RAZVOJ INFORMACIONIH SISTEMA i baze podataka

ALEMPIJE VELJOVIĆ

"Nesebična pomoć potencijalnom kvalitetu početnika da se dokaže, vredi više od svih mogućih ličnih uzleta".

prof. dr Vladimir B. Šolaja, dipl.ing.

Uvod

U vreme ekspanzije zahteva za što raznovrsnijim podacima sve je potrebnije i dobijanje kvalitetnih informacija. Ovu potrebu treba da zadovolji razvoj informacionih sistema i odgovarajućih sistema za upravljanje podacima (SUBP). Oni bi trebalo da integrišu teoretske postavke vezane za korišćenje metodologije *odozgo nadole (Top-Down)* sa implementacijom koja se izvodi metodologijom *odozdo nagore (Bottom-Up).*

Metodologija odozgo nadole se izvodi sa stanovišta rukovodećeg menadžmenta preduzeća, korišćenjem metode intervjua. Na taj način se definišu ciljevi, procesi, resursi i dr. Obrnutim putem, metodologijama odozdo nagore (analizom dokumenata) izvodi se generisanje baza podataka. Širinu u pristupu daje metodologija odozgo nadole, a preciznost metodologija odozdo nagore.

Za uspešno izvodenje aktivnosti vezanih za razvoj informacionih sistema potrebno je na samom početku raskrstiti sa zabludama, definisati ograničenja i dati odgovarajuće pretpostavke.

Najčešće zablude su:

- da neko drugi može obaviti ovaj posao, po principu "ključ u ruke";
- da se naručivanjem projekta može brže završiti posao;
- da se preslikavanjem postojećih aplikacija u novo hardversko i softversko okruženje može doći do novog sistema.

Kada se raskrsti sa zabludama, na sledećem koraku javljaju se *ograničenja* koja, pak, mogu sa svoje strane usporiti razvoj informacionih sistema, imajući u vidu:

- stepen organizovanosti sistema koji se analizira i koji zavisi od razrađenosti standardnih dokumenta i procedura za njihovu obradu i distribuiranje;
- ograničenja vezana za korisnike, koja su u vezi sa odbojnošću prema ideji uvođenja novog sistema;
- znanje projektnog tima, njihovu metodologiju rada i iskustvo za slične sisteme koji mogu biti od presudnog značaja.

Imajući u vidu ograničenja, treba ispuniti tri osnovne *pretpostavke* :

- Prva pretpostavka je vezana za *jedinstven sistem označavanja* i podrazumeva definisanje najčešće tzv. paralelnog sistema označavanja. Paralelnim sistemom označavanja se definišu: jedinstven identifikacioni broj, standardizovan naziv i odgovarajući klasifikacioni broj. Jedinstven identifikacioni broj ili IDENT BROJ je neimenovani redni broj (najčešće od šest cifara). Naziv je definisan po standardu JUS A.A0.006 i ima tačno propisanu strukturu. Klasifikacioni broj definiše grupe PREDMETA POSLOVANJA i svako mesto ima odgovarajuće značenje (do pet cifara).
- Druga pretpostavka odnosi se na *jedinstvenost modela procesa i podataka,* gde se podrazumeva, obično, primena jedinstvene metodologije vezane za projektovanje informacionog sistema korišćenjem CASE alata (Computer Aidid Software Engineering). Korišćenje metodologije projektovanja informacionog sistema IDEF0 i IDEF1X, tj. CASE alata BPwin i Erwin predmet je razmatranja ove knjige.

- Treća pretpostavka je vezana za *jedinstvenost sistema za upravljanje bazama podataka (SUBP).* U ovoj knjizi je usvojen, odnosno biće razmatran SUBP MS ACCESS.

Prevazilaženjem zabluda i ograničenja, uz poštovanje definisanih pretpostavki, mogućan je razvoj informacionog sistema koji će omogućiti definisanje nove strategije vođenja preduzeća. Ta strategija obezbeđuje integraciju svih informacionih tokova u preduzeću, a na osnovu toga i upravljanje procesima.

Modeliranje kao osnova razvoja informacionog sistema

Razvojem informacionog sistema (IS) treba definisati što objektivniju sliku realnog sveta, njegovih bivših i sadašnjih stanja, kao podlogu za procenu budućeg ponašanja i naravno, podlogu za dalji razvoj i primenu informatičke tehnologije.

Za opis rada poslovnog sistema veliki je problem to što ne mogu da se koriste prirodni jezici, zbog mnogih jezičkih dvosmislenosti. S druge strane, precizan opis preko formalnih jezika je nerazumljiv za većinu ljudi.

Stoga je potrebna tehnika koja će organizovati prirodne jezike na taj način da eliminiše dvosmislenost i omogući efikasnu komunikaciju i razumevanje. Pokazalo se da je postupak modeliranja jedna od najefektivnijih tehnika za razumevanje i jednoznačnu komunikaciju između projektanata i korisnika.

U procesu modeliranja, eliminišu se detalji, čime se umanjuje vidljiva kompleksnost sistema koji se proučava. Grafičke prezentacije (uglavnom pravougaonici i linije) koriste se da bi obezbedile da većina ljudi razmišlja o procesu modeliranja kao o slikovitoj prezentaciji (jedna slika zamenjuje 1000 reči). Pored grafičkog prikaza, potrebno je dati i precizne definicije predmeta koji se pojavljuju u modelu, kao i propratni tekst, koji je kritičan prema modelu koji ima svoju ulogu, kao sredstvo komunikacije.

Ovakav pristup nametnuo je potrebu za apstrakcijom, kojom se izvodi kontrolisano isključivanje detalja, tj. izvlače se zajedničke karakteristike u opisivanju nekog sistema. Tako je na višim nivoima apstrakcije sistem opisan jasnije, a na nižim detaljnije.

S druge strane, još uvek u velikim firmama postoje hardverske konfiguracije gde je korisnički softver razvijen, obično, u jeziku treće generacije (najčešće COBOL), bez odgovarajuće prateće dokumentacije, mada preduzeća žele da pređu na relativno jeftin i moćan kompjuterski sistem, obično je to mreža PC, definisana po principima klijent/server arhitekture.

S obzirom na to, modeliranje treba da:

- bude "jezik" za komunikaciju između korisnika i analitičara i
- omogući preciznu i formalizovanu "specifikaciju zahteva".

Treba, dakle, još jednom posebno istaći postupak modeliranja realnog sistema, koji zavisi od *sposobnosti, znanja i iskustva projektnog tima,* jer se ne mogu dati stroga formalna pravila modeliranja koja bi vodila do jedinstvenog modela složenog realnog sistema, bez obzira na to ko vrši modeliranje. Mogu se dati samo opšte metodološke preporuke, opšti metodološki pristupi, kao pomoć u tom složenom poslu.

Standardi
kao podrška modeliranju

Postupak razvoja informacionih sistema opisan u ovoj knjizi ima za osnovu standarde IDEF0 i IDEF1X, koji podržavaju modeliranje. Istorijski gledano, tokom 60-ih i 70-ih godina Douglas T. Ross je razvio tehniku modeliranja poznatu kao SADT (Structured Analysis & Design Technique). Prihvatajući SADT tehniku, avijacija SAD razvila je SADT kao deo ICAM (Integrated Computer Aided Manufacturing) programa tokom kasnih 70-ih, koji je dobilo naziv IDEF tehnika (**I**ntegation **DEF**inition). Cilj ICAM programa je bio da se poboljša proizvodna produktivnost primenjivanjem kompjuterske tehnologije. Učesnici u izgradnji ICAM programa su uvideli sve prednosti korišćenja IDEF tehnike, jer tekstualni opis ne predstavlja efikasan način za dokumentovanje procesa i podataka.

U ranim 90-im, IDEF Users Group, u kooperaciji sa National Institutes for Standards and Technology (NIST), preduzela je određene postupke za stvaranje standarda *IDEF0* za funkcionalno modeliranje i *IDEF1X (eXtend),* kao tehniku za informaciono modeliranje (semantičko modeliranje podataka), publikujući ih 1993. godine (U.S. Government standards documents). Ovi standardi su pod pokroviteljstvom Institute of Electrical and Electronics Engineers (IEEE), a prihvatila ih je i International Organization of Standards (ISO).

Cilj modeliranja je da se razviju tehnologije koje će omogućiti logičku i fizičku integraciju mreža hardverski i softverski veoma različitih konfiguracija. Tehnika IDEF modeliranja je prihvaćena kao osnova za sprovođenje postupka reinženjeringa poslovnih procesa.

Imajući u vidu sve ove činjenice, funkcionalno modeliranje IDEF0 omogućuje:

- izvršenje funkcionalne dekompozicije i dizajna na svim nivoima, za sistem sastavljen od ljudi, mašina, materijala, računara i informacija;
- stvaranje dokumentacije, paralelno sa reinženjeringom poslovnih procesa;
- bolju komunikaciju između projektnog tima, korisnika i menadžera;
- diskusiju u projektnom timu da bi se postiglo međusobno razumevanje;
- upravljanje velikim i složenim projektima;
- obezbeđenje elemenata potrebnih za informaciono modeliranje (IDEF1X metodologija).

Softverska realizacija IDEF0 standarda je BPwin (Bussines Process windows) firme LogicWorks (2) koji se koristi u ovoj knjizi.

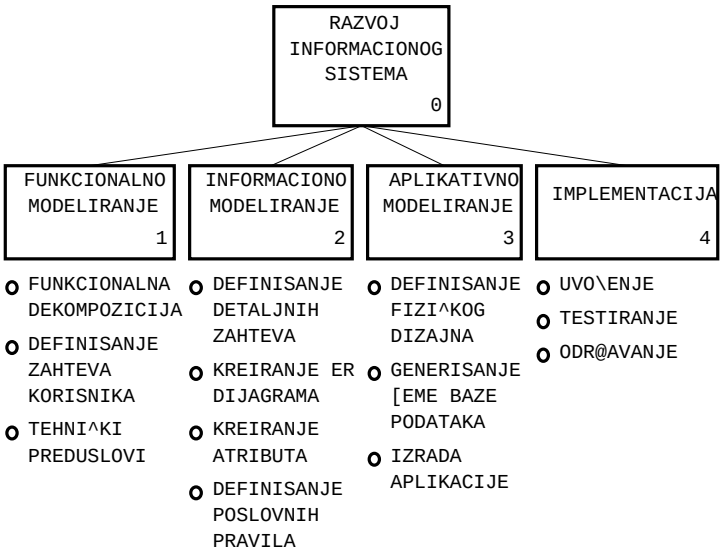
Drugi standard koji je IDEF Users Group definisala je *IDEF1X* tehnika za informaciono modeliranje. *Informaciono modeliranje (IDEF1X)* predstavlja apstraktno viđenje realnog sistema, tj. to je pojednostavljeno predstavljanje realnog sistema preko skupa objekata (entiteta), veza između objekata i atributa objekata. Informaciono modeliranje je pojam koji je definisan u okviru IDEF1X metodologije i definiše odgovarajući model podataka.

Zapravo, IDEF1X je semantički bogat modelar podataka treće generacije koji je realizovan u okviru softvera ERwin (Entity Relationships for windows) CASE alata (3). ERwin CASE alat omogućuje generisanje u neki od sistema za upravljanje bazama podataka (SUBP). Mora se naglasiti pojam *generisanje,* a ne programiranje, jer ERwin omogućuje da se direktno kreiraju tabele, veze, atributi i sva ograničenja koja su se nekada programirala.

Postupak razvoja informacionog sistema

Imajući u vidu postavke vezane za IDEF0 i IDEF1X metodologiju, kao i potrebe za reinženjeringom poslovnih procesa, može se reći da se razvoj informacionog sistema (RIS) izvodi kroz četiri sledeće faze (slika 1):

- Aktivnost 1. Funkcionalno modeliranje,
- Aktivnost 2. Informaciono modeliranje,
- Aktivnost 3. Aplikativno modeliranje i
- Aktivnost 4. Implementacija.



Slika 1. Stablo aktivnosti postupka razvoja informacionog sistema

S obzirom na to da se sve četiri aktivnosti detaljno razmatraju u posebnim poglavljima, ovde će biti dat samo kratak opis.

Prva aktivnost *"1. Funkcionalno modeliranje"* treba da omogući postavljanje modela, tj. definisanje studije koja koncipira reinženjering poslovnih procesa u širinu.

Ova aktivnost se definiše kroz tri podaktivnosti, kao:

- Aktivnost 1.1. Funkcionalna dekompozicija,
- Aktivnost 1.2. Definisanje zahteva korisnika,
- Aktivnost 1.3. Tehnički preduslovi.

U okviru podaktivnosti *"1.1. Funkcionalna dekompozicija"* polazi se od **svesti o potrebi razvoja informacionog sistema, i to** povezane, pre svega, sa donošenjem strategijske odluke rukovodećeg menadžmenta o sprovođenju reinženjeringa poslovnih procesa. Kao rezultat treba dobiti stablo poslovnih procesa kako ih vidi vodeći menadžment.

Podaktivnost *"1.2. Definisanje zahteva korisnika"* treba da omogući da se analizom dokumenata i sprovođenjem intervjua može **identifikovati okvir reinženjeringa poslovnih procesa.** Potrebno je pronaći i dobro osmotriti nepotpune (prekinute, isprekidane), neperspektivne procese koji guše ostvarivanje planiranih (željenih) rezultata. Ključ "leži" u odgovoru na pitanje: "Šta treba promeniti?". Drugim rečima, u ovoj fazi se proučavaju i procenjuju postojeći procesi, analiziraju se rezultati koje daje proces sada i kakvi se rezultati mogu očekivati u budućnosti. Dakle, snimkom trenutnih procesa omogućuje da se uoče uska grla (gde se nalaze) i problematične tačke.

Takođe, u ovoj aktivnosti treba predvideti i **alternativne prilaze.** Kada se definiše problem, timovi koji sprovode reinženjering ističu novi strateški pravac za realizaciju procesa i pridružena merila, a ujedno vrše i procenu novih poslovnih alternativa. Dakle, naročita pažnja se obraća na to da članovi tima treba da razumeju proces, tj . da imaju razjašnjene odgovore na pitanja šta se želi postići, zašto je potreban redizajn i kako treba da izgleda proces u budućnosti.

Funkcionalno modeliranje zahteva i odgovarajuće **tehničke preduslove** koji se definišu kroz odgovarajući hardver i softver, kadrovske potrebe i dinamiku realizacije, što treba definisati u okviru podaktivnosti *"1.3. Tehnički preduslovi"*. Rezultat ove faze rada je "studija " ili "idejni projekat" kao dokument za aktivnost "*2. Informaciono modeliranje*".

Aktivnost *"2. Informaciono modeliranje"* je ključni momenat gde do izražaja dolaze sposobnost i znanje visokostručnog kadra iz oblasti menadžmenta i informatike. U ovoj fazi poželjno je angažovanje i spoljnih eksperata.

Ova aktivnost se definiše kroz sledeće četiri podaktivnosti:

- Aktivnost 2.1. Definisanje detaljnih zahteva,
- Aktivnost 2.2. Kreiranje ER modela,
- Aktivnost 2.3. Kreiranje atributa i
- Aktivnost 2.4. Definisanje poslovnih pravila.

U okviru aktivnosti *"2.1. Definisanje detaljnih zahteva"* definišu se procesi redizajniranja. To zavisi od kriterijuma važnosti procesa i njene narušenosti, kao i od mogućnosti sprovođenja izmena. U ovoj fazi treba utvrditi koja stara pravila ostaju i koji se novi procesi pojavljuju, zatim izvršiti spajanje odgovarajućih operacija ili eliminisati nepotrebne i utvrditi logičan redosled koraka u procesu. Kao rezultat ovog rada treba da bude definisano detaljno stablo aktivnosti sa odgovarajućim detaljnim dekompozicionim dijagramima (po IDEF0 metodologiji) i verifikacijom top-menadžmenta preduzeća.

Aktivnost *"2.2. Kreiranje ER modela",* korišćenjem IDEF1X metodologije, predstavlja kvalitetno novi skok, jer treba da bude kreacija projektanata informacionog sistema. Do ovog trenutka, korišćenjem IDEF0 metodologije, opisivana je dinamika rada, što je prisutno kao iskustvo i tradicija u svakom preduzeću i što je definisano kroz aktivnost "1.Funkcionalno modeliranje".

Ova aktivnost otvara "crnu kutiju", koja je budućim korisnicima uvek bila nepoznata, jer nisu mogli da prate razmišljanja projektanata informacionog sistema. Prvi put korisnici uzimaju aktivno učešće i u ovom delu i prvi put projektanti informacionog sistema crtaju ono što predstavlja njihovo iskustvo i saznanje o poslovanju konkretnog preduzeća i što su oni osmislili u svojoj glavi.

Kroz identifikaciju entiteta, odnosno kroz definisanje objekata od interesa za posmatranje i definisanje veza definiše se ER model, postupkom *odozgo nadole,* tj. intervjuom sa budućim korisnicima.

Sledeća aktivnost *"2.3. Kreiranje atributa"* treba da da opis osobina u prethodno definisanim entitetima. Osobine entiteta se definišu kroz identifikaciju atributa za svaki entitet, definisanje odgovarajućih ključeva i sprovođenja postupka normalizacije. Ova aktivnost se izvodi postupkom *odozdo nagore,* tj. analizom dokumenata.

Aktivnost *"2.4. Definisanje poslovnih pravila"* predstavlja sintezu prethodne dve aktivnosti i treba da definiše poslovna ograničenja i pravila ponašanja.

Treća aktivnost: *"3. Aplikativno modeliranje"* posmatra se sa stanovišta izabranog sistema za upravljanje bazama podataka (SUBP).

Ova aktivnost se posmatra kroz sledeće tri podaktivnosti:

- Aktivnost 3.1. Definisanje fizičkog dizajna,
- Aktivnost 3.2. Generisanje šeme baze podataka,
- Aktivnost 3.3. Izrada aplikacije.

Aktivnost *"3.1. Definisanje fizičkog dizajna"* razmatra problematiku vezanu za izgradnju sistema za upravljanje bazama podataka (SUBP).

Aktivnost *"3.2. Generisanje šeme baze podataka"* definiše se za izabranu ciljnu platformu, gde se definišu fizičke tabele, kolone i relacije.

Aktivnošću *"3.3. Izrada aplikacije"* treba da se realizuje korisnički pogled na podatke, tj. da se definišu meniji, forme, upiti i izveštaji.

I na kraju, aktivnost *"4. Implementacija"* omogućuje izvođenje promena vezanih za način rukovođenja i primene informacionih tehnologija.

Ova aktivnost se posmatra kroz sledeće tri podaktivnosti:

- Aktivnost 4.1. Uvođenje,
- Aktivnost 4.2. Testiranje,
- Aktivnost 4.3. Održavanje.

Aktivnost *"4.1. Uvođenje"* treba da da ocenu urađene korisničke aplikacije, omogući izmene u toku uvođenja, izradi uputstva za korisnike i obuči same korisnike.

Aktivnost *"4.2. Testiranje"* treba da omogući testiranje sprovedenih aktivnosti u okviru postavljenog SUBP gde se ocenjuju performanse tog sistema.

Aktivnost *"4.3. Održavanje"* se izvodi kad se pređe u fazu eksploatacije novopostavljenog sistema. Ovde se drastično pokazuju sve manjkavosti i neregularnosti vezane za sprovedeni reinženjering poslovnih procesa i definišu odgovarajuće korektivne akcije.

Ono što omogućuje fleksibilno izvođenje svih aktivnosti prikazanih na slici 1.1. i što zaokružuje ceo ovaj posao je CASE alat, kojim se omogućuju automatsko registrovanje svih izmena i ažurno održavanje projektne dokumentacije.

Aktivnost 1. Funkcionalno modeliranje

Aktivnost 1.1. Funkcionalna dekompozicija

Aktivnost 1.2. Definisanje zahteva korisnika

Aktivnost 1.3. Tehnički preduslovi

Aktivnost *"1. Funkcionalno modeliranje"* omogućuje dekomponovanje poslovnih funkcija i planiranje potrebnih resursa za realizaciju funkcija. Funkcionalno modeliranje je vezano za korišćenje IDEF0 tehnike. IDEF0 je tehnika modeliranja aktivnosti baziranih na kombinaciji grafike i teksta, koji su predstavljeni na organizovan i sistematičan način da bi se povećala razumljivost, koja podržava analizu, obezbeđuje logiku za potencijalne izmene, specificira zahteve, ili, rečeno na drugi način, podržava analizu sistema po nivoima i integriše aktivnosti.

IDEF0 funkcionalni model se sastoji od hijerarhijskog niza dijagrama koji postepeno prikazuju sve više detalja o funkcijama i njihovoj međuvezi (interface) sa ostalim delovima sistema. IDEF0 modeliranje omogućuje analizu osobina određenog poslovnog procesa radi njegovog maksimalnog unapređenja.

Razlozi koji su motivisali nastanak IDEF0 funkcionalnog modeliranja su:

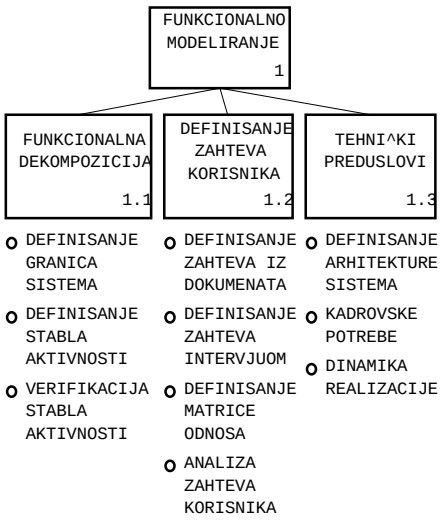
Prvo, služi kao dokumentacija i uputstvo za opis kompleksnih poslovnih procesa. Poznata je činjenica da što je dokumentacija veća - to se manje čita. Tačnije, dokument od jedne ili dve strane sa grafičkim prikazom biće najverovatnije pregledan. Dokument od 30 strana ima sve izglede da mesecima ne bude pročitan.

Drugo, omogućava brze organizacione promene, jer model procesa dokumentuje važne aktivnosti i omogućava uvid u kritične aktivnosti koje treba izvesti sa odgovarajućim resusima, što je bitan element u održavanju reinženjeringom definisanih poslovnih procesa.

Treće, što je i najvažnije, koristi kao prototipski pristup funkcionalnom modeliranju gde se na brz i jednostavan način proveravaju alternativne ideje. Mnogo je jednostavnije i jeftinije nacrtati model i proveriti ga na "papiru", nego izvršiti reorganizaciju sektora. To je veoma bitna osobina, jer brzi razvoj informacionih tehnologija uslovljava potrebu za reinženjeringom poslovnih procesa.

Na slici 2.1. prikazana je aktivnost *"1. Funkcionalno modeliranje"* kojom se definišu tri podređene aktivnosti:

- Aktivnost 1.1. Funkcionalna dekompozicija,
- Aktivnost 1.2. Definisanje zahteva korisnika,
- Aktivnost 1.3. Tehnički preduslovi.



Slika 2.1. Aktivnost 1. Funkcionalno modeliranje

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 2.1.

Aktivnost
1.1. Funkcionalna dekompozicija

Prilikom realizacije aktivnosti *"1.1. Funkcionalna dekompozicija"* posebna pažnja se poklanja zahtevima top-menadžmenta**,** jer se, zbog brzih promena u novim tržišnim uslovima, iz osnova menjaju koncept, principi, arhitektura, funkcije, prioriteti i dr. U **preduzeću,** praktično, treba da se reflektuje viđenje poslovanja vodećeg menadžmenta i da uspeh reinženjeringa poslovnih procesa zavisi od njihovih postavki datih u obliku vizija, misija i definisanih ciljeva.

Aktivnost *"1.1. Funkcionalna dekompozicija"* izvodi se kroz tri podređene aktivnosti:

- Aktivnost 1.1.1. Definisanje granica sistema,
- Aktivnost 1.1.2. Definisanje stabla aktivnosti,
- Aktivnost 1.1.3. Verifikacija stabla aktivnosti.

U daljem tekstu detaljno će biti obrazložene definisane aktivnosti.

Aktivnost
1.1.1. Definisanje granica sistema

Aktivnost *"1.1.1. Definisanje granica sistema"* je vezana za nabrajanje objekata koji će u sledećem koraku biti po hijerarhiji povezani u stablo aktivnosti.

U okviru utvrđivanja granica sistema treba jasno definisati ciljeve koji moraju da sadrže sledeće elemente:

- zašto se proces modelira;
- šta će proces da prikaže;
- šta će korisnik modela napraviti sa njim;
- čemu služi model.

Odgovori na ova pitanja treba da pomognu u fokusiranju postavljene problematike. Sledeća pitanja na koja treba dati odgovor su:

- koji su zadaci na datom radnom mestu;
- koji je redosled izvođenja koraka;
- kako se izvodi kontrola;
- koji se resursi koriste.

Dakle, treba identifikovati zadatke svakog zaposlenog i shvatiti odnose između zadataka. Za izvođenje ovih aktivnosti koristi se grafički jezik IDEF0. IDEF0 tehnika je svojevrsan grafički jezik koji omogućuje komunikaciju, razumljivu svim učesnicima u postupku reinženjeringa poslovnih procesa.

Da bi se realizovale naredne aktivnosti, u daljem tekstu detaljno će biti obrazložena struktura grafičkog jezika IDEF0.

Struktura grafičkog jezika IDEF0

Grafički jezik IDEF0 opisuje metodu funkcionalne dekompozicije preko skupa dijagrama, od kojih svaki predstavlja ograničenu količinu detalja definisanih odgovarajućom sintaksom i semantikom. Dijagrami su međusobno povezani tako da opisuju sistem, hijerarhijski, sa vrha naniže. Dijagrami se sastoje od pravougaonika koji predstavljaju neki deo celine. Povezani su međusobno usmerenim linijama koje predstavljaju veze između delova.

Postoje tri vrste IDEF0 prikaza: grafički, tekstualni i rečnik (glossary). Grafički prikaz definiše funkcije i veze funkcija preko pravougaonika i strelica i odgovarajuće sintakse i semantike. Tekst i rečnik pružaju dodatne informacije i podržavaju grafičke dijagrame.

Sintaksa grafičkog jezika IDEF0

Sintaksu grafičkog jezika IDEF0 čine pravougaonici (boxes), strelice (arrows) i pravila (rules).

Pravougaonik

Pravougaonici predstavljaju aktivnosti, definisane kao funkcije, procesi i transformacije (slika 2.2). Svaki pravougaonik ima naziv i broj u okviru granica pravougaonika. Za naziv aktivnosti se koristi aktivan glagol ili glagolska fraza koja opisuje funkciju. Broj se koristi da bi bio prepoznat predmet opisa pravougaonika u pridruženom tekstu.

Slika 2.2. Sintaksa pravougaonika (Box)

Aktivnost definisana u okviru pravougaonika ima tri karakteristike:

- naziv,
- vremensku dimenziju,
- rezultat rada.

Prvo, aktivnost mora imati *naziv,* tj. da ime aktivnosti ima, obično, strukturu formata tipa \glagol\subjekt\. Za svaki naziv mogu se dati definicije koje ne smeju biti duge, ali bi trebale u potpunosti da objasne svaku aktivnost.

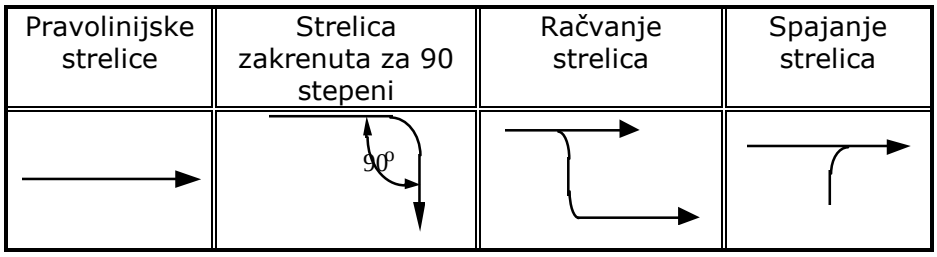
Drugo, aktivnost ima *vremensku dimenziju,* tj. određeno vreme koje mora proći između početka i kraja aktivnosti. Pre nego što se definiše nešto kao aktivnost, mora se imati u vidu da se u trenutku trajanja aktivnosti troši energija, koja može biti fizička, mehanička ili električna.

Treće, sve aktivnosti moraju dati *rezultat,* tj. odgovarajući izlaz. Aktivnosti koje ne proizvode odgovarajući rezultat mogu se definisati kao aktivnosti, ali samo zbog opisa, onakvog kakav je on u stvarnosti. Međutim, takve aktivnosti najpre će biti eliminisane.

Sledeći element sintakse grafičkog jezika IDEF0 je strelica.

Strelice (Arrows)

Strelica se sastoji od jedne ili više linija, sa vrhom strelice na jednom kraju. Strelice mogu biti pravolinijske ili savijene pod uglom od 90 stepeni i mogu se račvati ili spajati (slika 2.3).



Slika 2.3. Sintaksa strelica

Strelice predstavljaju podatke ili objekte vezane za aktivnosti. One ne znače samo tok ili sekvencu, kao u tradicionalnom modelu dijagrama toka podataka, već prenose podatke ili objekte vezane za posmatranu aktivnost.

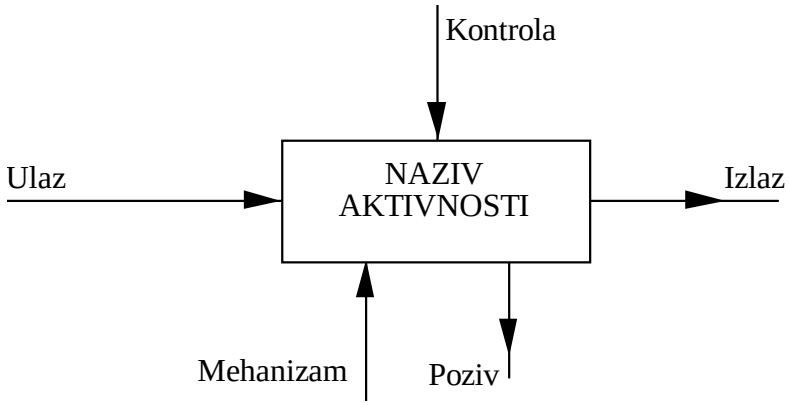
Svaka strelica je definisana nazivom (imenicom). Za opis naziva strelice se definiše i odgovarajući tekstualni opis.

U daljem tekstu detaljno će biti opisana semantika grafičkog jezika IDEF0.

Semantika grafičkog jezika IDEF0

Semantika grafičkog jezika IDEF0 ukazuje na značenje sintaksne komponente jezika i olakšava korektnost interpretacije kojom se opisuje notacija za aktivnosti i strelice.

Odnos između aktivnosti i strelica je određen pomoću strane pravougaonika (aktivnosti) na koji je strelica naslonjena (slika 2.4).



Slika 2.4. Pozicija strelica i uloge

Strelice sa leve strane pravougaonika se definišu kao ulazi (Input). *Strelice koje ulaze u pravougaonik odozgo* se definišu kao kontrole (Control). *Strelice koje izlaze* iz pravougaonika na desnoj strani predstavljaju izlaze (Output). Izlazi su podaci ili objekti, odnosno proizvodi aktivnosti.

Dakle, elementi prikazani na slici 2.4. mogu se opisati rečenicom: "Ulazi se preko aktivnosti transformišu u odgovarajući izlaz, dok kontrole specificiraju uslove pod kojima aktivnost daje korektan izlaz".

Strelice na donjoj strani pravougaonika predstavljaju mehanizme. Strelice okrenute

prema gore identifikuju značenje koje podržava izvršenje aktivnosti. Strelice mehanizma koje su okrenute nadole definišu se kao strelice poziva (Call arrows).

Imajući u vidu englesku notaciju, dijagrami se zovu i ICAM dijagrami, jer je to skraćenica od:

- **I** - Input, nešto što se upotrebljava u aktivnosti;
- **C** - Control, kontrole ili uslovi izvođenja aktivnosti;
- **O** - Output, rezultat izvođenja aktivnosti;
- **M** - Mehanizam, nešto što se koristi u aktivnosti ali se ne menja.

Imajući u vidu navedene postavke, postavlja se pitanje: koje resurse nose pojedini tipovi strelica.

Ulazna (Input) strelica predstavlja materijal ili informaciju koja se koristi ili transformiše radi definisanja izlaza (Output). Dozvoljava se mogućnost da određene aktivnosti ne moraju imati ulazne strelice.

Kontrolne (Control) strelice regulišu, odnosno odgovorne su za to kako, kada i da li će se aktivnost izvesti, odnosno kakvi će biti izlazi (Output). Svaka aktivnost mora imati najmanje jednu kontrolnu strelicu.

Kontrole su često u obliku pravila, politika, procedura, ili standarda. One utiču na aktivnost, ali ne mogu da budu transformisane ili upotrebljene. U slučaju da je cilj aktivnosti da promeni pravilo, politiku, proceduru ili standard, treba očekivati da će strelice koje sadrže tu informaciju, u stvari, biti ulaz.

Izlazne (Output) strelice su materijali ili informacije stvorene aktivnošću. Svaka aktivnost mora imati najmanje jednu izlaznu (Output) strelicu. Ne treba modelirati aktivnost koja ne stvara izlaz.

Strelice mehanizama su izvori koji izvode aktivnosti, a sami se ne "troše". Mehanizmi mogu biti ljudi, mašine i/ili oprema, tj. objekti koji obezbeđuju energiju potrebnu za izvođenje aktivnosti. Po slobodnoj volji projektanta, strelice mehanizama mogu biti i izostavljene iz aktivnosti.

Strelica poziv (Call) specifični je slučaj strelice mehanizma i ona označava da pozivajući pravougaonik nema vlastiti detaljniji dijagram, već daje detaljniji prikaz izveden na nekom drugom pravougaoniku u istom ili nekom drugom modelu. Više pozivajućih pravougaonika mogu pozivati isti pravougaonik na nekom drugom ili istom modelu. Imenuju se brojem dekompozicionog dijagrama, koji sadrži pozvani pravougaonik zajedno sa brojem pozivnog pravougaonika.

Kao školski primer, koji treba da posluži za prikaz IDEF0 metoda modeliranja, definisan je test-primer dokumenta "Karton isplata", prikazan na slici 2.5.

Primer dokumenta "Karton isplata"

Na slici 2.5. prikazan je pojednostavljen primer ručnog dokumenta "Karton isplata" koji će dalje služiti za opisivanje IDEF0 metodologije.

KARTON ISPLATA			
SIFRA RADNIKA	IME I PREZIME	STRANI JEZIK	ODELJENJE
7359	Zoran Starcevic	Engleski, Ruski	Razvoj
RADNO MESTO: Tehnolog		Francuski	
R.BR. ISPLATE	IZNOS ISPLATE	DATUM ISPLATE	PRIMEDBE
01	150,00	08.12.1996	
02	450,00	20.02.1997	
03	800,00	30.09.1997	

Slika 2.5. Ručni dokument "Karton isplata"

Preduzeće koje se razmatra, bavilo se uvozno-izvoznim poslovima, ali se zbog potreba na tržištu počelo baviti prevodilačkim poslom. Za ovu novu aktivnost bilo je potrebno napraviti dokument koji bi omogućio *praćenje vanrednih isplata prevodilaca* (slika 2.5). Dokument "Karton isplate" je klasičan primer loše urađenog ručnog dokumenta (koji se sastoji od zaglavlja i stavki) i treba da posluži kao primer kako se od ručnog dokumenta dolazi do gotove aplikacije. Na ovom jednostavnom primeru treba pokazati kako se izvodi prevođenje klasičnih ručnih dokumenata (pa i dokumenata sistema kvaliteta) u informatički razumljivu formu. *To je prvi korak koji treba da omogući integraciju informacionog sistema i zahteva sistema kvaliteta ISO 9000, kao polaznih elemenata vezanih za reinženjering poslovnih procesa.*

Na primeru dokumenta "Karton isplata" biće prikazane sve aktivnosti definisane na slici 1.1.

Na osnovu definisane sintakse i semantike grafičkog jezika IDEF0, pristupa se prvom koraku, tj. definisanju kontekstnog dijagrama.

Kontekstni dijagram

Prvi korak koji se preporučuje u aktivnosti *"1.1.1. Utvrđivanje opštih zahteva"* predstavlja izrada kontekstnog dijagrama. Na taj se način definišu okviri IDEF0 modela.

Kontekstni dijagram je definisan jednim pravougaonikom koji predstavlja *granicu modela* koji se proučava. U tom sistemu i van njega teku informacije preko strelica. Kontekstni dijagram je najviši nivo apstrakcije koji se dekompozicionim dijagramima prevodi u niži nivo apstrakcije. Na slici 2.6. prikazan je kontekstni dijagram pod imenom "Praćenje isplata" za dokument "Karton isplata" sa slike 2.5.

Granice modela se definišu da bi se, pre svega, znalo gde treba stati sa modeliranjem.

Ovaj problem se može posmatrati sa aspekta:

- širine (definisanja elemenata koji se posmatraju) i

- dubine (definisanja nivoa detaljnosti).

Širina modela je vezana za definisanje kontekstnog dijagrama (koji se u IDEF0 notaciji označava sa A0) i prvog nivoa dekompozicije nosi oznaku A1. U okviru kontekstnog dijagrama mora se voditi računa da treba definisati setove ulaza, kontrola i mehanizama koji proizvode set izlaza, tj. treba na ovom nivou uopštiti posmatranu problematiku sa manje detalja.

Dubina modela se definiše nivoima dekomponovanja, gde se definišu nivoi detaljnosti. Dekompozicija ide do mogućnosti definisanja programskih modula, koji se mogu opisati dijagramom toka podataka, tzv. Data Flow Diagram (DFD). O tome će biti više reči u opisu aktivnosti *"2.1.3. Definisanje dijagrama toka podataka".*

Aktivnost A0, koja se pojavljuje u kontekstnom dijagramu, opisuje okvire modela i mora biti određena aktivnom glagolskom frazom, kao, npr. "Praćenje isplata" (slika 2.6).

Preporučuje se da treba početi od definisanja izlaznih strelica, pa se pomerati prema ulazima, mehanizmima i kontrolama. Polazi se od činjenice da svaka aktivnost poseduje odgovarajuće izlaze koji se mogu identifikovati. Prilikom definisanja izlaza treba voditi računa i o negativnim izlazima, koji prouzrokuju tzv. povratne (feedback) strelice.

Sledeći elementi koje treba definisati su strelice ulaza, koji se na specifičan način transformiše (ili troši) radi stvaranja odgovarajućeg izlaza, potpomognut odgovarajućim mehanizmima i kontrolom.

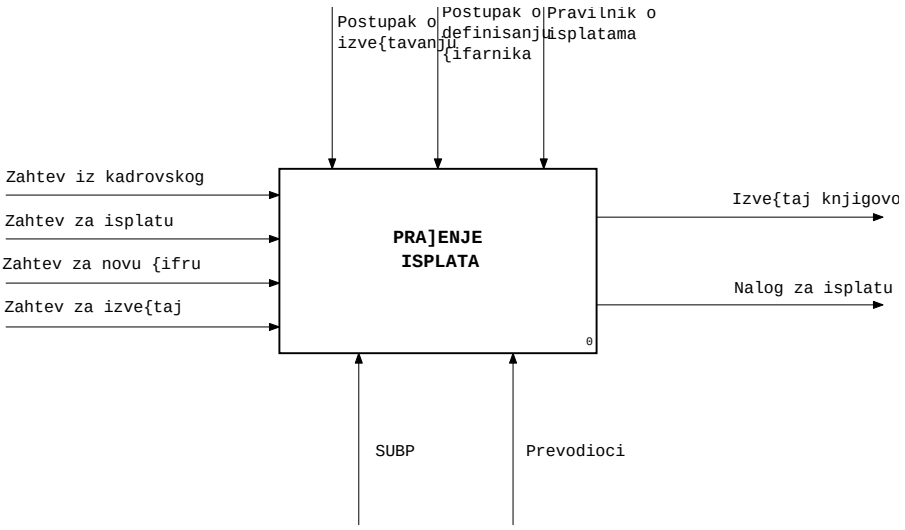
Na kraju treba proveriti:

- da li kontekstni dijagram obuhvata aktivnosti koje se modeliraju;
- da li je kontekstni dijagram konzistentan sa svrhom, uglom posmatranja i granicama;
- da li strelice uspostavljaju odgovarajući nivo detalja (ne sme ih biti više od šest po tipu strelice);
- da li su model prihvatili svi članovi grupe.

Imajući u vidu definisane pretpostavke, u daljem tekstu biće prikazan kontekstni dijagram za dokument "Karton isplata".

Kontekstni dijagram za dokument "Karton isplata"

Za pojednostavljeni primer dokumenta "Karton isplata" prikazani su svi elementi kontekstnog dijagrama "Praćenje Isplata".



Slika 2.6. Kontekstni dijagram za aktivnost "Praćenje isplata"

Kao što se na slici 2.6. vidi, sama aktivnost definisana je glagolskom frazom "Praćenje

isplata", dok su strelice definisane i grupisane kao:

- ulazni dokumenti:
  - "Zahtev iz kadrovskog", kojim se definiše zahtev za otvaranje novog kartona;
  - "Zahtev za isplatu", kojim se definiše zahtev za isplate prevodiocima;
  - "Zahtev za novu šifru", kojim se u šifarniku definiše nova šifra;
  - "Lista isplata", kojom se definiše lista isplata radnika;
  - "Zahtev za izveštaj", kojim se definiše tip izveštaja;
- izlazni dokumenti:
  - "Izveštaj knjigovodstvu", gde se knjigovodstveno prate isplate;
  - "Nalog za isplatu", kojim se omogućuje isplaćivanje preko žiro-računa.
- Kontrole su uputstva i postupci i za ovaj primer su:
  - postupak o izveštavanju;
  - postupak o definisanju šifarnika;
  - pravilnik o isplatama.
- Mehanizmi se definišu kao:
  - relacioni sistem za upravljanje bazom podataka (RSUBP);
  - prevodioci.

Imajući u vidu ovako postavljeni kontekstni dijagram, u sledećem koraku se definiše stablo aktivnosti.

Aktivnost
1.1.2. Definisanje stabla aktivnosti

Na osnovu definisanih granica sistema prelazi se na sledeću aktivnost *"1.1.2. Definisanje stabla aktivnosti"*, gde se uspostavljaju vertikalne (hijerarhijske) veze između aktivnosti.

Stablo aktivnosti se definiše primenom metode rešavanja problema odozgo nadole (top- -down), kada se složena aktivnost rastavlja na više podređenih aktivnosti, a zatim se pristupa rešavanju jednostavnih podređenih aktivnosti.

Drugim rečima, polazna složena aktivnost razvija se u hijerarhiju podređenih aktivnosti, čija je struktura tipa stabla. Koren stabla (to je najviši čvor stabla) sadrži polaznu aktivnost, dok listovi, tj. čvorovi koji nemaju potomke, sadrže aktivnosti čije je rešavanje relativno jednostavno. Rešavanjem svih podređenih aktivnosti iz listova rešena je i polazna složena aktivnost. Za aktivnost *"1.2. Definisanje zahteva korisnika"* ide se do trećeg nivoa, dok se detalji razmatraju u okviru aktivnosti *"2.1.1. Izrada detaljnog stabla aktivnosti".*

Dakle, stablo aktivnosti predstavlja hijerarhiju definisanih aktivnosti, očišćenu od strelica, i omogućuje funkcionalnu dekompoziciju i uvid u dubinu odvijanja veza između aktivnosti.

Ovaj pristup je veoma važan kada se uspostavlja okvir reinženjeringa poslovnih procesa koji mogu doći u razmatranje.

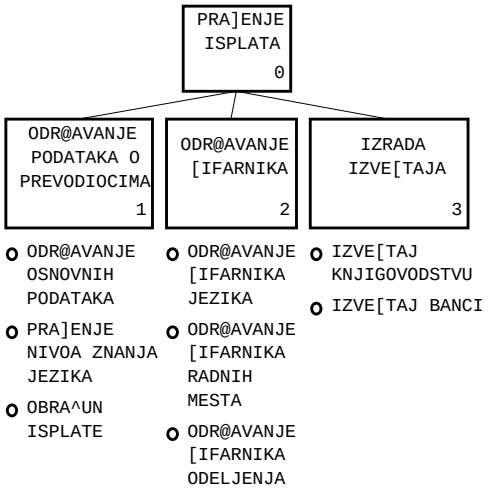
Aktivnost na vrhu (root) uvek je označena sa 0. Brojevi se koriste da bi prikazali koliko detalja sadrži aktivnost. Aktivnost A0 je dekomponovana (razdvojena) na 1, 2, 3 itd. Aktivnost 1 je dekomponovana u 11, 12, 13 itd. Nadređena aktivnost se zove 'roditelj' (parent), a podređene aktivnosti su deca (childs).

Razbijanje aktivnosti 'roditelj' na svoju decu treba da ima od dve do šest podređenih aktivnosti. Ako je više od šest podređenih aktivnosti, to znači pokušaj da se smesti previše detalja na jedan nivo.

Na osnovu svega što je dosad rečeno, treba definisati stablo aktivnosti za aktivnost "Praćenje isplata".

Stablo aktivnosti za aktivnost "Praćenje isplata"

Za primer dokumenta "Karton isplata" (prikazanog na slici 2.5), odnosno za kontekstni dijagram sa slike 2.6. prikazano je stablo aktivnosti na slici 2.7.



Slika 2.7. Stablo aktivnosti za kontekstnu aktivnost "Praćenje isplata"

Prilikom formiranja ovog stabla aktivnosti pošlo se od opšte činjenice da se većina dokumenata može podeliti na:

- održavanje podataka,
- održavanje šifarnika i
- izradu izveštaja.

Na osnovu analize dokumenta "Karton isplata" i sprovedenih intervjua, aktivnost *"1.Održavanje podataka o prevodiocima"* može se podeliti na podređene aktivnosti:

- Aktivnost 1.1. Održavanje osnovnih podataka, gde se definišu osnovni podaci o prevodiocima;
- Aktivnost 1.2. Praćenje nivoa znanja jezika, gde se definišu odgovarajući sertifikati;
- Aktivnost 1.3. Obračun isplata za obavljanje prevodilačke aktivnosti.

Aktivnost *"2. Održavanje šifarnika"* obuhvata:

- Aktivnost 2.1. Šifarnik jezika, gde se definišu šifra i standardni naziv stranog jezika;
- Aktivnost 2.2. Šifarnik radnih mesta, gde se definišu šifra i standardni naziv radnih mesta;
- Aktivnost 2.3. Šifarnik odeljenja, gde se definišu šifra i standardni naziv odeljenja.

Aktivnost *"3. Izrada izveštaja"* obuhvata:

- Aktivnost 3.1. Izveštaj knjigovodstvu, gde se izvode odgovarajući knjigovodstveni poslovi;
- Aktivnost 3.2. Izveštaj banci, kojim se izveštava o uplati na žiro-račun za obavljeni posao.

Dalja podela zavisi od postupka vođenja posla. Prikazano stablo aktivnosti treba u sledećem koraku obavezno da prihvati i verifikuje verifikaciono telo, o čemu će u daljem tekstu biti više reči.

Aktivnost
1.1.3. Verifikacija stabla aktivnosti

Aktivnost *"1.1.3.Verifikacija stabla aktivnosti"* veoma je važna, jer je to strateški momenat, tj. odluka rukovodstva da li je to ono što se želi postići reinženjeringom poslovnih procesa. Kao veoma bitna aktivnost, ona prva mora biti usaglašena i verifikovana. Ovaj elemenat se mora razmatrati prilikom klasičnog načina uvođenja dokumenata za obezbeđenje kvaliteta, po standardu ISO 9000.

Verifikaciono telo treba stalno da kontroliše rad stručnog tima, uz blagovremeno otklanjanje eventualnih grešaka i nedoslednosti u njihovom radu. S druge strane, zbog postojanja ovog tela, nije potreban supervizor projekta, jer je projekat kontrolisan i verifikovan u svim fazama izrade.

Verifikaciono telo ne mora da bude stalnog sastava; u zavisnosti od sadržaja koji se verifikuje, ono može da ima uži ili širi sastav.

Drugi veliki kvalitet je učestvovanje dela rukovodstva preduzeća u izradi projekta, od čijeg angažovanja isključivo zavisi kasnija uspešna realizacija projektovanih rešenja.

Nakon verifikacije stabla aktivnosti od strane top-menadžmenta, pristupa se definisanju zahteva korisnika koji će da potvrde ili koriguju ovako postavljenu tezu.

