IMRE PETKOVIĆ

BAZE PODATAKA

SKRIPTA

SUBOTICA, 2004.

SADRŽAJ

1. UVOD

“Danas je već opšte prihvaćen stav, da je svet sastavljen iz tri komponente: materije, energije i informacije. Takav prilaz svetu potvrdjuju kako živi organizmi, tako i organizovani sistemi koje je stvorio čovek. Bez informacija organizovani sistemi ne mogu opstati. Današnji sistemi, medjutim, pored toga što su organizovani, i održavaju svoju organizovanost. Objašnjenje takvog ponašanja sistema leži u neprekidnom izvlačenju informacija iz spoljašnjeg sveta o pojavama i procesima koji se u njemu odigravaju. Stabilnost sistema, koja ima presudan značaj za ocenu radne sposobnosti sistema, se procenjuje razmatranjem dinamike procesa koji se odvijaju u sistemima.“ (A.J.Lerner).

Informacija se danas svugde u svetu smatra resursom koji igra ključnu ulogu u životu i poslovanju organizacija svih vrsta i veličina. Strateški značaj informacija priznaju svi savremeni stručnjaci. Važnost i neophodnost kao atributi uz pojam informacija ističu ulogu otkrivanja, organizovanja, memorisanja i rukovanja saznanja (podataka – informacija). Informacije se mogu obezbediti i pomoću računara uz pomoć softverskog produkta (proizvoda) koji se zove informacioni sistem. Razvoj informacionog sistema je složen i mukotrpan poduhvat koji pored zavidnog nivoa stručnosti na polju informatike traži i druge sposobnosti: otvorenost prema problemima drugih, agilnost i istrajnost u otkrivanju i rešavanju problema i, svakako, komunikativnost. Savremeni informacioni sistemi objekte posmatranja (činjenice koje treba zapisivati i pratiti njihovu sudbinu), njihove osobine i sve bitne događaje formulisane pomoću podataka pohranjuju i crpe iz baza podataka. Upravo će to biti i predmet diskusije, odnosno upoznavanja u ovim beleškama. Predmet »Baze podataka« se osamostalio zahvaljujući stavu da predstavlja važnu i samostalnu celinu u okviru projektovanja i realizacije informacionih sistema. No, ova oblast se samostalno nikada ne obrađuje u praksi, jer bi to bila »l'art pour l'art«, tj. umetnost radi umetnosti. Projektovanje baze podataka predstavlja jednu od osnovnih aktivnosti projektovanja informacionog sistema, koja se svugde u svetu, na fakultetima i višim školama nezavisno izučava u okviru samostalnog predmeta. Ove beleške, međutim imaju zahtevniji cilj, jer pored projektovanja baza podataka obuhvataju i realizaciju baza podataka, kao i savremeni način obrade podataka pomoću relacionih baza podataka.

Materijal je prvenstveno pripremljen za studente na drugoj godini Više tehničke škole Odseka za informatiku koji slušaju predmet »Baze podataka« u trećem semestru sa fondom časova 2+2. Ovaj fond časova ograničava količinu gradiva, koje se može izlagati u okviru raspoloživog vremena. Shodno tome, beleške sadrže filtriran materijal iz ogromnog opusa

projektovanja i rukovanja bazama podataka. Primarni cilj predmeta, pa prema tome i ovog pomagala je prikaz prilaza i pricipa projektovanja, realizacije i obrade podataka pomoću baza podataka. Izložena saznanja u ovom učilu su pretežno teoretskog karaktera. Na vežbama studenti rešavaju konkretne primere i zadatke na konkretnim tehničkim sredstvima, a na osnovu izloženih metoda sa predavanja i instrukcija dobijenih na časovima malobrojnih tabličnih, pretežno praktičnih vežbi.

Na kraju bih želeo izneti svoje subjektivno mišljenje budućim informatičarima da su samo u tom slučaju birali odgovarajući smer (užu struku) za sebe, ako im je uža oblast intersovanja ujedno i svakidašnji hobi. I još nešto na kraju, želeo bih da im informatika ne bude samo zanimanje, već i životni poziv!

U beleškama sigurno postoje manje i/ili veće greške. Broj grešaka se smanjuje sa brojem pregleda (čitanja), ali se u potpunosti, za ovakvo kratko vreme, sve greške ne mogu eliminisati. Sve informacije u vezi poboljšanja kvaliteta materijala su dobrodošle.

mr Imre Petković, dipl. el. ing.
peti@vts.su.ac.yu

Subotica, 19. 08. 2003.

2. OBRADA PODATAKA

2.1.POJMOVI I SHVATANJA

Danas je korišćenje računara za izvršavanje raznoraznih poslova svakako neminovnost, a pored toga je postalo i moda. Ko nema računar, ili ne zna koristiti računar, nije više u "savremenom trendu". To je sasvim logično i prihvatljivo. Postoje, medjutim, razna pogrešna shvatanja u vezi informatike i računarske tehnike koja proističu uglavnom iz neznanja. Danas naime najveći broj korisnika – početnika računara smatra sebe informatičarem, a kao potvrdu za to navodi koliko pisama je napisao pomoću softvera Word2000, koliko crteža je sastavio u Paintbrush Picture okruženju i koliko je slao i primao E-mail-ova u zadnjih mesec dana. Sve se više čini da su se pojmovi cilj, način i sredstva za realizaciju (informacionog) sistema, odnosno obrade podataka izmešali: ako naime neki informacioni sistem (aplikacija) sporo radi ili ne radi besprekorno, kupuju snažniju mašinu i/ili nabavljaju najnovije verzije raznih softvera, umesto da se okrenu samoj aplikaciji, tj. obradi podataka, odnosno da analiziraju rešenje zadatka ili problema (možda baš u tom grmu leži zec!).

Treba naglasiti činjenicu da stručnjak koji je vešt i iskusan u korišćenju računara i raznih softvera nije ujedno (ili automatski) i informatičar! Nenadmašni u korišćenju računara i računarskog softvera se ne mogu smatrati informatičarem iz sledećeg prostog razloga: problemi u informatici nisu (samo ili pre svega) tehničke prirode! Primena računara, znači, nije ujedno i informatika. U informatici najveći problem predstavljaju brojnost i kompleksnost činilaca ili elemenata informacionih sistema.

Pojam informatike je definisan na nekoliko načina do danas. Ovde ćemo navesti jedan sažeti, kraći i jedan opširniji opis pojma informatike. Po opširnijoj definiciji **informatika se bavi raznim modalitetima (načinima) strujanja, prenosa informacija, metodima njene obrade i korišćenja, njenim delovanjima na produktivnost i efektivnost (efikasnost) njenom primenom za praćenje, kontrolu i upravljanje, kao i ulogom informacija u razvoju privrede i društva**[1]. Po ovom opisu informatika predstavlja jako opširan pojam: obuhvata integraciju informacija (podataka) u sistem, informatička sredstva, načine obrade informacija od trenutka nastanka informacija (tačnije podataka) do njene isporuke do krajnjeg korisnika, pa sve do raznoraznih uticaja informacija na privredne i društvene subjekte. Po ovom shvatanju informatika predstavlja interdisciplinarni pojam, koja u sebi sadrži telekomunikacije, računarsku

[1] Izvor: [20.], strana: 8.

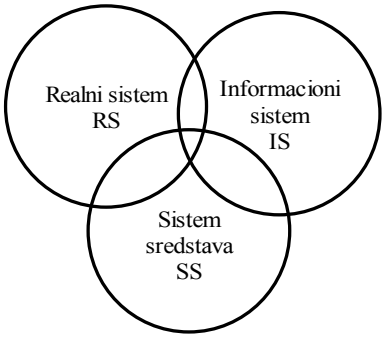
tehniku, primenu računarske tehnike, primenu mikroelektronike i robotike, organizaciju, upravljanje, kao i odgovarajuće oblasti privrednih i društvenih nauka. Ta definicija je preopširna za naše potrebe.

Po sažetoj definiciji **informatika je nauka za upoznavanje, efikasno i svrsishodno organizovanje, sredjivanje i obradu saznanja (podataka ili informacija)**. Informatičarem se može smatrati stručnjak koji je vešt u toj oblasti[2].

Informatičar u procesu shvatanja, predstavljanja i obrade realnog sveta i njegovog informatičkog razradjivanja uvek treba da ima u vidu sledeće tri suštine, kao i njihove medjusobne veze:

1. objektivna stvarnost (realni sistem – RS),
2. slika stvarnosti na osnovu poznatih saznanja o njoj (informacioni sistem – IS) i
3. skup sredstava (hardvera, softvera, lifeware-a i orgvera, tj. sistema sredstava – SS).

Tu se radi o tri sistema koji se delimično preklapaju, ali su u isto vreme i relativno nezavisni (slika 2.1.). Te tri suštine moraju biti medjusobno uskladjene, odnosno moraju biti u što većoj harmoniji. Taj stav se danas sve više gubi iz vida. Današnji je trend da se kupuju što snažnije mašine (računari) i najnovije verzije softvera.



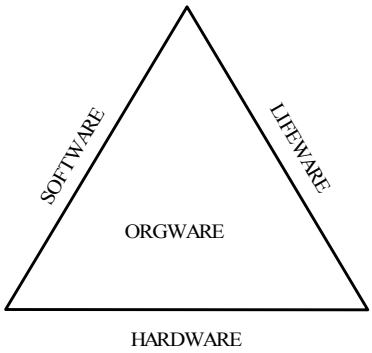
Slika 2.1.

Sistem sredstava se može prikazati i grafički[3], kao što se to nalazi na slici 2.2. Grafički prikaz dobro odslikava sintezu ili medjusobnu povezanost te četiri komponente sistema sredstava.

Hardver (hardware) čine materijalni sadržaji sistema sredstava: računar sa svim ulaznim, izlaznim i ulazno-izlaznim kao i uredjajima za prenos podataka na daljinu, dodatni uredjaji za sakupljanje, prenos, predobradu, memorisanje i distribuciju podataka. Jednom rečju hardver je „celokupno gvoždje", ili „sve što se može opipati". Hardver računara se intenzivno razvija. Naše aplikacije koriste jedva nekoliko procenata ukupnog hardvera, no današnji softveri su jako

[2] Izvor: [11.], strana: 35.
[3] Izvor: [6.], strana:

zahtevni prema performansama hardvera i hardver se samo upravo zbog njih i razvija. Prema današnjim procenama razvoj hardvera prednjači oko 10-15 godina ispred razvoja softvera.



Slika 2.2.

Softver obuhvata sve programsko-algoritamske sadržaje: operativni(e) sistem(e), uslužne programe, jezičke procesore (programske jezike), sisteme za rukovanje bazama podataka, editore tekstova i tabela kao i raznorazne softverske produkte za rešavanje opštih problema na svim mogućim poljima ljudske aktivnosti. Razvoj softvera je na prvi pogled brži i očigledniji od razvoja hardvera u zadnje vreme. Primena softvera, medjutim, kaska za njegovim razvojem (razvojem hardvera) čak 20 godina po nekim procenama. Neki softveri za rukovanje bazama podataka naime toliko su kompleksni, da ih laici skoro ni pokrenuti ne znaju, a i najsposobniji projektanti i programeri mogu upoznati i razumno (u praksi) primeniti svega nekoliko procenata njihovih mogućnosti.

Lifeware sistema sredstava predstavlja kadrove neophodne za izvršavanje poslova na svim segmentima u toku životnog ciklusa informacionog sistema. Kadrovi su danas dobro pripremljeni za korišćenje računara i najnovijih softvera, ali imaju pogrešan stav (i shvatanja) prema razvoju informacionih sistema.

Orgver sistema sredstava predstavlja njegovo organizaciono ustrojstvo. Povezuje prethodno opisane tri komponente u organizacionu strukturu sa ciljem da se ispune ciljevi informacionog sistema zacrtani u zahtevima naručioca. Orgver je u stvari metod organizacije saznanja.

U cilju besprekornog funkcionisanja informacionog sistema ili obrade podataka, sve četiri komponente sistema sredstava moraju biti na visokom nivuo harmonije, odnosno medjusobno dobro uskladjene i u potpunosti kompatibilne. Zapostavljanje bilo koje komponente rezultuje u drastičnom smanjenju efektivnosti informacionog sistema bez obzira na kvalitet i uskladjenost ostale tri komponente.

2.2.O SAZNANJU

Ljudsko biće neprekidno skuplja saznanja iz okolnog sveta pomoću svojih čulnih organa. Nepobitno je da i osmeh, prijatan miris i ukusno jelo može biti jednoznačno i važno saznanje, informacija, no ipak se za komunikaciju izmedju ljudi najviše koristi jezik. Kao što ćemo kasnije videti podatak je saznanje koje se može protumačiti, ali koje još nije protumačeno. Protumačeni podatak, ako nosi novo saznanje za lice koje ga je protumačilo, postaće informacija.

U prirodnom govoru (jeziku) svaki prost iskaz (prosta rečenica) ima subjekat i predikat. U slučaju da fali subjekat ili predikat, on se podrazumeva, odnosno može se zaključiti na osnovu prethodnih iskaza. Subjekat odredjuje o čemu želimo nešto izjaviti, a predikat sam iskaz u vezi subjekta rečenice.

2.1.1. Proces sticanja saznanja

Do samog sticanja saznanja može se doći apsolviranjem sledeća četiri koraka ili sledeće četiri aktivnosti:

1. opažanje (observation)
2. čulno primanje ili percepcija saznanja (perception),
3. umno primanje ili identifikacija sadržaja saznanja – shvatanje (comprehension) i
4. razumevanje (understanding).

U slučaju da se desi makar i jedna "greška" u prethodna četiri koraka ili izostane bilo koji korak, do aktivnosti tumačenja saznanja ne može doći.

Opažanje pretpostavlja prisustvo sredine, materije i/ili situacije koja ili koje nose saznanja (ako ne slušam radio, ne mogu steći saznanje koje se prenosi preko radio talasa, ako ne čitam oglase u liftu, nikad neću saznati ko šta oglašava na taj način).

Percepcija ukazuje na ulogu naših čulnih organa u procesu pretvaranja podataka u informacije. Za hendikepirane ljude, pretežno na polju sluha i vida, uslovi opažanja i percepcije, odnosno stvaranja ili nastajanja informacija su otežani.

Identifikacija sadržaja saznanja (shvatanje) se teže objašnjava od percepcije, ali nam je verovatno poznata situacija iz svakodnevnog života kad u pauzi za doručak grickajući kreker ili sendvič, uz to čitajući novine priupitamo našeg sagovornika: "Šta si ono rekao?". Percepcija saznanja se očito odigrala, medjutim, identifikacija njegovog sadržaja nije. Naša pažnja je bila usmerena na vesti iz novina. U prvom trenutku kada shvatimo da nam se sagovornik obratio,

otpočinje “rekonstrukcija” percepcije, da bi, nakon neuspele rekonstrukcije sledilo gore navedeno pitanje za ponavljanje sadržaja izrečene misli. Drugi karakterističan primer je obračun struje: nikad u životu nisam imao u ruci obračunski list za potrošenu električnu energiju sa kojeg sam mogao saznati koliko za potrošača košta jedan kilovat-čas električne energije! Taj podatak nisam mogao izdejstvovati ni kada sam bio šef sektora Informatike u Elektrodistribuciji “SUBOTICA” u Subotici! Cena jedinice električne energije se i tada, a i sad može dobiti agregiranjem (izračunavanjem) podataka sa obračunskog lista, ali to ni ja nisam nikada znao kako! Primetimo da je fizička organizacija i prikazivanje saznanja bitno utiču na identifikaciju saznanja: suštinska saznanja treba prikazati na upadljivim mestima i na način koji se odmah, na prvi pogled primeti.

Razumevanje saznanja predstavlja četvrti, zadnji korak pre njegovog protumačenja. Saznanje se najčešće formuliše u vidu vesti ili iskaza. Iskaz se može shvatiti samo u tom slučaju ako primaoc vesti razume svaku njenu reč, tj. kad generator iskaza i primaoc “govore istim jezikom”.

Protumačenje saznanja je u stvari povezivanje novodobijenog podatka ili iskaza sa ranijim saznanjima. Ako ne postoje ranija saznanja sa kojima bi se mogao povezati podatak, informacija naravno neće nastati. Neće se roditi informacija ni tada, ako se već povezivanje jednom dogodilo. Informacija nastaje u trenucima kada razmišljamo o samom sadržaju podatka, kad donosimo neki sud u vezi tog iskaza, kada pronadjemo mesto tog podatka u nekoj ili nekim klasifikacijama, kada formiramo neku misao, kada formulišemo neki zaključak.

Upravo iz prethodnih razloga na računaru uvek smeštamo “samo” podatke, koje posle, po potrebi i obradimo. Informacija se radja samo u glavama ljudi, a obrada podataka na računaru je samo onda odgovarajuća, ako obezbedjuje takve nizove podataka da ljudima olakšava i ubrzava proces njihovog protumačenja.

2.1.2. Podatak, informacija, znanje

Pojmovi kao što su podatak, informacija i znanje su objašnjeni na više mesta u stručnoj literaturi. Svugde u objašnjenjima postoji semantička razlika izmedju podatka i informacije.

Najprostiji prilaz u objašnjenju pojma podatak naglašava da je on zapis neke konkretne činjenice ili saznanja na proizvoljnom medijumu (papiru, magnetskom medijumu, itd.) nezavisno od njegove dalje sudbine, tj. nezavisno od toga da li će biti upotrebljen za donošenje poslovnih odluka i/ili upravljanje ili ne. Ako se podatak upotrebi za upravljanje, tj u procesu odlučivanja,

on se transformiše u informaciju. Zapis zadržava svojstvo podatka sve dok postoji verovatnoća da će se na neki način transformisati u informaciju, a posle gubi to svoje svojstvo[4].

Takvo shvatanje pojma podatka i informacije u današnje doba već ne može da opstane. Opšteprihvaćen je danas stav u vezi pojma podatka, da je on konkretizacija pojave (ili pojavnog oblika) informacije, grupa simbola koja se može obraditi na računaru, ali sa suženijim značenjem i manjim značajem od informacije. Informacija je bogatija u semantičkom smislu od podatka. Podatak je neprotumačeno saznanje ili neprotumačena informacija koja se može protumačiti. Informacija je, prema tome protumačeni podatak. Podaci nose u sebi informaciju, ali količina informacionog sadržaja podatka je takodje relativan pojam, jer u velikoj meri zavisi od primaoca podatka (informacija). Isti podatak, naime za jednog primaoca ostaje samo podatak (znači on ga nije mogao protumačiti ili već poseduje taj podatak), a za drugog primaoca postaje informacija (ako za njega predstavlja novo saznanje).

Podatak je, znači, takav neprotumačeni niz simbola ili znakova koji opisuje karakteristike (obeležja) stanja objekata ili uopšte promena u realnom sistemu sa ciljem kasnijeg korišćenja i/ili obrade, a obuhvata se u formi pogodnoj za memorisanje i prenos.

Istraživači koji se bave teorijom informacija se ni do današnjeg dana nisu mogli da se usaglase u vezi precizne definicije informacije. O tome svedoče sledeće formulacije definicije informacije.

B. Langefors ima sledeći stav:

"Informacijom se smatra svako znanje, poruka koja se može iskoristiti za neko dokazivanje ili omogućuje donošenje odluka."[5].

U Webster's Dictionary-ju se takodje nalazi definicija informacije:

Informacija je znanje, inteligencija, vest, odnosno njihova predaja ili primanje.

Interesantnu formulaciju je dao H. Rittel, norveški istraživač:

"Informacija je takva aktivnost koja rezultuje u izmeni nečijeg znanja."[6].

Novi momenat u sledećoj definiciji je taj da se informacijom smatra sredjeni niz već poznatih ili postojećih podataka:

[4] Izvor: [6.], strana:
[5] Izvor: [22.], strana: 13.
[6] Izvor: [22.], strana: 13.

"Informacija nije ništa drugo, već nesredjeni niz podataka ili činjenica dovedenih pomoću izvesnih procesa na upotrebljivu formu."[7].

Slično mišljenje o pojmu informacije ima i E. Noszkay:

"Informacija nije generalno novo saznanje, nego takvo novo odnosno otkriveno saznanje koje nastaje na osnovu sistematizacije, uporedjivanja, analize i vrednovanja već poznatih saznanja (podataka, činjenica)"[8].

Informacija je protumačeni podatak, iskaz, saznanje, saopštenje, vest, koja može da posluži za smanjenje neizvesnosti, za donošenje odluka, za upravljanje i uvek predstavlja kvalitet više od podatka i ostalih gore navedenih pojmova, jer angažuje svog primaoca: za razmišljanje, za neki izbor, za neku aktivnost itd.

Saznanja o realnom svetu vezana za njegove pojave i procese kroz iskustva i učenje omogućuju formulisanje zaključaka, analizu uzročno-posledičnih veza, apstrahiranje – izvlačenje bitnih crta – atributa, kao i sposobnost sticanja novih saznanja i označavaju sistem na višem nivou apstarkcije koji se naziva znanjem.

Znanje je, prema tome, **pojmovna slika realnosti u svesti čoveka o stvarima, objektima, činjenicama, pojavama i procesima zajedno sa svim njihovim medjusobnim i uzročno-posledičnim vezama**.

2.1.3. Sredine za prenos saznanja

Prvi korak u procesu sticanja saznanja je opažanje, a ono je uslovljeno prisustvom sredine koja (pre)nosi predmetno saznanje koje želimo usvojiti. Sredine za prenos saznanja određuju naši čulni organi. Tako sredina za prenos saznanja može biti slika, mimika, pokret, zvuk, ukus, miris, njihova proizvoljna kombinacija itd., i naravno, najčešća sredina za prenos saznanja: jezik.

Čovek najčešće koristi prostu (ili prostoproširenu) potvrdnu rečenicu:

Zgrada Vojvođanske banke je bela.

Svaka potpuna rečenica se sastoji od dvostrukog subjekta i dvostrukog predikata (prethodna rečenica nije potpuna u tom pogledu, potpuna rečenica bi glasila: **Boja zgrade**

[7] J Frates u [22.], strana: 13.
[8] Izvor: [20.], strana: 9.

Vojvođanske banke je bela.)[9]. Dvostruki subjekat se sastoji iz dva dela: prvi označava najbliži rod (genus proximum), a drugi specifičnu razliku (znak raspoznavanja – differentia specifica). Često se prvi deo subjekta zove generički, a drugi specifični subjekat. U našem primeru je najbliži rod **Zgrada**, a specifični razlika ili znak razlikovanja **Vojvođanska banka**. Delovi subjekta ne moraju uvek da se pojavljuju eksplicitno u rečenici. Na primer u slučaju nedostatka znaka razlikovanja rečenica glasi: **Zgrada je bela** (Kad nema najbližeg roda rečenicaima oblik: **A Vojvođanske banke je bela**). U tim slučajevima se podrazumevaju (na osnovu prethodnih iskaza ili rečenica) i kažemo da su implicitni.

Slično se može rezonovati u vezi (dvostrukog) predikata. Generički predikat (genus proximum) je »Boja« (u prvobitnoj rečenici taj deo predikata nije prisutan u rečenici, tj. On je implicitni predikat), a specifični predikat je **bela**.

U rečenicama prirodnog govora (jezika) prisutni su naravno i drugi mogući delovi rečenice, kao što su objekti, prilozi i pridevi. Njih možemo smatrati iskazima i svrstavati ih među generične i specifične predikate.

Izjave, odnosno rečence mogu biti i znatno složenije od gore navedenih prostih i proširenih rečenica. Kod složenih i višestruko složenih rečenica, međutim, uvek je moguće izvršiti podelu na proste i prostoproširene rečenice.

Važno je primetiti i naglasiti da je za proces sticanja saznanja neophodno da su prisutni sve četiri dimenzije saznanja: generički i specifični subjekat, a takđe i generički i specifični predikat. Ta činjenica je predstavljala i predstavlja osnovu za izgradnju raznoraznih modela podataka, o čemu će biti više reči u sledećem poglavlju.

2.3. NAČINI OBRADE SAZNANJA

Saznanja se danas (ako izuzmemo multimediju, tj. grafiku, sliku, zvk, itd) obradjuju na dva osnovna načina:

1. činjenični (faktografski) način obrade saznanja i
2. tekstualni način obrade saznanja.

U vezi faktografskog načina obrade saznanja treba se malo vratiti nazad u istoriji. Stari Grci su "činjenice koje treba zabeležiti" zvali rekordima. Još i danas se za skup bitnih saznanja u vezi značajnih činjenica i/ili dogadjaja od opšteg interesa koristi izraz rekord i to ne samo u našem jeziku. Ista reč se koristi u engleskom jeziku i za niz vezanih podataka na nekom računu.

[9] Izvor: [9], str. 15-16.

Rekordi (slogovi, zapisi) se o pojavama ili stvarima koji pripadaju istoj klasi stvari smeštaju u isti fajl (dosije ili registrator), tako da je jedna stavka je u stvari jedan rekord (slog). Rekord (slog) se sastoji od polja, a za sadržaj polja se koristi pojam ili izraz podatak.

Činjenični način obrade saznanja se danas zove obrada podataka. Obrada podataka se može realizovati na dva načina:

1. klasičnom organizacijom datoteka i
2. pomoću baze podataka.

Klasična organizacija podataka, odnosno obrada zasnovana na podacima smeštenih u pojedine datoteke ne vodi računa o realno postojećim vezama između pojedinih podataka. Podaci se smeštaju višestruko (redundantnost), shodno pojedinim obradama, što dovodi do nekonzistentnosti (nemogućnosti istovremenog ažuriranja vrednosti nekog podatka, smeštenih u različitim datotekama) Nema povezanosti izmedju datoteka, odnosno između slogova različitih datoteka (unatoč tome, što u realnom svetu možda ta veza i egzistira). Fizičko smeštanje podataka fajla je najvažnije kod ove vrste obrade podataka, o sadržaju, smislu, značaju i značenju podataka u fajlovima malo je ko brinuo.

Baza podataka je u stvari organizovani skup fajlova. Fajlovi (u terminologiji baza podataka zovu se tabele) su međusobno povezani, zbog objektivno postojećih povezanosti podataka u njima. Obrada podataka na osnovu baze podataka ne isključuje klasičnu obradu datoteka ili vertikalno kretanje u tabeli prilikom obrade, ali je prvenstveno transverzijalnog karaktera (više je krasi horizontalno kretanje izmedju tabela).

Tekstualni način obrade saznanja se često meša sa pojmovima obrada teksta i editiranje teksta. Editor teksta ne može da se koristi za rukovanje podacima i za stvaranje novih podataka. Kao što znamo tekstualni fajlovi nemaju stavke – rekorde, editor ne poznaje ni jednu od četiri dimenzije saznanja, a da o prirodno postojećim vezama između podataka i ne govorimo.

Tekstualni način obrade saznanja obuhvata dve moguće realizacije:

1. sisteme za obradu teksta i
2. sisteme za obradu tabela.

Sistemi za obradu teksta mogu prihvatiti rečenice (tekstove) bilo kojeg prirodnog jezika. Dok su reči, tj. podaci kod faktografskog načina obrade semeštene u pojedina tačno predviđena polja, u sistemima za obradu teksta reči se nalaze u neodređenom, prirodnom redosledu nastanka, kako ih je sastavljač teksta sačinio. Treba, međutim, primetiti da sistemi za obradu teksta razlikuju, odnosno mogu prepoznati dve dimenzije saznanja: generički i specifični subjekat. Mogu naime da sadrže saznanja o različitim stvarima, formirajući time različite "rekorde". Sa druge strane, u cilju efikasnog pretraživanja formiraju tzv. tezaurus u kojem se

smeštaju reči (sadržaji) i njihova mesta pojavljivanja u memorisanom tekstu. To je posebna indeksna tabela (datoteka) koja u jednom delu sadrži vrednost podatka (tj. sam podatak) na koju se može formulisati zahtev za pretraživanje, a u drugom delu sve adrese pojavljivanja te vrednosti u okviru teksta. Time se ostvaruje specifični predikat. Tumačenje ili interpretacija niza simbola (reči) koji se nalazi u tezaurusu time još nije razrešena. Taj problem se rešava mehanizmom koji se zove deskriptor. Sačinjen je za eliminaciju problema postojanja homonima (homonim je reč ili niz simbola sa više mogućih značenja). Fizička realizacija deskriptora je slična tezaurusu, naime prilikom ukucavanja niza simbola koji se smešta u tezaurus, u deskriptoru se memoriše tačno značenje upravo unešenog niza simbola. Time je realizovan geerički predikat. Korisnik pri unosu testa može označiti koje ključne reči treba zapamtiti u tezaurusu (ujedn se memoriše i njihovo mesto u tekstu), a deskriptor služi za pamćenje obeležja ili kategorije kojoj pomenuta ključna reč pripada. Time u stvari pomoću tezaurusa i deskriptora možemo pronaći sve konkretizacije izabranih nizova simbola iz tezaurusa, a i sve pojave svih konkretizacija memorisanih kategorija u deskriptorima.

Pojedini dokumenti se u sistemima za obradu tekstova mogu, na osnovu – recimo – ključnih reči u deskriptorima, eksplicitno grupisati u "fajl", koje sistem zajedno rukuje i obradjuje. Time je ostvaren tip entiteta ("fajl") i pojava tipa entiteta (dokument).

Sistemi za obradu tekstova, ipak, ne predstavljaju sistem za rukovanje bazom podataka jer nisu zasnovani na modelu podataka i ne mogu obradu realizovati na osnovu veza koje postoje izmedju "fajlova".

Banka podataka je skup pogodno organizovanih i celishodno smeštenih tekstualnih datoteka (dokumenata) kojima se može pristupiti pomoću obezbedjenih sistema za obradu tekstova. Povezivanje ovih dokumenata (datoteka) po sadržaju nije moguće, ali je tehnika njihove obrade ista.

Sistemi za obradu tabela se ponekad nazivaju (pogrešno) sistemima za rukovanje bazom podataka, ali to ne odgovara stvarnosti. Greška potiče možda od toga što se na logičkom nivou entitetu (pojmovni nivo) odgovara pojam relacije ili tabele. Tu se radi o homonimu: pojam tabela u terminologiji baze podataka označava skup slogova sa tačno navedenim redosledom smeštanja podataka. No, u klasičnoj tabeli (a to je drugi sadržaj pojma tabela) bilo koji sadržaj (bilo koji podatak) može da se smesti u bilo koju kolonu, bez kontrole se dopušta upis proizvoljne vrednosti u proizvoljnu kolonu. Kod klasičnih tabela ne postoje nikakve medjusobne veze.

2.4. INFORMACIONI SISTEM

Činjenični ili faktografski način obrade saznanja ili obrada podataka se realizuje posredstvom aplikacija ili realizovanih (isprogramiranih i istestiranih) informacionih sistema.

Kako samo ime kaže informacioni sistem je takodje sistem, dakle i za njega važe sve zakonitosti koje su otkrivene i dokazane u naučnoj oblasti teorije sistema. Treba, medjutim, naglasiti da pojam informacioni sistem nije ekvivalentan pojmu sistem informacija. Sistem informacija označava samo sredjivanje, odnosno organizaciju informacija, a informacioni sistem obuhvata i razne druge elemente sasvim različite naravi.

Upoznavanje sa suštinom informacionih sistema ne predstavlja prvenstveni cilj ovog predmeta, ali baze podataka predstavljaju deo informacionog sistema i u tom smislu treba da se upoznamo sa definicijom pojama informacionog sistema.

Informacioni sistem je organizovani skup sledećih činilaca ili elemenata kao i njihovih medjuveza:

1. podataka (informacija),
2. sa njima povezanih informacionih dogadjaja,
3. nad njima izvršenih aktivnosti,
4. potrebni resursi vezani za prethodna tri elementa, kao i za razvoj i funkcionisanje informacionog sistema
5. korisnika informacija (podataka) i
6. standarda i procedura koji upravljaju napred navedenim činiocima.[10]

Informacioni sistemi se razvijaju da bi zadovoljili potrebe organizacija različitih profila, kao i pojedinaca, a iza tih zahteva i potreba, u pozadini uvek stoje ljudi i njihova težnja da što brže i jednostavnije dodju do njima važnih podataka, informacija. Podaci i ljudi spadaju u činioce informacionih sistema, i kako su oni medjusobno povezani, tako su i sa ostalim elementima informacionog sistema u vezi.

Saznanja u okviru nekog konkretnog informacionog sistema i sami predstavljaju sistem. Sistem podataka je i sam jako složen. Od saznanja do podatka takodje treba stići (kao što je već napred izloženo), isto tako od podatka do informacije. Podatak, koji će za neke postati i informacija predstavlja osnovu za komunikaciju. Podatak ima razne svoje aspekte. Njega treba fizički smestiti na magnetske medijume računara, podaci su medjusobno i logički povezani, a svakako služe za stvaranje slike o realnom sistemu.

[10] Izvor: [11.], strana: 43.

2.5. KONCEPCIJA BAZE PODATAKA

Početkom šezdesetih godina javila se potreba za intenzivnijim razvojem programa u različitim oblastima (uglavnom u evidencijama stanja, obračunima potrošnje raznih energenata i knjiženjima). Tadašnji programski jezici namenjeni za takve svrhe, COBOL i PL1 su obezbedili potrebne mehanizme za definiciju i rukovanje podacima, ali su neophodno morali sadržati opis (definiciju) podataka u svakom pojedinom programu. Tada je artikulisana potreba da se rutine za obradu podataka i deklaracije za opis podataka uopštavaju i osamostale. Tako bi se programi oslobodili od ponavljajućih balasta u vezi definicije i obrade podataka, i ujedno bi efikasnost rada programera znatno porasla. Takav pristup je potpomagala pojava adresibilnih eksternih memorija (diskova), a i težnja analitičara za eliminisanje redundancije u podacima. Rodila se ideja o rukovanju podacima pomoću baze podataka.

Prethodno formulisani zahtevi su pokrenuli razradu takvog softverskog produkta, koji je na sasvim uopšteni način rešavao deklaraciju podataka kao i njihovo standardizovano rukovanje. Taj softver je kasnije prerastao u sistem (ili softver) za upravljanje bazom podataka. Programi su postali nezavisni od podataka, a u datotekama u kojima do tada bilo velike redundancije (ponavljanja istih podataka u različitim datotekama – razlog za to je taj, što su datoteke bile projektovane za pojedine programe) preklapanja su se smanjivala do najniže neophodne mere.

Paralelno tim procesima se povećavao broj tih stručnjaka, koji su uvideli i stalno naglašavali činjenicu, da je najstabilniji od činilaca informacionih sistema upravo skup podataka. Posedovanje dobro izgrađene baze podataka omogućuje realizaciju bilo kakve nove zahteve korisnika. Zato su preporučili da se razvojne aktivnosti informacionog sistema otpočinju sa projektovanjem i izgradnjom baze podataka.

Koncepcija baze podataka je ponudila mogućnost definisanja prirodno postojećih veza između podataka. Na početku prepoznavanja te alternative se rodila ideja o potpunom eliminisanju preklapanja (redundancije) podataka. Ta ideja je bila pogrešna, što su docnije i dokazali, jer je za obezbeđenje održavanja veza između podataka u bazi podataka neophodan jedan optimalni nivo redundancije.

Za formulisanje i postizanje optimalne redundancije je razrađen proces pod nazivom modeliranje podataka (razrada je trajala više godina – grubo tokom sedamdesetih godina). U tom traganju su najzaslužniji Bachmann, Codd i Halassy.

