zaključan, transakcija mora da čeka dok ga transakcija, koja ga je zaključala i ne otključa.

Primer 3.10. Pseudokod programa iz primera 3.9, proširen naredbama za zaključavanje i otključavanje, izgledao bi ovako:

ZAKLJUČAJ A
ČITAJ A
POSTAVI $A \leftarrow A + 1$
UPIŠI A
OTKLJUČAJ A

Neka su T_1 i T_2 različita izvršenja ovog programa. Ako se transakcija T_1 prva pokrene, a obeležje A nije zaključano, T_1 će ga zaključati. Transakcija T_2 će moći da zaključa A tek kada ga T_1 odključa. Saglasno tome, nije teško utvrditi da će nakon završetka transakcije T_2, vrednost obeležja A u bazi podataka iznositi 5. □

Nažalost, uvođenjem mehanizma zaključavanja objekata, ne rešavaju se svi problemi konkurentnog korišćenja baze podataka. Naime, zaključavanje može dovesti do pojave dva nova nepogodna fenomena. To su:

- izgladnjavanje i
- uzajamno zaključavanje.

Izgladnjavanjem se naziva situacija u kojoj jedna transakcija praktično neograničeno dugo vremena čeka da joj se omogući zaključavanje nekog objekta, jer SUBP stalno dodeljuje pravo zaključavanja tog objekta nekoj drugoj transakciji. Naziv ovog fenomena treba da ukaže da neka transakcija stalno traži, a nikako ne dobija pravo da izvrši zaključavanje određenog objekta.

Primer 3.11. Ako bi, u prethodnom primeru, tokom čekanja transakcije T_2 na završetak transakcije T_1, prvo neka transakcija T_3, a zatim, jedna nakon druge, transakcije T_4, T_5,... zatražile i dobile pravo da zaključaju obeležje A, transakcija T_2 bi mogla čekati na pravo zaključavanja obeležja A veoma, ako ne i neograničeno, dugo. □

Jednostavan mehanizam izbegavanja izgladnjavanja predstavlja uvođenje pravila da transakcije stiču pravo na zaključavanje nekog objekta prema redosledu postavljanja zahteva za to zaključavanje.

Mnogo ozbiljniji problem konkurentnog korišćenja baze podataka predstavlja pojava ***uzajamnog zaključavanja***. Do uzajamnog zaključavanja dolazi kada svaka transakcija iz jednog skupa T od dve ili više transakcija čeka da zaključa neki objekat, koji je već zaključala neka druga transakcija iz skupa T. Pošto je svaka transakcija iz T u stanju čekanja, ne može da otključa objekat, koji neka druga transakcija želi da zaključa, tako da sve transakcije iz T ostaju u stanju čekanja. To čekanje može trajati neograničeno dugo. Ovu situaciju ilustruje sledeći primer.

Primer 3.12. Neka su T_1 i T_2 dve transakcije. Sledeće naredbe ovih transakcionih programa su bitne s tačke gledišta pojave uzajamnog zaključavanja:

T_1 : *ZAKLJUČAJ A* *ZAKLJUČAJ B* *OTKLJUČAJ A* *OTKLJUČAJ B*

T_2 : *ZAKLJUČAJ B* *ZAKLJUČAJ A* *OTKLJUČAJ B* *OTKLJUČAJ A*.

Ostale naredbe ovih programa su nebitne za dalja razmatranja fenomena uzajamnog zaključavanja. Ako: izvršavanja T_1 i T_2 započnu približno istovremeno, T_1 zaključa A, a T_2 zaključa B, doći će do uzajamnog zaključavanja. Transakcija T_1 čeka da transakcija T_2 otključa B, dok transakcija T_2 čeka da T_1 otključa A. □

Postoji više postupaka za borbu protiv uzajamnog zaključavanja. Na ovom mestu će biti prikazana tri takva postupka. Prva dva sprečavaju pojavu fenomena uzajamnog zaključavanja, dok treći njegovu pojavu detektuje i poništava.

Prema prvom postupku, svaka transakcija mora zahtevati od SUBP dozvolu za sva zaključavanja odjednom. Ako ih dobije, nastavlja sa radom, inače prelazi u stanje čekanja.

Kod drugog postupka se uvodi neko linearno uređenje u skup svih objekata, tako da transakcije postavljaju svoje zahteve za zaključavanje tačno tim redosledom.

Primer 3.13. Lako se da zaključiti da u slučaju primene prvog postupka zaključavanja, transakcije T_1 i T_2 iz primera 3.12 neće doći u stanje uzajamnog zaključavanja, jer će SUBP dozvoliti transakciji T_1 da zaključa i A i B.

Isto tako, ako se u linearnom uređenju, A nalazi ispred B, tada i transakcija T_1 i transakcija T_2 moraju prvo tražiti da zaključaju A. Ako T_1 prva zaključa A, tada će transakcija T_2 morati da čeka da ga transakcija T_1 otključa, te ponovo neće doći do uzajamnog zaključavanja. □

U slučaju trećeg postupka se ništa ne preduzima da bi se pojava uzajamnog zaključavanja sprečila. Umesto toga, SUBP periodično proverava da li je do uzajamnog zaključavanja došlo. Ako otkrije pojavu uzajamnog zaključavanja, SUBP vraća bar jednu od transakcija iz skupa T na početak i poništava sve promene u bazi podataka, koje je ta transakcija izvršila.

3.5.3. Zaštita od neovlašćenog korišćenja

SUBP mora posedovati mehanizme za zaštitu baze podataka od neovlašćenog korišćenja u višekorisničkom radu. Da bi se ti mehanizmi mogli realizovati, uvode se pojmovi *korisničkog imena i korisničke lozinke*. Korisničko ime i korisničku lozinku dodeljuje administrator baze podataka svakom korisniku u cilju njegovog identifikovanja i sticanja dozvole za rad. Tu identifikaciju i davanje dozvole vrše operativni sistem i SUBP.

Jedan od mehanizama za zaštitu od neovlašćenog korišćenja, predstavlja podšema ili pogled. Selektivnim prikazivanjem tipova entiteta šeme u podšemi i izostavljanjem određenih obeležja tipa entiteta šeme iz njegove slike u podšemi, izostavlja se informacija o njihovom postojanju u šemi, tako da korisnik podšeme niti je svestan postojanja tih tipova entiteta i obeležja niti može da koristi odgovarajuće podatake. Međutim taj mehanizam nije dovoljno strog. Zato se uvodi pojam *privilegije*. Privilegije se povezuju sa elementima intenzionalnog opisa baze podataka, ili čak i za elemente ekstenzije. Pod elementom intenzionalnog opisa baze podataka podrazumeva se: obeležje, tip entiteta, tip poveznika ili podšema, odnosno pogled. Kada je reč o elementima ekstenzije, reč je o takvim podskupovima pojava tipova entiteta koji poseduju određenu osobinu. Privilegije se definišu za svakog korisnika i svaki elemenat intenzionalnog opisa baze podataka, a odnose se na dozvolu:

- samo čitanja pojava,
- čitanja i upisivanja novih pojava,
- čitanja i modifikovanja postojećeih pojava,
- čitanja i brisanja postojećih pojava ili
- čitanja i modifikovanja pojava bez ograničenja.

Privilegije se nazivaju još i *autorizacijom* ili *pravom pristupa*. Podatke o privilegijama, SUBP drži u takozvanoj *autorizacionoj tabeli*. Autorizaciona tabela sadrži trojke (korisnik, elemenat intenzionalnog opisa, privilegija), čime se definiše šta određeni korisnik sme da radi sa pojavama posmatranog elementa intenzionalnog opisa baze podataka.

3.5.4. Zaštita od uništenja

Baza podataka može doći u nekorektno stanje iz mnogo razloga. U njih spadaju:

- nekonzistentnost podataka, bilo zbog greške u definisanju, realizaciji ili izvršavanju uslova integriteta,
- greška u aplikativnom programu,
- pogrešno uneti podaci od strane korisnika,
- greška SUBP ili operativnog sistema,
- kvar računara,
- greška izazvana kolebanjima u: električnom napajanju, temperaturi i vlažnosti okoline,
- prirodne katastrofe i

- namerna oštećenja.

Za zaštitu baze podataka od uništenja, pristupa se postupku *oporavka* baze podataka. Termin oporavak se koristi za postupak poništavanja efekata neke otkrivene greške i prevođenje baze podataka iz nekorektnog u korektno stanje. U tom cilju, SUBP poseduje mehanizme za:

- kopiranje baze podataka na rezervni medijum,
- vođenje ***žurnal*** datoteke, koja sadrži sve promene sadržaja baze podataka tokom određenog intervala vremena,
- oporavak baze podataka.

Pravljenje kopija korektnih stanja baze podataka se izvršava ili nakon određenog broja promena sadržaja baze podataka ili u određenim periodima vremena. Pri tome se, uvek, čuva samo nekoliko poslednjih kopija.

Ključni mehanizam za zaštitu baze podataka od uništenja je vođenje žurnal datoteke. Svaka transakcija, koja je koristila bazu podataka, beleži se na žurnal dataoteku. Pri tome, obično se evidentiraju sledeći podaci: identifikacija inicijatora transakcije, tačno vreme iniciranja transakcije, jedinstvena identifikacija transakcije i, za trasnsakcije koje su dovele do izmene sadržaja baze podataka, beleže se adrese lokacija svih podataka, kojima je pristupano, kao i vrednosti podataka pre i posle izmene. Vrednosti podataka pre izmene se nazivaju *prethodnom*, a vrednosti podataka nakon izmene, *naknadnom slikom*.

Kada se utvrdi da je stanje baze podataka nekorektno, mehanizmi za oporavak baze podataka se koriste za vraćanje baze podataka u korektno stanje. Postoji dva tipa postupaka za oporavak. To su:

- oporavak unapred i
- oporavak unazad.

Oporavak unapred je takav postupak, kod kojeg se poslednja korektna kopija baze podataka menja, saglasno izmenama iz žurnal dataoteke, do stanja neposredno pre nastanka greške. Pri tome, na kopiju se primenjuju naknadne slike onih promena, koje su nastale nakon pravljenja kopije. Oporavak unapred se, obično, primenjuje pri oštećenju većih delova baze podataka i, tokom oporavka unapred, baza podataka se operativno ne koristi.

Oporavak unazad se koristi kada se neka transakcija, koja sadrži niz izmena sadržaja baze podataka, završi neuspešno. Oporavak unazad znači poništavanje svih izmena, koje je posmatrana transakcija izvršila nad bazom podataka. Pri tome se koriste odgovarajuće prethodne slike žurnal dataoteke. Oporavak unazad se izvršava u režimu operativnog korišćenja baze podataka.

3.5.5. Efikasnost

Pod efikasnim korišćenjem baze podataka, na ovom mestu će se smatrati kako brz razvoj i implementacija programa za korišćenje baze podataka, tako i brza selekcija i manipulisanje sadržajem baze podataka. Savremeni SUBP poseduju određene mehanizme za

realizaciju tih zadataka. Za brz razvoj i implementaciju programa, korisnicima stoje na raspolaganju:

- neproceduralni jezici i
- sledeći mehanizmi:
  - ograničenja,
  - okidači,
  - procedure baze podataka i
  - nezavisnost programa i podataka.

Za obezbeđenje efikasne selekcije i manipulisanja sadržajem baze podataka, savremeni SUBP koriste:

- optimizaciju upita,
- metode pristupa i
- različita rešenja zaključavanja.

3.5.5.1. *Produktivnost razvoja i implementacije programa*

Neproceduralnim jezicima su posvećeni drugi delovi ove knjige. Na ovom mestu će određena pažnja biti posvećena mehanizmima za povećanje produktivnosti rada programera.

Ograničenja su pasivni mehanizmi, koje SUBP koristi da spreči dovođenje sadržaja baze podataka u koliziju sa pravilima ponašanja i poslovanja u realnom sistemu. Da bi SUBP mogao da aktivira te mehanizme, uslovi integriteta, kao što su:

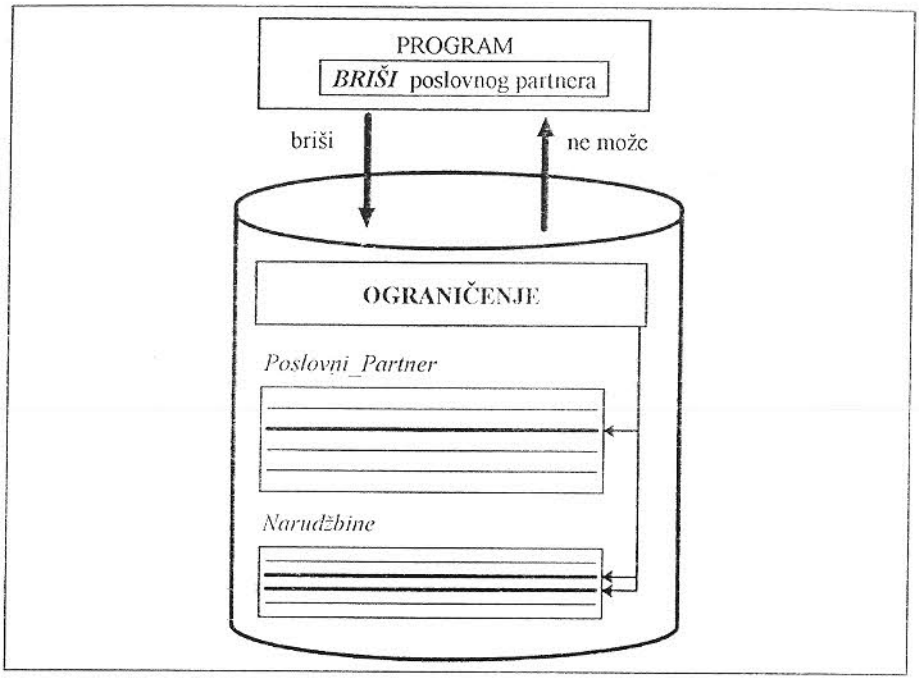
- integritet domena,
- jedinstvena vrednost ključa i
- kardinalitet tipa poveznika,

mu se saopštavaju putem naredbi jezika za opis podataka. Ograničenja su pasivni mehanizmi, jer se aktiviraju tek kada korisnik pokuša da unese podatke, koji narušavaju uslove integriteta.

Naravno, sva ograničenja se mogu ugraditi i u aplikativne programe, koji koriste bazu podataka. Međutim, to rešenje je mnogo lošije od onog kada se ograničenja ugrađuju u bazu podataka, iz dva razloga. Prvi leži u činjenici da sprovođenje uslova integriteta putem programa znači da se odgovornost za integritet baze podataka prenosi na programe (i programere). Na taj način se isti zadatak višestruko ponavlja i uvek ostaje nedoumica da li je baš svugde korektno sproveden. U takvim uslovima, o izmenama uslova integriteta baze podataka nema ni smisla govoriti, jer bi te izmene zahtevale intervencije u velikom broju programa. Drugi razlog je da se baza podataka može koristiti kako putem aplikativnih programa, tako i putem takozvanih "ad hoc" programa, koji se interaktivno kreiraju i odmah izvršavaju. Ako se pretpostavi da je sprovođenje uslova integriteta putem aplikativnih programa, koji se pažljivo pišu, prevode i optimiziraju, moguće, to upošte ne važi za "ad hoc" programe. Kada se saopšte SUBP, ograničenja postaju centralni mehanizam, koji ne može da zaobide nijedan program.

Primer 3.14. Slika 3.9 ilustruje situaciju, kada je SUBP saopšteno pravilo "u bazi podataka ne mogu postojati podaci o narudžbini, ako ne postoje podaci o poslovnom partneru, koji je tu narudžbinu izvršio". Ako program pokuša da izbriše podatke o nekom

poslovnom partneru, koji ima evidentirane narudžbine, SUBP će poslati programu poruku da to brisanje ne može da se izvrši. □



Slika 3.9.

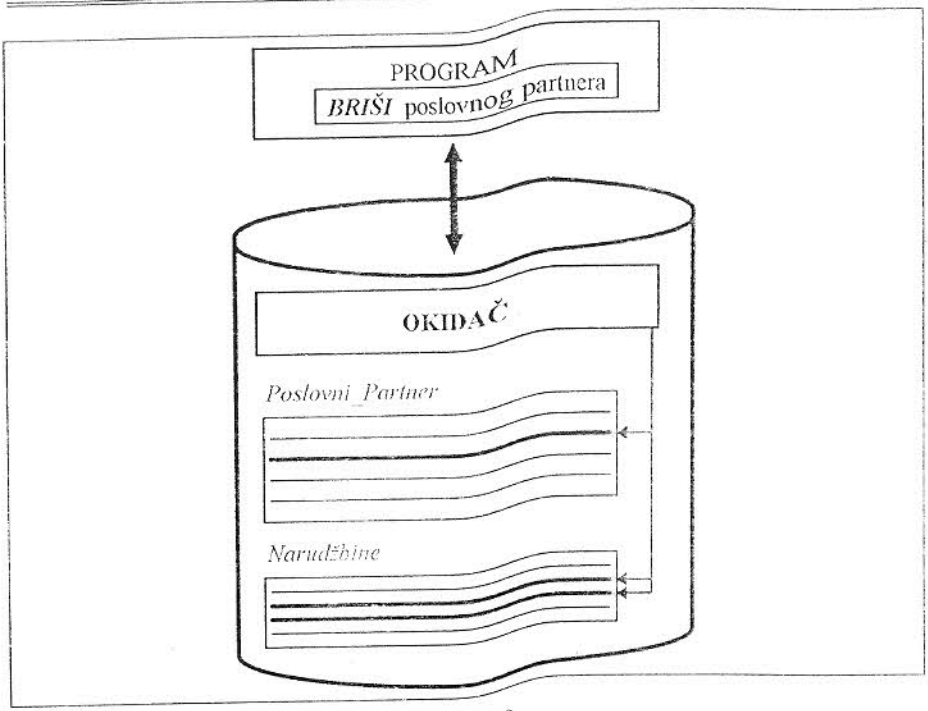
Okidač je programska procedura, povezana sa skupom pojava jednog tipa entiteta u bazi podataka. Te procedure se aktiviraju putem određenih događaja, kao što su:

- upis nove pojave,
- brisanje ili
- modifikacija postojeće pojave tipa entiteta.

Rad okidača je van kontrole programa, a njegovo izvršenje je obavezno. SUBP inicira izvršavanje okidača, čim nastupi događaj koji ga aktivira. I okidači su centralni mehanizmi, koje ne može zaobići nijedan program.

Primer 3.15. Slika 3.10 ilustruje situaciju, kada je okidač napravljen tako da, kada se iz programa pokušaju brisati podaci o jednom poslovnom partneru, okidač izbriše i podatke o svim njegovim porudžbinama. □

Okidači su aktivni mehanizmi, koji održavaju propisane odnose između podataka o različitim entitetima. Menjaju podatke o jednim entitetima, kada se menjaju podaci o drugim. Ni njihovog postojanja korisnici baze podataka nisu svesni.



Slika 3.10.

Za razliku od ograničenja i okidača, *procedure baze podataka* se ne sprovode automatski. To su programi, koji su po pravilu:

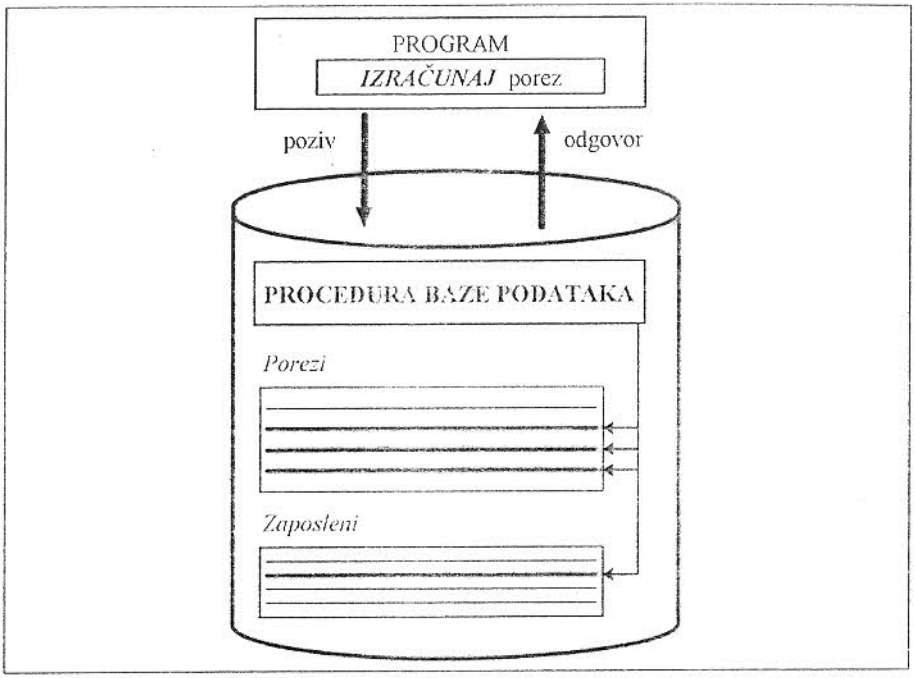
- kompleksni,
- prekompilirani i
- optimizirani.

Čuvaju se centralno, u bazi podataka. Korisnički programi ih, po potrebi, pozivaju da bi ih primenili na deo transakcije.

Primer 3.16. Na slici 3.11 je prikazan slučaj, kada program poziva proceduru baze podataka za izračunavanje poreza. Ta procedura koristi podatke o zaposlenima i o poreskim stopama, da bi izračunala porez. □

Okidači su namenjeni za aktivno sprovođenje uslova integriteta baze podataka, a ograničenja za pasivnu kontrolu. Procedure baze podataka obezbeđuju centralnu definiciju kompleksnih programskih kodova. Sve to dovodi do:

- povećanja produktivnosti programera najmanje za 30 posto,
- neuporedivo bolje kontrole baze podataka, nego kada se ti mehanizmi ugrađuju u programe repetitivnim pisanjem istog koda.



Slika 3.11.

Najveći efekat se postiže u održavanju programa, jer je centralnim definisanjem mehanizama:

- smanjen obim programskog koda, a
- promene u mehanizmima ne utiču na programe.

Nezavisnost programa i podataka, kako je opisana u tački 3.2 ovog poglavlja, stvarno je realizovana tek u relacionim sistemima za upravljanje bazam podataka. Mnoge izmene baze podataka, kao što su:

- dodavanje novog obeležja u opis tipa entiteta,
- dodavanje novog tipa entiteta u šemu baze podataka ili
- izmena fizičke strukture baze podataka,

uopšte ne utiču na programe. Čak šta više, te izmene se vrše dinamički, tokom izvršavanja programa. Takođe, zbog tih izmena, bazu podataka ne treba reorganizovati kopiranjem i ponovnim punjenjem. Najveći efekti se postižu u održavanju programa, jer je, zbog centralnog definisanja mehanizama:

- smanjen obim programskog koda, a
- izmene u mehanizmima ne utiču na programe.

3.5.5.2. *Performanse korišćenja baze podataka*

Kao što je već rečeno, optimizator upita, metode pristupa i postupci zaključavanja značajno utiču na performanse korišćenja baze podataka.

Optimizator upita predstavlja suštinsku komponentu za obezbeđenje performantne obrade relacione baze podataka. Potreba njegovog postojanja je posledica deklarativnosti manipulacionog jezika relacionog modela podataka. U tom jeziku, programer, pretežno koristeći intenzionalni opis baze podataka, definiše koji deo baze podataka treba da se selektuje, a zadatak je SUBP da, poznajući:

- fizičku strukturu baze podataka,
- brojeve pojava tipova entiteta i poveznika i
- raspodelu vrednosti obeležja,

odredi kako taj zahtev treba da se, na efikasan način, izvrši. Postoje tri osnovne vrste optimizatora. To su:

- sintaksni,
- zasnovan na ceni i
- statistički.

Sintaksni optimizator upita je zasnovan na strukturi samog upita, definisanog putem naredbi upitnog jezika. Tu strukturu određuje programer. Međutim, sa logičke tačke gledišta isti upit, može se izvršavati sa veoma različitim vremenima, u zavisnosti od redosleda navođenja obeležja, operatora i uslova. Takvo rešenje optimizatora upita zahteva od programera da, u stvari on, vodi računa o optimalnosti upita.

Optimizator zasnovan na ceni koristi restriktivnost operatora poređenja i podatke o brojevima pojava tipova entiteta i poveznika. Svaki operator poređenja ima svoju ocenu restriktivnosti ("=" je restriktivnije od "≥"). Kombinovanjem ocene restriktivnosti operatora sa brojevima pojava tipova entiteta i poveznika, na koje se operatori primenjuju, dobija se cena "plana izvršavanja" upita. SUBP analizira više mogućnih planova izvršavanja za dati upit, a realizuje najjeftiniji. Ovakvi optimizatori upita daju dobre rezultate u praksi.

Statistički optimizator upita radi na sličnom principu kao i optimizator zasnovan na ceni, ali koristi još i histograme sa raspodelom vrednosti ključa za donošenje odluke o najboljem planu izvršenja upita. Statistički optimizatori predstavljaju najbolje rešenje, ali je za njihovu primenu potrebno povremeno vršiti inoviranje histograma sa raspodelom vrednosti ključa.

Kada su *metode pristupa* u pitanju, SUBP ili koriste usluge upravljača datotekama operativnog sistema, ili imaju svoje metode pristupa. Svi relacioni SUBP podržavaju:

- serijsku organizaciju podataka (*pile*) i
- indeksnu organizaciju sa B - stablom,

a neki SUBP podržavaju još i:

- rasutu (*hash*) i
- indeks - sekvencijalnu organizaciju.

Bitan uticaj na performanse korišćenja baze podataka u višekorisničkom radu ostvaruje i veličina objekta *zaključavanja*. Najbolje performanse se postižu, kada se zaključavanje vrši na nivou pojave tipa entiteta ili čak samo dela pojave.

3.5.6. Distributivnost

Potreba efikasnog upravljanja i korišćenja distribuirane baze podataka diktira potrebu postojanja mehanizama, kao što su:

- distribuirani rečnik,
- dvofazni komit,
- automatski replikator,
- lokacijska transparentnost,
- klijent \ server arhitektura.

Sva ova pitanja su detaljno obrađena u drugim delovima bilo ove, bilo knjige "Principi projektovanja baza podataka".

3.5.7. Arhitektura sistema baze podataka

Na slici 3.12 je nacrtana struktura odnosa sistema za upravljanje bazom podataka. Najvažnije delove ovog sistema pretstavljaju:

- procesor upitnog jezika,
- procesor opisa šeme,
- upravljač baze podataka i
- rečnik.

Projekat šeme baze podataka, opisan putem takozvanog jezika za opis podataka, uvodi se u ***procesor opisa šeme***. Procesor opisa šeme je prevodilac (kompajler) jezika za opis podataka. Rezultat prevođenja je interni opis baze podataka, smešten u rečnik. Prevođenje projekta šeme u opis baze podataka se vrši relativno retko, jedanput na početku korišćenja baze podataka i, povremeno, prilikom retkih modifikacija projekta šeme baze podataka. U velikim višekorisničkim bazama podataka, izmena šeme je zadatak administratora baze podataka, kao i poslovi izmene podšema (pogleda), ili davanje privilegija (autorizacije) korisnicima za pristup bazi podataka. Autorizaciona tabela se, takođe, smešta u rečnik.

Na slici 3.12 je prikazan i ***procesor upita***, u koji se uvode programi za manipulisanje podacima iz dva izvora. Te izvore pretstavljaju:

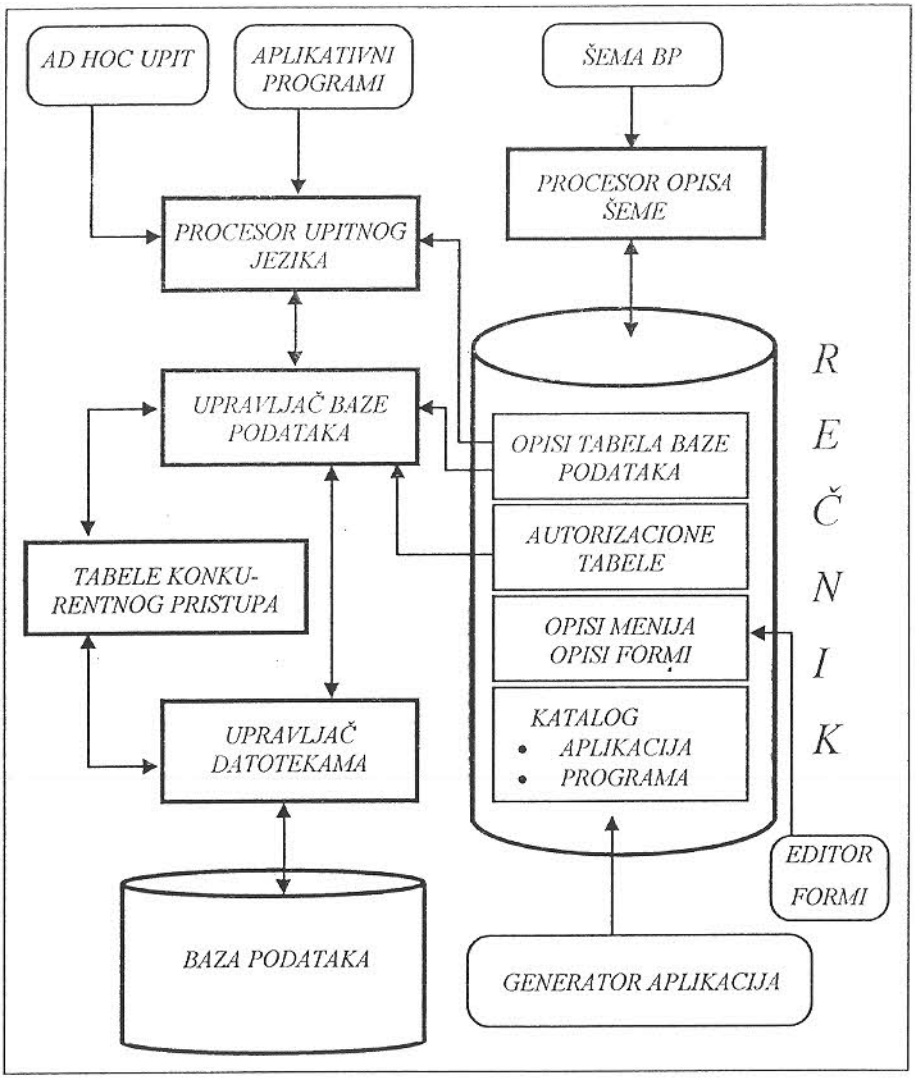
- korisnički upiti, koji se postavljaju direktno putem terminala ili nekog drugog interaktivnog radnog mesta i
- aplikativni programi, koji sadrže naredbe za selekciju i manipulisanje bazom podataka.

Prevodilac jezika domaćina ne predstavlja sastavni deo SUBP.

Zadaci ***upravljača baze podataka*** su da:

- preuzima naredbe od procesora upitnog jezika i prevodi ih sa nivoa apstrakcije opisa podšeme ili šeme u operacije na datotekama, odnosno fizičkoj strukturi baze podataka,
- proverava pravo pristupa korisnika elementima intenzionalnog opisa baze podataka,
- formira i održava specijalnu tabelu sa podacima o zaključavanju objekata,

- vodi borbu protiv pojave izgladnjavanja transakcija i uzajamnog zaključavanja,
- upravlja vođenjem žurnal datoteke i
- detektuje razne anomalne situacije, kao što je narušavanje integriteta baze podataka, ili potreba njenog oporavka.



Slika 3.12.

Kada je reč o uslovima integriteta baze podataka, poželjno je da jezik za opis podataka poseduje mogućnosti za eksplicitno definisanje ograničenja u opisu same šeme baze podataka. Na taj način, sprovođenje tih ograničenja, pa prema tome i održavanje baze podataka u konzistentnom stanju, postaje zadatak SUBP.

Na slici 3.12 je prikazan i upravljač datotekama. On može biti ili sastavni deo SUBP, ili je reč o opštem upravljaču datotekama operativnog sistema. Njegov je zadatak da, na osnovu zahteva upravljača bazom podataka, prenosi delove baze podataka u operativnu memoriju.

Rečnik podataka je baza podataka sa podacima o bazi podataka i njenim korisnicima. Rečnikom, kao bazom podataka, takođe upravlja SUBP, tako da se može koristiti putem upitnog jezika. Različiti SUBP poseduju rečnike sa različitim mogućnostima. Analiza tih mogućnosti izlazi izvan okvira ovog teksta. Na ovom mestu će samo biti ukazano da rečnik sadrži podatke o:

- logičkoj i fizičkoj (internoj) strukturi baze podataka,
- pogledima,
- korisnicima i njihovim pravima pristupa,
- aplikativnim programima i njihovim komponentama.*)

*) Komponentama aplikativnog programa se na ovom mestu smatraju: procedure, meniji, ekranske i štampane forme.

4. Glava

Relacioni model podataka

Jedan od prvih radova, posvećen modelu podataka, čiju osnovu čine relacije i njihova reprezentacija putem dvodimenzionalnih tabela, napisao je Codd [C1]. Pored njega, značajan doprinos razvoju ovog modela dao je niz autora, kao na primer Beeri, Bernstein, Fagin, Maier, Rissanen, Ullman i mnogi drugi. Motiv za definisanje relacionog modela podataka predstavljali su nedostaci, uočeni pri korišćenju do tada poznatih modela podataka, u koje su spadali hijerarhijski i mrežni modeli podataka. Zaključeno je da su uočeni nedostaci posledica tri osnovna uzroka. To su:

- nepostojanje jasne granice između logičkih i fizičkih aspekata baze podataka,
- strukturalna kompleksnost i
- navigacioni jezik za manipulisanje podacima.

4.1. Koncepcija relacionog modela podataka

Saglasno tome, pri razvoju relacionog modela podataka postavljena su tri odgovarajuća cilja:

- Prvi i najvažniji cilj bilo je uvođenje jasne granice između logičkih i fizičkih aspekata baze podataka, kako u domenu projektovanja, tako i u domenu korišćenja.
- Drugi cilj je bio da se modelu da strukturalna jednostavnost i da se obezbedi jednoobraznost shvatanja i pogleda na podatke - kako programera tako i krajnjih korisnika.
- Treći cilj je bio uvođenje deklarativnog jezika za definisanje i korišćenje baze podataka. To je značilo da jezik treba da poseduje osobinu neproceduralnosti i mogućnost definisanja operacija nad skupovima podataka, što dovodi do značajnog povećanja pro-

duktivnosti programera, a predstavlja i preduslov za neposredno korišćenje baze podataka od strane krajnjih korisnika.

4.1.1. Nezavisnost

Prema Codd-ovim rečima: "Najvažniji motiv istraživačkog rada, koji je rezultovao u relacionom modelu, bio je postizanje oštre i jasne granice između logičkih i fizičkih aspekata upravljanja bazom podataka". Taj cilj istraživanja Codd je nazvao nezavisnošću podataka. U dotadašnjim SUBP, opis podataka je bio preplavljen informacijama o karakteristikama fizičke strukture podataka. U svaki aplikativni program bila je ugrađena informacija o fizičkoj strukturi. Za takve programe se kaže da zavise o fizičkoj strukturi, jer ako se fizička struktura baze podataka izmeni, moraju se menjati i svi programi na koje ta izmena utiče, a to je, po pravilu, skupo.

Pojam fizičke strukture podataka odnosi se na sve aspekte takozvane "interne reprezentacije podataka", kao i na mehanizme pristupa podacima. Spomenuti mehanizmi su: raspodela slogova po zonama baze podataka, fizički redosled i grupisanje slogova (po stranicama)*), algoritam transformacije ključa u adresu, lanci slogova povezanih pokazivačima, indeksi (stabla traženja), hijerarhijski redosled slogova (u setovima) i slično.

Ključ za razumevanje osnovne ideje relacionog modela leži u razumevanju pojma nezavisnosti podataka, koja se postiže potpunim razdvajanjem oblika u kojem se podaci prezentiraju programu ili korisniku (logički aspekt), od oblika u kojem se ti podaci memorišu u bazi podataka (fizički aspekt). Postavljanjem jasne granice između ova dva oblika i delegiranjem zadataka prevođenja jednog oblika u drugi sistemu za upravljanje bazom podataka, eliminiše se potreba ugradnje bilo kakve informacije o fizičkoj strukturi podataka u programe.

U relacionom modelu je, kao najveća logička organizaciona jedinica podataka, odabrana *n* - arna relacija. S obzirom na matematičku definiciju, relacija je skup *n* - torki (ili kratko torki), pri čemu svaka komponenta torke predstavlja vrednost iz jednog skupa, takozvanog "domena obeležja". Broj *n* je broj obeležja u opisu relacije na nivou apstrakcije obeležja.

Ovaj opis se naziva šemom relacije i predstavlja imenovanu dvojku sa oznakom $N(R,\mathcal{K})$, gde je: N naziv relacije, R skup obeležja, a $\mathcal{K}$ skup uslova integriteta relacije. Uslovi integriteta iz $\mathcal{K}$ ukazuju na neophodne osobine torki relacije. Često se $\mathcal{K}$ svodi samo na ključ šeme relacije. Tada je reč o ograničenju da svaka torka mora imati jedinstvenu vrednost ključa.

Primer 4.1. Šemi relacije *Fakultet*({*FAK*, *NAZ*, *BPI*}, {*FAK*}), gde je *FAK* oznaka fakulteta, *NAZ* naziv fakulteta, *BPI* broj informatičkih predmeta i gde je *FAK* ključ (primarni), odgovara relacija:

$$r(Fakultet) = \{(PMF, Matematički, 7), (EKF, Ekonomski, 4), (ETF, Elektrotehnički, 9), (MAF, Mašinski, 7)\}.$$

*) U bazama podataka se jedinica razmene podataka između operativne i eksterne memorije naziva stranicom. Stranica je pojam analogan pojmu bloka kod datoteke.

Sve torke relacije *r(Fakultet)* zadovoljavaju ograničenje definisano ključem. Torka (*EKF, Elektronski , 8*) se ne može upisati u ovu relaciju, jer bi bio narušen uslov integriteta. □

Šemu relacione baze podataka predstavlja imenovana dvojka $N(S, I)$, gde je S skup šema relacija, a I skup uslova integriteta baze podataka (referencijalni integriteti, na primer). Relacionu bazu podataka sačinjava skup relacija.

Definisanje relacija i šema relacija kao komponenata logičkog aspekta relacione baze podataka, izbacilo je u prvi plan pitanje definisanja postupka za selekciju podataka iz baze podataka. Ovaj problem je rešen uvođenjem pojma asocijativnog adresiranja. Svaki podatak se u relacionoj bazi podataka jednoznačno adresira putem naziva šeme relacije, naziva obeležja i vrednosti ključa. Slično, da bi se izvršila selekcija torki sa zajedničkom osobinom iz relacije, definiše se naziv relacije i ta zajednička osobina. Asocijativnim adresiranjem se prepušta SUBP-u da odredi gde će podatak biti memorisan i kako će biti pronađen.

4.1.2. *Strukturalna jednostavnost*

Da bi se obezbedila strukturalna jednostavnost modela, kao reprezent relacije je usvojena dvodimenzionalna tabela, jer predstavlja lako razumljiv pojam i za programera i za krajnjeg korisnika. Prilikom korišćenja tabele kao reprezenta relacije, tabela nosi naziv relacije, a vrsta sa nazivima kolona (zaglavlje tabele) predstavlja skup obeležja šeme relacije. Nazivi kolona tabele predstavljaju obeležja, a same kolone sadrže elemente domena odgovarajućih obeležja. Vrste tabele predstavljaju n - torke. Redosled uvođenja kolona i vrsta u tabelu je proizvoljan i nevažan.

Fakultet

FAK	*NAZ*	*BIP*
FIL	*Filozofski*	*1*
PMF	*Matematički*	*7*
ETF	*Elektrotehn.*	*9*
EKF	*Ekonomski*	*4*
MAF	*Mašinski*	*7*

Projektant

MBR	*IME*	*PRZ*	*FAK*
m_3	*Ivo*	*Ban*	*PMF*
m_1	*Ana*	*Tot*	*MAF*
m_4	*Ana*	*Ras*	*FIL*
m_8	*Aca*	*Pap*	*ETF*
m_6	*Ivo*	*Ban*	*EKF*
m_5	*Eva*	*Tot*	*ETF*

Slika 4.1.

Primer 4.2. Na slici 4.1 prikazane su tabelarne predstave relacija *Fakultet* i *Projektant*. Obeležja ključeva šema relacija su podvučena u zaglavlju tabele. □

Usvajanje relacije, odnosno tabele, kao osnovne organizacione jedinice podataka na logičkom nivou, iniciralo je problem predstavljanja informacije o odnosima između podataka u različitim tabelama. U mrežnom modelu, odnosi se predstavljaju putem lanaca

slogova povezanih pokazivačima. Međutim, u modelu podataka sa jasno razdvojenim prezentacionim od formata memorisanja podataka, lanci sa pokazivačima (ako uopšte postoje) nevidljivi su za korisnika. Opredeljenje za tabelu, kao jedinu veću organizacionu jedinicu podataka, dovelo je i do opredeljenja da se informacija o odnosima između podataka predstavi unutar postojeće tabele ili putem nove tabele. Prvo rešenje se koristi u slučaju odnosa $1:1$ i $1:N$, a realizuje se uvođenjem pojma stranog ključa. Drugo rešenje se koristi u slučaju odnosa $M:N$ između podataka u tabelama. Treba zapaziti da se i u drugom slučaju povezivanje zasniva na stranom ključu.

Primer 4.3. Tabele *Fakultet* i *Projektant* na slici 4.1 su povezane uvođenjem obeležja *FAK* u tabelu *Projektant*. Obeležje *FAK* predstavlja ključ tabele *Fakultet* i strani ključ u tabeli *Projektant*. □

4.1.3. *Jezik podataka*

Rešenje problema predstavljanja podataka i njihovih veza putem tabela, iniciralo je potrebu razvoja tehnike za njihovo korišćenje. U mrežnom modelu, programer koristi pristupne mehanizme i lance sa pokazivačima. Takva navigacija ne samo da nije moguća u relacionom modelu, već nije ni poželjna, jer vodi zavisnosti programa od fizičke strukture i zahteva proceduralno programiranje. Obrazlažući opredeljenje za relacioni model, Codd je napisao: "Relacioni pogled na podatke obezbeđuje postupak za opis samo prirodne strukture podataka, bez ikakve informacije o mašinskoj reprezentaciji. Saglasno tome, ostvareni su preduslovi za definisanje jezika podataka visokog nivoa, koji će dati maksimalnu nezavisnost između programa s jedne strane i mašinske reprezentacije i organizacije podataka sa druge strane."

Za potrebe razvoja tog jezika, definisana su dva alata, ekvivalentna s obzirom na mogućnosti definisanja operacija manipulisanja podacima. To su: relaciona algebra i relacioni račun. Pošto su relacije skupovi, definisan je, u okviru relacione algebre, niz skupovnih operatora za manipulisanje podacima. Operator, koji više od bilo kog drugog, odvaja relacione od nerelacionih jezika, naziva se ***SPOJ*** (***JOIN***). Služi za formiranje nove tabele od podataka smeštenih u dve tabele. Pri tome koristi informaciju, realizovanu putem stranog ključa, o vezama između vrsta u te dve tabele.

Svakoj operaciji relacione algebre odgovara ekvivalentni logički iskaz relacionog računa. Jezik podataka ***SQL*** (Structured Query Language) zasnovan je na relacionom računu, a sintaksa i semantika su mu definisani standardom [AS]. SQL predstavlja interaktivni jezik za definisanje podataka, ažuriranje podataka i postavljanje upita. Sintaksa i semantika ovog jezika su prilagođene krajnjem korisniku. Bitne karakteristike SQL-a su deklarativnost i rad sa skupovima. Rezultat jednog, ad hoc postavljenog upita, na primer, je skup torki (nova relacija), koje zadovoljavaju definisane kriterijume. Osnovni upitni blok SQL-a ima oblik:

SELECT	*lista obeležja*
FROM	*lista relacija*
WHERE	*kvalifikacioni izraz.*

Rezultat upita je nova tabela, čije kolone su definisane listom obeležja (atributa) iza reči *SELECT*. U listi relacija (iza reči *FROM*), pobrojane su tabele iz kojih se uzimaju podaci, a kvalifikacioni izraz iza reči *WHERE* predstavlja logički izraz koji određuje osobine vrsta u rezultantnoj tabeli.

Primer 4.4. Ako se, na osnovu tabela sa slike 4.1, žele dobiti imena i prezimena projektanata, koji su na fakultetu slušali više od pet informatičkih predmeta, odgovarajući SQL izraz bi glasio:

SELECT *IME, PRZ, BIP*
FROM *Fakultet, Projektant*
WHERE *BIP > 5* ***AND*** *Projektant.FAK = Fakultet.FAK*

a odgovor bi bio tabela na slici 4.2. □

IME	*PRZ*	*BIP*
Ivo	*Ban*	*7*
Ana	*Tot*	*7*
Aca	*Pap*	*9*
Eva	*Tot*	*9*

Slika 4.2.

SQL nije predviđen za definisanje kompletnih postupaka obrade podataka. Zato postoji mogućnost da se njegove komande za rad sa bazom podataka ugrade u programe pisane u nekom jeziku treće ili četvrte generacije. Ukratko, proceduralni program "vidi" rezultat izvršenja SQL izraza kao sekvencijalnu datoteku i čita je slog po slog. Bitno je zapaziti da čitanje sekvencijalne datoteke predstavlja neuporedivo lakši zadatak od navigacije kroz mrežnu bazu podataka.

Opisane osnovne karakteristike relacionog modela podataka i činjenica da relacioni SUBP poseduju čitav niz savremenih rešenja, kakvo je, nprimer i aktivni rečnik podataka, omogućili su povećanje produktivnosti projektanata, programera i krajnjih korisnika u poslovima realizacije i održavanja informacionih sistema.

4.2. *Strukturalna komponenta relacionog modela podataka*

Na nivou ekstenzije, koncepte relacionog modela podataka predstavljaju: domen obeležja, torka, relacija i pojava baze podataka, dok na nivou intenzije, osnovne koncepte predstavljaju: obeležje, šema relacije i šema baze podataka. Pri tome, primitivne koncepte predstavljaju samo elemenat domena i obeležje. Svi ostali koncepti izvode se od primitivnih, primenom određenih formalno - matematičkih pravila. Osnove pomenutih pravila i

postupaka definisanja složenijih koncepata date su u narednom tekstu. Treba napomenuti da je pristup definisanju koncepata relacionog modela, prikazan u narednom tekstu, zasnovan na savremenim shvatanjima, prezentiranim, na primer u [M], [PBG] i [U], te se u određenim stavovima razlikuje od klasičnog, datog u [C1].

Polaznu osnovu za definisanje koncepata strukturalne komponente relacionog modela podataka predstavlja skup $\mathcal{U} = \{A_i \mid i = 1,...,m\}$. To je podskup skupa obeležja realnog sistema, ili njegovog dela koji je predmet posmatranja. Skup $\mathcal{U}$ je konačan i ovde se usvaja njegov najčešće korišćen naziv, ***univerzalni*** skup obeležja. Svako obeležje A_i opisuje neku odabranu osobinu realnog sistema, a pridružen mu je domen, u oznaci $dom(A_i)$, iz kojeg uzima vrednosti. Svi skupovi obeležja, koji se koriste u daljem tekstu, predstavljaju podskupove ovog univerzalnog skupa.

U daljem tekstu se koriste određene konvencije u notaciji, uobičajene u teoriji relacionih baza podataka. Oznaka XY se koristi kao skraćena notacija za uniju, ne nužno disjunktnih skupova X i Y. Takođe, umesto oznake $\{A\}$ za jednočlani skup, koristi se skraćena notacija A.

4.2.1. *R - vrednost*

Neka je dat skup obeležja $R \subset \mathcal{U}$, odnosno

$$R = \{A_i \mid i = 1,...,k\}.$$

Jedna ***R - vrednost***, u oznaci $t[R]$ ili kratko t ako je R poznato, je funkcija koja preslikava svako obeležje iz R u odgovarajuću vrednost, odnosno

$$t : R \rightarrow Dom,$$

gde je $Dom = \bigcup_{i=1}^{k} dom(A_i)$ i važi $t(A_i) \in dom(A_i)$, za $i = 1,...,k$. Pridruživanje $t(A_i) = a_i$, gde je

$$a_i \in dom(A_i),$$

naziva se A_i - vrednošću i često se opisuje kao par (A_i, a_i). Preslikavanje t se opisuje navođenjem skupa $\{(A_i, a_i) \mid A_i \in R\}$. Treba zapaziti da je redosled nabrajanja obeležja, pri definisanju R - vrednosti, nevažan. Preslikavanja ove vrste označavaju se malim slovima t, u, v.

Primer 4.5. Neka je $R = \{A, B, C\}$, $dom(A) = \{a_1, a_2, a_3\}$, $dom(B) = \{b_1, b_2\}$, $dom(C) = \{c_1, c_2\}$. Tada $t = \{(A, a_2), (B, b_1), (C, c_2)\}$ i $u = \{(A, a_1), (B, b_2), (C, c_2)\}$ predstavljaju dve različite, a t i $v = \{(C, c_2), (A, a_2), (B, b_1)\}$ dve iste R - vrednosti. □

4.2.2. *Restrikcija R - vrednosti*

Neka je $X \subseteq R$, a $t[R]$ jedna R - vrednost. ***Restrikcijom R - vrednosti*** $t[R]$ na skup obeležja X, naziva se takva X - vrednost $u[X]$, u kojoj je svakom obeležju A iz X pridružena ista vrednost kao i u R - vrednosti $t[R]$. Restrikcija R - vrednosti t na $X \subseteq R$ se označava sa $t[X]$, pošto je originalna oblast definisanosti R poznata iz konteksta. Pošto se R - vrednost t definiše kao $\{(A_i, a_i) \mid A_i \in R\}$, restrikciju $t[X]$ predstavlja skup $\{(A_i, a_i) \mid A_i \in X\}$.

Primer 4.6. Neka je R, t i u dato kao u primeru 4.5, a $X = \{A, B\}$. Tada je:

$t[X] = \{(A, a_2), (B, b_1)\}$ i $u[X] = \{(A, a_1), (B, b_2)\}$. □

4.2.3. *Relacija*

U relacionom modelu podataka se relacijom nad skupom obeležja R naziva svaki konačan skup R - vrednosti. U daljem tekstu će se relacija nad R označavati sa $r(R)$, ili kratko r, ako je R poznato. Ako su svi domeni konačni skupovi, nad jednim skupom obeležja R se može definisati konačan, ali veoma velik broj različitih relacija. Može se smatrati da je prelazak iz jedne u drugu relaciju posledica upisa novih, brisanja i modifikacije postojećih R - vrednosti relacije.

Pojam relacije (u smislu relacionog modela) u bliskoj je međuzavisnosti sa pojmom relacije u matematičkom smislu. U matematici se relacija definiše kao podskup Dekartovog proizvoda $dom(A_1) \times ... \times dom(A_k)$, odnosno kao skup uređenih k - torki $(a_1,..., a_k)$, gde je, za svako $i \in \{1,...,k\}$, $a_i \in dom(A_i)$.

Uređenje obeležja $A_1,...,A_k$ je u slučaju matematičke relacije značajno, jer nosi informaciju o semantici elemenata a_i k - torke. Dok je semantika $a_i = t(A_i)$ u definiciji R - vrednosti eksplicitna, semantika a_i u k - torci je implicitna i zavisi od uređenja obeležja $A_1,..., A_k$.

U daljem tekstu će se uvek pod pojmom relacije nad skupom obeležja R podrazumevati konačan skup R - vrednosti, jer odražava suštinu pojma relacije u relacionom modelu. Međutim, kada se putem indeksa izvrši jedno numerisanje obeležja, dolazi se do pojednostavljenja notacije. Tada se obeležje A_i opisuje kao i - to obeležje (po redu), a k - torka, ili kratko torka, $(a_1,..., a_k)$ se može koristiti za skraćeno označavanje R - vrednosti $\{(A_i, a_i) \mid A_i \in R\}$. U tom smislu se i R - vrednosti nazivaju torkama.

Skup svih relacija nad skupom obeležja R, označava se sa $SAT(R)$. Svaka relacija iz skupa $SAT(R)$ zadovoljava ograničenje $t[R]$, u smislu da se svakom a_i iz $t = (a_1,..., a_n)$ pridružuje vrednost iz $dom(A_i)$, za $A_i \in R$.

Tabela predstavlja najpogodnije sredstvo za reprezentovanje relacije. Zaglavlje tabele sadrži obeležja skupa R, a vrste R - vrednosti, odnosno torke. Predstavljanje relacije putem tabela sa obeležjima u zaglavlju, u potpunoj je saglasnosti sa definicijom relacije kao skupa R - vrednosti. Naravno redosled nabrajanja vrsta u tabeli je nevažan.

Primer 4.7. Na slici 4.3 su prikazane dve tabelarne predstave jedne iste relacije nad skupom obeležja $R = \{A, B, C\}$. Pri tome je $dom(A) = \{a_1, a_2, ...\}$, $dom(B) = \{b_1, b_2, ...\}$ i $dom(C) = \{c_1, c_2, ...\}$. Tabele predstavljaju istu relaciju bez obzira na izmenjene redoslede navođenja obeležja i torki. Primenjeni tabelarni način prikazivanja relacije sa navođenjem obeležja iz R u zaglavlju i elemenata domena obeležja u odgovarajućim kolonama, odgovara definisanju relacije kao skupa preslikavanja. □

$r(ABC) =$

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_1	c_2

$r(ABC) =$

C	A	B
c_2	a_2	b_1
c_1	a_1	b_1
c_1	a_1	b_2

Slika 4.3.

4.2.4. Projekcija relacije na skup obeležja

Neka je $X \subseteq R$, a r jedna relacija nad R. Projekciju ili restrikciju relacije r na skup obeležja X predstavlja relacija $r[X]$ definisana kao $r[X] = \{t[X] \mid t \in r(R)\}$. Projekcija relacije $r(R)$ na $X \subseteq R$, najčešće se označava na sledeći način

$$\pi_X(r(R)).$$

Primer 4.8. Neka je $X = \{A, B\}$. Projekciju relacije $r(ABC)$ iz primera 4.7 na skup obeležja X predstavlja relacija prikazana na slici 4.4. □

$r[AB] =$

A	B
a_1	b_1
a_1	b_2
a_2	b_1

Slika 4.4.

4.2.5. Šema relacije

Šema relacije je imenovana dvojka, u oznaci $N(R, C)$, gde je N naziv šeme relacije, $R \subseteq \mathcal{U}$ skup obeležja, a C skup ograničenja. Naziv šeme relacije N predstavlja neformalnu komponentu definicije, pošto se odnosi na realni svet i opisuje njenu semanti-

ku u prirodnom jeziku. Skup ograničenja C opisuje odnose između elemenata domena obeležja iz R.

Saglasno pretpostavci o postojanju šeme univerzalne relacije*), svako obeležje univerzalnog skupa obeležja poseduje samo jednu ulogu, te se umesto putem posebnog naziva N, šema relacije može jednoznačno identifikovati putem skupa obeležja R. Sam naziv N šeme relacije postaje nepotreban. U daljem tekstu će se često, ali ne uvek, šema relacije identifikovati putem svog skupa obeležja R. Ta se pogodnost neće koristiti kada je skup R veliki, ili zadat putem mnemonika, ili kada se želi istaći semantika šeme relacije. U tim slučajevima će se koristiti naziv šeme relacije, što ne znači odstupanje od pretpostavke o postojanju šeme univerzalne relacije.

Primer 4.9. Posmatra se klasa entiteta *Student* sa obeležjima: {*BRI, IME, PRZ, BPI*}. Pri tome je uočeno da važe sledeća ograničenja:

(γ_1) svaki student ima broj indeksa i ne postoje dva studenta sa istim brojem indeksa,

(γ_2) broj položenih ispita je veći ili jednak 0 i manji od 50 ($0 \leq BPI < 50$).

Šema relacije, koja predstavlja model ove klase entiteta, imala bi oblik:

$$Student(\{BRI, IME, PRZ, BPI\}, \{\gamma_1, \gamma_2\}). \square$$

Primer 4.10. Posmatraju se klase entiteta: N - nastavnik, S - student i P - predmet. Neka su N, S i P, redom, i oznake za odgovarajuće skupove obeležja koji reprezentuju ove klase entiteta. Između entiteta posmatranih klasa važe sledeći odnosi:

(γ_1) svaki nastavnik predaje najviše jedan predmet,

(γ_2) ako student sluša neki predmet, sluša ga kod samo jednog nastavnika.

Šema relacije, koja bi predstavljala model odnosa između podataka o entitetima posmatranih klasa, imala bi oblik ($\{N, S, P\}, \{\gamma_1, \gamma_2\}$). Saglasno pretpostavci o postojanju šeme univerzalne relacije, za referenciranje na ovu šemu relacije može se koristiti i notacija NPS. $\square$

4.2.6. *Pojava nad šemom relacije*

Šema relacije je dvojka (R, C). Broj različitih relacija nad skupom obeležja $R = \{A_1, \ldots, A_n\}$ jednak je kardinalnom broju partitivnog skupa Dekartovog proizvoda $dom(A_1) \times \ldots \times dom(A_n)$. Ne mora svaka od ovih relacija zadovoljavati sva ograničenja iz skupa C.

Definicija 4.1. Relacija $r(R)$, koja zadovoljava svako ograničenje iz skupa C, predstavlja *pojavu* nad šemom relacije (R, C). Skup svih pojava nad šemom relacije (R, C) obeležava se sa $SAT(R, C)$. $\square$

*) Ova pretpostavka je opisana u petoj glavi.

Definicija 4.2. Ograničenje $\gamma \in C$ šeme relacije (R, C) predstavlja Bulovu funkciju, koja svakoj relaciji nad R pridružuje ili vrednost T (tačan) ili vrednost $\perp$ (netačan). Ako ta funkcija pridruži relaciji $r(R)$ vrednost T, relacija zadovoljava ograničenje γ. □

Primer 4.11. Neka je r jedna relacija nad skupom obeležja $\{BRI, IME, PRZ, BPI\}$ iz primera 4.9. Tada sledeće dve Bulove funkcije reprezentuju dva definisana ograničenja:

(γ_1) svaki student ima broj indeksa i ne postoje dva studenta sa istim brojem indeksa:
$$\gamma_1(r) = \mathsf{T} \equiv (\forall u, v \in r)(u[BRI] = v[BRI] \Rightarrow u = v),$$

(γ_2) broj položenih ispita je veći ili jednak 0 i manji od 50 :
$$\gamma_2(r) = \mathsf{T} \equiv (\forall u \in r)(0 \leq u[BPI] < 50).$$

Na slici 4.5 je prikazana relacija, koja predstavlja, a na slici 4.6 je prikazana relacija koja ne predstavlja pojavu nad šemom relacije *Student*($\{BRI, IME, PRZ, BPI\}$, $\{\gamma_1, \gamma_2\}$). Relacije na obe slike predstavljaju relacije nad skupom obeležja *Student*, ali relacija na slici 4.6 ne zadovoljava ni ograničenje γ_1 (narušava ga prva i treća vrsta), ni ograničenje γ_2 (narušava ga prva vrsta). □

BRI	*IME*	*PRZ*	*BPI*
159	*Ivo*	*Ban*	*13*
013	*Ana*	*Tot*	*00*
113	*Ivo*	*Ban*	*27*

Slika 4.5.

BRI	*IME*	*PRZ*	*BPI*
159	*Ivo*	*Ban*	*51*
013	*Ana*	*Tot*	*00*
159	*Aco*	*Ban*	*27*

Slika 4.6.

Primer 4.12. Neka je r relacija nad skupom obeležja $\{N, P, S\}$ iz primera 4.10. Tada sledeće dve Bulove funkcije reprezentuju dva definisana ograničenja:

(γ_1) svaki nastavnik predaje najviše jedan predmet:
$$\gamma_1(r) = \mathsf{T} \equiv (\forall u, v \in r)(u[N] = v[N] \Rightarrow u[P] = v[P]),$$

(γ_2) ako student sluša neki predmet, sluša ga kod samo jednog nastavnika:
$$\gamma_2(r) = \mathsf{T} \equiv (\forall u, v \in r)(u[SP] = v[SP] \Rightarrow u[N] = v[N]).$$

Na slici 4.7 je prikazana relacija nad $\{N, P, S\}$ koja predstavlja, a na slici 4.8 je prikazana relacija nad $\{N, P, S\}$ koja ne predstavlja pojavu nad šemom relacije ($\{N, P, S\}$, $\{\gamma_1, \gamma_2\}$). Relacija na slici 4.8 ne zadovoljava ograničenje γ_2 . □

Sve relacije koje predstavljaju pojave iste šeme relacije imaju zajedničke osobine. Opis tih zajedničkih osobina predstavlja šema relacije. Šema relacije opisuje statičke, vremenski relativno nepromenljive osobine klase entiteta realnog sveta. Svaka pojava šeme relacije reprezentuje jedno stanje realne klase entiteta, izraženo putem reprezentacije stanja svakog entiteta te klase. Stanje entiteta se reprezentuje putem jedne R - vrednosti odnosno torke.

N	P	S
n_1	p_1	s_1
n_1	p_1	s_2
n_2	p_1	s_3
n_3	p_2	s_1
n_3	p_2	s_3

Slika 4.7.

N	P	S
n_1	p_1	s_1
n_1	p_1	s_2
n_2	p_1	s_1
n_3	p_2	s_1
n_3	p_2	s_3

Slika 4.8.

4.2.7. *Ključ šeme relacije*

Svaka šema relacije poseduje bar jedan ključ.

Definicija 4.3. Skup obeležja $X \subseteq R$ predstavlja ključ šeme relacije (R, C), ako za svako $r \in SAT(R, C)$ važe sledeća dva uslova:

1° $(\forall u, v \in r)(u[X] = v[X] \Rightarrow u = v)$ i

2° $(\forall Y \subset X)(\neg 1°)$. □

U definiciji 4.3, uslov 1° ukazuje na jedinstvenost vrednosti ključa, u smislu da se svakoj torci relacije dodeljuje druga vrednost ključa i da se putem vrednosti ključa može izvršiti jednoznačna identifikacija torke. Uslov 2° ukazuje na minimalnost skupa obeležja ključa.

Jedna šema relacije može imati više od jednog ključa. Svi ključevi jedne šeme relacije čine skup ***ekvivalentnih*** ključeva te šeme relacije. Nazivaju se ekvivalentnim, jer su sa logičke tačke gledišta svi ravnopravni. Jedan od ekvivalentnih ključeva se bira za ***primarni***. Putem vrednosti primarnog ključa se, najčešće, vrši traženje torki u relaciji, a također, poželjno je i da se jedino primarni ključ javlja kao strani ključ u drugim šemama relacija. Jedan od kriterijuma za određivanje primarnog ključa predstavlja navika korisnika baze podataka.

Primer 4.13. Šema relacije *Student*, iz primera 4.9, poseduje samo jedan ključ. To je obeležje *BRI*. Ako se skup obeležja te šeme relacije proširi obeležjem *MBS* (matični broj stanovnika), a skup C ograničenjem:

(γ_3) svaki student poseduje matični broj stanovnika (*MBS*) i ne postoje dva studenta sa istim matičnim brojem stanovnika,

dobija se šema relacije sa dva ključa. Drugi ključ je *MBS*. Prirodno je pretpostaviti da su korisnici fakultetskog infomacionog sistema naviknuti na korišćenje broja indeksa za jednoznačnu identifikaciju studenata i da će *BRI* biti proglašeno za primarni ključ.

Šema relacije *NPS*, iz primera 4.10, predstavlja model odnosa između klasa entiteta *Nastavnik*, *Predmet* i *Student*. Ako se pretpostavi da obeležja N, P i S predstavljaju primarne ključeve, redom, šema relacija *Nastavnik*, *Predmet* i *Student*, tada šema relacije *NPS* poseduje sledeća dva ključa: $K_1 = \{N, S\}$ i $K_2 = \{P, S\}$. K_1 i K_2 su ekvivalentni ključevi šeme relacije *NPS*. □

4.2.8. Šema baze podataka

Definicija 4.4. Šemu relacione baze podataka predstavlja imenovana dvojka $N(S, I)$, gde je N naziv šeme baze podataka, S konačan skup šema relacija, u oznaci $S = \{(R_i, C_i) \mid i = 1,...,n\}$, takvih da je $\mathcal{U} = \bigcup_{i=1}^{n} R_i$, a I predstavlja skup međurelacionih ograničenja i ograničenja, koja su posledica pravila poslovanja. □

Primer 4.14. Neka su N, P, K, U oznake skupova obeležja, karakterističnih, redom, za klase entiteta: *Nastavnik*, *Predmet*, *Katedra* i *Učionica*. Tada šema baze podataka *Fakultet* može sadržati sledeći skup šema relacija:

$$S = \{(\{K, N\}, \{\gamma_1\}), (\{K, P\}, \{\gamma_2\}), (\{N, P, U\}, \{\gamma_3\})\},$$

sa ograničenjima:

(γ_1) svaki nastavnik pripada samo jednoj katedri,
(γ_2) svaki predmet pripada samo jednoj katedri,
(γ_3) ako nastavnik predaje neki predmet, predaje ga uvek u istoj učionici

i sledeći skup ograničenja baze podataka $I = \{i_1, i_2, i_3\}$, gde su:

(i_1) samo nastavnik, koji pripada nekoj katedri, može izvoditi nastavu,
(i_2) samo predmet, koji pripada nekoj katedri se može predavati,
(i_3) svaki nastavnik predaje samo one predmete, koji pripadaju njegovoj katedri.

Treba zapaziti da ograničenje i_3 povezuje sve tri šeme relacije iz S. □

4.2.9. Pojava baze podataka

Šema baze podataka je definisana kao imenovana dvojka $N(S, I)$. Da bi se definisala pojava nad šemom baze podataka, potrebno je prvo definisati pojam pojave nad skupom šema relacija S i pojam ograničenja baze podataka.

Definicija 4.5. Pojava nad skupom šema relacija $S = \{(R_i, C_i) \mid i = 1,...,n\}$ je funkcija s, koja preslikava svako $(R_i, C_i) \in S$ u pojavu nad (R_i, C_i), odnosno

$$s : S \rightarrow \{r_i \mid (i = 1,...,n) \wedge (r_i \in SAT(R_i, C_i))\}. \square$$

Definicija 4.6. Ograničenje baze podataka $i \in \mathcal{I}$ je Bulova funkcija, koja svakoj pojavi s nad S pridružuje ili vrednost T ili vrednost $\perp$. Ako ta funkcija pridruži pojavi s vrednost T, tada s zadovoljava ograničenje i. □

Definicija 4.7. Pojava nad šemom baze podataka $N(S, \mathcal{I})$ je takva pojava s nad S, koja zadovoljava svako $i \in \mathcal{I}$, u oznaci $s \in SAT(S, \mathcal{I})$. □

Primer 4.15. Neka je s jedna pojava nad skupom šema relacija S iz primera 4.14. Ograničenja i_1, i_2 i i_3 iz primera 4.14 se mogu opisati na sledeći način:

$i_1(s) \equiv (\forall t \in r(NPU))(\exists u \in r(KN))(t[N] = u[N])$,
$i_2(s) \equiv (\forall t \in r(NPU))(\exists u \in r(KP))(t[P] = u[P])$ i
$i_3(s) \equiv (\forall t \in r(NPU))(\exists u \in r(KN) \wedge \exists v \in r(KP))(t[N] = u[N] \wedge t[P] = v[P] \wedge u[K] = v[K])$.

Na slici 4.9 je prikazana jedna pojava nad šemom baze podataka *Fakultet*. Skup relacija na slici predstavlja pojavu nad skupom šema relacija S, ali ne predstavlja pojavu nad šemom baze podataka *Fakultet*, jer nije zadovoljeno ograničenje i_5. Naime, nastavnik n_1 ne može predavati predmet p_3, jer n_1 pripada katedri k_1, a p_3 katedri k_2. Na slici 4.10 je prikazana jedna pojava nad šemom baze podataka *Fakultet*. □

K	N
k_1	n_1
k_1	n_2
k_2	n_3
k_2	n_4
k_2	n_5

K	P
k_1	p_1
k_1	p_2
k_2	p_3
k_2	p_4
k_3	p_5

N	P	U
n_1	p_1	u_1
n_1	p_3	u_2
n_2	p_1	u_1
n_3	p_3	u_1
n_4	p_3	u_3

Slika 4.9.

K	N
k_1	n_1
k_1	n_2
k_2	n_3
k_2	n_4
k_2	n_5

K	P
k_1	p_1
k_1	p_2
k_2	p_3
k_2	p_4
k_3	p_5

N	P	U
n_1	p_1	u_1
n_1	p_2	u_2
n_2	p_1	u_1
n_3	p_3	u_1
n_4	p_3	u_3

Slika 4.10.

5. Glava

Integritetna komponenta relacionog modela podataka

Na osnovu opisa strukturalne komponente relacionog modela podataka, sledi da ograničenja predstavljaju neophodni sastavni deo definicije šeme baze podataka. Koriste se, pri formiranju i ažuriranju baze podataka, u cilju održavanja sadržaja baze podataka u saglasnosti sa uočenim odnosima između obeležja stanja realnog sistema. Ograničenja se mogu klasifikovati kao:

- ograničenja torki,
- relaciona ograničenja i
- međurelaciona ograničenja.

Provera važenja ograničenja torke se sprovodi za svaku torku relacije posebno. Za proveru važenja relacionog ograničenja, moraju se posmatrati međusobni odnosi više torki jedne relacije. Ograničenja torke predstavljaju specijalan slučaj relacionog ograničenja, tako da ograničenja torke i relaciona ograničenja određuju koje relacije nad skupom obeležja predstavljaju pojave nad šemom relacije sa istim skupom obeležja, ali ništa ne govore o pojavama nad drugim šemama relacija, niti o njihovoj međusobnoj uslovljenosti. Putem relacionih ograničenja se reguliše lokalna konzistentnost - usaglašenost pojave sa ograničenjima, definisanim u šemi relacije.

Međurelaciona ograničenja regulišu globalnu usaglašenost baze podataka, kao pojave, sa ograničenjima, definisanim u šemi baze podataka. Međutim, baza podataka je konzistentna, ako je i lokalno i globalno konzistentna, tako da sledi da i relaciona ograničenja predstavljaju specijalan slučaj ograničenja baze podataka.

Primer 5.1. Ograničenje γ_2 iz primera 4.11, definisano sa $(\forall u \in r)(0 \leq u[BPI] \leq 50)$, predstavlja ograničenje torke. Ograničenje γ_1 iz istog primera, definisano sa $(\forall u, v \in r)(u[BRI] = v[BRI] \Rightarrow u = v)$ je relaciono ograničenje. Ograničenje i_3 iz primera 4.15, definisano sa

$$(\forall t \in r(NPU))(\exists u \in r(KN) \wedge \exists v \in r(KP))(t[N] = u[N] \wedge t[P] = v[P] \wedge u[K] = v[K]),$$

predstavlja međurelaciono ograničenje. □

U daljem tekstu, nakon definisanja pojma implikacionog problema, opisani su sledeći tipovi ograničenja relacionog modela podataka:

- funkcionalna zavisnost,
- višeznačna zavisnost,
- zavisnost spoja i
- zavisnost sadržavanja.

Implicitnu pretpostavku za aksiomatizaciju ovih ograničenja predstavljaju pretpostavke o postojanju šeme univerzalne relacije i njene pojave, te je i tim pojmovima poklonjena dužna pažnja. Takođe je obrađeno pitanje nekompletnih podataka i ograničenja koja su njihova posledica.

5.1. *Implikacioni problem*

U poglavlju, posvećenom strukturalnoj komponenti relacionog modela, ograničenja su posmatrana kao Booleove funkcije, koje nekoj relaciji ili nekom skupu relacija pridružuju jednu vrednost iz skupa $\{\mathsf{T}, \perp\}$. Neka je $C(\mathcal{U})$ oznaka za skup svih ograničenja, koja se mogu definisati nad relacijom $r(\mathcal{U})$. Tada je skup ograničenja C šeme relacije $N(\mathcal{U}, C)$, podskup skupa $C(\mathcal{U})$.

Pored ograničenja iz skupa C, pojava r nad šemom $(\mathcal{U}, C)$, zadovoljava i niz drugih ograničenja, koja su logička posledica skupa ograničenja C. Tako se dolazi do novog skupa ograničenja C', koji je, takođe, podskup skupa $C(\mathcal{U})$.

Primer 5.2. Neka su N i P oznake za skupove obeležja, karakteristične, redom, za klase entiteta *Nastavnik* i *Predmet* i neka je obeležje B godišnji fond časova predmeta. Neka, dalje, važe sledeća ograničenja:

(γ_1) svaki nastavnik n predaje samo jedan predmet p i
(γ_2) svaki predmet p se izvodi putem određenog fonda časova b.

Na osnovu ograničenja γ_1 i γ_2 se može izvesti zaključak da svaka pojava nad šemom relacije $(\{N, P, B\}, \{\gamma_1, \gamma_2\})$ mora zadovoljavati i ograničenje:

(γ_3) svaki nastavnik n je angažovan u nastavi b časova godišnje.

Ograničenje γ_3 je logička posledica skupa ograničenja $C = \{\gamma_1, \gamma_2\}$. Saglasno tome, šema relacije $(\{N, P, B\}, \{\gamma_1, \gamma_2\})$ se može zameniti šemom relacije $(\{N, P, B\}, \{\gamma_1, \gamma_2, \gamma_3\})$ i obratno. □

Na osnovu prethodnog primera se može zaključiti da različiti skupovi ograničenja nad istim skupom obeležja definišu isti skup pojava. Ova činjenica otvara pitanje minimizacije inicijalno definisanog skupa ograničenja. Naime, ako SUBP (ili program) treba da proveravaju zadovoljavanje ograničenja nakon svakog ažuriranja baze podataka, prirodno se nameće zaključak da je poželjno da skup ograničenja sadrži samo neophodne elemente. Pre prelaska na rešavanje problema minimizacije skupa ograničenja, potrebno je uvesti određeni broj pojmova.

Definicija 5.1. Neka je r relacija nad skupom obeležja $\mathcal{U}$, a C_1 i C_2 podskupovi skupa ograničenja $C(\mathcal{U})$. Tada:

- C_2 je ***logička posledica*** C_1, u oznaci $C_1 \models C_2$, ako svaka relacija r nad $\mathcal{U}$, koja zadovoljava C_1, zadovoljava i C_2,
- C_1 je ***ekvivalentno*** sa C_2, u oznaci $C_1 \equiv C_2$, ako važi $C_1 \models C_2$ i $C_2 \models C_1$. □

Primer 5.3. U primeru 5.2. je pokazano da $\{\gamma_1, \gamma_2\} \models \{\gamma_3\}$, a pošto trivijalno važi $\{\gamma_1, \gamma_2\} \models \{\gamma_1, \gamma_2\}$, zaključuje se da $\{\gamma_1, \gamma_2\} \models \{\gamma_1, \gamma_2, \gamma_3\}$. Isto tako, pošto $\{\gamma_1, \gamma_2\}$ predstavlja trivijalnu logičku posledicu skupa $\{\gamma_1, \gamma_2, \gamma_3\}$, zaključuje se da važi $\{\gamma_1, \gamma_2\} \equiv \{\gamma_1, \gamma_2, \gamma_3\}$. □

Definicija 5.2. Tip ograničenja ili ***tip zavisnosti*** je skup dvojki $(\mathcal{U}, \mathcal{T})$, gde je $\mathcal{U}$ skup obeležja, a $\mathcal{T}$ takav podskup skupa $C(\mathcal{U})$, da za svako $\tau \in \mathcal{T}$ važi $\mathcal{P}(\tau)$, gde je $\mathcal{P}(\tau)$ predikat. □

Primer 5.4. Ograničenje, definisano sa

$$(\forall u, v \in r(\mathcal{U}))(u[X] = v[X] \Rightarrow u = v),$$

za $X \subseteq \mathcal{U}$, naziva se zavisnošću ključa. Zavisnost ključa, u oznaci $zk(X)$, važi u $r(\mathcal{U})$, ako se svaka torka relacije $r(\mathcal{U})$ može jednoznačno identifikovati svojom X vrednošću. Zavisnosti ključa predstavljaju jedan tip zavisnosti, a dvojka $(\mathcal{U}, \{zk(X)\})$ jedan elemenat tog tipa zavisnosti, gde je $\mathcal{T} = \{zk(X) \mid zk(X)$ je zavisnost ključa$\} \subseteq C(\mathcal{U})$. □

U ovom poglavlju će se razmatrati sledeći tipovi ograničenja:

- zavisnosti ključa,
- funkcionalne zavisnosti,
- višeznačne zavisnosti,
- zavisnosti spoja,
- zavisnosti sadržavanja,
- kao i unije skupova ovih tipova zavisnosti.

Za rešavanje problema minimizacije skupa ograničenja određenog tipa, bitno je davanje odgovora na pitanje da li važi ekvivalencija $\mathcal{T} \setminus \{\tau\} \equiv \mathcal{T}$, gde je $\tau \in \mathcal{T}$. Ovo pitanje vodi definisanju implikacionog problema.

Definicija 5.3. Neka je $(\mathcal{U}, \mathcal{T})$ tip ograničenja. ***Implikacioni problem*** skupa ograničenja tipa $\mathcal{T}' \subseteq \mathcal{T}$ i ograničenja $\tau \in \mathcal{T}$ se iskazuje putem pitanja:

da li se može utvrditi da je τ posledica $\mathcal{T}'$ ($\mathcal{T}' \models \tau$). □

Da bi se rešio implikacioni problem za dati tip ograničenja $(\mathcal{U}, \mathcal{T})$, potreban je alat za izvođenje svih posledica nekog skupa ograničenja $\mathcal{T}$.

Za svaki tip ograničenja, karakterističan je pojam trivijalnog ograničenja.

Definicija 5.4. Ograničenje $\tau \in \mathcal{T}$ predstavlja ***trivijalno*** ograničenje, ako za svaku relaciju $r \in SAT(\mathcal{U})$ važi da zadovoljava τ. □

Primer 5.5. U primeru 5.4. je uveden pojam zavisnosti ključa. Neka je, za $X \subseteq \mathcal{U}$, $\mathcal{K} = \{zk(X)\}$ skup ograničenja tipa zavisnosti ključa. Skup ograničenja $\mathcal{K}$ trivijalno implicira važnost zavisnosti ključa $zk(\mathcal{U})$, jer $\mathcal{U}$ sadrži sve ključeve šeme relacije $(\mathcal{U}, C)$. Takođe, ako $\mathcal{K}$ implicira $zk(X)$, implicira i $zk(Y)$ za svako $Y \supseteq X$ takvo, da je i $Y \subseteq \mathcal{U}$. Ova pravila izvođenja novih zavisnosti ključa na osnovu postojećih, mogu se formalizovati na sledeći način:

($\mathbb{R}_1$) Ako je $\mathcal{U}$ skup obeležja, tada važi $zk(\mathcal{U})$.

($\mathbb{R}_2$) Ako je $X, Y \subseteq \mathcal{U}$ i $\{zk(X)\}$, tada važi $zk(XY)$. □

Skupovi pravila, kao što je skup $\{\mathbb{R}_1, \mathbb{R}_2\}$, nazivaju se ***sistemom aksioma*** za izvođenje ograničenja. Sistem aksioma predstavlja alat za izvođenje "novih" na osnovu "postojećih" ograničenja.

Neka je $\mathbb{A}$ skup aksioma za izvođenje ograničenja tipa $(\mathcal{U}, \mathcal{T})$, a $\mathbb{A}_1 \in \mathbb{A}$ i neka je $\mathcal{T} \cup \{\tau\} \subseteq C(\mathcal{U})$. Sa $\mathcal{T} \vdash_{\mathbb{A}} \tau$ se označava činjenica da se ograničenje τ može izvesti iz $\mathcal{T}$ pomoću aksioma iz $\mathbb{A}$, a sa $\mathcal{T} \vdash_{\mathbb{A}_1} \tau$ se označava činjenica da se ograničenje može izvesti iz $\mathcal{T}$ pomoću aksiome $\mathbb{A}_1$. Aksiome se nazivaju i ***pravilima izvođenja***. Naravno, poželjno je da $\mathcal{T} \vdash_{\mathbb{A}} \tau$ bude ekvivalentno sa $\mathcal{T} \models \tau$. Za definisanje uslova kada će ova ekvivalencija važiti, uvode se oznake:

$$\mathcal{T}^* = \{\tau \mid (\tau \in C(\mathcal{U})) \wedge (\mathcal{T} \models \tau)\} \text{ i}$$
$$\mathcal{T}^+_{\mathbb{A}} = \{\tau \mid (\tau \in C(\mathcal{U})) \wedge (\mathcal{T} \vdash_{\mathbb{A}} \tau)\}.$$

Definicija 5.5. Niz izvođenja ograničenja τ iz skupa ograničenja $\mathcal{T}$ pomoću sistema aksioma $\mathbb{A}$, u oznaci $\mathcal{T} \vdash_{\mathbb{A}} \tau$, je konačni niz ograničenja $(\tau_1, \ldots, \tau_n)$, $n \geq 1$, takav da važi:

1. $(\forall i \in \{1, \ldots, n\})(\tau_i \in \mathcal{T} \vee \{\tau_1, \ldots, \tau_{i-1}\} \vdash_{\mathbb{A}_j} \tau_i)$ i
2. $\tau_n = \tau$,

gde je $\mathbb{A}_j \in \mathbb{A}$. □

Primer 5.6. Neka je dato $\mathcal{U} = \{A, B, C\}$ i $\mathcal{T} = \{zk(A)\}$ skup zavisnosti ključa. Tada važi sledeći niz izvođenja

$$\{zk(A)\} \vdash_{\mathbb{R}_2} \{AB\} \vdash_{\mathbb{R}_2} \{ABC\}. \square$$

Definicija 5.6. Neka je $(\mathcal{U}, \mathcal{T})$ tip ograničenja, a $\mathbb{A}$ skup aksioma za izvođenje ograničenja tipa $(\mathcal{U}, \mathcal{T})$.

- $\mathbb{A}$ je ***neprotivrečan skup aksioma***, ako za svako $\mathcal{T}_1 \subseteq \mathcal{T}$ važi $\mathcal{T}_1^{+}{}_{\mathbb{A}} \subseteq \mathcal{T}_1^{*}$,
- $\mathbb{A}$ je ***kompletan skup aksioma***, ako za svako $\mathcal{T}_1 \subseteq \mathcal{T}$ važi $\mathcal{T}_1^{*} \subseteq \mathcal{T}_1^{+}{}_{\mathbb{A}}$,
- $\mathbb{A}$ je ***neredundantan skup aksioma***, ako važi implikacija $(\forall \mathbb{A}_1 \subseteq \mathbb{A})((\mathcal{T}^{+}{}_{\mathbb{A}_1} = \mathcal{T}^{+}{}_{\mathbb{A}}) \Rightarrow \mathbb{A}_1 = \mathbb{A})$. □

Ako je skup aksioma poznat iz konteksta, oznaka $\mathbb{A}$ za skup aksioma se, u notaciji, izostavlja.

Na osnovu definicije 5.6. sledi da, ako važi $\mathcal{T}^{+}{}_{\mathbb{A}} = \mathcal{T}^{*}$, tada će važiti i $\mathcal{T} \models \tau \Leftrightarrow \mathcal{T} \vdash_{\mathbb{A}} \tau$.

Primer 5.7. Skup aksioma $\{\mathbb{A}_1, \mathbb{A}_2\}$ je neprotivrečan, kompletan i neredundantan. Dokaz neprotivrečnosti i kompletnosti se ostavlja čitaocu. Na ovom mestu će biti pokazano da je skup $\mathbb{A} = \{\mathbb{A}_1, \mathbb{A}_2\}$ neredundantan.

Neka je $\mathcal{T} = \emptyset$ skup zavisnosti ključa. Tada je $\mathcal{T}^{+}{}_{\mathbb{A}} = \{zk(\mathcal{U})\}$, a $\mathcal{T}^{+}{}_{\mathbb{A}_2} = \emptyset$. Znači, aksioma $\mathbb{A}_1$ nije suvišna.

Neka je $\mathcal{U} = \{A, B, C\}$, $\mathcal{T} = \{zk(A)\}$. Tada je $\mathcal{T}^{+}{}_{\mathbb{A}} = \{zk(A), zk(AB), zk(AC), zk(ABC)\}$, a $\mathcal{T}^{+}{}_{\mathbb{A}_1} = \{zk(A), zk(ABC)\}$, što ukazuje da ni aksioma $\mathbb{A}_2$ nije redundantna u $\mathbb{A}$. □

5.2. *Funkcionalna zavisnost*

Funkcionalna zavisnost je prvi tip ograničenja, koji je definisan u okviru relacionog modela podataka još početkom sedamdesetih godina [C1].

Izraz oblika $f: X \rightarrow Y$ se naziva funkcionalnom zavisnošću. Pri tome, f predstavlja naziv funkcionalne zavisnosti, a X i Y predstavljaju skupove obeležja. Naziv f se često izostavlja, tako da se koristi skraćena notacija $X \rightarrow Y$. Kaže se da X funkcionalno određuje Y, ili da Y funkcionalno zavisi od X. To dalje znači da se svakom elementu $dom(X)$ može pridružiti najviše jedan elemenat iz $dom(Y)$. Poznavanjem jedne vrednosti obeležja X, može se tačno odrediti odgovarajuća vrednost Y. Funkcionalna zavisnost $f: X \rightarrow Y$ se može posmatrati i kao vremenski promenljiva funkcija. Atribut "vremenski promenljiva" ukazuje da se domen i kodomen ove funkcije menjaju u vremenu, kao i da, u različitim trenucima, istoj vrednosti iz $dom(X)$, f može pridružiti različite vrednosti iz $dom(Y)$.

Primer 5.8. Neka je $X = BRI$ (broj indeksa), a $Y = BPI$ (broj položenih ispita) i neka je definisana funkcionalna zavisnost $f: BRI \rightarrow BPI$. Tada, za $159 \in dom(BRI)$, u jednom trenutku može važiti $f(159) = 13$, što ukazuje da je student sa brojem indeksa *159* položio *13* ispita, a u drugom trenutku može važiti $f(159) = 14$, jer je student, u međuvremenu položio još jedan ispit.

Slično, u jednom trenutku, za $BRI = 831$, funkcija f može biti nedefinisana, jer student sa tim brojem indeksa ne postoji. Ako se, pri upisu, nekom studentu dodeli broj indeksa 831, tada će, inicijalno, važiti $f(831) = 0$. □

U literaturi se često koriste pojmovi ***leve*** i ***desne strane*** funkcionalne zavisnosti f, u oznaci, redom, $lhs(f)$ i $rhs(f)$. Takođe se uvodi i pojam ***skupa obeležja funkcionalne zavisnosti*** f, u oznaci $attr(f)$. Ako je $f: X \rightarrow Y$ funkcionalna zavisnost, tada je $lhs(f) = X$, $rhs(f) = Y$ i $attr(f) = XY$.

Definicija 5.7. Neka je r relacija nad skupom obeležja $\mathcal{U}$, a X i Y dva podskupa skupa $\mathcal{U}$. Skup torki relacije r ***zadovoljava funkcionalnu zavisnost*** $X \rightarrow Y$, ako za svake dve torke u i v u r važi implikacija

$$u[X] = v[X] \Rightarrow u[Y] = v[Y], \tag{5.1}$$

gde je $t[Z]$ vrednost obeležja Z u torki t. □

Ako relacija r zadovoljava funkcionalnu zavisnost $X \rightarrow Y$, kaže se i da ta funkcionalna zavisnost ***važi*** u relaciji r.

Primer 5.9. U relaciji r nad skupom obeležja $\{A, B, C\}$, prikazanoj na slici 5.1, važe sledeće funkcionalne zavisnosti: $f_1: A \rightarrow B$, $f_2: B \rightarrow A$, $f_3: AC \rightarrow B$, $f_4: BC \rightarrow A$. U r takođe važe, odnosno relacija r zadovoljava, funkcionalne zavisnosti oblika $A \rightarrow A$ ili $ABC \rightarrow A$. Relacija r ne zadovoljava funkcionalne zavisnosti: $f_5: C \rightarrow A$, $f_6: C \rightarrow B$ i $f_7: AB \rightarrow C$. □

$r =$

A	B	C
a	b	c
e	d	c
a	b	d

Slika 5.1.

Primer 5.10. U relaciji r nad skupom obeležja $\{BRI, IME, PRZ, PRD, OCE\}$, prikazanoj na slici 5.2, važe sledeće funkcionalne zavisnosti: $f_1: BRI \rightarrow IME$, $f_2: BRI \rightarrow PRZ$ i $f_3: \{BRI, PRD\} \rightarrow OCE$. U daljem tekstu će se, pri korišćenju mnemonika za oznake obeležja, koristiti notacija oblika $BRI + PRD$ kao zamena za $\{BRI, PRD\}$. Saglasno tome, $f_3: BRI + PRD \rightarrow OCE$. □

Funkcionalna zavisnost predstavlja intenzionalno ograničenje, jer se definiše nad skupom obeležja. Jedini način za utvrđivanje funkcionalnih zavisnosti je pažljiva analiza značenja obeležja i odnosa između elemenata njihovih domena u realnom svetu. Važenje funkcionalnih zavisnosti nad skupom obeležja se ne može utvrditi posmatranjem određenog broja relacija nad tim skupom obeležja. Ako je relacija prazan ili jednočlan skup, u njoj važe sve moguće funkcionalne zavisnosti. Posmatranjem relacija, može se jedino utvrditi koje funkcionalne zavisnosti sigurno ne važe nad skupom obeležja.

BRI	*IME*	*PRZ*	*PRD*	*OCE*
159	*Ivo*	*Ban*	*Mat*	*09*
013	*Ana*	*Ras*	*Fiz*	*10*
013	*Ana*	*Ras*	*Mat*	*06*
041	*Eva*	*Tot*	*Meh*	*06*
159	*Ivo*	*Ban*	*Meh*	*09*
013	*Ana*	*Ras*	*Meh*	*09*
099	*Ivo*	*Ban*	*Mat*	*07*

Slika 5.2.

Primer 5.11. Na osnovu relacije na slici 5.2, može se zaključiti da funkcionalna zavisnost $BRI \rightarrow OCE$ ne važi nad skupom obeležja $\{BRI, IME, PRZ, PRD, OCE\}$, realnog sistema, koji relacija na slici 5.2 približno odslikava. □

Neka je $\mathcal{F}$ skup funkcionalnih zavisnosti nad skupom obeležja $\mathcal{U}$, pri čemu je X, $Y \subseteq \mathcal{U}$, a $X \rightarrow Y$ jedna funkcionalna zavisnost. Saglasno definiciji 5.1, skup funkcionalnih zavisnosti $\mathcal{F}$ ***logički implicira*** $X \rightarrow Y$, u oznaci $\mathcal{F} \models X \rightarrow Y$, ako svaka relacija nad $\mathcal{U}$, koja zadovoljava sve zavisnosti iz $\mathcal{F}$, zadovoljava i $X \rightarrow Y$. Skup svih ***logičkih posledica*** skupa $\mathcal{F}$ označava se sa $\mathcal{F}^*$.

Primer 5.12. Lako je zaključiti da, ako su nad skupom $\mathcal{U}$ definisane funkcionalne zavisnosti $X \rightarrow Y$ i $Y \rightarrow Z$, tada, svaka relacija nad $\mathcal{U}$, koja zadovoljava ove dve funkcionalne zavisnosti, mora zadovoljavati i njihovu logičku posledicu $X \rightarrow Z$.

Da bi se proverila gornja tvrdnja, posmatra se relacija $r(\mathcal{U})$, koja bi zadovoljavala $X \rightarrow Y$ i $Y \rightarrow Z$, ali ne i $X \rightarrow Z$. Znači, u $r(\mathcal{U})$ postoje torke u i v takve, da je $u[X] = v[X]$ i $u[Z] \neq v[Z]$. Pošto $r(\mathcal{U})$ zadovoljava $X \rightarrow Y$, zaključuje se da je $u[Y] = v[Y]$. Pošto je $u[Y] = v[Y]$ i $Y \rightarrow Z$ u kontradikciji sa tvrđenjem $u[Z] \neq v[Z]$, zaključuje se da $r(\mathcal{U})$ zadovoljava $X \rightarrow Z$. □

Definicija 5.8. Neka je $\mathcal{F}$ skup funkcionalnih zavisnosti nad skupom obeležja $\mathcal{U}$. Skup svih relacija nad $\mathcal{U}$, koje zadovoljavaju skup funkcionalnih zavisnosti $\mathcal{F}$, označava se sa $SAT(\mathcal{U}, \mathcal{F})$. □

Na osnovu definicije 5.1 i definicije 5.8 sledi sledeća ekvivalencija

$$\mathcal{F} \models \mathcal{G} \equiv SAT(\mathcal{U}, \mathcal{F}) \subseteq SAT(\mathcal{U}, \mathcal{G}).$$

Primer 5.13. Neka je $\mathcal{F} = \{A \rightarrow B, B \rightarrow C\}$, a $\mathcal{G} = \{A \rightarrow C\}$. Da je $\mathcal{G}$ logička posledica $\mathcal{F}$ i da svako r nad $\mathcal{U}$, koje zadovoljava $\mathcal{F}$ zadovoljava i $\mathcal{G}$, pokazano je u primeru 5.12. Na slici 5.3, je prikazana relacija s nad $\mathcal{U}$, koja zadovoljava $\mathcal{G}$, ali ne i $\mathcal{F}$, što ilustruje tvrđenje $SAT(\mathcal{U}, \mathcal{F}) \subseteq SAT(\mathcal{U}, \mathcal{G})$. □

$s =$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

Slika 5.3.

5.2.1. *Armstrongove aksiome*

Armstrong [A] je izvršio aksiomatizaciju funkcionalnih zavisnosti, definišući jedan skup formalnih pravila, putem kojih se mogu izvesti sve logičke posledice nekog polaznog skupa funkcionalnih zavisnosti. Tokom vremena je definisano više ekvivalentnih sistema aksioma. Aksiome ovih sistema se nazivaju i pravilima izvođenja.

U daljem tekstu je definisan jedan sistem aksioma, označen sa $\mathfrak{I}$. Pretpostavka je da je skup funkcionalnih zavisnosti $\mathcal{F}$ definisan nad skupom obeležja $\mathcal{U}$ i da su X, Y, Z, W podskupovi skupa $\mathcal{U}$. Reč je o sledeće tri aksiome:

($\mathfrak{I}_1$) (***Refleksivnost.***) Ako je $Y \subseteq X$, tada važi $X \rightarrow Y$.
($\mathfrak{I}_2$) (***Proširenje.***) Ako je $X \rightarrow Y$ i $Z \subseteq W$, tada važi $XW \rightarrow YZ$.
($\mathfrak{I}_3$) (***Pseudotranzitivnost.***) Ako je $X \rightarrow Y$ i $YW \rightarrow Z$, tada važi $XW \rightarrow Z$.

Pravilo izvođenja $\mathfrak{I}_1$ dovodi do definisanja takozvanih ***trivijalnih*** funkcionalnih zavisnosti. To su one, za koje važi $rhs(f) \subseteq lhs(f)$. U slučaju pravila izvođenja $\mathfrak{I}_2$, treba zapaziti da zavisnost $X \rightarrow Y$ ili pripada $\mathcal{F}$ ili je izvedena iz $\mathcal{F}$ primenom pravila iz $\mathfrak{I}$. Kada je reč o pravilu $\mathfrak{I}_3$, za $W = \emptyset$, ono prelazi u ***tranzitivnost***.

Pre prelaska na dokazivanje da je:

- sistem aksioma $\mathfrak{I}$ ***neredundantan***, odnosno da ne sadrži suvišna pravila,
- ***neprotivrečan***, odnosno da primena $\mathfrak{I}$ daje samo funkcionalne zavisnosti, koje su logička posledica $\mathcal{F}$ i
- ***kompletan***, odnosno da se primenom pravila iz $\mathfrak{I}$ mogu dobiti sve logičke posledice zadatog skupa funkcionalnih zavisnosti $\mathcal{F}$,

daje se komentar pravila iz $\mathfrak{I}$ sa tačke gledišta funkcionalne zavisnosti kao vremenski promenljive funkcije.

Neka je $X = \{A_1, ..., A_n\}$, aksioma $\mathfrak{I}_1$ tvrdi da se uvek može uspostaviti vremenski promenljivo preslikavanje između skupa $dom(X) = dom(A_1) \times ... \times dom(A_n)$ i skupa $dom(Y) = dom(A_1) \times ... \times dom(A_m)$, za $m \leq n$.

Smisao aksiome $\mathfrak{I}_2$ je da, ako važi $f: X \rightarrow Y$, tada se može formirati nova funkcija, čiji domen predstavlja skup $dom(X) \times dom(W)$, a kodomen $dom(Y)$. Drugim rečima, vrednosti iz $dom(W)$ ne utiču na to koje vrednosti iz $dom(Y)$ funkcionalna zavisnost f pridružuje vrednostima iz $dom(X)$.

Aksioma $\mathfrak{I}_3$ predstavlja pravilo za kompoziciju funkcionalnih zavisnosti. Neka je $f: X \rightarrow Y$, $g: YW \rightarrow Z$, a $h: XW \rightarrow Z$. Primena h na $(x, w) \in dom(X) \times dom(W)$ se može posma-

trati u dva koraka. Prvo, primenom f na $x \in dom(X)$ dobija se jedno $y \in dom(Y)$. Zatim, primenom g na (y, w), dobija se jedno $z \in dom(Z)$. Znači, $h(x, w) = g(f(x), w)$. Ako je $W = \emptyset$, tada pseudotranzitivnost postaje tranzitvnost.

Lema 5.1. Sistem aksioma $\mathfrak{F}$ je neredundantan.

Dokaz. Da bi se pokazalo da je $\mathfrak{F}$ neredundantan skup, dovoljno je pokazati da se, na osnovu dva pravila, ne mogu izvesti posledice trećeg, pod pretpostavkom da važi $X, Y \subseteq \mathcal{U}$.

Da bi se pokazalo da $\mathfrak{F}_2$ i $\mathfrak{F}_3$ ne impliciraju posledice $\mathfrak{F}_1$, posmatra se $\mathcal{F} = \{A \to \emptyset\}$, za $A \in \mathcal{U}$. Tada je

$$\mathcal{F}^+_{\{\mathfrak{F}_2, \mathfrak{F}_3\}} = \{X \to Y \mid (Y \subseteq X) \wedge (A \in X)\}.$$

dok je

$$\mathcal{F}^+_{\mathfrak{F}} = \{X \to Y \mid (Y \subseteq X)\}.$$

Skup $\mathcal{F}^+_{\{\mathfrak{F}_2, \mathfrak{F}_3\}}$ ne sadrži funkcionalne zavisnosti oblika $X \to Y$ za $A \notin X$, kao ni funkcionalnu zavisnost $\emptyset \to \emptyset$. Sve ove funkcionalne zavisnosti skup $\mathcal{F}^+_{\mathfrak{F}}$ sadrži. Prema tome, može se zaključiti da $\mathcal{F}^+_{\mathfrak{F}} \neq \mathcal{F}^+_{\{\mathfrak{F}_2, \mathfrak{F}_3\}}$.

Da bi se pokazalo da $\mathfrak{F}_1$ i $\mathfrak{F}_3$ ne impliciraju posledice primene $\mathfrak{F}_2$, posmatra se skup funkcionalnih zavisnosti $\mathcal{F} = \{A \to B\}$, za $A, B \in \mathcal{U}$ i $A \neq B$. Tada je

$$\mathcal{F}^+_{\mathfrak{F}} = \{A \to B\} \cup \{X \to Y \mid (Y \subseteq X)\} \cup \{AX \to BY \mid X \neq \emptyset \wedge B \notin X \wedge Y \subseteq X\},$$

a

$$\mathcal{F}^+_{\{\mathfrak{F}_1, \mathfrak{F}_3\}} = \{A \to B\} \cup \{X \to Y \mid (Y \subseteq X)\}.$$

Skup $\mathcal{F}^+_{\{\mathfrak{F}_1, \mathfrak{F}_3\}}$, očigledno ne sadrži funkcionalnu zavisnost $AX \to BY$, za $X \neq \emptyset$, $B \notin X$ i $Y \subseteq X$, dok je skup $\mathcal{F}^+_{\mathfrak{F}}$ sadrži.