Aktivnost
1.2. Definisanje zahteva korisnika

Sa stanovišta projektanta koji izvodi reinženjering poslovnih procesa, aktivnost *"1.2. Definisanje zahteva korisnika"* ključni je momenat. U pitanju su informisanje i "učenje" projektanta, odnosno upoznavanje sa potrebama i željama korisnika, kako bi projektant mogao da uspostavi informacione veze i donese pravilne zaključke.

Ova aktivnost se deli u četiri podaktivnosti (slika 2.1):

- Aktivnost 1.2.1. Definisanje zahteva iz dokumenata;
- Aktivnost 1.2.2. Definisanje zahteva intervjuom;
- Aktivnost 1.2.3. Definisanje matrice odnosa;
- Aktivnost 1.2.4. Analiza zahteva korisnika.

U daljem tekstu detaljno će biti opisane definisane aktivnosti.

Aktivnost
1.2.1. Definisanje zahteva iz dokumenata

Aktivnost *"1.2.1. Definisanje zahteva iz dokumenata"* je pogled odozdo nagore i u nadređenoj aktivnosti *"1.2. Definisanje zahteva korisnika"* ima globalni karakter, jer se prikupljaju dokumenta iz cele firme.

Ako u preduzeću postoje definisane odgovarajuće službe za izradu organizacionih propisa i internih standarda, prikupljanje informacija iz postojeće dokumentacije je olakšano; u suprotnom, što je najčešći slučaj, ova aktivnost može veoma dugo da traje. Veliku pomoć u izvođenju ove aktivnosti mogu pružiti i odgovarajuće procedure i uputstva, definisani po standardu kvaliteta ISO 9000, jer sistematizuju klasičnu "ručnu" dokumentaciju.

Dakle, treba prikupiti:

- ulazne dokumente,
- izlazne dokumente,
- uzorke izveštaja,
- organizacione propise i dr.

Na ovom nivou postoji velika opasnost da se prikupe netačne informacije kojima se opisuje zastareli način rada i zato treba stalno proveravati da li je dobijena poslednja verzija dokumentacije.

Za prikupljenu dokumentaciju, potrebno je dati odgovore na sledeća pitanja:

- odakle potiču podaci;
- kako su podaci dobijeni;
- koliki su maksimum i minimum podataka dobijenih u praksi;

- da li se unose tačni podaci;
- da li se unose kompletni podaci;
- kako se koriste;
- koliko često se koriste;
- koliko je tačan taj izlaz danas;
- u kom obliku je prikazan;
- kada se izrađuje i koliko često;
- koliki su obim i broj kopija;
- kome se upućuje;
- šta startuje izlaz;
- koja se poboljšanja mogu uvesti.

Za postupak rada sa dokumentima definisane su i odgovarajuće procedure i organizacioni propisi, koje, ako postoje, treba proučiti i inovirati postojećom praksom, a ako ne postoje- - treba ih napisati, jer to neposredno utiče na postojeću organizaciju rada.

Analiza dokumenata pomaže analitičaru da formira mišljenje o čitkosti tih dokumenata, kao i da razume korisnikovu terminologiju, da bi u sledećem procesu *"1.2.2. Definisanje zahteva intervjuom"* postavio prava pitanja prilikom sprovođenja intervjua.

Na osnovu svega toga u sledećem koraku treba izvršiti analizu dokumenta "Karton isplata".

Analiza dokumenta "Karton isplata"

Dokument "Karton isplata" nastao je zbog potrebnog evidentiranja prevodilačkih poslova u okviru preduzeća. Dokument "Karton isplata" se smešta u istoimenu kartoteku, u kojoj se evidentiraju sve promene vezane za aktivnosti prevodilaca.

Podaci koji se unose u "Karton isplata" potiču iz kadrovske kartoteke, a na osnovu evidencije o obavljenim prevodilačkim uslugama.

Kao i svi ručni dokumenti, i dokument "Karton isplata" ima nedorečenosti u smislu nedefinisanja svih zahteva koje korisnik može da zahteva, a ne može da ih upiše, jer ne postoje "rubrike". Zbog nedorečenosti dokumenta ostavi se, obično, jedan deo dokumenta, naslovljen kao: "Primedbe" ili " Opis", što pokazuje da onaj koji je projektovao dokument ne poznaje problematiku i da su podaci u dokumentu "Karton isplata" neažurni i netačni i da ne odražavaju trenutno stanje. Ovaj dokument ne omogućava pravljenje različitih analiza, jer je nedorečen i ne sadrži sve informacije potrebne rukovodstvu, pa se stoga ne može "automatizovati", već se mora sprovesti intervju i saznati koje se informacije moraju definisati i projektovati.

Autor ovog teksta je primetio da je prilikom uvođenja sistema kvaliteta dosta dokumenata, upravo, ovakvog tipa. Sa stanovišta informatičara, koji mora da snimi ovakav dokument i omogući unos preko računara, informacije tipa "Opis" su bezvredne. Da bi se tekst iskoristio, treba detaljno čitati sadržaj teksta i na osnovu toga napraviti novi dokument, iz koga će korisnik moći krstićem da izabere jednu iz više ponuđenih kombinacija. Sa druge strane, ovako dobijena informacija se može statistički obraditi, što je jedan od zahteva standarda ISO 9000.

Stoga u sledećem koraku treba izvršiti definisanje zahteva intervjuom.

Aktivnost
1.2.2. Definisanje zahteva intervjuom

Aktivnost *"1.2.2. Definisanje zahteva intervjuom"* je pristup odozgo nadole, i treba da omogući definisanje:

- potreba za informacijama,
- ciljeva i
- problema kako ih vide rukovodioci i neposredni izvršioci.

To je ključni momenat u reinženjeringu poslovnih procesa, jer se ovde rukovodstvo izjašnjava o pitanju budućnosti, odnosno daljem poslovanju. Ova analiza treba da da i odgovore vezane za primenu Interneta u preduzeću.

Prvi korak su opšte pripreme za izvođenje intervjua koje su vezane za definisanje:

- liste rukovodilaca za intervjue,
- vremenskog rasporeda intervjua,
- teme za razgovor,
- potvrde termina,
- organizacije grupe za intervjue,
- vođenje zapisnika,
- priprema panela,
- opremanje prostorije,
- izbor opštih pitanja i
- probni intervju.

Ne sme se zaboraviti da rukovodilac treba da pošalje pismo sa objašnjenjem, svrhom, temama i pitanjima tako da bi se druga strana mogla pripremiti.

Posebno treba naglasiti organizaciju grupe za intervjue, gde treba definisati:

- koji će član tima voditi zapisnik,
- ko će prezentovati dosadašnje rezultate,
- ko će postavljati pitanja.

Teme treba unapred odrediti, jer se posle lakše sređuju. Preporučuje se izvođenje probnog intervjua u okviru članova tima, radi bolje pripreme i uvežbanosti.

Cilj svih aktivnosti je razvoj preporuka za buduće akcije. Naime, aktivnosti treba da omoguće da se za trenutno postojeće objekte poslovanja (organizacione jedinice, procese i dr.), aplikacije i datoteke identifikuje redundantnost podataka, razjasne odgovornosti i, uopšte, razume poslovanje.

Da bi se ostvarile ove aktivnosti pristupa se postupku intervjuisanja, počev od najviših rukovodilaca do neposrednih korisnika.

Treba još jednom naglasiti da intervju zahteva pre svega uključivanje najviših rukovodilaca i sagledavanje problema u poslovanju sa njihovog stanovišta.

Opšta pitanja za intervjuisanje su:

- koje su nadležnosti i odgovornosti;
- šta su osnovni ciljevi i kakve se promene mogu očekivati u određenoj oblasti za sledeću godinu i dalje;
- koji su kritični faktori u pogledu odgovornosti upravljanja i odlučivanja;

- kakvi su zahtevi za informacijama;
- koji su najveći problemi bili u poslednje vreme i šta je bilo potrebno za rešavanje tih problema (koje se informacije i koji efekti mogu očekivati);
- u kojim procesima se mogu postići poboljšanja i koje su potrebe za podacima;
- koje su prikupljene informacije najupotrebljivije.

Po završetku intervjua treba odgovore razmotriti, napisati rezime i izdvojiti uočene probleme.

Jedan od bitnih rezultata intervjua je i uspostavljanje boljih odnosa između članova tima i najviših rukovodilaca, što ima neposrednog uticaja na podršku koju rukovodioci treba da pruže u postupku reinženjeringa poslovnih procesa.

Obično se sprovodi pet do deset intervjua sa najvišim rukovodiocima, u trajanju od dva do tri sata. To je i najduža aktivnost u toku reinženjeringa poslovnih procesa.

Takođe, jedan od bitnih momenata je i analiza problema i njihovo povezivanje sa procesima poslovanja, jer predstavljaju uvodno uputstvo za postavljanje prioriteta vezanih za redosled realizacije pojedinih poslovnih funkcija.

Na kraju se definišu odgovarajući zaključci, koji treba da pokažu rukovodiocima da su njihova gledišta shvaćena i ugrađena u rezultate i zaključke projekta.

Izvršioci ovog procesa su:

- poseban tim za intervjue koji priprema i obavlja intervju, obrađuje odgovore i daje analizu intervjua;
- poseban tim za katalog aplikacija koji snima i radi katalogizaciju postojećih aplikacija;
- eksperti iz domena podsistema ili procesa koji predlažu organizaciono-informaciona rešenja pojedinih procesa ili podsistema;
- korisnici koji daju informacije o informacionim potrebama i zahtevima procesa u koje su uključeni i koji ocenjuju kritičnost procesa na osnovu dogovorenih merila;
- koordinacioni odbor koji:
- koordinira rad posebnih timova,
- usvaja predložena organizaciono-informaciona rešenja,
- određuje merila za ocenu kritičnosti procesa,
- usvaja analize posebnih timova i
- utvrđuje prioritete razvoja podsistema.

U sledećem koraku biće izvršeno definisanje zahteva intervjuom za dokument "Karton isplata".

Definisanje zahteva intervjuom za dokument "Karton isplata"

Za primer dokumenta "Karton isplata" uočene su manjkavosti koje se dopunjuju postupkom intervjua, gde se definišu dopunski zahtevi za informacijama, i to:

- odeljenje gde je prevodilac zaposlen,
- radno mesto na koje je radnik raspoređen,
- stručna sprema koju prevodilac poseduje,
- nivoi znanja jezika,
- primanja vezana za redovnu isplatu,
- stimulacija,

- usluge van preduzeća,
- datum stupanja u preduzeće,
- neposredno nadređeni rukovodilac.

Dakle, sprovedenim intervjuom definisani su zahtevi koji ne postoje u postojećem ručnom dokumentu i koje treba ugraditi u buduće rešenje. Ovaj veoma značajan momenat i zahteva maksimalno angažovanje zaposlenih, a pogotovu vodećeg menadžmenta.

Na osnovu prethodno izvedenih aktivnosti u sledećem koraku treba definisati matricu odnosa.

Aktivnost
1.2.3. Definisanje matrice odnosa

Aktivnost *"1.2.3.Definisanje matrice odnosa"* treba da definiše matricu veza između aktivnosti i dokumenata koji treba da povežu dokumenta određena kao ulazne ili izlazne informacije sa aktivnostima iz stabla aktivnosti na gornjem okvirnom nivou. Dokumenta su definisana odgovarajućim rubrikama, čijom se analizom određuju odgovarajući entiteti.

Entiteti na ovom nivou predstavljaju objekat koji se može opisati nekim osobinama. Za ovu aktivnost bitni su samo nazivi entiteta, bez ulaženja u definisanje osobina.

Svakom entitetu se pridodaje način na koji aktivnost koristi taj entitet, odnosno šta radi sa pojedinim instancama tog entiteta preko tzv. CRUD matrice.

Entitet se u okviru neke aktivnosti: i/ili kreira (C-Create), i/ili pretražuje (R-Retrive) i/ili ažurira (U-Update), i/ili briše (D-Delete), pa otuda i naziv CRUD matrica (definisana početna slova engleskog naziva).

Za aktivnost "Praćenje isplata" na slici 2.8. prikazana je CRUD matrica.

Naziv aktivnosti	*Naziv entiteta*	C R U D
ODRZAVANJE PODATAKA O PREVODIOCIMA	ISPLATA	C R U D
	OSOBA	C R U D
	JEZIK	R
	CERTIFIKAT	C R U D
	ODELJENJE	R
	RMESTO	R
ODRZAVANJE SIFARNIKA	JEZIK	C R U D
	ODELJENJE	C R U D
	RMESTO	C R U D
IZRADA IZVESTAJA	ISPLATA	R
	JEZIK	R
	ODELJENJE	R
	OSOBA	R
	CERTIFIKAT	R

Slika 2.8. CRUD matrica za primer aktivnosti "Praćenje isplata"

Za primer aktivnosti "Praćenje isplata" mogu se definisati sledeće aktivnosti:

- održavanje podataka o prevodiocima,
- održavanje šifarnika i
- izrada izveštaja.

U okviru ovako definisanih aktivnosti se definišu odgovarajući entiteti (OSOBA, ISPLATA, CERTIFIKAT, JEZIK, ODELJENJE, RADNO MESTO). Na slici 2.8. prikazane su CRUD veze između aktivnosti i entiteta.

U ovoj aktivnosti se definišu osnovne postavke dok se u okviru aktivnosti *"2.1.1.Definisanje detaljne matrice odnosa",* detaljnije specificiraju i atributi korišćenja, tzv. IRUN matrica (o čemu će kasnije više biti reči).

Na osnovu izvedenih, prethodno opisanih aktivnosti u sledećem koraku pristupa se analizi zahteva budućeg korisnika softverske aplikacije.

Aktivnost
1.2.4. Analiza zahteva korisnika

Aktivnost *"1.2.4. Analiza zahteva korisnika"* znači da postavke definisane u prethodnim koracima treba da verifikuje odgovarajući verifikacioni organ preduzeća.

Zadatak verifikacionog tela je da usvaja rezultate rada stručnog tima u kontrolnim tačkama koje predstavljaju okončanje pojedinih faza na izradi projekta.

Treba naglasiti da sve navedene aktivnosti i podaktivnosti moraju ići ovim redosledom i da definisanje tehničkih preduslova tek onda dolazi u razmatranje. U praksi se obično, radi naopako.

Aktivnost
1.3. Tehnički preduslovi

Tehnički preduslovi podrazumevaju, pre svega, računarski sistem sa definisanim:

- sistemom hardver (nešto opipljivo);
- sistemom softver (nešto neopipljivo, tj. preneta ljudska znanja i pamet);
- sistemom dokumentacije (opis za sistem hardver i sistem softver).

Sistem hardver čine: radna memorija, masovna memorija, ulazne jedinice, izlazne jedinice i centralna procesorska jedinica. Radna memorija sadrži operativnu (RAM-Random access memory) i postojanu (ROM-Read only memory, PROM-Programable read only memory, EPROM-Eresable programable read only memory) i dr.

U *masovnu memoriju* spadaju: magnetni diskovi, diskete, optički diskovi, magnetne trake, kasete, CD.

Ulazna jedinica omogućuje da se u računar unose instrukcije, programi, podaci; čine je: tastatura, disketna jedinica, miš, skener, optički štap i dr.

Izlazna jedinica se koristi da se preko nje saopštavaju rezultati rada računara; čine je: monitor, štampač, ploter i dr.

Centralna procesorka jedinica izvršava instrukcije uskladištene u operativnoj memoriji.

Sistem softver čine operativni sistemi, koji mogu biti *jednokorisnički* (CPM, MS-DOS) i *višekorisnički* (UNIX, OPEN/VMS, Windows 2000), zatim, *jezički procesori* koji se dele na interpretere (BASIC) i kompajlere (FORTRAN, PASCAL, C++), kao i *aplikativni softveri* u koje spadaju tekst-procesori (Word for Windows) i baze podataka.

Sistem dokumentacije čine dokumentacija za sistem hardver, dokumentacija za sistem softver i ostala dokumentacija.

Imajući sve to u vidu, aktivnost *"1.3. Tehnički preduslovi"* izvodi se kroz tri podređene aktivnosti:

- Aktivnost 1.3.1. Definisanje arhitekture sistema,
- Aktivnost 1.3.2. Kadrovske potrebe,
- Aktivnost 1.3.3. Dinamika realizacije i troškovi.

Aktivnost
1.3.1. Definisanje arhitekture sistema

Aktivnost *"1.3.1. Definisanje arhitekture sistema"* treba da ukaže na osnovne pretpostavke koje se moraju ispuniti da bi se mogao sprovoditi postupak reinženjeringa poslovnih procesa.

Tehnike i tehnologije računarstva, komunikacija, pogotovu razvoj Interneta i Intraneta, osnova su za pristupanje složenom poslu reinženjeringa poslovnih procesa. Otuda reinženjering poslovnih procesa treba zasnivati na najnovijim saznanjima, tehnikama i tehnologijama, kao i principima distribuirane obrade, korišćenja baza podataka, postizanja kompatibilnosti u mrežama računara i upotrebama uređaja za prikaz informacija. Prilaz razmatranja tehničko-tehnoloških resursa treba da je u saglasnosti sa otvorenom arhitekturom referentnog modela organizacije za standarde, gde je definisano sedam nivoa povezivanja i komuniciranja računarske i druge opreme:

- NIVO 7. APPLICATION (aplikacijski nivo)
- NIVO 6. PRESENTATION (prezentacijski nivo)
- NIVO 5. SESSION (nivo sesije)
- NIVO 4. TRANSPORT (transportni nivo)
- NIVO 3. NETWORK (mrežni nivo)
- NIVO 2. DATA LINK (nivo podaci - veze)
- NIVO 1. PHISICAL (fizički nivo)

Pri izboru opreme treba imati u vidu tehničko-tehnološke preduslove, koji su sagledavani prema strukturi arhitekture referentnog modela:

- kvalitetan komunikacioni sistem,
- visok stepen kompatibilnosti računarske opreme,
- otvorenost mrežne arhitekture,
- modularnost opreme krajnjih korisnika,
- efikasnost sistema upravljanja podacima,
- korišćenje softverskih proizvoda za razvoj aplikacije.

Kvalitetan komunikacioni sistem

Za potrebe potpune distribucije i autonomnosti krajnjih korisnika u pristupu reinženjeringa poslovnih procesa, komunikacioni sistem ima ključnu ulogu kao osnovni deo integracije. Otuda je jasno da kvalitetan komunikacioni sistem može mnogo da doprinese uspešnosti funkcionisanja pod uslovom da ne zavisi od računarske opreme, da obezbeđuje neograničeno komuniciranje među svim krajnjim korisnicima i da poseduje mogućnost integracije informacija u obliku podataka, glasa, teksta, slike i crteža. Lokalne mreže (LAN) predstavljaju odgovarajući komunikacioni podsistem za lokalna geografska područja, pošto efikasno omogućuju potpunu integraciju svih krajnjih korisnika u jedinstven sistem, a mogu da se povezuju i sa drugim mrežama za prenos podataka i informacija (MAN, WAN i GAN mreže). Za veća rastojanja koriste se javne mreže za prenos govora i podataka, privatne mreže i mreže posebnih namena. Takođe, kvalitetni i pouzdani modemski uređaji i koncentratori čine deo ovog sistema. Sve to predstavlja osnovne pretpostavke i za najnovije trendove razvoja Intranet-sistema, kojim treba da se omogući povezivanje u okviru preduzeća.

Kompatibilnost računarske opreme

Jedan od osnovnih problema za sprovođenje postupka reinženjeringa poslovnih procesa je ostvarivanje kompatibilnosti između računarske opreme radi njenog povezivanja i međusobnog rada.

Kompatibilnost računarske opreme je neophodna da bi mogla da čini distribuirani sistem, pogotovu kompatibilnost centralizovane opreme sa opremom na mestima krajnjih korisnika obrade podataka i informacija.

Međutim, kompatibilnost koja mora da postoji za potrebe fizičkog i logičkog povezivanja različite opreme nije dovoljna, već je neophodna i kompatibilnost i za ostale nivoe komuniciranja dva korisnika prema OSI (Open System Interconnection) referentnom modelu, gde je posebno bitna kompatibilnost aplikativnog nivoa i nivoa predstavljanja podataka i informacija.

Prilikom izbora računarske opreme treba težiti da između opreme postoji što veći stepen kompatibilnosti rada na svim nivoima posmatranja prema OSI referentnom modelu. U vezi s tim bitno je naglasiti da se i prilikom izbora softvera, tj. sistema za upravljanje bazama podataka mora voditi računa da postoji softverski proizvod koji se može instalirati na svim računarskim platformama. To je strateška odluka rukovodstva preduzeća i od te odluke može zavisiti tehnološki napredak preduzeća.

Otvorenost sistema distribuirane obrade

Distribuirani sistemi se baziraju na potrebi disperzije informacija i predstavljaju koordinirani skup mogućnosti obrade informacija u dva ili više nezavisna resursa. Osnovni zahtevi distribuiranim sistemima sagledavaju se u sklopu doprinosa razvoju informacionih sistema, i to prvenstveno radi:

- pružanja novih kvaliteta krajnjem korisniku,
- mogućnosti korišćenja zajedničkih resursa,
- rasterećenja centralizovanih resursa,
- autonomnosti delova sistema,
- veće fleksibilnosti integracije sistema.

Savremeni sistemi distribuirane obrade zahtevaju otvorenost mrežne arhitekture ovih sistema, sa tendencijom da se formiraju potpuno distribuirane strukture i da se obezbedi što veća integracija svih računarskih i drugih uređaja.

Modularnost računarske opreme

Troškovi realizacije distribuirane obrade su veoma bitan i, to najčešće, ograničavajući faktor izgradnje složenih distribuiranih sistema, koji treba sagledavati kao zbir troškova računarskih i drugih uređaja, troškova komunikacionog povezivanja i prenosa podataka, troškova sistemskih i aplikativnih mogućnosti međusobnog rada svih resursa, troškova vezanih za kadar potreban za razvoj, uvođenje i održavanje sistema, kao i troškova krajnjih korisnika za eksploataciju ovakvih sistema.

Računarska oprema u okviru klijent/server arhitekture, a za potrebe distribuiranog sistema, može se posmatrati u dva osnovna oblika, i to kao

- računarska oprema zajedničkih resursa (strana servera) i
- računarska oprema krajnjeg korisnika (strana klijenta).

Za realizaciju postupka reinženjeringa poslovnih procesa veoma je bitno da se ostvari modularnost računarske opreme, posebno opreme na mestima krajnjih korisnika. Takođe, bitno je da se optimalno postave zajednički resursi, koji mogu biti smešteni na jednoj lokaciji ili, pak, distribuirani na više lokacija sa definisanim odnosima.

Efikasnost sistema upravljanja podacima

Podaci predstavljaju osnovni resurs, pa je otuda upravljanje podacima u ovakvim sistemima od izuzetne važnosti, pogotovu kada je reč o korišćenju i upravljanju distribuiranim bazama podataka, gde sistemska podrška treba da osigura kontrolisanu distribuciju i redundansu podataka, brzo i precizno lociranje i manipulisanje podacima. Osnovni zahtevi upravljanja podacima su iskazani kroz potrebu unificiranog organizovanja i integracije podataka, kao i obezbeđivanja zaštite i pouzdanosti podataka. U okviru tih zahteva ogleda se efikasnost sistema za upravljanje podacima.

Prilikom realizacije distribuiranih sistema treba težiti da se koristi jedinstvenost sistema upravljanja podacima, i to pre svega zbog transparentnosti podataka između računara, što je posebno bitno za složene distribuirane sisteme, kod kojih se koriste distribuirane i integrisane baze podataka, a procesiranje vrši sa više računara.

Korišćenje sistemske podrške za razvoj aplikacija

Uspešnost distribuiranih sistema se posebno ogleda u nezavisnosti korisnika da bez mnogo napora i ulaganja razvijaju aplikacije za svoje potrebe. Stepen nezavisnosti, tj. uspešnosti distribuiranih sistema je utoliko veći ukoliko tehnička sredstva poseduju veću sistemsku podršku za razvoj aplikacija, prilagođenu što većem broju ljudi. Argumenti da se to što bolje ostvari, mogu se sagledati u korišćenju sistemske podrške, kao što su:

- jezici za upite bazama podataka,
- generatori izveštaja,
- neproceduralni jezici višeg nivoa,
- generatori aplikacija,
- parametrizovani namenski softverski paketi.

Korišćenje sistemske podrške za razvoj aplikacija je sve nužnije, jer se time obezbeđuju kraće vreme kreiranja aplikacije, povećanje uloge krajnjih korisnika i lakše održavanje aplikacija. Takođe, ostvaruje se i veća integracija krajnjih korisnika i zajedničkih računarskih resursa na bazi jednostavnosti softverskih sintaksi i fleksibilnosti u specifikaciji upita i zahteva. Pri tome treba imati u vidu da, sa stanovišta krajnjeg korisnika, to bude jedinstveno organizovano i sveobuhvatno namenjeno zahtevima korisnika.

Aktivnost 1.3.2. Kadrovske potrebe

Pod kadrovskim potrebama podrazumevaju se broj potrebnog kadra za realizaciju projekta reinženjeringa poslovnih procesa i potrebna obuka za korišćenje informacionih tehnologija.

Uz obezbeđivanje neophodne računarske opreme, komunikacione i ostale prateće opreme, od posebnog značaja su i kadrovski resursi, odnosno kvalitetna i u dovoljnoj meri zastupljena kadrovska podrška.

Saglasno tom prilazu, kao informatičku osnovu u sprovođenju reinženjeringa poslovnih procesa, treba imati minimum potrebnog kadra. Potrebni su:

- rukovodilac,
- vodeći projektant za modeliranje procesa i podataka,
- vodeći projektant softverskih rešenja,

- vodeći projektant baze podataka,
- sistem inženjer,
- referent dokumentacije.

Posebno značajnu ulogu treba da imaju kontinuirani proces obrazovanja kadra i automatizacija njihovog rada, kao i adekvatni oblici funkcionalnog organizovanja. To je podjednako značajno i za fazu razvoja i za fazu korišćenja. Upravo zato, shodno usvojenoj metodologiji, treba dati i pregled sledećih kurseva:

- kompjutersko opismenjavanje (WINDOWS, MS Word),
- integracija IS i zahteva sistema kvaliteta (modeliranje procesa - BPwin),
- modeliranje podataka - ERwin,
- generisanje prototipske aplikacije u MS ACCESS-u,
- rad sa tabelama - MS EXCEL,
- mrežni rad i INTERNET I NJEGOVI SERVISI.

Kompjutersko opismenjavanje (WINDOWS, MS Word)

Sadržaj kursa:

Računari i računarski sistemi. Sistem hardver. Sistem softver. Sistem dokumentacije. PC računari, konfiguracija, komponente, štampači.

WINDOWS operativni sistem: Osnovno o WINDOWS okruženju i načinu korišćenja prozora. Organizacija podataka. Pojam direktorijuma, stabla, navigacija. Rad sa disketom (formati, kapaciteti, formatizovanje). Tipovi podataka.

Microsoft Office - funkcije i elementi.

WORD for WINDOWS: Pozivanje programa WORD for WINDOWS. Dokument, font, format stranice, tabulatori, automatski back-up, korišćenje stilova, podešavanje osnovnih parametara stranice. Navigacija, unos teksta, brisanje, obeležavanje teksta. Operacije sa blokovima teksta. Ubacivanje prekida stranica, straničenje, heder-futer, fusnota. Ubacivanje datoteka, okvira i slika. Editor formula. Rad sa makroima. Štampanje dokumenta. Unakrsne tabele. Baze podataka.

Napomena: Opismenjeni korisnici su korisniji sagovornici, jer imaju više zahteva i prošireni su im vidici.

Integracija IS i zahteva sistema kvaliteta (Modeliranje procesa-BPwin)

Sadržaj kursa:

Zašto vršiti modeliranje procesa? Reinženjering kao element integracije ISO 9000 i modeliranja procesa. Zašto se proces modelira? Obezbeđivanje kvaliteta softvera u skladu sa standardima ISO 9000. Šta će proces da prikaže? Funkcionalno modeliranje. Sintaksa i semantika grafičkog jezika IDEF0. Granice modela. Kontekstni dijagram. Dekompozicioni dijagram. Stablo aktivnosti. Integracija procesa i podataka preko rečnika podataka. Dataflow (DFD) modeliranje. *Obuka u korišćenju CASE alata BPwin.*

Napomena: Obučeni korisnici celokupnu komunikaciju i opisivanje postupaka ostvaruju na grafičkom nivou i pokazuju bolje razumevanje.

Modeliranje podataka-ERwin

Sadržaj kursa:

Šta je to IDEF1X standard? Identifikacija kandidata za entitete. Identifikacija veze. Definisanje entiteta i relacija. Nezavisni entitet. Zavisni entitet. Asocijativni entitet. Identifikujuće i neidentifikujuće veze. Izrada ER modela. Atributi ER modela. Lista kandidata za atribute. Definisanje ključeva u modelu. Atributi i normalizacija. Definisanje poslovnih pravila. Ograničenja. Prikaz i verifikacija kardinalnosti veza. Definisanje referencijalnog integriteta. Identifikacija poslovnog domena. Identifikacija operacija.

Napomena: Obučeni korisnici su korisni sagovornici za modeliranje podataka, jer na ovom nivou, pre nego što se pređe na programiranje, treba razrešiti što više dilema.

Generisanje prototipske aplikacije u MS ACCESS-u

Sadržaj kursa:

Prevođenje strukture ERwin-a. Generisanje tabela. Generisanje formata i validacionih pravila. Korišćenje Wizard-a. Dizajniranje i korščenje formi. Upravljanje podacima. Dodeljivanje i oduzimanje privilegija drugim korisnicima. Upoređivanje Access-a 2.0 sa Access-om 97 i Access-om 2000 i veza sa SQL serverom. Prikaz realizovanog softvera za definisan model podataka.

Napomena: MS ACCESS je dobra platforma za prototipsko programiranje koje, po izboru odgovarajućeg SUBP, može da ostane klijent-strana u okviru klijent/server-arhitekture.

Rad sa tabelama-MS EXCEL

Sadržaj kursa:

MS EXCEL, vrste podataka. Unos numeričkih podataka, natpisa, datuma i vremena, logičkih izraza, formula. Unos adrese ćelije. Apsolutna i relativna adresa ćelije. Formatizovanje i poravnavanje podataka. Izbor fonta. Dodavanje okvira tabeli. Prikaz karakteristika tabele.

Prikaz formula i funkcija. Dodeljivanje funkcija. Promena formula. Matematičke i statističke funkcije. Funkcije baze podataka. Znakovne i datumske funkcije. Finansijske funkcije. Funkcije posebne namene. Kreiranje baze podataka.

Dodavanje zapisa i atributa. Brisanje zapisa i atributa. Pretraživanje zapisa i sortiranje.

Izrada izveštaja. Kreiranje i rad sa grafikonima. Promena vrste grafikona. Makroi.

Napomena: Ovaj kurs je prelazno rešenje da bi se omogućilo da se već kupljeni računari što pre napune podacima tabelarnog tipa.

Mrežni rad, Internet i njegovi servisi

Sadržaj kursa:

Korišćenje klijent/server-aplikacija. Razvojno okruženje klijent/server-aplikacija. Korisnički interfejsi u klijent-aplikacijama. Mreža u klijent/server-sistemu. Topologija mreže. Procesi u sistemu klijent/server. Šta je Internet? Organizacija Interneta. Povezivanje na Internet. Načini pristupa Internetu. Nivoi povezanosti. Šta nudi Internet? E-mail - elektronska pošta. WWW - multimedijalni pristup i dr. Pravljenje sopstvene world wide web (www) prezentacije. Struktura www prezentacije (HTML jezik). Pregled napravljene prezentacije sa www servera putem www browsera.

Napomena: Kurs je namenjen svim zaposlenima koji koriste Internet i njegove servise.

Predloženi kursevi treba da obezbede metodološki usaglašen način rada na izradi detaljnog projekta.

Pored ovih opštih kurseva, pri izradi detaljnog projekta treba obezbediti stručnu i konsultativnu pomoć, naročito pri rešavanju problema modeliranja podataka, odnosno organizacije baze podataka.

Aktivnost
1.3.3. Dinamika realizacije i troškovi

Kada je reč o dinamici realizacije i troškovima, neophodno je korišćenje nekog od softvera za upravljanje projektima (npr., MS Project).

Troškovi realizacije najčešće se posmatraju u okviru grupa poslova kao:

- troškovi razvoja aplikacija,
- troškovi tehničko-tehnoloških resursa,
- troškovi eksploatacije.

Svaka od ovih grupa poslova se, takođe, sastoji iz posebnih troškova i otuda specifikaciju troškova treba da čini sledeća struktura:

- troškovi razvoja:
  - obuka projektnog tima,
  - razvoj zajedničkih aplikacija,
  - stručna pomoć pri razvoju,
  - razvoj i dopuna sopstvenih aplikacija,
  - softverski proizvodi za razvoj aplikacija;
- troškovi tehničko-tehnoloških resursa:
  - računarska oprema zajedničkih resursa,
  - oprema komunikacionog sistema,
  - dopuna i kompletiranje računarske opreme,
  - prateća oprema i adaptacija prostora;
- troškovi eksploatacije:
  - održavanje opreme,
  - potrošnja električne energije,
  - korišćenje komunikacionih linija,
  - amortizacija opreme,
  - plate radnika.

Kao rezultat aktivnosti *"1.Funkcionalno modeliranje"* trebalo bi da proizađe dokument pod nazivom "Studija elemenata potrebnih za reinženjering poslovnih procesa" na osnovu koje se sprovodi sledeća aktivnost: *"2. Informaciono modeliranje".*

Aktivnost 2. Informaciono modeliranje

Aktivnost 2.1. Definisanje detaljnih zahteva

Aktivnost 2.2. Kreiranje ER dijagrama

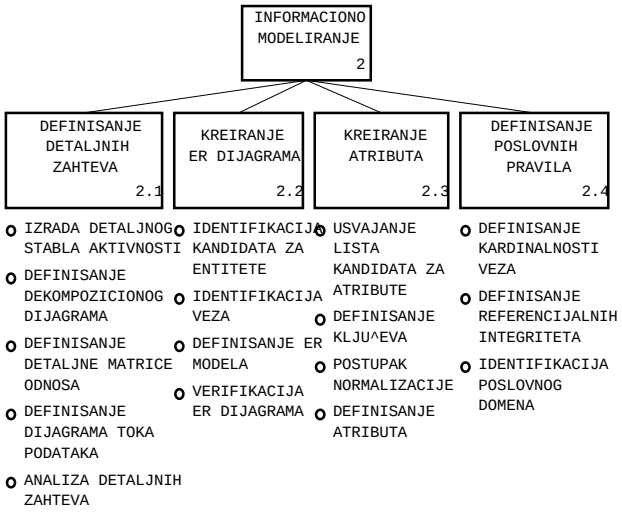
Aktivnost 2.3. Kreiranje atributa

Aktivnost 2.4. Definisanje poslovnih pravila

Aktivnost *"2. Informaciono modeliranje"* se izvodi na osnovu definisanih kritičnih funkcija za prethodno urađenu aktivnost *"1. Funkcionalno modeliranje"*. Kritične funkcije predstavljaju, svaka za sebe, po jedan glavni projekat koji se razvija u okviru aktivnosti: *"2. Informaciono modeliranje"*. Informacioni model prikazuje u kakvom su međusobnom odnosu podaci u nekom realnom sistemu.

Informaciono modeliranje omogućuje:

- definisanje sistemske dokumentacije koja se može koristiti za bazu podataka i za razvoj aplikacija da bi osoblje moglo da definiše sistemske zahteve;
- bolju komunikaciju međusobno i sa krajnjim korisnicima;
- jasnu sliku o poslovnim pravilima i
- 'logičku' sliku baze podataka koju mogu koristiti automatski alati za generisanje sistema za upravljanje bazama podataka (SUBP).



Slika 3.1 Stablo aktivnosti "2.Informaciono modeliranje"

Na slici 3.1. prikazano je stablo aktivnosti za *"2.Informaciono modeliranje"*, koje se sastoji od sledećih podređenih aktivnosti:

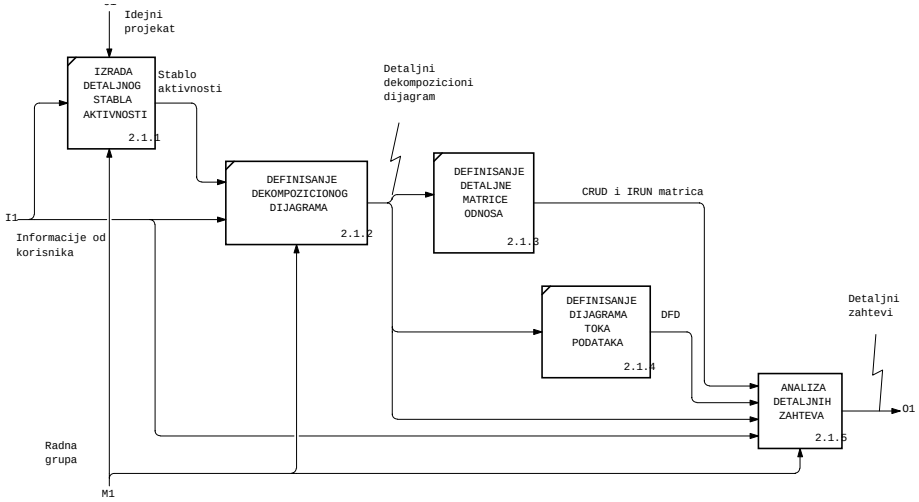
- Aktivnost 2.1. Definisanje detaljnih zahteva,
- Aktivnost 2.2. Kreiranje ER modela,
- Aktivnost 2.3. Kreiranje atributa,
- Aktivnost 2.4. Definisanje poslovnih pravila.

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 3.1.

Aktivnost
2.1. Definisanje detaljnih zahteva

Aktivnost *"2.1. Definisanje detaljnih zahteva"* se izvodi na osnovu prethodno urađene aktivnosti *"1. Funkcionalno modeliranje"*, tj. definisane studije kojom se određuju okviri koji se u ovoj aktivnosti za izabrane podsisteme detaljno razrađuju. Definisanje detaljnih zahteva treba da predstavlja reviziju postavljenih elemenata iz studije, uz dalje detaljno produbljivanje za izabranu funkciju za koju se sprovodi reinženjering poslovnih procesa. Definisanje detaljnih zahteva se izvodi posredstvom sledećih akivnosti (slika 3.1):

- Aktivnost 2.1.1. Izrada detaljnog stabla aktivnosti,
- Aktivnost 2.1.2. Definisanje dekompozicionog dijagrama,
- Aktivnost 2.1.3. Definisanje detaljnih matrica odnosa,
- Aktivnost 2.1.4. Definisanje dijagrama toka podataka,
- Aktivnost 2.1.5. Analiza detaljnih zahteva.



Slika 3.2. Dekompozicioni dijagram za aktivnost "2.1. Definisanje detaljnih zahteva"

Na slici 3.2. prikazan je dekompozicioni dijagram za aktivnost *"2.1. Definisanje detaljnih zahteva"*, gde je definisan tok informacija, počev od ulaznog dokumenta "Idejni projekat", preko nabrojanih aktivnosti i prikazanih na slici 3.2, do izlaza definisanog dokumentom "Detaljni zahtevi".

Dekompozicionim dijagramom za aktivnost *"2.1. Definisanje detaljnih zahteva"* definišu se tokovi informacija između podređenih aktivnosti. O dekompozicionom dijagramu u daljem tekstu biće više reči.

Aktivnost
2.1.1. Izrada detaljnog stabla aktivnosti

Ulaz u aktivnost *"2.1.1. Izrada detaljnog stabla aktivnosti"* je idejni projekat na osnovu dobijene informacije od korisnika. Sama aktivnost se izvodi za izabranu kritičnu funkciju, gde se vrši dekompozicija do nivoa primitivnih procesa za koje se realizuju odgovarajući scenariji događanja. Izlaz iz ove aktivnosti je detaljno stablo aktivnosti za kritičnu funkciju.

Način izrade stabla aktivnosti opisan je u prethodnom poglavlju. Ovde treba istaći da primenom BPwin-a CASE alata ovaj pristup ima novu dimenziju, jer omogućuje da se nastavi projekat na onim mestima gde se završava navedena studija. Ovakav pristup omogućuje stalnu ažurnost projekta i praćenje vremenske i novčane dinamike, vezane za sprovođenje reinženjeringa izabrane poslovne funkcije preduzeća.

Na osnovu definisanog detaljnog stabla aktivnosti u sledećem koraku prelazi se na definisanje dekompozicionog dijagrama.

Aktivnost
2.1.2. Definisanje dekompozicionog dijagrama

Ulaz u aktivnost *"2.1.2. Definisanje dekompozicionog dijagrama"* su dokumenta: "Detaljno stablo aktivnosti" i "Informacije od korisnika". U okviru ove aktivnosti uspostavljaju se horizontalne veze između podaktivnosti koje su povezane strelicama. Izlaz iz ove aktivnosti je "Detaljni dekompozicioni dijagram" .

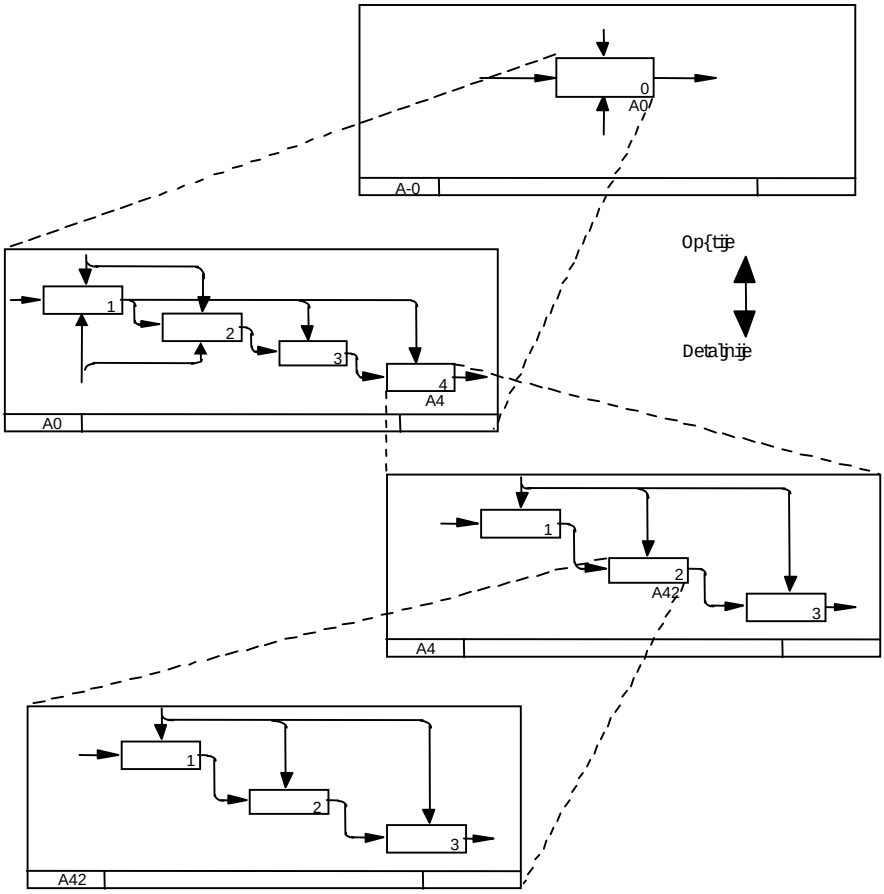
U okviru definisanja dekompozicionog dijagrama u daljem tekstu biće razmatrani:

- pravougaonici u dekompozicionom dijagramu,
- strelice u dekompozicionom dijagramu,
- grananje ili udruživanje strelica,
- ugao posmatranja,
- postojeći (as-is) i budući (to-be) model,
- formiranje troškovnih centara,
- tekstualni opis.

Pravougaonici u dekompozicionom dijagramu

Aktivnosti su, kao što je već rečeno, smeštene u pravougaonicima koji se crtaju u dijagonalnom smeru, od gornjeg levog ugla strane ka donjem desnom uglu. Svakoj aktivnosti mora se dodeliti naziv u obliku glagolske fraze, te mora imati najmanje jednu kontrolnu i jednu izlaznu strelicu.

Na slici 3.3. prikazana je struktura formiranja dekompozicionog dijagrama. Polazi se od kontekstnog dijagrama (opisan u prethodnom poglavlju) koji se definiše na najvišem nivou, pa se izvodi dekomponovanje u podređene (child) dijagrame. Svaka od podaktivnosti podređenog dijagrama može kreirati svoj dijagram na nižem nivou. Na taj način se definišu različiti nivoi apstrakcije, tj. na višim nivoima su opštije aktivnosti i grupisane strelice, koje se na nižim nivoima dekomponuju i detaljnije opisuju.



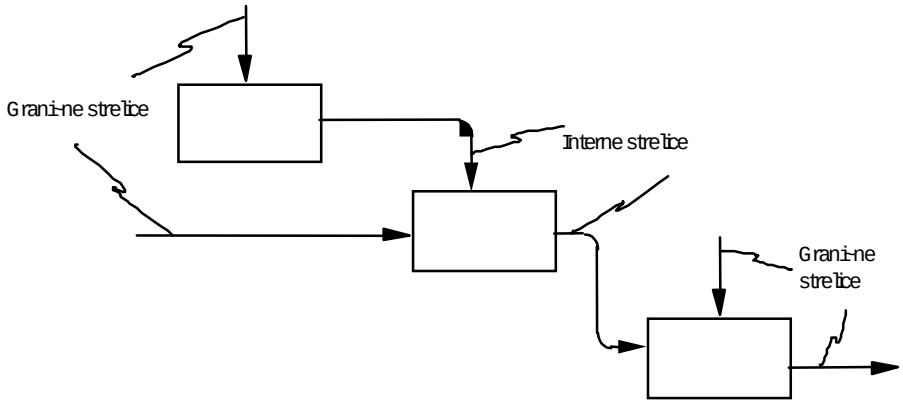
Slika 3.3. Dekompoziciona struktura IDEF0 metodologije

Strelice u dekompozicionom dijagramu

Strelice u okviru dekompozicionog dijagrama omogućuju tzv. horizontalno povezivanje definisanih aktivnosti.

Kao što se može videti, na slici 3.3. strelice definisane na kontekstnom dijagramu se prenose u podređeni dekompozicioni dijagram. Dakle, strelice definisane u aktivnosti koja prethodi ('roditelj') pojavljuju se u podređenom dekompozicionom dijagramu kao *granične strelice (boundary arrows),* tj. kao strelice koje nastaju van okvira posmatranog dijagrama (slika 3.4).

U okviru dekompozicionog dijagrama definišu se tzv. *eksplicitne ili interne strelice* koje povezuju aktivnosti (slika 3.4). Dekompozicioni dijagram bez unutrašnjih strelica ukazuje na organizacioni pristup dekompoziciji, a ne funkcionalni.



Slika 3.4. Granične i interne strelice (Boundary and Internal Arrows)

Ulazne granične strelice koje dolaze iz nadređenog dijagrama u podređeni dijagram mogu se deliti u više specifičnih strelica i obrnuto, izlazne granične strelice iz podređenog dekompozicionog dijagrama se grupišu i izlaze u nadređeni dijagram.

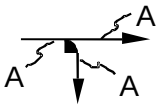
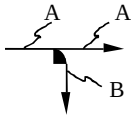
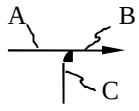
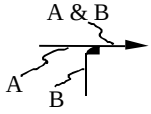
Strelice u okviru dekompozicionog dijagrama mogu se definisati kao:

- strelice koje se granaju ili udružuju,
- povratne strelice i
- skrivene strelice.

Grananje ili udruživanje strelica

Strelice predstavljaju svojevrsan cevovod koji se može granati ili udruživati, kao što je prikazano na slici 3.5.

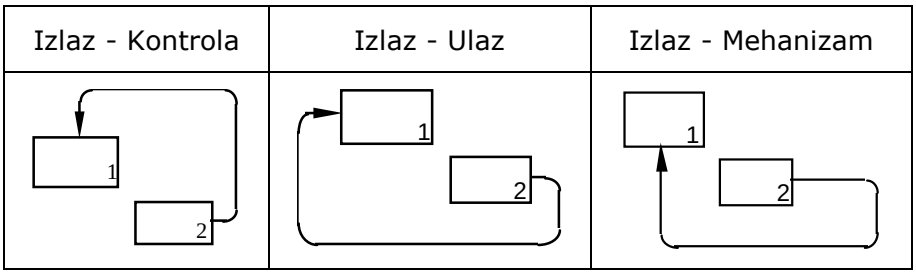
U prvom slučaju, prikazanom na slici 3.5, dolazi do grananja ulaza A u iste skupove informacija, dok se u drugom slučaju odvaja novi podskup B. U trećem slučaju se A grana u B i C, a u četvrtom se A i B integrišu u A&B. Preporučuje se da se definiše naziv svake strelice da ne bi došlo do dvosmislenosti.