3. MODELI PODATAKA

3.1. FORMALNO-MATEMATIČKI PRISTUP MODELIMA PODATAKA

Model podataka predstavja matematičko preslikavnje dela realnog sistema za koji se projektuje informacioni sistem. U cilju obezbeđenja podloge za realizaciju baze podataka i nad njom izgrađenog informacionog sistema model podataka treba da sadži informacije o:[1]

1. statičkim-strukturalnim osobinama
2. postojećim ograničenjima i
3. dinamičkim osobinama

realnog sistema.

Statičke osobine realnog sistema su relativno stabilne u vremenu i predodređuju strukturu buduće baze podataka. Ograničenja predstavljaju lepezu objektivno postojećih uslova u realnom svetu kao što su dozvoljene i nedozvoljene vrednosti podataka, pravila poslovanja i ponašanja realnog sistema. Ograničenja će se ugraditi u buduću bazu podataka u vidu uslova integriteta. Dinamičke osobine predstavljaju razvoj, odnosno promene realnog sistema, a odražavaju se u promenama podataka o komponentama sistema. Na osnovu dinamičkih osobina se definišu operacije, odnosno procedure (programi) koje će realizovati promene podataka odgovarajućih komponenata sistema shodno razvoju (promenama) u realnom svetu.

Shodno gore formulisanom sadržaju modela podataka razlikovaćemo tri njegove komponente: strukturalnu, integritetu i operacijsku komponentu.

Strukturalna komponenta modela podataka obuhvata osnovne (elementarne, primitivne) koncepte i pravila za izgradnju složenih koncepata, koji se grade od elementarnih koncepata. Koncept je apstrakcija entiteta ili tipa entiteta[2] (zajednički naziv za neki konkretan subjekat ili klasu sličnih subjekata, neki konkretan objekat ili klasu sličnih objekata i neki konkretan događaj ili klasu sličnih događaja), neka osobina entiteta ili konkretna vrednost osobine entiteta, odnosno veza između dva tipa entiteta (dve klase subjekata, dve klase objekata, dve klase događaja, određene klase subjekata i određene klase objekata, određene klase subjekata i određene klase događaja i najzad određene klase objekata i određene klase događaja), ili dva različita entiteta (dva konkretna subjekta, dva konkretna objekta, dva konkretna događaja,

[1] Izvor: [19.], strana: 1-2.
[2] Strožije određivanje pojmova entitet i tip entiteta se nalazi u daljem tekstu.

konkretnog subjekta i konkretnog objekta, konkretnog subjekta i konkretnog događaja i na kraju konkretnog objekta i konkretnog događaja).

Primitivni koncepti strukturalne komponente modela podataka su :

- opis osobine skupa entiteta (obeležje) i
- skup konkretnih vrednosti te osobine (domen).

Složeni koncepti strukturalne komponente modela podataka su sledeći:

- opis skupa entiteta (tip entiteta)
- opis veza između skupa entiteta (poveznik) i
- opis baze podataka (šema baze podataka).

Pomoću koncepata i pravila za izgradnju grade se dve vrste modela: modeli skupova entiteta (modeli tipa entiteta) i modeli konkretnih entiteta (modeli entiteta). Model tipa entiteta se sastoji od naziva tipa entiteta i skupa obeležja (naziva bitnih zajedničkih osobina skupa entiteta). Model entiteta se dobija navođenjem konkretnih vrednosti pojedinih obeležja odgovarajućeg tipa entiteta kojem posmatrani entitet pripada.

Napred izrečene pojmove, u cilju lakšeg shvatanja i pamćenja ilustrovaćemo sledećim primerom. Jedan model tipa entiteta vozila može da se formuliše i predstavlja na sledeći način:

VOZILO(registarski_broj, broj_šasije, tip_vozila, boja).

Model entiteta, jednog konkretnog vozila može biti

(SU 372-66,TMBEEA614TO334911,Škoda Felicija GLS, trula višnja).

Model tipa entiteta lice može imati sledeći oblik

LICE(identifikacioni_broj_lica, ime_i_prezime)

Između entiteta u realnom svetu postoje raznovrsne veze (povezanosti). U cilju predstavljanja tih veza u strukturalnoj komponenti modela podataka, formulišu se relacije u skup modela tipa entiteta ili u skup modela entiteta (skup modela konkretnih entiteta). Tako nastaju dve vrste struktura na **dva nivoa apstrakcije**. Struktura nad skupom modela tipa entiteta predstavlja viši nivo apstrakcije od strukture nad skupom modela konkretnih entiteta.

Model relacije između tipa entiteta LICE i tipa entiteta VOZILO može da se izrazi na sledeći način:

Poseduje(LICE, VOZILO).

Model veze između konkretnih entiteta po prethodnom predlogu izgleda kao što sledi:

((1234,Imre Petković), (SU 372-66,TMBEEA614TO334911,Škoda Felicija GLS, trula višnja).

Oblast strukturalne komponente modela podataka obuhvata i dva dosta često korišćena pojma, a to su: intenzija i ekstenzija. **Intenzija** predstavlja definicioni opis nekog skupa. To drugim rečima znači da je model skupa entiteta ili model tipa entiteta predstavlja intenziju.

Ekstenzija predstavlja barem jednu od mogućih pojava nekog skupa, odnosno jedan model entiteta (ili jedan skup konkretne pojave nednog tipa entiteta).

Integritetna komponenta modela podataka predstavlja skup ograničenja diktiranih od strane realnog sistema, a posledica su njegovog poslovanja i ponašanja. Pravila ponašanja i poslovanja realnog sistema artikulišu se pomoću

- ograničenja domena, odnosno skupa konkretnih vrednosti obeležja
- ograničenja veza u skupu modela tipa entiteta ili u skupu modela entiteta (u skupu modela konkretnih entiteta)
- ograničenja odnosa između konkretnih entiteta i njima pridruženih konkretnih vrednosti iz odgovarajućeg domena.

Na pogodan način formulisana ograničenja se u procesu projektovanja nazivaju uslovima integriteta baze podataka. Jedna moguća kombinacija načina formulisanja gore navedene tri vrste ograničenja je sledeća:

- za ograničenje obeležja A notacija $[a_1,a_2]$ označava interval dozvoljenih vrednosti, tj. $a_1 \leq A \leq a_2$.
- Za ograničenje veze niz simbola (LICE(0,N):VOZILO(1,1)) ima sledeće značenje (ako je naziv veze poseduje)
    - LICE ne mora biti povezan ni sa jednim vozilom (ne mora posedovati ni jedno vozilo), a može da bude u vezi sa više vozila (tj. može da ima više vozila)
    - VOZILO mora biti u vezi sa najmanje jednim licem, a najviše sa jednim licem (vozilo može biti vlasništvo samo jednog lica – barem po mojim današnjim saznanjima)
- Za ograničenje odnosa između konkretnih entiteta i njima pridruženih konkretnih vrednosti umesto notacije koristićemo rečenicu: svako VOZILO ima jedan registarski_broj, a jedan registarski_broj se dodeljuje samo (ili najviše) jednom VOZILU.

Ako je sadržaj baze podataka u skladu sa uslovima integriteta, za nju kažemo da je konzistentna (neprotivrečna).

Strukturalna i integritetna komponenta modela podataka određuju statičku strukturu dela realnog života za koji se projektuje informacioni sistem.

Operacijska komponenta modela podataka opisuje dinamičke osobine – tj. promene u realnom sistemu pomoću skupa operacija. Operacije se izvršavaju na konkretnim podacima (na pojavi baze podataka). Svaka operacija $o \in O$, ako se dozvoljava njeno izvršavanje, menja stanje

baze podataka (na osnovu korišćenih oznaka iz [19] za skup stanja: $D=\{d_i|i=1,...,n\}$, operacija se može formulisati kao $o:D\rightarrow D$). Izmena stanja baze podataka ne znači automatski izmenu pojave baze podataka (korisničke podatke smeštene u bazu), jer baza podataka pored korisnih podataka sadrži razne kontrolne mehanizme (zabranu pristupa grupi korisnika nekom skupu podataka) ili indikatora (indikator tekućeg sloga ili indikator aktvnosti). Sa druge strane izvršavanje operacije se ne dozvoljava, ako bi ona njenim izvršavanjem dovela sadržaj baze podataka u nekonzistentno stanje.

Operacije se najčešće odnose samo na mali deo pojave baze podataka, stoga se sastoje iz dva dela. U jednom delu se vrši selekcija podataka na koje podatke se primenuje aktivnost definisana u drugom delu operacije.

Selekcija podataka se realizuje na jedan od tri sledeća načina:

1. pomoću odnosa između podataka
2. pomoću vrednosti osobine
3. pomoću vrednosti indikatora aktivnosti (prevaziđen način).

Aktivnost definisana u drugom delu operacije predstavlja jednu od sledećih pet vrsta:

1. čitanje podataka
2. upis novih podataka
3. brisanjepostojećih podataka
4. modifikacija postojećih podataka
5. definisanje vrednosti indikatora aktivnosti (prevaziđen način).

Ako se selekcija realizuje samo po jednom činiocu prateći veze u statičkoj komponenti modela podataka (korišćenjem indikatora aktivnosti u toj strukturi), radi se o navigaciji, tj. takve selekcije izvršavaju navigacioni jezici. Za njih se kaže da su proceduralni, jer opisuju ne samo to šta se želi dobiti, nego na koji način to treba izvesti.

Jezici koji izbor podataka (selekciju) vrše po više činioca se nazivaju deklarativnim ili specifikacionim jezicima. Kod njih ne postoje procedure (neproceduralni jezici), jer formulišu samo zahtev šta se želi dobiti, ali ne i to na koji način treba to postići.

U toku sastavljanja modela podataka koristi se koncepcija apstrakcije u opisu (formulisanju) podataka. Apstrakcija je intelektualni postupak za formulisanje i predstavljanje samo opštih, bitnih i zajedničkih osobina pojedinih složenih koncepata strukturalne komponente modela podataka iz realnog sistema i zanemarivanje nebitnih osobina sa tačke gledišta ciljeva i granica informacionog sistema.

U toku razrade modela podataka koriste se sledeće tri vrste apstrakcije:

1. Klasifikacija ili tipizacija i uzorkovanje

2. Generalizacija i specijalizacija i
3. Agregacija i dekompozicija.

Klasifikuju se, recimo neke vrednosti (ili skup nekih vrednosti) u obeležja. tipovi entiteta se generalizuju u supertip (ili nadtip). Tip entiteta se može smatrati agregacijom njegovih obeležja.

3.2. RAZVOJ MODELA PODATAKA

Prvi modeli podataka nastali su u drugoj polovini šezdesetih godina. Nedostaci primećeni u postojećim modelima podataka su predstavljali podstrek za definisanje i razvoj novih, boljih modela. Razvijen je velik broj modela podataka, no u teoriji i praksi baze podataka su najznačajniji sledeći[3]:

1. mrežni model podataka,
2. hijerarhijski model podataka,
3. relacioni model podataka,
4. model entiteta i poveznika (model entiteta, atribta i poveznika),
5. funkcionalni model podataka,
6. model semantičkih hijerarhija,
7. semantički model podataka,
8. objektno-orijentisani model podataka i
9. logički model podataka.

U procesu projektovanja baze podataka koriste se model entiteta i poveznika, relacioni model podataka i objektno-orijentisani model podataka. Iz istorijskih razloga se još pominju i objašnjavaju mrežni i hijerarhijski model podataka.

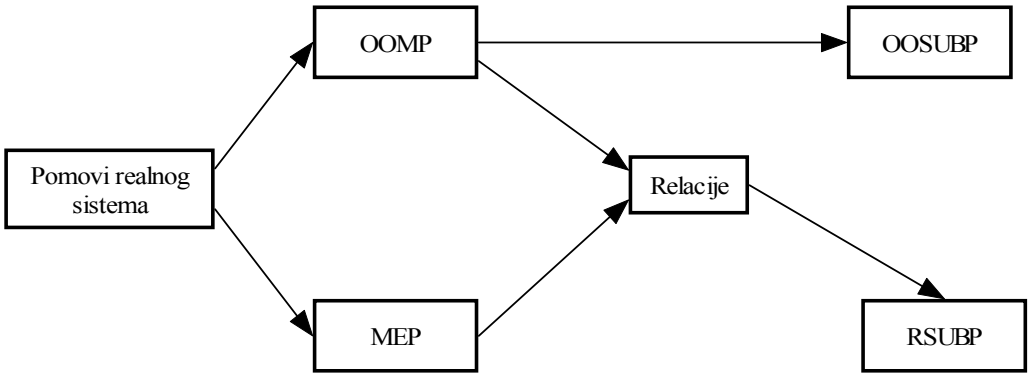
Ostali modeli podataka imaju uglavnom teoretski značaj i ukazuju na razvoj izražavanja semantike (značenja) podataka pomoću modela podataka. Semantički bogati modeli podataka imaju svoje koncepte za verno preslikavanje svih činilaca svakog realnog sistema.

3.3. SEMANTIČKO-PRAKTIČKI PRISTUP MODELIMA PODATAKA

U pragmatičkom pristupu razvoja (projektovanja i realizacije) baze podataka razlikuju se dve vrste modela podataka. U prvu grupu modela podataka spadaju oni koji omogućuju

projektovanje, a u drugu oni koji realizuju baze podataka. Za projektovanje se danas najčešće koriste model entiteta i poveznika i objektno-orijentisani model podataka. Za implementaciju projekta baze podataka izrađenog pomoću jednog od prethodno pomenuta dva modela podataka se koriste relacioni sistemi za rukovanje bazama podataka (koji realizuju relacioni model podataka) objektno-orijentisani sistemi za rukovanje bazama podataka (koji ostvaruju objektno-orijentisane modele podataka u praksi).

U procesu modeliranja (projektovanja) i implementacije baze podataka niz aktivnosti je sledeći[4]: pojmovi važni sa tačke gledišta funkcionisanja informacionog sistema se predstavljaju na izabranom jeziku za projektovanje šeme baze podataka (pomoću simbola modela entiteta i poveznika – MEP ili objektno-orijentisanog model podataka – OOMP), a posle se, po pravilu, projekat implementira pomoću relacionog sistema za upravljanje bazama podataka – RSUBP (kao što se vidi na sl. 3.1.). Postoji, naravno, i mogućnost konverzije objektno-orijentisanog modela podataka u neki objektno-orijentsani sistem za upravljanje bazama podataka – OOSUBP.



Sl. 3.1.

Na teoretskom (pa čak i na praktičnom) nivou tvrdnja o primeni dve vrste modela podataka (kako je to izloženo u prvom paragrafu ove tačke – ili na kraju prethodne tačke) nije sasvim tačna. U cilju lakšeg razumevanja ove tvrdnje treba da krenemo od važne karakteristike informacionog sistema.

Informacioni sistem je nehomogen sistem konkretnih i abstraktnih elemenata. U konkretne činioce spadaju fizički egzistirajuće stvari kao što su tehnička sredstva, standardi i korisnici, a u abstraktne elemente ubrajamo intelektualne proizvode ljudskog razmišljanja koji služe za verno odslikavanje fizičke stvarnosti, a to su dogadjaj, aktivnost i podatak. Konkretni elementi postoje, a abstraktne elemente treba smisliti, projektovati, formulisati i izraditi, a takodje i otkriti i opisati

[3] Po Moginu [19]

njihove medjusobne veze. Na osnovu prethodno izloženog se već naziru dve projekcije informacionog sistema, medjutim, pošto su podatak sa jedne strane i dogadjaj i aktivnost sa druge strane jako različite prirode, razlikuju se sledeće **tri projekcije informacionog sistema**:

1. projekcija podataka,
2. projekcija obrade i
3. projekcija okruženja.

Treba konstatovati da su saznanja i pojmovi u nekoj konkretnoj organizaciji poprilično stabilni. Česte korekcije i dogradnje su pretežno posledica nestručno izvedenih razvoja nego nestabilnosti podataka (pojmova). **Projekcija podataka** je relativno objektivna. Podatak je u velikoj meri nezavisan od ostalih činilaca informacionog sistema. Važno je istaći da je podatak u velikom stepenu nezavisan od obrade, sa kojom je možda u najtesnijoj vezi.

Projekcija obrade je u odnosu na projekciju podataka relativno nestabilna i subjektivna i, uopšteno govoreći nema takav stepen nezavisnosti od ostalih elemenata informaconog sistema kao što ima projekcija podataka. Projekcija obrade je relativno nestabilna iz razloga što korisnik može izmišljati razne vrste izveštaja i sopstvene dogadjaje bez izmene projekcije podataka. Obrada, dakle, u odnosu na podatak može biti višestruka. Promenljivost u obradi, medjutim, ne uslovljava i promenu strukture podataka, jer **logička nezavisnost podataka kao princip propisuje da promene u obradi podataka ne smeju uticati na strukturu podataka i obrnuto**.

Projekcija okruženja obuhvata korisnike, resurse za realizaciju informacionog sistema (vreme, novac, hardver i softver) i standarde, tj. realno postojeće stvari , a ne abstrakcije. Subjektivne želje korisnika, objektivne mogućnosti hardvera i softvera kao i kvazi objektivni standardi sprečavaju analitičara da realizuje po njemu najbolji mogući informacioni sistem. Promena okruženja u vremenu je neminovna pojava, ali to ne (bi) sme(lo) da izazove promenu strukture podataka, jer **princip fizičke nezavisnosti podataka formuliše stav da promene u sredstvima za obradu ne smeju uticati na strukturu podataka i obrnuto**.

Projekciju podataka, obrade i okruženja možemo shvatiti kao horizontalnu podelu činilaca informacionog sistema. Elementi ovih projekcija nisu nezavisni jer predstavljaju činioce (informacionog) sistema. Kako postoje povezanosti izmedju dva elemenata iz različite projekcije tako postoje i medjusobne veze izmedju činioca iz iste projekcije. Elementi projekcije podataka i obrade nastaju kao rezultat intelektualnog napora na raznim nivoima abstrakcije. Projekcija okruženja, pošto označava fizički postojeće resurse, poseduje samo jeadan abstrakcioni nivo:

[4] Izvor [27], strana 47.

fizički. Ostale dve projekcije pored fizičkog, imaju još dva abstrakciona nivoa, tako da se informacioni sistem posmatra i realizuje na sledeća tri abstrakciona nivoa:

1. pojmovnom (konceptualnom),
2. logičkom i
3. fizičkom nivou.

Konceptualni nivo obuhvata **sadržaj** ili **suštinu** dela realnog sistema koji se želi opisati pomoću informacionog sistema, a sâmo **rešenje** se predstavlja na logičkom i fizičkom nivou. Pomenuta tri nivoa informacionog sistema se mogu i moraju (ili morali bi) razlikovati pri razvoju samog sistema. Neispoštovanje ova tri abstrakciona nivoa može dovesti do sledeća dva tipa problema u kojima se mešaju suština i rešenje. Pri razvoju se najčešće predstavlja slika stvarnosti koja odgovara korisniku-naručiocu, a ta slika je po pravilu izvitoperena u odnosu na realni sistem. Sa druge strane razvojni tim pri izradi projekta je najčešće opterećen sa ograničenim mogućnostima sistema za rukovanje i obradu podataka, znači i rešenje će odražavati ta ograničenja, a ne stvarnu sliku dela realnog sveta. Gore opisani problemi su objektivni i naravno neizbežni. Naručioc je zainteresovan da njegovi zahtevi budu ispoštovani pri razvoju, a razvojni tim ne može zaobići ograničenja hardvera i softvera računara. Sve je to u savršenom redu, jedina je manjkavost ta, da projektanti pripremaju samo jedan plan u kojem su i babe i žabe, tj. i suština i rešenje. A zašto je to loše? Suština i koncepcija nekog poduhvata ili posla se znatno sporije menjaju nego računari ili softveri za obradu podataka. U slučaju da postoji samo jedan projekat u koji su ugradjena i ograničenja po zahtevu korisnika i ograničenja hardvera i softvera, kupovina novih računara ili prelazak na novi softver će iziskovati ponavljanje celog razvojnog posla iz početka.

Ispoštovanje tri nivoa informacionog sistema znači sastavljanje tri plana ili modela (na svakom nivou po jedan). Na konceptualnom nivou se predstavlja verna slika realnog sistema, na logičkom slika o stvarnosti koja odgovara korisniku, a na fizičkom slika koja može da se realizuje na računaru, a odgovara korisnikovom vidjenju stvarnosti. Niži nivoi informacionog sistema sve više izobličavaju pravu sliku stvarnosti, što je sasvim logično i razumljivo.

U razvoju projekcije podataka prvi je zadatak analitičara da upozna pojmove koje korisnik upotrebljava i da otkrije njihovu semantiku. Pojmovi u realnom svetu nisu nezavisni, shodno tome drugi zadatak analitičara je da ih organizuje, da sačini njihovu strukturu (da dokumentuje njihove medjusobne zavisnosti), tj. da izgradi pojmovnu strukturu ili pojmovni model podataka. **Pojmovna struktura ili pojmovni model podataka** (pojmovni nivo strukture podataka) je beskompromisna, verna i objektivna slika stvarnosti. Ili, drugim rečima:

konceptualni plan podataka je struktura podataka koja realni sistem odslikava bez ijedne greške.

Na zahtev korisnika – naručioca projektant svesno izobličava, »pokvari« pravu sliku stvarnosti. Uticaj ili zahteve korisnika možemo smatrati subjektivnim ili kvazi-subjektivnim ograničenjima kojima valja izaći u susret. Ponekad analitičar i nesvesno izobličava sliku realnog sistema. To se dogadja u slučajevima kada on ne obuhvata sve elemente ili činioce informacionih sistema, ili kada ne može doći do prave objektivne slike realnog sistema, a o tome on sam ne zna. **Logička struktura ili logički model podataka** sadrži izmenjenu pojmovnu strukturu podataka shodno definisanim ograničenjima okoline (uglavnom korisnika), ili zbog pogrešne vizije projektanta o realnom sistemu. **Logički plan podataka** je, prema tome, struktura podataka koja odražava izmenjenu sliku stvarnosti usled subjektivnih zahteva naručioca ili pogrešne percepcije (pogrešnog shvatanja) realnog sistema od strane projektanta.

U toku razvoja informacionog sistema analitičari moraju biti svesni svih ograničenja sredstava za obradu podataka (hardvera) i softvera za rukovanje i obradu podataka. Razvojni tim mora biti spreman na kompromise u vezi smeštanja i predstavljanja podataka na računaru. **Fizička struktura ili fizički model podataka** obuhvata način predstavljanja i smeštaja podataka na memorijskom medijumu računara i često se naziva **fizički plan podataka**.

Planove ili modele podataka na sva tri nivoa abstrakcije obavezno treba izraditi, i to u jedino mogućem i logičkom redosledu: prvo se realizuje konceptualni plan, zatim logički plan, a na kraju fizički plan podataka.

4. MODEL ENTITEA I POVEZNIKA – ER MODEL PODATAKA

Pod pojmom modela entiteta i poveznika se podrazumeva najpoznatija verzija modela podataka zasnovanih na entitetima i poveznicima koju je izradio P.P. Chen sredinom sedamdesetih godina. Naziv ER model potiče od skraćenice engleskih reči Entity Relationship. Ovaj model podataka predviđa grafičku predstavu rezultata procesa projektovanja baze podataka u obliku grafova i tabela. Intenzija modela se predstavlja grafovima, a ekstenzija tabelama. Bez obzira što postoje razrađene sve tri komponente ovog modela (strukturalna, itegritetna i operacijska), on se skoro isključivo koristi za projektovanje statičkog modela realnog sistema na pojmovnom (konceptualnom) nivou. Jedan od razloga je sigurno i taj što je ovaj model semantički vrlo bogat, a drugi je grafičko predstavljanje (jedan pogled vredi više od hiljadu reči...).

Bez obzira što je netrivijalno koliko u ovom modelu ima osnovnih koncepata (neki tvrde da ima dva, a drugi da postoje tri), ona je jednoznačno najrasprostranjenija vrsta modela podataka za konceptualno planiranje pojmovne strukture podataka informacionih sistema.

4.1. STRUKTURALNA KOMPONENTA MODELA ENTITETA I POVEZNIKA

Model entiteta i poveznika preslikava realni svet pomoću svega dva osnovna koncepta (kako to i sam naziv modela sugeriše): entiteta i poveznika (veze). Čak i ona grupa koja priznaje samo dva osnovna koncepta ovog modela podataka priznaje egzistenciju trećeg, po njima »gradivnog« elementa, od kojih se grade entiteti i poveznici, a taj element je obeležje.

4.1.1. Entitet, klasa entiteta i tip entiteta

Pojave realnog sveta koje želimo opisati informacionim sistemom, odnosno obuhvatiti u budućoj bazi podataka na konceptualnom nivou apstrakcije nazivamo entitetima. Tačnije entitet predstavlja sliku neke pojave realnog sistema na pojmovnom nivou apstrakcije. Entitet je nešto što se može jednoznačno identifikovati, on je “objekat posmatranja” ili “jedinica posmatranja”.

Taj termin se može koristiti za svaki realni subjekat, objekat, dogadjaj, pojavu ili apstraktni pojam.

Čovek uvek klasifikuje pojave realnog sveta. Klase pojava realnog sveta koje želimo opisati pomoću informacionog sistema na pojmovnom nivou abstrakcije krase neke osobine koje te pojave, svrstane u klase poseduju.

Pojave realnog sveta čovek uvrštava u klase putem apstrakcije koja je uvek zavisna od ciljeva samog sistema ili od namere čoveka. Klasifikacija entitea se uvek vrši po nekoj ili nekim osobinama. Te zajedničke osobine (koje se zovu obeležja) predstvaljaju osnovu za klasifikaciju. Od trenutka konstruisanja nekog apstraktnog tipa entiteta, njemu pripadajuće konkretne entitete nazivamo konkretizacijama ili pojavama tipa entiteta. Konstruisanje abstraktnog tipa entiteta znači selekciju i izbor važnih osobina klase entiteta sa tačke gledišta cilja, odnosno ciljeva informacionog sistema. To znači da u procesu konstruisanja tipa entiteta veliku ulogu igra abstrakcija, jer nikada se ne obuhvataju u tipu entiteta sve moguće osobine date klase entiteta. Konkretizacije tipa entiteta su u stvari vrednosti iz (vremenski zavisnog – promenljivog) skupa pojava pripadajućeg tipa entiteta. Tip entiteta predstavlja pojam na višem nivou apstrakcije od pojave tipa entiteta.

Pretpostavimo, u cilju ilustracije gore izloženih pojmova, da želimo voditi podatke o studentima Zoranu i Dejanu, prema tome oni su entiteti (objekti posmatranja) za nas. Odredimo li apstraktni tip entiteta STUDENT, od tog trenutka Zoran i Dejan gube svojstvo entiteta i postaju konkretizacije ili pojave tipa entiteta STUDENT. Naravno, konstruišući tip entiteta STUDENT neminovno je odrediti njegove važne osobine sa tačke gledišta informacionog sistema.

4.1.2. Obeležje, vrednost obeležja i domen

Pojave realnog sistema imaju osobine pomoću kojih se mogu opisati, klase pojava takodje. Osobine pojava realnog sveta, odnosno njihove vrednosti oogućavaju njihovu klasifikaciju, kako je već napred istaknuto. Slika osobine klase pojava realnog sveta na pojmovnom nivou abstrakcije se naziva obeležje (atribut). Drugim rečima osobine tipova entiteta na pojmovnom nivou se nazivaju obeležjima (atributima). Obeležje je takodje apstrakcija, jer čovek dodeljuje pojmovima nazive. Sve pojave (konkretizacije) istog tipa entiteta imaju barem jednu zajedničku osobinu (obeležje), na osnovu koje su svrstane u isti tip entiteta.

Kao ilustrativni primer navedimo moguća obeležja za apstraktni tip entiteta VOZILO: registarski_broj, broj_šasije, tip_vozila, boja, itd.

Svakom obeležju odgovara jedan skup mogućih vrednosti (pojava ili konkretizacija) koje pomenuto obeležje u konkretnim slučajevima može imati. Taj skup vrednosti se zove domen obeležja. Slično kao kod entiteta, ovde je obeležje apstraktni, a vrednost obeležja konkretni pojam.

Obeležje koje se ne može ili ne želi dalje dekomponovati na delove koji takođe predstavljaju obeležja smatra se elementarnim obeležjem. Niz elementarnih obeležja predstavlja složeno obeležje kojem se može dodeliti sopstveno ime ili naziv.

Entiteti, konkretizacije entiteta, obeležja i vrednosti obeležja služe za preslikavanje pojava i njenih osobina iz fizičke realnosti na konceptualnom nivou predstavljanja modela podataka, odnosno informacionog sistema. Treba, medjutim, istaći da su entitet i obeležje relativni pojmovi. Za boju po pravilu kažemo da je obeležje (napr. boja_automobila), i to s pravom, medjutim u fabrici boja i lakova "boja" treba da bude opisana pomoću raznih informacija (od kojih osnovnih boja je sastavljena i sa kojim procentima, koliki je procenat razredjivača u sastavu, kojom tehnologijom treba nanositi tu boju, itd.), tj. tu je boja entitet.

Obeležje ili grupa obeležja koja za svaku pojavu entiteta datog tipa uzima različite (jedinstvene) vrednosti nazivamo identifikatorom ili ključem (kandidatom za ključ). Ključ treba da ispunjava dva uslova:

1. uslov jedinstvenosti i
2. uslov minimalnosti.

Ako ključ zadovoljava samo prvi uslov onda je on superključ. Uslov minimalnosti formuliše zahtev da ključ ne sme imati suvišna obeležja ili, drugim rečima ključ ne sme da ima takav pravi podskup koji zadovoljava uslov jedinstvenosti. Uslov minimalnosti, naravno ima smisla kontrolisati samo u slučaju složenog ključa (ključ sastavljen od niza elementarnih obeležja).

Retko se nadje neka prirodna osobina – obeležje čije su konkretizacije (vrednosti) različite za svaku pojavu datog tipa entiteta. U tom slučaju se obično "veštački" uvodi jedno obeležje koje će imati različite vrednosti za svaku pojavu posmatranog tipa entiteta, ili se grupisanjem nekoliko atributa (obeležja) postiže da će njihove konkretizacije biti različite za svaku pojavu datog entiteta.

Na osnovu prethodnog razmišljanja se može zaključiti da postoje dve vrste identifikacije:

1. nominativna i
2. deskriptivna.

Kod nominativne identifikacije postoji jedno pogodno izabrano obeležje čija je vrednost različita za sve konkretizacije posmatranog tipa entiteta. Primer: broj_narudžbe koji jednoznačno identifikuje svaku narudžbu, ili navodjenjem jedinstvenog matičnog broja gradjana (JMBG) za identifikaciju lica.

Deskriptivna identifikacija se dobije kada se uključuje toliki broj obeležja u identifikator (ključ), da on jednoznačno identifikuje svaku pojavu tipa entiteta. Primer: deskriptivna identifikacija lica (gradjana) se realizuje pomoću identifikatora sastavljenog od sledećih obeležja: ime, prezime, datum_rodjenja, mesto_rodjenja, ime_oca, adresa.

U praksi se može desiti da jedan tip entiteta ima više takvih obeležja koja jednoznačno odredjuju njegove konkretizacije, tj. ima više kandidata za ključ. U tom slučaju se jedan od kadidata bira za ključ (primarni ključ).

4.1.3. Poveznik i tip poveznika

U prirodnom govoru retko koristimo kompletne proste iskaze (celovite proste rečenice). Rečenice ili iskaze najčešće povezujemo. Danas je već svim informatičarima poznato da saznanje ili podatak generalno ima tri svoje projekcije:

1. intenzionalnu,
2. ekstenzionalnu i
3. transverzijalnu.

Dva od gore navedena pojma, intenzija i ekstenzija su već poznata is prethodnih izlaganja.

Shodno tome intenziju tipa entiteta predstavlja naziv tipa entiteta i skup njegovih obeležja. Intenzija tipa entiteta se dobije navodjenjem svih njegovih obeležja posle navedenog naziva: $E(A_1,A_2,A_3,...,A_n)$. Oznaka za osobinu A_i potiče od skraćenice engleske reči atribute koja na našem jeziku znači obeležje. Kao što je već objašnjeno tip entiteta predstavlja samo najbitnije osobine klase entiteta realnog sveta (to je, inače, zajednička osobina svih modela, zbog abstrakcije u procesu formulisanja istih). Pretpostavimo da klasa entiteta ima skup osobina-obeležja $\{A_1,A_2,A_3,...,A_m\}$. Tada skup obeležja $\{A_1,A_2,A_3,...,A_n\}$ u intenziji tipa entiteta E predstavlja pravi ili nepravi podskup skupa osobina klase entiteta $\{A_1,A_2,A_3,...,A_m\}$.

Primer. Tip entiteta VOZILO se predstavlja na sledeći način:

VOZILO(registarski_broj, broj_šasije, tip_vozila, boja).

Klasu entiteta VOZILO danas krase i druge, možda čak i važne osobine, kao što su broj_retrovizora, broj_air_bagova, dužina_garantnog_roka, prosečna-potrošnja_goriva, itd., koje

se, kako se vidi iz tipa entiteta nisu našle u modelu (tipu) entiteta. Skup osobina za izgradnju tipa entiteta VOZILO znači pravi je podskup skupa osobina klase entiteta VOZILO.

Ekstenzija tipa entiteta odgovara skupu njegove konkretizacije (pojave). To znači da ekstenziju tipa entiteta dobijemo nabrajanjem svih njegovih pojava.

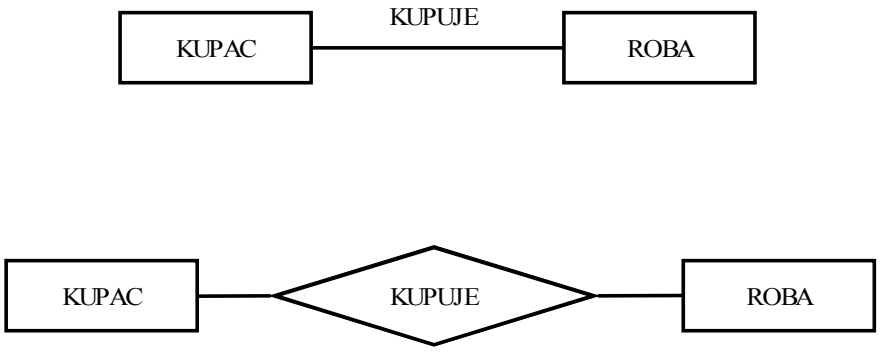
Transverzija predstavlja povezanost ili razmenu podataka izmedju pojava (entiteta) ili grupa pojava (tipova entiteta). Transverzija se može definisati na dva nivoa apstrakcije.

Povezanosti izmedju entiteta na pojavnom nivou ili odnosi izmedju pojava tipova entiteta nazivaju se vezama (to su konkretne veze) ili poveznicima. Generički odnosi izmedju tipova entiteta nazivaju se tipovima poveznika. U praktičnim realizacijama se najčešće koriste tipovi poveznika između dva tipa entiteta. Veze definisane između dva različita entiteta nazivaju se nehomogenim vezama. No dva tipa entiteta se ne moraju razlikovati: veze između dva ista tipa entiteta zovu se homogenim ili rekurzivnim vezama.