Da bi se pokazalo da $\mathfrak{F}_1$ i $\mathfrak{F}_2$ ne impliciraju posledice $\mathfrak{F}_3$, posmatra se skup funkcionalnih zavisnosti $\mathcal{F} = \{A \to B, B \to C\}$, gde su A, B i C međusobno različita obeležja iz $\mathcal{U}$. Tada je

$$\mathcal{F}^+_{\{\mathfrak{F}_1, \mathfrak{F}_2\}} = \{X \to Y \mid (Y \subseteq X) \vee ((C \in Y \wedge B \in X) \vee (B \in Y \wedge A \in X))\}.$$

Skup $\mathcal{F}^+_{\{\mathfrak{F}_1, \mathfrak{F}_2\}}$ ne sadrži funkcionalnu zavisnost $A \to C$, dok je skup $\mathcal{F}^+_{\mathfrak{F}}$ sadrži. □

Lema 5.2. Sistem aksioma $\mathfrak{F}$ je neprotivrečan.

Dokaz. Neprotivrečnost pravila $\mathfrak{F}_1$ sledi iz definicije interpretacije funkcionalne zavisnosti.

Da bi se dokazala neprotivrečnost pravila $\mathfrak{F}_2$, pretpostavlja se da relacija $r(\mathcal{U})$ zadovoljava funkcionalnu zavisnost $X \to Y$, ali ne i zavisnost $XW \to YZ$, gde je $Z \subseteq W$. Znači, u $r(\mathcal{U})$ postoje torke u i v takve da je $u[XW] = v[XW]$ i $u[YZ] \neq v[YZ]$, što se svodi na $u[Z] \neq v[Z]$, jer $X \to Y$ važi u $r(\mathcal{U})$. Međutim, to je u kontradikciji sa pretpostavkom da važi $Z \subseteq W$.

Da bi se dokazala neprotivrečnost pravila $\mathfrak{I}_3$, pretpostavlja se da relacija $r(\mathcal{U})$ zadovoljava funkcionalne zavisnosti $X\rightarrow Y$ i $YW\rightarrow Z$, ali ne i zavisnost $XW\rightarrow Z$. Znači, u $r(\mathcal{U})$ postoje torke u i v takve, da je $u[XW] = v[XW]$ i $u[Z] \neq v[Z]$. Saglasno tome, mora važiti i $u[YW] \neq v[YW]$, što se svodi na $u[Y] \neq v[Y]$. Međutim, to je u kontradikciji sa pretpostavkom da $r(\mathcal{U})$ zadovoljava $X\rightarrow Y$. □

Pošto je neprotivrečnost sistema aksioma $\mathfrak{I}$ dokazana, može se formulisati i dokazati sledeća lema.

Lema 5.3. Pravila izvođenja:

($\mathfrak{I}_4$) (*Unija.*) Ako je $X\rightarrow Y_1$ i $X\rightarrow Y_2$, tada važi $X\rightarrow Y_1Y_2$,
($\mathfrak{I}_5$) (*Dekompozicija.*) Ako je $X\rightarrow Y$ i $Z \subseteq Y$, tada važi $X\rightarrow Z$,

su posledica sistema aksioma $\mathfrak{I}$.

Dokaz. Da bi se proverila važnost pravila $\mathfrak{I}_4$, prvo se funkcionalna zavisnost $X\rightarrow Y_1$ proširi sa Y_2 (saglasno aksiomi $\mathfrak{I}_2$), što daje $XY_2\rightarrow Y_1Y_2$. Saglasno aksiomi $\mathfrak{I}_3$, na osnovu $X\rightarrow Y_2$ i $XY_2\rightarrow Y_1Y_2$, sledi $XX\rightarrow Y_1Y_2$, odnosno $X\rightarrow Y_1Y_2$. Da bi se proverila važnost pravila $\mathfrak{I}_5$, polazi se od $Y\rightarrow Z$, za $Z \subseteq Y$, tada, na osnovu $X\rightarrow Y$ i $\mathfrak{I}_3$, sledi $X\rightarrow Z$. □

Na osnovu pravila izvođenja $\mathfrak{I}_5$ sledi da se svaki skup funkcionalnih zavisnosti $\mathcal{F}$ može zameniti ekvivalentnim skupom $\mathcal{G}$, u kojem se na desnoj strani svake funkcionalne zavisnosti nalazi samo jedno obeležje.

Definicija 5.9. Iscrpnom primenom pravila izvođenja iz $\mathfrak{I}$ na skup funkcionalnih zavisnosti $\mathcal{F}$, dobija se skup funkcionalnih zavisnosti $\mathcal{F}^+$. Skup $\mathcal{F}^+$ sadrži sve funkcionalne zavisnosti iz $\mathcal{F}$ i sve one, koje se mogu izvesti iz $\mathcal{F}$ primenom pravila iz $\mathfrak{I}$. Skup $\mathcal{F}^+$ se naziva ***zatvaranjem (zatvaračem) skupa funkcionalnih zavisnosti*** $\mathcal{F}$. □

Za dokazivanje kompletnosti sistema aksioma $\mathfrak{I}$, potrebno je prvo definisati efikasan postupak za utvrdivanje odgovora na pitanje da li je neka funkcionalna zavisnost f u $\mathcal{F}^+$. U tom cilju se uvodi pojam zatvaranja skupa obeležja $X \subseteq \mathcal{U}$, s ozirom na $\mathcal{F}$. Ovaj pojam se uvodi i zbog potrebe rešavanja implikacionog problema za funkcionalne zavisnosti.

Definicija 5.10. Zatvaranje (zatvarač) skupa obeležja $X \subseteq \mathcal{U}$, s obzirom na skup funkcionalnih zavisnosti $\mathcal{F}$, definisan u $\mathcal{U}$, je

$$X^+_{\mathcal{F}} = \{A \mid X\rightarrow A \in \mathcal{F}^+\}. \square$$

Ako je $\mathcal{F}$ poznato iz konteksta, umesto $X^+_{\mathcal{F}}$, piše se kratko X^+.

Lema 5.4. Funkcionalna zavisnost $X\rightarrow Y$ je u $\mathcal{F}^+$ ako i samo ako važi $Y \subseteq X^+$.

Dokaz. ($\Rightarrow$) Neka je $Y \subseteq X^+$ i $Y = \{A_1,...,A_n\}$. Prema definiciji 5.10, $X\rightarrow A_i$ je posledica $\mathcal{F}$, za svako $i \in \{1,...,n\}$. Na osnovu pravila $\mathfrak{I}_4$ sledi $X\rightarrow Y \in \mathcal{F}^+$.

($\Leftarrow$) Neka je $X\rightarrow Y \in \mathcal{F}^+$. Na osnovu pravila $\mathfrak{I}_5$, za svako i sledi $X\rightarrow A_i$, te se zaključuje da je $Y \subseteq X^+$. □

Sada se može preći na dokazivanje kompletnosti sistema aksioma $\mathfrak{I}$. Ideja dokaza je u pokazivanju da, ako funkcionalna zavisnost $X \to Y$ nije u $\mathcal{F}^+$, ona ne predstavlja logičku posledicu skupa funkcionalnih zavisnosti $\mathcal{F}$.

Teorema 5.1. Sistem aksioma $\mathfrak{I}$ je neredundantan, neprotivrečan i kompletan.

Dokaz. Neredundantnost i neprotivrečnost su dokazane, redom, putem lema 5.1 i 5.2.

Neka je $\mathcal{F}(\mathcal{U})$ skup svih funkcionalnih zavisnosti nad skupom $\mathcal{U}$ i $\mathcal{F} \subseteq \mathcal{F}(\mathcal{U})$. Treba pokazati da važi $\mathcal{F}^* \subseteq \mathcal{F}^+$, odnosno $\mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^+ \subseteq \mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^*$. Neka je $X \to Y$ funkcionalna zavisnost, koja se ne može izvesti iz $\mathcal{F}$. Znači, $X \to Y \in \mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^+$. Tada postoji $A \in Y$ takvo, da važi $X \to A \in \mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^+$.

Da bi se pokazalo da važi i $X \to A \in \mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^*$, posmatra se relacija $r(\mathcal{U})$ koja sadrži samo dve torke u i v, definisane na sledeći način:

$u[A] = 0$ za svako $A \in \mathcal{U}$,
$v[A] = 0$ za svako $A \in X^+$,
$v[A] = 1$ za svako $A \in \mathcal{U} \setminus X^+$.

$r(\mathcal{U}) =$

	X^+	$\mathcal{U} \setminus X^+$
u	0	0
v	0	1

Prvo će biti pokazano da $r(\mathcal{U})$ zadovoljava svaku zavisnost iz $\mathcal{F}$. Neka je $W \to Z \in \mathcal{F}$, takva da je $r(\mathcal{U})$ ne zadovoljava. Tada važi $W \subseteq X^+$ i $Z \cap (\mathcal{U} \setminus X^+) \neq \emptyset$. Neka je $B \in Z \cap (\mathcal{U} \setminus X^+)$. Na osnovu leme 5.4 sledi $X \to W \in \mathcal{F}^+$, a na osnovu pravila $\mathfrak{I}_1$ važi $Z \to B$, te na osnovu pravila $\mathfrak{I}_3$, sledi $X \to B$. Međutim, to je u kontradikciji sa pretpostavkom da $B \notin X^+$. Znači, $r(\mathcal{U})$ zadovoljava $\mathcal{F}$, kao i sve logičke posledice $\mathcal{F}$.

Na osnovu tvrđenja $X \to A \in \mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^+$, zaključuje se da važi $A \notin X^+$. Pošto važi $A \notin X^+$, tada je $u[X^+] = v[X^+]$ i $u[A] \neq v[A]$, te $r(\mathcal{U})$ ne zadovoljava funkcionalnu zavisnost $X \to A$. Znači, $X \to A$ nije logička posledica $\mathcal{F}$, te važi $X \to A \in \mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^*$, što je i trebalo dokazati. □

Bitna posledica teoreme 5.1. je da važi $\mathcal{F}^* = \mathcal{F}^+_{\mathfrak{I}}$ i da se za izračunavanje svih logičkih posledica zadatog skupa funkcionalnih zavisnosti mogu koristiti pravila izvođenja iz $\mathfrak{I}$. Saglasno tome, zaključuje se da su tvrđenja $\mathcal{F} \models f$ i $\mathcal{F} \vdash_{\mathfrak{I}} f$ ekvivalentna, te se implikacioni problem za funkcionalne zavisnosti svodi na proveru tipa $f \in \mathcal{F}^+_{\mathfrak{I}}$. U daljem tekstu će se podrazumevati da se zatvaranje skupa funkcionalnih zavisnosti izračunava primenom aksioma iz $\mathcal{F}$, te će se, u notaciji za $\mathcal{F}^+$ izostavljati $\mathfrak{I}$ u indeksu.

5.2.2. Izračunavanje zatvaranja

Kada je $\mathcal{F}$ zadati skup funkcionalnih zavisnosti, primenom pravila izvođenja $\mathfrak{I}$ na $\mathcal{F}$, mogu se dobiti nove funkcionalne zavisnosti. Zatvaranje zadatog skupa funkcionalnih zavisnosti sadrži kompletnu informaciju o ograničenjima među podacima u ekstenziji. Međutim, već i za mali skup funkcionalnih zavisnosti $\mathcal{F}$, zatvaranje $\mathcal{F}^+$ sadrži veoma veliki broj funkcionalnih zavisnosti, od kojih su mnoge logički suvišne.

Primer 5.14. Neka je $\mathcal{F} = \{A \to B, B \to C\}$. Tada je

$$\mathcal{F}^+ = \{\emptyset \to \emptyset, A \to \emptyset, A \to A, A \to B, A \to C, A \to AB, A \to AC, A \to BC, A \to ABC, B \to \emptyset, B \to B, B \to C, B \to BC, C \to \emptyset, C \to C, AB \to \emptyset, AB \to A, AB \to B, AB \to C, AB \to AB, AB \to AC, AB \to BC, AB \to ABC, AC \to \emptyset, AC \to A, AC \to B, AC \to C, AC \to AB, AC \to AC, AC \to BC, AC \to ABC, BC \to \emptyset, BC \to B, BC \to C, BC \to BC, ABC \to \emptyset, ABC \to A, ABC \to B, ABC \to C, ABC \to AB, ABC \to AC, ABC \to BC, ABC \to ABC\}.$$

Čak i za mali broj obeležja i funkcionalnih zavisnosti, kao u ovom primeru, kardinalni broj zatvaranja $\mathcal{F}^+$ je veliki. Iscrpnom primenom pravila dekompozicije i unije na skup $\mathcal{F}^+$, dobijaju se, redom, skupovi $\mathcal{F}_d{}^+$ i $\mathcal{F}_u{}^+$, koji su ekvivalentni sa sa $\mathcal{F}^+$, ali imaju manji kardinalni broj.

Skup funkcionalnih zavisnosti $\mathcal{F}_d{}^+$ sadrži takve funkcionalne zavisnosti, koje, na desnoj strani imaju jednočlane skupove

$$\mathcal{F}_d{}^+ = \{\emptyset \to \emptyset, A \to \emptyset, A \to A, B \to \emptyset, B \to B, C \to \emptyset, C \to C, A \to B, A \to C, B \to C, AB \to \emptyset, AB \to A, AB \to B, AB \to C, AC \to \emptyset, AC \to A, AC \to B, AC \to C, BC \to \emptyset, BC \to B, BC \to C, ABC \to \emptyset, ABC \to A, ABC \to B, ABC \to C\}.$$

Nakon primene pravila izvođenja $\mathfrak{I}_4$ (unija) na skup $\mathcal{F}^+$, dobija se

$$\mathcal{F}_u{}^+ = \{\emptyset \to \emptyset, A \to ABC, B \to BC, C \to C, AB \to ABC, AC \to ABC, BC \to BC, ABC \to ABC\}.$$

U skupu $\mathcal{F}_u{}^+$ se svaka leva strana funkcionalne zavisnosti javlja samo jedanput. U daljem tekstu će se, kao zatvaranje skupa funkcionalnih zavisnosti $\mathcal{F}$, koristiti $\mathcal{F}_d{}^+$ ili $\mathcal{F}_u{}^+$, bez navođenja indeksa d ili u. □

Neka je $\mathcal{U}$ skup obeležja, a $\mathcal{F}$ jedan skup funkcionalnih zavisnosti u $\mathcal{U}$. Ako se iscrpnom primenom pravila izvođenja $\mathfrak{I}_4$ na skup $\mathcal{F}^+$ dobija skup, kod kojeg su leve strane svih funkcionalnih zavisnosti različite, tada je

$$|\mathcal{F}^+| = |\mathcal{P}(\mathcal{U})|,$$

gde je $\mathcal{P}(\mathcal{U})$ partitivni skup skupa $\mathcal{U}$. Drugim rečima, $|\mathcal{F}^+| = 2^{|\mathcal{U}|}$.

Funkcionalne zavisnosti predstavljaju jedan od uslova integriteta relacionog modela podataka i jedno od bitnih ograničenja relacione baze podataka. To dalje znači da, SUBP ili aplikativni program, treba da proveravju da li baza podataka zadovoljava svaku od funkcionalnih zavisnosti iz $\mathcal{F}$, nakon svakog ažuriranja baze podataka. Međutim, $\mathcal{F}$ može sadržati veliki broj logički suvišnih funkcionalnih zavisnosti. Da bi se izbegao rad sa

nepotrebno velikim skupom funkcionalnih zavisnosti, postavlja se zadatak pronalaženja takvog minimalnog podskupa skupa $\mathcal{F}$, na osnovu kojeg se mogu izvesti sve funkcionalne zavisnosti iz $\mathcal{F}^+$.

Za traženje "minimalnog" skupa funkcionalnih zavisnosti, bitni su pojmovi:

- redukovanog skupa funkcionalnih zavisnosti i
- neredundantnog pokrivanja skupa funkcionalnih zavisnosti.

5.2.3. *Redukcija*

Definicija 5.11. Funkcionalna zavisnost $X \to A \in \mathcal{F}^+$ je *redukovana* ili ima *redukovanu levu stranu* s obzirom na $\mathcal{F}$, ako važi

$$(\forall Y \subset X)(Y \to A \notin \mathcal{F}^+). \square$$

Definicija 5.12. Skup funkcionalnih zavisnosti $\mathcal{F}$ je *redukovan* ili ima *redukovane leve strane*, ako je svaka $f: X \to A$ iz $\mathcal{F}$ redukovana funkcionalna zavisnost. $\square$

Primer 5.15. Neka je $\mathcal{F} = \{AB \to C, A \to B\}$. Skup funkcionalnih zavisnosti $\mathcal{F}$ nije redukovan. Naime, na osnovu $\mathfrak{F}_3$ i $\mathcal{F}$, sledi, da ako važi $A \to B$ i $AB \to C$, tada važi $A \to C$, jer je $AA = A$, te se zaključuje da funkcionalna zavisnost $AB \to C$ nije redukovana u skupu $\mathcal{F}$. $\square$

ALGORITAM REDUKCIJE

```
PROCES Redukcija_fz
   Ulaz: F      (* zadati skup funkcionalnih zavisnosti *)
   Izlaz: G     (* skup funkcionalnih zavisnosti sa redukovanim levim stranama,
                   ekvivalentan sa F *)
POČETAK PROCESA Redukcija_fz
   POSTAVI  G←∅
   RADI  redukcija (∀f: X→B ∈ F)
      POSTAVI  Y ← X
      RADI  eliminacija (∀A ∈ Y)
         AKO JE  (Y \{A})→B ∈ F+  TADA
            POSTAVI  Y ← Y \{A}
         INAČE
         KRAJ AKO
      KRAJ RADI  eliminacija
      POSTAVI  G ← G ∪ {Y→A}
   KRAJ RADI  redukcija
KRAJ PROCESA Redukcija_fz
```

Slika 5.4.

Na slici 5.4. je prikazan algoritam za generisanje skupa funkcionalnih zavisnosti $\mathcal{G}$ sa redukovanim levim stranama, na osnovu polaznog skupa funkcionalnih zavisnosti $\mathcal{F}$. Pretpostavka je da svaka funkcionalna zavisnost u $\mathcal{F}$ poseduje samo jedno obeležje na svojoj desnoj strani.

Primer 5.16. Neka je $\mathcal{F} = \{AB \rightarrow C, A \rightarrow B, AD \rightarrow B\}$. Primena algoritma redukcije na skup funkcionalnih zavisnosti $\mathcal{F}$ dovodi do redukovanja levih strana funkcionalnih zavisnosti:

- $AB \rightarrow C$ se redukuje u $A \rightarrow C$, jer je $A \rightarrow C \in \mathcal{F}^+$,
- $AD \rightarrow B$ se redukuje u $A \rightarrow B$, jer je $A \rightarrow B \in \mathcal{F}^+$.

Konačno se dobija $\mathcal{G} = \{A \rightarrow C, A \rightarrow B\}$. Pri čemu važi $\mathcal{F}^+ = \mathcal{G}^+$, odnosno $\mathcal{F} \equiv \mathcal{G}$. □

Konačan rezultat redukcije leve strane funkcionalne zavisnosti f može zavisiti od redosleda, kojim se biraju elementi iz $lhs(f)$, u cilju eventualne eliminacije. Naime, obeležje na desnoj strani f može zavisiti od više različitih poskupova skupa $lhs(f)$.

Primer 5.17. Neka je $\mathcal{F} = \{ABC \rightarrow D, AB \rightarrow C, C \rightarrow B\}$. Za funkcionalnu zavisnost $ABC \rightarrow D$, algoritam redukcije daje jedno od sledeća dva rešenja: $Y_1 = AB$, ili $Y_2 = AC$. □

5.2.4. *Neredundantno pokrivanje*

Neka su $\mathcal{F}$ i $\mathcal{G}$ dva skupa funkcionalnih zavisnosti. Skupovi $\mathcal{F}$ i $\mathcal{G}$ su ekvivalentni ako i samo ako važi $\mathcal{F}^+ = \mathcal{G}^+$. Za ekvivalentne skupove funkcionalnih zavisnosti se kaže i da pokrivaju jedan drugog.

Definicija 5.13. Pokrivanje (pokrivač) skupa funkcionalnih zavisnosti $\mathcal{F}$ je svaki skup funkcionalnih zavisnosti $\mathcal{G}$, koji ima isto zatvaranje kao $\mathcal{F}$. □

Primer 5.18. Neka je $\mathcal{F}_1 = \{A \rightarrow B, B \rightarrow C\}$, a $\mathcal{F}_2 = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, AB \rightarrow A\}$. Lako se proverava da važi $\mathcal{F}_1^+ = \mathcal{F}_2^+$, te se zaključuje da $\mathcal{F}_1$ i $\mathcal{F}_2$ predstavljaju pokrivanja istog skupa funkcionalnih zavisnosti. □

Definicija 5.14. Skup $\mathcal{G}$ je ***neredundantno pokrivanje*** skupa zavisnosti $\mathcal{F}$, ako ne sadrži pravi podskup koji takođe predstavlja pokrivanje skupa $\mathcal{F}$. □

Definicija 5.15. Funkcionalna zavisnost f je ***redundantna*** u $\mathcal{F}$, ako se f može izvesti iz $\mathcal{F} \setminus \{f\}$, odnosno ako je $f \in (\mathcal{F} \setminus \{f\})^+$. □

Iz prethodnih definicija sledi da je ***pokrivanje redundantno***, ako sadrži redundantne funkcionalne zavisnosti. Na slici 5.5 je prikazan algoritam za izračunavanje neredundantnog pokrivanja $\mathcal{G}$ skupa funkcionalnih zavisnosti $\mathcal{F}$. Pretpostavka je da sve funkcionalne zavisnosti skupa $\mathcal{F}$ poseduju samo jedno obeležje na svojoj desnoj strani.

ALGORITAM NEREDUNDANTNOG POKRIVANJA

PROCES Neredundantno_pokrivanje
  Ulaz: $\mathcal{F} = \{X \to A \mid X \subseteq \mathcal{U} \wedge A \in \mathcal{U}\}$ (* zadati skup funkcionalnih zavisnosti *)
  Izlaz: $\mathcal{G}$ (* neredundantno pokrivanje skupa $\mathcal{F}$ *)
POČETAK PROCESA Neredundantno_pokrivanje
  POSTAVI $\mathcal{G} \leftarrow \mathcal{F}$
  RADI eliminacija $(\forall f \in \mathcal{G})$
    AKO JE $f \in (\mathcal{G} \setminus \{f\})^+$ *TADA*
      POSTAVI $\mathcal{G} \leftarrow \mathcal{G} \setminus \{f\}$
    INAČE
    KRAJ AKO
  KRAJ RADI eliminacija
KRAJ PROCESA Neredundantno_pokrivanje

Slika 5.5.

Primer 5.19. Skup funkcionalnih zavisnosti $\mathcal{F}_1$ iz primera 5.18, predstavlja neredundantno pokrivanje skupa $\mathcal{F}_2$, kao i samog sebe.

Neka je dat skup funkcionalnih zavisnosti $\mathcal{F} = \{A \to B, B \to C, A \to C\}$. Lako se proverava da $A \to B \notin (\mathcal{F} \setminus \{A \to B\})^+$, $B \to C \notin (\mathcal{F} \setminus \{B \to C\})^+$ i $A \to C \in (\mathcal{F} \setminus \{A \to C\})^+$, te se zaključuje da $\mathcal{G} = \{A \to B, B \to C\}$ predstavlja neredundantno pokrivanje za $\mathcal{F}$. □

Jedan skup funkcionalnih zavisnosti može imati više neredundantnih pokrivanja. Slično kao u slučaju algoritma redukcije i rezultat primene algoritma neredundantnog pokrivanja može zavisiti od redosleda kojim se biraju funkcionalne zavisnosti u petlji "eliminacija". To je posledica postojanja funkcionalnih zavisnosti oblika $X \to Y$ i $Y \to X$ u $\mathcal{G}^+$.

Primer 5.20. Skup funkcionalnih zavisnosti $\mathcal{G} = \{A \to B, A \to C, A \to D, B \to D, D \to B, BC \to E, AC \to E\}$ je redukovan, ali nije neredundantan. Naime, ili funkcionalna zavisnost $A \to B$ ili funkcionalna zavisnost $A \to D$ je tranzitivna, redom, zbog $D \to B$ ili $B \to D$. Takođe, funkcionalna zavisnost $AC \to E$ je pseudotranzitivna, zbog $A \to B$ i $BC \to E$.

Skup funkcionalnih zavisnosti $\mathcal{F} = \{A \to B, A \to C, B \to D, D \to B, BC \to E\}$ predstavlja jedno neredundantno pokrivanje skupa $\mathcal{G}$. Drugo neredundantno pokrivanje predstavlja $\mathcal{H} = \{A \to D, A \to C, B \to D, D \to B, DC \to E\}$, treće $\mathcal{I} = \{A \to B, A \to C, B \to D, D \to B, DC \to E\}$ i četvrto $\mathcal{J} = \{A \to D, A \to C, D \to B, B \to D, BC \to E\}$. □

Definicija 5.16. Skup $\mathcal{G}$ predstavlja ***kanoničko pokrivanje*** skupa funkcionalnih zavisnosti $\mathcal{F}$, ako su zadovoljeni sledeći uslovi:

1° $\mathcal{G}^+ = \mathcal{F}^+$,
2° $\mathcal{G}$ sadrži samo redukovane funkcionalne zavisnosti,
3° $\mathcal{G}$ je neredundantan skup i
4° desna strana svake funkcionalne zavisnosti u $\mathcal{G}$ sadrži samo jedno obeležje. □

Primer 5.21. Skup funkcionalnih zavisnosti $\mathcal{F} = \{A \rightarrow BC, BC \rightarrow D\}$ je redukovan i neredundantan, ali nije kanoničko pokrivanje. Skup funkcionalnih zavisnosti $\mathcal{G} = \{A \rightarrow B, A \rightarrow C, BC \rightarrow D\}$ je kanoničan. □

Kada je reč o funkcionalnim zavisnostima, pojam minimalnog pokrivanja je dosta neodređen. To može biti redukovano, neredundantno pokrivanje sa minimalnim brojem funkcionalnih zavisnosti, ili sa minimalnim brojem simbola, neophodnih za predstavljanje ograničenja, ili i jedno i drugo.

Primer 5.22. Skupovi funkcionalnih zavisnosti $\mathcal{F} = \{A \rightarrow BC, BC \rightarrow D, D \rightarrow A\}$, $\mathcal{G} = \{A \rightarrow BC, BC \rightarrow A, A \rightarrow D, D \rightarrow A\}$ i $\mathcal{H} = \{A \rightarrow BC, BC \rightarrow A, BC \rightarrow D, D \rightarrow BC\}$ su ekvivalentni, redukovani i neredundantni. Međutim, $|\mathcal{F}| = 3$, $|\mathcal{G}| = 4$ i $|\mathcal{H}| = 4$. Ako se sa $s(X)$ označi broj simbola upotrebljenih za označavanje skupa X, tada je: $s(\mathcal{F}) = 8$, $s(\mathcal{G}) = 10$ i $s(\mathcal{H}) = 12$. □

U daljem tekstu će se smatrati da je jedino bitno naći redukovano, neredundantno pokrivanje zadatog skupa funkcionalnih zavisnosti, bez obzira na opisane aspekte minimalnosti.

5.2.5. *Izračunavanje zatvaranja skupa obeležja*

Algoritam redukcije, slika 5.4, zahteva izračunavanje zatvaranja polaznog skupa $\mathcal{F}$ i sprovođenje provera tipa $(Y \setminus \{A\}) \rightarrow B \in \mathcal{F}^+$ (za svako $f \in \mathcal{F}$ takvo da je $|lhs(f)| > 1$). Algoritam neredundantnog pokrivanja, na slici 5.5, zahteva sprovođenje $|\mathcal{F}|$ provera tipa $f \in (\mathcal{G} \setminus \{f\})^+$. Izračunavanja zatvaranja skupa funkcionalnih zavisnosti traži iscrpnu primenu Armstrongovih aksioma. Kardinalni broj skupa $\mathcal{F}^+$ je veći ili jednak $2^{|\mathcal{U}|}$, gde je $\mathcal{U}$ skup obeležja, u kojem je $\mathcal{F}$ definisano. To dalje znači da je ocena kompleksnosti oba ova algoritma eksponencijalna.

Provere tipa $f \in \mathcal{F}^+$ (implikacioni problem za funkcionalne zavisnosti), sa polinomijalnom ocenom kompleksnosti, zasnovane su na pojmu zatvaranja skupa obeležja. Prema definiciji 5.10, zatvaranje skupa obeležja $X \subseteq \mathcal{U}$, u oznaci $X^+_{\mathcal{F}}$, sadrži sva obeležja iz $\mathcal{U}$, koja funkcionalno zavise od X. Saglasno tome, $B \in X^+_{\mathcal{F}}$ i $X \rightarrow B \in \mathcal{F}^+$ su ekvivalentna tvrđenja. Na slici 5.6 je prikazan algoritam za izračunavanje zatvaranja skupa obeležja X.

Teorema 5.2. Algoritam zatvaranja skupa obeležja je korektan.

Dokaz. Treba pokazati da važi $XPLUS = X^+$. Da važi $XPLUS \subseteq X^+$, sledi na osnovu pravila izvođenja $\mathfrak{I}_3$ (pseudotranzitivnost). Naime, pošto inicijalno važi funkcionalna zavisnost $X \rightarrow XPLUS$ i pošto svaka promena $XPLUS$ u algoritmu ne narušava ovu funkcionalnu zavisnost, jer $Y \subseteq XPLUS$ i $Y \rightarrow Z$ implicira $X \rightarrow XPLUS \cup Z$. Znači, za svako $A \in XPLUS$ važi $A \in X^+$.

Da bi se pokazalo da važi $X^+ \subseteq XPLUS$, polazi se od činjenice da je sistem aksioma $\mathfrak{I}$ kompletan, što znači da se putem $\mathfrak{I}$ može izračunati $\mathcal{F}^*$. Neka je $\mathcal{G}_k \subseteq \mathcal{F}^*$ skup svih logičkih posledica skupa funkcionalnih zavisnosti $\mathcal{F}$, za koje postoji niz izvođenja dužine

ne veće od k (k je proizvoljan prirodni broj). Indukcijom po dužini niza izvođenja će biti dokazana tvrdnja da za proizvoljno k važi:

(5.2) $$((\forall Y \rightarrow Z \in \mathcal{G}_k)(Y \subseteq XPLUS)) \Rightarrow Z \subseteq XPLUS.$$

ALGORITAM ZATVARANJA SKUPA OBELEŽJA

PROCES Zatvaranje
  Ulaz: $\mathcal{F}$, $\mathcal{U}$ i $X \subseteq \mathcal{U}$
  Izlaz: $XPLUS$
POČETAK PROCESA Zatvaranje
  POSTAVI $XPLUS \leftarrow X$
  POSTAVI $Q \leftarrow \emptyset$
  RADI form_X^+ *DOK JE* $Q \neq XPLUS$
    POSTAVI $Q \leftarrow XPLUS$
    RADI novo_obeležje $(\forall f \in \mathcal{F})$
      AKO JE $lhs(f) \subseteq XPLUS$ *TADA*
        POSTAVI $XPLUS \leftarrow XPLUS \cup rhs(f)$
      INAČE
      KRAJ AKO
    KRAJ RADI novo_obeležje
  KRAJ RADI form_X^+
KRAJ PROCESA Zatvaranje

Slika 5.6.

1. Tvrdnja važi za $k = 0$:

$\mathcal{G}_0$ sadrži, prema definiciji niza izvođenja, samo trivijalne zavisnosti i zavisnosti iz skupa $\mathcal{F}$, tako da implikacija (5.2) trivijalno važi.

2. Pretpostavimo da tvrđenje (5.2) važi za $\forall n \leq k$, gde je k prirodan broj i neka je dat skup posledica $\mathcal{G}_{k+1}$. Treba dokazati da važi:

(5.3) $$((\forall Y \rightarrow Z \in \mathcal{G}_{k+1})(Y \subseteq XPLUS)) \Rightarrow Z \subseteq XPLUS.$$

Neka je $Y \rightarrow Z$ u $\mathcal{G}_{k+1}$, i neka je $Y \subseteq XPLUS$. Ako je $Y \rightarrow Z$ u $\mathcal{G}_k$, (5.3) je zadovoljeno po pretpostavci indukcije. Ako $Y \rightarrow Z$ nije u $\mathcal{G}_k$, tada je $Y \rightarrow Z$ dobijeno primenom: ili pravila $\mathbb{I}_1$, ili pravila $\mathbb{I}_2$, ili pravila $\mathbb{I}_3$.

Ako je $Y \rightarrow Z$ dobijeno primenom pravila $\mathbb{I}_1$, tada je $Z \subseteq Y$, a pošto je, po pretpostavci $Y \subseteq XPLUS$, sledi $Z \subseteq XPLUS$. Ako je $Y \rightarrow Z$ dobijeno primenom pravila $\mathbb{I}_2$, tada u $\mathcal{G}_k$ postoji funkcionalna zavisnost $P \rightarrow Q$ i postoji $W \subseteq \mathcal{U}$ takvo, da je $Y = PW$, a $Z = QV$, za $V \subseteq W$. Pošto je $Y \subseteq XPLUS$ i pošto je po pretpostavci indukcije $Q \subseteq XPLUS$, onda je i $Z \subseteq XPLUS$, te se zaključuje da je (5.3) zadovoljeno. Ako je $Y \rightarrow Z$ dobijeno primenom pravila $\mathbb{I}_3$, tada u $\mathcal{G}_k$ postoje funkcionalne zavisnosti $Q \rightarrow V$ i $VW \rightarrow Z$, pri čemu je

$Y = QW$. Pošto je $Y \subseteq XPLUS$, i pošto su, po pretpostavci indukcije, V, W i Z takođe u $XPLUS$, zaključuje se da važi (5.3).

Pošto je dokazano da tvrdnja (5.2) važi za svaki prirodan broj k, a skup pravila je $\mathfrak{F}$ kompletan, može se tvrditi da važi implikacija

(5.4) $$(\forall Y \rightarrow Z \in \mathcal{F}^*)(Y \subseteq XPLUS \Rightarrow Z \subseteq XPLUS).$$

Ako se u (5.4) $Y \rightarrow Z$ zameni sa $X \rightarrow X^+$, dobija se

$$((X \rightarrow X^+ \in \mathcal{F}^*) \wedge (X \subseteq XPLUS)) \Rightarrow X^+ \subseteq XPLUS,$$

što je i trebalo dokazati, jer je pretpostavka ove implikacije uvek istinita. □

Primer 5.23. Neka je $\mathcal{F} = \{AB \rightarrow C,\ C \rightarrow A,\ BC \rightarrow D,\ ACD \rightarrow B,\ D \rightarrow EG,\ BE \rightarrow C,\ CG \rightarrow BD,\ CE \rightarrow AG\}$ i neka se traži $(BD)^+_{\mathcal{F}}$ primenom algoritma sa slike 5.6. Inicijalno je $XPLUS = BD$. Pošto je $D \rightarrow EG \in \mathcal{F}$ i $D \in XPLUS$, dobija se $XPLUS = BDEG$. U sledećem prolazu kroz $\mathcal{F}$ se utvrđuje da važi $BE \rightarrow C \in \mathcal{F}$, te je $XPLUS = BCDEG$, a u sledećem $C \rightarrow A \in \mathcal{F}$ i $C \in XPLUS$, te se dobija $XPLUS = ABCDEG$. Sledeći prolaz kroz $\mathcal{F}$ ne dovodi do promene $XPLUS$, te se zaključuje da je $(BD)^+_{\mathcal{F}} = ABCDEG$. □

Teorema 5.3. Ocena vremenske kompleksnosti algoritma zatvaranja skupa obeležja je $O(|\mathcal{U}|\,|\mathcal{F}|\,min\{|\mathcal{U}|, |\mathcal{F}|\})$.

Dokaz. Provera sadržavanja $lhs(f) \subseteq XPLUS$ se izvršava za svako f iz $\mathcal{F}$, te ocena vremenske kompleksnosti jednog prolaza kroz petlju "novo_obeležje" iznosi $O(|\mathcal{U}||\mathcal{F}|)$. Petlja "form_$X^+$" se izvršava najviše ili $|\mathcal{U}| + 1$, ili $|\mathcal{F}| + 1$ puta, pod pretpostavkom da se u svakoj iteraciji doda najviše jedno obeležje u $XPLUS$. Prema tome, ocena vremenske kompleksnosti algoritma zatvaranja skupa obeležja iznosi $O(|\mathcal{U}|\,|\mathcal{F}|\,min\{|\mathcal{U}|, |\mathcal{F}|\})$. □

U literaturi, na primer [BB, PBG, M], opisani su algoritmi za izračunavanje X^+, čija ocena kompleksnosti je linearna, s obzirom na dužinu ulaza $|\mathcal{U}|\,|\mathcal{F}|$. Da bi se to postiglo, potrebno je uvesti odredene pomoćne strukture podataka, koje obezbeđuju da se, pri izračunavanju X^+, svaka funkcionalna zavisnost testira i koristi najviše jedanput.

U svakom slučaju, bitno je zapaziti da algoritam na slici 5.6, za izračunavanje zatvaranja koristi $\mathcal{F}$, a ne $\mathcal{F}^+$ i da se implikacioni problem za funkcionalne zavisnosti može rešiti putem algoritma sa linearnom ocenom kompleksnosti.

Algoritam za izračunavanje zatvaranja skupa obeležja ima višestruku primenu u rešavanju implikacionog problema za funkcionalne zavisnosti. Koristi se pri:

- utvrđivanju ekvivalentnosti dva skupa funkcionalnih zavisnosti,
- redukciji levih strana funkcionalnih zavisnosti i
- traženju neredundantnog pokrivanja skupa funkcionalnih zavisnosti.

Da bi se proverila ekvivalentnost skupova funkcionalnih zavisnosti $\mathcal{F}$ i $\mathcal{G}$, potrebno je, za svako $f \in \mathcal{F}$ proveriti da li važi $rhs(f) \subseteq (lhs(f))^+_{\mathcal{G}}$ i za svako $g \in \mathcal{G}$ proveriti da li važi $rhs(g) \in (lhs(g))^+_{\mathcal{F}}$. Pozitivan ishod obe provere potvrđuje da su posmatrani skupovi funkcionalnih zavisnosti ekvivalentni.

Provere tipa $(X \setminus \{A\}) \rightarrow B \in \mathcal{F}^+$, pri redukciji, svode se na provere tipa $B \in (X \setminus \{A\})^+_{\mathcal{F}}$.

Provere tipa $f \in (\mathcal{F} \setminus \{f\})^+$, za $f: X \rightarrow B$, pri traženju neredudantnog pokrivanja, svode se na provere tipa $B \in X^+_{\mathcal{F} \setminus \{f\}}$.

Primer 5.24. Neka je dat skup funkcionalnih zavisnosti $\mathcal{F} = \{AB \rightarrow C, A \rightarrow B\}$. Da bi se proverilo da li je funkcionalna zavisnost $AB \rightarrow C$ redukovana, izračunava se $A^+_{\mathcal{F}} = \{A, B, C\}$. Pošto je $C \in A^+_{\mathcal{F}}$, zaključuje se da skup redukovanih funkcionalnih zavisnosti $\mathcal{G}$ ($\equiv \mathcal{F}$), sadrži funkcionalnu zavisnost $A \rightarrow C$, a ne sadrži $AB \rightarrow C$. □

Primer 5.25. Posmatra se skup funkcionalnih zavisnosti $\mathcal{G}$ iz primera 5.20. Da bi se proverilo da li je funkcionalna zavisnost $AC \rightarrow E$ suvišna, formira se $(AC)^+_{\mathcal{G} \setminus \{AC \rightarrow E\}} = \{A, C, B, D, E\}$. Pošto je $E \in \{A, C, B, D, E\}$, zaključuje se da je funkcionalna zavisnost $AC \rightarrow E$ suvišna (redudantna). □

5.2.6. *Projekcija skupa funkcionalnih zavisnosti*

Definicija 5.17. Projekcija (ili restrikcija) skupa funkcionalnih zavisnosti $\mathcal{F}$, definisanih u $\mathcal{U}$, na skup obeležja $W \subseteq \mathcal{U}$, u oznaci $\mathcal{F}|_W$, ili $\pi_W(\mathcal{F})$, je

$$\mathcal{F}|_W = \{X \rightarrow Y \mid (X \rightarrow Y \in \mathcal{F}^+) \wedge (XY \subseteq W)\}.$$

Ako je $f \in \mathcal{F}|_W$, kaže se da je funkcionalna zavisnost ***ugrađena*** u skup obeležja W. □

Primer 5.26. Neka je $\mathcal{U} = \{A, B, C, D\}$, $W = \{A, C, D\}$ i $\mathcal{F} = \{A \rightarrow B, BC \rightarrow D\}$ skup funkcioalnih zavisnosti nad skupom $\mathcal{U}$. Tada je $\mathcal{F}|_W = \{\varnothing \rightarrow \varnothing, A \rightarrow A, C \rightarrow C, D \rightarrow D, AC \rightarrow ACD, AD \rightarrow AD, CD \rightarrow CD, ACD \rightarrow ACD\}$, a kanonično pokrivanje $\mathcal{G}$ skupa $\mathcal{F}|_W$ je $\mathcal{G} = \{AC \rightarrow D\}$. Funkcionalna zavisnost $AC \rightarrow D$ pripada $\mathcal{F}^+$ na osnovu aksiome $\mathfrak{I}_3$ (pseudotranzitivnost). □

Teorema 5.4. Neka je r relacija nad $\mathcal{U}$, a $W \subseteq \mathcal{U}$. Ako funkcionalna zavisnost $X \rightarrow Y$, za $XY \subseteq W$, važi u $\pi_W(r)$, važi i u r.

Dokaz. Da bi se dokazalo tvrđenje teoreme, pretpostavlja se da $\pi_W(r)$ zadovoljava, a da r ne zadovoljava funkcionalnu zavisnost $X \rightarrow Y$. Tada u r postoje torke u i v takve da je $u[X] = v[X]$ i $u[Y] \neq v[Y]$. Međutim, pošto je $XY \subseteq W$, tada $\pi_W(r)$ sadrži $u[XY]$ i $v[XY]$, što znači da ni $\pi_W(r)$ ne zadovoljava funkcionalnu zavisnost $X \rightarrow Y$. □

5.2.7. *Armstrongova relacija*

Definicija 5.18. Neka je $\mathcal{F}$ skup funkcionalnih zavisnosti nad skupom obeležja $\mathcal{U}$. ***Armstrongova relacija*** za $\mathcal{F}$ je relacija nad $\mathcal{U}$ koja zadovoljava samo funkcionalne zavisnosti iz $\mathcal{F}^+$. □

Primer 5.27. Neka je $\mathcal{F} = \{AB \rightarrow C, C \rightarrow B\}$. Relacija r na slici 5.7. je Armstrongova relacija za $\mathcal{F}$, a relacija s nije Armstrongova relacija za $\mathcal{F}$, jer zadovoljava i funkcionalnu zavisnost $C \rightarrow A$, koja nije u $\mathcal{F}^+$. □

$r =$

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_1	c_1
a_2	b_2	c_3

$s =$

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_2	c_3

Slika 5.7.

Neka je dat skup funkcionalnih zavisnosti $\mathcal{F} \subseteq \mathcal{F}(\mathcal{U})$ i neka je $\mathcal{P}(\mathcal{U})$ partitivni skup skupa obeležja $\mathcal{U}$. Armstrongova relacija za skup funkcionalnih zavisnosti $\mathcal{F}$ se formira na sledeći način: elementi skupa $\mathcal{P}(\mathcal{U})$ se numerišu rednim brojevima. Neka je $\mathcal{P}(\mathcal{U}) = \{X_1, X_2, ..., X_k\}$. Za svaki element $X_i \in \mathcal{P}(\mathcal{U})$, za koji važi da je $X_i \neq \emptyset$ i $(X_i)^+_{\mathcal{F}} \neq \mathcal{U}$, formira se relacija $r_i(\mathcal{U})$, takva da sadrži dve torke: $r_i(\mathcal{U}) = \{t_{2i-1}, t_{2i}\}$, pri čemu je:

$$(\forall A \in \mathcal{U})(t_{2i-1}[A] = 2i - 1) \text{ i } (\forall A \in X_i{}^+_{\mathcal{F}})(t_{2i}[A] = 2i - 1) \wedge (\forall A \in \mathcal{U} \setminus X_i{}^+_{\mathcal{F}})(t_{2i}[A] = 2i).$$

U protivnom, ako je $(X_i)^+_{\mathcal{F}} = \mathcal{U}$, ili $X_i = \emptyset$, biće $r_i(\mathcal{U}) = \emptyset$. Formira se unija

$$r_A(\mathcal{U}) = \bigcup_{i=1}^{k} r_i(\mathcal{U}).$$

Za svaki $X_i \in \mathcal{P}(\mathcal{U})$ važi da odgovarajuća relacija $r_i(\mathcal{U})$ zadovoljava sve funkcionalne zavisnosti $f \in \mathcal{F}^+$, takve da je $lhs(f) = X_i$, a ne zadovoljava ni jednu funkcionalnu zavisnost $f \in \mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^+$, takvu da je $lhs(f) = X_i$. Pošto $r_A(\mathcal{U})$ predstavlja uniju relacija $r_i(\mathcal{U})$ po partitivnom skupu $\mathcal{P}(\mathcal{U})$, znači da su za formiranje skupova X_i uzete u obzir sve moguće kombinacije obeležja iz skupa $\mathcal{U}$. Imajući u vidu ovu, kao i činjenicu da su relacije $r_i(\mathcal{U})$ međusobno disjunktne, znači da $r_A(\mathcal{U})$ zadovoljava sve zavisnosti iz $\mathcal{F}^+$, a ne zadovoljava ni jednu zavisnost iz skupa $\mathcal{F}(\mathcal{U}) \setminus \mathcal{F}^+$. Na osnovu ovog obrazloženja, proizilazi da relacija r_A predstavlja traženu Armstrongovu relaciju.

5.3. *Funkcionalne zavisnosti kao skup ograničenja šeme relacije*

Često se skup ograničenja šeme relacije $N(R, C)$ svodi na skup funkcionalnih zavisnosti $\mathcal{F}$ u R, tako da se šema relacije definiše kao imenovana dvojka $N(R, \mathcal{F})$.

Primer 5.28. Šema relacije *Nastava* $(\{N, P, S\}, \{\gamma_1, \gamma_2\})$ iz primera 4.12 se može predstaviti kao *Nastava* $(\{N, P, S\}, \{N \rightarrow P, PS \rightarrow N\})$, jer je ograničenje:

- (γ_1) svaki nastavnik predaje samo jedan predmet, ekvivalentno sa funkcionalnom zavisnošću $N \rightarrow P$, a
- (γ_2) ako student sluša neki predmet, sluša ga kod samo jednog nastavnika, ekvivalentno sa funkcionalnom zavisnošću $PS \rightarrow N$. □

I ključ šeme relacije se može definisati putem funkcionalnih zavisnosti.

Definicija 5.19. Skup obeležja $X \subseteq R$ je ***ključ*** šeme relacije $(R, \mathcal{F})$ ako važi:

1° $X \rightarrow R \in \mathcal{F}^+$ i
2° $(\forall Y \subset X)(Y \rightarrow R \notin \mathcal{F}^+)$. □

ALGORITAM ZA IZRAČUNAVANJE JEDNOG KLJUČA

```
PROCES Izračunavanje_ključa
   Ulaz: (R, F)
   Izlaz: X          (* jedan ključ šeme relacije (R, F) *)
POČETAK PROCESA Izračunavanje_ključa
   POSTAVI X ← R
   RADI redukcija (∀A ∈ X)
      AKO JE A ∈ (X \{A})+ TADA
         POSTAVI X ← X \{A}
      INAČE
      KRAJ AKO
   KRAJ RADI redukcija
KRAJ PROCESA Izračunavanje_ključa
```

Slika 5.8.

Uslov 1° definicije 5.19 ukazuje na osobinu jedinstvenosti vrednosti ključa, a uslov 2° ukazuje na minimalnost skupa obeležja ključa šeme relacije. Na slici 5.8 je prikazan algoritam za izračunavanje jednog ključa šeme relacije $(R, \mathcal{F})$. Lako se uočava da korektnost ovog algoritma sledi na osnovu činjenice da on predstavlja direktnu realizaciju definicije 5.19. Ocena kompleksnosti ovog algoritma je $O(|R|^2|\mathcal{F}|min\{|R|, |\mathcal{F}|\})$.

Međutim, šema relacije $(R, \mathcal{F})$ može imati više ključeva. Svi oni su, međusobno, ***ekvivalentni***. Jedan od ekvivalentnih ključeva se bira za ***primarni***. Skup ključeva šeme relacije $(R, \mathcal{F})$ će se u daljem tekstu označavati sa $\mathcal{K}$.

Primer 5.29. Neka je $R = \{A, B, C, D, E, F\}$, a $\mathcal{F} = \{AB \rightarrow CDE, E \rightarrow A, CD \rightarrow B\}$. Skup $\mathcal{K} = \{AB, EB, ACD, ECD\}$ predstavlja skup ključeva šeme relacije $(R, \mathcal{F})$. □

Na slici 5.9 je prikazan jedan algoritam za izračunavanje svih ključeva šeme relacije $(R, \mathcal{F})$. Ovaj algoritam je zasnovan na opažanju da partitivni skup $\mathcal{P}(R)$ sadrži sve ključeve šeme relacije $(R, \mathcal{F})$ i da se proverom svakog od elementa skupa $\mathcal{P}(R)$ da li zadovoljava uslove 1° i 2° definicije 5.19, dolazi do skupa ključeva šeme relacije $(R, \mathcal{F})$.

ALGORITAM ZA IZRAČUNAVANJE SKUPA KLJUČEVA ŠEME RELACIJE

```
PROCES Izračunavanje_skupa_ključeva
   Ulaz: (R, F)
   Izlaz: K                (* skup ključeva šeme relacije (R, F) *)
POČETAK PROCESA Izračunavanje_skupa_ključeva
   POSTAVI K ← ∅
   RADI traži_ključ (∀X ⊆ R)
      AKO JE R = X+F TADA
         RADI redukcija (∀A ∈ X)
            AKO JE A ∈ (X \ {A})+F TADA
               POSTAVI X ← X \ {A}
            INAČE
            KRAJ AKO
         KRAJ RADI redukcija
         POSTAVI K ← K ∪ X
      INAČE
      KRAJ AKO
   KRAJ RADI traži_ključ
KRAJ PROCESA Izračunavanje_skupa_ključeva
```

Slika 5.9.

Lema 5.5. Algoritam za izračunavanje skupa svih ključeva je korektan, a ocena složenosti mu je eksponencijalna.

Dokaz. Algoritam je korektan, ako izračunava sve ključeve šeme relacije $(R, \mathcal{F})$. Treba prvo zapaziti da, ako je $\mathcal{F} = \emptyset$, tada je R jedini ključ šeme relacije $(R, \emptyset)$. Pošto je $R \subseteq R$, algoritam ga sigurno otkriva. Dalje, pošto se u iterativnom bloku pod nazivom "traži_ključ", analizira svaki podskup skupa R i nalazi njegovo zatvaranje, a u iterativnom bloku "redukcja_X", se redukuju svi oni podskupovi X skupa R, za koje važi $X^+ = R$, zaključuje se da će $\mathcal{K}$ sadržati sve $X \subseteq R$ takve, da zadovoljavaju uslove 1° i 2° definicije 5.19.

U algoritmu se zatvaranje izračunava za svaki od $2^{|R|}$ podskupova skupa R. Ocena kompleksnosti postupka za izračunavanje zatvaranja je $O(|R||\mathcal{F}|(min\{|R|, |\mathcal{F}|\})$. Redukcija se primenjuje na one podskupove, za koje važi $X^+ = R$. Broj takvih podskupova je veći ili jednak *1*, a manji od $2^{|R|}$. Ocena kompleksnosti postupka redukcije je $O(|R|^2|\mathcal{F}|(min\{|R|, |\mathcal{F}|\}))$. Sledi da je ocena kompleksnosti algoritma $O(|R|^3|\mathcal{F}|^2(min\{|R|, |\mathcal{F}|\})^2 2^{|R|})$. □

Ozbiljan nedostatak algoritma sa slike 5.9 leži u činjenici da mu je ocena kompleksnosti eksponencijalna bez obzira na kardinalni broj skupa $\mathcal{K}$. Ideja za konstruisanje algoritma, čija kompleksnost će zavisiti od kardinalnog broja skupa $\mathcal{K}$, zasnovana je na

opažanju da skup obeležja, oblika $V(X \setminus W)$, gde su $V, W \subseteq R$, a $X \in \mathcal{K}$, predstavlja jedan superključ šeme relacije $(R, \mathcal{F})$.

Lema 5.6. Neka je $\mathcal{K}$ skup ključeva šeme relacije $(R, \mathcal{F})$. Tada važe sledeće implikacije:

a° $(X \in \mathcal{K}) \wedge (V \subseteq R) \Rightarrow R = (V(X \setminus V^+))^+$,
b° $(X \in \mathcal{K}) \wedge (V \rightarrow W) \Rightarrow R = (V(X \setminus W))^+$.

Dokaz.

a° Na osnovu definicije 5.10, važi $V \rightarrow V^+ \in \mathcal{F}^+$. Na osnovu pravila izvođenja $\mathfrak{F}_2$, sledi $V(X \setminus V^+) \rightarrow (X \setminus V^+)V^+ \in \mathcal{F}^+$, odnosno $V(X \setminus V^+) \rightarrow X \in \mathcal{F}^+$, te se zaključuje da je $V(X \setminus V^+)$ jedan superključ šeme relacije $(R, \mathcal{F})$.

b° Ako se funkcionalna zavisnost $V \rightarrow W$ proširi, primenom pravila izvođenja $\mathfrak{F}_2$, sa $(X \setminus W)$, dobija se $V(X \setminus W) \rightarrow X$. Pošto je X ključ šeme relacije $(R, \mathcal{F})$, zaključuje se da je $V(X \setminus W)$ jedan superključ šeme relacije $(R, \mathcal{F})$. □

Teorema 5.5. Neka je $\mathcal{K}$ neprazan skup ključeva šeme relacije $(R, \mathcal{F})$. Skup $\mathcal{P}(R) \setminus \mathcal{K}$ sadrži bar još jedan ključ šeme relacije $(R, \mathcal{F})$, ako i samo ako važi

$$(5.5) \qquad ((\exists V \rightarrow W \in \mathcal{F}) \wedge (\exists X \in \mathcal{K}))(\forall Y \in \mathcal{K})(Y \not\subseteq V(X \setminus W)).$$

Dokaz. ($\Rightarrow$) Neka važi (5.5). Tada dokaz "ako" dela teoreme 5.5 sledi na osnovu dokaza implikacije **b°** leme 5.6. Naime, pošto je $V(X \setminus V)$ superključ šeme relacije $(R, \mathcal{F})$ i pošto važi $V(X \setminus W) \notin \mathcal{K}$, zaključuje se da u $\mathcal{P}(R) \setminus \mathcal{K}$ postoji $Z \subseteq V(X \setminus W)$, koje predstavlja ključ šeme relacije $(R, \mathcal{F})$, ali nije u $\mathcal{K}$.

($\Leftarrow$) Sada treba pokazati da

$$(5.6) \qquad (\exists Z \in (\mathcal{P}(R) \setminus \mathcal{K}))(\forall Y \in \mathcal{K})((Y \not\subseteq Z) \wedge (Z \rightarrow R \in \mathcal{F}^+))$$

implicira (5.5).

Prvo će biti ukazano na jednu osobinu ključeva šeme relacije, koja je bitna za dokazivanje samo ako dela teoreme. Neka su Z i Y dva ključa šeme relacije $(R, \mathcal{F})$. Da bi se utvrdilo da važi $Z \rightarrow Y \in \mathcal{F}^+$, koristi se niz funkcionalnih zavisnosti $f_1, \ldots, f_n$ iz $\mathcal{F}^+$. Svaka funkcionalna zavisnost ovog niza pripada ili skupu $\mathcal{G} = \{f \mid (lhs(f) \subseteq Z) \wedge (f \in \mathcal{F} \vee lhs(f) = rhs(f)\}$ ili skupu $\mathcal{J} = \{f \mid f$ je posledica primene pravila izvođenja $\mathfrak{F}_2$ i $\mathfrak{F}_3\}$ ili jednom od skupova $\mathcal{H}_i = \{f \mid \mathcal{P}(f) \wedge \mathcal{P}_i(f)\}$, $i = 1, \ldots, k$. Pri tome je $\mathcal{P}(f) = (lhs(f) \cap Z = \varnothing) \wedge (\exists A \in Y)(A \in (lhs(f))^+)$, $\mathcal{P}_1(f) = (\exists g \in \mathcal{G})(lhs(f) \subseteq rhs(g))$, a $\mathcal{P}_i(f) = (\exists g \in \mathcal{H}_{i-1})(lhs(f) \subseteq rhs(g))$, za $i = 2, \ldots, k$. Pri tome važi

$$Z = \bigcup_{f \in \mathcal{G}} lhs(f).$$

Na osnovu leme 5.6 **b°** sledi da je

$$\bigcup_{f \in \mathcal{H}_k} lhs(f) \left(Y \setminus \bigcup_{f \in \mathcal{H}_k} rhs(f) \right)$$

jedan superključ šeme relacije $(R, \mathcal{F})$. Neka je

$$Y_1 \subseteq \bigcup_{f \in \mathcal{H}_k} lhs(f)\left(Y \setminus \bigcup_{f \in \mathcal{H}_k} rhs(f)\right)$$

ključ. Za izvođenje zaključka da važi $Z \rightarrow Y_1 \in \mathcal{F}^+$, nisu potrebne funkcionalne zavisnosti iz $\mathcal{H}_k$. Isto tako je

$$\bigcup_{f \in \mathcal{H}_{k-1}} lhs(f)\left(Y_1 \setminus \bigcup_{f \in \mathcal{H}_{k-1}} rhs(f)\right)$$

jedan superključ, a

$$Y_2 \subseteq \bigcup_{f \in \mathcal{H}_{k-1}} lhs(f)\left(Y_1 \setminus \bigcup_{f \in \mathcal{H}_{k-1}} rhs(f)\right)$$

ključ šeme relacije $(R, \mathcal{F})$. Za izvođenje zaključka da važi $Z \rightarrow Y_2 \in \mathcal{F}^+$, nisu potrebne funkcionalne zavisnosti ni iz $\mathcal{H}_k$ ni iz $\mathcal{H}_{k-1}$. Opisanim postupkom se identifikuju i ključevi $Y_3, \ldots, Y_k$. Za izvođenje zaključka da važi $Z \rightarrow Y_k \in \mathcal{F}^+$, dovoljne su samo funkcionalne zavisnosti iz $\mathcal{G}$ i $\mathcal{J}$. Saglasno tome,

$$Y_k \subseteq \bigcup_{f \in \mathcal{G}} rhs(f).$$

Pretpostavlja se da važi (5.6). Neka je $Z \in (\mathcal{P}(R) \setminus \mathcal{K})$, $R = Z^+$, neka važi $(\forall Y \in \mathcal{K})(Y \not\subseteq Z)$ i neka je Y_k u $\mathcal{K}$, a $f: V \rightarrow W$ netrivijalna funkcionalna zavisnost iz $\mathcal{G}$. Tada je

$$\bigcup_{g \in \mathcal{G} \wedge g \neq f} lhs(g)\left(Y_k \setminus \bigcup_{g \in \mathcal{G} \wedge g \neq f} rhs(g)\right)$$

takođe superključ šeme relacije $(R, \mathcal{F})$. Neka je $X \in \mathcal{K}$ i

$$X \subseteq \bigcup_{g \in \mathcal{G} \wedge g \neq f} lhs(g)\left(Y_k \setminus \bigcup_{g \in \mathcal{G} \wedge g \neq f} rhs(g)\right)$$

Pošto je

$$lhs(f)\left(\left(\bigcup_{g \in \mathcal{G} \wedge g \neq f} lhs(g)\left(Y_k \setminus \bigcup_{g \in \mathcal{G} \wedge g \neq f} rhs(g)\right)\right) \setminus rhs(f)\right) \subseteq \bigcup_{g \in \mathcal{G}} lhs(g)$$

zaključuje se da je $V(X \setminus W) \subseteq Z$, čime je dokaz teoreme okončan. □

Primer 5.30. Neka je $R = \{A, B, C, D, E, F, G\}$, $\mathcal{F} = \{A \rightarrow E, B \rightarrow C, C \rightarrow D, D \rightarrow F, E \rightarrow G, FG \rightarrow AB\}$, a $\mathcal{K} = \{BE, EC, FG\}$. Da bi se ilustrovala primena ako dela teoreme 5.5, bira se funkcionalna zavisnost $E \rightarrow G$ iz $\mathcal{F}$. Skup obeležja $E(FG \setminus G)$ predstavlja ključ šeme relacije $(R, \mathcal{F})$, koji nije u $\mathcal{K}$.

Da bi se ilustrovala ideja "samo ako" dela teoreme 5.5, iz $\mathcal{P}(R) \setminus \mathcal{K}$ se bira skup obeležja AB. Lako se proverava da AB zadovoljava uslov $(\forall Y \in \mathcal{K})((Y \not\subseteq AB) \wedge (AB \rightarrow R \in \mathcal{F}^+))$. Znači, $Z = AB$ i neka je $Y = FG$. Za izvođenje zaključka da je $AB \rightarrow FG \in \mathcal{F}^+$ koristi se sledeći niz funkcionalnih zavisnosti:

- $A \rightarrow E$ i $E \rightarrow G$ daju $A \rightarrow G$,
- $B \rightarrow C$ i $C \rightarrow D$ i $D \rightarrow F$ daju $B \rightarrow F$,
- $B \rightarrow B$ i $A \rightarrow G$ daju $AB \rightarrow BG$,
- $B \rightarrow F$ i $AB \rightarrow BG$ daju $AB \rightarrow FG$.

Pri tome je:

- $\mathcal{G} = \{A \rightarrow E, B \rightarrow C, B \rightarrow B\}$,
- $\mathcal{J} = \{A \rightarrow G, B \rightarrow F, AB \rightarrow BG, AB \rightarrow FG\}$,
- $\mathcal{H}_1 = \{C \rightarrow D, E \rightarrow G\}$,
- $\mathcal{H}_2 = \{D \rightarrow F\}$.

Primena funkcionalne zavisnosti $D \rightarrow F$ na ključ FG, daje $Y_1 = D(FG \setminus F) = DG$, a primena funkcionalne zavisnosti $CE \rightarrow DG$ na ključ DG, daje $Y_2 = CE(DG \setminus DG) = CE$. Neka je $V \rightarrow W = A \rightarrow E$. Tada je $X \subseteq B(EC \setminus BC) = BE$, a $Z = A(BE \setminus E) = AB$. □

Primer 5.31. Neka je $R = \{A, B, C, D, E, F, G, H\}$, a $\mathcal{F} = \{AB \rightarrow E, BC \rightarrow F, AC \rightarrow G, EFG \rightarrow ABC, AG \rightarrow C\}$, a inicijalni skup ključeva $\mathcal{K} = \{ABDG, BCDEG, DEFG\}$. Skup obeležja $Z = ABCD$ takođe predstavlja ključ šeme relacije $(R, \mathcal{F})$. Neka je $Y = DEFG$. Da bi se izveo zaključak da važi $ABCD \rightarrow DEFG \in \mathcal{F}^+$, koriste se funkcionalne zavisnosti:

- $AB \rightarrow E$ i $CD \rightarrow CD$ daju $ABCD \rightarrow CDE$,
- $AC \rightarrow G$ i $ABCD \rightarrow CDE$ daju $ABCD \rightarrow CDEG$,
- $BC \rightarrow F$ i $ABCD \rightarrow CDEG$ daju $ABCD \rightarrow CDEFG$.

Tako se dobijaju skupovi $\mathcal{G} = \{AB \rightarrow E, AC \rightarrow G, CD \rightarrow CD, BC \rightarrow F\}$ i $\mathcal{J} = \{ABCD \rightarrow CDE, B \rightarrow F, AB \rightarrow BG, AB \rightarrow FG\}$. Skupovi $\mathcal{H}_i$, za $i = 1,..., k$ su prazni.

Neka je $f: V \rightarrow W = AC \rightarrow G$. Tada je $X \subseteq ABCD(DEFG \setminus CDEF) = ABCDG$. U $\mathcal{K}$ se nalazi ključ $ABDG$, te se zaključuje da je $X = ABDG$. Primenom funkcionalne zavisnosti na ključ X, saglasno lemi 5.6 **b°**, dobija se $Z = AC(ABDG \setminus G) = ABCD$. □

Na slici 5.10 prikazan je algoritam za generisanje dodatnih ključeva šeme relacije $(R, \mathcal{F})$. Pretpostavka je da je inicijalni, jednočlani, skup ključeva generisan putem algoritma sa slike 5.8. U algoritmu se koristi operacija $red(W, \mathcal{F})$, koja predstavlja poziv algoritma redukcije i njegovu primenu na skup obeležja W. Rezultat je redukcija skupa obeležja W, s obzirom na skup funkcionalnih zavisnosti $\mathcal{F}$.

Teorema 5.6. Algoritam za generisanje dodatnih ključeva je korektan, a složenost mu je polinomijalna.

Dokaz. Algoritam za generisanje dodatnih ključeva se završava nakon konačnog broja koraka, jer su skupovi $\mathcal{F}$ i $\mathcal{K}$ konačni. Korektan je, ako generiše sve ključeve šeme relacije $(R, \mathcal{F})$. Da je to tako, sledi na osnovu teoreme 5.5.

Ocena kompleksnosti algoritma je $O(|\mathcal{F}| \, |\mathcal{K}|^2)$. Naime, kompleksnost algoritma je određena ugnježđenim iterativnim blokovima:

- "traži_ključ", koji se izvršava $|\mathcal{K}|$ puta,
- "traži_fz", koji se izvršava $|\mathcal{F}|$ puta i
- "proveri_ključ", koji se izvršava $|\mathcal{K}|$ puta. □

ALGORITAM ZA GENERISANJE DODATNIH KLJUČEVA

```
PROCES Generisanje_dodatnih_ključeva
   Ulaz: (R, F), K1          (* K1 je skup ključeva šeme relacije (R, F) *)
   Izlaz: K                  (* K je skup svih ključeva šeme relacije (R, F) *)
POČETAK PROCESA Generisanje_dodatnih_ključeva
   POSTAVI K ← K1
   RADI traži_ključ (∀X ∈ K)
      RADI traži_fz (∀V→W ∈ F)
         POSTAVI t ← 0
         RADI proveri_ključ (∀Y ∈ K) DOK JE t = 0
            AKO JE Y ⊆ V(X \ W) TADA
               POSTAVI t ← 1
            INAČE
            KRAJ AKO
         KRAJ RADI proveri_ključ
         AKO JE t = 0 TADA
            POSTAVI K ← K ∪ red(V(X \ W), F)
         INAČE
         KRAJ AKO
      KRAJ RADI traži_fz
   KRAJ RADI traži_ključ
KRAJ PROCESA Generisanje_dodatnih_ključeva
```

Slika 5.10.

Ocena kompleksnosti algoritma sa slike 5.10 je kvadratna s obzirom na kardinalni broj skupa ključeva šeme relacije. Međutim, postoje šeme relacija, kod kojih je kardinalni broj skupa ključeva eksponencijalna funkcija broja obeležja u skupu R, kao što to sledeći primer ilustruje.

Primer 5.32. Može se pokazati da šema relacije $(R, \mathcal{F})$, gde je $|R| = |\mathcal{F}| = 2n$, može imati 2^n ključeva. Za $n = 1$, šema relacije $(\{A_1, A_2\}, \{A_1 \rightarrow A_2, A_2 \rightarrow A_1\})$ ima dva ključa. To su A_1 i A_2. Za $n = 2$, $R_2 = \{A_1, A_2, A_3, A_4\}$ i $\mathcal{F} = \{A_1 \rightarrow A_2, A_2 \rightarrow A_1, A_1A_3 \rightarrow R_2, A_1A_4 \rightarrow R_2\}$, skup ključeva je $\mathcal{K} = \{A_1A_3, A_1A_4, A_2A_3, A_2A_4\}$. Za $n = 3$, $R_3 = \{A_i \mid i = 1, 2, ..., 6\}$ i $\mathcal{F}_3 = \{A_1 \rightarrow A_2, A_2 \rightarrow A_1, A_1A_3 \rightarrow R_3 \setminus \{A_5, A_6\}, A_1A_4 \rightarrow R_3 \setminus \{A_5, A_6\}, A_1A_3A_5 \rightarrow R_3, A_1A_3A_6 \rightarrow R_3\}$, dobija se $\mathcal{K} = \{A_1A_3A_5, A_1A_3A_6, A_1A_4A_5, A_1A_4A_6, A_2A_3A_5, A_2A_3A_6, A_2A_4A_5, A_2A_4A_6\}$. Pretpostavlja se da šema relacije sa $|R_k| = |\mathcal{F}_k| = 2k$, konstruisana prikazanim postupkom postupnog proširivanja skupova R i $\mathcal{F}$, ima skup ključeva $\mathcal{K}_k = \{X_i \mid i = 1, 2, ..., 2^k\}$. Neka je $n = k + 1$, $R_{k+1} = \{A_i \mid i = 1, 2, ..., 2k + 2\}$, a $\mathcal{F}_{k+1} = \{f_1, f_2, f_{2k}, X_1A_{2k+1} \rightarrow R_{k+1}, X_1A_{2k+2} \rightarrow R_{2k+2}\}$. Skupovi obeležja X_1A_{2k+1} i $X_1A_{2(k+1)}$ predstavljaju ključeve šeme relacije $(R_{k+1}, \mathcal{F}_{k+1})$. Pošto važi $(\forall X_i \in \mathcal{K}_k)(X_i \rightarrow X_1 \in F_{k+1}^+)$, zaključuje se da šema relacije $(R_{k+1}, \mathcal{F}_{k+1})$ ima 2^{k+1} ključeva. □

Na osnovu primera 5.34 se može zaključiti da je traženje dodatnih ključeva šeme relacije, u krajnjem, mada samo iznimnom, slučaju, zadatak sa eksponencijalnom ocenom kompleksnosti.

Definicija 5.20. Ako je X jedan ključ šeme relacije $(R, \mathcal{F})$, tada važi funkcionalna zavisnost $X \rightarrow R$. Ta funkcionalna zavisnost se naziva ***zavisnošću ključa***. □

Neka je $\mathcal{K}$ skup ključeva šeme relacije $(R, \mathcal{F})$, a $\mathcal{F}$ jedno redukovano i neredundantno pokrivanje. Ako važi

$$(\forall f \in \mathcal{F})(lhs(f) \in \mathcal{K}),$$

tada $(R, \mathcal{F})$ i $(R, \mathcal{K})$ predstavljaju ravnopravne notacije za istu šemu relacije, jer je svaka funkcionalna zavisnost f iz $\mathcal{F}$ reprezentovana jednom zavisnošću ključa oblika $X \rightarrow R$, za $X \in \mathcal{K}$.

Primer 5.33. Šema relacije

$$Student(\{BRI, IME, PRZ, BPI, MBG\}, \{BRI, MBG\}),$$

gde je MBG matični broj građana, ravnopravno zamenjuje notaciju

$$Student(\{BRI, IME, PRZ, BPI, MBG\}, \{BRI \rightarrow IME + PRZ + BPI + MBG, MBG \rightarrow BRI\}).$$ □

Primer 5.34. Šema relacije $(\{N, P, S\}, \{N \rightarrow P, PS \rightarrow N\})$ ima dva ključa To su NS i PS. Međutim, leva strana funkcionalne zavisnosti $N \rightarrow P$ se ne nalazi u skupu ključeva, tako da šema relacije $(\{N, P, S\}, \{NS, PS\})$ ne reprezentuje isti deo realnog sveta. Naime, semantika funkcionalne zavisnosti $NS \rightarrow P$ je "ako profesor predaje studentu, predaje mu samo jedan predmet", dok je semantika funkcionalne zavisnosti $N \rightarrow P$ "svaki nastavnik predaje samo jedan predmet".

Da bi se isti realni sistem opisao postupkom, kod kojeg se šema relacije definiše putem skupa obeležja i skupa ključeva, bile bi neophodne dve šeme relacije. To su:

$$(\{N, P, S\}, \{PS\}) \text{ i } (\{N, P\}, \{N\}).$$

Prva ukazuje da:

- svaki student sluša određeni predmet kod samo jednog nastavnika,

a druga da:

- svaki nastavnik predaje samo jedan predmet.

Da je izostavljena druga šema relacije, izgubio bi se odnos između nastavnika i predmeta. Takođe treba zapaziti da u prvoj šemi relacije, NS nije definisano kao ključ, inače bi funkcionalne zavisnosti $NS \rightarrow P$ i $N \rightarrow P$ bile kolizione, nosile bi različitu semantiku. Međutim, sada je funkcionalna zavisnost $N \rightarrow P$ ograničenje, ugrađeno samo u šemu relacije $(\{N, P\}, \{N\})$, a ne i u šemu relacije $(\{N, P, S\}, \{PS\})$. To dalje znači da se u relaciju nad NPS mogu upisati torke u i v takve, da je $u[N] = v[N]$ i $u[P] \neq v[P]$. Da bi se to sprečilo, mora se uvesti sledeće međurelaciono ograničenje

$$(\forall t \in r(NPS))(\exists u \in r(NP))(t[NP] = u).$$

Ovo ograničenje se naziva zavisnošću sadržavanja i opisano je u daljem tekstu ovog poglavlja. □

5.4. Višeznačna zavisnost

Funkcionalne zavisnosti predstavljaju najvažniji, ali ne i jedini, tip relacionog ograničenja u teoriji i praksi relacionih baza podataka. Drugi tip relacionog ograničenja, koji takođe zaslužuje određenu pažnju, je višeznačna zavisnost.

Neka je r relacija nad $\mathcal{U}$, a X i Y takvi podskupovi skupa obeležja $\mathcal{U}$, da je $\mathcal{U} \setminus XY \neq \emptyset$. Intuitivno značenje višeznačne zavisnosti između skupova obeležja X i Y, u oznaci $X \rightarrow\rightarrow Y$, je da je svakoj X vrednosti pridružen po jedan skup Y vrednosti i da taj skup Y vrednosti ne zavisi od skupa $(\mathcal{U} \setminus XY)$ vrednosti. Kaže se da X višeznačno određuje Y, ili da je Y višeznačno određeno putem X.

Primer 5.35. Posmatra se skup obeležja $\{D, M, R\}$, gde je D oznaka za skup obeležja, koji opisuje klasu entiteta *Deo*, M oznaka za skup obeležja klase entiteta *Mašina*, a R oznaka za skup obeležja klase entiteta *Radnik*. Odnosi između entiteta ovih klasa je opisan sledećom rečenicom:

"Ako radnik r radi na mašini m, tada proizvodi svaki deo d, koji se izrađuje na mašini m."

Može se zaključiti da između skupa radnika, koji rade na mašini m i skupa delova, koji se proizvode na mašini m ne postoji bilo kakva uslovljenost, te da, nad skupom $\{D, M, R\}$ važi višeznačna zavisnost $M \rightarrow\rightarrow R$.

Međutim, na različitim mašinama se proizvode različiti skupovi delova. Radnik, radeći na određenim mašinama, na svakoj izrađuje odgovarajuće skupove delova. Zaključuje se da elementi onog skupa mašina i onog skupa delova, koji su povezani sa jednim radnikom, nisu nezavisni, te da višeznačna zavisnost $R \rightarrow\rightarrow M$ i $R \rightarrow\rightarrow D$, nad skupom $\{D, M, R\}$ ne važe.

Da je odnos između entiteta posmatranih klasa bio opisan na sledeći način:

"Radnik r izrađuje na mašini m samo neke od delova d, koji se na mašini m proizvode",

tada ne bi važila ni višeznažna zavisnost $M \rightarrow\rightarrow R$ ni zavisnost $M \rightarrow\rightarrow D$. □

Primer 5.36. Nad skupom obeležja $\{SMER, STUDENT, PREDMET\}$ važe višeznačne zavisnosti $SMER \rightarrow\rightarrow STUDENT$ i $SMER \rightarrow\rightarrow PREDMET$, ako svaki student smera sluša svaki predmet na smeru. S druge strane, ako student može biti upisan na samo jedan smer, tada važe i višeznačne zavisnosti $STUDENT \rightarrow\rightarrow SMER$ i $STUDENT \rightarrow\rightarrow PREDMET$. Poslednji zaključak sugeriše da višeznačna zavisnost $X \rightarrow\rightarrow Y$ predstavlja uopštenje funkcionalne zavisnosti $X \rightarrow Y$. □

Definicija 5.21. Neka je r jedna relacija nad $\mathcal{U}$, a X i Y poskupovi skupa obeležja $\mathcal{U}$. Relacija r zadovoljava ***višeznačnu zavisnost*** $X \rightarrow\rightarrow Y$, ako, za svake dve torke t, $s \in r$ takve da je $t[X] = s[X]$, postoje torke u, $v \in r$ takve, da važi:

1° $t[X] = u[X] = v[X] = s[X]$,
2° $t[Y] = u[Y]$ i $s[\mathcal{U}\setminus XY] = u[\mathcal{U}\setminus XY]$ i
3° $s[Y] = v[Y]$ i $t[\mathcal{U}\setminus XY] = v[\mathcal{U}\setminus XY]$. □

Definicija 5.21 ukazuje da, ako relacija r nad $\mathcal{U}$ zadovoljava višeznačnu zavisnost $X\rightarrow\rightarrow Y$, tada torke, dobijene zamenom Y vrednosti izmedu torki t i s, takode moraju postojati u r. Pri tome, X i Y ne moraju biti nepresečni skupovi obeležja. Za razliku od funkcionalne zavisnosti $X\rightarrow Y$, višeznačna zavisnost $X\rightarrow\rightarrow Y$ nije nezavisna od skupa obeležja $\mathcal{U}\setminus XY$. To dalje znači da višeznačna zavisnost ne predstavlja jednostavnu konstataciju da je jednoj X vrednosti pridružen neki skup Y vrednosti, već i da taj skup Y vrednosti mora biti invarijantan u odnosu na skup $\mathcal{U}\setminus XY$ vrednosti, pridružen istoj X vrednosti u relaciji r. Skup $\mathcal{U}\setminus XY$ je neprazan.

Treba zapaziti da se, u definiciji 5.21, umesto $s[X] = u[X]$ i $s[\mathcal{U}\setminus XY] = u[\mathcal{U}\setminus XY]$, može pisati $s[X(\mathcal{U}\setminus Y)] = u[X(\mathcal{U}\setminus Y)]$, jer je $X(\mathcal{U}\setminus XY) = X(\mathcal{U}\setminus Y)$. Takode treba zapaziti da je pominjanje i torke u i torke v u definiciji 5.21 suvišno, jer uslov mora važiti za svako t i s iz r, tako da postojanje torke u potvrduje i postojanje torke v.

Definicija 5.22. Višeznačna zavisnost $X\rightarrow\rightarrow Y$ je trivijalna, ako je zadovoljen jedan od uslova:

1° $Y\subseteq X$ ili
2° $XY = \mathcal{U}$. □

Trivijalna višeznačna zavisnost ne predstavlja ograničenje, jer je zadovoljava svaka relacija nad $\mathcal{U}$.

D	M	R
d_1	m_1	r_1
d_2	m_1	r_1
d_3	m_1	r_1
d_1	m_1	r_2
d_2	m_1	r_2
d_3	m_1	r_2
d_1	m_2	r_2
d_4	m_2	r_2
d_5	m_3	r_3

Slika 5.11.

STUD	*SMER*	*PRED*
s_1	a	p_1
s_1	a	p_2
s_2	a	p_1
s_2	a	p_2
s_3	b	p_1
s_3	b	p_3
s_4	c	p_4

Slika 5.12.

Primer 5.37. Na slici 5.11 je prikazana jedna relacija nad skupom $\{D, M, R\}$, koja zadovoljava višeznačne zavisnosti $M\rightarrow\rightarrow R$ i $M\rightarrow\rightarrow D$. Neka je $t = \{(M, m_1), (D, d_1), (R, r_1)\}$, a $s = \{(M, m_1), (D, d_2), (R, r_2)\}$, lako se proverava da relacija sadrži i torke $\{(M, m_1), (D, d_2), (R, r_1)\}$, $\{(M, m_1), (D, d_1), (R, r_2)\}$.

Na slici 5.12 je prikazana jedna relacija nad skupom obeležja {*STUDENT*, *SMER*, *PREDMET*}, koja zadovoljava višeznačne zavisnosti $SMER \rightarrow\rightarrow STUDENT$, $SMER \rightarrow\rightarrow PREDMET$, $STUDENT \rightarrow\rightarrow SMER$ i $STUDENT \rightarrow\rightarrow PREDMET$. Da bi se proverila važnost višeznačne zavisnosti $STUDENT \rightarrow\rightarrow SMER$, neka je $t = \{(STUDENT, s_1), (SMER, a), (PREDMET, p_1)\}$, a $s = \{(STUDENT, s_1), (SMER, a), (PREDMET, p_2)\}$, tada je $u = s$ i $v = t$. Višeznačna zavisnost $STUDENT \rightarrow\rightarrow SMER$ je, u stvari, funkcionalna zavisnost. □

Primeri 5.38 i 5.39 ukazuju, između ostalog, da se, kao i u slučaju funkcionalnih zavisnosti, može govoriti o logičkim posledicama višeznačnih zavisnosti. Logička posledica višeznačne zavisnosti $X \rightarrow\rightarrow Y$ je višeznačna zavisnost $X \rightarrow\rightarrow \mathcal{U} \setminus XY$, a logička posledica funkcionalne zavisnosti $V \rightarrow W$ je višeznačna zavisnost $V \rightarrow\rightarrow \mathcal{U} \setminus VW$. Na osnovu važenja višeznačne zavisnosti $V \rightarrow\rightarrow \mathcal{U} \setminus VW$, se pak zaključuje da funkcionalna zavisnost $V \rightarrow W$ implicira višeznačnu zavisnost $V \rightarrow\rightarrow W$. Sve ove logičke posledice ukazuju na komplementarnost višeznačne zavisnosti, što se označava sa $X \rightarrow\rightarrow Y \mid \mathcal{U} \setminus XY$.

Međutim, izvođenje zaključaka o logičkim posledicama skupa višeznačnih zavisnosti i o interakciji između funkcionalnih i višeznačnih zavisnosti, je težak zadatak. To ukazuje na potrebu definisanja sistema aksioma za formalno izvođenje zaključaka o posledicama zadatog skupa višeznačnih i funkcionalnih zavisnosti.

5.4.1. *Sistem aksioma za funkcionalne i višeznačne zavisnosti*

Na slici 5.13 je prikazan jedan sistem aksioma, označen sa $\mathfrak{M}$, za izvođenje zaključaka o posledicama zadatog skupa višeznačnih i funkcionalnih zavisnosti $\mathcal{O}$, definisanog nad skupom obeležja $\mathcal{U}$. Aksiome $\mathfrak{I}_1$, $\mathfrak{I}_2$ i $\mathfrak{I}_3$ su ovde uključene u cilju kompletnosti. Upoređujući pravila $\mathfrak{I}_1$ do $\mathfrak{I}_3$ sa pravilima $\mathfrak{M}_1$ do $\mathfrak{M}_3$, može se zaključiti:

- da kod višeznačnih zavisnosti ne važi pravilo refleksivnosti, dok kod funkcionalnih zavisnosti ne važi pravilo komplementarnosti,
- da je pravilo tranzitivnosti kod višeznačnih zavisnosti strože, nego kod funkcionalnih zavisnosti.

Tvrđenje aksiome $\mathfrak{M}_1$, $X \rightarrow\rightarrow Y \Rightarrow X \rightarrow\rightarrow \mathcal{U} \setminus XY$, se, u literaturi, skraćeno zapisuje na sledeći način $X \rightarrow\rightarrow Y \mid \mathcal{U} \setminus XY$.

Neredundantnost sistema aksioma $\mathfrak{M}$ se može dokazati primenom, u principu, sličnog postupka kao i pri dokazivanju neredundantnosti sistema aksioma $\mathfrak{I}$ u lemi 5.1. U cilju ilustracije ove tvrdnje, biće pokazano da ne važe tvrđenja:

$$\{SMER \rightarrow\rightarrow STUDENT\} \vdash_{\mathfrak{M} \setminus \{\mathfrak{M}_1\}} \{SMER \rightarrow\rightarrow PREDMET\} \text{ i}$$
$$\{STUDENT \rightarrow SMER\} \vdash_{\mathfrak{M} \setminus \{\mathfrak{I}\mathfrak{M}_1\}} \{STUDENT \rightarrow\rightarrow PREDMET\}.$$

SISTEM AKSIOMA $\mathfrak{M}$

$\mathfrak{F}_1$	(refleksivnost)	$Y \subseteq X \Rightarrow X \rightarrow Y$,
$\mathfrak{F}_2$	(proširenje)	$X \rightarrow Y$ i $V \subseteq W \Rightarrow XW \rightarrow VY$,
$\mathfrak{F}_3$	(pseudotranzitivnost)	$X \rightarrow Y$ i $WY \rightarrow Z \Rightarrow XW \rightarrow Z$,
$\mathfrak{M}_1$	(komplementarnost)	$X \rightarrow\rightarrow Y \Rightarrow X \rightarrow\rightarrow \mathcal{U} \setminus XY$,
$\mathfrak{M}_2$	(proširenje)	$X \rightarrow\rightarrow Y$ i $V \subseteq W \Rightarrow XW \rightarrow\rightarrow VY$,
$\mathfrak{M}_3$	(tranzitivnost)	$X \rightarrow\rightarrow Y$ i $Y \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow Z \setminus Y$,
$\mathfrak{FM}_1$	($\mathfrak{F}$ implicira $\mathfrak{M}$)	$X \rightarrow Y \Rightarrow X \rightarrow\rightarrow Y$,
$\mathfrak{FM}_2$	($\mathfrak{MF}$ tranzitivnost)	$X \rightarrow\rightarrow Y$ i $Y \rightarrow Z \Rightarrow X \rightarrow Z \setminus Y$.

Slika 5.13.

Do zaključka da aksioma $\mathfrak{M}_1$ (komplementarnost) nije redundantna u sistemu aksioma $\mathfrak{M}$, dolazi se na osnovu činjenice da se primenom aksioma iz $\mathfrak{M} \setminus \{\mathfrak{M}_1\}$ na polazni skup višeznačnih zavisnosti, mogu dobiti samo funkcionalne zavisnosti oblika $X \rightarrow Y$ i, primenom aksiome $\mathfrak{M}_2$ (proširenje), višeznačna zavisnost $SMER \cup X \rightarrow\rightarrow STUDENT \cup Y$, za $Y \subseteq X \subseteq \{SMER, STUDENT, PREDMET\}$.

Do zaključka da aksioma $\mathfrak{FM}_1$ ($\mathfrak{F}$ implicira $\mathfrak{M}$) nije redundantna u sistemu aksioma $\mathfrak{M}$, dolazi se na osnovu činjenice da se primenom aksioma iz $\mathfrak{M} \setminus \{\mathfrak{FM}_1\}$ na polazni skup funkcionalnih zavisnosti, mogu dobiti samo funkcionalne zavisnosti, koje su posledica aksioma $\mathfrak{F}_1$ i $\mathfrak{F}_2$.

Teorema 5.7. Sistem aksioma $\mathfrak{M}$ je neprotivrečan.

Dokaz. Neprotivrečnost aksioma $\mathfrak{F}_1$, $\mathfrak{F}_2$ i $\mathfrak{F}_3$ je dokazana u lemi 5.2.

Aksioma $\mathfrak{M}_1$. Neka je r relacija nad $\mathcal{U}$, koja zadovoljava $X \rightarrow\rightarrow Y$, a $u, v \in r$ torke, za koje važi $u[X] = v[X]$. Tada postoji $t \in r$ takva, da je $t[XY] = u[XY]$ i $t[\mathcal{U} \setminus XY] = v[\mathcal{U} \setminus XY]$. Ako r zadovoljava i $X \rightarrow\rightarrow \mathcal{U} \setminus XY$, tada postoji torka $s \in r$ takva, da je:

$$s[X] = u[X],\ s[\mathcal{U} \setminus X(\mathcal{U} \setminus XY)] = u[\mathcal{U} \setminus X(\mathcal{U} \setminus XY)] \text{ i } s[\mathcal{U} \setminus XY] = v[\mathcal{U} \setminus X].$$

Pošto je $\mathcal{U} \setminus (X(\mathcal{U} \setminus XY)) = Y$, zaključuje se da je $s = t$.

Aksioma $\mathfrak{M}_2$. Neka je r relacija nad $\mathcal{U}$, koja zadovoljava višeznačnu zavisnost $X \rightarrow\rightarrow Y$, $V \subseteq W$, a $u, v \in r$ torke za koje važi $u[XW] = v[XW]$. Treba pokazati da postoji torka $t \in r$ takva, da važi $u[XWY] = t[XWY]$ i $v[\mathcal{U} \setminus XYW] = t[\mathcal{U} \setminus XYW]$. Pošto r zadovoljava $X \rightarrow\rightarrow Y$, tada važi $u[XY] = t[XY]$ i $v[\mathcal{U} \setminus XY] = t[\mathcal{U} \setminus XY]$. Ako je $W \subseteq XY$, tada je $u[W] = v[W] = t[W]$. Ako je $W \subseteq \mathcal{U} \setminus XY$, tada je $v[W] = t[W] = u[W]$. Ako je $W = PQ$, gde je $P = W \cap XY$, a $Q = W \cap (\mathcal{U} \setminus XY)$, tada je $t[P] = u[P] = v[P]$, a $t[Q] = v[Q] = u[Q]$, odnosno $t[W] = u[W] = v[W]$.

Aksioma $\mathfrak{M}_3$. Neka je r relacija nad $\mathcal{U}$, koja zadovoljava višeznačne zavisnosti $X \rightarrow\rightarrow Y$ i $Y \rightarrow\rightarrow Z$, $u, v \in r$ torke za koje važi $u[X] = v[X]$. Treba pokazati da postoji torka

$w \in r$ takva, da je $w[X(Z \setminus Y)] = v[X(Z \setminus Y)]$ i $w[X(\mathcal{U} \setminus (Z \setminus Y))] = u[X(\mathcal{U} \setminus (Z \setminus Y))]$. Pošto r zadovoljava višeznačnu zavisnost $X \rightarrow\rightarrow Y$, postoji torka t u r takva, da je: $t[XY] = u[XY]$ i $t[X(\mathcal{U} \setminus Y)] = v[X(\mathcal{U} \setminus Y)]$. Pošto r zadovoljava $Y \rightarrow\rightarrow Z$, saglasno aksiomi $\mathfrak{M}_2$, zadovoljava i $XY \rightarrow\rightarrow XZ$. Na osnovu $XY \rightarrow\rightarrow XZ$ i $t[XY] = u[XY]$, sledi da, u r, postoji w takva da važi: $t[XYZ] = w[XYZ]$ i $u[XY(\mathcal{U} \setminus Z)] = w[XY(\mathcal{U} \setminus Z)]$. Na osnovu $t[X(\mathcal{U} \setminus Y)] = v[X(\mathcal{U} \setminus Y)]$ i $t[XYZ] = w[XYZ]$, sledi da v i w imaju iste $(X(\mathcal{U} \setminus Y)) \cap (XYZ) = X(Z \setminus Y)$ vrednosti. Pošto je $X(\mathcal{U} \setminus (Z \setminus Y)) = XY(\mathcal{U} \setminus Z)$, zaključuje se da je aksioma $\mathfrak{M}_3$ neprotivrečna.

Aksioma $\mathbb{FM}_1$. Neprotivrečnost aksiome $\mathbb{FM}_1$ sledi na osnovu definicije funkcionalne zavisnosti. Naime, neka je r relacija nad $\mathcal{U}$, koja zadovoljava funkcionalnu zavisnost $X \rightarrow Y$, a $u, v \in r$ torke za koje važi $u[XY] = v[XY]$. Ako r zadovoljava višeznačnu zavisnost $X \rightarrow\rightarrow Y$, tada postoji $t \in r$ takva, da važi $u[XY] = t[XY]$ i $v[\mathcal{U} \setminus XY] = t[\mathcal{U} \setminus XY]$, a to je baš torka v.

Aksioma $\mathbb{FM}_2$. Neka je r relacija nad $\mathcal{U}$, koja zadovoljava $X \rightarrow\rightarrow Y$ i $Y \rightarrow Z$, a $u, v \in r$ torke, za koje važi $u[X] = v[X]$. Pošto r zadovoljava $X \rightarrow\rightarrow Y$, postoji $t \in r$ takva, da je $u[XY] = t[XY]$ i $v[\mathcal{U} \setminus XY] = t[\mathcal{U} \setminus XY]$. S obzirom na $Y \rightarrow Z$ i $u[Y] = t[Y]$, sledi $u[Z] = t[Z]$. Na osnovu $u[Z] = t[Z]$ i $v[\mathcal{U} \setminus XY] = t[\mathcal{U} \setminus XY]$, sledi da torke u i v imaju iste $Z \setminus XY$ vrednosti, odnosno $u[Z \setminus XY] = v[Z \setminus XY]$. Konačno se zaključuje da važi $X \rightarrow Z \setminus Y$. □

Primer 5.38. Neka je $\mathcal{U} = \{DAN, SAT, UCI, PRD, NAS, STUD\}$ i neka važe sledeći odnosi između ovih obeležja:

1° jedan nastavnik predaje samo jedan predmet,
2° ako isti predmet predaje više nastavnika, svi studenti, koji slušaju predmet, slušaju ga kod svih nastavnika,
3° određenog dana i sata u jednoj učionici, nastavnik predaje određeni predmet.

DAN	*SAT*	*UCI*	*PRD*	*NAS*	*STUD*
d_1	h_1	u_1	p_1	n_1	s_1
d_1	h_1	u_1	p_1	n_1	s_2
d_1	h_1	u_1	p_1	n_1	s_3
d_1	h_2	u_2	p_2	n_2	s_4
d_1	h_2	u_2	p_2	n_2	s_5
d_2	h_3	u_2	p_1	n_3	s_1
d_2	h_3	u_2	p_1	n_3	s_2
d_2	h_3	u_2	p_1	n_3	s_3
d_3	h_2	u_3	p_1	n_1	s_1
d_3	h_2	u_3	p_1	n_1	s_2
d_3	h_2	u_3	p_1	n_1	s_3

Slika 5.14.

Na osnovu odnosa 1° i 3°, zaključuje se da nad skupom obeležja $\mathcal{U}$ važe sledeće funkcionalne zavisnosti $\mathcal{F} = \{NAS \rightarrow PRD,\ DAN+SAT+UCI \rightarrow PRD+NAS\}$. Na osnovu odnosa 2° se zaključuje da nad skupom $\mathcal{U}$ važi sledeći skup višeznačnih zavisnosti $\mathcal{M} = \{PRD \rightarrow\!\!\rightarrow DAN+SAT+UCI+NAS\}$. Na slici 5.14 je prikazana jedna relacija nad $\mathcal{U}$, koja zadovoljava $\mathcal{F}$ i $\mathcal{M}$.

Primenom aksiome $\mathfrak{M}_1$ na višeznačnu zavisnost

$$PRD \rightarrow\!\!\rightarrow DAN+SAT+UCI+NAS,$$

dobija se višeznačna zavisnost $PRD \rightarrow\!\!\rightarrow STUD$. Do zaključka o važenju ove višeznačne zavisnosti se moglo doći i na sledeći, mnogo duži, način:

- na osnovu funkcionalne zavisnosti $DAN+SAT+UCI \rightarrow PRD+NAS$ i aksiome $\mathfrak{M}_1$, sledi $DAN+SAT+UCI \rightarrow\!\!\rightarrow PRD+NAS$, a odatle, na osnovu $\mathfrak{M}_3$, sledi $DAN+SAT+UCI \rightarrow\!\!\rightarrow STUD$,
- na osnovu višeznačne zavisnosti $DAN+SAT+UCI \rightarrow\!\!\rightarrow STUD$ i aksiome $\mathfrak{M}_2$, sledi $DAN+SAT+UCI+NAS \rightarrow\!\!\rightarrow NAS+STUD$,
- na osnovu višeznačne zavisnosti $DAN+SAT+UCI+NAS \rightarrow\!\!\rightarrow NAS+STUD$, višeznačne zavisnosti $PRD \rightarrow\!\!\rightarrow DAN+SAT+UCI+NAS$ i aksiome $\mathfrak{M}_3$, sledi $PRD \rightarrow\!\!\rightarrow STUD$.

□

Nakon dokaza neprotivrečnosti sistema aksioma $\mathfrak{M}$, može se dokazati i sledeća lema.

Lema 5.7. Pravila izvođenja:

$\mathfrak{M}_4$ (presek) $\quad X \rightarrow\!\!\rightarrow Y$ i $X \rightarrow\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\rightarrow Y \cap Z$,
$\mathfrak{M}_5$ (razlika) $\quad X \rightarrow\!\!\rightarrow Y$ i $X \rightarrow\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\rightarrow Y \setminus Z$,
$\mathfrak{M}_6$ (unija) $\quad X \rightarrow\!\!\rightarrow Y$ i $X \rightarrow\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\rightarrow YZ$,

predstavljaju posledicu sistema aksioma $\mathfrak{M}$.

Dokaz. Prvo će biti pokazano da je $\mathfrak{M}_4$ posledica $\mathfrak{M}$. Na osnovu $X \rightarrow\!\!\rightarrow Y$ i aksiome $\mathfrak{M}_1$, sledi $X \rightarrow\!\!\rightarrow (\mathcal{U} \setminus XY)$, te se primenom aksiome $\mathfrak{M}_2$, dobija $X \rightarrow\!\!\rightarrow X(\mathcal{U} \setminus Y)$. Na osnovu $X \rightarrow\!\!\rightarrow Z$ i aksiome $\mathfrak{M}_2$, zaključuje se da važi $X(\mathcal{U} \setminus Y) \rightarrow\!\!\rightarrow Z$, za $\varnothing \subseteq \mathcal{U} \setminus Y$. Primenom aksiome $\mathfrak{M}_3$ na $X \rightarrow\!\!\rightarrow X(\mathcal{U} \setminus Y)$ i $X(\mathcal{U} \setminus Y) \rightarrow\!\!\rightarrow Z$, dobija se $X \rightarrow\!\!\rightarrow Z \setminus X(\mathcal{U} \setminus Y)$. Pošto je $Z \setminus X(\mathcal{U} \setminus Y) = (Z \cap Y) \setminus X$, primenom aksiome $\mathfrak{M}_2$ (za $X \cap Y \cap Z \subseteq X$), na $X \rightarrow\!\!\rightarrow (Z \cap Y) \setminus X$, dobija se $X \rightarrow\!\!\rightarrow Z \cap Y$.

Da bi se pokazalo da je $\mathfrak{M}_5$ posledica $\mathfrak{M}$, posmatraju se višeznačne zavisnosti $X \rightarrow\!\!\rightarrow Y$ i $X \rightarrow\!\!\rightarrow \mathcal{U} \setminus XZ$. Poslednja višeznačna zavisnost je posledica primene aksiome $\mathfrak{M}_1$ na $X \rightarrow\!\!\rightarrow Z$. Na osnovu pravila $\mathfrak{M}_4$ sledi $X \rightarrow\!\!\rightarrow Y \cap (\mathcal{U} \setminus XZ)$. Pošto je $Y \cap (\mathcal{U} \setminus XZ) = Y \setminus XZ$, zaključuje se da važi $X \rightarrow\!\!\rightarrow Y \setminus XZ$. Primenom aksiome $\mathfrak{M}_2$ (za $X \cap Y \subseteq X$), dobija se $X \rightarrow\!\!\rightarrow Y \setminus Z$.