Puni skupovi informacija u svim granama.	
Podela na pun skup u prvoj i podskup informacija u drugoj grani	
Podela na različite podskupove informacija	
Integracija strelica	

Slika 3.5. Struktura račvanja i udruživanja strelica

Povratne strelice

Povratna (feedback) petlja je pojava kada izlaz iz aktivnosti predstavlja ulaz, kontrolu ili mehanizam neke ranije aktivnosti u dekompozicionom dijagramu. Ovakvi izlazi su, obično, neke negativne karakteristike.



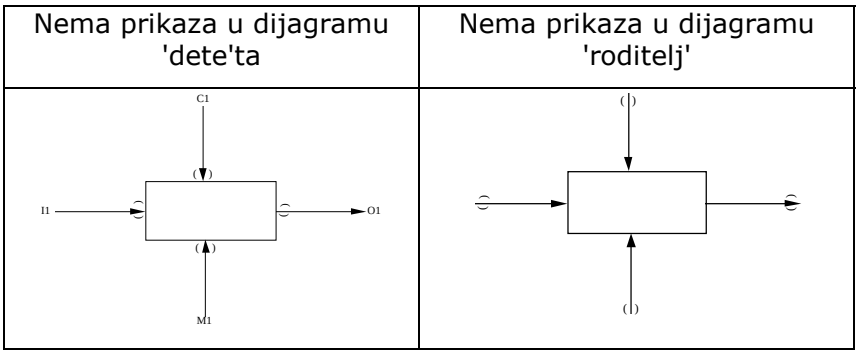
Slika 3.6. Vrste povratnih izlaza

U prvom i drugom slučaju prikazanom na slici 3.6. povratna petlja i ulazna povratna petlja kao kontrola predstavljaju iteraciju ili rekurziju, dok u trećem slučaju povratna petlja Izlaz - Mehanizam ima ulogu podrške na nivou resursa za izvođenje određene aktivnosti.

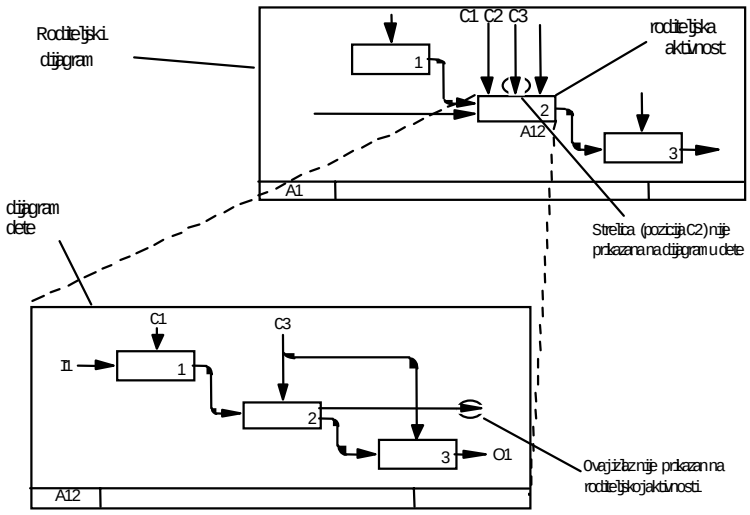
Skrivene strelice

Poseban tip strelica su tzv. *skrivene (tunneled) strelice* koje nastaju ako se želi da strelice ne budu viđene na nadređenom ili podređenom dekompozicionom dijagramu. To je situacija kada je dijagram prenatrpan, pa se zbog jasnoće izbegava njegovo prikazivanje. Druga situacija je ako se želi prikaz budućih događaja.

Može biti i slučaj da strelica predstavlja kontrolu koja se primenjuje na svakoj aktivnosti u dekompoziciji, a da je bila propuštena radi jasnoće u nižim nivoima. Oznaka da je strelica sakrivena su zagrade (slika 3.7).



Slika 3.7. Vrste skrivenih strelica



Slika 3.8. Primer skrivenih strelica (Example of Tunneled Arrows)

Da bi se ovo pojasnilo poslužiće primer skrivenih strelica na slici 3.8. Ako su skrivene strelice nacrtane u 'roditelj'skom dijagramu (pozicija C2) onda se ne pojavljuju u dijagramu 'dete'ta. Ukoliko su, pak, skrivene strelice nacrtane u dijagramu 'dete'ta, onda se ne vide u dijagramu 'roditelj'.

Ugao posmatranja

Prilikom IDEF0 modeliranja mogu se razmatrati različita gledišta za rešavanje postavljenog problema. Pri tom se mora usvojiti jedno kompromisno rešenje koje će se dalje koristiti. Međutim, različita gledišta se mogu predstaviti korišćenjem takozvanih FEO (For Exposition Only) dijagrama samo za prikazivanje. FEO dijagrami se, obično, koriste za prezentaciju kada se žele ukloniti neki detalji zbog lakšeg razumevanja problema. Za razliku od IDEF0 grafičkog dijagrama, FEO dijagram ne mora da poštuje IDEF0 pravila. Ovaj prilaz se najčešće koristi za prezentacije.

Postojeći (as-is) i budući (to-be) model

Prilikom definisanja postupka za reinženjering poslovnih procesa definišu se dva pristupa: jedan koji opisuje postojeće stanje (analiza dokumenata, tj. pogled odozdo) i drugi kojim se definiše budući funkcionalni IDEF0 model, koji se razvija paralelno sa IDEF1X modeliranjem podataka (intervju, tj. pogled odozgo). Ako je velika razlika između postojećeg i budućeg modela, ponekad treba napraviti treći model, kojim se definišu dokumenta za potrebe prelaza sa postojećeg na budući model. Prilikom definisanja postojećeg stanja potrebno je izbegavati zamku tipa "TREBALO BI DA", jer se ne želi opis idealne situacije već onakve kakva stvarno jeste.

Formiranje troškovnih centara

Metoda IDEF0 omogućava formiranje troškovnih centara i vezu aktivnosti i troškovnog centra. Za svaku aktivnost se definišu frekvenca, dužina trajanja i cena te aktivnosti na najnižem nivou. Troškovi aktivnosti na višem nivou sadrže troškove aktivnosti nižeg nivoa. Modelar poslovnih procesa BPwin podržava ABC (Activity Based Costing) sistem i ima interfejs ka ABC alatima, namenjenim onima koji menadžment strategiju baziraju na aktivnostima i žele da prate troškove vezane za sprovođenje reinženjeringa poslovnih procesa (videti Aktivnost *"1.3.2. Dinamika realizacije i troškovi").*

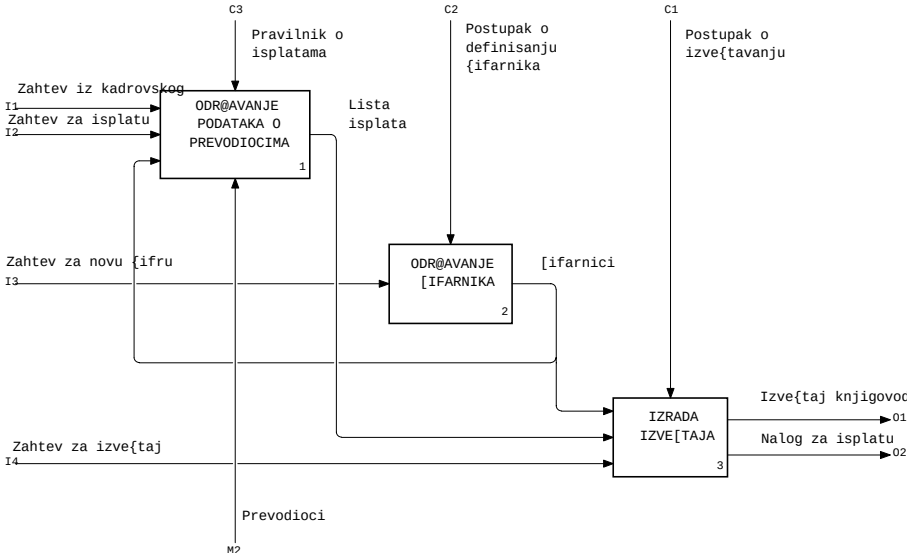
Tekstualni opis

Pored prikazanog grafičkog opisivanja, potrebno je, kao dopunu, i tekstom opisati aktivnosti. Tekstom se opisuju do detalja namena i funkcionisanje svake aktivnosti u modelu. Tekstualni opis se često koristi da bi se obezbedio kompletan rezime poslovnog procesa, tj. opisuju se relevantni detalji do tančina. To je bitno ako se sprovode elementi vezani za internu standardizaciju i definisani postupci i uputstva vezani za seriju standarda ISO 9000, jer postupci i uputstva postaju sastavni deo grafičkog opisa.

Imajući sve to u vidu, definisan je dekompozicioni dijagram za Aktivnost *"Praćenje isplata".*

Dekompozicioni dijagram za aktivnost "Praćenje isplata"

Na osnovu definisanog kontekstnog dijagrama (prikazanog na slici 2.6) i stabla aktivnosti (sa slike 2.7) definiše se dekompozicioni dijagram prikazan na slici 3.9. Sa prethodno definisanog kontekstnog dijagrama (prikazanog na slici 2.6) automatski su prenesene granične strelice, a na osnovu stabla aktivnosti (slika 2.7) pojavljuju se nepovezane tri aktivnosti. Potrebno je granične strelice povezati sa odgovarajućim aktivnostima i definisati interne strelice koje će povezati aktivnosti između sebe.



Slika 3.9. Dekompozicioni dijagram za aktivnost "Praćenje isplata"

Za aktivnost *"1.Održavanje podataka o prevodiocima"* treba, na osnovu ulaznih dokumenata: "Zahtev iz kadrovskog" i "Zahteva za isplatu", definisati interni izlazni dokument " Lista isplata", koji sadrži spisak osoba za isplatu preko žiro-računa.

Aktivnost *"2. Održavanje šifarnika"* je pretpostavka za uspešnu efikasnost buduće korisničke aplikacije. Za ovu aktivnost ulaz je "Zahtev za novu šifru", a izlaz su "Šifarnici", koji su ulaz za sledeću aktivnost *"3. Izrada izveštaja"* i povratna ulazna informacija za prethodnu aktivnost *"1. Održavanje podataka o prevodiocima"*. Za primer o kome je dosada bilo govora u okviru ove aktivnostu, održavaju se šifarnici ODELJENJA, JEZIKA i RADNIH MESTA.

Stoga se u test-primeru aktivnost *"3. Izrada izveštaja"*, na osnovu ulaza "Zahtev za izveštaj" i "Lista isplata", prave dva izlazna dokumenta:

- "Izveštaj knjigovodstvu", gde se knjigovodstveno vodi evidencija o obavljenom poslu;
- "Nalog za izplatu", kojim se šalje novac za isplatu na žiro-račun u banci.

Na osnovu definisanog dekompozicionog dijagrama u sledećem koraku je definisanje detaljnih matrica odnosa.

Aktivnost
2.1.3. Definisanje detaljnih matrica odnosa

Tokovi podataka prikazani strelicama povezuju se sa entitetima i sa svim ili samo sa pojedinim atributima tih entiteta. Ako se posmatraju načini na koje strelica koristi entitet, onda se definišu CRUD matrice (slika 3.10), što je skraćenica od:

- kreirati entitete, što se označava sa C (Create);
- pretraživati entitete, što se označava sa R (Retrive);
- ažurirati entitete, što se označava sa U (Update);
- brisati entitete, što se označava sa D (Delete).

Kada se za svaku strelicu definiše, osim načina korišćenja entiteta, i povezanost entiteta sa atributima, onda se definiše IRUN (slika 3.10) matrica, što je skraćenica od:

- ubacivanje atributa, što se označava sa I (Insert);
- pretraživanje atributa, što se označava sa R (Retrive);
- ažuriranje atributa, što se označava sa U (Update);
- dodela nula vrednosti, što se označava sa N (Null field).

Dakle, mogu se kreirati dve matrice:

- CRUD matrica, na kojoj bi sa jedne strane bili entiteti, a sa druge strane aktivnosti koje koriste te entitete i
- IRUN matrica, sa definicijom korišćenja svakog atributa u određenom entitetu i određenoj aktivnosti.

Veza IDEF0 funkcionalnog modela i IDEF1X informacionog modela je omogućena preko rečnika podataka (data dictionary manager). Na taj način (indirektnom sinhronizacijom) dozvoljeno je sinhronizovanje jednog modela podataka sa više modela procesa, što se u praksi ne samo često dešava već i preporučuje. ERwin CASE alat omogućuje importovanje rečnika podataka iz ERwin-ovog modela podataka u BPwin model procesa i obrnuto.

Kao što se može zaključiti, prebacivanje rečnika podataka je omogućeno u oba smera, tako da je, sa stanovišta projektanta, nebitno u kom trenutku je uočio potrebe za promenama na datom modelu. U slučaju da se pojavi potreba da se definiše novi entitet ili atribut, ili da se izmeni postojeći, to se može uraditi u BPwin-u, a zatim nadograditi rečnik podataka kao veza sa ERwin-om. Moguće je, takođe, u BPwin-u definisati entitete ili atribute koji za model podataka ne treba da budu vidljivi. Entiteti i atributi iz rečnika podataka se pridružuju objektima koji opisuju kretanje podataka kroz procese, kao i način kreiranja, održavanja i upotrebe podataka.

Kao jedan od važnih momenata je postupak "inverznog inženjerstva" (reverse engineering) kada se od gotovog softverskog proizvoda, posredstvom ERwin-a, formira njegova specifikacija (model podataka), pa se importuje u dekompozicioni dijagram BPwin, gde se opisuju entiteti i atributi za kasniji, tzv. direktni inženjering, tj. modeliranje podataka korišćenjem ERwin-a (videti *"2. Aktivnost Informaciono modeliranje"*).

"Inverzni inženjering" je potreban, jer postoji, npr., skup COBOL-skih programa u kojima su definisane gomile datoteka koje se još uvek koriste, ali za koje, osim izvornog koda, ne postoji nikakva dokumentacija. Da bi se ove informacije iskoristile postupkom "inverznog inženjerstva", treba praviti standardnu specifikaciju dela sistema, pa na osnovu nje nastavljati razvoj, tj. definisati rečnik podataka u BPwin-u. U 4. delu za aktivnost *"3. Aplikativno modeliranje"* o ovom postupku biće više reči. Treba napomenuti da se na ovaj način ne može dobiti potpuna specifikacija, jer implementacija nekog softvera je semantički siromašnija od njegove specifikacije, pa se ne može rekonstruisati potpuna semantika sistema, mada se na taj način ipak pokušava sa korišćenjem onoga što je već projektovano kao softver i što već radi.

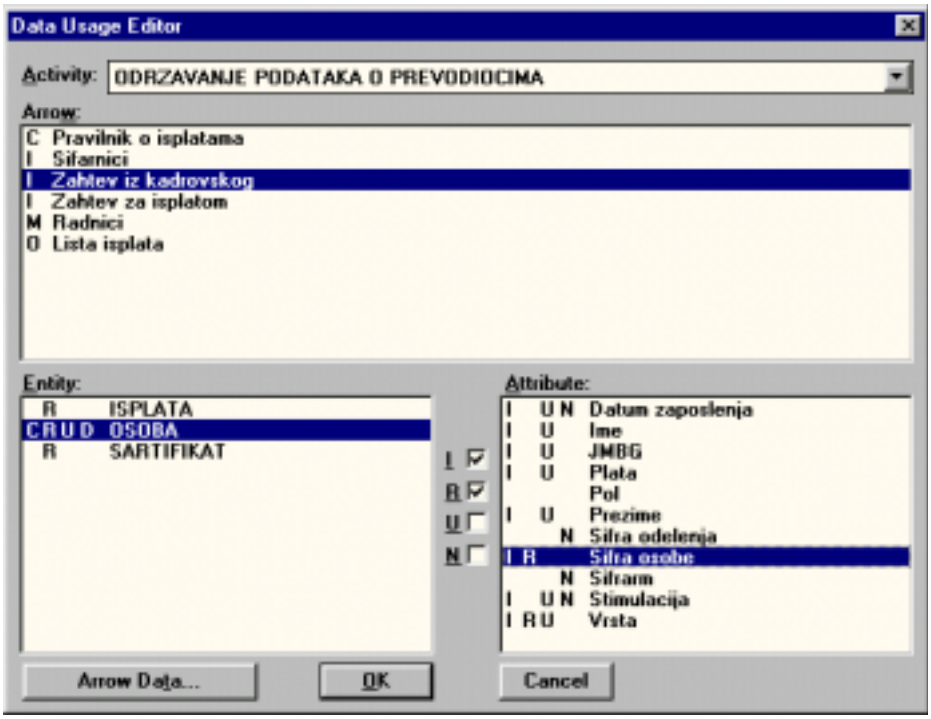
Imajući sve to u vidu, za primer dokumenta "Karton isplata" biće definisane detaljne CRUD i IRUN matrice.

Matrice odnosa za dokument "Karton isplata"

Na slici 3.10. prikazan je iz BPwin-a Data Usage Editor, gde se za aktivnost *"1. Održavanje podataka"* o prevodiocima i za ulazni dokument (I): "Zahtev iz kadrovskog", definišu entiteti: ISPLATA, OSOBA I CERTIFIKAT. Za entitet OSOBA, koji u ovoj aktivnosti ima sve četiri operacije (CRUD), definišu se entiteti na kojima se izvode operacije (kao što je prikazano na slici 3.11).

Korišćenjem načina opisa prikazanog na slici 3.10. i na slici 3.11. prikazan je izveštaj u obliku kombinovane CRUD i IRUN matrice koja treba da precizira do detalja pravila koja važe za entitete i atribute i da omogući preciznu izradu entitetnog dijagrama.

Kao i u prethodnim fazama, potrebno je verifikovati izvedene aktivnosti, pa se kroz aktivnost *"2.1.5. Analiza detaljnih zahteva"* opisuje način verifikacije detaljnih zahteva.



Slika 3.10. Prikaz CRUD i IRUN matrice

Naziv aktivnosti	Naziv entiteta	CRUD	Naziv atributa	IRUN
ODRZAVANJE PODATAKA O PREVODIOCIMA	OSOBA	CRUD	DATUMZ	I UN
			IME	I U
			PLATA	I U
			PREZIME	I U
			SIFRA ODELJENJA	N
			SIFRA RM	N
			SIFRA OSOBE	IR
			STIMULACIJA	I UN
			JMBG	I U
			VRSTA	I R
	CERTIFIKAT	CRUD	SIFRA JEZIKA	R
			SIFRA OSOBE	R
			STEPEN ZNANJA	R U
	ISPLATA	CRU	BROJ ISPLATE	IR
			SIFRA OSOBE	R
			DATUM ISPLATE	I U
			IZNOS	I U
ODRZAVANJE SIFARNIKA	JEZIK	CRUD	NAZIV JEZIKA	I U
			SIFRA JEZIKA	IRU
	ODELJENJE	CRUD	MESTO	I U
			NAZIV ODELJENJA	I U
			SIFRA ODELJENJA	IRU
	RAD.MESTO	CRUD	SIFRA RM	IRU
			NAZIV RM	I U

Slika 3.11. CRUD i IRUN matrica za dokument "Karton isplata"

Aktivnost
2.1.4. Definisanje dijagrama toka podataka

Dijagram toka podataka (DTP) grafički je opis koji povezuje procese, tokove podataka (dokumenta), skladišta podataka (kartoteke). Drugim rečima, dijagram toka podataka se koristi za prikaz na onom nivou gde su mogu definisati ulazi, izlazi, tokovi, skladišta i procesi.

Dijagram toka podataka ili Data Flow Diagram (DFD) ugrađen je u softverski proizvod BPwin i koristi se pri modeliranju procesa. Dijagram toka podataka fokusira problem tokova podataka između procesa, a vrši i analizu skladišta podataka radi maksimalnog povećanja njihove raspoloživosti i smanjenja vremena pretraživanja.

Dijagram toka podataka treba da nadomesti nedostatke koji postoje u IDEF0 metodologiji i definišu se za niže nivoe dekompozicionog dijagrama.

Dakle, DTP je skup procesa koji se paralelno izvršavaju i ne treba ga mešati sa dijagramom toka programa (flow chart). Dijagram toka programa je akcioni dijagram, tj. sekvencijalni proces, dok je dijagram toka podataka-skup paralelnih procesa i veza tokova podataka i skladišta podataka između njih.

Primer razlika između dijagrama toka podataka i dijagrama toka programa (flow chart) može se opisati na sledeći način:

"Ako je redosled aktivnosti u jednom preduzeću takav da jedan radnik obrađuje jedan dokument dok svi ostali ne rade ništa, pa kad završi obradu dokumenta, aktivira se sledeći radnik a prethodni čeka, onda je dijagram toka programa identičan dijagramu toka podataka."

Dijagram toka podataka se grafički opisuje preko:

- procesa (Processes),

- toka podataka (Data Flow),
- skladišta podataka (Data stores),
- spoljnih objekata (External agents).

Proces (Processes)

Proces je niz operacija kojima se transformišu ulazni podaci u izlazne.

Svaki proces obrade podataka definisan je:

- nekim ulaznim tokom podataka,
- procesom koji se obavlja na osnovu jasno specificirane logike obrade, korišćenjem podataka iz ulaznog toka ili nekog skladišta podataka, a čiji je rezultat obrade izlazni tok podataka i/ili ažurirani podaci u skladištu podataka.

Proces je označen:

- brojnom oznakom kojom je referenciran proces i
- nazivom procesa.

Način definisanja brojne oznake je vezan za označavanje nivoa dekomponovanja u dekompozicionom dijagramu.

Grafički se proces predstavlja zaobljenim pravougaonikom, tako da, za razliku od IDEF0, stranice pravougaonika nisu orijentisane po ICOM metodologiji, tj. ulazi i izlazi se mogu definisati sa bilo koje strane.

Tok podataka (Data Flow)

Tok podataka je put (grafički označen strelicom) kojim protiču grupe podataka (dokumenta, formulari, obrasci), koji pokazuje između kojih elemenata se odvija tok podataka.

Tok podataka se predstavlja orijentisanom pravom i ima jedinstveno ime, koje odražava značenje podataka i vezuje se za proces sa bilo koje strane pravougaonika (za razliku od IDEF0 metodologije gde je važno sa koje strane se prilazi pravougaoniku).

Skladište podataka (Data stores)

Skladište podataka (kartoteka, fascikla, datoteka) služi za čuvanje podataka, odnosno definiše se kao tok podataka u mirovanju; povezano je isključivo s procesima sistema preko tokova podataka.

Skladište podataka treba da omogući:

- akumuliranje sadržaja toka podataka,
- povezivanje isključivo sa procesima preko toka podataka, tako da:
  - tok podataka KA skladištu označava operaciju održavanja, tj. ubacivanja, izbacivanja i promene sadržaja;
  - tok podataka OD skladišta označava korišćenje skladišta za izveštavanje.

Grafički se skladište podataka predstavlja pravougaonikom bez desne strane.

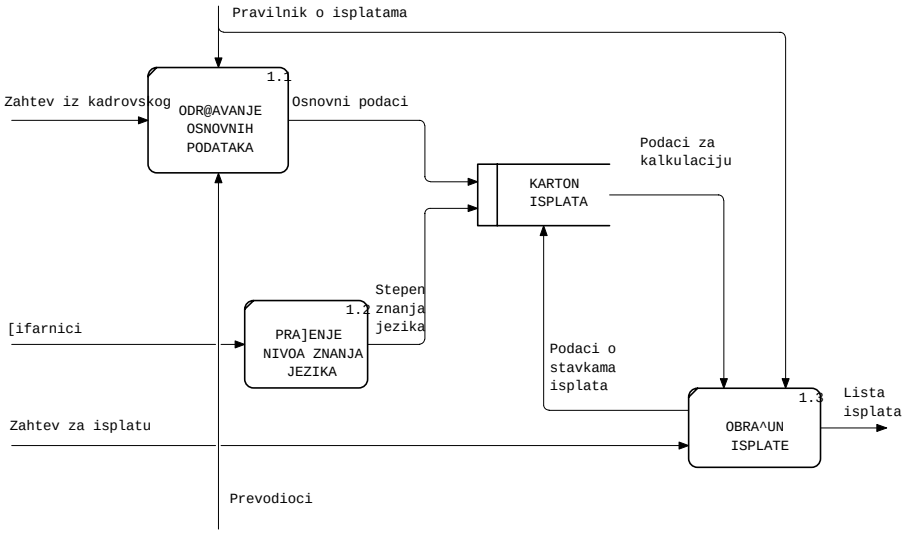
Spoljni objekat (External agents)

Spoljni objekat (External agents ili interface) treba da izvrši povezivanje sa objektima van konteksta posmatranog sistema i predstavlja ponore i/ili izvore tokova podataka. Inače, predstavlja se dvostrukim pravougaonikom, u koji se upisuje ime objekta.

Imajući sve to u vidu, za Aktivnost *"1. Održavanje podataka o prevodiocima"* (sa slike 3.9.) biće definisan dijagram toka podataka.

Dijagram toka podataka za aktivnost "1. Održavanje podataka o prevodiocima"

Ako se posmatra test-primer prikazan na slici 3.12. za nadređenu aktivnost *"1. Održavanje podataka o prevodiocima"*, definisani su tri procesa i jedno skladište podataka ("Karton isplata").



Slika 3.12. Dijagram toka podataka za aktivnost "1. Održavanje podataka o prevodiocima"

Aktivnost *"1.1. Održavanje osnovnih podataka"* treba, na osnovu ulazne informacije "Zahteva iz kadrovskog", da omogući evidentiranje osnovnih podataka o prevodiocima, a kao izlaz - da definiše osnovne podatke o prevodiocima koji se smeštaju u skladište podataka "Karton isplata". Osnovni podaci o prevodiocu se nalaze na zaglavlju kartona isplata (videti sliku 2.5.).

Za potrebe procesa *"1.2. Praćenje nivoa znanja jezika"* prate se elementi stepena znanja stranih jezika pojedinih prevodilaca. Za ovaj proces ulaz su odgovarajući šifarnici jezika, a izlaz je stepen znanja koje poseduje prevodilac. Stepen znanja se evidentira u karton isplata i smešta skladište podataka "Karton isplata".

Na osnovu ulaza "Zahtev za isplatu" i podataka za kalkulaciju iz kartona isplata u procesu *"1.3. Obračun isplate"* obračunava se isplata za obavljeni posao i pravi se lista isplata.

Kao i u prethodnim fazama, potrebno je verifikovati izvedene aktivnosti, pa se kroz sledeću aktivnost *"2.1.5. Provera detaljnih zahteva"* opisuje način verifikacije detaljnih zahteva.

Aktivnost
2.1.5. Analiza detaljnih zahteva

Aktivnost *"2.1.5. Analiza detaljnih zahteva"* treba da bude svojevrsna rekapitulacija do sada obavljenih aktivnosti (slika 3.13).

Ativnost *"1. Provera prikupljenih informacija"* predstavlja proveru prethodne faze, vezane za postavljanje modela reinženjeringa, jer postoji mogućnost da u međuvremenu dođe do nekih izmena.

Ulaz je definisana studija iz prethodne faze i zahtev da se izvrši ekspertiza, a izlaz su izmenjene ili nove činjenice koje zadovoljavaju postavljene ciljeve projekta i koje izvode odgovarajući eksperti, uz pomoć učesnika u izradi studije.

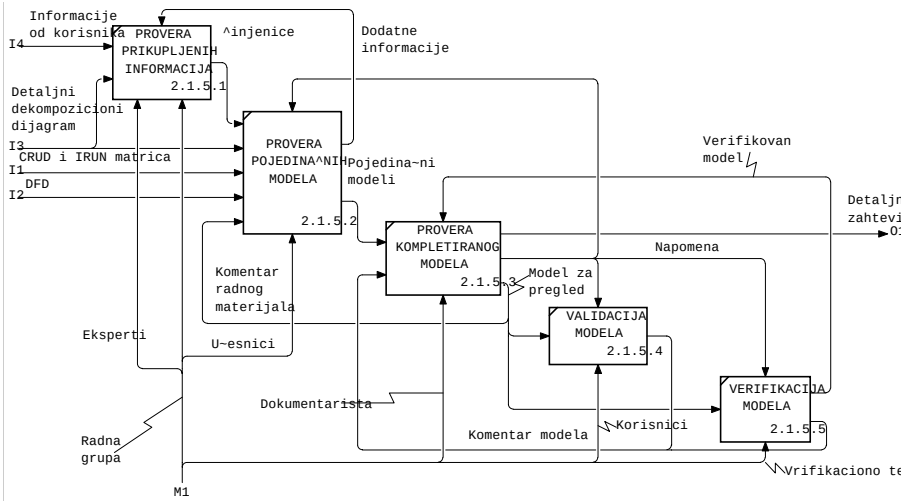
Sledećom aktivnosti *"2. Provera pojedinačnih modela"*, u svetlu novih činjenica, definisanih prethodnom aktivnošću učesnika, tj. autora pojedinačnog modela može zahtevati dodatne informacije, a kao rezultat ove aktivnosti su pojedinačni modeli.

S obzirom na to da se prilikom izvođenja aktivnosti *"3. Provera kompletiranog modela"* moraju napraviti pojedini kompromisi i odstupanja, to kao povratna informacija ide "Komentar radnog materijala ", kao i model za pregled.

Kako su korisnici konačne sudije, to oni u okviru aktivnosti *"4. Validacija modela"* treba da prekontrolišu valjanost modela. Izlaz iz ove aktivnosti su komentari korisnika koji predstavljaju ulaz za prethodnu aktivnost.

Na kraju, potrebno je da odgovarajuće verifikaciono telo izvrši proveru modela u okviru aktivnosti *"5. Verifikacija modela"*. Ako postoje primedbe, one su kao komentar modela ulaz u aktivnost *"3. Provera kompletiranog modela"*, a ako je model verifikovan - on se šalje kao kontrolna informacija u aktivnost *"3. Provera kompletiranog modela"*. Nakon verifikacije kao izlaz se dobija detaljni IDEF0 model koji je rezultat aktivnosti *"2.1. Definisanje detaljnih aktivnosti".*

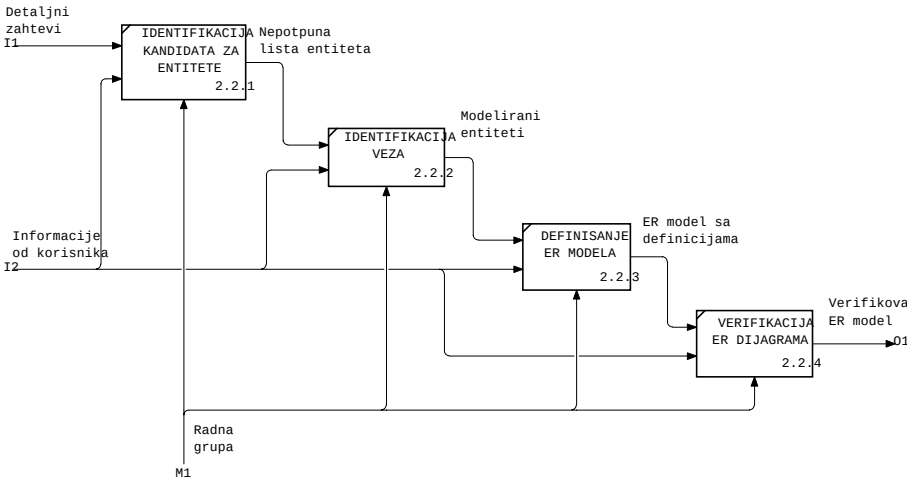
Dosadašnjim aktivnostima modelirani su procesi, tj. definisana je dinamika sistema, dok će u sledećem koraku biti izvršeno modeliranje podataka, tj. biće definisana statika sistema.



Slika 3.13. Redosled koraka provere detaljnih zahteva

Aktivnost
2.2. Kreiranje ER dijagrama

Na osnovu prethodno izvedenih opsežnih priprema i definisanih entiteta i atributa u aktivnosti "2.2. Kreiranje ER dijagrama" pristupa se modeliranju podataka, tj. definisanju entitetnog dijagrama. Za kreiranje ER dijagrama treba koristiti tehniku za opisivanje strukture podataka i poslovnih pravila pod nazivom *model podataka***,** kojim se definišu entiteti *(entities).* Pri tom svaki entitet ima svoje osobine, tj. atribute *(attributes),* a sve je to povezano vezama *(relationships)*. Prema ustaljenim konvencijama, velikim slovima u jednini pišu se entiteti, a atributi i veze malim slovima.



Slika 3.14. Dekompozicioni dijagram za aktivnost "2.2. Kreiranje ER dijagrama"

U okviru aktivnosti *"2.2. Kreiranje ER dijagrama"* definišu se sledeće podaktivnosti:

- Aktivnost 2.2.1. Identifikacija kandidata za entitete,
- Aktivnost 2.2.2. Identifikacija veza,
- Aktivnost 2.2.3. Definisanje ER modela,
- Aktivnost 2.2.4. Verifikacija ER modela.

Na slici 3.14. prikazan je dekompozicioni dijagram kojim se povezuju definisane aktivnosti, koje su predmet daljih razmatranja.

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 3.14.

Aktivnost
2.2.1. Identifikacija kandidata za entitete

S obzirom na to da svaki sledeći korak zahteva kontrolu i reviziju prethodnog, to su potrebne i ovde provera i identifikacija kandidata za entitete.

Za aktivnost "*2.2.1. Identifikacija kandidata za entitete*" polazi se od objekata posmatranja. Objekt posmatranja je sve što se može jednoznačno identifikovati, pa samim tim i izolovati iz okoline i opisati. Tako je objekt posmatranja i "entitet**".** Entitet je osoba, stvar, događaj, pojam (realni ili apstraktni) koji je od trajnog interesa za preduzeće, tj. nešto što se želi pojedinačno posmatrati.

Za potrebe definisanja ER dijagrama, na primer, mogu se posmatrati sledeći objekti, i to:

- fizički objekti (vozilo, mašina,...),
- osobe,
- mesta (adrese, koordinate na karti,...),
- organizacije (preduzeća, zavod,...),
- grupe/klase/tipovi (tip proizvoda, klasa poslova,...),
- ugovori,
- potraživanja (narudžbe, fakture,...),
- prenos/ premeštaj (stvari, vozila, novca,...),
- pridruženje (zadatak - osoba, vozila, vožnja,...),
- pripadnost/članstvo (komponente - sastavi,...) i dr.

Za navedene moguće entitete treba:

- odrediti prikladne radne nazive;
- napraviti grupe entiteta (ako ih je više od 15);
- po grupama, tražiti dodatne entitete (posmatrati najvažniji entitet);
- po potrebi, rearanžirati grupe.

Kako se entiteti opisuju preko svojih osobina, tj. atributa, to se identifikacija atributa može izvesti i na sledeći način.

Treba poći od postavke da svaki atribut u jednom trenutku vremena ima neku vrednost, zatim analizirati tu vrednost i na osnovu toga proširiti listu entiteta na sledeći način:

- Na osnovu prethodne analize strukture teksta, imenicu po analogiji smatrati entitetom.
- Na osnovu sličnosti ATRIBUTA koji mogu pripadati entitetu, uočava se značajna razlika (sličnost), što može da ukaže na to da je reč o različitim (istim) objektima.
- U toku identifikovanja entiteta, za svaki tip entiteta mora postojati jedan atribut (ili grupa atributa) koji jedinstveno identifikuje konkretni entitet u okviru tog tipa.
- Atribut koji identifikuje drugi tip entiteta je entitet.
- Entitet je i atribut koji je istovremeno i atribut drugog entiteta.
- Na osnovu **pasivne i aktivne uloge** veze mogu se definisati odgovarajući tipovi entiteta (slika 3.15):

ULOGA VEZE	TIP ENTITETA
RUKOVODI	RUKOVODILAC
RUKOVOĐEN	ODELJENJE
NARUCUJE	KUPAC
NARUCEN	PROIZVOD

Slika 3.15. Uticaj uloge veze na definisanje entiteta

- Na osnovu **atributa na dokumentima** mogu se identifikovati entiteti (slika 3.16):

ATRIBUT	PITANJE	ODGOVOR
BOJA	BOJA CEGA?	PROIZVOD
STAROST	STAROST CEGA?	RADNIK
DATUM	DATUM CEGA?	NARUDZBE

Slika 3.16. Uticaj atributa na definisanje entiteta

- Na osnovu interesa posmatranja, atribut može biti entitet, a može biti i atribut entiteta, što zavisi od interesovanja, odnosno od toga koji se deo realnog sveta i koji pogled na njega želi predstaviti. Npr., ako je osnovni objekt od interesa-kuća, onda je entitet-kuća, a atribut-ulica, a ako su od interesa-ulice onda su atributi-kuće. Objekti od interesa posmatranja prikazani su na slici 3.17.

Entitet "KUCA"	
Atributi	ulica kucni broj godina izgradnje broj spratova broj stanova

Entitet "ULICA"	
Atributi	broj kuca levo broj kuca desno duzina kolovoza sirina kolovoza

Slika 3.17. Interes posmatranja

Na osnovu definisanih kandidata za entitete pristupa se identifikaciji mogućih veza između entiteta.

Aktivnost
2.2.2. Identifikacija veza

Celokupna dosadašnja aktivnost bila je usmerena na definisanje entiteta i atributa kroz CRUD i IRUN matrice, dok se u okviru aktivnosti *"2.2.2. Identifikacija veza"* prvi put definišu veze između entiteta. Veza predstavlja interakciju među objektima, tj. znanje o njihovoj međusobnoj povezanosti. Vezama se definišu zavisnosti između entiteta, odnosno opisuju načini povezivanja (uzajamna dejstva) entiteta.

Predmet daljih razmatranja predstavlja identifikacija veza, i to:

- povezivanje entiteta na osnovu odgovarajućih interesa;
- definisanje zavisnosti entiteta (gde se definišu nezavisni i zavisni entiteti);
- definisanje značenja zavisnosti (definisanjem referencijalnog integriteta);
- izvođenje varijacija zavisnosti (nalaženjem optimalne);
- izbor entiteta 'roditelj' i entiteta 'dete'.

Veze u rečenici su 'glagolske fraze' koje pokazuju kakav je odnos među entitetima. Premda glagolske fraze ne opisuju pravila precizno, one dozvoljavaju osobi koja posmatra model da

stekne osnovni osećaj o povezanosti entiteta. Dobra je praksa ako čitanje modela daje valjane rečenice (iskaze).

Na slici 3.18. dat je predlog mogućih fraza kojima se definisu veze između entiteta.

odnosi	vazi	koristi	izgrađuje
zahteva	poslata	postize	potvrđuje
sadrzi	poredi	određuje	prihvata
definise	trazi	upravlja	proverava
salje	odgovora	potvrđuje	nalazi
poseduje	identifikuje	specificira	povezuje
realizuje	podstice	nabavlja	razdvoja
ukljucuje	ima	upucuje	selektuje

Slika 3.18. Glagolske fraze kojima se definišu veze

Imajući u vidu način definisanja potencijalnih entiteta i njihovih veza, u sledećoj aktivnosti biće definisani ER modeli.

Aktivnost
2.2.3. Definisanje ER modela

Kao što je u prethodnom poglavlju istaknuto, realan svet se sastoji iz objekata (entiteta) posmatranja, koji mogu biti ili realni objekti (osoba, mašina, vozilo, kuća), ili apstraktni koncept (preduzeće, radno mesto), događaj (prekršaj, prevoz) ili odnosi (student- predmet, muž-žena). Entiteti su, obično, imenovani imenicama, kao OSOBA, TELEFON, JEZIK, ISPLATA.

Preciznije, može se razmišljati o nekom entitetu kao o setu ili kolekciji (skupu) individualnih objekata zvanih *primerci ili instance (instances).* Jedan primerak je jedan pojavni oblik datog entiteta. Svaki primerak mora imati identitet različit od svih ostalih primeraka.

Npr., može se reći da entitet OSOBA predstavlja skup svih mogućih osoba definisanih kao primerci entiteta (slika 3.19.).

Postupak izvođenja aktivnosti *"2.2.3. Definisanje ER modela"* sadrži sledeće korake:

- definisanje nezavisnih entiteta, tj. pronalaženje 'roditelj';
- definisanje zavisnih entiteta;
- definisanje veza.

OSOBA

Sifra osobe	Prezime	Ime	JMBG	Plata	Stimu-lacija	Datum zaposlenja
827369	STEVIC	ZORAN	1411952710331	8000	0	17.12.80
827499	ALAGIC	MILAN	2503965345611	16000	3000	20.02.81
827521	VUKIC	MILOS	1304970554321	12500	5000	22.02.81
827566	JOVIC	MARA	1511956710343	29750	0	02.04.81
827654	MARTIC	ZORA	2406965345311	12500	14000	28.09.81
827698	BOBIC	IVAN	2304950554322	28500	0	01.05.81
827782	CEBIC	GORAN	2311952710441	24500	0	09.06.81
827788	SUSIC	ZORAN	1103965345611	30000	0	09.06.86
827839	KLJAKIC	STEVAN	1404970554321	50000	0	17.11.81
827844	TUBIC	MIRA	1611956710343	15000	0	08.09.81
827876	ALIMPIC	PETAR	2706965345311	11000	0	19.09.87
827900	JAKIC	VLADA	2904950554322	9500	0	03.12.81
827902	FILIPIC	DRAGAN	1305970554821	30000	0	03.12.81

Slika 3.19. Instance entiteta OSOBA

Definisanje nezavisnih entiteta

Nezavisni entitet je objekat koji ima jednu osobinu koja ga može jednoznačno identifikovati, tj. nezavisni entiteti imaju vlastitu identifikaciju (ne zavise od drugih entiteta). Grafički se nezavisni entiteti prikazuju pravougaonikom u okviru koga se upisuje naziv tipa entiteta u jednini (slika 3.20).

Slika 3.20. Grafički prikaz nezavisnog entiteta OSOBA

Kao što se može videti na slici 3.20, entitet OSOBA predstavlja skup svih mogućih osoba sa kojima se komunicira. Svaki primerak entiteta OSOBA je jedan korisnik. Svaki entitet sastoji se od odgovarajućih primeraka ili instanci, kao što je prikazano na slici 3.19.

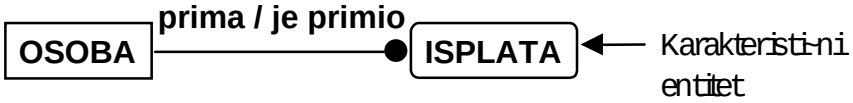
Definisanje zavisnih entiteta

Zavisni entiteti su entiteti čije egzistencija i identifikacija zavise od drugog ili drugih entiteta. Zavisni entiteti se dele u četiri grupe, i to:

- *karakteristične* entitete, tj. entitete koji se ponavljaju više puta za određeni nezavisni entitet;
- *asocijativne* entitete, koji predstavljaju vezu više entiteta;
- *projektne* entitete, koji su slični asocijativnim entitetima, samo što nemaju sopstvene atribute;
- entitete *kategorije,* koji predstavljaju potkategoriju entiteta.

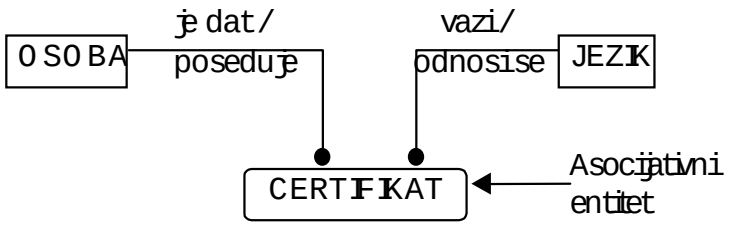
Grafički se zavisni entiteti prikazuju kao zaobljeni pravougaonici u okviru kojih se upisuje naziv tipa entiteta u jednini.

Karakteristični entitet ili slab entitet predstavlja grupu atributa koji se ponavljaju više puta za jedan entitet i koji se identifikuju preko nezavisnog entiteta; npr., entiteti OSOBA i ISPLATA. Za entitet ISPLATA se kaže da je karakterističan entitet, jer zavisi od entiteta OSOBA (slika 3.21).



Slika 3.21. Veza karakterističnog entiteta ISPLATA i nezavisnog entiteta OSOBA

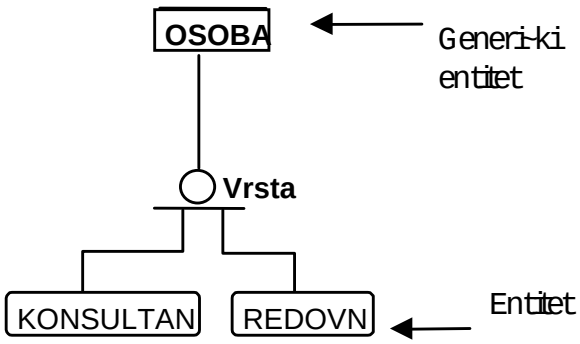
Asocijativni entiteti su sastavljeni od više veza između dva ili više entiteta, kao što se može videti na slici 3.22. Npr., ako OSOBA zna više JEZIK-a i jedan JEZIK poznaje više OSOBA, tada je CERTIFIKAT asocijativni entitet koji opisuje veza između entiteta: OSOBA i JEZIK. Dakle, asocijativni entiteti nose informaciju o višeznačnoj vezi.



Slika 3.22. Veza asocijativnog entiteta CERTIFIKAT sa nezavisnim entitetima OSOBA i JEZIK

Projektni entitet (designative) sličan je asocijativnom entitetu, s tim što nema sopstvene atribute. U prethodnom primeru, CERTIFIKAT se može koristiti kao projektni entitet pod uslovom da ne nosi nikakvu informaciju, izuzev identifikacionih oznaka za OSOBU i JEZIK.

Entitet kategorija (category) zavisan je entitet koji ima tzv. vezu tipa potkategoriju (sub--category). Kod entiteta tipa kategorija definišu se: nadređeni entitet koji ima zajedničke osobine (npr., entitet OSOBA) i podređeni entiteti (entiteti: KONSULTANT i REDOVNI) koji se identifikuju ključem nadređenog i poseduju svoje specifične osobine (slika 3.23).



Slika 3.23. Primer potkategorije entiteta

Definisanje veza

Za razliku od hijerarhijskih i mrežnih modela, gde se relacije prikazuju na fizičkom nivou kao pointeri na slogove, relacioni model prikazuje relacije na logičkom nivou i te relacije se zovu veze.

Kao što se u realnom svetu uspostavljaju određene veze između objekata, po istoj analogiji se definišu i veze između entiteta. Veza je asocijacija između dva ili više entiteta, tj. predstavlja odnos koji postoji među objektima, bilo u realnosti, ili u mislima. Veza u IDEF1X metodologiji se prikazuje kao linija koja povezuje dva entiteta, sa tačkom na jednom kraju i glagolskom frazom napisanom duž linije.

Entitet od koga je uspostavljena veza zove se 'roditelj' (parent), a entitet ka kome je uspostavljena veza zove se 'dete' (child).

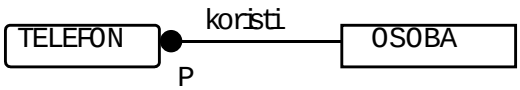
Veza 'roditelj'-'dete' je asocijacija između entiteta gde su svi primerci entiteta 'roditelj' asocirani sa nula, jedan ili više primeraka entiteta 'dete', a svi primerci entiteta 'dete' su asocirani sa nula ili jedan - primerkom entiteta 'roditelj'.

Drugim rečima, način povezivanja dva entiteta ima osobine koje se nazivaju kardinalnost, koja pokazuje "koliko nečega" od dva entiteta može biti uključeno (sadržano).

Da bi se sve to bolje razumelo, poslužiće primer sa slike 3.24. Prvo treba definisati rečenicu gde je glagol <koristi> veza između entiteta OSOBA i TELEFON.

OSOBA <koristi> jedan ili više TELEFON-a

U ovom primerku veza između dva entiteta je izabrana tako da ima oblik koji je poznat kao jedan ili više. To znači da je jedan (i samo jedan) primerak entiteta osoba (entitet 'roditelj') povezan sa više primeraka entiteta telefon (entitet 'dete'). Na osnovu definisane rečenice, korišćenjem glagolske fraze <koristi>, definiše se primer relacije između entiteta 'roditelj' OSOBA i entiteta 'dete' TELEFON (slika 3.24).