4.1.4. Intenzija ER modela i njeno grafičko predstavljanje

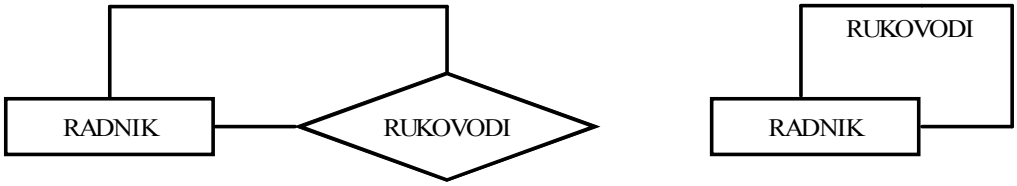
Intenziju ER modela podataka sačinjavaju tip entiteta i tip poveznika. Ova dva osnovna koncepta se grade od »osnovnog gradivnog elementa«, od obeležja. Grafičko predstavljanje statičke strukture realnog sistema u formi intenzije ER modela realizuje se pomoću ER dijagrama.

Postoje dve vrste ER dijagrama koje prikazuju dva nivoa detaljnosti: nivo detaljnosti tipa entiteta i tipa poveznika i nivo detaljnosti obeležja. Dva stila ER dijagrama nivoa detaljnosti tipa entiteta i tipa poveznika su prikazana na sl. 4.1. Tip entiteta se crta kao pravougaonik sa centriranim upisom naziva tipa entiteta. Tip poveznika se predstavlja običnim potegom koji povezuje dva tipa entiteta (a iznad potega se upisuje naziv tipa entiteta) ili se crta kao romb sa



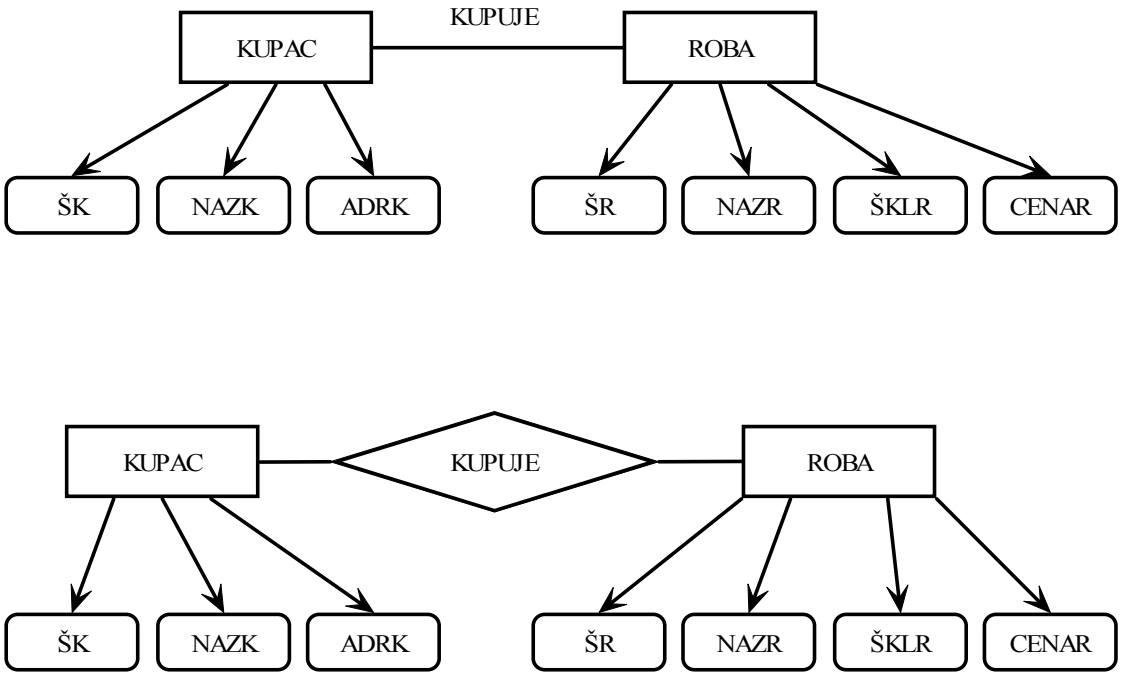
Sl. 4.1.

centriranim upisom naziva tipa poveznika. Za rekurzivne veze način predstavljanja dijagrama na ovom nivou detaljnosti je isti s tim, što postoji samo jedan tip entiteta, odnosno da tip entiteta povezuje jedan tip entiteta sa samim sobom (sl.4.2.). U tom slučaju u okviru istog tipa entiteta jedan skup pojave tipa entiteta je povezan sa drugim skupom pojave tipa entiteta koji redovno predstavlja podskup pomenutog prvog skupa. Skupovi pojave tipa entiteta imaju različite uloge u rekurzivnoj vezi.



Sl. 4.2.

Nivo detaljnosti obeležja na ER dijagramima se ispoljava u pojavljivanju samih obeležja ili domena obeležja (ako više obeležja imaju isti domen) pored tipova entiteta i tipova poveznika. To znači da ER dijagrami nivoa detaljnosti obeležja u stvari predstavljaju proširene dijagrame nivoa detaljnosti tipa entiteta i tipa poveznika. Ilustrativni primer za ovaj nivo detaljnosti se nalazi na sl. 4.3.



Sl. 4.3.

U praksi projektovanja statičke strukture realnog sistema se najčešće koriste ER dijagrami nivoa detaljnosti tipa entiteta i tipa poveznika.

4.1.5. Ekstenzija ER modela i njeno grafičko predstavljanje

Ekstenziju ER modela čine pojava tipa entiteta, skup pojava tipa entiteta, pojava tipa poveznika i skup pjava tipa poveznika. Predstavljanje ekstenzije ER modela se vrši pomoću tabela: priprema se po jedna tabela za svaki tip entiteta i tip poveznika. Tabele predstavljaju samo okvir, jer ekstenziju ER modela sačinjavaju same vrednosti (podaci) u tabelama. Takvo predstavljanje ekstenzije u potpunosti odgovara načinu predstavljanja modela podataka u relacionom modelu. Čak se, kao kod relacionog modela, i ove tabele nazivaju relacijama (entiteta i poveznika). Ilustracija predstavljanje ekstenzije ER modela se nalazi na sl. 4.4.

NARUDŽBA

Br_narudžbe	**Šifra_kupca**	**Naziv_kupca**	**Datum_nar**
17494	112	Carnex	12.09.2000.
15944	147	Pionir	15.09.2000.
13675	174	Sever	05.10.2000.
19395	174	Sever	14.07.2000.
25495	112	Carnex	15.09.2000.
...	...		...

NARUDŽBA_1

Br_narudžbe	**Šifra_kupca**	**Datum_nar**
17494	112	12.09.2000.
15944	147	15.09.2000.
13675	174	05.10.2000.
19395	174	14.07.2000.
25495	112	15.09.2000.
...	...	...

KUPAC

Šifra_kupca	**Naziv_kupca**
112	Carnex
147	Pionir
174	Sever
...	

Sl. 4.4.

4.2. INTEGRITETNA KOMPONENTA ER MODELA

Integritetna komponenta ER modela obuhvata dve važne vrste ograničenja koja se predstavljaju na konceptualnom nivou:

1. kardinalnost tipa poveznika i
2. integritet domena.

4.2.1. Kardinalnost tipa poveznika

Najvažnija crta tipa poveznika je njegova kardinalnost. Tipovi poveznika se mogu podeliti u tri grupe, što ujedno i znači da dva tipa entiteta mogu biti u sledećim odnosima: 1:1, 1:N i M:N. Primer za kardinalnost veze 1:1 je veza izmedju vozila i obaveznog osiguranja, tj. da jedno vozilo može da ima samo jedno obavezno osiguranje, a takodje jedno osiguranje se može odnositi samo na jedno vozilo. Kardinalnost tipa 1:N je prisutna u povezanosti izmedju lica (vlasnika) i vozila, tj. da jedno vozilo može imati samo jednog vlasnika, a jedno fizičko lice može imati više vozila. Primer za tip kardinalnosti M:N je veza izmedju vozila i saobraćajnih nezgoda, tj. u jednoj saobraćajnoj nezgodi mogu učestvovati više vozila, a da jedno vozilo može »preživeti« ili učestvovati u više saobraćajnih nezgoda (naravno, i nadajmo se, ne istovremeno).[1]

Postoje mišljenja da se svaki tip poveznika, bez obzira na njegovu kardinalnost, mora opisati podacima (saznanjima). Treba, medjutim, imati na umu i činjenicu da sve pojave koje se žele opisati saznanjima nazivaju entitetima, tj. ako se veza opiše podacima postaće entitet (primer robe, narudžbe i stavke narudžbe), ili kako se još naziva entitet-poveznik. U stručnoj literaturi se podacima opisan tip poveznika (entitet-poveznik) naziva gerundom. Gerund je tip entiteta koji se dobije pretvaranjem tipa poveznika u tip entiteta. Potreba za stvaranjem gerunda se javlja kada su neke pojave jednog tipa poveznika povezane sa nekim pojavama drugog tipa poveznika. U tom slučaju se povezani tipovi poveznika pretvaraju u gerunde.

Ima preporuka sa anglosaksonskog govornog područja da povezanost označava (veže) dve suštine u zavisnosti od toga sa koje strane se posmatra ta veza, pa prema tome treba dodeliti dva naziva za istu fizičku vezu. Povezanost je jedna bez obzira što povezuje dva (tipa) entiteta i zato je pogrešno posmatrati sa dve različite strane i dodeliti joj dva naziva.

Za povezanost tipa kardinalnosti M:N se kaže da ne predstavlja pravu vezu izmedju dva entiteta koje povezuje, tj. ti entiteti nisu u vezi u pravom smislu reči. Dva entiteta su samo tada povezani ako se identifikator (ključ) jednog entiteta može pronaći u drugom entitetu. Tip poveznika kardinalnosti M:N nije poželjan i zato ga treba ubacivanjem novog entiteta–poveznika razbiti na dva tipa poveznika kardinalnosti 1:N.

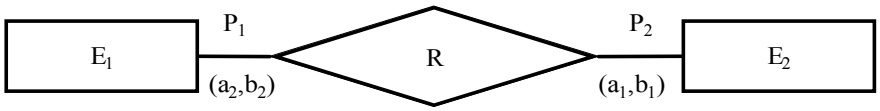
Poveznik ili povezanost je pojam koji se najteže shvata, medjutim veze predstavljaju ključ za efikasno memorisanje i rukovanje povezanih saznanja.

Semantika kardinaliteta tipa poveznika

Podrobnijom analizom opštih osobina veza izmedju različitih tipova entiteta, a takodje izmedju različitih pojava entiteta istog tipa, može se doći do opšteg opisa kardinaliteta tipa poveznika koji ima sledeći oblik:

$$R\,(E_1\,(a_2\,,b_2):E_2\,(a_1\,,b_1))$$

Grafički prikaz veze opisane prethodnim opštim opisom tipa kardinaliteta se nalazi na sl. 4.5.



Sl. 4.5.

U prethodnom izrazu R označava naziv tipa poveznika, E_1 i E_2 su nazivi entiteta, a_1 i a_2 su minimalni, a b_1 i b_2 su maksimalni kardinaliteti tipa poveznika R.

U slučaju kad se govori o kardinalitetu poveznika, kao u tački 4.2.1., obično se podrazumevaju maksimalni kardinaliteti. Shodno maksimalnim kardinalitetima (kardinalitetima grupe 1:1, 1:N i M:N) upoznatih kod definisanja poveznika vrši će se analiza mogućih kombinacija tipa i kardinalnosti, tj. minimalne i maksimalne kardinalnosti veza.

Svaka veza (svaki tip veze) definiše dva tipa preslikavanja. Na osnovu gornje opšte definicije tipa poveznika ta dva tipa preslikavanja su sledeća:

$$P_1 : E_1(a_2, b_2) \rightarrow E_2$$
$$P_2 : E_2(a_1, b_1) \rightarrow E_1$$

U preslikavanju P_1 E_1 predstavlja domen, a E_2 kodomen, pri čemu a_2 i b_2 označavaju minimalni (najmanji) i maksimalni (najveći) broj elemenata kodomena u koji se preslikava jedan element domena, respektivno. Shodno tome

$$a_2 = \min(\text{card}(P_1)) \text{ i}$$
$$b_2 = \max(\text{card}(P_1)).$$

Ponekad se minimalne vrednosti preslikavanja (napr. a_2 za preslikavanje P_1) ne označavaju na dijagramima, već se formulišu rečima (što je krajnje nepodesno za dijagramski prikaz). Za nulu vrednost minimalne kardinalnosti se kaže da je reč o parcijalnoj (opcionoj) vezi, a kad je minimalna kardinalnost jednaka jedinici, kaže se da je veza totalna (obavezna).

[1] Primeri uzeti iz [11.]

Preslikavanje se može predstaviti i u sledećem obliku:

$$P_1 : E_1(DG, GG) \rightarrow E_2$$

pri čemu je, naravno:

$$DG = \min(\text{card}(P_1)) \text{ i}$$

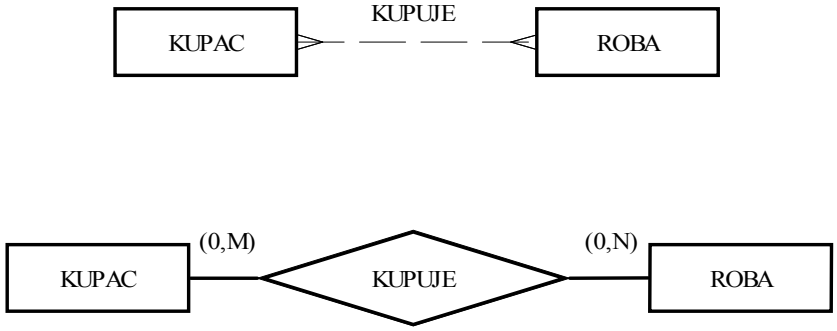
$$GG = \max(\text{card}(P_1)).$$

Kardinalitet grupe M:N

Kod ove grupe kardinaliteta su maksimalni kardinaliteti fiksni i poznati: $b_1 = N$ i $b_2 = M$. Minimalni kardinaliteti se slobodno menjaju i mogu uzimati bilo koje vrednosti. Četiri osnovna slučaja se analiziraju u ovoj grupi kardinaliteta kao što sledi u daljem tekstu.

1. **$a_1 = 0$ i $a_2 = 0$** **[** $R(E_1(0, M) : E_2(0, N))$ **]**

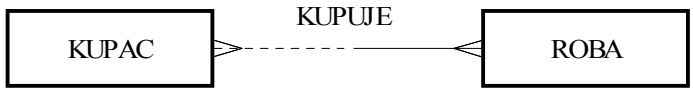
   Minimalni kardinaliteti ukazuju na činjenicu da u obe klase mogu postojati entiteti koji nisu povezani sa bilo kojim entitetom druge klase, ali, naravno, nije isključen ni taj slučaj kada su svi entiteti posmatrana dva tipa entiteta (posmatranih klasa) medjusobno povezani. Grafička predstava ove vrste veze je prikazana na sl. 4.6. Entiteti klase E_1 (KUPAC) ne moraju biti povezani ni sa jednim entitetom, mogu biti povezani sa jednim ili sa više entiteta klase E_2 (ROBA). To važi i obratno, tj. entiteti klase E_2 (ROBA) mogu biti povezani sa jednim (ali ne moraju ni sa jednim) ili više entiteta klase E_1 (KUPAC).



Sl. 4.6.

2. **a_1 =1 i a_2 =0** **[** R (E_1 (0 , M) : E_2 (1 , N)) **]**

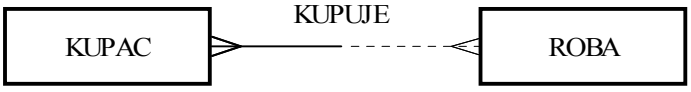
Formulacija ove veze je sedeća: svi entiteti tipa entiteta E_2 moraju biti povezani bar sa jednim entitetom klase E_1. To znači da je egzistencija entiteta u klasi E_2 je uslovljena njihovom povezanošću sa barem jednim entitetom iz tipa entiteta E_1. Entiteti iz tipa entiteta E_1 ne moraju biti povezani ni sa jednim entitetom, mogu biti povezani sa jednim ili sa više entiteta klase E_2. Grafička predstava ove veze se nalazi na sl. 4.7.



Sl. 4.7.

3. **a_1 =0 i a_2 =1** **[** R (E_1 (1 , M) : E_2 (0 , N)) **]**

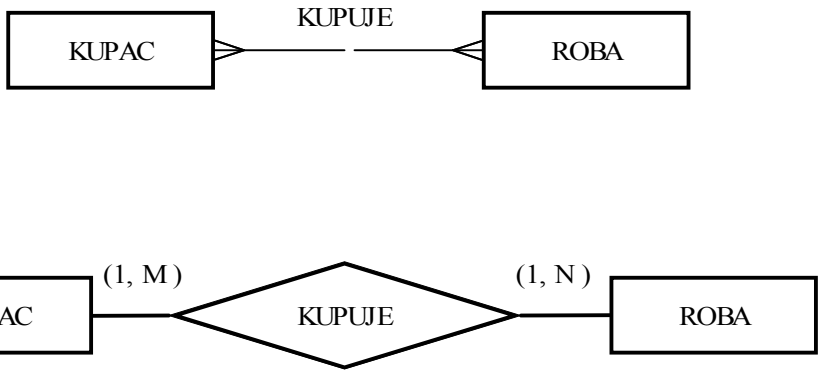
Ova veze se može opisati na sledeći način: svi entiteti tipa entiteta E_1 moraju biti povezani bar sa jednim entitetom klase E_2. To znači da je egzistencija entiteta u klasi E_1 je uslovljena njihovom povezanošću sa barem jednim entitetom iz klase entiteta E_2. Entiteti iz tipa entiteta E_2 ne moraju biti povezani ni sa jednim entitetom, mogu biti povezani sa jednim ili sa više entiteta klase E_1. Grafička predstava ove veze se nalazi na sl. 4.8.



Sl. 4.8.

4. **a_1 =1 i a_2 =1** **[** $R(E_1(1,M):E_2(1,N))$ **]**

   Ovaj četvrti tip veze grupe »više prema više« ima sledeće semantičko značenje. Svaki entitet jednog tipa entiteta 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. Konkretno, svi entiteti tipa entiteta E_1 moraju biti povezani bar sa jednim (a, naravno, mogu i sa više) entitetom klase E_2, a takodje svi entiteti klase E_2 moraju biti povezani bar sa jednim (a, naravno, mogu i sa više) entitetom tipa entiteta E_2. Grafička predstava ove veze se nalazi na sl. 4.9.



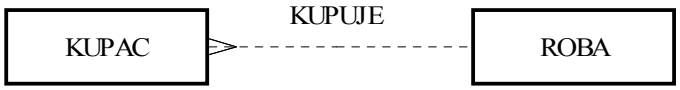
Sl. 4.9.

Kardinalitet grupe 1:N

Za ovu grupu kardinaliteta važi da su maksimalni kardinaliteti $b_1 = N$ i $b_2 = 1$. Kombinacija minimalnih kardinaliteta i u ovom slučaju daje četiri različite veze.

1. **a_1 =0 i a_2 =0** **[** $R(E_1(0,1):E_2(0,N))$ **]**

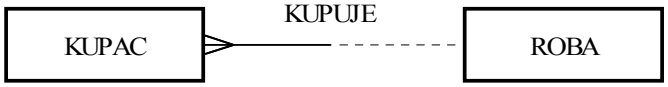
   U ovom slučaju svaki entitet tipa entiteta E_1 ne mora biti povezan ni sa jednim, a može biti povezan sa najviše jednim entitetom iz klase E_2. Entiteti klase E_2 ne moraju biti povezani ni sa jednim entitetom, mogu biti povezani sa jednim ili sa više entiteta klase E_1. Na jeziku grafike ova veza je formulisana na sl 4.10.



Sl. 4.10.

2. **a_1 =0 i a_2 =1** **[** R (E_1 (1,1) : E_2 (0, N)) **]**

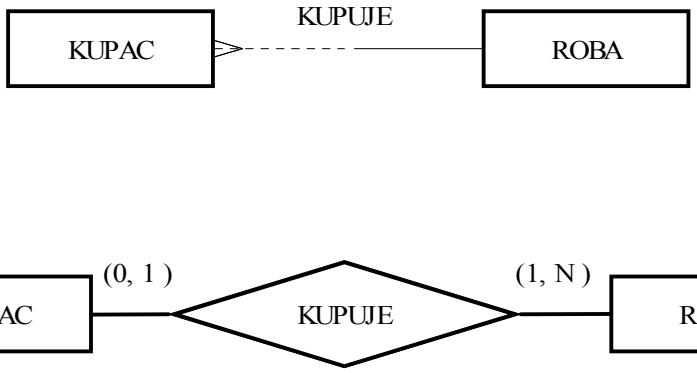
   Za ovu vezu je karakteristično, da svaki entitet klase E_1 mora biti povezan sa jednim i samo jednim entitetom iz klase entiteta E_2, a entiteti tipa entiteta E_2 ne moraju biti povezani ni sa jednim entitetom, mogu biti povezani sa jednim ili sa više entiteta klase E_1. U grafičkoj formi ova veza je predstavljena na sl 4.11.



Sl. 4.11.

3. **a_1 =1 i a_2 =0** **[** R (E_1 (0,1) : E_2 (1, N)) **]**

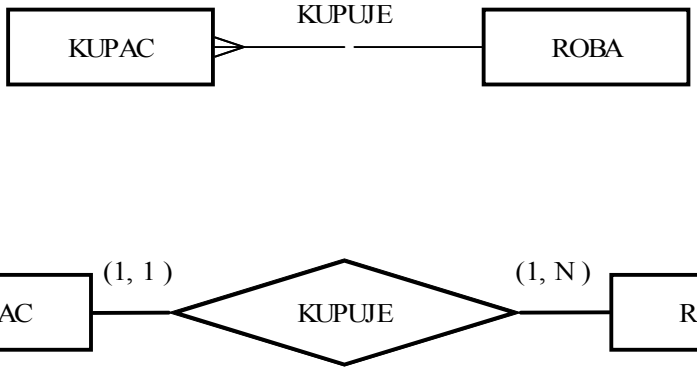
   Semantika ove veze je sledeća. Svaki entitet tipa entiteta E_1 ne mora biti povezan ni sa jednim, a može biti povezan sa najviše jednim entitetom iz klase E_2, a istovremeno svaki entiteti klase E_2 mora biti povezan sa barem jednim, a može biti povezan sa većim brojem entiteta klase E_1. Kao što se vidi sada su entiteti klase E_2 egzistencijalno zavisni od postojanja veze sa entitetima klase E_1. Odgovarajući grafik je na sl. 4.12.



Sl. 4.12.

4. **a_1 =1 i a_2 =1** **[** $R(E_1(1,1):E_2(1,N))$ **]**

Ova vrsta veze se rečima može opisati na sledeći način. Svaki entitet klase E_1 mora biti povezan sa jednim i samo jednim entitetom klase E_2, a entiteti tipa entiteta E_2 moraju biti povezani bar sa jednim (a mogu i sa više) entitetom klase E_1. Ovde su entiteti obe povezane klase su egzistencijalno uslovljeni postojanjem barem jedne konkretne veze sa entitetima druge klase. Ovu formulaciju predstavlja grafik 4.13.



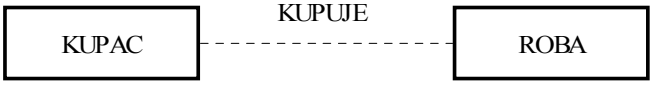
Sl. 4.13.

Kardinalitet grupe 1:1

Kod ove grupe kardinaliteta su maksimalni kardinaliteti isti i imaju vrednost $b_1 = b_2 = 1$, i kao kod prethodne dve grupe kardinaliteta kombinacije dva minimalna kardinaliteta rezultuju četiri moguće konkretne veze.

1. **a_1 =0 i a_2 =0** **[** $R(E_1(0,1):E_2(0,1))$ **]**

Ova veza predstavlja situaciju kada svaki entitet jedne klase može biti povezan sa najviše jednim entitetom druge klase. Drugim rečima svaki entitet tipa entiteta E_1 ne mora biti povezan ni sa jednim, a može biti povezan sa najviše jednim entitetom iz klase

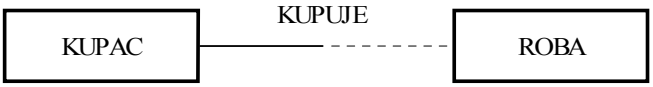


Sl. 4.14.

E_2, a to važi i za entitete klase E_2, koji ne moraju biti povezani ni sa jednim, a mogu biti povezani sa najviše jednim entitetom iz klase E_1. Slika sl. 4.14. prikazuje ovu vezu.

2. **a_1 =0 i a_2 =1** **[** $R(E_1(1,1):E_2(0,1))$ **]**

Svaki entitet klase E_1 za ovu vezu mora biti povezan sa jednim i samo jednim entitetom iz klase entiteta E_2, a konkretni entiteti tipa entiteta E_2 ne moraju, a mogu biti povezani sa najviše jednim entitetom klase E_1. Crtež na sl 4.15. prikazuje ovu vezu.

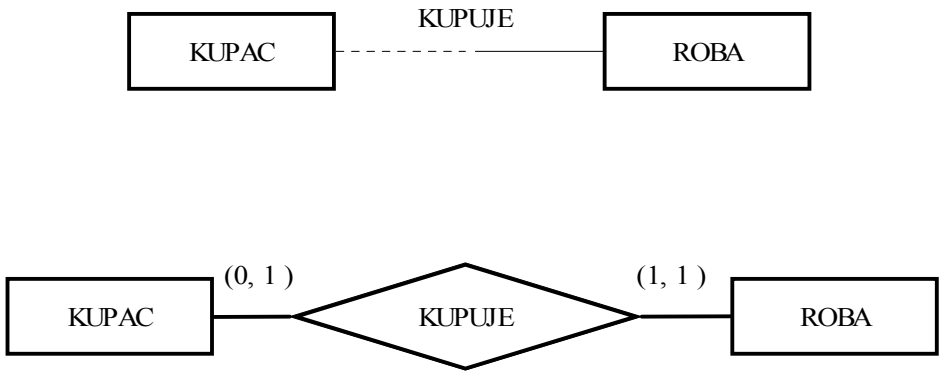


Sl. 4.15.

3. **a_1 =1 i a_2 =0** **[** $R(E_1(0,1):E_2(1,1))$ **]**

Semantiku ove veze je moguće formulisati na sledeći način. Svaki entitet tipa entiteta E_1 ne mora biti povezan ni sa jednim, a može biti povezan sa najviše jednim entitetom iz klase E_2, a istovremeno svaki entiteti klase E_2 mora biti povezan sa barem jednim i

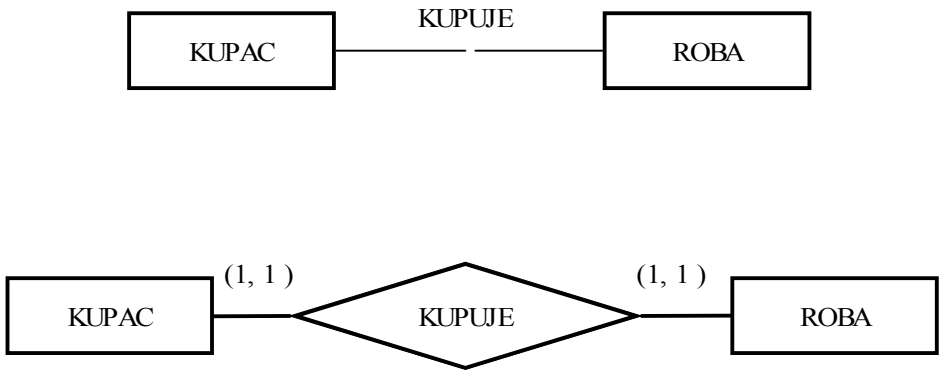
samo jednim entitetom iz klase E_1. Egzistencijalna zavisnost entiteta klase E_2 od postojanja veze sa entitetima klase E_1 je očigledna. Odgovarajući grafik je na sl. 4.16.



Sl. 4.16.

4. **a_1 =1 i a_2 =1** **[** $R(E_1(1,1):E_2(1,1))$ **]**

Poslednja vrsta veze u ovoj grupi se rečima može opisati na sledeći način. Svaki entitet klase E_1 mora biti povezan sa jednim i samo jednim entitetom klase E_2, a isto to važi i za entitete klase E_2, tj. svaki entitet klase E_2 mora biti povezani bar sa jednim i samo jednim entitetom klase E_1. Ovde je prisutna egzistencijalna uslovljenost svih entiteta obe povezane klase od postojanja barem jedne konkretne veze sa entitetima druge klase. Slikovito predstavljanje ove veze se nalazi na sl. 4.17.



Sl. 4.17.

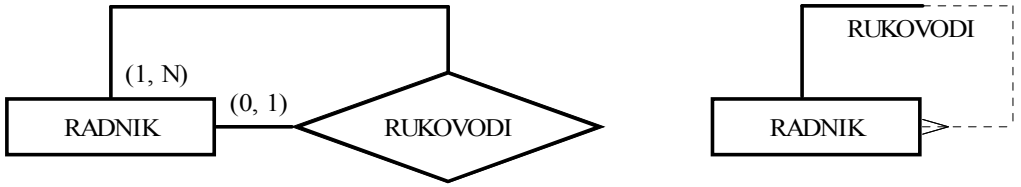
Rekurzivne veze

Odnosi izmedju različitih entiteta istoga tipa se na konceptualnom nivou apstrakcije nazivaju rekurzivnim vezama. Apstrakcijom konkretnih odnosa izmedju entiteta istoga tipa se dolazi do pojma rekurzivnog tipa poveznika. Kod rekurzivnog tipa poveznika (ako se uporedjuje sa »klasičnim tipom poveznika«) isti tip entiteta predstavlja i prvu i drugu klasu entiteta, pri čemu jedna klasa obično predstavlja podskup druge klase. Kardinaliteti rekurzivnog tipa poveznika mogu uzimati sve moguće vrednosti objašnjenih u prethodnom delu ove tačke.

Kao primer za rekurzivni tip poveznika može da posluži tip entiteta radnik i rekurzivni tip poveznika Rukovodi sa kardinalitetima:

$$\text{Rukovodi}\,(\text{Radnik} - \text{"podredjeni"}\,(0,1) : \text{Radnik} - \text{"nadredjeni"}\,(1,N))$$

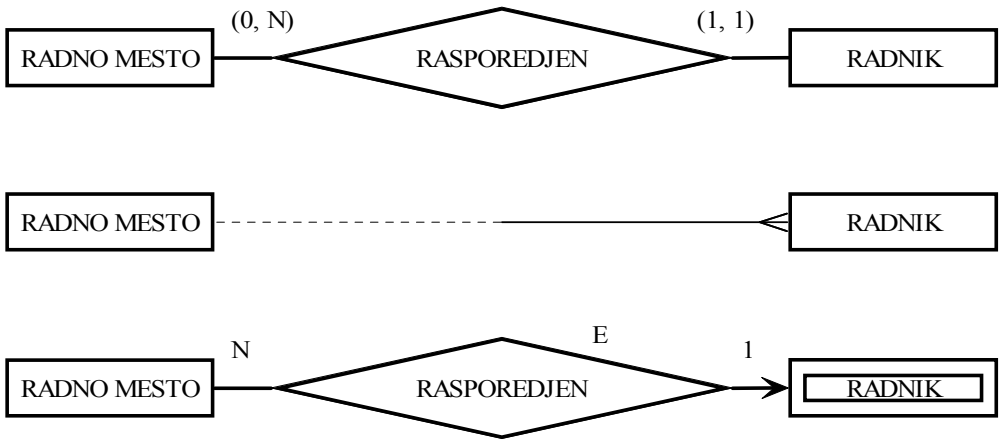
Grafička predstava rekurzivnog tipa poveznika Rukovodi je na sl. 4.18.



Sl. 4.18.

Slabi tip entiteta

Na dijagramima tipova entiteta i tipova poveznika (ER – dijagrami po engleskim rečima Entity Relationship) ponekad se zadaju samo maksimalne vrednosti kardinaliteta. Na takvim crtežima egzistencijalno zavisan tip entiteta (onaj koji ima minimalnu kardinalnost 1) se označava dvostrukim pravougaonikom i naziva se slabi tip entiteta. Linija koja povezuje tip poveznika (koji ima oznaku E za egzistencijalnu zavisnost) sa slabim tipom entiteta je usmerena ka slabom tipu entiteta. Često se i tip poveznika koji povezuje slabi tip entiteta sa drugim tipom entiteta naziva slabim tipom poveznika. Egzistencijalno nezavisni tipovi entiteta i njihovi tipovi poveznika se smatraju regularnim i ponekad se to posebno ističe u cilju razlikovanja od slabih tipova entiteta i poveznika. Takav primer ilustruje sl. 4.19. na nama svim poznatim “jezicima” grafičke predstave.

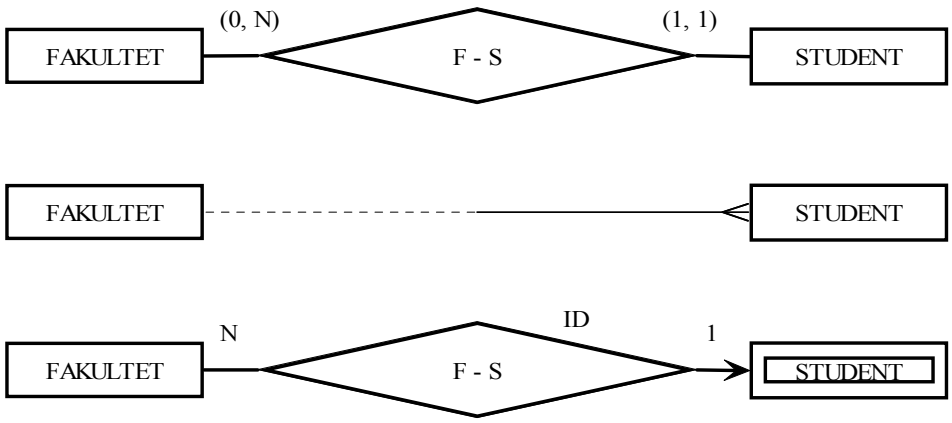


Sl. 4.19.

Identifikaciono zavisan tip entiteta

Identifikaciono zavisan tip entiteta predstavlja specijalan slučaj slabog tipa entiteta. Ako slabi tip entiteta nema svoj (jedinstveni) identifikator, nego se identifikuje pomoću identifikatora egzistencijalno nezavisnog entiteta sa kojim je u neposrednoj vezi, onda se on zove identifikaciono zavisni tip entiteta. Pojave tipa identifikaciono zavisnog entiteta se ne mogu jednoznačno identifikovati pomoću vrednosti nekog svog obeležja, već im treba pridružiti identifikator regularnog tipa entiteta sa kojim ga povezuje identifikaciono zavisan tip poveznika. Identifikaciona zavisnost uvek znači i egzistencijalnu zavisnost, a to obrnuto ne mora automatski važiti.

Pretpostavimo primer studenta koji ima za svoju identifikaciju broj indeksa na svom matičnom fakultetu, no to ga ne može identifikovati na nivou države. Na nivou države njegov identifikator može biti broj indeksa i identifikator (šifra) matičnog fakulteta koji je jedinstven na nivou države (sl. 4.20.).



Sl. 4.20.

4.2.2. Integritet domena

Integritet domena je ograničenje koje se u opštem slučaju predstavlja pomoću trojke: tip podatka, dužina podatka, uslov. Tip i dužina podatka ograničavaju vrednosti obeležja iz domena na tip i broj simbola, a uslov precizira prefinjenije zahteve. Uslov se formuliše izrazom ili funkcijom. Izrazi sadrže konstante, operacije poređenja i logičke operacije za stvaranje složenih uslova (ograničenja). Funkcije obuhvataju obeležja u svojim argument listama, tj. pored konstanti sadrže i obeležja u opisu ili realizuju neki algoritam.