Da bi se pokazalo da je $\mathfrak{M}_6$ posledica $\mathfrak{M}$, posmatraju se višeznačne zavisnosti $X \rightarrow\!\!\rightarrow \mathcal{U} \setminus Y$ i $X \rightarrow\!\!\rightarrow \mathcal{U} \setminus Z$, koje se dobijaju primenom aksiome $\mathfrak{M}_1$ i aksiome $\mathfrak{M}_2$, za $X \setminus (X \cap Y) \subseteq X$ i $X \setminus (X \cap Z) \subseteq X$, redom, na višeznačne zavisnosti $X \rightarrow\!\!\rightarrow Y$ i $X \rightarrow\!\!\rightarrow Z$. Primena pravila izvođenja $\mathfrak{M}_4$ na $X \rightarrow\!\!\rightarrow \mathcal{U} \setminus Y$ i $X \rightarrow\!\!\rightarrow \mathcal{U} \setminus Z$ daje $X \rightarrow\!\!\rightarrow (\mathcal{U} \setminus Y) \cap (\mathcal{U} \setminus Z)$. Primena

aksiome $\mathfrak{M}_1$ na $X\rightarrow\rightarrow(\mathcal{U}\setminus Y)\cap(\mathcal{U}\setminus Z)$ daje $X\rightarrow\rightarrow\mathcal{U}\setminus((\mathcal{U}\setminus Y)\cap(\mathcal{U}\setminus Z))$. Pošto je $\mathcal{U}\setminus((\mathcal{U}\setminus Y)\cap(\mathcal{U}\setminus Z))=YZ$, zaključuje se da važi $X\rightarrow\rightarrow YZ$. □

5.4.2. *Zatvaranje skupa funkcionalnih i višeznačnih zavisnosti*

Neka je $\mathcal{D}$ skup funkcionalnih i višeznačnih zavisnosti, definisanih nad skupom obeležja $\mathcal{U}$, $\mathcal{D}^*$ skup svih logičkih posledica skupa $\mathcal{D}$, a $\mathcal{D}^+_{\mathfrak{M}}$ zatvaranje skupa $\mathcal{D}$, dobijeno iscrpnom primenom sistema aksioma $\mathfrak{M}$ na $\mathcal{D}$. Da bi se dokazala kompletnost sistema aksioma $\mathfrak{M}$, potrebno je pokazati da je $\mathcal{D}^*=\mathcal{D}^+_{\mathfrak{M}}$.

Postupak izračunavanja zatvaranja $\mathcal{D}^+_{\mathfrak{M}}$, u skraćenoj notaciji $\mathcal{D}^+$, ima eksponencijalnu ocenu kompleksnosti, jer je $|\mathcal{D}^+|\geq 2^{|\mathcal{U}|}$. S druge strane, dokazivanje jednakosti $\mathcal{D}^*=\mathcal{D}^+$ se svodi na dokazivanje ekvivalentnosti iskaza $\mathcal{D}\models d$ i iskaza $d\in\mathcal{D}^+$, redom, za svako $d\in\mathcal{D}^+$ i za svako $d\in\mathcal{D}^*$. Za rešavanje implikacionog problema za skup funkcionalnih i višeznačnih zavisnosti, uvodi se pojam baze zavisnosti skupa obeležja $X\subseteq\mathcal{U}$, s obzirom na $\mathcal{D}$.

Definicija 5.23. Neka je $\mathcal{D}$ skup funkcionalnih i višeznačnih zavisnosti, definisan nad skupom obeležja $\mathcal{U}$, a $X\subseteq\mathcal{U}$. ***Baza zavisnosti*** za X, s obzirom na $\mathcal{D}$, je skup $db(X)=\{Y_i\mid Y_i\subseteq\mathcal{U}\}$ takav, da važi:

1° $X\rightarrow\rightarrow Z\in\mathcal{D}^+\Rightarrow Z=\bigcup_{j=1}^{k}Y_j$, $Y_j\in db(X)$,

2° $X\rightarrow Z\in\mathcal{D}^+\Rightarrow Z\subseteq X^+$,

3° $A\in X^+\Rightarrow\{A\}\in db(X)$. □

Sledeća teorema ukazuje na postupak za izračunavanje skupa $db(X)$.

Teorema 5.8. Skup obeležja $\mathcal{U}$ se može particionirati u skupove $Y_1,\ldots,Y_k$ tako, da $X\rightarrow\rightarrow Z$, za $Z\subseteq\mathcal{U}$, važi ako i samo ako Z predstavlja uniju nekih od Y_i.

Dokaz. Postupak particioniranja počinje od skupa $\{X,\mathcal{U}\setminus X\}$, jer $X\rightarrow\rightarrow\mathcal{U}\setminus X$ trivijalno važi. Neka je, nakon određenog broja koraka, dobijen skup $\{W_1,\ldots,W_n\}$, pri čemu važi $(\forall i\in\{1,\ldots,n\})(X\rightarrow\rightarrow W_i)$. Ako važi $X\rightarrow\rightarrow Z$, a Z nije jednako uniji nekih od W_i, svako W_i takvo, da je $Z\cap W_i\neq\emptyset$ i $W_i\setminus Z\neq\emptyset$, zamenjuje se sa $Z\cap W_1$ i $W_i\setminus Z$. Pošto je $\mathcal{U}$ konačan skup, mora se doći do situacije da, za svako $X\rightarrow\rightarrow Z$, Z predstavlja uniju skupova dobijenih particioniranjem, čime je izračunavanje $db(X)=\{Y_1,\ldots,Y_n\}$ završeno. S druge strane, s obzirom na pravilo izvođenja $\mathfrak{M}_6$, X višeznačno određuje uniju elemenata svakog podskupa skupa $db(X)$. □

Da bi se proverilo da li neka višeznačna zavisnost $X\rightarrow\rightarrow Z$ predstavlja posledicu datog skupa funkcionalnih i višeznačnih zavisnosti $\mathcal{D}$, definisanog nad skupom obeležja

$\mathcal{U}$, dovoljno je izračunati bazu zavisnosti za X i proveriti da li Z predstavlja uniju nekog podskupa skupa $db(X)$.

Primer 5.39. Neka je $\mathcal{U} = \{DAN, SAT, UCI, PRD, STUD, NAS\}$, a $\mathcal{O} = \{NAS \rightarrow PRD,\ DAN + SAT + UCI \rightarrow PRD + NAS\}$. Da bi se proverilo važenje višeznačne zavisnosti $DAN + SAT + UCI \rightarrow\rightarrow STUD$, treba izračunati $db(\{DAN, SAT, UCI\})$. Inicijalno je

$$db(\{DAN, SAT, UCI\})_1 = \{\{DAN, SAT, UCI\}, \{PRD, NAS, STUD\}\}.$$

Na osnovu aksiome $\mathfrak{I}_1$ važe funkcionalne, pa i višeznačne zavisnosti

$$DAN + SAT + UCI \rightarrow DAN,\ DAN + SAT + UCI \rightarrow SAT \text{ i } DAN + SAT + UCI \rightarrow UCI,$$

te se dobija

$$db(\{DAN, SAT, UCI\})_2 = \{\{DAN\}, \{SAT\}, \{UCI\}, \{PRD, NAS, STUD\}\}.$$

Nakon primene funkcionalne zavisnosti $DAN + SAT + UCI \rightarrow PRD + NAS$, dobija se, vodeći računa o uslovu 3° iz definicije 5.23,

$$db(\{SAT, DAN, UCI\})_3 = \{\{DAN\}, \{SAT\}, \{UCI\}, \{PRD\}, \{NAS\}, \{STUD\}\}.$$

Pošto je dalje particioniranje, očigledno, nemoguće, zaključuje se da je

$$db(\{SAT, DAN, UCI\})_3 = db(\{SAT, DAN, UCI\}).$$

Takođe, pošto je $\{STUD\} \in db(\{DAN, SAT, UCI\})$, zaključuje se da višeznačna zavisnost $DAN + SAT + UCI \rightarrow\rightarrow STUD$ predstavlja logičku posledicu skupa zavisnosti $\mathcal{O}$, (jer su aksiome iz $\mathfrak{M}$ neprotivrečne). □

Primer 5.40. Neka je $\mathcal{U} = \{A, B, C, D, E, F, G\}$, a $\mathcal{M} = \{AB \rightarrow\rightarrow CDE,\ AB \rightarrow\rightarrow EFG\}$ skup višeznačnih zavisnosti. Treba izračunati $db(AB)$.

Pošto, na osnovu aksiome $\mathfrak{I}_1$, sledi $AB \rightarrow A$, $AB \rightarrow B$, zaključuje se da $\{A\}$ i $\{B\}$ pripadaju $db(AB)$. Na osnovu pravila izvođenja $\mathfrak{M}_4$ (presek), važi višeznačna zavisnost $AB \rightarrow\rightarrow E$, te se zaključuje da i $\{E\}$ pripada $db(AB)$. Na osnovu pravila izvođenja $\mathfrak{M}_5$ (razlika), važe višeznačne zavisnosti $AB \rightarrow\rightarrow CD$ i $AB \rightarrow\rightarrow FG$, te se zaključuje da i $\{C, D\}$ i $\{F, G\}$ pripadaju $db(AB)$. Pošto se primenom pravila izvođenja ne mogu dobiti druge zavisnosti, zaključuje se da je

$$db(AB) = \{\{A\}, \{B\}, \{C, D\}, \{F, G\}, \{E\}\}.$$

Na osnovu izračunatog $db(AB)$, može se, na primer, zaključiti da $AB \rightarrow\rightarrow C \notin \mathcal{M}^+$. □

Primeri 5.41 i 5.42 su imali namenu da ilustruju zasnovanost tvrđenja teoreme 5.8 na skupu aksioma $\mathfrak{M}$, a ne i da ukažu na efikasnost postupka izračunavanja baze zavisnosti $db(X)$. U literaturi [Ga], na primer, je dat korektan, skoro linearan, algoritam za izračunavanje baze zavisnosti. Njegova ideja je zasnovana na particionairanju skupa $\mathcal{U}$, korišćenjem samo zavisnosti iz definisanog skupa $\mathcal{O}$ i korišćenjem specijalnih struktura

podataka. Sledeći primer ilustruje tvrđenje da se baza zavisnosti može izračunati samo jednim prolazom kroz skup zavisnosti $\mathcal{D}$.

Primer 5.41. Neka je $\mathcal{U} = \{A, B, C, D, E, F, G\}$, $\mathcal{D} = \{AB \rightarrow\rightarrow CD, C \rightarrow\rightarrow F, C \rightarrow\rightarrow E\}$, a $X = AB$. Inicijalno je $db(AB)_1 = \{\{A\}, \{B\}, \{C, D, E, F\}\}$. Nakon primene $AB \rightarrow\rightarrow CD$, dobija se $db(AB)_2 = \{\{A\}, \{B\}, \{C, D\}, \{E, F, G\}\}$. Primena $C \rightarrow\rightarrow F$ daje $db(AB)_3 = \{\{A\}, \{B\}, \{C, D\}, \{E, G\}, \{F\}\}$. Konačno, primenom $C \rightarrow\rightarrow E$, dobija se $db(AB)_4 = \{\{A\}, \{B\}, \{C, D\}, \{E\}, \{F\}, \{G\}\}$. Pošto ponovni prolazi kroz skup $\mathcal{D}$ ne bi doveli do promene $db(AB)_4$, zaključuje se da je $db(AB) = db(AB)_4$. □

Pojam baze zavisnosti predstavlja polaznu osnovu za dokazivanje kompletnosti sistema aksioma $\mathfrak{M}$. Ti dokazi se mogu naći u literaturi, na primer [PBG], [B]. Na osnovu činjenice da je sistem aksioma $\mathfrak{M}$ neprotivrečan i kompletan, sledi zaključak da se implikacioni problem za funkcionalne i višeznačne zavisnosti, u oznaci $\mathcal{D} \models d$, gde je $lhs(d) = X$, svodi na izračunavanje $db(X)$ i proveru da li $rhs(d)$ predstavlja uniju nekog podskupa baze zavisnosti za X.

5.4.3. *Projekcija skupa višeznačnih zavisnosti*

Kao i u slučaju funkcionalnih zavisnosti, često se javlja potreba utvrđivanja koje višeznačne zavisnosti iz skupa $\mathcal{M}$, definisanog na $\mathcal{U}$, važe i na skupu obeležja $W \subseteq \mathcal{U}$. Drugim rečima, potrebno je izračunati projekciju skupa višeznačnih zavisnosti $\mathcal{M}$ na skup obeležja W, u oznaci $\mathcal{M}|_W$.

Teorema 5.9. Ako je $X \rightarrow\rightarrow Y$ višeznačna zavisnost, koja važi na $\mathcal{U}$, a $X \subseteq W \subseteq \mathcal{U}$, tada na skupu obeležja W važi $X \rightarrow\rightarrow Y \cap W$.

Dokaz. Neka je r relacija nad $\mathcal{U}$, koja zadovoljava $X \rightarrow\rightarrow Y$. Neka u r postoje torke u, v i t, za koje važi $u[X] = v[X]$, $u[XY] = t[XY]$ i $v[X(\mathcal{U} \setminus Y)] = t[X(\mathcal{U} \setminus Y)]$. Saglasno tome, u r važi i $u[X(Y \cap W)] = t[X(Y \cap W)]$ i $v[X((\mathcal{U} \setminus Y) \cap W)] = t[X((\mathcal{U} \setminus Y) \cap W)]$. Pošto je $(\mathcal{U} \setminus Y) \cap W = W \setminus Y$, sledi $v[X(W \setminus Y)] = t[X(W \setminus Y)]$, a pošto je $X \subseteq W$, a W igra ulogu univerzalnog skupa u slučaju projekcije, zaključuje se da višeznačna zavisnost $X \rightarrow\rightarrow Y \cap W$ važi u $\pi_W(r)$, što je i trebalo pokazati. □

Pošto višeznačna zavisnost $X \rightarrow\rightarrow Y$ može biti logička posledica skupa višeznačnih zavisnosti $\mathcal{M}$, zaključuje se da je za izračunavanje $\mathcal{M}|_W$ potrebno prvo izračunati zatvaranje $\mathcal{M}^+$, pa onda izvršiti projektovanje na skup obeležja W, odnosno da je

$$\mathcal{M}|_W = \{X \rightarrow\rightarrow Y \cap W \mid (X \rightarrow\rightarrow Y \in \mathcal{M}^+) \wedge (X \subseteq W)\}.$$

Primer 5.42. Neka je $\mathcal{U} = \{A, B, C, D, E\}$, $\mathcal{M} = \{A \rightarrow\rightarrow B, BC \rightarrow\rightarrow D\}$, a $W = \{A, C, D, E\}$. Treba izračunati $\mathcal{M}|_W$.

Iscrpnom primenom aksioma $\mathfrak{M}_1$, $\mathfrak{M}_2$ i $\mathfrak{M}_3$ na skup višeznačnih zavisnosti $\mathcal{M}$, dobija se $\mathcal{M}^+$. Lako se proverava da važi $\{AC \rightarrow\rightarrow BC, BC \rightarrow\rightarrow D, AC \rightarrow\rightarrow D\} \subset \mathcal{M}^+$, te

da je $\mathcal{M}|_W = \{AC\rightarrow\rightarrow D,\ AC\rightarrow\rightarrow E\}$, jer su $AC\rightarrow\rightarrow D$ i $AC\rightarrow\rightarrow E$ jedine netrivijalne višeznačne zavisnosti iz $\mathcal{M}^+$, koje važe na skupu obeležja $\{A, C, D, E\}$, a nisu dobijene primenom nekog od pravila izvođenja iz $\mathcal{M}$. □

5.4.4. *Ugrađena višeznačna zavisnost*

Teorema 5.4 tvrdi da ako $X\rightarrow Y$ važi u $\pi_W(r)$, gde je $XY \subseteq W \subseteq U$, a r relacija nad U, tada $X\rightarrow Y$ važi i u r. Kada je reč o višeznačnim zavisnostima, može se desiti da $X\rightarrow\rightarrow Y$ važi u $\pi_W(r)$, ali da ne važi u r. Višeznačne zavisnosti, koje važe u projekciji relacije $r(U)$ na skup obeležja $W \subset U$, ali ne i u relaciji $r(U)$, nazivaju se *ugrađenim* (u relaciju $r(U)$). Višeznačne zavisnosti, koje važe u $r(U)$, nazivaju se *potpunim*.

Primer 5.43. Posmatra se skup obeležja $\{SMER, PRED, STUD, GRUP\}$, za koji su uočena sledeća ograničenja:

(γ_1) svaki student, upisan na smer, sluša svaki predmet na smeru.
(γ_2) student sluša predmet samo u određenoj grupi.
(γ_3) neke predmete studenti slušaju svi u jednoj grupi, a druge u više grupa.

Na prvi pogled, ograničenje γ_1 vodi zaključku o važenju višeznačne zavisnosti $SMER\rightarrow\rightarrow STUD$. Međutim, ograničenje γ_3 ukazuje da ta pretpostavka nije tačna, što potvrđuje i relacija r na slici 5.15. Naime, da višeznačna zavisnost $SMER\rightarrow\rightarrow STUD$ važi, tada bi relacija r, pored torki $(a, s_1, p_1, 1)$ i $(a, s_2, p_2, 1)$, morala sadržati i torku $(a, s_2, p_1, 1)$, što bi bilo u koliziji sa torkom $(a, s_2, p_1, 2)$, koja je u relaciji r, i sa ograničenjem γ_2.

SMER	*STUD*	*PRED*	*GRUP*
a	s_1	p_1	1
a	s_2	p_1	2
a	s_1	p_2	1
a	s_2	p_2	1
b	s_3	p_1	1
b	s_4	p_1	3
b	s_3	p_3	1
b	s_4	p_3	2

Slika 5.15.

Funkcionalna zavisnost $STUD+PRED\rightarrow GRUP$ je jedina netrivijalna zavisnost, koja važi nad skupom obeležja $\{SMER, STUD, PRED, GRUP\}$. Međutim na skupu obeležja $\{SMER, STUD, PRED\}$ važi višeznačna zavisnost $SMER\rightarrow\rightarrow STUD$, što se lako proverava posmatranjem projekcije relacije r sa slike 5.15 na skup obeležja $\{SMER,$

STUD, PRED}. Višeznačna zavisnost $SMER \rightarrow\rightarrow STUD$ je ugrađena u skup obeležja {*SMER, STUD, PRED*}, ali ne pripada skupu zavisnosti $\mathcal{D}$, definisanom na skupu obeležja {*SMER, STUD, PRED, GRUP*} i ne pripada skupu zavisnosti $\mathcal{D}|_{\{SMER, STUD, PRED\}}$. □

Sistem aksioma $\mathcal{AR}$ se ne može koristiti za izvođenje zaključaka o logičkim posledicama ugrađenih višeznačnih zavisnosti. Na primer, ako na skupu $W \subset \mathcal{U}$, *za* $XY \subseteq W$, važi višeznačna zavisnost $X \rightarrow\rightarrow Y$, ne može se tvrditi da, na skupu $\mathcal{U}$, važi bilo višeznačna zavisnost $X \rightarrow\rightarrow Y$, bilo $X \rightarrow\rightarrow \mathcal{U} \setminus XY$.

5.5. *Zavisnost spoja*

Zavisnost spoja je ograničenje, koje, pri postojanju niza torki sa jednim osobinama, uslovljava postojanje još jedne torke sa nekim drugim osobinama u relaciji $r(\mathcal{U})$. Osobine niza torki, kao i dodatne torke, određene su vrednostima podskupova $X_1,...,X_k$ skupa obeležja $\mathcal{U}$. Kao što će biti pokazano kasnije, podskupovi $X_1,...,X_k$ predstavljaju logički povezane, ali relativno nezavisne celine.

Definicija 5.24. Neka je r relacija nad skupom obeležja $\mathcal{U}$, a $X_1,...,X_k \subseteq \mathcal{U}$ i $\mathcal{U} = \bigcup_{i=1}^{k} X_i$. Relacija r zadovoljava ***zavisnost spoja***, u oznaci $\bowtie(X_1,..., X_k)$, ako važi

(5.7) $(\forall t_1,..., t_k \in r)((\forall i, j \in \{1,...,k\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j]) \Rightarrow$

$(\exists t \in r)(\forall i \in \{1,...,k\})(t[X_i] = t_i[X_i]))$. □

U interpretaciji značenja definicije 5.24 treba voditi računa da izraz (5.7) mora biti zadovoljen za svaki redosled torki $t_1,..., t_k$ i za svaka dva podskupa X_i, X_j, kao i da torke $t_1,..., t_k$ ne moraju biti međusobno različite. Torka t se, često, naziva ***ciljnom torkom***, a skupovi obeležja $X_1,...,X_k$ ***objektima*** ili ***komponentama*** zavisnosti spoja.

Prazna relacija i relacija nad skupom $\mathcal{U}$, koja sadrži samo jednu torku, zadovoljavaju svaku zavisnost spoja, za čije komponente važi $\mathcal{U} = \bigcup_{i=1}^{k} X_i$.

Primer 5.44. Na slici 5.16 je prikazana relacija $r(X_1X_2X_3)$, takva da zadovoljava zavisnost spoja $\bowtie(X_1, X_2, X_3)$, pri čemu važi da je $X_1 \cap X_3 = \emptyset$.

Saglasno definiciji 5.24, proizilazi da ako relacija r poseduje torke t_1, t_2 i t_3, tada, da bi zadovoljavala zavisnost spoja $\bowtie(X_1, X_2, X_3)$, mora sadržati i ostalih pet torki. Tako, na primer, važi da:

- za niz torki (t_1, t_2, t_3), torka t_4 zadovoljava uslov (5.7), što je ilustrovano na slici 5.17,
- za niz torki (t_1, t_2, t_2), torka t_5 zadovoljava uslov (5.7), što je ilustrovano na slici 5.18,
- za niz torki (t_1, t_7, t_3), torka t_4 zadovoljava uslov (5.7), što je ilustrovano na slici 5.19,

- za niz torki (t_1, t_3, t_4), uslov (5.7) trivijalno važi, pošto je usled činjenice $t_1[X_1 \cap X_2] \neq t_3[X_1 \cap X_2]$ narušena pretpostavka implikacije (5.7). □

r	X_1			X_3	
		X_2			
t_1	0	1	0	0	0
t_2	2	1	2	3	2
t_3	4	4	4	3	4
t_4	0	1	2	3	4
t_5	0	1	2	3	2
t_6	2	1	0	0	0
t_7	2	1	2	3	4
t_8	4	4	4	3	2

Slika 5.16.

	X_1			X_3	
		X_2			
t_1	0	1	0	0	0
t_2	2	1	2	3	2
t_3	4	4	4	3	4
t_4	0	1	2	3	4

Slika 5.17.

	X_1			X_3	
		X_2			
t_1	0	1	0	0	0
t_2	2	1	2	3	2
t_5	0	1	2	3	2

Slika 5.18.

r	X_1			X_3	
		X_2			
t_1	0	1	0	0	0
t_3	4	4	4	3	4
t_4	0	1	2	3	4
t_7	2	1	2	3	4

Slika 5.19.

Primer 5.45. Na slici 5.20 je prikazana relacija r nad skupom obeležja $\{A, B, C\}$, koja ne zadovoljava zavisnost spoja $\triangleright\triangleleft(AB, BC)$, jer ne poseduje torku (a_2, b_1, c_2), a poseduje, redom, torke (a_2, b_1, c_1) i (a_1, b_1, c_2). S druge strane, relacija r zadovoljava zavisnost spoja $\triangleright\triangleleft(AC, AB)$, jer, bez obzira na redosled posmatranja torki (a_1, b_1, c_1) i (a_1, b_1, c_2), jedna od njih predstavlja ciljnu torku. □

$r =$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1

Slika 5.20.

Primer 5.46. Relacija nad skupom obeležja $\{K, N, P, U\}$, prikazana na slici 5.21 zadovoljava, a relacija na slici 5.22 ne zadovoiljava zavisnost spoja $\triangleright\triangleleft(KN, KP, NPU)$. Relacija sa slike 5.22 ne zadovoljava zavisnost spoja $\triangleright\triangleleft(KN, KP, NPU)$, jer ne sadrži torku (k_1, n_1, p_2, u_2). □

K	N	P	U
k_1	n_1	p_1	u_1
k_1	n_2	p_2	u_1
k_2	n_3	p_3	u_2

Slika 5.21.

K	N	P	U
k_1	n_1	p_1	u_1
k_1	n_2	p_2	u_1
k_2	n_1	p_2	u_2

Slika 5.22.

Višeznačna zavisnost $X \rightarrow\rightarrow Y \mid \mathcal{U} \setminus XY$ predstavlja specijalan slučaj zavisnosti spoja. Naime, za $k = 2$, $X_1 = XY$ i $X_2 = X(\mathcal{U} \setminus Y)$, izraz (5.7) postaje

$$(\forall t_1, t_2 \in r)(t_1[X] = t_2[X] \Rightarrow (\exists t \in r)(t[XY] = t_1[XY] \wedge t[X(\mathcal{U} \setminus Y)] = t_2[X(\mathcal{U} \setminus Y)]),$$

što predstavlja uslov, koji relacija r nad $\mathcal{U}$ mora ispuniti da bi zadovoljavala višeznačnu zavisnost $X \rightarrow\rightarrow Y \mid \mathcal{U} \setminus XY$.

Primer 5.47. Relacija r nad $\{A, B, C\}$ iz primera 5.47 zadovoljava funkcionalnu zavisnost $A \rightarrow B$, te zadovoljava i višeznačnu zavisnost $A \rightarrow\rightarrow B \mid C$ i zavisnost spoja $\bowtie(AB, AC)$. □

Operator prirodnog spoja i zavisnost spoja predstavljaju usko povezane pojmove, o čemu govori teorema 5.10. Pre prelaska na definisanje i dokazivanje teoreme 5.10, biće dokazana sledeća lema.

Lema 5.8. Ako je r relacija nad $\mathcal{U}$, a $X_1, \ldots, X_k \subseteq \mathcal{U}$ i $\mathcal{U} = \bigcup_{i=1}^{k} X_i$, tada važi

$$r \subseteq \bowtie_{i=1}^{k} \pi_{X_i}(r).$$

Dokaz. Neka je $t \in r$, a $r_i = \pi_{X_i}(r)$. Tada važi $(\forall i \in \{1, \ldots, k\})(t[X_i] \in r_i)$. Prema definiciji prirodnog spoja je $\bowtie_{i=1}^{k} r_i = \{t[\mathcal{U}] \mid (\forall i \in \{1, \ldots, k\})(t[X_i] \in r_i)\}$. Znači, svako $t \in r$ je u $\bowtie_{i=1}^{k} r_i$, te se zaključuje da važi $r \subseteq \bowtie_{i=1}^{k} \pi_{X_i}(r)$. □

Primer 5.48. Na slici 5.23 je prikazana relacija r nad $\{A, B, C\}$ iz primera 5.47, njene projekcije po komponentama zavisnosti spoja $\bowtie(AB, BC)$ i rezultat prirodnog spajanja tih projekcija. Lako se uočava da važi $r \subset \pi_{AB}(r) \bowtie \pi_{BC}(r)$.

$r =$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1

$r[AB] =$

A	B
a_1	b_1
a_2	b_1

$r[BC] =$

B	C
b_1	c_1
b_1	c_2

$r[AB] \bowtie r[BC] =$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_2

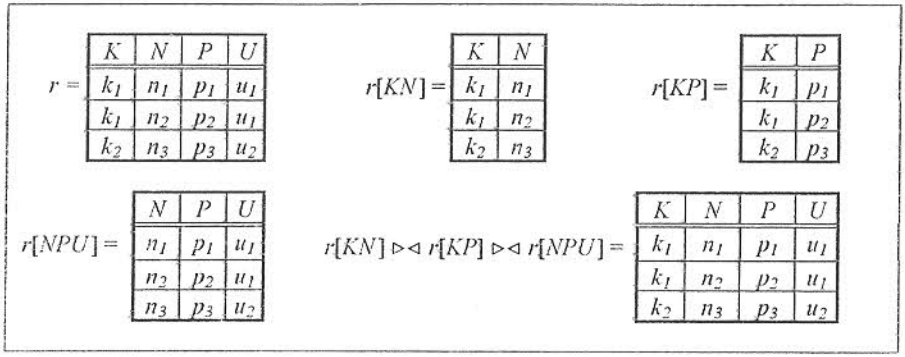
Slika 5.23.

Na slici 5.24 je prikazana relacija r nad skupom obeležja $\{K, N, P, U\}$ iz primera 5.48, njene projekcije po komponentama zavisnosti spoja $\bowtie(KN, KP, NPU)$ i rezultat prirodnog spajanja tih projekcija. Lako se uočava da važi

$$r = \pi_{KN}(r) \bowtie \pi_{KP}(r) \bowtie \pi_{NPU}(r). \square$$

Primer 5.50 sugeriše da je

$$r \subset \bowtie_{i=1}^{k} \pi_{X_i}(r),$$



$r =$

K	N	P	U
k_1	n_1	p_1	u_1
k_1	n_2	p_2	u_1
k_2	n_3	p_3	u_2

$r[KN] =$

K	N
k_1	n_1
k_1	n_2
k_2	n_3

$r[KP] =$

K	P
k_1	p_1
k_1	p_2
k_2	p_3

$r[NPU] =$

N	P	U
n_1	p_1	u_1
n_2	p_2	u_1
n_3	p_3	u_2

$r[KN] \triangleright\!\triangleleft r[KP] \triangleright\!\triangleleft r[NPU] =$

K	N	P	U
k_1	n_1	p_1	u_1
k_1	n_2	p_2	u_1
k_2	n_3	p_3	u_2

Slika 5.24.

ako r ne zadovoljava zavisnost spoja $\triangleright\!\triangleleft(X_1,...,X_k)$, a da je $r = \underset{i=1}{\overset{k}{\triangleright\!\triangleleft}} \pi_{X_i}(r)$, ako r zadovoljava zavisnost spoja $\triangleright\!\triangleleft(X_1,...,X_k)$.

Teorema 5.10. Dat je skup obeležja $\mathcal{U}$. Relacija r iz $SAT(\mathcal{U})$ zadovoljava zavisnost spoja $\triangleright\!\triangleleft(X_1,...,X_k)$, gde je $X_1,...,X_k \subseteq \mathcal{U}$ i $\mathcal{U} = \bigcup_{i=1}^{k} X_i$, ako i samo ako važi

$$r = \underset{i=1}{\overset{k}{\triangleright\!\triangleleft}} \pi_{X_i}(r). \tag{5.8}$$

Dokaz. ($\Rightarrow$) Neka je r relacija iz $SAT(\mathcal{U})$ koja zadovoljava uslov (5.8), ali ne i uslov (5.7), tj. neka važi

$$(\exists t_1,...,t_k \in r)(\forall i,j \in \{1,...,k\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j]) \wedge (\forall t \in r)(\exists i \in \{1,...,k\})(t[X_i] \neq t_i[X_i])). \tag{5.9}$$

Neka je za svako i iz $\{1,...,k\}$, $r_i = \pi_{X_i}(r)$, tada važi $(\forall i \in \{1,...,k\})(\exists u_i \in r_i)(t_i[X_i] = u_i)$. Takođe će važiti $(\forall i,j \in \{1,...,k\})((\exists u_i \in r_i) \wedge (\exists u_j \in r_j))(u_i[X_i \cap X_j] = u_j[X_i \cap X_j])$, te se zaključuje da $\underset{i=1}{\overset{k}{\triangleright\!\triangleleft}} r_i$ sadrži torku t, koja zadovoljava uslov $(\forall i \in \{1,...,k\})(t[X_i] = u_i)$. Međutim, s obzirom na (5.9) takvo t nije u r, te se, na osnovu leme 5.8, zaključuje da važi $r \subset \underset{i=1}{\overset{k}{\triangleright\!\triangleleft}} \pi_{X_i}(r)$, što je u kontradikciji sa (5.8).

($\Leftarrow$) Neka je r takva relacija iz $SAT(\mathcal{U})$, koja zadovoljava zavisnost spoja $\triangleright\!\triangleleft(X_1,...,X_k)$ i neka, suprotno tvrđenju (5.8), a saglasno lemi 5.8, važi

$$r \subset \underset{i=1}{\overset{k}{\triangleright\!\triangleleft}} \pi_{X_i}(r). \tag{5.10}$$

Neka je $t \in \overset{k}{\underset{i=1}{\triangleright\!\triangleleft}} \pi_{X_i}(r) \setminus r$. Tada, za svako $i \in \{1,\ldots,k\}$ važi $t[X_i] \in \pi_{X_i}(r) = r_i$. Pošto je svako r_i dobijeno projektovanjem r po X_i, zaključuje se da važi

$$(\forall i \in \{1,\ldots,k\})(\exists t_i \in r)(t_i[X_i] = t[X_i]).$$

Pošto je t dobijeno prirodnim spajanjem po jedne torke $t_i[X_i]$ iz svakog r_i, zaključuje se da važi i

$$(\forall i, j \in \{1,\ldots,k\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j].$$

Po pretpostavci, t nije u r, te važi

$$(\exists t_1,\ldots,t_k \in r)((\forall i, j \in \{1,\ldots,k\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j]) \wedge (\forall u \in r)(u \neq t)),$$

što je u kontadikciji sa pretpostavkom da r zadovoljava uslov (5.7). □

Primer 5.50 ilustruje i tvrđenje teoreme 5.10.

Posledica 5.1. Data je šema relacije $(\mathcal{U}, C)$ i zavisnost spoja $\triangleright\!\triangleleft(X_1,\ldots,X_k)$, gde je $X_1,\ldots,X_k \subseteq \mathcal{U}$ i $\mathcal{U} = \bigcup_{i=1}^{k} X_i$. Važi da je $C \models \triangleright\!\triangleleft(X_1,\ldots,X_k)$, ako i samo ako je za svaku relaciju $r \in SAT(\mathcal{U}, C)$ zadovoljeno $r = \overset{k}{\underset{i=1}{\triangleright\!\triangleleft}} \pi_{X_i}(r)$.

Dokaz. Činjenica $(\forall r \in SAT(\mathcal{U}, C))(r = \overset{k}{\underset{i=1}{\triangleright\!\triangleleft}} \pi_{X_i}(r))$ je, na osnovu teoreme 5.10, ekvivalentna tvrđenju da svaka relacija $r \in SAT(\mathcal{U}, C)$ zadovoljava zavisnost spoja $\triangleright\!\triangleleft(X_1,\ldots, X_k)$, što je, saglasno definiciji logičke posledice, ekvivalentno činjenici $C \models \triangleright\!\triangleleft(X_1,\ldots, X_k)$. □

Iz teoreme 5.10, takođe, sledi da je zavisnost spoja $\triangleright\!\triangleleft(X_1,\ldots,X_k)$, gde je $X_1,\ldots,X_k \subseteq \mathcal{U}$ i $\mathcal{U} = \bigcup_{i=1}^{k} X_i$, trivijalna ako važi

$$(\exists X_i \in \{X_1,\ldots,X_k\})(\mathcal{U} = X_i). \tag{5.11}$$

Definicija 5.25. Zavisnost spoja $\triangleright\!\triangleleft(X_1,\ldots, X_k)$ nad skupom obeležja $\mathcal{U}$ je ***ugrađena***, ako važi $\bigcup_{i=1}^{k} X_i \subset \mathcal{U}$. Ako važi $\bigcup_{i=1}^{k} X_i = \mathcal{U}$, zavisnost spoja je ***potpuna***. □

5.5.1. *Definisanje zavisnosti spoja putem predikata*

Dok je intuitivno značenje pojma funkcionalne zavisnosti očigledno i lako primenljivo, jer se njihovo definisanje vrši na nivou intenzije, uopštavanjem odnosa između elemenata dva domena, takvo tvrđenje bi se moglo teže braniti kada su u pitanju

višeznačne zavisnosti. Definisanje zavisnosti spoja, koja sadrži više od dve komponente, na osnovu posmatranja skupa obeležja i odnosa između elemenata njihovih domena je, praktično, veoma teško rešiv zadatak. Jedno rešenje ovog problema, za slučaj određene klase zavisnosti spoja, predložen je u [FMU]. Rešenje polazi od definisanja skupa legalnih relacija nad skupom obeležja $\mathcal{U}$ putem predikata, čije promenljive predstavljaju obeležja iz podskupova $X_1,..., X_k$ skupa $\mathcal{U}$. Zatim se dokazuje da svaka relacija tako definisanog skupa relacija nad $\mathcal{U}$, zadovoljava zavisnost spoja $\triangleright\triangleleft(X_1,..., X_k)$.

Neka je $X_i = \{ A_i^1,..., A_i^n \}$ podskup skupa obeležja $\mathcal{U}$, za koji važi:

1° $1 \le |X_i| \le |\mathcal{U}|$ i

2° elementi $dom(A_i^j)$, za $j = 1,..., n$ su međusobno logički povezani u realnom sistemu,

tada je $\mathcal{P}_i(X_i)$ predikat, koji opisuje relaciju r_i između elemenata domena obeležja iz X_i:

$$r_i = \{t[X_i] \mid \mathcal{P}_i(X_i \mid t)\},$$

pri čemu je sa $\mathcal{P}_i(X_i \mid t)$ označena interpretacija predikata $\mathcal{P}_i(X_i)$ s obzirom na torku t i važi da je $\mathcal{P}_i(X_i \mid t) \in \{\mathsf{T}, \perp\}$. Relacija r_i, dakle, sadrži samo one torke t, za koje je zadovoljen predikat $\mathcal{P}_i(X_i)$ (tj. $\mathcal{P}_i(X_i \mid t) = \mathsf{T}$).

Definicija 5.26. Skup predikata $\mathcal{P} = \{\mathcal{P}_1(X_1),...,\mathcal{P}_k(X_k)\}$, takvih da je $\mathcal{U} = \bigcup_{i=1}^{k} X_i$, definiše relaciju

$$r = \{t[\mathcal{U}] \mid \mathcal{P}_1(X_1 \mid t) \wedge ... \wedge \mathcal{P}_k(X_k \mid t)\}. \quad (5.12)$$ □

Relacija r nad skupom obeležja $\mathcal{U}$, definisana putem izraza (5.12), sadrži sve one i samo one torke, koje zadovoljavaju svaki od predikata iz skupa $\mathcal{P}$. To znači da je skupom $\mathcal{P}$ definisan i predikat $\mathcal{P}(\mathcal{U}) = \mathcal{P}_1(X_1) \wedge ... \wedge \mathcal{P}_k(X_k)$. Saglasno definiciji 5.26, svaki predikat služi kao svojevrstan "filter", koji proverava da li svaka komponenta $t_i = (a_1,..., a_n)$ torke t reflektuje jednu od tekućih vrednosti skupa obeležja X_i u realnom sistemu.

Pri promeni stanja realnog sistema, menja se skup istinitih vrednosti bar jednog predikata $\mathcal{P}_i(X_i)$. Ta promena dovodi do generisanja nove relacije saglasno formuli (5.12). Tokom vremena, formula (5.12) generiše, za isti skup predikata $\mathcal{P}$, veći broj relacija nad skupom obeležja $\mathcal{U}$. Međutim, u svakom trenutku, samo jedna od tih relacija predstavlja vernu sliku realnog sistema.

Definicija 5.26 ima za posledicu i činjenicu da se nijedan predikat $\mathcal{P}(X)$ ne može predstaviti putem konjunkcije predikata $\mathcal{R}(Y) \wedge Q(Z)$, za $X = YZ$.

Primer 5.49. Neka je $\mathcal{U} = \{K, N, P, U\}$, gde je K skup obeležja karakterističnih za katedru, N za nastavnika, P za predmet, a U za učionicu. Neformalni opis značenja tih obeležja i njihovih veza u realnom sistemu bi mogao biti:

- nastavnik n radi na katedri k,
- predmet p pripada katedri k,
- nastavnik n predaje predmet p u učionici u.

U gornjem opisu su upotrebljena tri predikata. To su: predikat "radi na" sa promenljivama K i N, predikat "pripada" sa promenljivama K i P i predikat "predaje u" sa promenljivama

N, P i U. Saglasno tome, relacije nad skupom obeležja $\{K, N, P, U\}$ se mogu opisati na sledeći način

$$\{(k, n, p, u) \mid (n \text{ "radi na" } k) \wedge (p \text{ "pripada" } k) \wedge (n \text{ "predaje" } p \text{ "u" } u)\}.$$

Treba zapaziti da se predikat "n predaje p u u" ne može zameniti sa dva predikata i to: predikatom "n predaje p" i predikatom "p se izvodi u u". Tromesni predikat $\mathcal{P}(N, P, U)$ ukazuje da n izvodi nastavu iz p u određenim učionicama u, dok neki drugi nastavnik, isti predmet može izvoditi u drugim učionicama. Dvomesni predikat $\mathcal{R}(P, U)$ ukazuje u kojim se sve učionicama u izvodi predmet p. Konjunkcija predikata $Q(N, P) \wedge \mathcal{R}(P, U)$ bi ukazivala da svaki nastavnik, koji predaje predmet p, predaje ga u svakoj od učionica u, gde se p uopšte izvodi. □

Primer 5.50. Posmatra se skup obeležja $\{D, A, S, P, M, K, C\}$, gde je D (dobavljač), A (adresa dobavljača), S (stanje računa dobavljača), P (porudžbina), M (materijal), K (količina) i C (cena). Za dati skup obeležja se mogu definisati sledeći predikati sa odgovarajućim skupovima obeležja (promenljivih):

- d ima a ($\mathcal{P}_1(\{A, D\})$),
- s je stanje računa d ($\mathcal{P}_2(\{D, S\})$),
- p je upućeno d ($\mathcal{P}_3(\{D, P\})$),
- p sadrži m u količini k ($\mathcal{P}_4(\{K, M, P\})$),
- d isporučuje m po ceni c ($\mathcal{P}_5(\{C, D, M\})$).

Relacije nad skupom obeležja $\{A, C, D, K, M, P, S\}$ su definisane sa

$$\{(a, c, d, k, m, p, s) \mid \mathcal{P}_1(\{A, D\}) \wedge \mathcal{P}_2(\{D, S\}) \wedge \mathcal{P}_3(\{D, P\}) \wedge \mathcal{P}_4(\{K, M, P\}) \wedge \mathcal{P}_5(\{C, D, M\})\}.$$

Kao i u prethodnom primeru, treba zapaziti da se predikat "d isporučuje m po ceni c" ne može podeliti na dva predikata, predikat "d isporučuje m" i predikat "c je cena za m". Poslednja dva predikata opisuju realni svet u kome materijal m ima cenu c, bez obzira na isporučioca d, dok predikat "d isporučuje m po ceni c" ukazuje da, u principu, svaki isporučilac materijala m može tražiti drugu cenu c. □

Postoji direktna veza između zavisnosti spoja, definisane putem skupa svojih komponenata $\{X_1, \ldots, X_k\}$ i skupa predikata, takvih da:

- svaki skup obeležja X_i predstavlja skup promenljivih tačno jednog predikata i
- da je putem tih predikata opisana relacija nad skupom obeležja $\bigcup_{i=1}^{k} X_i$.

Pre prelaska na dokazivanje povezanosti ovih pojmova, treba zapaziti da neka relacija r nad skupom obeležja $\mathcal{U}$ predstavlja mogućnu relaciju, definisanu izrazom (5.12), ako i samo ako za svako $i \in \{1, \ldots, k\}$ važi:

- p_i je takva relacija nad skupom obeležja X_i, da sadrži sve torke, koje zadovoljavaju predikat $\mathcal{P}_i(X_i)$ i
- $r = \bowtie_{i=1}^{k} p_i$.

Teorema 5.11. Relacija r nad skupom obeležja $\mathcal{U} = \bigcup_{i=1}^{k} X_i$, se može konstruisati na osnovu izraza (5.12) od istinitih vrednosti predikata $\mathcal{P}_i(X_i)$, ako i samo ako relacija r zadovoljava zavisnost spoja $\bowtie(X_1,...,X_k)$.

Dokaz. ($\Rightarrow$) Pretpostavlja se da relacija r zadovoljava zavisnost spoja $\bowtie(X_1,...,X_k)$, odnosno da važi $r = \bowtie_{i=1}^{k} \pi_{X_i}(r)$. Neka je, za svako $i \in \{1,...,k\}$, $\mathcal{P}_i(X_i)$ predikat za koji relacija $\pi_{X_i}(r)$ predstavlja skup istinitih vrednosti. Pošto relacija r zadovoljava zavisnost spoja $\bowtie(X_1,...,X_k)$, saglasno teoremi 5.10, svako t iz r zadovoljava svaki od odabranih predikata $\mathcal{P}_1(X_1),...,\mathcal{P}_k(X_k)$, te važi $r = \{t \mid \mathcal{P}_1(X_1|\, t) \wedge ... \wedge \mathcal{P}_k(X_k \mid t)\}$.

($\Leftarrow$) Pretpostavlja se da je relacija r konstruisana primenom izraza (5.12) od predikata $\mathcal{P}_1(X_1),..., \mathcal{P}_k(X_k)$. Tada $\pi_{X_i}(r)$ mora predstavljati skup istinitih vrednosti predikata $\mathcal{P}_i(X_i)$. Pošto je, po definiciji 5.26, r relacija, koja sadrži sve one torke, koje zadovoljavaju svaki od predikata $\mathcal{P}_1(X_1),...,\mathcal{P}_k(X_k)$, zaključuje se da je $r = \bowtie_{i=1}^{k} \pi_{X_i}(r)$, odnosno da relacija r zadovoljava zavisnost spoja $\bowtie(X_1,...,X_k)$. □

Primer 5.51. Saglasno teoremi 5.11, relacija nad skupom obeležja $\{K, N, P, U\}$, definisana u primeru 5.51, zadovoljava zavisnost spoja $\bowtie(KN, KP, NPU)$, a relacija nad skupom obeležja $\{A, C, D, K, M, P, S\}$, definisana u primeru 5.52, zadovaljava zavisnost spoja $\bowtie(AD, DP, DS, KMP, CDM)$. □

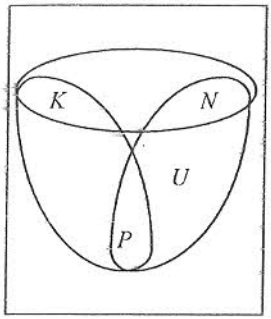
5.5.2. *Predstavljanje zavisnosti spoja putem hipergrafa*

Kada je $k \geq 3$, u notaciji $\bowtie(X_1,...,X_k)$ je teško uočiti strukturu zavisnosti spoja. Zato se za predstavljanje zavisnosti spoja, često, koriste hipergrafovi.

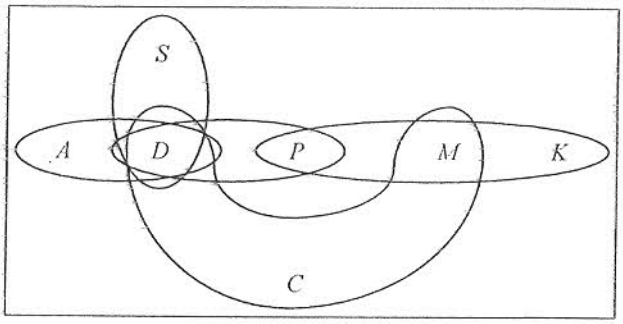
Definicija 5.27. Hipergraf je uređeni par (N, E), gde je N konačan skup čvorova, a E skup hipergrana (hiperivica). Svaka hipergrana iz E predstavlja neprazan podskup skupa N. □

Hipergrafovi predstavljaju generalizaciju neusmerenih grafova. Neusmereni graf je hipergraf kod kojeg sve ivice povezuju tačno dva čvora. U daljem tekstu se termin *hipergrana* skraćuje kao *grana*.

Zavisnost spoja se repezentuje putem hipergrafa tako, da svako obeležje iz $\mathcal{U}$ predstavlja jedan čvor, a komponente zavisnosti spoja predstavljaju grane. Grane hipergrafa se crtaju zatvorenom krivom linijom, koja obuhvata čvorove - elemente jedne komponente zavisnosti spoja.



Slika 5.25.



Slika 5.26.

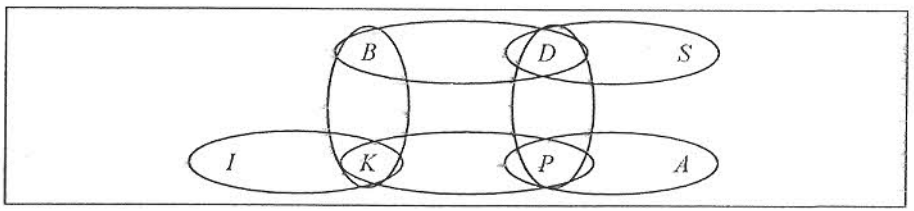
Primer 5.52. Na slici 5.25 je prikazan hipergraf zavisnosti spoja $\triangleright\triangleleft(KN, KP, NPU)$, a na slici 5.26 je prikazan hipergraf zavisnosti spoja $\triangleright\triangleleft(AD, DP, DS, KMP, CDM)$. □

5.5.3. *Ciklični i aciklični hipergrafovi zavisnosti spoja*

Postoji veći broj ekvivalentnih definicija acikličnosti hipergrafa zavisnosti spoja. Pojam cikličnosti i acikličnosti je detaljno obrađen u literaturi, na primer [FMU], [AP], [BFMY], [PBG]. Pre uvođenja postupka za proveru acikličnosti zavisnosti spoja, biće ukazano na intuitivno značenje cikličnosti zavisnosti spoja, putem narednog primera.

Primer 5.53. Jedan od klasičnih primera [FMU] za ilustraciju cikličnosti hipergrafa zavisnosti spoja, predstavlja opis sledećeg realnog sistema putem predikata:

- banka b prima depozite (uloge) d,
- banka b daje kredite k,
- depozit d ima stanje s,
- kredit k ima iznos i,
- poslovni partner p ima depozit (ulog) d,
- poslovni partner p je digao kredit k,
- poslovni partner p ima adresu a.



Slika 5.27.

GRAHAMOV ALGORITAM

PROCES Cikličnost_grafa

Ulaz: $\triangleright\triangleleft(X_1,\ldots,X_k)$ (* Zavisnost spoja, takva da je $\mathcal{U} = \bigcup_{i=1}^{k} X_i$ *)

Izlaz: *BROJAC* (* *BROJAC* ≥ 0, ako je sadržaj promenljive *BROJAC* = 0, zavisnost spoja je aciklična, inače je ciklična*)

POČETAK PROCESA Cikličnost_grafa

POSTAVI BROJAC $\leftarrow \sum_{i=1}^{k} |X_i|$

POSTAVI STARI $\leftarrow 0$

RADI redukcija *DOK JE STARI* $\neq$ *BROJAC*

POSTAVI STARI $\leftarrow$ *BROJAC*

RADI sadržavanje $(\forall i \in \{1,\ldots,k\})$

AKO JE $X_i \neq \emptyset$ *TADA*

AKO JE $(\exists j \in \{1,\ldots,k\})((i \neq j) \wedge (X_i \subseteq X_j))$ *TADA*

POSTAVI BROJAC $\leftarrow$ *BROJAC* $- |X_i|$

POSTAVI $X_i \leftarrow \emptyset$

INAČE

KRAJ AKO

INAČE

KRAJ AKO

KRAJ RADI sadržavanje

RADI izolovana_obeležja $(\forall A \in \mathcal{U})$

POSTAVI BROB $\leftarrow 0$

RADI brojanje $\forall i \in \{1,\ldots,k\}$ *DOK JE BROB* < 2

AKO JE $A \in X_i$ *TADA*

POSTAVI BROB $\leftarrow$ *BROB* $+ 1$

POSTAVI $n \leftarrow i$

INAČE

KRAJ AKO

KRAJ RADI brojanje

AKO JE BROB $= 1$ *TADA*

POSTAVI BROJAC $\leftarrow$ *BROJAC* $- 1$

POSTAVI $X_n \leftarrow X_n \setminus \{A\}$

INAČE

KRAJ AKO

KRAJ RADI izolovana_obeležja

KRAJ RADI redukcija

KRAJ PROCESA Cikličnost_grafa

Slika 5.28.

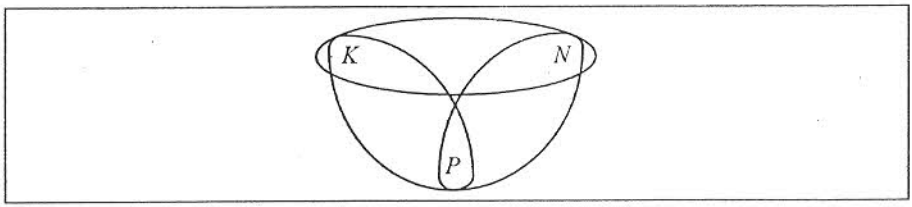
Na slici 5.27 nacrtan je hipergraf zavisnosti spoja $\triangleright\triangleleft(BD, BK, DS, KI, PD, PK, PA)$. Cikličnost hipergafa na slici 5.27 je posledica činjenice da od poslovnog partnera do banke postoje dva puta, preko kredita i preko depozita. Ova činjenica predstavlja i uzrok jednog od problema, koji se javlja u vezi sa cikličnim hipergrafovima. Ako se putem upita u bazu podataka traže banke povezane sa datim poslovnim partnerom, upit nije jednoznačan. Naime, poslovni partner može imati depozitne i kreditne odnose sa više banaka, te se javlja dilema da li se upit odnosi na banke, gde poslovni partner ima depozite, na banke, gde je polovni partner digao kredit, ili i na jedne i na druge.

Cikličnost hipergrafa sa slike 5.27 je posledica činjenice da, u posmatranom realnom sistemu, poslovni partner igra dve uloge, ulogu ulagača i ulogu korisnika kredita. □

Kao alat za testiranje acikličnosti hipergrafa, na ovom mestu će biti primenjen takozvani Grahamov algoritam [G], prikazan na slici 5.28. Osnovni zadatak Grahamovog algoritma je da izvrši redukciju hipergrafa zavisnosti spoja. Pojam redukcije se odnosi na uzajamne primene postupaka "sadržavanje" i "izolovani čvorovi". U tom cilju, algoritam prvo proverava da li postoje takve komponente date zavisnosti spoja, koje su sadržane u drugim komponentama. Te komponente se isključuju iz hipergrafa u postupku pod nazivom "sadržavanje". Nakon toga se, u postupku "izolovana_obeležja" traže i isključuju iz hipergrafa sva obeležja, koja se javljaju u samo jednoj komponenti zavisnosti spoja. Algoritam se realizuje naizmeničnim izvršavanjem postupaka "sadržavanje" i "izolovana_ obeležja", dok se ne dođe do tačke kada više nijedna od akcija nije moguća. U radu [YO] je dokazana korektnost Grahamovog algoritma, kao i da je zavisnost spoja aciklična, ako je redukcija njenog hipergrafa prazan hipergraf i da je redukcija hipergrafa ciklične zavisnosti spoja, neprazan hipergraf.

Primer 5.54. Neka je data zavisnost spoja $\triangleright\triangleleft(ABC, ABF, ACE, BCD)$. Ova zavisnost spoja, očigledno, ne poseduje komponente koje su sadržane u drugim komponentama, ali poseduje izolovana obeležja F, E i D. Nakon njihovog uklanjanja, dobijaju se komponente - grane hipergrafa: ABC, AB, AC i BC. Pošto su AB, AC i BC sadržane u ABC, postupak "sadržavanje" algoritma sa slike 5.28 ostavlja u hipergrafu samo granu ABC. Pošto sada postoji samo jedna grana, sva njena obeležja su izolovana, te je krajnji rezultat primene Grahamovog algoritma na hipergraf zavisnosti spoja $\triangleright\triangleleft(ABC, ABF, ACE, BCD)$, prazan hipergraf i zaključak da je zavisnost spoja aciklična. □

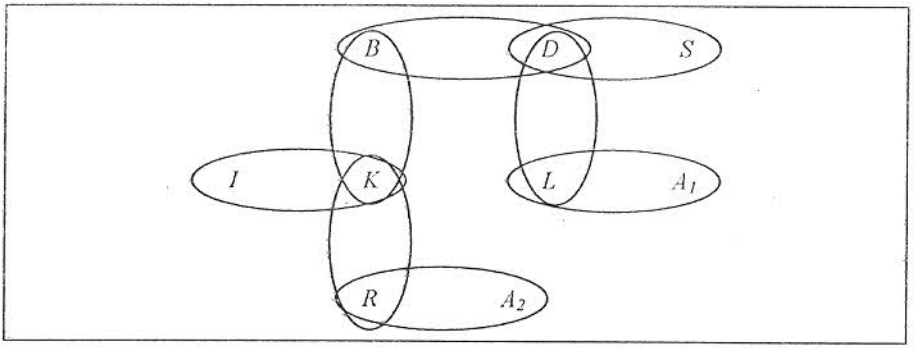
Primer 5.55. Lako se da utvrditi da je Grahamova redukcija hipergrafa zavisnosti spoja $\triangleright\triangleleft(KN, KP, NPU)$ neprazan hipergraf, prikazan na slici 5.29, te se zaključuje da je ta zavisnost spoja ciklična. □



Slika 5.29.

Proširenje skupa obeležja $\mathcal{U}$ novim obeležjima, predstavlja mogući postupak za transformaciju cikličnog u aciklični hipergraf. U literaturi, na primer [FMU], [PBG], se predlaže preimenovanje obeležja sa višestrukom ulogom. Preimenovanjem se u skup obeležja $\mathcal{U}$, umesto svakog obeležja sa višestrukom ulogom, uvodi po jedno novo obeležje za svaku od uloga. Jedan drugi postupak je zasnovan na pretpostavci da skup $\mathcal{U}$ ne sadrži sva obeležja, bitna za izgradnju modela realnog sistema. U ovom pristupu, obeležja sa više uloga ostaju u skupu obeležja $\mathcal{U}$, ali se on proširuje i nekim novim obeležjima. Oba ova postupka transformacije cikličnog u acikličan hipergraf ilustrovana su putem narednih primera.

Primer 5.56. U primeru 5.55, obeležje P (poslovni partner) ima dve uloge. Ako se obeležje P zameni obeležjima L (ulagač) i R (korisnik kredita), a obeležje A obeležjima A_1 (adresa ulagača) i A_2 (adresa korisnika kredita), dobija se hipergraf, prikazan na slici 5.30, koji je acikličan. □

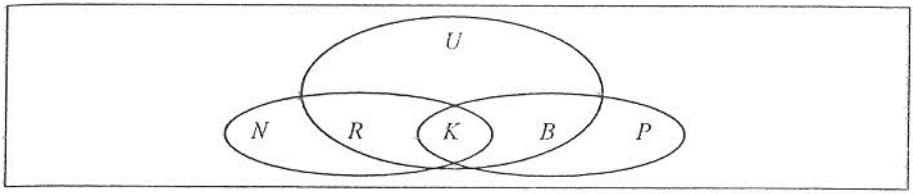


Slika 5.30.

Primer 5.57. Može se pretpostaviti da u realnom sistemu, opisanom u primeru 5.51, pored obeležja K (katedra), N (nastavnik), P (predmet) i U (učionica), egzistiraju i obeležja R (redni broj nastavnika u evidenciji) i B (redni broj predmeta u odgovarajućoj evidenciji). Opis tog realnog sistema bi bio:

- nastavnik n ima redni broj r u evidenciji katedre k,
- predmet p ima redni broj b u evidenciji katedre k,
- nastavnik sa rednim brojem r na katedri k predaje predmet sa rednim brojem b na (istoj) katedri k u učionici u.

Pod pretpostavkom da svaki nastavnik i svaki predmet pripada samo jednoj katedri, skupovi obeležja $\{N\}$ i $\{K, R\}$ nose istu semantiku. Isto važi i za skupove obeležja $\{P\}$ i $\{B, K\}$. Ova tvrdnja se može potkrepiti i činjenicom da u skupu obeležja $\{B, K, N, P, R\}$ važi skup funkcionalnih zavisnosti $\{N \rightarrow R,\ N \rightarrow K,\ R+K \rightarrow N,\ P \rightarrow B,\ P \rightarrow K,\ B+K \rightarrow P\}$. Saglasno tome, navedeni predikati definišu zavisnost spoja $\triangleright\triangleleft(KNR,\ BKP,\ BKRU)$, čiji hipergraf, prikazan na slici 5.31, je acikličan. □



Slika 5.31.

U principu, cikličnosti se uvek mogu izbeći uvođenjem novih obeležja u skup $\mathcal{U}$. Međutim, broj tako uvedenih novih obeležja može biti veoma veliki. Kao ilustracija ove tvrdnje može poslužiti skup $\mathcal{U} = \{A_1,...,A_n\}$ i zavisnost spoja, koja sadrži sve parove obeležja (A_i, A_j) za $i \neq j$ i $i, j \in \{1,...,n\}$.

5.5.4. *Implikacioni problem za funkcionalne i zavisnosti spoja*

Mada je problem aksiomatizacije zavisnosti spoja izučavan u određenom broju radova, kao što su [Sc1] i [BV], u teoriji relacionih baza podataka se smatra da je pitanje postojanja konačnog, neprotivrečnog i kompletnog skupa aksioma za izvođenje zavisnosti spoja još otvoren problem. S druge strane, razvijen je postupak za rešavanje implikacionog problema za funkcionalne i potpune zavisnosti spoja. Postupak će se nazivati ***implikacionim algoritmom.****) Pošto višeznačne zavisnosti predstavljaju specijalan slučaj zavisnosti spoja, postupak je primenljiv i na višeznačne zavisnosti. Sam implikacioni algoritam je zasnovan na pojmovima: ***preslikavanja simbola*** i ***tabloa.***

Preslikavanje simbola

Da bi se definisali pojmovi preslikavanja simbola i tabloa, uvode se oznake za skup vrednosti i skup simbola. ***Skup vrednosti***, u oznaci *Dom*, predstavlja uniju:

$$Dom = \bigcup_{A \in U} dom(A).$$

Skup simbola, u oznaci *Sym*, predstavlja uniju dva skupa promenljivih:

$$Sym = V_p \cup V_n.$$

Skup $V_p = \{\alpha_A \mid A \in \mathcal{U}\}$ predstavlja skup ***poznatih promenljivih,*****) indeksiranih putem obeležja iz skupa $\mathcal{U}$. Skup simbola $V_n = \{\beta_A^1, \beta_A^2, ... \mid A \in \mathcal{U}\}$ predstavlja beskonačan

*) U literaturi se ovaj postupak naziva CHASE algoritmom. Engleska reč chase se odnosi na poteru, gonjenje. U ovom slučaju, ukazuje na traženje posledica zadatog skupa zavisnosti iscrpljivanjem.

**) Engleski: distinguished variables

skup ***nepoznatih promenljivih***, indeksiranih putem obeležja iz skupa $\mathcal{U}$ i putem elemenata iz skupa prirodnih brojeva. □

U daljem tekstu će se poznate promenljive nazivati i ***karakterističnim***, a nepoznate ***nekarakterističnim*** promenljivama.

Definicija 5.28. Preslikavanje simbola u vrednosti je neka funkcija g: $Sym \rightarrow Dom$, dok je ***preslikavanje vrednosti u simbole*** bilo koja funkcija h: $Dom \rightarrow Sym$. □

Pojam preslikavanja simbola u vrednosti se može generalizovati na preslikavanje n-torki simbola u n-torke vrednosti, putem funkcije g : $Sym^n \rightarrow Dom^n$. Ako je $s = (a_1, ..., a_n)$ n-torka, čiji elementi pripadaju skupu Sym, primenom funkcije g na s, dobija se n-torka $g(s) = g(a_1, ..., a_n) = (g(a_1), ..., g(a_n))$, čiji elementi predstavljaju vrednosti iz Dom. Analogno, uvodi se preslikavanje n-torki vrednosti u n-torke simbola: h: $Dom^n \rightarrow Sym^n$, tako da za bilo koju n-torku vrednosti t važi: $h(t) = h(d_1, ..., d_n) = (h(d_1), ..., h(d_n))$.

Prirodno dalje proširenje pojmova preslikavanja n-torki simbola i n-torki vrednosti predstavljaju pojmovi preslikavanja skupova n-torki simbola i n-torki vrednosti.

Definicija 5.29. Neka je $\{s_1, ..., s_k\}$ skup n-torki simbola, a $\{t_1, ..., t_m\}$ skup n-torki vrednosti. Funkcija g preslikava n-torke prvog u n-torke drugog skupa, ako važi

$$(\forall i \in \{1, ..., k\})(\exists j \in \{1, ..., m\})(g(s_i) = t_j).$$

Funkcija h preslikava n-torke drugog u n-torke prvog skupa, ako važi

$$(\forall i \in \{1, ..., m\})(\exists j \in \{1, ..., k\})(h(t_i) = s_j). \square$$

Primer 5.58. Na slici 5.32 je prikazana jedna relacija r nad skupom obeležja $\mathcal{U} = \{A, B, C\}$, kao skup trojki vrednosti, jedna tabela τ nad istim skupom obeležja $\mathcal{U}$, kao skup trojki promenljivih, tabelarno zadata funkcija h, koja preslikava trojke vrednosti u trojke promanljivih i funkcija g, koja preslikava trojke promenljivih u trojke vrednosti.

Funkcija h preslikava sve četiri trojke vrednosti iz r u trojku $(\alpha_A, \alpha_B, \beta_C^1)$, dok funkcija g preslikava trojku $(\alpha_A, \alpha_B, \beta_C^1)$ u (a_1, b_1, c_1), a trojku $(\beta_A^2, \alpha_B, \alpha_C)$ u trojku (a_2, b_1, c_2). Funkcija g ujedno predstavlja ilustraciju tvrđenja da se može naći funkcija koja će zadatu tabelu τ preslikati u podskup skupa torki zadate relacije r. □

$r =$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2
a_1	b_1	c_2
a_2	b_1	c_1

$\tau =$

A	B	C
α_A	α_B	β_C^1
β_A^2	α_B	α_C

x	$h(x)$
a_1	α_A
b_1	α_B
c_1	β_C^1
a_2	β_C^2
c_2	α_C

x	$g(x)$
α_A	a_1
α_B	b_1
α_C	c_2
β_A^2	a_2
β_C^1	c_1

Slika 5.32.

Tablo

Definicija 5.30. Tablo τ nad skupom obeležja $\mathcal{U}$ je konačan skup vrsta (n-torki) l, takvih da svaki l predstavlja jedno preslikavanje $\mathcal{U} \to V_p \cup V_n$. □

Neka je dat tablo τ i neka je $l \in \tau$. Skup obeležja za koji vrsta l sadrži poznate promenljive se označava kao: $\widetilde{l} = \{A \in \mathcal{U} \mid l[A] \in V_p\}$.

Primer 5.59. Tabela τ na slici 5.32 predstavlja tablo nad skupom obeležja $\{A, B, C\}$. □

Implikacioni algoritam služi, u opštem slučaju, za proveru pretpostavke da je zavisnost d posledica skupa zavisnosti $\mathcal{D}$. Osnovna ideja ovog algoritma je u sledećem:

- prvo se konstruiše inicijalni tablo, $\tau(d)$, za zavisnost d,
- zatim se inicijalni tablo modifikuje iscrpnom primenom zavisnosti iz $\mathcal{D}$,
- zavisnosti iz $\mathcal{D}$ se primenjuju na tablo da bi se on doveo u određeno stanje,
- primena zavisnosti na tablo se prekida ili po prevođenju tabloa u željeno stanje, ili u slučaju da dalja primena zavisnosti ne dovodi do dalje modifikacije tabloa,
- ako se tablo prevede u željeno stanje, zaključuje se da važi $\mathcal{D} \models d$, inače se zaključuje da važi $\neg(\mathcal{D} \models d)$,
- tablo τ, koji se više ne može modifikovati primenom zavisnosti iz $\mathcal{D}$, označava se sa $chase_{\mathcal{D}}(\tau)$, ili samo $chase(\tau)$, ako se $\mathcal{D}$ podrazumeva iz konteksta.

Na ovom mestu će pretežna pažnja biti posvećena rešavanju implikacionog problema za slučaj da je $\mathcal{D}$ konačan skup funkcionalnih zavisnosti $f_1, f_2, \ldots, f_k$ i zavisnosti spoja $j_1, j_2, \ldots, j_m$ a da je d neka zavisnost spoja j.

Primena zavisnosti na tablo znači njegovo prevođenje u stanje, koje zadovoljava svaku funkcionalnu zavisnost iz $\mathcal{D}$ (saglasno definiciji 5.7) i svaku zavisnost spoja iz $\mathcal{D}$ (saglasno definiciji 5.24). To dalje znači da ako funkcionalna zavisnost $X \to Y$ pripada $\mathcal{D}$, a tablo sadrži vrste l_i i l_j takve da je $l_i[X] = l_j[X]$ i $l_i[Y] \neq l_j[Y]$, tablo se modifikuje izjednačavanjem $l_i[Y]$ i $l_j[Y]$ simbola. Isto tako, ako $\mathcal{D}$ sadrži zavisnost spoja $\bowtie(X_1, \ldots, X_k)$, a tablo sadrži vrste $l_1, l_2, \ldots, l_k$, takve da je $(\forall i, j \in \{1, \ldots, k\})(l_i[X_i \cap X_j] = l_j[X_i \cap X_j])$, ali ne i vrstu l, za koju važi $(\forall i \in \{1, \ldots, k\})(l[X_i] = l_i[X_i])$, tablo se proširuje vrstom l.

Definicija 5.31. Inicijalni tablo za zavisnost spoja j: $\bowtie(X_1, \ldots, X_k)$, gde je $\mathcal{U} = \bigcup_{i=1}^{k} X_i$, u oznaci $\tau(j)$, je $\tau(j) = \{l_1, l_2, \ldots, l_k\}$, takav da važi:

1° $(\forall i \in \{1, \ldots, k\})(\forall A \in X_i)(l_i[A] = \alpha_A)$ i

2° $(\forall i \in \{1, \ldots, k\})(\forall A \notin X_i)(l_i[A] = \beta_A^i)$. □

Primer 5.60. Tablo na slici 5.32 predstavlja inicijalni tablo za zavisnost spoja $\bowtie(AB, BC)$. □

Tablo se, u opštem slučaju, može shvatiti kao podskup neke formalne relacije r nad skupom obeležja $\mathcal{U}$. Konstrukcija inicijalnog tabloa $\tau(j)$ za zavisnost spoja j:

$\triangleright\!\triangleleft(X_1,..., X_k)$, podseća na definiciju 5.24. Vrste $l_1, l_2,..., l_k$ tabloa $\tau(j)$ se dobijaju preslikavanjem torki $t_1, t_2,..., t_k$ relacije r nad $\mathcal{U}$, putem funkcije h: $Dom \rightarrow V_p \cup V_n$.

Torke $t_1,..., t_k$ zadovoljavaju uslov $(\forall i, j \in \{1,..., k\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j])$. Slika svakog $t_i[X_i]$ u tablou $\tau(j)$ sadrži poznate promenljive, tako da bi slika torke t, koja zadovoljava uslov $(\forall i \in \{1,..., k\})(t[X_i] = t_i[X_i])$, bila vrsta l, koja sadrži samo poznate promenljive ($\widetilde{l} = \mathcal{U}$).

Sada se može obrazložiti i pojam "*željenog stanja tabloa*". S obzirom na postupak konstruisanja inicijalnog tabloa $\tau(j)$, sledi da ako skup zavisnosti $\mathcal{D}$ implicira zavisnost j, tada će se primenom zavisnosti iz skupa $\mathcal{D}$ na tablo $\tau(j)$, u tablou pojaviti vrsta l, za koju će važiti $\widetilde{l} = \mathcal{U}$. Pojava te vrste predstavlja željeno stanje tabloa.

Primer 5.61. Neka je $\mathcal{D} = \{C \rightarrow D, \triangleright\!\triangleleft(AB, BCD)\}$, a j: $\triangleright\!\triangleleft(AB, BC, CD)$. Na slici 5.33 je prikazan inicijalni tablo $\tau(j)$. Pošto je $l_2[C] = l_3[C]$ i $l_2[D] \neq l_3[D]$, primena funkcionalne zavisnosti $C \rightarrow D$ na tablo $\tau(j)$, dovodi do njegove modifikacije, prema slici 5.34. Ako se sada, na tablo sa slike 5.34 primeni zavisnost spoja $\triangleright\!\triangleleft(AB, BCD)$, dobija se tablo, prikazan na slici 5.35. Pošto vrsta l_4 sadrži samo poznate promenljive, zaključuje se da važi $\mathcal{D} \models j$. □

	A	B	C	D
l_1	α_A	α_B	β_C^1	β_D^1
l_2	β_A^2	α_B	α_C	β_D^2
l_3	β_A^3	β_B^3	α_C	α_D

Slika 5.33.

	A	B	C	D
l_1	α_A	α_B	β_C^1	β_D^1
l_2	β_A^2	α_B	α_C	α_D
l_3	β_A^3	β_B^3	α_C	α_D

Slika 5.34.

	A	B	C	D
l_1	α_A	α_B	β_C^1	β_D^1
l_2	β_A^2	α_B	α_C	α_D
l_3	β_A^3	β_B^3	α_C	α_D
l_4	α_A	α_B	α_C	α_D
l_5	β_A^2	α_B	β_C^1	β_D^1

Slika 5.35.

Implikacioni algoritam

Na slici 5.36 prikazan je u psedokodu implikacioni algoritam.

Stav 5.1. Implikacioni algoritam se završava nakon konačnog broja koraka.

Dokaz. Inicijalni tablo za zavisnost spoja j sadrži k vrsta sa po $|\mathcal{U}|$ simbola. Znači, inicijalni tablo sadrži najviše $k|\mathcal{U}|$ simbola. Sa $k|\mathcal{U}|$ različitih simbola se može konstruisati najviše $(k|\mathcal{U}|)^{|\mathcal{U}|}$ različitih vrsta. Pošto implikacioni algoritam ne uvodi nove promenljive u postupak, najviše nakon $(k|\mathcal{U}|)^{|\mathcal{U}|}$ prolaza kroz postupak modifikacije tabloa, algoritam će se i završiti. □

Direktna posledica stava 5.1 je zaključak da je ocena kompleksnosti implikacionog algoritma eksponencijalna, s obzirom na veličinu ulaza.

IMPLIKACIONI ALGORITAM

PROCES Impl_alg
Ulaz: $\mathcal{U}, \mathcal{D}, j$ (* $\mathcal{D}$ je skup zavisnosti nad $\mathcal{U}$ i j je zavisnost spoja *)
Izlaz: $\tilde{l}$ (* ako $\mathcal{D} \models j$, tada $\tilde{l} = \mathcal{U}$, inače $\tilde{l} = \emptyset$ *)
POČETAK PROCESA Impl_alg
POSTAVI $TAB \leftarrow \tau(j)$ (*inicijalni tablo*)
POSTAVI $STARI \leftarrow \emptyset$
RADI modifikacija ***DOK JE*** $TAB \neq STARI$
POSTAVI $STARI \leftarrow TAB$
RADI funk_zav $(\forall X \rightarrow A \in \mathcal{D})$
RADI torke_2 $(\forall l_i, l_j \in TAB)$
AKO JE $l_i[X] = l_j[X]$ *TADA*
AKO JE $l_i[A] = \alpha_A \vee l_j[A] = \alpha_A$ *TADA*
POSTAVI $l_i[A] \leftarrow \alpha_A$
POSTAVI $l_j[A] \leftarrow \alpha_A$
INAČE
POSTAVI $l_i[A] \leftarrow \beta_A^{min(i,j)}$
POSTAVI $l_j[A] \leftarrow \beta_A^{min(i,j)}$
KRAJ AKO
KRAJ AKO
KRAJ RADI torke_2
KRAJ RADI funk_zav
RADI zavisnost_spoja $(\forall \triangleright\triangleleft(Y_1, \ldots, Y_n) \in \mathcal{D})$
RADI torke_n $(\forall l_1, \ldots, l_n \in TAB)$
AKO JE $(\forall i, j \in \{1, \ldots, n\})(l_i[Y_i \cap Y_j] = l_j[Y_i \cap Y_j])$ *TADA*
RADI formiranje_1 $(\forall i \in \{1, \ldots, n\})$
POSTAVI $l[Y_i] \leftarrow l_i[Y_i]$
POSTAVI $TAB \leftarrow TAB \cup \{l\}$
KRAJ RADI formiranje_1
KRAJ AKO
KRAJ RADI torke_n
KRAJ RADI zavisnost_spoja
KRAJ RADI modifikacija
AKO JE $(\exists l \in TAB)(\forall A \in \mathcal{U})(l[A] = \alpha_A)$ *TADA*
POSTAVI $\tilde{l} \leftarrow \mathcal{U}$
INAČE
POSTAVI $\tilde{l} \leftarrow \emptyset$
KRAJ AKO
KRAJ PROCESA *Impl_alg*

Slika 5.36.

Teorema 5.12. Tablo $chase(\tau(j))$ sadrži vrstu l, takvu da je $\widetilde{l} = \mathcal{U}$, ako i samo ako $\mathcal{D} \models j$.

Dokaz. ($\Rightarrow$) Pretpostavlja se da važi $\mathcal{D} \models j$: $\triangleright\triangleleft(X_1,..., X_k)$. Treba pokazati da $chase(\tau(j))$ sadrži vrstu l, za koju važi $\widetilde{l} = \mathcal{U}$. Neka je g: $Sym \rightarrow Dom$ injektivna funkcija, takva da svaku promenljivu indeksiranu sa A preslikava u $dom(A)$. Neka je $r = g(chase(\tau(j)))$. Pošto $chase(\tau(j))$ zadovoljava $\mathcal{D}$, zadovoljava ga i relacija r, a saglasno pretpostavci, r zadovoljava i zavisnost spoja j. S obzirom na konstrukciju inicijalnog tabloa, važi $(\forall i, j \in \{1,..., k\})(g(l_i[X_i \cap X_j]) = g(l_j[X_i \cap X_j]))$, te se zaključuje da relacija r sadrži torku $g(l)$, za koju važi $(\forall i \in \{1,..., k\})(g(l[X_i]) = g(l_i[X_i]))$. Pošto je g injekcija, sledi da $chase(\tau(j))$ sadrži vrstu l, za koju je $\widetilde{l} = \mathcal{U}$.

($\Leftarrow$) Pretpostavlja se da važi $(\exists l \in chase(\tau(j)))(\widetilde{l} = \mathcal{U})$. Treba pokazati da važi $\mathcal{D} \models j$: $\triangleright\triangleleft(X_1,..., X_k)$. Neka je r relacija iz $SAT(\mathcal{U}, \mathcal{D})$, a $t_1,..., t_k$ proizvoljne torke iz r, takve da zadovoljavaju pretpostavku definicije 5.24: $(\forall i, j \in \{1,..., k\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j])$. Treba pokazati da r sadrži odgovarajuću torku t: $(\forall i \in \{1,..., k\})(t[X_i] = t_i[X_i])$.

S obzirom na postupak konstruisanja $\tau(j)$, može se naći funkcija g: $Sym \rightarrow Dom$, koja preslikava vrste $l_1,..., l_k$ tabloa $\tau(j)$, redom, u torke $t_1,..., t_k$ relacije r. Znači, $g(\tau(j)) \subseteq r$. Indukcijom po broju izvršenih modifikacija inicijalnog tabloa $\tau(j)$ će biti pokazano da važi $g(chase(\tau(j))) \subseteq r$, iz čega sledi da za torku $l \in chase(\tau(j))$, za koju je $\widetilde{l} = \mathcal{U}$, važi $g(l) \in r$, pri čemu $g(l)$ ispunjava zahtevani uslov $(\forall i \in \{1,..., k\})(g(l)[X_i] = t_i[X_i])$.

1^0 Baza indukcije. Neka je broj izvršenih modifikacija inicijalnog tabloa $n = 0$. To znači da je $chase(\tau(j)) = \tau(j)$, iz čega sledi da je $g(chase(\tau(j))) = g(\tau(j)) \subseteq r$.

2^0 Pretpostavka indukcije. Neka je τ_n tablo, dobijen tokom primene implikacionog algoritma na $\tau(j)$, pri čemu je izvršeno najviše n modifikacija $\tau(j)$. Pretpostavlja se da važi $g(\tau_n) \subseteq r$.

3^0 Dokaz indukcije. Neka je τ_{n+1} tablo, dobijen modifikacijom tabloa τ_n, putem jedne funkcionalne zavisnosti, ili zavisnosti spoja iz $\mathcal{D}$.

- Ako je τ_n modifikovan putem funkcionalne zavisnosti $X \rightarrow Y$, onda je on sadržao vrste l_i i l_j takve da je $l_i[X] = l_j[X]$, te je i $g(l_i[X]) = g(l_j[X])$. Pošto relacija r zadovoljava $\mathcal{D}$, važi $g(l_i[Y]) = g(l_j[Y])$. Tablo τ_{n+1} je dobijen izjednačavanjem promenljivih u Y kolonama. Pošto su obe izjednačene promenljive već preslikavane u iste vrednosti, sledi $g(\tau_{n+1}) \subseteq r$.
- Ako je τ modifikovan putem zavisnosti spoja $\triangleright\triangleleft(Y_1,..., Y_n)$, dobijena je nova vrsta l', te je $\tau_{n+1} = \tau_n \cup \{l'\}$. Pošto relacija r zadovoljava skup zavisnosti $\mathcal{D}$, zaključuje se da je $g(l')$ u r, odnosno da važi $g(\tau_{n+1}) \subseteq r$.

Konačno, zaključuje se da važi $g(chase(\tau(j))) \in r$, odnosno $g(l) = t \in r$. □

Primer 5.62. Neka je $\mathcal{U} = \{A, B, C, D, E, F, G\}$, $\mathcal{D} = \{\triangleright\triangleleft(ABCD, DEFG), \triangleright\triangleleft(ABDEG, ACDFG)\}$, j: $\triangleright\triangleleft(ABD, ACD, DEG, DFG)$. Treba proveriti da li $\mathcal{D} \models j$. Na slici 5.37 je prikazan tablo $\tau(j)$. Pošto važi $l_1[ABCD \cap DEFG] = l_3[ABCD \cap DEFG]$, primenom odgovarajuće zavisnosti spoja se dobija vrsta l_5, za koju važi $l_5[ABCD] = l_1[ABCD]$ i $l_5[DEFG] = l_3[DEFG]$. Takođe važi $l_2[ABCD \cap DEFG]$ i $l_4[ABCD \cap DEFG]$,

te se dobija vrsta l_6 sa $l_6[ABCD] = l_2[ABCD]$ i $l_6[DEFG] = l_4[DEFG]$. Sada važi $l_5[ABDEG \cap ACDFG] = l_6[ABDEG \cap ACDFG]$, te se primenom odgovarajuće zavisnosti spoja iz $\mathcal{D}$, dobija vrsta l_7, koja, saglasno slici 5.38, sadrži samo poznate promenljive. □

	A	B	C	D	E	F	G
l_1	α_A	α_B	β_C^1	α_D	β_E^1	β_F^1	β_G^1
l_2	α_A	β_B^2	α_C	α_D	β_E^2	β_F^2	β_G^2
l_3	β_A^3	β_B^3	β_C^3	α_D	α_E	β_F^3	α_G
l_4	β_A^4	β_B^4	β_C^4	α_D	β_E^4	α_F	α_G

Slika 5.37.

	A	B	C	D	E	F	G
l_1	α_A	α_B	β_C^1	α_D	β_E^1	β_F^1	β_G^1
l_2	α_A	β_B^2	α_C	α_D	β_E^2	β_F^2	β_G^2
l_3	β_A^3	β_B^3	β_C^3	α_D	α_E	β_F^3	α_G
l_4	β_A^4	β_B^4	β_C^4	α_D	β_E^4	α_F	α_G
l_5	α_A	α_B	β_C^1	α_D	α_E	β_F^3	α_G
l_6	α_A	β_B^2	α_C	α_D	β_E^4	α_F	α_G
l_7	α_A	α_B	α_C	α_D	α_E	α_F	α_G

Slika 5.38.

Primer 5.63. Neka je $\mathcal{U} = \{K, N, P, U\}$, $\mathcal{D} = \{N \to K, P \to K, NP \to U\}$, a j: $\triangleright\triangleleft(KN, KP, NPU)$. Primena implikacionog algoritma na inicijalni tablo, prikazan na slici 5.39, rezultuje u tablou na slici 5.40. Zaključuje se da je zavisnost spoja j: $\triangleright\triangleleft(KN, KP, NPU)$ logička posledica skupa funkcionalnih zavisnosti u $\mathcal{D}$. □

K	N	P	U
α_K	α_N	β_P^1	β_U^1
α_K	β_N^2	α_P	β_U^2
β_K^3	α_N	α_P	α_U

Slika 5.39.

K	N	P	U
α_K	α_N	β_P^1	β_U^1
α_K	β_N^2	α_P	β_U^2
α_K	α_N	α_P	α_U

Slika 5.40.

Uz neznatne korekcije, implikacioni algoritam sa slike 5.36 se može koristiti i za proveru da li funkcionalna zavisnost $f: X \to Y$ predstavlja logičku posledicu skupa funkcionalnih zavisnosti $\mathcal{F}$. U tom cilju:

- gradi se inicijalni tablo $\tau(f)$ sa dve vrste l_1 i l_2, takve da važi:
  - $(\forall A \in XY)(l_1[A] = \alpha_A)$
  - $(\forall A \notin XY)(l_1[A] = \beta_A^1)$
  - $(\forall A \in X)(l_2[A] = \alpha_A)$
  - $(\forall A \notin X)(l_2[A] = \beta_A^2)$,
- zaključuje se da $\mathcal{F} \models f$, ako važi $(\forall l \in chase(\tau(f)))(\forall A \in Y)(l[A] = \alpha_A)$.

Primer 5.64. Neka je $\mathcal{F} = \{AB \to C, C \to D, D \to B\}$ i $f: AD \to C$. Na slici 5.41 je prikazan inicijalni tablo za funkcionalnu zavisnost $AD \to C$. Prvi prolaz kroz postupak "funkcionalna_zavisnost", dovodi samo do izjednačavanja β_B^1 i β_B^2 simbola, zbog funk-

cionalne zavisnosti $D \rightarrow B$ (slika 5.42). Tek u narednom prolasku kroz isti postupak, dolazi do izjednačavanja C simbola, na osnovu funkcionalne zavisnosti $AB \rightarrow C$ (slika 5.43), te se zaključuje da je $f: AD \rightarrow C$ logička posledica skupa $\mathcal{F}$. □

	A	B	C	D
l_1	α_A	β_B^1	α_C	α_D
l_2	α_A	β_B^2	β_C^2	α_D

Slika 5.41.

	A	B	C	D
l_1	α_A	β_B^1	α_C	α_D
l_2	α_A	β_B^1	β_C^2	α_D

Slika 5.42.

	A	B	C	D
l_1	α_A	β_B^1	α_C	α_D
l_2	α_A	β_B^1	α_C	α_D

Slika 5.43.

Višeznačne zavisnosti i zavisnost spoja

Postoji relativno jednostavan postupak za izvođenje višeznačne zavisnosti na osnovu zadate zavisnosti spoja. Postupak je zasnovan na posmatranju hipergrafa zavisnosti spoja.

Definicija 5.32. Put od čvora A do čvora B hipergrafa, je niz grana $E_1,\ldots,E_k$ takav da važi:

1^o A je u E_1,

2^o B je u E_k i

3^o $(\forall i \in \{1,\ldots,k\})(E_i \cap E_{i+1} \neq \emptyset)$. □

Definicija 5.33. Dva čvora hipergrafa su povezana, ako postoji put od jednog čvora do drugog. □

Primer 5.65. Čvorovi K i U hipergrafa na slici 5.25 su povezani, jer postoji put KN, NPU između ta dva čvora. □

Teorema 5.13. Neka je j: $\triangleright\triangleleft(X_1,\ldots,X_k)$ zavisnost spoja, takva da je $\mathcal{U} = \bigcup_{i=1}^{k} X_i$, a $Y \subseteq \mathcal{U}$. Višeznačna zavisnost $Y \rightarrow\rightarrow Z$, gde je $Y \cap Z = \emptyset$, predstavlja logičku posledicu zavisnosti spoja j, ako i samo ako Z predstavlja uniju nekih od povezanih komponenata - hipergrafa zavisnosti spoja j, kada se čvorovi skupa Y uklone*) iz hipergrafa.

Dokaz. ($\Rightarrow$) Pretpostavlja se da je Z unija povezanih komponenata hipergrafa zavisnosti spoja j, nakon uklanjanja čvorova iz Y. Tada je $\mathcal{U} \setminus Z$ unija preostalih povezanih komponenata. Da bi se pokazalo da višeznačna zavisnost $Y \rightarrow\rightarrow Z$ predstavlja logičku posledicu zavisnosti spoja j, koristiće se implikacioni algoritam. Neka je $Y = \{A_1,\ldots, A_k\}$, $Z = \{A_{k+1},\ldots, A_m\}$, a $\mathcal{U} \setminus YZ = \{A_{m+1},\ldots, A_n\}$. Inicijalni tablo za višeznačnu zavisnost $Y \rightarrow\rightarrow Z$, koja je ekvivalentna zavisnosti spoja $\triangleright\triangleleft(YZ, Y(\mathcal{U} \setminus YZ))$, je prikazan na slici 5.44.

---

*) Čvor se uklanja kako iz skupa čvorova, tako i iz svih grana hipergrafa, koje ga sadrže. Pri tome, grana ostaje u hipergrafu sve dok se ne ukloni i poslednji čvor koji obuhvata.

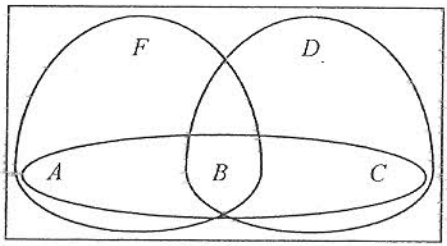
Y	Z	$\mathcal{U}\setminus YZ$
$\alpha_{A_1}...\alpha_{A_k}$	$\alpha_{A_{k+1}}...\alpha_{A_m}$	$\beta_{A_{m+1}}^{\ 1}...\beta_{A_n}^{\ 1}$
$\alpha_{A_1}...\alpha_{A_k}$	$\beta_{A_{k+1}}^{\ 2}...\beta_{A_m}^{\ 2}$	$\alpha_{A_{m+1}}...\alpha_{A_n}$

Slika 5.44.

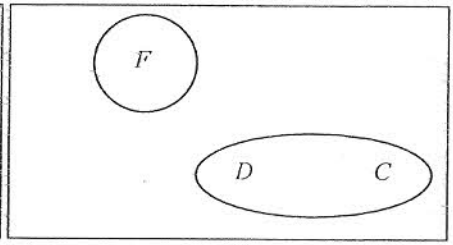
Svaka komponenta X_i zavisnosti spoja j: $\triangleright\triangleleft(X_1,...,X_k)$ predstavlja podskup bilo skupa YZ ili $Y(\mathcal{U}\setminus YZ)$, inače skupovi Z i $\mathcal{U}\setminus Z$ ne bi predstavljali unije povezanih komponenata zavisnosti spoja j, nakon uklanjanja čvorova iz Y. Saglasno rečenom, primena zavisnosti spoja j na inicijalni tablo za zavisnost spoja $\triangleright\triangleleft(YZ, Y(\mathcal{U}\setminus YZ))$ proizvešće vrstu sa poznatim promenljivama.

($\Leftarrow$) Neka $Y\rightarrow\rightarrow Z$ predstavlja logičku posledicu zavisnosti spoja j. Uvodi se suprotna pretpostavka, po kojoj postoji komponenta X_i zavisnosti spoja j takva da je $X_i \cap Z \neq \emptyset$ i, nakon uklanjanja čvorova iz Y, $X_i \cap (\mathcal{U}\setminus YZ) \neq \emptyset$. Sada inicijalni tablo za višeznačnu zavisnost $Y\rightarrow\rightarrow Z$ ne sadrži samo karakteristične promenljive za komponentu X_i ni u kolonama za YZ ni u kolonama za $(\mathcal{U}\setminus YZ)$. Primena implikacionog algoritma neće dovesti do generisanja vrste sa samo karakterističnim promeljivama, te se zaključuje da zavisnost spoja j ne implicira višeznačnu zavisnost $Y\rightarrow\rightarrow Z$, što je kontradiktorno polaznoj pretpostavci. □

Primer 5.66. Na slici 5.45 je prikazan hipergraf aciklične zavisnosti spoja $\triangleright\triangleleft(ABC, ABF, BCD)$. Neka je $Y = \{A, B\}$. Rezultat uklanjanja čvorova iz Y iz hipergrafa sa slike 5.45, prikazan je na slici 5.46. Zaključuje se da zavisnost spoja $\triangleright\triangleleft(ABC, ABF, BCD)$ implicira višeznačnu zavisnost $AB\rightarrow\rightarrow F \mid CD$. □

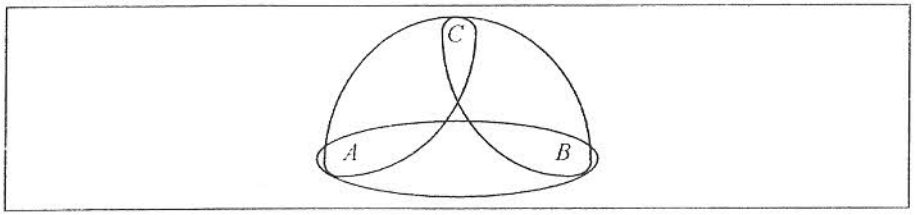


Slika 5.45.



Slika 5.46.

Primer 5.67. Na slici 5.47 je prikazan hipergraf ciklične zavisnosti spoja $\triangleright\triangleleft(AB, BC, AC)$. Izbor bilo kog podskupa skupa obeležja $\{A, B, C\}$ za Y, vodi zaključku da zavisnost spoja $\triangleright\triangleleft(AB, BC, AC)$ implicira samo trivijalne višeznačne zavisnosti, jer nijedno opredelenje za Y ne dovodi do rastavljanja hipergrafa na više od jedne povezane komponente. □



Slika 5.47.

Teorema 5.13 definiše postupak za pronalaženje skupa višeznačnih zavisnosti, koje su logička posledica jedne zavisnosti spoja. Ako skup ograničenja šeme relacije sadrži i netrivijalne funkcionalne zavisnosti, neke druge višeznačne zavisnosti su posledica tih funkcionalnih zavisnosti.

5.6. *Zavisnost sadržavanja i referencijalni integritet*

Zavisnost sadržavanja predstavlja ograničenje, koje se značajno razlikuje od ograničenja kakva su funkcionalne zavisnosti i zavisnosti spoja. Funkcionalna zavisnost i zavisnost spoja uspostavljaju vezu između skupova obeležja X i Y ($X, Y \subseteq \mathcal{U}$), ne vodeći računa o odnosu između domena tih obeležja. Zavisnost sadržavanja između skupova obeležja X i Y uvodi uslov da mora važiti $dom(X) \subseteq dom(Y)$, da bi se ta zavisnost između skupova X i Y uopšte mogla definisati. Za skupove obeležja X i Y, koji zadovoljavaju uslov $dom(X) \subseteq dom(Y)$, kaže se da su domen kompatibilni.

Primer 5.68. Neka je $\mathcal{U}$ = {*MBR*, *IME*, *PRZ*, *ZAN*, *RUK*, *IMR*, *PRR*}, gde većina mnemonika obeležja ima već ranije uvedena značenja, osim *RUK* (matični broj rukovodioca), *IMR* (ime rukovodioca) i *PRR* (prezime rukovodioca). Svaka torka relacije nad $\mathcal{U}$ sadrži podatke o jednom radniku i njegovom rukovodiocu. Međutim, i svaki rukovodilac je radnik, tako da mu odgovara torka sa njegovim matičnim brojem radnika (*MBR*) i ostalim podacima. Prirodno se nameće zaključak da mora važiti $dom(IMR) \subseteq dom(IME)$, $dom(PRR) \subseteq dom(PRZ)$ i $dom(RUK) \subseteq dom(MBR)$. Ako je r relacija nad skupom obeležja $\mathcal{U}$, važiće i

$$\pi_{\{RUK, PRR, IMR\}}(r) \subseteq \pi_{\{MBR, PRZ, IME\}}(r). \square$$

Definicija 5.34. Neka je r relacija nad $\mathcal{U}$, $A_1,..., A_n$ i $B_1,..., B_n$ dva niza različitih obeležja iz $\mathcal{U}$. Relacija r zadovoljava zavisnost sadržavanja $[A_1,..., A_n] \subseteq [B_1,..., B_n]$, ako važi

$$(\forall t \in r)(\exists u \in r)(\forall i \in \{1,..., n\})(t[A_i] = u[B_i]). \square$$

Treba podvući dve činjenice, koje proističu iz definicije 5.34. Jedna je da je redosled nabrajanja obeležja u zavisnosti sadržavanja bitan, da bi se održao uslov domenske kompatibilnosti. Druga je da torke sa istim A_i i B_i vrednostima mogu, a ne moraju biti iste.

Zavisnost sadržavanja se često definiše i kao međurelaciono ograničenje, putem kojeg se kontroliše usaglašenost (konzistentnost) vrednosti onih domen kompatibilnih obeležja, koja se javljaju u više šema relacija. Zadovoljavanje ovih zavisnosti sadržavanja se proverava pri ažuriranju relacija.

Definicija 5.35. Neka su r_i i r_j relacije nad različitim šemama relacija $N_i(R_i, C_i)$ i $N_j(R_j, C_j)$, $A_1,..., A_n \in R_i$, a $B_1,..., B_n \in R_j$. Zavisnost sadržavanja

$$N_i[A_1,..., A_n] \subseteq N_j[B_1,..., B_n]$$

je zadovoljena, ako važi

$$(\forall t_i \in r_i)(\exists t_j \in r_j)(\forall k \in \{1,..., n\})(t_i[A_k] = t_j[B_k]). \square$$

Primer 5.69. Na slici 5.48 su prikazane pojave nad šemama relacija $N_1(\{K, N\}, \{N\})$, $N_2(\{K, P\}, \{P\})$ i $N_3(\{N, P, U\}, \{NP\})$. Ove pojave zadovoljavaju sledeće zavisnosti sadržavanja:

$$N_3[N] \subseteq N_1[N],\ N_3[P] \subseteq N_2[P],\ N_2[P] \subseteq N_3[P] \text{ i } N_2[K] \subseteq N_1[K].$$

Pojave na slici 5.48 ne zadovoljavaju sledeće zavisnosti sadržavanja:

$$N_1[N] \subseteq N_3[N] \text{ i } N_1[K] \subseteq N_2[K].$$

$r(KN) =$

K	N
k_1	n_1
k_1	n_2
k_2	n_3
k_3	n_4

$r(KP) =$

K	P
k_1	p_1
k_2	p_2
k_2	p_3
k_2	p_4

$r(NPU) =$

N	P	U
n_1	p_1	u_1
n_3	p_2	u_1
n_3	p_3	u_2
n_3	p_4	u_3

Slika 5.48.

Neka je N_1 = *Nastavnik*, N_2 = *Predmet*, a N_3 = *Nastava*. Semantika zavisnosti sadržavanja *Nastava*[N] $\subseteq$ *Nastavnik*[N] je da nastavu ne može izvoditi nastavnik koji ne pripada nekoj od katedri. Slično, semantika zavisnosti sadržavanja *Nastava*[P] $\subseteq$ *Predmet*[P] je da se ne može izvoditi nastava iz predmeta koji ne pripada nijednoj katedri.

Ineresantno je napomenuti da bi se definisanjem zavisnosti sadržavanja *Nastavnik*[K] $\subseteq$ *Predmet*[K] uvelo ograničenje, čija semantika bi bila da nastavnik n može pripadati katedri k samo ako ta katedra ima bar jedan predmet p. Analognu semantičku interpretaciju bi imala i zavisnost sadržavanja *Predmet*[K] $\subseteq$ *Nastavnik*[K]. Semantička interpretacija zavisnosti sadržavanja *Nastavnik*[N] $\subseteq$ *Nastava*[N] bi bila da nastavnik ne može pripadati bilo kojoj katedri, ako ne izvodi nastavu iz bar jednog predmeta. $\square$

Analiza semantike, koju nose zavisnosti sadržavanja, sprovedene u primeru 5.71, ukazuje da je zavisnost sadržavanja semantički bogato ograničenje.

$\mathfrak{I}_1$ (trivijalne zavisnosti sadržavanja) $[A_1,..., A_n] \subseteq [A_1,..., A_n]$,

$\mathfrak{I}_2$ (projekcija/permutacija) $[A_1,..., A_n] \subseteq [B_1,..., B_n] \Rightarrow [A_{i_1},..., A_{i_k}] \subseteq [B_{i_1},..., B_{i_k}]$

(za svaki podniz i_1 ,..., i_k različitih celih brojeva iz skupa $\{1,..., n\}$),

$\mathfrak{I}_3$ (tranzitivnost)

$([A_1,..., A_n] \subseteq [B_1,..., B_n]) \wedge ([B_1,..., B_n] \subseteq [C_1,..., C_n]) \Rightarrow [A_1,..., A_n] \subseteq [C_1,..., C_n]$.