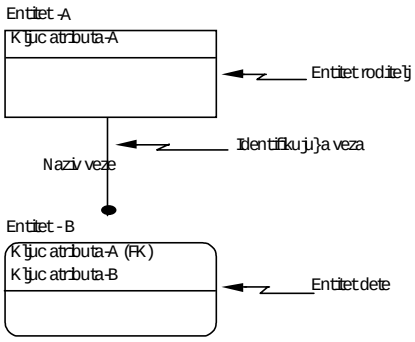
Slika 3.24. Primer relacije

U daljem tekstu detaljno će biti razmatrani sledeći tipovi veza:

- *identifikujuće* veze, koje entitet 'dete' identifikuju kroz njegovu vezu sa entitetom 'roditelj';
- *neidentifikujuća* veza, koja ne identifikuje 'dete' preko identifikatora 'roditelj';
- *veza kategorije,* tj. veza prema podtipovima;
- neodređujuća veza **-** *više prema više***.**

Identifikujuće veze

Veza se zove *identifikujuća* zato što ključevi entiteta 'roditelj' predstavljaju deo identiteta entiteta 'dete', tj. entitet 'dete' zavisi od entiteta 'roditelj' preko identifikatora. Dakle, ako se primerak entiteta 'dete' identifikuje preko asocijacije sa entitetom 'roditelj', onda se veza definiše kao identifikujuća veza, i svaki primerak entiteta 'dete' mora biti povezan sa najmanje jednim primerkom entiteta 'roditelj'.



Slika 3.25. Identifikujuća veza

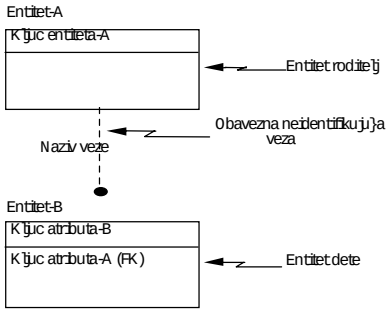
Identifikujuća veza je prikazana punom linijom i povezuje entitet 'roditelj' sa entitetom 'dete' sa tačkom na strani entiteta 'dete' (slika 3.25).

U identifikujućoj vezi entitet 'roditelj' ima svoj nezavisni primarni ključ (Ključ entiteta-A), a entitet 'dete' ima složeni ključ koji se sastoji od svog ključa (Ključ entiteta-B) i prenesenog roditeljskog ključa ŠKljuč entiteta-A (FK)Ć. Dakle, instance entiteta 'roditelj' se definišu nezavisno a instance entiteta 'dete' se ne mogu identifikovati bez identifikatora entiteta 'roditelj'.

Neidentifikujuće veze

Ako se svaki primerak entiteta 'dete' može jedinstveno identifikovati, bez znanja veze sa primerkom entiteta 'roditelj', onda se takva veza definiše kao *neidentifikujuća veza***.**

Neidentifikujuće veze su prikazane isprekidanom linijom koja povezuje entitet 'roditelj' i entitet 'dete' sa tačkom na strani entiteta 'dete'.



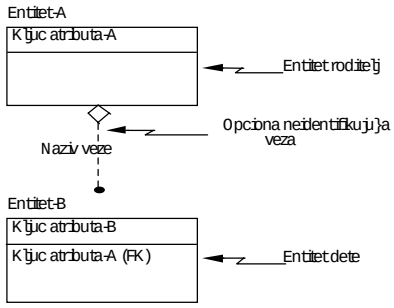
Slika 3.26. Neidentifikujuća obavezna veza

Neidentifikujuća ili slaba veza zavisi od načina definisanja ključeva od 'roditelj' ka 'dete'tu na dva načina:

- kao obavezna neidentifikujuća veza i
- kao neobavezna (opciona) neidentifikujuća veza.

Ako je veza (relationships) *obavezna* (No Nulls ili Mandatory) iz perspektive 'roditelj', onda je 'dete' *egzistencijalno zavisno* od 'roditelj' (slika 3.26). No nulls ili Mandatory znači da je obavezan unos prenesenog ključa entiteta 'roditelj' u okviru entiteta 'dete' [Ključ entiteta A (FK) slika 3.26].

Ako je veza *neobavezna* (Nulls Allowed ili Optional), tada 'dete' niti je egzistencijalno niti identifikaciono zavisno, ali poštuje tu vezu. Null Allowed ili Optional znači da nije obavezan (može biti Null) unos prenesenog ključa entiteta 'roditelj' u okviru entiteta 'dete' [Ključ entiteta A (FK) slika 3.27].



Slika 3.27. Neidentifikujuća neobavezna veza

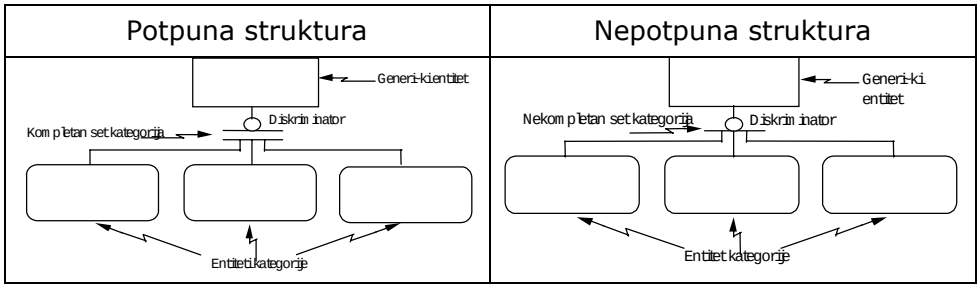
ERwin koristi romb (diamond) da bi naznačio slučaj egzistencijalne i identifikacione *nezavisnosti*. Romb može postojati samo u slabim vezama (pošto je jaka veza u okviru primarnog ključa, a primarni ključ ne može da ima NULL vrednost).

Veza kategorije

Veza kategorije je definisana za hijerarhijsku vezu između nadređenog generalizovanog (generičkog) entiteta koji sadrži zajedničke osobine podređenih entiteta kategorije (slika 3.28).

Ovaj tip veze se deli na:

- *potpunu strukturu* (tzv. kompletan set kategoriju) kada je zatvoren skup entiteta kategorije;
- *nepotpunu strukturu* (tzv. nekompletnu set kategoriju) kada nije zatvoren skup entiteta kategorije.



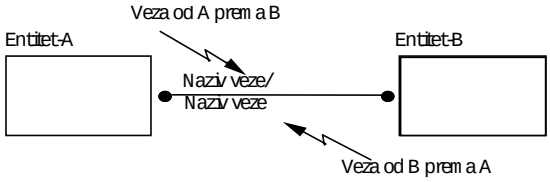
Slika 3.28.Vrste entiteta kategorije

Potpuna struktura se definiše za tačno određeni broj entiteta kategorije i ne može se više nijedan uključiti; nepotpuna struktura ostavlja mogućnost uključivanja drugih entiteta kategorije.

Potpuni opis vezan za ovaj tip veze detaljno će biti opisan u sledećem poglavlju.

Neodređujuća veza

Neodređujuća veza je nespecificirana, a govori o tome da se jedan entitet (Entitet A) pridružuje većem broju entiteta drugog tipa (Entitet B) i obrnuto (slika 3.29).



Slika 3.29. Veza više prema više

Ovaj tip veze zahteva, prema IDEF1X metodologiji, da se prevede uvođenjem asocijativnog entiteta između entiteta A i entiteta B.

Izvođenje prethodnih aktivnosti zahteva i odgovarajuću verifikaciju, što je objašnjeno u sledećoj aktivnosti.

Aktivnost
2.2.4. Verifikacija ER dijagrama

Verifikacija ER dijagrama zahteva opštu proveru entiteta, i to: naziva, pojavljivanja, zavisnosti i dr. Provera se izvodi pitanjima: šta ako..., imajući uvek u vidu da se to mora obavezno izvoditi sa korisnicima. Za verifikaciju treba koristiti analogiju sa strukturom teksta, kao što je prikazano na slici 3.30.

TEKST	ER KONCEPT
IMENICA	ENTITET
GLAGOL	VEZA
PRIDEV	ATRIBUT ENTITETA
PRILOG	ATRIBUT VEZE
GLAGOL.IMENICA	MESOVITI TIP ENTITETA-VEZA
RECENICA	OBJEKTI POVEZANI VEZAMA
POGLAVLJE	LOKALNI PODMODEL

Slika 3.30. Analogija teksta i ER koncepta

U sledećem koraku biće definisan ER dijagram za dokumen "Karton isplata".

ER dijagram za dokument "Karton isplata"

Razvijajući dalje primer, na slici 3.31. prikazan je ER dijagram definisan za dokument "Karton isplata".

Poslovna pravila 'roditelj'	**Opis**
A JEZIK odnosi se many CERTIFIKATs. (*JEZIK se odnosi na više CERTIFIKATA*).	*Jedan JEZIK odnosi se na nula, jedan ili više CERTIFIKATA. To znači da može biti definisan JEZIK, a da nije izdat nijedan CERTIFIKAT.*
A ODELJENJE radi many OSOBAs. (*U ODELJENJU radi više OSOBA)*	*ODELJENJU pripada nula, jedan ili više OSOBA. To znači da odeljenje može da bude bez zaposlenih.*
Based on Pol, A OSOBA is A ZENA or A MUSKARAC. *(OSOBA je ZENA ili MUSKARAC)*	*U odnosu na atribut Pol entiteta OSOBA, OSOBA je ZENA ili MUSKARAC. Obavezan unos jedne od alternativa.*
Based on Vrsta, A OSOBA is A REDOVNI, A KONSULTANT, or another OSOBA type. *(OSOBA je REDOVNI ili KONSULTANT ili neki drugi tip OSOBE).*	*U odnosu na atribut Vrsta entiteta OSOBA, OSOBA je REDOVNI ili KONSULTANT. Nije obavezan unos jedne od alternativa.*
A OSOBA je rukovodilac many OSOBAs.	*OSOBA je rukovodilac nula, jedan ili više OSOBA.*

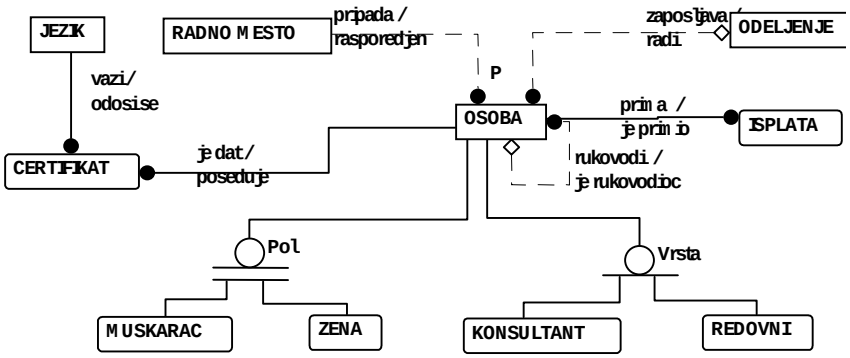
(OSOBA je rukovodilac više OSOBA)	
A OSOBA poseduje many CERTIFIKATs. *(OSOBA poseduje više certifikata).*	*Jedna OSOBA može posedovati nula, jedan ili više CERTIFIKATA.*
A OSOBA je primila many ISPLATAs. *(OSOBA je primila više ISPLATA)*	*OSOBA prima nula, jednu ili više ISPLATA.*
A RADNO MESTO raspoređen at least one OSOBA. *(Na RADNO MESTO mora biti raspoređena najmanje jedna OSOBA)*	*RADNO MESTO se ne može otvoriti ako ne postoji najmanje jedna zaposlena OSOBA.*

Poslovna pravila 'roditelj' počinju rečenicom u kojoj je naziv entiteta 'roditelj' na početku rečenice, a naziv entiteta 'dete' na kraju.

Poslovna pravila 'dete'ta	**Opis**
A ISPLATA je primio exactly one OSOBA. *(Jednu ISPLATU prima jedna i samo jedna OSOBA).*	*Jedna ISPLATA se odnosi isključivo za jednu OSOBU.*
A KONSULTANT is a type of OSOBA. *(KONSULTANT je tip OSOBE)*	*KONSULTANT je tip ili specijalizacija entiteta OSOBA.*
A MUSKARAC is a type of OSOBA. *(MUSKARAC je tip OSOBE)*	*MUSKARAC je tip ili specijalizacija entiteta OSOBA.*
A OSOBA raspoređen exactly one RADNO MESTO. *(OSOBA je raspoređena obavezno na jedno RADNO MESTO).*	*To znači da se NE može primiti novi čovek u preduzeće a da ne bude raspoređen ni na jedno RADNO MESTO.*
A OSOBA je rukovodilac zero or one OSOBAs. *(OSOBA je rukovodilac nula ili jednoj OSOBI).*	*OSOBA može, a ne mora da bude rukovodilac i može samo da bude jedan rukovodilac.*
A OSOBA radi zero or one ODELJENJEs. *(OSOBA radi u nijednom ili jednom ODELJENJU).*	*To znači da se može primiti novi čovek u preduzeće, a da ne bude raspoređen ni u jedno odeljenje.*
A REDOVNI is a type of OSOBA. *(REDOVNI je tip OSOBE)*	*REDOVNI je tip ili specijalizacija entiteta OSOBA.*
A CERTIFIKAT odnosi se exactly one JEZIK. *(CERTIFIKAT odnosi se na tačno jedan JEZIK)*	*Jedan CERTIFIKAT vezan je za najmanje jedan i najviše jedan JEZIK.*
A CERTIFIKAT poseduje exactly	*Jedan CERTIFIKAT poseduje*

one OSOBA. *(CERTIFIKAT poseduje tačno jedna OSOBA).*	*najmanje jedna i najviše jedna OSOBA.*
A ZENA is a type of OSOBA. *(ZENA je tip OSOBE).*	*ZENA je tip ili specijalizacija entiteta OSOBA.*

Poslovna pravila 'dete'ta počinju rečenicom u kojoj je naziv entiteta 'dete'ta na početku rečenice, a naziv entiteta 'roditelj' na kraju.



Slika 3.31. ER dijagram za dokument "Karton isplata"

ER dijagram prikazan na slici 3.31. predstavlja rezultat, sa jedne strane, analize dokumenta "Karton isplata" (prikazanog na slici 2.5. - pogled odozdo) i sa, druge, rezultat želja i potreba za informacijama, dobijenih na osnovu sprovedenog intervjua (pogled odozgo).

ER dijagram sa slike 3.31. urađen je u ERwin-u i može se predstaviti i na sledeći način - u obliku izveštaja definisanog kao poslovna pravila. Tako se razlikuju poslovna pravila 'roditelj' i poslovna pravila 'dete'ta.

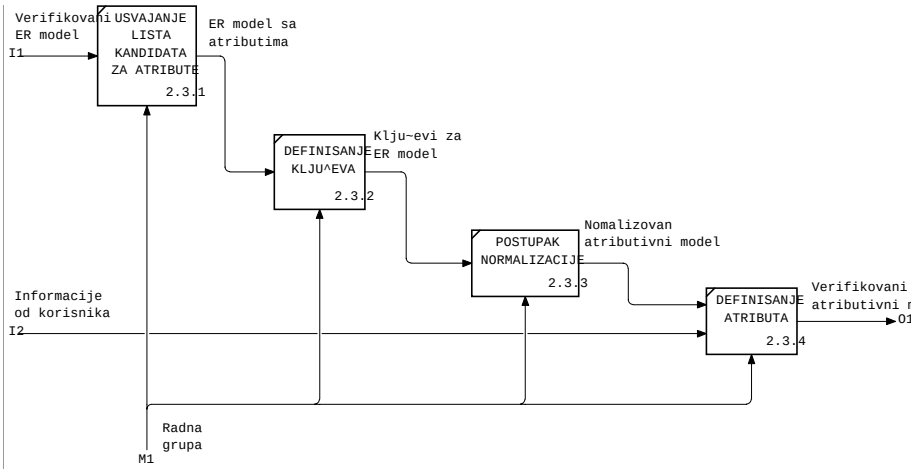
ER modelom definisani su okviri modela podataka, a sledećom aktivnošću biće izvršeno kreiranje atributa za svaki entitet.

Aktivnost
2.3. Kreiranje atributa

Svaki objekat realnog sveta je definisan osobinama, pa po toj analogiji i entiteti imaju atribute kojima se opisuju karakteristična svojstva. U okviru aktivnosti *"2.3. Kreiranje atributa"* definisane su sledeće podaktivnosti:

- Aktivnost 2.3.1. Definisanje lista kandidata za atribute,
- Aktivnost 2.3.2. Definisanje ključeva,
- Aktivnost 2.3.3. Postupak normalizacije,
- Aktivnost 2.3.4. Definisanje atributa.

Na slici 3.32. prikazan je dekompozicioni dijagram za aktivnost *"2.3. Kreiranje atributa".*



Slika 3.32. Dekompozicioni dijagram za aktivnost "2.3. Kreiranje atributa"

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 3.32.

Aktivnost 2.3.1. Definisanje liste kandidata za atribute

Ulaz u aktivnost *"2.3.1. Definisanje liste kandidata za atribute"* je verifikovani ER model, koji u okviru ove aktivnosti kao izlaz ima ER model sa atributima.

Da bi se definisala lista kandidata za atribute, mora se dati odgovor na pitanje šta je interes posmatranja. Da li će po nekoj osobini objekat biti entitet ili atribut, tj. koju će ulogu imati, zavisi od pogleda koji se želi predstaviti.

Osnovna pravila koja se koriste u realizaciji aktivnosti *"2.3.1. Definisanje liste kandidata za atribute"* su:

- Svaki entitet ima proizvoljan broj atributa, što znači da nema ograničenja u broju atributa.
- Određeni atribut pripada jednom i samo jednom entitetu, tako da isti atribut ne može biti opisan u okviru dva ili više entiteta.
- Svako pojavljivanje entiteta ima vrednosti za sve atribute tog entiteta.
- Atribut određenog pojavljivanja entiteta može imati samo jednu vrednost, pa primerak entiteta za određeni atribut ima jednu vrednost.
- Svaki atribut predstavlja jednu određenu činjenicu, tako da i svako značenje vrednosti atributa mora imati jedno dosledno značenje.

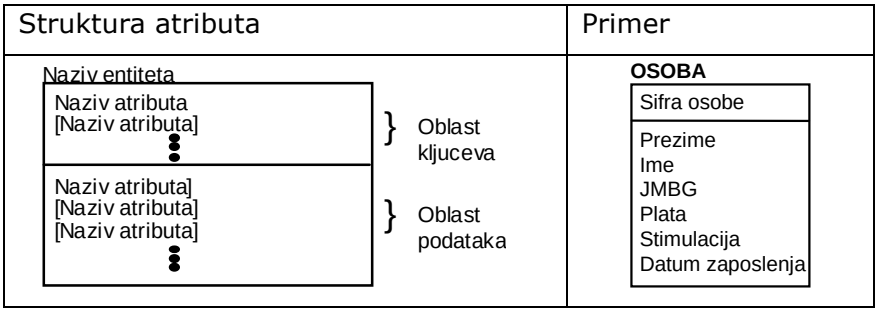
Na osnovu definisane liste kandidata za atribute u sledećem koraku biće definisani svi tipovi ključeva.

Aktivnost 2.3.2. Definisanje ključeva

Aktivnost *"2.3.2. Definisanje ključeva"* ima zadatak da za svaki entitet bude definisan atribut ili kombinacija atributa, čije vrednosti jedinstveno identifikuju svaki primerak entiteta.

Taj atribut ili grupa atributa naziva se *primarni ključ.* Ako ključ čini samo jedan atribut, onda je *prost ključ;* u suprotnom je *složen*. Kao što se vidi na slici 3.33, atributi mogu biti definisani u oblasti ključeva (primarni ključ) ili u oblasti podataka.

Na primeru prikazanom na slici 3.33. entitet OSOBA je predstavljen crtanjem pravougaonika sa imenom entiteta na vrhu i svim atributima u okviru pravougaonika. Atribut "Sifra osobe" je u oblasti ključa, a svi ostali atributi su u oblasti podataka.



Slika 3.33. Struktura atributa

U daljem tekstu detaljno će biti razmotreni sledeći tipovi ključeva:

- primarni ključ,
- alternativni i inverzni ključevi i
- preneseni ključevi.

Primarni ključ

Atributi ili grupe atributa koji mogu biti izabrani kao primarni ključevi nazivaju se 'atributi kandidati za ključ'. Kandidat za ključ mora jedinstveno da identifikuje svaki primerak entiteta. Iz toga sledi da nijedan deo primarnog ključa ne može biti NULL, odnosno prazan (empty) ili nedostajući (missing). Od kandidata za ključeve bira se jedan koji se naziva primarni ključ.

Osnovno pitanje koje se postavlja jeste kako izabrati primarni ključ kojim se identifikuje jednoznačno entitet. Izbor ključa nekog entiteta može biti jedan atribut kojim se definiše jednostavni primarni ključ ili kombinacija atributa kojom se definiše složeni primarni ključ.

Ako se pogleda entitet OSOBA sa slike 3.33, atributi su:

- Sifra osobe
- JMBG
- Prezime
- Ime
- Plata
- Stimulacija
- Datum zaposlenja

Neka je, na primer, atribut 'Sifra osobe' prvi kandidat za ključ, zato što je jedinstven za svaku osobu. Sledeći atribut JMBG bi mogao da bude kandidat za ključ ako su pretpostavke o jedinstvenosti broja korektne. Međutim, verovatno je da neće sve osobe imati JMBG, jer treba koristiti usluge i stranaca. Mada se na prvi pogled čini da ovaj atribut može biti kandidat za ključ, treba ga ostaviti po strani. Sledeći atributi 'Ime i Prezime' ne bi mogli da budu dobar kandidat za ključ, jer mogu postojati dva Zorana Petrovića.

Kombinacija 'Ime i Prezime' i 'Datum zaposlenja' mogla bi biti kandidat za ključ (osim ako ne postoje, npr., dva Zorana Petrovića zaposlena istog dana).

Premda je to nekorektno, neka u ovom primeru kombinacija 'Ime i Prezime' i 'Datum zaposlenja' određuju specifičnu osobu. Tako je dobijen drugi kandidat za ključ: kombinacija 'Ime i Prezime' i 'Datum zaposlenja'.

Dakle, dobijena su dva kandidata za ključ: jedan je 'Sifra osobe' a drugi je grupa atributa koja sadrži atribute: 'Ime i Prezime' i 'Datum zaposlenja' i sada treba izabrati primarni ključ.

Koji će od ova dva kandidata biti izabran za primarni ključ zavisi od analize koja se sprovodi na sledeći način:

- Prvo i najvažnije prilikom izbora primarnog ključa je naći atribut koji neće menjati vrednost za vreme 'života' svakog primerka entiteta, jer ključ određuje identitet primerka entiteta (ako se promeni ključ, to je već drugi primerak). Ovaj uslov

ispunjavaju i jedan i drugi kandidat za ključ.

- Drugo, treba tražiti razumno male ključeve. Ako se može naći jedan atribut, generalno gledano, pronađen je dobar ključ. Taj uslov ispunjava prvi kandidat za ključ.
- Treće, potrebno je izbegavati upotrebu 'inteligentnih' ključeva (na primer, gde struktura brojeva identifikuje grupisanje, lociranje, klasifikaciju, datume itd.). Taj uslov ispunjava prvi kandidat za ključ, jer njegova šifra je neimenovani identifikacioni broj.

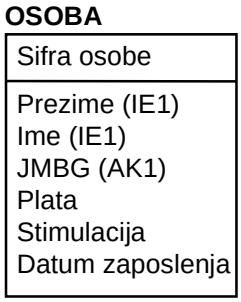
Na osnovu ovako obrazloženih elemenata za izbor primarnog ključa za entitet 'OSOBA' bira se atribut 'Sifra osobe' kao primarni ključ, jer zadovoljava sva tri definisana kriterijuma.

Alternativni i inverzni ključevi

Kandidati za ključ koji nisu izabrani za primarne ključeve mogu se definisati kao alternativni ključevi (Akn).

Alternativni ključ (AKn) predstavlja atribut ili grupa atributa koji jedinstveno identifikuju primerke entiteta. ERwin generiše jedinstven indeks za svaki alternativni ključ. O indeksima će kasnije više biti reči. Ako treba definisati indeks nad atributom ili grupom atributa koji ne identifikuju jedinstveno primerke entiteta definiše se tzv. Inversion Entry (IEn) indeks.

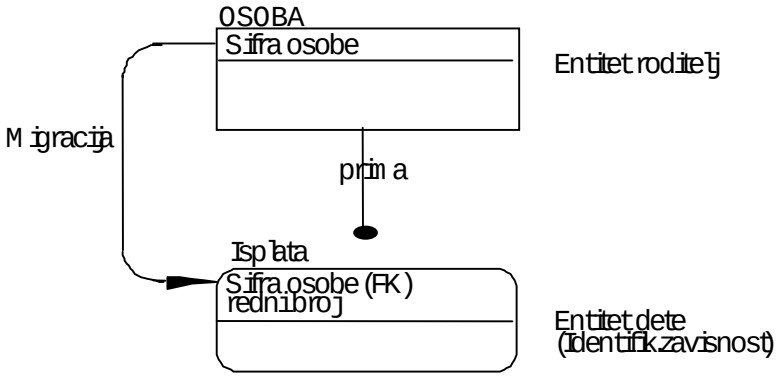
Tako primer sa slike 3.34. dobija alternativni ključ JMBG. Kako je JMBG jedinstven, a svi zaposleni ne moraju imati ovaj broj (npr., zaposleni stranci), to se on usvaja kao alternativni ključ koji ima osobinu jednoznačnosti, ali je zato NULL podatak (ne mora se unositi). Atributi: 'Ime' i 'Prezime' definisani su kao inverzni atributi, tj. kao IE ključ, jer nemaju definisanu jednoznačnost (može se dogoditi da više zaposlenih ima isto 'Ime' i Prezime').



Slika 3.34. Alternativni i inverzni ključevi

Preneseni ključevi

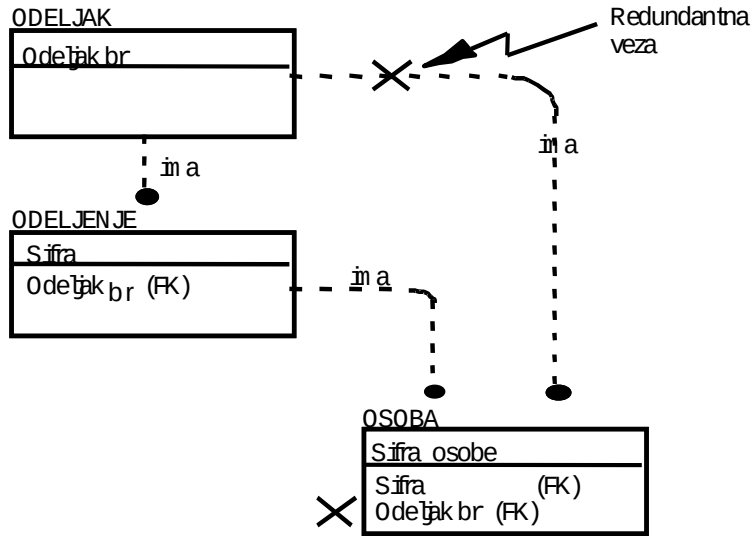
Preneseni ključ je kolekcija atributa koji u posmatranom entitetu nisu ključ, ali su zato ključ u nekom drugom entitetu. Preneseni ključ (Foreign Key) jeste atribut koji povezuje entitet 'dete' sa entitetom 'roditelj' i određen je oznakom FK, koja dolazi iza imena atributa.



Slika 3.35. Migracija primarnog ključa u preneseni ključ

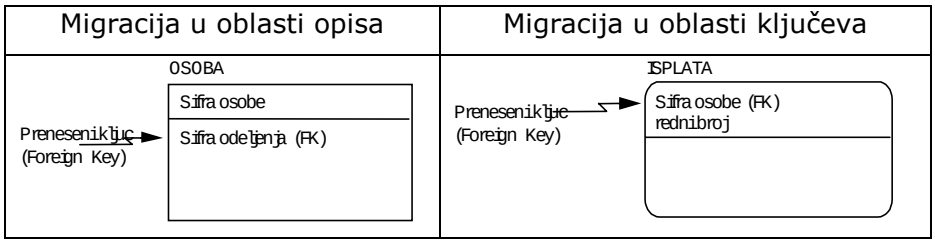
Na slici 3.35. prikazan je postupak migracije primarnog ključa entiteta OSOBA (entitet 'roditelj') u entitet ISPLATA (entitet 'dete'), tj. veza prenosi ključ entiteta 'roditelj' u entitet 'dete'.

Kada se vrši migracija ključeva, tj. povezivanje entiteta, može se dogoditi i situacija prikazana na slici 3.36, gde se pojavljuje i redundantna veza koju treba izbrisati.



Slika 3.36. Primer redundandne veze

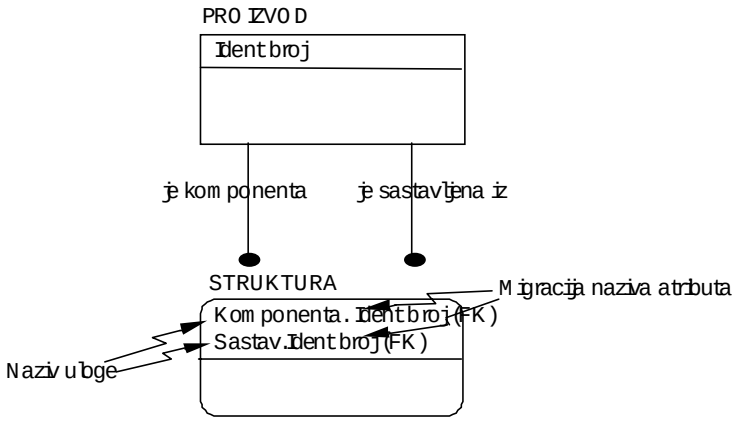
Kao što se na slici 3.37. vidi, migracija ključeva može biti u oblasti opisa i tada je u pitanju neidentifikujuća veza (označena isprekidanom linijom kao na slici 3.36) ili, pak, u oblasti ključeva, kada je u pitanju identifikujuća veza (označena punom linijom kao na slici 3.35).



Slika 3.37. Različiti tipovi migracija

Preneseni ključevi mogu imati i drugo ime, što se definiše kao uloga atributa u entitetu. Ime uloge (Rolename) predstavlja novo ime za prenesene ključne atribute koji definišu ulogu koju ti atributi igraju u tom entitetu. Ime uloge deklariše novi atribut, čije ime treba da opiše poslovno pravilo ugrađeno preko veze koja zahteva preneseni ključ (slika 3.38).

Definisanje imena uloge biće pokazano na primeru definisanja strukture proizvoda.



Slika 3.38. Primer za definisanje imena uloge

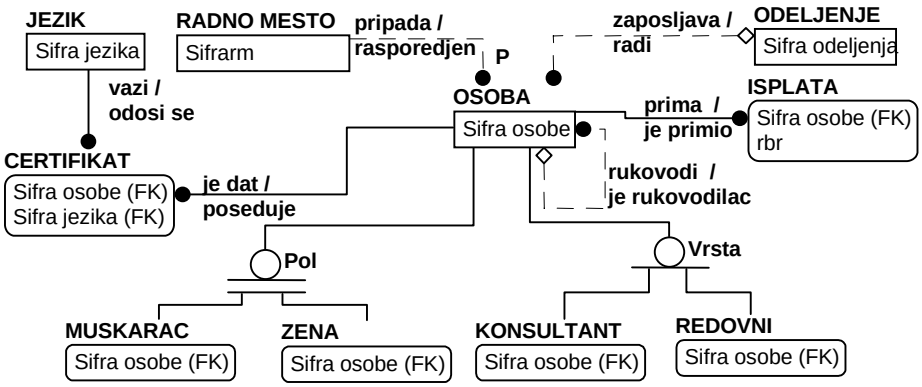
Kako entitet STRUKTURA ima složeni ključ od istog nadređenog entiteta PROIZVOD, to se definisanjem uloge definišu različita imena atributa i identifikuje entitet STRUKTURA.

Detalji za definisanje ovog tipa veze biće kasnije detaljno opisani.

Imajući sve to u vidu, za primer ER modela dokumenta "Karton isplata", u sledećem koraku biće izvršeno definisanje primarnih ključeva.

Definisani primarni ključevi za dokument "Karton isplata"

Na slici 3.39. prikazan je ER model sa definisanim ključevima za primer "Karton isplata", gde entitet OSOBA identifikuje atribut 'Sifra osobe'. Za entitete tipa šifarnika, kao što je entitet ODELJENJE, primarni ključ je 'Sifra odeljenja', za entitet JEZIK primarni ključ je atribut 'Sifra jezika', a za entitet RADNO MESTO primarni ključ je 'Sifrarm'. Zavisni entitet ISPLATA ima složeni ključ koji se sastoji od atributa 'Sifra osobe+rbr', a asocijativni entitet CERTIFIKAT ima složeni ključ od atributa 'Sifra osobe+Sifra jezika'. Potkategorije MUSKO, ZENSKO, KONSULTANT, REDOVNI imaju primarne ključeve prenesene sa generalizovanog entiteta OSOBA ('Sifra osobe').



Slika 3.39. ER model sa definisanim primarnim ključevima

Na osnovu prethodno izvedenih aktivnosti (slika 3.32) u sledećem koraku potrebno je sprovesti postupak normalizacije.

Aktivnost
2.3.3. Postupak normalizacije

Prilikom definisanja atributa, kao što je rečeno, pristupa se modeliranju podataka odozdo nagore (Bottom Up). Ovaj pristup veoma je prihvatljiv za početnike u ovoj oblasti, jer polazi od opipljivih informacija definisanih na dokumentima i u kartotekama. Osnovu za ovaj način modeliranja podataka čine analiza funkcionalnih zavisnosti i postupak normalizacije.

U okviru aktivnost *"2.3.3.Postupak normalizacije"* uklanjaju se sve strukture koje stvaraju redundansu podataka, pa je slogan normalizacije: *Jedna činjenica na jednom mestu.*

Pravilno izveden postupak normalizacije podataka omogućuje korektno izvođenje aktivnosti *"3. Aplikativno modeliranje"*, koja ima strukturu kojom osigurava da u radu sa njom neće biti neželjenih anomalija, kao što su, npr., uništavanje određenih podataka ili neusklađenost između memorisanih podataka kao posledica ažuriranja baze podataka.

Drugim rečima, postupak normalizacije predstavlja transformaciju početnog entiteta u jednu ili više korektnih entiteta ili veza u kojima su svi atributi potpuno funkcijski zavisni od ključa, a međusobno funkcijski nezavisni.

Da bi se mogao opisati postupak normalizacije, treba prethodno opisati pojam funkcijske zavisnosti.

Ako je svakoj vrednosti atributa A u relaciji R priključena samo jedna vrednost atributa B u istoj relaciji, onda je atribut B funkcijski zavisan od atributa A asocijacijom tipa 1.

Funkcijska zavisnost se može definisati između složenog ključa (više atributa) i jednostavnog atributa.

Ako se svakom paru vrednosti atributa A i B relacije R može priključiti tačno jedna vrednost C iste relacije, tada je atribut C funkcijski zavisan od sastavljenog atributa A i B.

Potpuna funkcijska zavisnost se definiše na osnovu definicije funkcijske zavisnosti.

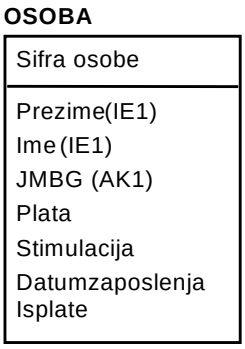
Atribut B je potpuno funkcijski zavisan od atributa A iste relacije, ako je funkcijski zavisan od atributa A, a ne od nekog sastavnog dela atributa A.

Na osnovu ovako izvedenih definicija na primeru entiteta OSOBA biće pokazan postupak normalizacije kroz definisanje prve (1NF), druge (2NF) i treće (3NF) normalne forme.

Definisanje prve normalne forme (1NF)

Može li se na osnovu posmatranja entiteta OSOBA (sa slike 3.40) uočiti greška?

Zbog lakšeg razumevanja, treba prevesti entitet OSOBA u tabelu sa definisanim primercima (koristi se i termin instance).



Slika 3.40. Entitet OSOBA

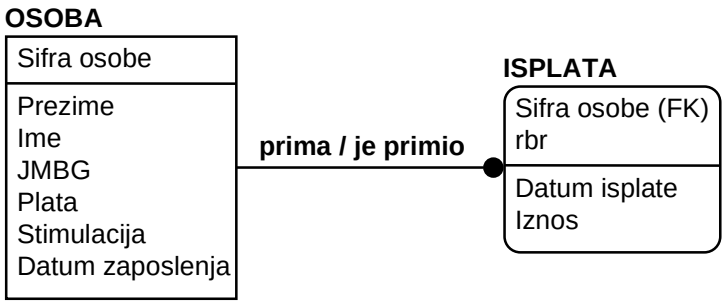
OSOBA

Sif.oso	Prezime	Ime	JMBG	Plata	Stimu	Dat.zapos.	Isplate
827369	STEVIC	ZORAN	141195271033	8000	0	17.12.80	200,300
827499	ALAGIC	MILAN	250396534561	16000	3000	20.02.81	400
827521	VUKIC	MILOS	130497055432	12500	5000	22.02.81	800,300
827566	JOVIC	MARA	151195671034	29750	0	02.04.81	
827654	MARTIC	ZORA	240696534531	12500	1400	28.09.81	200,100
827698	BOBIC	IVAN	230495055432	28500	0	01.05.81	
827782	CEBIC	GORAN	231195271044	24500	0	09.06.81	
827788	SUSIC	ZORAN	110396534561	30000	0	09.06.86	3000,20
827839	KLJAKIC	STEVA	140497055432	50000	0	17.11.81	
827844	TUBIC	MIRA	161195671034	15000	0	08.09.81	
827876	ALIMPIC	PETAR	270696534531	11000	0	19.09.87	
827900	JAKIC	VLADA	290495055432	9500	0	03.12.81	300,400
827902	FILIPIC	DRAGA	130597055482	30000	0	03.12.81	

Slika 3.41. Tabela OSOBA za entitet OSOBA sa slike 3.40.

Problem je u atributu "Isplate". U prethodnim poglavljima u vezi sa entitetima i atributima naglašeno je da sva imena moraju biti u jednom primerku, tj. da se u jedan atribut ne može smestiti više isplata.

S obzirom na to da nije poznato koliko isplata treba zapamtiti, koliko je prostora za to potrebno i šta raditi ako ima više isplata nego prostora, to onda ovakva tabela krši prvu normalnu formu. Da bi se popravila prethodna tabela, treba na neki način ukloniti atribute isplate iz entiteta OSOBA. Jedan od načina je da se to uradi prikazan je na slici 3.42.



Slika 3.42. Veza entiteta OSOBA i ISPLATA

Prikaz dat na slici 3.42. preveden u tabele OSOBA i ISPLATA sa definisanim instancama (slika 3.43) izgleda ovako:

OSOBA

Sif_oso	Prezime	Ime	JMBG	Plata	Stim	Datum
827369	STEVIC	ZORAN	141195271033	8000	0	17.12.80
827499	ALAGIC	MILAN	250396534561	1600	300	20.02.81
827521	VUKIC	MILOS	130497055432	1250	500	22.02.81
827566	JOVIC	MARA	151195671034	2975	0	02.04.81
827654	MARTIC	ZORA	240696534531	1250	140	28.09.81
827698	BOBIC	IVAN	230495055432	2850	0	01.05.81
827782	CEBIC	GORAN	231195271044	2450	0	09.06.81
827788	SUSIC	ZORAN	110396534561	3000	0	09.06.86
827839	KLJAKIC	STEVA	140497055432	5000	0	17.11.81
827844	TUBIC	MIRA	161195671034	1500	0	08.09.81

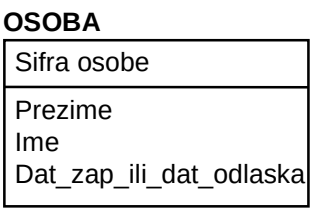
ISPLATA

Sifra osobe	rbr	Datum	Iznos
7369	1	12.12.97	2.433,00
7369	2	12.11.97	2.322,00
7521	1	10.10.97	212
7521	2	11.11.97	232
7521	3	11.12.97	2.122

Slika 3.43. Primerak entiteta za sliku 3.42.

Pošto je otkrivena grupa podataka koji se ponavljaju i od njih stvoren novi entitet ISPLATA, time je učinjen prvi korak prema normalizovanom modelu.

Kao još jedan primer vezan za definisanje prve normalne forme, treba navesti slučaj koji se najčešće pojavljuje - višeznačna upotreba istog atributa, gde se jednim atributom, npr., "Dat_zap ili Dat_odlaska" (slika 3.44) definišu dve činjenice.



Slika 3.44. Višeznačna upotreba istog atributa

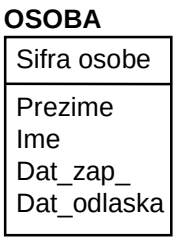
OSOBA

Sif osobe	Prezime	Ime	Dat zapos ili dat odlaska
827369	STARCEVI	ZORAN	17.12.80: 12.12. 1995
827499	ALAGIC	MILAN	20.02.81
827521	VUKIC	MILOS	22.02.81
827566	JOVIC	MARA	02.04.81
827654	MARTIC	ZORA	28.09.81
827698	BOBIC	IVAN	01.05.81:13.09.1990
827782	CEBIC	GORAN	09.06.81
827788	SUSIC	ZORAN	09.06.86
827839	KLJAKIC	STEVAN	17.11.81
827844	TUBIC	MIRA	08.09.81:14.05.1987

Slika 3.45. Primerak entiteta sa slike 3.44.

Dakle, problem je u atributima: 'dat_zap ili dat_odlaska' koji predstavljaju jednu od dve činjenice, početak rada u firmi i prestanak rada u firmi. Ne postoji mogućnosti da se otkrije šta taj datum predstavlja, kao što ne postoji mogućnost da se zapamte oba datuma, iako su, možda, oba poznata. Rešenje nije u tome da atribut može da sadrži dve činjenice, već da postoje dva atributa koji govore o početku i završetku rada.

Stoga je potrebno da se ugrade dva atributa koji nose dve različite informacije (sl. 3.46. i sl. 3.47.) prikazuju izvedenu 1NF za ovaj slučaj.



Slika 3.46. Jednoznačna upotreba atributa

OSOBA

Sif osobe	Prezime	Ime	Dat zapos	dat odlaska
827369	STARCEVI	ZORAN	17.12.80	12.12.1995
827499	ALAGIC	MILAN	20.02.81	
827521	VUKIC	MILOS	22.02.81	
827566	JOVIC	MARA	02.04.81	
827654	MARTIC	ZORA	28.09.81	
827698	BOBIC	IVAN	01.05.81	13.09.1990
827782	CEBIC	GORAN	09.06.81	
827788	SUSIC	ZORAN	09.06.86	
827839	KLJAKIC	STEVAN	17.11.81	

Slika 3.47. Primerak entiteta sa slike 3.46.

I u primeru sa sl. 3.40. i sl. 3.44. pogrešna rešenja ne zadovoljavaju prvu normalnu formu. Menjajući strukturu, sasvim sigurno, jedan se atribut pojavljuje samo jednom u entitetu i nosi samo jednu činjenicu.

Dakle, prva normalna forma je ispunjena ako svaki od atributa entiteta ima jedno značenje i ne više od jedne vrednosti za svaki primerak. Ako je sigurno da svi entiteti i atributi ne nose više činjenica, model zadovoljava prvu normalnu formu.

Ovi elementi su ugrađeni i u ERwin CASE alatu koji prihvata bilo koje ime za definiciju entitetat ili atribut, ali postoje ograničenja:

- ERwin će obavestiti o ponovnom korišćenju imena entiteta, zavisno od postavke opcije o jedinstvenom imenu.
- ERwin će obavestiti o ponovnom korišćenju imena atributa, osim ako je to ime uloge (Rolename). Kada je pridruženo ime uloge za atribut, može biti korišćeno u različitim entitetima.
- Erwin neće dozvoliti ulazak prenesenih ključeva u entitet više nego jednom, osim ako

mu je dodeljeno ime uloge svaki put.

Sprečavajući da se višestruko koristi isto ime, ERwin vodi korisnika da svaki podatak smesti tačno na jedno mesto.

Međutim, potrebna je dalja analiza, jer prva normalna forma nije dovoljna pa se prelazi na definisanje druge normalne forme.

Druga normalna forma (2NF)

Definicija druge normalne forme je sledeća:

Entitet A zadovoljava drugu normalnu formu ako zadovoljava prvu i ako svaki atribut koji nije ključ potpuno zavisi od primarnog ključa.

Za opis ove definicije poslužiće primer višestrukog pojavljivanja istih činjenica.

Ako bi se na primeru (slika 3.48) u entitet ISPLATA stavio atribut ' Datum zaposl.' može se uočiti da ovaj atribut zavisi od dela ključa entiteta ISPLATA (Sifra osobe), a ne od celog ključa entiteta ISPLATA.

ISPLATA

Sifra osobe	rbr	Datum	Datum	Iznos
827369	1	12.12.97	17.12.80	2.433.00
827369	2	12.11.97	17.12.80	2.322.00
827521	2	11.11.97	22.02.81	232
827521	3	11.12.97	22.02.81	2.122

Slika 3.48. Instance tabele ISPLATE koje nisu u 2NF

Da bi se zadovoljila druga normalna forma potrebno je prebaciti atribut 'Datum zaposl.' u entitet OSOBA. Dakle, entitet krši drugu normalnu formu ako podatak može biti pronađen kada se zna samo deo ključa entiteta.

Može da nastane greška druge normalne forme ako se postavi neki atribut nekorektno, a ne postoji algoritam koji bi bez dodatnih informacija, pored onih u modelu, otkrio grešku. U entitetnom dijagramu Erwin ne može znati da ime koje je dodeljeno atributu može predstavljati listu objekata.

Na osnovu korekcija sprovedenih u okviru 2NF pristupa se definisanju treće normalne forme.

Treća normalna forma (3NF)

Definicija treće normalne forme je sledeća:

Entitet zadovoljava treću normalnu formu ako svaki atribut koji nije ključ zavisi od ključa, čitavog ključa i ne služi ničemu drugom osim ključa.

Na primer, bila bi povređena treća normalna forma ako se u entitet ISPLATA ugradi atribut 'Suma isplata'. 'Suma isplata' zavisi od atributa 'Isplata' i može se izračunati.

Pored ovih formi postoje i četvrta i peta i Boyce-Codd-ova forma, a njihova upotreba zavisi od skupa transakcija koje treba izvršiti i ovde se neće razmatrati.

Mora se naglasiti da iskusni projektanti već razmišljaju u 3NF .

Na osnovu prethodno izvedenih aktivnosti (slika 3.32) u sledećem koraku treba definisati atribute.

Aktivnost
2.3.4. Definisanje atributa

Definisanje atributa se izvodi u tri koraka:

- identifikacijom atributa,
- alociranjem atributa i
- revizijom atributa.

Identifikacija atributa se definiše na osnovu zahteva korisnika i poslovne dokumentacije.

Alociranje atributa se izvodi u zavisnosti od toga da li atribut zavisi od ključa ili je opisni.

Revizijom atributa se eliminiše višestruko nastupanje vrednosti atributa pojedinog entiteta i pri tom se za svaki atribut postavljaju pitanja:

- da li je potrebno tražiti višestruke vrednosti za isto pojavljivanje entiteta (Sifra odeljenja, Naziv odeljenja);
- da li atribut može pripadati nekom drugom entitetu;
- da li postoje atributi koji ne nastupaju za neko pojavljivanje entiteta;
- da li postoje izvedeni atributi (koje treba ili odstraniti ili dodati);
- da li postoje atributi bez entiteta;
- da li je prikladan identifikator/ ključ.