Nula vrednost je specijalno ograničenje domena. Ovo ograničenje nema nikakve veze sa brojem 0. Nula vrednost označava da li obeležje može imati nedefinisanu vrednost ili vrednost »prazno«. Nula vrednost ima sledeća tri semantička značenja:

1. postojeća, ali u trenutku upisivanja nepoznata vrednost
2. postojeća i poznata prazna vrednost ili
3. neprimereno svojstvo.

4.3. OPERACIJSKA KOMPONENTA MODELA ENTITETA I POVEZNIKA

Model entiteta i poveznika (Entity Relationship Model – ER Model) se intenzivno koristi u izgradnji pojmovnog modela podataka (za logičko projektovaje baze podataka), što predstavlja statički model sistema podataka. ER model nema realizovanog sistema za upravljanje bazom podataka. Prvobitno (od strane tvorca modela P. P. Chen-a) operacijska komponenta modela nije bila razrađena. Kasnije, tokom osamdesetih godina XX. veka predloženo je više jezika za manipulisanje podacima za ovaj model. I sam Chen je definisao operacijsku komponentu ER modela koja je slična SQL jeziku relacionog modela podataka. U nastavku se nalazi opis jezika za manipulaciju podataka razrađen na Fakultetu organizacionih nauka u Beogradu.

Operacijska komponenta sadrži dve različite klase operacija, i to:

1. operacije pretraživanja (izveštavanja) i
2. operacije ažuriranja (održavanja).

Operacije pretraživanja odnosno izveštavanja se vezuju za upitne jezike danas pretežno relacionog tipa (SQL SELECT), i zbog toga se ovde pomenute operacije ne razmatraju. Njima će biti posvećena nužno potrebna pažnja u glavi 8.

4.3.1. Operacije ažuriranja baze podataka

Među operacije za ažuriranje baze podataka spadaju sledeće osnovne operacije: upiši, briši, promeni, poveži, razveži i preveži. Prve tri operacije se odnose na entitet (pojavu tipa entiteta), a naredne tri na konkretne veze (pojave tipa poveznika).

Operacije ažuriranja nad bazom podataka nisu ograničene samo na jedan tip entiteta (na sve pojave jednog tipa entiteta), već se mogu širiti kroz bazu podataka. Pri izvršavanju osnovnih operacija za ažuriranje baze podataka mora se održavati integritet baze podataka: mora se kontrolisati da li su sva modelom definisana strukturna i vrednosna ograničenja zadovoljena.

Strukturna pravila integriteta predstavljaju akcije (niz operacija) koje se izvršavaju u situacijama kada neka osnovna operacija naruši neko strukturno ograničenje. Moguća strukturna pravila integriteta (SPI) su:

1. RESTRICTED (R) – operacija koja bi narušila neko strukturno ograničenje se ne izvodi (njeni efekti se poništavaju),
2. CASCADES (C) – osnovna operacija se nastavlja nad tipom entiteta kod kojeg je integritet (strukturno ograničenje) narušen takvim operacijama koje će obezbediti zadovoljavanje (održavanje) uslova integriteta,
3. NULLIFIES (N) – oporavak narušenog integriteta se obezbeđuje upisivanjem nula (još nepoznatog) entiteta (još nepoznate pojave tipa entiteta) koji zamenjuje entitet čiji nedostatak (nepostojanje) narušio uslov integriteta, i
4. DEFAULT (D) – oporavak narušenog integriteta se obezbeđuje upisivanjem default (pretpostavljenog) entiteta (pretpostavljeno pojavljivanje tipa entiteta) koji zamenjuje entitet čiji nedostatak (nepostojanje) narušio uslov integriteta.

Semantika i sintaksa osnovnih operacija za ažuriranje (održavanje) baze podataka se definiše navođenjem naziva operacije, ulaznih parametara, strukturnog pravila integriteta (SPI) – kao specifičnog ulaznog parametra, preduslova za izvršenje operacije, efekata operacije (postuslova), a u opisu semantike (algoritma) pojedinih operacija koriste se sledeće oznake (prema Draganu Mihajloviću i Laboratorije za informacione sisteme FON-a):

- X – naziv tipa entiteta sa kojim se vrši operacija,
- x – entitet, tj. jedna pojava tipa entiteta X,
- Y – naziv tipa entiteta – kodomena posmatranog preslikavanja,
- y – entitet, tj. jedna pojava tipa entiteta Y – kodomena posmatranog preslikavanja,
- X.P – naziv preslikavanja,
- x.P – podskup preslikavanja P za dati entitet x (pojavu tipa entiteta X),

- $Y.P^{-1}$ – inverzno preslikavanje preslikavanju P,
- $y.P^{-1}$ – podskup inverznog preslikavanja P za dati entitet y (pojavu tipa entiteta Y),
- vspi – jedno od strukturnih pravila integriteta: RESTRICTED (R), CASCADES (C), NULLIFIES (N), DEFAULT (D),
- nula_y – nula entitet tipa entiteta Y,
- default_y – default entitet tipa entiteta Y.

(1) **UPIŠI_X**($x,y_1,y_2,...,y_n$)

SPI: vspi $X.P_1$, vspi $X.P_2$,..., vspi $X.P_n$

Argumenti ove operacije su entitet x (jedna pojava tipa entiteta X) i po jedan entitet iz tipova entiteta – kodomena za sva obavezna preslikavanja (za koja minimalna kardinalnost jednaka 1)

pre: ne postoji x u X (ne postoji entitet x u tipu entiteta X)
post: postoji x u X

if nisu zadati svi parametri
  then zadaj sve parametre;
end if;
if x JE_U X
  then odbij_operaciju;
end if;
for each $X.P_i$ iz SPI **do**
  if $card(y_i.P_i^{-1}) = GG(P_i^{-1})$
    then odbij_operaciju;
end for;
upiši_X(x);
for each P_i iz SPI **do**
  if y_i JE_U Y_i
    then poveži_$X.P_i(x,y)$
    else case vspi **of**
      R: odbij_operaciju;
      C: UPIŠI_$Y_i(y_i)$

poveži_X.P_i(x,y_i);

N: poveži_X.P_i(x,nula_y);

D: poveži_X.P_i(x,default_y);

end case;

end if;

end for;

end;

(2) BRIŠI_X(x)

SPI: vspi X.P_1, vspi X.P_2,..., vspi X.P_n

Argument ove operacije je jedan entitet x, tačnije samo njegov identifikator

pre: postoji x u X

post: ne postoji x u X

if x NOT JE_U X

then odbij_operaciju;

end if;

for each X.P_i **do**

if POSTOJI y JE_U x.P_i AND card(y.P_i^{-1}) = DG(P_i^{-1}) AND X.P_i JE_U SPI

then case vspi **of**

R: odbij_operaciju;

C: **for each** y iz x.P_i **do**

razveži_y.P_i^{-1}(y,x);

BRIŠI_Y(y);

end for;

N: preveži_Y.P_i^{-1}(y,x,nula_x);

D: preveži_Y.P_i^{-1}(y,x,default_x);

end case;

end if;

end for;

briši_X(x);

end;

(3) PROMENI_X(x_{1s},x_{1n})

Argumenti ove operacije su stari i novi entitet, a strukturna pravila integriteta nisu navedena, jer operacija promeni ne narušava strukturalni integritet (strukturalna ograničenja). Ovoj operaciji odgovarajuća operacija u relacionom modelu, ako se menja ključ ili spoljni ključ relacije može da dovede do narušavanja integritet baze podataka.

pre: postoji x_{1s} u X
post: entitet x_{1s} dobija nove konkretizacije obeležja (x_{1n})

(4) POVEŽI_X.P(x,y)
SPI: vspi Y

pre: ne postoji <x,y> (konkretno preslikavanje) u preslikavanju X.P
post: postoji <x,y> u preslikavanju X.P

if x NOT JE_U X
    then odbij_operaciju;
end if;
if <x,y> JE_U X.P
    then odbij_operaciju;
end if;
if card(x.P) = GG(X.P)
    then odbij_operaciju;
end if;
if card($y.P^{-1}$) = GG($Y.P^{-1}$)
    then odbij_operaciju;
end if;
if y JE_U Y
    then poveži_X.P(,y)
    else case vspi **of**
        R: odbij_operaciju;
        C: UPIŠI_Y(y)

```
            poveži_X.P(x,y);
        N: poveži_X.P(x,nula_y);
        D: poveži_X.P(x,default_y);
      end case;
  end if;
  end;
```

(5) RAZVEŽI_X.P(x,y)

SPI: vspi Y

pre: postoji <x,y> (konkretno preslikavanje) u klasi preslikavanja X.P

post: ne postoji <x,y> u klasi preslikavanja X.P

```
if <x,y> NOT JE_U X.P
   then odbij_operaciju;
end if;
if card(x.P) = DG(X.P)
   then odbij_operaciju;
end if;
if card(y.P^-1) > DG(Y.P^-1)
   then razveži_X.P(x,y)
   else case vspi of
      R: odbij_operaciju;
      C: BRIŠI_Y(y)
         Razveži_X.P(x,y);
      N: preveži_Y.P^-1(y,x,nula_x);
      D: preveži_Y.P^-1(y,x,default_x);
   end case;
end if;
end;
```

(6) PREVEŽI_X.P(x,y_1,y_2)

SPI: vspi Y(sa), vspi Y(na)

pre: postoji $<x,y_1>$ (konkretno preslikavanje) u klasi preslikavanja X.P i
ne postoji $<x,y_2>$ (konkretno preslikavanje) u klasi preslikavanja X.P

post: ne postoji $<x,y_1>$ (konkretno preslikavanje) u klasi preslikavanja X.P i
postoji $<x,y_2>$ (konkretno preslikavanje) u klasi preslikavanja X.P

if $<x,y_1>$ NOT JE_U X.P
then odbij_operaciju;
end if;
if $<x,y_2>$ JE_U X.P
then odbij_operaciju;
end if;
if card(x.P) = DG(X.P)
then odbij_operaciju;
end if;
if card($y_2.P^{-1}$) = GG($X.P^{-1}$)
then odbij_operaciju;
end if;
if card($y_1.P^{-1}$) > DG($Y.P^{-1}$)
then **if** y_2 JE_U Y
then razveži_X.P(x,y_1);
poveži_X.P(x,y_2);
else razveži_X.P(x,y_1);
POVEŽI_X.P(x,y_2); {pod uslovom vspi(na)}
end if;
else **if** y_2 JE_U Y
then poveži_X.P(x,y_2);
RAZVEŽI_X.P(x,y_1); {pod uslovom vspi(sa)}
else POVEŽI_X.P(x,y_2); {pod uslovom vspi(na)}
RAZVEŽI_X.P(x,y_1); {pod uslovom vspi(sa)}
end if;
end if;
end;

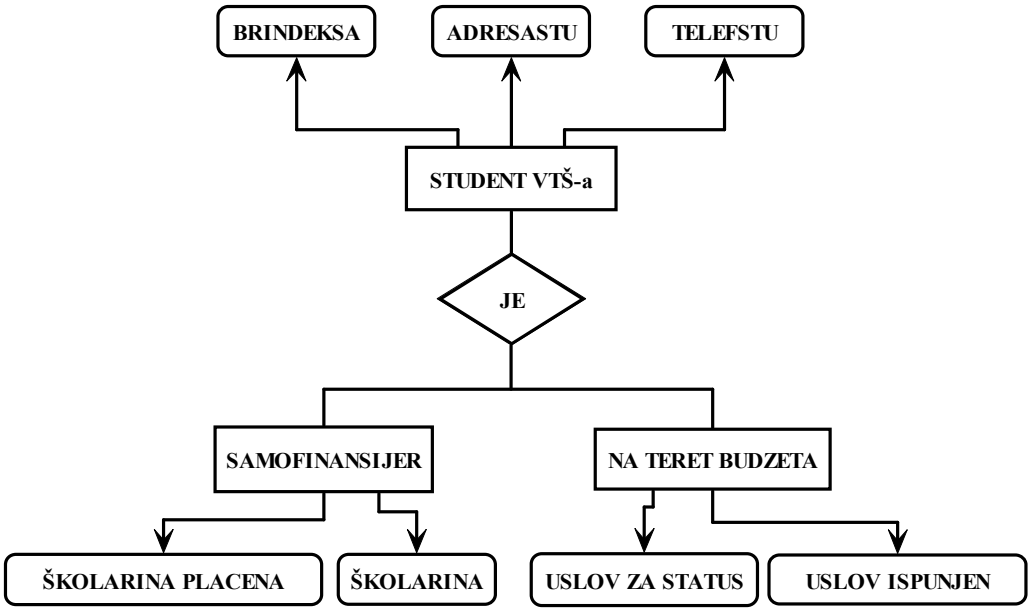
4.4.PROŠIRENJA MODELA ENTITETA I POVEZNIKA

Nakon šireg prihvatanja i primene modela entiteta i poveznika, ukazala se potreba za uvođenjem novih koncepata i postupaka u cilju obogaćenja semantike ER modela. Bogatija sematika modela obezbeđuje efikasnije opisivanje realnog sistema. Proširenje modela entiteta i poveznika obuhvata sledeće nove koncepte: superklasa, potklasa, kategorija i gerund. Za realizaciju prethodno navedenih novih koncepata su neophodne sledeće apstrakcije (postupci): generalizacija, specijalizacija, kategorizacija, agregacija i asocijacija.

Superklasa i potklasa kao novi koncepti su nastali prepoznavanjem situacije da realni entiteti iste klase ili kategorije imaju iste vrednosti nekog ili nekih obeležja. Grupisanje tih entiteta po zajedničkim obeležjima se vrši pomoću generalizacije ili specijalizacije. To je proces koji dovodi do tzv. IS_A hijerarhije entiteta. Predstavljanje grupisanjem dobijenih podskupova entiteta pomoću koncepta superklase i potklase se može realizovati nakon što se:

1. zajedničke osobine (obeležja) grupišu u superklasu i
2. specifične osobine (obeležja) se grupišu u odgovarajuće potklase.

Specifična obeležja predstavljaju svojstvenu osobinu samo dotične potklase, za entitete drugih potklasa predstavljaju neprimereno svojstvo. Geometrijska prezentacija IS_A hijerarhije se nalazi na sl. 4.21. Superklasa sadrži zajedniče osobine-obeležja (i primarni ključ, naravno), a potklasa naleđuje primarni ključ superklase i ima svoja specifična obeležja. Potklasa (preko vrednosti primarnog ključa) nasleđuje svoje osobine-obeležja od svoje superklase, zbog toga se ona ne može izdvojeno razmatrati (bez svoje superklase).



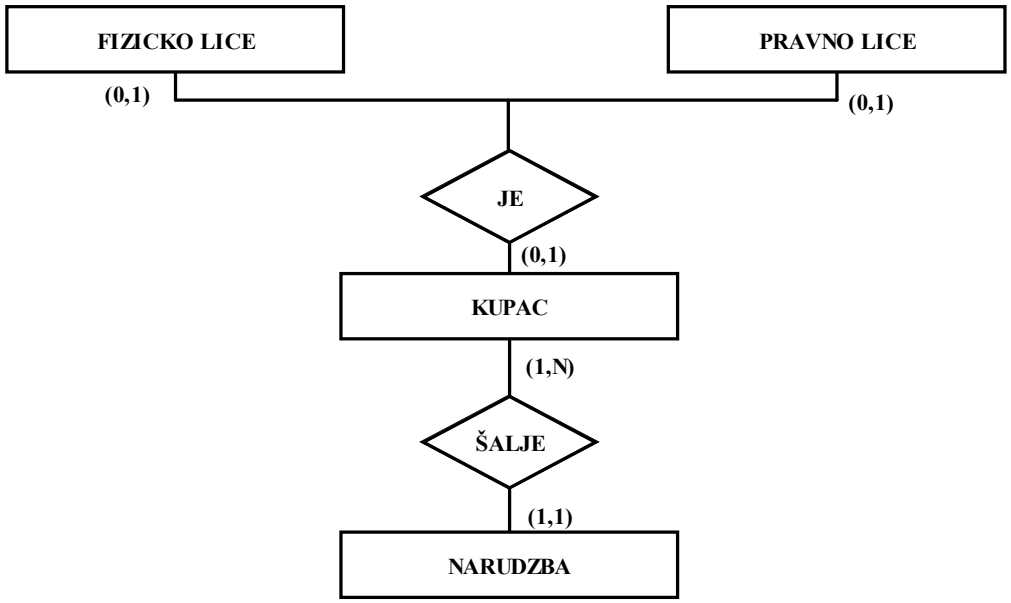
sl.4.21.

Proces izgradnje IS_A hijerarhije se zove specijalizacija, ako izgradnja krene od tipa entiteta (buduća superklasa) iz koje se izdvajaju potklase sa odgovarajućim obeležjima. Generalizacija je obrnuti proces u odnosu na specijalizaciju: kreće se od grupe različitih tipova entiteta (buduće potklase) prepoznavanjem i izdvajanjem zajedničkih obeležja (na taj način se gradi superklasa).

Kardinalnost poveznika u IS_A hijerarhiji se određuje na isti način kao za svaki drugi tip poveznika, s tim, da je kardinalnost preslikavanja sa skupa pojava potklasa na skup pojave superklase je uvek 1:1 (i to, po pravilu, se i ne označava na ER dijagramima). Kardinalnost tipa poveznika u IS_A hijerarhiji iz pravca superklase prema potklasama je specifična, jer **u okviru minimalnog kardinaliteta se posmatra broj pojava veza** (ili broj konkretnih povezanosti) između pojava dve klase, a maksimalni kardinalitet odražava broj pojava različitih potklasa koje odgovaraju jednoj konkretizaciji superklase. *Ako svakoj pojavi superklase odgovara bar jedna pojava neke od potklasa*, **minimalni kardinalitet je 1** (**totalno preslikavanje**), a *ako bar jednoj pojavi superklase ne odgovara ni jedna pojava bilo koje podklase*, **minimalni kardinalitet je 0** (**parcijalno preslikavanje**). *Ako svakoj pojavi superklase odgovara pojava iz najviše jedne potklase*, maximalni kardinalitet je 1 (disjunktno preslikavanje), a *ako bar jednoj pojavi superklase odgovaraju pojave iz više od jedne potklase*, maximalni kardinalitet je N (presečno preslikavanje). Primer na sl. 4.21. prikazuje superklasu (nadtip entiteta) STUDENT VTŠ-a sa potklasama (podtipovima entiteta) SAMOFINANSIJER i NA TERET BUDZETA. Nadtip entiteta ima kardinalnost tipa poveznika (u IS_A hijerarhiji) 1:1, tj. radi se o totalnom i disjunktnom preslikavanju. Verbalno objašnjenje je sledeće: pojava nadtipa entiteta STUDENT VTŠ-a ima za posledicu da će se pojaviti bar jedna pojava u nekoj potklasi (student je samofinansijer, ili studira na teret budžeta) – minimalni kardinalitet 1, tj. totalno preslikavanje; pojava nadtipa entiteta STUDENT VTŠ-a ima za posledicu da će se pojaviti pojava u striktno samo jednoj potklasi (student je ili samofinansijer, ili studira na teret budžeta, treće mogućnosti nema) – maksimalni kardinalitet 1, tj. radi se o disjunktnom preslikavanju.

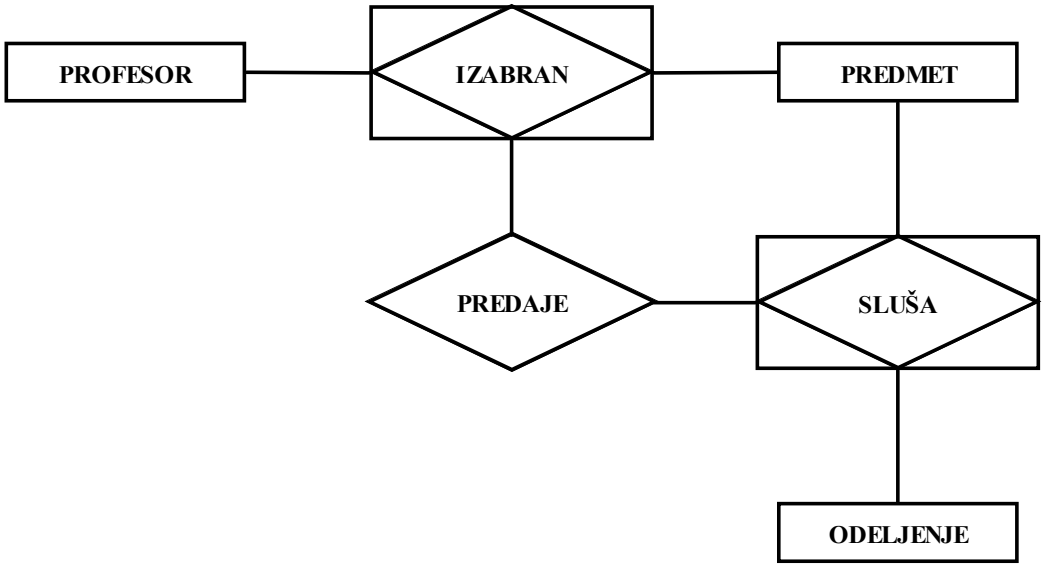
Kategorija kao novi koncept u modelu entiteta i poveznika nastaje u situacijama kada potklasa, na osnovu uloga pojava različitih tipova entiteta, neke od pojava tih, različitih tipova entiteta se obuhvata u jednu kategoriju. Tada potklasa predstavlja kategoriju. Ilustracija za grafičku predstavu kategorije se nalazi na sl.4.22. Narudžbu može da pošalje kako fizičko tako i pravno lice. KUPAC je, znači, podskup unije konkretnih fizičkih i pravnih lica i predstavlja kategoriju. Superklase su PRAVNO LICE i FIZIČKO LICE. Pošto Kupac nasleđuje obeležja od pravnog ili fizičkog lica (ali, naravno ne istovremeno od oba) i identifikatori nadtipa entiteta

PRAVNO LICE i FIZIČKO LICE su različiti, za kategoriju se uvodi novi identifikator (obeležje) – ključ kategorije (koji se često naziva surogatom).



sl.4.22.

Gerund je već pominjan u prethodnom delu teksta, i rečeno je da predstavlja vezni tip entiteta ili tip entiteta koji se dobije pretvaranjem tipa poveznika. Neki ga zovu kao mešoviti tip entitet-veza. Primena gerunda je neophodna u situacijama kada je potrebno povezati dva tipa poveznika (tada se tipovi poveznika transformišu u gerunde) ili kada treba povezati tip poveznika sa tipom entiteta (i ovde se tip poveznika pretvara u gerund). Identifikator gerunda je



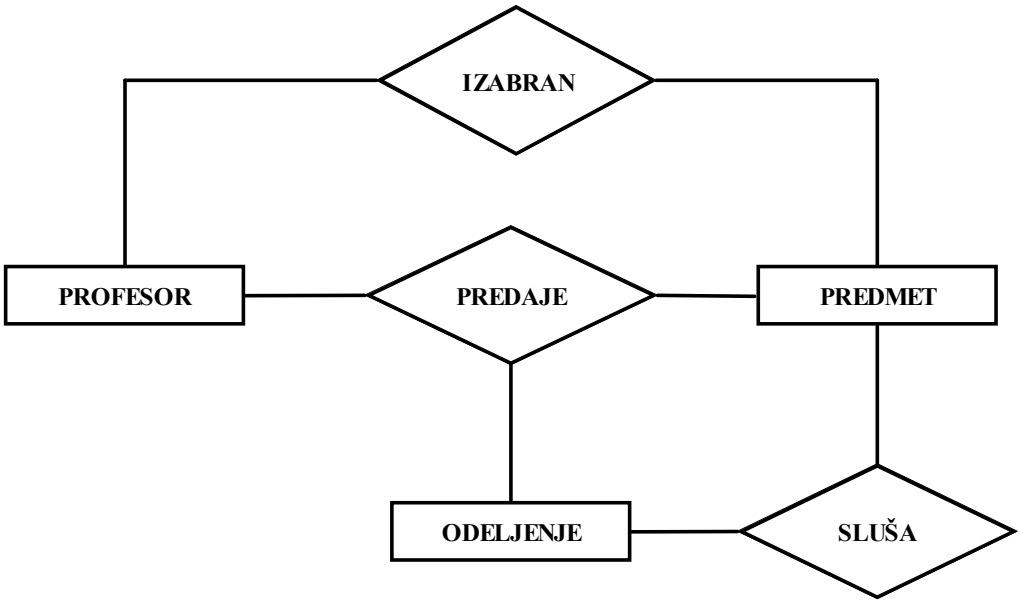
sl.4.23.

složene konstrukcije koji se formira prenošenjem identifikatora dva povezana tipa entiteta. Drugim rečima gerund nema sopstveni identifikator, već ga identifikuju identifikatori tipova entiteta koje on povezuje. Na dijagramima se gerund prikazuje rombom upisanim u pravougaonik. Primer primene gerunda se nalazi na sl. 4.23.

Semantika značenja dela ER dijagrama sa sl.4.23. je sledeća:

1. profesor **pr** je **IZABRAN** za predmet **pt**,
2. predmet **pt SLUŠA** odeljenje **od** i
3. profesor **pr** predmet **pt** za koji je **IZABRAN PREDAJE** odeljenju **od** koji taj predmet **SLUŠA**.

»Isti deo« realnog sistema predstavljen ER dijagramom, ali bez primene gerunda se nalazi na sl.4.24.



sl.4.24.

Da bi ukazali na bogatiju moć i preciznost izražavanja pomoću gerunda, analiziramo značenje dela ER dijagrama na sl.4.24.:

1. profesor **pr** je **IZABRAN** za predmet **pt**,
2. predmet **pt SLUŠA** odeljenje **od** i
3. profesor **pr** predmet **pt PREDAJE** odeljenju **od**.

Treba istaći da na sl.4.24. nema nikakvog ograničenja ko će kakav predmet predavati kojem odeljenju, tj. može se desiti da će profesor **pr** držati predavanje iz predmeta **pt**, za koji i nije izabran odeljenju **od** koji uopšte ne sluša taj predmet **pt**. Smisaono značenje dela ER dijagrama na sl.4.23. i sl.4.24. je sasvim drugo, razlike su ogromne.

4.5.IZRADA MODELA ENTITETA I POVEZNIKA (**ER** MODELA)

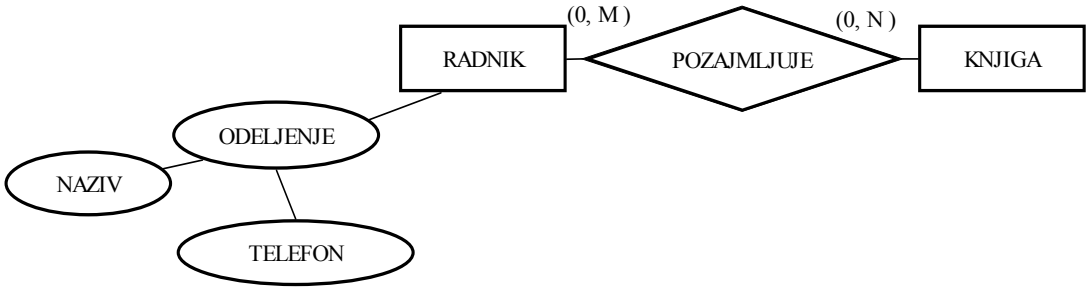
ER model služi za prikaz statičkog modela realnog sistema. Model obuhvata bitne pojave (entitete) o kojima vode evidenciju u delu realnog sveta koji se modelira i o međusobnim vezama (poveznicima) između bitnih pojava. Preduslov za izradu ER modela je upoznavanje realnog sveta što se može apsolvirati na osnovu tekstualnog ili usmenog objašnjavanja suštine (bitnih pojmova), veza između bitnih pojmova, funkcionisanja, ciljeva i pravila poslovanja, itd. Neformalne tekstove za opis realnog sistema treba pretvarati pomoću raspoloživih postupaka u koncepte ER modela. Heuristička uputstva [19] pružaju grube vodilje pri projektovanju modela realnog sistema na osnovu tekstualnog opisa istog:

1. imenice predstavljaju potencijalne tipove entiteta,
2. glagolski oblici ukazuju na moguće poveznike ili gerunde,
3. izrazi tipa »barem jedan«, »više« i slično predstavljaju moguće kardinalitete tipova poveznika ili gerunda,
4. postojanje različitih uloga entiteta jednog tipa u vezama sa entitetima drugih tipova ukazuje na
    a. potrebu uvođenja više tipova poveznika između dotičnih tipova entiteta, ili
    b. potrebu uvođenja IS_A hijerarhije,
5. veze između entiteta u okviru istog tipa entiteta signalizira potrebu izgradnje rekurzivnog tipa entiteta (uloge entiteta se kod takvih veza obavezno navode),
6. vremensko prethođenje entiteta jednog tipa u odnosu na entitete nekog drugog tipa označava egzistencijalnu zavisnost entiteta drugog tipa od entiteta prvog tipa (minimalni kardinalitet poveznika sa pravca drugog tipa biće 1),
7. »potreba takvog selektivnog povezivanja entiteta tri ili više skupova, kod kojeg u vezi mogu učesvovati samo entiteti, koji su već u nekakvoj drugoj vezi sa entitetima jednog (ili više) drugih skupova, ukazuje na neophodnost korišćenja gerunda, kao modela tih drugih veza«[2]
8. postojanje specifičnih vrednosti obeležja entiteta istog tipa signalizira potrebu uvođenja IS_A hijerarhije,
9. obeležje može pripadati samo jednom entitetu ili gerundu,
10. osobina tipa entiteta ili gerunda može postati obeležje samo ako je bitna sa tačke gledišta realizacije ciljeva informacionog sistema u razvoju.

[2] izvor [19], str. 51-52.

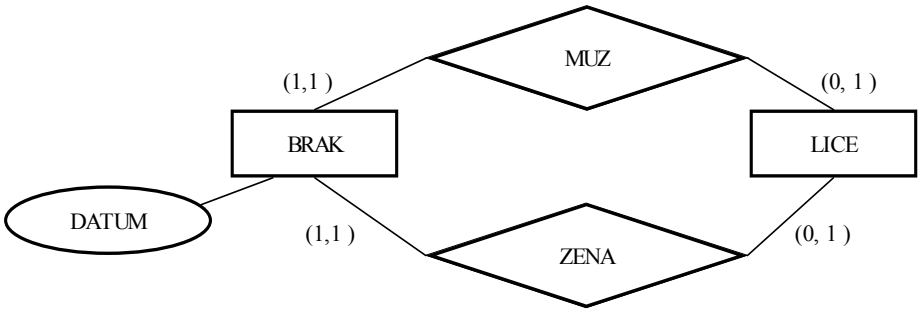
Preslikavanje pojmova iz realnog sveta nije jednoznačno. Različite predstave istog pojma ne mora označavati loše rešenje. U narednim primerima se ukazuje na relativnost predstavljanja. U prvom primeru isti pojam (ODELJENJE) može da se javlja i kao entitet i kao obeležje. ER dijagram na sl.4.25. prikazuje deo strukture nekog preduzeća, na kojem ODELJENJE predstavlja nezavisan objekat posmatranja o kojem želimo evidentirati podatke. Sl.4.26. predstavlja deo modela podataka za biblioteku nekog preduzeća. U tom modelu ODELJENJE nema taku važnu ulogu (nema svoju samostalnost – posebno nas ne interesuje, već samo kao opis radnika), da bi se o njemu vodila evidencija, već predstavlja samo fizičko okruženje radnika koji posuđuje knjige iz pomenute biblioteke.

Sl. 4.25.



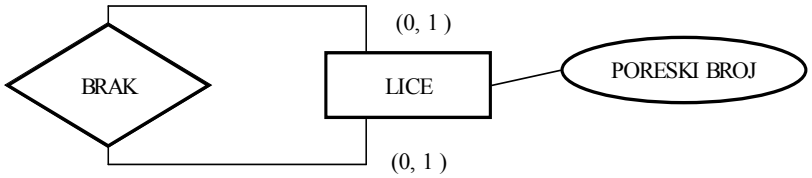
Sl. 4.26.

U drugom primeru izabrani pojam (BRAK) u zavisnosti od realnog sistema i ciljeva informacionog sistema igra takođe dvojnu ulogu: jednom kao entitet (sl. 4.27.), a u drugom slučaju kao veza (sl. 4.28.). U matičnom uredu se evidentiraju brakovi i vode se podaci o



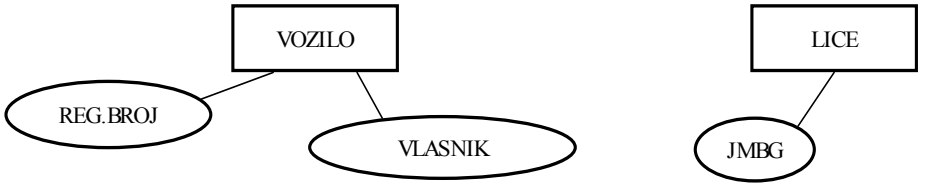
Sl. 4.27.

pojedinim sklopljenim brakovima (ne samo datum sklapanja, već i svedoci, matičar, mesto, itd.). Tu je brak naravno entitet. Za poreske vlasti (sl. 4.28.) sam brak nije interesantan sa svim prethodno nabrajanim pojedinostima: za njih brak služi samo za identifikaciju porodice, tačnije prihoda stečenih od strane članova porodice. Ovde brak služi za povezivanje lica sa odgovarajućim poreskim podacima.

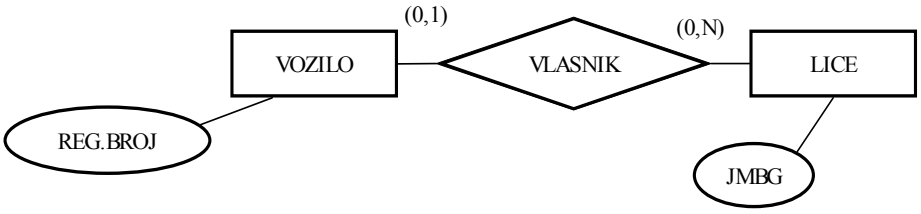


Sl. 4.28.

U trećem primeru je VLASNIK taj pojam koji se prikazuje u različitim okolnostima kao obeležje ili veza. VLASNIK se može prikazati kao obeležje u okviru tipa entiteta VOZILO (sl.4.29.), mada se to u stručnim krugovima ne smatra besprekornim rešenjem. Prihvatljivije je da se VLASNIK prikaže kao veza (sl. 4.30) između LICA i VOZILA.



Sl. 4.29.



Sl. 4.30.

5. RELACIONI MODEL PODATAKA

Relacioni model podataka je nastao početkom sedamdesetih godina, kada je Codd objavio njegov čuveni rad u vezi ovog, tada novog modela podataka. Daljem razvoju relacionog modela su doprineli takvi istaknuti naučnici kao što su Fagin, Ullmann, Rissanen Beeri, Bernstein, itd. Nedostaci u do tada primenjena dva (mrežni i hijerarhijski) modela podataka su definisali cilj istraživanja u vezi razrade novog (relacionog) modela, a to su (Mogin):

1. razgraničenje logičke od fizičke strukture baze podataka
2. obezbeđenje strukturalne jednostavnosti modela i
3. razrada deklarativnog (neproceduralnog) jezika za manipulisanje podacima.