Slika 5.49.

Kao za funkcionalne i višeznačne zavisnosti, i za zavisnosti sadržavanja postoji sistem aksioma. Intuitivna jednostavnost tog sistema aksioma, dozvoljava da se on samo navede bez dokazivanja neredundantnosti, neprotivrečnosti i kompletnosti, slika 5.49.

Definicija 5.36. Referencijalni integritet je međurelaciona zavisnost sadržavanja $N_i[A_1,..., A_n] \subseteq N_j[B_1,..., B_n]$, kod koje $\{B_1,..., B_n\}$ predstavlja primarni ključ šeme relacije sa nazivom N_j. □

Primer 5.70. U primeru 5.71 zavisnost sadržavanja $N_3[P] \subseteq N_2[P]$ i $N_3[N] \subseteq N_1[N]$ predstavljaju uslove referencijalnog integriteta. □

5.7. *Pretpostavka o šemi univerzalne relacije*

I bez eksplicitnog naglašavanja, autori mnogih radova iz oblasti relacionih baza podataka polaze od pretpostavke o postojanju šeme univerzalne relacije. Prema ovoj pretpostavci, šemu univerzalna relacije predstavlja dvojka $(\mathcal{U}, C)$, gde je $\mathcal{U}$ univerzalni skup obeležja, a C skup svih ograničenja nad $\mathcal{U}$, a šeme relacija (R_i, C_i) skupa $S = \{(R_i, C_i) \mid i = 1,..., n\}$ šeme baze podataka, dobijene su dekomponovanjem šeme univerzalne relacije tako da važi barem $\mathcal{U} = \bigcup_{i=1}^{n} R_i$.

Prirodna posledica pretpostavke o postojanju šeme univerzalne relacije je i pretpostavka o postojanju univerzalne relacije, kao pojave nad šemom $(\mathcal{U}, C)$. Preciznije, reč je o pretpostavci da postoji skup pojava $SAT(\mathcal{U}, C)$. Prema ovoj pretpostavci, sve relacije $r_1,..., r_n$ baze podataka nad šemom sa skupom šema relacija $S = \{(R_i, C_i) \mid i = 1,..., n\}$ dobijaju se projektovanjem relacije $r \in SAT(\mathcal{U}, C)$ na skupove obeležja R_i,

$$(\forall i \in \{1,..., n\})(r_i = \pi_{R_i}(r)). \tag{5.13}$$

Pretpostavka da su sve relacije baze podataka dobijene projektovanjem neke, hipotetične univerzalne relacije r iz $SAT(\mathcal{U}, C)$ na skupove obeležja $R_1,..., R_n$, vodi i ka zaključku da se primenom operatora prirodnog spajanja na aktuelne relacije $s_1,..., s_n$ nad šemama relacija $(R_1, C_1),..., (R_n, C_n)$, mora dobiti neka, eventualno druga, relacija r' iz $SAT(\mathcal{U}, C)$ i da relacija r' treba da bude takva da važi $(\forall i \in \{1,..., n\})(s_i = \pi_{R_i}(r'))$. Treba

naglasiti da bazu podataka predstavlja skup relacija $\{s_i | i = 1,..., n\}$, koje se, u postupku ažuriranja baze podataka, menjaju, a ne univerzalna relacija r. Ova razmatranja vode ka definisanju pretpostavke o čistoj univerzalnoj relaciji.

Definicija 5.37. Neka je $\{s_1,..., s_n\}$ jedno stanje baze podataka nad šemom $(S, \mathcal{I})$, gde je $S = \{(R_i, C_i) | i = 1,..., n\}$, a $(\mathcal{U}, C)$ šema univerzalne relacije, takva da važi $\bowtie(R_1,..., R_n) \in C$. Skup relacija $\{s_1,..., s_n\}$ zadovoljava pretpostavku o ***čistoj univerzalnoj relaciji***, ako važi

$$(5.14) \qquad (\exists r \in SAT(\mathcal{U}, C))((r = \overset{n}{\underset{i=1}{\bowtie}} s_i) \wedge (\forall i \in \{1,..., n\})(s_i = \pi_{R_i}(r))). \square$$

Primer 5.71. Neka je $\mathcal{U} = \{A, B, C\}$, $S = \{(\{A, B\}, \{A\}), (\{B, C\}, \{B\})\}$ i $C = \{A \rightarrow B, B \rightarrow C, \bowtie(AB, BC)\}$. Na slici 5.50 je prikazana jedna univerzalna relacija $r \in SAT(\mathcal{U}, C)$ i jedno stanje baze podataka nad šemom $(S, \varnothing)$, koje zadovoljava pretpostavku o čistoj univerzalnoj relaciji. Na slici 5.51 je prikazano jedno stanje baze podataka nad istim skupom šema relacija, koje ne zadovoljava pretpostavku o čistoj univerzalnoj relaciji, a na slici 5.52 ponovo jedno stanje koje pretpostavku zadovoljava. □

$r(ABC) =$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1

$r(AB) =$

A	B
a_1	b_1
a_2	b_1

$r(BC) =$

B	C
b_1	c_1

Slika 5.50.

$r(AB) =$

A	B
a_1	b_1
a_2	b_1

$r(BC) =$

B	C
b_1	c_1
b_2	c_2

Slika 5.51.

$r(AB) =$

A	B
a_1	b_1
a_2	b_1
a_3	b_2

$r(BC) =$

B	C
b_1	c_1
b_2	c_2

Slika 5.52.

Treba još jednom istaći da su šema univerzalne relacije i univerzalna relacija pretpotavke, neophodne za razvoj teorije relacionih baza podataka. Univerzalna relacija se, u praksi, fizički ne realizuje. Baza podataka, u opštem slučaju, sadrži više relacija, a ne jednu, unverzalnu relaciju.

U daljem tekstu je ukazano na neke od posledica pretpostavke o postojanju šeme univerzalne relacije i njene pojave.

5.7.1. Jedinstvena uloga obeležja

Jednu od posledica pretpostavke o šemi univerzalne relacije predstavlja pojam jedinstvene uloge obeležja. Naime, ako sva obeležja šeme baze podataka treba da se nađu u jednoj šemi relacije - šemi univerzalne relacije, tada svako obeležje može imati samo jednu ulogu, jedno značenje. Inače se ne može konstruisati univerzalna relacija, koja zadovoljava uslov (5.14).

Primer 5.72. Posmatra se skup šema relacija

$S = \{Zaposlenje(\{MBR, DAT\}, \{MBR\}), Radnik(\{MBR, IME, DAT\}, \{MBR\})\}$.

Obeležje *DAT*, u šemi relacije *Zaposlenje*, ima ulogu datuma zaposlenja radnika, a u šemi relacije *Radnik*, ulogu datuma rođenja. Na slici 5.53 je prikazana jedna pojava nad skupom S.

$r(Zaposlenje) =$

MBR	*DAT*
m_1	d_1
m_2	d_2
m_3	d_1

$r(Radnik) =$

MBR	*IME*	*DAT*
m_1	i_1	d_3
m_2	i_2	d_4
m_3	i_3	d_5

Slika 5.53.

Pošto je $\{MBR, DAT\} \cap \{MBR, IME, DAT\} = \{MBR, DAT\}$, primena operatora prirodnog spoja na relacije $r(Zaposlenje)$ i $r(Radnik)$ rezultuje u praznoj relaciji, tako da prikazana pojava baze podataka ne zadovoljava uslov (5.14).

Na slici 5.54 je prikazana relacija nad skupom obeležja $\{MBR, IME, DAT\}$, koja sadrži uniju torki relacija $r(Zaposlenje)$ i $r(Radnik)$. Ne ulazeći u razmatranje korektnosti konstruisanja ove relacije, može se konstatovati da ona ne zadovoljava uslov (5.14), jer:

- niti zadovoljava funkcionalnu zavisnost $MBR \rightarrow DAT$,

MBR	*IME*	*DAT*
m_1	i_1	d_1
m_1	i_1	d_3
m_2	i_2	d_2
m_2	i_2	d_4
m_3	i_3	d_2
m_3	i_3	d_5

Slika 5.54.

- niti relacije sa slike 5.53 predstavljaju njene projekcije po odgovarajućim skupovima obeležja.

Problem je posledica činjenice da obeležje *DAT* ima dve uloge. Da su upotrebljena dva obeležja, *DAZ*, kao datum zaposlenja i *DAR*, kao datum rođenja, tada bi šema univerzalna relacije bila ({*MBR*, *IME*, *DAZ*, *DAR*}, {*MBR*}), a univerzalna pojava bi se mogla konstruisati prirodnim spajanjem relacija nad skupovima obeležja {*MBR*, *DAZ*} i {*MBR*, *IME*, *DAR*}. □

Primer 5.73. Model sastavnice se u relacionom modelu podataka rešava sledećim skupom šema relacija

$$\{Proizvod(\{IBP, POP\}, \{IBP\}), Sastavnica(\{IBP, IBK, KOL\}, \{IBP \to IBK\})\},$$

gde je *IBP* identifikacioni broj proizvoda, *POP* podaci o proizvodu, *IBK* identifikacioni broj komponente, a *KOL* količina, odnosno broj komada posmatrane komponente, koja se ugrađuje u prizvod. Međutim, jedan proizvod se može ugraditi, kao komponenta, u druge proizvode. S druge strane, svaka komponenta je neki proizvod. Obeležja *IBP* i *IBK* su uvedena da bi se razdvojile ove dve uloge proizvoda. Pri tome važi $dom(IBK) \subseteq dom(IBP)$. Na slici 5.55 su prikazane pojave nad šemama relacija *Proizvod* i *Sastavnica*. □

$r(Proizvod) =$

IBP	*POP*
i_1	p_1
i_2	p_2
i_3	p_3
i_4	p_4

$r(Sastavnica) =$

IBP	*IBK*	*KOL*
i_1	i_2	k_1
i_1	i_3	k_2
i_2	i_3	k_3
i_2	i_4	k_4

Slika 5.55.

Pri definisanju zavisnosti spoja putem predikata, uvodi se ograničenje da se predikat $\mathcal{P}(X, Y, Z)$ ne može rastaviti na predikate $\mathcal{R}(X, Y)$ i $\mathcal{Q}(Y, Z)$. Ovo ograničenje se može posmatrati i u svetlu pretpostavke o postojanju šeme univerzalne relacije, odnosno u svetlu uslova da svako obeležje sme imati samo jednu ulogu.

Primer 5.74. Neka je *I* isporučilac, *D* deo, a *P* proizvod. Neka je predikat $\mathcal{P}(I, D, P)$ definisan sa

- *i* isporučuje *d* za ugradnju u *p*,

predikat $\mathcal{R}(I, D)$ definisan sa

- *i* isporučuje *d*,

a predikat $\mathcal{Q}(D, P)$ definisan sa

- *d* se može ugraditi u *p*.

Zamena predikata $\mathcal{P}(I, D, P)$ sa $\mathcal{R}(I, D) \wedge Q(D, P)$ bi značila da obeležje D ima dve uloge. To su:

- deo d, koji isporučilac uopšte može da isporuči i
- deo d, koji bi se mogao ugraditi u proizvod p.

Tek presek skupa delova, koji se mogu isporučiti i skupa delova, koji se mogu ugraditi, daje skup delova, koji se aktuelno isporučuje i ugrađuje, što i jeste uloga obeležja D u predikatu $\mathcal{P}(I, D, P)$. □

Pošto, saglasno pretpostavci o postojanju šeme univerzalne relacije, šeme relacija jedne šeme baze podataka, sadrže različite skupove obeležja, poseban naziv N šeme relacije sa skupom obeležja $\{X, Y, Z\}$ postaje nepotreban. Umesto naziva N može se koristiti notacija XYZ kao identifikator šeme relacije. Ovaj zaključak, mada logički ispravan, ne vodi računa o činjenici da naziv pretstavlja komponentu semantike šeme relacije.

U praksi se, ne retko, jedinstvena uloga obeležja ne poštuje, tako da se sreću rešenja sa obeležjima homonimima. U različitim šemama relacija se javljaju obeležja sa istim nazivom, ali različitim ulogama. Tada se naziv šeme relacije koristi kao prefiks nazivu obeležja u cilju razdvajanja različitih uloga. De fakto, korišćenjem naziva šeme relacije kao prefiksa, zadovoljava se pretpostavka o postojanju šeme univerzalne relacije.

Posledicu pretpostavke o šemi univerzalne relacije predstavlja i činjenica da svakom obeležju može biti pridružen samo jedan domen.

5.7.2. *Izvođenje funkcionalnih zavisnosti*

Armstrongove aksiome služe za sintaktičko izvođenje zaključaka o posledicama zadatog skupa funkcionalnih zavisnosti. Međutim, funkcionalne zavisnosti poseduju i semantičku komponentu.

Pretpostavka o postojanju šeme univerzalne relacije ima značajan uticaj na semantiku funkcionalnih zavisnosti. Ova pretpostavka garantuje da između dva skupa obeležja X i Y može postojati najviše jedna funkcionalna zavisnost od X ka Y. Drugim rečima, za dati skup funkcionalnih zavisnosti $\mathcal{F}$ i funkcionalnu zavisnost $X \rightarrow Y$, ili važi $X \rightarrow Y \notin \mathcal{F}$ ili u $\mathcal{F}$ postoji samo jedna funkcionalna zavisnost $X \rightarrow Y$. Nad istim skupom obeležja XY ne mogu postojati dve različite funkcionalne zavisnosti, na primer $f: X \rightarrow Y$ i $g: X \rightarrow Y$.

Primer 5.75. Date su funkcionalne zavisnosti f_1: *IDBRMA*→*NAZIV* i f_2: *NAZIV*→*BRRAD*. Semantika f_1 je da identifikacioni broj materijala *IDBRMA* određuje naziv materijala *NAZIV*. Semantika f_2 je da naziv organizacione jedinice *NAZIV* određuje broj zaposlenih radnika u organizacionoj jedinici. Na osnovu pravila izvođenja $\mathfrak{I}_3$ sledi da važi f_3: *IDBRMA*→*BRRAD*, što je očigledno besmisleno. Do pogrešnog zaključka se došlo zbog nepoštovanja pretpostavke o šemi univerzalne relacije. Obeležje *NAZIV*, očigledno, ima dve uloge. □

Primer 5.76. Neka je f_1 : $SEK \rightarrow RUK$, gde je SEK sektor, a RUK rukovodilac, i neka je f_2 : $RUK + SPR \rightarrow BRD$, gde je SPR sprat zgrade, a BRD broj radnika. Semantika ovih funkcionalnih zavisnosti je da svaki sektor ima samo jednog rukovodioca (f_1) i da jedan rukovodilac, na određenom spratu zgrade, rukovodi određenim brojem radnika (f_2). Primenom pravila izvođenja $\mathbb{I}_3$, dobija se f_3 : $SEK + SPR \rightarrow BRD$, kojom se određuje broj onih radnika na određenom spratu, kojima rukovodi rukovodilac određenog sektora. Ako jedan rukovodilac upravlja sa više sektora, tada f_3 nije semantički jednaka sintaksno identičnoj funkcionalnoj zavisnosti g : $SEK + SPR \rightarrow BRD$, koja određuje broj radnika na spratu iz određenog sektora, jer sada na jednom spratu mogu raditi radnici iz više sektora, a da imaju istog rukovodioca. Da bi se f_3 i g razlikovale ne samo po semantici, već i po sintaksi, potrebno je izvršiti preimenovanje obeležja u f_2 i g. Na primer, f_2 : $RUK + SPR \rightarrow BRR$, gde je BRR broj radnika na spratu, kojima upravlja rukovodilac sektora i g : $SEK + SPR \rightarrow BRS$, gde je BRS broj radnika sektora na spratu.

Na slici 5.56 su prikazane relacije nad skupovima obeležja $\{RUK, SEK\}$, $\{BRR, RUK, SPR\}$, $\{BRR, SEK, SPR\}$ i $\{BRS, SEK, SPR\}$. Ove relacije zadovoljavaju, redom, sledeće funkcionalne zavisnosti: $SEK \rightarrow RUK$, $RUK + SPR \rightarrow BRR$, $SEK + SPR \rightarrow BRR$ i $SEK + SPR \rightarrow BRS$. Sa slike se da lako utvrditi da rukovodilac r_1 upravlja sektorima s_1 i s_2, da pod rukovodstvom r_1 na spratu p_1 radi *15*, a na spratu p_2 *4* radnika, kao i da na spratu p_1 radi *10* radnika iz sektora s_1 i *5* radnika iz sektora s_2. Takođe, relacija $r(BRR, SEK, SPR)$ ukazuje da je semantika funkcionalne zavisnosti $SEK + SPR \rightarrow BRR$ "sektor i sprat određuju broj radnika na spratu, kojima upravlja rukovodilac sektora". □

$r(RUK, SEK) =$

RUK	SEK
r_1	s_1
r_1	s_2
r_2	s_3

$r(BRR, RUK, SPR) =$

BRR	RUK	SPR
15	r_1	p_1
04	r_1	p_2
12	r_2	p_2

$r(BRR, SEK, SPR) =$

BRR	SEK	SPR
15	s_1	p_1
04	s_1	p_2
15	s_2	p_1
04	s_2	p_2
12	s_3	p_2

$r(BRS, SEK, SPR) =$

BRS	SEK	SPR
10	s_1	p_1
05	s_2	p_1
02	s_1	p_2
02	s_2	p_2
12	s_3	p_2

Slika 5.56.

U analiziranim primerima sintaksna izvođenja su bila pogrešna zbog činjenice da je jedno obeležje imalo više uloga. Problem je rešavan preimenovanjem obeležja. To preimenovanje pomera poznavanje semantike funkcionalnih zavisnosti na nivo sintakse, u cilju dobijanja ispravnih rezultata pri primeni pravila izvođenja.

5.8. Nula vrednosti

U svim dosadašnjim razmatranjima se polazilo od pretpostavke da su torke relacije kompletne, odnosno da svaka torka pojave r nad šemom relacije (R, C), za svako obeležje A iz R, sadrži vrednost iz domena obeležja A. U realnim bazama podataka se, često, javlja potreba za upisom nove torke u neku relaciju, mada su vrednosti nekih (ne svih) obeležja, za tu torku još uvek nepoznate. Međutim, relacija se definiše kao skup R - vrednosti, a R - vrednost kao funkcija koja svakom obeležju A iz R pridružuje jednu vrednost iz $dom(A)$. To dalje znači da bi upis torke sa nepoznatim vrednostima za neka obeležja, smeo da se izvrši samo ako ta nepoznata vrednost pripada domenu svakog od tih obeležja.

Da bi se prevazišao opisani problem, uvodi se pojam ***nula vrednosti***. Odmah treba naglasiti da tu nije reč o broju nula. Postoje tri osnovne kategorije nula vrednosti. To su:

- postojeća, ali nepoznata vrednost (vrednost postoji, ali se još ne zna koja je),
- nepostojeća vrednost (javlja se kada za neki entitet određeno obeležje predstavlja neprimenljivo svojstvo),
- neinformativna nula (ne zna se da li vrednost uopšte postoji ili ne).

Primer 5.77. Posmatra se šema relacije *Radnik*({*MBR, IME, PRZ, GOD, ZAN, DKL, KTL*}, {*MBR*}), gde mnemonici većine obeležja imaju uobičajena značenja, osim *DKL* (daktilografska klasa) i *KTL* (broj kućnog telefona).

Pošto svaki radnik ima godinu rođenja, upis torke sa nepoznatom vrednošću za obeležje *GOD*, ukazuje da je reč o nula vrednosti tipa "postojeća, ali nepoznata vrednost".

Obeležja *ZAN* i *DKL* su, na određeni način povezana. Za sve vrednosti iz $dom(ZAN)$, osim za "*daktilograf*", obeležje *DKL* predstavlja neprimenljivo svojstvo. Nepoznata vrednost za obeležje *DKL* se može interpretirati kao nula vrednost tipa "nepostojeća vrednost" i sadržaće je torke svih onih radnika, koji nisu daktilografi.

Obeležje *KTL* može poslužiti za ilustraciju pojma neinformativne nule. Pošto neki radnici mogu odbiti da daju broj svog kućnog telefona, mada ga imaju, nepoznata vrednost obeležja *KTL* ne nosi preciznu informaciju o statusu radnikovog kućnog telefona. Naime, može biti reč o postojećoj, ali nepoznatoj vrednosti, a može biti reč i o nepostojećoj vrednosti. □

Za označavanje nula vrednosti se uvodi ili jedan specijalan simbol, na primer ω, u domene svih obeležja, koja mogu uzimati nula vrednosti, ili se uvode različito označeni simboli ω_i, za $i = 1,2,...$ U ovom poslednjem slučaju, svakoj nepoznatoj vrednosti, svakog obeležja, pridružuje se druga nula vrednost. Različite nula vrednosti se nazivaju označenim nulama.

Primer 5.78. Na slici 5.57 je prikazana tabela sa različito označenim nula vrednostima, koja ilustruje razmatranja iz primera 5.79. □

MBR	*IME*	*PRZ*	*GOD*	*ZAN*	*DKL*	*KTL*
159	*Ivo*	*Ban*	*1940*	*ekon*	ω_1	*36-41*
013	*Ana*	*Tot*	*1960*	*prav*	ω_2	ω_3
009	*Eva*	*Pap*	*1968*	*dakt*	*I*	*51-48*
201	*Aca*	*Kon*	ω_4	*elek*	ω_5	*20-33*
011	*Iva*	*Car*	ω_6	*dakt*	*II*	ω_7

Slika 5.57.

Uvođenje nula vrednosti u relacioni model podataka utiče na:

- definisanje upita,
- izvođenje ažuriranja,
- definisanje ograničenja i
- pretpostavku o postojanju univerzalne relacije.

Uticaj nula vrednosti na pretpostavku o postojanju univerzalne relacije opisan je u Prilogu 1 knjige Principi projektovanja baza podataka.

Primer 5.79. Većina upita u bazu podataka, definisana bilo prirodnim jezikom ili putem nekog upitnog programskog jezika, mora se redefinisati da bi dali očekivani odgovor i u slučaju kada u relaciji postoje torke sa nula vrednostima. Na primer, upit "Prikaži sve radnike, rođene *1950* godine", mora se preformulisati u "Prikaži sve radnike, koji su sigurno rođeni *1950* godine". □

5.8.1. Ograničenja u bazi podataka sa nula vrednostima

Uvođenje nula vrednosti u bazu podataka dovodi do potrebe da se definišu i nova ograničenja. Najjednostavnije od tih ograničenja je da se dozvoli, ili ne dozvoli, upis nula vrednosti u kolone pojedinih obeležja relacione tabele. Ovo ograničenje se realizuje u okviru *SQL* naredbe *CREATE TABLE*.

U daljem tekstu je nešto više pažnje posvećeno ograničenjima, pod nazivima:

- integritet entiteta i
- egzistencijalno ograničenje.

Definicija 5.38. Relacija r nad šemom relacije $(R, \mathcal{K})$ zadovoljava uslov ***integriteta entiteta***, ako su ispunjena sledeća dva uslova:

$$(\forall X \in \mathcal{K})(\forall t_i, t_j \in r)(t_i[X] = t_j[X] \Rightarrow t_i = t_j) \text{ i}$$

$$(\forall X \in \mathcal{K})(\forall A \in X)(\forall t \in r)(t[A] \neq \omega). \ \square$$

Integritet entiteta je vezan za pojam ključa šeme relacije i ima dve komponente. To su:

- jedinstvenost vrednosti ključa i

- zahtev da niti kompletan ključ niti bilo koje obeležje ključa, ne sme imati nepoznatu vrednost u pojavi nad šemom relacije.

Obe komponente integriteta entiteta su posledica činjenice da vrednosti ključa šeme relacije treba da obezbede jednoznačnu identifikaciju torki u pojavi nad šemom relacije. Taj zadatak se ne bi mogao izvršiti, kada bi ključ sadržao nula vrednost. Naime, ako se ta nula vrednost interpretira kao postojeća, ali nepoznata vrednost, a to bi bila jedino ispravna interpretacija za nula vrednosti obeležja ključa, to bi značilo i da ta nula vrednost može biti jednaka svakoj ne nula vrednosti iz domena obeležja, čime bi se izgubila mogućnost jednoznačne identifikacije torki.

Posledica uslova integriteta entiteta je i da se vrednost ključa ne može, pri ažuriranju, modifikovati.

U literaturi se, neki put navodi, da samo primarni ključ šeme relacije treba da zadovolji uslov integriteta entiteta. To je posledica činjenice da se primena uslova integriteta entiteta na sve ekvivalentne ključeve šeme relacije, u praksi pokazuje suviše strogim ograničenjem. Međutim, rešenje te dileme pre treba tražiti u pažljivom definisanju ključeva šema relacije, jer ključ sa nula vrednostima nije ključ. Naime, ako se koristi jedna, neoznačena nula vrednost, takav ključ gubi osobinu jedinstvenosti. Ako se koriste označene nula vrednosti, prividno se obezbeđuje osobina jedinstvenosti. Međutim, takva jedinstvena nula vrednost je nepoznata korisniku, te se ne može koristiti za traženje torki u bazi podataka.

Definicija 5.39. Egzistencijalno ograničenje je iskaz oblika $X \varepsilon Y$, sa semantikom X zahteva Y, koje ukazuje da, za X, $Y \subseteq \mathcal{U}$, u relaciji r nad $\mathcal{U}$, važi

$$(\forall t \in r)((\forall A \in X)(t[A] \neq \omega) \Rightarrow (\forall A \in Y)(t[A] \neq \omega)). \square$$

Egzistencijalno ograničenje se često definiše i kao nešto stroži, međurelacioni uslov integriteta.

Definicija 5.40. Neka su (R_1, C_1) i (R_2, C_2) šeme relacija, r_1 i r_2 pojave nad tim šemama relacija, $X = \{B_1,..., B_n\} \subseteq R_1$ i $Y = \{A_1,..., A_n\} \subseteq R_2$, a $R_2[A_1,..., A_n] \subseteq R_1[B_1,..., B_n]$ uslov referencijalnog integriteta. Relacije r_1 i r_2 zadovoljavaju uslov egzistencijalnog ograničenja, u oznaci $R_1[X] \varepsilon R_2[Y]$, ako važi

$$\pi_X(r_1) = \pi_Y(r_2). \square$$

Uslov međurelacionog egzistencijalnog ograničenja ne dozvoljava pojavu nula vrednosti u bilo kojoj komponenti skupova obeležja X i Y, jer je kombinovan sa uslovom referencijalnog integriteta. Uslov referencijalnog integriteta, s druge strane, je zasnovan na pojmu ključa šeme relacije, odnosno pojmu integriteta entiteta.

Međurelaciono egzistencijalno ograničenje se može izraziti i putem dve zavisnosti sadržavanja, od kojih jedna predstavlja uslov referencijalnog integriteta.

Primer 5.80. Date su dve šeme relacije $(\{A, B\}, \{A\})$ i $(\{B, C\}, \{B\})$. Na slici 5.58 su prikazane dve pojave nad ovim šemama relacija, koje zadovoljavaju međurelaciono egzistencijalno ograničenje $BC[B] \varepsilon AB[B]$, dok su na slici 5.59 prikazane pojave nad ovim šemama relacija, koje ne zadovoljavaju ovo egzistencijalno ograničenje. Ograničenje je

narušeno iz dva razloga. Pojava $r(AB)$ na slici 5.59 sadrži torku (a_5, ω) sa nula vrenošću i ne sadrži toku sa $B = b_4$ vrednošću.

Pojave na slikama 5.58 i 5.59 zadovoljavaju uslov integriteta entiteta. Uslov integriteta entiteta bi bio narušen, ako bi se u relaciji $r(AB)$ upisala torka (ω, b_1). □

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_4	b_3

B	C
b_1	c_1
b_2	c_2
b_3	c_2

Slika 5.58.

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_4	b_3
a_5	ω

B	C
b_1	c_1
b_2	c_2
b_3	c_2
b_4	c_3

Slika 5.59.

5.8.2. *Funkcionalne zavisnosti i nula vrednosti*

Najvažnija primena funkcionalnih zavisnosti u bazama podataka sa nula vrednostima tipa postojeća, ali nepoznata vrednost, je u eliminaciji nula vrednosti. Funkcionalne zavisnosti se koriste za takozvano *popunjavanje* nula. Popunjavanje nula se realizuje prilikom modifikacije torki sa nula vrednostima.

Neka je $(R, \mathcal{K})$ šema relacije sa skupom ključeva $\mathcal{K}$, a r jedna njena pojava. Neka, dalje, za neko obeležje A iz R, takvo da je $(\forall X \in \mathcal{K})(A \notin X)$, važi $(\exists t \in r)(t[A] = \omega_i)$, gde je ω_i označena nula vrednost. Modifikacija torke t, pri kojoj se nula vrednost ω_i zamenjuje nekom vrednošću a iz $dom(A)$, koncepcijski se realizuje u tri koraka. To su:

- torka u, takva da je $t[R \setminus \{A\}] = u[R \setminus \{A\}]$ i $u[A] = a$, se upisuje u pojavu r,
- na osnovu funkcionalne zavisnosti $X \rightarrow A$ (X je ključ šeme relacije R) se zaključuje da je $t[\omega_i] = a$, te se $t[\omega_i]$ zamenjuje sa a,
- iz r se eliminiše ili t ili u.

Definicija 5.41. Neka je (R, C) šema relacije, a $X \rightarrow A \in C$. Pravila za popunjavanje označenih nula vrednosti u torkama t_1 i t_2 relacije $r(R)$ takvim, da je $t_1[X] = t_2[X]$, su:

- ako je $t_1[A] = \omega_i$, a $t_2[A] \in dom(A)$, ω_i se zamenjuje sa $t_2[A]$,
- ako je $t_1[A] = \omega_i$, $t_2[A] = \omega_j$, $i < j$, vrši se zamena ω_j sa ω_i. □

Repetitivna primena pravila za popunjavanje označenih nula vrednosti je veoma slična implikacionom algoritmu za tabloe. Vrednosti iz domena obeležja odgovaraju karakterističnim promenljivama, a nula vrednosti odgovaraju nekarakterističnim promenljivama.

6. Glava

Operacijska komponenta relacionog modela podataka

Operacijska (operaciona) komponenta modela podataka služi za opis dinamičkih karakteristika realnog sistema, za razliku od strukturalne i integritetne komponente, koje služe za predstavljanje statičke strukture realnog sistema, iskazane putem šeme baze podataka. Operacijskom komponentom modela podataka se definiše jezik za izražavanje upita nad bazom podataka i jezik putem kojeg se vrši ažuriranje pojave baze podataka u cilju njenog usaglašavanja s tekućim stanjem realnog sistema. Zbog toga se, često, operacijska komponenta naziva ***dinamičkom*** komponentom modela podataka. Operacijska komponenta sadrži tri potkomponente:

- ***upitni jezik***,
- ***jezik za ažuriranje podataka*** i
- ***jezik za definiciju podataka***.

Upitni jezik i jezik za ažuriranje podataka se jednim imenom nazivaju ***jezik za manipulaciju podacima***[*]. Pomoću upitnog jezika i jezika za ažuriranje podataka se, u praksi, realizuju informacioni zahtevi svih korisnika informacionog sistema, tako da se ovi jezici pojavljuju u formi:

- nezavisnog, interaktivnog jezika,
- jezika, ugrađenog u jezik treće generacije, ili
- jezika, ugrađenog u jezik (odnosno, alat) četvrte generacije.

---

[*] U literaturi se, ponekad, samo jezik za ažuriranje podataka naziva jezik za manipulaciju podacima (Data Manipulation Language).

Jezik za definiciju podataka služi za realizaciju implementacione šeme baze podataka.

U nastavku ove glave će biti dat prikaz operacijske komponente relacionog modela podataka. Za razliku od mrežnih i hijerarhijskih sistema za upravljanje bazama podataka, operacijsku komponentu relacionog sistema za upravljanje bazama podataka najčešće u praksi reprezentuje jedan ***jedinstveni*** jezik podataka, ***visokog nivoa deklarativnosti***. Visoki nivo deklarativnosti jezika podataka podrazumeva da korisnik takvog jezika na deklarativan način iskazuje ono ***šta*** želi, ***bez specificiranja načina*** izvršavanja željene akcije.

Tačke 6.1. i 6.2. sadrže, redom, prikaz relacione algebre i relacionog računa - teoretskih modela upitnih (manipulacionih) jezika relacionih baza podataka. Tačka 6.3. se bavi pitanjem ekvivalentnosti relacione algebre i relacionog računa sa stanovišta ekspresivne moći, dok tačka 6.4. obrađuje problem ažuriranja baze podataka i transakcione obrade podataka. Tačka 6.5. daje prikaz široko upotrebljavanog jezika relacionih sistema za upravljanje bazama podataka SQL.

6.1. Relaciona algebra

Relaciona algebra predstavlja jedan teoretski model upitnog jezika relacionih baza podataka. Relaciona algebra je ***skupovno*** orijentisani, ***visoko deklarativni***, upitni jezik. U osnovi relacione algebre leži matematička teorija skupova. Izrazi relacione algebre, što će u ovoj tački biti precizno definisano, sastoje se od operatora i operanda relacione algebre, pri čemu su operandi skupovi torki, ili reprezentanti skupova torki, a rezultat izraza relacione algebre je, takođe, nekakav skup torki, odnosno reprezentant skupa torki.

Sledeći primer daje prikaz jednog izraza relacione algebre i objašnjava način njegovog formiranja, u cilju sticanja globalne predstave o relacionoj algebri. Svi operatori koji se u primeru koriste, u okviru ove tačke se, zatim, i formalno definišu.

Primer 6.1. Neka je data je šema baze podataka $\mathcal{S} = (S, \mathcal{I})$, pri čemu je skup šema relacija: $S = \{Radnik(\{MBR, IME, PRZ, PLT, SEF\}, \{MBR\}), Projekat(\{SPR, NAP, RUK\}, \{SPR\}), Radproj(\{MBR, SPR, BRC\}, \{MBR + SPR\})\}$, a skup međurelacionih ograničenja: $\mathcal{I} = \{Radproj[MBR] \subseteq Radnik[MBR], Radproj[SPR] \subseteq Projekat[SPR], Projekat[RUK] \subseteq Radnik[MBR], Radnik[SEF] \subseteq Radnik[MBR]\}$. Obeležja šema relacija imaju sledeća značenja: *MBR* - matični broj radnika, *IME* - ime radnika, *PRZ* - prezime radnika, *PLT* - plata radnika, *SEF* - matični broj direktno nadređenog rukovodioca radnika (šefa), *SPR* - šifra projekta, *NAP* - naziv projekta, *RUK* - rukovodilac projekta i *BRC* - broj časova mesečnog angažovanja radnika na projektu. Neka je $rbp = \{radnik, projekat, radproj\}$ baza podataka nad šemom $\mathcal{S}$. Upit:

U0. prikazati nazive projekata, kao i šifre, imena i prezimena rukovodilaca odgovarajućih projekata, na kojima radi radnik s matičnim brojem $MBR = 10$

se realizuje putem sledećeg izraza relacione algebre:

$$\pi_{NAP+RUK+PRZ+IME}((projekat \triangleright\!\triangleleft \pi_{SPR}(\sigma_{MBR=10}(radproj)))\ [RUK = MBR]\ radnik).$$

Ovaj izraz se izračunava na sledeći način:

1. Izračunava se podizraz $E_1 = \sigma_{MBR = 10}$ (*radproj*), kojim se iz relacije *radproj* izdvajaju (selekcioniraju) samo takve torke t_i, za koje važi da je $t_i(MBR) = 10$ i od njih formira nova relacija r_1. Rezultat podizraza E_1 je, dakle, relacija r_1 nad skupom obeležja $\{MBR, SPR, BRC\}$, što se označava kao $r_1(MBR, SPR, BRC)$.
2. Izračunava se podizraz $E_2 = \pi_{SPR}(E_1)$ na taj način što se vrši restrikcija svih torki t_i iz relacije r_1 na obeležje SPR i od restrikicija $t_i[SPR]$ formira nova relacija $r_2(SPR)$.
3. Izračunava se podizraz E_3 = *projekat* $\triangleright\!\triangleleft$ E_2, tako što se vrši spajanje svih torki iz relacije *projekat* sa svim torkama iz relacije r_2, za koje važi da imaju iste vrednosti nad zajedničkim obeležjem SPR. Kao rezultat će se dobiti relacija $r_3(SPR, NAP, RUK)$, koja će sadržati sve i samo one torke iz tabele *projekat*, za koje važi da u relaciji r_2 postoji odgovarajuća torka sa istom vrednošću nad obeležjem SPR.
4. Izračunava se podizraz $E_4 = E_3$ [$RUK = MBR$] *radnik*, na taj način što se vrši spajanje svih torki t_i iz relacije r_3 sa svim torkama t_j iz relacije *radnik*, za koje važi da su im vrednosti nad obeležjem RUK, tj. MBR iste: $t_i(RUK) = t_j(MBR)$. Pri tome se dobija relacija $r_4(SPR, NAP, RUK, MBR, IME, PRZ, PLT, SEF)$.
5. Konačno, izraz relacione algebre $E = \pi_{NAP + RUK + PRZ + IME}(E_4)$ se izračunava tako što se izvrši restrikcija svih torki t_i iz relacije r_4 na skup obeležja $\{NAP, RUK, PRZ, IME\}$ i od restrikcija $t_i[NAP + RUK + PRZ + IME]$ formira relacija $r(NAP, RUK, PRZ, IME)$, koja predstavlja rezultat navedenog izraza relacione algebre, odnosno odgovor na upit U0. □

6.1.1. *Iskazivanje upita putem relacione algebre*

Za iskazivanje upita putem relacione algebre se koriste dve vrste operatora: standardni operatori matematičke teorije skupova (unija, presek, razlika) i specijalni operatori relacione algebre. U nastavku će biti definisati svi operatori relacione algebre i pojam ekstenzionog izraza relacione algebre.

Definicija 6.1. Date su relacije $r(R)$ i $s(R)$, pri čemu je $R \subseteq \mathcal{U}$, gde je $\mathcal{U}$ univerzalni skup obeležja. ***Unija***, ***presek*** i ***razlika*** relacija r i s su, redom, binarne operacije $\cup$, $\cap$, $-$: $SAT(R) \times SAT(R) \rightarrow SAT(R)$, definisane na sledeći način:

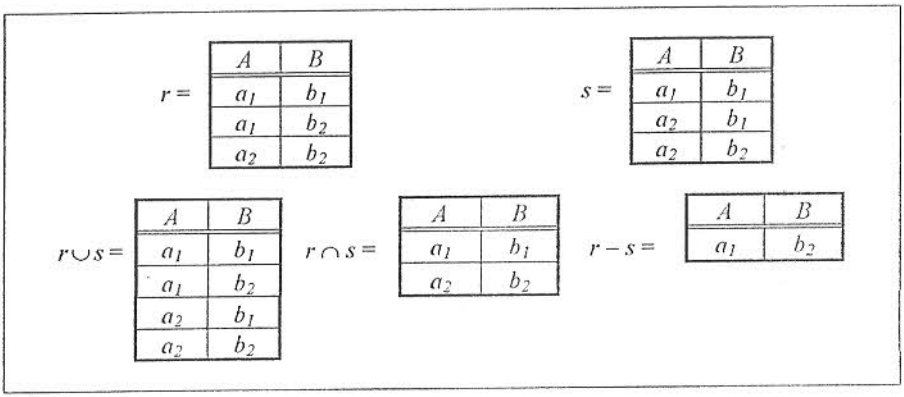
$$r(R) \cup s(R) = \{t \in Tuple(R) \mid t \in r \vee t \in s\},$$

$$r(R) \cap s(R) = \{t \in Tuple(R) \mid t \in r \wedge t \in s\} \text{ i}$$

$$r(R) - s(R) = \{t \in Tuple(R) \mid t \in r \wedge t \notin s\},$$

gde $SAT(R)$ predstavlja ***skup svih relacija***, a $Tuple(R)$ predstavlja ***skup svih torki*** nad skupom obeležja R. □

Primer 6.2. Na slici 6.1 su prikazane dve relacije r i s, kao i rezultat primene operatora unije, preseka i razlike na te dve relacije. □

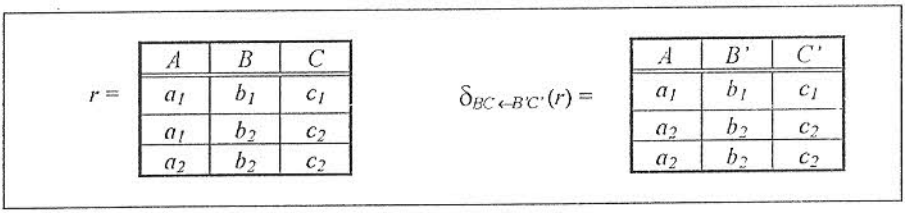


Slika 6.1.

Definicija 6.2. Data je relacija $r(R)$, $R \subseteq \mathcal{U}$, skupovi obeležja $X \subseteq R$, $Y \subseteq \mathcal{U} \setminus R$, funkcija dom: $\mathcal{U} \to \mathcal{O}_S$, koja svakom obeležju $A \in \mathcal{U}$ pridružuje domen $dom(A) \in \mathcal{O}_S$ i bijekcija h: $X \to Y$, takva da je $(\forall A \in X)(dom(h(A)) \subseteq dom(A))$. ***Preimenovanje obeležja*** nad relacijom r je funkcija $\delta_{X \leftarrow Y} : SAT(R) \to SAT((R \setminus X)Y)$, definisana na sledeći način:

$$\delta_{X \leftarrow Y}(r(R)) = \{t \in Tuple((R \setminus X)Y) \mid (\exists t' \in r)(t[R \setminus X] = t'[R \setminus X] \wedge (\forall A \in X)(t(h(A)) = t'(A)))\}. \square$$

Primer 6.3. Na slici 6.2 je prikazana relacija r i rezultat primene operatora preimenovanja $\delta_{BC \leftarrow B'C'}(r)$, pri čemu je $h(B) = B'$, $h(C) = C'$, $dom(B') \subseteq dom(B)$ i $dom(C') \subseteq dom(C)$. $\square$



Slika 6.2.

Iz definicije 6.1 je vidljivo da operatori unije, preseka, ili razlike mogu da se primene samo nad relacijama koje su definisane nad istim skupom obeležja. Takve relacije se nazivaju ***unijski kompatibilnim*** relacijama. Jedna od primena unarnog operatora preimenovanja jeste dovođenje, ukoliko je to moguće, unijski nekompatibilnih relacija u stanje unijske kompatibilnosti, u cilju omogućavanja primene operatora $\cup$, $\cap$, ili $-$.

Da bi se mogao definisati unarni operator selektovanja torki iz relacije, potrebno je prvo definisati pojam selekcione formule i pojam interpretacije selekcione formule. Selekciona formula služi za zadavanje kriterijuma po kojem se izdvajaju (selektuju) torke iz

relacije. Putem naredne dve definicije, formalizuju se pravila (sintaksa) za formiranje selekcionih formula.

Definicija 6.3. Neka je dat univerzalni skup obeležja $\mathcal{U}$, funkcija *dom*, skup relacionih operatora $\mathcal{R} = \{<, >, \leq, \geq, \neq, =\}$ i konačan skup parcijalnih, izračunljivih, logičkih funkcija $\mathcal{F} = \{ f_i^{k_i} \mid i \in \{1,..., n\}\}$*), pri čemu za svaki $i \in \{1,..., n\}$ k_i predstavlja arnost, a f_i naziv funkcije. ***Atomarna (atomična) selekciona formula*** je izraz oblika:

- $A\ \theta B$, pri čemu je $A, B \in \mathcal{U}$ i $\theta \in \mathcal{R}$,
- $A\ \theta c$, ili $c\ \theta A$, pri čemu je $A \in \mathcal{U}$, $c \in dom(A)$ i $\theta \in \mathcal{R}$, ili
- $f_i(x_1,..., x_{k_i})$, pri čemu je $f_i \in \mathcal{F}$ i $(\forall j \in \{1,..., k_i\})(x_j \in \mathcal{U} \vee x_j \in Dom)$, gde je $Dom = \bigcup_{A \in U} dom(A)$. □

Definicija 6.4. Dati su $\mathcal{U}$, *dom* i skup logičkih operacija $\mathcal{L} = \{\neg, \wedge, \vee\}$.

1. Svaka atomarna selekciona formula je ***selekciona formula***.
2. Ako su F i G selekcione formule, onda su ***selekcione formule*** izrazi: $(\neg F)$, $(F \wedge G)$ i $(F \vee G)$.
3. ***Selekciona formula*** je svaki izraz, dobijen primenom pravila 1. i 2. konačno mnogo puta. □

Dogovorom o prioritetu logičkih operacija, po kojem je najprioritetnija operacija negacije ($\neg$), zatim konjukcije ($\wedge$) i na kraju disjunkcije ($\vee$), uvodi se konvencija o uklanjanju zagrada iz selekcionih formula.

Sledeća definicija uvodi pravila za interpretaciju (izračunavanje vrednosti) selekcione formule. Interpretacija selekcione formule se uvek vrši s obzirom na neku torku. Interpretacija selekcione formule je logička funkcija, što znači da vrednost selekcione formule u bilo kojoj interpretaciji može biti T - tačno, ili $\perp$ - netačno.

Definicija 6.5. Interpretacija selekcione formule F s obzirom na proizvoljnu torku $t \in Tuple(R)$, $R \subseteq \mathcal{U}$, je parcijalna logička funkcija F: $Tuple(R) \rightarrow \{T, \perp\}$, definisana putem sledećih pravila:

- Neka je $F = A\ \theta B$. $F(t)$ je definisana ako i samo ako važi da $A, B \in R$ i da je relacija θ definisana kao: $\theta \subseteq dom(A) \times dom(B)$. Pri tome je $F(t) = T$, ako važi $(t(A), t(B)) \in \theta$, a inače je $F(t) = \perp$.
- Neka je $F = A\ \theta c$, ili $F = c\ \theta A$. $F(t)$ je definisana ako i samo ako važi da $A \in R$ i da je relacija θ definisana kao: $\theta \subseteq dom(A)^2$. Pri tome je $F(t) = T$, ako važi $(t(A), c) \in \theta$, odnosno $(c, t(A)) \in \theta$, a inače je $F(t) = \perp$.
- Neka je $F = f_i(x_1,..., x_{k_i})$. $F(t)$ je definisana ako i samo ako je f_i definisana nad domenima obeležja, upotrebljenih u f_i. Pri tome je $F(t) = f_i(t(x_1),..., t(x_{k_i}))$, gde je:

---

*) Pošto je relaciona algebra teoretski model upitnog jezika, logičke funkcije skupa $\mathcal{F}$ nisu konkretizovane u okviru ove definicije.

$$(\forall j \in \{1,\ldots,k_i\})\left(t(x_j) = \begin{cases} t(x_j), & x_j \in \mathcal{U} \\ x_j, & x_j \in Dom \end{cases}\right).$$

- Neka je $F = \neg F'$. Tada je $F(t) = \neg F'(t)$. Operacija negacije se primenjuje na uobičajeni način.
- Neka je $F = H \wedge G$, odnosno $F = H \vee G$. Tada je $F(t) = H(t) \wedge G(t)$, odnosno $F(t) = H(t) \vee G(t)$. Logička operacija konjukcije, tj. disjunkcije se primenjuje na uobičajeni način. □

Primer 6.4. Posmatra se skup obeležja $R = \{A, B, C\}$, pri čemu je $dom(A) = dom(B) = \mathbf{N}$, gde je N skup prirodnih brojeva, a $dom(C) = \{a, b, c\}$. Pretpostavlja se da su nad skupom N definisane sve binarne relacije iz skupa $\mathcal{R} = \{<, >, \leq, \geq, \neq, =\}$, na uobičajen način, dok je nad skupom $dom(C)$ definisana samo relacija =, takođe na uobičajen način. Neka je parcijalna logička funkcija $plus(x, y, z) \in \mathcal{T}$ definisana tako da vraća vrednost T, ako je $z = x + y$, a inače vraća vrednost $\perp$. Neka je, takođe, zadata torka t nad skupom obeležja R, takva da je $t(A) = 2$, $t(B) = 3$ i $t(C) = c$.

- Za selekcionu formulu F_1: $B > A \wedge plus(A, B, 5) \wedge C = c$ važi da je $F_1(t) = \mathsf{T}$, pošto je ispunjeno: $t(B) > t(A) \wedge plus(t(A), t(B), 5) \wedge t(C) = c$.
- Za selekcionu formulu F_2: $\neg(B > A \wedge plus(A, B, 5) \wedge C = c)$ važi da je $F_2(t) = \perp$.
- Za selekcionu formulu F_3: $B > C \wedge plus(A, B, 5)$ važi da interpretacija $F_3(t)$ nije definisana, jer izraz $t(B) > t(C)$ nije definisan. Drugim rečima, formula F_3 nije primenljiva na skup obeležja R s obzirom na preslikavanje *dom*. □

Definicija 6.6. Data je relacija $r(R)$, $R \subseteq \mathcal{U}$. ***Selekcija*** relacije r po selekcionoj formuli F je unarna operacija: $\sigma_F : SAT(R) \rightarrow SAT(R)$, definisana na sledeći način:

$$\sigma_F(r(R)) = \{t \in r \mid F(t) = \mathsf{T}\}. \square$$

Primer 6.5. Na slici 6.3 je prikazana relacija *radnik*(R), $R = \{MBR, IME, PRZ, PLT, SEF\}$, u kojoj se nalaze osnovni podaci o radnicima. Značenje obeležja iz skupa R je isto kao u primeru 6.1.

- Upit: "Prikazati radnike čija plata je veća od *1000* i manja od *3000* i nemaju svog rukovodioca" se realizuje putem selekcije:

$$s_1 = \sigma_{(PLT > 1000) \wedge (PLT < 3000) \wedge (SEF = \omega)}(radnik).$$

- Neka je definisana logička funkcija $subs(x, y, z) \in \mathcal{T}$, takva da vraća vrednost T, ako i samo ako je prvih y znakova stringa x jednako stringu z. Upit: "Prikazati radnike koji se prezivaju '*Car*', a ime im počinje znakom '*M*' " se realizuje putem selekcije:

$$s_2 = \sigma_{F_2}(radnik), \text{ pri čemu je } F_2 : (PRZ = \text{'}Car\text{'}) \wedge subs(IME, 1, \text{'}M\text{'}).$$

Relacije s_1 i s_2 su, takođe, prikazane na slici 6.3. □

radnik =

MBR	*IME*	*PRZ*	*PLT*	*SEF*
10	*Mile*	*Car*	*1600*	ω
20	*Mira*	*Car*	*1500*	*10*
30	*Ivo*	*Han*	*1800*	*10*
40	*Ana*	*Tot*	*2800*	*20*
50	*Ana*	*Tot*	*3200*	*20*
60	*Boro*	*Kun*	*4600*	ω

s_1 =

MBR	*IME*	*PRZ*	*PLT*	*SEF*
10	*Mile*	*Car*	*1600*	ω

s_2 =

MBR	*IME*	*PRZ*	*PLT*	*SEF*
10	*Mile*	*Car*	*1600*	ω
20	*Mira*	*Car*	*1500*	*10*

Slika 6.3.

Može se pokazati da operator selekcije poseduje osobine komutativnosti i asocijativnosti.

Definicija 6.7. Data je relacija $r(R)$, $R \subseteq \mathcal{U}$ i skup obeležja $X \subseteq R$. ***Projekcija*** relacije r je unarni operator: $\pi_X : SAT(R) \rightarrow SAT(X)$, definisan na sledeći način:

$$\pi_X(r(R)) = \{t \in Tuple(X) \mid (\exists t' \in r)(t = t'[X])\}. \square$$

Primer 6.6. Data je relacija *radnik* iz primera 6.5.

- Upit: "Prikazati imena i prezimena radnika" se realizuje putem projekcije:

$$p_1 = \pi_{IME,PRZ}(radnik).$$

- Upit: "Prikazati matične brojeve, imena i prezimena radnika koji zarađuju više od *1900* dinara" se realizuje pomoću izraza:

$$p_2 = \pi_{MBR,IME,PRZ}(\sigma_{PLT > 1900}(radnik)).$$

Na slici 6.4 su prikazane relacije p_1 i p_2. $\square$

p_1 =

IME	*PRZ*
Mile	*Car*
Mira	*Car*
Ivo	*Han*
Ana	*Tot*
Boro	*Kun*

p_2 =

MBR	*IME*	*PRZ*
40	*Ana*	*Tot*
50	*Ana*	*Tot*
60	*Boro*	*Kun*

Slika 6.4.

Izraz kojim je prikazan drugi upit iz primera 6.6 ukazuje na to da operatori projekcije i selekcije ne moraju, u opštem slučaju, da budu međusobno komutativni (selekciona formula $PLT > 1900$ se može interpretirati nad relacijom *radnik*, ali se ne može interpretirati na projekciji $\pi_{MBR,IME,PRZ}(radnik)$).

Teorema 6.1. Neka je $ATTR(F)$ skup obeležja koja se pojavljuju u selekcionoj formuli F. Za proizvoljnu relaciju $r(R)$ i bilo koji $X \subseteq R$ važi implikacija:

$$ATTR(F) \subseteq X \Rightarrow \sigma_F(\pi_X(r)) = \pi_X(\sigma_F(r)).$$

Dokaz. Prvo će biti dokazano da važi $\sigma_F(\pi_X(r)) \subseteq \pi_X(\sigma_F(r))$. Odabira se proizvoljna torka $t \in \sigma_F(\pi_X(r))$. Po definiciji selekcije je $t \in \pi_X(r) \wedge F(t)$, odakle sledi, po definiciji projekcije, da $(\exists t' \in r)(t = t'[X] \wedge F(t))$. Neka je torka t_0 ta koja zadovoljava navedeni uslov, što znači da važi $t_0 \in r \wedge t = t_0[X] \wedge F(t)$. Iz pretpostavke $ATTR(F) \subseteq X$ i činjenica $t = t_0[X] \wedge F(t)$ sledi da je $t_0 \in r \wedge t = t_0[X] \wedge F(t_0)$, što znači da je $t_0 \in \sigma_F(r) \wedge t = t_0[X]$, odnosno da $(\exists t_0 \in \sigma_F(r))(t = t_0[X])$, odakle sledi da $t \in \pi_X(\sigma_F(r))$. Pošto je t proizvoljno odabrana torka, zaključuje se da važi $\sigma_F(\pi_X(r)) \subseteq \pi_X(\sigma_F(r))$.

Dokaz da je $\pi_X(\sigma_F(r)) \subseteq \sigma_F(\pi_X(r))$ se sprovodi analogno prethodnom postupku. Iz $\sigma_F(\pi_X(r)) \subseteq \pi_X(\sigma_F(r))$ i $\pi_X(\sigma_F(r)) \subseteq \sigma_F(\pi_X(r))$ sledi tvrdenje teoreme. □

Teorema 6.2. Data je relacija $r(R)$ i skupovi obeležja $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_n \subseteq R$, $n \geq 1$. Važi da je $\pi_{X_1}(\pi_{X_2}(\ldots\pi_{X_n}(r)\ldots)) = \pi_{X_1}(r)$. □

Operatorom projekcije je obezbeđena mogućnost izbora samo onih obeležja relacije, koja su od interesa za realizaciju konkretnog upita. Nasuprot ovom operatoru, operator prirodnog spajanja, koji će u nastavku biti definisan, daje mogućnost formiranja nove relacije na osnovu sadržaja dve druge relacije. Prirodnim spajanjem se povezuju torke dve relacije sa istim vrednostima nad zajedničkim skupom obeležja.

Definicija 6.8. Date su relacije $r(R)$ i $s(S)$, pri čemu je $R, S \subseteq \mathcal{U}$. ***Prirodni spoj*** relacija r i s je funkcija $\triangleright\!\triangleleft : SAT(R) \times SAT(S) \rightarrow SAT(RS)$, definisana na sledeći način:

$$r(R) \triangleright\!\triangleleft s(S) = \{t \in Tuple(RS) \mid t[R] \in r \wedge t[S] \in s\}.$$ □

Primer 6.7. Na slici 6.5 su prikazane relacije r i s i prirodni spoj ovih relacija. Sve torke relacije $r \triangleright\!\triangleleft s$ su dobijene spajanjem svih torki relacija r i s sa istim vrednostima nad skupom obeležja $ABC \cap BCD = BC$. □

$r =$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_2	b_2	c_1
a_2	b_2	c_2

$s =$

B	C	D
b_1	c_1	d_1
b_2	c_3	d_2
b_2	c_1	d_1
b_2	c_1	d_3

$r \triangleright\!\triangleleft s =$

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_1	c_1	d_1
a_2	b_2	c_1	d_1
a_2	b_2	c_1	d_3

Slika 6.5.

Sama definicija prirodnog spajanja ne prikazuje postupak formiranja prirodnog spoja dve proizvoljne relacije. Sledeća teorema predstavlja osnov postupka izračunavanja prirodnog spoja.

Teorema 6.3. Date su relacije $r(R)$ i $s(S)$, pri čemu je $R, S \subseteq \mathcal{U}$. Važi:

$$(\forall t_r \in r)(\forall t_s \in s)(t_r[R \cap S] = t_s[R \cap S] \Leftrightarrow t_r t_s \in r \rhd\lhd s),$$

pri čemu je sa $t_r t_s$ označen spoj torke t_r sa torkom t_s, takav da je $t_r t_s[R] = t_r$ i $t_r t_s[S] = t_s$.

Dokaz. ($\Rightarrow$) Neka su date proizvoljne torke $t_r \in r$ i $t_s \in s$ takve da je $t_r[R \cap S] = t_s[R \cap S]$ i neka je $t = t_r t_s$ spoj ove dve torke. Pošto je $t_r t_s[R] = t_r$, sledi da $t_r t_s[R] \in r$. Analogno, sledi da $t_r t_s[R] \in s$, a na osnovu $t_r t_s[R] \in r$ i $t_r t_s[S] \in s$ sledi da je $t_r t_s \in r \rhd\lhd s$.

($\Leftarrow$) Neka za $t_r \in r$ i $t_s \in s$ važi da je $t_r t_s \in r \rhd\lhd s$. Po definiciji prirodnog spoja je $t_r t_s[R] \in r$ i $t_r t_s[S] \in s$. Iz činjenica $t_r t_s[R] \in r$ i $t_r \in r$ sledi da je $t_r t_s[R] = t_r$. Analogno se može zaključiti da je $t_r t_s[S] = t_s$. Pošto je $R \cap S \subseteq R$, zaključuje se da je $t_r[R \cap S] = (t_r t_s[R])[R \cap S] = t_r t_s[R \cap S]$. Analogno, važi da je $t_s[R \cap S] = (t_r t_s[S])[R \cap S] = t_r t_s[R \cap S]$. Iz $t_r[R \cap S] = t_r t_s[R \cap S]$ i $t_s[R \cap S] = t_r t_s[R \cap S]$ sledi $t_r[R \cap S] = t_s[R \cap S]$. □

Primer 6.8. Na slici 6.6 su prikazane relacije r i s i prirodni spoj ovih relacija, pri čemu treba zapaziti da je $AB \cap CD = \emptyset$. □

$r =$

A	B
a_1	b_1
a_2	b_1

$s =$

C	D
c_1	d_1
c_1	d_2

$r \rhd\lhd s =$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_2	b_1	c_1	d_1
a_2	b_1	c_1	d_2

Slika 6.6.

Može se pokazati da operator prirodnog spajanja poseduje osobine komutativnosti i asocijativnosti.

Teorema 6.4. Date su relacije $r(R)$ i $s(S)$, pri čemu je $R, S \subseteq \mathcal{U}$ i selekciona formula F. Važi implikacija:

$$ATTR(F) \subseteq R \Rightarrow \sigma_F(r \rhd\lhd s) = \sigma_F(r) \rhd\lhd s.$$

Dokaz. Prvo će biti pokazano da je $\sigma_F(r \rhd\lhd s) \subseteq \sigma_F(r) \rhd\lhd s$. Odabira se proizvoljna torka $t \in \sigma_F(r \rhd\lhd s)$. Sledi da je $t \in r \rhd\lhd s \wedge F(t)$, odnosno $t[R] \in r \wedge t[S] \in s \wedge F(t)$. Pošto je $ATTR(F) \subseteq R$ iz $F(t)$ sledi $F(t[R])$, tj. može se tvrditi da je $t[R] \in r \wedge t[S] \in s \wedge F(t[R])$, a po definiciji selekcije je $t[R] \in \sigma_F(r) \wedge t[S] \in s$. Po definiciji prirodnog spoja je $t \in \sigma_F(r) \rhd\lhd s$, čime je tvrdnja dokazana.

Dokaz činjenice $\sigma_F(r) \rhd\lhd s \subseteq \sigma_F(r \rhd\lhd s)$ se ostavlja čitaocu. □

Teorema 6.4 ukazuje na činjenicu da je, sa stanovišta relacione algebre, nevažno da li se prvo vrši selekcija torki iz relacije pa njeno prirodno spajanje s drugom relacijom, ili se prvo vrši prirodno spajanje pa selekcija. Sa stanovišta performansi izvršavanja upita

na računaru, međutim, redosled operacija je veoma važan. Spajanje relacija je vremenski zahtevna operacija i nije svejedno da li se prvo spajaju sve torke dve relacije pa se tek onda odaberu samo tražene torke, ili se prvo odaberu tražene torke, koje se zatim spajaju. U prvom slučaju se, u praksi, može dogoditi da se vrši spajanje nekoliko stotina (ili čak nekoliko hiljada torki), a u drugom slučaju se spajanje može svesti na svega nekoliko traženih torki. Razlika u performansama može postati još izraženija ukoliko traženi upit zahteva i prenos podataka kroz računarsku mrežu, pod pretpostavkom da se relacije koje treba spajati nalaze na različitim čvorovima.

U cilju postizanja zadovoljavajućih performansi izvršavanja upita, savremeni relacioni sistemi za upravljanje bazama podataka poseduju, kao sastavni deo upitnog jezika, modul za optimizaciju upita. Jedan od zadataka ovog modula jeste da u realnom vremenu pronađe što efikasniju strategiju za generisanje odgovora na korisnikov upit.

Sledeće teoreme (čiji se dokazi izostavljaju) iskazuju odnose prirodnog spajanja i projekcije relacija.

Teorema 6.5. Date su relacije $r(R)$ i $s(S)$, pri čemu je $R, S \subseteq \mathcal{U}$. Važi da je $\pi_R(r \rhd\lhd s) \subseteq r$. □

Teorema 6.6. Data je relacija $r(\mathcal{U})$ i skupovi obeležja R i S, pri čemu je $RS = \mathcal{U}$, $R \neq \emptyset$ i $S \neq \emptyset$. Važi da je $r \subseteq \pi_R(r) \rhd\lhd \pi_S(r)$. □

Teorema 6.7. Date su relacije $r(R)$ i $s(S)$, pri čemu je $RS = \mathcal{U}$, $R \neq \emptyset$, $S \neq \emptyset$. Važi da je $\pi_R(r \rhd\lhd s) \rhd\lhd \pi_S(r \rhd\lhd s) = r \rhd\lhd s$. □

Definicija 6.9. Date su relacije $r(R)$ i $s(S)$, pri čemu je $R, S \subseteq \mathcal{U}$ i $R \cap S = \emptyset$. ***Dekartov proizvod*** relacija r i s je funkcija $\times : SAT(R) \times SAT(S) \rightarrow SAT(RS)$, definisana na sledeći način:

$$r(R) \times s(S) = \{t \in Tuple(RS) \mid t[R] \in r \wedge t[S] \in s\}. \square$$

Dekartov proizvod se, dakle, definiše samo za relacije koje nemaju zajednička obeležja i u tom slučaju se relacija $r \times s$ izračunava na isti način kao i relacija $r \rhd\lhd s$. Relacija $r \rhd\lhd s$ iz primera 6.7 je istovetna relaciji $r \times s$, za zadate r i s iz istog primera.

Definicija 6.10. Date su relacije $r(R)$ i $s(S)$, pri čemu je $R, S \subseteq \mathcal{U}$ i $R \cap S = \emptyset$. ***Teta spajanje*** relacija r i s po selekcionoj formuli F je funkcija $[F] : SAT(R) \times SAT(S) \rightarrow SAT(RS)$, definisana na sledeći način:

$$r(R)\,[F]\,s(S) = \sigma_F(r \times s). \square$$

Specijalan slučaj teta spajanja relacija $r(R)$ i $s(S)$ je ***ekvi spajanje***, kod kojeg je selekciona formula oblika: $F : A_1 = B_1 \wedge \ldots \wedge A_n = B_n$, pri čemu je $(\forall i \in \{1,\ldots, n\})(A_i \in R \wedge B_i \in S)$.

Primer 6.9. Na slici 6.7 su prikazane relacije r i s. Značenja obeležja su sledeća: *PNI* - polazak autobusa iz Niša, *DBG* - dolazak autobusa u Beograd, *PBG* - polazak autobusa iz Beograda i *DNS* dolazak autobusa u Novi Sad. Upit: "prikazati sve mogućnosti

za putovanje na relaciji Niš - Novi Sad s presedanjem u Beogradu u toku dana" se realizuje teta spajanjem: $r\ [PBG > DBG]\ s$. Na slici 6.7 je prikazan rezultat ovog teta spajanja. □

$r =$

PNI	DBG
6^{00}	8^{30}
10^{00}	12^{30}
14^{00}	17^{00}
18^{00}	20^{30}

$s =$

PBG	DNS
7^{00}	8^{45}
9^{00}	10^{15}
14^{00}	15^{30}
20^{00}	21^{15}

$r\ [PBG > DBG]\ s =$

PNI	DBG	PBG	DNS
6^{00}	8^{30}	9^{00}	10^{15}
6^{00}	8^{30}	14^{00}	15^{30}
6^{00}	8^{30}	20^{00}	21^{15}
10^{00}	12^{30}	14^{00}	15^{30}
10^{00}	12^{30}	20^{00}	21^{15}
14^{00}	17^{00}	20^{00}	21^{15}

Slika 6.7.

Definicija 6.11. Date su relacije $r(R)$ i $s(S)$, pri čemu je $R \subseteq \mathcal{U}$ i $S \subseteq R$. ***Količnik*** (*deljenje*) relacija r i s je funkcija: $\div : SAT(R) \times SAT(S) \rightarrow SAT(R \setminus S)$, definisana na sledeći način:

$$r(R) \div s(S) = \{t \in Tuple(R \setminus S) \mid (\forall t_s \in s)(\exists t_r \in r)(t_r[R \setminus S] = t \wedge t_r[S] = t_s)\}.$$ □

Primer 6.10. Na slici 6.8 su prikazane relacije r i s, pri čemu je značenje obeležja sledeće: A - tip aviona i P - oznaka pilota. Upit: "prikazati pilote koji mogu da lete na svim tipovima aviona" se realizuje putem deljenja $r(PA) \div s(A)$. Rezultat ove operacije je, takođe, prikazan na slici 6.8. □

$r =$

P	A
Car	747
Han	767
Tot	DC9
Car	767
Tot	737
Car	DC9
Car	737

$s =$

A
747
767
737
DC9

$r \div s =$

P
Car

Slika 6.8.

U nastavku teksta će relaciona algebra i ekstenzioni izraz relacione algebre biti formalno definisani. Neformalno, ekstenzioni izraz relacione algebre se sastoji od operatora i operanada, pri čemu su operandi konstantne relacije i relacije iz baze podataka, dok su operatori neki od prethodno definisanih operatora.

Definicija 6.12. Neka je dat univerzalni skup obeležja $\mathcal{U}$ i funkcija *dom*.

1. ***Konstantna relacija*** nad obeležjem $A \in \mathcal{U}$, u oznaci $<A : a>$, je jednočlana relacija $\{t\}$, takva da je $t(A) = a \in dom(A)$.
2. Neka su $r_{\mathcal{A}}(R)$ i $s_{\mathcal{A}}(S)$ konstantne relacije, takve da je $R \cap S = \emptyset$. Tada je $r_{\mathcal{A}}(R) \times s_{\mathcal{A}}(S)$ konstantna relacija.
3. Neka su $r_{\mathcal{A}}(R)$ i $s_{\mathcal{A}}(R)$ konstantne relacije. Tada je $r_{\mathcal{A}}(R) \cup s_{\mathcal{A}}(R)$ konstantna relacija.
4. Konstantna relacija je sve ono što se može dobiti primenom, konačno mnogo puta, pravila 1, 2, ili 3. □

Definicija 6.13. Relaciona algebra je struktura

$$\mathcal{A}_r = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega, \mathcal{O}),$$

pri čemu je $\mathcal{U}$ univerzalni skup obeležja, $\mathcal{O}_S$ skup domena, $dom : \mathcal{U} \to \mathcal{O}_S$ domenska funkcija, $\mathcal{S} = (S, \mathcal{I})$ šema relacione baze podataka, rbp relaciona baza podataka nad $\mathcal{S}$, $\Omega = \mathcal{R} \cup \mathcal{T} \cup \mathcal{L}$ skup relacionih operatora, logičkih funkcija i logičkih operatora i $\mathcal{O} = \{\cup, \cap, -, \delta, \sigma_F, \pi, \triangleright\!\triangleleft, \times, [F], \div\}$. □

Definicija 6.14. Data je relaciona algebra $\mathcal{A}_r = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega, \mathcal{O})$.

1. Svaka konstantna relacija predstavlja ***ekstenzioni izraz*** relacione algebre.
2. Svaka relacija $r \in rbp$ predstavlja ***ekstenzioni izraz*** relacione algebre.
3. Neka je E ekstenzioni izraz relacione algebre. Tada su ekstenzioni izrazi relacione algebre formule: $\delta_{X \leftarrow Y}(E)$, $\sigma_F(E)$ i $\pi_X(E)$.
4. Neka su E i H ekstenzioni izrazi relacione algebre. Tada su ekstenzioni izrazi relacione algebre formule: $(E \cup H)$, $(E \cap H)$, $(E - H)$, $(E \triangleright\!\triangleleft H)$, $(E \times H)$, $(E\ [F]\ H)$ i $(E \div H)$.
5. Ekstenzioni izraz relacione algebre je sve ono što se može dobiti primenom, konačno mnogo puta, pravila 1, 2, 3. i 4. □

Interpretacija, odnosno vrednost ekstenzionog izraza relacione algebre je relacija koja se dobija kao rezultat primene operatora iz skupa $\mathcal{O}$ na konkretne relacije, saglasno definicijama operatora. Kaže se da je ekstenzioni izraz ***legalno formiran***, ukoliko su ispoštovana sva pravila za primenu operatora relacione algebre, data definicijama 6.1-6.12. Nelegalno formiran izraz nema svoju interpretaciju.

Primer 6.11. Posmatra se šema baze podataka $\mathcal{S} = (S, \mathcal{I})$ i njena pojava $rbp = \{radnik, projekat, radproj\}$ iz primera 6.1. U nastavku će biti prikazani ekstenzioni izrazi relacione algebre za nekoliko karakterističnih upita nad bazom podataka rbp:

U1. Prikazati prezimena i imena radnika koji se prezivaju *'Petrović'*, a ime im započinje slovom *'M'*. Na raspolaganju je logička funkcija $subs(x, y, z) \in \mathcal{T}$, definisana na isti način kao u primeru 6.5.

$$\pi_{PRZ+IME}(\sigma_{PRZ = 'Petrović' \wedge subs(IME,\ 1,\ 'M')}(radnik)).$$

U2. Prikazati prezimena, imena i plate rukovodilaca (RUK) projekata:

$$\pi_{PRZ+IME+PLT}(radnik\ [MBR = RUK]\ projekat).$$

U3. Prikazati prezimena, imena i plate radnika koji imaju veću platu od radnika s matičnim brojem *30* :

$$\pi_{PRZ+IME+PLT}(radnik\ [PLT > PLT']\ \delta_{PLT \leftarrow PLT'}(\pi_{PLT}(\sigma_{MBR=30}(radnik)))).$$

U4. Prikazati matične brojeve, prezimena i imena svih rukovodilaca (*SEF*):

$$\pi_{MBR+PRZ+IME}((\delta_{SEF \leftarrow MBR}(\pi_{SEF}(radnik)) - <SEF : \omega>)) \triangleright\triangleleft radnik).$$

U5. Prikazati radnike koji nisu rukovodioci ni jednog projekta:

$$(\pi_{MBR}(radnik) - \delta_{RUK \leftarrow MBR}(\pi_{RUK}(projekat))) \triangleright\triangleleft radnik.$$

U6. Prikazati matične brojeve, prezimena i imena radnika koji rade na projektu sa šifrom *110* :

$$\pi_{MBR+PRZ+IME}(radnik \triangleright\triangleleft \sigma_{SPR=110}(radproj)).$$

U7. Prikazati matične brojeve, prezimena i imena radnika koji rade na projektu na kojem radi i radnik s matičnim brojem *40*:

$$\pi_{MBR+PRZ+IME}(\pi_{SPR}(\sigma_{MBR=40}(radproj)) \triangleright\triangleleft radproj \triangleright\triangleleft radnik).$$

U8. Prikazati matične brojeve, prezimena i imena radnika koji rade na svim projektima:

$$\pi_{MBR+PRZ+IME}((radproj \div \pi_{SPR}(projekat)) \triangleright\triangleleft radnik). \square$$

6.1.2. *Ekspresivna moć i generativni deo relacione algebre*

Skup svih upita, koji se mogu izraziti putem legalnih ekstenzionih izraza relacione algebre, predstavlja ***ekspresivnu moć*** relacione algebre. Ekspresivna moć relacione algebre se uzima kao standard, tako da se od svakog upitnog jezika, namenjenog za praktičnu upotrebu u okviru nekog sistema za upravljanje bazama podataka, zahteva da ima ekspresivnu moć veću ili jednaku ekspresivnoj moći relacione algebre.

Generativni deo relacione algebre $\mathcal{A}_r = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega, \mathcal{O})$ predstavlja minimalni skup operatora i operanada relacione algebre, koji ima istu ekspresivnu moć kao i relaciona algebra.

Može se dokazati da struktura $\mathcal{G}(\mathcal{A}_r) = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega, \mathcal{O}')$, pri čemu je:

- $\mathcal{O}' = \{\cup, -, \delta, \sigma_F, \pi, \triangleright\triangleleft\}$,
- kao konstantne relacije se koriste samo relacije oblika $<A : a>$ (date definicijom 6.12, tačka 1), a
- kao selekcione formule se koriste samo atomarne selekcione formule (date definicijom 6.3)

predstavlja generativni deo relacione algebre.

Teorema 6.8. Ekspresivna moć strukture $\mathcal{G}(\mathcal{A}_r)$ je jednaka ekspresivnoj moći relacione algebre $\mathcal{A}_r$.

Dokaz. Treba dokazati da se svaki ekstenzioni izraz relacione algebre može izraziti putem ekvivalentnog ekstenzionog izraza strukture $\mathcal{G}(\mathcal{A}_r)$.

1. Neka je $\mathcal{A}_r$ ekstenzioni izraz $E = r_c$, gde je r_c konstantna relacija, dobijena primenom definicije 6.12. Ekvivalentni ekstenzioni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ će se sastojati iz konstantnih relacija oblika $<A : a>$, povezanih operatorom $\triangleright\triangleleft$, na mestu gde je pri formiranju izraza E korišćeno pravilo 2. definicije 6.12, ili operatorom $\cup$, na mestu gde je pri formiranju izraza E korišćeno pravilo 3. definicije 6.12.
2. $\mathcal{A}_r$ ekstenzioni izraz $E = r \in rbp$ predstavlja, istovremeno, i ekstenzioni izraz strukture $\mathcal{G}(\mathcal{A}_r)$.
3. Neka je $\mathcal{A}_r$ ekstenzioni izraz $E = H \cap G$, pri čemu su H i G ekstenzioni izrazi strukture $\mathcal{G}(\mathcal{A}_r)$. Tada je ekvivalentni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ oblika $H - (H - G)$. (Dokaz tvrđenja je ostavljen čitaocu.)
4. Neka je $\mathcal{A}_r$ ekstenzioni izraz $E = H \times G$, pri čemu su H i G ekstenzioni izrazi strukture $\mathcal{G}(\mathcal{A}_r)$. Tada je ekvivalentni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ oblika $H \triangleright\triangleleft G$. (Dokaz tvrđenja je ostavljen čitaocu.)
5. Neka je $\mathcal{A}_r$ ekstenzioni izraz $E = H \div G$, pri čemu su H i G ekstenzioni izrazi strukture $\mathcal{G}(\mathcal{A}_r)$, čiji rezultat predstavljaju, redom, relacije nad skupom obeležja R i S. Tada je ekvivalentni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ oblika $\pi_{R\setminus S}(H) - \pi_{R\setminus S}((\pi_{R\setminus S}(H) \triangleright\triangleleft G) - H)$. (Dokaz tvrđenja je ostavljen čitaocu.)
6. Neka je $\mathcal{A}_r$ ekstenzioni izraz $E = \sigma_F(H)$, pri čemu je H ekstenzioni izraz strukture $\mathcal{G}(\mathcal{A}_r)$. Ako je F atomarna selekciona formula, tada je E, istovremeno, izraz strukture $\mathcal{G}(\mathcal{A}_r)$. Ako je F oblika $F = F' \wedge F''$, tada je ekvivalentni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ oblika $\sigma_{F'}(H) - (\sigma_{F'}(H) - \sigma_{F''}(H))$. Ako je F oblika $F = F' \vee F''$, tada je ekvivalentni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ oblika $\sigma_{F'}(H) \cup \sigma_{F''}(H)$. Ako je F oblika $F = \neg F'$, tada je ekvivalentni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ oblika $H - \sigma_{F'}(H)$. (Dokazi tvrđenja su ostavljeni čitaocu.)
7. Neka je $\mathcal{A}_r$ ekstenzioni izraz $E = H\ [F]\ G$, pri čemu su H i G ekstenzioni izrazi strukture $\mathcal{G}(\mathcal{A}_r)$. Tada se ekvivalentni izraz strukture $\mathcal{G}(\mathcal{A}_r)$ dobija transformacijom E u oblik $\sigma_F(H \triangleright\triangleleft G)$, a zatim se primenjuje transformacija, prikazana u tački 6. ovog dokaza. (Dokaz tvrđenja je ostavljen čitaocu.) □

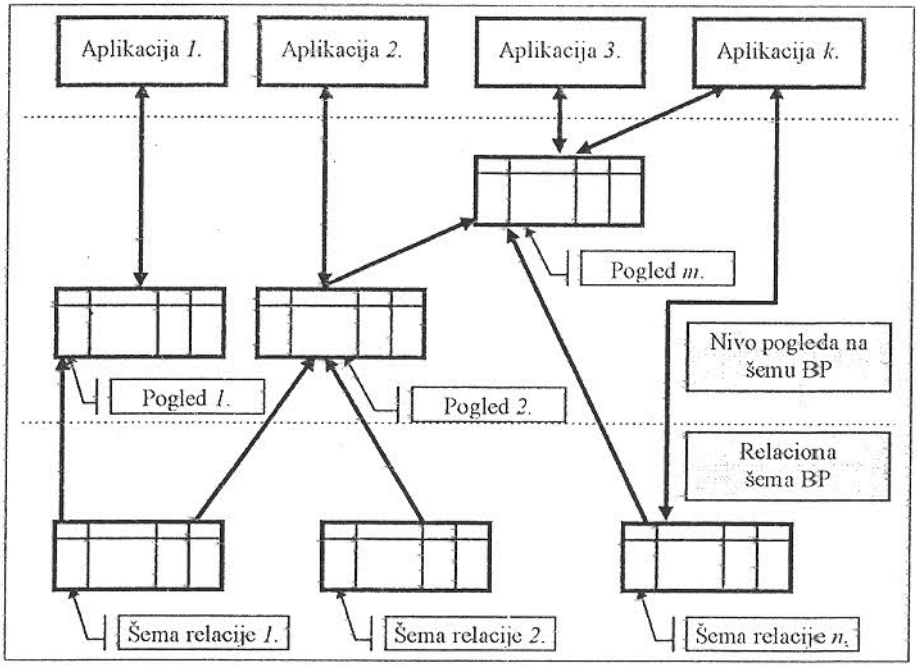
6.1.3. *Pojam pogleda i kreiranje pogleda putem relacione algebre*

Jedna od dobrih osobina relacionog modela podataka jeste obezbeđenje logičke nezavisnosti, odnosno visokog nivoa nezavisnosti aplikativnih programa od celokupne šeme baze podataka. Krajnjeg korisnika aplikativnog programa informacionog sistema zanimaju podaci, pokriveni samo jednim delom šeme baze podataka. Programeru aplikativnog programa je, za realizaciju, takođe potreban tačno određeni deo šeme baze podataka, dok projektanti šemu baze podataka posmatraju kao jednu celinu. Zbog toga je

"sliku" šeme baze podataka, kao celine, potrebno transformisati u "sliku" skupa pogleda na šemu baze podataka.

U relacionom modelu podataka, pogled predstavlja specijalnu šemu relacije, koja se formira na osnovu nekih šema relacija šeme baze podataka, ili drugih, prethodno kreiranih, pogleda. Na slici 6.9 je šematski prikazan način formiranja nekih pogleda. Tako, na primer, pogled *1.* je formiran na osnovu šeme relacije *1*, pogled *2.*je formiran na osnovu šema relacija *1* i *2*, dok je pogled *m* formiran na osnovu šeme relacije *n* i pogleda *2*. Relacije, koje predstavljaju pojave pogleda, ne sadrže svoje sopstvene podatke, već se ovi podaci dobijaju na osnovu podataka iz odgovarajućih relacija baze podataka. S druge strane, relacije baze podataka nad odgovarajućom šemom sadrže sopstvene (konkretne) podatke pa se zbog toga one često nazivaju ***baznim relacijama***.

Kreiranje pogleda predstavlja jedan od mehanizama za formiranje tzv. eksternog nivoa šeme baze podataka (slika 6.9), kojim se obezbeđuje logička nezavisnost programa od podataka. Putem pogleda se deo šeme baze podataka, potreban za realizaciju aplikativnog programa, transformiše u jednu šemu relacije, nad kojom će se program realizovati. ***Pogled***, prema tome, predstavlja šemu relacije, koja služi za informacino pokrivanje jednog korisničkog zahteva. Na slici 6.9 je, na primer, vidljivo da će se pri realizaciji i upotrebi aplikacije *1.* koristiti pogled *1*, dok će se pri realizaciji i upotrebi aplikacije *k* koristiti kako pogled *m*, tako i šema relacije *n*.



Slika 6.9.

Intenzioni izraz, koji će u nastavku biti definisan, reprezentuje pogled, formiran putem relacione algebre.

Definicija 6.15. Data je relaciona algebra $\mathcal{A}_r = (\mathcal{U}, \mathcal{D}_S, dom, \mathcal{S}, rbp, \Omega, \mathcal{O})$.

1. Bilo koja konstantna relacija, ili bilo koji naziv konstantne relacije predstavlja ***intenzioni izraz*** relacione algebre.
2. Svaka oznaka šeme relacije $N(R, \mathcal{K})$ predstavlja ***intenzioni izraz*** relacione algebre. Kao oznaka šeme relacije $N(R, \mathcal{K})$ se, saglasno pretpostavci o postojanju šeme univerzalne relacije, koristi skup obeležja R, a može se koristiti i njen naziv N.
3. Neka je E intenzioni izraz relacione algebre. Tada su intenzioni izrazi relacione algebre formule: $\delta_{X\leftarrow Y}(E)$, $\sigma_F(E)$ i $\pi_X(E)$.
4. Neka su E i H intenzioni izrazi relacione algebre. Tada su intenzioni izrazi relacione algebre formule: $(E \cup H)$, $(E \cap H)$, $(E - H)$, $(E \rhd\lhd H)$, $(E \times H)$, $(E\ [F]\ H)$ i $(E \div H)$.
5. Intenzioni izraz relacione algebre je sve ono što se može dobiti primenom, konačno mnogo puta, pravila 1, 2, 3.i 4. □

Interpretacija, odnosno vrednost intenzionog izraza relacione algebre se dobija putem interpretacije odgovarajućeg ekstenzionog izraza, koji se dobija kada se oznake šema relacija zamene konkretnim relacijama baze podataka.

Definicija 6.16. Neka je E intenzioni izraz relacione algebre, $SR(E)$ skup šema relacija izraza E, a $R(E) \subseteq \mathcal{U}$ skup obeležja, definisan izrazom E. ***Pogled*** nad intenzionim izrazom relacione algebre E predstavlja šema relacije $E(R(E), \mathcal{F})$, pri čemu je $\mathcal{F}$ skup funkcionalnih zavisnosti, dat kao:

$$\mathcal{F} = \left(\bigcup_{(R_i, \mathcal{K}_i) \in SR(E)} \{K \rightarrow A \mid K \in \mathcal{K}_i \wedge A \in R_i\} \right)\Big|_{R(E)} . \square$$

Primer 6.12. Neka je data šema baze podataka iz primera 6.1. Neka je sa R obeležen skup obeležja šeme relacije *Radnik*, sa P skup obeležja šeme relacije *Projekat*, a sa RP skup obeležja šeme relacije *Radproj*. Pogled: "radnici koji rade na svim projektima" se formira putem intenzionog izraza relacione algebre:

$$RSP = (RP \div \pi_{SPR}(P)) \rhd\lhd R),$$

i ima oblik $RSP(\{MBR, IME, PRZ, PLT, SEF\}, \{MBR \rightarrow IME, MBR \rightarrow PRZ, MBR \rightarrow PLT, MBR \rightarrow SEF\})$, što je, u ovom slučaju, ekvivalentno šemi relacije kod koje je skup ograničenja izražen putem ključeva: $RSP(\{MBR, IME, PRZ, PLT, SEF\}, \{MBR\})$. □

6.2. Relacioni račun

Drugi teoretski model upitnog jezika relacionih baza podataka predstavlja relacioni račun. Relacioni račun je definisan na principima ***predikatskog računa***, a u odnosu

na relacionu algebru poseduje dve prednosti: višeg je nivoa deklarativnosti i poseduje mehanizme za eksplicitno definisanje tipa i konteksta promenljive, što je od važnosti za računarsku implementaciju jezika.

Naglašena deklarativnost relacionog računa dolazi od načina definisanja formule relacionog računa, koja je, po svojoj strukturi, predikatskog tipa, za razliku od izraza relacione algebre, kod kojeg redosled primene operatora delimično ukazuje i na samu proceduru interpretacije izraza. Može se primetiti da su definicije 6.1 - 6.11, kojima je formalizovana interpretacija operatora relacione algebre, predikatskog tipa. Kroz ove definicije se može sagledati motivacija za definisanje relacionog računa, pošto su principi izgradnje formula relacionog računa vrlo slični principima definisanja operatora relacione algebre, datim kroz definicije 6.1 - 6.11.

Tip i kontekst promenljive relacionog računa se definišu u odnosu na univerzalni skup obeležja $\mathcal{U}$, saglasno upotrebi univerzalnog i egzistencijalnog kvantifikatora.

Postoje dve vrste relacionog računa:

- relacioni račun *nad torkama* i
- relacioni račun *nad domenima*.

Principi formiranja formula relacionog računa nad torkama i relacionog računa nad domenima su isti. Razlika je u tome da kod relacionog računa nad torkama promenljiva formule reprezentuje torku, definisanu nad skupom obeležja X, $X \subseteq \mathcal{U}$, dok kod relacionog računa nad domenima promenljiva formule predstavlja vrednost obeležja iz univerzalnog skupa $\mathcal{U}$. U nastavku ove tačke će biti dat prikaz relacionog računa nad torkama.

Sledeći primer daje prikaz jednog izraza relacionog računa nad torkama i objašnjava način njegovog formiranja, u cilju sticanja globalne predstave o relacionom računu. Pojmovi izraza i formule relacionog računa, koji se u primeru koriste, u okviru ove tačke se, zatim, i formalno definišu.

Primer 6.13. Neka su dati šema baze podataka $\mathcal{S} = (S, \mathcal{I})$ i njena pojava $rbp = \{radnik, projekat, radproj\}$ iz primera 6.1. Neka je sa R obeležen skup obeležja šeme relacije *Radnik*, sa P skup obeležja šeme relacije *Projekat*, a sa RP skup obeležja šeme relacije *Radproj*. Upit:

U0. prikazati nazive projekata, kao i šifre, imena i prezimena rukovodilaca odgovarajućih projekata, na kojima radi radnik s matičnim brojem $MBR = 10$

se realizuje putem sledećeg izraza relacionog računa nad torkama:

$$E = \{x(NAP + RUK + PRZ + IME) \mid F(x)\},$$

pri čemu je $F(x)$ sledeća formula relacionog računa nad torkama:

$$(\exists y)(R)(\exists z)(RP)(\exists w)(P)(radnik(y) \wedge radproj(z) \wedge projekat(w) \wedge z(MBR) = 10 \wedge z(SPR) = w(SPR) \wedge w(RUK) = y(MBR) \wedge y[PRZ + IME] = x[PRZ + IME] \wedge w[RUK + NAP] = x[RUK + NAP]).$$

Rezultat izraza E treba da bude skup svih torki $t \in Tuple(NAP + RUK + PRZ + IME)$ za koje je zadovoljena formula $F(x)$, kada torka t preuzme ulogu promenljive x. Zamena promenljive x torkom t u formuli F se označava na sledeći način: $F(x \mid t)$.

Da bi se izračunao izraz relacionog računa E, potrebno je interpretirati formulu $F(x)$. Za interpretiranje formule $F(x)$, potrebno je imati u vidu da su y, z i w promenljive

relacionog računa, pri čemu promenljiva y reprezentuje torku nad skupom obeležja R, z reprezentuje torku nad skupom obeležja RP, a w torku nad skupom P. Činjenica *radnik*(y) znači da svaka torka koju reprezentuje promenljiva y mora pripadati relaciji *radnik*. Analogno, *radproj*(z), odnosno *projekat*(w), znači da svaka torka koju reprezentuje promenljiva z, tj. w, mora pripadati relaciji *radproj*, tj. *projekat*. Neka je formula $F(x)$, u cilju lakše analize, prikazana na sledeći način:
$F(x)$:

(*) $(\exists z)(RP)(radproj(z) \wedge z(MBR) = 10 \wedge ($
(**) $(\exists w)(P)(projekat(w) \wedge w(SPR) = z(SPR) \wedge w[RUK + NAP] = x[RUK + NAP] \wedge ($
(***) $(\exists y)(R)(radnik(y) \wedge y(MBR) = w(RUK) \wedge y[PRZ + IME] = x[PRZ + IME]))))))$.

Zagrade, primenjene u ovom prikazu formule $F(x)$, ukazuju na mogući redosled "operacija", koje treba primeniti u cilju izračunavanja izraza E. Jedan od načina (postupaka) za izračunavanje izraza E, saglasno datom prikazu formule $F(x)$, je:

1. Na osnovu (*), izdvojiti sve torke $t_z \in radproj$, takve da je ispunjeno $t_z(MBR) = 10$.
2. Na osnovu (**), za svaku izdvojenu torku t_z, pronaći sve torke $t_w \in projekat$, takve da važi $t_w(SPR) = t_z(SPR)$. Za svaku, tako izdvojenu, torku t_w, formirati torku $t_x[RUK + NAP]$, pri čemu će biti $t_x[RUK + NAP] = t_w[RUK + NAP]$.
3. Na osnovu (***), za svaku izdvojenu torku t_w, pronaći sve torke $t_y \in radnik$, takve da je $t_y(MBR) = t_w(RUK)$. Za svaku tako izdvojenu torku t_y, proširiti odgovarajuću torku t_x, tako da bude $t_x \in Tuple(NAP + RUK + PRZ + IME)$, pri čemu važi $t_x[PRZ + IME] = t_y[PRZ + IME]$.

Činjenica da su pri sprovođenju postupka 1-3. ispitane sve važeće kombinacije torki iz relacija *radproj*, *projekat* i *radnik*, znači da će rezultat izraza E biti skup **svih** torki t_x, za koje važi $F(x \mid t_x)$.

Treba zapaziti da je formula $F(x)$ deklarativnog karaktera, što znači da postupak izračunavanja izraza E **ne zavisi** od načina njenog formiranja. □

6.2.1. *Iskazivanje upita putem relacionog računa nad torkama*

Za izražavanje upita se koriste ekstenzioni izrazi relacionog računa. Osnovu za definisanje ekstenzionog izraza predstavlja definicija formule relacionog računa. U nastavku teksta će biti date definicije atomarne formule i formule relacionog računa. Nakon toga, uvode se pojmovi: "legalna formula", "vezana" i "slobodna" promenljiva, "kontekst" i "tip" promenljive relacionog računa. Na taj način, uvode se pravila (sintaksa) za formiranje formula relacionog računa. Zatim se razmatra problem interpretacije (značenja) sintaksno ispravnih, legalnih formula.

Definicija 6.17. Neka su dati: univerzalni skup obeležja $\mathcal{U}$, funkcija *dom*, šema relacione baze podataka $\mathcal{S}$, relaciona baza podataka *rbp*, skup relacionih operatora $\mathcal{R} = \{<, >, \leq, \geq, \neq, =\}$, skup promenljivih $\mathcal{V} = \{x, y, z, x_1, y_1, z_1, \ldots, x_n, y_n, z_n, \ldots\}$ i konačan skup

parcijalnih, izračunljivih, logičkih funkcija $\mathcal{F} = \{ f_i^{k_i} \mid i \in \{1,..., n\}\}$, pri čemu za svaki $i \in \{1,..., n\}$ k_i predstavlja arnost, a f_i naziv funkcije. ***Atomarna formula relacionog računa (atom)*** je izraz oblika:

- $r_i(x)$, pri čemu je $r_i \in rbp$, a $x \in \mathcal{V}$,
- $x(A)\ \theta\ y(B)$, pri čemu je $A, B \in \mathcal{U}$, $x, y \in \mathcal{V}$ i $\theta \in \mathcal{R}$,
- $x(A)\ \theta\ c$, ili $c\ \theta\ x(A)$, pri čemu je $A \in \mathcal{U}$, $x \in \mathcal{V}$, $c \in dom(A)$ i $\theta \in \mathcal{R}$, ili
- $f_i(x_1,..., x_{k_i})$, pri čemu je $f_i \in \mathcal{F}$ i $(\forall j \in \{1,..., k_i\})(x_j \in Dom \vee (\exists A \in \mathcal{U})(\exists x \in \mathcal{V})(x_j = x(A)))$, gde je $Dom = \bigcup_{A \in U} dom(A)$. □

Primer 6.14. Posmatra se formula $F(x)$ iz primera 6.13. Sledeći zapisi predstavljaju atomarne formule relacionog računa, sadržane u $F(x)$:

- *radnik*(y),
- *radproj*(z),
- *projekat*(w),
- $z(MBR) = 10$,
- $z(SPR) = w(SPR)$ i
- $w(RUK) = y(MBR)$. □

Definicija 6.18. Neka je $\mathcal{A}_{\mathcal{F}}$ skup svih atomarnih formula relacionog računa, $\mathcal{U}$ univerzalni skup obeležja, $\mathcal{V}$ skup promenljivih i neka je dat skup simbola $\mathcal{L}_{\mathcal{R}} = \{\neg, \wedge, \forall\}$.

1. Svaka atomarna formula iz skupa $\mathcal{A}_{\mathcal{F}}$ je ***formula relacionog računa.***
2. Ako su F i G formule relacionog računa, onda su ***formule relacionog računa*** izrazi: $(\neg F)$, $(F \wedge G)$ i $(\forall x)(R)(F)$, pri čemu je $x \in \mathcal{V}$, a $R \subseteq \mathcal{U}$.
3. ***Formula relacionog računa*** je svaki izraz, dobijen primenom pravila 1. i 2, konačno mnogo puta. □

Dogovorom o prioritetu logičkih operacija, po kojem je najprioritetnija operacija negacije $(\neg)$, a zatim konjukcije $(\wedge)$, uvodi se konvencija o uklanjanju zagrada iz formula relacionog računa.

Umesto formule relacionog računa $\neg(\neg F \wedge \neg G)$ se može koristiti formula $(F \vee G)$. Umesto formule $(\neg F \vee G)$ se može koristiti formula $(F \Rightarrow G)$, dok se formula $(F \Rightarrow G) \wedge (G \Rightarrow F)$ može zameniti formulom $(G \Leftrightarrow F)$. Formula $\neg(\forall x)(R)(\neg F)$ može biti zamenjena formulom $(\exists x)(R)(F)$. Zamena za formulu oblika: $x(A_1) = y(A_1) \wedge ... \wedge x(A_n) = y(A_n)$ je formula $x[X] = y[X]$, pri čemu je $X = A_1 ... A_n$.

Primer 6.15. Posmatra se formula $F(x)$ iz primera 6.13. Sledeći zapisi predstavljaju formule relacionog računa, sadržane u $F(x)$:

- $F_1(x, y)$: $y[PRZ + IME] = x[PRZ + IME]$,
- $F_2(x, y, w)$: $radnik(y) \wedge y(MBR) = w(RUK) \wedge F_1(x, y)$,
- $F_3(x, w)$: $(\exists y)(R)(F_2(x, y, w))$,
- $F_4(x, z, w)$: $projekat(w) \wedge w(SPR) = z(SPR) \wedge w[RUK + NAP] = x[RUK + NAP] \wedge F_3(x, w)$,

- $F_5(x, z)$: $(\exists w)(P)(F_4(x, z, w))$,
- $F_6(x, z)$: $radproj(z) \wedge z(MBR) = 10 \wedge F_5(x, z)$ i
- $F(x)$: $(\exists z)(RP)(F_6(x, z))$. □

Iz definicije 6.18 proizilazi da svaka promenljiva u formuli relacionog računa može biti slobodna, odnosno globalna (ako je van opsega delovanja univerzalnog ($\forall$), ili egzistencijalnog ($\exists$) kvantifikatora), ili vezana, tj. lokalna (ako je u opsegu delovanja kvantifikatora). Promenljiva relacionog računa može imati svoju oblast definisanosti - skup obeležja na koji se odnosi, kao i kontekst - skup obeležja, nad kojim je promenljiva primenjena u formuli. U nastavku će biti data definicija pojma tipa (oblasti definisanosti) i konteksta (opsega delovanja) promenljive, kao i pojma vezane i slobodne promenljive. Biće, takođe, definisan pojam legalne formule relacionog računa. Skup svih slobodnih promenljivih formule će biti označavan kao $free(F)$. Za kontekst promenljive x u formuli F će biti korišćena oznaka $men(x, F)$, dok će se za tip koristiti oznaka $type(x, F)$.

Definicija 6.19. Neka je data formula relacionog računa F.

- Neka je $F = r_i(x)$, pri čemu je $r_i(R_i) \in rbp$. x je ***slobodna*** promenljiva u F: $free(F) = \{x\}$. ***Tip*** i ***kontekst*** promenljive se definišu kao: $type(x, F) = men(x, F) = R_i$. F je ***legalna*** formula.
- Neka je $F = x(A)\ \theta\ y(B)$. x i y su ***slobodne*** promenljive u F: $free(F) = \{x, y\}$, $type(x, F)$ i $type(y, F)$ su nedefinisani, što će biti označavano na sledeći način: $type(x, F) = type(y, F) = UNDEF$. Za kontekst promenljivih x i y važi: $men(x, F) = A$ i $men(y, F) = B$. F je ***legalna*** formula ako i samo ako je relacija θ definisana kao: $\theta \subseteq dom(A) \times dom(B)$.
- Neka je $F = x(A)\ \theta\ c$, ili $F = c\ \theta\ x(A)$. x je ***slobodna*** promenljiva u F: $free(F) = \{x\}$, tip je nedefinisan: $type(x, F) = UNDEF$, dok je $men(x, F) = A$. F je ***legalna*** formula ako i samo ako je relacija θ definisana kao: $\theta \subseteq dom(A)^2$.
- Neka je $F = f_i(x_1, ..., x_{k_i})$, pri čemu je $f_i \in \mathcal{F}$. Za svaki argument funkcije x_j, za koji važi da je $x_j = x(A)$, pri čemu je $A \in \mathcal{U}$ i $x \in \mathcal{V}$, x je ***slobodna*** promenljiva u F: $free(F) = \{x \mid x_j = x(A) \wedge x_j \in \{x_1, ..., x_{k_i}\}\}$, $type(x, F) = UNDEF$, dok je $men(x, F) = A$. F je ***legalna*** formula ako i samo ako je f_i definisana nad domenima obeležja, upotrebljenih u f_i.
- Neka je $F = (\neg G)$, pri čemu je G legalna formula relacionog računa. Sve vezane (slobodne) promenljive formule G ostaju vezane (slobodne) i u formuli F: $free(F) = free(G)$. Svaka slobodna promenljiva x formule G zadržava svoj tip i kontekst: $type(x, F) = type(x, G)$ i $men(x, F) = men(x, G)$. F je legalna formula.
- Neka je $F = (G \wedge H)$, pri čemu su G i H legalne formule relacionog računa. Sve slobodne (vezane) promenljive formule G i H ostaju slobodne (vezane) i u formuli F: $free(F) = free(G) \cup free(H)$. Svaka slobodna promenljiva $x \in free(F)$ ima kontekst: $men(x, F) = men(x, G) \cup men(x, H)$. Tip promenljive x se određuje na sledeći način:
  - Ako su $type(x, G) = type(x, H) = UNDEF$, tada je i $type(x, F) = UNDEF$. Formula F je legalna.