Za atribute se mogu definisati :

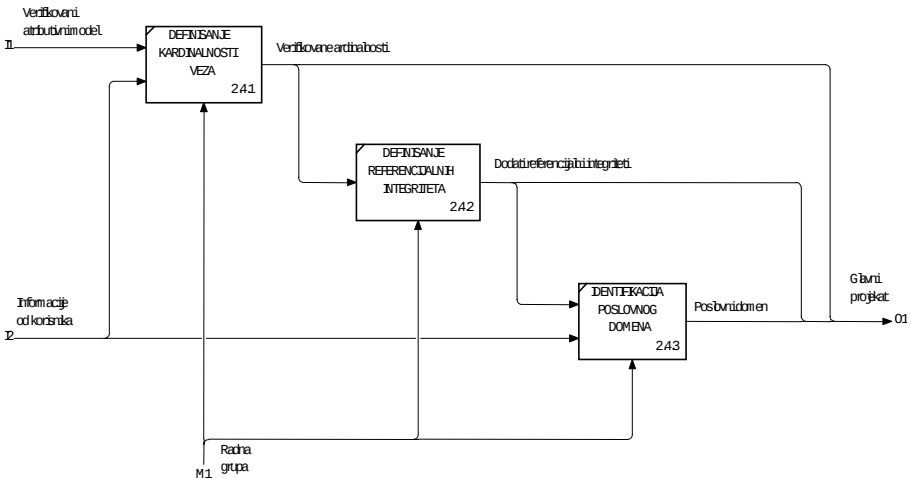
- set vrednosti,
- pravila dozvoljenih vrednosti,
- null vrednosti,
- dosledno značenje, tj. međusobno isključivanje (npr., Atribut: Pol; Vrednost: muškarac, žena).

Pravila za definisanje atributa biće definisana u sledećem poglavlju, kao tzv. poslovna pravila.

Aktivnost
2.4. Definisanje poslovnih pravila

Za aktivnost *"2.4. Definisanje poslovnih pravila"* potrebne su sledeće podaktivnosti (slika 3.49):

- Aktivnost 2.4.1. Definisanje kardinalnosti veza,
- Aktivnost 2.4.2. Definisanje referencijalnog integriteta i
- Aktivnost 2.4.3. Identifikacija poslovnog domena.



Slika 3.49. Dekompozicioni dijagram za aktivnost "2.4. Definisanje poslovnih pravila"

Definisanje poslovnih pravila vezano je za tzv. strukturna dinamička pravila integriteta koja se definišu uređenom trojkom:

<Ograničenje, Operacija , Akcija>

Strukturna dinamička pravila integriteta su:

- ograničenja kojima se definišu dozvoljena stanja baze podataka,
- operacije koje mogu potencijalno ugroziti ograničenja i
- akcije koje treba preduzeti ukoliko dođe do narušavanja ograničenja.

Ograničenja

Ograničenja se posmatraju preko:

- strukturnih ograničenja,
- ograničenja nad standardnim domenom,
- ograničenja nad vrednošću domena i

- ograničenja na kardinalnost.

Ograničenja su strukturna ukoliko su prikazana strukturom modela podataka, što se, pre svega, odnosi na:

- *integritet entiteta* - gde ne mogu da postoje dva primerka entiteta u istom tipu entiteta tako da imaju istu vrednost atributa koji čine identifikator, tj. ne postoje dva tipa entiteta koji imaju isti skup atributa kao identifikator;
- *referencijalni integritet*, gde se definišu:
  - ograničenje postojanja (egzistencijalna zavisnost) jednog entiteta u zavisnosti od drugog entiteta;
  - ograničenje mogućnosti identifikacije jednog objekta bez poznavanja identifikatora nekog drugog objekta;
  - specijalni tipovi veze kojima se definišu podtipovi egzistencijalno i identifikaciono, zavisno od nadređenog generalizovanog entiteta.

Ograničenja nad *standardnim domenom* definišu se, npr, kao:

- tip podatka (character, numeric, boolen),
- dužina podatka CHARACTER (30) i dr.

Ograničenja nad vrednošću domena (vrednost atributa) mogu se podeliti na:

- operatore poređenja (<,>, =, >=, <=);
- IN listu vrednosti koja formira listu konstanti iz odgovarajućeg domena, eksplicitnim navođenjem svih dozvoljenih vrednosti (npr., Stepen IN ['G,P,C']);
- BETWEEN opseg dozvoljenih vrednosti, gde atributi objekata i veza uzimaju vrednosti iz domena, ali uz postavljena ograničenja na ove vrednosti, tako da atribut može poprimiti samo uži skup vrednosti iz domena (npr., BETWEEN 10 AND 200);
- **NOT NULL** kada dato polje ne može da dobije nula vrednost, tj. mora uvek da ima neku vrednost.

Ograničenja na *kardinalnost* veza definišu se između:

- entiteta 'roditelj' i entiteta 'dete' i to kao:
  - kardinalnost tipa Zero, One or More, gde se jedan primerak entiteta 'roditelj' pridružuje nijednom, jednom ili većem broju primeraka entitetu 'dete';
  - kardinalnost tipa One or More (P), gde se jedan primerak entiteta 'roditelj' pridružuje najmanje jednom ili većem broju primeraka entiteta 'dete';
  - kardinalnost tipa Zero or One (Z), gde se jedan primerak entiteta 'roditelj' pridružuje nijednom ili jednom primerku entiteta 'dete';
  - kardinalnost tipa konkretne vrednosti (Exactly), gde se jedan primerak entiteta 'roditelj' pridružuje tačno definisanom broju primeraka entiteta 'dete'.
- entiteta 'dete' prema entitetu 'roditelj' kao:
  - TOTALNO učešće, gde svi primerci entiteta 'dete' učestvuju bar u jednoj vezi (No Nulls) sa entitetom 'roditelj';
  - PARCIJALNO (delimično) učešće, gde samo pojedini primerci entiteta 'dete' učestvuju u vezi (Nulls Allowed) sa entitetom 'roditelj'.

Operacije

Operacije koje potencijalno ugrožavaju ograničenja su standardne operacije ažuriranja, tzv. IRD operacije, što je skraćenica od:

- ubacivanje novog sloga (Insert),

- izmena sloga (Replace) i
- brisanje sloga (Delete).

Operacija *ubacivanje (insert)* omogućuje sledeća dodavanja podataka:

- kreira objekat i proverava da li je vrednost ključa objekta moguća ili već postoji objekat sa tom vrednošću;
- kreira vezu i proverava da li postoje objekti sa datim vrednostima ključa;
- dodaje vrednost objektu ili vezi i proverava da li je ta vrednost dozvoljena.

Operacija *izmena (replace)* omogućuje sledeće izmene podataka:

- izmenu vrednosti neključnog atributa objekta;
- izmenu vrednosti atributa koji je deo ključa, što znači da treba izmeniti tu vrednost u svim objektima i u svim vezama sa objektom, kao i izmeniti tu vrednost u svim slabim objektima u kojima je ta vrednost spuštena kao deo ključa;
- izmenu vrednosti neključnog atributa u vezi.

Operacija *brisanje (delete)* omogućuje sledeća brisanja podataka:

- brisanja objekata i veze u kojima se pojavljuje vrednost ključa objekta;
- brisanje veze u tipu veze;
- brisanje objekta 'roditelj' i svih objekata 'dete', čije postojanje zavisi od datog objekta.

Akcije

Za iskazivanje strukturnih pravila integriteta, tj. za iskazivanje potpune specifikacije buduće baze podataka, definišu se *akcije* koje treba preduzeti kada neka operacija ažuriranja baze podataka naruši definisano ograničenje.

Mogu se definisati sledeći tipovi akcija:

- RESTRICT (R), tj. akcija *odbijanja* operacije kojom se efekti te operacije poništavaju ako je uslov integriteta narušen;
- CASCADE (C), tj. akcija *prosleđivanja* operacije na vezni entitet;
- DEFAULT (D), tj. akcija kojom se kreira *specifično pojavljivanje* tzv."default objekta" koji označava "pretpostavljeni objekat" i zamenjuje objekat čije je nepostojanje uzrok narušavanja integriteta;
- SET NULL (SN), tj. akcija koja treba da eliminiše da primerak entiteta "visi" u sistemu, tj. atribut koji uspostavlja vezu setuje se na null vrednost. Specificira se "nul objekat" koji označava "još nepoznato pojavljivanje datog tipa objekta" i zamenjuje objekat čije je nepostojanje uzrok narušavanja integriteta;
- NONE, što znači da ne postoji ograničenje i da se operacija neometano izvodi.

Ovako nabrojana strukturna dinamička pravila integriteta biće u sledećim aktivnostima detaljno opisana.

Aktivnost
2.4.1. Definisanje kardinalnosti veza

S obzirom na to da su u prethodnom poglavlju navedeni tipovi veza, trebalo bi sada izneti njihove osobine. Naime, veze (relationships) imaju osobinu koja se zove *kardinalnost preslikavanja,* koja definiše:

- kardinalnost preslikavanja 'roditelj'-'dete' i
- kardinalnost preslikavanja 'dete'-'roditelj'.

Kardinalnost preslikavanja 'roditelj'-'dete' definiše "koliko mnogo" instanci entiteta 'roditelj' je povezano sa "koliko mnogo" instanci entiteta 'dete'. Po IDEF1X metodologiji, definišu se četiri načina definisanja kardinalnosti preslikavanja 'roditelj'-'dete':

- Zero , One or More - bez oznake
  - Svaki 'roditelj' povezuje se sa nula, jednom ili više instanci 'dete';
- One or More - označen slovom "P"
  - Svaki 'roditelj' povezuje se sa jednom instancom ili više instanci 'dete';
- Zero or One - označen slovom "Z"
  - Svaki 'roditelj' povezuje se sa nula ili jednom instancom 'dete';
- Tačno n - gde je n broj
  - Svaki 'roditelj' povezuje se sa tačno n instanci 'dete'.

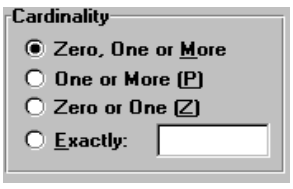
Kardinalnost preslikavanja 'dete'-'roditelj' definiše se kao:

- kardinalnost preslikavanja gde *je dozvoljena null vrednost* stranog ključa (Nulls Allowed), što je moguće samo za neidentifikujuće veze i obeležava se (sa strane 'roditelj') romboidom;
- kardinalnost preslikavanja gde *nije dozvoljena null vrednost* stranog ključa (No Nulls).

Na slici 3.50. prikazane su opcije prilikom definisanja kardinalnosti preslikavanja.

U daljem tekstu biće prikazane sve varijante kardinalnosti preslikavanja za sledeće tipove veza:

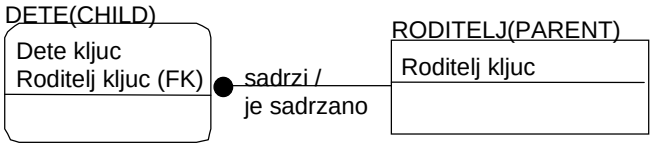
- identifikujuće veze,
- neidentifikujuća veza ,
- rekurzivna veza,
- veza kategorije,
- neodređujuća veza tipa više prema više i
- N-arne veze.

Veza 'roditelj'-'dete'	Veza 'dete'-'roditelj'
	

Slika 3.50. Opcije u ERwin-u kojima se definiše kardinalnost preslikavanja

Kardinalnost identifikujućih veza

Kod identifikujućih veza ključevi entiteta 'roditelj' učestvuju u identifikovanju entiteta 'dete', tj. svaka instanca entiteta 'dete' mora biti povezana sa najmanje jednom instancom entiteta 'roditelj'. *Identifikujuće veze* su prikazane punom linijom koja povezuje entitet 'roditelj' i entitet 'dete' sa tačkom na entitetu 'dete' (slika 3.51). Ovakav tip veze govori da je entitet 'dete' identifikujući, zavisno od entiteta 'roditelj'.

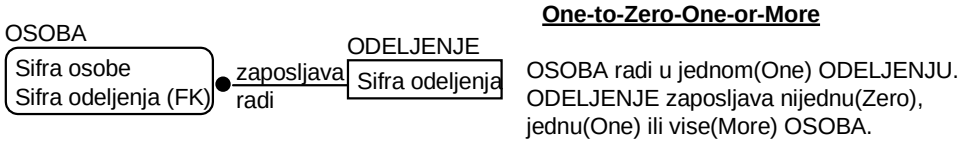


Slika 3.51. Kardinalnost preslikavanja identifikujuće veze

Sledeće varijante tipova kardinalnosti identifikujućih veza biće prikazane na primeru entiteta ODELJENJE ('roditelj') i entiteta OSOBA ('dete'), imajući u vidu opcije prikazne na slici 3.50. gde je fiksirana opcija 'dete'-'roditelj' na No Nulls (ONE).

Kardinalnost veze tipa One-to-Zero-One-or-More

Ako je svako ODELJENJE (PARENT) povezano sa nula, jednom ili više instanci OSOBA (CHILD), gde OSOBA zavisi od ODELJENJA, kardinalnost je prikazana na slici 3.52.

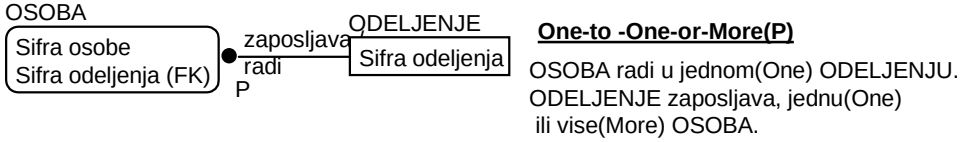


Slika 3.52. Kardinalnost veze tipa One-to-Zero-One-or-More

Ova veza govori da Odeljenje nema nijednu zaposlenu Osobu, ili ima jednu ili više zaposlenih Osoba. Kako je to identifikujuća veza, OSOBA zavisi od ODELJENJA, pa se OSOBA ne može identifikovati ukoliko identifikator entiteta ODELJENJA nije poznat. Dakle, entitet OSOBA egzistencijalno zavisi od entiteta ODELJENJE. *Ovaj tip identifikujuće veze se preporučuje za korišćenje, jer je najmanje ograničavajući.*

Kardinalnost veze tipa One-to-One-or-More (P)

Ako je svako ODELJENJE (PARENT) povezano sa jednom ili više instanci OSOBA (CHILD), gde OSOBA zavisi od ODELJENJA, kardinalnost je prikazana na slici 3.53.



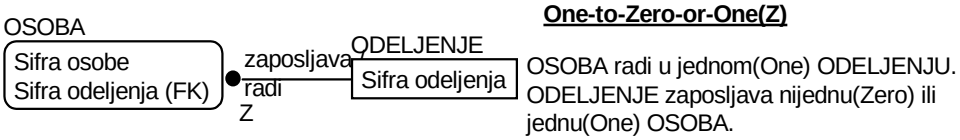
Slika 3.53. Kardinalnost veze tipa One-to-One-or-More (P)

Ova veza zahteva da Odeljenje *mora* (One-P) da zapošljava bar jednu Osobu, a može i više (More). Kako je to identifikujuća veza, entitet OSOBA je zavisan od entiteta ODELJENJE, pa se OSOBA *ne može* identifikovati ukoliko identifikator entiteta ODELJENJE nije poznat. Dakle, OSOBA zavisi od entiteta ODELJENJE (isključena je opcija Nulls).

Kardinalnost veze tipa One-to-Zero-or-One (Z)

Ako je svako ODELJENJE (PARENT) povezano sa nula ili jednom instancom OSOBA (CHILD),

gde OSOBA zavisi od ODELJENJA, kardinalnost je prikazana na slici 3.54.

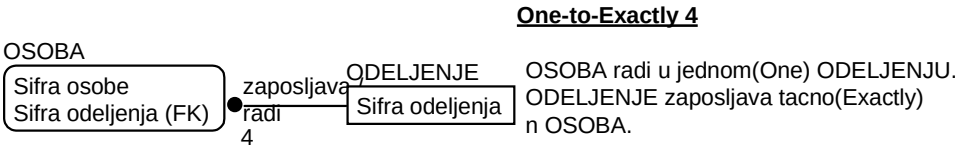


Slika 3.54. Kardinalnost veze tipa One-to-Zero-or-One (Z)

Ova veza govori da Odeljenje ne mora (Zero) da zapošljava nijednu osobu ili zapošljava samo jednu Osobu (One). Kako je to identifikujuća veza, entitet OSOBA je zavisan od entiteta ODELJENJE, pa se OSOBA *ne može* identifikovati ukoliko identifikator entiteta ODELJENJE nije poznat. Dakle, entitet OSOBA je zavisan od entiteta ODELJENJE (isključena je opcija Nulls).

Kardinalnost veze tipa One-to-Exactly n

Ako je svako ODELJENJE (PARENT) povezano sa tačno n instanci OSOBA (CHILD), npr.4, gde OSOBA zavisi od ODELJENJA, kardinalnost je prikazana na slici 3.55.



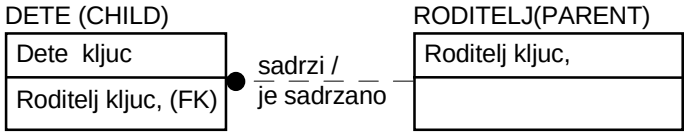
Slika 3.55. Kardinalnost veze tipa One-to-Exactly n

Ova veza omogućuje da osoba radi samo u jednom Odeljenju (One), dok u obrnutoj vezi odeljenje zapošljava tačno n osoba. Kako je to identifikujuća veza, entite OSOBA je zavisan od entiteta ODELJENJE, pa se OSOBA *ne može* identifikovati ukoliko identifikator entiteta ODELJENJE nije poznat. Dakle, entitet OSOBA je zavisan od entiteta ODELJENJE (isključena je opcija Nulls).

Kardinalnost neidentifikujućih veza

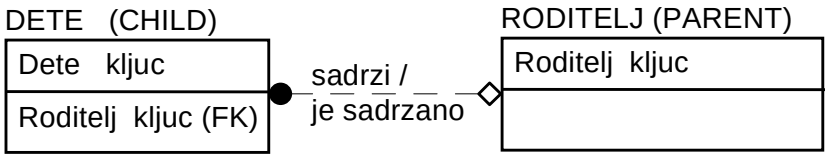
Veza je neidentifikujuća ako se može identifikovati instanca entiteta 'dete' bez znanja veze sa entitetom 'roditelj'. Grafički se ova veza predstavlja isprekidanom linijom. U zavisnosti od kardinalnosti 'dete'-'roditelj', (slika 3.50.), ovaj tip kardinalnosti može ili ne može imati egzistencijalnu zavisnost.

Egzistencijalna zavisnost instance entiteta 'dete' od instance entiteta 'roditelj' vezana je za obaveznost veze (No Nulls) i grafički se definiše na slici 3.56.



Slika 3.56. Obaveznost veze

Ako je veza neobavezna (Nulls Allowed), tada 'dete' nije egzistencijalno zavisno, ali poštuje tu vezu i grafički se predstavlja rombom (slika 3.57).

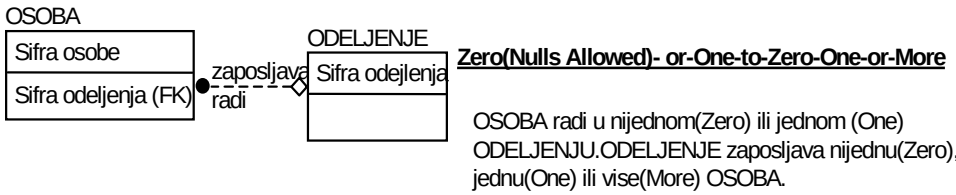


Slika 3.57. Neobavezne veze

Sledeće varijante tipova kardinalnosti neidentifikujućih veza biće razmotrene na primeru entiteta ODELJENJE ('roditelj') i entiteta OSOBA ('dete').

Kardinalnost veze tipa Zero(Null Allowed)-or-One-to-Zero-One-or-More

Ako je svako ODELJENJE (PARENT) povezano sa nula, jednom ili više instanci OSOBA (CHILD), gde OSOBA može, a ne mora da pripada ODELJENJU, kardinalnost je prikazana na slici 3.58.

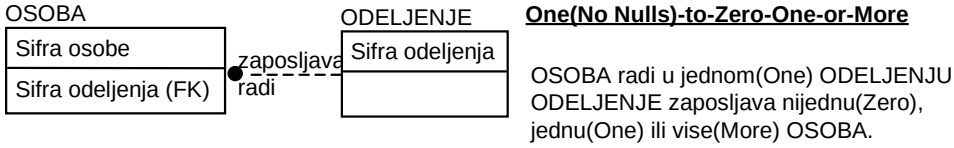


Slika 3.58. Kardinalnost veze tipa Zero(Null Allowed)-or-One-to-Zero-One-or-More

Ova veza dozvoljava da Osoba ne radi u nijednom (Nulls Allowed) Odeljenju ili da radi u jednom Odeljenju. U Odeljenje može biti neraspoređena (veza zapošljava) nejedna (Zero), ili raspoređena jedna (One) ili više (More) Osoba. Ova veza je najmanje ograničavajuća, jer omogućuje definisanje 'roditelj' (ODELJENJE) i 'dete' (OSOBA) bez ograničenja i međusobnih zavisnosti.

Kardinalnost veze tipa One(No Nulls)-to-Zero-One-or-More

Ako je svako ODELJENJE (PARENT) povezano sa nula, jednom ili više instanci OSOBA (CHILD), gde OSOBA mora da pripada ODELJENJU, kardinalnost je prikazana na slici 3.59.

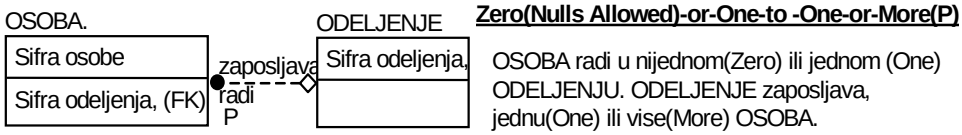


Slika 3.59. Kardinalnost veze tipa One(No Nulls)-to-Zero-One-or-More

Ova veza dozvoljava da Osoba mora da radi u jednom (No Nulls) i samo jednom Odeljenju. U Odeljenje može biti raspoređena (veza zapošljava) nejedna (Zero), jedna (One) ili više (More) Osoba.

Kardinalnost veze tipa Zero(Nulls Allowed)-or-One-to-One-or-More(P)

Ako je svako ODELJENJE (PARENT) povezano sa jednom ili više instanci OSOBA (CHILD), gde OSOBA može, a ne mora da zavisi od ODELJENJA, kardinalnost je prikazana na slici 3.60.

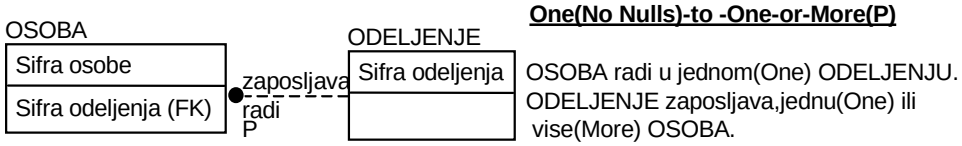


Slika 3.60. Kardinalnost veze tipa Zero(Nulls Allowed)-or-One-to-One-or-More(P)

Ova veza dozvoljava da Osoba može da ne radi u nijednom (Nulls Allowed) Odeljenju ili da radi u jednom Odeljenju. U Odeljenje *mora* biti raspoređena (veza zapošljava) najmanje jedna (One-P) ili više (More) Osoba. Ova veza je ograničavajuća, jer zahteva da u Odeljenju mora biti najmanje jedna Osoba, dok definisanje podataka o Osobama ne zavisi od Odeljenja u kome će biti.

Kardinalnost veze tipa One(No Nulls)-to-One-or-More(P)

Ako je svako ODELJENJE (PARENT) povezano sa jednom ili više instanci OSOBA (CHILD), gde OSOBA mora da pripada ODELJENJU, kardinalnost je prikazana na slici 3.61.

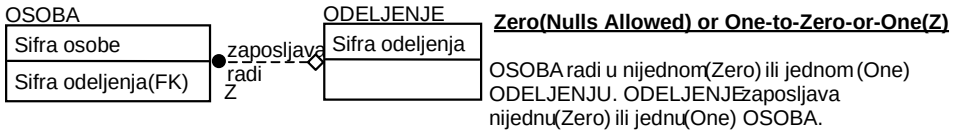


Slika 3.61. Kardinalnost veze tipa One(No Nulls)-to-One-or-More(P)

Ova veza zahteva da Osoba mora da radi u jednom (No Nulls) i samo jednom Odeljenju. U Odeljenje mora biti raspoređena (veza zapošljava) najmanje jedna (One-P) ili više (More) Osoba. Ova veza je ograničavajuća jer zahteva da u odeljenju mora biti najmanje jedna Osoba i da ta Osoba radi isključivo u jednom odeljenju. Ova veza zahteva programsko rešenje vezano za brisanje poslednje Osobe.

Kardinalnost veze tipa Zero(Nulls Allowed)-or-One-to-Zero-or-One(Z)

Ako je svako ODELJENJE (PARENT) povezano sa nula ili jednom instancom OSOBA (CHILD), gde OSOBA može, a ne mora da zavisi od ODELJENJA, kardinalnost je prikazana na slici 3.62.

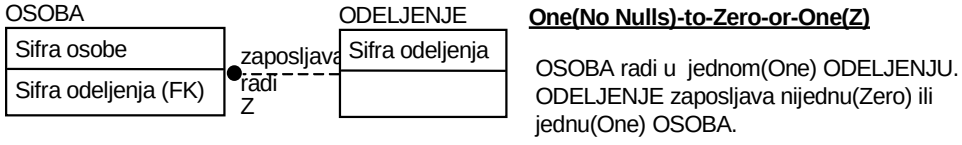


Slika 3.62. Kardinalnost veze tipa Zero(Nulls Allowed)-or-One-to-Zero-or-One(Z)

Ova veza dozvoljava da Osoba može da radi u nijednom (Nulls Allowed) ili jednom Odeljenju. U Odeljene može biti raspoređena (veza zapošljava) nijedna (Zero) ili jedna (One) Osoba.

Kardinalnost veze tipa One(No Nulls)-to-Zero-or-One(Z)

Ako je svako ODELJENJE (PARENT) povezano sa nula ili jednom instancom OSOBA (CHILD), gde OSOBA mora da pripada ODELJENJU, kardinalnost je prikazana na slici 3.63.

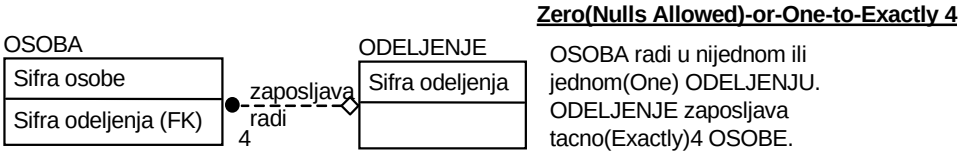


Slika 3.63. Kardinalnost veze tipa One(No Nulls)-to-Zero-or-One(Z)

Ova veza dozvoljava da Osoba mora da radi u jednom (No Nulls) i samo jednom Odeljenju. U Odeljenje može biti raspoređena (veza zapošljava) nijedna (Zero) ili jedna (One) Osoba.

Kardinalnost veze tipa Zero(Nulls Allowed)-or-One-to-Exactly n

Ako je svako ODELJENJE (PARENT) povezano sa tačno n (4) instanci OSOBA (CHILD), gde OSOBA može, a ne mora da zavisi od ODELJENJA, kardinalnost je prikazana na slici 3.64.

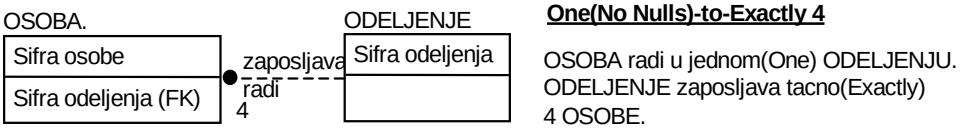


Slika 3.64. Kardinalnost veze tipa Zero(Nulls Allowed)-or-One-to-Exactly n

Ova veza dozvoljava da Osoba može da radi u nijednom (Nuls Allowed) ili jednom Odeljenju. U Odeljenje može biti raspoređeno (veza zapošljava) tačno n Osoba.

Kardinalnost veze tipa One(No Nulls)-to-Exactly n

Ako je svako ODELJENJE (PARENT) povezano sa tačno n (4) instanci OSOBA (CHILD), gde OSOBA mora da pripada ODELJENJU, kardinalnost je prikazana na slici 3.65.



Slika 3.65. Kardinalnost veze tipa - One (No Nulls) to Exactly n

Ova veza dozvoljava da Osoba mora da radi u Odeljenju. U Odeljenje može biti raspoređeno (veza zapošljava) tačno n Osoba.

Kardinalnost rekurzivnih veza

Rekurzivne veze se definišu kao:

- rekurzija nad jednom tabelom i
- rekurzija nad dve tabele.

Rekurzija nad jednom tabelom

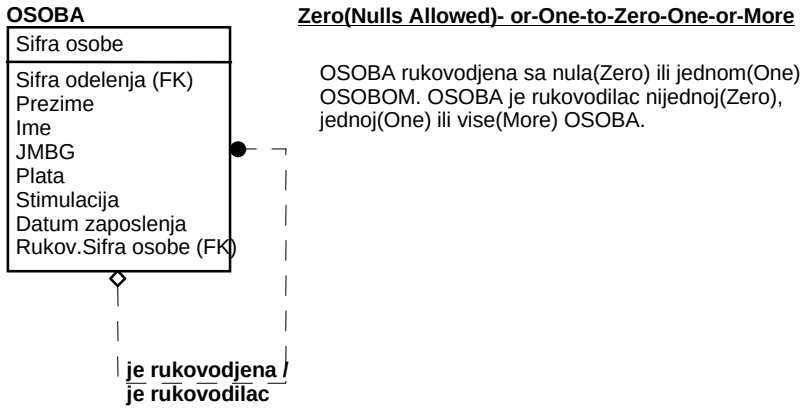
Rekurzija nad jednom tabelom ili rekurzija *hijerarhijskog* tipa definisana je za entitet koji je ujedno i 'roditelj' i 'dete'. Često se naziva i "udica" (*fish hook*). Hijerarhijska rekurzija omogućuje da 'roditelj' može imati više dece, ali deca mogu imati samo jednog 'roditelj'. Sve rekurzivne veze hijerarhijskog tipa moraju biti slabe veze. Kao i sve slabe veze, one ubacuju ključ entiteta 'roditelj' u prostor podataka entiteta 'dete'.

Na slici 3.66. prikazan je primer rekurzije nad entitetom OSOBA. Za atribut 'Sifra osobe' definiše se novi atribut (kojim se definiše rukovodilac) čije je ime "Rukov.Sifra osobe (FK)". Atribut "Rukov" je novo ime (*Rolename*) za atribut 'Sifra osobe'. Razlog za definisanje atributa "Rukov" je u tome što se u istom entitetu pod istim imenom ne može dva puta pojaviti atribut "Sifra osobe". To bi bila greška u normalizaciji, jer je jedan od osnovnih pravila normalizacije da ako dve stvari imaju isto ime, onda su one ista stvar.

Uloga (*Rolename*) nešto je više od novog imena za atribut, jer nosi sa sobom i zaštitu, odnosno identitet instance entiteta.

Za primer prikazan na slici 3.66. definiše se rekurzivna (ne identifikujuća) veza, gde

preneseni ključevi mogu biti NULL podatak.

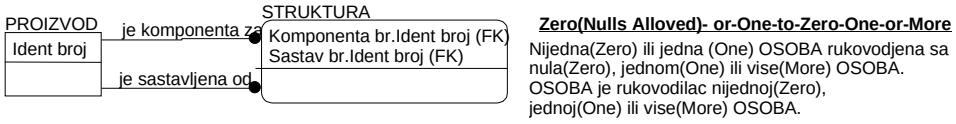


Slika 3.66. Rekurzivna veza hijerarhijskog tipa

Rekurzija nad dve tabele

Rekurzija nad dve tabele ili rekurzija mrežnog tipa (još se zove i dvotabelarna rekurzija) predstavlja vezu 'roditelj'-'dete', gde 'roditelj' može imati veći broj dece i 'dete' može imati veći broj roditelja. To je specijalni slučaj 'Many-to-Many' veze nad samim sobom i treba je razbiti na dve veze 1:M. Stoga i u tom slučaju treba definisati *Rolename* da bi se predstavilo prenošenje ključeva. U tom slučaju definišu se identifikujuće veze i atribut ključevi sa NOT NULL vrednostima.

Na slici 3.67. prikazana je tipična rekurzija mrežnog tipa, gde se rekurzija definiše nad dva entiteta PROIZVOD i STRUKTURA. Entitet PROIZVOD sadrži instance vezane za proizvode, sklopove, podsklopove, delove, materijale i dr. Entitet STRUKTURA povezuje proizvod sa odgovarajućim podređenim sklopovima, a te sklopove sa podređenim podsklopovima i tako redom do delova i materijala. To je osnova za definisanje tzv. sastavnica proizvoda.



Slika 3.67. Rekurzivna veza mrežnog tipa

Veza kategorije

Veza kategorije uspostavlja vezu između nadređenih entiteta i njegovih zavisnih, klasnih entiteta. Preko ove veze entitet koji se specijalizuje u klase prosleđuje primarni ključ klasnim zavisnim entitetima. Ova veza se definiše postupkom generalizacije, tj. postupkom apstrakcije podataka, u kome se *skup sličnih tipova* entiteta predstavlja NADTIPOM, tj. generičkim (generalizovanim) entitetom (IDEF1X metodologija).

Pod sličnim tipovima objekata podrazumevaju se oni koji imaju JEDAN broj ISTIH (zajedničkih) atributa, tipova veza i/ili operacija sa drugim objektima.

Kardinalnost veze kategorije od generičkog oblika ka entitetu kategorije je tipa "One-to--zero-or-one" i sadrži implicitni izraz '*is a',* a čita se na sledeći način:

"Svaka (One) instanca generičkog entiteta može, a ne mora (zero-or-one) da ima instancu kategorije, dok svaka instanca entiteta kategorije je (is a) instanca generičkog oblika."

Dakle, objekti su hijerarhijski GENERALIZOVANI ako se skup srodnih objekata posmatra kao jedan objekat. Svi atributi sa nižeg nivoa, koji su zajednički, prenose se na viši nivo, dok se

na nižem nivou definišu entiteti kategorije koji sadrže atribute koji su svojstveni samo tom nivou.

U generalizacionoj hijerarhiji objekata (entiteta) važi pravilo nasleđivanja osobina i nasleđivanja operacija nadtipa.

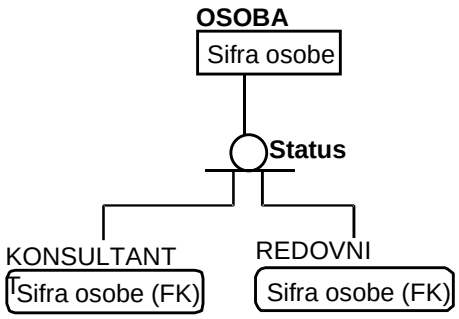
IDEF1X metodologija simbolički obezbeđuje razlikovanje dva tipa entiteta kategorije, tj. nepotpune i potpune strukture.

Nepotpune strukture

Nepotpuna struktura se definiše kada nije sigurno da su identifikovani svi oblici entiteta kategorije (jednostruka linija na dnu simbola klase naznačuje da mogu biti uključene druge kategorije). Za ovakvu vezu se definiše atribut diskriminator u entitetu koji se specijalizira i koji obezbeđuje ekskluzivnost veze. Ako se diskriminator ne definiše, veza se može smatrati neekskluzivnom.

Na primeru prikazanom na slici 3.68. generalizovani entitet OSOBA može imati dva pojavna oblika: KONSULTANT ili REDOVNI. Diskriminator "Vrsta" omogućuje definisanje oba tipa angažovanja, ali ne ograničava da može biti više podtipova.

Diskriminator 'kategorije' je atribut koji pokazuje kako razlikovati entitete jedne kategorije od entiteta druge kategorije. U navedenom primeru atribut 'vrsta' je diskriminator kategorije i definisan je u entitetu OSOBA.



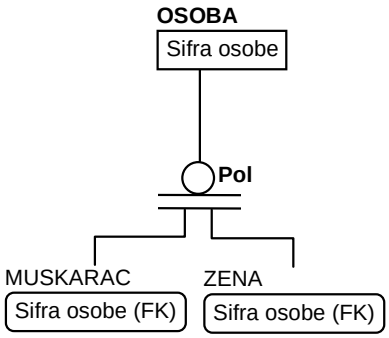
Slika 3.68. Nepotpuna struktura

U IDEF1X formatu nepotpuna struktura se definiše pod nazivom "incomplete category cluster" i ima osobinu da primerak generalizovanog (generičkog) entiteta može postojati bez asocijacije sa primerkom bilo kog specijalizovanog (kategorisanog) entiteta, tj. može se definisati primerak entiteta OSOBA bez obaveze definisanja primeraka entiteta KONSULTANT ili REDOVNI.

Potpune strukture

Potpuna struktura se definiše za slučajeve kada je sigurno da nema višeklasnih entiteta u koje bi se specijalizovao određeni entitet (dvostruka linija na dnu simbola klase naznačuje da ne mogu biti uključene druge kategorije). U literaturi se još zove ekskluzivna specijalizacija.

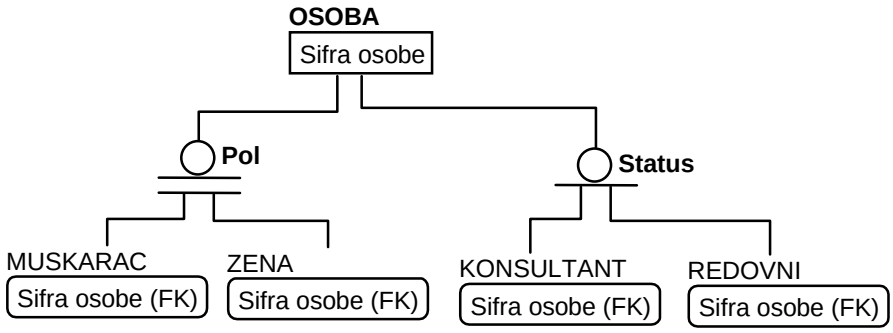
Na slici 3.69. prikazan je primer potpune strukture gde generički entitet OSOBA ima entitete kategorije MUSKO ili ZENSKO. U IDEF1X formatu potpuna struktura se definiše pod nazivom "complete category cluster" i ima osobinu da primerak generalizovanog (generičkog) entiteta ne može postojati bez asocijacije sa primerkom specijalizovanog (kategorisanog) entiteta. Diskriminator 'Pol' definisan u entitetu OSOBA, *isključuje* pojavljivanje treće varijante.



Slika 3.69. Potpuna struktura

Korišćenje I (And) ILI (Or) struktura

Prikazane nepotpune i potpune strukture su primeri, ILI struktura, gde se definiše, kao na slici 3.68, generički entitet OSOBA, koji ima entitete kategorije KONSULTANT ili REDOVNI ili da je, kao na slici 3.69, OSOBA ili MUSKARAC ili ZENA. Struktura I (And) dobija se kada se definišu dva ili više diskriminatora, kategorije za generički entitet. U datom primeru, na slici 3.70, generički entitet OSOBA može imati četiri kombinacije, od kojih je jedna da jedna osoba istovremeno može biti i MUSKARAC I KONSULTANT.



Slika 3.70. Kombinacija potpune i nepotpune strukture

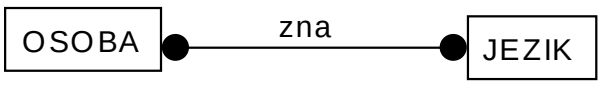
Neodređujuća veza tipa više prema više

Neodređujuća veza tipa 'više prema više' je tip nespecificirane veze koja povezuje dva entiteta i u kojoj se primerak jednog entiteta vezuje sa nula, jednim ili više primeraka drugog entiteta, a svaki primerak drugog entiteta se povezuje sa nula, jednim ili više primeraka prvog entiteta.

Veze 'više prema više' su nacrtane sa tačkama na oba kraja. Prema primeru sa slike 3.71. veze 'više prema više', OSOBA zna više JEZIKA i svaki JEZIK poznaje više OSOBA. Veze 'više prema više' se, obično, koriste u preliminarnoj fazi razvoja entitetnog dijagrama.

Ovde ništa nije pogrešno, samo u slučaju da se jednostavno pokušava reći da OSOBA "zna" više JEZIKA, kao i da JEZIK "poznaje" više OSOBA. 'Many-to-Many' veza (puna linija sa tačkama na oba kraja) zove se nespecifična veza.

Osnovno pravilo koje se ovde mora poštovati je da se mora izvesti prevođenje 'Many-to- -Many' veze u 'Zero-to-Many' vezu.



Slika 3.71. Veza Many-to-Many

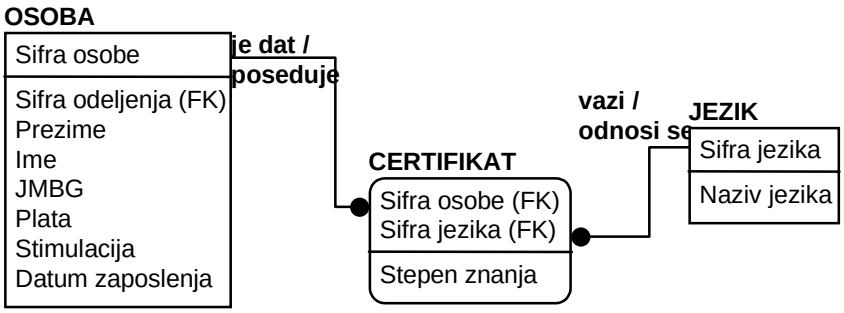
Ovaj problem se rešava tako što se dodaje asocijativni entitet nazvan CERTIFIKAT (slika 3.72) i na taj način prevede 'Many-to-Many' veza u par One-to-Zero-One-or-Many veze. Kao što je prikazano na slici 3.72, ključ entiteta OSOBA ("Sifra osobe") prelazi u asocijativni entitet, a, isto tako, i ključ entiteta JEZIK ("Sifra jezika") prelazi u asocijativni entitet tako da je asocijativni entitet CERTIFIKAT egzistenciono - identifikaciono zavisan od entiteta JEZIK i OSOBA: egzistencijalno jer fizički ne može da postoji, a identifikaciono jer sadrži primarne ključeve entiteta JEZIK i OSOBA.

Definisanjem entiteta CERTIFIKAT izvršena je tzv. agregacija. *Agregacija* je apstrakcija u kojoj se skup tipova objekata i njihovih veza tretira kao jedinstveni AGREGIRANI TIP objekta ili asocijativni tip entiteta. U literaturi se ovaj tip entiteta još zove 'mešoviti tip objekta veze'Š2Ć.

Kardinalnost preslikavanja, od komponente (OSOBA) ka agregaciji (CERTIFIKAT) identifikujuća je veza (ključ 'roditelj' je sastavni deo ključa 'dete'), tj. Zero-One-or-Many, a inverzno preslikavanje od 'dete' (CERTIFIKAT) prema ''roditelj'u' (OSOBA) - No Nulls (One).

Identifikator agregiranog entiteta, po pravilu, nema svoj identifikator, već ga identifikuju objekti koje on agregira (kao što je prikazano na slici 3.72).

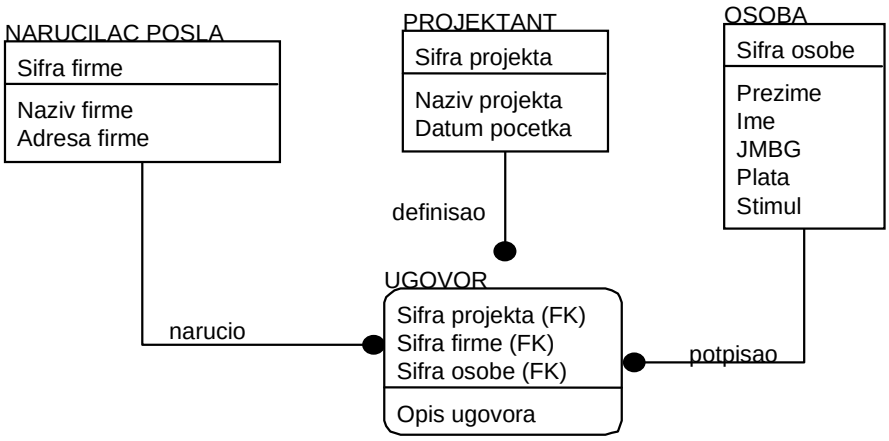
Agregirani entitet se u modelu podataka tretira kao i kod drugih entiteta, tj. on može da ima svoje atribute i/ili da bude u vezi sa nekim drugim entitetima (moguće agregiranim, takođe), ili da ima svoje podtipove i slično.



Slika 3.72. Prikaz asocijativne veze za entitet CERTIFIKAT

Kardinalnost N-arne veze

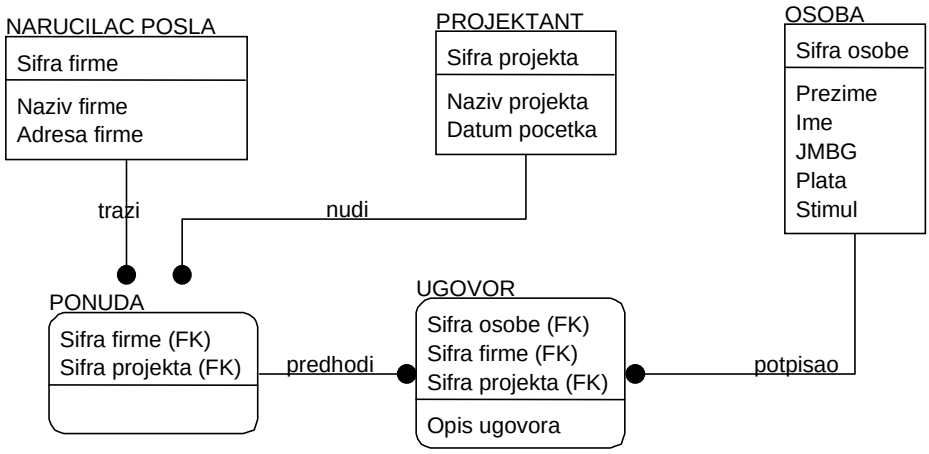
Postavljanje veze između tri ili više entiteta se izvodi na način koji je sličan asocijativnom entitetu. Na primer, treba posmatrati postupak poslovnog odlučivanja koji je prikazan na slici 3.73.



Slika 3.73. N-arna veza za primer entiteta UGOVOR

Entitet UGOVOR ovde predstavlja trostruku vezu od entiteta NARUCILAC POSLA, PROJEKAT i ZAPOSLEN. Struktura sa slike 3.73. indicira da više NARUCILACA POSLA naručuje više PROJEKATA u kojima učestvuje više OSOBA.

Međutim, nameće se pitanje: *"kako ponuditi projekat pre nego što je ugovoren"*(slika 3.74.).



Slika 3.74. Zamena trostuke veze sa dve dvostruke

Da bi se dobio odgovor na postavljeno pitanje, treba "trostruku" vezu zameniti sa dve "dvostruke". Odgovarajući na razne vrste ovakvih pitanja, N-arne veze se pretvaraju u seriju "dvostrukih" veza. Retkost je videti prave trostruke veze u IDEF1X modelu. One su čak mnogo ređe nego "četvorostruka", "petostruka"...

ERwin CASE alat je zasnovan na IDEF1X metodologiji koja, za razliku od drugih jezika modeliranja, insistira da sve veze (relationships) budu binarne (da spajaju tačno dva entiteta). To zaista ne znači da su nepotrebne 3,4 ili "n" veza (relationships), samo u slučaju da su ove situacije adekvatno adresirane.

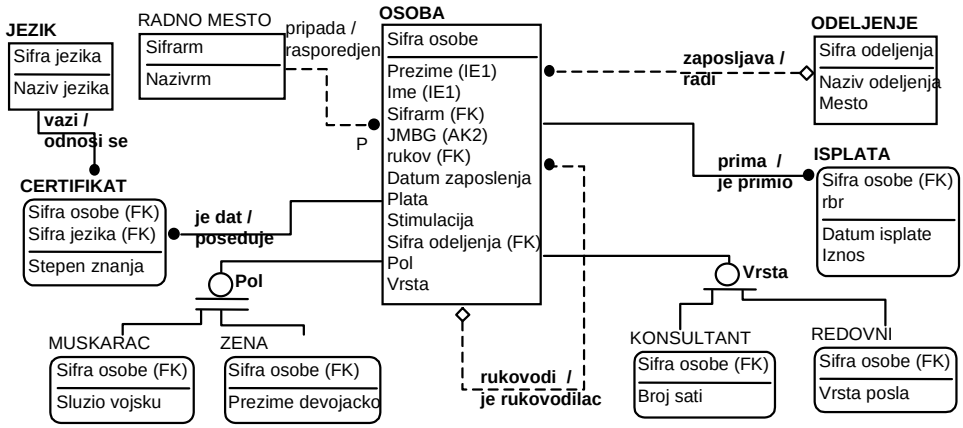
Imajući sve to u vidu, za primer dokumenta "Karton isplata" u sledećem koraku biće definisan atributivni model.