Istraživanja u vezi razgraničavanja logičke od fizičke strukture podataka su dovela su do logičke i fizičke nezavisnosti podataka, dva fundamentalna principa koji su ispoštovani, deklarisani u relacionom modelu i predstavljaju kvalitetan napredak u odnosu na prethodne modele podataka.

Strukturalna jednostavnost u relacionom modelu je postignuta prihvatanjem relacije, odnosno dvodimenzionalne tabele kao reprezenta modela. Odnosi između podataka u različitim tabelama su realizovani podacima u okviru postojeih tabela (prosleđivanjem stranog ključ) ili pomoću nove tabele (nove relacije).

Za razvoj deklarativnog jezika za manipulisanje podacima iskorišćeni su relaciona algebra i relacioni račun. Relaciona algebra je u toku razvoja deklarativnog jezika proširen većim brojem skupovnih operatora za potrebe relacionog modela. Operacije relacione algebre se mogu predstaviti jednim logičkim iskazom relacionog računa. Deklarativni jezik za manipulisanje podacima (SQL) se bazira na relacionom računu. SQL je prvenstveno namenjen za definisaje i ažuriranje podataka i za postavljanje upita bilo koje vrste (koji se odnose na podatke u jednoj ili u više tabela). SQL nije projektovan za izvršenje obrade podataka, nego za ugradnju u jezike treće i četvrte generacije, a tamo već postoje naredbe za obradu podataka.

5.1. STRUKTURALNA KOMPONENTA RELACIONOG MODELA

Strukturalna komponenta ukazuje na elemente (koncepte) modela od kojih se gradi relacioni model. Na nivou intenzije koncepti su obeležje, šema relacije (relacija) i šema baze podataka, a na nivou ekstenzije su domen obeležja, torka, relacija i pojava baze podataka. Osnovni ili

primitivni koncepti (elementi) su samo obeležje i element domena obeležja i pomoću njih se grade, na osnovu matematičkih pravila ostali elementi modela.

Relacija se u matematici definiše kao podskup Dekartovog proizvoda domena obeležja u strogo određenom redosledu: $dom(A_1)\times dom(A_2) \times\ldots\times dom(A_k)$. Relacija je, znači, skup uređenih k-torki sa datim redosledom obeležja (atributa), tako da u $(a_1,a_2,\ldots,a_k)$ važi za svako $i\in\{1,2,\ldots,k\}$, $a_i\in dom(A_i)$. U relacionom modelu se preuzima matematička definicija relacije, s tim, da se strogi redosled obeležja iz matematičke definicije ne traži.

Relacije se predstavljaju tabelama, gde (sadržaji) zaglavlja predstavljaju (konkretna) obeležja, a redovi su konkretne relacije. Redosled pojavljivanja vrsta u tabeli je takođe nevažan.

Šema relacije je koncept na intenzionalnom nivou i predstavlja "sliku" relacije (koncepta na ekstenzionalnom nivou). Šemu relacije predstavlja dvojka kojoj se dodeljuje ime: R(A, O) gde je R naziv šeme relacije, A skup obeležja (atributa), a O je skup ograničenja. Šema relacije opisuje statičku strukturu (osobine) jednog tipa entiteta. Svaka pojava šeme relacije predstavlja jedan konkretan entitet (eksplicitnom listom vrednosti njegovih osobina) dotičnog tipa entiteta. Naravno, svaka pojava nad šemom relacije mora odgovarati celokupnom skupu ograničenja O.

Ključ šeme relacije je obeležje ili skup obeležja sa sledećim osobinama: ključ, sa jedne strane, mora biti jedinstven, tj. pomoću njegove vrednosti se moraju jednoznačno identifikovati redovi (torke, odnosno, konkretni entiteti). Sa druge strane ključ mora biti minimalan, tj. nijedan njegov pravi podskup ne sme imati osobine ključa. Šema relacije teoretski može imati više ekvivalentnih ključeva (ili kandidata za ključ), od kojih se jedan bira za primarni (a ostali postaju alternativni ključevi).

Šema baze podataka se, slično šemi relacije, predstavlja kao imenovana dvojka S(SR, MRO), gde je S naziv šeme baze podataka, SR je konačan skup šeme ralacija, a MRO je skup međurelacionih ograničenja i ograničenja, koja su posledica pravnih i drugih pravila poslovanja. Pojava baze podataka je takva pojava koja obuhvata pojavu ili pojave šema relacije iz šeme baze podataka, ali samo takve pojave mogu doći u obzir koje odgovaraju svim ograničenjima iz skupa MRO.

5.2. INTEGRITETNA KOMPONENTA RELACIONOG MODELA

Ograničenja predstavljaju suštinsku komponentu definicije šeme baze podataka. Ona su neophodna pri iniciranju i ažuriranju baze podataka i obavezno se kontrolišu pri obradi u cilju

obezbeđenja konzistentnog sadržaja (stanja) baze podataka. Ograničenja se mogu svrstavati u sledeće grupe (Mogin, str. 93):

1. ograničenja torki,
2. relaciona ograničenja i
3. međurelaciona ograničenja.

Ograničenja torki su u stvari specijalan slučaj relacionih ograničenja. Prve dve vrste ograničenja obezbeđuju da relacije predstavljaju pojave nad šemom relacije kojoj pripadaju po skupu obeležja. Relaciona ograničenja, znači, predstavljaju osnovu za kontrolu da li svaka torka (svaki red tabele) predstavlja pojavu nad šemom relacije kojoj pripada, pomoću njih se kontroliše lokalna konzistentnost.

Međurelaciona ograničenja služe za kontrolu globalne konzistentnosti baze podataka (kao pojave), tj. kontroliše se pojava baze podataka, da li odgovara svim međurelacionim ograničenjima (MRO) definisanim u šemi baze podataka. Baza podataka mora biti i lokalno i globalno konzistentna, tako se međurelaciona ograničenja mogu smatrati kao specijalan slučaj ograničenja baze podataka.

Najveći broj ograničenja se ugrađuje, odnosno proizilazi iz strukture baze podataka. Ograničenja naravno postoje i na pojmovnom nivou, tj. u modelu podataka, i, upravo ta ograničenja se u modelu podataka preko integritetne komponente relacionog modela ugrađuju (pretežno) u strukturu baze podataka.

U narednom izlaganju će biti reči o sledećim ograničenjima: ograničenja domena, integritet entiteta (ključ), funkcionalna zavisnost, tranzitivna zavisnost, višeznačna zavisnost, zavisnost spoja, zavisnost sadržavanja i uopštena ograničenja.

U praksi projektovanja baza podataka uređenost relacija se meri normalnim formama (videti kasnije). Bez obzira što je definisano šest normalnih formi, nivo kvaliteta treće normalne forme već zadovoljava projektante u najvećem broju slučajeva. Shodno tome višeznačne zavisnosti i zavisnosti spoja nećemo razmatrati jer su one vezane za četvrtu i petu normalnu formu.

Ograničenja domena označavaju činjenicu, da obeležje (atribut) može uzimati vrednsoti iz određenog domena (eventualno opsega vrednosti).

Integritet entiteta se sastoji od dva ograničenja:

1. ključ predstavlja obeležje ili skup obeležja koji jednoznačno identifikuje svaku torku nad šemom relacije (svaki red u tabeli), tj. vrednosti ključa moraju biti jedinstvene i

2. ni ključ, ni bilo koje obeležje u sastavu ključa ne sme imati nepoznatu (nul) vrednost.

Ipak treba još reći nekoliko reči o pojmu ključa. Obeležje ili grupa obeležja koja za svaku torku nad šemom relacije uzima različite vrednosti nazivamo ključem, eventualno kandidatom za ključ. Ključ treba da je minimalne konstrukcije. To drugim rečima znači, da ne postoji nijedan pravi podskup kandidata za ključ koji sam može biti kandidat za ključ (sve zavisnosti u kojima figuriše ključ moraju biti potpune).

Ako u nekoj šemi relacije postoji opisno, neidentfikujuće (neključno) obeležje koje je u drugoj šemi relacije ključ, onda ga zovemo spoljnim ključem.

Ako se ključ sastoji od jednog obeležja, onda se zove prost ključ. U slučaju da se ključ sastoji od više spoljnih ključeva, zove se složeni ključ. Ako se ključ sastoji od jednog ili više spoljnih ključeva i od barem jednog obeležja koje pripada tipu netiteta onda se naziva hijerarhijski ključ.

Naj poznatija zavisnost izmedju obeležja je **funkcionalna zavisnost**. Izraz oblika f: X→Y, ili, skraćeno X→Y predstavlja funcionalnu zavisnost, pri čemu X i Y u opštem slučaju predstavljaju skupove obeležja. Prikazana funkcionalna zavisnost se rečima može formulisati na sledeći način: X funkcionalno odredjuje Y, ili, Y funkcionalno zavisi od X. Funkcionalna zavisnost se može opisati i na drugi način: svakom elementu domena X (svakoj konkretizaciji obeležja X) se može pridružiti najviše jedan elemenat iz domena Y (jedna konkretizacija obeležja Y).

Funkcionalna zavisnost u obrnutom smeru, u opštem slučaju označava nezavisnost, odnosno u slučaju X→Y važi da Y-/->X. Primetimo da funkcionalna zavisnost definiše hijerarhiju izmedju obeležja.

Primer: Matični_broj_studenta→Ime_i_prezime_studenta, gde izmedju konkretizacija (vrednosti domena) ova dva obeležja postoji veza tipa N:1. Veći broj studenata sa istim imenom i prezimenom imaju različite matične brojeve (N), a svakom matičnom broju odgovara samo jedno ime studenta (1).

Funkcionalna zavisnost može biti jaka, kada svakoj »levoj strani«, tj. X-u mora postojati »desna strana«, tj. Y, ili slaba, kada postoje X-evi za koje ne postoje Y-i.

Primer: funkcionalna zavisnost Matični_broj_studenta→Ime_i_prezime_studenta je jaka, jer svakom matičnom broju studenta uvek pripada ime i prezime studenta, a Matični_broj_studenta→Vrsta_primljenog_kredita predstavlja slabu funkcionalnu zavisnost, pošto ima studenata koji ne primaju nikakav kredit.

Funkcionalna zavisnost može biti potpuna (full dependence) ili nepotpuna (partial dependence). Ako se na levoj strani zavisnosti nalazi obeležje, a ne skup obeležja, za zavisnost se kaže da je elementarna, u suprotnom se radi o složenoj zavisnosti. Klasifikacija funkcionalnih zavisnosti na potpune i nepotpune se primenjuje samo na složene zavisnosti. Potpuna je složena funkcionalna zavisnost u tom slučaju kad desna strana (obeležje na desnoj strani) zavisi od kompletne leve strane (celog skupa obeležja na levoj strani). To znači da odstranjivanjem bilo kojeg obeležja iz skupa obeležja na levoj strani kod potpune zavisnosti uništimo funcionalnu zavisnost. Ako izvlačenjem bilo kojeg obeležja iz skupa obeležja na levoj strani složene funkcionalne zavisnosti automatski ne dolazi i do uništavanja funkcionalne zavisnosti, onda se radi o nepotpunoj funkcionalnoj zavisnosti.

Primer: posmatrajmo relaciju POLOŽENI_ISPITI, i dve funkcionalne zavisnosti u okviru njega:

POLOŽENI_ISPITI (Matični_broj_studenta+Šifra_predmeta, ocena, ..., Naziv_predmeta)

Matični_broj_studenta+Šifra_predmeta→ocena

Matični_broj_studenta+Šifra_predmeta→Naziv_predmeta

Očigledno je da je prva funkcionalna zavisnost potpuna, pošto ni prvo ni drugo obeležje na levoj strani ponaosob ne odredjuju desnu stranu. Druga funkcionalna zavisnost je nepotpuna, jer izbacivanjem matičnog broja studenta zavisnost ostaje dalje na snazi pošto šifra predmeta odredjuje naziv predmeta.

Semantički funkcionalna zavisnost f: X→Y predstavlja vremenski promenljivu funkciju. To znači da se leva strana (domen) i desna strana (kodomen) funkcionalne zavisnosti menjaju u vremenu, odnosno da u različitim trenucima, istoj vrednosti iz domena X, f može pridružiti različite vrednosti iz domena Y. Kao primer za prethodnu konstataciju može da posluži funkcionalna zavisnost REG_BR_VOZILA→BR_OBAVEZNIH_SERVISA u kojoj se pridružena vrednost levoj strani menja u vremenu: broj obavljenih obaveznih servisa raste u toku životnog ciklusa automobila.

Funkcionalna zavisnost predstavlja intenzionalno ograničenje, jer se formuliše pomoću oznaka i pojmova koji se koriste pri definisanju intenzije relacije. Jedini način za utvrdjivanje funkcionalne zavisnosti se otkrivaju, formulišu i utvrđuju pažljivom analizom realnog sistema: pojava, njegovih obeležja i njihovih domena.

Tranzitivna zavisnost se definiše na sledeći način. Označimo skup obeležja (atributa) neke relacije sa A, i neka postoje skupovi obeležja X i Y ($X,Y \subseteq A$) i X→Y. Kaže se da Y tranzitivno zavisi od X, ako postoji takav skup obeležja $Z \subseteq A$, da X→Z i Z→Y, ali tako da ne važi Z→X i Y→Z.

Jednostavnija formulacija tranzitivne zavisnosti je sledeća. Neključno obeležje Y posmatrane relacije tranzitivno zavisi od ključa relacije X samo u tom slučaju ako ga (obeležje Y) funkcionalno odredjuje i neko neključno obeležje Z, koje je, naravno, funkcionalno zavisno od ključa relacije X.

Zavisnost sadržavanja je uslov integriteta koji se može formulisati na sledeći način: relacije r i p su relacije nad šemama relacije $R(A,O_R)$ i $P(B,O_P)$, $A_1,A_2,...A_n \in A_i$, a $B_1,B_2,...B_n \in B_i$ koje zadovoljavaju zavisnost sadržavanja, u oznaci

$$R[A_1,A_2,...A_n] \subseteq P[B_1,B_2,...B_n]$$

ako važi da je

$$(\forall t_i \in r)(\exists t_j \in p)(\forall k \in \{1,2,...n\})(t_i[A_k] = t_j[B_k]).$$

Zavisnost sadržavanja formuliše egzistencijalno ograničenje koje ukazuje na zahtev da se pre upisa torke u r mora kontrolisati postojanje barem jedne torke u p takve da je $t_i[A_k]= t_j[B_k]$ za $\forall k \in \{1,2,...,n\}$.

Referencijalni integritet kao poseban slučaj međurelacione zavisnosti sadržavanja $R[A_1,A_2,...A_n] \subseteq P[B_1,B_2,...B_n]$ poseduje poseban uslov da podskup obeležja $\{ B_1,B_2,...B_n \}$ mora biti primarni ključ u šemi relacije P.

Nul-vrednost ili **nula vrednost** predstavlja specifično ograničenje za vrednost obeležja. Naziv nula vrednost nema nikakve veze sa brojem 0, već označava sledeće tri situacije:

1. signifikantna nula (izvesno se zna, da obeležje još nema vrednost),
2. nesignifikantna nula (ne zna se da li obeležje ima neku konkretnu vrednost) i
3. neprimenjeno svojstvo (not-applicable, napr. devojačko prezime za muškarce)

5.3. OPERACIJSKA KOMPONENTA RELACIONOG MODELA PODATAKA

Strukturalna i integritetna komponenta relacionog modela služe za opis statičke strukture realnog sistema. Operacijska komponenta odražava dinamiku realnog sveta (predstavlja dinamičke karakteristike) i služi za artikulaciju jezika za manipulisanje podacima. Taj jedinstveni jezik za manipulisanje podacima služi (ili projektovan je) pre svega za upit i ažuriranje baze podataka. Jezik za manipulaciju podacima se temelji na relacionom računu i

relacionoj algebri i predstavlja visoko deklarativni jezik (naredbe sadrže ono što se želi dobiti, a ne i način kako doći do željenog rezultata).

Realaciona algebra se zasniva na matematičkoj teoriji skupova, a relacioni račun na predikatskom računu, a može se dokazati ekvivalentnost relacione algebre i relacionog računa sa tačke gledišta ekspresivne moći (Mogin, str. 166) u vezi manipulacije podacima.

Za formulisanje upita relaciona algebra poseduje dve vrste operatora:

1. standardne operacije nad skupovima u koje spadaju
   - unija,
   - presek i
   - razlika (neki tu svrstavaju i Dekartov proizvod) i
2. specijalne operacije koje sačinjavaju
   - selekcija,
   - projekcija,
   - spoj i
   - količnik.

Pošto su standardne operacije nad skupovima poznate, navešćemo ih samo ukratko. Treba, međutim, istaći da se navedene tri standardne operacije mogu primeniti samo nad relacijama koje su unijski kompatibilne (definisane su nad istim skupom obeležja).

Unija relacija $R_1(A)$ i $R_2(A)$ u oznaci $R_1(A) \cup R_2(A)$ je skup svih torki t iz R_1, R_2 ili obe relacije, odnosno:

$$R_1 \cup R_2 = \{t \mid t \in R_1 \vee t \in R_2\}$$

Presek relacija $R_1(A)$ i $R_2(A)$ u oznaci $R_1(A) \cap R_2(A)$ je skup svih takvih torki t koje su istovremeno prisutne i u R_1, i u R_2 , tj.:

$$R_1 \cap R_2 = \{t \mid t \in R_1 \wedge t \in R_2\}$$

Razlika između relacija $R_1(A)$ i $R_2(A)$ u oznaci $R_1(A) - R_2(A)$ je skup takvih torki t iz relacije R_1, koje se istovremeno ne pojavljuju u relaciji R_2 , u oznaci:

$$R_1 - R_2 = \{t \mid t \in R_1 \wedge t \notin R_2\}$$

Dekartov proizvod se može definisati i nad relacijama. Obeležja Dekartovog proizvoda predstavljaju uniju skupova obeležja relacija koja učestvuju u Dekartovom proizvodu. Stoga, ako relacije, učesnici u Dekartovom proizvodu imaju prazan skup preseka skupova svojih obeležja, onda Dekartov proizvod predstavlja relaciju. Ako presek skupova obeležja relacija u Dekartovom proizvodu nije prazan skup, proizvod bi sadržao dva puta neka obeležja (obeležja iz preseka skupova obeležja), a to u relaciji nije dozvoljeno.

Selekcija je unarna operacija, tj. definiše se nad jednom relacijom. Pomoću selekcije koja se ponekad naziva i restrikcijom, se izdvajaju torke koje odgovaraju nekom kriterijumu (zadovoljavaju neki uslov). Uslov u selekciji može da sadrži konstante, obeležja relacije koja učestvuje u selekciji, aritmetičke operatore poređenja i logičke operatore. Selekcija se definiše izrazom

$\sigma_F(R) = \{t \mid t \in R \wedge t \text{ zadovoljava } F\}$.

Semantika gornjeg izraza je sledeća: selekcija nad relacijom R prema uslovu F je relacija koja sadrži sve torke iz R koje zadovoljavaju uslov F.

Projekcija je takođe unarna operacija koja obezbeđuje mogućnost izdvajanja (izbora) onih obeležja relacije koja su interesantna za neki upit. Izraz za projekciju relacije R na skup obeležja X ima sledeći oblik

$\pi_X(R) = \{t[X] \mid t \in R\}$.

Operacija spoj predstavlja složenu binarnu operaciju sa sledećim koracima:

1. stvaranje Dekartovog proizvoda
2. izdvajanje torki iz Dekartovog proizvoda shodno definisanim uslovima
3. iz relacije dobijene u prethodnoj tački se izdvajaju po jedno od identičnih obeležja (samo kod prirodnog spoja).

Theta spoj je najuopšteniji slučaj spoja (uslov je da vrednosti označenih obeležja A_i iz jedne i B_j iz druge polazne relacije budu povezani kao $A_i \Theta B_j$, gde je $\Theta \in \{=, \text{NOT}=, >, \geq, <, \leq\}$. Ako nema uslova izdvajanja theta spoj postaje Dekartov proizvod, a u slučaju kada Θ predstavlja = spoj ima naziv equi-join (izjednačavanje). Ako iz rezultata dobijenog equi-joinom odstranimo po jedno od identičnih obeležja, dobijamo prirodni spoj (natural join).

Deljenje je složena operacija relacione algebre koja se može definisati preko jednostavnih operacija opisanih u prethodnom delu teksta na sledeći način:

$$R/P(X) = \pi_X(R) - \pi_X((\pi_X(R) \nabla P) - R).$$

U gornjem izrazu su R(X,Y) i P(Z) relacije tako da je Y unijski kompatibilan sa Z (uslov dozvaljava i mogućnost da se skupovi obeležja Y i Z budu identični, tj. Y=Z). Deljenje daje kao rezultat relaciju Q(X) koja je najveći podskup od $\pi_X(R)$ za koji važi da je $Q \nabla R$ (∇ je oznaka za Dekartov proizvod) sadržan u R.

Deljenje se, recimo, može dobro primeniti na upite tipa «koji radnici su kvalifikovani za rad na svim tipovima mašine» za dole navedenu relaciju K(W,M). Ovaj upit se definiše izrazom (deljenjem) K(W,M)/P(M):

$$K/P(W) = \pi_W(K) - \pi_W((\pi_W(K) \nabla P) - K).$$

K(W,M)

W	M
R_1	M_1
R_2	M_1
R_1	M_2
R_3	M_4
R_2	M_2
R_3	M_1
R_2	M_4
R_4	M_4
R_3	M_2

P(M)

M
M_1
M_2
M_4

$\pi_W(K)$

W
R_1
R_2
R_3
R_4

$\pi_W(K) \nabla P$

W	M
R_1	M_1
R_1	M_2
R_1	M_4
R_2	M_1
R_2	M_2
R_2	M_4
R_3	M_1
R_3	M_2
R_3	M_4
R_4	M_1
R_4	M_2
R_4	M_4

$(\pi_W(K) \nabla P) - K$

W	M
R_1	M_4
R_4	M_1
R_4	M_2

$\pi_W((\pi_W(K) \nabla P) - K)$

W
R_1
R_4

$\pi_W(K) - \pi_W((\pi_W(K) \nabla P) - K)$

W
R_2
R_3

5.4. REALIZACIJA RELACIONOG MODELA

Tok projektovanja baze podataka diktira korake u procesu razvoja informacionog sistema, a u okviru njega i u razvoju projekcije podataka istog. Aktivnosti na izradi projekcije podataka počinju razradom modela entiteta i poveznika (ER model) na pojmovnom nivou, nakon čega sledi prevođenje ER modela u relacioni model koji se može neposredno implementirati pomoću nekog od relacionog sistema za upravljanje bazama podataka (RSUBP).

Prevođenju modela podataka (pojmovni nivo) u relacioni model (logički nivo) je posvećeno posebno poglavlje u ovim beleškama, gde se opisuju tehnike pomenutog prevođenja. Primena tih tehnika označava proces nastanka (realizacije) relacionog modela koji se kasnije pretvara u bazu podataka (fizički nivo) pomoću nekog RSUBP.

6. NORMALIZACIJA

Nornalizacija je proces u kojem šeme relacije dovode na jedan kvalitetan i prihvatljiv oblik sa tačke gledišta redundanse i anomalije ažuriranja. Kod relacionih baza podataka nužno postoji redundansa podataka koju ne možemo (i ne smemo) eliminisati. Setimo se da kod relacionih baza podataka ne postoje pointeri ili poveznici koji bi fizički realizovali vezu između dve šeme relacije. Veza između dve šeme relacije se u relacionim bazama podataka ostvaruje smeštanjem ključa jedne relacije u drugu (među neključna obeležja ili kao deo ključa). Cilj procesa normalizacije je postizanje minimalno nužne redundanse podataka.

Značajniji problem od prethodnog predstavljaju anomalije ažuriranja. Postoje tri vrste anomalije, ali svaka odnjih se javlja zbog iste situacije, a to je obuhvatanje osobina (obeležja) različitih pojava (entiteta, objekata) u istu šemu relacije.

Anomalija upisa (dodavanja) se, recimo, javlja kod upisa torki za šemu relacije koja memoriše podatke o katedri sa pripadajućim predmetima i profesorima koji rade na tim predmetima. Katedru nije moguće upisati u bazu dok ne postoji barem jedan profesor koji radi bar na jednom predmetu dotične katedre.

Anomalija brisanja se u vezi šeme relacije opisane u prethodnom pasusu ispoljava u činjenici gubljenja informacija o predmetu, ako se briše profesor koji je sam radio na pomenutom predmetu, odnosno, u slučaju da se briše jedini profesor na nekoj katedri, gube se informacije i o toj katedri.

Anomalija menjanja (izmene, modifikacije) se javlja u gornjem primeru kad se modifikuju podaci o profesoru koji radi na više predmeta (eventualno na više katedri). Tada modifikaciju treba izvršiti na svim torkama gde se pojavljuju podaci o posmatranom profesoru.

Normalne forme formulišu uslov(e) za šeme relacije. One ujedno i odražavanju stanje (kvalitet strukture) šeme relacije (na pojmovnom nivou unutrašnje strukture tipa entiteta). Postoji veći broj normalnih formi, koji se označavaju u obliku xNF, u kojem se na mestu x-a nalazi broj, a NF potiče od skraćenice engleskih reči Normal Form. Da bi ukazali na jedini izuzetak u označavanju dajemo spisak svih normalnih formi od najslabije do najkvalitetnije: 0NF, 1NF, 2NF, 3NF, BCNF (Boyce-Codd-ova normalna forma), 4NF i 5NF.

Normalizacija je nastala kao matematički metod za poboljšanje nenormalizovane relacije (ili relacije niže normalne forme) u skup manjih normalizovanih relacija (u relacije željene normalne forme). Uz navođenje strogo formalnog (matematičkog) postupka normalizacije neminovno je ukazati i na potrebu primene paralelne semantičke analize (greške u shvatanju i

razumevanju fizičke realnosti i njenom strukturiranju) uz izvršavanje koraka normalizacije. Normalizacija se kao proces može realizovati metodom dekompozicije i metodom sinteze.

Dekompozicija kao postupak izvršenja normalizacije kreće od jedne nenormalizovane relacione šeme (šeme univerzalne relacije), a izvršava se postupno, u koracima. U svakom koraku se polazne relacije prevode (razbijaju, podele) u relacije koje su u višoj normalnoj formi. Uslov reverzibilnosti mora biti zadovoljen pri svakom koraku, tj. na osnovu novodobijenih šema relacije se uvek može rekonstruisati prvobitna struktura (šema univerzalne relacije). Takva dekompozicija se zove dekompozicija bez gubitaka (non-loss decomposition). (Aktivnost suprotna projekciji je spoj. Uslov spajanja dva tipa entiteta je postojanje istog obeležja u oba tipa entiteta. Uobičajeno je da se od spoja takodje traži reverzibilnost, tj. da realizuje novi spajani tip entiteta bez gubitaka informacija u njima - non-loss join.)

Sinteza kao postupak realizacije normalizacije polazi od skupa obeležja i skupa zavisnosti formulisanih na osnovu realnog sistema. Sinteza daje šeme relacije željenog kvaliteta (u zahtevanoj normalnoj formi) u jednom prolazu.

U izlaganjima se prikazuje metoda dekompozicije kao sredstvo za realizaciju normalizacije. U narednim pasusima se objašnjava proces dekomponovanja jedne šeme relacije.

Neka skup obeležja šeme relacije R_1 A, i neka postoje podskupovi $X \subseteq A$ i $Y \subseteq A$ tako, da je $X \cup Y=A$. Ograničenje da je presek podskupova X i Y ne sme biti prazan skup svakako mora biti zadovoljeno ($X \cap Y \neq \emptyset$). Kaže se da X i Y predstavljaju dekompozicioni par u šemi relacije R_1, ako iz egzistencije (postojanja) dve pojave šeme relacija R_1, recimo za pojave t_i i t_j (t_i = ext (R_1) i t_j = ext (R_1), to su redovi t_i i t_j u tabeli 6.1.) za koje je ispunjen uslov $t_i=t_j$ na preseku dva skupa $X \cap Y$, sledi da postoji takva konkretizacija tipa entiteta (red t) za koju $t=t_i$ na skupu X, a $t=t_j$ na skupu Y. Definicija se može formalno napisati na sledeći način:

$$\text{Ako } \forall t_i, t_j \text{ takvi da } t_i(X \cap Y) = t_j(X \cap Y) \text{ i } \exists t \text{ takav da } t(X) = t_i(X) \text{ i } t(Y) = t_i(Y),$$

onda X i Y predstavljaju dekompozicioni par u šemi relacije R_1.

Primer: Neka je X = {A_1, A_2, A_3, A_4} i Y = {A_3, A_4, A_5, A_6}, tj $X \cap Y$ = {A_3, A_4}, a šema relacije ima ekstenzije (pojave) prikazane u sledećoj tabeli 6.1.

Tabela 6.1.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**	**A_6**
t_1	**a**	**b**	**c**	**d**	**e**	**f**
t_2	**g**	**f**	**c**	**d**	**h**	**e**
t_3	**a**	**b**	**c**	**d**	**h**	**e**
t_4	**.**	**.**	**.**	**.**	**.**	**.**

Na osnovu prikazane ekstenzije u tabeli za tip entiteta E_1 se može zaključiti da X i Y predstavljaju dekompozicioni par, jer

1. $t_1 (X \cap Y) = t_2 (X \cap Y) = \{c, d\}$ i
2. $t_3 (X) = t_1 (X)$ i $t_3 (Y) = t_2 (Y)$

Šema relacije sadrži ponavljajuća saznanja ako sadrži takva obeležja koja nisu funkcionalno zavisna od ključa šeme relacije.

6.1. NENORMALIZOVANA ŠEMA RELACIJE

Šema relacije je u nenormalizovanoj formi (0NF) ako u njoj postoji bar jedno takvo neključno obeležje koje je funkcionalno nezavisno od ključa.

Primer: STUDENT (Br_indeksa, Ime_i_prz, ..., Šifra_pr, Naziv_pr, Datum_ispita, Ocena). Šema relacije STUDENT je prikazan i grafički na sl. 6.1. Obeležja šifra predmeta (Šifra_pr), naziv predmeta (Naziv_pr), datum ispita i ocena u nekim redovima (pojavama tipa entiteta) mogu uzimati više vrednosti. Ključ šeme relacije STUDENT je Br_indeksa.

STUDENT

Br_indeksa	**Ime_i_prz**	**...**	**Šifra_pr**	**Naziv_pr**	**Datum_ispita**	**Ocena**
E494	Marko Marković		112	Matematika	12.09.2000.	9
			147	Fizika	15.09.2000.	7
			174	Elektrotehnika	05.10.2000.	6
E495	Petar Petrović		174	Elektrotehnika	14.07.2000.	8
			112	Matematika	15.09.2000.	8
...	...		...	...	...	...

Sl. 6.1.

6.2. PRVA NORMALNA FORMA

Šema relacije se sa sigurnošću nalazi barem u prvoj normalnoj formi (1NF) ako svako neključno obeležje funkcionalno zavisno od ključa (kandidata ključa, tj. u svakoj pojavi šeme relacije za svaku konkretizaciju obeležja stoji tačno jedna vrednost).

U cilju dovodjenja šeme relacije na 1NF, nužno je iz šeme relacije odstraniti sva obeležja koja nisu funkcionalno zavisna od ključa (imaju višestruke vrednosti). Prvobitna šema relacije se dekomponuje na dve nove šeme realcije: u prvoj ostaju obeležja koja su funkcionalno zavisna od ključa u prvobitnoj šemi relacije, a ona sa višestrukim vrednostima iz prvobitne šeme relacije se smeštaju u drugu šemu relacije dopunjujući ih sa ključem prvobitne šeme relacije. Razlog tog "dopunjavanja" je dvojake prirode: bez dopune druga novodobijena šema relacije ne bi imala

(jednoznačan) ključ, a sa druge strane izgubili bi postojeću vezu između podataka (da ključ prvobitne šeme relacije nije "prosleđen" drugoj novodobijenoj šemi relacije, spajanjem se nikad ne bi magla dobiti prvobitna šema relacije). Tu dekompoziciju bez gubitaka prikazuje sl. 6.2.

Nova šema relacije je dobila naziv POLOŽENI_ISPITI i ima složeni ključ sastavljen iz dva obeležja, a ona su: Br_indeksa i Šifra_pr. Ključ prve novodobijene šeme relacije (to je u stvari ona "stara", ali "okrnjena šema relacije) je isti kao i prvobitne.

STUDENT

Br_indeksa	**Ime_i_prz**	**...**
E494	Marko Marković	
E495	Petar Petrović	
...	...	

POLOŽENI_ISPITI

Br_indeksa	**Šifra_pr**	**Naziv_pr**	**Datum_ispita**	**Ocena**
E494	112	Matematika	12.09.2000.	9
E494	147	Fizika	15.09.2000.	7
E494	174	Elektrotehnika	05.10.2000.	6
E495	174	Elektrotehnika	14.07.2000.	8
E495	112	Matematika	15.09.2000.	8
...	...	...	...	...

Sl. 6.2.

Neki stručnjaci smatraju da 0NF forma i ne predstavljaju relaciju, jer se pod pojmom relacije podrazumeva dvodimenzionalna tabela sa ad hoc atomarnim vrednostima u elementima tabele.

6.3. DRUGA NORMALNA FORMA

Šema relacije je samo tada u barem drugoj normalnoj formi (2NF) ako se nalazi u 1NF i ako je svako neključno obeležje potpuno funkcionalno zavisno od (kandidata) ključa.

Na osnovu prethodne definicije se mogu izvući dva zaključka:

1. ako šema relacije ima prost ključ, a nalazi se u 1NF onda je sigurno i u 2NF i
2. ako šema relacije nema neključna (neidentifikujuća) obeležja, takodje je sigurno, da se nalazi u 2NF.

U procesu dovodjenja šeme relacije u 2NF prvo se otkrivaju nepotpune funkcionalne zavisnosti da bi se posle te zavisnosti uklonile. Algoritam dovodjenja šeme relacije u 2NF se sastoji od tri koraka:

1. iz ključa šeme relacije se izdvajaju ona obeležja (grupe obeležja) koja i sama funkcionalno odredjuju neidentifikujuća (neključna) obeležja (ta obeležja su »kriva« za »nepotpunost« funkcionalne zavisnosti),
2. obeležja izabrana u tački 1. sa njima pripadajućim neključnim obeležjima (koja su od njih funkcionalno zavisna) se izdvajaju u posebne šeme relacija, a
3. neključna obeležja koja su u potpunoj funkcionalnoj zavisnosti od (celokupnog) ključa u prvobitnoj šemi relacije, ostaju na svom starom mestu, tj. u prvobitnoj šemi relacije.

Vezu između dve novodobijene šeme relacije predstavlja(ju) obeležje (a) izabrano(a) u tački 1. algoritma dovodjenja šeme relacije u 2NF (»krivci« za nepotpunu funkcionalnu zavisnost). Za svaku nepotpunu funkcionalnu zavisnosti se definiše po jedna nova šema relacije.

Pošto šema relacije STUDENT ima prost ključ, dovodjenje na 2NF se može prikazati na šemi relacije POLOŽENI_ISPITI. U toj šemi relacije deo ključa Šifra_pr, odnosno šifra predmeta funkcionalno odredjuje Naziv_pr , ili naziv predmeta, što znači da njih treba izdvojiti u novu šemu relacije PREDMET. Prodiskutovane šeme relacije u drugoj normalnoj formi se nalaze na sl. 6.3.