- Ako je $type(x, G) \neq UNDEF$ *), a $type(x, H) = UNDEF$, tada je F legalna, ako i samo ako je $men(x, H) \subseteq type(x, G)$. U tom slučaju je $type(x, F) = type(x, G)$.
- Ako je $type(x, G) = UNDEF$, a $type(x, H) \neq UNDEF$, tada je F legalna, ako i samo ako je $men(x, G) \subseteq type(x, H)$. U tom slučaju je $type(x, F) = type(x, H)$.
- Ako je $type(x, G) = type(x, H) \neq UNDEF$, tada je F legalna ako i samo ako je $type(x, G) = type(x, H)$. U tom slučaju je $type(x, F) = type(x, H) = type(x, G)$.

- Neka je $F = (\forall x)(R)(G)$, pri čemu je G legalna formula relacionog računa. F je legalna ako i samo ako je $x \in free(G) \wedge men(x, G) \subseteq R \wedge (type(x, G) \neq UNDEF \Rightarrow type(x, G) = R)$. Promenljiva x postaje vezana, dok sve ostale slobodne (vezane) promenljive formule G ostaju slobodne (vezane) i u F: $free(F) = free(G) \setminus \{x\}$. Tip i kontekst promenljive x postaju nedefinisani: $type(x, F) = men(x, F) = UNDEF$. Za bilo koju drugu promenljivu $y \neq x$ formule G važi da je $type(y, F) = type(y, G)$ i $men(y, F) = men(y, G)$. □

Primer 6.16. Posmatraju se formule F_1, F_2, F_3, F_4, F_5, F_6 i F iz primera 6.15. Sa R je označen skup obeležja šeme relacije *Radnik*, sa RP šeme relacije *Radproj*, a sa P šeme relacije *Projekat*, iz primera 6.1. Slobodne promenljive, tipovi i konteksti promenljivih u ovim formulama su:

- $F_1(x, y)$: $free(F_1) = \{x, y\}$,
  $men(x, F_1) = men(y, F_1) = \{PRZ, IME\}$,
  $type(x, F_1) = type(y, F_1) = UNDEF$.
- $F_2(x, y, w)$: $free(F_2) = \{x, y, w\}$,
  $men(x, F_2) = men(x, F_1) = \{PRZ, IME\}$,
  $men(y, F_2) = men(y, F_1) \cup \{MBR\} = \{MBR, PRZ, IME\}$,
  $men(w, F_2) = \{RUK\}$,
  $type(x, F_2) = type(x, F_1) = UNDEF$,
  $type(y, F_2) = R$,
  $type(w, F_2) = UNDEF$.
- $F_3(x, w)$: $free(F_3) = \{x, w\}$,
  $men(x, F_3) = men(x, F_2) = \{PRZ, IME\}$,
  $men(y, F_3) = UNDEF$,
  $men(w, F_3) = men(w, F_2) = \{RUK\}$,
  $type(x, F_3) = type(x, F_2) = UNDEF$,
  $type(y, F_3) = UNDEF$,
  $type(w, F_3) = type(w, F_2) = UNDEF$.
- $F_4(x, z, w)$: $free(F_4) = \{x, z, w\}$,
  $men(x, F_4) = men(x, F_3) \cup \{RUK, NAP\} = \{RUK, NAP, PRZ, IME\}$,
  $men(y, F_4) = men(y, F_3) = UNDEF$,
  $men(z, F_4) = \{SPR\}$,
  $men(w, F_4) = men(w, F_3) \cup \{SPR, RUK, NAP\} = \{SPR, RUK, NAP\}$,
  $type(x, F_4) = type(x, F_3) = UNDEF$,
  $type(y, F_4) = type(y, F_3) = UNDEF$,

*) Činjenica da je tip promenljive x u nekoj formuli F definisan, što se označava kao $type(x, F) \neq UNDEF$, znači da važi $type(x, F) \subseteq \mathcal{U}$, gde je $\mathcal{U}$ univerzalni skup obeležja.

$type(z, F_4) = UNDEF$,
$type(w, F_4) = P$.

- $F_5(x, z)$: $free(F_5) = \{x, z\}$,
$men(x, F_5) = men(x, F_4) = \{RUK, NAP, PRZ, IME\}$,
$men(y, F_5) = men(y, F_4) = UNDEF$,
$men(z, F_5) = men(z, F_4) = \{SPR\}$,
$men(w, F_5) = UNDEF$,
$type(x, F_5) = type(x, F_4) = UNDEF$,
$type(y, F_5) = type(y, F_4) = UNDEF$,
$type(z, F_5) = type(z, F_4) = UNDEF$,
$type(w, F_5) = UNDEF$.
- $F_6(x, z)$: $free(F_6) = \{x, z\}$,
$men(x, F_6) = men(x, F_5) = \{RUK, NAP, PRZ, IME\}$,
$men(y, F_6) = men(y, F_5) = UNDEF$,
$men(z, F_6) = men(z, F_5) \cup \{MBR\} = \{SPR, MBR\}$,
$men(w, F_6) = men(w, F_5) = UNDEF$,
$type(x, F_6) = type(x, F_5) = UNDEF$,
$type(y, F_6) = type(y, F_5) = UNDEF$,
$type(z, F_6) = RP$,
$type(w, F_6) = type(w, F_5) = UNDEF$.
- $F(x)$: $free(F) = \{x\}$,
$men(x, F) = men(x, F_6) = \{RUK, NAP, PRZ, IME\}$,
$men(y, F) = men(y, F_6) = UNDEF$,
$men(z, F) = UNDEF$,
$men(w, F) = men(w, F_6) = UNDEF$,
$type(x, F) = type(x, F_6) = UNDEF$,
$type(y, F) = type(y, F_6) = UNDEF$,
$type(z, F) = UNDEF$,
$type(w, F) = type(w, F_6) = UNDEF$.

Formule F_1, F_2, F_3, F_4, F_5, F_6 i F su legalne. □

Sledećom definicijom se uvode pravila za interpretaciju legalne formule. Praviła za interpretaciju formule predstavljaju osnov za izračunavanje izraza relacionog računa. Legalna formula relacionog računa sa n slobodnih promenljivih definiše, nakon interpretacije svih vezanih promenljivih, odgovarajući predikat iste dužine, čija će interpretacija, zatim, zavisiti od konkretnih vredosti koje date, slobodne promenljive poprime.

Definicija 6.20. Neka je data legalna formula relacionog računa F, sa skupom slobodnih promenljivih $free(F) = \{x_1,..., x_n\}$, u oznaci $F(x_1,..., x_n)$. ***Interpretacija*** formule F s obzirom na skup torki $\{t_1,...t_n\}$, u oznaci $F(x_1 | t_1,..., x_n | t_n)$, je logička funkcija $F: \mathcal{V}^n \rightarrow \{\mathsf{T}, \perp\}$, definisana putem sledećih pravila:

- Neka je $F(x) = r_i(x)$, pri čemu je $r_i(R_i) \in rbp$. Tada je $F(x \mid t) = \mathsf{T}$, ako važi $t \in r_i$, a inače je $F(x \mid t) = \perp$.
- Neka je $F(x, y) = x(A)\ \theta\ y(B)$. Tada je $F(x \mid t_1, y \mid t_2) = \mathsf{T}$, ako važi $(t_1(A), t_2(B)) \in \theta$, a inače je $F(x \mid t_1, y \mid t_2) = \perp$.

- Neka je $F(x) = x(A)\ \theta\ c$, ili $F(x) = c\ \theta\ x(A)$. Tada je $F(x \mid t) = \mathsf{T}$, ako važi $(t(A), c) \in \theta$, odnosno $(c, t(A)) \in \theta$, a inače je $F(x \mid t) = \bot$.
- Neka je $F(x_1,..., x_n) = f_i(x_1,..., x_{k_i})$, gde je $f_i \in \mathcal{F}$, $n \le k_i$. Tada je $F(x_1 \mid t_1,..., x_n \mid t_n) = f_i(\xi_1,..., \xi_{k_i})$, pri čemu je ispunjeno:

$$(\forall j \in \{1,..., k_i\})\left(\xi_j = \begin{cases} t_j(A), & \xi_j = x_i(A) \wedge A \in \mathcal{U} \\ \xi_j, & \xi_j \in Dom \end{cases}\right).$$

- Neka je $F(x_1,..., x_n) = \neg F'(x_1,..., x_n)$. Tada je $F(x_1 \mid t_1,..., x_n \mid t_n) = \neg F'(x_1 \mid t_1,..., x_n \mid t_n)$. Operacija negacije se primenjuje na uobičajeni način.
- Neka je $F(x_1,..., x_n) = H(x_{h_1},..., x_{h_n}) \wedge G(x_{g_1},..., x_{g_n})$. Tada je:

$$F(x_1 \mid t_1,..., x_n \mid t_n) = H(x_{h_1} \mid t_{h_1},..., x_{h_n} \mid t_{h_n}) \wedge G(x_{g_1} \mid t_{g_1},..., x_{g_n} \mid t_{g_n}).$$

  Logička operacija konjukcije se primenjuje na uobičajeni način.
- Neka je $F(x_1,..., x_n) = (\forall x)(R)(G(x_1,..., x_n, x))$. Tada je:

$$F(x_1 \mid t_1,..., x_n \mid t_n) = (\forall x)(R)(G(x_1 \mid t_1,..., x_n \mid t_n, x)).$$

  Univerzalni kvantifikator se primenjuje na uobičajeni način, s obzirom na skup torki $Tuple(R)$: $F(x_1 \mid t_1,..., x_n \mid t_n) = \bigwedge_{t \in Tuple(R)} (G(x_1 \mid t_1,..., x_n \mid t_n, x \mid t))$. □

Primer 6.17. Posmatraju se formule F_1, F_2, F_3, F_4, F_5, F_6 i F iz primera 6.15 i šema baze podataka $\mathcal{S} = (S, \mathcal{I})$ iz primera 6.1. Na slici 6.10 je prikazana jedna njena pojava $rbp = \{radnik, projekat, radproj\}$. Neka je data torka $t = (\langle NAP : P1\rangle, \langle RUK : 20\rangle, \langle PRZ : Car\rangle, \langle IME : Mira\rangle) \in Tuple(NAP + RUK + PRZ + IME)$.

radnik =

	MBR	*IME*	*PRZ*	*PLT*	*SEF*
t_1	10	Mile	Car	1600	ω
t_2	20	Mira	Car	1500	10
t_3	30	Ivo	Han	1800	10
t_4	40	Ana	Tot	2800	20
t_5	50	Ana	Tot	3200	20
t_6	60	Boro	Kun	4600	ω

projekat =

	SPR	*NAP*	*RUK*
u_1	110	P1	20
u_2	120	P2	30

radproj =

	SPR	*MBR*	*BRC*
v_1	110	10	30
v_2	110	20	20
v_3	120	10	10
v_4	120	30	15

Slika 6.10.

Interpretacija formule $F(x)$ za torku t je data kao:

$$F(x \mid t) = \bigvee_{t_z \in Tuple(RP)} (F_6(x \mid t, z \mid t_z)) =$$

$$= \bigvee_{t_z \in Tuple(RP)} (t_z \in radproj \wedge t_z(MBR) = 10 \wedge F_5(x \mid t, z \mid t_z)) =$$

$$= \bigvee_{t_z \in radproj} (t_z(MBR) = 10 \wedge \bigvee_{t_w \in Tuple(P)} (F_4(x \mid t, z \mid t_z, w \mid t_w))) =$$

$$= \bigvee_{t_z \in radproj} (t_z(MBR) = 10 \wedge \bigvee_{t_w \in Tuple(P)} (t_w \in projekat \wedge t_w(SPR) = t_z(SPR) \wedge$$
$$t_w[RUK + NAP] = t[RUK + NAP] \wedge F_3(x \mid t, w \mid t_w))) =$$

$$= \bigvee_{t_z \in radproj} (t_z(MBR) = 10 \wedge \bigvee_{t_w \in projekat} (t_w(SPR) = t_z(SPR) \wedge t_w[RUK + NAP] =$$
$$t[RUK + NAP] \wedge F_3(x \mid t, w \mid t_w))) =$$

$$= \bigvee_{t_z \in radproj} (\bigvee_{t_w \in projekat} (t_z(MBR) = 10 \wedge t_w(SPR) = t_z(SPR) \wedge t_w[RUK + NAP] =$$
$$t[RUK + NAP] \wedge F_3(x \mid t, w \mid t_w))) =$$

$$= \bigvee_{t_z \in radproj} (\bigvee_{t_w \in projekat} (t_z(MBR) = 10 \wedge t_w(SPR) = t_z(SPR) \wedge t_w[RUK + NAP] =$$
$$t[RUK + NAP] \wedge \bigvee_{t_y \in Tuple(R)} (F_2(x \mid t, y \mid t_y, w \mid t_w)))) =$$

$$= \bigvee_{t_z \in radproj} \bigvee_{t_w \in projekat} \bigvee_{t_y \in radnik} (t_z(MBR) = 10 \wedge t_w(SPR) = t_z(SPR) \wedge t_w[RUK +$$
$$NAP] = t[RUK + NAP] \wedge t_y(MBR) = t_w(RUK) \wedge t_y[PRZ + IME] = t[PRZ + IME]) =$$

$$= \bigvee_{t_z \in radproj} \bigvee_{t_w \in projekat} \bigvee_{t_y \in radnik} (t_z(MBR) = 10 \wedge t_w(SPR) = t_z(SPR) \wedge t_w[RUK +$$
$$NAP] = (\langle RUK : 20\rangle, \langle NAP : P1\rangle) \wedge t_y(MBR) = t_w(RUK) \wedge t_y[PRZ + IME] =$$
$$(\langle PRZ : Car\rangle, \langle IME : Mira\rangle))) =$$

$$= v_1(MBR) = 10 \wedge u_1(SPR) = v_1(SPR) \wedge u_1[RUK + NAP] = (\langle RUK : 20\rangle, \langle NAP :$$
$$P1\rangle) \wedge t_2(MBR) = u_1(RUK) \wedge t_2[PRZ + IME] = (\langle PRZ : Car\rangle, \langle IME : Mira\rangle) \vee$$
$$\ldots = \mathsf{T} \vee \ldots = \mathsf{T}.$$

Pošto važi $F(x \mid t) = \mathsf{T}$, zaključuje se da će torka t pripadati rezultatu izraza $E = \{x(NAP + RUK + PRZ + IME) \mid F(x)\}$ iz primera 6.13.

Na sličan način, za torku $t' = (\langle NAP : P1\rangle, \langle RUK : 20\rangle, \langle PRZ : Car\rangle, \langle IME : Mile\rangle)$ se može zaključiti da je $F(x \mid t') = \perp$. □

Definicija 6.21. Relacioni račun nad torkama predstavlja strukturu

$$\mathcal{TC} = (\mathcal{U}, \mathcal{O}_\mathcal{S}, dom, \mathcal{S}, rbp, \Omega),$$

pri čemu je $\mathcal{U}$ univerzalni skup obeležja, $\mathcal{O}_\mathcal{S}$ skup domena, $dom : \mathcal{U} \to \mathcal{O}_\mathcal{S}$ domenska funkcija, $\mathcal{S}$ šema relacione baze podataka, rbp relaciona baza podataka nad $\mathcal{S}$, a $\Omega = \mathcal{R} \cup \mathcal{T} \cup \mathcal{L}_\mathcal{R}$ skup relacionih operatora, logičkih funkcija, logičkih operatora i kvantifikatora. ***Izraz relacionog računa*** $\mathcal{TC}$ je oblika:

$$\{x(R) \mid F(x)\},$$

pri čemu važi:

1. $R \subseteq \mathcal{U}$,
2. F je legalna formula nad $\mathcal{TC}$, sa jednom slobodnom promenljivom: $free(F) = \{x\}$. Ako je $type(x, F) \neq UNDEF$, onda važi da je $type(x, F) = R$. Ako je $type(x, F) = UNDEF$, tada mora važiti $men(x, F) \subseteq R$. □

Definicija 6.22. Neka je $E = \{x(R) \mid F(x)\}$ izraz relacionog računa $\mathcal{TC}$. Interpretacija (vrednost) izraza E, u oznaci $I(E)$, je skup torki:

$$I(E) = \{t \in Tuple(R) \mid F(x \mid t) = \mathsf{T}\}. \square$$

Primer 6.18. Dati su šema baze podataka $\mathcal{S}$ i baza podataka *rbp* iz primera 6.1. U nastavku će biti prikazani izrazi relacionog računa za upite **U1-U8** iz primera 6.11, nad bazom podataka *rbp*. U cilju jednostavnijeg prikaza izraza, skup obeležja šeme relacije *Radnik* će biti obeležen sa R, skup obeležja šeme relacije *Projekat* sa P, a skup obeležja šeme relacije *Radproj* sa RP.

U1. Prikazati prezimena i imena radnika koji se prezivaju '*Petrović*', a ime im započinje slovom '*M*'. Na raspolaganju je logička funkcija $subs(x, y, z) \in \mathcal{T}$, definisana u primeru 6.5.

$$\{x(PRZ + IME) \mid (\exists y)(R)(radnik(y) \wedge y(PRZ) = \text{'Petrović'} \wedge subs(y(IME), 1, \text{'M'}) \wedge y[PRZ + IME] = x[PRZ + IME])\}.$$

U2. Prikazati prezimena, imena i plate rukovodilaca (*RUK*) projekata:

$$\{x(PRZ + IME + PLT) \mid (\exists y)(R)(radnik(y) \wedge (\exists z)(P)(projekat(z) \wedge z(RUK) = y(MBR) \wedge y[PRZ + IME + PLT] = x[PRZ + IME + PLT]))\}.$$

U3. Prikazati prezimena, imena i plate radnika koji imaju veću platu od plate radnika s matičnim brojem *30*:

$$\{x(PRZ + IME + PLT) \mid (\exists y)(R)(radnik(y) \wedge y[PRZ + IME + PLT] = x[PRZ + IME + PLT] \wedge (\exists z)(R)(radnik(z) \wedge z(MBR) = 30 \wedge y(PLT) > z(PLT)))\}.$$

U4. Prikazati matične brojeve, prezimena i imena svih rukovodilaca (*SEF*):

$$\{x(MBR + PRZ + IME) \mid (\exists y)(R)(\exists z)(R)(radnik(y) \wedge radnik(z) \wedge y(MBR) = z(SEF) \wedge y[MBR + PRZ + IME] = x[MBR + PRZ + IME])\}.$$

U5. Prikazati radnike koji nisu rukovodioci ni jednog projekta:

$$\{x(R) \mid radnik(x) \wedge (\forall y)(P)(projekat(y) \wedge y(RUK) \neq x(MBR))\}.$$

U6. Prikazati matične brojeve, prezimena i imena radnika koji rade na projektu sa šifrom *110*:

$$\{x(MBR + PRZ + IME) \mid (\exists y)(R)(\exists z)(RP)(radnik(y) \wedge radproj(z) \wedge y(MBR) = z(MBR) \wedge z(SPR) = 110 \wedge y[MBR + PRZ + IME] = x[MBR + PRZ + IME])\}.$$

U7. Prikazati matične brojeve, prezimena i imena radnika koji rade na projektu na kojem radi i radnik s matičnim brojem *40*:

$$\{x(MBR + PRZ + IME) \mid (\exists y)(R)(radnik(y) \wedge y[MBR + PRZ + IME] = x[MBR + PRZ + IME] \wedge (\exists z)(RP)(radproj(z) \wedge y(MBR) = z(MBR) \wedge (\exists w)(RP)(radproj(w) \wedge w(MBR) = 40 \wedge z(SPR) = w(SPR))))\}.$$

U8. Prikazati matične brojeve, prezimena i imena radnika koji rade na svim projektima:

$$\{x(MBR + PRZ + IME) \mid (\exists y)(R)(radnik(y) \wedge y[MBR + PRZ + IME] = x[MBR + PRZ + IME] \wedge (\forall w)(P)(projekat(w) \wedge (\exists z)(RP)(radproj(z) \wedge w(SPR) = z(SPR) \wedge z(MBR) = y(MBR))))\}. \square$$

Može se primetiti da interpretacija izraza relacionog računa $I(E)$ ne mora biti konačan skup torki, odnosno relacija, što znači da algoritam za izračunavanje vrednosti izraza, konstruisan na osnovu definicija 6.20 i 6.22, neće okončati.

Primer 6.19. Dati su šema baze podataka $\mathcal{S}$ i baza podataka *rbp* iz primera 6.1. Neka je domen makar jednog obeležja iz skupa P beskonačan. Vrednosti izraza:

$$E = \{x(P) \mid \neg projekat(x)\} \text{ i}$$

$$E' = \{x(R) \mid radnik(x) \wedge (\forall y)(P)(\neg projekat(y) \wedge y(RUK) \neq x(MBR))\}$$

ne predstavljaju relacije, a algoritam koji interpretira ove izraze neće okončati! Razlog za to je činjenica da skup torki $Tuple(P) \setminus projekat$ nije konačan. $\square$

Definicija 6.23. Izraz relacionog računa $\{x(R) \mid F(x)\}$, čija interpretacija definiše konačan skup torki, predstavlja ***siguran izraz*** relacionog računa. $\square$

Primer 6.20. Izrazi E i E' iz primera 6.19 nisu sigurni. $\square$

Način da se obezbedi da svaki izraz relacionog računa bude siguran, jeste uvođenje pojma ograničene interpretacije izraza.

Definicija 6.24. Dat je proizvoljni izraz nad $\mathcal{TC}$ $E = \{x(R) \mid F_E(x)\}$. Neka je $REL(F_E) \subseteq rbp$ skup svih relacija, upotrebljenih pri formiranju formule $F_E(x)$ i neka je za svako obeležje $A \in R$, $Const(F_E, A)$ skup svih konstanti c, upotrebljenih u formuli F_E, u kontekstu obeležja A : $y(A)\ \theta\ c$, $c\ \theta\ y(A)$, ili $f_i(\lambda_1, ..., \lambda_n)$, tako da $(\exists \lambda_i,\ \lambda_j \in \{\lambda_1, ..., \lambda_n\})(\lambda_i = y(A) \wedge \lambda_j = c)$. ***Aktivni domen*** obeležja $A \in R$, s obzirom na formulu F_E, u oznaci $adom(A, F_E)$ predstavlja skup vrednosti:

$$adom(A, F_E) = Const(F_E, A) \cup \{a \in dom(A) \mid (\exists r(S) \in REL(F_E))(A \in S \wedge (\exists t \in r)(t(A) = a))\}. \square$$

Skup svih torki, definisanih aktivnim domenima svih obeležja skupa R, izraza nad $\mathcal{TC}$ $E = \{x(R) \mid F_E(x)\}$, u oznaci $Tuple_E(R)$, je skup:

$$Tuple_E(R) = \{t \in Tuple(R) \mid (\forall A \in R)(t(A) \in adom(A, F_E))\}.$$

Definicija 6.25. Neka je $E = \{x(R) \mid F(x)\}$ izraz relacionog računa $\mathcal{TC}$. ***Ograničena interpretacija*** izraza E, u oznaci $I_L(E)$, je skup torki:

$$I_L(E) = \{t \in Tuple_E(R) \mid F(x \mid t) = \mathsf{T}\}. \square$$

Definicija 6.25 zahteva izmenu poslednje tačke definicije 6.20, kojom se reguliše interpretacija univerzalnog kvantifikatora. Univerzalni kvantifikator se, u ograničenoj interpretaciji, interpretira s obzirom na skup $Tuple_E(R)$.

Teorema 6.9. Svaki izraz relacionog računa, pri ograničenoj interpretaciji, predstavlja siguran izraz.

Dokaz. Sledi neposredno iz definicije ograničene interpretacije izraza i činjenice da je skup $Tuple_E(R)$ konačan. □

6.2.2. *Kreiranje pogleda putem relacionog računa*

Pogledi se, u relacionom računu, prave putem izraza nad $\mathcal{TC}$, koji se formiraju putem ***intenzionih formula***. Definicija intenzione formule relacinog računa je identična definiciji formule relacionog računa (definicija 6.18). Definicija atomarne intenzione formule relacionog računa se razlikuje od definicije atomarne formule relacionog računa (definicija 6.17) samo u prvoj tački, koja u slučaju atomarne intenzione formule ima oblik: "atomarna intenziona formula je izraz $R_i(x)$, pri čemu je R_i oznaka šeme relacije, a $x \in \mathcal{V}$. Kao oznaka šeme relacije se može koristiti skup obeležja šeme relacije (saglasno pretpostavci o postojanju šeme univerzalne relacije), ili naziv šeme relacije".

Definicija 6.26. Neka je $E = \{x(R) \mid F(x)\}$ izraz relacionog računa, pri čemu je F intenziona formula nad $\mathcal{TC}$, a $SR(F)$ skup šema relacija koji se pojavljuje u formuli F. ***Pogled*** nad izrazom relacionog računa E predstavlja šemu relacije $E(R, \mathcal{F})$, pri čemu je $\mathcal{F}$ skup funkcionalnih zavisnosti, dat kao:

$$\mathcal{F} = \left(\bigcup_{(R_i, \mathcal{K}_i) \in SR(F)} \{K \rightarrow A \mid K \in \mathcal{K}_i \wedge A \in R_i\} \right)_R . \square$$

Primer 6.21. Neka je data šema baze podataka $\mathcal{S}$ iz primera 6.1. Neka je sa R obeležen skup obeležja šeme relacije *Radnik*, sa P skup obeležja šeme relacije *Projekat*, a sa RP skup obeležja šeme relacije *Radproj*. Pogled: "radnici koji rade na svim projektima" se formira putem izraza relacionog računa:

$$RSP = \{x(R) \mid R(x) \wedge (\forall w)(P)(P(w) \wedge (\exists z)(RP)(RP(z) \wedge w(SPR) = z(SPR) \wedge z(MBR) = x(MBR)))\}$$

i ima oblik $RSP(\{MBR, IME, PRZ, PLT, SEF\}, \{MBR \rightarrow IME, MBR \rightarrow PRZ, MBR \rightarrow PLT, MBR \rightarrow SEF\})$, što je, u ovom slučaju, ekvivalentno šemi relacije kod koje je skup ograničenja izražen putem ključeva: $RSP(\{MBR, IME, PRZ, PLT, SEF\}, \{MBR\})$. □

6.3. *Ekvivalentnost relacione algebre i relacionog računa*

U ovoj tački će biti pokazano da je ekspresivna moć relacione algebre i relacionog računa nad torkama ista.

Teorema 6.10. Za svaki ekstenzioni izraz $E_{\mathcal{A}}$ relacione algebre $\mathcal{A}_r = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega, \mathcal{O})$ postoji ekvivalentni izraz E_C relacionog računa nad torkama $\mathcal{TC} = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega)$.

Dokaz. Dovoljno je dokazati da za svaki izraz $E_{\mathcal{A}}$ generativnog dela relacione algebre $\mathcal{G}(\mathcal{A}_r)$ postoji ekvivalentni izraz relacionog računa nad torkama E_C. Dokaz će biti sproveden metodom indukcije po dužini ekstenzionog izraza relacione algebre.

Baza indukcije. $E_{\mathcal{A}}$ je ekstenzioni izraz nad $\mathcal{G}(\mathcal{A}_r)$, dobijen primenom pravila 5. definicije 6.14, najviše jedanput, što znači da $E_{\mathcal{A}}$ ne sadrži operatore relacione algebre.

- Neka je $E_{\mathcal{A}} = <A : a>$. Ekvivalentni izraz nad $\mathcal{TC}$ je:

$$E_C = \{x(A) \mid x(A) = a\}.$$

- Neka je $E_{\mathcal{A}} = r_i$, pri čemu je $r_i(R_i) \in rbp$. Ekvivalentni izraz nad $\mathcal{TC}$ je:

$$E_C = \{x(R_i) \mid r_i(x)\}.$$

Pretpostavka indukcije. Za proizvoljne izraze $G_{\mathcal{A}}$ i $H_{\mathcal{A}}$ nad $\mathcal{G}(\mathcal{A}_r)$, dobijene primenom pravila 5. definicije 6.14, najviše k puta (k je prirodan broj), postoje ekvivalentni izrazi nad $\mathcal{TC}$, redom, $G_C = \{x(R) \mid F_G(x)\}$ i $H_C = \{x(S) \mid F_H(x)\}$, pri čemu je R skup obeležja definisan izrazom $G_{\mathcal{A}}$, a S skup obeležja definisan izrazom $H_{\mathcal{A}}$.

Dokaz indukcije. Neka je izraz $E_{\mathcal{A}}$ nad $\mathcal{G}(\mathcal{A}_r)$, dobijen primenom pravila 5. definicije 6.14, $k + 1$ puta.

- Neka je $E_{\mathcal{A}} = \delta_{X \leftarrow Y}(G_{\mathcal{A}})$, pri čemu je $X = A_1 \ldots A_n$ i $Y = B_1 \ldots B_n$. Ekvivalentni izraz nad $\mathcal{TC}$ je:

$$E_C = \{x((R \setminus X)Y) \mid (\exists y)(R)(F_G(y) \wedge x[R \setminus X] = y[R \setminus X] \wedge x(B_1) = y(A_1) \wedge \ldots \wedge x(B_n) = y(A_n))\}.$$

- Neka je $E_{\mathcal{A}} = \sigma_{A = a}(G_{\mathcal{A}})$, $E_{\mathcal{A}} = \sigma_{A = B}(G_{\mathcal{A}})$, ili $E_{\mathcal{A}} = \sigma_{f_i(x_1, \ldots, x_n)}(G_{\mathcal{A}})$. Ekvivalentni izrazi nad $\mathcal{TC}$ su, redom:

$$E_C = \{x(R) \mid F_G(x) \wedge x(A) = a\}.$$

$$E_C = \{x(R) \mid F_G(x) \wedge x(A) = x(B)\},$$

$$E_C = \{x(R) \mid F_G(x) \wedge f_i(\zeta_1, \ldots, \zeta_n)\}.$$

pri čemu je:

$$(\forall j \in \{1,\ldots, n\})\left(\zeta_j = \begin{cases} x(A), & x_j = A \\ x_j, & x_j \in Dom \end{cases}\right).$$

- Neka je $E_{\mathcal{A}} = \pi_X(G_{\mathcal{A}})$. Ekvivalentni izraz nad $\mathcal{TC}$ je:

$$E_C = \{x(X) \mid (\exists v)(R)(F_G(v) \wedge x[X] = v[X])\}.$$

- Neka je $E_{\mathcal{A}} = G_{\mathcal{A}} \triangleright\triangleleft H_{\mathcal{A}}$. Ekvivalentni izraz nad $\mathcal{TC}$ je:

$$E_C = \{x(RS) \mid (\exists v)(R)(\exists z)(S)(F_G(v) \wedge F_H(z) \wedge x[R] = v[R] \wedge x[S] = z[S])\}.$$

- Neka je $E_{\mathcal{A}} = G_{\mathcal{A}} \cup H_{\mathcal{A}}$. Ekvivalentni izraz nad $\mathcal{TC}$ je:

$$E_C = \{x(R) \mid F_G(x) \vee F_H(x)\}.$$

- Neka je $E_{\mathcal{A}} = G_{\mathcal{A}} - H_{\mathcal{A}}$. Ekvivalentni izraz nad $\mathcal{TC}$ je:

$$E_C = \{x(R) \mid F_G(x) \wedge \neg F_H(x)\}.$$

Na osnovu primene principa indukcije, zaključuje se da za svaki izraz relacione algebre postoji odgovarajući izraz relacionog računa. □

Teorema 6.11. Za svaki izraz $E_C = \{x(R) \mid F(x)\}$ relacionog računa nad torkama $\mathcal{TC} = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega)$ sa ograničenom interpretacijom, postoji ekvivalentni ekstenzioni izraz $E_{\mathcal{A}}$ relacione algebre $\mathcal{A}_r = (\mathcal{U}, \mathcal{O}_S, dom, \mathcal{S}, rbp, \Omega, \mathcal{O})$.

Dokaz. Dokaz će biti sproveden metodom indukcije po dužini formule izraza relacionog računa nad torkama.

Baza indukcije. E_C je izraz nad $\mathcal{TC}$ sa ograničenom interpretacijom, čija formula je dobijena primenom pravila 3, definicije 6.18, najviše jedanput, što znači da je formula izraza E_C atomarna.

- Neka je $E_C = \{x(R_i) \mid r_i(x)\}$, tako da je $r_i \in rbp$. Ekvivalentni izraz nad $\mathcal{A}_r$ je:

$$E_{\mathcal{A}} = r_i.$$

- Neka je $E_C = \{x(A) \mid x(A)\ \theta\ c\}$, $A \in \mathcal{U}$, $c \in dom(A)$ i $\theta \in \mathcal{R}$. Ekvivalentni izraz nad $\mathcal{A}_r$ je:

$$E_{\mathcal{A}} = \sigma_{A\,\theta\,c}(r_A),$$

pri čemu je r_A konstantna relacija, takva da $r_A = Tuple_{E_C}(A)$.

- Neka je $E_C = \{x(A) \mid f_i(c_1,\ldots, c_k, x(A), c_{k+1},\ldots, c_n)\}$, $k \geq 0$, $n \geq k$, $f_i \in \mathcal{T}$, $A \in \mathcal{U}$, $((\forall i \in \{1,\ldots, n\}))(c_i \in Dom)$. Ekvivalentni izraz nad $\mathcal{A}_r$ je:

$$E_{\mathcal{A}} = \sigma_{f_i(c_1,\ldots,c_k,A,c_{k+1},\ldots,c_n)}(r_A),$$

pri čemu je $r_A = Tuple_{E_C}(A)$.

U svim ostalim slučajevima atomarnih formula, E_C nije izraz relacionog računa $\mathcal{TC}$.

Pretpostavka indukcije. $G_C = \{y(V) \mid F_G(y)\}$ i $H_C = \{z(W) \mid F_H(z)\}$ su proizvoljni izrazi nad $\mathcal{UC}$ sa ograničenom interpretacijom, čija formula je dobijena primenom pravila 3. definicije 6.18. najviše k puta (k je prirodan broj). Ekvivalentni izrazi nad $\mathcal{A}$, su, redom, $G_{\mathcal{A}}$ i $H_{\mathcal{A}}$.

Dokaz indukcije. $E_C = \{x(R) \mid F_E(x)\}$ je proizvoljan izraz nad $\mathcal{UC}$ sa ograničenom interpretacijom, takav da je $F_E(x)$ dobijena primenom pravila 3. definicije 6.18, $k + 1$ puta.

- Neka je $E_C = \{x(R) \mid \neg F_G(x)\}$, $R = V$. Ekvivalentni izraz nad $\mathcal{A}$, je:

$$E_{\mathcal{A}} = Tuple_{E_C}(R) - G_{\mathcal{A}}.$$

- Neka je $E_C = \{x(R) \mid F_G(x) \wedge F_H(x)\}$, $R = V = W$. Ekvivalentni izraz nad $\mathcal{A}$, je:

$$E_{\mathcal{A}} = G_{\mathcal{A}} \cap H_{\mathcal{A}}.$$

- Neka je $E_C = \{x(R) \mid (\forall z)(W)(F'_E(x, z))\}$, $R = V$, pri čemu se $F'_E(x, z)$ može prikazati u obliku konjukcije tri formule: $F'_E(x, z) = F_G(x) \wedge F_H(z) \wedge F''_E(x, z)$, tako da formula $F''_E(x, z)$ ne može da se razloži na nezavisne potformule po promenljivama x i z. Saglasno definicijama 6.17. i 6.18, a imajući u vidu činjenicu da $F''_E(x, z)$ ne može da se razloži na konjukciju nezavisnih formula po promenljivoj x, ili z, $F''_E(x, z)$ se može prikazati u obliku konjuktivne normalne forme:

$$F''_E(x, z) = \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{m_i} \xi_{ij}(x, z) \right), \tag{6.1}$$

pri čemu važi da je:

$$(\forall i \in \{1, \ldots, k\})(\forall j \in \{1, \ldots, m_i\})(\xi_{ij}(x, z) = x(A)\ \theta\ z(B) \vee \xi_{ij}(x, z) = \neg(x(A)\ \theta\ z(B)) \vee \xi_{ij}(x, z) = f(\lambda_1, \ldots, \lambda_p) \vee \xi_{ij}(x, z) = \neg f(\lambda_1, \ldots, \lambda_p)),$$

tako da je $A \in R$, $B \in W$ i $(\exists \lambda_k, \lambda_l \in \{\lambda_1, \ldots, \lambda_p\})(\lambda_k = x(A) \wedge \lambda_l = z(B))$. Neka je $W = B_1 \ldots B_n$. Definiše se funkcija preimenovanja obeležja h: $W \rightarrow (W \setminus R)Y$, pri čemu je $Y = C_1 \ldots C_n$ i $(\forall i \in \{1, \ldots, n\})(C_i \notin RW)$, na sledeći način:

$$h(B_i) = \begin{cases} B_i, & B_i \notin R \cap W \\ C_i, & B_i \in R \cap W \end{cases}.$$

Izraz (6.1) se transformiše u selekcionu formulu relacione algebre:

$$F''_{xz} = \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{m_i} \zeta_{ij}(x, z) \right), \tag{6.2}$$

takvu da za $(\forall i \in \{1, \ldots, k\})(\forall j \in \{1, \ldots, m_i\})$ važi:

$$\zeta_{ij}(x,z)=\begin{cases} A\,\theta\ h(B), & \xi_{ij}(x,z)=x(A)\,\theta\ z(B)\\ \neg(A\,\theta\ h(B)), & \xi_{ij}(x,z)=\neg(x(A)\,\theta\ z(B))\\ f(\nu_1,\ldots,\nu_p), & \xi_{ij}(x,z)=f(\lambda_1,\ldots,\lambda_p)\\ \neg f(\nu_1,\ldots,\nu_p), & \xi_{ij}(x,z)=\neg f(\lambda_1,\ldots,\lambda_p)\end{cases} \text{ i}$$

$$(\forall \nu_l \in \{\nu_1,\ldots,\nu_p\})(\nu_l=\begin{cases} A, & \lambda_l=x(A)\\ h(B), & \lambda_l=z(B)\\ \lambda_l, & \lambda_l\neq x(A)\wedge\lambda_l\neq z(B)\end{cases}).$$

Pošto su, po pretpostavci indukcije, $G_{\mathcal{A}}$ i $H_{\mathcal{A}}$ konačni skupovi torki nad $\mathcal{A}_r$, koji odgovaraju, redom, izrazima nad $\mathcal{TC}$ $G_C = \{x(R) \mid F_G(x)\}$ i $H_C = \{z(W) \mid F_H(z)\}$, ekvivalentni izraz izraza $E_C = \{x(R) \mid (\forall z)(W)(F_G(x) \wedge F_H(z) \wedge F''_E(x, z))\}$ nad $\mathcal{A}_r$ je:

$$(G_{\mathcal{A}}\,[F''_{xz}]\,\delta_{R\cap W\leftarrow Y}(H_{\mathcal{A}}))\div\delta_{R\cap W\leftarrow Y}(H_{\mathcal{A}}),$$

gde je selekciona formula teta spajanja F''_{xz} data izrazom (6.2).

- Neka je $E_C = \{x(R) \mid (\forall z)(W)(F'_E(x, z))\}$, pri čemu se $F'_E(x, z)$ može prikazati u obliku konjukcije dve formule: $F'_E(x, z) = F_H(z) \wedge F''_E(x, z)$, tako da formula $F''_E(x, z)$ ne može da se razloži na nezavisne potformule, po promenljivoj x i promenljivama x i z. Formula $F''_E(x, z)$ se, prema definicijama (6.17) i (6.18) može prikazati putem izraza (6.1) i može se (saglasno analizi datoj u trećem slučaju dokaza indukcije) transformisati u selekcionu formulu F''_{xz} relacione algebre (6.2). Pošto je, po pretpostavci indukcije, $H_{\mathcal{A}}$ konačan skup torki, koji odgovara izrazu $H_C = \{z(W) \mid F_H(z)\}$, ekvivalentni izraz izraza $E_C = \{x(R) \mid (\forall z)(W)(F_H(z) \wedge F''_E(x, z))\}$ nad $\mathcal{A}_r$ je:

$$(Tuple_{G_C}(R)\,[F''_{xz}]\,\delta_{R\cap W\leftarrow Y}(H_{\mathcal{A}}))\div\delta_{R\cap W\leftarrow Y}(H_{\mathcal{A}}).$$

- Neka je $E_C = \{x(R) \mid (\forall z)(W)(F'_E(x, z))\}$, pri čemu se $F'_E(x, z)$ može prikazati u obliku konjukcije dve formule: $F'_E(x, z) = F_G(x) \wedge F''_E(x, z)$, tako da formula $F''_E(x, z)$ ne može da se razloži na nezavisne potformule, po promenljivoj z i promenljivama x i z. Formula $F''_E(x, z)$ se, prema definicijama (6.17) i (6.18) može prikazati putem izraza (6.1) i može se (saglasno analizi datoj u trećem slučaju dokaza indukcije) transformisati u selekcionu formulu F''_{xz} relacione algebre (6.2). Pošto je, po pretpostavci indukcije, $G_{\mathcal{A}}$ konačan skup torki koji odgovara izrazu $G_C = \{x(R) \mid F_G(x)\}$, ekvivalentni izraz izraza $E_C = \{x(R) \mid (\forall z)(W)(F_G(x) \wedge F''_E(x, z))\}$ nad $\mathcal{A}_r$ je:

$$(G_{\mathcal{A}}\,[F''_{xz}]\,\delta_{R\cap W\leftarrow Y}(Tuple_{H_C}(W)))\div\delta_{R\cap W\leftarrow Y}(Tuple_{H_C}(W)).$$

- Neka je $E_C = \{x(R) \mid (\forall z)(W)(F'_E(x, z))\}$, pri čemu se $F'_E(x, z)$ ne može prikazati u obliku konjukcije nezavisnih potformula po x, z i x i z. Formula $F'_E(x, z)$ se, prema definicijama (6.17) i (6.18) može prikazati putem izraza (6.1) i može se (saglasno analizi datoj u prethodnom slučaju) transformisati u selekcionu formulu F'_{xz} relacione algebre (6.2). Ekvivalentni izraz izraza E_C nad $\mathcal{A}_r$ je:

$$(Tuple_{G_C}(R)\ [F'_{xz}]\ \delta_{R\cap W\leftarrow Y}(Tuple_{H_C}(W))) \div \delta_{R\cap W\leftarrow Y}(Tuple_{H_C}(W)).$$

Na osnovu primene principa indukcije, zaključuje se da za svaki siguran izraz relacionog računa postoji odgovarajući izraz relacione algebre. □

Saglasno teoremama 6.10 i 6.11 zaključuje se da je ekspresivna moć relacione algebre i relacionog računa nad torkama sa ograničenom interpretacijom, jednaka.

6.4. *Ažuriranje baze podataka*

Ažuriranje baze podataka predstavlja proces usaglašavanja stanja baze podataka sa stanjem realnog sistema. Ažuriranje relacione baze podataka predstavlja niz operacija upisa, brisanja i modifikovanja torki odgovarajućih relacija, pri čemu treba obezbediti da baza podataka, nakon obavljenog ažuriranja, ostane u konzistentnom stanju. Pre definicija osnovnih operacija ažuriranja baze podataka, uvodi se pojam lokalnog i globalnog zadovoljenja skupa ograničenja nad šemom baze podataka.

Definicija 6.27. Neka je *rbp* skup relacija nad šemom baze podataka $\mathcal{S} = (S, \mathcal{I})$. Kaže se da je *rbp* ***lokalno konzistentan*** skup s obzirom na S, ako je za svaku relaciju $r \in rbp$, formiranu nad šemom $(R_i, C_i) \in S$, zadovoljen lokalni skup ograničenja C_i. □

Definicija 6.28. Neka je *rbp* skup relacija nad šemom baze podataka $\mathcal{S} = (S, \mathcal{I})$. Kaže se da je *rbp* ***globalno konzistentan*** skup s obzirom na $\mathcal{S}$ (***pojava*** nad $\mathcal{S}$), ako važi da *rbp* globalno zadovoljava sva ograničenja iz skupa:

$$C_G = \mathcal{I} \cup \left(\bigcup_{(R_i, C_i) \in S} C_i \right). \square$$

Definicija 6.29. Upis (***dodavanje***) nove torke t u relaciju $r \in SAT(R, C)$*), gde je R skup obeležja, a C lokalni skup ograničenja šeme relacije $(R, C) \in S$, predstavlja funkciju:

$$ADD: SAT(R, C) \times Tuple(R) \rightarrow SAT(R, C),$$

definisanu na sledeći način:

$$ADD(r, t[R]) = r \cup \{t[R]\}. \square$$

Operacija dodavanja nove torke, saglasno definisanom preslikavanju ADD: $SAT(R, C) \times Tuple(R) \rightarrow SAT(R, C)$, obezbeduje da nakon izvršenog upisa torke $t \in Tuple(R)$ u relaciju $r \in SAT(R, C)$, r ostane u konzistentnom stanju. Drugim rečima, zabranjen je svaki pokušaj upisa torke u relaciju r, kojim bi se narušio lokalni skup ograničenja C.

*) Sa $SAT(R, C)$ je označen skup svih relacija iz $SAT(R)$, takvih da zadovoljavaju sva ograničenja, zadata lokalnim skupom ograničenja C.

Definicija 6.30. Brisanje postojeće torke t iz relacije $r \in SAT(R, C)$, gde je $\mathcal{K}$ skup ključeva šeme relacije $(R, C) \in S$ (pri čemu važi $C \models \mathcal{K}$), a $K \in \mathcal{K}$ jedan ključ, predstavlja funkciju:

$$DEL: SAT(R, C) \times Tuple(K) \rightarrow SAT(R, C),$$

definisanu na sledeći način:

$$DEL(r, t[K]) = r - \{t' \in r \mid t'[K] = t[K]\}. \square$$

Prilikom primene operacije *DEL*, torka koju treba izbrisati se identifikuje putem zadate vrednosti ključa $K \in \mathcal{K}$. Kao i u slučaju upisa, preslikavanjem $DEL: SAT(R, C) \times Tuple(K) \rightarrow SAT(R, C)$, dozvoljena je primena operacije brisanja samo u onim slučajevima kada se ne narušavaju ograničenja iz skupa C.

Definicija 6.31. Modifikacija podataka torke t, relacije $r \in SAT(R, C)$, pri čemu je

$$\hat{\mathcal{K}} = \bigcup_{K_i \in \mathcal{K}} K_i$$

skup primarnih obeležja, a $K \in \mathcal{K}$ jedan ključ iz skupa ključeva, šeme relacije $(R, C) \in S$, predstavlja funkciju:

$$MOD: SAT(R, C) \times Tuple(K) \times Tuple(X) \rightarrow SAT(R, C),$$

pri čemu je $X \subseteq R \setminus \hat{\mathcal{K}}$, definisanu na sledeći način:

$$MOD(r, t_1[K], t_2[X]) = r - \{t' \in r \mid t'[K] = t_1[K]\} \cup \{t'' \in Tuple(R) \mid t''[R \setminus X] = t_1[R \setminus X] \wedge t''[X] = t_2[X]\}. \square$$

Pri primeni operacije *MOD*, torka koju treba modifikovati se identifikuje putem zadate vrednosti ključa $K \in \mathcal{K}$. Modifikacija podataka torke zabranjuje promenu vrednosti ključa šeme relacije, u cilju izbegavanja problema vezanih za obezbeđenje globalne konzistentnosti baze podataka. Modifikacija se, u slučaju promene vrednosti nekog obeležja ključa, realizuje putem kompozicije operacija *DEL* i *ADD*. Preslikavanje $MOD: SAT(R, C) \times Tuple(K) \times Tuple(X) \rightarrow SAT(R, C)$ obezbeđuje lokalnu konzistentnost relacije nad šemom (R, C).

Primer 6.22. Data je šema baze podataka $\mathcal{S} = (S, \mathcal{I})$ iz primera 6.1 i pojava $rbp = \{radnik, projekat, radproj\}$, prikazana na slici 6.10.

- Operacija upisa nove torke $ADD(radnik, t)$, pri čemu je $t = (\langle MBR : 20\rangle, \langle PRZ : Car\rangle, \langle IME : Ana\rangle, \langle PLT : 1100\rangle, \langle SEF : 10\rangle)$ nije legalna, jer se narušava lokalno ograničenje ključa šeme relacije *Radnik*.
- Operacija $DEL(radnik, (\langle MBR : 60\rangle))$ obezbeduje brisanje torke, identifikovane pomoću vrednosti ključa $MBR = 60$, pri čemu lokalni skup ograničenja ovom operacijom nije narušen.
- Operacija $MOD(radnik, (\langle MBR : 10\rangle), (\langle PRZ : Car\rangle, \langle IME : Mile\rangle, \langle PLT : 1800\rangle, \langle SEF : \omega\rangle))$ obezbeđuje modifikaciju podataka torke relacije *radnik*, identifikovane putem vrednosti ključa $MBR = 10$ i ne narušava lokalni skup ograničenja. $\square$

Pojam transakcije

Operacijama *ADD*, *DEL* i *MOD* je obezbeđena lokalna konzistentnost relacija, nad odgovarajućim šemama. Globalna konzistentnost baze podataka, međutim, može biti narušena pojedinačnom primenom operacija ažuriranja, ukoliko nije predviđeno sprovođenje kontrole očuvanja međurelacionih ograničenja (kao što su: referencijalni integriteti, zavisnosti sadržavanja i pravila poslovanja). Pored toga, priroda informacionih zahteva korisnika je takva da se oni često realizuju putem niza operacija upita nad bazom podataka i ažuriranja baze podataka.

Zbog prethodno navedenih razloga, skupovi operacija upita i ažuriranja baze podataka, kojima se obezbeđuje ispunjenje jednog korisničkog zahteva, organizuju se u jedinice obrade podataka, koje se nazivaju transakcije.

Definicija 6.32. Data je šema baze podataka $\mathcal{S} = (S, \mathcal{I})$ i njena pojava *rbp*. ***Transakcija*** nad bazom podataka *rbp* predstavlja takav niz operacija $T = (op_1, \ldots, op_n)$, $n \geq 1$, pri čemu je $(\forall i \in \{1, \ldots, n\})(op_i \in \{ADD, DEL, MOD, SEL\})$ (operacije *ADD*, *DEL* i *MOD* su date, redom, definicijama 6.27, 6.28 i 6.29, a *SEL* predstavlja izraz upitnog jezika nad datom bazom podataka), koji se u celosti izvršava ili u celosti ne izvršava nad bazom podataka i ima osobinu da, pri uspešnom završetku, prevodi bazu podataka iz jednog u drugo (ne nužno različito) globalno konzistentno stanje. □

Neka uspešno izvršena transakcija $T = (op_1, \ldots, op_n)$ prevodi pojavu *rbp* nad šemom $\mathcal{S} = (S, \mathcal{I})$, u skup relacija *rbp'*, što će biti označeno na sledeći način: $rbp' = T(rbp)$. Pri tome se postupak $T(rbp) = (op_1, \ldots, op_n)(rbp)$ obavlja po sledećim pravilima:

- $rbp_1 = rbp$,
- $(\forall i \in \{1, \ldots, n-1\})((op_i, \ldots, op_n)(rbp_i) = (op_{i+1}, \ldots, op_n)(op_i(rbp_i)))$ i
- $(\forall i \in \{1, \ldots, n\})(rbp_{i+1} = op_i(rbp_i))$, gde je

$$op_i(rbp_i) = \begin{cases} (rbp_i \setminus \{r\}) \cup \{ADD(r, t[R])\}, & op_i = ADD \\ (rbp_i \setminus \{r\}) \cup \{DEL(r, t[K])\}, & op_i = DEL \\ (rbp_i \setminus \{r\}) \cup \{MOD(r, t_1[K], t_2[R \setminus K])\}, & op_i = MOD \end{cases}.$$

Definicijom 6.32 se garantuje globalna konzistentnost baze podataka, što znači da *rbp'*, u slučaju uspešnog okončanja transakcije, predstavlja pojavu nad $\mathcal{S}$, odnosno da zadovoljava sva ograničenja iz C_G.

Primer 6.23. Data je šema baze podataka $\mathcal{S} = (S, \mathcal{I})$ iz primera 6.1 i pojava $rbp = \{radnik, projekat, radproj\}$, prikazana slici 6.10. Neka se posmatra transakcija koja se sastoji iz samo jedne operacije: $T = (op_1) = (DEL(radnik, (<MBR : 10>)))$. Pri tome je $T(rbp) = op_1(rbp) = (rbp \setminus \{radnik\}) \cup \{DEL(radnik, (<MBR : 10>)\}$. $T(rbp)$, međutim, ne predstavlja pojavu nad šemom $\mathcal{S}$, jer operacija $DEL(radnik, (<MBR : 10>)$ narušava referencijalne integritete $Radproj[MBR] \subseteq Radnik[MBR]$ i $Radnik[SEF] \subseteq Radnik[MBR]$, što za posledicu ima da transakcija T ne može biti uspešno završena. □

Kada se posmatra konkretan sistem za upravljanje bazama podataka i konkretan operativni sistem, onda se transakcija poistovećuje sa jednim procesom (tj. jednim izvršenjem aplikativnog programa) nad bazom podataka.

Primer 6.24. Data je šema relacije *Rezervacije*({*BRLET, DATUM, VREME, BRSLSD*}, {*BRLET* + *DATUM*}), sa sledećim značenjem obeležja: *BRLET* - broj leta aviokompanije, *DATUM* - datum letenja, *VREME* - vreme poletanja i *BRSLSD* - broj slobodnih sedišta.

Na slici 6.11 je prikazan pseudokod programa transakcije kojom se obavlja rezervacija mesta za konkretni let. Uspešnost transakcije se potvrđuje naredbom ***POTVRDI_TRANS***, dok se neuspešna transakcija poništava naredbom ***PONIŠTI_TRANS***. □

```
TRANSAKCIJA rezervacija_mesta
   Ulaz: :brlet          (* broj leta za koji se zahteva rezervacija *)
         :datum          (* datum, za koji se zahteva rezervacija *)
         :N              (* traženi broj rezervacija *)
   Izlaz: ind ∈ {0, 1, 2}  (* indikator uspešnosti transakcije *)
POČETAK_TRANSAKCIJE rezervacija_mesta
   SEL E←{x(R) | rezervacije(x) ∧ x(BRLET) = :brlet ∧ x(DATUM) = :datum)}
   (* R je skup obeležja šeme relacije Rezervacije *)
   AKO E = ∅ TADA
      POSTAVI ind ← 1                  (* Ne postoji traženi let zadatog datuma *)
      PONIŠTI_TRANS                    (* Transakcija se poništava *)
   INAČE
      AKO x(BRSLSD) ≥ :N TADA
         MOD (rezervacije, (:brlet, :datum), (x(VREME), x(BRSLSD) - :N))
         POSTAVI ind←0                 (* Transakcija uspešno okončana *)
         POTVRDI_TRANS                 (* Transakcija se potvrđuje kao uspešna *)
      INAČE
         POSTAVI ind←2                 (* Ne postoji traženi broj rezervacija *)
         PONIŠTI_TRANS                 (* Transakcija se poništava *)
      KRAJ AKO
   KRAJ AKO
KRAJ_TRANSAKCIJE rezervacija_mesta
```

Slika 6.11.

Naredba *POTVRDI_TRANS* (*COMMIT*) šalje informaciju sistemu za upravljanje bazama podataka da je transakcija, sa stanovišta krajnjeg korisnika, uspešno obavljena. Naredba *PONIŠTI_TRANS* (*ROLLBACK*) šalje informaciju sistemu za upravljanje bazama podataka da transakcija nije uspešno obavljena, što znači da efekti izvršenja transakcije u celosti moraju biti poništeni - baza podataka se ostavlja u konzistentnom stanju koje je važilo pre početka izvršenja transakcije.

Transakcija, čije su sve operacije tipa *SEL*, se naziva ***upitna transakcija***. U daljem tekstu, podrazumevaće se da je transakcija upitna, samo ako to bude posebno naglašeno.

Primer 6.25. Date su šeme relacija *Zagl_porudžbine*({*BRPOR*, *DATUM*, *DOBAV*}, {*BRPOR*}) i *Stavka_por*({*BRPOR*, *BRSTAV*, *ART*, *KOL*}, {*BRPOR* + *ART*}), kao i međurelaciona ograničenja:

$$Stavka\_por[BRPOR] \subseteq Zagl\_porudžbine[BRPOR] \text{ i}$$

$$Zagl\_porudžbine[BRPOR] \subseteq Stavka\_por[BRPOR],$$

pri čemu su: *BRPOR* - broj porudžbine, *DATUM* - datum izdavanja porudžbine, *DOBAV* - oznaka dobavljača kojem se šalje porudžbina, *ART* - oznaka artikla koji se poručuje i *KOL* - količina poručenog artikla.

Transakcija, kojom se kreira nova porudžbina, sastoji se iz jedne *ADD* operacije, kojom se upisuje torka u relaciju *zagl_porudžbine* i niza *ADD* operacija, kojima se, za svaku stavku porudžbine, upisuje po jedna torka u relaciju *stavka_por*. Ukoliko prilikom izvršenja makar jedne *ADD* operacije transakcije dođe do greške (pri čemu je greška i to da ne postoji ni jedna *ADD* operacija za dodavanje torke u relaciju *stavka_por*), transakcija, kao celina, ne sme biti izvršena. □

Jedan od bitnih zahteva koji se postavljaju pred savremeni sistem za upravljanje bazama podataka jeste obezbeđenje mogućnosti transakcione obrade podataka u uslovima višekorisničkog režima rada. Ovom pitanju je posvećen Prilog 2, pod nazivom *Višekorisnički režim transakcione obrade podataka*.

6.5. SQL - jezik relacionih sistema za upravljanje bazama podataka

SQL (Structured Query Language - strukturirani upitni jezik) predstavlja standardni*) jezik relacionih sistema za upravljanje bazama podataka. Nastao je 1974. godine u Istraživačkoj laboratoriji IBM-a. To je jezik visokog nivoa deklarativnosti, koji objedinjuje funkcije jezika za definiciju podataka, jezika za manipulaciju podacima i upitnog jezika. Pomoću SQL-a se obavlja celokupna komunikacija na relaciji sistem za upravljanje bazama podataka - korisnik sistema za upravljanje bazama podataka.

6.5.1. Namena i zadaci SQL-a u okviru sistema za upravljanje bazama podataka

SQL je ***namenjen*** sledećim tipovima korisnika sistema za upravljanje bazama podataka:

- ***administratorima baze podataka*** (DBA), za obavljanje poslova administracije,

*) Sintaksa SQL-a predstavlja kompromis najuticajnijih proizvođača relacionih sistema za upravljanje bazama podataka.

- *programerima* (PG), za izradu aplikacija nad bazom podataka i
- *krajnjim korisnicima* (KK), za postavljanje upita nad bazom podataka.

SQL se javlja u formama:

- *interaktivnog jezika* sistema za upravljanje bazama podataka (ISQL),
- *ugrađenog jezika* u jezik III generacije, kao što su C, COBOL, FORTRAN, Oracle*PL/SQL, itd. (ESQL) i
- *sastavnog dela jezika IV generacije*, kao što su Oracle/SQL*Forms, Ingres/4GL, itd. (FSQL).

U tabeli na slici 6.12 su prikazane potrebe tipova korisnika, s obzirom na navedene pojavne forme SQL-a. Oznaka *I* ima značenje "intezivno upotrebljava", *P* ima značenje "povremeno upotrebljava", a *R* "retko upotrebljava". Prazna polja tabele znače da dati tip korisnika ne upotrebljava navedenu formu SQL-a.

	ISQL	ESQL	FSQL
DBA	*I*	*P*	*P*
PG	*P*	*I*	*I*
KK	*R*		

Slika 6.12.

Saglasno nameni i vrstama korisnika koji ga upotrebljavaju, SQL obezbeduje realizaciju sledećih zadataka:

1. izražavanje upita putem upitnog jezika (naredba SELECT),
2. ažuriranje baze podataka putem jezika za manipulaciju podacima (naredbe INSERT, DELETE i UPDATE),
3. realizacija implementacione šeme baze podataka i definisanje fizičke organizacije baze podataka (naredbe CREATE, DROP, ALTER),
4. realizacija pogleda (naredbe CREATE VIEW, DROP VIEW),
5. automatsko održavanje rečnika podataka,
6. transakcijska obrada podataka (naredbe COMMIT, ROLLBACK, SAVEPOINT),
7. zaključavanje resursa (naredba LOCK TABLE),
8. zaštita podataka od neovlašćenog pristupa (naredbe GRANT, REVOKE),
9. praćenje zauzeća resursa i performansi rada sistema za upravljanje bazama podataka (naredbe AUDIT; EXPLAIN PLAN),
10. obezbeđenje proceduralnog načina obrade podataka "slog po slog" (naredbe za rad s kursorom: OPEN, FETCH i CLOSE).

Treba naglasiti da iako se za SQL kaže da je standardni jezik relacionih sistema za upravljanje bazama podataka, njegova sintaksa nije u potpunosti standardizovana i u ne maloj meri zavisi od proizvođača sistema za upravljanje bazama podataka. Viši stepen standardizacije je postignut u onim delovima SQL-a, koji se odnose na obavljanje funkcija 1, 2, 4, 6 i 8 (pri čemu su funkcije 1. i 2. visoko standardizovane samo u domenu interaktivnog SQL-a). Nivo standardizacije 3. funkcije je vrlo nizak i odnosi se samo na osnovnu strukturu naredbi za kreiranje tabela i indeksa.

U nastavku ove tačke će biti obrađen deo interaktivnog SQL-a, koji se odnosi na obavljanje funkcija: 1, 2, 4, kao i dela funkcije 3.

6.5.2. *Izražavanje upita i osnovna struktura naredbe SELECT*

Sve vrste upita se u SQL-u izražavaju putem naredbe SELECT. Ova naredba prati logiku relacionog računa nad torkama. Može se reći da je SQL, u domenu upitnog jezika, jedna implementacija relacionog računa. Može se pokazati da je ekspresivna moć upitnog jezika SQL pri tome i veća od ekspresivne moći relacionog računa sa sigurnim izrazima, prikazanog u tački 6.2. Osnovna struktura SELECT naredbe je:

SELECT *<lista_obeležja>*
FROM *<lista_tabela>*
WHERE *<uslov_selekcije>*

pri čemu *<lista_obeležja>* sadrži obeležja nad kojima se formira rezultat upita, *<lista_tabela>* sadrži nazive tabela*), potrebne za realizaciju upita, a *<uslov_selekcije>* izražava uslov selekcije podataka iz tabela koje su navedene iza službene reči FROM.

Naredba SELECT će, u nastavku, biti prikazana putem primera, bez većeg upuštanja u tumačenje strogih sintaksnih formalizama. Kao osnova za sve naredne primere, služiće šema baze podataka $\mathcal{S} = (S, \mathcal{I})$ iz primera 6.1.

6.5.2.1. *Upiti nad jednom tabelom*

Za izlistavanje celokupnog sadržaja tabele, ili za realizaciju projekcije tabele na skup obeležja, koriste se službene reči SELECT i FROM.

Primer 6.26. Realizacija upita "prikazati sadržaj tabele *radnik*":

SELECT *
FROM radnik

Oznaka * umesto *<liste_obeležja>* ukazuje na to da se prikazuju sva obeležja tabele *radnik*. □

Projekcija tabele se realizuje navođenjem konkretnih obeležja u *<listi_obeležja>*. Redosled obeležja u listi ukazuje i na redosled prikaza kolona tabele.

Primer 6.27. Realizacija upita "prikazati imena i prezimena svih radnika, kao i njihove matične brojeve":

SELECT IME, PRZ, MBR
FROM radnik □

*) Termin *tabela* u SQL-u predstavlja sinonim za pojam *relacija*.

Primer 6.28. Realizacija upita "prikazati imena i prezimena svih radnika":

```
SELECT IME, PRZ
FROM radnik
```

U rezultatu ovog upita se može dogoditi ponavljanje identičnih torki, ukoliko u tabeli *radnik* postoji više od jednog radnika s istim imenom i prezimenom. □

Uklanjanje višestrukog ponavljanja identičnih torki iz rezultata upita se obavlja pomoću službene reči DISTINCT, koja se navodi neposredno iza službene reči SELECT.

Primer 6.29. Realizacija upita "prikazati različita imena i prezimena svih radnika":

```
SELECT DISTINCT IME, PRZ
FROM radnik
```

Rezultat ovog upita neće sadržati višestruko ponovljene identične torke. □

Operator selekcije torki iz tabele se realizuje upotrebom klauzule WHERE, pri čemu *<uslov_selekcije>* definiše logički izraz, koji se interpretira za svaku torku tabele posebno. Logički izraz može biti složen iz više predikatskih izraza, povezanih logičkim simbolima AND, OR, ili NOT. Predikatski izrazi mogu biti relacioni izrazi tipa $x < y$, $x > y$, $x <= y$, $x >= y$, $x != y$, $x = y$, sa značenjem, redom, "manje", "veće", "manje ili jednako", "veće ili jednako", "različito", "jednako", ili mogu biti izrazi oblika:

- x IS NULL - x je nula vrednost,
- x IS NOT NULL - x nije nula vrednost,
- x BETWEEN y AND z - x je u intervalu vrednosti $[y, z]$,
- x NOT BETWEEN y AND z - x nije u intervalu vrednosti $[y, z]$,
- x LIKE z - x odgovara uzorku z (detaljnije objašnjenje sledi kasnije),
- x NOT LIKE z - x ne odgovara uzorku z (detaljnije objašnjenje sledi kasnije),
- x IN (*<lista_vrednosti>*) - x pripada specificiranoj listi vrednosti,
- x NOT IN (*<lista_vrednosti>*) - x ne pripada specificiranoj listi vrednosti,
- $x\ \theta$ ANY (*<lista_vrednosti>*), $\theta \in \{<, >, <=, >=, !=, =\}$. Na primer:

  x = ANY(*<lista_vrednosti>*)

  znači: x je jednako makar jednoj vrednosti u *<listi_vrednosti>*,
- $x\ \theta$ ALL(*<lista_vrednosti>*), $\theta \in \{<, >, <=, >=, !=, =\}$. Na primer:

  x = ALL(*<lista_vrednosti>*)

  znači: x je jednako svim vrednostima u *<listi_vrednosti>*,
- EXISTS (*<lista_vrednosti>*) - *<lista_vrednosti>* nije prazan skup vrednosti i
- NOT EXISTS (*<lista_vrednosti>*) - *<lista_vrednosti>* je prazan skup vrednosti.

U tekstu koji sledi, biće prezentovani primeri upotrebe ovih predikatskih izraza.

Primer 6.30. Realizacija upita "prikazati sve radnike, koji zarađuju više od *3600* dinara i koji nemaju svog rukovodioca":

```
SELECT *
FROM radnik
WHERE PLT > 3600 AND SEF IS NULL
```

Podizraz SEF IS NULL obezbeđuje selektovanje samo onih torki tabele *radnik*, za koje je vrednost torke nad obeležjem *SEF* nula vrednost. □

Sledeći primer prikazuje istovremenu primenu projekcije i selekcije podataka iz tabele *radnik* i ukazuje na primenu predikatskih izraza BETWEEN AND i LIKE.

Primer 6.31. Realizacija upita "prikazati matične brojeve, imena i prezimena svih radnika, čija zarada se kreće u rasponu od *2800* do *4000* dinara (uključujući i granične vrednosti) i čije prezime počinje slovom '*P*' ":

```
SELECT MBR, IME, PRZ
FROM radnik
WHERE    (PLT BETWEEN 2800 AND 4000) AND
         (PRZ LIKE 'P%')
```

Simbol % u predikatskom izrazu LIKE zamenjuje bilo koji niz znakova. □

Predikatski izraz:

<*obeležje*> LIKE <*uzorak*>

omogućava selektovanje torki, takvih da im vrednost nad obeležjem <*obeležje*> odgovara zadatom uzorku <*uzorak*>. Uzorak je string, koji pored ostalih, može sadržati specijalne simbole % i _. Simbol % zamenjuje pojavu nula ili više drugih znakova, dok simbol _ zamenjuje tačno jednu pojavu bilo kog drugog znaka.

Kao operand <*liste_obeležja*>, ili <*uslova_selekcije*> se može pojaviti i aritmetički izraz. Operandi aritmetičkog izraza su obeležja, konstante, ili skalarne funkcije, dok aritmetički operatori mogu biti +, -, *, /, u značenju, redom, sabiranje, oduzimanje, množenje i deljenje. Dozvoljena je upotreba običnih zagrada u cilju izmene uobičajenog prioriteta izvršenja operacija. Skup skalarnih funkcija nije standardizovan.

Primer 6.32. Realizacija upita "prikazati matične brojeve, imena, prezimena i godišnje plate svih radnika, zaokružene na dva decimalna mesta, čija dvostruka mesečna zarada nije veća od *4000* dinara":

```
SELECT MBR, IME, PRZ, ROUND(12 * PLT, 2)
FROM radnik
WHERE NOT (2 * PLT > 4000)
```

Funkcija ROUND(x, y) se koristi za zaokruživanje vrednosti x na naznačeni broj decimalnih mesta y. □

U sledećem primeru se demonstrira upotreba predikatskog izraza IN i logičke operacije OR.

Primer 6.33. Realizacija upita "prikazati matične brojeve radnika, čije mesečno angažovanje na projektu iznosi *12*, *15*, ili *20* sati, ili rade na projektu sa šifrom *110* ":

```
SELECT MBR
FROM radproj
WHERE BRC IN (12, 15, 20)  OR  SPR = 110
```

□

6.5.2.2. Uređivanje izlaznih rezultata upita

SQL - SELECT omogućava, putem klauzule ORDER BY, uređivanje izlaznih rezultata upita u rastućem, ili opadajućem redosledu. Klauzula:

ORDER BY *<podlista_obeležja>*

obezbeđuje prikaz izlaznih rezultata, uređenih saglasno obeležjima navedenim u *<podlisti_obeležja>*. Redosled obeležja u *<podlisti_obeležja>* određuje prioritet po kojem se vrši uređivanje. Iza svakog obeležja u *<podlisti_obeležja>* se može navesti službena reč ASC, kojom se zahteva rastući redosled prikaza podataka, ili DESC, kojom se zahteva opadajući redosled prikaza podataka. Ukoliko se drugačije ne naglasi, podaci se uređuju u rastućem redosledu.

ORDER BY je uvek poslednja klauzula naredbe SELECT.

Primer 6.34. Realizacija upita "prikazati radnike koji imaju svog rukovodioca, saglasno rastućem redosledu matičnog broja rukovodioca, zatim, saglasno rastućem redosledu prezimena radnika (unutar redosleda po šifri rukovodioca), a na kraju po opadajućem redosledu imena radnika":

```
SELECT MBR, PRZ, IME, SEF, PLT
FROM radnik
WHERE SEF IS NOT NULL
ORDER BY SEF ASC, PRZ ASC, IME DESC
```

Službene reči ASC su mogle biti izostavljene iz teksta upita. □

6.5.2.3. Upotreba skupovnih funkcija

Deklarativni jezici, kao što je SQL, su skupovno orijentisani jezici. Zbog toga, postoji mogućnost definsanja specijalnih funkcija, čiji argument predstavlja skup podataka, a izlazni rezultat je jedna vrednost, izračunata na osnovu zadatog skupa podataka. Takve funkcije se u SQL-u nazivaju ***skupovne funkcije***. Standardne SQL skupovne funkcije su:

- MAX(*<obeležje>*), koja vraća maksimalnu vrednost za *<obeležje>*, uzimajući u obzir sve selektovane torke,
- MIN(*<obeležje>*), koja vraća minimalnu vrednost za *<obeležje>*, uzimajući u obzir sve selektovane torke,
- COUNT(*), koja vraća ukupan broj selektovanih torki,
- COUNT(*<obeležje>*), koja vraća ukupan broj selektovanih torki, za koje vrednost *<obeležja>* nije nula vrednost,
- COUNT(DISTINCT *<obeležje>*), koja vraća ukupan broj različitih torki, za koje vrednost *<obeležja>* nije nula vrednost,
- SUM(*<obeležje>*), koja vraća zbir vrednosti datog *<obeležja>*, za sve selektovane torke, uključujući višestruko ponavljanje istih torki,
- SUM(DISTINCT *<obeležje>*), koja vraća zbir vrednosti datog *<obeležja>*, za sve različite selektovane torke,

- AVG(*<obeležje>*), koja vraća srednju vrednost datog *<obeležja>*, za sve selektovane torke, uključujući višestruko ponavljanje istih torki i
- AVG(DISTINCT *<obeležje>*), koja vraća srednju vrednost datog *<obeležja>*, za sve različite selektovane torke.

Primer 6.35. Realizacija upita "prikazati ukupan broj zaposlenih radnika, srednju vrednost plate svih radnika i ukupan godišnji iznos plata svih radnika":

```
SELECT COUNT(*), AVG(PLT), 12 * SUM(PLT)
FROM radnik
```
□

Primer 6.36. Realizacija upita "prikazati ukupan broj radnika koji imaju svog rukovodioca (*SEF*)":

```
SELECT COUNT(SEF)
FROM radnik
```
□

Primer 6.37. Realizacija upita "prikazati ukupan broj rukovodilaca (*SEF*) radnika":

```
SELECT COUNT(DISTINCT SEF)
FROM radnik
```
□

Primer 6.38. Realizacija upita "prikazati minimalnu i maksimalnu platu radnika":

```
SELECT MIN(PLT), MAX(PLT)
FROM radnik
```
□

6.5.2.4. Spajanje tabela

Spajanje tabela putem SELECT naredbe predstavlja dekartovo spajanje i realizuje se navođenjem naziva tabela koje se spajaju u klauzuli FROM. Klauzula WHERE, pri tome, može sadržati i uslov po kojem se selektuju torke iz dobijenog dekartovog spoja tabela, čime se realizuje operacija teta spajanja iz relacione algebre.

Primer 6.39. Realizacija dekartovog spoja tabela *radnik* i *projekat*:

```
SELECT radnik.*, projekat.*
FROM radnik, projekat
```

Izraz radnik.* označava listu svih obeležja tabele radnik, a analogno važi i za izraz projekat.*. □

Primer 6.40. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate svih rukovodilaca projekata (*RUK*)":

```
SELECT DISTINCT MBR, PRZ, IME, PLT
FROM radnik, projekat
WHERE RUK = MBR
```
□

Isti nazivi obeležja tabela koje se navode u FROM klauzuli moraju biti "kvalifikovani" i nazivom tabele. Na taj način se definiše kontekst obeležja. Sledeći primer ilustruje kvalifikaciju obeležja nazivom tabele.

Primer 6.41. Realizacija prirodnog spoja tabela *radnik*, *projekat* i *radproj*:

```
SELECT radnik.MBR, PRZ, IME, PLT, SEF, projekat.SPR, NAP, RUK, BRC
FROM radnik, radproj, projekat
WHERE radnik.MBR = radproj.MBR AND radproj.SPR = projekat.SPR
```
□

Umesto punog naziva tabele, u SELECT naredbi se može koristiti skraćeni naziv ("drugo ime") tabele, koji se navodi neposredno iza punog naziva tabele u FROM klauzuli.

Primer 6.42. Realizacija upita "prikazati matične brojeve, prezimena, imena, plate i broj časova mesečnog angažovanja svih radnika koji rade na projektu sa šifrom *110* ":

```
SELECT r.MBR, r.PRZ, r.IME, r.PLT, rp.BRC
FROM radnik r, radproj rp
WHERE r.MBR = rp.MBR AND rp.SPR = 110
```
□

6.5.2.5. *Upit s višestrukom upotrebom iste tabele i rekurzivni upit*

SELECT naredba dozvoljava višestruko navođenje iste tabele u FROM klauzuli. U tom slučaju, za označavanje konteksta upotrebe tabele, koristi se skraćeni naziv tabele.

Primer 6.43. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate svih radnika koji zarađuju više od radnika s matičnim brojem *30* ":

```
SELECT r1.MBR, r1.PRZ, r1.IME, r1.PLT
FROM radnik r1, radnik r2
WHERE r1.PLT > r2.PLT AND r2.MBR = 30
```

Kontekst *r2* upotrebe tabele *radnik* je "plata radnika s matičnim brojem *30* ", a kontekst *r1* je "lista radnika koji imaju veću platu od plate, selektovane u tabeli *r2* ". □

Postoje varijante SQL-a, koje omogućavaju rekurzivno izlistavanje podataka iz tabele, upotrebom klauzula CONNECT BY i START WITH u SELECT naredbi.

Primer 6.44. Realizacija upita "prikazati, rekurzivno (po obeležju *SEF*), matične brojeve, prezimena i imena svih radnika, saglasno hijerarhijskoj strukturi rukovođenja, čiji je glavni rukovodilac radnik s matičnim brojem *10*, uključujući i njega":

```
SELECT MBR, PRZ, IME
FROM radnik
CONNECT BY PRIOR MBR = SEF
START WITH MBR = 10
```

Opcija PRIOR ispred obeležja *MBR* obezbeđuje da se hijerarhijska struktura rukovođenja izlistava "odozgo na dole" (prethodna vrednost obeležja *MBR* u narednom koraku postaje vrednost obeležja *SEF*). Klauzula START WITH obezbeđuje da se struktura rukovođenja

izlista počev od radnika čiji je matični broj *10*. Nakon prikaza radnika s matičnim brojem *10*, selektuju se sve torke za koje je *SEF* = *10*, a zatim vrednost obeležja *MBR* svake selektovane torke postaje vrednost obeležja *SEF* i postupak se rekurzivno nastavlja. □

6.5.2.6. *Ugnježđeni upiti*

Ugnježđeni upiti su upiti kod kojih je jedna SELECT naredba ugrađena u WHERE klauzulu druge SELECT naredbe. Ugnježđeni upiti se realizuju putem predikatskih izraza: ANY, ALL, IN i EXISTS, kao i njihovih negacija. SQL dozvoljava višestruko ugnježđavanje upita.

Primer 6.45. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate svih radnika koji zaraduju više od proseka":

```
SELECT MBR, PRZ, IME, PLT
FROM radnik
WHERE PLT >ANY*) (SELECT AVG(PLT) FROM radnik)
```

Odgovor na ovaj upit se generiše tako što se prvo selektuju podaci ugnježđenog upita "SELECT AVG(PLT) FROM radnik", a potom se za svaku torku glavnog upita primenjuje kriterijum, naveden u klauzuli WHERE. □

Primer 6.46. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate svih radnika čije je mesečno angažovanje na nekom od projekata manje ili jednako najmanjem mesečnom angažovanju bilo kog radnika na projektu sa šifrom *10* ":

```
SELECT DISTINCT r.MBR, r.PRZ, r.IME, r.PLT
FROM radnik r, radproj rp
WHERE r.MBR = rp.MBR AND
      rp.BRC <= (SELECT DISTINCT BRC
                 FROM radproj
                 WHERE SPR = 10)
```

□

Primer 6.47. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate svih radnika koji nisu rukovodioci ni jednog projekta":

```
SELECT MBR, PRZ, IME, PLT
FROM radnik
WHERE MBR NOT IN (SELECT DISTINCT RUK
                  FROM projekat)
```

□

Treba primetiti da je predikatski izraz =ANY ekvivalentan izrazu IN, dok je izraz !=ALL ekvivalentan predikatskom izrazu NOT IN.

Svi, do sada navedeni ugnježđeni upiti, izvršavaju se tako što se prvo selektuju torke podupita pa se, nakon toga, za svaku torku glavnog upita, izračunava vrednost

*) U slučaju da ugnježđeni podupit kao rezultat daje tačno jednu vrednost, tada se službena reč ANY, odnosno ALL može izostaviti.

logičkog izraza WHERE klauzule. Takvi upiti se nazivaju ***nezavisni ugnježđeni upiti***, pošto rezultat podupita nije zavisan od podataka glavnog upita. Postoje, međutim, situacije u kojima podupit može zavisiti od podataka koji se selektuju glavnim upitom. Takvi upiti se nazivaju ***zavisni ugnježđeni upiti***. Sledeći primeri predstavljaju zavisne ugnježđene upite.

Primer 6.48. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate radnika, čiji je broj sati angažovanja na nekom od projekata veći od prosečnog broja sati angažovanja na tom projektu":

```
SELECT DISTINCT r.MBR, r.PRZ, r.IME, r.PLT
FROM radnik r, radproj rp1
WHERE r.MBR = rp1.MBR AND rp1.BRC >ANY
                  (SELECT AVG(BRC)
                   FROM radproj rp2
                   WHERE rp2.SPR = rp1.SPR)
```

U ovom slučaju selektovanje torki podupita se vrši za svaku torku glavnog upita po jednom! □

Primer 6.49. Realizacija upita "prikazati prva tri radnika po visini zarade u preduzeću":

```
SELECT *
FROM radnik r1
WHERE 3 >
        (SELECT COUNT(*)
         FROM radnik r2
         WHERE r2.PLT > r1.PLT)
```

Navedenim upitom, selektuju se samo oni radnici, za koje važi da je broj radnika, s platom koja je veća od plate datog radnika, manji od *3*. □

Realizacija upita koji bi se u relacionom računu izrazili putem univerzalnog kvantifikatora (odnosno, putem deljenja u relacionoj algebri) je u SQL-u nešto složenija, pošto univerzalni kvantifikator nije direktno implementiran.

Primer 6.50. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate radnika koji su angažovani na svakom projektu":

```
SELECT r.MBR, r.PRZ, r.IME, r.PLT
FROM radnik r
WHERE NOT EXISTS
        (SELECT p.SPR
         FROM projekat p
         WHERE SPR NOT IN
                 (SELECT SPR
                  FROM radproj rp
                  WHERE rp.SPR = p.SPR AND rp.MBR = r.MBR))
```

Upit je realizovan tako što je izvršena njegova dvojna negacija: selektovani su radnici za koje ne postoji projekat na kojem ne rade. □

6.5.2.7. *Upiti s grupisanjem podataka*

Opcija:

GROUP BY *<lista_obeležja>*

naredbe SELECT obezbeđuje particioniranje skupa selektovanih torki saglasno istim vrednostima skupa obeležja datog pomoću *<liste_obeležja>*. U tom slučaju, postoji mogućnost primene skupovnih funkcija nad pojedinačnim grupama dobijene particije.

Primer 6.51. Realizacija upita "prikazati za svakog radnika matični broj, prezime, ime, ukupan broj projekata i ukupno mesečno angažovanje na projektima na kojima radi":

```
SELECT r.MBR, r.PRZ, r.IME, COUNT(*), SUM(rp.BRC)
FROM radnik r, radproj rp
WHERE r.MBR = rp.MBR
GROUP BY r.MBR, r.PRZ, r.IME
```

U ovom slučaju se vrši particioniranje prirodnog spoja tabela *radnik* i *radproj*, saglasno istim vrednostima obeležja *MBR*, *PRZ* i *IME*. Funkcije COUNT(*) i SUM(rp.BRC) će, za svaku grupu torki (odnosno za svakog radnika), dati po jedan rezultat. □

Pri upotrebi klauzule GROUP BY postoji orgraničenje da iza klauzule SELECT mogu da se pojave samo skupovne funkcije primenjene na obeležja koja ne postoje u klauzuli GROUP BY i samo ona obeležja koja su navedena u klauzuli GROUP BY.

Uz klauzulu GROUP BY može da se pojavi i opcija:

HAVING *<uslov_selekcije_grupe>*,

koja obezbeđuje zadavanje kriterijuma selekcije celokupnih grupa. Logički izraz *<uslov_selekcije_grupe>* se formira na isti način kao i logički izraz klauzule WHERE. Prilikom primene naredbe SELECT, treba paziti na to da WHERE uslov daje kriterijum selekcije **pojedinačnih torki**, dok HAVING uslov daje kriterijum selekcije **kompletnih grupa torki**.

Primer 6.52. Realizacija upita "prikazati za svakog radnika, *koji radi na više od 3 projekta*, matični broj, prezime, ime, ukupan broj projekata i ukupno mesečno angažovanje na projektima na kojima radi":

```
SELECT r.MBR, r.PRZ, r.IME, COUNT(*), SUM(rp.BRC)
FROM radnik r, radproj rp
WHERE r.MBR = rp.MBR
GROUP BY r.MBR, r.PRZ, r.IME
HAVING COUNT(*) > 3
```

U ovom slučaju se vrši selektovanje samo onih grupa projekata za datog radnika, u kojima postoji više od tri torke. □

Primer 6.53. Realizacija upita "prikazati projekte na kojima je prosečno angažovanje veće od prosečnog angažovanja na svim projektima":

```
SELECT p.SPR, p.NAP, p.RUK, AVG(rp.BRC)
FROM projekat p, radproj rp
WHERE p.SPR = rp.SPR
GROUP BY p.SPR, p.NAP, p.RUK
HAVING AVG(rp.BRC) > (SELECT AVG(BRC) FROM radproj)
```
□

6.5.2.8. *Uniranje podataka*

Standardna mogućnost jezika SQL jeste uniranje skupova torki, dobijenih pomoću dve SELECT naredbe, upotrebom skupovnog operatora UNION. Neke varijante SQL-a omogućavaju i upotrebu skupovnih operatora razlike (MINUS) i preseka (INTERSECT). Skupovi torki koji se uniraju (ili presecaju, odnosno oduzimaju) treba da budu unijski kompatibilni.

Primer 6.54. Realizacija upita "prikazati matične brojeve, prezimena, imena i plate svih radnika koji zarađuju više od proseka, ili nemaju svog rukovodioca":

```
SELECT MBR, PRZ, IME, PLT
        FROM radnik
        WHERE PLT > ANY (SELECT AVG(PLT) FROM radnik)
UNION
SELECT MBR, PRZ, IME, PLT
        FROM radnik
        WHERE SEF IS NULL
```
□

Na kraju tačke 6.5.2, kao rezime, daje se globalni prikaz sintakse naredbe SELECT. Pojmovi u uglastim zagradama mogu biti izostavljeni, pojmovi u vitičastim zagradama se mogu ponavljati nula ili više puta, dok pojmovi razdvojeni simbolom | , predstavljaju alternative. Simbolom \ se označava početak i kraj pojma kkoji se pojavljuje tačno jedanput.

```
SELECT [DISTINCT]    \ * | tabela.* | aritmetički_izraz \
                     {, tabela.* | aritmetički_izraz }
FROM tabela [skraćeni_naziv] {, tabela [skraćeni_naziv] }
[WHERE uslov_selekcije]
[CONNECT BY uslov_povezivanja [START WITH uslov_selekcije] ]
[GROUP BY obeležje {, obeležje} [HAVING uslov_selekcije] ]
[ \ UNION | INTERSECT | MINUS \  SELECT ... ]
[ORDER BY obeležje [ASC | DESC] {, obeležje [ASC | DESC] } ]
```

Slika 6.13.

6.5.3. Kreiranje pogleda putem SQL-a

Pogledi se u SQL-u kreiraju upotrebom naredbe za izražavanje upita SELECT. Naredba za kreiranje pogleda je:

CREATE VIEW *<naziv_pogleda>* [(*<lista_obeležja>*)]
AS SELECT *<upit>*

Pojam *<lista_obeležja>* predstavlja niz obeležja pogleda. Značenje i redosled obeležja iz *<liste_obeležja>* treba da odgovara značenju i redosledu liste klauzule SELECT. Pogledu se, nakon što je kreiran, može pristupati u cilju selekcije podataka, na isti način, kao da je u pitanju tabela baze podataka.

Primer 6.55. Pogled "svi rukovodioci projekta" se kreira na sledeći način:

```
CREATE VIEW ruk_proj (MRP, IME, PRZ. PLT) AS
        SELECT MBR, IME, PRZ, PLT
        FROM radnik
        WHERE MBR IN
                (SELECT DISTINCT RUK
                 FROM projekat)
```

Značenje obeležja *MRP* je matični broj rukovodioca projekta. □

Primer 6.56. Realizacija upita: "prikazati matične brojeve, imena i prezimena rukovodilaca projekata, kao i broj projekata kojima rukovode":

```
SELECT rk.MRP, rk.IME, rk.PRZ, COUNT(*)
        FROM ruk_proj rk, projekat p
        WHERE rk.MRP = p.RUK
        GROUP BY rk.MRP, rk.IME, rk.PRZ
```

□

Definicija pogleda se briše iz rečnika podataka naredbom

DROP VIEW *<naziv_pogleda>*,

pri čemu je *<naziv_pogleda>* naziv onog pogleda, kojeg treba izbrisati.

6.5.4. Ažuriranje baze podataka putem jezika SQL

Ažuriranje baze podataka se realizuje putem naredbi:

- za dodavanje novih torki u tabelu (INSERT),
- za brisanje torki iz tabele (DELETE) i
- modifikaciju torki tabele (UPDATE).

Dodavanje nove torke

Sintaksa naredbe za dodavanje nove torke ima oblik:

INSERT INTO <*ime_tabele*> [(<*lista_obeležja*>)]
\ VALUES (<*lista_konstanti*>) | SELECT ... \

<*lista_obeležja*> predstavlja skup obeležja tabele s nazivom <*ime_tabele*>. Ukoliko se <*lista_obeležja*> ne navede, onda se podrazumeva redosled obeležja koji je utvrđen pri kreiranju tabele (pogledati tačku 6.5.5).

Dodavanje nove torke u tabelu se može obaviti na dva načina:

- navođenjem vrednosti obeležja torke, putem <*liste_konstanti*>, koja treba da odgovara <*listi_obeležja*>, ili
- selektovanjem torki putem naredbe SELECT.

Primer 6.57. Prikaz postupka dodavanja po jedne nove torke u tabele *radnik*, *projekat* i *radproj* :

INSERT INTO radnik (MBR, PRZ, IME, PLT, SEF)
VALUES (10, 'Car', 'Ana', 4600, NULL)

INSERT INTO projekat
VALUES (100, 'Univerzitetski IS', 10)

INSERT INTO radproj (MBR, SPR, BRC)
VALUES (10, 100, 30)

Specijalna konstanta NULL označava da obeležje *SEF* dobija nula vrednost. □

Primer 6.58. Prikaz postupka dodavanja jedne nove torke u tabelu *tab1*, na osnovu selekcije torki iz tabele *tab2*:

INSERT INTO tab1 (A, B, C, D)
SELECT A, B, C, D
FROM tab2
WHERE A > B □

Brisanje postojećih torki

Sintaksa naredbe za brisanje postojećih torki ima oblik:

DELETE FROM <*naziv_tabele*> [WHERE <*uslov_selekcije*>]

Opcija WHERE ima istu funkciju kao i u slučaju SELECT naredbe. Ukoliko se ne navede, sve torke iz tabele će biti izbrisane.

Primer 6.59. Prikaz postupka brisanja torke iz tabele *radnik* :

DELETE FROM radnik
WHERE MBR = 40 □

Modifikacija postojećih torki

Sintaksa naredbe za modifikaciju postojećih torki ima oblik:

UPDATE <*naziv_tabele*>
SET <*obeležje*> = <*aritm_izraz*> {, <*obeležje*> = <*aritm_izraz*>}
[WHERE <*uslov_selekcije*>]

ili:

UPDATE <*naziv_tabele*>
SET (<*lista_obeležja* >) = (SELECT ...)
{, (<*lista_obeležja*>) = (SELECT ...)}
[WHERE <*uslov_selekcije*>]

Opcija WHERE ima istu funkciju kao i u slučaju SELECT naredbe. Ukoliko se ne navede, sve torke tabele će biti modifikovane. Prvi oblik naredbe UPDATE se koristi u situaciji kada se vrednosi obeležja modifikuju direktno, putem aritmetičkog izraza <*aritm_izraz*>. Drugi oblik se koristi u situaciji kada se <*lista_obeležja*> modifikuje indirektno, putem naredbe SELECT. Rezultat takve SELECT naredbe mora biti jedna torka, koja po podacima odgovara <*listi_obeležja*>.

Primer 6.60. Prikaz postupka direktnog modifikovanja torke tabele *radnik*:

```
UPDATE radnik
SET PLT = 1.1 * PLT, SEF = 10
WHERE MBR = 40
```

□

Primer 6.61. Prikaz postupka indirektnog modifikovanja torke tabele *projekat*:

```
UPDATE projekat
SET (RUK) =     (SELECT SEF
                 FROM radnik
                 WHERE MBR = 40)
WHERE SPR = 100
```

□

6.5.5. *Osnove definisanja fizičke strukture baze podataka putem jezika SQL*

U okviru ove tačke će biti opisan osnovni oblik naredbi za:

- kreiranje tabele (CREATE TABLE),
- brisanje tabele (DROP TABLE),
- kreiranje stabla pristupa (CREATE INDEX) i
- brisanje stabla pristupa (DROP INDEX).

Osnovna sintaksa naredbe za kreiranje nove tabele ima oblik:

CREATE TABLE <*naziv_tabele*>
(<*obeležje*> <*tip_podatka*> [NOT NULL]
{, <*obeležje*> <*tip_podatka*> [NOT NULL] })

Prikazani deo sintakse naredbe CREATE TABLE je isti za sve varijante SQL-a. Ova naredba poseduje, međutim, veliki broj dodatnih klauzula, koje se odnose na deklaraciju lokalnih i međurelacionih ograničenja, kao i definisanje parametara fizičke strukture tabele, koji zavise od konkretnog sistema za upravljanje bazama podataka.

Za svako *<obeležje>* tabele se navodi *<tip_podatka>*, koji, takođe, zavisi od vrste sistema za upravljanje bazama podataka. Klauzula NOT NULL obezbeđuje zabranu dodele nula vrednosti obeležju tabele. Ukoliko se drugačije ne naglasi, za obeležja se dozvoljava dodela nula vrednosti.

Brisanje definicije tabele (kao i svih njenih torki) se obavlja putem naredbe:

DROP TABLE *<naziv_tabele>*

Primer 6.62. Prikaz postupka kreiranja tabela *radnik*, *projekat* i *radproj*:

```
CREATE TABLE radnik
        (MBR    NUMBER(6) NOT NULL,
         PRZ    VARCHAR(24) NOT NULL,
         IME    VARCHAR(12) NOT NULL,
         PLT    MONEY,
         SEF    NUMBER(6)
        )

CREATE TABLE projekat
        (SPR    NUMBER(6) NOT NULL,
         NAP    VARCHAR(24) NOT NULL,
         RUK    NUMBER(6)
        )

CREATE TABLE radproj
        (SPR    NUMBER(6) NOT NULL,
         MBR    NUMBER(6) NOT NULL,
         BRC    NUMBER(2)
        )
```

□

Stabla pristupa (indeksi) se, zajedno sa klauzulom NOT NULL naredbe CREATE TABLE, koriste u cilju implementacije integriteta entiteta (ograničenja ključa) i u cilju obezbeđenja efikasnog traženja i pretraživanja podataka tabele. Osnovna sintaksa naredbe za kreiranje indeksa je:

CREATE [UNIQUE] INDEX *<naziv_indeksa>*
ON *<naziv_tabele>*
(*<obeležje>* [ASC | DESC] {, *<obeležje>* [ASC | DESC] })

Stablo pristupa se realizuje kao jedna od varijanti B - stabla. Klauzula UNIQUE ukazuje da će indeks, a samim tim i tabela, sadržati jedinstvene vrednosti nad navedenim nizom obeležja. Ukoliko se drugačije ne naglasi, indeks, odnosno tabela, može sadržati višestruke pojave istih vrednosti navedenih obeležja. Redosled obeležja u naredbi CREATE INDEX je važan i ukazuje na uređenje podataka, pokrivenih indeksom (koje može biti rastuće (ASC), ili opadajuće (DESC)). Ukoliko se službena reč ASC, tj. DESC ne navede uz

obeležje, tada se uzima rastuće uređenje vrednosti datog obeležja. Efekti izvršenja naredbe SELECT će uvek biti isti, bez obzira da li su nad tabelom definisani indeksi, ili ne. Performanse izvršenja naredbe SELECT, kao i naredbi za ažuriranje baze podataka se, međutim, mogu značajno razlikovati, u zavisnosti od toga da li odgovarajući indeksi postoje, ili ne.

Kao i u slučaju naredbe CREATE TABLE, i naredba CREATE INDEX poseduje dodatne klauzule, koje su specifične za pojedine tipove sistema za upravljanje bazama podataka.

Brisanje indeksa se postiže upotrebom naredbe:

DROP INDEX *<naziv_indeksa>*

Treba napomenuti da se indeksi nad tabelom automatski brišu pri upotrebi naredbe DROP TABLE.

Primer 6.63. Prikaz postupka kreiranja indeksa nad primarnim ključevima tabela *radnik*, *projekat* i *radproj*:

```
CREATE UNIQUE INDEX idx_radnik
ON radnik (MBR ASC)

CREATE UNIQUE INDEX idx_projekat
ON projekat (SPR)

CREATE UNIQUE INDEX idx_radproj
ON radproj (MBR, SPR)
```

□

Konkretni sistemi za upravljanje bazama podataka poseduju, pored navedenih, i čitav niz drugih SQL naredbi za definisanje fizičke strukture baze podataka.

7. Glava

Objektno - orijentisani model podataka

Tokom osamdesetih godina je došlo do primene računara u jednom broju novih oblasti. U te nove oblasti primene spadaju:

- računarom podržano projektovanje u mašinstvu, elektrotehnici, arhitekturi, građevinarstvu i informatici,
- multimedijalni sistemi i
- baze znanja.

Sve ove nove oblasti primene zahtevaju manipulisanje velikim količinama podataka i mogle bi imati koristi od primene SUBP. Međutim, priroda podataka u tim primenama se teško uklapa u relacione okvire. Na primer, sistemi za računarom podržano projektovanje treba da omoguće izgradnju modela kompleksnih objekata i održavanje različitih verzija istog objekta. Multimedijalni sistemi sadrže tekstove varijabilne dužine, grafiku, slike, audio i video podatke, što rezultuje u zahtevu za efikasnim memorisanjem i manipulisanjem nizovima velike i promenljive dužine. Konačno, sistemi zasnovani na znanju zahtevaju semantički bogate podatke i kompleksne operacije. Sve ove nove oblasti primene stavljaju akcenat na još dva zahteva. To su: produktivnost programera i performanse obrade.

Opisani zahtevi su doveli i do razvoja novog modela podataka, koji bi trebalo da ih zadovolji. Taj novi model podataka se naziva objektno - orijentisanim. Razvija se na idejama objektno - orijentisanih programskih jezika i semantičkih modela podataka. Ceo pristup se, često, naziva objektno - orijentisanom ***paradigmom***. Termin paradigma se odnosi na određeni način mišljenja o nečem. Kada je reč o objektno - orijentisanoj paradigmi, reč je o softverskom predstavljanju realnih entiteta putem parova (struktura podataka, ponašanje). Ti parovi predstavljaju nedeljivu celinu, a nazivaju se objektima. Pri tome, ponašanje se odnosi na programsku realizaciju postupaka, putem kojih objekti me-

njaju stanje ili samo daju informaciju o svom stanju. Stanje objekta je definisano strukturom podataka, putem koje je realni entitet predstavljen.

Objektno - orijentisani model podataka je još u razvoju. Ne postoji opšte prihvaćen stav koji od, međusobno različitih, objektno - orijentisanih jezika treba da predstavlja osnovu za njegovu operacijsku komponentu. Model nije unificiran, jer oslanjanje na različite programske jezike dovodi do daljih razlika u verzijama modela podataka. Cilj ovog poglavlja je da ukaže na stanje razvoja objektno - orijentisanog modela podataka. Samo poglavlje je podeljeno u pet delova. U prvom su nešto opširnije opisani zahtevi novih oblasti primena u odnosu na baze podataka, kao i osnovni razlozi nepogodnosti relacionog modela da tim zahtevima udovolji. Mada su strukturalna i operacijska komponenta kod objektno - orijentisanog modela podataka međusobno čvrsto povezane, drugi, treći i četvrti deo poglavlja je posvećen detaljnom opisu strukturalne komponente modela sa samo neophodnim refleksijama na operacijsku komponentu. Peti deo je posvećen integritetnoj komponenti. Kao operacijska komponenta modela podataka, u šestom delu je opisan programski jezik C++. Konačno, sedmi deo je posvećen komparaciji potencijala objektno - orijentisanog i stvarnih mogućnosti relacionog modela podataka.