Puni atributivni model za dokument "Karton isplata"

Na tom nivou, a za dosadašnji primer vezan za "Karton isplata" biće prikazan na slici 3.75. atributivni model za ER dijagram koji sadrži definisane atribute.

Prikazani atributivni model se može opisati na sledeći način:

Osnovni entitet je OSOBA i poseduje, kao primarni ključ, atribut " Sifra osobe" i set opisnih atributa.



Slika 3.75. Puni atributivni model ze dokument "Karton isplata"

Čisto opisni atributi u entitetu OSOBA su: "Plata", "Stimul" i "Datum zaposlenja". Atribut "JMBG (AK1)" alternativni je ključ i po njemu se može izvršiti sekundarno indeksiranje. Atributi: "Prezime (IE1)" i "Ime (IE2)" atributi su tipa "Inverse Entry" i oni se koriste za indeksiranje, s tim što indeks nije jednoznačan, tj. vrednosti ključa se mogu ponavljati.

Atribut "Sifra odeljenja (FK)" preneseni je ključ od entiteta ODELJENJE i dozvoljava unos Null podataka, tj. ne mora se odmah definisati ODELJENJE gde OSOBA radi. Atribut: "Sifrarm (FK)" preneseni je ključ entiteta RADNO MESTO i ne dozvoljava unos Null podataka, tj. mora se obavezno uneti radno mesto Osobe.

Atribut: "Pol" diskriminator je kategorije MUSKARAC ili ZENA i automatski zahteva izbor jedne od opcija. Atribut "Vrsta" je diskriminator kategorija entiteta: KONSULTANT i REDOVNI i ne zahteva odmah odluku o vrsti, tj. naknadno se može definisati da li je osoba konsultant ili redovni.

Entitet ODELJENJE, kojim se definiše istoimeni šifarnik, treba da omogući da ne postoji nijedna Osoba zaposlena u njemu (Zero-One-or-Many). Entitet RADNO MESTO zahteva da makar jedna Osoba ima odgovarajuće radno mesto (One-or-Many).

Na slici 3.75. definisan je i asocijativni entitet CERTIFIKAT koji je "povukao" ključeve entiteta OSOBA i entiteta JEZIK i ima složeni ključ (Sifra osobe+Sifra jezika) i opisni atribut: "Stepen znanja", kojim se definiše stepen znanja jezika.

Entitet ISPLATA je primer slabog entiteta, čiji je složeni ključ povukao ključ jakog (OSOBA) entiteta "Sifra osobe (FK)" i atribut "rbr" kojim se definiše redni broj isplate. Opisni atributi entiteta ISPLATA su: "Datum isplate" i "Iznos".

Entiteti kategorije povukli su: primarni ključ "Sifra osobe" iz entiteta OSOBA i imaju svoje opisne atribute. Tako, entitet MUSKARAC ima opisni atribut: "Sluzio vojsku", a entitet ZENA ima opisni atribut: "Prezime devojacko". Entitet KONSULTANT ima opisni atribut: "Broj sati"

i entitet REDOVNI opisni atribut "Vrsta posla".

Na osnovu prethodno izvedenih aktivnosti (slika 3.49) u sledećem koraku biće definisan referencijalni integritet.

Aktivnost 2.4.2. Definisanje referencijalnog integriteta

Uprošćeno rečeno, referencijalni integritet obezbeđuje korektno povezivanje objekata, jer objekat koji nije predstavljen u odgovarajućem skupu objekata ne može da učestvuje u nekoj od veza predstavljenih u modelu podataka.

Referencijalni integritet je vezan za postojanje prenesenog ključa za neki entitet (npr. "Sifra odeljenja" u entitetu OSOBA). Entitet gde se nalazi preneseni ključ zove se 'dete' (CHILD), a entitet gde se definiše primarni ključ je 'roditelj' (PARENT). Primarni ključ se može preneti i postati preneseni ključ u okviru identifikujuće ili neidentifikujuće veze.

Preneseni ključ u identifikujućoj vezi je sastavni deo primarnog ključa entiteta 'dete', pa se 'dete' identifikuje preko 'roditelj', čime učestvuje u definisanju integriteta entiteta. Kod neidentifikujuće veze preneseni ključevi nisu deo primarnog ključa ''dete'ta', pa se 'dete' ne može identifikovati preko 'roditelj'. Ova razlika je veoma važna kada treba podržati vezu između 'roditelj' i ''dete'ta' kod operacija: ubacivanje (insert), brisanje (delete) i ažuriranje (update), što čini suštinu referencijalnog integriteta.

Na osnovu toga mogu se definicije integriteta entiteta i referencijalnog integriteta dopuniti sledećim tvrdnjama.

Integritetom entiteta se onemogućuje pojava da se mogu uneti dva entiteta sa istom vrednošću primarnog ključa ili da je ključ NULL podatak.

Referencijalni integritet entiteta zahteva da unesena vrednost atributa odgovara vrednosti atributa koji je primarni ključ drugog entiteta. Referencijalni integritet opisuje ponašanje modela kada, usled operacija održavanja, dolazi do narušavanja kardinalnosti veza. To ponašanje modela ili *strukturna pravila integriteta (SPI)* definišu se kao strukturna ograničenja.

Referencijalni integritet se definiše za svaku vezu, posebno za stranu entiteta 'roditelj', a posebno za stranu entiteta 'dete', i to za operacije: *insert (ubacivanje), delete (brisanje) i update (ažuriranje).*

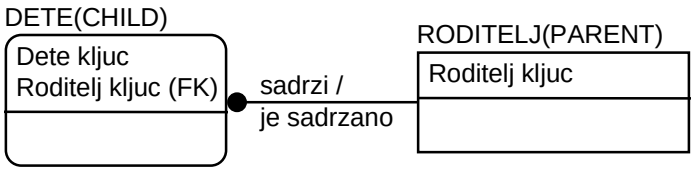
U daljem tekstu biće prikazane sve varijante referencijalnog integriteta za sledeće tipove veza:

- identifikujuće veze,
- neidentifikujuća veza,
- rekurzivne veze,
- veza kategorije,
- neodređujuća veza tipa više prema više i
- N-arne veze.

Referencijalni integritet za identifikujuće veze

Kad je u pitanju identifikujuća veza između entiteta 'roditelj' i 'dete', tada primarni ključ entiteta 'roditelj' postaje deo primarnog ključa entiteta 'dete'. Prethodno opisana pravila kardinalnosti govore da za svaku instancu 'dete' postoji jedna instanca 'roditelj'. To je veza

koja specificira egzistencijalnu i identifikacionu zavisnost između entiteta 'dete' od 'roditelj'.



Slika 3.76. Kardinalnost preslikavanja identifikujuće veze

Radi daljeg razumevanja potrebno je razmotriti detaljno pravila za operacije: **delete (**brisanje), **insert (**ubacivanje) i **update (**ažuriranje).

Pravila brisanja za identifikujuće veze

Pravila brisanja se mogu izvesti korišćenjem dveju akcija:

- prvo, može se obrisati svaka instanca ''dete'ta' čiji je nasleđeni ključ obrisana instanca 'roditelj' (akcija CASCADE), ili
- drugo, alternativno, može se sprečiti brisanje 'roditelj' ako postoji bilo koje 'dete' čiji bi deo primarnog ključa bio nekompletan (akcija RESTRICT).

U okviru ERwin CASE alata postavljene su difoltne vrednosti za brisanje entiteta 'dete'- -NONE, dok je difoltna vrednost za brisanje 'roditelj'-RESTRICT.

CASCADE omogućuje da sve instance ''dete'ta' obuhvaćene brisanjem instance 'roditelj' budu obrisane, što je definisano rečenicom PARENT DELETE CASCADE (D:C).

RESTRICT definiše ograničenje tako da instanca 'roditelj' ne može biti obrisana dok sve instance ''dete'ta' koje imaju nasleđeni ključ ne budu prethodno obrisane. Ako postoji neki nasleđeni, brisanje je zabranjeno.

Pravila ubacivanja (Insert) i ažuriranja (Update) za identifikujuće veze

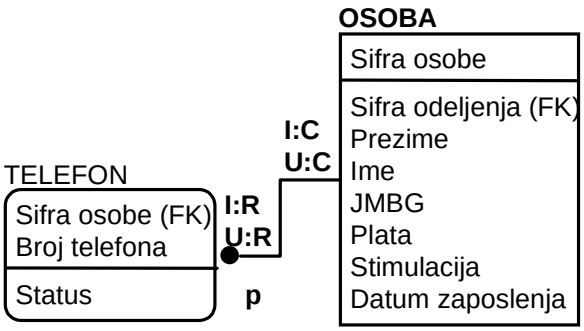
Pravila za ubacivanje (INSERT) i ažuriranje (UPDATE) upravljaju šta će biti kada je primerak dodat ili ažuriran. Ažuriranje je, u suštini, ubacivanje, ali sa nekim dodatnim pravilima.

Za operacije **ubacivanje (INSERT) i izmena (UPDATE)**, red može biti dodat ili izmenjen samo ako svi referencirani preneseni ključevi odgovaraju postojećim redovima u referenciranim tabelama.

Pravila ubacivanja/ažuriranja mogu se izvesti korišćenjem dveju akcija:

- prvo, ne može se ubaciti/ažurirati instanca ''dete'ta' bez instance 'roditelj' (akcija RESTRICT), ili
- drugo, alternativno, za ubačenog/ažuriranog 'roditelj' treba ubaciti/ažurirati i bilo koje 'dete', čiji bi deo primarnog ključa bio ključ 'roditelj' (akcija CASCADE).

U okviru ERwin CASE alata difoltna vrednost za ubacivanje/ažuriranje entiteta 'dete' je RESTRICT, dok je difoltna vrednost za ubacivanje 'roditelj'-NONE, a za ažuriranje RESTRICT.



Slika 3.77. Primer identifikujuće veze

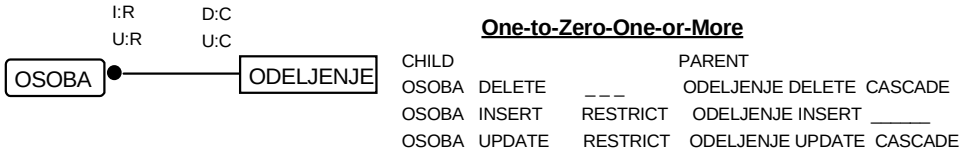
Na primeru veze OSOBA/TELEFON to izgleda ovako.

Složeni ključ entitetaTELEFON sastoji se od ključa odgovarajućeg entiteta OSOBA (Sifra osobe) i sopstvenog ključa Broj telefona. Zbog toga, se ne može ubaciti (I:R) ili izmeniti (U:R) TELEFON za koji ne postoji OSOBA (oznaka P). Obrnuto, prilikom ubacivanja (INSERT) ili izmene (UPDATE) instance za entitet OSOBA, automatski (CASCADE) se izvodi ubacivanje i izmena instance TELEFONA (I:C, U:C).

U daljem tekstu prikazaće se referencijalni integritet za identifikujuće veze.

Referencijalni integritet za tip veze One-to-Zero-or-More

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa nula, jednom ili više instanci OSOBA (CHILD) i gde OSOBA egzistencijalno - identifikaciono zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.78.



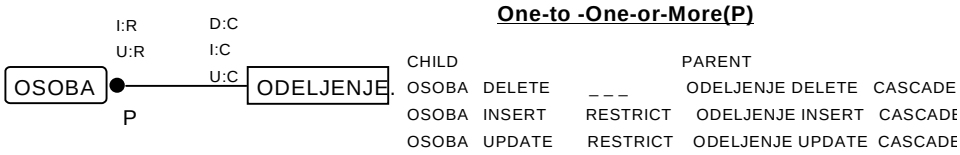
Slika 3.78 Referencijalni integritet za tip veze One-to-Zero-or-More

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Ubacivanje (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Bez ograničenja
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) bez 'roditelj' Odeljenje
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Brišu se sve Osobe za izbrisanog 'roditelj' Odeljenje (D:C)
	Restrict	Ne dozvoljava brisanje Odeljenja ako je 'roditelj' Osobi
Entitet OSOBA		Bez ograničenja
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Automatska izmena Osoba za izmene nad Odeljenjem (U:C)
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za tip veze One-to-One-or-More (P)

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa jednom ili više instanci OSOBA (CHILD) i gde OSOBA egzistencijalno - identifikaciono zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.79.



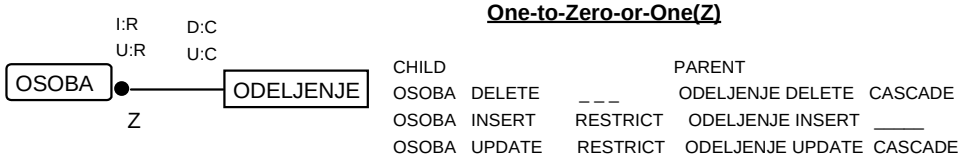
Slika 3.79. Referencijalni integritet za tip veze One-to-One-or-More (P)

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Unose se Osobe za uneseno Odeljenje (I:C)
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) bez 'roditelj' Odeljenje
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Brišu se sve Osobe za izbrisana Odeljenja (D:C) 'roditelj'
	Restrict	Ne briše Odeljenje (ako je poslednji 'roditelj' Osobi)
Entitet OSOBA	Cascade	Briše 'roditelj' Odeljenje ako su izbrisane Osobe koja su deca Odeljenja
	Restrict	Ne dozvoljava se brisanje Osobe ako nema više Osoba za istog 'roditelj' Odeljenje
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Automatska izmena Osoba za izmene nad Odeljenjem (U:C)
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za tip veze One-to-Zero-or-One (Z)

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa nula ili jednom instancom OSOBA (CHILD) i gde OSOBA egzistencijalno - identifikaciono zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.80.



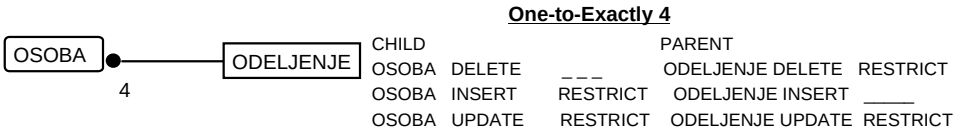
Slika 3.80. Referencijalni integritet za tip veze One-to-Zero-or-One (Z)

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Bez ograničenja
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) bez 'roditelj' Odeljenje ili ako postoji Osoba sa istim ''roditelj'em' Odeljenje
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Ako postoji, briše se Osoba za izbrisanog 'roditelj' Odeljenja (D:C)
	Restrict	Ne moze se dozvoliti brisanje Odeljenja ako postoji Osoba definisan 'roditelj'
Entitet OSOBA		Bez ograničenja
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Automatska izmena Osoba za izmene nad Odeljenjem (U:C)
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za tip veze One-to-Exactly n

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa tačno n instanci (npr. 4) OSOBA (CHILD) i gde OSOBA egzistencijalno - identifikaciono zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.81.



Slika 3.81. Referencijalni integritet za tip veze One-to-Exactly n

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

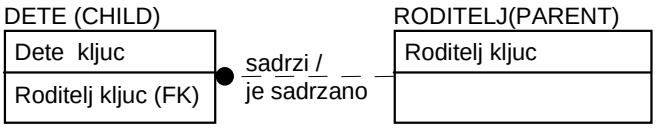
Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Unos Odeljenja mora pratiti tačno n Osoba za koje je Odeljenje 'roditelj'
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) ako ne postoje Odeljenja i n-1 unesenih Osoba za istog 'roditelj'
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Za izbrisano Odeljenje brišu se sve Osobe
	Restrict	Ne može se obrisati Odeljenje (D:R), ako je vezano za Osobu
Entitet OSOBA	Cascade	Briše se 'roditelj' Odeljenje i sve druge Osobe za koje je Odeljenje 'roditelj'
	Restrict	Ne može se dozvoliti brisanje Osobe
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Restrict	Ne može se izmeniti Odeljenje (U:R) ako je vezano za Osobu
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za neidentifikujuće veze

Kad je u pitanju neidentifikujuća veza između entiteta 'roditelj' i 'dete', primarni ključ entiteta 'roditelj' se nalazi u opisnom delu entiteta 'dete'. Prethodno opisana pravila kardinalnosti govore da za svaku instancu 'dete' postoji jedna instanca 'roditelj'. To je slaba veza koja specificira da je 'dete' egzistencijalno zavisno ili nezavisno od 'roditelj'.

Referencijalni integritet za egzistencijalno zavisnu neidentifikujuću vezu

Egzistencijalna zavisnost instance entiteta 'dete' od instance entiteta 'roditelj' vezana je za obaveznost veze (No Nulls) i grafički se definiše na slici 3.82.



Slika 3.82. Obaveznost veze (No Nulls)

Potrebno je detaljno razmotriti pravila za operacije: **delete** (brisanje), **insert (**ubacivanje) i **update** (ažuriranje).

Pravila brisanja za egzistencijalno zavisne neidentifikujuće veze

Pravila brisanja mogu se izvesti korišćenjem triju akcija:

- prvo, može se obrisati svaka instanca ''dete'ta' čiji je nasleđeni ključ obrisana instanca 'roditelj' (akcija CASCADE);
- drugo, alternativno, može se sprečiti brisanje 'roditelj' ako postoji bilo koje 'dete', čiji bi

deo primarnog ključa bio nekompletan (akcija RESTRICT);

- treće, za svaku obrisanu instancu 'roditelj' može se instanca entiteta 'dete' postavititi kao Set Default.

U okviru ERwin CASE alata difoltna vrednost za brisanje entiteta 'dete' ne zahteva izvođenje prethodne tri akcije (NONE), dok je difoltna vrednost za brisanje 'roditelj' RESTRICT.

CASCADE omogućuje da sve instance ''dete'ta' obuhvaćene brisanjem instance 'roditelj' budu obrisane, što je definisano rečenicom PARENT DELETE CASCADE (D:C).

RESTRICT definiše ograničenje tako da instanca 'roditelj' ne može biti obrisana dok sve instance ''dete'ta' koje imaju nasleđeni ključ ne budu prethodno obrisane. Ako postoji neki nasleđeni, brisanje je zabranjeno.

SET DEFAULT definiše standardnu vrednost instance entiteta 'dete' za izbrisanog 'roditelj'.

Pravila ubacivanja (Insert) i ažuriranja (Update) za egzistencijalno zavisne neidentifikujuće veze

Pravila za ubacivanje (INSERT) i ažuriranje (UPDATE) upravljaju šta će biti kada je red u tabeli dodat ili ažuriran. Ažuriranje je, u suštini, ubacivanje, ali sa nekim dodatnim pravilima.

Za operacije **ubacivanje (INSERT) i izmena (UPDATE)**, red može biti dodat ili izmenjen samo ako svi referencirani preneseni ključevi odgovaraju postojećim redovima u referenciranim tabelama.

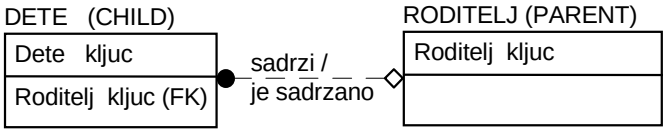
Pravila ubacivanja/ažuriranja mogu se izvesti korišćenjem triju akcija:

- prvo, ne može se ubaciti/ažurirati instanca ''dete'ta' bez instance 'roditelj' (akcija RESTRICT);
- drugo, alternativno, za ubačenog/ažuriranog 'roditelj' treba ubaciti/ažurirati i bilo koje 'dete', čiji bi deo primarnog ključa bio ključ 'roditelj' (akcija CASCADE);
- treće, prilikom ubacivanja/ažuriranja instance entiteta 'dete' može se postaviti standardna vrednost instance entiteta 'dete' za 'roditelj' koji ne postoji.

U okviru ERwin CASE alata difoltna vrednost za ubacivanje/ažuriranje entiteta 'dete' je RESTRICT, dok je difoltna vrednost za ubacivanje 'roditelj' - NONE a za ažuriranje RESTRICT.

Referencijalni integritet za egzistencijalno nezavisnu neidentifikujuću vezu

Ako je veza **neobavezna** (Nulls Allowed), tada 'dete' nije egzistencijalno zavisno, ali poštuje tu vezu i grafički se predstavlja rombom (slika 3.83).



Slika 3.83. Neobaveznost veze (Nulls Allowed)

Potrebno je detaljno razmotriti pravila za operacije: **delete (**brisanje), **insert (**ubacivanje) i **update (**ažuriranje).

Pravila brisanja za egzistencijalno nezavisne neidentifikujuće veze

Pravila brisanja mogu se izvesti korišćenjem četiri akcije:

- prvo, može se obrisati svaka instanca ''dete'ta' čiji je nasleđeni ključ obrisana instanca 'roditelj' (akcija CASCADE);
- drugo, alternativno, može se sprečiti brisanje 'roditelj' ako postoji bilo koje 'dete' čiji bi deo primarnog ključa bio nekompletan (akcija RESTRICT);

- treće, za svaku obrisanu instancu 'roditelj' može se instanca entiteta 'dete' postaviti kao Set Default;
- četvrto, za svaku obrisanu instancu 'roditelj' može se instanca entiteta 'dete' može se postaviti kao Set Null.

U okviru ERwin CASE alata difoltna vrednost za brisanje entiteta 'dete' ne zahteva izvođenje prethodne četiri akcije (NONE), dok je difoltna vrednost za brisanje 'roditelj' SET NULL.

CASCADE omogućuje da sve instance ''dete'ta' koje će biti obuhvaćene brisanjem instance 'roditelj' budu obrisane, što je definisano rečenicom PARENT DELETE CASCADE (D:C).

RESTRICT definiše ograničenje tako da instanca 'roditelj' ne može biti obrisana dok sve instance ''dete'ta' koje imaju nasleđeni ključ ne budu prethodno obrisane. Ako postoji neki nasleđeni, brisanje je zabranjeno.

Set Null ili Set Default pokazuje da je instanca 'roditelj' obrisana, ali je ključ 'roditelj' zadržan kao podatak u entitetu 'dete'. Brisanje nije CASCADE, pa se ključ postavlja kao NULL ili Set Default vrednost. Prednost tog pristupa je da se mogu sačuvati informacije o ''dete'tu' dok je efektivna veza između 'roditelj' i ''dete'ta' prekinuta.

Pravila ubacivanja (Insert) i ažuriranja (Update)

Pravila za ubacivanje (INSERT) i ažuriranje (UPDATE) upravljaju šta će biti kada je red u tabeli dodat ili ažuriran. Ažuriranje je, u suštini, ubacivanje, ali sa nekim dodatnim pravilima.

Za operacije **ubacivanje (INSERT) i izmena (UPDATE)**, red može biti dodat ili izmenjen samo ako svi referencirani preneseni ključevi odgovaraju postojećim redovima u referenciranim tabelama.

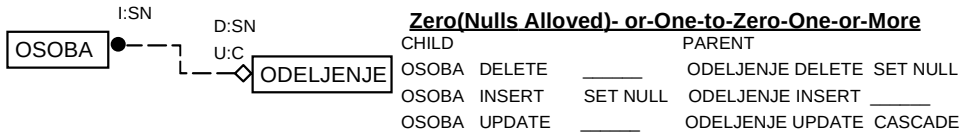
Pravila ubacivanja/ažuriranja mogu se izvesti korišćenjem četiri akcije:

- prvo, ne može se ubaciti/ažurirati instanca ''dete'ta' bez instance 'roditelj' (akcija RESTRICT);
- drugo, alternativno, može se za ubačenog/ažuriranog 'roditelj' ubaciti/ažurirati i bilo koje 'dete' čiji bi deo primarnog ključa bio ključ 'roditelj' (akcija CASCADE);
- treće, prilikom ubacivanja/ažuriranja instance entiteta 'dete' može se postaviti standardna vrednost instance entiteta 'dete' za 'roditelj' koji ne postoji;
- četvrto, prilikom ubacivanja/ažuriranja instance entiteta 'dete' može se postaviti Null vrednost instance entiteta 'dete' za 'roditelj' koji ne postoji.

U okviru ERwin CASE alata difoltna vrednost za ubacivanje/ažuriranje entiteta 'dete' je Set Null, dok je difoltna vrednost za ubacivanje 'roditelj' - NONE, a za ažuriranje Set Null.

Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to Zero-One-or-More

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa nula, jednom ili više instanci OSOBA (CHILD) i gde OSOBA može, a ne mora da pripada ODELJENJU, referencijalni integritet je prikazan na slici 3.84.



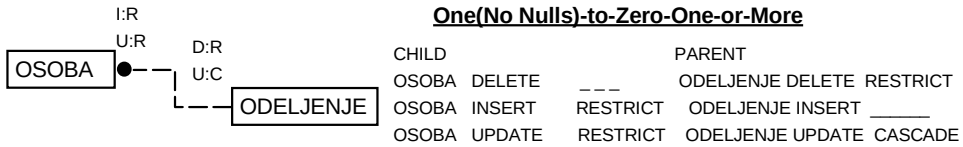
Slika 3.84. Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to-Zero-One-or-More

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Bez ograničenja
Entitet OSOBA	Set Null	Ne može se uneti Osoba bez 'roditelj' Odeljenje, sem ako se preneseni ključ Osobe ne setuje na NULL (I:SN)
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Brišu se sve Osobe za izbrisano Odeljenje kao 'roditelj'
	Restrict	Ne dozvoljava se brisanje Odeljenja ako postoji Osoba za koje je Odeljenje'roditelj'
	Set Null	Sve Osobe za koje je Odeljenje'roditelj' setuju preneseni ključ Osobe na NULL (D:SN)
Entitet OSOBA		Bez ograničenja
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Izmene se sve Osobe za izmenu Odeljenja (U:C)
Entitet OSOBA		Bez ograničenja

Referencijalni integritet za tip veze One (No Nulls)-to-Zero-One-or-More

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa nula, jednom ili više instanci OSOBA (CHILD) i gde OSOBA mora da pripada ODELJENJU, referencijalni integritet je prikazan na slici 3.85.



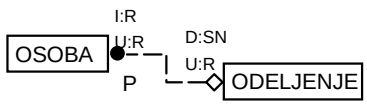
Slika 3.85. Referencijalni integritet za tip veze One (No Nulls)-to-Zero-One-or-More

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Bez ograničenja
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) bez 'roditelj' Odeljenje
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cacsade	Brišu se sve Osobe za izbrisano Odeljenje kao 'roditelj'
	Restrict	Ne može se obrisati Odeljenje (D:R) ako je vezano za Osobu
Entitet OSOBA		Bez ograničenja
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Brišu se sve Osobe za izbrisana Odeljenja (D:C)
Entitet OSOBA	Restrict	Ne mogu se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to-One-or-More (P)

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa jednom ili više instanci OSOBA (CHILD) i gde OSOBA može, a ne mora da zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.86.



Zero(Nulls Allowed)-or-One-to -One-or-More(P)

CHILD		PARENT	
OSOBA	DELETE	_ _ _	ODELJENJE DELETE SET NULL
OSOBA	INSERT	RESTRICT	ODELJENJE INSERT _____
OSOBA	UPDATE	RESTRICT	ODELJENJE UPDATE RESTRICT

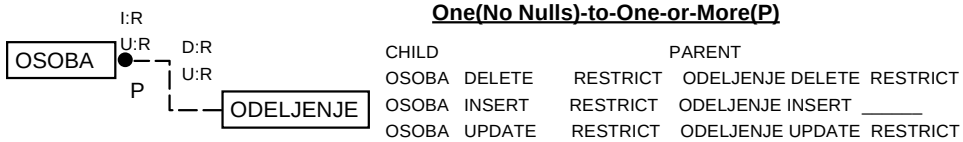
Slika 3.86. Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to-One-or-More (P)

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Unos Odeljenja mora pratiti unos najmanje jedne Osobe za koje je Odeljenje 'roditelj'
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) ako ne postoji Odeljenje, sem ako se reneseni ključ Odeljenja u Osobi setuje na NULL
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Brišu se sve Osobe za koje je izbrisano Odeljenje kao 'roditelj'
	Restrict	Ne dozvoljava se brisanje Odeljenja (ako je poslednji 'roditelj' Osobi)
	Set Null	Za sve Osobe gde Odeljenje 'roditelj' setuju se preneseni ključevi od odeljenja na NULL (D:SN)
Entitet OSOBA	Cascade	Briše se Odeljenje za izbrisanu Osobu
	Restrict	Ne dozvoljava se brisanje Osobe ako preneseni ključ Odeljenja u Osobu nije NULL
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Restrict	Ne može se izmeniti Odeljenje (U:R) ako je vezano za Osobu
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za tip veze One (No Nulls)-to-One-or-More (P)

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa jednom ili više instanci OSOBA (CHILD) i gde OSOBA mora da pripada ODELJENJU, referencijalni integritet je prikazan na slici 3.87.



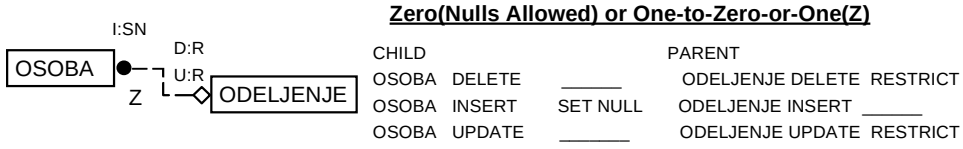
Slika 3.87. Referencijalni integritet za tip veze One (No Nulls)-to-One-or-More (P)

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Unos Odeljenja mora pratiti unos najmanje jedne Osobe za koje je Odeljenje 'roditelj'
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) ako ne postoji 'roditelj' Odeljenje
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Brišu se sve Osobe za koje je izbrisano Odeljenje kao 'roditelj'
	Restrict	Ne dozvoljava se brisanje Odeljenja (ako je poslednji 'roditelj' Osobi)
Entitet OSOBA	Cascade	Briše se 'roditelj' Odeljenje ako je izbrisana Osoba poslednje 'dete' Odeljenja
	Restrict	Zabrana brisanja Osobe (D:R) ako je poslednja u Odeljenju
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Restrict	Ne može se izmeniti Odeljenje (U:R) ako je vezano za Osobu
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to-Zero-or-One (Z)

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa nula ili jednom instancom OSOBA (CHILD) i gde OSOBA može, a ne mora da zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.88.



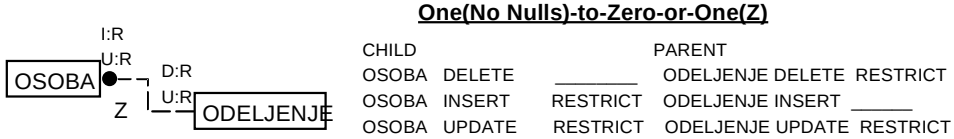
Slika 3.88. Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to-Zero-or-One (Z)

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Bez ograničenja
Entitet OSOBA	Set Null	Osoba se može uneti ako je preneseni ključ Odeljenja setovan na NULL (I:SN)
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Ako postoji, briše se Osoba za izbrisano Odeljenje kao 'roditelj'
	Restrict	Ne može se obrisati Odeljenje (D:R) ako je vezano za Osobu
	Set Null	Za izbrisanog 'roditelj' Odeljenje odgovarajući preneseni ključevi u Osobi setuju se na NULL
Entitet OSOBA		Bez ograničenja
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Restrict	Ne može se izmeniti Odeljenje (U:R) ako je vezano za Osobu
Entitet OSOBA	Restrict	Bez ograničenja

Referencijalni integritet za tip veze One (No Nulls)-to-Zero-or-One (Z)

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa nula ili jednom instancom OSOBA (CHILD) i gde OSOBA mora da pripada ODELJENJU, referencijalni integritet je prikazan na slici 3.89.



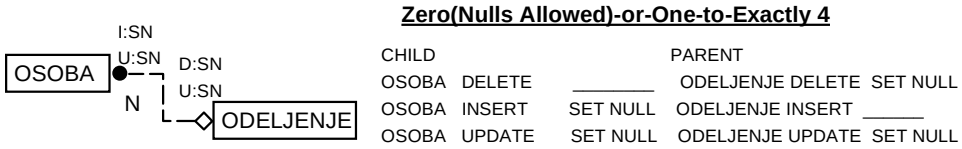
Slika 3.89. Referencijalni integritet za tip veze One (No Nulls)-to-Zero-or-One (Z)

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Bez ograničenja
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) ako ne postoji Odeljenje, ili ako postoji Osoba za isti 'roditelj' Odeljenje
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Ako postoji, briše se Osoba za izbrisano Odeljenje kao 'roditelj'
	Restrict	Ne može se obrisati Odeljenje (D:R) ako je vezano za Osobu
Entitet OSOBA		Bez ograničenja
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Restrict	Ne može se izmeniti Odeljenje (U:R) ako je vezano za Osobu
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to-Exactly n

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa tačno n instanci OSOBA (CHILD) i gde OSOBA može, a ne mora da zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.90.



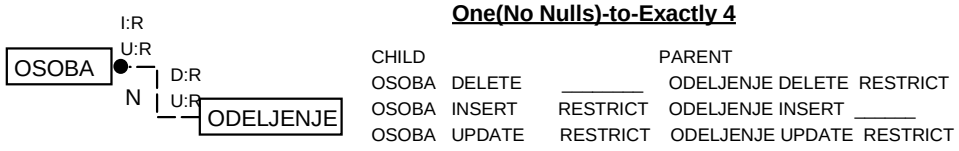
Slika 3.90. Referencijalni integritet za tip veze Zero (Nulls Allowed)-or-One-to-Exactly n

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Unos Odeljenja mora pratiti tačno n Osoba za koje je Odeljenje 'roditelj'
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) ako ne postoje Odeljenja i n-1 unesenih Osoba za isto Odeljenje, sem ako se ključ Odeljenja setuje na NULL u Osobi
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Za izbrisano Odeljenje brišu se sve Osobe
	Restrict	Ne može se obrisati Odeljenje (D:R) ako je vezano za Osobu
	Set Null	Za sve Osobe gde je Odeljenje 'roditelj' setuju se preneseni ključevi Odeljenja na NULL
Entitet OSOBA	Cascade	Ako preneseni ključ Odeljenja u Osobu nije NULL, briše se 'roditelj' Odeljenje i sve Osobe za koje je Odeljenje 'roditelj', inače briše se Osoba
	Restrict	Nije dozvoljeno brisanje Osoba
	Set null	Briše Osobu i u svim Osobama gde je Odeljenje 'roditelj' setuje preneseni ključ Odeljenje na NULL
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Set Null	Izmenom Odeljenja preneseni ključevi su Set Null (D:SN)
Entitet OSOBA	Set Null	Izmenom Osobe preneseni ključevi su Set Null (D:SN)

Referencijalni integritet za tip veze One (No Nulls)-to-Exactly n

Za kardinalnost gde je svako ODELJENJE (PARENT) povezano sa tačno n instanci OSOBA (CHILD) i gde OSOBA mora da zavisi od ODELJENJA, referencijalni integritet je prikazan na slici 3.91.



Slika 3.91. Referencijalni integritet za tip veze One (No Nulls)-to-Exactly n

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE		Unos Odeljenja mora pratiti tačno n Osoba za koje je Odeljenje 'roditelj'
Entitet OSOBA	Restrict	Ne može se uneti Osoba (I:R) ako ne postoje Odeljenja i n-1 unesenih Osoba za istog 'roditelj'
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Cascade	Za izbrisano Odeljenje brišu se sve Osobe
	Restrict	Ne može se obrisati Odeljenje (D:R) ako je vezano za Osobu
Entitet OSOBA	Cascade	Brišu se 'roditelj' Odeljenje i sve druge Osobe za koje je Odeljenje 'roditelj'
	Restrict	Ne može se dozvoliti brisanje Osobe
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Entitet ODELJENJE	Restrict	Ne može se izmeniti Odeljenje (U:R) ako je vezano za Osobu
Entitet OSOBA	Restrict	Ne može se izmeniti Osoba (U:R) ako ne postoji Odeljenje

Referencijalni integritet za rekurzivne veze

Referencijalni integritet za rekurzivne veze definiše se kao:

- referencijalni integritet hijerarhijskog tipa (rekurzija nad jednom tabelom)
- referencijalni integritet mrežnog tipa (rekurzija nad dve tabele).

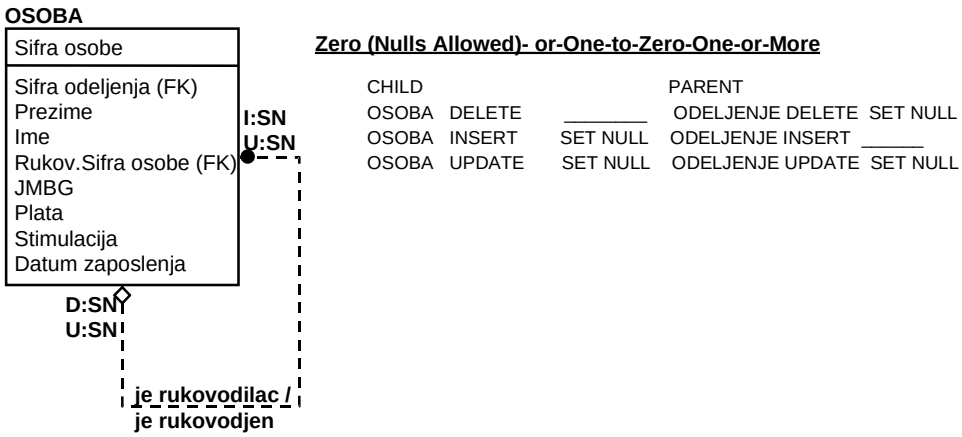
Referencijalni integritet rekurzivne veze hijerarhijskog tipa

Referencijalni integritet rekurzije hijerarhijskog tipa je veza entiteta nad samim sobom i definiše se kao neidentifikujuća veza.

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije ubacivanje (INSERT) ključa entiteta 'roditelj' ne zahteva nikakvu akciju (NONE) nad okolinom, dok se ubacivanjem instance entiteta 'dete' postavlja referencijalni integritet na Set Null (I:SN).

Operacija brisanja (DELETE) ključa entiteta 'roditelj' izvodi se akcijom Set Null (D:SN), dok brisanje instance entiteta 'dete' ne zahteva nikakvu akciju (NONE).

Izvodenjem akcije ažuriranja (UPDATE), instance entiteta: 'dete' i 'roditelj' postavljaju se u Set Null (U:SN).



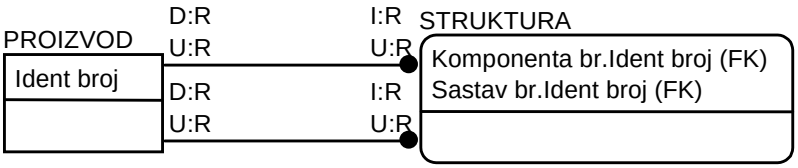
Slika 3.92. Referencijalni integritet rekurzivne veze hijerarhijskog tipa

Referencijalni integritet rekurzivne veze mrežnog tipa

Referencijalni integritet rekurzivne veze mrežnog tipa je specijalni slučaj 'Many-to-Many' veze nad samim sobom i treba je razbiti na dve veze 1:M.

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije ubacivanje (INSERT) ključa entiteta PROIZVOD ne zahteva nikakvu akciju (NONE) nad okolinom.

Izvođenje operacije brisanja (DELETE) i izmena (UPDATE) ključa entiteta PROIZVOD su zabranjeni akcijom RESTRICT (D:R i U:R). Entitet STRUKTURA poseduje složen ključ, pa su operacije i izmena (UPDATE) i ubacivanje (INSERT) zabranjene (U:R, I:R), jer zavise od postojanja ključa entiteta PROIZVOD. Operacija brisanje (DELETE) entiteta STRUKTURA ne zahteva nikakvu akciju (NONE) nad okolinom.



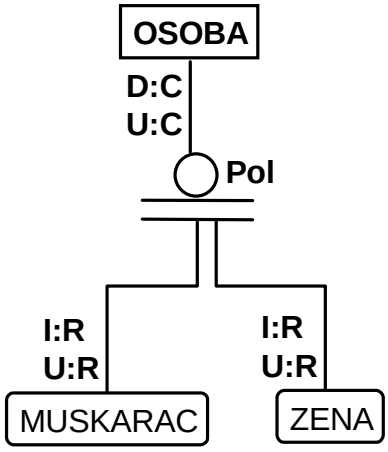
Slika 3.93. Referencijalni integritet rekurzivne veze mrežnog tipa

Referencijalni integritet za tip veze kategorija

Potrebno je posmatrati način definisanja referencijalnog integriteta za generalizovani entitet OSOBA u varijanti nepotpune i potpune strukture.

Referencijalni integritet potpune strukture

Ako se posmatra referencijalni integritet potpune strukture može se uočiti da se izmenom (UPDATE) ili brisanjem (DELETE) ključa entiteta OSOBA akcijom CASCADE (D:C, U:C) prenose ove operacije i na podtipove i da se one automatski izvršavaju. S druge strane, ne može se izvršiti operacija ubacivanje (INSERT) ili izmena (UPDATE) ključeva odgovarajućih podtipova (I:R, U:R). Ova zabrana definisana je akcijom RESTRICT (R).



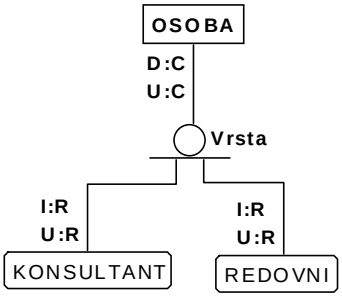
Slika 3.94. Referencijalni integritet potpune strukture

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet OSOBA		Unošenje Osobe mora pratiti unošenje i Muskarca ili Zene za isti ključ kao Osoba za diskriminator kategorije Pol
Entitet MUSKARAC		Ne može se uneti Muskarac bez unesenog 'roditelj' Osoba za kategoriju definisanu u diskriminatoru Pol kao "M".
Entitet ZENA		Ne može se uneti Zena bez unesenog'roditelj' Osoba za kategoriju definisanu u diskriminatoru Pol kao "Z".
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet OSOBA	Cascade	Briše se primerak kategorije za izbrisanu Osobu kao 'roditelj'
	Restrict	Nije dozvoljeno brisanje Osobe
Entitet MUSKARAC	Cascade	Briše 'roditelj' Osobu zajedno sa kategorijom Muskarac
	Restrict	Nije dozvoljeno brisanje Muskarac
Entitet ZENA	Cascade	Briše 'roditelj' Osobu zajedno sa kategorijom Zena
	Restrict	Nije dozvoljeno brisanje Zena

Referencijalni integritet nepotpune strukture

Ako se posmatra referencijalni integritet nepotpune strukture može se uočiti da se izmenom (UPDATE) ili brisanjem (DELETE) ključa entiteta OSOBA, i akcijom CASCADE (D:C, U:C), prenose ove operacije i na podtipove i one se automatski izvršavaju. S druge strane, ne može se izvršiti operacija ubacivanje (INSERT) ili izmena (UPDATE) ključeva odgovarajućih podtipova (I:R, U:R). Ova zabrana definisana je akcijom RESTRICT (R).



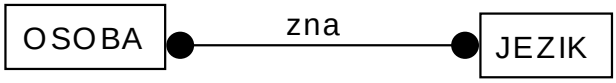
Slika 3.95. Referencijalni integritet nepotpune strukture

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Entitet OSOBA		Unošenje Osobe može se definisati bez unošenja Konsultanta ili Redovnog i pri tom indikator diskriminatora nije označen
Entitet KONSULTANT		Ne može se uneti Konsultant bez unesenog 'roditelj' Osoba za kategoriju definisanu u diskriminatoru Vrsta kao "K"
Entitet REDOVNI		Ne može se uneti Redovni bez unesenog 'roditelj' Osoba za kategoriju definisanu u diskriminatoru Vrsta kao "R".
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Entitet OSOBA	Cascade	Briše se primerak kategorije za izbrisanu Osobu kao 'roditelj'
	Restrict	Nije dozvoljeno brisanje Osobe
Entitet KONSULTANT	Cascade	Briše 'roditelj' Osobu zajedno sa kategorijom Konsultant
	Restrict	Nije dozvoljeno brisanje Konsultanta
	Set Null	Ne briše se Osoba, već se atribut Vrsta postavlja na Set Null
Entitet REDOVNI	Cascade	Briše 'roditelj' Osobu zajedno sa kategorijom Redovni
	Restrict	Nije dozvoljeno brisanje instance entiteta Redovni

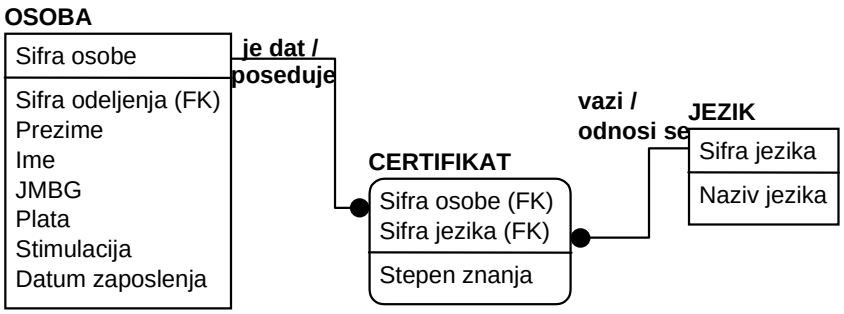
Referencijalni integritet za neodređujuću vezu

Kao što je pokazano u prethodnom poglavlju kada je definisana kardinalnost veza, osnovno pravilo koje se ovde mora poštovati je da se mora izvesti prevođenje 'Many-to-Many' veze u 'Zero-to-Many' vezu.



Slika 3.96. Veza 'Many-to-Many'

Ovaj problem se rešava tako što se dodaje asocijativni entitet nazvan CERTIFIKAT i na taj način prevodi 'Many-to-Many' veza u par One-to-Zero-One-or-Many vezu (slika 3.97).



Slika 3.97. Kardinalnost veza entiteta CERTIFIKAT

Potrebno je detaljno razmotriti pravila za operacije: *insert* **(**ubacivanje)**,** *delete* **(**brisanje) i *update* **(**ažuriranje) za primer sa slike 3.97.

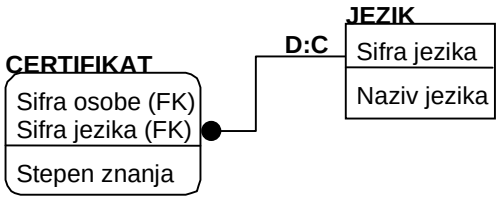
Pravila brisanja

Pravila brisanja mogu se izvesti korišćenjem dveju operacija:

- prvo, može se obrisati svaka instanca CERTIFIKAT čiji je nasleđeni ključ obrisana instanca entiteta JEZIK (akcija CASCADE), ili
- drugo, može se sprečiti brisanje JEZIKA ako postoji bilo koji CERTIFIKAT čiji bi deo primarnog ključa bio nekompletan (akcija RESTRICT).

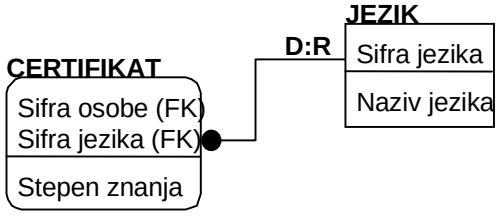
Ove situacije koje su u vezi sa pravilima brisanja nazivaju se **CASCADE** i **RESTRICT,** a odluka koje pravilo treba da se koristi biće definisana u modelu.

Dakle, **CASCADE** omogućuje da sve instance CERTIFIKAT obuhvaćene brisanjem instance JEZIKA budu, takođe, obrisane, što je definisano rečenicom PARENT (JEZIK) DELETE CASCADE (D:C), tj. ako je u pitanju, npr., instanca "nemački jezik" u entitetu JEZIK, onda će se izbrisati automatski svi certifikati izdati za "nemački jezik" (slika 3.98).



Slika 3.98. Brisanje entiteta CERTIFIKAT akcijom CASCADE

RESTRICT definiše ograničenje tako da instanca JEZIKA ne može biti obrisana dok sve instance CERTIFIKAT koje imaju nasleđeni ključ ne budu prethodno obrisane. Ako postoji neki nasleđeni, brisanje je "ograničeno". U dosadašnjem primeru moraju prethodno biti izbrisane sve instance entiteta CERTIFIKAT za "nemački jezik", pa tek onda da se pristupi brisanju instance "nemački jezik" u entitetu JEZIK (slika 3.99).