STUDENT

Br_indeksa	**Ime_i_prz**	**...**
E494	Marko Marković	
E495	Petar Petrović	
...	...	

PREDMET

Šifra_pr	**Naziv_pr**
112	Matematika
147	Fizika
174	Elektrotehnika
174	Elektrotehnika
112	Matematika
...	...

POLOŽENI_ISPITI

Br_indeksa	**Šifra_pr**	**Datum_ispita**	**Ocena**
E494	112	12.09.2000.	9
E494	147	15.09.2000.	7
E494	174	05.10.2000.	6
E495	174	14.07.2000.	8
E495	112	15.09.2000.	8
...	...	...	...

Sl. 6.3.

6.4. TREĆA NORMALNA FORMA

Šema relacije se garantovano nalazi bar u trećoj normalnoj formi (3NF), ako je u 2NF i ako nijedno neidentifikujuće (neključno) obeležje nije tranzitivno zavisno od (kandidata) ključa šeme relacije.

O unutrašnjoj strukturi šeme relacije se može tvrditi da su u njoj sva opisna (neidentifikujuća) obeležja zavisna od identifikatora (ključa – 1NF), i to potpuno funkcionalno zavisna od (celokupnog složenog) ključa (2NF), a da su funkcionalno nezavisna od ostalih opisnih obeležja (3NF).

Dovodjenje šeme relacije u treću normalnu formu se izvršava na osnovu sledećeg algoritma:

1. iz šeme relacije se izdvajaju ona neključna obeležja (grupe obeležja) koja i sama funkcionalno odredjuju druga neidentifikujuća (neključna) obeležja (ta obeležja su »kriva« za tranzitivne zavisnosti),
2. obeležja izabrana u tački 1. sa njima pripadajućim neključnim obeležjima (koja su od njih funkcionalno zavisna) se izdvajaju u posebne šeme relacija, a
3. neključna obeležja koja su u potpunoj (i netranzitivnoj) funkcionalnoj zavisnosti od ključa u prvobitnoj šemi relacije, ostaju na svom starom mestu, tj. u prvobitnoj šemi relacije.

Vezu između dve novodobijene šeme relacije predstavlja(ju) obeležje (a) izabrano(a) u tački 1. algoritma dovodjenja šeme relacije u 3NF (»krivci« za tranzitivnu zavisnost). Za svaku tranzitivnu zavisnost se definiše po jedna nova šema relacije.

Primer za dovodjenje šeme elacije na 3NF je uzet iz oblasti trgovine. Konkretno se radi o evidenciji narudžbe. Šema relacije NARUDŽBA koja je u drugoj normalnoj formi, se dekompozicijom pretvara u 3NF (sl. 6.4.).

Proces normalizacije (dovodjenje unutrašnje strukture šeme relacije na sve bolje normalne forme) je realizovan metodom dekompozicije, podelom prvobitnog na dve ili više nove šeme relacije. Normalizacija omogućuje postizanje sve savršenije interne strukture šeme relacije.

Dekompozicija se ostvaruje projekcijom (odstranjivanjem) obeležja koja nisu funkcionalno zavisna od ključa (1NF), koja su nepotpuno (delimično) zavisna od složenog ključa (2NF), ili, pak zavise i od drugih opisnih (neključnih) obeležja pored postojanja njihove funkcionalne zavisnosti od ključa (3NF) u nove šeme relacije. Postavlja se, medjutim, specijalan zahtev za dekompoziciju: mora predstavljati reverzibilan proces, odnosno da se na osnovu novodobijenih šema relacija uvek može rekonstruisati prvobitna struktura. Takva dekompozicija se zove dekompozicija bez gubitaka (non-loss decomposition).

NARUDŽBA

Br_narudžbe	**Šifra_kupca**	**Naziv_kupca**	**Datum_nar**
17494	112	Carnex	12.09.2000.
15944	147	Pionir	15.09.2000.
13675	174	Sever	05.10.2000.
19395	174	Sever	14.07.2000.
25495	112	Carnex	15.09.2000.
...	...		...

NARUDŽBA_1

Br_narudžbe	**Šifra_kupca**	**Datum_nar**
17494	112	12.09.2000.
15944	147	15.09.2000.
13675	174	05.10.2000.
19395	174	14.07.2000.
25495	112	15.09.2000.
...	...	...

KUPAC

Šifra_kupca	**Naziv_kupca**
112	Carnex
147	Pionir
174	Sever
...	

Sl. 6.4.

Aktivnost suprotna projekciji je spoj. Uslov spajanja dve šeme relacije je postojanje istog obeležja u obe šeme relacije. Uobičajeno je da se od spoja takodje traži reverzibilnost, tj. da realizuje novu spojenu šemu relacije bez gubitaka informacija u njima (non-loss join).

7. PREVOĐENJE ER MODELA U RELACIONI MODEL

Projektovanje baze podataka se realizuje na pojmovnom (konceptualnom) i logičkom (neki ga zovu implementacioni) nivou. Konceptualno projektovanje se vrši pomoću ER modela. Današnji sistemi za upravljanje bazama podataka su, međutim, zasnovani na relacionom modelu. Prevođenje ER modela u relacioni model je tako »nužno zlo« u današnjoj praksi.

ER model je semantički bogatiji od relacionog modela. Prevođenje između koncepata strukture jednog i drugog modela zbog prethodno navedene činjenice ne može biti 1:1, tj svakom konceptu strukture jednog modela ne mora odgovarati jedan koncept strukture drugog modela. Isto važi i za operacije i za ograničenja u dva pomenuta modela.

Postoje razni pristupi za prevođenje ER u relacioni model. Neki od njih su previše teoretskog, a drugi, pak, neadekvatno »slobodne« formulacije. Postupak opisan u ovom delu sledi stil formulisan u [28. Mihajlović].

7.1. PREVOĐENJE ENTITETA

Prevođenje entiteta u relaciju predstavlja relativno jednostavan zadatak koji se može formalizovati i automatizovati. Evo opisa tih kombinacija koje mogu nastati kod prevođenja tipa entiteta.

E1. Svaki »običan« tip entiteta kao i gerund iz ER dijagrama postaje šema relacije. Naziv tipa entiteta (gerunda) postaje naziv šeme realcije, a obeležja tipa entiteta (gerunda) postaju obeležja šeme relacije. Identifikator tipa entiteta (gerunda) postaje ključ šeme relacije.

E2. Slabi tip entiteta će postati šema relacije. Naziv slabog tipa entiteta će postati naziv šeme relacije, a obeležja slabog tipa entiteta će postati obeležja šeme relacije. Ključ šeme relacije (slabog tipa entiteta) će postati identifikator slabog tipa entiteta, a ključ šeme relacije nadređenog tipa entiteta se pojavljuje kao obično neključno obeležje u šemi relacije slabog tipa (kao kod prevođenja veza sa kardinalnošću (1,1) : (0,N) – videti u kasnijem izlaganju) – predstavljajući time spoljni ključ u pomenutoj relaciji. Egzistencijalna zavisnost propisuje uslov da spoljni ključ ne sme imatu nul vrednost u šemi relacije slabog tipa entiteta.

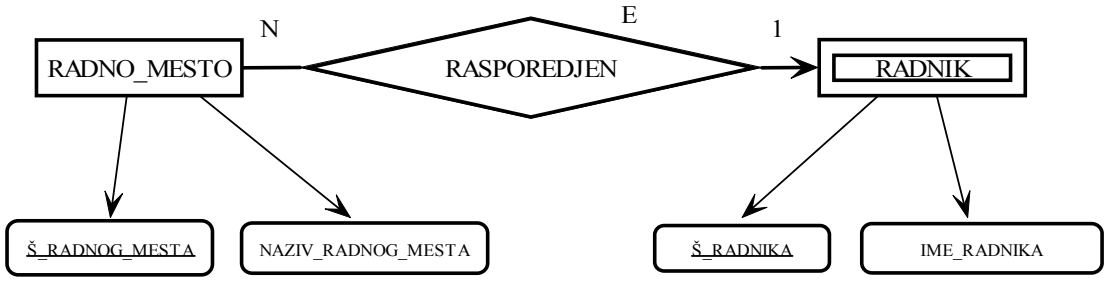
E3. Identifikacioni tip entiteta će postati šema relacije. Naziv identifikacionog tipa entiteta će postati naziv šeme relacije, a obeležja identifikacionog tipa entiteta će postati obeležja šeme relacije. Ključ šeme relacije (identifikacionog tipa entiteta) će postati identifikator nadređenog tipa entiteta zajedno sa obeležjima identifikacionog tipa entiteta koji zajednički jednoznačno određuju (identifikuju) svaku pojavu slabog tipa entiteta.

E4. Tip entiteta koji odgovara nadtipu (superklasi) postaje šema relacije. Naziv nadtipa postaje naziv šeme relacije. Obeležja nadtipa postaju obeležja šeme relacije, a identifikator nadipa postaje ključ šeme relacije.

E5. Tip entiteta koji odgovara podtipu (potklasi) postaće šema relacije. Naziv podtipa će postati naziv šeme relacije, a obeležja podtipa, obeležja šeme relacije. Šema relacije (koja odgovara podtipu) preuzeće identifikator nadtipa (posmatranog podtipa) koji će postati njen ključ.

E6. Kod hijerarhije tipa kategorija prevođenje nadtipa se vrši po opisu u tački 4. Razlika u odnosu na prevođenje hijerarhije tipa generalizacija/specifikacija (opisanog u tačkama 4 i 5) se javlja kod prevođenja potklase (podtipa). Kategorija predstavlja potklasu u istoimenoj hijerarhiji, ali ona ima poseban, novokonstruisani (u procesu formiranja kategorije) ključ. Prevođenje kategorije se vrši po tački 1.

Za primer prevođenja slabog (egzistencijalno zavisnog) tipa entiteta uzmimo naveden primer iz glave 4 koji se nalazi na sl. 7.1.



Sl. 7.1.

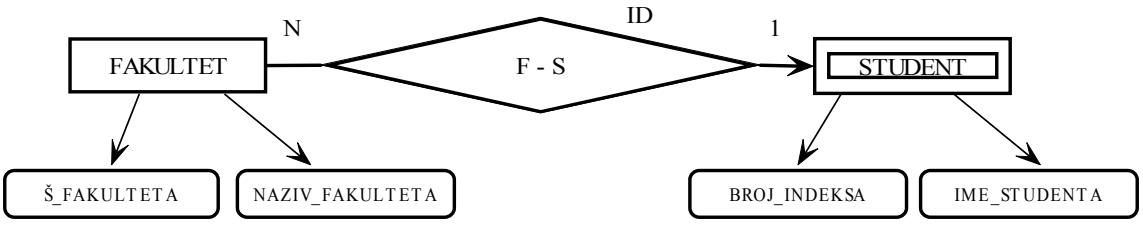
Šeme relacije se mogu dobiti primenom pravila E1 i E2, a imaju oblik kao što sledi:

RADNO_MESTO (Š_RADNOG_MESTA, NAZIV_RADNOG_MESTA)

RADNIK (Š_RADNIKA, Š_RADNOG_MESTA, IME_RADNIKA)

Egzistencijalna zavisnost propisuje uslov da spoljni ključ u šemi relacije RADNIK (Š_RADNOG_MESTA) ne sme imati null vrednost. Rečima to znači da radnik ne može biti upisan u bazu podataka ako nije raspoređen na neko konkretno radno mesto.

Identifikaciono zavisan tip entiteta se prevodi nešto drugačije od slabog tipa entiteta. Primer iz glave 4. se nalazi na sl. 7.2.



Sl. 7.2.

Šeme relacije koje odgovaraju primeru sa sl. 7.2. su sledeće:

FAKULTET (Š_FAKULTETA, NAZIV_FAKULTETA)

STUDENT (Š_FAKULTETA, BROJ_INDEKSA, IME_STUDENTA)

Slično kao u slučaju slabog (egzistencijalno zavisnog) tipa entiteta, ni identifikaciono zavisan tip entiteta se ne može upisati u bazu podataka ako njegov nadređeni tip entiteta ne postoji (kod slabog tipa entiteta taj uslov je obezbeđen zabranom nul vrednosti za spoljni ključ, a ovde se ta zabrana odnosi na delove ključa tj. na ključna obeležja) . Ovde će ključ šeme relacije nadređenog tipa entiteta postati deo složenog (hijerarhijskog) ključa šeme relacije identifikaciono zavisnog tipa entiteta.

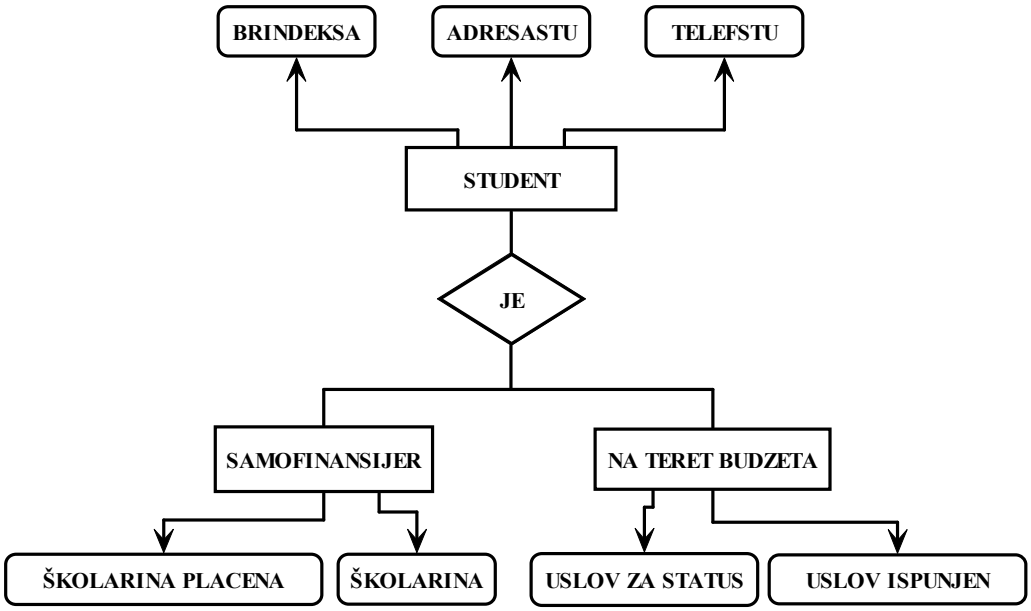
Za strukturu supertip-podtip se odnose pravila E4 i E5, i za primer sa sl. 7.3. šeme relacije su sledeće:

STUDENT (BRINDEKSA, ADRESASTU, TELEFSTU)

SAMOFINANSIJER (BRINDEKSA, ŠKOLARINA PLAĆENA, ŠKOLARINA)

NA TERET BUDZETA (BRINDEKSA, USLOV ZA STATUS, USLOV ISPUNJEN)

Za objašnjenje načina prevođenja **gerunda i kategorije** nećemo navoditi primere, jer bi se oni sveli na one koji su objašnjeni u okviru prethodnih prevođenja. Gerund se prevodi kao običan tip entiteta, a u hijerarhiji kategorije učesnici nadtip i kategorija se prevode po pravilu E4 (nadtip) i E1 (kategorija).



Sl.7.3.

7.2.PREVOĐENJE POVEZNIKA

Prevođenje koncepta poveznika ER modela može, a ne mora rezultovati novu šemu relacije. Prostiranjem ključeva se u odgovarajućim slučajevima takođe može predstaviti vezu u ER modelu.

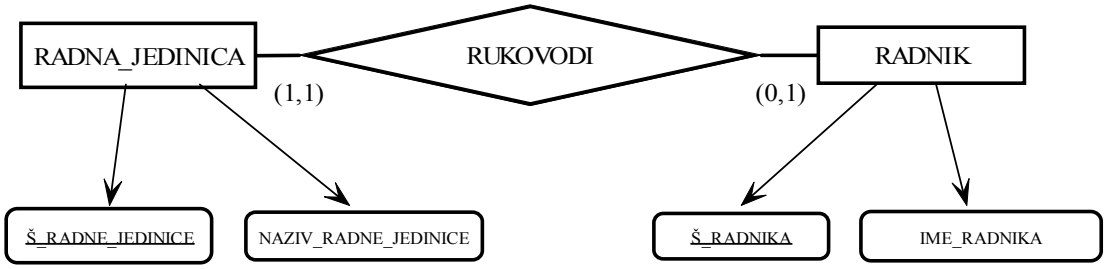
7.2.1. Poveznici sa kardinalnošću 1:1

V11-1. Veze sa kardinalnošću (1,1) : (1,1) zajedno sa oba tipa entiteta koji su vezani takvom vezom se prevode u jednu šemu relacije. Tu se, znači, vrši spajanje dva tipa entiteta u jednu šemu relacije, a poveznik neće postati nova relacija. Ključ šeme relacije se bira od dva kandidata za ključ (identifikatora jednog i drugog tipa entiteta).

V11-2. Veze sa kardinalnošću (0,1) : (1,1) sa dva tipa entiteta u vezi se prevode u dve šeme relacije. Tipovi entiteta u ovakvoj vezi se prevode u šeme relacije prema pravilu E1. iz 7.1., a veza između tipova entiteta neće postati nova šema relacije. Veza se ovde realizuje prostiranjem ključeva: identifikator jednog tipa entiteta se smešta kao obično obeležje u drugu šemu relacije. Novosmešteno obeležje u drugoj šemi relacije predstavlja spoljni ključ. Spoljni ključ treba da se realizuje u šemi relacije koja odgovara tipu entiteta sa strane (1,1). Drugo moguće rešenje (pojava spoljnog ključa sa

strane (0,1)) bi prouzrokovalo pojavu nul – vrednosti za taj spoljni ključ (videti sledeći primer).

ER dijagram sa sl. 7.4. odslikava deo realnog sveta u kojem postoje radne jedinice i



Sl. 7.4.

radnici. Radna jedinica mora imati barem jednog radnika za rukovodioca, a može imati najviše jednog. Radnik ne mora biti rukovodilac ni jedne radne jedinice, a može rukovoditi najviše jednom radnom jedinicom. Moguće pretvaranje ovog ER dijagrama daje dva rešenja od koja se samo jedno može prihvatiti. Rešenje br.1.:

RADNA_JEDINICA (Š_RADNE_JEDINICE, NAZIV_RADNE-JEDINICE)

RADNIK (Š_RADNIKA, IME_RADNIKA, Š_RADNE_JEDINICE_KOJOM_RUKOVODI)

Rešenje br.2.:

RADNA_JEDINICA (Š_RADNE_JEDINICE, NAZIV_RADNE_JEDINICE, Š_RADNIKA_RUKOVODIOCA)

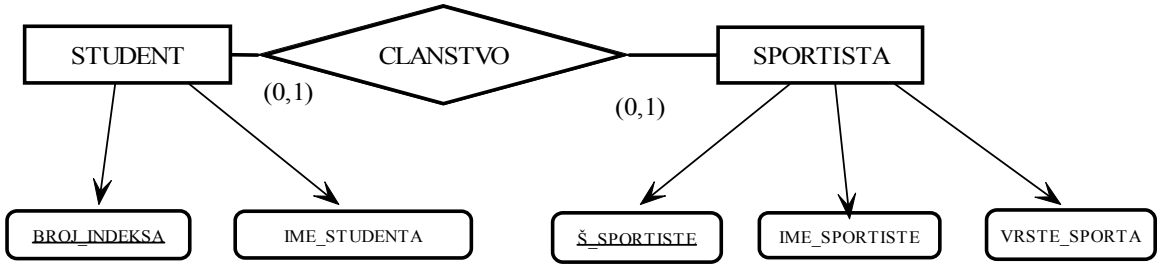
RADNIK (Š_RADNIKA, IME_RADNIKA)

Prvo rešenje prouzrokuje velik broj nula-vrednosti za spoljni ključ u konkretizacijama šeme relacija (redovima tabele) RADNIK, naime svi radnici koji nisu rukovodioci imaće nula-vrednost za obeležje Š_RADNE_JEDINICE_KOJOM_RUKOVODI. Drugo rešenje neće dovesti do nula-vrednosti spoljnog ključa u šemi relacije RADNA_JEDINICA, jer tip entiteta RADNA_JEDINICA mora imati rukovodioca (i to, samo jednog).

Ima tu još jedan momenat koji iziskuje kratko objašnjenje. Prilikom realizacije prostiranja ključa i u jednom i u drugom rešenju naziv spoljnog ključa se razlikuje od naziva primarnog ključa iz druge šeme relacije (Š_RADNE_JEDINICE – Š_RADNE_JEDINICE_KOJOM_RUKOVODI i Š_RADNIKA – Š_RADNIKA_RUKOVODIOCA). Sa jedne strane, modifikovani nazivi spoljnih ključeva ukazuju na njihovu ograničenu ulogu u dotičnoj šemi relacije (u šemi relacije RADNIK ne može figurisati bilo koja postojeća radna jedinica, već samo ona kojom upravo posmatrani radnik rukovodi), a, sa druge strane, promenjeni nazivi odražavaju i vezu usled koje su primarni ključevi dospeli u predmetnu šemu relacije.

V11-3. Veze sa kardinalnošću (0,1) : (0,1) sa dva tipa entiteta u vezi se pretvaraju u tri šeme relacije. Tipovi entiteta u ovakvoj vezi se prevode u šeme relacije po tački E1. iz 7.1., a veza između tipova entiteta će postati nova šema relacije. Šema relacija koja predstavlja vezu imaće obeležja iz oba tipa entiteta, koja su istovremeno i kandidati za ključ za posmatranu relaciju.

ER dijagram na sl. 7.5. predstavlja situaciju u kojoj student ne mora, ali ako hoće može da postane član sportskog društva. Isto tako sportista ne mora, ali može studirati. Šeme relacije koje odgovaraju dijagrama sa sl. 7.5. su sledeće:

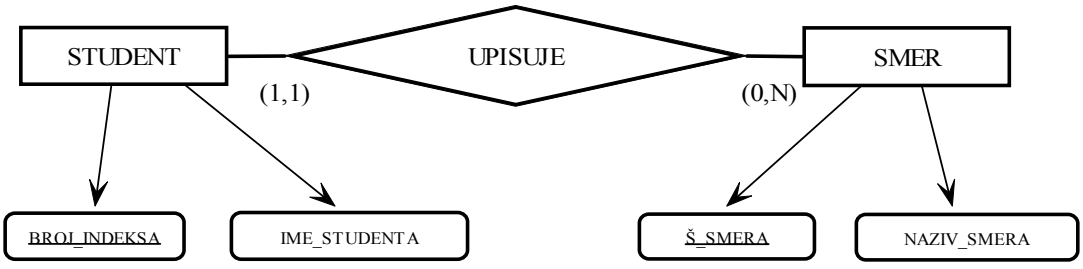


Sl. 7.5.

STUDENT (BROJ_INDEKSA, IME_STUDENTA)
SPORTISTA (Š_SPORTISTE, IME_SPORTISTE, VRSTE_SPORTA)
CLANSTVO (BROJ_INDEKSA, Š_SPORTISTE)

7.2.2. Poveznici sa kardinalnošću 1:N

V1N-1. Same veze sa kardinalnošću (1,1) : (0,N) i (1,1) : (1,N) neće postati šeme relacija, već se veza predstavlja prostiranjem ključeva. Identifikator tipa entiteta na strani za koju je maksimalni kardinalitet M se smešta kao obično obeležje u drugu šemu relacije (koja odgovara tipu entiteta na strani za koju je maksimalni kardinalitet 1). Novosmešteno obeležje u drugoj šemi relacije predstavlja spoljni ključ.



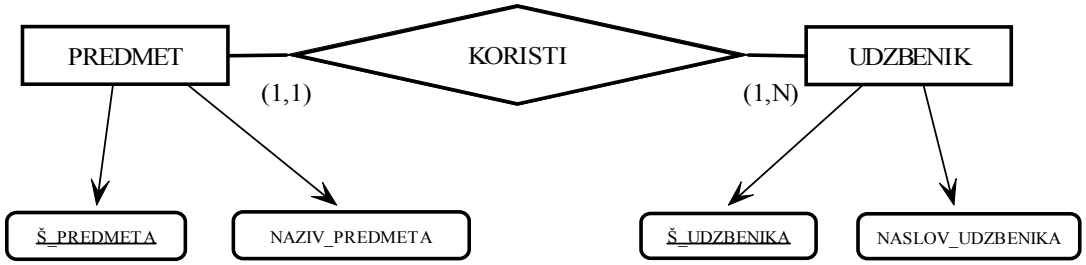
Sl. 7.6.

Primer na sl. 7.6. predstavlja situaciju u kojoj student mora da bira jedan smer studiranja (ali tačno jedan, a ne više), istovremeno smer egzistira bez obzira da li je neko od studenata uopšte birao taj smer (postoji i bez jednog prijavljenog studenta), a, naravno smer može upisati više studenata. Šeme relacije su sledeće:

STUDENT (BROJ_INDEKSA, IME-STUDENTA, Š_SMERA_UPISANOG)

SMER (Š_SMERA, NAZIV_SMERA)

ER dijagram na sl. 7.7. odražava san studenata: svaki predmet mora imati udžbenik (ali samo jedan), pri čemu udžbenik mora da se koristi barem za jedan predmet, a može i za više predmeta. Šeme relacije su kao što sledi:



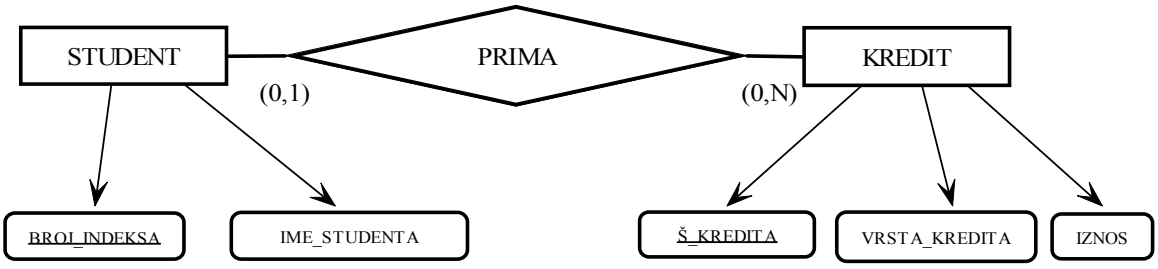
Sl. 7.7.

PREDMET (Š_PREDMETA, NAZIV-PREDMETA, Š_UDZBENIKA_KORIŠĆENOG)

UDZBENIK (Š_UDZBENIKA, NASLOV_UDZBENIKA)

V1N-2. Veze sa kardinalnošću (0,1) : (0,N) i (0,1) : (1,N) će postati posebna, nova šema relacija. Obeležja šeme relacije koja odgovara ovim vezama obrazuju identifikatori tipova entiteta koji su u vezi. Ključ novoobrazovane šeme relacije predstavlja identifikator tipa entiteta na strani za koju je maksimalni kardinalitet 1.

Sl. 7.8. predstavlja deo realnog sveta studenata primaoca kredita. Taj deo ER modela se



Sl. 7.8.

prevodi u relacioni model pomoću sledećih šema relacija:

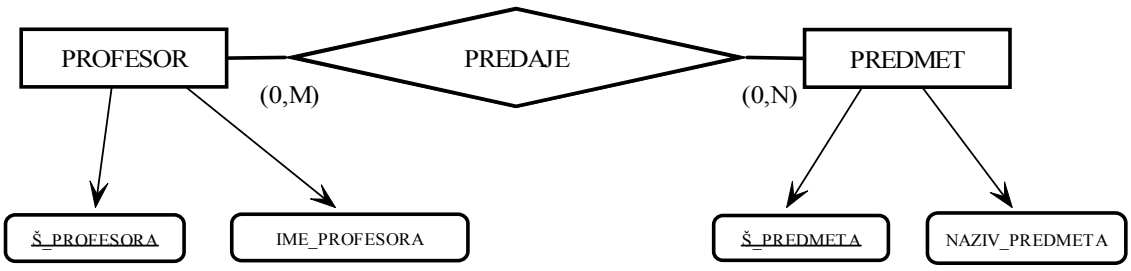
STUDENT (BROJ_INDEKSA, IME_STUDENTA)
KREDIT (Š_KREDITA, VRSTA_KREDITA, IZNOS)
PRIMA (BROJ_INDEKSA, Š_KREDITA)

V1N-3. Veza između nadređenog i slabog tipa entiteta, superklase (nadtipa) i potklase (podtipa) kod generalizacije/specijalizacije, kao i nadtipa i kategorije kod kategorizacije ne postaje posebna šema relacije. Prevođenje ovih veza je opisano u 7.1. (tačke od 1. do 6.).

7.2.3. Poveznici sa kardinalnošću M:N

VMN-1. Sve moguće kombinacije veza sa kardinalnošću M:N, a tu spadaju (0,M) : (0,N), (1,N) : (0,M), (0,N) : (1,M) i (1,N) : (1,M) postaju nove šeme relacije. Obeležja ove dodatne šeme relacije postaće identifikatori tipa entiteta koji su u vezi. Ključ ove šeme relacije će biti složen ključ sastavljen od oba identifikator tipa entiteta u vezi. To znači da će nova šema relacije sadržati samo ključna obeležja (neće sadržati ni jedno neključno obeležje). Neki je, zbog toga zovu relacija-ključ.

Primer na sl. 7.9. prikazuje tipove entiteta PROFESOR i PREDMET i njihov međusobni odnos. Ima takvih profesora koji još ili već ne rade ni na jednom predmetu u svojstvu nastavnika. Slučaj je sličan sa predmetima: iz nekih se još ili već ne održavaju predavanja iz razloga što su predviđeni u novom nastavnom planu ili u nekom starom nastavnom planu po kojem se više ne radi. Prevođenje dijagrama sa sl. 7.9. daje sledeće relacije:

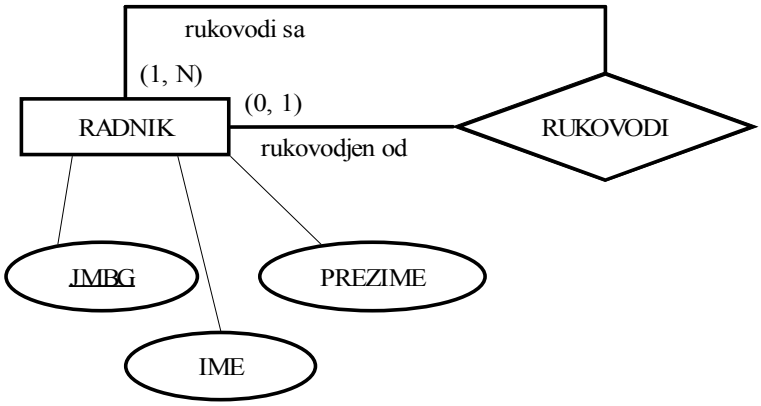


Sl. 7.9.

PROFESOR (Š_PROFESORA, IME_PROFESORA)
PREDMET (Š_PREDMETA, NAZIV_PREDMETA)
PREDAJE (Š_PROFESORA, Š_PREDMETA)

7.2.4. Rekurzivni tip poveznika

Rekurzivni tip poveznika se ponekad naziva i unarnom vezom. Tu se radi o vezi koja povezuje pojave istog tipa entiteta. Prevođenje unarnih veza se realizuje na isti način kao za obične, prethodno opisane binarne veze. Prevođenje zavisi od kardinalnosti tipa poveznika. Postoji, međutim, jedna specifičnost koja krasi rekurzivne tipove poveznika. Pri prevođenju unarnih veza spoljni ključ bi u šemi relacije imao isto ime kao i primarni ključ, što je nedozvoljeno, zato se vrši preimenovanje spoljašnjeg ključa.



Sl. 7.10.

Postoje dva prihvatljiva rešenja:

1.

RADNIK(JMBG, IME, PREZIME)

RUKOVODI(JMBG, JMBG_RUKOVODIOCA)

2.

RADNIK(JMBG, IME, PREZIME, JMBG_RUKOVODIOCA)

7.3. POSEBNA PRAVILA INTEGRITETA ZA SAČUVANJE SEMANTIKE ER MODELA

U napred izloženom je već istaknuto da je ER model semantički bogatiji od relacionog modela. Neka ograničenja definisana u ER modelu ne mogu da se artikulišu isključivo

prevođenjem entiteta i poveznika**.** U cilju sačuvanja semantike ER modela se pored prevođenjem dobijenih šema relacija moraju definisati posebna pravila integriteta. Ta posebna pravila integriteta se mogu formulisati na osnovu ER dijagrama analizom jedne veze sa pripadajuća dva tipa entiteta. Posebna pravila integriteta mogu biti statička i dinamička.

Statička pravila integriteta definišu uslove koji moraju biti zadovoljeni pre i posle izvršenja proizvoljne operacija nad bazom podataka.

Dinamička pravila integriteta definišu procedure u relacionom modelu koje odgovaraju operacijama u ER modelu (osnovnim operacijama operacijske komponente ER modela), a ujedno obezbeđuju održavanje (ostvarenje) uslova integriteta. Pošto je operacijska komponenta ER modela u potpunosti je analizirana u glavi 4, dinamička pravila integriteta nećemo analizirati.

7.3.1. Posebna pravila integriteta vezana za entitete

PPI-E1. Vrednost identifikatora tipa entiteta, shodno tome primarnog ključa šeme relacije kao celine, a i bilo koje njegove komponente ne sme imati nul – vrednost.

PPI-E2. Vrednost identifikatora slabog tipa entiteta, odnosno u njemu odgovarajućoj šemi relacije vrednost primarnog ključa ne može imati vrednost koja ne postoji kao vrednost primarnog ključa u šemi relacije koja odgovara nadređenom tipu entiteta.

PPI-E3. Vrednost identifikatora podtipa, tj. vrednost primarnog ključa šeme relacije koja odgovara podtipu ne može imati vrednost koja ne postoji kao vrednost ključa u šemi relacije koja odgovara nadtipu.

PPI-E4. Za vrednost identifikatora tj. vrednost kjluča šeme relacije koja odgovara gerundu (sigurno složeni kjluč) važi pravilo integriteta pod 1. u ovoj tački.

PPI-E5. Za vrednost identifikatora kategorije (potklase u hijerarhiji »kategorija«) odnosno u njemu odgovarajućoj šemi relacije za vrednost primarnog ključa (pošto je veštački uveden jedinstveni ključ – surogat) važi pravilo integriteta pod 1. u ovoj tački.

7.3.2. Posebna pravila integriteta za veze sa kardinalnošću 1:1

PPI-V11-1. Za veze sa kardinalnošću (1,1) : (1,1) važi ograničenje kao posebno pravilo integriteta da nijedan od dva kandidata za ključ ne smeju imati nul – vrednost. Kao što je poznato veze sa kardinalnošću (1,1) : (1,1) zajedno sa oba tipa entiteta koji su

vezani takvom vezom se prevode u jednu šemu relacije, pri čemu se ključ šeme relacije bira od dva kandidata za ključ (identifikatora jednog i drugog tipa entiteta u vezi).

PPI-V11-2. Za veze sa kardinalnošću (0,1) : (1,1) važi pravilo integriteta koje glasi: spoljni ključ u šemi relacije koja odgovara tipu entiteta sa strane (1,1) ne sme imati vrednost ako ta ista vrednost ne postoji kao vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (0,1).

PPI-V11-3. Za veze sa kardinalnošću (0,1) : (0,1) važi sledeće pravilo integriteta: za treću šemu relacije koja predstavlja vezu nijedan od dva kandidata za ključ ne smeju imati nul – vrednost.