7.1. Nove oblasti primene baza podataka

Računarom podržano projektovanje, baze znanja i multimedijalni sistemi predstavljaju potencijalno nove oblasti primene baza podataka. Redosled nabrajanja ovih oblasti primene, ukazuje i na redosled uvođenja odgovarajućih softverskih sistema u praksu. Softverski sistemi za računarom podržano projektovanje (Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Computer Aided Software Engineering (CASE)) su se pojavili prvi, te će naredna analiza zahteva u odnosu na sisteme baza podataka biti, pretežno, zasnovana na potrebama tih sistema.

Softverski sistemi za računarom podržano projektovanje se koriste za rešavanje kompleksnih zadataka inženjerske prakse. U takve zadatke spadaju izrade projekata: mašinskih proizvoda, proizvodnih linija, kompletnih fabrika, brodova, aviona, zgrada, mostova, integrisanih kola velike gustine, računara ili informacionih sistema. Inženjersko projektovanje postavlja jedan broj novih zahteva u odnosu na model podataka i sistem za upravljanje bazom podataka. Ti zahtevi su posledica sledećih specifičnosti primene računara u inženjerskom projektovanju:

- ***Nehomogenost podataka.*** Inženjerski projekti sadrže heterogen skup objekata. Za te projektne objekte je karakterističan veliki broj različitih tipova, svaki sa malim brojem pojava.
- ***Nizovi promenljive i nizovi velike dužine.*** Digitalizovani crteži predstavljaju, često, sastavni deo inženjerskih projekata. U digitalizovanom obliku, ceo crtež je predstavljen kao veoma dug niz bajtova, od kojih svaki nosi informaciju o nivou sivila jedne tačke, izuzetno malih dimenzija, na crtežu. Pored toga, oni poseduju i zapise promenljive dužine sa tekstualnim podacima.

- *Kompleksni objekti.* Inženjerski projekti, po pravilu, sadrže kompleksne objekte, koji se mogu rekurzivno deliti u manje objekte. Kompleksan objekat ima hijerarhijsku strukturu podataka. Bez obzira na to, korisnik mora moći da manipuliše tim objektom kao da je jedinstven.
- *Upravljanje verzijama.* U inženjerskim projektima je neophodno voditi zapise o evoluciji rešenja. Projektanti često eksperimentišu sa više verzija jednog objekta, pre nego što izaberu onu, koja najbolje zadovoljava postavljene zahteve. U principu, jedan objekat može imati samo jednu prethodnu verziju i više paralelnih narednih verzija. Svaka od narednih verzija se naziva *alternativom*. Verzije jednog objekta formiraju strukturu stabla.
- *Evolucija šeme.* Inženjerski projekti, obično, prolaze dug evolutivni period. Tokom tog perioda, intenzivno se menja statička struktura projekta. Te promene se reflektuju kroz potrebu da se dinamički menja i šema baze podataka.
- *Ekvivalentni objekti.* Postoji više mogućih pogleda na isti objekat projektovanja. Na primer, jedan VLSI čip se može predstaviti na nivou logičkih kola, u cilju provere logičke funkcionalnosti, na nivou tranzistora, u cilju analize brzine rada, ili na nivou šeme povezivanja, u cilju provere primenjenih pravila projektovanja. Mada se reprezentacije znatno razlikuju, reč je o samo različitim reprezentacijama istog objekta. Te reprezentacije se nazivaju *ekvivalentnim objektima.* Prava svrha ekvivalentnih objekata je da uvedu ograničenja na bazu podataka. Naime, ako se deo projekta izmeni u jednoj repezentaciji, sistem treba ili da tu promenu sprovede i u drugim reprezentacijama, ili da barem isti deo, u drugim reprezentacijama, označi kao nevažeći.
- *Modularnost.* Velika složenost projekta nameće potrebu paralelnog rada većeg broja projektanata na njegovim delovima - *modulima*. Moduli projekta nisu medusobno nezavisni. Pri tome, svaki modul sme da menja samo njegov projektant, a drugim projektantima treba da budu poznate njegove ulazno - izlazne karakteristike, da bi sa njime povezali svoj modul.
- *Dugačke transakcije.* Transakcije u inženjerskom projektovanju uz primenu računara traju veoma dugo. Projektant započinje transakciju, putem koje menja neki objekat i radi na njemu nedeljama, pre nego što, završivši transakciju, stavlja rezultate svog rada na uvid i drugim korisnicima baze podataka.

Multimedijalne baze podataka. U modernom kancelarijskom informacionom ili drugom multimedijalnom sistemu, podaci uključuju ne samo tekstove i brojeve, već i slike, grafiku i digitalizovane zvučne i video zapise. Takvi multimedijalni podaci se memorišu kao nizovi bajtova promenljive dužine, a delovi tih nizova su medusobno spregnuti, radi lakšeg povezivanja tekstova, slika, zvuka i video zapisa.

Baze znanja. Veštačka inteligencija i ekspertni sistemi predstavljaju informacije kao činjenice i pravila, koje se mogu zajedno posmatrati kao baza znanja. U tipičnim primenama veštačke inteligencije, predstavljanje znanja zahteva strukture podataka sa bogatom semantikom. Operacije u bazi znanja su kompleksnije nego u tradicionalnim bazama podataka. Na primer, kada se želi dodati neko novo pravilo, sistem mora proveriti da li je ono suvišno ili u kontradikciji sa dotadašnjim pravilima. Kompleksnost takvih provera raste veoma brzo sa porastom baze znanja.

7.1.1. Ograničenja relacionog modela podataka

Relacioni sistemi za upravljanje bazama podataka nisu u stanju da zadovolje zahteve ovih novih oblasti primene računara. Za to postoje dva razloga. Jedan je posledica činjenice da se, pri projektovanju relacionih sistema za upravljanje bazama podataka, jednostavno nije vodilo računa o tim novim oblastima primene. Drugi razlog leži u činjenici da sam relacioni model podataka ne poseduje mogućnosti za efikasno zadovoljenje zahteva, koje postavljaju nove oblasti primene.

Relacioni sistemi za upravljanje bazama podataka nisu razvijani tako da bi efikasno podržavali:

- Rad sa nehomogenim skupovima podataka, jer su projektovani za efikasno upravljanje bazama podataka, kod kojih broj pojava (torki) daleko premašuje broj tipova entiteta (šema relacija).
- Intenzivne izmene šeme baze podataka, jer su razvijani pod pretpostavkom da se šema baze podataka retko menja, pa su u njih ugrađeni i, relativno, skromni mehanizmi za izmenu šeme.
- Upravljanje različitim verzijama jednog objekta. Modifikacijom se, u relacionoj bazi podataka, gube prethodne vrednosti određenih podataka. Ne postoje ugrađeni mehanizmi za efikasno vođenje evidencije o promenama stanja objekata. S druge strane, to su prirodni zahtevi primena, kao što su: planiranje, simulacija i podrška odlučivanju.
- Upravljanje ekvivalentnim objektima, posebno kada je reč o evidentiranju potrebe izmene jednog objekta, na osnovu izmena u njemu ekvivalentnom objektu.
- Upravljanje dugačkim transakcijama. Transakcija je takav niz akcija čitanja i pisanja u bazu podataka, koji ostavlja bazu podataka u konzistentnom stanju. Nakon završetka transakcije, sve promene, koje je ona izazvala u bazi podataka, postaju vidljive odjednom, ili, ako je došlo do bilo kakve neregularne situacije, upravljač transakcijama poništava sve promene u bazi podataka, kao da transakcija nije ni izvršavana. Upravljači transakcijama relacionih sistema su razvijani za transakcije, koje kratko traju. Ako dođe do pada sistema tokom izvršavanja dugačke transakcije, putem koje se modifikuje neki projektantski objekat, strategija rekonstrukcije stanja baze podataka na osnovu kopije nekog prethodnog stanja i sadržaja žurnal datoteke, rezultovala bi u gubitku nedelja rada, jer bi se rekonstrukcija baze podataka izvršila saglasno stanju pre početka posmatrane transakcije. Primena računara u inženjerskom projektovanju traži od sistema za upravljanje bazom podataka da i parcijalni rezultati neke transakcije budu vidljivi i da omogući takvu rekonstrukciju baze podataka da i parcijalni rezultati projektantskih transakcija budu spašeni, a ne da rekonstrukcija znači vraćanje u stanje nakon poslednje uspešno završene transakcije.

Mada jednostavan i zasnovan na strogim teorijskim osnovama, relacioni model nameće određena ograničenja na reprezentovanje realnih entiteta u bazama podataka. Organizovanjem podataka u relacije (tabele), relacioni model pretpostavlja horizontalnu i vertikalnu homogenost podataka. Horizontalno, svaka torka date relacije poseduje istu definiciju za obeležja. Vertikalno, dato obeležje uzima vrednosti iz istog domena u svakoj

torci. Pri tome, domeni su ograničeni na primitivne tipove podataka, kao što su: celi i realni brojevi, karakteri i slično, ograničene dužine. Svakoj torci odgovara samo jedna vrednost iz domena svakog obeležja. Potreba da se reprezentacijama entiteta jedne klase pridruži skup vrednosti iz nekog domena, dovodi ili do pojave većeg broja torki sa pretežno (ali ne potpuno) istim sadržajem, ili do dekompozicije šeme relacije, kao modela klase realnih entiteta, na nekoliko novih šema relacija. Broj torki sa pretežno istim sadržajem jednak je kardinalnom broju skupa vrednosti iz domena, koji treba pridružiti jednom entitetu. Takvo predstavljanje podataka o entitetima dovodi do nepoželjne redundanse podataka,

Ako se šema relacije rastavi (dekomponuje) na više šema relacija, tada se kompleksan objekat memoriše putem zapisa, rasutih po različitim relacijama. Međutim, za korisnika taj kompleksni objekat i dalje predstavlja jednu logičku celinu i jedinicu posmatranja. Dekompozicija značajno smanjuje performanse računarske obrade podataka, jer uvodi potrebu čestog korišćenja operatora prirodnog spajanja, u cilju rekonstruisanja kompleksnih objekata.

Dalje, relacioni model eksplicitno ne uključuje semantiku, kao deo reprezentacije realnih entiteta. Umesto toga, aplikacioni programi interpretiraju semantiku podataka. Kada se primeni dekompozicija, ne samo što se smanjuje efikasnost obrade, već se povećava i verovatnoća gubljenja semantike povezane sa podacima. S druge strane, jezik podataka relacionog modela se ne može sam koristiti za izvođenje operacija, potrebnih u bazama znanja.

Može se zaključiti da relacioni model podataka, sam po sebi, nije pogodan za:

- izgradnju struktura sa podacima promenljive i velike dužine,
- predstavljanje kompleksnih objekata,
- izvršavanje operacija takve kompleksnosti, kakvu zahtevaju primene u bazama znanja.

Primer 7.1. U relacionoj bazi podataka se, često, podaci o radniku nalaze u relacijama nad šemama *Radnik*, *Odelenje*, *Radno_Mesto*, *Radnik_Projekat*. U objektno-orijentisanom sistemu, objekti realnog sveta predstavljeni su kao jedan objekat baze podataka. Jedan od osnovnih ciljeva objektno - orijentisanog modela podataka je da verno preslika objekat realnog sveta u njegovu softversku reprezentaciju. Drugim rečima, *Odelenje*, *Radno_Mesto*, *Projekat* mogu biti tipovi objekata baze podataka, ako je to potrebno, ali *Radnik* postaje kompleksni tip objekat, a njegova pojava sadrži sve podatke o samom radniku, o odelenju u kojem radi, o radnom mestu na koje je raspoređen i o projektima na kojima je angažovan. □

Potreba intenzivne primene operatora prirodnog spajanja za rekonstrukciju kompleksnih objekata od torki, rasutih po raznim relacijama, pokazuje da relacioni sistemi nemaju odgovarajuće performanse, za primenu u novim oblastima. Zbog toga su, mnogi današnji sistemi, namenjeni za inženjersko projektovanje, izgrađeni na sistemu datoteka. Jedna studija [CH] pokazuje da realizacija CAD sistema putem relacionog SUBP dovodi do petostrukog povećanja vremena pretraživanja podataka u poređenju sa implementacijom putem datoteka. Međutim, izgradnja putem sistema datoteka dovodi do poznatih problema zavisnosti programa i podataka, nekompatibilnosti i nekonzistentnosti podataka.

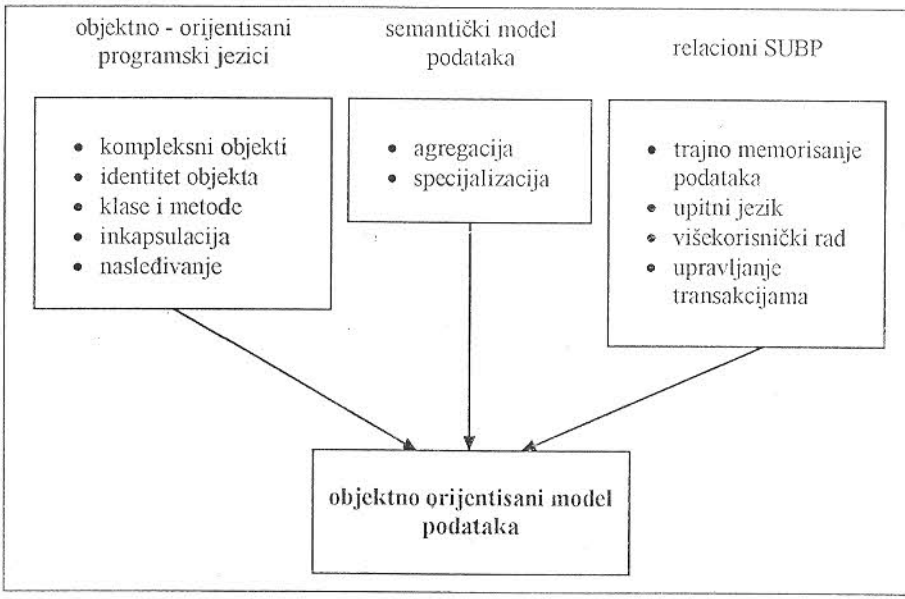
S druge strane, za relacione SUBP su razvijeni principi izgradnje čitavog niz važnih mehanizama, kao što su:

- neproceduralni jezik podataka,
- automatska optimizacija upita,
- uslovi integriteta,
- upravljač transakcijama,
- zaštita od neovlašćenog korišćenja,
- zaštita od uništenja,
- distribucija baze podataka i distribucija njene obrade,

koje će i novi, objektno - orijentisani SUBP morati da koriste. Relacioni sistemi će se dalje koristiti u oblastima gde su zahtevi u odnosu na tipove podataka i na koncepte za izgradnju modela realnih entiteta primereni mogućnostima modela. Međutim, za određene primene su relacioni SUBP jednostavno nepogodni.

7.1.2. *Poreklo objektno - orijentisanog modela podataka*

Objektno - orijentisani model podataka (dalje OOMP) je razvijan na drugi način nego relacioni model podataka. Umesto da se pođe od jednostavne definicije i snažnog matematičkog aparata, razvoj OOMP je zasnovan na tradicijama:



Slika 7.1.

- objektno - orijentisanih programskih jezika i
- semantičkih modela podataka.

Objektno - orijentisani SUBP se razvijaju na tradicijama SUBP, zasnovanih na drugim modelima podataka. Slika 7.1 pokazuje koji koncepti iz različitih oblasti su korišćeni za razvoj objektno - orijentisanog modela podataka i SUBP.

Prva pojava objekta, kao koncepta, javlja se u Simuli, simulacionom jeziku razvijenom u ranim sedamdesetim godinama. Međutim, istraživači su počeli pokazivati interes za objektno - orijentisane jezike tek nakon pojave Smalltalk - a [GR]. Od onda, u literaturi je opisan veći broj takvih jezika. Danas, pored Smalltalk -a, intenzivno se koriste objektni Pascal i C++. Interes za primenu objektno - orijentisanih jezika u inženjerskim primenama je posledica:

- njihove sposobnosti da efikasno koriste strukture sa praktično proizvoljnim tipovima podataka,
- postojanja bogatih biblioteka sa gotovim klasama (pojam klase je objašnjen u daljem tekstu) i
- mogućnosti ponovnog korišćenja već proverenog programskog koda.

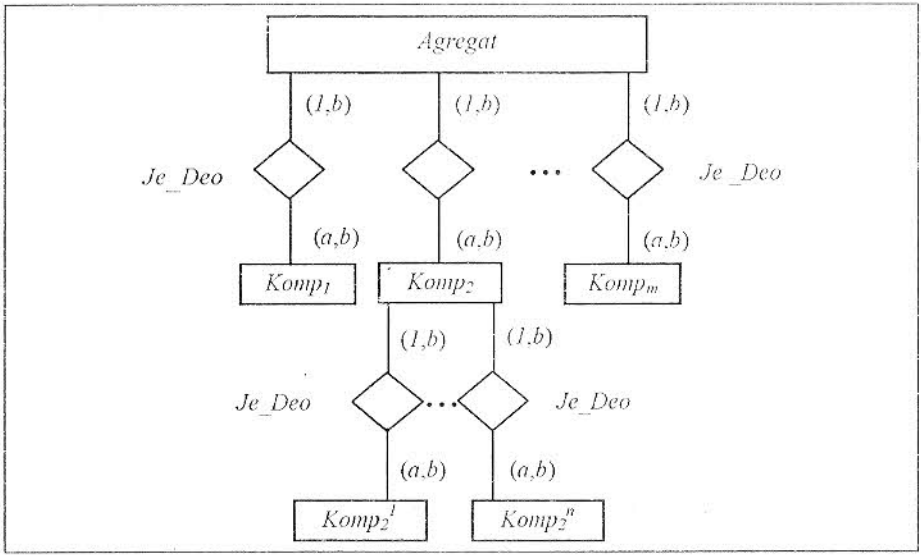
Razlika između objektno - orijentisanog programskog jezika i objektno - orijentisane baze podataka je da baza podataka podrazumeva postojanje permanentno memorisanih objekata na medijumu eksterne memorije, dok objekti u objektno - orijentisanom programskom jeziku postoje samo tokom izvršavanja programa. Dodavanje mogućnosti trajnog memorisanja podataka programskom okruženju, dovelo je do pojave objektno - orijentisanih SUBP.

Semantički modeli podataka su druga od tri oblasti, koje su uticale na razvoj objektno - orijentisanog modela podataka. U semantičke modele podataka spadaju: prošireni ER model, funkcionalni model i semantički model podataka. Semantički modeli podataka imaju dva moćna koncepta. To su: agregacija i specijalizacija.

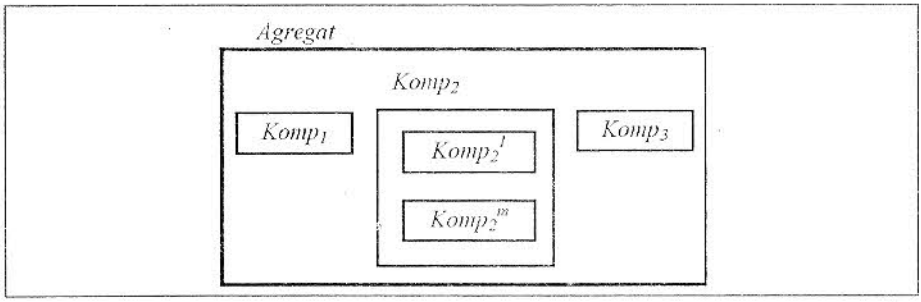
Agregacija je postupak izgradnje složenih objekata - agregata, od objekata - komponenata. Postoje dva slučaja sličnosti između agregata i koncepata ER modela podataka. Prvi je kada se agregacijom obeležja dobija tip entiteta - agregirani objekat. Drugi je slučaj kada se kombinuju međusobno povezani tipovi entiteta u agregirani tip entiteta na višem nivou apstrakcije. Odnos između tipa entiteta - komponente i agregata se naziva vezom tipa "je deo". Pojava agregata je egzistencijalno zavisna od pojava svojih komponenata. Na slici 7.2 je prikazana reprezentacija agregata i komponenata putem koncepata ER modela podataka. Treba naglasiti da i komponenta može predstavljati agregat. Agregacija je jedan od postupaka za predstavljanje hijerarhijskog odnosa između entiteta različitih realnih klasa.

Međutim, agregat je jedna celina, te je, sa te tačke gledišta, adekvatnija njegova geometrijska reprezentacija prema slici 7.3. U svakom slučaju, agregat je apstraktni tip entiteta, koji sadrži heterogene komponente.

Korišćenjem termina relacionog modela podataka, pojam agregata se može objasniti na sledeći način: šema relacije, reprezent agregata ima obeležja, koja i sama mogu biti šeme relacija, tako da vrednosti obeležja u torci mogu predstavljati torke nekih drugih relacija. Pojava šeme relacije u ulozi obeležja neke druge šeme relacije, naziva se "***ugnježdavanjem***". Ugnježdavanje može imati proizvoljnu dubinu.



Slika 7.2.



Slika 7.3.

Specijalizacija je apstrakcija proširenog ER modela podataka, te je detaljno obrađena u glavi 2 ove knjige. Ukratko, elementi neke klase se mogu specijalizovati i grupisati u potklase. Jedna potklasa nasleđuje sve osobine svojih superklasa. Elementi potklase pripadaju i superklasi. Element potklase nasleđuje osobine odgovarajućeg elementa člana superklase, ali poseduju i svoje, specifične osobine.

7.2. Osnovni koncepti objektno - orijentisanog modela podataka

Objektno - orijentisana paradigma donosi čitav niz novih pojmova i mehanizama za predstavljanje realnih entiteta. Ti pojmovi i mehanizmi predstavljaju koncepte objektno - orijentisanog modela podataka. U nove pojmove i mehanizme spadaju:

- objekat,
- metoda,
- poruka,
- inkapsulacija (učaurenje),
- klasa,
- nasleđivanje i
- identitet objekta.

Svi ovi koncepti su opisani i ilustrovani u ovoj tački. Zbog kompleksnosti i važnosti uloge, koju igraju u objektno - orijentisanom modelu podataka, mehanizam nasleđivanja i pojam identiteta objekta opisani su i u posebnim tačkama ovog poglavlja.

7.2.1. Osnovni tipovi podataka

Svaki programski jezik podržava neki skup tipova podataka. Kada je reč o konvencionalnim programskim jezicima, kao što je Pascal ili, kada je reč o jezicima podataka mrežnog ili relacionog sistema za upravljanje bazama podataka, oni podržavaju ***osnovne tipove podataka***, kao što su: celi (*int*) i realni (*real*) brojevi, karakteri (*chr*), datum (*date*), novac (*money*) i logički podaci, koji uzimaju vrednosti iz skupa {*true*, *false*}. Takođe, podržavaju tipske konstruktore za izgradnju složenijih tipova podataka, kao što su: nizovi, liste, slogovi i skupovi. Elementi domena, pridruženog obeležju, predstavljaju se putem nekog od osnovnih tipova podataka. Za svaki osnovni tip podataka, karakterističan je jedan broj operacija. Te operacije su ugrađene u razne programe.

Primer 7.2. Elementi domena, pridruženog obeležju *IME*, predstavljaju se putem karaktera ili niza karaktera, a elementi domena, pridruženog obeležju *KOL* (količina materijala), predstavljaju se putem celih ili realnih brojeva. Primeri operacija nad ovim tipovima podataka su, redom: povezivanje nizova karaktera, primena aritmetičkih operacija na cele ili realne brojeve. □

7.2.2. Pojam objekta

Objekat je jedan od mogućih modela realnog entiteta. Taj model ima dve komponente. To su: stanje i ponašanje. ***Stanje*** objekta se reprezentuje putem strukture nad skupom podataka o realnom entitetu. ***Ponašanje*** objekta je skup procedura, koji se

nazivaju ***metodama*** ili *operacijama*. Ti programi služe za izmenu stanja objekta (ažuriranje strukture podataka) ili samo za davanje informacije o njegovom stanju. Bez insistiranja na preciznosti, metode se mogu posmatrati i kao programi, koji opisuju ponašanje realnog entiteta.

Primer 7.3. Ako je Ivo Ban programer sa platom od *400* dinara, tada bi sledeća semantički određena linearna struktura podataka mogla predstavljati strukturalni deo odgovarajućeg objekta

$$((IME, Ivo), (PRZ, Ban), (ZAN, programer), (PLT, 400)),$$

a skup

$$\{zaposli, povećaj\_platu, prikaži\_platu, prikaži\_podatke, penzioniši\}$$

bi sadržao nazive metoda, koji se mogu primeniti na tu strukturu podataka. □

Mada je objekat dvojka (stanje, ponašanje), često se eksplicitno naglašava samo njegova strukturalna komponenta, dok se postojanje skupa metoda implicitno podrazumeva. Opravdanje za takvu nepreciznost, može se tražiti u činjenici da svi objekti, modeli realnih entiteta jedne klase, dele iste metode, a da struktura, u određenoj meri, odražava individualnost objekta. Naredna definicija opisuje moguće strukture objekta, koristeći termin objekat u smislu struktura podataka objekta.

Definicija 7.1. Neka je U skup obeležja, a D = {ceo broj, realan broj, niz karaktera fiksne ili promenljive dužine, datum, novac,...} skup osnovnih tipova podataka. Tada važi:

1° Svaki elemenat svakog osnovnog tipa podataka je ***primitivni objekat***.

2° Ako su: $X_1,..., X_n$ različita obeležja u U, a $o_1,..., o_n$ objekti, tada je

$$o = ((X_1, o_1),..., (X_n, o_n))$$

jedan ***objekat - torka***. Objekat o_i je vrednost obeležja X_i, u oznaci $o_i = o.X_i$.

3° Ako su $o_1,..., o_n$ različiti objekti, tada je

$$o = \{o_1,..., o_n\}$$

jedan ***objekat - skup*** i važi $o_i \in o$. □

Definicija 7.1 ukazuje na veoma širok dijapazon mogućnih struktura podataka objekta. Objekat može imati tako jednostavnu strukturu, kakvu ima jedan ceo broj, ili kompleksniju, kakva je relaciona torka ili skup objekata, do veoma kompleksne strukture, kakva je struktura stabla nad skupom objekata. Bitno je zapaziti da obeležja iz U mogu uzimati vrednosti iz proizvoljnog domena. Vrednost obeležja može biti ceo broj ili niz karaktera, ali i torka, skup, ili neka kompleksna struktura podataka.

Primer 7.4. Pošto broj *5* pripada skupu celih brojeva, broj *5* je jedan primitivni objekat. Slično, pošto "*Ana*" pripada skupu nizova karaktera, i "*Ana*" je primitivni objekat.

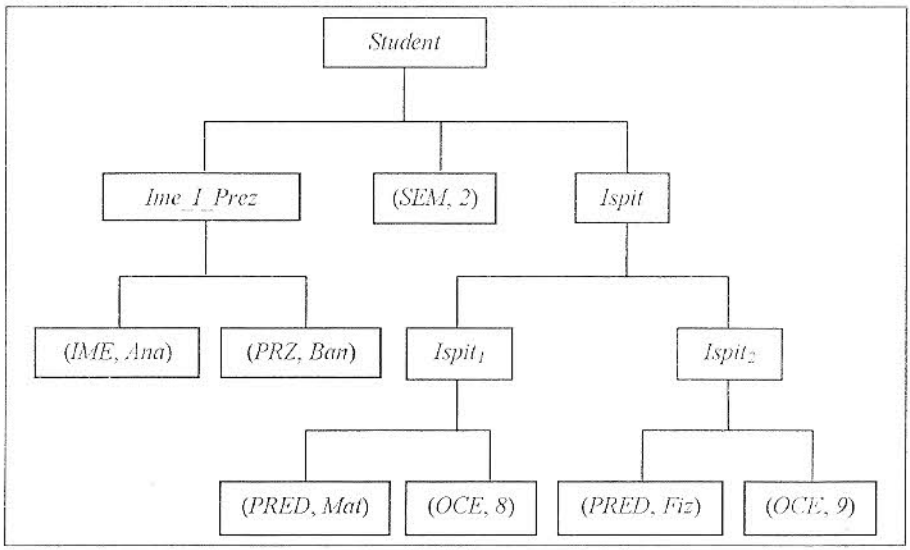
Objekat ((*IME, Ana*), (*PRZ, Ban*)) je objekat - torka, koji sadrži dva primitivna objekta, a objekat {((*PRED, Matematika*), (*OCE, 8*)), ((*PRED, Fizika*), (*OCE, 9*))} je objekat skup. Objekat

((*IME_I_PREZIME*(*IME, Ana*), (*PRZ, Ban*)), (*SEM, 2*), (*ISPIT*, {((*PRED, Matematika*), (*OCE, 8*)), ((*PRED, Fizika*), (*OCE, 9*))}))

reprezentuje realni entitet - studenta Anu Ban, putem jednog složenog objekta - torke. Objekat je složen, jer sadrži:

- jedan objekat - torku tipa *Ime_i_Prezime* (koji, sa svoje strane sadrži dva primitivna objekta),
- jedan primitivni objekat tipa ceo broj, (koji reprezentuje upisani semestar) i
- jedan objekat - skup, čiji elementi, objekti - torke reprezentuju položene predmete sa ocenama.

Na slici 7.4 je ovaj objekat prikazan kao struktura stabla nad skupom objekata. Pri tome, listovi stabla sadrže primitivne objekte, dok drugi čvorovi predstavljaju agregate. Agregati su, na slici 7.4, samo naznačeni putem svojih naziva, mada i sami sadrže podatke. □

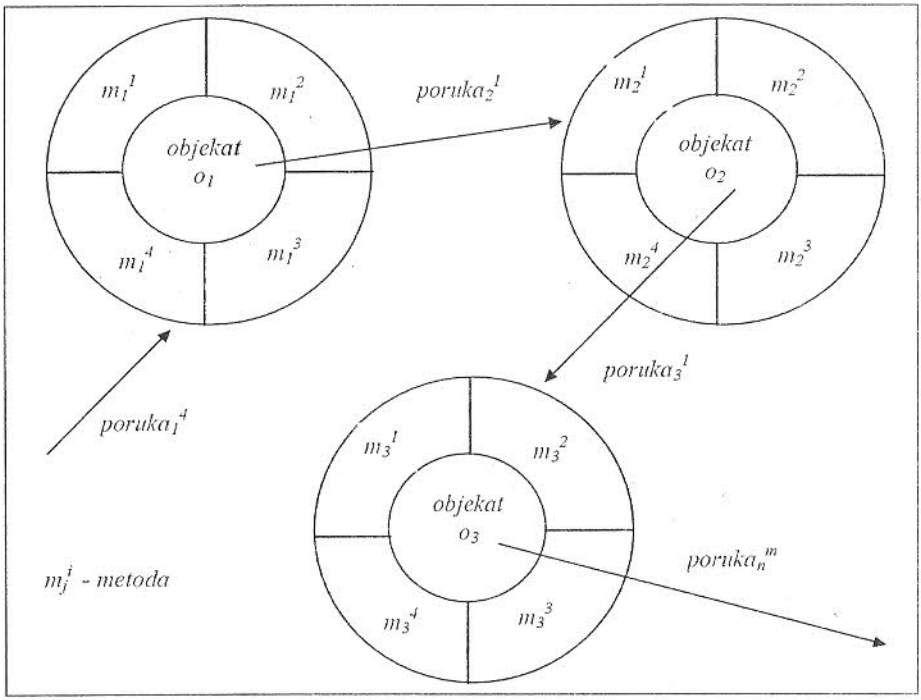


Slika 7.4.

Putem definicije 7.1 je opisana logička struktura podataka objekta, koja se može značajno razlikovati od njegove memorijske reprezentacije - fizičke strukture. Mogućim fizičkim realizacijama logičke strukture podataka objekta, više prostora je posvećeno u delu 7.4 ovog poglavlja.

7.2.3. *Poruka*

Jedino metode, pridružene posmatranom objektu, imaju pravo da pristupaju njegovoj strukturi podataka, bilo u cilju ažuriranja stanja objekta, ili samo davanja informacije o nekoj komponenti njegovog stanja. Metode se pozivaju putem ***poruka***. Poruka je zahtev, upućen objektu, da izvrši neku od svojih metoda. Objekti međusobno komuniciraju putem poruka.



Slika 7.5.

Osnovni oblik poruke je (<prijemnik>, <naziv metode>). Deo poruke, označen sa <prijemnik>, predstavlja adresu objekta, koji treba da primi i protumači poruku. Ta adresa je ili identifikator objekta, ili neki izraz, putem kojeg se objekat označava. Naime, svaki objekat ima jedan ***identifikator***, koji ga jednoznačno identifikuje. Drugi objekti se obraćaju posmatranom objektu navođenjem tog identifikatora u okviru poruke. Za svaku poruku, koju objekat razume, postoji odgovarajuća metoda, koja izvršava poruku. Deo poruke, označen sa <naziv metode>, ukazuje o kojoj metodi je reč. Na osnovu naziva, objekat poziva odgovarajuću metodu. Poruka može sadržati i parametre, neophodne za izvršavanje metode. Objekat vraća pošiljaocu rezultat izvršenja metode, pozvane porukom. Rezultat je

obično neki drugi objekat. Na slici 7.5 je prikazan princip komunikacije između objekata. Prema slici 7.5, objekat o_1 prima poruku p^1_4 i, da bi na nju odgovorio, odlučuje da uputi poruku p^2_1 objektu o_2. Objekat o_2 , sa svoje strane, šalje poruku p^3_1 objektu o_3.

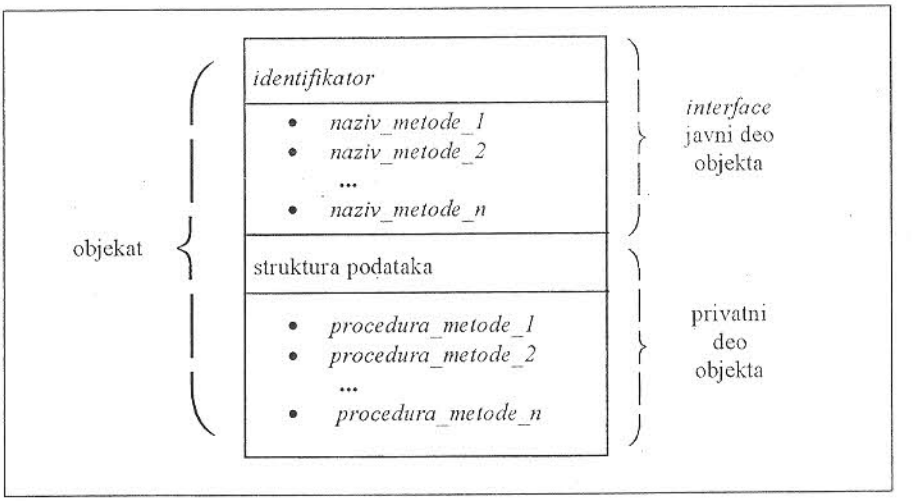
Paradigma objekat / poruka pretpostavlja:

- da svi podaci predstavljaju objekte,
- da svaki objekat poseduje skup metoda,
- da su nazivi tih metoda poznati drugim objektima,
- da svaki objekat odgovara samo na svoje poruke i
- da objekti međusobno komuniciraju samo putem poruka.

Metoda je entitet sličan programskoj proceduri. Opisuje redosled akcija, koje treba izvršiti po prijemu poruke. Više prostora opisu postupaka za izgradnju metoda, dato je u delu 7.6 ovog poglavlja.

7.2.4. *Inkapsulacija*

Svaki objekat ima svoj privatni i svoj javni deo. ***Privatni deo*** objekta čine: struktura podataka i skup metoda za tu strukturu podataka. Taj deo objekta se naziva i ***implementacijom.*** ***Javni*** deo objekta čine: identifikator i nazivi metoda. To je njegov ***interfejs*** (sprega sa okolinom). Poruke pristupaju interfejsu i specificiraju koje metode bi, na objektu, trebalo da se izvrše, ali ne i kako da se te metode izvrše. Objekat, koji primi poruku, određuje kako će se izvršiti tražena metoda. Na slici 7.6, ilustrovani su pojmovi privatnog i javnog dela objekta.



Slika 7.6.

Uvođenjem privatnog dela objekta, uvodi do *princip skrivanja informacija*. To znači da su detalji realizacije objekta skriveni od svih programa izvan objekta. Često se kaže da objekat *inkapsulira* podatke i programe. To znači da jedan objekat ne može da vidi unutrašnjost "kapsule" drugog objekta, ali može da ga koristi, pozivajući, putem poruka, njegov programski deo. To je slično pozivima procedura u konvencionalnim programskim jezicima. Inkapsulaciju treba shvatiti i kao ograničenje da se svi pristupi objektima mogu vršiti jedino primenom njihovih metoda.

7.2.5. Klasa

Mnogi objekti u realnom svetu poseduju slične karakteristike i izvršavaju slične operacije. U proizvodnom pogonu metalskog kompleksa, nalaze se glodalice, rendisaljke, bušilice. Mada je svaki od ovih objekata različit, svi pripadaju klasi mašina za obradu metala rezanjem. Svi objekti u klasi mašina za obradu metala rezanjem poseduju zajednička obeležja (električni motor, glavno vreteno, na primer) i izvršavaju zajedničke operacije (obrada metala). Prema tome, kategorizacijom jednog struga, kao člana klase mašina za obradu rezanjem, zna se nešto o njegovim obeležjima i operacijama, čak i kada se ne poznaju njegove precizne funkcije.

Objektno - orijentisani model podataka se ne svodi samo na pojam objekta. Osnovni pojam objektno - orijentisanog modela podataka je *klasa*. Klasa je dvojka. Jedna komponenta te dvojke je *tip objekta*, koji nosi informaciju o strukturi podataka. Druga komponenta je skup metoda. To su operacije, koje se primenjuju na sve objekte, čija struktura je definisana putem tipa objekta. Klasa je skup takvih objekata, koji imaju iste karakteristike (tip strukture i metode). Svaki individualni objekat je *pojava* neke klase.

Definicija 7.2. Neka je U skup obeležja, $X \subset U$, $A \in U$, a D = {ceo broj, realan broj, niz karaktera fiksne ili promenljive dužine, datum, novac,...} skup osnovnih tipova podataka. Tada važi:

1° Dvojka (A, d), gde je $d \in D$, je ***tip objekta***.
2° Ako su $T_1,..., T_k$ tipovi objekata, tada je i $((X_1, T_1),...,(X_k, T_k))$ ***tip objekta torka***.
3° Ako je T tip objekta, tada je i $\{T\}$ ***tip objekta skup***. Objekat tipa $\{T\}$ je skup objekata tipa T. □

Komponenta tipa objekta je dvojka (obeležje, tip podataka). Putem obeležja je definisana semantika komponente tipa objekta. Tip podataka je domen obeležja, ali elementi tog domena mogu biti veoma složeni. Elementi domena mogu uzimati vrednosti iz skupa vrednosti nekog osnovnog tipa podataka, ali i iz skupa objekata tipa torka ili skup. Pošto su i elementi osnovnih tipova podataka objekti, sa svojom strukturom i ponašanjem, dolazi se do zaključka da domen predstavlja klasu. Domen se, neki put naziva i *interpretacijom*. Interpretacija je skup, iz kojeg neki semantički koncept uzima vrednosti. Treba zapaziti da je definicija pojma tipa objekta rekurzivna. Tip objekta se definiše kao struktura nad skupom tipova objekata.

Primer 7.5. Tip objekta, pod nazivom *TipArtikla* se, koristeći uvedenu notaciju, može definisati na sledeći način

$$TipArtikla = ((NAZIV, Chr), (IDA, Int))$$

gde je *IDA* identifikacioni broj artikla. Nazivi domena obeležja *NAZIV* i *IDA* pisani su velikim početnim slovima, da bi se ukazalo da je reč o tipovima, odnosno klasama. Tip objekta *StavkaNrudžbine* je

$$StavkaNarudžbine = ((ARTIKAL, TipArtikla), (KOLIČINA, Int)).$$

U komponenti (*ARTIKAL*, *TipArtikla*) tipa objekta *StavkaNarudžbine*, *ARTIKAL* je obeležje, a *TipArtikla* je klasa. Vrednosti obeležja *ARTIKAL* su pojave klase *TipArtikla*. Sada se može definisati tip objekta *TipPorudžbine*, kao

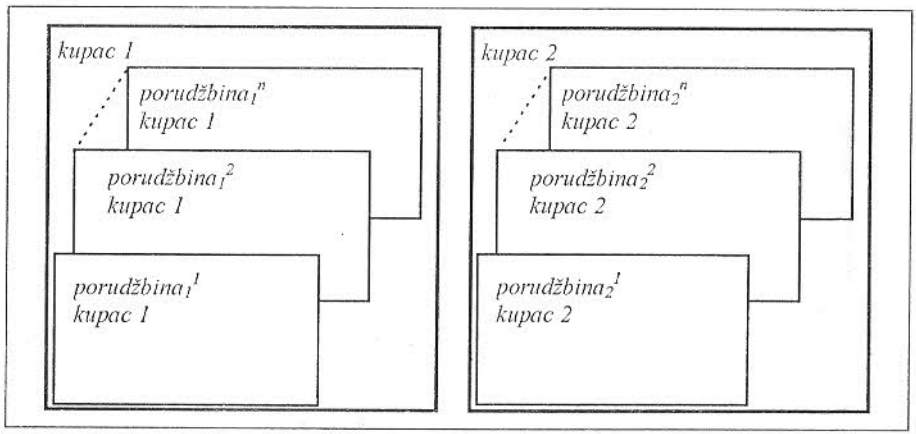
$$TipPorudžbine = ((BRP, Int), (SADRŽI, \{StavkaNarudžbine\}),$$

gde je *BRP* broj porudžbine i gde je obeležju *SADRŽI* pridružen tip podatka - domen, čiji elementi su skupovi objekata tipa *StavkaNarudžbine*. Kupci se mogu predstaviti putem sledećeg tipa objekta

$$TipKupca = ((NAZIV, Chr), (ADRESA, Chr), (STANJE\_RAČ, Int), (PORUDŽBINE, \{TipPorudžbine\})).$$

Da bi se uspostavila veza između narudžbina i kupaca, tip objekta *TipPorudžbine* se može modifikovati na sledeći način

$$TipPorudžbine = ((BRP, Int), (KUPAC, TipKupca), (SADRŽI, \{StavkaNarudžbine\}).$$



Slika 7.7.

Treba zapaziti da tip objekta *TipKupca* sadrži, kao svoju komponentu, skup pojava klase *TipPorudžbine*, a da tip objekta *TipPorudžbine*, kao svoju komponetu, sadrži

jednu pojavu klase *TipKupca*. Na slici 7.7 su prikazane dve pojave klase *Kupac*, svaka sadrži određeni broj odgovarajućih pojava klase *Porudžbina*.□

Klasa se opisuje putem naredbi objektno - orijentisanog jezika. To opisivanje klase se naziva ***deklaracijom*** klase. U okviru deklaracije klase se navode:

- nazivi njenih obeležja sa pripadajućim domenima i
- nazivi metoda sa odgovarajućim programskim kodom.

Putem naziva obeležja i domena se definiše tip objekta, a nazivi obeležja i metoda čine interfejs klase. Primeri deklaracija klasa u sintaksi objektno - orijentisanog jezika C++ su dati u tački 7.6.

Klasa je generator svojih pojava (objekata istog tipa). Skupu aktuelnih objekata, odgovara pojam ***ekstenzije*** u klasičnim modelima podataka. U svakom objektno - orijentisanom sistemu postoji mehanizam za generisanje pojava klase. Pošto je i sama klasa vrsta objekta, ona bi mogla da vodi računa o svim objektima, koje je generisala, te i da upravlja svojom ekstenzijom. Međutim, objektno - orijentisani jezici, kakvi su Smalltalk i C++, eksplicitno ne podržavaju pojam ekstenzije klase. Ne postoji eksplicitna mogućnost za uvid i manipulisanje svim aktuelnim pojavama klase. Svakoj pojavi klase se pristupa samo na osnovu poznavanja njenog identifikatora. Ne postoji naredba sa efektom SQL naredbe SELECT *. S druge strane, pojam ekstenzije je izuzetno važan za sisteme baza podataka. Jedan od njihovih osnovnih zadataka je da obezbede efikasnu obradu velikog broja objekata istog tipa. Relacioni sistemi za upravljanje bazom podataka obezbeđuju upravljanje ekstenzijom, jer je sam model podataka građen tako, da se definisanjem šeme relacije (tipa) obezbeđuje pristup svim aktuelnim pojavama.

U većini objektno - orijentisanih jezika rad sa ekstenzijom klase se postiže putem objekata tipa ***kolekcija***. Kolekcije su klase, ugrađene u sam jezik, tako da nije potrebno posebno ih definisati. Postoji više vrsta kolekcija. U njih spadaju: ***skup*** i ***torba***. Torba je kolekcija u kojoj se može nalaziti više jednakih objekata. Objekat tipa kolekcija se može upotrebiti u cilju postizanja efekta upravljanja ekstenzijom. Objekti tipa kolekcija sadrže druge objekte. Postupak rada sa kolekcijom u ulozi ekstenzije je sledeći. Prvo se definiše jedna pojava (objekat) klase kolekcija. Toj pojavi se pridružuje ime (semantika) ekstenzije neke klase. Pošto je reč o objektu tipa skup objekata, on će sadržati sve one pojave (elemente ekstenzije), koji se u njega eksplicitno upišu.

Primer 7.6. Neka je kreirana klasa *Zaposleni* i neka je potrebno omogućiti upravljanje njenom ekstenzijom. Tada se prvo definiše nova pojava ugrađene klase tipa kolekcija. Ako je *Radnici* naziv ekstenzije klase *Zaposleni*, tada bi se *Radnici* deklarisali kao nova pojava ugrađene klase, recimo, klase *Skup*. Svaki objekat klase *Zaposleni*, nakon svog kreiranja u okviru klase *Zaposleni*, mora se eksplicitnom naredbom uvrstiti i u objekat - skup sa identifikatorom *Radnici*. Metode ugrađene klase *Skup*, omogućavaju upravljanje ekstenzijom klase *Zaposleni*, pošto je objekat sa identifikatorom *Radnici* pojava klase *Skup*. □

7.2.6. *Apstraktni tipovi podataka*

Apstraktni tipovi podataka predstavljaju uopštenje pojma osnovnog tipa podatka. Koncepti:

- inkapsulacije,
- javnog i privatnog dela,
- poruke i metode,

u osnovi su vezani za pojam apstraktnih tipova podataka. U objektno - orijentisanom pristupu, ovi koncepti se realizuju putem klasa. Saglasno tome, i apstraktni tip podatka se može posmatrati kao dvojka (s, m), gde je s struktura, a m skup metoda. Međutim, između pojmova apstraktni tip podatka i klasa, postoji i određena razlika. Apstraktni tip podataka opisuje skup svih mogućih objekata sa datom strukturom i datim ponašanjem. Taj skup može biti neograničen. Nasuprot apstraktnom tipu podataka, klasa opisuje objekte svoje ekstenzije, koja sadrži konačan broj pojava klase.

7.2.7. *Nasleđivanje*

Objekti nasleđuju osobine (tip strukture i metode) od svoje klase. Međutim, i klasa može nasleđivati osobine od drugih klasa. Klasa, koja nasleđuje osobine od druge klase, naziva se ***potklasom***. Klasa, čije osobine nasleđuje jedna ili više drugih klasa, naziva se ***superklasom***. U literaturi se, za potklasu, sreću i nazivi: potomak ili dete, a za superklasu: predak ili roditelj. Svaka potklasa nasleđuje sve ili samo deo osobina svoje superklase. Nasleđene osobine nije potrebno ponovo definisati u potklasi. Međutim, nove osobine se mogu dodati potklasama, ako je to potrebno. Potklasa nasleđuje strukturu tako da sva obeležja superklase postaju obeležja potklase. Potklasa nasleđuje metode, tako da njeni objekti mogu da koriste sve metode superklase. Međutim, potklasa može imati i svoje metode, neke od metoda superklase mogu biti zabranjene za objekte potklase, a neka od nasleđenih metoda se može i prilagoditi za potklasu. Pošto potklasa može imati svoje potklase, dolazi se do pojma hijerarhije klasa. ***Hijerarhija klasa*** definiše "je podvrsta" odnos između klasa. Koncept nasleđivanja kaže da osobine, definisane za neku klasu, mogu naslediti sve potklase te klase. To je slično konceptu specijalizacije u semantičkim modelima podataka.

Korišćenje klasa i nasleđivanja je veoma bitno za savremeno softversko inženjerstvo. ***Ponovno korišćenje*** programa se postiže kreiranjem novih klasa, zasnovanih na obeležjima i metodama postojećih klasa. Tada treba samo da se specificira kako se nova potklasa razlikuje od svoje superklase, a ne i da se definiše kompletno nova klasa. Nova klasa nasleđuje već proverene metode od svojih superklasa i te metode ne treba ponovo programirati.

7.2.8. Identitet objekta

U objektno orijentisanom modelu podataka, svakom objektu, osim primitivnom, pridružuje se jedinstveni identifikator. Primitivnom objektu se identifikator ne pridružuje, jer primitivni objekti imaju jedinstvene vrednosti, koje ih samoidentifikuju. U svojstvu identifikatora se može koristiti adresa memorijske lokacije, u koju je objekat smešten, ali postoje i druga rešenja. Važno je da je identifikator nezavisan od bilo kojeg obeležja tipa objekta, pa i ključa. Na taj način se omogućava ažuriranje svakog obeležja (promena stanja) objekta bez opasnosti da se naruši identitet objekta. Identifikator ostaje nepromenjen dok se objekat ne izbriše iz baze podataka. Identifikator se koristi kao referenca na objekat, kao surogat - zamena za objekat. Identifikator obezbeđuje jedinstvenost *identiteta* objekta, ali, ako je to memorijska adresa, nju korisnik ne može poznavati. Postoje i druga rešenja za realizaciju identiteta objekta. Ta rešenja su opisana u tački 7.4.

7.2.9. Paralela između termina objektno - orijentisanog i drugih modela podataka

Smatra se da objektna - orijentacija predstavlja ne evolutivni, već revolutivni skok u razvoju ne samo baza podataka već kompletnog softvera. Revolucionarnost objektnog modela podataka u odnosu na druge, ogleda se, pre svega, u načinu izgradnje softverske reprezentacije realnih entiteta. Novi model sadrži i čitav niz novih pojmova sa specifičnim nazivima. Za lakše shvatanje sličnosti, ali i razlika između koncepata objektnog i drugih modela podataka, pogodno je povući paralelu između sličnih, ali nikako identičnih pojmova i njihovih naziva.

Terminologija

o - o model	*tumačenje*	*drugi modeli podataka*
Objekat	apstraktna predstava nekog realnog entiteta	*pojava tipa entiteta, torka*
Klasa	dvojka (tip objekta, skup metoda), koja generiše objekte sa zajedničkim osobinama	*tip entiteta, šema relacije*
Apstraktni tip podataka	opis skupa objekata sa istim tipom strukture i ponašanjem	
Hijerarhija klasa	struktura, dobijena uvođenjem relacije "je podvrsta" u skup klasa	*šema baze podataka*
Osobina	neka rutina, aktivnost ili obeležje, koje služi kao deo definicije klase	*obeležje*

o - o model	*tumačenje*	*drugi modeli podataka*
Metoda	osobina, koja obavlja neku operaciju na objektu	*telo procedure*
Promenljiva pojave	obeležje, kao osobina objekta	*polje*
Pojava klase	objekat	*pojava tipa entiteta, torka*
Poruka	protokol, koji se sastoji od metode ili rutine i identifikatora (ili adrese) jednog objekta	*poziv procedure*
Skrivanje informacija	odvajanje javnog od privatnog dela objekta ili klase	
Inkapsulacija	sakrivanje informacija	
Nasleđivanje	takva relacija između dve klase, kod koje jedna klasa, dete, preuzima sve relevantne osobine druge klase, roditelja	*ISA hijerarhija*

7.3. *Nasleđivanje*

Objektna orijentacija ima dva cilja. To su:

- izgradnja što je moguće vernijeg modela realnog sveta i
- obezbeđenje uslova za proširljivost i ponovno korišćenje softvera.

Oba cilja se postižu putem koncepta ***nasleđivanja***. Naime, hijerarhije nasleđivanja predstavljaju prirodan mehanizam za izgradnju taksonomijske strukture realnog sistema. (Taksonomijska struktura predstavlja sistematizaciju delova realnog sistema.) Takođe, putem nasleđivanja grade se nove klase na osnovu postojećih. Time se izbegava projektovanje struktura i programiranje metoda novih klasa uvek iz početka. Nove klase nasleđuju ponašanje i obeležja od postojećih. Nasleđivanjem ponašanja postiže se deljeno korišćenje programskog koda, a time ponovno korišćenje softvera.

Nasleđivanje vuče korene od postupaka predstavljanja znanja u veštačkoj inteligenciji. U veštačkoj inteligenciji se nasleđivanje predstavlja putem *IS_A* (je) relacije i ograničeno je na nasleđivanje strukture, kako na nivou intenzije, tako i na nivou ekstenzije. Specijalizacija predstavlja najčešće korišćeni postupak za realizaciju nasleđivanja.

U objektno - orijentisanim jezicima, nasleđivanje se odnosi, pre svega, na klase, znači intenziju. Potklasa nasleđuje osobine (tip strukture i metode) od svoje superklase, tako da potklasa predstavlja intenzionalnu specijalizaciju svoje superklase. Pojava potklase se može posmatrati kao ekstenzionalna specijalizacija svoje klase, međutim, pojava potklase nije specijalizacija odgovarajuće pojave superklase. Šta više, ta odgovarajuća pojava superklase ne mora ni postojati.

Relacije "je potklasa" i "je superklasa" su tranzitivne. Saglasno tome, ako je X potklasa klase Y, a Y potklasa klase Z, tada je X potklasa klase Z. Pošto potklasa može imati svoje potklase, dolazi se do pojma hijerarhije klasa. ***Hijerarhija klasa*** definiše "je vrsta" odnos između klasa. Koncept nasleđivanja kaže da osobine, definisane za neku klasu, nasleđuju sve potklase te klase u hijerarhiji. To je slično konceptu specijalizacije u semantičkim modelima podataka.

Primer 7.7. Neka je klasa *Osoba* definisana kao sledeći tip objekta

$$((IME, Chr), (PRZ, Chr), (DAT\_ROĐ, Date)),$$

i sledeći skup metoda

$$\{prikaži\_ime, prikaži\_prezime, prikaži\_godine, izmeni\_prezime\}.$$

Ako se žele imati i objekti, koji reprezentuju zaposlene, potrebno je kreirati klasu *Zaposleni*, kao potklasu klase *Osoba*. Objekti klase *Zaposleni* će, automatski naslediti obeležja *IME*, *PRZ* i *DAT_ROĐ*. Tipu objekta klase *Zaposleni* se može dodati obeležje *PLATA* sa domenom *money*. Nasleđeni skup metoda se može proširiti metodama: *zaposli*, *otpusti*, *povećaj_platu*, *prikaži_platu*. Ako neki objekat klase *Zaposleni* primi poruku *prikaži_ime*, odgovoriće na nju pozajmljujući odgovarajuću metodu od klase *Osoba*. Poruka *prikaži_godine* se može zabraniti za klasu *Zaposleni*.

Postupak definisanja novih klasa, kao potklasa već postojećih, može se produžiti definisanjem klase *Programer* kao potklase klase *Zaposleni*. Sada se novoj klasi može pridružiti obeležje *PROG_JEZ* sa domenom {*Prog_jez*}, gde je *Prog_jez* tip objekta klase sa istim imenom. □

Postoji sedam aspekata nasleđivanja. To su:

- ***Nasleđivanje i podtip:*** U mnogim objektno - orijentisanim jezicima se pojmovi nasleđivanja i podtipa posmatraju kao sinonimi. Ipak, korisno ih je posmatrati kao posebne koncepte. Podtip je logička kategorija. Nasleđivanje je samo mehanizam za programsku realizaciju tog logičkog koncepta.
- ***Vidljivost nasleđenih obeležja i metoda.*** Obeležja superklase mogu biti javna i privatna. Privatna obeležja su nevidljiva za potklasu, te ih ne može ni naslediti.
- ***Nasleđivanje i inkapsulacija.*** Vidljivost obeležja narušava inkapsulaciju. U stvari, postoji konflikt između nasleđivanja i inkapsulacije, jer objekat - pojava potklase, koristi metode superklase, koja za njega predstavlja tuđu klasu. U svakom slučaju, nasleđivanje je, pre svega, mehanizam za deljenje programskog koda i tipa strukture.
- ***Postupci specijalizacije.*** Nasleđivanje se ostvaruje specijalizacijom postojećih klasa. Klase se specijalizuju proširenjem njihove strukture ili ponašanja. Alternativno, klase se mogu specijalizovati i ograničavanjem skupa obeležja ili skupa metoda postojećih klasa.
- ***Metaklase.*** Uvođenjem koncepta metaklase, kao klase, koja opisuje skup klasa, klase tog skupa postaju objekti, pojave metaklase.

- *Nasleđivanje objekata.* Većina objektno - orijentisanih jezika podržava nasleđivanje samo na nivou intenzije, a ne i na nivou ekstenzije. Kod takozvanih prototipskih sistema, objekat, kao pojava potklase, nasleđuje objekat, pojavu superklase.
- *Višestruko nasleđivanje.* U mnogim situacijama je poželjno da jedna potklasa nasledi osobine od više superklasa. To se naziva višestrukim nasleđivanjem.

Korišćenje klasa i nasleđivanja je veoma bitno za savremeno softversko inženjerstvo. ***Ponovno korišćenje*** programa se postiže kreiranjem novih klasa, zasnovanih na obeležjima i metodama postojećih klasa. Treba samo da se specificira kako se nova potklasa razlikuje od svoje superklase, a ne da se definiše kompletno nova klasa. Potklasa nasleđuje već proverene metode od svojih superklasa i te metode ne treba ponovo programirati.

7.3.1. Podtipovi

Tip - podtip je semantički odnos između dva tipa objekata. Nasleđivanje je programski mehanizam, putem kojeg se u objektno - orijentisanim jezicima realizuje semantički koncept podtipa.

Definicija 7.3. Tip T_1 je ***podtip*** tipa T_2, ako svaka pojava tipa T_1 predstavlja pojavu i tipa T_2. Tada je i tip T_2 ***supertip*** za tip T_1. □

Definicija koncepta podtipa ukazuje da se pojava podtipa može koristiti kadgod se u nekoj operaciji, ili u nekoj konstrukciji očekuje pojava supertipa. Ovaj fenomen se naziva ***principom zamenljivosti.***

Primer 7.8. Skup prostih brojeva je podtip skupa prirodnih brojeva. Svaka operacija, definisana za elemente skupa prirodnih brojeva, primenljiva je i na elemente skupa prostih brojeva. □

Relacija "je podtip" je:

- po definiciji, refleksivna (svaki tip je sam sebi podtip),
- antisimetrična (ako je tip T_1 podtip tipa T_2, a tip T_2 podtip tipa T_1, tada je $T_1 = T_2$) i
- tranzitivna (ako je T_1 podtip tipa T_2, a T_2 podtip tipa T_3, tada je i T_1 podtip tipa T_3).

Parcijalni poredak, koji ova relacija uvodi u skup tipova posmatranog sistema, u objektno - orijentisanim jezicima se realizuje putem hijerarhije nasleđivanja. U opštem slučaju, hijerarhija nasleđivanja je usmereni aciklički graf - *latisa*.

Odnos tip - podtip se može uspostaviti između:

- skupova,
- strukturiranih tipova i
- metoda.

7.3.2. Podskupovi kao podtipovi

Tip je reprezent skupa objekata sa istim osobinama. Saglasno tome, prirodni kandidati za podtipove su podskupovi. Pri tome, tip predstavlja ne samo skup objekata, već i skup operacija nad tim objektima. Naravno, operacije se realizuju putem metoda. Kada je reč o primeni operacija na objekte podskupa, mogu se javiti dva slučaja. Jedan je, kada se operacije na objektima podskupa ponašaju na isti način kao te iste operacije (metode) na objektima skupa. Za tip T_1 se kaže da je ***kompletan podtip*** tipa T_2, ako je ponašanje operacija isto bilo da je reč o objektima tipa ili podtipa.

Drugi je slučaj, kada se operacije ponašaju različito pri primeni na objekte skupa i podskupa. Različito ponašanje se ogleda u činjenici da određena ograničenja, koja se postavljaju na rezultate primene operacija na objekte tipa, nisu zadovoljena pri primeni istih metoda na objekte podtipa. Rešenje ovog problema se traži u relaksaciji ograničenja pri primeni metoda na objekte podtipa.

Primer 7.9. Neka je C skup celih brojeva, P skup prostih brojeva, a E skup parnih brojeva. Uobičajene aritmetičke operacije u skupu C su: sabiranje, oduzimanje i množenje. Sa formalno - algebarske tačke gledišta, pri opisivanju metoda za ove operacije, treba voditi računa o:

1° ***signaturi*** operatora (nazivu, tipovima podataka operanada i rezultata) i
2° aksiomama, koje definišu semantiku operacija.

U aksiome spadaju: asocijativnost i komutativnost sabiranja i množenja, distributivnost množenja u odnosu na sabiranje, na primer. Pored ovih algebarskih osobina, operacije sabiranja, oduzimanja i množenja u skupu C poseduju i važnu osobinu ***zatvaranja***. Ove operacije preslikavaju parove celih brojeva u skup celih brojeva. Pri primeni operacija sabiranja, oduzimanja i množenja na objekte skupa P, dolazi do narušavanja ograničenja, koje nameće osobina zatvaranja. Skup P nije kompletan podtip tipa C. Za razliku od njega, skup E je kompletan podtip tipa C.

Rešenje opisanog problema se postiže takvim redefinisanjem metoda u skupu P, da se ograničenje, koje nameće zatvaranje, izostavlja, a signature operacija se definišu kao preslikavanje sa skupa parova prostih brojeva u skup celih brojeva. □

7.3.3. Strukturirani tipovi kao podtipovi

Neka su $T_1,..., T_n$ tipovi, a $X_1,..., X_n$ obeležja. Interpretacija svakog tipa T je njegov domen, u oznaci $dom(T)$. (Interpretacija je skup iz kojeg tip uzima vrednosti.) Tada je:

$$((X_1, T_1),..., (X_n, T_n))$$

tip torka, čiju interpretaciju predstavlja skup torki takvih, da svaka ima, barem za obeležja $X_1,..., X_n$, vrednosti, redom, iz domena $dom(T_1),..., dom(T_n)$. Treba zapaziti da u interpre-

taciji tipa mogu postojati i takve torke, koje pored vrednosti za obeležja $X_1,..., X_n$ poseduju i vrednosti za neka druga obeležja.

Primer 7.10. Posmatra se tip *Osoba*((*IME*, *Niz_chr*), (*PRZ*, *Niz_chr*), (*STAR*, *Int*)). Domen tipa *Niz_chr* je skup svih nizova karaktera, a domen tipa *Int* je skup celih brojeva. U domen tipa *Osoba* spadaju i ((*IME*, *Ana*), (*PRZ*, *Car*), (*STAR*, *23*)) i ((*IME*, *Eva*), (*PRZ*, *Tot*), (*STAR*, *48*), (*PLT*, *400*)). □

Definicija 7.4. Neka je " ≤ " relacija "je podtip".

10 Ako za svako $i = 1,..., n$, važi $T_i \leq S_i$, tada, za $n \leq m$, važi i

$$((X_1, T_1),..., (X_n, T_n),..., (X_m, T_m)) \leq ((X_1, S_1),..., (X_n, S_n)).$$

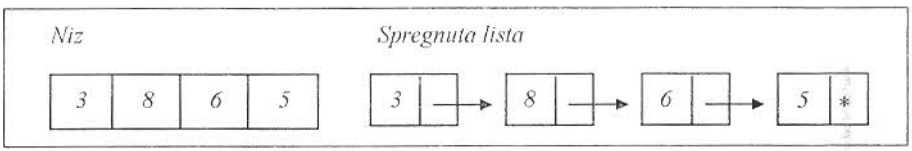
20 Ako su $S_1 = \{T_1\}$ i $S_2 = \{T_2\}$ tipovi i važi $T_1 \leq T_2$, tada važi i $S_1 \leq S_2$. □

Primer 7.11. Saglasno definiciji 7.4, važi

$$((IME, Niz\_chr), (PRZ, Niz\_chr), (PLT, \{1,..., 400\})) \leq ((IME, Niz\_chr), (PLT, Int)).$$ □

7.3.4. Nasleđivaje i podtipovi

Nasleđivanje je mehanizam objektno - orijentisanih programskih jezika, putem kojeg se realizuje relacija "je podtip". Međutim, relacija "je podtip" je semantička kategorija. Ona snabdeva skup tipova hijerarhijom ponašanja, a ne vodi računa o implementaciji (tip strukture podataka i metode). Nasleđivanje je implementaciona kategorija. Potklasa nasleđuje i tip strukture i metode od svoje superklase. Saglasno tome, nasleđivanje je uži pojam, pojam sa strožim ograničenjima od relacije "je podtip".



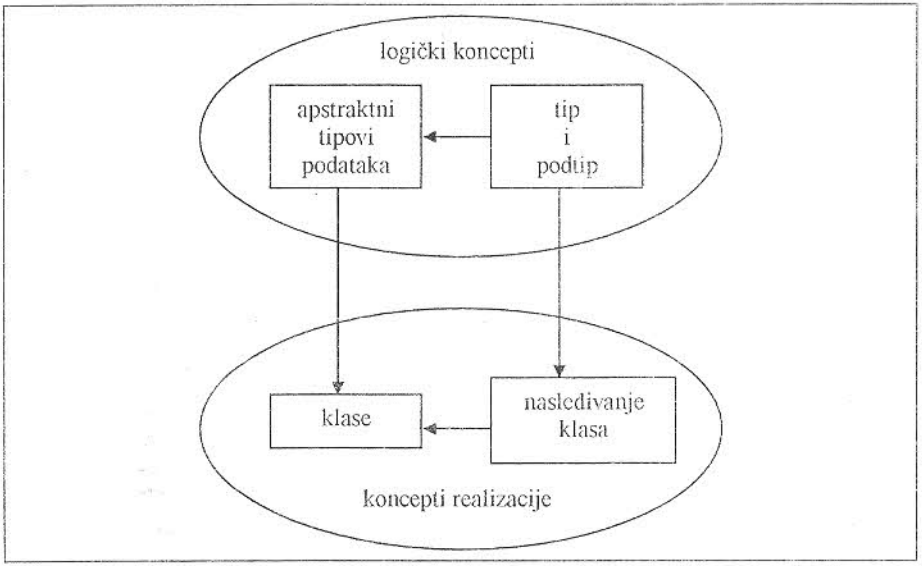
Slika 7.8.

Primer 7.12. *Skup*, kao tip, je podtip *Torbe*. *Torba* je kolekcija, koja može sadržati više jednakih objekata. Međutim, ta dva tipa mogu imati potpuno različite implementacije. Na primer, *Skup* može imati strukturu niza, a *Torba* strukturu spregnute liste, slika 7.8.

Zbog različite implementacije struktura podataka, po nazivu iste metode (unija, presek, razlika, upiši, briši, nađi), takođe moraju biti potpuno različito realizovane.

Da je reč o nasleđivanju, *Skup*, kao potklasa *Torbe*, nasledio bi i strukturu i metode od *Torbe*. □

Odnos između relacije nasleđivanja i relacije "je podtip", sličan je odnosu između klase i apstraktnog tipa podataka, slika 7.9. Kao što se apstraktni tipovi podataka realizuju putem klasa, tako se strukture tip - podtip realizuju putem nasleđivanja. Samo nasleđivanje može biti i selektivno, u smislu da se za potklasu neke osobine superklase mogu zabraniti.



Slika 7.9.

7.3.5. *Nasleđivanje klasa*

U objektno - orijentisanim jezicima, nasleđivanje klasa se eksplicitno deklariše. Putem odgovarajuće sintakse, nova klasa se deklariše kao potklasa neke postojeće klase. Pri tome, ako objektno - orijentisani jezik podržava višestruko nasleđivanje, jedna klasa može biti deklarisana kao potklasa više superklasa.

Potklasa može nasleđivati od svojih superklasa:

- obeležja sa njihovom fizičkom strukturom,
- metode, što vodi delenju i ponovnoj upotrebi programskog koda i
- interface, što znači da objekti potklase mogu pozivati metode superklase.

Nasleđivanje obeležja i metoda donosi određene probleme. U njih spadaju:

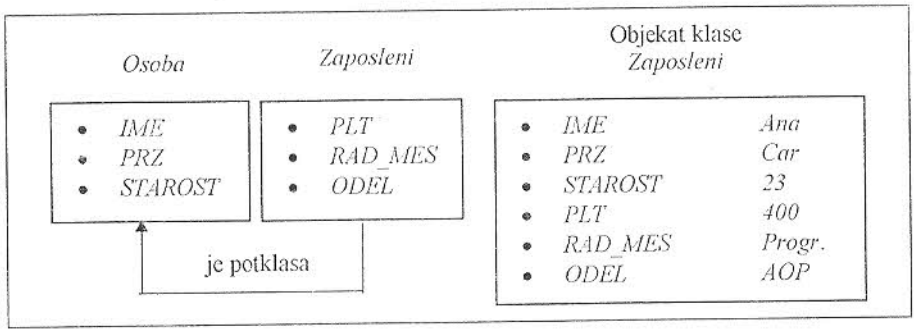
- narušavanje inkapsulacije,
- potreba da se nasleđene metode modifikuju u cilju prilagođavanja zahtevima potklase i
- potreba selektivnog nasleđivanja obeležja i metoda.

Ovim problemima će biti posvećena određena pažnja u narednim tačkama.

7.3.5.1. Nasleđivanje obeležja

Klase opisuju strukturu svojih objekata putem obeležja. U svim objektno - orijentisanim jezicima pojave potklase moraju, za nasleđena obeležja, zadržati isti tip podataka kao i pojave njihovih superklasa.

Neka su za klasu K_n deklarisane komponente (X_1, T_1),..., (X_n, T_n), gde su X_1 ,..., X_n obeležja, a T_1,...,T_n tipovi podataka. Ako se klasa K_m, sa komponentama (X_{n+1}, T_{n+1}),..., (X_m, T_m), deklariše kao potklasa klase K_n, objekti klase K_m će sadržati vrednosti za obeležja X_1,..., X_n, X_{n+1},..., X_m. Činjenica da objekat klase K_m sadrži vrednosti za obeležja klase K_n, ne znači da on te vrednosti nasleđuje od nekog objekta klase K_n, već da mu se te vrednosti eksplicitno dodeljuju, prilikom njegovog konstruisanja.



Slika 7.10.

Primer 7.13. Na slici 7.10 su prikazana obeležja klase *Osoba*, njene potklase *Zaposleni* i jedan objekat klase *Zaposleni*. □

Nasleđivanje obeležja i metoda dovodi do narušavanja inkapsulacije, jer objekti potklase koriste tuđe metode (metode svoje superklase) za obradu poruka, koje se odnose na nasleđena obeležja. Ova činjenica ima određene negativne posledice. Struktura objekta je definisana putem obeležja klase. Međutim, i obeležja se u implementaciji mogu realizovati putem različitih struktura. Metode striktno zavise od implementacije strukture obeležja. Pojam implementacije strukture obeležja se odnosi na način realizacije fizičke strukture podataka posmatranog obeležja. Izmena implementacione strukture obeležja, povlači izmenu programskog koda odgovarajućih metoda i modifikaciju svih objekata klase.

Primer 7.14. Domen obeležja može biti, između ostalog i neka klasa tipa skup. U implementaciji, tip objekta klase skup može imati strukturu bilo niza bilo spregnute liste. Programski kod svih metoda za to obeležje značajno zavisi od izbora implementacione strukture. Metoda za traženje elementa u nizu, ne može se primeniti za traženje tog istog elementa u spregnutoj listi. □

Potklasa nasleđuje obeležja sa njihovom implementacionom strukturom od superklase. Pri tome, da bi se postiglo deljenje i ponovno korišćenje programskog koda, objekti potklase pozivaju metode superklase, za obradu poruka, koje se odnose na nasleđena obeležja. Ako se implementaciona struktura nekog obeležja superklase izmeni, moraju se, na odgovarajući način, modifikovati strukture svih objekata potklase, kako bi se na njih mogle primenjivati izmenjene metode superklase. Ovakvim povezivanjem potklase sa superklasom, gubi se i osobina lokalnosti. Promene u superklasi izazivaju promene u svim njenim potklasama.

7.3.5.2. *Nasleđivanje metoda*

U hijerarhiji nasleđivanja, potklase nasleđuju metode, definisane u njihovim superklasama. Pri tome, nasleđene metode postaju deo interfejsa objekata potklase, tako da ih objekti potklase mogu pozivati. Čak i ako metoda pripada superklasi, izvršava se na objektu, koji je poruku primio, a ne na nekom objektu superklase.

ALGORITAM IZVRŠENJA PORUKE

```
POSTAVI K ← P                    (* inicijalizacija tekuće klase *)
POSTAVI i ← 0
RADI traženje_M DOK JE i = 0
   AKO JE M ∈ M_k TADA
      IZVRŠI M
   INAČE
      AKO JE K je koren hijerarhije klasa TADA
         POSTAVI i ← 1          (* greška, metoda M ne postoji *)
      INAČE
         POSTAVI K ← S
      KRAJ AKO
   KRAJ AKO
KRAJ RADI traženje_M
```

Slika 7.11.

Opšti algoritam izvršenja poruke, poslate objektu potklase, zasnovan je na traženju odgovarajuće metode u hijerarhiji klasa. Traženje počinje od potklase, čijem objektu je poruka upućena. Ako ta klasa ne sadrži traženu metodu, traženje se nastavlja u direktno nadređenoj superklasi. Neka je P potklasa objekta, koji je primio poruku sa nazivom metode M, K tekuća klasa hijerarhije, M_k skup metoda klase K, a S direktno nadređena superklasa tekuće klase K. Na slici 7.11. je prikazan pseudokod algoritma traženja metode M.

Nasleđena metoda se, za potrebe potklase, može i modifikovati (redefinisati). Modifikovana metoda superklase postaje deo implementacije potklase, tako da, saglasno algoritmu sa slike 7.11, objekti potklase izvršavaju modifikovanu metodu, čak i ako originalna i modifikovana metoda imaju iste nazive.

Primer 7.15. Klasa *Skup* je potklasa klase *Torba*. Klasa *Torba* može, a klasa *Skup* ne sme sadržati dva jednaka objekta. Metoda *upiši_novi_elemenat* klase *Torba* se, za potrebe klase *Skup*, mora modifikovati tako da se, pre upisa novog elementa, proveri da li takav element već postoji. □

Neki put je potrebno izvršiti originalnu, a ne modifikovanu metodu. U cilju izbegavanja nejednoznačnosti, zbog istih naziva metoda, u objektno - orijentisanim jezicima se koristi jedan od sledeća dva postupka:

- korišćenje pseudopromenljive ***super*** i
- kvalifikacija metode putem imena klase.

U slučaju korišćenja pseudopromenljive *super*, poruka sa frazom *super M* ukazuje da traženje metode *M* treba započeti od klase, direktno nadređene klasi, čijem objektu je poruka upućena. Ako se koristi kvalifikacija metode, tada fraza *S.M* ukazuje da traženje treba započeti od klase *S*.

7.3.5.3. Preklapanje naziva metoda

Jedan od važnih koncepata objektne orijentacije je ***preklapanje naziva metoda***[*]. Preklapanje naziva omogućava da se metode sa istim imenom, ali različitom semantikom i implementacijom primene na objekte različitog tipa. Međutim, preklapanje naziva nije karakteristično samo za objektno - orijentisane programske jezike.

Primer 7.16. U skoro svim programskim jezicima, aritmetičke operacije se koriste za sabiranje, oduzimanje, množenje i deljenje bilo celih bilo brojeva u pokretnom zarezu, mada su implementacije tih operacija za cele brojeve i brojeve u pokretnom zarezu potpuno različite. U Pascalu, na primer, "+" znači: sabiranje, povezivanje nizova i uniju skupova.

Kao dalja ilustracija višestruke upotrebe, može poslužiti činjenica da, iako su aritmetičke operacije binarne, one mogu lako da se primene i kao operacije nad nizovima brojeva. Neka su N_1 i N_2 dva niza od n celih brojeva. Tada se može definisati procedura

RADI sabiranje $\forall i \in \{1,...,n\}$
  POSTAVI $N[i] = N_1[i] + N_2[i]$
KRAJ RADI sabiranje.

Kao rezultat se dobija niz N, čiji elementi predstavljaju zbir odgovarajućih elemenata nizova N_1 i N_2. □

---

[*] Preklapanje naziva metoda se na engleskom naziva overloading

U konvencionalnim programskim jezicima je preklapanje naziva dozvoljeno samo za neke operacije i to na osnovnim tipovima podataka. U objektno - orijentisanim programskim jezicima se preklapanje naziva dozvoljava za svaku metodu i za proizvoljne tipove podataka. Preklopljene metode treba da se razlikuju bilo po broju bilo po tipovima svojih argumenata. Kada se metoda sa preklopljenim nazivom pozove, objektno - orijentisani sistem bira odgovarajuću implementaciju upoređujući tipove argumenata u poruci sa tipovima argumenata, deklarisanim za metodu. Povezivanje naziva metode sa specifičnom implementacijom se vrši ili statički, prilikom prevođenja, ili dinamički, prilikom izvršavanja program, što zavisi od objektno - orijentisanog programskog jezika.

7.3.5.4. *Dinamičko povezivanje*

Dinamičko ili ***kasno povezivanje*** znači da objektno - orijentisani sistem povezuje naziv sa kodom metode u trenutku izvršavanja programa, a ne prilikom njegovog prevođenja. Naime, od klase, kojoj pripada objekat - primalac poruke, zavisi i koji od programskih kodova sa istim nazivom će biti pozvan za izvršavanje.

Primer 7.17. Posmatra se poruka $c + c_1$ upućena objektno - orijentisanom sistemu. Da bi utvrdio koja metoda stvarno treba da se primeni, objektno - orijentisani sistem šalje poruku "+ c_1" objektu c. Objekat c proverava da li se u njegovom interfejsu nalazi metoda sa nazivom "+" i izvršava je. Pri tome, u proceduru te metode je ugrađena specifičnost postupka sabiranja, karakteristična za tip podataka objekta c. □

Dobra strana dinamičkog povezivanja je da se njegovom primenom izbegava upotreba naredbi uslovnog skoka (tipa *AKO ...TADA ... INAČE*) u cilju: ispitivanja tipa podataka objekta - primaoca poruke i pozivanja odgovarajućeg programskog koda. Međutim, primena mehanizama preklapanja naziva i kasnog povezivanja dovodi do degradacije performansi izvršavanja programa.

7.3.5.5. *Parametarski polimorfizam*

Polimorfizam je sposobnost poprimanja različitih oblika. Kada je reč o programskim jezicima, ***polimorfizam*** ukazuje da iste (ne samo po imenu) operacije mogu da se primene na objekte različitog tipa. Saglasno tome, preklapanje naziva metoda predstavlja jedan oblik polimorfizma. ***Parametarski polimorfizam*** je mnogo stroži pojam od preklapanja naziva. Koristi tipove podataka kao parametre, koji se prosleđuju metodi. Ti parametri informišu metodu nad kojim od generičkih tipova podataka ona treba da se izvrši. To dalje znači:

- da se procedura metode definiše s obzirom na proizvoljan tip podataka T, te da, saglasno tome, može da se primeni na različite tipove podataka i
- da se poruka objektu upućuje navođenjem trojke $m[T](i)$, gde je m ime metode, T tip podataka objekta - primaoca poruke, a i njegov identifikator.

U poruci se metodi prosleđuje stvarni tip podataka objekta - primaoca poruke kao parametar. Taj stvarni tip podataka se posmatra kao generički tip podataka, nastao od proizvoljnog tipa podataka T.

Primer 7.18. Neka je m ime metode za sabiranje elemenata liste, a cl identifikator objekta - liste sa celim brojevima, kao elementima. Sabiranje elemenata liste cl se vrši upućivanjem poruke sa parametrom Int, kao specijalizacijom za tip podataka T

$$m[Int](cl).$$

Ako su elementi liste rl realni brojevi, poruka glasi

$$m[Real](rl).$$

Procedura metode m je pisana za sabiranje elemenata liste, bez obzira na tip podataka tih elemenata. Liste sa celim ili realnim brojevima su generički podtipovi liste sa proizvoljnim tipom podataka. □

Osnovni efekat korišćenja parametarskog polimorfizma je stvarno delenje programskog koda. U slučaju preklapanja naziva, samo je ime isto za, u suštini, različite procedure.

7.3.5.6. Jedan pristup selektivnom nasleđivanju

Najopštiji pristup selektivnom nasleđivanju se realizuje uvođenjem pojmova: javne, privatne i zaštićene osobine (obeležja, metode). Ukratko i bez pretenzija na strogu preciznost, ako je osobina deklarisana kao:

- ***javna***, tada objekat bilo koje klase može da joj direktno: pristupi, da njome manipuliše ili da je pozove,
- ***privatna***, tada samo objekti posmatrane klase mogu da joj direktno: pristupe, da njome manipulišu ili da je pozovu,
- ***zaštićena***, tada samo objekti posmatrane klase i njenih potklasa mogu direktno: da joj pristupe, da njome manipulišu ili da je pozovu.

Pri tome, direktno korišćenje osobine se odnosi na pristupanje, manipulisanje i pozivanje samo putem metoda klase, u kojoj je osobina inicijalno deklarisana.

I nasleđivanje može biti javno, privatno ili zaštićeno. Pri tome, ako je nasleđivanje:

- javno, javne osobine superklase postaju javne osobine potklase, a zaštićene osobine superklase postaju zaštićene osobine potklase,
- zaštićeno, javne i zaštićene osobine superklase postaju zaštićene osobine potklase,
- privatno, javne i zaštićene osobine superklase postaju privatne osobine potklase.

Privatne osobine superklase se ne nasleđuju.

Opisani mehanizmi selektivnog nasleđivanja su preciznije razrađeni i ilustrovani u tački 7.6, posvećenoj operacijskoj komponenti objektno - orijentisanog modela podataka.

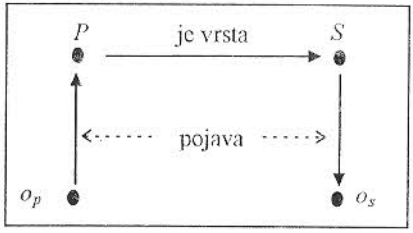
7.3.6. Nasleđivanje interfejsa

Nasleđivanjem, metode superklase postaju pristupačne objektima potklase. U suštini, nasleđivanje metoda znači širenje interfejsa objekata potklase nazivima metoda superklasa.

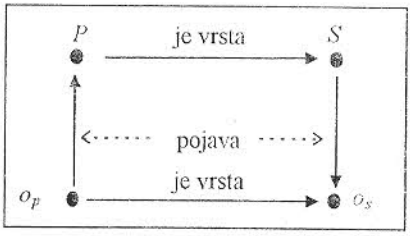
Nasleđivanje se koristi za specijalizaciju, za opisivanje specifičnih osobina nekog skupa objekata. Objekti tog skupa predstavljaju modele takvih entiteta, koji u realnom sistemu predstavljaju podskup nekog drugog skupa entiteta. Modeli entiteta tog drugog skupa predstavljaju objekte superklase. Između skupova entiteta superklase i potklase važi relacija sadržavanja. Kardinalni broj skupa entiteta potklase je manji od kardinalnog broja skupa entiteta superklase. Međutim, nasleđivanjem, interfejs objekata potklase postaje nadskup interfejsa objekata superklase. Isto važi i za skupove obeležja potklase i superklase. Saglasno tome, nasleđivanje se može posmatrati ne samo kao specijalizacija, već i kao proširenje.

7.3.7. Nasleđivanje objekata

Nasleđivanje se odnosi samo na klase, ne i na objekte. Između skupova objekata potklase i superklase ne mora važiti relacija sadržavanja, mada je u realnom sistemu svaki entitet potklase i entitet superklase. Objektno - orijentisani jezici ne podržavaju automatski relaciju sadržavanja između skupova objekata superklase i potklase, već zahtevaju da se o tome eksplicitno vodi računa pri pisanju programa. Slika 7.12 ukazuje da između objekta o_p potklase P i objekta o_s superklase S ne postoji nasleđivanje.



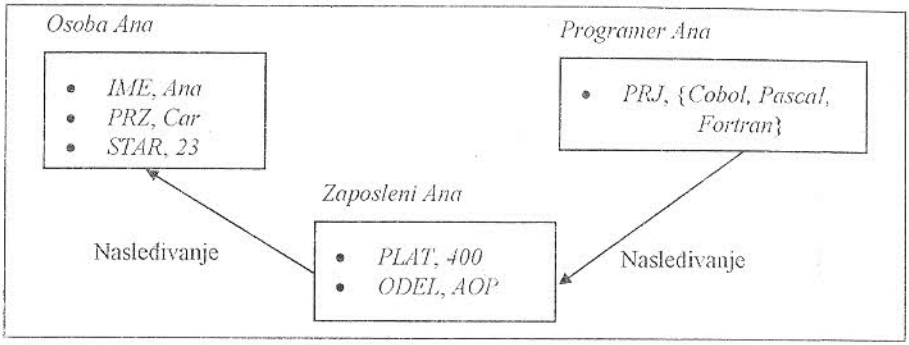
Slika 7.12.



Slika 7.13.

Kada bi nasleđivanje objekata bilo dozvoljeno, tada bi objekat o_p nasleđivao stanje objekta o_s za sva ona obeležja, koja klasa P nasleđuje od klase S. Tada bi za svaki objekat o_p klase P postojao jedan objekat o_s klase S, od kojeg bi o_p nasleđivao stanje, slika 7.13. Pri tome, o_p ne bi sadržao podatke, koje sadrži objekat o_s.

Primer 7.19. Na slici 7.14 je predstavljena ideja nasleđivanja objekata za slučaj hijerarhije *Osoba - Zaposleni - Programer*. □

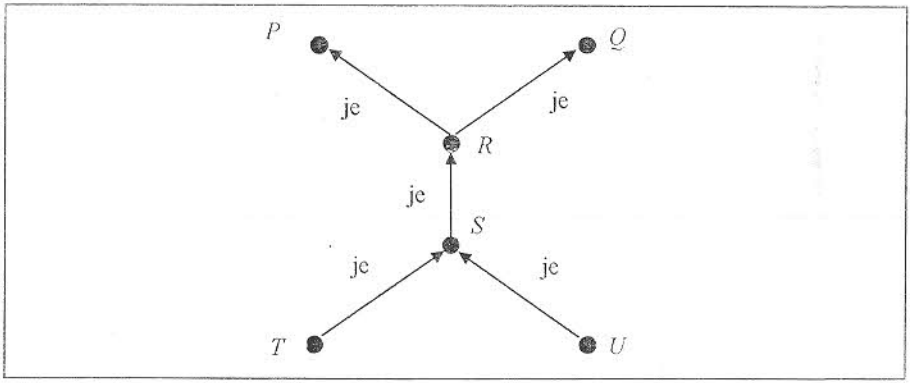


Slika 7.14.

Sistemi, koji podržavaju nasleđivanje objekata, nazivaju se ***prototipskim sistemima***. Osnovna ideja prototipskih sistema je da se, pri projektovanju modela realnog sistema, prvo definišu prototipski objekti, kao individualni slučajevi, a da se prototipski objekti potom specijalizuju i generalizuju.

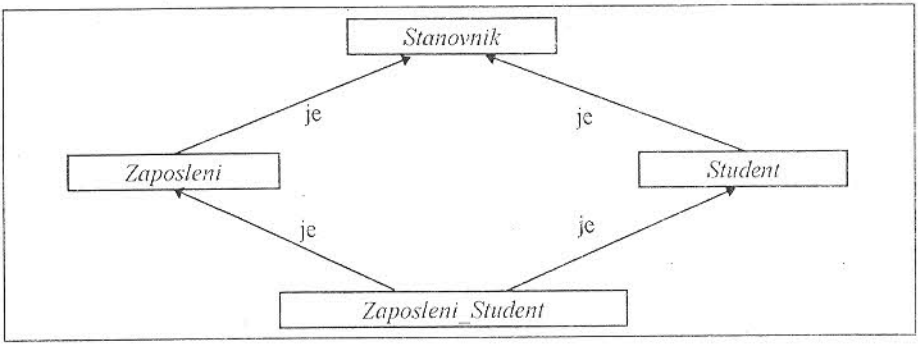
7.3.8. Višestruko nasleđivanje

Ako svaka klasa ima najviše jednu superklasu, tada je hijerarhija klasa struktura stabla. Neki put je korisno za neku klasu da ima više superklasa. Tako se dolazi do generalizacije hijerarhije klasa, koja se naziva ***usmerenim acikličnim grafom***. Taj graf se naziva i ***latisom***. U latisi klasa, jedna klasa može nasleđivati osobine od više superklasa. To se naziva ***višestrukim*** nasleđivanjem. Na slici 7.15 je nacrtana jedna latisa klasa. Da je reč o latisi, a ne o stablu klasa, govori činjenica da klasa R ima dve superklase P i Q.



Slika 7.15.

Primer 7.20. Posmatraju se klase: *Stanovnik*, *Zaposleni*, *Student* i *Zaposleni_Student*. Klasa *Zaposleni_student* ima dve direktno nadređene i jednu tranzitivnu superklasu. Direktno nadređene klase su: *Zaposleni* i *Student*. *Zaposleni_Student* nasleđuje osobine od sve tri klase. Na slici 7.16 je prikazana ova hijerarhija klasa i jedna pojava klase *Zaposleni_Student*. □



Slika 7.16.

7.4. *Identitet objekta*

U objektno - orijentisanom modelu podataka, svaki objekat dobija identitet, koji mu je permanentno pridružen i ostaje nepromenjen, bez obzira na promene stanja objekta. Identitet je interna osobina objekta. Namena mu je da reprezentuje objekat, bez obzira na to kako se objektu pristupa, šta sadrži ili gde se nalazi. Identitet objekta je pojam, koji ne zavisi od hardverskog - softverskog okruženja i načina realizacije objekta.

Kada se kaže da je namena identiteta da reprezentuje objekat, želi se ukazati da je cilj uvođenja identiteta objekta uvođenje takve karakteristike, koja odražava onu bitnost objekta, kakvu poseduju i realni entiteti.

Primer 7.21. Svaka osoba, tokom vremena prolazi kroz niz strukturalnih promena. Rađa se, završava školu, zapošljava se. Stiče nove osobine, kao što su: supružnik, deca, položaj na poslu. Može doći do promene imena, prezimena, broja zdravstvene knjižice. Ipak, pri svim tim promenama, ostaje nešto što je jedinstveno, karakteristično za svaku osobu, njen identitet, njena bitnost. □

Identitet objekta pridružuje objektu, kao modelu realnog entiteta, osobinu bitnosti. Pored ovog, semantičkog značenja, identitet predstavlja i mehanizam za povezivanje objekta sa drugim objektima. Jedan objekat postaje komponenta drugih objekata tako, što se njegov identitet javlja kao vrednost određenog obeležja tih drugih objekata.

Ako bi se želeli koristiti samo osnovni tipovi podataka za vrednosti obeležja objekta, a ne i identitet objekta, tada postoje dva rešenja. To su:

- replikacija podataka i
- strani ključ, kao u relacionom modelu.

Oba rešenja imaju određene nedostatke. Korišćenje replikacije podataka značajno komplikuje ažuriranje, jer predstavlja potencijalni izvor nekonzistentnosti podataka. Neka su o_1 i o_2 objekti klase, čiji tip objekta sadrži komponentu (X, T). T je tip objekta neke druge klase. Neka je, dalje, t objekat tipa T i neka važi da objekti o_1 i o_2 imaju i treba da imaju iste X vrednosti ($o_1.X = t \wedge o_2.X = t$). Ako se ažuriranjem izmeni X vrednost, recimo, objekta o_1, mora se, na isti način, izmeniti i X vrednost objekta o_2. Ako se ta izmena ne izvrši, baza podataka će ostati nekonzistentna. Ovakva, višestruka ažuriranja predstavljaju ozbiljan problem u praksi.

Primer 7.22. Neka je *Zaposleni* = ((*IME*, *Chr*), (*PRZ*, *Chr*), (*RAD_MES*, *Radno_Mesto*)) tip objekta, čije obeležje *RAD_MES* ima, kao domen, klasu *RadnoMesto*. Komponente tipa objekta *Radno_Mesto* su (*NAZRM*, *Chr*) i (*BRBOD*, *Int*). Ako bi se koristila replikacija, tada bi dva objekta klase *Zaposleni* imala sledeću strukturu:

((*IME*, *Ana*), (*PRZ*, *Tot*), (*NAZRM*, *programer*), (*BRBOD*, *400*)) i
((*IME*, *Aca*), (*PRZ*, *Zec*), (*NAZRM*, *programer*), (*BRBOD*, *400*)).

Ako, u realnom sistemu dođe do izmene broja bodova za radno mesto programer, tu izmenu treba sprovesti u svim objektima klase *Zaposleni*, za koje važi *NAZRM* = *programer*. □

Rešenje, zasnovano na korišćenju stranog ključa, donosi sledeća dva problema. Jedan je posledica činjenice da se sada podaci o jednom realnom entitetu nalaze u više objekata. Povezivanje tih podataka zahteva primenu operatora prirodnog spajanja, a prirodni spoj je operacija, čije izvršavanje zahteva dosta vremena. Drugi problem leži u činjenici da se vrednosti ključa ne smeju modifikovati. To dalje eliminiše mogućnost korišćenja prirodnih obeležja klase entiteta u ulozi ključa.

Primer 7.23. Obeležje *NAZRM* se ne bi smelo proglasiti ključem tipa objekta *Radno_Mesto*, jer se nazivi konkretnih radnih mesta u realnom sistemu mogu, tokom vremena menjati. Te izmene bi dovele do značajnih intervencija u bazi podataka. □

Uvođenjem identiteta objekta, ovi problemi se eliminišu. Prednosti identiteta objekta i direktne reprezentacije kompleksnih objekata, opisani su u narednom tekstu. Modeli, koji podržavaju identitet objekta, nazivaju se "objektno - orijentisanim", mada se taj termin, strogo govoreći, odnosi samo na one modele koji podržavaju kompleksne objekte i inkapsulaciju. Modeli, koji ne podržavaju identitet objekta, nazivaju se orijentisanim na vrednosti.

7.4.1. *Pojam identiteta objekta*

Objekti se grade od objekata. Primitivni objekti, kao elementi osnovnih tipova podataka, predstavljaju polazne gradivne elemente za formiranje kompleksnijih objekata. Klase osnovnih tipova podataka su, po pravilu, ugrađene u objektno - orijentisane pro-

gramske jezike i druge objektno - orijentisane sisteme, što znači da ih ne treba posebno deklarisati. Te klase, po pravilu, nemaju svojih obeležja. Primitivnim objektima se ne pridružuju identiteti. Smatra se da primitivni objekti identifikuju sami sebe, putem svoje vrednosti. Identitet neprimitivnog objekta se realizuje pridruživanjem jedinstvenog *identifikatora*.

Za operacije sa identitetima objekata, uvode se dva osnovna predikata. To su:

- predikat $x == y$, za proveru identičnosti objekata x i y i
- predikat $x = y$, za proveru jednakosti stanja objekata x i y.

Pri tome, identičnost dva objekta znači da imaju isti identitet i isto stanje, a jednakost stanja ne implicira jednakost identiteta.

Primer 7.24. Broju *3*, kao elementu klase *Int*, se identitet ne dodeljuje, jer ne mogu postojati dva primitivna objekta *3*. Oba suda, $3 == 3$ i $3 = 3$, su istinita.

Međutim, ako se objektu o_1 = ((*IME*, *Aca*), (*PRZ*, *Zec*)) dodeli identifikator i_1, a objektu o_2 = ((*IME*, *Aca*), (*PRZ*, *Zec*)) dodeli identifikator i_2, tada je $o_1 == o_2$ neistinit sud, a $o_1 = o_2$ istinit sud. □

Jedna od osnovnih pretpostavki objektno - orijentisanog modela podataka je da postoji neograničen skup identifikatora $I = \{i_1, i_2, ...\}$, takav da važi:

1° Samo jedan identifikator $i \in I$ se pridružuje svakom neprimitivnom objektu.

2° Identifikator $i \in I$ se generiše i pridružuje objektu u trenutku njegovog nastanka i ostaje mu pridružen bez obzira na izmene njegovog stanja.

3° Svaki identifikator se može pridružiti jednom i samo jednom objektu. Ako je identifikator generisan, on mora biti pridružen nekom objektu. Objekat i njegov identifikator se ne mogu posmatrati odvojeno. Identifikator jedinstveno identifikuje objekat i ostaje mu pridružen i traje dok objekat postoji.

Svaki objekat poseduje sledeće tri osobine:

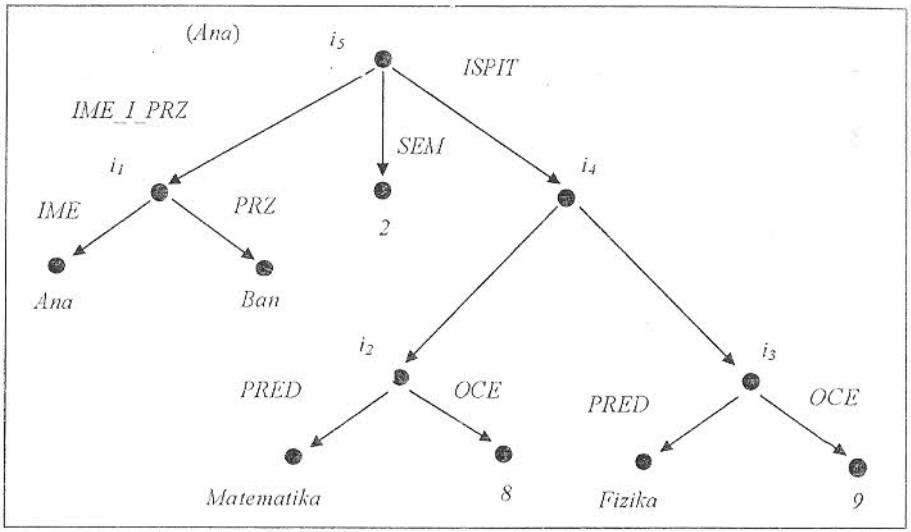
- objekat je pojava neke klase, čime je određen njegov tip,
- objektu je pridružen identifikator, putem kojeg se realizuje pojam identiteta objekta i
- objekat poseduje stanje, u notaciji $((X_1, i_1), ..., (X_n, i_n))$, gde je X obeležje klase, a i ili identifikator ili primitivni objekat. Ako je o objekat tipa kolekcija, njegovo stanje je $\{i_1, i_2, ..., i_m\}$, gde je i identifikator objekta u kolekciji.

Postoji više tehnika za grafičku reprezentaciju fizičke strukture objekta. U ovom tekstu će se koristiti tehnika usmerenih acikličnih grafova. Svaki objekat predstavlja se kao čvor sa pridruženim identifikatorom, ili pridruženom vrednošću primitivnog objekta. Potezima se pridružuju nazivi obeležja.

Primer 7.25. Neka je objektu ((*IME*, *Ana*), (*PRZ*, *Ban*)) pridružen identifikator i_1, objektu ((*PRED*, *Matematika*), (*OCE*, *8*)) identifikator i_2, ((*PRED*, *Fizika*), (*OCE*, *9*)) i_3, a objektu - skupu $\{i_2, i_3\}$, identifikator i_4. Objekat sa podacima o studentu Ani, upisanoj u drugi semestar i o njenim ispitima, ima oblik

$$i_5((IME\_I\_PRZ, i_1), (SEM, 2), (ISPIT, i_4)).$$

Posmatranom složenom objektu je pridružen identifikator i_5. Identifikatori predstavljaju vrednosti svih onih obeležja, čiji domeni sadrže neprimitivne objekte. Na slici 7.17 prikazana je moguća geometrijska predstava fizičke strukture objekta *Ana*. □



Slika 7.17.