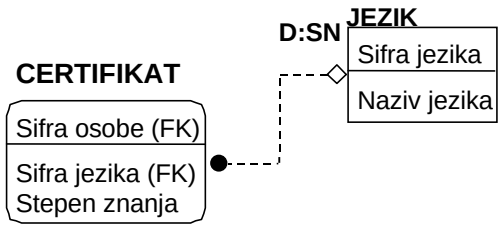


Slika 3.99. Brisanje entiteta CERTIFIKAT akcijom CASCADE

Zašto treba ograničiti brisanje?

Jedan od razloga može biti da treba ostaviti instancu "nemački jezik" u entitetu CERTIFIKAT. Ako dođe do CASCADE brisanja, izgubiće se ova dopunska informacija.

To treba rešiti tako što entitet CERTIFIKAT neće egzistencijalno ili identifikaciono zavisiti od JEZIKA, tj. biće promenjen referencijalni integritet upotrebom slabe veze između dva entiteta (slika 3.100).



Slika 3.100. Prikaz načina ograničavanja brisanja

U navedenom slučaju, situacija je bitno različita. Kao što je već poznato, spoljnom ključu, koji je "donesen" preko slabe veze, dozvoljene su NULL vrednosti. Tako se za slabe veze javlja treća opcija koja se zove **Set Null**.

Set Null pokazuje da ako je instanca JEZIKA obrisana, ali je ključ JEZIKA zadržan kao podatak u entitetu CERTIFIKAT, brisanje nije CASCADE kao u prethodnom primeru i ne može se zabraniti brisanje. Stoga se postavlja ključ kao NULL vrednost. Prednost tog pristupa je da se mogu sačuvati informacije o CERTIFIKATU, dok je efektivna veza između entiteta JEZIK i entiteta CERTIFIKAT prekinuta.

Pravila za ubacivanje i izmenu

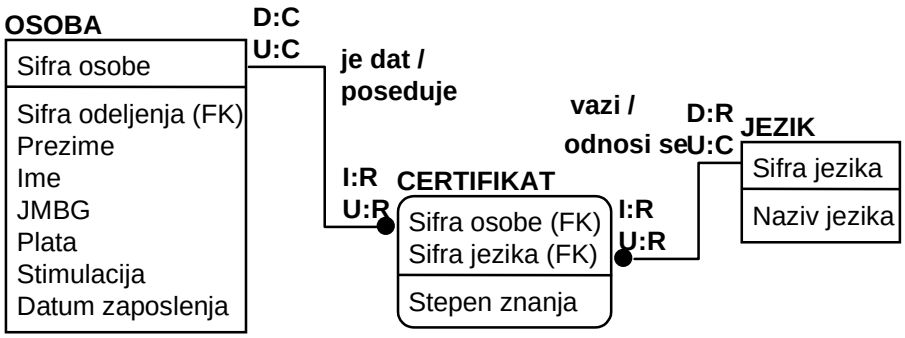
Odluka da se upotrebljava CASCADE ili Set Null odražava se na poslovne odluke o održavanju "istorije" znanja o vezi predstavljenoj preko prenesenih ključeva.

Naime, pravila brisanja (DELETE) upravljaju šta će se desiti u bazi podataka kada se red u tabeli obriše.

Pravila za ubacivanje (INSERT) i ažuriranje (UPDATE) upravljaju šta će biti kada je red u tabeli dodat ili ažuriran. Ažuriranje je, u suštini, ubacivanje, ali sa nekim dodatnim pravilima.

Za operacije *ubacivanje (INSERT) i izmena (UPDATE)*, red može biti dodat ili izmenjen samo ako svi referencirani preneseni ključevi odgovaraju postojećim redovima u referenciranim tabelama.

S obzirom na to da se sada mogu izmeniti poslovna pravila, to se asocijativni tip entiteta CERTIFIKAT koji povezuje entitete OSOBA i JEZIK može se prikazati kao na slici 3.101.



Slika 3.101. Referencijalni integritet za entitet CERTIFIKAT

Referencijalni integritet može se opisati na sledeći način:

Ako se posmatra referencijalni integritet veze OSOBA - CERTIFIKAT, može se uočiti da se izmenom (UPDATE) ili brisanjem (DELETE) ključa entiteta OSOBA - akcijom CASCADE (D:C, U:C) - prenose ove operacije i na ključeve entiteta CERTIFIKAT i one se automatski izvršavaju. S druge strane, ne može se izvršiti operacija ubacivanje (INSERT) ili izmena (UPDATE) entiteta CERTIFIKAT (I:R, U:R) ako ne postoji OSOBA. Ova zabrana definisana je akcijom RESTRICT (R).

Kada se se posmatra referencijalni integritet veze JEZIK - CERTIFIKAT, može se uočiti da se izmenom (UPDATE) ili ubacivanjem (INSERT) ključa entiteta JEZIK - akcijom CASCADE (D:C, U:C) - prenose ove operacije i na ključeve entiteta CERTIFIKAT i one se automatski izvršavaju.

Brisanje (DELETE) ključa entiteta JEZIK je onemogućeno akcijom RESTRICT (R).

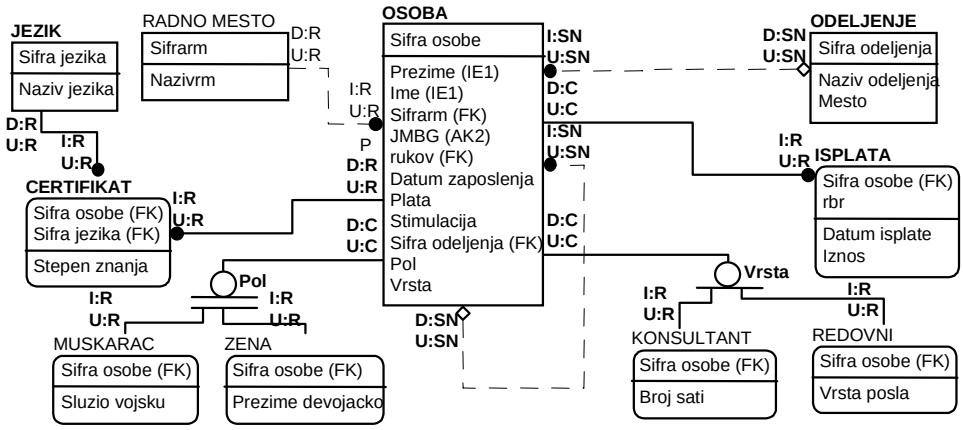
S druge strane, ne može se izvršiti operacija ubacivanje (INSERT) ili izmena (UPDATE) entiteta CERTIFIKAT (I:R, U:R) ako ne postoji OSOBA. Ova zabrana definisana je akcijom RESTRICT (R).

Mada svaka veza agregira dva entiteta, samo binarne veze kojima bi imalo smisla dodati neke atribute, ili koje bi imalo smisla povezivati sa drugim entitetima, treba tretirati kao agregirane entitete (agregacije). Ako postoji potreba da se direktno predstavi veza tri ili više entiteta, tada se i takva veza tretira kao agregacija.

Imajući sve to u vidu za primer dokumenta "Karton isplata", u sledećem koraku biće definisana strukturna pravila integriteta za dokument "Karton isplata".

Prikaz strukturnih pravila integriteta za dokument "Karton isplata"

Za primer dokumenta "Karton isplata" referencijalni integritet je prikazan na slici 3.102. U daljem tekstu biće opisan grafički prikaz i definisana tablica referencijalnih integriteta.



Slika 3.102. Referencijalni integritet za dokument "Karton isplata"

Referencijalni integritet veze ODELJENJE-OSOBA

Referencijalni integritet definisan ovim tipom veze pri izvođenju operacije ubacivanje (INSERT) ključa entiteta ODELJENJE ne zahteva nikakvu akciju (NONE) nad okolinom. Pri izvođenju operacija: brisanje (DELETE) i izmena (UPDATE) ključa entiteta ODELJENJE akcijom Set Null (D:SN i U:SN) preneseni ključ ŠSifra ODELJENJA (FK)Ć entiteta OSOBA setuje se na NULL. Entitet OSOBA za operaciju brisanje (DELETE) ne zahteva nikakvu akciju (NONE) nad okolinom. Izvođenjem operacije unosa (INSERT) i izmene (UPDATE) entiteta OSOBA, preneseni ključ entiteta ODELJENJE se setuje na NULL (I:SN i U:SN).

Referencijalni integritet veze RADNO MESTO-OSOBA

Referencijalni integritet definisan tipom veze RADNO MESTO-OSOBA pri izvođenju operacije ubacivanje (INSERT) ključa entiteta RADNO MESTO ne zahteva nikakvu akciju (NONE) nad okolinom. Operacije: brisanje (DELETE) i izmena (UPDATE) ključa entiteta RADNO MESTO su zabranjene (D:R, U:R), sve dok postoji preneseni ključ u entitetu OSOBA. Nad entitetom OSOBA operacije: izmena (UPDATE) i ubacivanje (INSERT) i brisanje (DELETE) nisu dozvoljene (U:R, I:R, D:R), jer zavise od postojanja ključa entiteta RADNO MESTO.

Referencijalni integritet veze OSOBA-ISPLATA

Referencijalni integritet definisan tipom veze OSOBA-ISPLATA pri izvođenju operacije ubacivanje (INSERT) ključa entiteta OSOBA ne zahteva nikakvu akciju (NONE) nad okolinom.

Pri izvođenju operacije brisanja (DELETE) ključa entiteta OSOBA istovetna operacija se izvodi akcijom CASCADE (D:C) i nad ključem entiteta ISPLATA (prvo se brišu sve ISPLATE za izabrani ključ OSOBE, a potom i OSOBA). Pri izvođenju operacije izmena (UPDATE) ključa entiteta OSOBA istovetna operacija se izvodi akcijom CASCADE (U:C) i nad ključem entiteta ISPLATA. Entitet ISPLATA poseduje složen ključ i obe operacije - izmena (UPDATE) i ubacivanje (INSERT), jer zavise (U:R, I:R) od postojanja ključa entiteta OSOBA. Operacija brisanje (DELETE) entiteta ISPLATE ne zahteva nikakvu akciju (NONE) nad okolinom.

Referencijalni integritet veze OSOBA-CERTIFIKAT-JEZIK

U primeru veze OSOBA-CERTIFIKAT-JEZIK postoji identifikujuća veza između JEZIKA i CERTIFIKATA. Primarni ključ entiteta JEZIKA i OSOBE postaju složeni primarni ključ entiteta CERTIFIKAT. Pravila kardinalnosti govore da za svaku instancu CERTIFIKATA postoji jedna instanca JEZIKA i OSOBE. Priroda ove veze specificira da CERTIFIKAT egzistencijalno - identifikaciono zavisi od JEZIKA i OSOBE.

Referencijalni integritet se može opisati na sledeći način (slika 3.102):

- Nad entitetom OSOBA operacije: DELETE i UPDATE su zabranjene akcijom RESTRICT, što je označeno sa D:R i U:R i znači da instanca entiteta OSOBA ne može biti izbrisana ili izmenjena dok prethodno ne budu izbrisane ili izmenjene sve instance entiteta CERTIFIKAT.
- Nad entitetom CERTIFIKAT operacije: ubacivanje (INSERT) i izmena (UPDATE) zabranjene su akcijom RESTRICT (R), što se označava sa I:R i U:R i znači da se instanca entiteta CERTIFIKAT ne može ubaciti ili ažurirati ako prethodno nisu definisane instance entiteta OSOBA i JEZIK.
- Nad entitetom JEZIKA operacije: DELETE I UPDATE su zabranjene akcijom RESTRICT, što je označeno sa D:R i U:R i znači da instanca entiteta JEZIK ne može biti izbrisana ili izmenjena dok prethodno ne budu izbrisane ili izmenjene sve instance entiteta CERTIFIKAT.

Pod pretpostavkom da se želi obrisati jedna instanca JEZIKA (npr., nemački jezik) tada treba obrisati sve instance CERTIFIKAT koji je deo nasleđenog ključa iz obrisane instance JEZIKA, zato što je deo ključa postao NULL vrednost, a NULL vrednost nije dozvoljena kao vrednost bilo kog dela primarnog ključa.

Referencijalni integritet veze OSOBA-MUSKARAC-ZENA

Ako se posmatra referencijalni integritet potpune strukture može se uočiti da se izmenom (UPDATE) ili brisanjem (DELETE) ključa entiteta OSOBA - akcijom CASCADE (D:C, U:C) - prenose ove operacije i na podtipove (MUSKARAC i ZENA) i one se automatski izvršavaju. S druge strane, ne može se izvršiti operacija ubacivanje (INSERT) ili izmena (UPDATE) odgovarajućih podtipova (I:R, U:R) ako ne postoji OSOBA. Ova zabrana je definisana akcijom RESTRICT (R).

Referencijalni integritet veze OSOBA-KONSULTANT-REDOVNI

Ako se posmatra referencijalni integritet nepotpune strukture može se uočiti da se izmenom (UPDATE) ili brisanjem (DELETE) ključa entiteta OSOBA - akcijom CASCADE (D:C, U:C) - prenose ove operacije i na podtipove (KONSULTANT i REDOVNI) i one se automatski izvršavaju. S druge strane, ne može se izvršti operacija ubacivanje (INSERT) ili izmena (UPDATE) odgovarajućih podtipova (I:R, U:R) ako ne postoji OSOBA. Ova zabrana je definisana akcijom RESTRICT (R).

Dakle, referencijalni integritet definisan za dosadašnji navedeni primer, a prikazan na slici 3.102, može imati sledeće događaje ili operacije :

- ubacivanje 'roditelj' (Parent Insert - **PI)**
- brisanje 'roditelj' (Parent Delete - **PD)**
- izmena 'roditelj' (Parent Update - **PU)**
- ubacivanje 'dete' (Child Insert - **CI)**
- brisanje 'dete' (Child Delete - **CD)**
- izmenu 'dete' (Child Update - **CU)**.

Način predstavljanja tabele strukturnih pravila integriteta (SP) za primer sa slike 3.102. dat je na sledeći način:

Entitet 'roditelj'	Entitet 'dete'	ime veza	U - UPDATE I - INSERT D - DELETE
OSOBA	ISPLATA	Prima	U - Cascade I - Restrict D - Cascade
	OSOBA	Rukovodi	U - Set Null I - Set Null D - Set Null
	MUSKARAC	is a	U - Cascade I - Restrict D - Cascade
	ZENA	is a	U - Cascade I - Restrict D - Cascade
	REDOVNO	is a	U - Cascade I - Restrict D - Cascade
	KONSULTANT	is a	U - Cascade I - Restrict D - Cascade
	CERTIFIKAT	Zna	U - Restrict I - Restrict D - Restrict
ODELJENJE	OSOBA	Zaposljava	U - Set Null I - Set Null D - Set Null
RADNO MESTO	OSOBA	Pripada	U - Restrict I - Restrict D - Restrict
JEZIK	CERTIFIKAT	govori	U- Restrict I - Restrict D - Restrict

Slika 3.103. Prikaz pravila integriteta

Na osnovu prethodno izvedenih aktivnosti (slika 3.49.) u sledećem koraku treba identifikovati poslovne domene.

Aktivnost
2.4.3. Identifikacija poslovnog domena

Entiteti se, kao što je već rečeno, opisuju preko svojih svojstava, odnosno atributa, od kojih svaki (atribut) u jednom trenutku vremena ima neku vrednost. Tačnije, atributi uzimaju vrednost iz skupa mogućih vrednosti koji se zove domen.

Pošto domen predstavlja skup vrednosti, u njemu se svaka vrednost javlja samo jednom. Svi elementi skupa se međusobno razlikuju. Atribut nema svojstvo skupa, jer se određene vrednosti mogu ponavljati, pa nije moguća jednoznačna identifikacija pojedinih elemenata.

Domen se može definisati po IDEF1X metodologiji kao bazni i tipski domen.

Bazni domen

Svaki domen se tretira kao podtip baznog *domena*. Naime, pod baznim domenima se podrazumevaju *tipovi podataka* (po IDEF1X standardu to su Character, Numeric ili Boolen), koji postoje u standardnim programskim jezicima (ceo broj, niz karaktera i slično) i samim tim *nasleđuju* karakteristike i operacije koje se mogu definisati nad ovim domenima.

Tipski domen

Tipski domen se definiše preko svog imena, baznog domena i, eventualno, prostih ili složenih ograničenja datih odgovarajućim domen-pravilima (domain rules) kojima se definišu moguće vrednosti u domenu.

Definišu se sledeća prosta ograničenja, tj. domen-pravila:

- *relacioni operatori* (operatora poređenja <,>,=,...);
- *IN lista vrednosti*, kao skup pojedinačnih vrednosti koje se definišu eksplicitnim navođenjem svih dozvoljenih vrednosti (IN ['A,B,C']);
- *BETWEEN rang domena,* koji se definiše tako da se vrednosti iz domena uzimaju iz užeg skupa vrednosti (BETWEEN 10 AND 200);
- *NOT NULL* kada se želi iskazati da neki atribut ne može da ima nula vrednost, pa se uvodi specifičan predikat NOT NULL, preko koga se definiše odgovarajući domen; obrnuto, pretpostavlja se da svaki domen uključuje u sebe i *nula vrednost* (specijalnu vrednost koja se dodeljuje nekom atributu objekta kada se njegova vrednost ne poznaje).

Dakle, specifiraju se *dozvoljene vrednosti* domena koje se definišu validacionim izrazima pomoću ključnih reči BETWEEN, IN i preko relacionih operatora (operatora poređenja <,>,=,...) i slično.

Složena ograničenja se sastoje iz prostih, povezanih logičkim operatorima AND, OR i NOT.

Imajući sve to u vidu za primer dokumenta "Karton isplata", u sledećem koraku treba definisati poslovni domen.

Poslovni domen na primeru dokumenta "Kartona isplata"

Poslovni domen za dosadašnji primer "Karton isplata" je prikazan u obliku tabele na slici 3.104.

Naziv domena	Opis domena	Tip podatka (Domen)	Null Opcija	Default naziv	Validaciono pravilo (Ograničenje)	Validacioni naziv
default		Text(18)			<>IsNull()	Obavezan unos
Broj	Domen definiše sve brojeve	Integer	NOT NULL			Obavezan unos
Datum	Opšti datum kome je vrednost sistemski datum	Date/ Time		Današnj i datum	<=Date()	Datum zapošljenja
Naziv	Ovaj domen je definisan za sve atribute koji opisuju nazive	Text(30)	NOT NULL		<>IsNull()	Obavezan unos
Oznaka	Ovaj domen definiše identifikacione oznake	Text(6)	NOT NULL		<>IsNull()	Obavezan unos
Primanja	Domen opisuje plate i honorare	Currency	NOT NULL			
Stimulacija	Domen za definisanje intervala i tipa stimulacije	Currency			BETWEEN 500 AND 1000	Interval stimulacije
Stepen	Stepen znanja P-piše;G-govori; C-čita	Text(3)	NOT NULL	Stepen	INŠ'P,G,C'Ć	Stepen

Slika 3.104. Tabela poslovnih domena za primer dokumenta "Karton isplata"

U tabeli su prikazani:

- naziv domena za koji se vezuju odgovarajući atributi;
- domen, definisan kao tip podataka;
- null opcija, kojom se definiše obaveznost unosa podataka;
- default vrednost, kao specificirana vrednost koja se ubacuje (insert) u kolonu kada se unose podaci;
- default naziv (komentar vezan za default vrednost);
- ograničenje ili validaciono pravilo kojim se ograničavaju set vrednosti datog domena;
- validacioni naziv, koji se pojavljuje kada je narušeno zadato ograničenje **(**validacioni naziv je očigledna ili neka opšta poruka).

Za primer dokumenta "Karton isplata" definisani su sledeći domeni (slika 3.104):

- U okviru kolone Naziv domena definiše se *default domen* (<default>) koji je ERwin predifinisani domen, koji se automatski setuje, a svi ostali domeni nasleđuju osobine tog domena koji je 'roditelj'. Definiše se tipom podatka Text (18) i ima obavezan unos.
- *Broj* je domen kojim se definišu: svi brojevi, Integer tip podatka i NOT NULL.
- *Datum* je domen kojim se definišu atributi tipa datuma i koji, po difoltu, uzima današnji datum i ima obavezan unos.
- *Naziv* je domen za definisanje svih atributa, koji se opisuju tipom podataka Text, dužine

30 i gde je obavezan unos.

- *Oznaka* je domen kojim se definišu: ident brojevi svih šifarnika i tip podataka Text dužine 6 sa obaveznim unosom (NOT NULL).
- *Primanja* su domen kojim se definišu: atribut plata i tip podataka Currency i obavezan je unos.
- *Stimulacija* je domen definisan intervalom između 500 i 1000 dinara.
- *Stepen* je domen definisan operatorom IN gde se definišu konkretne vrednosti.

Osnovna veza između atributa i domena je u tome što se atribut, koji je identičan sa svojim domenom, može koristiti kao ključ entiteta.

Na ovaj način izvršena je i kompletirana aktivnost *"2. Informaciono modeliranje"* gde su sve aktivnosti obavljane nezavisno od izabranog sistema za upravljanje bazama podataka (SUBP). U sledećoj aktivnosti *"3. Aplikativno modeliranje"* poslovi su vezani za izbor SUBP i predstavljaju konkretnu realizaciju vezano za izradu korisničke aplikacije.

Aktivnost 3. Aplikativno modeliranje

Aktivnost 3.1. Definisanje fizičkog dizajna

Aktivnost 3.2. Generisanje šeme baze podataka

Aktivnost 3.3. Izrada aplikacije

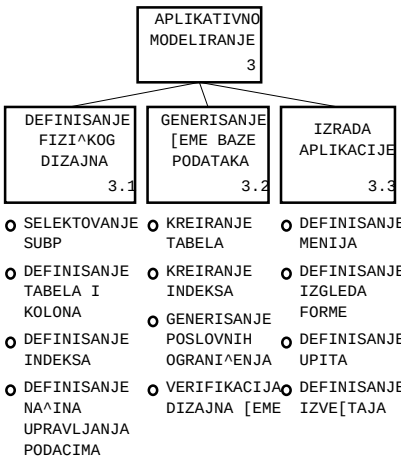
Aktivnošću *"2.Informaciono modeliranje"* završen je tzv. logički dizajn, pa sada predstoji da se izvođenjem aktivnosti *"3. Aplikativno modeliranje"* izvrši fizički dizajn za izabrani sistem za upravljanje bazom podataka - SUBP (Data Base Managment System - DBMS).

Sistem za upravljanje bazama podataka (SUBP) softverski je sistem za čuvanje i pretraživanje podataka i predstavlja skup programa čija je prvenstvena namena da na zahtev aplikativnih programa vrši manipulaciju podacima. To je i jedan od načina da se korisnicima (neprogramerima) omogući direktno korišćenje računara, tj. pristup i rukovanje podacima.

Uopšteno govoreći, baza podataka (BP) predstavlja zbirku uzajamno povezanih podataka, memorisanih sa kontrolisanom redundansom, da bi optimalno služili različitim aplikacijama. Podaci su memorisani nezavisno od programa koji ih koriste. Za dodavanje novih podataka i modificiranje ili pretraživanje postojećih podataka koriste se zajednički i kontrolisani pristupi.

Redundansa u podacima mora biti reducirana na najmanju moguću meru i strogo nadgledana, da bi na taj način bila osigurana usklađenost podataka u svakom momentu.

Aktivnost *"3. Aplikativno modeliranje"* treba da omogući projektantima baze podataka da fizički kreiraju efikasnu bazu podataka i da pomogne projektantskom timu u razvoju aplikacije i odabiru načina pristupa podacima.



Slika 4.1. Stablo aktivnosti za aktivnost "3. Aplikativno modeliranje"

Na slici 4.1 prikazano je stablo aktivnosti za aktivnost *"3. Aplikativno modeliranje"* čine ga sledeće aktivnosti:

- Aktivnost 3.1. Definisanje fizičkog dizajna,
- Aktivnost 3.2. Generisanje sheme baze podataka,
- Aktivnost 3.3. Izrada aplikacije.

Prve dve aktivnosti se definišu u okviru ERwin CASE alata, dok se treća aktivnost izvodi u odgovarajućem alatu za izabrani sistem za upravljanje bazom podataka i nije podržana IDEF1X standardom. Za prikaz izrade aplikacije za test-primer dokumenta "Karton isplata", biće korišćen MS ACCESS SUBP, jer je njegova upotreba i najčešća.

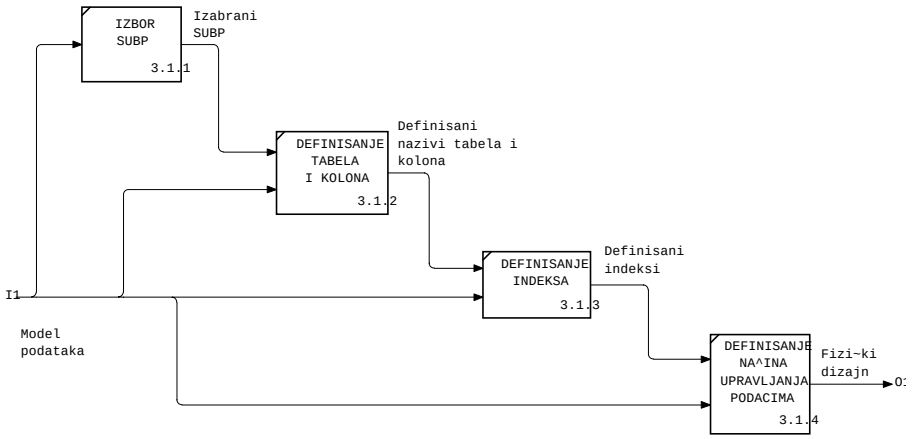
U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 4.1.

Aktivnost
3.1. Definisanje fizičkog dizajna

Aktivnost "3.1. Definisanje fizičkog dizajna" prevodi logički u fizički model, tj. entitet u tabele, atribute u kolone, kao i odgovarajuća ograničenja. ERwin nudi mogućnost istovremenog definisanja logičkog i fizičkog nivoa i jednostavno prebacivanje sa logičkog na fizički pogled na model.

Definisanje fizičkog dizajna se izvodi posredstvom sledećih aktivnosti (slika 4.1):

- Aktivnost 3.1.1. Izbor SUBP,
- Aktivnost 3.1.2. Definisanje tabela i kolona,
- Aktivnost 3.1.3. Definisanje indeksa,
- Aktivnost 3.1.4. Definisanje načina upravljanja podacima.



Slika 4.2. Dekompozicioni dijagram za aktivnost "3.1. Definisanje fizičkog dizajna"

Na slici 4.2. prikazan je dekompozicioni dijagram za aktivnost *"3.1. Definisanje fizičkog dizajna"*, gde je definisan tok informacija, počev od ulaznog dokumenta "Model podataka", preko nabrojanih aktivnosti definisanog fizičkog dizajna.

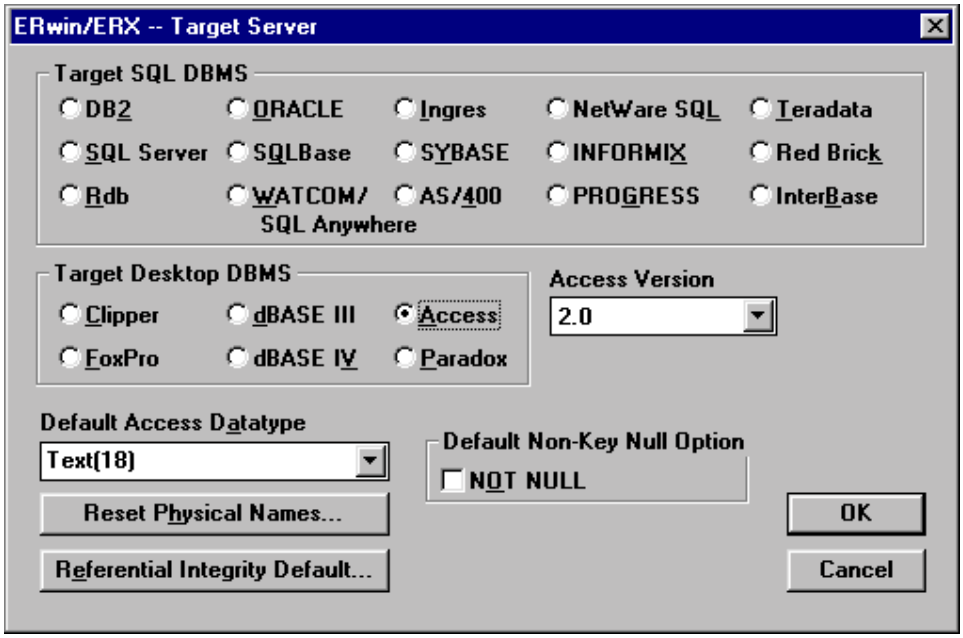
U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 4.2.

Aktivnost
3.1.1. Izbor SUBP

Aktivnost "3.1.1. Izbor SUBP" treba da definiše SUBP, gde će fizička šema biti kreirana i gde će se specificirati default tipovi podataka, null opcije i druge default opcije koje se koriste za generisanje kolona.

ERwin nudi komunikaciju prema sledećim SUBP (slika 4.3): DB2, ORACLE, Ingress, NetWare SQL, SQL Server, SQLBase, SYBASE, INFORMIX, Rdb, WATCOM, AS/40, Progres, Clipper, dBase III, dBase IV, Access, FoxPro i Peradox.

Svaki od ovih SUBP podržava određene tipove podataka i određenu sintaksu za definisanje strukture modela (sintaksa za opis relacija, kolona i domena na fizičkom nivou).



Slika 4.3. Izbor servera u ERwin-u

Za potrebe izrade test-aplikacije za dokument "Karton isplata" biće korišćen MS ACCESS sistem za upravljanje bazama podataka, kao što je pokazano na slici 4.3.

Osnovni problem je što je ERwin semantički bogat model, dok je SUBP daleko semantički siromašniji. Prebacivanje modela podataka iz ERwin-a u neki od komercijalnih SUBP treba da reši složen koncepcijski problem, a to je kako bogatstvo struktura modela ERwin-a prebaciti u neki SUBP koji ima relativno skromne semantičke strukture za opis modela. Rešavanje ove problematike je predmet daljih izlaganja.

Polazi se od činjenice da se podaci definišu *tipom podataka* koji predstavlja skup vrednosti koje imaju izvesne zajedničke karakteristike.

Tipovi podataka mogu biti:

- numerički (celobrojni, realni i logički),
- znakovni i
- strukturirani.

Celobrojni tip je podskup skupa celih brojeva, *realni* tip je podskup skupa realnih brojeva, a *logički* - tip podataka koji kvalifikuje stanje istine ili laži.

Znakovne tipove podataka čini konačan skup znakova.

Strukturirani tipovi podataka su skupovi vrednosti čija struktura ima određeni smisao. Nosioci strukturiranih podataka su strukturirane promenljive. U okviru SQL SUBP definišu se, sledeći strukturirani tipovi, imajući u vidu logički model podataka:

- entiteti postaju tabele,
- atributi se definišu kao kolone,
- instance ili primerci postaju redovi,
- u preseku reda i kolone definišu se polja.

Tabela se sastoji od kolona i redova, koji se mogu sagledavati u bilo kom redosledu, bez uticaja na sadržaj tabele. Tabelarno prikazivanje je prirodan način prikazivanja veza između podataka.Tabele moraju biti tako sastavljene da se nijedna veza ne izgubi.

Kolone su definisane nazivom i u jednoj koloni postoji samo jedna vrsta podataka. Ako se koristi ERwin, kolona se generiše iz atributa logičkog modela, pa se tip i dužina kolone generišu na osnovu osobina atributa ili ih određuje sam korisnik, dodeljujući im SIMBOLIČKI naziv.

Red se može definisati kao skup međusobno povezanih polja, koja sadrže osnovne podatke o nečemu.

Polje je osnovna i najmanja jedinica podatka koja može biti numerička, alfabetska ili alfanumerička.

Redovi se razlikuju među sobom, pa duplikati redova nisu dozvoljeni.

Kad se posmatra, na primer, tabela ODELJENJE sa kolonama definisanim vertikalno i redovima definisanim horizantalno:

ODELJENJE

SIFRAO	NAZIVO	MESTO
10	PRIPREMA	PANCEVO
20	RAZVOJ	NOVI SAD
30	PRODAJA	BEOGRAD
40	PROIZVODNJA	PAZOVA

za tabelu ODELJENJE definisani su:

- tri kolone (SIFRAO, NAZIVO i MESTO) i
- četiri reda (podaci za odeljenje 10, 20, 30 i 40)

Posmatranjem tabele može se videti da je svaki red sastavljen od polja i da svako polje sadrži vrednost polja na preseku reda i kolone.

Npr. u prvom redu tabele ODELJENJE, vrednost polja:

- SIFRAO (broj odeljenja) - 10 ,
- NAZIVO (naziv odeljenja) - PRIPREMA,
- MESTO (lokacija odeljenja) - PANCEVO.

Više između sebe povezanih tabela čini relacionu bazu podataka. Dakle, *relaciona baza podataka* je takav tip strukture koji se koristi za izražavanje odnosa između podataka u obliku jednostavnih dvodimenzionalnih tabela. Najveće teoretske osnove dao je dr E.F. Codd koji je objavio prve rezultate svoga istraživanja već 1969. godine.

Veliki broj SUBP sa atributom "relacioni", koji se pojavio na tržištu sedamdesetih godina, od trenutka kada je E.F.Codd dao prvu teorijsku definiciju ovoga modela, primorao je njegovog tvorca da 1981. godine definiše, a kasnije i dopunjava, kriterijume koje neki sistem treba da zadovolji da bi se mogao zvati relacionim.

Kriterijumi za ocenu relacionih SUBP-a

Definisano je 12 pravila koji omogućuju da se oceni do kog je nivoa neki SUBP relacioni.

1. Struktura SUBP se predstavlja samo tabelama.
2. Svaki podatak u bazi podataka dostupan je preko kombinacije imena tabele, vrednosti primarnog ključa i imena kolone, bez unapred zadatih pristupnih puteva i bez rekurzije ili iteracije.
3. Specifičan indikator, različit od "blanka" ("praznog") niza karaktera, nule ili bilo kog broja, koristi se za predstavljanje nula vrednosti, bez obzira na tip podatka.
4. SUBP treba da poseduje katalog (rečnik) podataka, koji se logički predstavlja na isti način kao i sama baza podataka.
5. SUBP mora posedovati bar jedan jezik čije se naredbe mogu izraziti kao niz karaktera sa dobro definisanom sintaksom i koji podržava: (1) definiciju podatka, (2) definiciju pogleda, (3) manipulaciju podatka, interaktivno i kroz programe, (4) definiciju pravila integriteta, (5) autorizaciju (sigurnost), (6) granice transakcija (BEGIN, COMMIT, ROLLBACK).
6. SUBP treba da poseduje efikasan algoritam pomoću koga može da odredi za svaki definisani pogled, u trenutku njegovog definisanja, da li se i koje operacije održavanja mogu da primene na taj pogled. Rezultat takvog algoritma se smešta u katalog baze podataka.
7. I nad tabelama i nad pogledima mogu se izvršavati ne samo operacije pretraživanja već i operacije održavanja baze podataka.
8. Aplikacioni program i interaktivna komunikacija treba da ostanu neizmenjeni kada se promeni fizička organizacija baze podataka.
9. Aplikacioni program i interaktivna komunikacija ostaju nepromenjeni kada se bilo koje promene, koje ne menjaju odgovarajući sadržaj tabele, unesu u baznu tabelu.
10. Pravila integriteta treba da se definišu u okviru definicije baze podataka i čuvaju se u rečniku podataka (Ne implementiraju se kroz aplikacione programe).
11. Sve navedene karakteristike treba da budu nezavisne od distribucije baze podataka.
12. Ako SUBP može da radi sa nekim jezikom treće generacije, u kome se obrađuje jedan red tabele u jednom trenutku vremena, kroz taj jezik se ne mogu zaobići pravila integriteta zadana preko samog relacionog SUBP.

Funkcije SUBP

S obzirom na prethodno definisana 12 pravila, sistemi za upravljanje bazama podataka imaju dve osnovne funkcije. Prva funkcija se koristi za memorisanje i održavanje podataka, koji izražavaju svojstva tabela (objekata posmatranja). Za izvođenje ove funkcije koriste se jezik za definisanje podataka i struktura podataka (Data Definition Language ili DDL). O tom jeziku biće više reči u poglavlju 4.2.

Druga funkcija se koristi za kontrolisan pristup do memorisanih podataka i prikazivanje podataka (vrednosti svojstava tabela) na zahtev korisnika. Za izvođenje ove funkcije koristi se jezik za manipulaciju podacima (Data Manipulation Language ili DML). O tom jeziku biće više reči u poglavlju 4.3.

Treba istaći da SUBP mora vršiti nadzor nad simultanim korišćenjem baze podataka. U načelu, svakom je korisniku data mogućnost da vrši manipulaciju sa memorisanim podacima. Tako, može se dogoditi da više korisnika istovremeno pokuša modifikovanje istih podataka. Stoga SUBP mora sinhronizovati simultane zahteve više korisnika, da ne bi došlo do gubljenja podataka.

Kada se to ima u vidu, rad sa SUBP treba da omogući:

- menjanje postojećih podataka u okviru baze podataka,
- brisanje postojećih redova,
- dodavanje novog reda,
- traženje (izdvajanje) konkretnog reda i
- pretraživanje baze podataka.

Mora se naglasiti da se sve promene u vezi sa poljem (unošenje novih podataka, menjanje postojećih i brisanje) moraju sprovoditi u okviru reda.

Postupak *menjanja* polja je sledeći:

- red u kome se vrši izmena podataka pronalazi se u memoriji pomoću adrese ili ključa;
- red se iz eksterne memorije prenosi u operativnu memoriju;
- vrši se izmena sadržaja datih polja;
- red se ponovo zapisuje u eksternu memoriju.

Postupak *brisanja* već postojećih redova zahteva pažnju - da se brišu pravi redovi, tj. da se poštuju pravila vezana za integritet baze podataka, kao i referencijalni integritet.

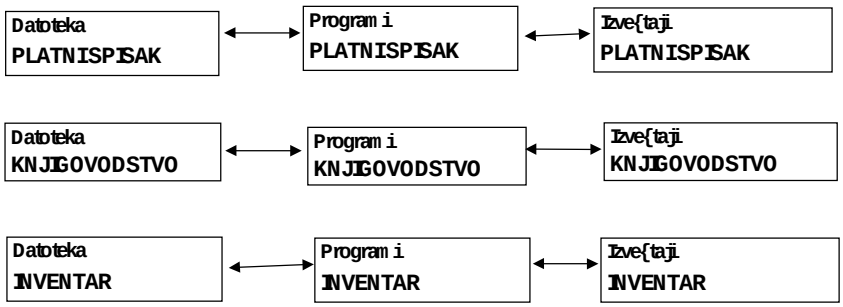
Postupak *dodavanja* novih redova zahteva obezbeđenje potrebnog memorijskog prostora i poštovanje integriteta baze podataka, kao i referencijalnog integriteta.

Traženje podrazumeva traženje konkretnog reda koje se obavlja preko primarnog ključa. Traženje je uspelo ako je vrednost primarnog ključa reda identična vrednosti ključa datog u argumentu.

Pretraživanje se vrši putem argumenta koji je dat kao LOGIČKI izraz. Sastavlja se lista zahtevanih svojstava i kriterijuma po kojima se vrši pretraživanje tabela i izdvajaju svi oni redovi koji u potpunosti udovoljavaju zahtevima (videti glavu 4.3).

Razlike između SUBP i datoteka

Korišćenjem klasičnih programskih jezika treće generacije svaki programer definiše programe i datoteke prema svojim potrebama i shvatanjima. Kao što se može videti na slici 4.4, parcelizovani način rada dovodi do redundanse (ponavljanja) sličnih datoteka u različitim aplikacijama.



Slika 4.4. Primer korišćenja datoteka

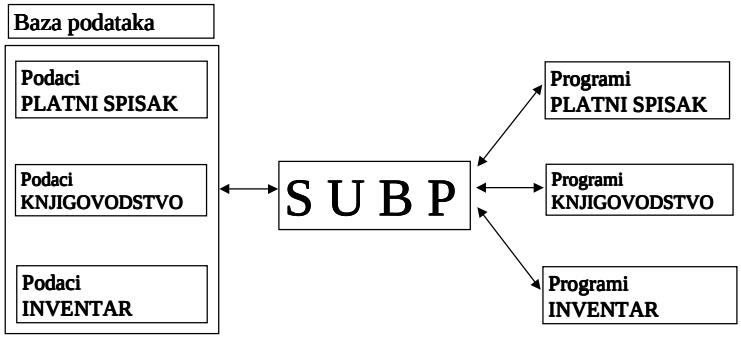
Datoteke nastaju kao produkt izolovano razvijenih programa (aplikacija), korišćenjem programskih jezika. Kao što se može videti na slici 4.4, datoteke su izrazito usklađene s

programima i prema njima imaju vlasnički odnos. Međutim, nisu prilagođene za upotrebu kod novih programa, jer oni nisu bili uzeti u obzir pri definisanju datoteke, a mogli bi uspešno da koriste neke podatke. Ovaj problem se polovično rešava promenom dizajna datoteke ili definisanjem dodatne datoteke za potrebe novog programa. Međutim, negativna reakcija je pojava višestrukog memorisanja podataka, ili tzv. redundanse podataka.

Za razliku od rada sa datotekama, korišćenje SUBP omogućuje da *neprogrameri* mogu pristupiti podacima i njima manipulisati:

- *direktnim* radom sa računarom svakog korisnika,
- *jednostavnim* definisanjem obrade (obradu definišu osobe koje se bave fizikom problema, a ne obradom podataka).

Dakle, SUBP treba da omogući svim ovlašćenim korisnicima korišćenje zajedničkih podataka (slika 4.5), skladištenje podataka sa minimalnom redundansom, logičku i fizičku nezavisnost programa od podataka, jedinstveno komuniciranje sa bazom podataka preko jezika bliskih korisniku. Nezavisnost programa i podataka se postiže kada se može dopunjavati i modifikovati struktura baze podataka bez posledica po postojeće korisnikove programe. Jezik blizak korisniku je tzv. SQL upitni jezik.



Slika 4.5. Primer korišćenja SUBP

SQL upitni jezik

Neproceduralni jezik SQL (Structured Query Language) dizajniran je tako da ga sa uspehom mogu koristiti i ljudi bez tehničkih znanja s područja obrade podataka, takozvani krajnji korisnici. SQL jezik je neproceduralan, jer specificira operacije u smislu ŠTA treba uraditi, a ne KAKO.

SQL upitni jezik omogućuje da korisnici mogu ad hoc formulisati i postavljati pitanja i veoma brzo dobijati odgovor, a da pri tom ne zavise od saradnje sa programerom. Programeri su na taj način rasterećeni i posvećuju se izradi kvalitetnih programa za aplikacije, koje zahtevaju mnogo tehničkog znanja (npr., veoma frekventne transakcije, kod kojih je važno da se postigne što kraće vreme odziva).

SQL omogućuje povezivanje sa klasičnim višim programskim jezicima, kao što su: COBOL, PL/I, PASCAL, FORTRAN, C i dr.

SQL jezik je usvojio Komitet Američkog nacionalnog Instituta za standarde (ANSI) kao standardni jezik relacionih baza podataka.

U relacionim bazama podataka se definiše struktura podataka u obliku tabela. SQL relacioni jezik pravi nove tabele, definišući podskupove i/ili kombinujući postojeće tabele. Dakle, SQL omogućuje da se jednom relacionom naredbom mogu čitati, ažurirati, ili brisati redovi memorisani u relacionoj bazi podataka.

SQL sadrži:

- konstrukcije analogne relacionoj algebri (videti prilog br.1),
- konstrukcije analogne relacionom računu (videti prilog br. 1),

- jezik za opis baza podataka (videti glavu 4.2),
- vezivanje SQL sa nekim standardnim jezikom (CURSOR operacija),
- kontrolne konstrukcije za upravljanje konkurentnom obradom,
- oporavak baze podataka (videti glavu 4.1),
- definisanje zaštite baze podataka (videti glavu 4.1),
- definisanje integriteta baze podataka (videti glavu 4.1).

SQL je jezik za:

- interaktivno definisanje baze podataka (Data Definition Language ili DDL) - jezik kojim treba da se omoguće kreiranje, dodavanje, brisanje tabela, pogleda, sinonima i indeksa, pa stoga poseduje sledeće komande (pogledati aktivnost *"3.2. Generisanje šeme baze podataka"*):

CREATE TABLE	▶	kreiranje tabele,
CREATE VIEW	▶	kreiranje pogleda,
CREATE SYNONIM	▶	kreiranje sinonima, *
CREATE INDEX	▶	kreiranje indeksa,
ALTER TABLE	▶	dodavanje kolona ili redefinisanje u tabeli,
DROP TABLE	▶	brisanje tabele,
DROP VIEW	▶	brisanje pogleda,*
DROP SYNONYM	▶	brisanje sinonima,*
DROP INDEX	▶	brisanje indeksa;

- pretraživanje podataka gde treba da se omogući izbor redova iz tabele (SQL za pretraživanje podataka koristi naredbu SELECT - pogledaj aktivnost *"3.3.3. Definisanje upita"*);
- manipulaciju podacima (Data Manipulation Language ili DML) koja treba da omogući ubacivanje redova, izmene i brisanje memorisanih podataka u tabeli (SQL za manipulaciju podacima poseduje određene komande - pogledaj aktivnost *"3.3.3.Definisanje upita"*):

INSERT	▶	ubacivanje redova u tabelu,
UPDATE	▶	izmena memorisanih podataka,
DELETE	▶	brisanje memorisanih podataka;

- upravljanje podacima (Data Control Language - DCL) kojim treba da se omoguće dodavanje i opoziv privilegija, prenos transakcija iz buffer-a u tabele, zaključavanje tabela i definisanje pogleda (SQL za upravljanje podacima poseduje određene komande - pogledati aktivnost *"3.1.4 Definisanje načina upravljanja podacima"*):

GRANT CONNECT	▶	dodeljivanje privilegije,
REVOKE	▶	opoziv privilegije,
COMMIT	▶	prenos transakcija iz bafera u tabelu,
ROLLBACK	▶	poništavanje izmena u tabelama pre commita,
LOCK TABLE	▶	zaključavanje tabele,*
AUDIT	▶	definisanje ORACLE pregleda.*

Napomena: Zvezdica (*) znači da su naredbe definisane u okviru ORACLE verzije SQL-a.

Na osnovu izabranog SUBP pristupa se u sledećem koraku - definisanju tabela i kolona.

Aktivnost
3.1.2. Definisanje tabela i kolona

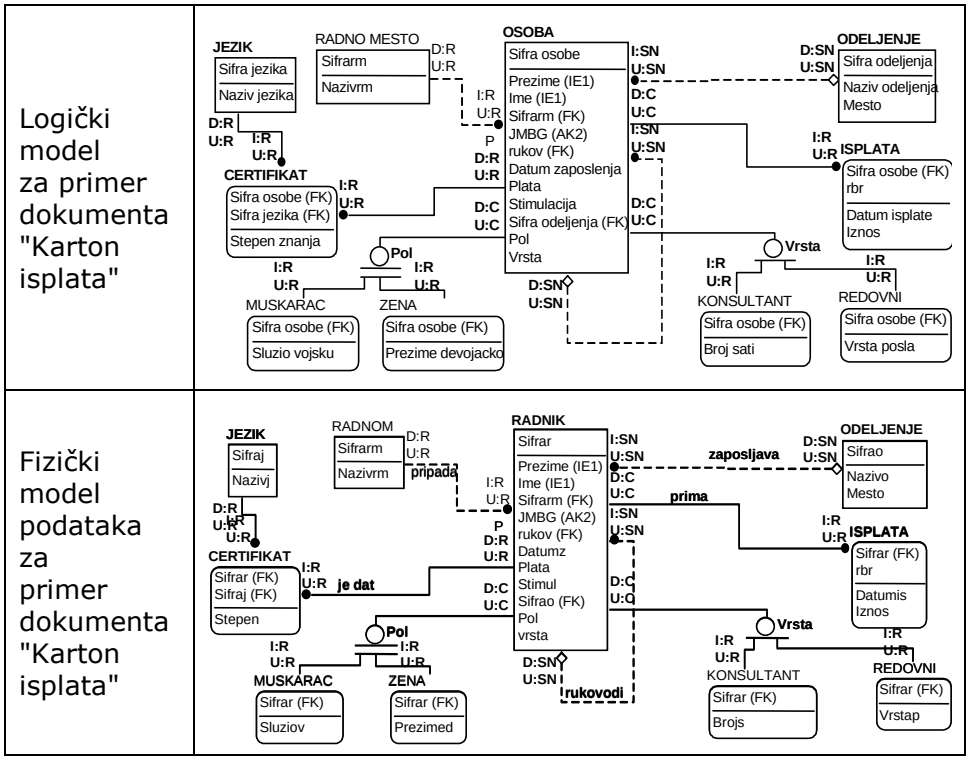
Implementacija entiteta i njihovih atributa u tabele i kolone nekog SUBP, korišćenjem ERwin-a, relativno je jednostavan posao. Programski modul ERwin-a za izgradnju fizičkog modela čita opis entiteta i atributa i formira tabele i polja fizičkog modela (slika 4.6).