7.3.3. Posebna pravila integriteta za veze sa kardinalnošću 1:N

PPI-V1N-1. Posebno pravilo integriteta za veze sa kardinalnošću (1,1) : (1,M) glasi: neka konkretna vrednost obeležja (spoljnog ključa) se ne može pojaviti u šemi relacije koja odgovara tipu entiteta sa strane (1,1), ako se ta ista vrednost se ne pojavljuje kao vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (1,M), a istovremeno se vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (1,M) mora, barem jednom, pojaviti kao vrednost odgovarajućeg obeležja (spoljnog ključa) u šemi relacije koja odgovara tipu entiteta sa strane (1,1). Prethodno opisano ograničenje nalaže istovremeno ažuriranje obe relacije kojima se predstavljaju veze sa kardinalnošću (1,1) : (1,M) u ER modelu.

PPI-V1N-2. Za veze sa kardinalnošću (1,1) : (0,M) posebno pravilo integriteta glasi: neka konkretna vrednost obeležja (spoljnog ključa) se ne može pojaviti u šemi relacije koja odgovara tipu entiteta sa strane (1,1), ako se ta ista vrednost se ne pojavljuje kao vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (1,M).

PPI-V1N-3. Za veze sa kardinalnošću (0,1) : (1,M) važi ograničenje kao posebno (dosta složeno) pravilo integriteta koje se može formulisati na sledeći način. Neka konkretna vrednost obeležja (spoljnog ključa) se ne može pojaviti u šemi relacije koja odgovara vezi (treća relacija), ako se ta ista vrednost ne pojavljuje kao vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (1,M), a istovremeno se vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (1,M) mora, barem jednom, pojaviti kao vrednost odgovarajućeg obeležja (spoljnog ključa) u šemi relacije koja odgovara vezi (ovo ukazuje na uzajamni integritet što ima za posledicu istovremeno ažuriranje relacije veze i relacije na strani (1,M). Sa

druge strane neka konkretna vrednost obeležja (primarnog ključa) se ne može pojaviti u šemi relacije koja odgovara vezi, ako se ta ista vrednost se ne pojavljuje kao vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (0,1).

PPI-V1N-4. Za veze sa kardinalnošću (0,1) : (0,M) ograničenje kao posebno pravilo integriteta se može formulisati na sledeći način. Neka konkretna vrednost obeležja (spoljnog ključa) se ne može pojaviti u šemi relacije koja odgovara vezi, ako se ta ista vrednost ne pojavljuje kao vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (0,M), a istovremeno mora važiti da se ne može pojaviti neka konkretna vrednost obeležja (primarnog ključa) u šemi relacije koja odgovara vezi, ako se ta ista vrednost ne pojavljuje kao vrednost primarnog ključa u šemi relacije koja odgovara tipu entiteta sa strane (0,1).

7.3.4. Posebna pravila integriteta za veze sa kardinalnošću M:N

Kod tipova veza sa kardinalnošću M:N prevođenje se realizuje posebnom (veznom)šemom relacije. Ključ vezne šeme relacije je u ovom slučaju složen i sastavljen je od ključeva i jedne i druge šeme relacije koje su u vezi.

PPI-VMN-1. Za veze sa kardinalnošću (0,M) : (0,N) važi ograničenje kao posebno pravilo integriteta koje se može opisati na sledeći način. Neka je jedna konkretizacija složenog ključa u veznoj relaciji {**a**,**b**}, pri čemu je **a** vrednost obeležja Sif_A (ključa prve relacije – relacije A u vezi), a **b** je vrednost obeležja Sif_B (ključa druge relacije – relacijeB u vezi). U tom slučaju vrednost **a** mora egzistirati kao vrednost ključa u relaciji A, a vrednost **b** mora postojati kao vrednost ključa u relaciji B.

PPI-VMN-2. Za veze sa kardinalnošću (0,M) : (1,N) važi strožije ograničenje od prethonog kao posebno pravilo integriteta i može se formulisati na sledeći način. Neka je jedna konkretizacija složenog ključa u veznoj relaciji {**a**,**b**}, pri čemu je **a** vrednost obeležja Sif_A (ključa prve relacije – relacije A u vezi), a **b** je vrednost obeležja Sif_B (ključa druge relacije – relacijeB u vezi). U tom slučaju vrednost **a** mora egzistirati kao vrednost ključa u relaciji A, a vrednost **b** mora postojati kao vrednost ključa u relaciji B. (Do ove tačke pravilo integriteta je ekvivalentno prethodnom ograničenju.). Pored toga, u relaciji B ne može postojati vrednost ključa **b**, ako ta ista vrednost **b** ne postoji kao deo konkretizacije ključa u veznoj relaciji. Ažuriranje relacije B i vezne relacije se, dakle mora istovremeno uraditi.

PPI-VMN-3. Za veze sa kardinalnošću (1,M) : (1,N) važi još strožije ograničenje od prethonog i može se formulisati na sledeći način. Neka je jedna konkretizacija složenog ključa u veznoj relaciji {**a**,**b**}, pri čemu je **a** vrednost obeležja Sif_A (ključa prve relacije – relacije A u vezi), a **b** je vrednost obeležja Sif_B (ključa druge relacije – relacijeB u vezi). U tom slučaju vrednost **a** mora egzistirati kao vrednost ključa u relaciji A, a vrednost **b** mora postojati kao vrednost ključa u relaciji B. (Do ove tačke pravilo integriteta je ekvivalentno prethodnom ograničenju.). Pored toga, u relaciji B ne može postojati vrednost ključa **b**, ako ta ista vrednost **b** ne postoji kao deo konkretizacije ključa u veznoj relaciji. A isto važi i za drugu relaciju: u relaciji A ne može postojati vrednost ključa **a**, ako ta ista vrednost **a** ne postoji kao deo konkretizacije ključa u veznoj relaciji. Ažuriranje sve tri relacije se, dakle mora istovremeno uraditi.

8. SQL – JEZIK ZA UPRAVLJANJE RELACIONIM BAZAMA PODATAKA

Relacioni model je strukturalno jednostavan i zasniva se na relacijama, odnosno dvodimenzionalnim tabelama. Kod razrade prvog jezika za upravljanje bazama podataka relacionih sistema prvenstveni cilj je bio da se realizuje takav deklarativni (opisni upravljački) jezik koji će omogućiti obradu relacija pomoću skupovnih operacija relacione algebre. Prvi jezik za upravljanje relacionim bazama podataka je objavljen 1974. godine pod nazivom SEQUEL (Structured English Query Language) ukazajući i nazivom da se radi o strukturalnom (sintaktički strogo određenom) jeziku baziranom na razumljivim engleskim rečima. Skraćenica obuhvata i reč upitni (Query) naglašavajući efikasnost ovog jezika u izražavanju upita, mada ovaj jezik može da posluži za razne druge aktivnosti u vezi upravljanja relacionim bazama podataka. Kasnije se naziv jezika promenio na SQL pod kojim imenom je i danas poznat.

Prvi korak na putu standardizacije SQL jezika je napravio ANSI (American National Standards Institute) 1986. godine, a godinu dana kasnije je i ISO (International Standards Organisation) objavio ISO SQL standard. Dve korekcije SQL standarda su obavljene 1989., 1992. i 1999. godine. SQL86 preporuke su obuhvatale samo osnovne elemente iz tada postojećih SQL realizacija, tako da su tadašnji SQL jezici pretežno prevazišli objavljeni standard. SQL89 je korigovao nedostale preporuke kod postojećih realizacija, a SQL92 i SQL99 sadrže viziju razvoja za budućnost.

SQL nije softver za upravljanje bazama podataka već samo jezik za rukovanje relacionim bazama. SQL je samo jedna komponenta softvera za rukovanje relacionim bazama podataka. Pored funkcije rukovanja podacima u relacionim bazama SQL obezbeđuje i druge vrste naredbi. Naredbe SQL jezika se dele na kategorije na osnovu kojih se definišu »podjezici« SQL jezika:

1. Jezik za definisanje podataka (DDL – Data Definition Language),
2. Jezik za manipulisanje podacima (DML – Data Manipulation Language),
3. Jezik za upit (Query) i
4. Jezik za upravljanje podacima (DCL – Data Control Language).

U nastavku će biti prikazane naredbe SQL86 jezika, jer su najprostije, a kompatibilne su sa naredbama novijih standarda. Krenućemo kategorijama SQL naredbi shodno njihovoj nameni, a prema gore datom redosledu. Za primenu SQL jezika za pisanje standardnih aplikacija, skup SQL naredbi treba da se proširi proceduralnim naredbama. Za realizaciju prethodno formulisanog zahteva postoje dve mogućnosti:

1. ugradnja SQL-a u postojeće proceduralne jezike i

2. proširenje SQL jezika proceduralnim instrukcijama (naredbama).

8.1. JEZIK ZA DEFINISANJE PODATAKA (DDL)

Prvi korak u primeni baze podataka je definisanje (prazne) strukture za smeštaj podataka koja će kasnije biti punjena i obrađivana. Za relacione baze podataka prazna struktura označava samu bazu podataka i njene tabele. Baza podataka predstavlja okvir u kojem tabele egzistiraju, a koje tabele moraju jedinstveno ime pomoću kojeg se mogu jednoznačno identifikovati. Unutar tabele se nalaze polja (nazivi kolona) tabele. Kako nazivi tabela moraju imati jedinstveno ime unutar iste baze podataka tako i nazivi kolona moraju biti jedinstveni unutar iste tabele. Kod kreiranja tabele se zadaju naziv tabele i nazivi i tipovi polja koja saćinjavaju tabelu.

Referencijalni integritet predstavlja sastavni deo relacionog modela, no SQL86 ne sadrži definicije svih mogućih uslova integriteta.

Instrukcija pomoću koje se definiše baza podataka ima sledeću formu:

CREATE DATABASE *naziv_baze_podataka*;

Izvršavanjem ove instrukcije nastaje baza podataka pod navedenim nazivom u aktuelnom (default) folderu.

Brisanje baze podataka je moguće izvršiti ako je ona zatvorena (nije aktivna) i to pomoću sledeće naredbe:

DROP DATABASE *naziv_baze_podataka*;

Za dobijanje informacija (naziv, matični folder baze podataka, datum kreiranja, identifikator korisnika-kreatora baze) o svim postojećim bazama podataka koristi se instrukcija oblika:

SHOW DATABASE;

Pre korišćenja bazu podataka treba aktivirati (uvek samo jedna baza podataka može biti aktivna) pomoću:

START DATABASE *naziv_baze_podataka*;

Zatvaranje (deaktiviranje) baze podataka se realizuje sledećom naredbom:

STOP DATABASE;

Naredba za kreiranje prazne (definisanje strukture za smeštaj podataka) tabele je sledećeg oblika:

CREATE TABLE *naziv_tabele* (*ime_polja*$_1$ *tip_polja*$_1$*(dužina_polja*$_1$*) [integritet_polja*$_1$*]*
[,..., ime_polja$_n$ *tip_polja*$_n$*(dužina_polja*$_n$*) [integritet_polja*$_n$*]]*
[, integritet_za_grupu_polja])

- *ime_polja*$_i$ - jedinstveno ime naziva kolone (polja) tabele
- *tip_polja*$_i$ - ukazuje na tip podatka koji će se smestiti u posmatrano polje (svaki relacioni sistem za upravljanje bazama podataka ima svojstven skup mogućih tipova podataka unapred definisanih dimenzija – dužina)
- *integritet_polja*$_i$ - predstavlja uslov integriteta koji mora biti zadovoljen za dotično polje
  vrsta uslova integriteta za polje može biti:
  - ✓ UNIQUE - sadržaj polja mora biti jedinstven u okviru tabele i
  - ✓ NOT NULL - polje ne sme ostati nepopunjen (ne može sadržati nula-vrednost)
- *integritet_za_grupu_polja* od gore navedena dva uslova integriteta za polje samo prvi može da se primeni za grupu polja navođenjem imena polja u zagradi:
  - ✓ UNIQUE (*ime_polja*$_1$ [,..., *ime_polja*$_n$])

Kao primer za kreiranje tabele može da posluži šema relacije student u sledećem obliku (izgledu):

STUDENT (<u>BRINDEKSA</u>, IMESTU, GODRODJSTU, ADRESASTU, TELEFSTU)

```
CREATE TABLE STUDENT (BRINDEKSA CHAR(8) NOT NULL UNIQUE,
                      IMESTU CHAR(20) NOT NULL,
                      GODRODJSTU NUMBER(4),
                      ADRESASTU CHAR(20),
```

TELEFSTU CHAR(12));

U standardu SQL86 nije postojala naredba za modifikaciju strukture tabele (naredba ALTER kojom je bilo moguće dodavati polja (nazive kolona) tabele je tek kasnije uveden), već samo za njeno brisanje u obliku:

DROP TABLE *naziv_tabele*;

Tabele kreirane pomoću CREATE TABLE instrukcije su i fizički postojeće tabele u koje se fizički smeštaju podaci. Pored ovakvih, fizički postojećih tabela u bazi podataka mogu definisati i pogledi (VIEW), fizički nepostojeće, izvedene tabele. Sistem za rukovanje bazama podataka memoriše samo njegovu definiciju. Podaci u jednom redu VIEW tabele fizički nigde nisu zajedno smešteni kako ih vidimo u prikazu VIEW tabele, već se sakupljaju iz različitih (fizički postojećih) tabela shodno definiciji samog VIEW-a:

CREATE VIEW *naziv_tabele_pogleda* [(*ime_polja*$_1$ [, *ime_polja*$_{2...}$ *ime_polja*$_n$])] AS *niz_operacija*;

Naznačena imena polja u CREATE VIEW naredbi ukazuju na mogućnost korišćenja različitih imena polja u VIEW-u od imena prvobitnih, fizički postojećih polja u osnovnim tabelama. Ako se ne specificiraju imena polja u CREATE VIEW naredbi onda će nazivi za polja u pogledu naslediti nazive iz osnovnih, fizički postojećih tabela. U CREATE VIEW instrukciji *niz_operacija* označava skup upitnih aktivnosti formulisan pomoću SELECT naredbe (format SELECT nardbe se kasnije objašnjava). Kao primer definišimo pogled studenata rođenih 1982. godine koji će sadržati broj indeksa, ime i telefon:

CREATE VIEW student1982 AS SELECT brindeksa, imestu, telefstu
FROM student WHERE godrodjstu = 1982;

Brisanje definicije pogleda (VIEW tabele) se vrši na isti način kao bilo koje fizički postojeće tabele:

DROP VIEW *naziv_tabele_pogleda*;

8.2.JEZIK ZA MANIPULISANJE PODACIMA (DML)

Posle definisanja imamo prazne tabele na raspolaganju. SQL86 standard omogućava tri osnovne aktivnosti za manipulisanje podacima:

1. unos podataka,
2. modifikaciju podataka i
3. brisanje podataka.

8.2.1. Naredbe DML-a za unos podataka

Za unos podataka se koristi INSERT naredba čija je sintaksa sledeća:

INSERT INTO *naziv_tabele*
[*ime_polja*$_1$, *ime_polja*$_2$,..., *ime_polja*$_n$]
VALUES (*vrednost_polja*$_1$, *vrednost_polja*$_2$,..., *vrednost_polja*$_n$);

Unos konkretnog studenta u tabelu STUDENT se izvršava na sledeći način:

```
INSERT INTO student
        brindeksa, imestu, godrodjstu, adresastu, telefstu
        VALUES ("I-18/2003", "Petar Petrović", 1984, "Subotica Rumska 17", NULL);
```

U primeru student ne poseduje telefon (ili samo u trenutku upisa broj njegovog telefona je nepoznat), tako će polje telefstu ostati prazan (NULL ili nula-vrednost).

Postoji i drugi oblik INSERT naredbe koji služi ne za pojedinačni nego za masovni unos (masovno punjenje) podataka iz neke druge tabele

INSERT INTO *naziv_tabele niz_operacija*;

8.2.2. Naredba DML-a za modifikaciju podataka

U slučaju pogrešno unetih podataka neophodno je ispravljati loše unošene podatke. U naredbi za modifikaciju je moguće definisati uslov pomoću kojeg se određuju redovi na koje se modifikacije odnose. Format naredbe je sledeći:

UPDATE *naziv_tabele*

SET $ime\_polja_1 = vrednost\_polja_1$

[,$ime\_polja_2 = vrednost\_polja_2$

.

.

$ime\_polja_n = vrednost\_polja_n$]

[**WHERE** *uslov*];

U prethodnoj naredbi *uslov* može biti i složen logički izraz u kojem proste izraze vezuju logičke AND, OR i NOT operacije. U prostim izrazima mogu da se koriste sledeći relacioni operatori: = (jednako), > (veće), < (manje), >= (veće ili jednako – ne manje od), <= (manje ili jednako – ne veće od), <> (nije jednako), BETWEEN, IN, LIKE i IS NULL.

Uobičajeni relacioni operatori su poznati iz svakidašnje prakse. Zadnja četiri relaciona operatora su specifična, pa će njihova sintaksa biti objašnjena u narednim pasusima.

Pomoću operatora BETWEEN se vrši kontrola neke vrednosti da li se nalazi unutar nekog zadatog intervala vrednosti ili ne. Operator se primenjuje u sledećem formatu:

$izraz_1$ **BETWEEN** $izraz_2$ **AND** $izraz_3$

U gore formulisanom izrazu $izraz_2$ je manji ili jednak izrazu $izraz_3$ i vraća istinitu vrednost ako je vrednost izraza $izraz_1$ veća ili jednaka vrednosti izraza $izraz_2$, odnosno istovremeno manja ili jednaka vrednosti izraza $izraz_3$.

IN operator upoređuje vrednost polja tabele sa elementima jedne liste. Oblik uslova formulisanog pomoću IN operatora je sledeći:

ime_polja **IN** ($element\_liste_1$ [, $element\_liste_2$,..., $element\_liste_n$])

Gornji izraz daje istinitu vrednost ako se vrednost *ime_polja* poklapa sa vredošću nekog elementa liste $element\_liste_i$, i=1,2, ..., n.

Operator LIKE služi za upoređivanje stringova (tip polja je CHAR). Nije rigorozan kao operator = (kod operatora = upoređeni stringovi moraju biti potpuno ekvivalentni), već se pomoću njega mogu formulisati delimične jednakosti ili ekvivalencije pojedinih delova stringova. Formulisanje uslova treba da sledi sledeću sintaktiku:

ime_polja **LIKE** 'abcd' [**ESCAPE** 'x']

U gornjem izrazu 'abcd' predstavlja sadržaj sa kojim se upoređuje vrednost *ime_polja*. Mustra 'abcd' može da sadrži i karaktere za maskiranje ili jocker karaktere, i to samo dva dozvoljena karaktera:

_ - taj znak zamenjuje jedan karakter i

% - zamenjuje proizvoljan niz karaktera.

Da bi se mogli i ovi jocker karakteri koristiti u mustri za upoređivanje uveden je ESCAPE opcija. ESCAPE opcija deklariše neki specijalni karakter iza kojeg u mustri rezervisani jocker karakter nema svoju funkciju maskiranja, već ga treba smatrati običnim znakom. U sledećem primeru formulišemo uslov za ispitivanje da li sadržaj *ime_polja* počinje nizom znakova 50%:

ime_polja LIKE '50#%%' ESCAPE '#'

ESCAPE opcijom deklarisan karakter # (taraba) obezbeđuje da će se znak % iza njega smatrati običnim karakterom za upoređivanje, no to se ne odnosi na sledeći (drugi po redu) znak % (jer ispred njega ne stoji ESCAPE-om definisan specijalni simbol) – on označava da se ostali znaci uopšte ne upoređuju (bitno je samo da prva tri znaka budu sledeća: 50%).

IS NULL operator vrši ispitivanje sadržaja polja na praznu (nepopunjenu) vrednost. Vraća istinitu vrednost ako je polje *ime_polja* prazno:

ime_polja **IS NULL**

Pomoću ključne reči NOT je moguće negirati sva četiri do sada objašnjena operatora stavljanjem reči NOT ispred svakog od operatora osim IS NULL operatora. Kod njega se negiranje formuliše u obliku IS NOT NULL.

U prostim logičkim izrazima se mogu koristiti i poznati aritmetički operatori: + (sabiranje), - (oduzimanje), * - množenje i / - deljenje.

Kao primer za modifikaciju će nam poslužiti previd unosa kod imena prethodno insertovanog studenta: student se istinski zove Petar Kočić, a ne Petar Petrović:

```
UPDATE student
        SET imestu = "Petar Kočić"
        WHERE brindeksa = "I-18/2003";
```

8.2.3. Naredba DML-a za brisanje podataka

Konkretni entiteti svakog tipa entiteta imaju svoj životni ciklus: nastaju, razvijaju se i nestaju. Shodno tome u nekim situacijama treba obezbediti i fizičko brisanje podataka iz neke tabele baze podataka. Naredba za brisanje konkretnog entiteta (reda tabele baze podataka) ima sledeći oblik:

DELETE FROM *naziv_tabele*
[**WHERE** *uslov*];

Obavezni deo naredbe sadrži samo naziv tabele, međutim, izvršavanjem tog oblika naredbe za brisanje može imati kobne posledice jer briše eve redove navedene tabele. Navođenjem konkretnog *uslov*a u WHERE klauzuli mogu se izabrati oni redovi tabele koji su predviđeni za brisanje. Izbrisaće se oni redovi za koje je *uslov* istinit. Formulisanje *uslov*a je detaljno objašnjen kod opisa prethodne naredbe (za modifikaciju saržaja reda tabele).

Opasnost od gubljenja podataka postoji u slučaju ako se naredba za brisanje neodgovorno ili nepromišljeno koristi. Kod većine sistema za rukovanje bazama podataka, međutim, postoji mogućnost restauracije prethodnog stanja, tako da se kod ovakvih sistema brisani podaci mogu spasiti izdavanjem (izvršavanjem) sledeće instrukcije:

ROLLBACK

Ta naredba se objašnjava u delu obrade transakcija jer je ona predviđena za poništavanje kompletnog niza naredbi definisanog u okviru jedne transakcije koju treba atomski izvršavati.

8.3. NAREDBA ZA UPIT PODATAKA

U SQL86 (a i u kasnijim modifikacijama, standardima ovog jezika) postoji samo jedna naredba za upit u bazu podataka, a to je čuvena ili famozna SELECT naredba. Pošto SELECT mora biti u stanju da formuliše sve objašnjene operacije relacione algebre korisne za sastavljanje upita, logično je očekivati da je struktura ove naredbe dosta složena. Postoje dva pristupa za upoznavanje SELECT naredbe. Prvi pristup kreće od najprostijeg oblika u objašnjenjima ka najsloženijoj formi, a drugi sledi obrnuti smer: kreće od pregleda komletne SELECT rečenice i

objašnjava pojedine delove naredbe. Naš prikaz će slediti drugi način pristupa u upoznavanju naredbe za upit. Bez obzira na način upoznavanja SELECT rečenice treba zapamtiti da rezultat izvršenja naredbe za upit predstavlja privremenu tabelu koja nestaje izvršavanjem neke druge (SQL) instrukcije. Tabela rezultata (R-tabela) se izgrađuje postepeno na osnovu prisutnih klauzula (glavnih delova SELECT rečenice) u samoj naredbi za upit. Unutar SELECT rečenice se mogu nalaziti unutrašnji upiti (tzv. unutrašnji ili ugrađeni SELECT-i), a takođe i same naredbe za upit se mogu povezati pomoću UNION operacije . Struktura kompletne SELECT naredbe u SQL86 standardu je sledeće (naredba je pisana **bold**-om, a objašnjenje *italic*-om):

SELECT ...	*izbor kolone (polja) - **projekcija***
FROM ...	*zadavanje naziva tabela – **Dekartov proizvod***
[WHERE ...]	*izbor redova - **selekcija***
[[GROUP BY ...	*grupisanje*
[HAVING ...]]	*izbor između grupa*
[ORDER BY ...]	*sortiranje rezultata (privremene R-tabele)*
[SAVE TO TEMP ...];	*memorisanje privremene R-tabele*
[UNION ...]	*povezivanje R-tabela*

Struktura SELECT rečenice ukazuje minimalnu moguću naredbu koja mora sadržati pored (naravno) SELECT klauzule i FROM klauzulu. Moramo, znači, deklarisati odakle želimo i šta želimo prikazati (ili listati u R-tabeli). Sve ostale klauzule su opcionalne. Interesantno je, međutim ukazati na redosled izvršavanje pojedinih klauzula u SELECT rečenici jer se redosled izvršavanja same naredbe, tj. njenih klauzula se ne poklapa sa propisanim redosledom definisanja. Evo redosleda izvršavanja klauzula SELECT naredbe:

FROM ...	*zadavanje naziva tabela – **Dekartov proizvod***
[WHERE ...]	*izbor redova - **selekcija***
[[GROUP BY ...	*grupisanje*
[HAVING ...]]	*izbor između grupa*
SELECT ...	*izbor kolone (polja) - **projekcija***
[ORDER BY ...]	*sortiranje rezultata (privremene R-tabele)*
[SAVE TO TEMP ...];	*memorisanje privremene R-tabele*

8.3.1. SELECT klauzula

Kompletna forma SELECT podrečenice (klauzule) je sledeća:

SELECT [ALL | DISTINCT] *ime_polja*$_1$ [,..., *ime_polja*$_n$] | *
FROM *naziv_tabele*$_1$ [,..., *naziv_tabele*$_n$];

Semantika SELECT klauzule je sledeća: selektuje navedena polja na osnovu navedenih ograničenja (ALL | DISTINCT) – projekcija – iz tabele(a) navedene(ih) u FROM klauzuli i na taj način konstituiše R-tabelu. Opcije u SELECT klauzuli su sledeće:

ALL – R-tabela će sadržati i duplirane redve (ako oni postoje u tabeli) – ta opcija je default, tj. podrazumeva se u slučaju da nema navedene opcije,

DISTINCT – R-tabela neće sadržati duplirane redove,

***** - sve kolone (sva polja) tabele(a) nevedene(ih) u FROM klauzuli će se pojaviti u R-tabeli,

ime_polja$_1$ [,..., *ime_polja*$_n$] – samo polja navedena u ovoj listi će se pojaviti u R-tabeli.

Primer:
SELECT * FROM student;
lista sve podatke o svim studentima iz tabele STUDENT.

Primer:
SELECT brindeksa, imestu, godrodjstu FROM student;
konstituiše R-tabelu sa nazivima kolona brindeksa, imestu i godrodjstu gde smešta ove podatke o svim studentima.

Primer:
SELECT DISTINCT godrodjstu FROM student;
lista sve različite godine rođenja studenata.

Za elemenat liste polja projekcije SELECT naredbe (SELECT klauzule SELECT rečenice kao celine) važe sledeća ograničenja. Elemenat liste može biti:

- ime polja – ako više tabela imaju polja pod istim imenom, ime polja treba da se navede u obliku: *naziv_tabele*. *ime_polja*,

- izraz sastavljen od imena polja tabela navedenih u from klauzuli,
- funkcija za agregiranje podataka,
- drugi izraz.

Primer - **aritmetički izraz**:
SELECT brindeksa, imestu,YEAR(DATE())-godrodjstu FROM student;
lista pored broja indeksa i imena i podatak koliko student ima godina.

Primer - **karakterski izraz**:
SELECT LEFT(brindeksa,1,1),UPPER(imestu) FROM student;
sastavlja R-tabelu sa oznakom smera (prvi znak u broju indeksa) i imena studenata – velikim slovima štampanih.

Funkcije za agregiranje podataka

Funkcije za agregiranje omogućuju stvaranje jedinstvenog polja za formirane grupe (definisane pomoću GROUP BY klauzule – videti kasnije), na taj način, da generišu jednu vrednost dotičnog polja za celu grupu. Tako će jedan red R-tabele sadržati podatke za jednu grupu podataka. Operatori koji se mogu koristiti formalno kao funkcije (odatle i naziv: funkcije za agregiranje podataka), a služe za stvaranje jedne vrednosti za celu grupu su sledeći:

1. Operator (funkcija) za prebrojavanje ima sledeći oblik

COUNT([DISTINCT] *ime_polja* **| *)**

Na ovakav način definosano numeričko polje R-tabele će sadržati vrednost izbrojanih članova elemenata (vrednosti) u navedenom polju tabele. Ako je naznačena i reč DISTINCT polje će sadržati samo vrednost koja ukazuje na različite vrednosti u navedenom polju *ime_polja*. Navedeno polje *ime_polja* može biti proizvoljnog tipa, ali u brojanje neće uzeti null-vrednosti. Zato, ako nas interesuje broj redova u tabele treba da formu: COUNT(*)

Primer:
SELECT COUNT (brindeksa) FROM student;

R-tabela vraća samo jednu vrednost, i to, ukupan broj evidentiranih studenata (pošto je polje brindeksa po definiciji ne može imati null-vrednost).

2. Operator za sabiranje (sumiranje) se opisuje na sledeći način

SUM (**[ALL]** *izraz* | **[ALL|DISTINCT]** [*ime_polja*])

Sabirati je moguće pomoću ovog operatora vrednosti definisane pomoću proizvoljnog validnog *izraz*, ali se, naravno, može tražiti zbir svih (**ALL**) ili samo različitih (**DISTINCT**) vrednosti nekog polja. Polje treba da je numeričko, odnosno izraz numerički.

Primer:
SELECT SUM (DISTINCT(godrodjstu)) FROM student;
Jedini mogući primer za prikazivanje SUM operatora (pošto je samo polje godrodjstu numeričkog tipa) je predstavljen prethodnom rečenicom (bez obzira što nema puno smisla). Upit sabira godine rođenja studenata, ali svaku godinu rođenja uzima u obzir samo jednom pri stvaranju zbira.

3. Operator za računanje proseka se može koristiti u sledećem obliku

AVG (**[ALL]** *izraz* | **[ALL|DISTINCT]** [*ime_polja*])

Slično kao kod sumiranja, može se tražiti prosek nekog izraza za posmatranu grupu, ali i prosečna vrednost neke kolone (polja) uzimajući u obzir sve moguće (**ALL**) ili samo različite vrednosti (**DISTINCT**) kolone. Prosek se može tražiti za numerička polja, odnosno izraze.

Primer:
SELECT AVG(DISTINCT((godrodjstu)) FROM student;

4. Funkcija za nalaženje minimalne vrednosti ima sledeću prostu formu

MIN(*ime_polja*)

ime_polja predstavlja polje koje je tekstualnog, datumskog ili vrednosnog (numeričkog) tipa.

Primer:
SELECT MIN(godrodjstu) FROM student;
Prethodni upit daje godinu rođenja najstarijeg(ih) studen(a)ta.

5. Operator za nalaženje maksimalne vrednosti

MAX(*ime_polja*)

Polje za koje tražimo najveću vrednost treba da je takvog tipa koji se može uređivati (numerički, tekstualni, datumski tip).

Primer:
SELECT MAX(godrodjstu) FROM student;
Prethodni SELECT vraća jednu vrednost: godinu rođenja najmlađeg(ih) studen(a)ta.

8.3.2. FROM klauzula

Ovaj deo SELECT rečenice je već naveden u SELECT klauzuli, ali u stvari predstavlja samostalnu klauzulu bez koje ne važi (nije kompletan) upit. Ujedno FROM klauzula je jedina obavezna klauzula koja je neophodna (formalno) za bilo koji upit. Opis formata FROM klauzule je

FROM *naziv_tabele*$_1$ [*skraćeni_naziv_tabele*$_1$]
[,..., *naziv_tabele*$_n$ [*skraćeni_naziv_tabele*$_n$]]

U ovom delu SELECT rečenice se definiše skup tabela iz kojih sistem za rukovanje bazama podataka stvara Dekartov proizvod. Po pravilu kompletan Dekartov proizvod izveden na osnovu nabrajanih tabela nije potreban ni za jedan konkretan upit, stoga u sledećim delovima SELECT rečenice se smanjuje (filtrira, selektuje) broj redova kompletnog Dekartovog proizvoda formulisanjem odgovarajućih uslova. U najvećem broju slučajeva za povezivanje dve tabele se koristi equijoin veza na osnovu ključa i spoljašnjeg ključa (povezivanje onih redova dve tabele

za koje važi jednakost-ekvivalencija ključa i spoljašnjeg ključa). U manjem broju praktičnih primera može se desiti povezivanje tabele sa samim sobom (operacija se zove self-join, a primenjuje se kod rekurzivnih veza). U FROM klauzuli je moguće zadati skraćeni naziv tabele u slučaju da ima više naznačenih tabela u klauzuli i u daljim klauzulama je moguća primena skraćenog naziva tabele. Kod rekurzivne veze (kod self-joina tabele) skraćeni nazivi se moraju zadati i, takođe, treba da se razlikuju.

8.3.3. WHERE klauzula

WHERE klauzula, kao što je napred rečeno selektuje redove iz Dekartovog proizvoda više tabela koji se dobija uparivanjem svakog sa svakim redom iz svih tabela. Rezultantna R-tabela će sadržati samo one redove za koje je uslov istinit (koji redovi zadovoljavaju uslov). Sintaktika ove klauzule je vrlo prosta:

WHERE *uslov*

Naznačen *uslov* u klauzuli je logički izraz. Ova klauzula je već praktično obrađena kod naredbe za modifikaciju podataka UPDATE, jer su tamo analizirane sve moguće forme uslova koje se mogu naznačiti I u ovoj klauzuli u okviru *uslov*-a.

Primer:
SELECT * FROM student WHERE telefstu IS NOT NULL;
Tabela rezultata upita (R-tabela) sadrži vrednosti svih polja tabele STUDENT za one studente koji poseduju telefon.

8.3.4. GROUP BY klauzula

To je opcija SELECT naredbe koja služi za grupisanje podataka. Grupu sačinjavaju oni redovi tabele koji po nekom proizvoljno formulisanom kriterijumu imaju iste vrednosti. Format GROUP BY opcije je sledeći:

GROUP BY ime_polja_1 [,..., ime_polja_n]

Na osnovu vrednosti pojedinih polja se sačinjavaju grupe. Redovi koji imaju iste konkretizacije u kolonama tabele sa nazivom *ime_polja1* će spadati u istu grupu. Za grupu(e) se po pravilu primenjuju agregacione funkcije (o kojima je već bilo reči).

Za ovu klauzulu važe sledeća ograničenja:

1. Iza rezervisane reči SELECT se nalaze imena polja po kojima se vrši grupisanje i agregacione funkcije (koje se izvršavaju nad svim grupama ponaosob),
2. Iza rezervisane reči SELECT se mogu nalaziti samo ona imena polja koja se nalaze istovremeno i iza GROUP BY izraza,
3. Redosled imena polja ukazuje na ugneždenje grupa (kako se vrši grupisanje podataka unutar pojedinih grupa definisanih prethodnim imenom polja u GROUP BY klauzuli i
4. Argumenti GROUP BY klauzule mogu biti samo nazivi polja, a ne i izvedena polja, odnosno izrazi.

Primer:

```
SELECT godrodjstu, COUNT(*)
        FROM student
        WHERE telefstu IS NOT NULL
        GROUP BY godrodjstu;
```

To je prava forma pisanja SELECT rečenice (svaka nova klauzula se piše u novi red, a na kraju rečenice se nalazi znak za »tačku«, tj. za završetak rečenice: »;«). Gornji upit lista godine rođenja studenata i ukupan broj studenata koji imaju telefon, a rođeni su te godine koja stoji kao prvi podatak u redu.