Zahvaljujući postojanju identifikatora objekta, dozvoljeno je definisati dva objekta, koji uzajamno sadrže jedan drugog kao komponente.

Primer 7.26. U primeru 7.5 definisani su tipovi objekata:

$$TipKupca = ((NAZIV, Chr), (ADRESA, Chr), (STANJE\_RAČ, Int), (PORUDŽBINE, \{TipPorudžbine\}))$$

i

$$TipPorudžbine = ((BRP, Int), (KUPAC, TipKupca), (SADRŽI, \{StavkaPorudžbine\}).$$

Na prvi pogled, ova cirkularna veza između pojava dva tipa objekta je nedozvoljena, jer jedan objekat treba da predstavlja sam svoju komponentu. Međutim, zahvaljujući postojanju identifikatora objekta, i takve strukture su regularne. Pojava tipa objekta *TipKupca* sadrži, uz obeležje *PORUDŽBINE*, identifikator - pokazivač na jedan objekat tipa kolekcija. Taj objekat tipa kolekcija sadrži identifikatore - pokazivače na one pojave klase *TipPorudžbine*, koje je izvršio posmatrani kupac. Svaka od ovih pojava klase *TipPorudžbine*, sadrži, uz obeležje *KUPAC*, identifikator posmatrane pojave tipa objekta *TipKupca*. □

U objektno - orijentisanim programskim jezicima, identifikator je pokazivač - memorijska adresa lokacije, u kojoj se objekat nalazi. Pokazivač je jedna od mogućih implementacija identiteta. Treba naglasiti da između pojma identiteta objekta i njegove implementacije, postoji fundamentalna razlika. Identitet je semantički koncept.

7.4.2. Postupci za implementaciju identiteta objekta

Postoji više postupaka za realizaciju identiteta objekta. U njih spadaju:

- virtuelne i fizičke adrese,
- korisnički nazivi i
- surogati.

Identifikator u obliku virtuelne ili fizičke adrese predstavlja brzo, ali nefleksibilno rešenje. Brzo, jer se na osnovu adrese brzo dolazi do objekta. Nefleksibilno, jer svaka promena lokacije za smeštaj objekta, zahteva i promenu identifikatora. Sve ostale tehnike zahtevaju primenu indirektnog pristupa objektu. Indirektni pristup objektu je posledica potrebe postojanja:

- tabele sa parovima (identifikator, adresa), za objekte u operativnoj memoriji i
- indeksa (stabla traženja) sa parovima (identifikator, adresa), za objekte na eksternoj memoriji.

Postupci, koji traže indirektni pristup su fleksibilniji, jer promena lokacije ne dovodi do promene identifikatora, ali zahtevaju najmanje dva pristupa pri traženju objekta. Najmanje jedan pristup tabeli ili stablu traženja i jedan pristup samom objektu.

Pobrojana rešenja za realizaciju identiteta objekta se razlikuju još i po tome što adrese i korisnički nazivi predstavljaju identifikatore, spolja pridružene objektu. Nisu sastavni deo objekta. Surogat predstavlja sastavni deo objekta, ali je to sistemski generisan identifikator, koji je nezavisan od stanja objekta. Surogat ne treba poistovetiti sa ključem u relacionom modelu podataka.

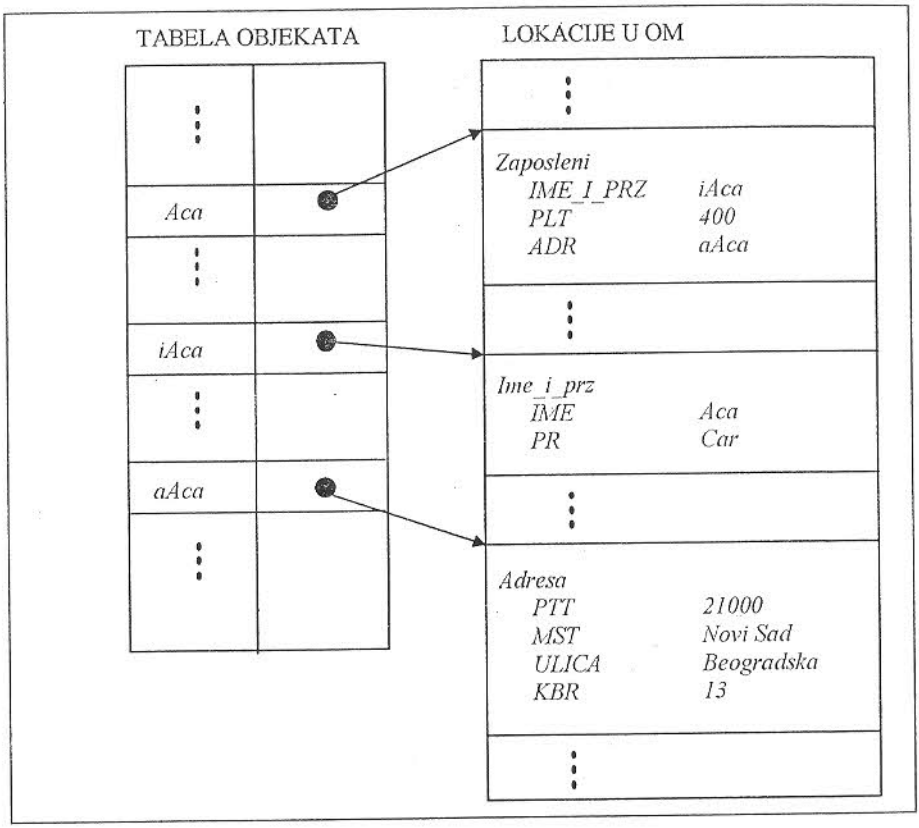
7.4.2.1. Adresa kao identifikator

Korišćenje adrese u svojstvu identifikatora predstavlja, verovatno, najjednostavniji postupak realizacije identiteta objekta. Generisanje novih objekata se vrši dinamički u operativnoj memoriji, pri izvršavanju objektno - orijentisanog programa. Kada neka klasa dobije naredbu da generiše novi objekat, ona prvo rezerviše memorijski prostor - lokaciju za smeštaj tog objekta, a adresu početka te lokacije proglasi za identifikator objekta. Nakon toga, objekat se može preneti na eksternu memoriju, gde ostaje trajno memorisan. Na eksternoj memoriji, objekat dobija drugi identifikator. Taj identifikator je relativna ili apsolutna adresa lokacije na eksternoj memoriji. Prilikom učitavanja objekta u operativnu memoriju, vrši se njegova transformacija iz jedne u drugu reprezentaciju. Ta transformacija znači i učitavanje i transformaciju svake komponente složenog objekta. Ne treba ispustiti iz vida da objekat, kao direktne komponente, sadrži samo primitivne objekte, a umesto neprimitivnih objekata, sadrži njihove identifikatore.

7.4.2.2. Korisnički nazivi kao identifikatori

Dodela korisničkih naziva je najčešći postupak pridruživanja identifikatora objektima. Jedno rešenje ovog postupka je da se svakom objektu dodeli jedinstveni naziv i da

taj naziv bude jedinstven, bez obzira na klasu objekta. Taj naziv postaje identifikator objekta. Ovakav postupak dodele identifikatora je fleksibilniji od postupka sa korišćenjem adrese, ali uvodi potrebu indirektnog pristupa. Indirektni pristup se vrši putem takozvane ***tabele objekata***, koja sadrži parove (identifikator, adresa).



Slika 7.18.

Primer 7.27. Neka je *Zaposleni* = ((*IME_I_PRZ*, *Ime_i_prz*), (*PLT*, *400*), (*ADR*, *Adresa*)) tip objekta klase *Zaposleni*, gde su *Ime_i_prz* i *Adresa*, takođe klase. Neka je, pri generisanju objekta sa podacima o zaposlenom Aci Caru, dodeljen identifikator:

- *Aca* odgovarajućem objektu klase *Zaposleni*,
- *iAca* odgovarajućem objektu klase *Ime_i_prz* i
- *aAca* odgovarajućem objektu klase *Adresa*.

Tada bi tabela objekata i sadržaji odgovarajućih lokacija u operativnoj memoriji, mogli da se predstave kao na slici 7.18. Na slici su adrese predstavljene kao potezi ka početku lokacije. □

Neki objektno - orijentisani jezici, kao C++, koriste i adrese i korisničke nazive za realizaciju identiteta objekta. Nazivi se koriste pri generisanju objekata, a njihove adrese pri izgranji kompleksnih objekata. Ako je objekat komponenta nekog drugog objekta, tada taj drugi objekat sadrži njegovu adresu, kao referencu na svoju komponentu.

Korišćenje naziva u svojstvu identifikatora, može dovesti do dodele dva različita naziva istom objektu. Do takve situacije može doći u slučajevima kada isti objekat ima više uloga. Sama dodela različitih naziva se realizuje putem odgovarajuće naredbe programskog jezika. Jedini način da se kasnije ustanovi da je reč o istom objektu je primena predikata za proveru identičnosti i jednakosti.

Primer 7.28. Objektu sa podacima o jednom studentu može biti dodeljen naziv s_1, kada se on posmatra kao najbolji student na drugoj godini, a naziv s_2, kao studentu koji se prijavio na konkurs za dodelu stipendija. Primena testa jednakosti, $s_1 = s_2$, je jedini način da se utvrdi da je reč o istom realnom entitetu. □

7.4.2.3. Surogati

Surogati su globalno jedinstveni identifikatori. Za njih je karakteristično:

- da ih generiše objektno - orijentisani sistem,
- da su potpuno nezavisni bilo od stanja bilo od adrese objekta,
- da predstavljaju integralni deo objekta,
- da se mogu koristiti za pristup objektima na eksternom memorijskom uređaju.

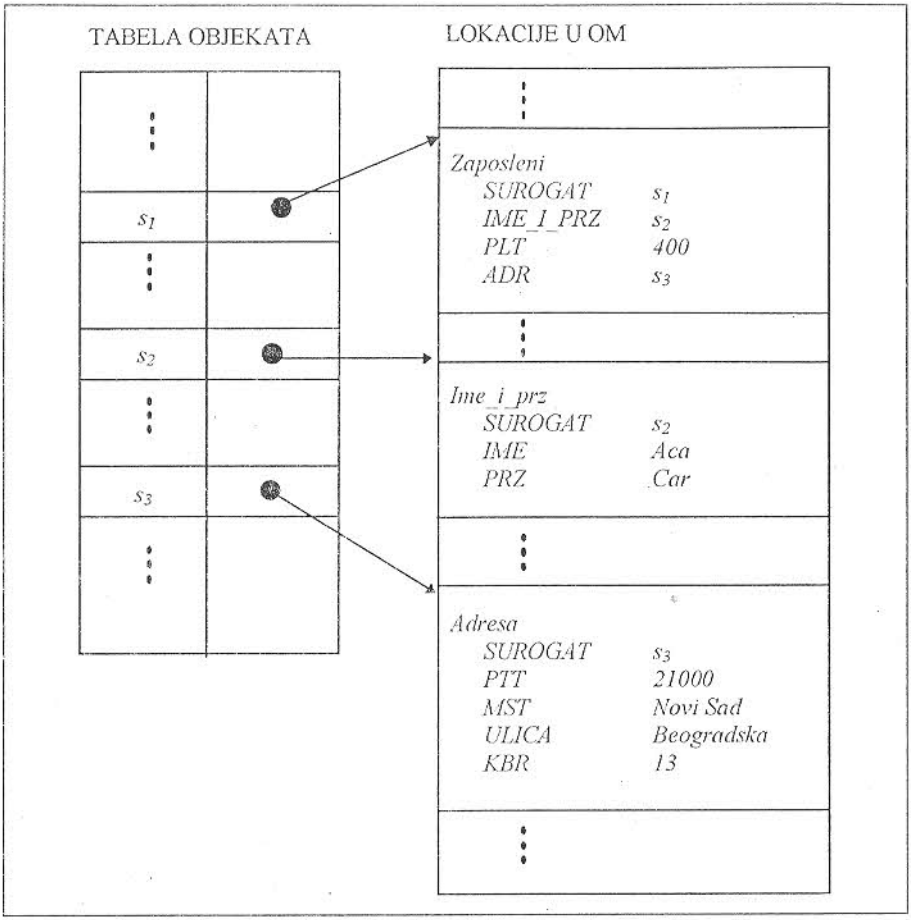
Surogat se uključuje u objekat kao vrednost specijalnog obeležja. Pošto se surogat memoriše zajedno sa objektom, objekat se može premeštati iz jedne lokacije u drugu, kopirati i replicirati, a da se njegov identifikator ne promeni. Međutim, i korišćenje surogata dovodi do potrebe indirekcije u adresiranju.

Primer 7.29. Na slici 7.19 je data geometrijska reprezentacija tabele objekata i sadržaja lokacija u operativnoj memoriji, za slučaj primene surogata kao identifikatora objekta klase *Zaposleni* iz primera 7.23. □

Pojam surogata ne treba poistovećivati sa pojmom ključa relacionog modela podataka, mada postoje određene sličnosti. Razlike su u sledećem:

- vrednost ključa određuje korisnik i može joj pristupiti,
- vrednost ključa je jedinstvena samo u okviru jedne relacije - tabele,
- vrednost surogata generiše objektno - orijentisani sistem, krajnji korisnik tu vrednost ne može videti i
- vrednost surogata je globalno , a ne lokalno jedinstvena.

Smatra se [KA] da primena surogata, kao identifikatora, predstavlja najbolje rešenje za realizaciju pojma identiteta objekta.



Slika 7.19.

7.5. *Integritetna komponenta objektno -orijentisanog modela podataka*

Putem transakcija, sistem za upravljanje bazom podaka preslikava jedno konzistentno stanje baze podataka u drugo. Uslovi konzistentnosti baze podataka se obično izražavaju putem predikata, koje treba da zadovolji tekuće stanje baze podataka. Predikati se definišu i za objekte i za obeležja. Svi uslovi integriteta baze podataka relacionog modela podataka, kao što su:

- nula vrednost,
- integritet entiteta,
- referencijalni integritet i
- integritet domena

treba da budu zadovoljeni i u objektno - orijentisanoj bazi podataka.

7.5.1. *Nula vrednost*

Ograničenje posedovanja nula vrednosti za određena obeležja se, u objektno - orijentisanim sistemima:

- ili navodi eksplicitno u deklaraciji strukture tipa objekta klase,
- ili rešava u okviru metoda za konstruisanje objekata klase.

Deklarisanje u okviru strukture tipa objekta klase se vrši, ako objektno - orijentisani jezik takvo deklarisanje podržava. Sintaksa te deklaracije zavisi od objektno - orijentisanog programskog jezika. Koriste se fraze tipa "required" ili "not null".

Ako se pitanje nula vrednosti rešava u okviru metoda za konstruisanje objekata klase, tada se za konstruisanje objekata uvodi više metoda. Jedna od njih generiše objekte sa vrednostima svih obeležja klase, a druge generišu objekte sa vrednostima samo nekog pravog podskupa skupa obeležja klase. Samo obeležja, koja se inicijalizuju navođenjem nekih vrednosti u svim metodama za konstruisanje objekata klase, ne mogu imati nula vrednosti.

Primer 7.30. Metode za konstruisanje objekata su precizno i detaljno obrađene u tačkama 7.6.1 i 7.6.3. Na ovom mestu se samo neformalno ilustruje postupak za konstruisanje objekata sa nula vrednostima nekih obeležja.

Posmatra se tip objekta *Osoba* = ((*MBR*, *Int*), (*IME*, *Chr*), (*PRZ*, *Chr*), (*KTL*, *Int*)), gde jedino obeležje *KTL* (broj kućnog telefona) može imati nula vrednost. Tada bi postojale dve metode za konstruisanje objekata tipa *Osoba*. Jedna, oblika *Osoba*(*int*, *chr*, *chr*, *int*), bi zahtevala inicijalizaciju svih obeležja klase. Druga, oblika *Osoba*(*int*, *chr*, *chr*), bi zahtevala inicijalizaciju samo obeležja *MBR*, *IME* i *PRZ*, pri konstruisanju objekta. U telu druge metode bi se moralo precizno definisati kakav tip podataka i koju vrednost treba objektno - orijentisani sistem da upiše kao nula vrednost za obeležje *KTL*. Pošto se objekti klase *Osoba* mogu konstruisati samo primenom jedne od ovih metoda, svaki od objekata bi imao ne nula vrednosti za obeležja *MBR*, *IME* i *PRZ*. □

7.5.2. *Integritet entiteta*

Identitet objekta je jedinstven, ali, po pravilu, nije pod kontrolom korisnika. Identifikator, kao implementacija identiteta objekta, ne nosi nikakvu dodatnu semantiku o objektu. Ne predstavlja ni uobičajeno sredstvo za traženje određenog objekta u velikoj bazi podataka. Za korisnika je mnogo pogodnije da vrši traženje na osnovu vrednosti ključa.

Identitet objekta ne predstavlja zamenu za ključ, koji definiše korisnik. Zbog toga, bez obzira na postojanje identiteta objekta, često je potrebno obezbediti da vrednosti nekog obeležja, recimo X, imaju jedinstvene i ne nula vrednosti u ekstenziji klase.

Postoje barem dva pristupa rešavanju problema integriteta entiteta u objektno - orijentisanim sistemima. Prema jednom, definisanje integriteta entiteta, kao ograničenja, vrši se navođenjem fraza tipa "unique" i "required" pri opisivanju domena obeležja. Međutim, objektno - orijentisani jezici, često, ne podržavaju takve fraze, tako da se definisanje integriteta entiteta rešava u okviru metoda za konstruisanje objekata klasa.

Pošto klasa nema mehanizam za upravljanje svojom ekstenzijom, kontrola integriteta entiteta, prema drugom postupku, zahteva generisanje odgovarajućeg objekta tipa kolekcija, koji sadrži informaciju o ekstenziji posmatrane klase i omogućava pristupanje svakom objektu klase. Pristupanje svakom objektu klase je potrebno da bi se, pri generisanju novog objekta, moglo proveriti da li već postoji objekat, koji ima istu X vrednost.

Pošto se u objektno - orijentisanom modelu podataka ne postavljaju tako stroga ograničenja u odnosu na obeležje, koje bi igralo ulogu ključa u nekom drugom modelu podataka, vrednosti tog obeležja se mogu modifikovati. U relacionom modelu podataka, jedan razlog zabrane modifikovanja vrednosti ključa leži u činjenici da vrednost ključa igra ulogu identiteta objekta (torke). Drugi razlog predstavlja potreba očuvanja važnosti referencijalnog integriteta. Kao što je pokazano u narednoj tački, u objektno - orijentisanom modelu podataka se referencijalni integritet rešava putem identiteta objekta, tako da ni uslov referencijalnog integriteta ne predstavlja razlog za zabranu izmene vrednosti obeležja, koje igra ulogu ključa.

7.5.3. *Referencijalni integritet*

Kada je reč o kompleksnim objektima, identitet objekta obezbeđuje uslov referencijalnog integriteta. Komponente neprimitivnog objekta sadrže ili primitivne objekte ili identifikatore drugih objekata. Identifikatori drugih objekata predstavljaju direktnu realizaciju referencijalnog integriteta. U ovom slučaju, referencijalni integritet je posledica same strukture objekta, tako da nema potrebe da se definiše poseban mehanizam za njegovu kontrolu.

Da bi se izbegli problemi pri brisanju objekta - komponente ili takva promena stanja neprimitivnog objekta, pri kojoj vrednost referentnog obeležja dobija nedefinisanu vrednost, potrebno je predvideti da referentno obeležje može imati i nula vrednost.

Primer 7.31. Neka je dat tip objekta *Zaposleni* = ((*IME_I_PRZ*, *Ime_i_Prz*), (*PLT*, *Int*), (*RAD_MES*, *Radno_Mesto*)). Da bi se izbegli problemi sa nerasporedenim radnicima, pri deklarisanju obeležja *RAD_MES*, ne treba zabraniti nula vrednosti. □

Kontrola referencijalnog integriteta između objekata potklase i objekata superklase, u objektno - orijentisanom modelu podataka predstavlja otvoreno pitanje. Nasleđivanje objekata ne postoji, te nije podržano ni preslikavanje sa skupa objekata potklase u skup objekata superklase. Čak i ako postoji objekat superklase, koji bi prema svom stanju

objekata klase *Sup* upisuje identifikator novog objekta u objekat *Ekst_Sup*, a metoda za generisanje objekata klase *Veza*, upisuje identifikator novog objekta u objekat *Ekst_Veza*.

POSTUPAK OBEZBEĐENJA REFERENCIJALNOG INTEGRITETA

KONSTRUIŠI *novi objekat* (*y, z*) *klase Pot*
POSTAVI *idpot = identifikator novog objekta*
POSTAVI *int←0* (**int je indikator uspešnosti traženja**)
RADI traženje_objekta _tipa_*Sup* $\forall i \in$*Ekst_Sup* ***DOK JE*** *int = 0*
  AKO JE $o_{idpot}.Z = o_i.Z$ ***TADA*** (*o_i je objekat klase *Sup* sa identifikatorom *i**)
    POSTAVI *int←1*
    POSTAVI *idsup = i*
  INAČE
  KRAJ AKO
KRAJ RADI traženje_objekta _tipa_*Sup*
AKO JE *int = 0* ***TADA***
  KOSTRUIŠI *novi objekat* (*x, z*) *klase SUP*
  POSTAVI *idsup = identifikator novog objekta*
INACE
KRAJ AKO
KONSTRUIŠI *novi objekat* (*idpot, idsup*) *klase Veza*

Slika 7.21.

7.5.4. *Integritet domena*

Integritet domena je specifikacija određenih ograničenja, koja vrednosti obeležja treba da zadovolje. Ta ograničenja mogu biti kompleksnija od onih, opisanih u drugoj glavi ove knjige, jer domen u objektno - orijentisanom modelu podataka može biti klasa sa proizvoljno kompleksnom strukturom objekata.

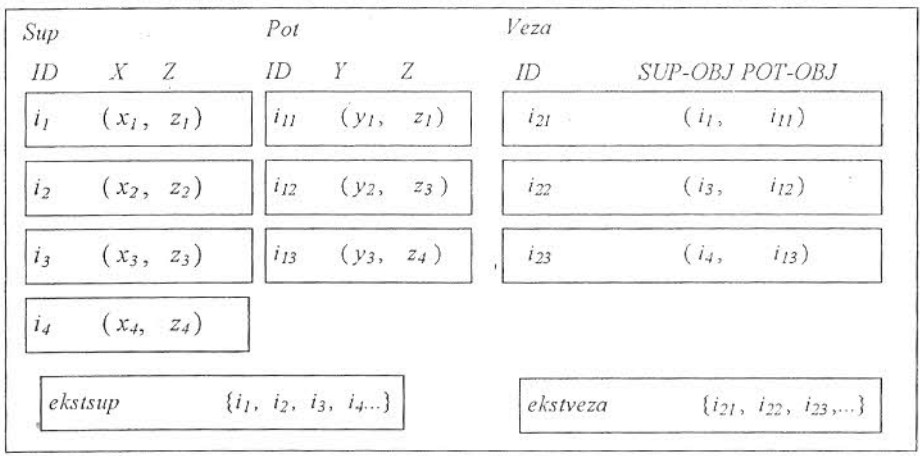
Jedan od načina za kontrolu integriteta domena u objektno - orijentisanom modelu podataka je da se definiše takva potklasa neke postojeće klase, čiji objekti će uzimati samo dozvoljene vrednosti, pa da se ta potklasa proglasi za domen obeležja.

Primer 7.32. Neka je dato obeležje *OCE* i ograničenje $5 \leq OCE \leq 10$. Tada se može definisati klasa *Ocena*, kao potklasa klase *Int*, tako da objekti iz klase *Ocena* uzimaju vrenosti iz segmenta [*5, 10*]. □

mogao biti slika posmatranog objekta potklase, to su dva objekta sa različitim identifikatorima. Jednakost stanja objekta superklase i dela stanja objekta potklase se može proveriti, ali obezbediti modifikaciju jednog pri promeni drugog, nije jednostavan zadatak.

Sledeći postupak predstavlja mogućno rešenje problema kontrole referencijalnog integriteta između objekata potklase i objekata superklase. Neka je *Sup* superklasa, *Pot* potklasa, a *Ekst_Sup* objekat tipa kolekcija, koji sadrži objekte superklase. Neka je *XZ* skup obeležja klase *Sup*, *YZ* skup obeležja klase *Pot*, gde je *Z* skup zajedničkih obeležja. U cilju sprovođenja kontrole integriteta, uvodi se nova klasa *Veza* sa tipom objekta ((*SUP*, *Sup*), (*POT*, *Pot*)), čiji objekti *o* zadovoljavaju uslov $o.SUP[Z] = o.POT[Z]$. Drugim rečima, jedan objekat klase *Veza* sadrži identifikatore po jednog objekta klase *Pot* i jednog, njemu odgovarajućeg, objekta klase *Sup*. Takođe se uvodi novi objekat *Ekst_Veza* tipa kolekcija, koji sadrži identifikatore objekata tipa *Veza*. Pri generisanju svakog novog objekta o_p klase *Pot*, proverava se da li objekat *Ekst_Sup* već sadži identifikator objekta o_s, takvog da važi $o_p.Z = o_s.Z$. U slučaju potvrdnog odgovora, generiše se odgovarajući objekat klase *Veza*, a identifikator tog objekta se uključuje i u objekat *Ekst_Veza*. Inače, prvo se generiše novi objekat klase *Sup*, pa objekat klase *Veza*. Objekat *Ekst_Veza* obezbeđuje preslikavanje sa skupa objekata potklase u skup objekata superklase, a koristi se:

- pri modifikaciji nekog obeležja iz *Z*, bilo u o_p, bilo u o_s, da bi se ista izmena izvršila i u onom drugom i
- pri brisanju objekta o_s klase *Sup*, da bi se izbrisao i odgovarajući objekat o_p klase *Pot*, kada se briše i objekat (i_{o_s}, i_{o_p}) u klasi *Veza*.



Slika 7.20.

Objekti *Ekst_Sup* i *Ekst_Veza* su potrebni kao mehanizmi za upravljanje ekstenzijama klasa *Sup* i *Veza*. Na slici 7.20 je prikazana ideja opisane realizacije kontrole referencijalnog integriteta, a na slici 7.21 je dat pseudokod opisanog postupka obezbeđenja uslova za kontrolu referencijalnog integriteta. Pri tome, pretpostavka je da metoda za konstruisanje

7.6. Operacijska komponenta objektno - orijentisanog modela podataka

U prvoj polovini devedesetih godina, veći broj objektno - orijentisanih programskih jezika je ušao u svakodnevnu upotrebu. U te jezike svakako spadaju: Smalltalk, C++, Visual Basic,... Svi ti jezici se, u preovlađujućem obimu, koriste baš kao programski jezici, a ne kao jezici podataka za deklarisanje struktura i za manipulisanje objektno - orijentisanim bazama podataka. Za takve primene, koje zahtevaju formiranje i korišćenje baze podataka na eksternom memorijskom uređaju, potrebno je uvesti određena proširenja, koja se, pre svega odnose na mehanizme za razmenu podataka između programa i baze podataka. U trenutku pisanja ove knjige, ta proširenja se, još uvek nalaze u eksperimentalnoj ili razvojnoj fazi.

Od pomenutih programskih jezika, C++ se nameće kao jedan od ozbiljnih kandidata, koji će poslužiti za realizaciju operacijske komponente objektno - orijentisanih baza podataka. Zbog toga je izvršeno opredelenje da se putem njegove sintakse ilustruju postupci za deklarisanje i realizaciju koncepata objektno - orijentisanog modela podataka.

Jezik C++ je danas svakako najpopularniji objektno-orijentisani jezik opšte namene. To je veoma moćan, fleksibilan jezik, udoban za programiranje. Programeri su ga veoma rado i brzo prihvatili, baš kao i njegovog prethodnika - jezik C. Nažalost, C++ nije nimalo jednostavan jezik. Ponekad se čoveku učini da je to jezik bez granica i da ga ne može lako sagledati u celini. Jezik C++ nije čisti objektno-orijentisani programski jezik koji bi korisnika "naterao" da ga koristi na objektno-orijentisani način. Može se koristiti i kao "malo bolji C". Treba ga koristiti kao sredstvo za implementaciju objektno-orijentisanog modela i kao smernice za razmišljanje o problemu.

U cilju ilustracije operacijske komponente objektno-orijentisanog modela podataka nije neophodno poznavati sve detalje jednog tako složenog jezika kao što je C++, već je potrebno pravilno upotrebljavati one delove koji su u datom trenutku potrebni. Potrebno je da čitalac uoči šta se sve može uraditi u jeziku C++ i kako se to radi, bez insistiranja na sintaksi, kada za to ne postoji potreba. Cilj ove tačke nije programiranje u jeziku C++, pa se, u skladu sa tim, neće detaljno razmatrati organizacija programa, povezivanje i osnovni koncepti jezika C++, a posebno oni nasleđeni iz jezika C. Ta pitanja se razmatraju u literaturi, na primer [Str, Mi].Razmatranja će se ograničiti samo na objektno-orijentisane koncepte jezika C++, kao primer operacijske komponente objektno - orijentisanog modela.

Najvažniji objektno-orijentisani koncepti jezika C++ opisani i ilustrovani u ovoj tački su:

- klase,
- preklapanje operatora,
- nasleđivanje i
- generički mehanizam.

U primerima, neki delovi programskog teksta su samo skicirani i ispisani na srpskom jeziku (kurzivom) tako da se mogu lako razlikovati od ostalog teksta na engleskom jeziku, koji

predstavlja gotov programski kod. Ključne reči jezika C++ su, u primerima, pisane masnim slovima.

7.6.1. *Klase*

Klasa je osnovna organizaciona jedinica programa u objektno-orijentisanom jeziku C++. Konceptom klase se u jeziku C++ realizuju principi apstrakcije i inkapsulacije. Klasom se definiše tip u programu. Klasa se definiše navođenjem deklaracije klase, kojom se u program uvodi ime klase kao ime novog, korisničkog tipa. Grupisanjem podataka i funkcija u jedinstvenu strukturu koja realizuje novi tip, podržava se princip apstrakcije tipa, kao jedinstvene strukture podataka i operacija nad njom.

Primer 7.33. U primeru 7.5 uvedena je klasa *Osoba*, čije pojave imaju svojstva: ime, prezime, zanimanje i platu i koja se mogu predstaviti obeležjima *IME*, *PRZ*, *ZAN*, *PLT*. Svaka pojava klase treba da odgovori na akciju, koja traži da se osoba zaposli, da se osobi poveća plata, da se osobi prikaže plata, da se osoba predstavi (prikaže osnovne podatke o sebi) ili da se osoba penzioniše. Ove akcije se mogu predstaviti funkcijama *Zaposli*, *Povećaj_platu*, *Prikaži_platu*, *Prikaži_podatke* i *Penzioniši*. U jeziku C++ ova klasa može se predstaviti na sledeći način:

```
// Deklaracija klase
class Osoba {
        tip_podataka_koji_sadrži_ime_Osobe IME;
        tip_podataka_koji_sadrži_prezime_Osobe PRZ;
        tip_podataka_koji_sadrži_zanimanje_Osobe ZAN;
        tip_podataka_koji_sadrži_platu_Osobe PLT;
public:
        funkcija Zaposli();
        funkcija Povecaj_platu();
        funkcija Prikazi_platu();
        funkcija Prikazi_podatke();
        funkcija Penzioniši();
};
```

U narednom delu teksta objašnjavaju se koncepti i ključne reči jezika C++, koji su uvedeni u prethodnom primeru. Na samom početku primera prikazana je upotreba komentara u jeziku C++. Svi znaci iza duple kose crte (//), pa do kraja reda predstavljaju komentare programera i ne utiču na rad programa.

Ključna reč ***class*** jezika C++ označava da se počinje sa deklaracijom klase. Deklaracija u jeziku C++ predstavlja iskaz kojim se neko ime uvodi u program. Time se prevodiocu "objašnjava" šta predstavlja ime koje se njime uvodi i omogućava mu se da zna kako se to ime može koristiti u programu. Deklaracija klase uvodi ime - identifikator klase u program pomoću reči koja sledi iza ključne reči *class*. Sama deklaracija klase nalazi se između para vitičastih zagrada ({}) i obuhvata obeležja i metode. Deklaracija obuhvata

"spoljnji" izgled klase odnosno interfejs klase. Elementi, navedeni u deklaraciji klase, nazivaju se ***članovima klase***. Metode se u jeziku C++ realizuju funkcijama. Funkcija, koja je deo klase, naziva se ***funkcija članica***. Deklaracija klase završava se znakom tačka-zarez (;). Ova deklaracija ne obuhvata specifikaciju šta treba da uradi svaka od funkcija. Deklaracija sadrži informaciju da klasa poseduje funkciju i da se ona može izvršiti za svaki objekat klase. ***Telo funkcije*** je programski kod, koji se posebno definiše, van deklaracije klase, iskazom koji se naziva ***definicija funkcije***. Definicija, generalno, predstavlja iskaz, kojim se stvarno rezerviše memorijski prostor za neku promenljivu ili daje telo funkcije.

Primer 7.34.

```
// Definicija funkcije
// ...
funkcija Osoba::Prikazi_platu() {
  ispiši platu osobe u rečenici "Osoba ... ima platu ... din".
}
// ... □
```

Izraz "Osoba::" u definiciji funkcije u prethodnom primeru označava da funkcija *Prikazi_platu* pripada klasi *Osoba*. Telo funkcije je, u ovom primeru, opisano tekstom, a nalazi se između para vitičastih zagrada ({}). Svaka funkcija vraća rezultat, koji je određeni tip podataka. Tip podatka rezultata funkcije navodi se ispred identifikatora funkcije i u konkretnim primerima, umesto reči "*funkcija*", treba da stoji tip podatka. U prethodnom primeru, izraz "Osoba::Prikazi_platu()", predstavlja ***identifikator funkcije***.

Niz znakova "// ...", korišćen u prethodnom primeru, označava postojanje određenog programskog teksta koji, s obzirom na cilj uvođenja primera, nema neku posebnu ulogu, ali je njegovo postojanje neophodno zbog kompletnosti primera. Ista notacija primenjivaće se i u narednim primerima.

U cilju obezbeđenja inkapsulacije, pristup članovima klase kontroliše se deklaracijom javnih, privatnih i zaštićenih članova. ***Javnim*** članovima mogu pristupiti svi korisnici klase. ***Privatnim*** članovima mogu pristupiti samo funkcije članice klase. ***Zaštićenim*** članovima mogu pristupiti i članovi nasleđene klase. Upotrebom ključnih reči ***public***, ***private*** i ***protected*** unutar deklaracije klase članovi klase se deklarišu, redom, kao javni, privatni ili zaštićeni.

Tipovi podataka koje podržava jezik C++ mogu se klasifikovati u dve grupe: ugrađeni i korisnički. ***Ugrađeni tipovi*** su tipovi podataka koji postoje u samom jeziku i obuhvataju ***osnovne tipove podataka*** (*char*, *int*, *float*, *double*, *enum*, *void*) i deo ***izvedenih tipova*** (*nizovi*, *reference*, *pokazivači*, *konstante*, *funkcije*). ***Korisnički tipovi*** su tipovi podataka koje definiše programer. Grupa korisničkih tipova obuhvata drugi deo izvedenih tipova (*klase*, *strukture*, *unije*). Većina osobina tipova podataka nasleđena je iz jezika C, pa se u ovom tekstu neće detaljnije objašnjavati. Objašnjenja će se ograničiti samo na tipove podataka koji se koriste u primerima.

Klasa predstavlja tip za koji se mogu kreirati pojave. Ako je klasa tip, objekat se može smatrati promenljivom tog tipa u programu. Objekti date klase se u programu kreiraju i koriste na način prikazan u sledećem primeru.

Primer 7.35.

```
// U programu se deklarišu promenljive tipa Osoba:
Osoba Ivo;
// a zatim se koriste poruke:
Ivo.Zaposli();              // poziv funkcije Zaposli objekta Ivo;
Ivo.Povecaj_platu();        // poziv funkcije Povecaj_platu objekta Ivo;
Ivo.Prikazi_platu();        // poziv funkcije Prikazi_platu objekta Ivo;
Ivo.Prikazi_podatke();      // poziv funkcije Prikazi_podatke objekta Ivo;
Ivo.Penzionisi();           // poziv funkcije Penzionisi objekta Ivo; . □
```

Pošto objekat ima neko svoje unutrašnje stanje, koje treba uspostaviti na određen način na početku života objekta, potrebno je obezbediti mehanizam da se objekat, automatski, po nastanku dovede u ispravno početno stanje. Proces kreiranja objekta u vreme izvršavanja programa i njegovo dovođenje u ispravno stanje predstavlja njegovu ***inicijalizaciju***.

Funkcije klase koje obezbeđuju inicijalizaciju objekata klase nazivaju se konstruktorima. ***Konstruktor*** je specijalna funkcija članica klase koja se implicitno poziva pri kreiranju objekata klase. Konstruktor ima isto ime kao i klasa, nema tip koji vraća, može imati argumente i može se preklopiti, što znači da klasa može posedovati više konstruktora sa različitim tipovima argumenata. U jeziku C++ postoji mogućnost da se za jedno ime funkcije navede više različitih deklaracija i tada se za to ime kaže da je preklopljeno. To praktično znači da je moguće deklarisati i definisati više različitih funkcija sa istim identifikatorom.

Primer 7.36. Konstruktor u klasi *Osoba* iz primera 7.32. može se deklarisati na sledeći način:

```
// Deklaracija klase
class Osoba {
        tip_podataka_koji_sadrži_ime_Osobe IME;
        tip_podataka_koji_sadrži_prezime_Osobe PRZ;
        tip_podataka_koji_sadrži_zanimanje_Osobe ZAN;
        tip_podataka_koji_sadrži_platu_Osobe PLT;
public:
        funkcija Osoba(argumenti: za ime, prezime, zanimanje i platu); //konstruktor
        funkcija Zaposli();
        funkcija Povecaj_platu();
        funkcija Prikazi_platu();
        funkcija Prikazi_podatke();
        funkcija Penzioniši();
};

// Definicija konstruktora
// ...
funkcija Osoba::Osoba(argumenti: za ime, prezime, zanimanje i platu) {
```

smesti argumente za ime, prezime, zanimanje i platu u obeležja IME, PRZ, ZAN, PLT;
}
// ...

Objekti date klase se u programu kreiraju i koriste na sledeći način:

// U programu se inicijalizuju promenljive tipa *Osoba*:
Osoba Ivo ("Ivo","Ban","programer",400);

// a zatim se koriste poruke:

```
Ivo.Zaposli();              // poziv funkcije Zaposli objekta Ivo;
Ivo.Povecaj_platu();        // poziv funkcije Povecaj_platu objekta Ivo;
Ivo.Prikazi_platu();        // poziv funkcije Prikazi_platu objekta Ivo;
Ivo.Prikazi_podatke();      // poziv funkcije Prikazi_podatke objekta Ivo;
Ivo.Penzionisi();           // poziv funkcije Penzionisi objekta Ivo; . □
```

Moguće je definisati i funkciju članicu klase, koja će se automatski pozivati kada objekat prestaje da postoji. Ova funkcija naziva se ***destruktor***. Konstruktora za jednu klasu može biti više, dok je destruktor samo jedan i funkcija destruktora se ne može eksplicitno pozvati. Destruktor se takođe zove isto kao i klasa, ali ispred njega dolazi poseban znak "~". Destruktor nema tip koji vraća, nema argumente i ne može se preklopiti. Redosled poziva destruktora je obrnut od redosleda poziva konstruktora.

Primer 7.37. Destruktor u Klasi Osoba iz primera 7.35. se deklariše na sledeći način:

```
// Deklaracija klase
class Osoba {
        tip_podataka_koji_sadrži_ime_Osobe IME;
        tip_podataka_koji_sadrži_prezime_Osobe PRZ;
        tip_podataka_koji_sadrži_zanimanje_Osobe ZAN;
        tip_podataka_koji_sadrži_platu_Osobe PLT;
public:
        funkcija Osoba(argumenti: za ime, prezime, zanimanje i platu)  //konstruktor
        funkcija ~Osoba();                                             //destruktor
        funkcija Zaposli();
        funkcija Povecaj_platu();
        funkcija Prikazi_platu();
        funkcija Prikazi_podatke();
        funkcija Penzioniši();
}; . □
```

Nakon ovog početnog upoznavanja sa strukturom jezika C++ i objašnjenja za određene ključne reči, u sledećem primeru klase će se deklarisati bez navođenja programskog teksta na srpskom jeziku.

Primer 7.38. Klase iz primera 7.5

TipArtikla = ((*NAZIV* , *Chr*), (*IDA* , *Int*))
StavkaNarudžbine = ((*ARTIKAL*, *TipArtikla*), (*KOLIČINA*, *Int*)).
TipKupca = ((*NAZIV*, *Chr*), (*ADRESA*, *Chr*), (*STANJE_RAČ*, *Int*), (*PORUDŽBINE*, {*TipPorudžbine*})).
TipPorudžbine = ((*BRP*, *Int*), (*KUPAC*, *TipKupca*), (*SADRŽI*, {*StavkaNarudžbine*}).

mogu se deklarisati u jeziku C++ na sledeći način:

```
class TipArtikla {
        char NAZIV[35];
        int IDA;
public:
        TipArtikla ();
        ~TipArtikla();
//...
};
class StavkaNarudzbine {
        TipArtikla ARTIKAL;
        int KOLICINA;
public:
        StavkaNarudzbine ();
        ~StavkaNarudzbine();
//...
};
class TipKupca {
        char NAZIV[35];
        char ADRESA[20];
        int STANJE_RAC;
        TipPorudzbine PORUDZBINE;
public:
        TipKupca ();
        ~TipKupca();
//...
};
class TipPorudzbine {
        int BRP;
        TipKupca KUPAC;
        StavkaNarudzbine SADRZI;
public:
        TipPorudzbine ();
        ~TipPorudzbine();
//...
};
```

U prethodnom primeru ključne reči *char* i *int* označavaju osnovne tipove podataka. U programskom jeziku C++ niz znakova se smešta u memoriju kao jednodimenziona tabela tipa *char* kojoj se na kraju dodaje nula-znak "\0". Pošto u jeziku ne postoje operatori za operacije nad nizovima znakova, da bi se taj niz znakova mogao obradivati, potrebno mu je dodeliti pokazivač, koji pokazuje na početak niza. Deklaracija tipa *char* * označava izvedeni tip pokazivača na promenljive tipa *char*. U primeru 7.37, deklaracija *char* **NAZIV* deklarisala bi promenljivu *NAZIV* kao pokazivač. Ovaj podatak pokazuje na adresu u memoriji, u kojoj je smeštena neka promenljiva tipa *char*. U praktičnom radu se, deklaracija promenljivih tipa *char* pomoću pokazivača, češće koristi. Deklaracija pomoću nizova znakova je preglednija i jednostavnija sa stanovišta primera u tekstu.

7.6.2. *Preklapanje operatora*

Operatori ugradeni u jezik C++ mogu se definisati za korisničke tipove (klase). Preklapanje operatora je definisanje novog značenja za neki operator u programu. Preklopljeni operator se definiše kao funkcija čije je ime taj operator, iza čega sledi spisak parametara i telo same funkcije.

Primer 7.39. Preklapanje operatora se može ilustrovati na klasi *Datum*:

```
class Datum {
        int DAN,MESEC,GODINA;
public:
        Datum(int, int, int);
        Datum operator + (int);
        //...
};
Datum::Datum (int D, int M, int G) {
        DAN = D;
        MESEC = M;
        GODINA = G;
}
Datum Datum::operator + (int n) {
        // Programski kod za preklopljeni operator +
        // n - broj dana
        // Aproksimativni algoritam
        int D,M,G;
        G = n / 365;
        M = (n - G * 365 ) / 30;
        D =  n - G * 365 - M * 30;
        GODINA = GODINA + G;
        MESEC = MESEC + M;
        DAN = DAN + D;
```

```
}
void main (){
        Datum Danas (6.11.1995);          // poziv konstruktora;
        Danas = Danas + 10;               // Danas je 16.11.1995
        Danas.operator+ (5);              // Danas je 21.11.1995
        //...
}. □
```

U prethodnom primeru preklopljen je ugrađeni operator + za korisnički tip *Datum*. Ovaj primer obezbeđuje vršenje operacije sabiranja celog broja sa korisničkim tipom *Datum*. Primer je namenjen za ilustraciju preklapanja operatora, pa je u tom cilju i u cilju jednostavnosti implementacije, primenjen aproksimativni algoritam, koji pretpostavlja da sve godine imaju 365 dana i da svi meseci imaju po 30 dana. U primeru su uvedene i dve nove ključne reči ***void*** i ***main***. Tip *void* je poseban ugrađeni tip koji predstavlja prazan skup vrednosti. Ne postoje objekti ovog tipa već se on koristi za formiranje izvedenih tipova. Upotrebljava se kao tip rezultata funkcije koje ne vraćaju nikakvu vrednost ili kao tip pokazivača koji mogu da ukazuju na bilo koji objekat u memoriji. Obavezna funkcija *main* predstavlja glavni program. Program počinje izvršavanje pozivom funkcije sa identifikatorom *main*.

7.6.3. Nasleđivanje

Nasleđivanje u jeziku C++ realizuje se konceptom ***izvedenih*** klasa. Izvođenje klasa je koncept jezika C++ koji definiše izvedenu klasu K_1 (potklasu) iz klase K_2 (superklase), kao klasu čiji objekti imaju sve članove superklase i one članove koji su eksplicitno deklarisani u potklasi.

Primer 7.40. Klase iz primera 7.7 se, korišćenjem koncepta izvođenja, mogu deklarisati na sledeći način:

```
class Date {
//... Deklaracija tipa Date
}
class Money {
//... Deklaracija tipa Money
}
class Prog_jez {
//... Deklaracija tipa Prog_jez
}
class Osoba {
        char IME[15];
        char PRZ[20];
        Date DAT_ROD;
        void Prikazi_godine();
```

```
public:
        Osoba();
        ~Osoba();
        void Prikazi_ime();
        void Prikazi_prezime();
        void Izmeni_prezime();
};
class Zaposleni : public Osoba {
        Money PLATA;
public:
        Zaposleni();                    // Preuzima ulogu metode zaposli iz primera 7.7;
        ~Zaposleni();                   // Preuzima ulogu metode otpusti iz primera 7.7
        void Povecaj_platu();
        void Prikazi_platu();
};
class Programer : public Zaposleni {
        Prog_jez PROGJEZ;
public:
        Programer();
        ~Programer();
        //...
};. □
```

Potklasa se deklariše neobaveznim navođenjem jedne od ključnih reči *public*, *protected* ili *private* i obaveznim navođenjem naziva superklase iza znaka ":" u zaglavlju deklaracije klase. Upotrebom ključnih reči *public*, *protected* i *private* deklariše se ***javno***, ***zaštićeno*** i ***privatno izvođenje***. Ključna reč *public* u zaglavlju deklaracije potklase znači da javni i zaštićeni članovi superklase ostaju javni i zaštićeni članovi potklase. Javnim i zaštićenim članovima superklase se iz funkcija potklase pristupa neposredno kao i sopstvenim članovima. Zaštićenim izvođenjem, javni i zaštićeni članovi superklase postaju zaštićeni članovi potklase, a privatnim izvođenjem javni i zaštićeni članovi superklase postaju privatni članovi potklase.

Objekti potklase nasleđuju sva obeležja superklase, bez obzira na način izvođenja i poseduju svoje posebne članove koji su navedeni u deklaraciji potklase. U zavisnosti od načina izvođenja, pristup određenim nasleđenim članovima potklase je na nekim mestima u programu dozvoljen ili ne. Potklasa ne nasleđuje konstruktore, destruktore i, ako postoji, funkciju članicu *operator=* superklase.

U prethodnom primeru većina funkcija članica je zbog jednostavnosti deklarisana kao funkcije bez argumenata koje ne vraćaju nikakvu vrednost.

Primer 7.41. Na slici 7.10 su prikazana obeležja klase *Osoba*, njene potklase *Zaposleni* i jedan objekat klase *Zaposleni*. Objekat potklase *Zaposleni* se kreira na sledeći način:

```
class Money {
```

```
//.. Deklaracija tipa Money
}
class Osoba {
        char IME[15];
        char PRZ[20];
        int STAROST;
public:
        Osoba(char,char,int);
        ~Osoba();
};
class Zaposleni : public Osoba {
        Money PLT;
        char RAD_MES[20];
        char ODEL[20];
public:
        Zaposleni(char,char,int,int,char,char);
        ~Zaposleni();
};
void main () {
        // ...
        // Negde u programu

        Zaposleni Ana ("Ana", "Car", 23, 400, "progr.", "AOP");

        // ...
}. □
```

Objekat *Ana* se kreira pozivom konstruktora *Zaposleni*. Pri kreiranju objekta potklase, prvo se bira jedan od konstruktora potklase, koji će se pozvati. Pre izvršenja tog konstruktora, poziva se konstruktor superklase. Konstruktor superklase se bira prema argumentima navedenim u zaglavlju konstruktora potklase iza znaka ":". Izraz iza ":" se naziva ***inicijalizatorom***. Konstruktor superklase inicijalizuje onaj deo objekta potklase, koji sadrži vrednosti obeležja superklase. Zatim se inicijalizuju obeležja potklase redosledom kojim su deklarisana. Na kraju se izvršava telo konstruktora potklase. Definicija konstruktora klase *Zaposleni* prikazana je u sledećem primeru.

Primer 7.42.

```
// Definicija konstruktora
// ...
void Zaposleni::Zaposleni( char IME[15], char PRZ[20], int STAROST[20], int PLT,
char RAD_MES[20], char ODEL[20]) : Osoba (IME[15], PRZ[20], STAROST[20]) {
// ... Telo konstruktora
}
// ... □
```

Redosled izvršavanja destruktora je suprotan redosledu izvršavanja kostruktora. Prvo se izvršava destruktor potklase, zatim se ukidaju članovi suprotno redosledu inicijalizacije i na kraju se poziva destruktor superklase.

Jezik C++ omogućava da neka klasa u isto vreme bude izvedena iz više klasa. Tada svaki objekat potklase nasleđuje sve članove svih svojih superklasa. ***Višestruko nasleđivanje*** se, slično kao i jednostruko, realizuje konceptom ***višestrukog izvođenja***. Klasa se može izvesti iz više superklasa. Potklasa se višestruko izvodi tako što se u zaglavlju njene deklaracije navode imena njenih superklasa, razdvojene zarezom. Značenje ključnih reči *public*, *protected* i *private* je isto kao i kod jednostrukog izvođenja.

Primer 7.43. Posmatraju se klase: *Stanovnik*, *Zaposleni*, *Student* i *Zaposleni_Student*. Klasa *Zaposleni_student* ima dve direktno nadređene i jednu tranzitivnu superklasu. Direktno nadređene klase su: *Zaposleni* i *Student*. *Zaposleni_Student* nasleđuje osobine od sve tri klase. Na slici 7.16 je prikazana ova hijerarhija klasa i jedna pojava klase *Zaposleni_Student*.

```
class Stanovnik {
// ...
};
class Zaposleni : public Stanovnik {
// ...
};
class Student : public Stanovnik {
// ...
};
class Zaposleni_Student : public Zaposleni, public Student {
// ...
}; . □
```

7.6.4. Generički mehanizam

Generički mehanizam u jeziku C++ omogućava generisanje klasa i funkcija. Ovaj mehanizam realizuje se pomoću ***šablona*** (*template*). Šabloni klasa uvode u program parametrizovane tipove. Generički mehanizam omogućava da se u programu automatski generišu klase i funkcije prema datom šablonu. Šablon predstavlja definiciju klase ili funkcije, koja u sebi sadrži kao formalne argumente tipove podataka.Te tipove podataka će, u fazi generisanja konkretne klase ili funkcije, zameniti konkretni tipovi. Klase i funkcije generisane iz istog šablona imaju istu realizaciju, ali manipulišu različitim tipovima.

Primer 7.44. Neka je kreirana klasa *Zaposleni* i neka je potrebno omogućiti upravljanje njenom ekstenzijom. Tada se prvo definiše nova pojava ugrađene klase tipa kolekcija. Ako je *Radnici* naziv ekstenzije klase *Zaposleni*, tada bi se *Radnici* deklarisali

kao nova pojava ugrađene klase, recimo, klase *Skup*. Svaki objekat klase *Zaposleni*, nakon svog kreiranja u okviru klase *Zaposleni*, mora se eksplicitnom naredbom uvrstiti i u objekat - skup sa identifikatorom *Radnici*. Metode ugrađene klase *Skup*, omogućavaju upravljanje ekstenzijom klase *Zaposleni*, pošto je objekat sa identifikatorom *Radnici* pojava klase *Skup*. Moguća realizacija korišćenjem C++ generičkog mehanizma je sledeća:

```
template <class T>
        class Skup {
                T *Niz[100];
                static i;
        public:
                Skup() {};
                void UpisiClan (T *ut) {
                        i++;
                        niz [i] = ut;
                };
}
class Zaposleni {
        // obeležja klase Zaposleni
public:
        Zaposleni();
        ~Zaposleni();
        // funkcije  klase Zaposleni...
};
// ...
void main () {
        // ...
        Skup<Zaposleni> Radnici;
        Zaposleni Ivo;
        Radnici.UpisiClan (&Ivo);
        // ...
}. □
```

Deklaracija šablona u potpunosti odgovara deklaraciji obične klase, samo što ispred nje stoji ključna reč ***template*** <...>, koji označava da se radi o deklaraciji šablona, čiji je formalni argument tip *T* naveden između znakova " < " i " > ".

U prethodnom primeru uvedeno je i nekoliko novih ključnih reči jezika C++. Konstrukcija tipa "T *Niz[100]" deklariše niz pokazivača na tip *T*. Ključna reč ***static*** deklariše, a ujedno i definiše promenljivu *i* kao statički objekat. ***Statički objekat*** ima životni vek do kraja izvršavanja celog programa. Inicijalizuju se samo jednom. U primeru, *i* se koristi kao brojač članova niza. Jezik C++ podržava inicijalizaciju funkcija u okviru deklaracije klase, pa je na takav način izvršena inicijalizacija funkcije *UpisiClan*. Poziv funkcije *UpisiClan* kao parametar prenosi adresu objekta.

7.7. Komparacija relacionog i objektno - orijentisanog modela podataka

Objektno - orijentisani model podataka, kao novija paradigma, poseduje niz prednosti u odnosu na relacionu i druge klasične paradigme. U prednosti objektno - orijentisanog pristupa se ubrajaju: predstavljanje kompleksnih objekata, produktivnost programiranja, jedinstven jezik podataka, potencijalno bolje performanse, pogodnost za zadovoljenje zahteva novih oblasti primene, fleksibilnost, ograničenost ponašanja i snaga semantičke izražajnosti. Produktivnost programiranja, jedinstven jezik podataka, potencijalno bolje performanse i pogodnost za zadovoljenje zahteva novih primena, detaljnije su obrazloženi u posebnim tačkama ovog dela knjige. Fleksibilnost, ograničenost ponašanja i snaga semantičke izražajnosti se samo kratko komentarišu u nastavku teksta.

Fleksibilnost je posledica činjenice da su izmene tipova objekata i njihovih metoda lokalnog karaktera. Te izmene u relacionom modelu su kompleksnije, jer jednoj klasi objektno - orijentisanog modela odgovara (po pravilu) veći broj šema relacija.

Inkapsulacija ***ograničava*** ponašanje svakog objekta na prethodno definisane metode. Saglasno tome, lakše se otkrivaju i rešavaju problemi, nastali zbog greške u nekoj metodi. Takođe, manja je verovatnoća da dođe do nekontrolisanog prostiranja efekata te greške u bazi podataka.

Korišćenje svih postupaka (osim asocijacije) semantičkih modela podataka daje snagu objektno - orijentisanom modelu za verno modeliranje realnog sveta. Apstrakcija asocijacija, koju u semantičkim modelima predstavlja tip poveznika, u objektno - orijentisanom modelu se postiže indirektno, razmenom poruka između objekata. Ne postoji mogućnost da se u objektno - orijentisanom modelu direktno izrazi povezanost između dva objekta, jer se svaki objekat posmatra kao samostalan pojam.

U literaturi se neki put navodi da objektno - orijentisani model može, a relacioni ne može da podrži verzije istog objekta i duge transakcije. Iako objektno - orijentisana paradigma otklanja određene nedostatke relacionog modela, sam objektno - orijentisani model ne spominje verzije i duge transakcije, kao ni relacioni. Verzije i duge transakcije se povezuju sa objektno - orijentisanim modelom, jer su to mogućnosti, koje nedostaju relacionim SUBP, a bitne su za one aplikacije, za koje je objektno - orijentisani model pogodniji od relacionog, te se i očekuje da će objektno - orijentisani SUBP moći da zadovolje te zahteve.

7.7.1. Produktivnost programiranja

Činjenica da postojeći programski kod može skoro, ali ne u potpunosti da zadovolji zahteve nove aplikacije, verovatno predstavlja izvor najvećih frustracija korisnika relacionog i drugih klasičnih pristupa, pri pokušaju da se neki od postojećih programa ponovo upotrebi.

Organizovanjem objekata u klase, koje inkapsuliraju ponašanje objekata na način sličan apstraktnim tipovima podataka i organizujući klase u hijerarhiju nasleđivanja, objektno - orijentisano programiranje može obezbediti i dugo traženo delenje i ponovno korišćenje programskog koda. Programski kod, koji strukturi podataka daje smisao, više nije rasut po aplikativnim programima. U objektno - orijentisanom jeziku, nova klasa se može kreirati od već postojeće, navođenjem samo razlika u odnosu na neku već postojeću klasu. Sve ostalo nova klasa nasleđuje od stare. Obeležja i metodi ne treba da se definišu i pišu ispočetka, što smanjuje mogućnost unošenja novih grešaka. Nasleđivanje omogućava da se isti program može primeniti na objekte, koji pripadaju različitim klasama. U slučaju velikog broja klasa sa velikim brojem obeležja i metoda, koristi od delenja obeležja i metoda mogu biti velike.

7.7.2. *Jedinstven jezik*

Relacioni sistemi baza podataka su projektovani za realizaciju važne ali ograničene klase primena. Te primene se odnose na slučajeve kada treba memorisati velike količine dobro strukturiranih podataka i na njima izvršavati relativno jednostavne operacije. Primere predstavljaju baze poslovnih podataka preduzeća, rezervacije karata u aviosaobraćaju ili baze podataka u društveno političkim zajednicama. U takvim bazama podataka dominiraju:

- operacije pretraživanja na osnovu relativno malog i prethodno definisanog broja kriterijuma,
- ažuriranje podataka i
- takozvana navigacija između relativno malog broja datoteka ili relacija.

Posledica takvih primena je i razdvojenost programskog jezika za definisanje, selekciju i manipulisanje podacima, kakav je SQL jezik podataka, od programskog jezika opšte namene, kakvi su programski jezici treće generacije i proceduralni jezici četvrte generacije. Naredbe jezika podataka (za selekciju i manipulisanje podacima) se ugrađuju u programe, pisane u programskom jeziku opšte namene. Jezik podataka ima zadatak da obezbedi efikasan pristup bazi podataka, ali su njegove mogućnosti za izražavanje kompleksnih upita ograničene. Po pravilu, ne podržava definisanje rekurzivnih upita.

Primer 7.45. U svakom jeziku za selekciju i manipulisanje bazom podataka, može se definisati upit tipa "Ko je rukovodilac radnika x", ali retko kada i upit "Ko su sve rukovodioci radnika x". Poslednji upit podrazumeva da se kao odgovor dobije kompletna rukovodna hijerarhija za radnika x. Pošto to znači da treba tražiti rukovodiočevog rukovodioca dok takav postoji, ovakav upit se naziva rekurzivnim. □

Programski jezici opšte namene su, po pravilu, pogodni za definisanje proizvoljnih procedura pa često i rekurzivnih. Međutim, nisu predvidjeni za efikasno pretraživanje baze podataka. S druge strane, u primenama vezanim za baze podataka u inženjerskom projektovanju, grafici, geografskim informacionim sistemima i softverskom inženjerstvu, javlja se potreba integracije jezika podataka i jezika opšte namene u jedan jezik. Naime,

programski jezik opšte namene i jezik podataka, po pravilu se značajno razlikuju po sintaksi i modelu podataka, što značajno otežava njihovo zajedničko korišćenje.

Objekno - orijentisani jezici su proceduralni jezici. Poseduju osobine jezika opšte namene za obavljanje operacija u operativnoj memoriji. Poseduju i sintaksu za definisanje klasa i hijerarhije klasa (tj. za definisanje šeme baze podataka). Ti jezici predstavljaju dobru osnovu za izgradnju jedinstvenog jezika za rad sa bazom podataka. Potrebno ih je proširiti:

- metodama za smeštanje podataka na medijum eksternog memorijskog uređaja i
- deklarativnim upitnim jezikom.

7.7.3. *Bolje performanse*

Povezivanje objekata putem identifikatora predstavlja potencijalni izvor boljih performansi objektno - orijentisane u odnosu na relacionu bazu podataka. U objektno - orijentisanoj bazi podataka, vrednost obeležja X objekta o_1 je identifikator i_2, objekta o_2. Ako je neki aplikativni program već učitao objekat o_1 i sada treba da učita objekat o_2, SUBP to realizuje koristeći njegov identifikator i_2. Pristup objektu o_2 je direktan, na osnovu adrese. U relacionoj bazi podataka, X vrednost torke t_1 je vrednost ključa x torke t_2. Da bi se učitala torka t_2 u operativnu memoriju, potrebno je, prvo izvršiti traženje u indeksu ključa X, pa onda preneti torku t_2 u operativnu memoriju, što, u opštem slučaju, zahteva više od jednog pristupa.

Zaposleni

ID	*MBR*	*IME*	*PRZ*	*RM*
i_1	*159*	*Ivo*	*Ban*	i_3
i_2	*113*	*Ana*	*Tot*	i_4

Radno_mesto

ID	*NAZ_RM*	*PLT*	*STS*
i_3	*programer*	*450*	*visoka*
i_4	*projektant*	*500*	*visoka*

Slika 7.22.

Primer 7.46. Na slici 7.22 su prikazane dve pojave klase *Zaposleni* i dve pojave klase *Radno_Mesto*. Klasa *Radno_Mesto* je domen obeležja *RAD_MES* u klasi *Zaposleni*. Vrednost, memorisana u polju *RAD_MES* je identifikator jednog objekta klase *Radno_Mesto*. Pristup tom objektu klase *Radno_Mesto* se vrši putem identifikatora, a to znači direktno, na osnovu adrese na disku.

Postojeće relacione baze podataka dozvoljavaju korišćenje samo osnovnih tipova podataka u svojstvu domena obeležja. Povezivanje torki dve relacije se vrši putem stranog ključa. Ako je neki aplikacioni program pročitao torku (*159, Ivo, Ban, programer*) i treba sada da pristupi torci (*programer, 450, visoka, 2*), sistem mora izvršiti upit, koji prolazi kroz torke relacije *Radno_Mesto* tražeći torku sa vrednošću obeležja *NAZ_RM* = *pro-*

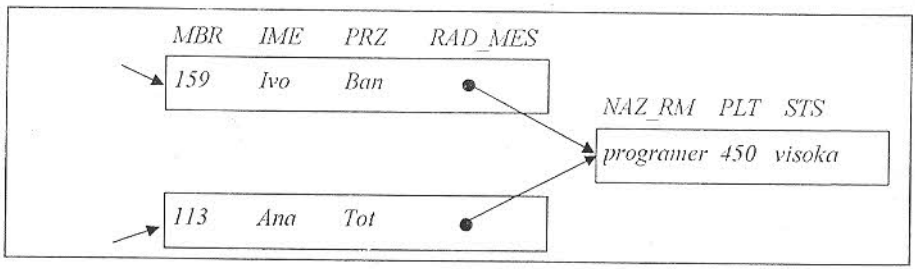
gramer. To traženje zahteva, u opštem slučaju više pristupa disku nego direktni pristup, na osnovu adrese, kao u slučaju objektno - orijentisane baze podataka. Slika 7.23 prikazuje relacionu reprezentaciju objektno - orijentisane baze podataka sa slike 7.22. □

MBR	*IME*	*PRZ*	*NAZ_RM*
159	*Ivo*	*Ban*	*programer*
113	*Ana*	*Tot*	*programer*

NAZ_RM	*PLT*	*STS*
programer	*450*	*visoka*
projektant	*500*	*visoka*

Slika 7.23.

Drugi aspekt doprinosa identifikatora boljim performansama objektno - orijentisanih baza podataka u poređenju sa relacionim je da se, pri prenosu objekata u operativnu memoriju, identifikatori, memorisani zajedno sa objektima, pretvaraju u memorijske pokazivače (virtuelne adrese). Pošto relacione baze podataka ne poznaju pojam identifikatora, one ne mogu ni memorisati memorijske pokazivače u torkama. Pristup relacionoj torci, koja je u operativnoj memoriji, vrši se indirektno, korišćenjem stranog ključa i odgovarajućeg indeksa. Mogućnost navigacije kroz memorijski rezidentne objekte je potpuno strana relacionom modelu, a pad performansi se ne može neutralisati jednostavnim obezbeđenjem velikog prostora za bafere u operativnoj memoriji. U aplikacijama, koje zahtevaju repetitivnu navigaciju kroz spregnute objekte u operativnoj memoriji, objektno - orijentisana baza podataka može biti znatno brža od relacione.



Slika 7.24.

Primer 7.47. Na slici 7.24 je prikazana memorijska reprezentacija objekata sa slike 7.23. Usmereni potezi ukazuju na povezanost objekata različitih klasa putem memorijskih adresa. □

7.7.4. *Objektno - orijentisani model podataka i zahtevi novih primena*

Objektno - orijentisani model podataka obezbeđuje mogućnosti za predstavljanje identiteta objekta, apstraktne podatke i odnose između objekata. Daje korisniku fleksibilnost i mogućnost širenja pri radu sa kompleksnim podacima. Sada će biti pokazano kako OOMP može da ukloni nedostatke relacionog modela u primeni na projektne baze podataka.

Nehomogeni podaci. Relacioni model postavlja homogene torke u relaciju, analogno, u objektno - orijentisanom modelu, klasa generiše slične objekte. Međutim, struktura podataka klase nije dvodimenzionalna. Može da sadrži višeznačna obeležja, što znači da se jednom objektu može pridružiti više vrednosti iz jednog domena. U relacionom modelu, ovakvi zahtevi dovode ili do redundanse podataka ili do dekomponovanja šeme relacije. Zajedničke osobine klasa se mogu izvući u cilju formiranja superklase. Hijerarhija klasa (latisa) obezbeđuje organizovan način upravljanja tipovima podataka.

Ekvivalentni objekti. Objektno - orijentisani model koristi generičke objekte za predstavljanje semantike ekvivalentnih objekata. Generički objekat poseduje obeležja koja identifikuju različite reprezentacije istog objekta. Definicijama generičkih objekata se mogu dodati ograničenja, tako da se modifikacije jedne reprezentacije reflektuju u drugim.

Razvoj šeme. Konvencionalni SUBP ne podržavaju efikasne mehanizme za evoluciju šeme najviše zbog toga što poslovne primene baze podataka ne zahtevaju česte izmene šeme. Dok je dodavanje takvih mehanizama tradicionalnim SUBP teško, objektno - orijentisani model dozvoljava definisanje korisničkih operacija, što olakšava širenje šeme. Evolucija šeme uključuje promene definicija unutar jedne klase, kao i definicije latise klasa. Može se definisati skup invarijanti i pravila, tako da šema ostane konzistentna kako evoluira.

Multimedija. Objektno - orijentisani model podataka predstavlja mnogo prirodniju osnovu za implementaciju funkcija, neophodnih za upravljanje multimedijskim podacima. Multimedijski podaci se definišu kao podaci proizvoljnog tipa (brojevi, nizovi karaktera, *Zaposleni*, *Preduzeće*, slika, tekst, zvuk, grafika, film, dokument sa slikama, tekstom i tako dalje). Pri tome, to su nizovi velike i promenljive dužine. Objektno - orijentisani model podataka stvarno omogućava definisanje proizvoljnih tipova podataka, pa i nizova proizvoljne i promenljive dužine. Definišu se kao nov tip podataka (klasa). Saglasno tome, niz nije ništa drugo nego objekat sa jedinstvenim identifikatorom. Nizu se može pristupiti putem njegovog identifikatora, ili na osnovu sadržaja. Međutim, objektno orijentisana paradigma, sama po sebi, ne rešava probleme efikasnog memorisanja, čitanja i ažuriranja multimedijskih podataka. To su ozbiljni problemi, koji treba da se reše u izgradnji objektno - orijentisanog SUBP.

1. Prilog

Generalizovane zavisnosti podataka

U ovom prilogu će biti prezentiran jedan od načina za generalizovani prikaz zavisnosti podataka, uvedenih u glavi 5, pri čemu se polazi od pretpostavke da se sve zavisnosti podataka (tj. "zakonitosti" koje važe između podataka) mogu, na unificiran način, prikazati putem jednog, ili više tabloa i određenih pravila za njihovu interpretaciju. Tablo, na taj način, preuzima ulogu minimalnog simboličkog uzorka podataka koji zadovoljava određene zavisnosti.

Na početku priloga se uvode pojmovi generalizovanih T - zavisnosti*) i generalizovanih E - zavisnosti,**) a zatim se demonstrira način prikaza ostalih tipova zavisnosti putem generalizovanih zavisnosti. Nakon toga se uvodi pojam izvedenih F - zavisnosti, pri čemu se pokazuje da funkcionalne zavisnosti predstavljaju specijalan slučaj izvedenih F - zavisnosti.

U nastavku priloga, činjenica da je tablo τ definisan nad skupom obeležja R, će biti označavana kao $\tau(R)$. Ukoliko se skup obeležja tabloa podrazumeva, tada se on može izostaviti. Pretpostavlja se, takođe, da se bilo koji tablo τ definiše nad beskonačnim, prebrojivim skupom simbola - promenljivih: $Sym = \{x, y, z, w, x_1,..., x_k,..., x_1^1,..., x_m^1,..., a, b, c,...\}$, pri čemu je nebitno (za razliku od problematike, izložene u tački 5.5.4) da li su pri formiranju tabloa upotrebljene poznate, ili nepoznate promenljive. Pored toga, koristi se i pojam funkcije preslikavanja simbola u vrednosti g: $Sym \rightarrow Dom$, gde je

$$Dom = \bigcup_{A \in U} dom(A),$$

---

*) Na engleskom *template dependency*, ili *tuple dependency*, što bi doslovno značilo "šablon zavisnost", odnosno "torka-zavisnost".

**) Na engleskom *equality dependency*, što bi doslovno značilo "zavisnost jednakosti".

kao i poopštenje, koje se odnosi na preslikavanje skupa torki simbola u skup torki vrednosti. Specijalno, ovde će biti upotrebljavan pojam preslikavanja tabloa τ, kao skupa torki simbola, u relaciju r, kao skup torki vrednosti, u oznaci $g(\tau) = r$, koje se još naziva i *interpretacija tabloa*.

Analogno pojmu projekcije relacije na skup obeležja, uvodi se pojam ***projekcije tabloa*** $\tau(R)$ na skup obeležja $X \subseteq R$: $\pi_X(\tau) = \{t[X] \mid t \in \tau\}$.

1.1. Generalizovana T - zavisnost

Sledećom definicijom se uvodi način formiranja ("sintaksa") generalizovane T - zavisnosti, a zatim se definiše i način interpretacije generalizovane T - zavisnosti.

Definicija 1.1. Izraz oblika $<\tau(R), \tau'(X)>$, pri čemu su $\tau(R) = \{t_i \mid i \in \{1,...,k\}\}$ i $\tau'(X) = \{t_i \mid i \in \{1,...,m\}\}$ tabloi, takvi da važi

1^0 $X \subseteq R \subseteq \mathcal{U}$, gde je $\mathcal{U}$ univerzalni skup obeležja, i

2^0 $(\forall A_i \in X)(\forall t_j \in \tau'(X))(\exists B \in R)(t_j[A_i] \in \pi_B(\tau(R)))$,

se naziva ***generalizovana T - zavisnost***, ili kraće T - zavisnost.

Primer 1.1. Na slici 1.1 su prikazani tabloi $\tau(ABCD)$, $\tau'(ABCD)$ i $\tau''(ABC)$. Izraz $<\tau, \tau'>$ predstavlja generalizovanu T - zavisnost, dok izraz $<\tau, \tau''>$ ne predstavlja T - zavisnost, jer je zbog upotrebljenih simbola x_2 i z_5 narušen uslov 2^0 definicije 1.1.

τ	A	B	C	D
t_1	x_1	y_1	z_1	w_1
t_2	x_1	y_2	z_2	w_2

τ'	A	B	C	D
t'	x_1	y_1	z_2	w_2

τ''	A	B	C
t_1''	x_1	y_1	z_5
t_2''	x_2	y_2	z_2

Slika 1.1.

Ako za proizvoljnu T - zavisnost $<\tau(R), \tau'(X)>$ važi da je $|\tau'(X)| = 1$, tada će ona biti prikazivana u obliku $<\tau(R), (x_1,...,x_m)(X)>$, tj. $<\tau(R), t(X)>$, pri čemu je $t = (x_1,..., x_m) \in \tau'(X)$.

Skup svih mogućih T - zavisnosti nad skupom obeležja $\mathcal{U}$ je skup $\mathcal{G}_T(\mathcal{U}) = \{<\tau(R), \tau'(X)> \mid X \subseteq R \subseteq \mathcal{U}\}$. Može se primetiti da je skup $\mathcal{G}_T(\mathcal{U})$ beskonačan, prebrojiv skup.

Ukoliko za proizvoljnu T - zavisnost $<\tau(R), \tau'(X)>$ važi da je $X = R$, onda se ona naziva ***potpuna*** T - zavisnost. Ako za proizvoljnu T - zavisnost $<\tau(R), \tau'(X)>$ važi da je $X \subset R$, reč je o ***ugrađenoj*** T - zavisnosti.

Definicija 1.2. Neka je dat skup simbola Sym i unija svih domena Dom. Relacija $r(R)$ zadovoljava T - zavisnost $<\tau(R), \tau'(X)>$, u oznaci $r \models <\tau(R), \tau'(X)>$, ako važi

$$(\forall g: Sym \rightarrow Dom)(g(\tau(R)) \subseteq r \Rightarrow g(\tau'(X)) \subseteq \pi_X(r)).$$

Primer 1.2. Posmatraju se generalizovana T - zavisnost $<\tau, \tau'>$ iz prethodnog primera i relacije r_1 i r_2, prikazane na slici 1.2. Primenom definicije 1.2, može se pokazati da relacija r_1 zadovoljava T - zavisnost $<\tau, \tau'>$, a r_2 ne zadovoljava $<\tau, \tau'>$.

Da bi se pokazalo da važi $r_1 \models <\tau, \tau'>$, formiraju se sve moguće funkcije preslikavanja simbola u vrednosti g_i, za koje važi da je $g_i(\tau) \subseteq r_1$. Na slici 1.3 su prikazane sve moguće interpretacije tabloa $g_i(\tau)$, koje ispunjavaju uslov $g_i(\tau) \subseteq r_1$. Može se proveriti da u svim slučajevima, za $i \in \{1,..., 13\}$, važi $g_i(\tau') \subseteq r_1$. Pri tome, treba zapaziti da svaka dvotorčana relacija $g_i(\tau)$ nastaje na dva različita načina:

- $g_i(l_1) = t_1^i \wedge g_i(l_2) = t_2^i$, ili
- $g_i(l_2) = t_1^i \wedge g_i(l_1) = t_2^i$.

U oba slučaja je zadovoljen uslov $g_i(l') \in r_1$.

Da bi se pokazalo da uslov $r_2 \models <\tau, \tau'>$ ne važi, dovoljno je pronaći jednu funkciju g: $Sym \rightarrow Dom$, takvu da je $g(\tau) \subseteq r_2$ i $g(\tau') \not\subseteq r_2$, odnosno $g(l') \notin r_2$. Posmatra se funkcija g_9, pri čemu je $g_9(l_1) = t_1^9 \wedge g_9(l_2) = t_2^9$ i važi $g_9(\tau) \subseteq r_2$. S druge strane, zaključuje se da $g_9(l') = (a_1, b_1, c_2, d_2) \notin r_2$. □

r_1	A	B	C	D
	a_1	b_1	c_1	d_1
	a_1	b_1	c_2	d_2
	a_1	b_2	c_1	d_1
	a_1	b_2	c_2	d_2
	a_2	b_1	c_1	d_1
	a_2	b_2	c_1	d_1

r_2	A	B	C	D
	a_1	b_1	c_1	d_1
	a_2	b_1	c_2	d_2
	a_1	b_2	c_1	d_1
	a_1	b_2	c_2	d_2

Slika 1.2.

Generalizovana T - zavisnost $<\tau(R), \tau'(X)>$ je, u opštem slučaju, ***tipski neinterpretirana zavisnost*** (u daljem tekstu će se koristi termin ***netipska zavisnost***), ukoliko njena interpretacija zahteva međusobno upoređivanje vrednosti različitih obeležja, odnosno ako važi:

$$(1.1) \qquad (\exists A \in X)(\exists l_j \in \tau'(X))(\exists B \in R)(A \neq B \wedge l_j[A] \in \pi_B(\tau(R))).$$

To znači da mora postojati domenska kompatibilnost svih obeležja T - zavisnosti, za koja važi uslov (1.1).

U nastavku teksta se definišu pojmovi tipski interpretiranog tabloa i tipski interpretirane T - zavisnosti.

Definicija 1.3. Tablo $\tau(R)$, $R \subseteq \mathcal{U}$, se naziva ***tipski interpretiran tablo*** (ili kraće ***tipski tablo***), ako važi:

$$(\forall A, B \in R)(A \neq B \Rightarrow \pi_A(\tau(R)) \cap \pi_B(\tau(R)) = \emptyset).$$

$g_1(\tau)$	A	B	C	D
t_1^1	a_1	b_1	c_1	d_1

$g_2(\tau)$	A	B	C	D
t_1^2	a_1	b_1	c_2	d_2

$g_3(\tau)$	A	B	C	D
t_1^3	a_1	b_2	c_1	d_1

$g_4(\tau)$	A	B	C	D
t_1^4	a_1	b_2	c_2	d_2

$g_5(\tau)$	A	B	C	D
t_1^5	a_1	b_1	c_1	d_1
t_2^5	a_1	b_1	c_2	d_2

$g_6(\tau)$	A	B	C	D
t_1^6	a_1	b_2	c_1	d_1
t_2^6	a_1	b_2	c_2	d_2

$g_7(\tau)$	A	B	C	D
t_1^7	a_1	b_1	c_1	d_1
t_2^7	a_1	b_2	c_1	d_1

$g_8(\tau)$	A	B	C	D
t_1^8	a_1	b_1	c_2	d_2
t_2^8	a_1	b_2	c_2	d_2

$g_9(\tau)$	A	B	C	D
t_1^9	a_1	b_1	c_1	d_1
t_2^9	a_1	b_2	c_2	d_2

$g_{10}(\tau)$	A	B	C	D
t_1^{10}	a_1	b_1	c_2	d_2
t_2^{10}	a_1	b_2	c_1	d_1

$g_{11}(\tau)$	A	B	C	D
t_1^{11}	a_2	b_1	c_1	d_1

$g_{12}(\tau)$	A	B	C	D
t_1^{12}	a_2	b_2	c_1	d_1

$g_{13}(\tau)$	A	B	C	D
t_1^{13}	a_2	b_1	c_1	d_1
t_2^{13}	a_2	b_2	c_1	d_1

Slika 1.3.

U slučaju da tablo $\tau(R)$ nije tipski, onda za sve parove obeležja $A, B \in R$, koji narušavaju uslov definicije 1.3, mora biti obezbeđena domenska kompatibilnost: $dom(A) \subseteq dom(B)$, ili $dom(B) \subseteq dom(A)$.

Definicija 1.4. Generalizovana T - zavisnost $<\tau(R), \tau'(X)>$, pri čemu su $\tau(R) = \{t_i \mid i \in \{1,...,k\}\}$ i $\tau'(X) = \{t_i \mid i \in \{1,...,m\}\}$ tipski tabloi, naziva se ***tipski interpretiranom*** (ili samo ***tipskom***) T - zavisnošću, ako je ispunjen uslov:

$$(\forall A_i \in X)(\forall B \in R)(B \neq A_i \Rightarrow \pi_{A_i}(\tau'(X)) \cap \pi_B(\tau(R)) = \emptyset)). \tag{1.2}$$

Koristeći se uslovom 2^0 definicije 1.1 i formulom (1.2), može se dokazati da je za tipsku T - zavisnost $<\tau(R), \tau'(X)>$ ispunjen sledeći uslov:

$$(\forall A_i \in X)(\pi_{A_i}(\tau'(X)) \subseteq \pi_{A_i}(\tau(R))).$$

Primer 1.3. T - zavisnost $<\tau, \tau'>$, definisana u primeru 1.1, predstavlja tipsku zavisnost.

1.2. *Generalizovana E - zavisnost*

Kao i u prethodnom slučaju, prvo se uvodi pojam generalizovane E - zavisnosti, a zatim i način njene interpretacije. U opštem slučaju, generalizovana E - zavisnost se, takođe, definiše kao ***tipski neinterpretirana zavisnost*** (ili samo ***netipska zavisnost***), a zatim se definiše potklasa tipski interpretiranih E - zavisnosti.

Definicija 1.5. Izraz oblika $<\tau(R), E>$, pri čemu je $\tau(R) = \{l_i \mid i \in \{1,...,k\}\}$ tablo, a $E = \{ Eq^i(\lambda^i_p, \lambda^i_q) \mid i \in \{1,...,m\}\}$ konačan skup predikata, takvih da važi

$$(1.3) \qquad (\forall i \in \{1,...,m\})((\exists A \in R)(\lambda^i_p \in \pi_A(\tau(R))) \wedge (\exists B \in R)(\lambda^i_q \in \pi_B(\tau(R))))$$

se naziva ***generalizovana E - zavisnost***, ili kraće E - zavisnost.

Ukoliko se simboli tabloa λ^i_p i λ^i_q nekog predikata $Eq^i(\lambda^i_p, \lambda^i_q) \in E$, pojavljuju nad različitim obeležjima:

$$\lambda^i_p \in \pi_A(\tau(R)) \wedge \lambda^i_q \in \pi_B(\tau(R)) \wedge A \neq B,$$

tada je $<\tau(R), E>$ netipska generalizovana E - zavisnost i zahteva se da obeležja A i B budu domenski kompatibilna ($dom(A) \subseteq dom(B)$, ili $dom(B) \subseteq dom(A)$).

Primer 1.4. Na slici 1.4 je prikazan tablo $\tau(ABCD)$. Izraz $<\tau, E>$, pri čemu je $E = \{Eq(x_2, x_3)\}$, predstavlja netipsku generalizovanu E - zavisnost. Pri tome, obeležja A i D moraju biti domenski kompatibilna.

τ	A	B	C	D
l_1	x_1	y_1	z_1	x_3
l_2	x_1	y_2	z_2	x_4
l_3	x_2	y_2	z_1	x_5

Slika 1.4.

Definicija 1.6. Generalizovana E - zavisnost $<\tau(R), E>$ se naziva ***tipski interpretiranom*** (ili samo ***tipskom***) E - zavisnošću, ako je $\tau(R)$ tipski tablo, a za svaki predikat $Eq^i(\lambda^i_p, \lambda^i_q) \in E$ važi da:

$$(1.4) \quad (\exists A \in R)(\{ \lambda^i_p, \lambda^i_q \} \subseteq \pi_A(\tau(R)) \wedge (\forall B \in R)(B \neq A \Rightarrow \{ \lambda^i_p, \lambda^i_q \} \cap \pi_B(\tau(R)) = \varnothing)).$$

Pri tome se predikat $Eq^i(\lambda^i_p, \lambda^i_q)$ prikazuje u obliku $Eq^i_A(\lambda^i_p, \lambda^i_q)$, gde je A obeležje za koje važi uslov $\{\lambda^i_p, \lambda^i_q\} \subseteq \pi_A(\tau(R))$.

Primer 1.5. Na slici 1.5 je prikazan tablo $\tau(ABCD)$. Izraz $<\tau, E>$, pri čemu je $E = \{Eq_D(w_1, w_3)\}$, predstavlja tipsku generalizovanu E - zavisnost, dok izraz $<\tau, E'>$, pri čemu je $E' = \{Eq_C(z_1, z_2), Eq_D(w_1, w_4)\}$, ne predstavlja E - zavisnost, jer je za simbol w_4 narušen uslov (1.3).

τ	A	B	C	D
l_1	x_1	y_1	z_1	w_1
l_2	x_1	y_2	z_2	w_2
l_3	x_2	y_2	z_1	w_3

Slika 1.5.

Skup svih mogućih E - zavisnosti nad skupom obeležja $\mathcal{U}$ se obeležava sa $\mathcal{G}_E(\mathcal{U})$. Može se primetiti da je $\mathcal{G}_E(\mathcal{U})$ beskonačan, prebrojiv skup.

Skup svih generalizovanih zavisnosti, u oznaci $\mathcal{G}(\mathcal{U})$, je skup

$$\mathcal{G}(\mathcal{U}) = \mathcal{G}_T(\mathcal{U}) \cup \mathcal{G}_E(\mathcal{U}).$$

Ako za proizvoljnu (tipsku) E - zavisnost $<\tau(R), E>$ važi da je $|E| = 1$, onda će ona biti prikazivana u obliku $<\tau(R), Eq(\lambda^j_p, \lambda^j_q)>$, odnosno $<\tau(R), Eq_A(\lambda^j_p, \lambda^j_q)>$.

Definicija 1.7. Neka je dat skup simbola *Sym* i unija svih domena *Dom*. Relacija $r(R)$ zadovoljava E - zavisnost $<\tau(R), E>$ (u oznaci $r \models <\tau(R), E>$) ako važi:

$$(\forall g: Sym \rightarrow Dom)(g(\tau(R)) \subseteq r \Rightarrow (\forall\, Eq(\lambda^j_p, \lambda^j_q) \in E)(g(\lambda^j_p) = g(\lambda^j_q))).$$

Primer 1.6. Posmatraju se generalizovana E - zavisnost $<\tau, E>$ iz prethodnog primera i relacije r_1 i r_2, prikazane na slici 1.6. Primenom definicije 1.7, može se pokazati da relacija r_1 zadovoljava E - zavisnost $<\tau, E>$, a r_2 ne zadovoljava $<\tau, E>$.

r_1	A	B	C	D
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_2	c_2	d_2
t_3	a_2	b_2	c_1	d_1

r_2	A	B	C	D
u_1	a_1	b_1	c_1	d_1
u_2	a_1	b_2	c_2	d_2
u_3	a_2	b_2	c_1	d_2

Slika 1.6.

Da bi se pokazalo da važi $r_1 \models <\tau, E>$, formiraju se sve moguće funkcije preslikavanja simbola u vrednosti g_i, za koje važi da je $g_i(\tau) \subseteq r_1$. Na slici 1.7 su prikazane sve

moguće interpretacije tabloa $g_i(\tau)$, koje ispunjavaju uslov $g_i(\tau) \subseteq r_1$. Može se proveriti da u svim slučajevima, za $i \in \{1,...,4\}$, važi $g_i(w_1) = g_i(w_3)$. Pri tome treba uočiti da je jedina mogućnost preslikavanja pri kojem važi $g_1(\tau) \subseteq r_1$, ona za koju je ispunjeno da $g_1(l_1) = t_1^1$, $g_1(l_2) = t_2^1$ i $g_1(l_3) = t_3^1$ i tada je $g_1(w_1) = g_1(w_3) = d_1$.

Da bi se pokazalo da uslov $r_2 \models <\tau, E>$ ne važi, dovoljno je pronaći jednu funkciju g: $Sym \rightarrow Dom$, takvu da je $g(\tau) \subseteq r_2$ i $g(w_1) \neq g(w_3)$. Posmatra se funkcija g, takva da je $g(l_1) = u_1$, $g(l_2) = u_2$ i $g(l_3) = u_3$, što znači da važi $g(\tau) \subseteq r_2$. S druge strane, zaključuje se da $g(w_1) = d_1$ i $g(w_3) = d_2$, tako da je $g(w_1) \neq g(w_3)$. □

$g_1(\tau)$	A	B	C	D
t_1^1	a_1	b_1	c_1	d_1
t_2^1	a_1	b_2	c_2	d_2
t_3^1	a_2	b_2	c_1	d_1

$g_2(\tau)$	A	B	C	D
t_1^2	a_1	b_1	c_1	d_1

$g_3(\tau)$	A	B	C	D
t_1^3	a_1	b_2	c_2	d_2

$g_4(\tau)$	A	B	C	D
t_1^4	a_2	b_2	c_1	d_1

Slika 1.7.

1.3. *Generalizovani način prikaza zavisnosti podataka*

U nastavku priloga će biti prezentirani načini za prikaz funkcionalnih zavisnosti, višeznačnih zavisnosti, zavisnosti spoja i zavisnosti sadržavanja putem generalizovanih zavisnosti.

1.3.1. *Generalizovani način prikaza funkcionalnih zavisnosti*

Funkcionalne zavisnosti se mogu prikazati putem tipskih E - zavisnosti, čiji tablo sadrži dve torke, a predikati Eq se definišu za svako obeležje desne strane date funkcionalne zavisnosti.

Teorema 1.1. Neka je data funkcionalna zavisnost $X \rightarrow A$, pri čemu je $X \subseteq \mathcal{U}$ i $A \in \mathcal{U}$, gde je $\mathcal{U}$ univerzalni skup obeležja, i neka je data tipska E - zavisnost $<\tau_{XA}(\mathcal{U}), Eq_A(a_1, a_2)>$, takva da je $\tau_{XA}(\mathcal{U}) = \{l_1, l_2\}$, pri čemu važi da je:

$$l_1[X] = l_2[X] \wedge l_1[A] = a_1 \wedge l_2[A] = a_2.$$

Grafička reprezentacija tabloa je data na slici 1.8, pri čemu je oznakom "-" naznačeno da simbol na datoj poziciji tabloa nije od važnosti. Posmatra se proizvoljna relacija $r \in SAT(\mathcal{U})$. Važi $r \models \langle \tau_{XA}(\mathcal{U}), Eq_A(a_1, a_2) \rangle$, ako i samo ako r zadovoljava $X \rightarrow A$.

$\tau_{XA}(\mathcal{U})$	X	A	$\mathcal{U} \setminus X$
l_1	x_1	a_1	-
l_2	x_1	a_2	-

Slika 1.8.

Dokaz. ($\Rightarrow$) Pretpostavlja se da r zadovoljava $X \rightarrow A$. Neka je $g(\tau_{XA}(\mathcal{U})) = \{t_1, t_2\}$ proizvoljna interpretacija tabloa $\tau_{XA}(\mathcal{U})$, takva da je $g(\tau_{XA}(\mathcal{U})) \subseteq r$. Pošto je $t_1[X] = t_2[X]$, saglasno funkcionalnoj zavisnosti $X \rightarrow A$, važi da je $t_1[A] = t_2[A]$, odnosno $g(a_1) = g(a_2)$. Pošto je $g(\tau_{XA}(\mathcal{U}))$ proizvoljno preslikavanje, sledi da:

$$(\forall g: Sym \rightarrow Dom)(g(\tau_{XA}(\mathcal{U})) \subseteq r \Rightarrow g(a_1) = g(a_2)),$$

što znači da važi $r \models \langle \tau_{XA}(\mathcal{U}), Eq_A(a_1, a_2) \rangle$.

($\Leftarrow$) Pretpostavlja se da važi $r \models \langle \tau_{XA}(\mathcal{U}), Eq_A(a_1, a_2) \rangle$. Neka su $t_1, t_2 \in r$ proizvoljno odabrane torke relacije r, takve da je $t_1[X] = t_2[X]$. Neka je g: $Sym \rightarrow Dom$ takvo preslikavanje, za koje važi da je $g(\tau_{XA}(\mathcal{U})) = \{t_1, t_2\}$.*) Na osnovu pretpostavke o važenju E - zavisnosti $\langle \tau_{XA}(\mathcal{U}), Eq_A(a_1, a_2) \rangle$, sledi da je $g(a_1) = g(a_2)$, odnosno $t_1[A] = t_2[A]$. Pošto su t_1 i t_2 proizvoljno odabrane torke, zaključuje se da $(\forall t_1, t_2 \in r)(t_1[X] = t_2[X] \Rightarrow t_1[A] = t_2[A])$, odnosno da r zadovoljava $X \rightarrow A$.

Na osnovu teoreme 1.1 i pravila dekompozicije desne strane funkcionalne zavisnosti, sledi da se bilo koja zavisnost $X \rightarrow Y$, pri čemu je $X, Y \subseteq \mathcal{U}$, može prikazati putem ekvivalentne E - zavisnosti $\langle \tau_{XY}(\mathcal{U}), E \rangle$, takve da za svako obeležje $A_i \in Y$ postoji jedan predikat $Eq_{A_i}(a_1^i, a_2^i) \in E$:

$$E = \{Eq_{A_i}(a_1^i, a_2^i) \mid A_i \in Y\}.$$

Primer 1.7. Generalizovana E - zavisnost $\langle \tau(\mathcal{U}), Eq_C(z_1, z_2) \rangle$, pri čemu je tablo $\tau(\mathcal{U})$ prikazan na slici 1.9, ekvivalentna je funkcionalnoj zavisnosti $AB \rightarrow C$.

$\tau(\mathcal{U})$	A	B	C	D	E
l_1	x_1	y_1	z_1	u_1	w_1
l_2	x_1	y_1	z_2	u_2	w_2

Slika 1.9.

*) S obzirom na to da važi $l_1[X] = l_2[X]$ i $t_1[X] = t_2[X]$, zaključuje se da mora postojati funkcija g, s navedenom osobinom.

1.3.2. *Generalizovani način prikaza višeznačnih zavisnosti*

Višeznačne zavisnosti se mogu prikazati putem tipskih T - zavisnosti, čiji prvi tablo sadrži dve torke, s jednakim simbolima nad obeležijma leve strane date višeznačne zavisnosti i različitim simbolima nad ostalim obeležjima, a drugi tablo sadrži jednu torku koja predstavlja kombinaciju simbola torki prvog tabloa.

Teorema 1.2. Neka je dat skup obeležja $R \subseteq \mathcal{U}$ i višeznačna zavisnost $X \rightarrow\rightarrow Y \mid R \setminus XY$ (skraćeno $X \rightarrow\rightarrow Y$), pri čemu je $X, Y \subseteq R$, gde je $\mathcal{U}$ univerzalni skup obeležja i neka je data tipska T - zavisnost $<\tau_{XY}(\mathcal{U}), I(R)>$, takva da je $\tau_{XY}(\mathcal{U}) = \{I_1, I_2\}$, pri čemu važi:

$$I_1[X] = I_2[X] \wedge (\forall A \in R \setminus X)(I_1[A] \neq I_2[A]), \tag{1.5}$$

a I je torka, takva da važi:

$$I[XY] = I_1[XY] \wedge I[R \setminus XY] = I_2[R \setminus XY]. \tag{1.6}$$

Grafička reprezentacija tabloa je data na slici 1.10, pri čemu je oznakom "-" naznačeno da simbol na datoj poziciji tabloa nije od važnosti. Neka je $r \in SAT(\mathcal{U})$. Važi $r \models <\tau_{XY}(\mathcal{U}), I(R)>$, ako i samo ako r zadovoljava $X \rightarrow\rightarrow Y$.

$\tau_{XY}(\mathcal{U})$	X	Y	$R \setminus XY$	$\mathcal{U} \setminus R$
	x_1	y_1	z_1	-
	x_1	y_2	z_2	-

$I(R)$	X	Y	$R \setminus XY$
	x_1	y_1	z_2

Slika 1.10.

Dokaz. ($\Rightarrow$) Pretpostavlja se da r zadovoljava višeznačnu zavisnost $X \rightarrow\rightarrow Y$. Neka je $g(\tau_{XY}(\mathcal{U})) = \{t_1, t_2\}$ proizvoljna interpretacija tabloa, takva da je $g(\tau_{XY}(\mathcal{U})) \subseteq r$. Pošto je $t_1[X] = t_2[X]$, saglasno višeznačnoj zavisnosti $X \rightarrow\rightarrow Y$, važi da $(\exists t' \in r)(t'[XY] = t_1[XY] \wedge t'[R \setminus XY] = t_2[R \setminus XY])$. Prema uslovu (1.6), na osnovu kojeg je formirana torka $I(R)$ i prethodnoj činjenici da za neku torku t' važi: $t' \in r \wedge t'[XY] = t_1[XY] \wedge t'[R \setminus XY] = t_2[R \setminus XY]$, zaključuje se da je ispunjen uslov $g(I) = t' \in r$. Pošto je $g(\tau_{XY}(\mathcal{U}))$ proizvoljna interpretacija, sledi da:

$$(\forall g: Sym \rightarrow Dom)(g(\tau_{XY}(\mathcal{U})) \subseteq r \Rightarrow g(I(R)) \in r),$$

odakle sledi da $r \models <\tau_{XY}(\mathcal{U}), I(R)>$.

($\Leftarrow$) Pretpostavlja se da važi $r \models <\tau_{XY}(\mathcal{U}), I(R)>$. Neka su $t_1, t_2 \in r$ proizvoljno odabrane torke relacije r, takve da je $t_1[X] = t_2[X]$. Neka je $g(\tau_{XY}(\mathcal{U}))$ takva interpretacija

tabloa, za koju važi da je $g(\tau_{XY}(\mathcal{U})) = \{t_1, t_2\}$.*) Na osnovu pretpostavke o važenju T - zavisnosti $<\tau_{XY}(\mathcal{U}), I(R)>$, sledi da je $g(I(R)) \in r$, a na osnovu načina formiranja torke simbola I (uslov (1.6)) sledi da $(\exists t' \in r)(t'[XY] = t_1[XY] \wedge t'[R \setminus XY] = t_2[R \setminus XY])$, pri čemu navedeno svojstvo ispunjava torka $g(I(R))$. Pošto su t_1 i t_2 proizvoljno odabrane torke, zaključuje se da:

$$(\forall t_1, t_2 \in r)(t_1[X] = t_2[X] \Rightarrow (\exists t' \in r)(t'[XY] = t_1[XY] \wedge t'[R \setminus XY] = t_2[R \setminus XY])),$$

što znači da r zadovoljava $X \rightarrow\rightarrow Y$.

Može se primetiti da ukoliko se kao uslov prethodne teoreme uvede jednakost $R = \mathcal{U}$, tada je u pitanju ekvivalentnost potpune T - zavisnosti i potpune višeznačne zavisnosti. Ukoliko, međutim, važi da je $R \subset \mathcal{U}$, onda je reč o ekvivalentnosti ugrađene T - zavisnosti sa ugrađenom višeznačnom zavisnosti. Prethodna teorema se mogla formulisati i tako da relacija r bude definisana nad skupom obeležja R, umesto $\mathcal{U}$, $(r \in SAT(R))$, što bi značilo da r zadovoljava potpunu višeznačnu zavisnost $X \rightarrow\rightarrow Y$.

Primer 1.8. Posmatra se tablo $\tau(\mathcal{U})$ sa slike 1.9. Važe sledeće ekvivalencije generalizovanih T - zavisnosti i višeznačnih zavisnosti:

- $<\tau(\mathcal{U}), (x_1, y_1, z_1, u_2)(ABCD)> \equiv \{AB \rightarrow\rightarrow C \mid D\}$,
- $<\tau(\mathcal{U}), (x_1, y_1, z_1, w_2)(ABCE)> \equiv \{AB \rightarrow\rightarrow C \mid E\}$,
- $<\tau(\mathcal{U}), (x_1, y_1, u_1, w_2)(ABDE)> \equiv \{AB \rightarrow\rightarrow D \mid E\}$ i
- $<\tau(\mathcal{U}), (x_1, y_1, z_1, u_1, w_2)(ABCDE)> \equiv \{AB \rightarrow\rightarrow CD \mid E\}$.

1.3.3. *Generalizovani način prikaza zavisnosti spoja*

Zavisnosti spoja se mogu prikazati putem tipskih T - zavisnosti, čiji prvi tablo sadrži onoliko torki, koliko komponenata sadrži zavisnost spoja, sa istim simbolima nad obeležijma koja pripadaju presecima komponenata date zavisnosti. Drugi tablo sadrži jednu torku koja predstavlja kombinaciju simbola torki prvog tabloa.

Teorema 1.3. Neka je dat skup obeležja $R \subseteq \mathcal{U}$ i zavisnost spoja $\triangleright\triangleleft(X_1,...,X_n)$, pri čemu je $\bigcup_{i=1}^{n} X_i = R$. Neka je data tipska T - zavisnost $<\tau(\mathcal{U}), I(R)>$, takva da je $\tau(\mathcal{U}) = \{I_1,..., I_n\}$, pri čemu važi da je:

$$(1.7) \qquad (\forall i, j \in \{1,...,n\})(I_i[X_i \cap X_j] = I_j[X_i \cap X_j] \wedge (\forall A \in R \setminus (X_i \cap X_j))(I_i[A] \neq I_j[A]))$$

a $I(R)$ torka, takva da važi:

$$(1.8) \qquad (\forall i \in \{1,...,n\})(I[X_i] = I_i[X_i]).$$

Neka je $r \in SAT(\mathcal{U})$. Važi $r \models <\tau(\mathcal{U}), I(R)>$, ako i samo ako r zadovoljava $\triangleright\triangleleft(X_1,...,X_n)$.

*) S obzirom na važenje uslova (1.5), zaključuje se da mora postojati funkcija g, s navedenom osobinom.

Dokaz. ($\Rightarrow$) Pretpostavlja se da r zadovoljava $\triangleright\triangleleft(X_1,...,X_n)$. Neka je $g(\tau(\mathcal{U})) = \{t_1,...,t_n\}$ proizvoljna interpretacija tabloa, takva da je $g(\tau(\mathcal{U})) \subseteq r$. Pošto je, saglasno formuli (1.7), $(\forall i,j \in \{1,...,n\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j])$, na osnovu zavisnosti spoja $\triangleright\triangleleft(X_1,...,X_n)$ se izvodi zaključak da $(\exists t \in r)(\forall i \in \{1,...,n\})(t[X_i] = t_i[X_i])$. Saglasno uslovu (1.8), na osnovu kojeg je formirana torka $l(R)$ i prethodnoj činjenici da za neku torku t važi: $t \in r \wedge (\forall i \in \{1,...,n\})(t[X_i] = t_i[X_i])$, zaključuje se da je ispunjen uslov $g(l) = t \in r$. Pošto je $g(\tau(\mathcal{U}))$ proizvoljna interpretacija, sledi da

$$(\forall g: Sym \rightarrow Dom)(g(\tau(\mathcal{U})) \subseteq r \Rightarrow g(l(R)) \in r),$$

odakle proizilazi činjenica $r \models <\tau(\mathcal{U}), l(R)>$.

($\Leftarrow$) Pretpostavlja se da važi $r \models <\tau(\mathcal{U}), l(R)>$. Neka su $t_1,...,t_n \in r$ proizvoljno odabrane torke relacije r, takve da je $(\forall i,j \in \{1,...,n\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j])$. Neka je $g(\tau(\mathcal{U}))$ takva interpretacija tabloa, za koju važi da je $g(\tau(\mathcal{U})) = \{t_1,...,t_n\}$.*) Na osnovu pretpostavke o važenju T - zavisnosti $<\tau(\mathcal{U}), l(R)>$, sledi da je $g(l) \in r$, a na osnovu načina formiranja torke l sledi da $(\exists t \in r)(\forall i \in \{1,...,n\})(t[X_i] = t_i[X_i])$, pri čemu navedeno svojstvo ispunjava torka $g(l)$. Pošto su $t_1,...,t_n$ proizvoljno odabrane torke, zaključuje se da

$$(\forall t_1,...,t_n \in r)((\forall i,j \in \{1,...,n\})(t_i[X_i \cap X_j] = t_j[X_i \cap X_j]) \Rightarrow (\exists t \in r)(\forall i \in \{1,...,n\})(t[X_i] = t_i[X_i])),$$

odakle proizilazi da r zadovoljava $\triangleright\triangleleft(X_1,...,X_n)$.

Može se primetiti da ukoliko se kao uslov prethodne teoreme uvede jednakost $R = \mathcal{U}$, tada je u pitanju ekvivalentnost potpunih zavisnosti. Ukoliko, međutim, važi da je $R \subset \mathcal{U}$, onda je reč o ekvivalentnosti ugrađene T - zavisnosti i ugrađene zavisnosti spoja. Teorema 1.3 je, takođe, mogla biti formulisana i tako da važi $r \in SAT(R)$ umesto $r \in SAT(\mathcal{U})$ pa bi zavisnost spoja $\triangleright\triangleleft(X_1,...,X_n)$ bila potpuna s obzirom na r.

Primer 1.9. Posmatra se tablo $\tau(\mathcal{U})$ sa slike 1.9. T - zavisnost $<\tau(\mathcal{U}), Q(x_1, y_1, z_1, u_1, w_2)(ABCDE)>$ je ekvivalentna zavisnosti spoja $\triangleright\triangleleft(ABCD, ABE)$.

Primer 1.10. T - zavisnost $<\tau(\mathcal{U}), (a_1, b_1, c_1, d_2)(ABCD)>$, pri čemu je $\tau(\mathcal{U})$ dat na slici 1.11, je ekvivalentna ugrađenoj zavisnosti spoja $\triangleright\triangleleft(ABC, BCD, CDA)$.

$\tau(\mathcal{U})$	A	B	C	D	E
	a_1	b_1	c_1	d_1	e_1
	a_2	b_1	c_1	d_2	e_2
	a_1	b_2	c_1	d_2	e_3

Slika 1.11.

*) S obzirom na važenje uslova (1.7), zaključuje se da mora postojati funkcija g, s navedenom osobinom.

1.3.4. Generalizovani način prikaza zavisnosti sadržavanja

Zavisnosti sadržavanja se mogu iskazivati putem T - zavisnosti, koje u opštem slučaju ne moraju biti tipske, za razliku od funkcionalnih i zavisnosti spoja, koje se iskazuju, redom, putem generalizovanih tipskih E - zavisnosti i tipskih T - zavisnosti. Netipska T - zavisnost, koja reprezentuje zavisnost sadržavanja $[A_1,...,A_n] \subseteq [B_1,...,B_n]$, treba da sadrži tabloe sa po jednom torkom simbola, takve da je torka prvog tabloa nad obeležjima niza $[A_1,...,A_n]$ jednaka torki drugog nad $[B_1,...,B_n]$ i da je drugi tablo definisan nad skupom $\{B_1,...,B_n\}$.

Teorema 1.4. Neka su dati skupovi X, $Y \subseteq \mathcal{U}$, prikazani putem nizova obeležja: $X = A_1,...,A_n$ i $Y = B_1,...,B_n$, za koje važi domenska kompatibilnost

$$(\forall i \in \{1,..., n\})(dom(A_i) \subseteq dom(B_i))$$

i zavisnost sadržavanja $[A_1,...,A_n] \subseteq [B_1,...,B_n]$. Posmatra se T - zavisnost $<\tau(\mathcal{U}), I_2(Y)>$, pri čemu je $\tau(\mathcal{U}) = \{I_1\}$ tipski tablo, takva da važi:

(1.9) $$(\forall i \in \{1,..., n\})(I_1[A_k] = I_2[B_k]).$$

Neka je data relacija $r \in SAT(\mathcal{U})$. Važi $r \models <\tau(\mathcal{U}), I_2(Y)>$, ako i samo ako r zadovoljava $[A_1,...,A_n] \subseteq [B_1,...,B_n]$.

Dokaz. ($\Rightarrow$) Pretpostavlja se da r zadovoljava $[A_1,...,A_n] \subseteq [B_1,...,B_n]$, odnosno da važi uslov $(\forall t_1 \in r)(\exists t_2 \in r)(t_1[X] = t_2[Y])$. Neka je $g(\tau(\mathcal{U}))$ proizvoljna interpretacija tabloa, takva da važi $g(\tau(\mathcal{U})) \subseteq r$, odnosno $g(I_1) \in r$. Treba dokazati da je $g(I_2(Y)) \in \pi_Y(r)$. Neka je $g(I_1) = t_1$ i neka je $t_2 \in r$ torka za koju važi $t_1[X] = t_2[Y]$. Saglasno uslovu (1.9), tj. činjenici $I_1[X] = I_2$, sledi da je $g(I_1)[X]$*) $= g(I_2)$. Pošto je $g(I_1) = t_1$ i $t_1[X] = t_2[Y]$, zaključuje se da važi $g(I_2) = t_2[Y]$, što znači da je $g(I_2) \in \pi_Y(r)$.

($\Leftarrow$) Pretpostavlja se da važi $r \models <\tau(\mathcal{U}), I_2(Y)>$. Neka je $t_1 \in r$ proizvoljno odabrana torka. Treba dokazati da postoji takva torka $t_2 \in r$, za koju je ispunjeno $t_1[X] = t_2[Y]$. Uzima se proizvoljna funkcija g: $Sym \rightarrow Dom$, takva da je $t_1 = g(I_1)$. Pošto važi $r \models <\tau(\mathcal{U}), I_2(Y)>$, zaključuje se da $g(I_2) \in \pi_Y(r)$, odakle sledi da postoji torka $t_2 \in r$, za koju je ispunjeno $t_2[Y] = g(I_2)$. S druge strane, iz činjenice da važi $I_1[X] = I_2$, sledi $g(I_1)[X] = g(I_2)$, a pošto je $t_1 = g(I_1)$ i $t_2[Y] = g(I_2)$, znači da važi $t_1[X] = t_2[Y]$.

Primer 1.11. Netipska T - zavisnost $<\tau(\mathcal{U}), (v_1, z_1)(DE)>$, pri čemu je $\tau(\mathcal{U})$ prikazan na slici 1.12, reprezentuje zavisnost sadržavanja $[BC] \subseteq [DE]$. Posmatraju se relacije $r_1 \in SAT(\mathcal{U})$ i $r_2 \in SAT(\mathcal{U})$, prikazane na slici 1.13.

*) Činjenica $(g(I))[X]$ će, u nastavku teksta, biti skraćeno označavana u notaciji $g(I)[X]$, što treba, u opštem slučaju, razlikovati od notacije za $g(I[X])$.

Može se proveriti da za svaku torku $t \in r_1$ i preslikavanje $g : Sym \to Dom$, koje zadovoljava uslov $g(l_1) = t$, važi da je $g(y_1, z_1) \in \pi_{DE}(r_1)$. Na osnovu teoreme 1.4, sledi da r_1 zadovoljava $[BC] \subseteq [DE]$, tj važi $\pi_{BC}(r) \subseteq \pi_{DE}(r)$.

Posmatra se torka $t_3 \in r_2$. Neka je g preslikavanje za koje važi $g(l_1) = t_3$, odakle sledi da mora biti $g(y_1) = b_3$ i $g(z_1) = c_3$, tj. $g(y_1, z_1)(DE) = (b_3, c_3)$. S druge strane, ne postoji torka $t \in r_2$, za koju bi bio ispunjen uslov $g(y_1, z_1) \in \pi_{DE}(r_2)$, pošto $(b_3, c_3) \notin \pi_{DE}(r_2)$. Saglasno teoremi 1.4, r_2 ne zadovoljava zavisnost sadržavanja $[BC] \subseteq [DE]$. □

$\tau(\mathcal{U})$	A	B	C	D	E
l_1	x_1	y_1	z_1	y_2	z_2

Slika 1.12.

$r_1(\mathcal{U})$	A	B	C	D	E
t_1	a_1	b_1	c_1	b_1	c_1
t_2	a_2	b_2	c_2	b_1	c_1
t_3	a_1	b_2	c_2	b_2	c_2

$r_2(\mathcal{U})$	A	B	C	D	E
t_1	a_1	b_1	c_1	b_1	c_1
t_2	a_2	b_2	c_2	b_1	c_1
t_3	a_1	b_3	c_3	b_2	c_2

Slika 1.13.

Na osnovu teoreme 1.4 se može zaključiti da T - zavisnost $\langle \tau(\mathcal{U}), l_2(Y) \rangle$, koja je ekvivalentna zavisnosti sadržavanja $[X] \subseteq [Y]$, predstavlja tipsku zavisnost, ako i samo ako je $X = Y$.

Zavisnost sadržavanja, definisana nad dve različite šeme relacije kao međurelaciono ograničenje, oblika $R_1[A_1,...,A_n] \subseteq R_2[B_1,...,B_n]$, može se, takođe, prikazati putem T - zavisnosti koja ne mora biti tipska i koja ima osobine, uvedene teoremom 1.4.

Teorema 1.5. Neka su date šeme relacija (R_1, C_1) i (R_2, C_2) $(R_1, R_2 \subseteq \mathcal{U})$ i domenski kompatibilni skupovi $X \subseteq R_1$ i $Y \subseteq R_2$, prikazani putem nizova obeležja: $X = A_1,...,A_n$ i $Y = B_1,...,B_n$. Posmatra se zavisnost sadržavanja $R_1[A_1,...,A_n] \subseteq R_2[B_1,...,B_n]$ i T - zavisnost $\langle \tau(\mathcal{U}), l_2(Y) \rangle$, pri čemu je $\tau(\mathcal{U}) = \{l_1\}$, a $\pi_{R_1R_2}(\tau(\mathcal{U}))$ predstavlja tipski tablo i važi uslov (1.9). Proizvoljne relacije $r_1 \in SAT(R_1)$ i $r_2 \in SAT(R_2)$ zadovoljavaju $R_1[A_1,...,A_n] \subseteq R_2[B_1,...,B_n]$, ako i samo ako važi da:

(1.10) $$(\forall g : Sym \to Dom)(g(l_1)[R_1] \in r_1 \Rightarrow g(l_2) \in \pi_Y(r_2)).$$

Dokaz. Izostavlja se, jer je analogan dokazu teoreme 1.4. □

Primer 1.12. T - zavisnost $\langle \tau(\mathcal{U}), (z_1)(C) \rangle$, pri čemu je $\tau(\mathcal{U})$ prikazan na slici 1.14, reprezentuje zavisnost sadržavanja $R_1[C] \subseteq R_2[C]$, gde je $R_1 = BC$, a $R_2 = CDE$. Simbol "-" označava bilo koje promenljive na datoj poziciji u tablou, pošto one nisu bitne sa

stanovišta zavisnosti $R_1[C] \subseteq R_2[C]$. Treba primetiti da je $<\tau(\mathcal{U}), (z_1)(C)>$, u ovom primeru, tipska T - zavisnost, jer je $X = Y = C$. Posmatraju se relacije $r_1 \in SAT(R_1)$ i $r_2 \in SAT(R_2)$, prikazane na slici 1.15.

Posmatra se torka $t_3 \in r_1$. Neka je g preslikavanje za koje važi $g(l_1) = t_3$, odakle sledi da mora biti $g(z_1) = c_3$, tj. $g(l_2)[C] = (c_3)$. S druge strane, ne postoji torka $t \in r_2$, za koju bi bio ispunjen uslov $g(l_2)[C] = t[C]$, pošto $(c_3) \notin \pi_C(r_2)$. Saglasno teoremi 1.5, r_1 i r_2 ne zadovoljavaju zavisnost sadržavanja $R_1[C] \subseteq R_2[C]$. □

$\tau(\mathcal{U})$	A	B	C	D	E
l_1	-	y_1	z_1	v_1	w_1

Slika 1.14.

$r_1(R_1)$	B	C
t_1	b_1	c_1
t_2	b_2	c_2
t_3	b_1	c_3

$r_2(R_2)$	C	D	E
t_1	c_1	d_1	e_1
t_2	c_2	d_1	e_1
t_3	c_2	d_2	e_2

Slika 1.15.

1.4. *Izvedene F - zavisnosti*

U ovoj tački se uvodi pojam izvedene F - zavisnosti, s jedne strane kao specijalnog slučaja generalizovane E - zavisnosti, a s druge strane kao generalizacije pojma funkcionalne zavisnosti. Za definisanje pojma izvedene F - zavisnosti je bitan pojam lanca, koji će u nastavku biti definisan u formi rekurzije.

Definicija 1.8. Dat je univerzalni skup obeležja $\mathcal{U}$.

1^0 Izraz oblika (A), pri čemu je $A \in \mathcal{U}$ obeležje iz univerzalnog skupa, se naziva ***lanac*** nad skupom $\mathcal{U}$. Umesto izraza (A) se, skraćeno, može koristiti izraz A.

2^0 Neka su $\xi_1,...,\xi_n$, $n \in \mathbf{N}$, konačni skupovi lanaca L^i_j, nad skupom obeležja $\mathcal{U}$:

$$(\forall i \in \{1,...,n\})(\xi_i = \{ L^i_1,...,L^i_{n_i} \}).$$

Izraz oblika $(\xi_1,..,\xi_n)$ predstavlja ***lanac*** nad $\mathcal{U}$.

3^0 ***Lanac*** je sve i samo ono što se dobije primenom, konačno mnogo puta, pravila 1^0 i 2^0.

□

Primer 1.13. Neka je dat univerzalni skup obeležja $\mathcal{U} = ABCDEFGH$. Obeležja A, B, C i D predstavljaju lance i skupove lanaca, istovremeno. Zapisi (A, B) i (C, D) predstavljaju lanace. $\{(A, B), (C, D)\}$ i F su dva skupa lanaca, dok je $(\{(A, B), (C, D)\}, F)$ lanac ova dva skupa. □

Sa $attr(\xi)$ će biti označavan skup svih obeležja, koja učestvuju u skupu lanaca ξ, a sa $attr(L_j^i)$ skup svih obeležja, koja učestvuju u lanacu L_j^i. Sledećom definicijom se formalno uvodi pojam izvedene F - zavisnosti.

Definicija 1.9. Neka je dat univerzalni skup obeležja $\mathcal{U}$, skup $Y \subseteq \mathcal{U}$ i skup lanaca ξ, takav da važi $attr(\xi) \subseteq \mathcal{U}$. Izraz oblika $\xi \rightarrow Y$ se naziva izvedena F - zavisnost nad skupom obeležja $\mathcal{U}$. □

Skup obeležja koja učestvuju u izvedenoj F - zavisnosti $\xi \rightarrow Y$ će biti označen kao $attr(\xi \rightarrow Y)$.

Skup svih mogućih F - zavisnosti nad skupom $\mathcal{U}$ je:

$$\mathcal{IF}(\mathcal{U}) = \{\xi \rightarrow Y \mid attr(\xi) \subseteq \mathcal{U}, Y \subseteq \mathcal{U}\}.$$

$\mathcal{IF}(\mathcal{U})$ predstavlja beskonačan, prebrojiv skup zavisnosti.

U cilju definisanja interpretacije izvedene F - zavisnosti, uvodi se, u rekurzivnoj formi koja prati logiku definicije 1.8, notacija, kojom se iskazuju odnosi između dve torke neke relacije, s obzirom na zadati skup lanaca. Specijalan slučaj nekog "odnosa" dve torke s obzirom na zadati skup lanaca predstavlja poznata relacija jednakosti dve torke nad zadatim skupom obeležja. Neka je data proizvoljna relacija $r \in SAT(\mathcal{U})$ i neka se posmatraju torke $u, v \in r$:

- uX_rv, pri čemu je $X \subseteq \mathcal{U}$, označava činjenicu da je $u[X] = v[X]$,
- uL_rv, pri čemu je $L = (\xi_1, .., \xi_n)$ lanac nad skupom obeležja $\mathcal{U}$, označava činjenicu da:

$$(\exists t_1, ..., t_{n-1} \in r)(u\,\xi_{1_r} t_1 \wedge t_1 \xi_{2_r} t_2 \wedge ... \wedge t_{n-2} \xi_{n-1_r} t_{n-1} \wedge t_{n-1} \xi_{n_r} v),$$

- $u\,\xi_r v$, pri čemu je $\xi = \{L_1, ..., L_k\}$ skup lanaca, označava činjenicu da:

$$(\forall L \in \{L_1, ..., L_k\})(uL_rv). \tag{1.11}$$

Navedenom notacijom su uvedena i pravila za interpretaciju lanca i skupa lanca s obzirom na posmatrane torke u i v.

Ukoliko se relacija r podrazumeva, onda se izrazi uX_rv, uL_rv i $u\,\xi_r v$, skraćuju, redom, kao uXv, uLv i $u\,\xi v$. Izraz (1.11) se može, takođe, napisati u formi: $uL_1v \wedge ... \wedge uL_kv$.

Definicija 1.10. Relacija $r(\mathcal{U})$ zadovoljava F - zavisnost $\xi \rightarrow Y$, što se označava kao $r \models \xi \rightarrow Y$, ako važi:

$$(\forall u, v \in r)(u\,\xi v \Rightarrow u[Y] = v[Y]).$$

Primer 1.14. Neka su dati $\mathcal{U} = ABCDE$ i izvedena F - zavisnost *if*:

$$\{(A, B), (C, D)\} \rightarrow E.$$

Navedena *if* se interpretira na sledeći način: $(\forall u, v \in r)(u\{(A, B), (C, D)\}v \Rightarrow uEv)$, odnosno, nakon razvijanja izraza $u\{(A, B), (C, D)\}v$ i uEv:

$$(\forall u, v \in r)(\exists t_1, t_2 \in r)((u(A) = t_1(A) \wedge t_1(B) = v(B) \wedge u(C) = t_2(C) \wedge t_2(D) = v(D)) \Rightarrow u(E) = v(E)).$$

Na slici 1.16 je prikazana relacija r, takva da je $r \models \{(A, B), (C, D)\} \rightarrow E$. Može se proveriti da jedine dve torke za koje važi pretpostavka F - zavisnosti *if* su t_1 i t_3: $t_1\{(A, B), (C, D)\}_r t_3$, jer važi $t_1(A) = t_2(A) \wedge t_2(B) = t_3(B) \wedge t_1(C) = t_4(C) \wedge t_4(D) = t_3(D)$. U tom slučaju važi i $t_1 E_r t_3$, tj. $t_1(E) = t_3(E)$, odakle sledi i činjenica $r \models if$. □

r	A	B	C	D	E
t_1	a_1	b_1	c_1	d_1	e_1
t_2	a_1	b_2	c_2	d_2	e_2
t_3	a_2	b_2	c_3	d_3	e_1
t_4	a_3	b_3	c_1	d_3	e_2

Slika 1.16.

Primer 1.15. Dat je skup obeležja $R = ABDE$ i $if\ \{(A, B, A), D\} \rightarrow E$. Neka je $r \in SAT(R)$ relacija, prikazana na slici 1.17. Može se utvrditi da $r \not\models \{(A, B, A), D\} \rightarrow E$, pošto važi da je $t_1\{(A, B, A), D\}t_4$, jer je $t_1 A t_2 \wedge t_2 B t_3 \wedge t_3 A t_4 \wedge t_1 D t_4$, ali je $t_1(E) \neq t_4(E)$. □

r	A	B	D	E
t_1	1	1	1	1
t_2	1	2	2	2
t_3	2	2	3	3
t_4	2	3	1	4

Slika 1.17.

Može se videti da je bilo koja funkcionalna zavisnost $X \rightarrow Y$ istovremeno i izvedena F - zavisnost $\{X\} \rightarrow Y$, odnosno $X \rightarrow Y$, saglasno konvenciji da je $\{X\} = X$. Na taj način, važi $\mathcal{F}(\mathcal{U}) \subseteq \mathcal{IF}(\mathcal{U})$.

Moguće je, takođe, dokazati činjenicu da je svaka F - zavisnost istovremeno i odgovarajuća tipska E - zavisnost, što znači da važi implikacija $\mathcal{G}_E(\mathcal{U}) \models \mathcal{IF}(\mathcal{U})$. Na taj način se svaka F - zavisnost može prikazati putem jedne tipske E - zavisnosti.

Primer 1.16. Izvedena F - zavisnost $\{(A, B), (C, D)\} \rightarrow E$ je ekvivalentna tipskoj E - zavisnosti $\langle T(\mathcal{U}), Eq_E(u_1, u_3)\rangle$, pri čemu je tablo $T(\mathcal{U})$ prikazan na slici 1.18. □

$T(\mathcal{U})$	A	B	C	D	E
l_1	x_1	y_1	z_1	w_1	u_1
l_2	x_1	y_2	z_2	w_2	u_2
l_3	x_2	y_2	z_3	w_3	u_3
l_4	x_3	y_3	z_1	w_3	u_4

Slika 1.18.

U nastavku priloga, prezentira se jedan mogući (neprotivurečan) sistem aksioma (pravila izvođenja) za F - zavisnosti, koji će biti označen sa $\mathbb{IF}$, pri čemu problem njegove kompletnosti i neredundantnosti neće biti razmatran. Sistem $\mathbb{IF}$ sadrži sledeće aksiome:

$\mathbb{IF}_1$ (*refleksivnost*) $\quad \xi \to Y,\ W \subseteq Y \vdash \xi \to W,$

$\mathbb{IF}_2$ (*uniranje*) $\quad \xi_1 \to Y,\ \xi_2 \to Z \vdash \xi_1 \cup \xi_2 \to YZ,$

$\mathbb{IF}_3$ (*lančanje*) $\quad \xi_1 \to Y,\ \xi_2 \to Y \vdash \{(\xi_1, \xi_2)\} \to Y$ i

$\mathbb{IF}_4$ (*tranzitivnost*) $\quad \xi \to Y,\ Y \to Z \vdash \xi \to Z.$

Iz ovih pravila, kao specijalan slučaj, slede Armstrongove aksiome za funkcionalne zavisnosti $\mathbb{F}_1$ - $\mathbb{F}_3$.

Primer 1.17. Neka su dati skup obeležja $R = ABCDE$ i skup funkcionalnih zavisnosti $\mathcal{F} = \{A \to C,\ B \to C,\ CD \to E\}$. Iz $A \to C$ i $B \to C$, na osnovu $\mathbb{IF}_3$, sledi $\{(A, B)\} \to C$. Iz $\{(A, B)\} \to C$ i $D \to D$, po pravilu $\mathbb{IF}_2$, izvodi se $\{(A, B), D\} \to CD$, što u kombinaciji sa $CD \to E$, po $\mathbb{IF}_4$, daje $\{(A, B), D\} \to E$. Zaključuje se da važi

$$\mathcal{F} \models \{(A, B), D\} \to E.$$

Na slici 1.19 je prikazana relacija $r \in SAT(R, \mathcal{F})$. Pošto je $\{(A, B), D\} \to E$ logička posledica skupa $\mathcal{F}$, zaključuje se da r zadovoljava navedenu F - zavisnost.

S druge strane, posmatra se skup obeležja $X = ABDE \subset R$ i projekcija skupa funkcionalnih zavisnosti $\mathcal{F}|_X = \{AD \to E,\ BD \to E\}$. U relaciji $s \in SAT(X, \mathcal{F}|_X)$, prikazanoj na slici 1.19, ne važi F - zavisnost $\{(A, B), D\} \to E$, zbog činjenica da je $t_1\{(A, B), D\}\, t_3$ i $t_1[E] \neq t_3[E]$, iako je ispunjeno $attr(\{(A, B), D\} \to E) \subseteq X$. To znači da posmatrana F - zavisnost nije posledica skupa $\mathcal{F}|_X$:

$$\mathcal{F}|_X \not\models \{(A, B), D\} \to E. \ \square$$

r	A	B	C	D	E
t_1	1	1	1	1	1
t_2	1	2	1	2	2
t_3	2	2	1	1	1

s	A	B	D	E
t_1	1	1	1	1
t_2	1	2	2	2
t_3	2	2	1	3

Slika 1.19.

Primer 1.17 ilustruje situaciju u kojoj postoji pojava s nad šemom relacije $(X, \mathcal{F}|_X)$, gde je $X \subset R$, takva da se ne može proširiti do pojave r nad šemom relacije $(R, \mathcal{F})$, pri čemu treba da bude zadovoljen uslov $\pi_X(r) = s$. Uočljivo je, pri tome, da za odgovarajuću F - zavisnost if važi implikacija $\mathcal{F} \models if$, ali je narušena implikacija $\mathcal{F}|_X \models if$, iako je $attr(if) \subseteq X$. Navedena situacija inicira probleme, u literaturi poznate pod nazivima "problem proširenja šeme relacije" i "problem zatvorenosti projekcije familije funkcionalnih zavisnosti" [GZ, Hu, KMo, Lu]. Formulacija i rešenje ovih problema se bazira, između ostalih, i na pojmu izvedenih F - zavisnosti. Detaljnije razmatranje pomenutih problema postoji, delom, u knjizi Principi projektovanja baza podataka, a takođe i u naznačenoj literaturi.

Teorema 1.6. Sistem aksioma $\mathfrak{Ix}$ je neprotivurečan.

Dokaz. ($\mathfrak{Ix}_1$) Korektnost pravila $\mathfrak{Ix}_1$ proizilazi direktno iz definicije 1.10 i činjenica da iz $u[Y] = v[Y]$ i $W \subseteq Y$ sledi jednakost $u[W] = v[W]$.

($\mathfrak{Ix}_2$) Treba pokazati da pod pretpostavkom važenja F - zavisnosti $\xi_1 \rightarrow Y$ i $\xi_2 \rightarrow Z$ u bilo kojoj relaciji $r \in SAT(\mathcal{U})$, važi i F - zavisnost $\xi_1 \cup \xi_2 \rightarrow YZ$. Neka su u, $v \in r$ proizvoljne torke za koje važi $u\ \xi_1 \cup \xi_2\ v$. Na osnovu pravila za interpretaciju skupa lanaca, sledi da važi $u\ \xi_1\ v \wedge u\ \xi_2 v$. Na osnovu $u\ \xi_1 v$ i $\xi_1 \rightarrow Y$ sledi $u\ Y v$. Analogno, važi $u\ Z\ v$. Iz $u\ Y v$ i $u\ Z\ v$ sledi $u\ YZ\ v$, što znači da važi F - zavisnost $\xi_1 \cup \xi_2 \rightarrow YZ$.

($\mathfrak{Ix}_3$) Treba pokazati da pod pretpostavkom važenja F - zavisnosti $\xi_1 \rightarrow Y$ i $\xi_2 \rightarrow Y$ u bilo kojoj relaciji $r \in SAT(\mathcal{U})$, važi i F - zavisnost $\{(\xi_1, \xi_2)\} \rightarrow Y$. Neka su u, $v \in r$ proizvoljne torke za koje važi $u\ \{(\xi_1, \xi_2)\}\ v$. Na osnovu pravila za interpretaciju lanca, sledi da postoji torka $t \in r$, takva da važi $u\ \xi_1 t \wedge t\ \xi_2 v$. Iz $u\ \xi_1 t$ i $\xi_1 \rightarrow Y$ sledi $u\ Y t$, a analogno se zaključuje da važi $t\ Y v$. Iz $u\ Y t$ i $t\ Y v$ sledi $u\ Y v$, što znači da važi F - zavisnost $\{(\xi_1, \xi_2)\} \rightarrow Y$.

($\mathfrak{Ix}_4$) Treba pokazati da pod pretpostavkom važenja F - zavisnosti $\xi \rightarrow Y$ i $Y \rightarrow Z$ u bilo kojoj relaciji $r \in SAT(\mathcal{U})$, važi i F - zavisnost $\xi \rightarrow Z$. Neka su u, $v \in r$ proizvoljne torke za koje važi $u\ \xi\ v$. Iz $u\ \xi\ v$ i $\xi \rightarrow Y$ sledi da važi $u\ Y v$, što u kombinaciji sa $Y \rightarrow Z$ daje zaključak $u\ Z\ v$, odakle proizilazi da je zadovoljena F - zavisnost $\xi \rightarrow Z$. □

U cilju definisanja još jednog pravila za izvođenje F - zavisnosti, uvodi se pojam zamene obeležja skupom lanaca.

Definicija 1.11. Neka su dati skupovi lanaca ξ_i, ξ_j i obeležje $A \in \mathcal{U}$. ***Zamena obeležja A skupom lanaca ξ_i u skupu lanaca ξ_j***, u oznaci $\mu_{\xi_i \rightarrow A}[\xi_j]$, se definiše na sledeći način:

1. Neka je dat lanac $L = (A)$, odnosno $L = A$. Tada je $\mu_{\xi_i \rightarrow A}[L] = (\xi_i)$.
2. Neka je dat lanac $L = (B)$, odnosno $L = B$, pri čemu je $B \neq A$. Tada je $\mu_{\xi_i \rightarrow A}[L] = L$.
3. Neka je dat lanac $L = (\xi_1', ..., \xi_k')$. Tada je $\mu_{\xi_i \rightarrow A}[L] = (\mu_{\xi_i \rightarrow A}[\xi_1'], ..., \mu_{\xi_i \rightarrow A}[\xi_k'])$.
4. Neka je $\xi_j = \{L_1, ..., L_m\}$. Tada je $\mu_{\xi_i \rightarrow A}[\xi_j] = \{\mu_{\xi_i \rightarrow A}[L_1], ..., \mu_{\xi_i \rightarrow A}[L_k]\}$. □

Definicijom 1.11 je, dakle, obezbeđeno da se, po potrebi, sve i samo pojave obeležja A u skupu lanaca ξ_j zamene skupom lanaca ξ_i, pri čemu "struktura" lanca ξ_j ostaje nepromenjena.

Primer 1.18. Neka su dati skupovi lanaca $\xi = \{(A, B), (C, D), A\}$ i $\xi' = \{(E, F)\}$. Zamena obeležja A u ξ, skupom ξ', ima oblik:

$$\mu_{\xi'\to A}[\xi] = \mu_{\xi'\to A}[\{(A, B), (C, D), A\}] = \{\mu_{\xi'\to A}[(A, B)], \mu_{\xi'\to A}[(C, D)], \mu_{\xi'\to A}[A]\} =$$
$$\{(\mu_{\xi'\to A}[A], \mu_{\xi'\to A}[B]), (\mu_{\xi'\to A}[C], \mu_{\xi'\to A}[D]), (\{(E, F)\})\} =$$
$$\{(\{(E, F)\}, B), (C, D), (\{(E, F)\})\}. \square$$

Teorema 1.7. Neka su date F - zavisnosti $\xi_i\to A$ i $\xi_j\to Y$. Važi sledeća implikacija:

(1.12) $$\{\xi_i\to A, \xi_j\to Y\} \models \mu_{\xi_j\to A}[\xi_j]\to Y.$$

Dokaz. Ako $A \notin attr(\xi_j)$, tada je $\mu_{\xi_i\to A}[\xi_j] = \xi_j$ pa je dokaz trivijalan. Neka je $A \in attr(\xi_j)$ i neka je $r \in SAT(\mathcal{U})$ relacija koja zadovoljava skup zavisnosti $\{\xi_i\to A, \xi_j\to Y\}$. Treba dokazati da r zadovoljava $\mu_{\xi_i\to A}[\xi_j]\to Y$. Neka su $u, v \in r$ proizvoljno odabrane torke za koje važi $u\,\mu_{\xi_i\to A}[\xi_j]v$. Za svaku pojavu skupa lanaca ξ_i u okviru ξ_j, u interpretaciji skupa lanca ξ_j će se pojaviti izraz oblika $u'\,\xi_i\,u''$, pri čemu su $u', u'' \in r$. Na osnovu F - zavisnosti $\xi_i\to A$, iz $u'\,\xi_i\,u''$ sledi da važi $u'\,A\,u''$. S obzirom na način formiranja skupa lanaca $\mu_{\xi_i\to A}[\xi_j]$ iz skupa ξ_j, može se zaključiti da važi uslov $u\,\xi_j\,v$. Na osnovu $\xi_j\to Y$ se zaključuje da važi $u\,Y\,v$, što znači da $(\forall u, v \in r)(u\,\mu_{\xi_i\to A}[\xi_j]\,v \Rightarrow u\,Y\,v)$, odnosno da važi $r \models \mu_{\xi_i\to A}[\xi_j]\to Y$. $\square$

Na osnovu teoreme 1.7, proizilazi još jedno pravilo za F - zavisnosti:

$\mathfrak{IF}_5$ (*zamena*) $$\xi_i\to A, \xi_j\to Y \vdash \mu_{\xi_i\to A}[\xi_j]\to Y.$$

Primer 1.19. Neka je dat skup F - zavisnosti $\Sigma = \{\{(B, C)\}\to A, \{A, (C, D)\}\to E\}$. Važi da je $\Sigma \models \{(\{(B, C)\}), (C, D)\}\to E$, odnosno $\Sigma \models \{(B, C), (C, D)\}\to E$, pošto je

$$\{(B, C), (C, D)\} = \{(\{(B, C)\}), (C, D)\} = \mu_{\{(B, C)\}\to A}[\{A, (C, D)\}].$$

Neka je data relacija $r \in SAT(ABCDE)$ i neka su $u, v \in r$ torke za koje važi $u\,\{(\{(B, C)\}), (C, D)\}\,v$. Ovaj izraz se interpretira na sledeći način: $u\,\{(B, C)\}\,v \wedge u(C, D)v$. Iz $\{(B, C)\}\to A$ sledi da je $u\,A\,v \wedge u(C, D)v$, odnosno $u\,\{A, (C, D)\}\,v$, a na osnovu $\{A, (C, D)\}\to E$ proizilazi $u\,E\,v$. $\square$

Izraz oblika $\mu_{\xi_1\to A_1}[\mu_{\xi_2\to A_2}[\ldots\mu_{\xi_n\to A_n}[\xi]\ldots]]$ se, skraćeno, može zapisivati u obliku:

$$\mu_{\xi_1\to A_1,\ \xi_2\to A_2,\ \ldots,\ \xi_n\to A_n}[\xi].$$

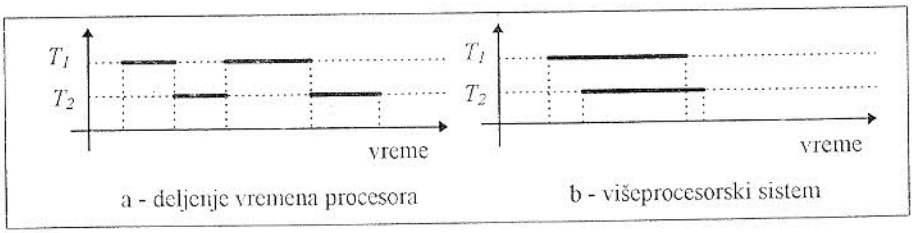
2. Prilog

Višekorisnički režim transakcione obrade podataka

Jedan od bitnih zahteva koji se postavljaju pred savremeni sistem za upravljanje bazama podataka jeste mogućnost višekorisničkog načina rada nad bazom podataka. Sistemi za upravljanje bazama podataka koji obezbeđuju mogućnost višekorisničkog načina rada funkcionišu pod operativnim sistemima koji su po svojoj koncepciji multiprogramski i višekorisnički. Računarski sistem koji opslužuje zahteve sistema za upravljanje bazama podataka (tzv. server), u tom slučaju, može biti organizovan kao:

- jednoprocesorski, ili
- višeprocesorski.

U prvom slučaju, paralelizam obrade transakcija se postiže deljenjem procesorskog vremena od strane više transakcija (slika 2.1.a), dok se u jednom trenutku, zaista, izvršava samo jedna transakcija. U drugom slučaju, postoji mogućnost paralelnog izvršavanja različitih transakcija, od strane različitih procesora (slika 2.1.b).



Slika 2.1.

Koncepcija ažuriranja baze podataka pri transakcionoj obradi podataka u višekorisničkom režimu rada je dosta složenija nego u jednokorisničkom režimu, koji ne dozvoljava paralelizam pri obradi transakcija. Razlog za to leži u činjenici da paralelizam pri obradi transakcija dovodi do sledećih problema:

1. Očuvanje konzistentnosti baze podataka pri paralelnoj pojavi transakcionih zahteva za ažuriranjem istog resursa, ili za korišćenjem resursa koji se ažurira. Pod pojmom *resurs* se podrazumevaju podaci, korišćeni od strane transakcije.
2. Oporavak (dovođenje u konzistentno stanje) baze podataka, nakon pojave greške pri obradi podataka.
3. Mogućnost međusobnog blokiranja transakcija.
4. Potreba opsluživanja svih pokrenutih transakcija u realnom vremenu.

Upitne transakcije ne izazivaju navedene probleme, pošto one ne utiču na promenu pojave baze podataka.

U cilju razrešenja problema 1 - 4, višekorisnički sistem za upravljanje bazama podataka je opremljen s dva podsistema:

- podsistem za upravljanje transakcijama i
- podsistem za oporavak baze podataka.

U nastavku teksta će biti detaljnije diskutovani problemi 1- 4, kao i neki od načina za njihovo rešavanje.

2.1. Očuvanje konzistentnosti baze podataka u transakcionoj obradi

Uzroci koji mogu dovesti do narušavanja konzistentnosti stanja baze podataka pri paralelnoj obradi transakcija su:

- ***gubitak ažuriranja***,
- ***privremenost ažuriranja*** i
- ***narušavanje serijabilnosti***

transakcija. U nastavku će ovi uzroci biti detaljnije diskutovani. U tom cilju, transakcije će biti prikazivane u onom obliku kako ih "vidi" sistem za upravljanje bazama podataka pri izvršavanju, uključujući i tzv. ***vremenski raspored*** - tok njihovog izvršavanja u vremenu. Pretpostavka je da je na raspolaganju jednoprocesorski server, a vremenske jedinice će, radi lakšeg uočavanja problema, biti diskretizovane i prikazane rednim brojevima.

Za ovakav prikaz transakcije će se koristiti, osim naredbe dodele vrednosti, i naredbe *čitaj_resurs*(X) i *piši_resurs*(X). Naredba *čitaj_resurs*(X) označava selekciju podataka (resursa), iz bloka podataka u radnu zonu programa, pri čemu je X oznaka resursa, a naredba *piši_resurs*(X) označava ažuriranje podataka (resursa), pri čemu se podaci iz radne zone programa prenose u blok podataka.

2.1.1. *Problem gubitka ažuriranja i koncept zaključavanja*

Primer 2.1. Na slici 2.2 je, pojednostavljeno, prikazano izvršenje u vremenu (vremenski raspored) dve transakcije T_1 i T_2. T_1 odgovara programu transakcije za rezervaciju mesta u avionu, prikazanom na slici 6.11 (primer 6.24), a T_2 bi mogla odgovarati programu transakcije za istovremeno rezervisanje sedišta na dva povezana leta. U situaciji prikazanoj na slici 2.2, resurs X označava broj slobodnih sedišta za konkretno zadati let i datum.

Vreme	T_1	T_2
1.	*čitaj resurs*(X)	
2.	$X = X - N$	
3.		*čitaj resurs*(X)
4.		$X = X - M$
5.		*piši resurs*(X)
6.	*piši resurs*(X)	
7.		*čitaj resurs*(Y)
8.		$Y = Y - M$
9.		*piši resurs*(Y)

Slika 2.2.

Nakon izvršenja obe transakcije, broj slobodnih sedišta za zadati let i datum bi trebalo da iznosi $X - M - N$, međutim, saglasno situaciji prikazanoj na slici 2.2, taj broj će iznositi $X - N$, čime je narušena konzistentnost baze podataka, što će, u ovom primeru, za posledicu imati da će neki od putnika ostati bez rezervisanog mesta u avionu. □

Narušavanje konzistentnosti podataka, koje je analogno slučaju prikazanom u primeru 2.1, je poznato pod nazivom problem gubitka ažuriranja, a nastaje u situaciji kada izvršenje dve (ili više) transakcija nije sinhronizovano, tako da se paralelno vrši ažuriranje istog resursa.

Napomenimo da je problem gubitka ažuriranja ovde demonstriran na primeru operacije modifikacije podataka, ali se ovaj problem, isto tako, može javiti i pri primeni operacija dodavanja, ili brisanja podataka iz baze.

2.1.1.1. *Tehnika zaključavanja resursa*

Jedno od mogućih i najčešće primenjivanih rešenja problema gubitka ažuriranja je rešenje putem tehnike ***zaključavanja resursa***.

Status resursa je promenljiva, pridružena resursu od strane sistema za upravljanje bazama podataka, čija vrednost služi za sinhronizaciju pristupa resursu od strane različitih transakcija. Ideja se sastoji u tome da dok jedna transakcija pristupa resursu, druge transakcije budu blokirane za pristup istom resursu. Pri tome, vrednost statusa resursa održava

isključivo sistem za upravljanje bazama podataka. Na taj način se smanjuje paralelizam transakcione obrade podataka, ali se zato rešava problem gubitka ažuriranja. Ističu se dva moguća koncepta tehnike zaključavanja:

- koncept binarnog zaključavanja i
- koncept zajedničkog i ekskluzivnog zaključavanja.

Koncept binarnog zaključavanja resursa podrazumeva da status bilo kog resursa $L(X)$ može da sadrži dve moguće vrednosti: $L(X) = 1$ - zaključan i $L(X) = 0$ - slobodan. Naredbom *zaključaj*(X) transakcija izražava zahtev za zaključavanje resursa X, pre prve *čitaj_resurs*(X), ili *piši_resurs*(X) naredbe. Ukoliko je resurs X slobodan, transakcija će ga dobiti na raspolaganje, a $L(X)$ dobija vrednost *1*. Ukoliko je resurs X zauzet, transakcija se prebacuje u red čekanja za X. Naredbom *otključaj*(X) transakcija oslobađa resurs, koji joj je prethodno bio dodeljen. Ukoliko postoji transakcija u redu čekanja za X, resurs X se dodeljuje narednoj transakciji, a ako je red čekanja prazan, tada $L(X)$ dobija vrednost *0*.

Ovakav koncept zaključavanja je suviše restriktivan sa stanovišta paralelizma, pošto samo jedna transakcija u istom trenutku može posedovati zaključani resurs. Zbog toga se u praksi binarno zaključavanje retko primenjuje.

Koncept zajedničkog i ekskluzivnog zaključavanja je, u različitim varijantama, široko primenjivan u praksi. Zajedničkim zaključavanjem se dozvoljava paralelni pristup više upitnih transakcija istom resursu, pri čemu je, za to vreme, onemogućeno ažuriranje resursa. Ekskluzivnim zaključavanjem se, s druge strane, dozvoljava da najviše jedna transakcija pristupi istovremeno resursu, u cilju njegovog ažuriranja. Sistem za upravljanje bazama podataka pridružuje resursu X dve promenljive:

- ***Status*** resursa $L(X)$, sa tri moguće vrednosti: $L(X) = 2$ - ekskluzivno zaključan, $L(X) = 1$ - zajednički zaključan i $EL(X) = 0$ - slobodan.
- ***Broj transakcija*** $NT(X)$, $NT(X) \geq 0$, čija vrednost ukazuje na broj transakcija koje su zajednički zaključale X. Vrednost $NT(X) = 0$ ukazuje na činjenicu da resurs nije zajednički zaključan.

Ekskluzivno zaključavanje resursa X se obezbeđuje pomoću naredbe *zaključaj_eks*(X) (pseudokod algoritma je prikazan na slici 2.3), zajedničko zaključavanje se obezbeđuje naredbom *zaključaj_zaj*(X) (pseudokod algoritma je prikazan na slici 2.4), a otključavanje se realizuje putem naredbe *otključaj*(X) (pseudokod algoritma je prikazan na slici 2.5).

```
PROCES zaključaj_eks
   Ulaz: X                              (* Oznaka resursa *)
POČETAK PROCESA zaključaj_eks
   AKO L(X) = 0 TADA                    (* Resurs je slobodan *)
      POSTAVI L(X)←2          (* Resurs se ekskluzivno zaključava *)
   INAČE                                (* Resurs je ekskluzivno zaključan *)
      POZOVI prenesi_u_red(X)           (* Transakcija se prenosi u red čekanja *)
   KRAJ AKO
KRAJ PROCESA zaključaj_eks
```

Slika 2.3.

```
PROCES zaključaj_zaj
   Ulaz: X                                  (* Oznaka resursa *)
POČETAK PROCESA zaključaj_zaj
   AKO L(X) = 0 TADA                        (* Resurs je slobodan *)
      POSTAVI L(X)←1                        (* Resurs se zajednički zaključava *)
      POSTAVI NT(X)←1
   INAČE
      AKO L(X) = 1 TADA                     (* Resurs je zajednički zaključan *)
         POSTAVI NT(X)←NT(X) + 1
      INAČE                                 (* Resurs je ekskluzivno zaključan *)
         POZOVI prenesi_u_red(X)            (* Transakcija se prenosi u red čekanja *)
      KRAJ AKO
   KRAJ AKO
KRAJ PROCESA zaključaj_zaj
```

Slika 2.4.

Saglasno konceptu zajedničkog i ekskluzivnog zaključavanja, pri formiranju bilo koje transakcije T, moraju biti ispoštovana sledeća pravila:

1. T sadrži naredbu *zaključaj_zaj*(X), ili *zaključaj_eks*(X) pre prve naredbe *čitaj_resurs*(X) - zabrana pristupa resursu koji nije zaključan.
2. T sadrži naredbu *zaključaj_eks*(X) pre prve naredbe *piši_resurs*(X) - zabrana pristupa resursu radi ažuriranja, ako nije ekskluzivno zaključan.
3. T sadrži naredbu *otključaj*(X) nakon poslednje naredbe *piši_resurs*(X), ili *čitaj_resurs*(X) - nakon upotrebe, resurs se mora otključati.
4. T ne sadrži naredbu *zaključaj_zaj*(X), nakon prethodno upotrebljene naredbe *zaključaj_zaj*(X) - zabranjuje se višestruko zajedničko zaključavanje resursa od strane iste transakcije.
5. T ne sadrži naredbu *zaključaj_eks*(X), nakon prethodno upotrebljene naredbe *zaključaj_eks*(X) - zabranjuje se višestruko ekskluzivno zaključavanje resursa od strane iste transakcije.
6. T ne sadrži naredbu *otključaj*(X), ukoliko prethodno nije upotrebljena naredba *zaključaj_zaj*(X), ili *zaključaj_eks*(X) - zabranjuje se otključavanje nezaključanog resursa.

Treba primetiti da transakcija može dati resurs X prvo zajednički da zaključa (*zaključaj_zaj*(X)), a zatim da zaključavanje preinači u ekskluzivno (*zaključaj_eks*(X)). Moguće je, takođe, da se resurs X prvo ekskluzivno zaključa (*zaključaj_eks*(X)), a da se zatim zaključavanje preinači u zajedničko (*zaključaj_zaj*(X)).

Primer 2.2. Na slici 2.6 je prikazan mogući vremenski raspored izvršavanja transakcija T_1 i T_2 iz primera 2.1, pri čemu je pomoću koncepta zajedničkog i ekskluzivnog zaključavanja razrešen problem gubitka ažuriranja. □

```
PROCES otključaj
   Ulaz: X                                  (* Oznaka resursa *)
POČETAK PROCESA otključaj
   POSTAVI Temp(X)←1                        (* Resurs se inicijalno ne oslobađa *)
   AKO L(X) = 2 TADA                        (* Resurs je ekskluzivno zaključan *)
      POSTAVI Temp(X)←0                     (* Resurs se oslobađa *)
   INAČE
      AKO L(X) = 1 TADA                     (* Resurs je zajednički zaključan *)
         POSTAVI NT(X)←NT(X) - 1
         AKO NT(X) = 0 TADA
            POSTAVI Temp(X)←0               (* Resurs se oslobađa *)
         INAČE
         KRAJ AKO
      INAČE                                 (* Greška: Resurs je već slobodan *)
      KRAJ AKO
   KRAJ AKO
   AKO Temp(X) = 0 TADA                     (* Resurs je oslobođen *)
      AKO prazan_red(X) TADA                (* Red čekanja za X je prazan *)
         POSTAVI L(X)←0      (* Resurs se otključava *)
      INAČE                                 (* Red čekanja za X nije prazan *)
         AKO eks_zaključ(X) TADA
         (* Transakcija u redu je inicirala naredbu zaključaj_eks(X) *)
            POSTAVI L(X)←2                  (* Resurs se ekskluzivno zaključava *)
         INAČE
         (* Transakcija u redu je inicirala naredbu zaključaj_zaj(X) *)
            POSTAVI L(X)←1                  (* Resurs se zajednički zaključava *)
            POSTAVI NT(X)←1
         KRAJ AKO
         POZOVI aktiviraj_trans(X)          (* Transakcija iz reda se aktivira *)
      KRAJ AKO
   KRAJ AKO
KRAJ PROCESA otključaj
```

Slika 2.5.

2.1.1.2. *Nivoi zaključavanja resursa*

Saglasno "dimenzijama" resursa koji se zaključava (odnosno, količini podataka), postoje sledeći nivoi zaključavanja:

- nivo vrednsti obeležja,
- nivo torke relacije,
- nivo bloka (stranice) podataka,
- nivo relacije (tabele) i
- nivo baze podataka.

Vreme	T_1	T_2
1.		*zaključaj eks(X)*
2.		*čitaj resurs(X)*
3.		$X = X - M$
4.		*piši resurs(X)*
5.		*otključaj(X)*
6.	*zaključaj eks(X)*	
7.	*čitaj resurs(X)*	
8.		*zaključaj eks(Y)*
9.		*čitaj resurs(Y)*
10.	$X = X - N$	
11.	*piši resurs(X)*	
12.	*otključaj(X)*	
13.		$Y = Y - M$
14.		*piši resurs(Y)*
15.		*otključaj(Y)*

Slika 2.6.

Navedeni redosled odgovara porastu restriktivnosti zaključavanja. Najrestriktivniji je nivo zaključavanja baze podataka i namenjen je isključivo administratoru baze podataka za obavljanje zadataka kao što su arhiviranje, ili dearhiviranje baze podataka. Ostali nivoi zaključavanja su namenjeni za upotrebu u interaktivnoj obradi podataka.

Porastom restriktivnosti zaključavanja se smanjuje nivo paralelizma obrade transakcija, ali se smanjuje i obim posla podsistema za upravljanje transakcijama. Današnji sistemi za upravljanje bazama podataka podržavaju u interaktivnoj obradi zaključavanje na nivou torke, bloka i tabele. Osnovno zaključavanje jedinice resursa se na početku realizuje kao najmanje restriktivno zaključavanje. Pretpostavimo da je to zaključavanje na nivou torke. Kada broj torki jedne relacije koji je u istom trenutku zaključan pređe unapred zadatu vrednost, tada zaključavanje prelazi u žaključavanje na nivou bloka. U trenutku kada broj zaključanih blokova relacije pređe određenu vrednost, zaključavanje resursa se realizuje kao zaključavanje na nivou tabele. Navedena strategija zaključavanja resursa je poznata pod nazivom ***eskalacija zaključavanja***. Eskalacijom zaključavanja se rasterećuje podsistem za upravljanje transakcijama, pošto je utrošak procesorskog vremena pri upravljanju zaključavanjem nižeg nivoa restriktivnosti veći.

2.1.2. *Problem privremenosti ažuriranja*

Primer 2.3. Neka se posmatra, ponovo, vremenski redosled transakcija T_1 i T_2, prikazan na slici 2.6. Neka je u vremenskom trenutku 14. došlo do greške pri izvršavanju naredbe *piši_resurs(Y)*. U tom slučaju, transakcija T_2 mora biti poništena. Ovo poništavanje obavlja sistem za upravljanje bazama podataka, na isti način kao da je u pro-

gramu transakcije zahtevano njeno poništavanje naredbom ***PONIŠTI_TRANS*** (koja je diskutovana u okviru dela 6.4).

Pri poništavanju transakcije T_2 baza podataka treba da ostane u stanju koje je važilo pre pokretanja transakcije T_2. To znači da i transakcija T_1, koja je uspešno završena, mora biti takođe poništena, jer je u T_1 upotrebljena vrednost resursa X, koju je transakcija T_2 već ažurirala. U protivnom, ako se T_1 ne poništi, baza podataka ostaje u nekonzistentnom stanju. □

Situacija, prikazana u primeru 2.3, se može multiplikovati, tako da poništavanje druge transakcije izazove potrebu poništavanja treće, a poništavanje ove, zatim, izazove potrebu poništavanja četvrte transakcije, itd. Problem ***kaskadnog poništavanja transakcija***, u cilju vraćanja baze podataka u poslednje konzistentno stanje, je poznat pod nazivom problem privremenosti ažuriranja.

Problem privremenosti ažuriranja se rešava uvođenjem sledećeg ograničenja: resursi koje transakcija T ekskluzivno zaključava u cilju ažuriranja, mogu postati dostupni ostalim transakcijama, tek pošto transakcija bude potvrđena, ili poništena.

Primer 2.4. Na slici 2.7 je prikazan vremenski redosled transakcija iz primera 2.1, koji ne dovodi do problema privremenosti ažuriranja. Naredbe *otključaj*(X) i *otključaj*(Y) su, sada, poslednje naredbe transakcije, tako da transakcija T_1 može pristupiti resursu X tek po završetku transakcije T_2. Ukoliko se u vremenskom trenutku 8. dogodi greška u izvršavanju naredbe *piši_resurs*(Y), sistem za upravljanje bazama podataka poništava transakciju T_2, nakon čega se otključavaju resursi X i Y, tako da će transakcija T_1 preuzeti podatke za X, koji su bili važeći pre početka transakcije T_2. □

Vreme	T_1	T_2
1.		*zaključaj_eks*(X)
2.		*čitaj_resurs*(X)
3.		$X = X - M$
4.		*piši_resurs*(X)
5.		*zaključaj_eks*(Y)
6.		*čitaj_resurs*(Y)
7.		$Y = Y - M$
8.		*piši_resurs*(Y)
9.		*otključaj*(X)
10.		*otključaj*(Y)
11.	*zaključaj_eks*(X)	
12.	*čitaj_resurs*(X)	
13.	$X = X - N$	
14.	*piši_resurs*(X)	
15.	*otključaj*(X)	

Slika 2.7.

Plan izvršenja transakcije

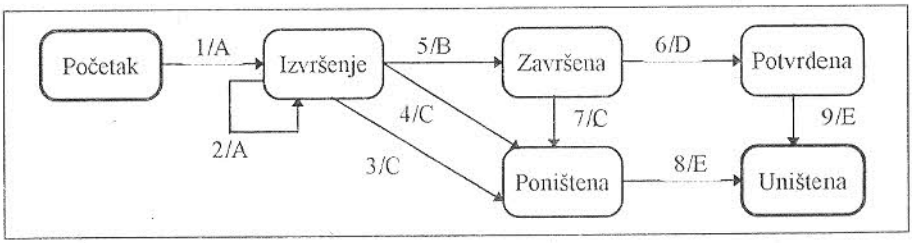
Problem privremenosti ažuriranja inicira potrebu definisanja plana izvršenja transakcije, tj. prikaza mogućih scenarija izvršenja transakcije, s obizrom na strukturu transakcionog programa i eventualne greške, koje se pri izvršenju transakcije mogu dogoditi. Planom izvršenja transakcije se daje formulacija pojma potvrdivanja i poništavanja transakcije.

Plan izvršenja transakcije je prikazan na slici 2.8, u formi konačnog automata, čiji čvorovi predstavljaju moguća stanja transakcije, a grane predstavljaju akcije sistema za upravljanje bazama podataka, koje prevode transakciju iz jednog u drugo stanje. Svaka akcija je inicirana nekim događajem. Mogući događaji, koji iniciraju akcije su:

1. kreirana transakcija,
2. uspešno izvršena naredba transakcionog programa tipa *čitaj_resurs*, ili *piši_resurs*,
3. neuspešno izvršena naredba transakcionog programa tipa *čitaj_resurs*, ili *piši_resurs*, (usled pojave greške tipa: "podaci nisu pronađeni", "deljenje s nulom", itd),
4. izvršena naredba ***PONIŠTI_TRANS***,
5. izvršena naredba ***POTVRDI_TRANS***,
6. uspešno sproveden postupak završetka transakcije,
7. detektovana sistemska greška (npr. međusobna blokada transakcija, koja će biti opisana u tački 2.3),
8. poništena transakcija i oslobođeni svi zaključani resursi i
9. potvrđena transakcija i oslobođeni svi zaključani resursi.

Svaka akcija je propraćena iniciranjem nekog zahteva, pri čemu zahtevi mogu biti:

A. zahtev za izvršenje naredbe transakcionog programa,
B. zahtev za završetak transakcije,
C. zahtev za poništenje transakcije,
D. zahtev za potvrdu transakcije i
E. zahtev za uništenje transakcije.



Slika 2.8.

2.1.3. Problem narušavanja serijabilnosti redosleda i dvofazni protokol zaključavanja

Serijski vremenski redosled transakcija je redosled, u kojem se kompletne transakcije izvršavaju jedna za drugom, bez deljenja procesorskog vremena. Serijski redosled transakcija odgovara pojmu monoprogramskog režima obrade.

Primer 2.5. Redosled transakcija T_1 i T_2, prikazan na slici 2.7, je serijski redosled. □

U režimu paralelnog opsluživanja transakcija, bitno je obezbediti da vremenski redosled transakcija, koji najčešće nije serijski, bude ekvivalentan nekom serijskom redosledu. U protivnom, dolazi do narušavanja konzistentnosti baze podataka.

Definicija 2.1. Vremenski redosledi transakcija S_1 i S_2 su ekvivalentni ako se sastoje iz istog skupa transakcija $T = \{T_1,...,T_n\}$ i važi da:

- Za bilo koju naredbu *čitaj_resurs*(X), transakcije $T_i \in T$, redosleda S_1, koja učitava vrednost X, zapisanu naredbom *piši_resurs*(X), transakcije $T_j \in T$, važi da će i u redosledu S_2, *čitaj_resurs*(X), transakcije $T_i \in T$, učitati vrednost, zapisanu putem *piši_resurs*(X), transakcije $T_j \in T$.
- Za bilo koji resurs X, važi da ako poslednja *piši_resurs*(X) naredba (ukoliko postoji) redosleda S_1 pripada transakciji $T_i \in T$, tada će i u redosledu S_2 poslednja *piši_resurs*(X) naredba biti naredba transakcije $T_i \in T$. □

Definicijom 2.1 se garantuje da će ekvivalentni redosledi transakcija pristupati traženim resursima u istom redosledu, tako da će produkovati isti rezultat ažuriranja baze podataka, za bilo koje početno stanje baze.

Definicija 2.2. Neka je dat redosled S_1, koji nije serijski. Ukoliko postoji serijski redosled S_2, takav da je ekvivalentan redosledu S_1, tada se S_1 naziva ***serijabilni redosled***. □

U cilju očuvanja konzistentnosti baze podataka, zahteva se da svaki vremenski redosled transakcija bude serijabilan.

Primer 2.6. Vremenski redosled transakcija T_1 i T_2 iz primera 2.2 (slika 2.6) je ekvivalentan serijskom redosledu iz primera 2.4 (slika 2.7), tako da predstavlja serijabilan redosled. □

U cilju testiranja serijabilnosti redosleda, definiše se tzv. graf vremenske zavisnosti transakcija (T, ρ). Čvorove tog grafa predstavljaju sve transakcije vremenskog redosleda. Uvode se sledeće pretpostavke:

a) svaki resurs se pre upotrebe zaključava u režimu zajedničkog i ekskluzivnog zaključavanja.
b) pre ažuriranja bilo kog resursa, prvo se obavlja njegovo čitanje, a resurs se ažurira samo na osnovu njegove prethodno pročitane vrednosti.

U tom slučaju, skup grana grafa zavisnosti transakcija ρ se formira prema sledećim pravilima:

$(T_i, T_j) \in \rho$, ako postoji resurs X, takav da važi:

- $Vreme(T_i, zaključaj\_zaj(X)) \leq Vreme(T_j, zaključaj\_eks(X)) \vee$
- $Vreme(T_i, zaključaj\_eks(X)) \leq Vreme(T_j, zaključaj\_eks(X)) \vee$
- $Vreme(T_i, zaključaj\_eks(X)) \leq Vreme(T_j, zaključaj\_zaj(X))$,

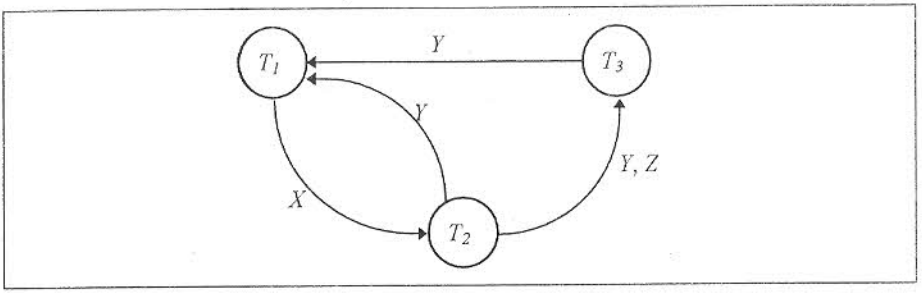
pri čemu $Vreme(T_i, zaključaj\_zaj(X))$, tj. $Vreme(T_i, zaključaj\_eks(X))$ označava vremenski trenutak u kojem transakcija T_i zajednički, odnosno ekskluzivno zaključava resurs X.

Može se dokazati da je vremenski redosled serijabilan, ako i samo ako je graf zavisnosti transakcija (T, ρ) aciklički. Ekvivalentni serijski redosled se, u tom slučaju, dobija topološkim uređivanjem u niz, čvorova grafa (T, ρ).

Primer 2.7. Na slici 2.9 je prikazan jedan neserijabilni vremenski redosled transakcija, dok je na slici 2.10 prikazan graf vremenske zavisnosti transakcija tog redosleda, pri čemu je uz svaku granu prikazana i oznaka resursa, čije zaključavanje uzrokuje zavisnost transakcija. □

Vreme	T_1	T_2	T_3
1.		*zaključaj_zaj(Z)*	
2.		*čitaj_resurs(Z)*	
3.		*zaključaj_eks(Y)*	
4.		*čitaj_resurs(Y)*	
5.		*piši_resurs(Y)*	
6.		*otključaj(Z)*	
7.		*otključaj(Y)*	
8.			*zaključaj_eks(Y)*
9.			*čitaj_resurs(Y)*
10.			*zaključaj_eks(Z)*
11.			*čitaj_resurs(Z)*
12.	*zaključaj_eks(X)*		
13.	*čitaj_resurs(X)*		
14.	*piši_resurs(X)*		
15.	*otključaj(X)*		
16.			*piši_resurs(Y)*
17.			*piši_resurs(Z)*
18.			*otključaj(Y)*
19.			*otključaj(Z)*
20.		*zaključaj_zaj(X)*	
21.		*čitaj_resurs(X)*	
22.		*otključaj(X)*	
23.	*zaključaj_zaj(Y)*		
24.	*čitaj_resurs(Y)*		
25.	*otključaj(Y)*		

Slika 2.9.



Slika 2.10.

U slučaju da pretpostavka b) ne važi (što je u praksi realan slučaj), tada je način formiranja grafa zavisnosti transakcija komplikovaniji, jer se u obzir uzimaju zavisnosti skupova svih ažuriranih resursa sa skupovima svih pročitanih resursa transakcija.

Dvofazni protokol zaključavanja

Dinamičko testiranje serijabilnosti redosleda je u praksi teže primenljivo. Za to postoje dva razloga:

- Ukoliko se, pri testiranju, ustanovi da redosled nije serijabilan, to bi značilo da sve transakije koje su u redosledu učestvovale moraju biti poništene, bez obzira na to što se čak neke od njih nalaze u stanju završenosti. U slučaju egzistencije većeg broja transakcija redosleda (što je u praksi uobičajeno), ovaj zahtev postaje veoma strog.
- Vremenski interval nailaska zahteva za opsluživanje transakcije je slučajna veličina, tako da se pojavljuje problem određivanja početka i kraja jednog vremenskog redosleda, što značajno komplikuje implementaciju algoritma serijabilnosti redosleda.

Zbog toga se, umesto dinamičkog testiranja serijabilnosti redosleda, pribegava uvođenju pravila za strukturiranje transakcija, koje će obezbediti da redosled bude sigurno serijabilan. Jedno rešenje takvog tipa predstavlja dvofazni protokol zaključavanja transakcija.

Dvofazni protokol zaključavanja transakcija je pravilo po kojem sve naredbe zaključavanja resursa prethode prvoj naredbi otključavanja resursa. Na taj način, transakcija prolazi kroz dve faze izvršenja:

I ***Faza zaključavanja resursa*** (faza ***širenja transakcije***), u kojoj se ni jedan resurs ne sme otključati i

II ***Faza otključavanja resursa*** (faza ***skupljanja transakcije***), u kojoj se ni jedan resurs ne sme zaključati.

U cilju izbegavanja pojave problema privremenosti ažuriranja, faza otključavanja resursa se obavlja u toku procesa potvrđivanja, ili poništavanja transakcije, tako da resursi postaju dostupni ostalim transakcijama tek pošto tekuća transakcija bude poništena, ili potvrđena.

Dvofaznim protokolom zaključavanja se dozvoljava tzv. "***nadogradnja zaključavanja***". To je situacija u kojoj resurs može biti prvo zajednički zaključan, a tek kasnije se zaključavanje preinači u ekskluzivno. Prebacivanjem faze otključavanja resursa

u proces potvrđivanja ili poništavanja transakcije, zabranjuje se mogućnost da se ekskluzivno zaključavanje resursa preinači u zajedničko zaključavanje.

Može se dokazati da dvofazni protokol zaključavanja garantuje serijabilnost bilo kog vremenskog redosleda transakcija. Cena garantovanja serijabilnosti je, međutim, umanjenje paralelizma pri izvršavanju transakcija.

Primer 2.8. Na slici 2.11 je prikazan serijabilni vremenski redosled transakcija T_1, T_2 i T_3, iz primera 2.7. Sve transakcije su formirane saglasno dvofaznom protokolu zaključavanja. Graf vremenske zavisnosti transakcija (T, ρ), u ovom slučaju, ima oblik: $T = \{T_1, T_2, T_3\}$ i $\rho = \{(T_3, T_2), (T_3, T_1), (T_1, T_2)\}$. (T, ρ) je, dakle, aciklički graf. Odgovarajući serijski redosled je: T_3, T_1, T_2. □

Vreme	T_1	T_2	T_3
1.			*zaključaj eks(Y)*
2.			*čitaj resurs(Y)*
3.			*zaključaj eks(Z)*
4.			*čitaj resurs(Z)*
5.	*zaključaj eks(X)*		
6.	*čitaj resurs(X)*		
7.	*piši resurs(X)*		
8.			*piši resurs(Y)*
9.			*piši resurs(Z)*
10.			*otključaj(Y)*
11.			*otključaj(Z)*
12.		*zaključaj zaj(Z)*	
13.		*čitaj resurs(Z)*	
14.	*zaključaj zaj(Y)*		
15.	*čitaj resurs(Y)*		
16.	*otključaj(X)*		
17.	*otključaj(Y)*		
18.		*zaključaj eks(Y)*	
19.		*čitaj resurs(Y)*	
20.		*piši resurs(Y)*	
21.		*zaključaj zaj(X)*	
22.		*čitaj resurs(X)*	
23.		*otključaj(Z)*	
24.		*otključaj(Y)*	
25.		*otključaj(X)*	

Slika 2.11.

2.2. Oporavak baze podataka

Čitav niz različitih događaja i grešaka može izazvati potrebu za primenu nekog od postupaka dovođenja baze podataka u konzistentno stanje. Postupci održavanja baze podataka u konzistentnom stanju i dovođenja baze podataka u konzistentno stanje se, kraće, nazivaju postupci *oporavaka* baze podataka. Uzroci, koji traže pokretanje postupaka oporavka baze podataka, se mogu klasifikovati u sledeće tipove:

1. ***Sistemska greška*** ("pad sistema") - softverska ili hardverska greška, koja uzrokuje pad sistema i gubitak sadržaja operativne memorije.
2. ***Transakcijska greška*** - greška u izvršenju transakcije (deljenje s nulom, prekoračenje dozvoljenog opsega celobrojne vrednosti, logička greška u programu transakcije, nasilni prekid transakcije od strane operatera, itd).
3. ***Poništavanje transakcije*** - programirani zahtev za poništavanje transakcije (naredba *PONIŠTI_TRANS*, obrada "izuzetaka", tipa "podaci ne postoje", "negativan saldo na računu", itd).
4. ***Konkurencijska greška*** - greška koju je izazvala nelegalna situacija u višekorisničkom režimu obrade transakcija (npr. međusobno blokiranje transakcija, koje će biti razmatrano u tački 2.3, ili potreba kaskadnog poništavanja transakcija, pod pretpostavkom da problem privremenosti ažuriranja nije preventivno rešen).
5. ***Greška na disku*** - greška pri upisu, ili čitanju podataka s diska, usled neispravnosti samog uređaja.
6. ***Katastrofalna fizička greška*** - delimično, ili kompletno uništenje podataka na disku usled organizacione greške u radu osoblja, elementarne nepogode, problema s napajanjem, itd.

U slučaju da dođe do greške tipa 1- 4, sistem za upravljanje bazama podataka održava dovoljan skup podataka, na osnovu kojeg je u mogućnosti da izvrši oporavak baze podataka u poslednje konzistentno stanje i ponovo pokrene sve prekinute (nepotvrđene) transakcije. U slučaju pojave greške tipa 5, ili 6, sistem za upravljanje često nije u mogućnosti da izvrši oporavak baze podataka u poslednje konzistentno stanje i pokrene sve prekinute transakcije. Jedna od mogućnosti u toj situaciji je da se, nakon ispravljanja greške, izvrši restauracija poslednje arhivirane kopije baze podataka. Promene baze podataka, nastale nakon trenutka kada je primenjena arhivirana kopija napravljena se, pri tome, neće izgubiti, ukoliko na raspolaganju postoji neoštećen sistemski dnevnik, koji sadrži podatke o svim promenama stanja baze podataka, nastalim nakon trenutka poslednjeg arhiviranja. O sistemskom dnevniku će u daljem tekstu biti više reči.

U nastavku teksta će biti izložena koncepcija obezbeđenja preduslova za oporavak, kao i samog oporavka baze podataka, u slučaju nastanka greške tipa 1- 4. Pri tome će biti dat i detaljniji prikaz postupaka poništavanja i potvrđivanja transakcije, koji su uvedeni u delu teksta pod naslovom "Plan izvršenja transakcije".

2.2.1. *Sistemski dnevnik*

Sistem za upravljanje bazama podataka kreira i tokom rada održava posebnu datoteku, pod nazivom ***sistemski dnevnik*** (ili ***žurnal - datoteka***). Na osnovu sistemskog dnevnika (i, eventualno, arhivirane kopije baze podataka) moguće je izvršiti oporavak baze podataka u poslednje, moguće, konzistentno stanje. Zbog svog značaja, sistemski dnevnik treba, kao i bazu podataka, periodično arhivirati.

Neka je T identifikaciona oznaka transakcije, a X oznaka resursa. Sistemski dnevnik se sastoji iz slogova, pri čemu svaki slog može sadržati jedan od sledećih tipova podataka:

- [*početak_transakcije*, T],
- [*čitaj_resurs*, T, X],
- [*piši_resurs*, T, X, *s_vred*, *n_vred*],
- [*potvrda_transakcije*, T] i
- [*kontrolna_tačka*].

Slog [*početak_transakcije*, T] se zapisuje u sistemski dnevnik prilikom kreiranja transakcije.

Slog [*čitaj_resurs*, T, X] se zapisuje u sistemski dnevnik neposredno nakon izvršenja naredbe *čitaj_resurs*(X), transakcije T. Postupci za oporavak i ažuriranje baze podataka, zasnovani na tehnici odloženog ažuriranja (videti tačku 2.2.3) ne zahtevaju formiranje ove vrste sloga.

Slog [*piši_resurs*, T, X, *s_vred*, *n_vred*] se zapisuje u sistemski dnevnik neposredno nakon izvršenja naredbe *piši_resurs*(X), transakcije T. Pri tome, *s_vred* predstavlja staru vrednost resursa X, a *n_vred* predstavlja ažuriranu vrednost resursa X. Postupci za oporavak i ažuriranje baze podataka, zasnovani na tehnici odloženog ažuriranja (videti tačku 2.2.3) ne zahtevaju zapis polja *s_vred*.

Slog [*potvrda_transakcije*, T] se zapisuje u sistemski dnevnik nakon sprovedenog postupka potvrđivanja transakcije i oslobađanja zaključanih resursa. Transakcija T, za koju postoji u sistemskom dnevniku slog [*potvrda_transakcije*, T] predstavlja potvrđenu transakciju. Potvrđivanje transakcije koja se nalazi u stanju završenosti (slika 2.8) započinje sprovođenjem testa uspešnosti transakcije. Ukoliko test uspešnosti detektuje konkurencijsku grešku, potvrđivanje transakcije se prekida i prelazi se na postupak poništavanja transakcije. Ukoliko, pak, test uspešnosti ne detektuje nikakvu konkurencijsku grešku, potvrđivanje transakcije se nastavlja izvođenjem sledećih koraka, u navedenom redosledu:

- Sadržaji svih blokova podataka sistemskog dnevnika koji se nalaze u operativnoj memoriji se prenose na disk. Ovaj postupak se naziva ***bezuslovno zapisivanje blokova na disk***.
- Oslobađaju se svi zaključani resursi transakcije.
- U sistemski dnevnik se upisuje slog [*potvrda_transakcije*, T], čime efekti transakcije postaju vidljivi ostalim transakcijama.

Postupak poništavanja transakcije koji je uvek posledica transakcijske ili konkurencijske greške, ne zahteva nikakvo ažuriranje sistemskog dnevnika, tako da ne postoji poseban tip sloga kojim bi se u sistemskom dnevniku označio trenutak poništavanja transakcije.

Slog [*kontrolna_tačka*] se zapisuje u sistemski dnevnik nakon sprovedenog postupka ažuriranja baze podataka na osnovu podataka iz sistemskog dnevnika. Postupak ažuriranja baze podataka i upisivanje u sistemski dnevnik sloga [*kontrolna_tačka*] se sprovodi periodično, u tačno utvrđenim vremenskim intervalima, ili nakon dostizanja definisanog broja potvrđivanja završenih transakcija, od trenutka kada je upisana poslednja [*kontrolna_tačka*]. Sam postupak podrazumeva izvršenje sledećih koraka, u navedenom redosledu:

- Privremeno suspendovanje svih tekućih transakcija.
- Izdvajanje iz sadržaja sistemskog dnevnika podataka o svim potvrđenim transakcijama, nakon poslednje pojave sloga [*kontrolna_tačka*].
- Ažuriranje baze podataka, saglasno hronološki izdvojenim podacima iz sistemskog dnevnika o potvrđenim transakcijama *).
- Bezuslovno zapisivanje sadržaja svih blokova podataka baze podataka na disk.
- Upisivanje u sistemski dnevnik sloga [*kontrolna_tačka*].
- Reaktiviranje svih privremeno suspendovanih transakcija.

2.2.2. *Tehnike ažuriranja i oporavka baze podataka*

Ažuriranje baze podataka se može realizovati putem tehnike:

- ***odloženog ažuriranja***, kod kojeg se baza podataka ažurira samo na osnovu sadržaja sistemskog dnevnika, tek pošto transakcija bude potvrđena putem sloga [*potvrda_transakcije*, *T*], ili
- ***trenutnog ažuriranja***, kod kojeg se baza podataka ažurira tokom izvršenja transakcija, bez obzira da li su potvrđene, ili ne.

Tehnika odloženog ažuriranja ne zahteva nikakav poseban postupak poništavanja transakcije. Poništavanje transakcije se svodi samo na oslobađanje zaključanih resursa. S druge strane, tehnika trenutnog ažuriranja zahteva da se efekti poništenih transakcija, u reverznom redosledu, uklone iz baze podataka.

Opisani postupak ažuriranja sistemskog dnevnika, kao i postupci odloženog, ili trenutnog ažuriranja baze podataka obezbeđuju da u slučaju "pada sistema", efekti transakcija, za koje postoji u sistemskom dnevniku slog [*potvrda_transakcije*, *T*] pre pojave poslednjeg sloga [*kontrolna_tačka*], ostaju zabeleženi u bazi podataka. Efekti transakcija, koje su potvrđene nakon poslednje pojave sloga [*kontrolna_tačka*] u sistemskom dnevniku, u slučaju pada sistema, neće biti zabeleženi u bazi podataka. To znači da, pri oporavku baze podataka, ove transakcije ulaze u postupak ažuriranja baze podataka na osnovu sadržaja sistemskog dnevnika.

Budući da za poništene i nepotvrđene transakcije ne postoji u sistemskom dnevniku slog [*potvrda_transakcije*, *T*], pri postupku ažuriranja baze podataka, ove transakcije neće biti uzete u obzir, ako je primenjena tehnika odloženog ažuriranja baze. Ako je,

*) Ovaj korak se sprovodi samo u slučaju primene tehnike odloženog ažuriranja baze podataka (videti tačku 2.2.2).

međutim, primenjena tehnika trenutnog ažuriranja baze, efekti ovih transakcija moraju, u reverznom redosledu, biti uklonjeni iz baze podataka.

Sve transakcije, za koje ne postoji slog [*potvrda_transakcije*, *T*] u sistemskom dnevniku, u okviru sprovođenja postupka oporavka nakon sistemske greške, biće ponovo pokrenute.

U nastavku će biti dat nešto detaljniji opis tehnike odloženog ažuriranja baze podataka.

2.2.3. *Tehnika odloženog ažuriranja i dvofazno potvrđivanje transakcije*

Ideja tehnike odloženog ažuriranja se sastoji u tome da se baza podataka ažurira, na osnovu transakcije *T*, tek pošto se u sistemski dnevnik upiše slog [*potvrda_transakcije*, *T*]. Do tada, sve promene koje je transakcija izazvala, ostaju zabeležene u radnoj zoni transakcije i prenose se u sistemski dnevnik.

Tehnika odloženog ažuriranja se realizuje putem tzv. ***dvofaznog protokola potvrđivanja transakcije*** *). Protokol se sastoji iz sledećih pravila, koja, istovremeno, predstavljaju i faze ažuriranja baze podataka:

I Transakcija ne sme da izvrši ažuriranje baze podataka dok ne bude potvrđena.

II Transakcija ne sme da bude potvrđena dok sva ažuriranja resursa ne budu registrovana u sistemski dnevnik i dok se ne izvrši bezuslovan zapis blokova sistemskog dnevnika na disk.

Na kraju, treba zaključiti da primena:

- dvofaznog protokola zaključavanja resursa transakcije,
- pravila da se zaključani resursi oslobađaju u postupku poništavanja, ili potvrđivanja transakcije i
- dvofaznog protokola potvrđivanja transakcije i upotreba sistemskog dnevnika,

rešavaju probleme gubitka ažuriranja, privremenosti ažuriranja (tj. kaskadnog poništavanja) i narušavanja serijabilnosti redosleda. Pri tome, obezbeđeni su potrebni mehanizmi za oporavak baze podataka u slučaju pojave sistemske, transakcijske, ili konkurencijske greške, kao i u slučaju poništavanja transakcije.

2.3. *Međusobno blokiranje transakcija*

Međusobna blokada transakcija, ili ***zastoj***, ili ***uzajamno zaključavanje***, je situacija u kojoj dve ili više transakcija čekaju na oslobađanje resursa, koje se sigurno neće desiti.

---

*) Pojam dvofaznog protokola potvrđivanja transakcije treba razlikovati od pojma dvofaznog protokola zaključavanja resursa transakcije.

Primer 2.9. Na slici 2.12 je prikazana situacija međusobne blokade transakcija T_1 i T_2. Transakcija T_1 čeka na oslobađanje resursa X, koji je zaključan od strane transakcije T_2, dok transkacija T_2 čeka na oslobađanje resursa Y, koji je zaključan od strane transakcije T_1. □

Vreme	T_1	T_2	
1.	*zaključaj zaj(Y)*		
2.	*čitaj resurs(Y)*		
3.		*zaključaj zaj(X)*	
4.		*čitaj resurs(X)*	
5.	*zaključaj eks(X)*		
6.		*zaključaj_eks(Y)*	← *Zastoj*

Slika 2.12.

Izbegavanje zastoja se može postići putem protokola o redosledu zaključavanja resursa.

Prema jednom takvom protokolu, uvodi se redosled zaključavnja resursa. Svi resursi (relacije baze podataka) dobijaju jednistveni prioritet zaključavanja, tako da svaka transakcija, poštujući protokol o prioritetu, zaključava tražene resurse po opadajućim vrednostima dodeljenih prioriteta.

Drugi mogući protokol za prevenciju zastoja se svodi na to da se svi traženi resursi transakcije zaključavaju odjednom. Ukoliko je makar jedan od traženih resursa zauzet, ni jedan resurs se ne zaključava, a transakcija, nakon izvesnog vremena, ponovo pokušava da sprovede zaključavanje resursa.

Sledeće rešenje za eliminaciju problema zastoja jeste dinamičko testiranje pojave zastoja. Umesto uvođenja protokola o redosledu zaključavanja, koji još više smanjuje mogući paralelizam transakcija, dozvoljava se da transakcije neometano zaključavaju tražene resurse, u bilo kakvom poretku. U određenim vremenskim intervalima, sistem za upravljaje bazama podataka pokreće program za detekciju i razrešenje zastoja.

Algoritam detekcije zastoja se svodi na pronalaženje ciklusa u tzv. ***grafu zavisnosti transakcija po zaključavanju*** $(\boldsymbol{T}, \psi)$. Čvorovi grafa zavisnosti $(\boldsymbol{T}, \psi)$ su transakcije, a grane se definišu na sledeći način: $(T_i, T_j) \in \psi$, ako transakcija T_j, čeka na zaključavanje resursa X, koji je već zaključan od strane transakcije T_i. Ukoliko $(\boldsymbol{T}, \psi)$ poseduje makar jedan ciklus, došlo je do zastoja.

Razrešenje zastoja se sastoji u tome da se jedna od transakcija u ciklusu nasilno poništi, čime se izaziva konkurencijska greška nad datom transakcijom.

Primer 2.10. Graf zavisnosti transakcija po zaključavanju za situaciju, prikazanu na slici 2.12, ima oblik $(\{T_1, T_2\}, \{(T_1, T_2), (T_2, T_1)\})$. Zastoj se, u ovom slučaju, rešava poništavanjem transakcije T_1, ili transakcije T_2. □

Dinamičko testiranje zastoja predstavlja dobro rešenje u slučaju da se zastoji neće u praksi često pojavljivati. To se može očekivati u situacijama kada transakcije

zaključavaju resurse za kratko vreme, kada zaključavaju relativno mali broj resursa i kada je nivo zaključavanja nizak (nivo zaključavanja torke, ili stranice).

2.4. Opsluživanje transakcija u realnom vremenu

Intezitet kreiranja transakcija u višekorisničkom režimu rada može biti dosta visok. Moguća situacija u slučaju visokog inteziteta zahteva za opsluživanje transakcija jeste da neka transakcija ne može da dođe na red za zaključavanje traženog resursa. Ova situacija se naziva "***izgladnjavanje transakcije***". Uzrok nastajanja izgladnjavanja može biti u tome da transakcija ima nizak prioritet opsluživanja, a intezitet nailaska transakcija visokog nivoa prioriteta, koje traže zaključavanje istog resursa, je veliki.

Problem izgladnjavanja transakcija se rešava na nivou protokola za opsluživanje transakcija. Najjednostavnije rešenje je ono po kojem transakcije dobijaju traženi resurs po prioritetu u kojem su zahtevale zaključavanje resursa ("prvi tražio - prvi opslužen"). Ukoliko postoji drugačiji protokol o dodeljivanju prioriteta opsluživanja, tada se mora predvideti da prioritet opsluživanja tokom čekanja transakcije na resurs raste. Time se obezbeđuje da transakcija sigurno bude opslužena u realnom vremenu.

Literatura

[A] Armstrong W. W., *" Dependecy Structures of Data base Relationships"* Proc. IFIP 74, North Holland, Amsterdam, 1974, pp. 580 - 583.

[ABU] Aho A. V., Beeri C., Ullman J. D., *"The Theory of Joins in Relational Databases"*, ACM TODS, Vol. 4, No. 3, Sept. 1979, pp. 297 - 314.

[Ansi] *"Database Language SQL, American National standard X3.135-1986"*, USA, 1986.

[AP] Atzeni P., Parker D. S., *"Properties of Acyclic Database Schemata"*, Advances in Database Theory II, Gallaire H., Minker J., Nicolas M., eds, Plenum Press, New York, 1983, pp. 27 -52.

[B] Beeri C., *"On the Membership Problem for Functional and Multivalued Dependencies"*, ACM TODS, Vol. 5, No. 3, Sept. 1980, pp. 241 - 259.

[BB] Beeri C., Bernstein P.A., *"Computational Problems Related to the Design of Normal Form Relational Schemas"*, ACM Transactions on Database Systems, Vol.4, No.1, March, 1979, pp. 30-59.

[BFMY] Beeri C., Fagin R., Maier D., Yannakakis M., *"On the Desirability of Acyclic Database Schemes"* Jour. of ACM, Vol. 30, No. 3, July 1983, pp. 479 513.

[BV] Beeri C., Vardi M., *"A Proof Procedure for Data Dependencies"*, Jour. of ACM, Vol. 31, No. 4, Oct. 1984, pp. 718 - 741.

[Che] Chen P. P., *"The Entity - Relationship Model - Toward a Unified View of Data"*, ACM TODS, Vol.1, No.1, March, 1976, pp. 9 -36.

[C] Codd E. F., *The Relational Model for Database Management Version 2*, Addison-Wesley-Publishing-Company, USA, 1990.

[C1] Codd E. F., *"Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks"*, Comm. ACM, Vol.13, No. 6, Jun 1970, pp. 377 - 387.

[C2] Codd E. F., *"Extending the Database Relational Model to Capture More Meaning"*, ACMTODS, Vol. 4, No. 4, Dec. 1979, pp. 397 - 434.

[CdS] Casanova M. A., Amaral de Sa J. E., *"Mapping Uninterpreted Schemes into Entity - Relationship Diagrams: Two Applications to Conceptual Schema Design"*, IBM Jour. of Res. & Dev., Vol. 28, No. 1, Jan. 1984, pp. 82 - 94.

[CH] Cheng J. -b., Hurson A. R. *"Effective Clustering of Complex Objects in Object - Oriented Databases"*, Proc. ACM SIGMOD Int'l Conf. Management of Data, ACM, New York 1991, pp. 22 - 31.

[CS] Cvetković D., Simić S., *Diskretna matematika - matematika za kompjuterske nauke*, Naučna knjiga, Beograd, 1990.

[Da] Date C. J., The Relational Institute and The Codd and Date Consulting Group, *A Guide to the SQL Standard*, Addison-Wesley Publishing Company, 1987.

[DM] Diedrich I., Milton J., *"New Methods and Fast Algorithms for Database Normalization"*, ACM Transactions on Database Systems, Vol 13, No. 3, Sept. 1988, pp. 339-365.

[EN] Elmasri R., Navathe S., *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, Inc, Redwood City, USA, 1989.

[Fa1] Fagin R., *"Multivalued Dependencies and a New Normal Form for Relational Databases"*, ACM TODS, Vol. 2, No. 3, Sept. 1977, pp. 262 - 278.

[Fa2] Fagin R., *"Horn Clauses and Database Dependencies"*, Journal of the ACM, Vol. 29, No. 4, October 1982, pp. 952-985.

[Fa3] Fagin R., *"Degrees of Acyclicity for Hypergraphs and Relational Database Schemes"*, Jour. of ACM, Vol. 30, No. 3, July 1983, pp. 514 - 550.

[FMU] Fagin R., Mendelzon O. A., Ullman D. J., *"A Simplified Universal Relation Assumption and Its Properities"*, ACM TODS, Vol. 7, No. 3, September 1982, pp. 343-360.

[G] Graham M. H., *"On the Universal Relation"*, Computer Systems Researc Group Report, University of Toronto, Dec. 1979.

[Ga] Galil Z., *"An Almost Linear Algorithm for Computing a Dependency Basis in Relational Database"*, Jour. of ACM, Vol. 29, No. 1, Jan. 1982, pp. 96 - 102.

[GMV] Graham M. H., Mendelzon A. O., Vardi M. Y., *"Notions of Dependency Satisfaction"*, Jour. of ACM, Vol. 33, No. 1, January 1986, pp. 105-129.

[GR] Goldberg A., Robson D., *Smalltalk - 80: The Language and its Implementation* Reading, MA, Addison - Wesley, 1983.

[GZ] Ginsburg S., Zaiddan S. M., *"Properties of Functional-Dependency Families"* Journal of the ACM, Vol. 29, No. 3, July 1982, pp. 678-698.

[IL] Imielinski T., Lipski W., *"Incomplete Information in Relational Databases"* Jour. of ACM, Vol. 31, No. 4, Oct. 1984, pp. 761 - 791.

[HM] Hammer M., Mc Leod D., *"Database Desciption with SDM: A Semantic Data Model"*, ACM TODS, Vol. 6, No. 3, Sept. 1981, pp. 351 - 386.

[HPC] Hurson A. R., Pakzad S. H., Cheng J. - b. *"Object - Oriented Databasse Management Systems: Evolution and Performance Issues"*, Computer, Vol. 26, No. 2, February 1993, pp. 48 - 60.

[HS] Honeyman P., Scoire E., *"New Caracterization of Independence"*, Proceedings of the ACM-SIGMOD Conference, 1983, pp. 92-96.

[Hu] Hull R., *"Finitely Specifiable Implicational Dependency Families"*, Journal of the ACM, Vol. 31, No. 2, April 1984, pp. 210-226.

[Ing] *INGRES/SQL Reference Manual fot the UNIX and VMS Operating Systems*, Release 6.4, December 1991.

[KA] Khoshafian S., Abnous R., *Object - Orientation*, J. Wiley, N. York, 1990.

[KMo] Kuzmanov S., Mogin P., *"The Relation Scheme Extensions"*, Informatica, 4/90, 1990, pp. 31-38.

[L] Lochovsky F. H., ed., *Entity - Relationship Approach to Database Design and Querying*, North Holland, Amsterdam, 1989.

[La] Lazarević B., Nešković S., Marjanović Z., Ilić S., *"Prošireni model objekti - veze"*, Interni izveštaj Laboratorije za informacione sisteme Fakulteta organizacionih nauka Univerziteta u Beogradu ILIS91/3, Beograd, 1991.

[LTK] Ling T. - W., Tompa F. W., Kameda T., *"An Improved Third Normal Form for Relational Databases"*, ACM TODS, Vol. 6, No. 2, June 1981, pp. 329 - 346.

[Lu] Luković I., *Integracija šema modula baze podataka informacionog sistema*, Doktorska disertacija, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, 1996.

[Lus] Lusardi F., *The Database Experts' Guide to SQL*, Intertext Publications/Multiscience Press, Inc, McGraw-Hill, Inc, New Yuork, 1988.

[M] Maier D., *The Theory of Relational Databases*, Computer Science Press, Inc. rockville, Maryland, 1983.

[Ma] March S. T., ed., *Entity - Relationship Approach*, North - Holland, Amsterdam, 1988.

[Me] Mendelzon O. A., *"Database states and Their Tableaux"*, ACM Transactions on Database Systems, Vol. 9, No. 2, June 1984, pp. 264-282.

[Mi] Milićev L., *Objektno - orijentisano programiranje na jeziku C++*, Mikro knjiga, Beograd, 1995.

[ML] Mogin P., Luković I., *"O kontroli integriteta baze podataka"*, Zbornik radova XXXVII jugoslovenske konferencije ETAN, Beograd, 20-23. 09. 1993, VIII sveska, pp. 55-60.

[MLK1] Mogin P., Luković I., Karadžić Ž., *"Ka automatskoj identifikaciji ograničenja baze podataka"*, Zbornik radova XXXVI jugoslovenske konferencije ETAN, Kopaonik, 08-11. 06. 1992, Sveska IX, pp. 19-26.

[MLK2] Mogin P., Luković I., Karadžić Ž., *"Relational Database Schema Design and Application Generating using IIS*CASE Tool"*, Proceedings of International Conference on Technical Informatics, Timisoara, Romania, 16-19. 11. 1994, Vol. 5, pp. 49-58.

[Mo] Mogin P., *Strukture podataka i organizacija datoteka*, Student, Novi Sad, 1994.

[Mo1] Mogin P., *Uvod u baze podataka*, Fakultet tehničkih nauka - Institut za industrijske sisteme, Novi Sad, 1989.

[Mo2] Mogin P., *Organizacija datoteka i uvod u baze podataka*, Viša škola za organizaciju i informatiku, Novi Sad, 1991.

[MR] Mogin P., Ristić S., *"Kandidati za primarni ključ šeme relacije"*, Zbornik radova XXXVIII jugoslovenske konferencije ETRAN, Niš, 07-09. 06. 1994, III sveska, pp. 99-100.

[MUV] Maier D., Ullman J. D., Vardi M. Y., *"On the Foundations of the Universal Relatio Model"*, ACM TODS, Vol. 9, No. 2, June 1984, pp. 283 - 308.

[N] Ng P. A., *"Further Analysis of the Entity - Relationship Approach to Database Design"*, IEEE Tran. on Soft. Eng., Vol. SE - 7, No. 1, Jan. 1981, pp. 85 - 99.

[Ora] *SQL Language Reference Manual*, Version 6.0, Oracle Corporation, 1990.

[PBG] Paredaens J., Bra D. P., Gyssens M., Gucht V. D., *The Structure of the Relational Database Model*, Springer-Verlag, Berlin Heidelberg, 1989.

[RB] Rine D. C., Bhargava B, *"Oject - Oriented Computing"*, Computer, Vol. 25, No. 10, October, 1992, pp. 6 - 10.

[S] Stein J., *"Object -Oriented Programming and Databases"*, Dr. Dobb's Journal, March 1988, pp.18 - 34.

[Sa] Sagiv Y., *"Characterization of Globally Consistent Databases and their Correct Access Path"*, TR University of Illinois, July, 1981.

[SAAF] Senko M. E., Altman E. B., Astrahan M. M., Fehder P. L., *"Data Structures and Accessing in Database Systems"*, IBM Syst. Jour., Vol. 12, No. 1, Jan. 1973, pp. 30 - 93.

[Sc1] Sciore E., *"A Complete Axiomatization of Full Join Dependecies"*, Jour. ACM, Vol. 29, No. 2, Apr. 1982, pp. 373 -393.

[Sc2] Scoire E., *"Improving Database Schemes by Adding Attributes"*, Proceedings of SIGACT - SIGMOD PODS, 1983.

[SS] Smith J., Smith D., *"Database Abstractions: Aggregatio and Generalization"*, ACM TODS, Vol. 2, No. 2, June 1977, pp. 105 - 133.

[St] Stevens A., *C++ Database Development*, MIS:Press, New York, 1994.

[Str] Stroustrup B., *The C++ Programming Language*, Sec. Ed., Addison - Wesley, Reading, Mass., 1993.

[SU] Sadri F., Ullman J. D., *"Template Dependencies: A Large Class of Dependencies in Relational Databases and Its Complete Axiomatization"*, Jour. of ACM, Vol. 29, No. 2, April 1982, pp. 363-372.

[TL] Tsirchitzis D. C., Lochovsky F. H., *Data Models*, Prentice - Hall, Inc., Englewood Cliffs, N. J., 1982.

[U1] Ullman D. J., *Principles of Database Systems*, Computer Science Press, Inc. Rockville, Maryland, 1982.

[U2] Ullman D. J., *Database and Knowledge - Base Systems*, Computer Science Press, Inc. Rockville, Maryland, 1988.

[W] Wagner P., *"Dimensions of Object - Oriented Modeling"*, Computer, Vol. 25, No. 10, October, 1992, pp. 12 - 20.

[Y] Yang C. C., *Relational Databases*, Prentice-Hall, A Division of Simon & Schuster, Inc. Englewood Cliffs, New Jersey, 1986.

[YO] Yu C. T., Ozsoyoglu Z. M.,*"On Determining Tree - Query Membership of a Distributed Query"*, Computer Science Rpt. 80 - 1, University of Alberta, Edmonton, 1980.

Indeks

– A –

– B –

– C –

– Č –

– D –

– E –

– F –

– G –

– H –

– I –

– J –

– K –

– L –

– M –

– N –

– R –

– S –

– Š –

– T –

– U –

– V –

– Z –

– Ž –

Sadržaj knjige "Principi projektovanja baza podataka"

Sponzori

Generalni sponzor:

PARALLEL, Beograd

Sponzori:

1. ENERGOSOFT, Novi Sad
2. MICROSYS, Novi Sad
3. INTERA, Novi Sad
4. PRESENT, Novi Sad
5. FTN - INSTITUT ZA INDUSTRIJSKE SISTEME, Novi Sad
6. ZGOP, Novi Sad
7. INTERSMART, Novi Sad