8.3.5. HAVING klauzula

HAVING klauzula služi za filtriranje grupa dobijenih GROUP BY klauzulom. Nekad nas, naime, ne interesuju sve grupe dobijene (formirane) GROUP BY klauzulom, tada koristimo ovu klauzulu da bi birali iz raspoloživih grupa. Semantika HAVING klauzule je sledeća:

HAVING *uslov*

Od grupa sastavljenih pomoću GROUP BY klauzule u R-tabelu će dospeti samo one koje zadovoljavaju uslov formulisan u HAVING klauzuli.

Primer:

```
SELECT godrodjstu, COUNT(*) FROM student
        WHERE telefstu IS NOT NULL
        GROUP BY godrodjstu HAVING godrodjstu BETWEEN 1980 AND 1985;
```

Upit u ovom primeru lista godine rođenja studenata i ukupan broj studenata koji imaju telefon, a rođeni su te godine koja stoji kao prvi podatak u redu, ali ne učestvuju sve moguće vrste godine rođenja (sve grupe formirane od strane godrodjstu), već samo grupe (ili godine) od 1980 do 1985 godine.

8.3.6. ORDER BY klauzula

Ova klauzula uređuje podatke (redove) R-tabele po određenom kriterijumu formulisanom u samoj klauzuli. Izgled ORDER BY klauzule je sledeći:

ORDER BY *ime_polja*$_1$ | *redni_broj_polja*$_1$ [ASC|DESC]
[,..., *ime_polja*$_n$ | *redni_broj_polja*$_n$ [ASC|DESC]]

Klauzula uređuje R-tabelu po vrednostima navedenih polja *ime_polja*$_n$. Ako klauzula ima više navedenih polja, sređivanje tabele se vrši po vrednostima prvonavedenog polja, unutar iste vrednosti prvog polja se uređuje R-tabela po vrednostima drugog polja itd. Ograničenja u vezi ove klauzule su sledeća:

1. Polje *ime_polja*$_n$ u ORDER BY klauzuli mora se pojaviti i u SELECT klauzuli.
2. Vrednost *redni_broj_polja*$_n$ označava redni broj polja u SELECT klauzuli.
3. ASC označava uređivanje po rastućem redosledu po vrednosti polja iza kojeg stoji opcija (ova se opcija podrazumeva ako se ništa ne navodi iza imena ili rednog broja polja).
4. DESC izaziva uređivanje po opadajućoj vrednosti polja.

Primer:

```
SELECT imestu, godrodjstu, adresastu
        FROM student
        ORDER BY adresastu, godrodjstu, imestu;
```

Upit lista studente po adresi (iz iste ulice) po godinama rođenja i po imenima (abeceni redosled studenata po grupama sa istom godinom rođenja u okviru iste ulice). Lista obuhvata sve ulice iz baze.

8.3.7. SAVE TO TEMP klauzula

U prethodnom delu je objašnjeno da R-tabela predstavlja privremenu tabelu koja nestaje nakon izdavanja sledeće SQL naredbe. U slučaju da želimo spasiti rezultat nekog upita treba da navedemo ovu klauzulu. Format klauzule je sledeći:

SAVE TO TEMP *naziv_tabele* [ime_polja_1,..., ime_polja_n]

Navođenjem imena polja moguće je preimenovati polja iz prvobitne tabele. Za ovu klauzulu je formulisano samo jedno ograničenje:

1. Ako se u SELECT klauzuli nalazi makar jedno izvedeno polje (bilo kakav izraz ili agregaciona funkcija) dodela imena polja je obavezna.

Primer:

```
SELECT godrodjstu, COUNT(brindeksa)
        FROM student
        GROUP BY godrodjstu
        SAVE TO TEMP pomocna(godrodjstu, broj_studenata_rodjenih_te_godine);
```

Upit daje kao rezultat R-tabelu koja sadrži u jednom redu godinu i broj studenata iz tabele STUDENT rođenih te godine. Rezultat se smešta u pomoćnu tabelu (pomocna) koja može da posluži za dalju obradu. Zbog agregacione funkcije u SELECT klauzuli treba dodeliti imena poljima rezultantne tabele.

8.3.8. Unutrašnji ili ugrađeni SELECT-i

Ugrađeni SELECT-i se mogu smestiti u *uslov* WHERE i/ili HAVING klauzule uz ispoštovanje sledeća dva ograničenja:

1. unutrašnji SELECT ne sme da sadrži ORDER BY klauzulu
2. u ugrađenom SELECT-u se ne može koristiti SAVE TO TEMP klauzula.

Ugrađeni SELECT takođe može da sadrži unutrašnje SELECT rečenice, tj. na taj način se mogu ugrađivati SELECT rečenice jedna u drugu. Unutrašnji SELECT se uvek stavlja u zagradu. Redosled izvršavanja SELECT rečenica je strogo određen: prvo se izvršava unutrašnji, pa tek onda spoljašnji SELECT, i, napokon, upoređuju se dobijene vrednosti. Rezultat (R-tabela) unutrašnjeg SELECT-a se upoređuje sa svakim redom spoljašnjeg (glavnog) SELECT-a.

Ugrađene SELECT rečenice se mogu klasifikovati na osnovu sledeća dva kriterijuma:

1. Broj vrednosti u R-tabeli unutrašnjeg SELECT-a i
2. Broj izvršavanja ugrađene SELECT rečenice

8.3.8.1. Broj vrednosti u R-tabeli ugrađenog SELECT-a

U vezi broja vrednosti u R-tabeli unutrašnjeg SELECT-a mogu nastati dve situacije:

1. R-tabela ugrađenog SELECT-a sadrži samo jednu vrednost
2. unutrašnji SELECT vraća više vrednosti u R-tabeli.

Ako unutrašnji upit ima samu jednu vrednost u *uslov*-u u WHERE i/ili HAVING klauzuli se koristi jedan od logičkih operatora za upoređivanje vrednosti (=, <, >, ≠, ≤, ≥) rezultata unutrašnjeg upita i neke druge vrednosti (konkretne vrednosti nekog konkretnog polja).

Ako ugrađeni upit vraća više vrednosti u *uslov*-u u gore pomenutim klauzulama treba da se koristi jedan od IN, ANY, ALL ili EXISTS predikata u sledećem obliku:

predikat (unutrašnji SELECT)

Uz svaki predikat može da stoji i reč NOT (NOT IN, NOT ANY, NOT ALL; NOT EXISTS). Značenje predikata je sledeće:

- IN – istinita vrednost se vraća ako je vrednosti polja navedenog ispred IN predikata jednaka nekoj vrednost unutrašnje R-tabele. Spoljašnji SELECT pretražuje listu generisanu od strane unutrašnjeg SELECT-a i one rekorde (redove) svrstava u R-

tabelu za koje se vrednost polja (navedeno ispred predikata) nalazi u listi generisanoj od strane ugrađenog SELECT-a.

- ANY – istinitu vrednost vraća ako je naznačeno uprođenje istinito za neku vrednosti unutrašnje R-tabele. Spoljašnji SELECT pomoću ANY predikata selektuje one rekorde (redove) za koje je uslov naveden pre predikata zadovoljen makar za jednu vrednost liste generisane od strane ugrađenog SELECT-a.
- ALL – vraća istinitu vrednost ako je navedeno upoređenje istinito za svaku vrednost unutrašnje R-tabele. Spoljašnji SELECT pomoću ALL predikata selektuje one rekorde (redove) za koje je uslov naveden pre predikata zadovoljen za sve vrednosti liste generisane od strane ugrađenog SELECT-a.
- EXISTS – uzima istinitu vrednost ako lista generisana od strane unutrašnjeg SELECT-a nije prazna.

Za navođenje primera unutrašnjeg SELECT-a sa gore navedenim predikatima neophodno je imati više od jedne tabele. Definišimo zato sledeće dve tabele.

```
CREATE TABLE PREDMET (SIFPREDMETA CHAR(8) NOT NULL UNIQUE,
                      NAZPREDMETA CHAR(20) NOT NULL,
                      GODUVODJPR NUMBER(4),
                      NEDELJNIFONDPRED NUMBER(1),
                      NEDELJNIFONDVEZB NUMBER(1),
                      NEDELJNIFONDLABVEZB NUMBER(1));

CREATE TABLE POLOZISPIT (BRINDEKSA CHAR(8) NOT NULL UNIQUE,
                         SIFPREDMETA CHAR(8) NOT NULL UNIQUE,
                         DATUMISPITA DATE(),
                         OCENA NUMBER(2));
```

Primer: IN predikat

Listajmo imena studenata koji su položili predmet pod nazivom Matematika1

```
SELECT imestu
  FROM student
  WHERE brindeksa IN (SELECT brindeksa FROM polozispit
                      WHERE sifpredmeta = (SELECT sifpredmeta FROM predmet
```

```
                                    WHERE nazpredmeta = "Matematika1"));
```

Napomena: Ako u bazi postoji više (makar dva) predmeta sa istim nazivom "Matematika1", a sa drugim šiframa sifpredmeta onda ugrađeni SELECT drugog nivoa vraća više vrednosti sifpredmeta i znak jednakosti u uslovu WHERE klauzule treba da se zameni predikatom IN, tj. upit će imati sledeći oblik:

```
SELECT imestu
  FROM student
  WHERE brindeksa IN (SELECT brindeksa FROM polozispit
                          WHERE sifpredmeta IN (SELECT sifpredmeta FROM predmet
                                    WHERE nazpredmeta = "Matematika1"));
```

Primer: ANY predikat

Listajmo imena studenata koji imaju veću ocenu iz predmeta "Matematika1" nego neki, bilo koji student rođen 1980. godine.

```
SELECT imestu
  FROM student
  WHERE brindeksa IN (SELECT brindeksa FROM polozispit
                WHERE sifpredmeta IN (SELECT sifpredmeta FROM predmet
                                    WHERE nazpredmeta = "Matematika1") and
              ocena>ANY (SELECT ocena FROM predmet
                    WHERE nazpredmeta = "Matematika1" and
                    brindeksa IN (SELECT brindeksa FROM student
                                    WHERE godrodjstu=1980)));
```

Ugrađeni SELECT navišeg nivoa obezbeđuje brojeve indeksa studenata rođenih 1980. godine. Ove brojeve indeksa koristi sledeći nivo SELECT-a, da bi za njih selektovao sve ocene iz predmeta "Matematika1". Sledeći SELECT manjeg nivoa upoređuje ocenu iz predmeta "Matematika1" za sve studente i bira brojeve indeksa onih studenata koji imaju veću ocenu od bilo koje (ANY) ocene iz predmeta "Matematika1" dodeljenih studentima rođenih 1980. godine. Spoljašnji SELECT služi samo da bi listali imena studenata.

Napomena: Uz ANY predikat je moguće koristiti logičke operatore (relacije) =, < i >. Kombinacija ANY= je ekvivalentna IN predikatu.

Primer: ALL predikat

Listajmo imena studenata koji imaju veću ocenu iz predmeta "Matematika1" od svakog studenta koji je položio pomenuti predmet, a rođen 1980. godine.

```
SELECT imestu
  FROM student
  WHERE brindeksa IN (SELECT brindeksa FROM polozispit
                WHERE sifpredmeta IN (SELECT sifpredmeta FROM predmet
                                WHERE nazpredmeta = "Matematika1") and
                      ocena>ALL (SELECT ocena FROM predmet
                            WHERE nazpredmeta = "Matematika1" and
                            brindeksa IN (SELECT brindeksa FROM student
                                        WHERE godrodjstu=1980)));
```

Napomena: Ovaj upit se može i drugačije rešiti (bez ALL predikata), jer je uslov »ocena studenta iz predmeta "Matematika1" treba da bude veća od svake ocene« je ekvivalentan uslovu »ocena studenta iz predmeta "Matematika1" treba da bude veća od najveće ocene« iz predmeta "Matematika1" za studente rođene 1980. godine. SELECT koji rešava problem na prethodno opisan način ima sledeći oblik:

```
SELECT imestu
  FROM student
  WHERE brindeksa IN (SELECT brindeksa FROM polozispit
                WHERE sifpredmeta IN (SELECT sifpredmeta FROM predmet
                                WHERE nazpredmeta = "Matematika1") and
                      ocena > (SELECT MAX(ocena) FROM predmet
                            WHERE nazpredmeta = "Matematika1" and
                            brindeksa IN (SELECT brindeksa FROM student
                                        WHERE godrodjstu=1980)));
```

Primer: EXISTS predikat

Listajmo imena studenata koji su položili predmet "Matematika1".

```
SELECT imestu
  FROM student
  WHERE EXISTS (SELECT * FROM polozispit
                WHERE sifpredmeta IN (SELECT sifpredmeta FROM predmet
                                      WHERE nazpredmeta = "Matematika1") and
                polozispit.brindeksa=student.brindeksa);
```

8.3.8.2. Broj izvršavanja ugrađene SELECT rečenice

Na početku izlaganja u vezi unutrašnjih SELECT-a rečeno je da se prvo izvršava (vrednuje) unutrašnja, pa tek posle glavna SELECT rečenica. Vrednovanje SELECT-a počinje, znači, od najdublje ugrađenog SELECT-a. Taj scenario izvršavanja važi samo u slučaju kad se unutrašnji SELECT izvršava samo jednom. Prema broju izvršavanja ugrađene SELECT rečenice razlikujemo dva slučaja:

1. unutrašnji SELECT se izvršava samo jednom (najčešći slučaj) i
2. ugrađeni SELECT se izvršava toliko puta koliko spoljašnji SELECT ima redova (unutrašnji SELECT se izvršava jednom za svaki red spoljašnjeg SELECT-a).

U vezi odvijanja algoritma izvršavanja složenog SELECT-a (sa ugrađenom SELECT rečenicom) nema posebnih momenata za objašnjavanje: prvo se izvršava unutrašnji SELECT, a posle spoljašnji SELECT uzimajući u obzir vrednost ili vrednosti koje je obezbedio unutrašnja SELECT rečenica.

Ugrađeni SELECT se izvršava više puta jedino u slučaju kad postoji predaja parametara između unutrašnjeg i spoljašnjeg SELECT-a (kad se u unutrašnjem SELECT-u poziva – pojavljuje promenljiva spoljašnjeg SELECT-a).

Primer: Unutrašnji SELECT se izvršava jednom za svaki red spoljašnjeg SELECT-a.
Listajmo studente (brindeksa) koji su dobili najbolju ocenu iz pojedinih predmeta.

```
SELECT sifpredmeta, brindeksa
        FROM polozispit pisp
        WHERE ocena=(SELECT MAX(ocena) FROM polozispit piun
                     WHERE piun.sifpredmeta=pisp.sifpredmeta);
```

8.4. JEZIK ZA UPRAVLJANJE PODACIMA (DCL)

DCL obuhvata naredbe SQL-a koje se odnose na dodelu i oduzimanje (ograničenje) prava za izvršenje pojedinih tipova operacija u relacionim bazama podataka. Bazu podataka koriste različiti ljudi i, naravno neće moći svi korisnici da obavljaju sve moguće aktivnosti (operacije) vezane za njeno ažuriranje. Pošto je baza podataka jedna u celoj firmi (organizaciji), među tim podacima se nalaze i strogo poverljivi kojima može pristupiti samo mali broj izabranih korisnika. Svi korisnici imaju svoje korisničko ime i lozinku za pristup podacima u bazi podataka, a svako korisničko ime se pridružuje jednoj klasi sa tačno definisanim pravima za izvršenje pojedinih operacija. Kontrola prava pristupa se definiše za svaki objekat u relacionoj bazi kao što je, recimo, tabela.

Operacije se u cilju kontrole prava pristupa tabelama za klase korisnika dele na sledeće četiri grupe:

1. SELECT – upit u tabelu,
2. INSERT – dodavanje redova u tabelu,
3. DELETE – brisanje redova iz tabele i
4. UPDATE – modifikacija podataka u tabeli.

Na taj način se za svaku tabelu (objekat) mogu definisati privilegije (operacije koje se mogu izvršiti sa njihove strane) pojedinim korisnicima. Naravno postoji vrsta korisnika sa »urođenim« ili podrazumevanim najvećim pravima pristupa, a to su, naravno, administratori baze podataka.

Dodela privilegija se realizuje pomoću naredbe GRANT koja ima sledeću sintaksu:

GRANT ALL PRIVILEGES | *vrsta_privilegije*
    ON *naziv_tabele*
    TO PUBLIC | *korisnik* [WITH GRANT OPTION];

Vrednost *vrsta_privilegije* je jedna od gore navedene četiri operacije: SELECT, INSERT, DELETE i UPDATE. Na mestu *korisnik*-a treba da stoji korisničko ime (može i više) korisnika. Ako je naznačena PUBLIC navedene opcije (ili sve ako u izrazu stoji ALL PRIVILEGES) se odnose na sve korisnike. Ako je navedena WITH GRANT OPTION opcija, naznačeni korisnik će imati pravo dodele privilegija.

Oduzimanje privilegije(a) se može realizovati pomoću REVOKE SQL naredbe (koja omogućava oduzimanje svih privilegija odjednom ili jende ili nekoliko navedenih privilegija označenom korisniku – slično kao kod GRANT naredbe):

REVOKE ALL PRIVILEGES | *vrsta_privilegije*
  ON *naziv_tabele*
  FROM PUBLIC | *korisnik*;

Značenje delova SQL naredbe *vrsta_privilegije*, *naziv_tabele* i *korisnik* su već poznate na osnovu prethodnih SQL naredbi.

9. ISPITNA PITANJA

1. Šta je informatika (Čime se bavi informatika)?
2. Interdisciplinarni karakter informatike po opširnoj definiciji.
3. Pojam informatike po sažetoj definiciji.
4. Koje tri suštine treba da ima u vidu informatičar pri projektovanju informacionog sistema?
5. Kako su povezani realni sistem, informacioni sistem i sistem sredstava?
6. Šta čini sistem sredstava i kako su delovi sistema sredstava povezani?
7. Šta je hardver?
8. Šta je softver?
9. Šta predstavlja lifeware?
10. Šta obuhvata orgver?
11. Šta je proces sticanja saznanja (koje aktivnosti obuhvata)?
12. Šta je podatak, šta informacija, a šta je znanje?
13. Jezik kao sredina za prenos saznanja.
14. Od čega se sastoji potpuni iskaz – izjava (potpuna rečenica)?
15. Važnost prisustva četiri dimenzije saznanja.
16. Činjenični način obrade saznanja (vrste, nazivi moguće obrade).
17. Tekstualni način obrade saznanja (vrste, nazivi moguće obrade).
18. Šta je banka podataka?
19. Šta je informacioni sistem?
20. Kako je nastala koncepcija baze podataka i šta ona obuhvata?
21. Šta je model podataka?
22. O kojim osobinama realnog sistema treba da sadr+i model podataka?
23. Šta su statičke osobine sistema?
24. Šta su dinamičke osobine sistema?
25. Šta su ograničenja?
26. Koje su strukturalne komponente modela podataka?
27. Koji su primitivni akoji složeni koncepti strukturalne komponente modela podataka?
28. Kakvi modeli se grade pomoću koncepata i pravila za izgradnju?
29. Veze između dva tipa entiteta i njihova dva nivoa apstrakcije.
30. Intenzija i ekstenzija strukturalne komponente podataka.
31. Šta je integritetna komponenta modela podataka?

32. Koje vrste ograničenja postoje?
33. Kada je baza podataka konzistentna (neprotivrečne)?
34. Šta opisuje operacijska komponenta modela podataka?
35. Od čega se sastoji (gradi) operacija?
36. Kako se realizuje selekcija podataka?
37. Koje su moguće aktivnosti u okviru operacije
38. Šta je navigacija?
39. Šta karakteriše proceduralne jezika?
40. Šta je karakteristično za deklarativne jezike?
41. Šta pokriva pojam apstrakcije?
42. Koje vrste apstrakcije se koriste za izradu modela podataka?
43. Koji su najpoznatiji modeli podataka?
44. Koji modeli podataka su bitni u procesu projektovanja baza podataka?
45. Kako se realizuje proces projektovanja i implementacije baza podataka?
46. Šta je informacioni sistem?
47. Koje su tri projekcije informacionog sistema?
48. Koje su karakteristike projekcije podataka?
49. Opiši projekciju obrade!
50. Šta je logička nezavisnost podataka?
51. Šta obuhvata projekcija okruženja?
52. Šta je fizička nezavisnost podataka?
53. Apstrakcioni nivoi informacionog sistema.
54. Šta obuhvataju pojedini apstrakcioni nivoi?
55. Koji problemi nastaju mešanjem nivoa?
56. Koliko modela podataka treba izraditi pri projektovanju?
57. Šta je pojmovni model (konceptualni plan) podataka?
58. Šta je logički model (plan) podataka?
59. Šta je fizički model (plan) podataka?
60. Šta je model entiteta i poveznika (ER model podataka)?
61. Kako se predstavlja ER model podataka?
62. Koji su osnovni koncepti ER modela?
63. Šta je entitet, klasa entiteta i tip entiteta?
64. Uloga apstrakcije i formulisanju klase entiteta i tipa entiteta.
65. Šta je obeležje, vrednost obeležja i domen?

66. Relativnost pojmova entitet, obeležje i veza.
67. Šta je identifikator (ključ)?
68. Koja dva uslova treba da ispuni identifikator?
69. Šta je superključ?
70. Šta je kandidat za ključ?
71. Koje vrste identifikacije postoje?
72. Kakva je nominalna identifikacija?
73. U čemu je suština deskriptivne identifikacije?
74. Šta je "veštačka" identifikacija i kojoj vrsti identifikacije pripada?
75. Koje su tri projekcije saznanja?
76. Razlika između intenzije klase entiteta i tipa entiteta.
77. Šta je ekstenzija tipa entiteta?
78. Šta je transverzija?
79. Šta je poveznik i tip poveznika?
80. Kakve mogu biti povezanosti sa tačke gledišta entiteta koji su u vezi?
81. Šta je intenzija ER modela i kako se predstavlja?
82. Nivoi detaljnosti intenzije ER dijagrama.
83. Grafički simboli ER dijagrama.
84. Šta je ekstenzija ER modela i kako se predstavlja?
85. Šta obuhvata integritetna komponenta ER modela?
86. Šta je kardinalnost veze i kako se one dele po kardinalnosti?
87. Opšti prikaz kardinaliteta tipa poveznika.
88. Preslikavanja u okviru tipa poveznika.
89. Parcijalna (opciona) i totalna (obavezna) veza.
90. Kardinalitet grupe M:N.
91. Kardinalitet grupe 1:N.
92. Kardinalitet grupe 1:1.
93. Šta su rekurzivne veze?
94. Šta je slabi tip entiteta?
95. Šta je identifikaciono zavisan tip entiteta?
96. Šta predstavlja integritet domena?
97. Šta je nula (null) vrednost?
98. Šta je operacijska komponenta ER modela?
99. Koje klase operacija sadrži operacijska komponenta ER modela?

100. Koje su operacije ažuriranja baze podataka?
101. Šta su strukturna pravila integriteta?
102. Koja su strukturna pravila integriteta?
103. Koja su proširenja ER modela?
104. Šta je superklasa, a šta potklasa?
105. Specijalizacija i generalizacija kod izgradnje superklase-potklase.
106. Kardinalnost poveznika u IS_A hijerarhiji (superklasa-potklasa).
107. Šta je kategorija?
108. Šta je gerund?
109. Heuristička uputstva za izradu ER modela.
110. Relativnost predstavljanja realnog sveta konceptima ER modela.
111. Pojava, tvorac i ciljevi relacionog modela.
112. Jezik za manipulaciju podacima relacionog modela i osnove za njegov razvoj.
113. Koncepti relacionog modela na nivou intenzije i ekstenzije.
114. Šta je relacija u matematičkom smislu?
115. Šta se smatra relacijom u relacionom modelu?
116. Kako se predstavljaju relacije?
117. Šta je šema relacije?
118. Šta je ključ šeme relacije?
119. Šta je šema baze podataka?
120. Koja su ograničenja u relacionom modelu?
121. Šta predstavljaju ograničenja domena?
122. Kakva ograničenja sačinjavaju integritet entiteta?
123. Šta je spoljni ključ?
124. Pojam prostog, složenog i hijerarhijskog ključa.
125. Šta je funkcionalna zavisnost i kako se predstavlja?
126. Šta je jaka, a šta slaba funkcionalna zavisnost?
127. Šta je potpuna, a šta nepotpuna funkcionalna zavisnost?
128. Funkcionalna zavisnost kao vremenski promenljiva funkcija.
129. Šta je tranzitivna zavisnost?
130. Šta je zavisnost sadržavanja?
131. Šta je referencijalni integritet?
132. Čemu služi operacijska komponenta relacionog modela?
133. Vrste operatora relacione algebre za formulisanje upita.

134. Šta su unja, presek i razlika?
135. Kako se definiše dekartov proizvod?
136. Šta je selekcija,a šta projekcija?
137. Šta je theta spoj?
138. Kako se definiše deljenje?
139. Kako se realizuje relacioni model podataka?
140. Šta je normalizacija?
141. Šta je svrha normalizacije?
142. Koje vrste anomalije su poznate?
143. Koje su poznate normalne forme?
144. Koji su metodi za realizaciju normalizacije?
145. Šta je suština dekompozicije (bez gubitaka), a šta sinteze?
146. Šta je nenormalizovana forma šeme relacije?
147. Kada je šema relacije u 1NF?
148. Kako se šema relacije u 0NF prevodi u šemu relacije u 1NF?
149. Kada je šema relacije u 2NF?
150. Kako se šema relacije u 1NF prevodi u šemu relacije u 2Nf?
151. Kada se šema relacije nalazi u 3NF?
152. Kako se šema relacije u 2NF prevodi u šemu relacije u 3Nf?
153. Zašto treba prevoditi ER model podataka u relacioni model?
154. Koji je model podataka semntički bogatiji: ER model ili relacioni model?
155. Da li preslikavanje između ER modela i relacionog modela može biti tipa 1:1? Zašto?
156. U šta se prevodi tip entiteta ER modela?
157. Koliko ima pravila prevođenja za tip entiteta ER modela?
158. U šta se prevodi gerund?
159. Kako se prevodi slabi tip entiteta?
160. Kako se prevodi identifikaciono zavisan tip entiteta?
161. Kako se prevodi IS_A hijerarhija?
162. Kako se prevodi kategorija?
163. Kako se predstavlja povezik u relacionom modelu?
164. Kako se prevode veze sa kardinalnošću (1,1) : (1,1)?
165. Kako se prevode veze sa kardinalnošću (0,1) : (1,1)?
166. Kako se prevode veze sa kardinalnošću (0,1) : (0,1)?
167. Kako se prevode veze sa kardinalnošću (1,1) : (0,N) i (1,1) : (1,N)?

168. Kako se predstavljaju veze sa kardinalnošću (0,1) : (0,N) i (0,1) : (1,N)?
169. Kako se predstavljaju veze sa kardinalnošću M:N?
170. Šta je karakteristično pri prevođenju rekurzivnog tipa poveznika?
171. Zašto postoje posebna pravila integriteta?
172. Kako se formulišu posebna pravila integriteta?
173. Kakva mogu biti posebna pravila integriteta?
174. Šta definišu statička, a šta dinamička pravila integriteta?
175. Koliko ima posebnih pravila integriteta za entitete?
176. Na šta se konkretno odnose posebna pravila integriteta za entitete?
177. Koliko ima posebnih pravila integriteta za veze sa kardinalnošću 1:1?
178. Koliko ima posebnih pravila integriteta za veze sa kardinalnošću 1:N?
179. Koliko ima posebnih pravila integriteta za veze sa kardinalnošću M:N?
180. Kako se razvijao SQL jezik?
181. Koji standardi jezika postoje?
182. Da li je SQL jezik za upravljanje bazama podataka?
183. Kako se klasifikuju naredbe SQL jezika (koji su "podjezici")?
184. Kako se SQL može primeniti za pisanje standardnih aplikacija?
185. Šta je DDL?
186. Koje su naredbe DDL-a?
187. Kakva je sintaksa naredbe za generisanje tabele?
188. Kako se definiše pogled?
189. Da li je pogled fizička tabela?
190. Šta je DML?
191. Za šta služe DML naredbe?
192. Šta je sintaksa unosa podataka (redova) u tabelu?
193. Kakva je sintaksa naredbe za modifikaciju podataka u tabeli?
194. U WHERE klauzuli, u uslovu koje operacije mogu da se primene?
195. Kako se koriste relacioni operatori BETWEEN, IN, LIKE i IS NULL?
196. Šta je sintaksa naredbe za brisanje podataka?
197. Šta realizuju naredbe COMMIT i ROLLBACK?
198. Za šta služi SELECT rečenica?
199. Šta je rezultat izvršenja SELECT rečenice?
200. Da li se mogu povezati upiti (SELECT rečenice) i ako jeste, kako?
201. Kako izgleda struktura kompletne SELECT rečenice i šta označavaju pojedini delovi?

202. Šta sadrži minimalna struktura SELECT rečenice?
203. Kojim redosledom se izvršavaju klauzule SELECT rečenice?
204. Kakva je forma SELECT klauzule SELECT rečenice?
205. Koja ograničenja važe za elemente liste polja projekcije?
206. Koje su funkcije za agregiranje podataka?
207. Koja je sintaksa COUNT funkcije?
208. Koje su mogućnosti SUM funkcije?
209. Kako se traži prosek pomoću AVG funkcije?
210. Kako se koriste MIN i MAX funkcije?
211. Za šta služi FROM klauzula i kakva joj je sintaksa?
212. Šta je specifično u FROM klauzuli u slučaju rekurzivnih veza?
213. Čemu služi WHERE klauzula?
214. Kako se koristi GROUP BY klauzula (šta je forma)?
215. Koja ograničenja važe za GROUP BY klauzulu?
216. Za šta služi HAVING klauzula?
217. Kako se uređuju redovi R-tabele (koja je sintaksa naredbe)?
218. Koja su ograničenja ORDER BY klauzule?
219. Za šta služi SAVE TO TEMP klauzula (i šta je njen format)?
220. Koja su ograničenja prisutna kod SAVE TO TEMP klauzule?
221. Šta su ugrađeni SELECT-i?
222. Koja ograničenja važe za unutrašnje SELECT rečenice?
223. Kakav je redosled izvršavanja ugneždenih SELECT rečenica?
224. Na osnovu kojih kriterijuma se klasifikuju ugrađeni SELECT-i?
225. Kakvi mogu biti unutrašnji SELECT-i sa tačke gledišta broja vrednosti u R-tabeli?
226. Koji logički operatori za upoređivanje stoje u uslovuu ako unutrašnji upit vraća samo jednu vrednost?
227. Koji logički operatori za upoređivanje stoje u uslovuu ako unutrašnji upit vraća više vrednosti?
228. Kako se koriste IN, ANY, ALL i EXISTS predikati?
229. Koliko puta može biti izvršen unutrašnji SELECT?
230. Zašto se može izvršiti unutrašnji SELECT i više puta?
231. Šta je DCL?
232. Kod kontrole prava pristupa tabelama kako se grupišu operacije?
233. Na kom nivou se može odrediti kontrola prava pristupa korisniku?

234. Kako se vrši dodela privilegije korisniku?

235. Format SQL naredbe za oduzimanje dodeljene privilegije.

10.LITERATURA

1. Bana I.: Az SSADM rendszerszervezési módszertan, LSI, Budapest, 1994.
2. Bennatan E.M.: SoftverProject Management: A Practitioner's Approach, Second Edition, McGRAW-HILL Book Company Europe, 1995.
3. Čarapić M., Pendić Z., Furht B., Veselinović D.: Osnovi projektovanja informacionih sistema zasnovanih na primeni računara, Tehnička knjiga, Beograd, 1974.
4. Date C.J.: "An Introduction to Database Systems", Vol. 1, Third Edition, Addison-Wesley Publ. Co., Reading, MA, 1981.
5. Đurković J., Tumbas P.: Analiza i projektovanje informacionih sistema – CASE STUDY, EF Subotica – ALEF Novi Sad, 1997.
6. Görög M.: Általános projektmenedzsment, Aula Kiadó, Budapest, 1996.
7. Halassy B.: Az adatbázis-tervezés alapjai és titkai, IDG, Budapest, 1994.
8. Halassy B.: Az információs rendszerek alapfogalmai, SZÁMALK, Budapest, 1982.
9. Halassy B.: Adatbázisok kezelésének alapvető kérdései, SZÁMALK, Budapest, 1982.
10. Halassy B.: Ember – Információ – Rendszer, IDG, Budapest, 1996.
11. Halassy B.: Adatmodellezés, Nemzeti Tanakönyvkiadó, Budapest, 2002.
12. Kuzmanov S., Mogin P., Petković I., Popović M.: "Automatizacija projektovanja šeme baze podataka", JAHORINA '88.
13. Lazarević B., Jovanović V., Vučković M..:"Projektovanje informacionih sistema" I deo, Naučna knjiga, Beograd, 1986.
14. Lazarević B.: "Prošireni model objekti-veze" interni materijal, Laboratorija za informacione sisteme FON-a, Beograd, april 1990.
15. Lazarević B.: Baze podataka, Radni materijal za interne kurseve, Beograd, 1989.
16. Lerner A.J.: Principi kibernetike, Tehnička knjiga, Beograd, 1970.
17. Marjanović Z.: ORACLE – relacioni sistem za upravljanje bazom podataka, Breza, Beograd, 1990.
18. Mogin P., Luković I., Govedarica M.: Principi projektovanja baza podataka, FTN-STYLOS, Novi Sad, 2000.
19. Mogin P., Luković I.: Principi baza podataka, FTN-STYLOS, Novi Sad, 1996.
20. Purba S., Sawh D., Shah B.: How to Manage a Succesfull Software Project, John Wiley & Sons, 1995.
21. Strukturirana sistemska analiza, IskraDelta, Ljubljana, 1984.
22. Šuc L.: Projektovanje baza podataka, Izobraževalni center DELTA, Ljubljana, 1985.
23. Szelezsán J.: Adatbázisok, LSI, Budapest, 1996.
24. Ullman J.D.: "Principles of Database Systems", Computer Science Press, 1980.
25. Ullmann J.D., Widom J.: A First Course in Database Systems, Prentice Hall Inc., 1997. (prevod na mađarski jezik, PANEM, Budapest, 1997.).
26. Veljović A.: Modeliranje podataka – Erwin (materijal za kurs), CIT, Beograd, 1998.
27. Veljović A.: Modeliranje procesa – Bpwin (materijal za kurs), CIT, Beograd, 1998.
28. Mihajlović D.: Informacioni sistemi i projektovanje baza podataka, FTN, Novi Sad, 1998.
29. Roger S. Pressman: Software Engineering – A practitioner's Approach, The McGraw-Hill Companies, Inc., New York, 1997.
30. Ian Sommerville: Software Engineering, Addison-Wesley, Harlow, England, 1997.
31. B. Jošanov, P.Tumbas: Softverski inženjering, Viša poslovna škola, Novi Sad, 2002.
32. Eric J. Naiburg, Robert A. Maksimchuk: UML za projektovanje baza podataka, Addison-Wesley (2001)-CET(2002), Beograd.
33. Chris Todman: Projektovanje skladišta podataka (Designing a Data Warehouse), Prentice Hall PTR (2001)- CET (2001), Beograd.

34. A. Silberschatz, H. F. Korth, S. Sudarshan: Instrucror's Manual to Accompany Database System Concepts, Mcgraw-Hill Companies, Inc., Boston, 1997.
35. A. Silberschatz, H. F. Korth, S. Sudarshan: Database System Concepts, Mcgraw-Hill Companies, Inc., Boston, 1997.