Proces implementacije veza i njihovih definicija je mnogo složeniji posao. Radi ilustracije, sledi razmatranje prevođenja ERwin modela u MS Access dekstop SUBP.

Dakle, ERwin definiše tabele i kolone automatski, tj. nazivi tabela po defaultu dobijaju imena na osnovu naziva entiteta, a nazivi atributa po defaultu postaju nazivi kolona. I druge osobine se dodeljuju kao default setovane vrednosti (vrednosti koja će biti insertovana u kolonu). U Target Server editoru (Erwin) postavljaju se default vrednosti za (slika 4.3):

- Default Access Datatype kojim se definišu default tip i veličina kolone, npr., text (18);
- Reset Physical Name kojim se izvodi resetovanje fizičkih imena;
- Referential Integrity Default kojim se definišu default vrednosti referencijalnog integriteta.

Na osnovu definisanih default osobina prelazi se na realizaciju fizičkog modela. Na slici 4.6. prikazani su logički i fizički model podataka za primer dokumenta "Karton isplata", gde se automatski preuzimaju isti nazivi tabela kao nazivi entiteta i u okviru njih nazivi kolona, a osobine kolona se preuzimaju na osnovu naziva atributa.



Slika 4.6. Uporedni prikaz logičkog i fizičkog modela podataka

Može se videti na slici 4.6. da su za potrebe ove knjige promenjena pojedina imena za tabele i kolone da bi se ukazalo na razliku između logičkog i fizičkog modela. Naime, u okviru fizičkog modela definisane su skraćenice za kolone umesto punih imena atributa u logičkom modelu (pogledati entitet OSOBA i tabelu RADNIK).

Ako se žele promeniti default vrednosti za kolone biće korišćeni ERwin editori:

- Column property Editor, kojim se definišu osobine kolona;
- Domain Editor, kojim se definišu osobine domena;

- Validation Rule editor, kojim se definišu validaciona pravila;
- Valid Value Editor, kojim se definišu validacione vrednosti;
- Access Default Editor, kojim se definišu default vrednosti.

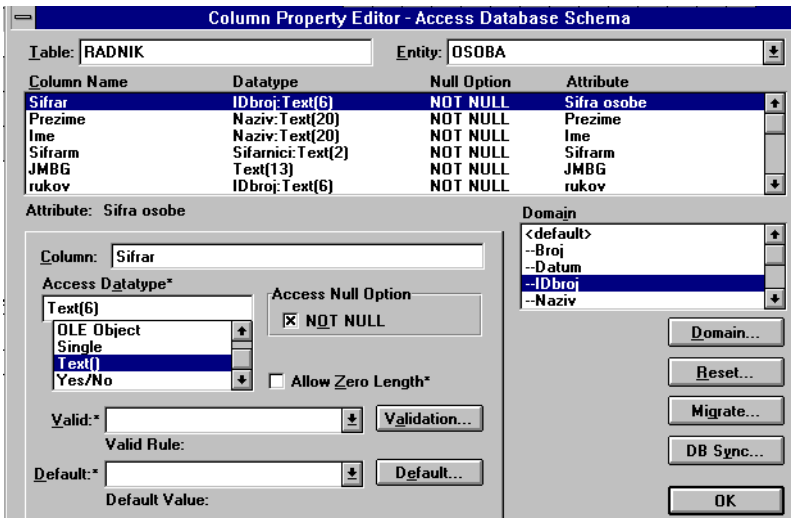
Na primeru sa slike 4.6. za tabelu RADNIK, korišćenjem editora za kolone, biće izvršene izmene za default vrednosti (slika 4.7).

Definisanje osobina kolona

Osobine kolona se definišu korišćenjem editora za definisanje kolona (Column property Editor) gde se mogu praviti izmene nad default vrednostima kolona (slika 4.7).

U okviru ovog editora prikazuju se ime selektovanog entiteta i ime odgovarajuće tabele, a u sledećem nivou spisak kolona sa definisanim osobinama. U donjem delu za izabranu kolonu definišu se osobine kolone, i to:

- tip podatka,
- domen,
- validacija,
- default vrednost.



Slika 4.7. Editor za opisivanje kolona tabele RADNIK

Kad je reč o dokumentu "Karton isplata", definisane su osobine kolona za tabelu RADNIK i prikazane na slici 4.7.

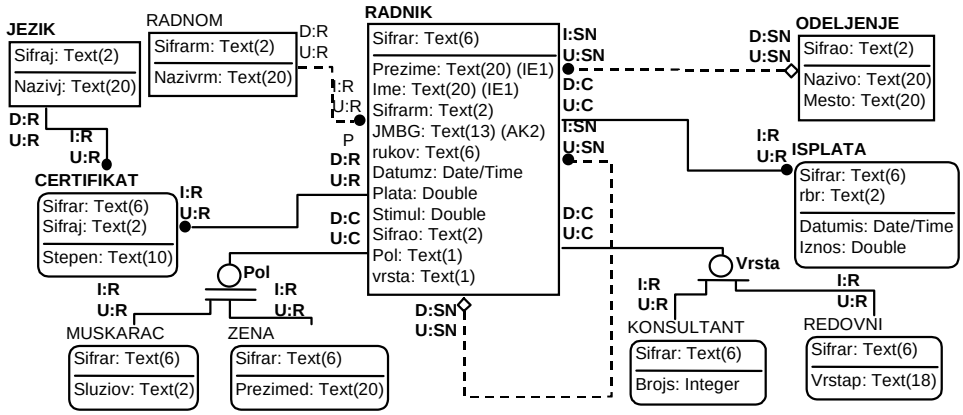
U okviru tipa podataka (Datatype) prikazani su: domen (npr., IDbroj) i posle dve tačke- tip i veličina domena ŠText (6)Ć, pa se kompletira sledeći izgled: IDbroj:Text (6)**.**

Ono što je sada interesantno sa slike 4.7. jeste osobina kolone Access Datatype gde se definiše tip podatka. U tabeli 4.1. prikazani su MS ACCESS tipovi podataka koji su dostupni u okviru ERwin-a za dodelu pojedinim kolonama.

Tabela 4.1. MS ACCESS tipovi podataka

Tip podatka	Opis tipa podatka
Byte	Podtip od numeričkog tipa podatka Number.
Counter	Broj koji se automatski dodeljuje i ne menja se.
Currency	Novčani iznos koji predstavlja broj sa četiri decimale.
Data/Time	Datum i vreme.
Double	Podtip od numeričkog tipa podatka Number sa pokretnim zarezom duple preciznosti.
Integer	Podtip od numeričkog tipa podatka Number. Predstavlja celobrojnu vrednost i zauzima 2 bajta.
Long Integer	Podtip od numeričkog tipa podatka. Predstavlja celobrojnu vrednost i zauzima četiri bajta.
Memo	Obimni tekst od 32000 karaktera.
OLE Objecat	Objekat tipa slika, zvuk, animacija.
Single	Podtip od numeričkog tipa podatka Number sa pokretnim zarezom obične preciznosti.
Text	Bilo koji tekst dužine do 255 znakova.
Yes/No	Logički tip podatka (True/False).

Na osnovu definisanih osobina kolona (slika 4.7) na slici 4.8. prikazan je fizički model za primer dokumenta "Karton isplata" sa definisanim tipovima i veličinama podataka za odgovarajuće kolone.



Slika 4.8. Fizički model podataka sa definisanim tipovima i veličinom podataka

Grafička prezentacija data na slici 4.8. u tablici 4.2. prikazana je za primer dokumenta "Karton isplata" kao uporedna tabela.

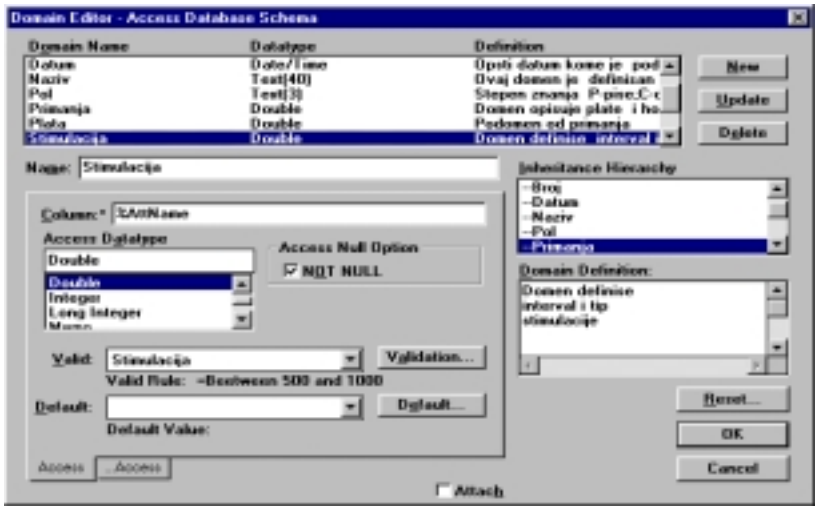
Tabela br.4.2. Uporedna tabela.

ENTITETI	TABELE	ATRIBUTI	KOLONE
ISPLATA	ISPLATA	Sifra osobe(PK)(FK)	Sifrar Text(6)NOT NULL
		rbr(PK)	rbr Text(2)NOT NULL
		Datum isplate	Datumis Date/Time
		Iznos	Iznos Double NOT NULL
JEZIK	JEZIK	Sifra jezika(PK)	Sifraj Text(2) NOT NULL
		Naziv jezika	Nazivj Text(20) NOT NULL
KONSULTANT	KONSULTANT	Sifra osobe(PK)(FK)	Sifrar Text(6) NOT NULL
		Broj sati	Brojs Integer NOT NULL
MUSKARAC	MUSKARAC	Sifra osobe(PK)(FK)	Sifrar Text(6) NOT NULL
		Sluzio vojsku	Sluziov Text(2) NOT NULL
ODELJENJE	ODELJENJE	Sifra deljenja(PK)	Sifrao Text(2) NOT NULL
		Naziv odeljenja	Nazivo Text(20) NOT NULL
		Mesto	Mesto Text(20) NOT NULL
OSOBA	RADNIK	Sifra osobe (PK)	Sifrar Text(6) NOT NULL
		Prezime (IE1)	Prezime Text(20)NOT NULL
		Ime(IE1)	Ime Text(20) NOT NULL
		Sifrarm(FK)	Sifrarm Text(2) NOT NULL
		JMBG(AK2)	JMBG Text(13) NOT NULL
		rukov(FK)	rukov Text (6) NOT NULL
		Datum zaposlenja	DatumzDate/TimeNOT NULL
		Plata	Plata Double NOT NULL
		Stimulacija	Stimul Double NOT NULL
		Sifra odeljenja(FK)	Sifrao Text(2)
		Pol	Pol Text(1) NOT NULL
		Vrsta	vrsta Text(1)
RADNO MESTO	RADNOM	Sifrarm(PK)	Sifrarm Text(2) NOT NULL
		Nazivrm	Nazivrm Text(20) NOT NULL
REDOVNI	REDOVNI	Sifra osobe(PK)(FK)	Sifrar Text (6) NOT NULL
		Vrsta posla	Vrstap Text (18)
CERTIFIKAT	CERTIFIKAT	Sifra osobe(PK)(FK)	Sifrar Text(6) NOT NULL
		Sifra Jezika(PK)(FK)	Sifraj Text(2) NOT NULL
		Stepen znanja	Stepen Text(10) NOT NULL
ZENA	ZENA	Sifra osobe(PK) (FK)	Sifrar Text (6) NOT NULL
		Prezime devojacko	Prezimed Text(20) NOT NULL

Sledeći korak je da se definišu osobine domena. Pritiskom na tipku Domain... (slika 4.7) prelazi se u novu formu gde se definišu osobine domena (slika 4.9).

Definisanje osobina domena

U okviru aktivnosti "2.4.3. Identifikacija poslovnog domena" identifikovani su domeni za primer dokumenta "Karton isplata", a sada treba pokazati tzv. domen editor pomoću koga se definišu osobine domena (slika 4.9). Editor omogućuje da se definiše domen, i to: tip domena, NULL opcija, validaciona pravila, default vrednost kao nasleđivanje hijerarhije domena (definisanje roditeljskog domena). U ovom editoru definišu se ograničenja nad standardnim domenom i pri tom određuju: tip podatka (tablica 4.2) i dužina podatka [npr., Text (30)].

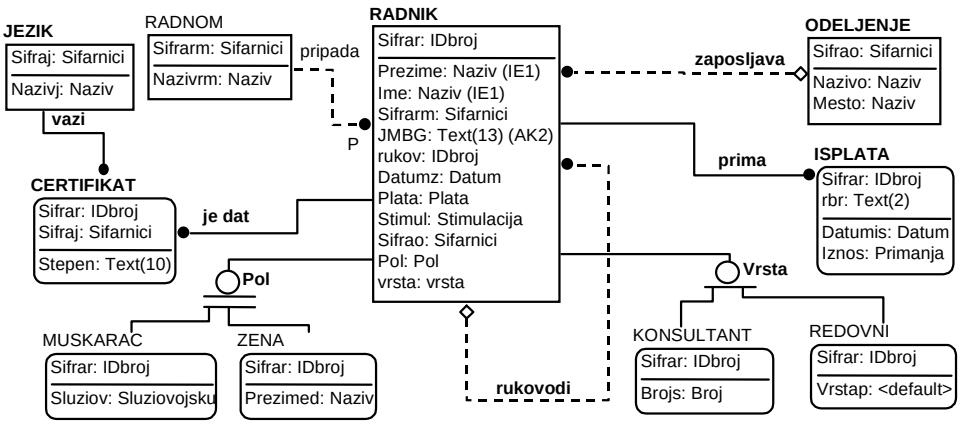


Slika 4.9. Domen editor

Dodeljivanjem **NULL** opcije definiše se Null vrednost polja, tj. ako je NOT NULL polje mora uvek da ima i neku vrednost; ako je NULL - ne mora.

Na slici 4.10. prikazan je fizički model podataka sa definisanim atributima i njima definisanim odgovarajućim domenima.

Kao što se može videti na slici 4.10, tabele su opisane kolonom i odgovarajućim nazivima domena. Može se primetiti da domen može, a ne mora da ima isto ime kao kolona. Domeni prikazani na slici 4.10. u tablici 4.3. detaljno su opisani.



Slika 4.10. Fizički model podataka sa definisanim domenima

Tablica br.4.3. Opis domena za primer "Karton isplata"

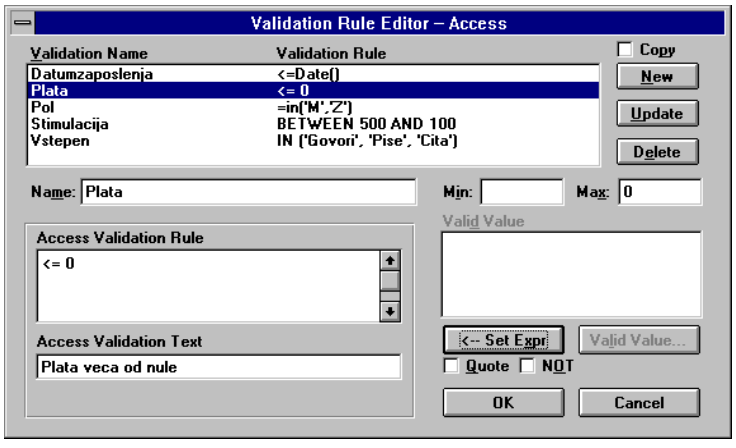
DOMEN	TIP I VELIČINA	OPIS DOMENA
<default>	Text (18)	Default vrednost definisana u ERwin-u
Broj	Integer	Domenom Broj definišu se svi brojevi
Datum	Date/Time	Podrazumevajuća vrednost sistemskog datuma
IDbroj	Text (6)	Jedinstven paralelni sistem označavanja
Naziv	Text (20)	Definiše atribute koji opisuju nazive
Primanja	Double	Domen opisuje plate i honorare
Plata	Double	Poddomen od domena primanja
Pol	Text (1)	Domen definisan IN listom vezan za definisanje potpune strukture (M- Muski Z-Zenski)
Sifarnici	Text (2)	Domen opisuje sifarnike
Sluziovojsku	Text (2)	Domen definisan IN listom (DA, NE)
Stepen	Text (10)	Stepen znanja jezika definisan IN listom (P,G,C)
Stimulacija	Double	Domen definiše interval i tip stimulacije
vrsta	Text (1)	Domen definisan IN listom vezan za definisanje nepotpune strukture

Pritiskom na dugme Validation... (slika 4.9) prelazi se na sledeći nivo gde se definišu validaciona pravila (slika 4.11), tj. definiše se ograničenje nad vrednošću domena.

Definisanje validacionih pravila

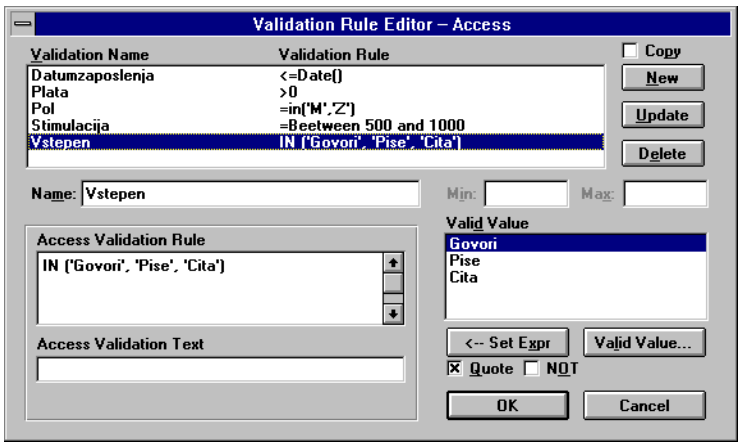
Validaciono pravilo ili vrednosno ograničenje je izraz kojim se utvrđuje *rang prihvatljivih vrednosti* za kolonu. Na slici 4.11. prikazana je lista validacionih pravila, tj. ograničenja nad vrednošću domenom (vrednost atributa) definisanih za model podataka "Karton isplata", i to:

- *Operatore poređenja* (<, >, =, >=, <=) definisan je u navedenom primeru za validacioni naziv Datum zaposlenja kao validaciono pravilo <= Date () i validacioni naziv Plata kao validaciono pravilo >0 (slika 4.11).



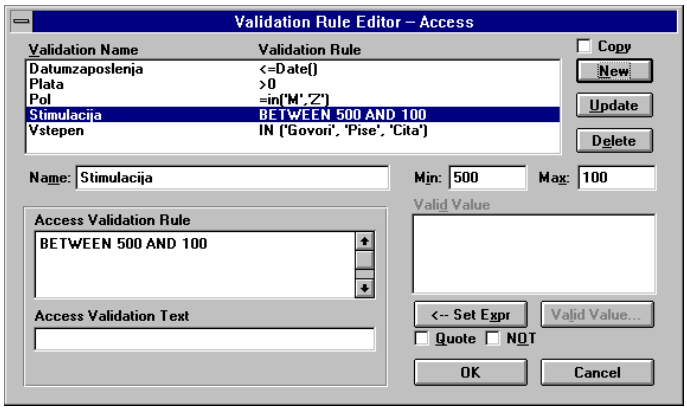
Slika 4.11. Editor validacionih pravila za primer operatora poređenja

- *IN lista* se formira kao lista od konstanti iz odgovarajućeg domena (slika 4.12), tj. eksplicitnim navođenjem svih dozvoljenih vrednosti. Za validacioni naziv Pol validaciono pravilo je IN ['M','Z'] i validacioni naziv Vstepen validaciono pravilo je IN ['Govori','Pise','Cita'].



Slika 4.12. Prikaz IN liste

- *BETWEEN opseg dozvoljenih vrednosti* gde atribut može poprimiti samo uži skup vrednosti iz domena; npr., za validacioni naziv Stimulacija definiše se validaciono pravilo BETWEEN 500 and 1000 (slika 4.13).



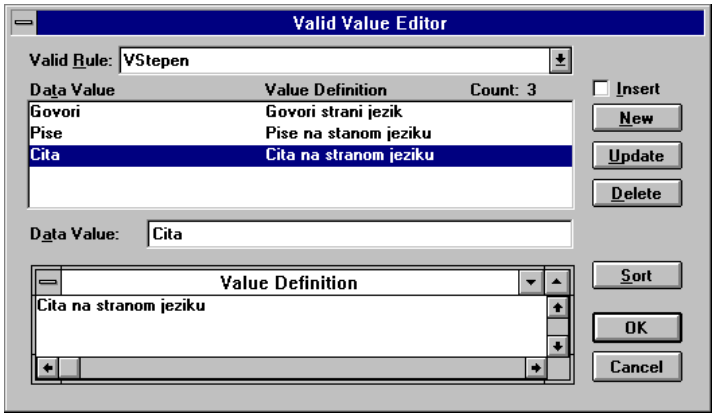
Slika 4.13. Prikaz BETWEEN operatora

Ako se pritisne dugme Valid Value (slika 4.13) prelazi se u novi editor, gde se definišu validacione vrednosti (slika 4.14).

Definisanje validacionih vrednosti

U ovom editoru kreira se fiksirana lista svih dozvoljenih vrednosti koje se mogu memorisati u kolonu i dodeliti validacionom pravilu. Kao primer dokumenta "Karton isplata" treba definisati validacione vrednosti (slika 4.14) za:

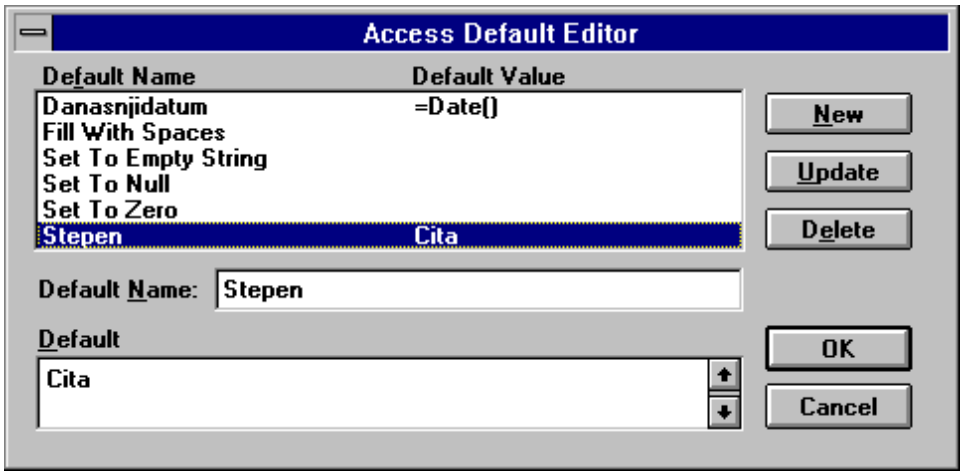
- validaciono pravilo Vstepen kao Govori, Pise , Cita i
- validaciono pravilo Pol kao M i Z.



Slika 4.14. Editor za definisanje validacionih vrednosti

Definisanje default vrednosti

Ako se pritisne tipka Default... (slika 4.9) prelazi se na definisanje default vrednosti (slika 4.15). U default editoru kreira se default vrednost koja se automatski dodeljuje za izabranu kolonu. Na slici 4.15. prikazane su definisane default vrednosti za definisan naziv Stepen default vrednost Cita.



Slika 4.15. Editor za definisanje default vrednosti

Trigeri i procedure

Trigeri predstavljaju imenovane blokove SQL koda koji su prekompajlirani i memorisani da bi ubrzali postavljanje upita, brže izvršili validaciju nad podacima ili neku vrlo često korišćenu operaciju. Najveća prednost je u tome da se jednom programirani blokovi SQL-a prenose na različite platforme bez modifikacija.

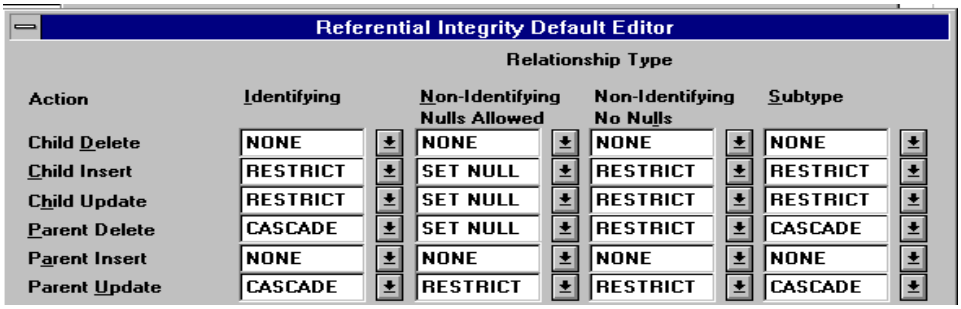
Dakle, trigeri su imenovani skupovi prekompajliranih SQL izraza 'memorisanih na serveru' koji se izvršavaju kada se pojavi odgovarajući događaj; oni govore SUBP kako da izvrši ubacivanje, ažuriranje ili brisanje u tabeli, i to onako kako je zamišljeno.

Trigeri referencijalnog integriteta (u daljem tekstu: RI trigeri) posebna su vrsta trigera koji se koriste da bi se održao integritet između dve tabele između kojih je uspostavljena relacija, odnosno oni govore SUBP šta da učini sa redom u onoj drugoj tabeli u kojoj je preneseni ključ jednak primarnom ključu u prvoj tabeli ako se u prvoj tabeli briše, ažurira ili dodaje novi red.

ERwin kreira RI trigere automatski i automatski dodeljuje podrazumevani RI triger svakom entitetu u vezi. SQL kod RI trigera se generiše na osnovu tri kriterijuma:

- vrste pravila referencijalnog integriteta koji se primenjuje na vezu (RESTRICT, CASCADE, SET NULL,SET DEFAULT ILI NONE)
- vrste veze, tj. da li je identifikujuća, neidentifikujuća ili podtip veze;
- vrste uloge entiteta u vezi, tj. da li je tabela 'dete' ili 'roditelj'.

Pritiskom na tipku Referential Integrity Default... (slika 4.3) prelazi se na editor za definisanje default vrednosti referencijalnog integriteta (slika 4.16).



Slika 4.16. Default Editor pravila referencijalnog integriteta

Default editor pravila referencijalnog integriteta omogućuju da se promeni pravilo referencijalnog integriteta dodeljeno određenoj vezi jednostavnim odabirom. Ipak ono što daje snagu ERwin-u je mogućnost da korisnik definiše odgovarajuće šablone za kreiranje RI trigera, odnosno da sam napiše svoj SQL kod koji će biti dodeljen vezi. Takođe, mogu se ugraditi trigeri koji izvršavaju posebne zadatke, kao što su izračunavanja vrednosti nekih atributa na osnovu drugih atributa.

Procedure u ERwin-u su skup prekompajliranih SQL izraza, sličnih trigerima, s tom razlikom što *se ne izvršavaju zavisno od događaju već po pozivu*. Osim toga, procedure mogu da prihvate parametre, ili mogu da vrate parametre, vrednosti, poruke ili, pak, da pozovu neku drugu proceduru. Naravno, procedure treba čitave da se pišu, jer ne postoji ugrađen šablon. One, takođe, mogu da se povezuju sa čitavim šemama, a ne samo sa entitetima.

Na osnovu prethodno izvedenih aktivnosti (slika 4.2) u sledećem koraku potrebno je definisati indekse.

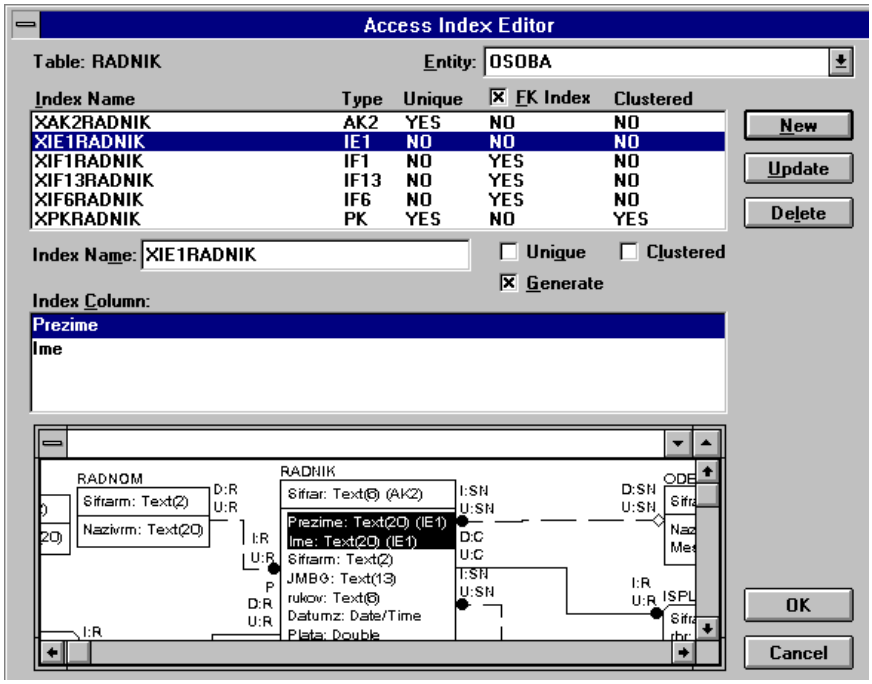
Aktivnost
3.1.3.Definisanje indeksa

U tabelama, podaci se smeštaju po redosledu njihovog unošenja. Postupak pretraživanja zahtevanog podatka u tabeli može dugo da potraje, pa se za rešavanje problema pretraživanja koriste specijalni tipovi tabela pod imenom *indeksi,* u kojima se nalaze adrese redova.

Npr., za tabelu RADNIK automatski se formira indeksna tabela vezana za primarni ključ (PK) nad kolonom Sifrar (slika 4.17).

Mogu se kreirati i separatni indeksi za svaku kolonu u tabeli ako je česta potreba za pretraživanjem pojedinačnih vrednosti memorisanih u koloni.

Takođe, može se formirati i indeks nad više kolona kao, npr., nad kolonama: Prezime i Ime u tabeli RADNIK.



Slika 4.17. Prikaz Access indeks editora

Kada se generiše fizička šema baze podataka, ERwin automatski kreira pojedine indekse za:

- primarne ključeve nad svakom tabelom (PK),
- alternativne ključeve (AKx),
- prenesene ključeve (IF),
- inverzne ključeve (IE),
- kolone koje imaju često pretraživanje.

Korišćenjem Index editora, u okviru naziva indeksa definiše se sledeća struktura oznake:

"X" + "PK" (ili "AK", " IF" ili "IE" + "n") + Naziv tabele entiteta.

Tablica br.4.4. Indeksi za tabele iz primera "Karton isplata"

Naziv tabele	Izraz za indeks
RADNIK	XPKRADNIK Unique Index Columns: Sifrar XAK2RADNIK Unique Index Columns: JMBG XIE1RADNIK Index Columns: Prezime Ime XIF1RADNIK Index Columns: Sifrao XIF13RADNIK Index Columns: Sifrarm XIF6RADNIK Index Columns: rukov
ISPLATA	XPKISPLATA Unique Index Columns: Sifrar rbr XIF2ISPLATA Index Columns: Sifrar
KONSULTANT	XPKE/9 Unique Index Columns Sifrar XIF8E/9 Index Columns: Sifrar
JEZIK	XPKJEZIK Unique Index Columns Sifraj
MUSKARAC	XPKMUSKARAC Unique Index Columns: Sifrar XIF11MUSKARAC Index Columns: Sifrar
ODELJENJE	XPKODELJENJE Unique Index Columns: Sifrao
RADNOM	XPKRADNOM Unique Index Columns: Sifrarm
REDOVNI	XPKE/8 Unique Index Columns: Sifrar XIF9E/8 Index Columns: Sifrar
CERTIFIKAT	XPKZNA JEZIK Unique Index Columns: Sifrar Sifraj XIF3ZNA JEZIK Index Columns: Sifrar XIF4ZNA JEZIK Index Columns Sifraj
ZENA	XPKZENA Unique Index Columns: Sifrar XIF12ZENA Index Columns: Sifrar

Npr., za tabelu RADNIK i odgovarajuću kolonu formira se, po default-u sledeća struktura indeksa:

- Sifrar (PK) ima indeks XPKRADNIK;
- Prezime (AK1)+Ime (AK1) ima indeks XAK1RADNIK;
- JMBG (AK2) ima indeks XAK2RADNIK;
- Sifrao (IF1) ima indekS XIF1RADNIK;
- Rukov (IF6) ima indeks XIF6RADNIK.

Kako su alternativni ključevi, takođe, jednoznačni, a instance kolona: Prezime i Ime se mogu ponoviti, to isključivanjem opcije "Unique" AK1 prelazi u IE1 (slika 4.17).

Primarni indeks XPKRADNIK ima uključenu opciju "Clustered" koja omogućuje da se podaci u tabelama fizički smeštaju uz indekse, čime se poboljšavaju performanse upita.

U tablici 4.4. prikazani su formirani indeksi u ERwin-u za primer fizičkog modela podataka "Karton isplata".

Na osnovu prethodno izvedenih aktivnosti (slika 4.2) u sledećem koraku potrebno je definisati način upravljanja podacima.

Aktivost 3.1.4.Definisanje načina upravljanja podacima

Aktivnost *"3.1.4. Definisanje načina upravljanja podacima"* je bitna funkcija organizacije podataka koja obuhvata:

- *skladištenje,* pod čime se podrazumevaju kontrola redosleda upisivanja podataka, načini pristupa i adresiranja podataka i način fizičkog predstavljanja podataka;
- *ponovno pristupanje,* tj. određivanje mesta nalaženja podataka (adresiranje), formiranje podataka i određivanje redosleda podataka;
- *kontrolu,* tj. unutrašnje regulisanje toka odvijanja postupka upravljanja podacima, određivanje prava pristupa podacima, čime osigurava podatke da ne dođe do gubitaka i obezbeđuje ažurnost podataka na sistemu.

Ova aktivnost se ostvaruje u okviru klijent/server-arhitekture i treba da se nalazi na jednoj ili više hardverskih platformi, gde server izvodi zajedničke servise za klijenta, tj. upravlja podacima preko ugrađenih poslovnih pravila i centalizovanih procedura. Klijent ograničeno i kontrolisano opterećuje server, distribuirano procesira informacije i zadržava samostalnost vezanu za rad u lokalu (pogledati detalje u poglavlju 4.3).

Upravljanje podacima treba da podrži:

- integritet baze podataka,
- transakcionu obradu podataka,
- sigurnost podataka,
- zaključavanje podataka i
- oporavak baze podataka.

Integritet baze podataka

Integritet baze podataka treba da omogući tačnost, korektnost i konzistentnost podataka i da označi probleme zaštite baze podataka od pogrešnog ažuriranja (pogrešnih ulaznih podataka, greški operatera i programera, sistemskih otkaza).

U ovom delu je deo o problematici vezanoj za očuvanje integriteta pri izvodenju jedne transakcije. O transakcionoj obradi podataka biće više reči kasnije.

U okviru integriteta baze podataka treba definisati odgovarajuća pravila integriteta kojima se određuje koje uslove podaci u bazi podataka treba da zadovolje, kada se vrši provera i koje akcije treba preduzeti ako definisani uslovi nisu zadovoljeni.

Integritet baze podataka je definisan pravilima integriteta na serveru baze podataka i sadrži:

- *rečnik podataka,* tj. odgovarajuću meta-bazu podataka;
- *integritet domena* kojim se definiše dozvoljeni skup vrednosti;
- *integritet tabele* gde svaki red u tabeli mora da bude jedinstven;
- *referencijalni integritet* ili integritet relacija;
- *autoreferencijalni integritet* gde se u istoj tabeli definišu spoljni i originalni ključ;
- *poslovna pravila* ili tzv. korisnička pravila i ograničenja.

Rečnik podataka je baza podataka o bazi podataka (meta-baza podataka) i u njoj su opisi tabela, relacija između tabela, kao i svi objekti tipa formi, izveštaja, upita, makroa i programa koji se čuvaju u meta-bazi, koja se logički predstavlja kao i sama baza podataka.

U rečnik podataka smeštaju se pravila integriteta koja su definisana pomoću nekog jezika visokog nivoa. Rečnik podataka je korektan pristup; no, međutim, neki SUBP ne podržavaju integritet na taj način, jer se podrška prebacuje u aplikacione programe, odnosno ostavlja se mogućnost da programer, uz pomoć generatora aplikacija, sam vodi računa o integritetu baze podataka.

Integritet domena je dozvoljeni skup vrednosti, gde svaka vrednost u polju mora da pripada domenu te kolone. Dakle, red se nalazi u tabeli ako su vrednosti u svim kolonama tog reda u domenu kolona, a integritet domena kolone je održan ako zapis može biti upisan u tabelu samo u slučaju da tip podatka u svakoj koloni odgovara dozvoljenom tipu za datu kolonu.

Integritet tabela je ispunjen ako je svaki red u tabeli jedinstven, tj. ako postoji jednoznačna identifikacija reda koji se definiše primarni ključem, u kojem se može nalaziti jedna ili više kolona. Kolona čija se vrednost koristi za identifikaciju ostalih kolona naziva se ključem. Postoje dve vrste ključeva: primarni i sekundarni.

Primarni ključ jednoznačno određuje red; stoga i ne mogu postojati dva reda sa istim vrednostima ključa. Ako u redu nema kolone koja bi jednoznačno odredila neki red, mogu se naći dva ili više koji zajedno određuju jednoznačnost. Tada se govori o *združenom* ključu. Sekundarni ključ je kolona po kojoj se vrši pretraživanje. Sekundarni ključ ne mora biti jednoznačan.

Referencijalni integritet ili integritet relacija omogućuje vezu između raznih kolona i tabela. Vrednosti u jednoj koloni ili grupa kolona *referišu* se vrednostima u drugoj koloni ili skupu kolona i pri tom se definiše za tabelu 'dete' preneseni ključ (foreign key), a za tabelu 'roditelj' - primarni ključ (primary key).

Autoreferencijalni integritet je definisan spoljnim i orginalnim ključem u istoj tabeli, gde je jedna kolona u tabeli povezana sa vrednostima u drugoj koloni iste tabele.

Poslovna pravila ili tzv. korisnička pravila su pravila za očuvanje integriteta podataka koja zadaje korisnik; npr., zabrana izmene podataka van radnog vremena. Uskladištena procedura je, takođe, primer za poslovna pravila, gde se definišu pravila za prevođenje skupa SQL iskaza i programski iskazi za kontrolu toka obrade. Takođe, poslovnim pravilima se deklarišu promenljive, kao i odgovarajući operatori za dodelu vrednosti. Uskladištena procedura automatski se "ispaljuje" pod određenim uslovima i nalazi se u serveru baze podataka, što omogućuje da se saobraćaj u mreži svede na minimum.

Ograničenjima se definiše integritet podataka između tabela; ta ograničenja su vezana za rang validnih vrednosti.

Transakciona obrada podataka

Transakcija je operacija kojom se izvodi serija izmena nad jednom ili više tabela, tj. transakcija je izvršenje neke logičke jedinice rada korisnika baze podataka. Osnovni cilj baze podataka je da omogući efikasnu obradu transakcija. Više izvršenja istog programa mogu se u sistemu odvijati konkurentno, svako od njih predstavlja transakciju.

Skup aktivnosti nad bazom podataka koje se izvršavaju po principu "sve ili ništa", tj. ili su sve aktivnosti uspešno obavljene ili je baza podataka ostala nepromenjena, atomski je skup aktivnosti. Očigledno je da "logička jedinica rada" predstavlja taj atomski skup aktivnosti, tj. "atomsku transakciju".

Definišu se dva tipa transakcija:

- DML transakcija kojom se povezuje veći broj DML naredbi, što se definiše još i kao eksplicitna transakcija i
- DDL transakcija kojom se izvodi samo jedna naredba, i to kao "automatska" (implicitna) transakcija.

Drugim rečima, transakcija predstavlja izvršenje jednog skupa operacija (npr., SELECT, UPDATE, INSERT i DELETE) nekog programa koji počinje sa prvom izvodljivom DML ili DDL naredbom (BEGIN TRANSACTION), a završava se:

- operacijom COMMIT (ako je ceo skup operacija uspešno izvršen),
- ROLLBACK (ako ceo skup operacija nije uspešno izvršen),
- DDL naredbom,
- važećom greškom (slično deadlock),
- log off,
- greškom mašine.

Nakom izvršenja jedne transakcije, sledeća SQL naredba automatski startuje drugu po redu transakciju.

Drugim rečima, na nivou SQL naredbi upravljanje transakcijom se upravlja sa:

- naredbom COMMIT, kojom se:
  - definišu stalne izmene u bazi podataka,
  - brišu svi CHECKPOINT u transakciji,
  - definiše kraj transakcije,
  - realizuje tzv. transakciono zaključavanje i
  - omogućuje eksplicitni kraj transakcije;
- naredbom CHECKPOINT za duge transakcije kojom se deli transakcija u male 'porcije', tj. čuva rad u transakciji u jednoj tački sa opcijom za kasniji COMMIT;
- naredbom ROLLBACK, kojom se poništavaju sve izmene u tekućoj transakciji, odnosno kojom se:
  - brišu svi CHECKPOINT u transakciji i
  - oslobađaju sva transakciona zaključavanja;
- naredbom ROLLBACK[WORK] TO CHECKPOINT kojom se poništava samo deo transakcije.

U radu sa bazom podataka koji se koriste u transakcijama definišu se dve operacije, i to:

- čitanje (read) sa baze podataka, korišćenjem naredbe SELECT i
- ispisivanje (write) u bazu podataka, korišćenjem naredbi INSERT, UPDATE i DELETE.

Jednim programom se može predstaviti sekvenca transakcija, iako je najčešće jedan program jedna transakcija.

Transakcije ne mogu biti "ugnježdene", tj. u jednom programu se operacija BEGIN TRANSACTION može izvršiti samo ako taj program nema nijednu drugu transakciju u izvršenju. Naredbe: COMMIT i ROLLBACK se izvršavaju samo ako je neka transakcija u izvršenju.

Transakcije moraju da prate i odgovarajuće izlazne poruke, koje ne smeju da se prenose direktno, već tek nakon planiranog završetka transakcije (COMMIT ili ROLLBACK).

Posebna pažnja se poklanja sistemu poruka u postupku oporavka baze podataka.

Poruke se ne prihvataju direktno, već se ulazne poruke stavljaju u ulazni red, a izlazne poruke u izlazni red. Pri planiranju završetka transakcije (COMMIT ili eksplicitni ROLLBACK) izvršava se:

- upisivanje log izlazne poruke,
- stvarno se prenose izlazne poruke i
- izbacuju se ulazne poruke iz ulaznog reda.

Pri neplaniranom ROLLBACK (izvršava se automatski ako dođe do neke greške u programu tipa "overflow", deljenja sa nulom i slično) dolazi do izbacivanja izlaznih poruka iz izlaznog reda.

Stoga su u MS ACCESS-u elementi transakcionog rada ugrađeni u okviru ACCESS BASIC, gde su mogući: definisani početak transakcije (BeginTrans), zapisivanje rezultata rada transakcije (CommitTrans) i vraćanje na stanje pre početka transakcije (Rollback).

Sigurnost podataka

Sigurnost podataka odnosi se na mehanizam zaštite podataka od neovlašćenog korišćenja, koji je ugrađen u SUBP. Zaštitom podataka se definiše koji subjekt zaštite (npr., Eremija) nad kojim tabelama (npr., RADNIK) može da izvede neku operacuju (npr., SELECT, UPDATE, INSERT i DELETE) i pod kojim uslovima (npr., samo za odeljenje 10).

Polazi se od činjenice da je korisnik vlasnik kreirane tabele i da samo on kao vlasnik može da je koristi osim ako i drugima ne dozvoli pristup. Za dodeljivanje privilegije korišćenja drugim korisnicima koristi se GRANT naredba koja se sastoji iz tri osnovne klauzule:

GRANT <Privilegija>

ON <tabele ili pogled>

TO <korisnik ili grupa korisnika>

[WITH GRANT OPTION];

Privilegije mogu biti :

- SELECT,
- UPDATE,
- INSERT,
- DELETE,
- INDEX,
- EXPAND (dodavanje atributa relacije),
- ALL (važi za sve navedene privilegije),
- RESOURCE (omogućuje korisniku kreiranje objekata baze podataka, kao što su: tabele, indeksi, klasteri),
- DBA (obavlja administrativne zadatke, kao što su: CREATE TABLESPACE i CREATE ROLLBACK SEGMENT),
- DBA privilegijom korisnik može definisati:
  - SELECT iz bilo koje tabele i pogleda,
  - CREATE objekata baze podataka za druge korisnike,
  - DROP drugih korisnika objekata baze podataka, uključujući tabele, sinonime i linkovane baze podataka,
  - GRANT varijante privilegija baze podataka,
  - CREATE public sinonima i linkovanje baze podataka,
  - EXPORT i IMPORT baze podataka.

WITH GRANT OPTION daje dozvolu davanja privilegija drugom korisniku.

Kada se, na primer, uvedu privilegije nad tabelom RADNIK, a onda treba da se nekom drugom da privilegija nad tom tabelom, i to korisniku PERI, piše se ovako:

```
GRANT SELECT ON RADNIK TO PERA;
```

Ako bi trebalo da se da nekom drugom privilegija za samo SELECT nad tabelom RADNIK, i

to korisniku VLADA piše se ovako:

```
GRANT SELECT ON RADNIK TO VLADA;
```

Ukoliko bi trebalo da se da nekom drugom privilegija - za samo UPDATE (PLATA, STIMUL) nad tabelom RADNIK, i to korisniku STEFAN piše se ovako:

```
GRANT UPDATE (PLATA, STIMUL) ON RADNIK TO STEFAN;
```

Privilegije se mogu davati i za poglede nad naredbama: SELECT, INSERT, UPDATE i DELETE.

Na primer, ako se ne želi da korisnik ALEMPIJE ima pogled na kolone: PLATA i STIMUL, onda je bolje da se privilegije ograničavaju na pogled, a ne na tabele.

Prvo se definiše pogled na tabelu RADNIK koja ne sadrži kolone: PLATA I STIMUL.

```
CREATE VIEW ZAPS AS
SELECT SIFRAR,PREZIME,SIFRARM,RUKOV,DATUMZ,SIFRAO
FROM RADNIK;
```

Za davanje privilegije korisniku ALEKSANDAR nad pogledom ZAPS piše se:

```
GRANT SELECT ON ZAPS TO ALEKSANDAR;
```

Može se definisati, kao primer, i pogled nad tabelom RADNIK koja daje podatke samo o onim zaposlenima sa istim brojem odeljenja koji ima i osoba koja koristi pogled.

```
CREATE VIEW IMERAD AS
SELECT *
FROM RADNIK
WHERE SIFRAO IN
(SELECT SIFRAO
FROM RADNIK
WHERE PREZIME = USER);
```

USER definiše ime prijavljenog korisnika. Ako CEBIC, koji je u odeljenju 10, pristupio pogledu može videti samo zaposlene u odeljenju 10. Ovim pogledom može se dati korisniku CEBIC privilegija da vrši, npr., ažuriranje na sledeći način:

```
GRANT SELECT, UPDATE
ON IMERAD
TO CEBIC;
```

Ako neki zaposleni pređe u drugo odeljenje (polje SIFRAO se menja), novi rukovodilac automatski dobija pristup podacima dok ga stari rukovodilac gubi.

Kada treba poništiti privilegije koristi se klauzula REVOKE.

Npr., ako se želi oduzeti privilegija korisniku ALEMPIJE za ubacivanje podataka u tabelu RADNIK piše se:

```
REVOKE INSERT
ON RADNIK
FROM ALEMPIJE;
```

Zaključavanje podataka

Zaključavanje podataka je mehanizam za upravljanje konkurentnim pristupom podacima i izvodi se kada korisnik pokuša da izvrši izmenu nad podacima u bazi podataka.

Zaključavanje se vrši prilikom:

- sintaksne analize,
- izvršavanja komandi i

- pristupanja redovima.

Postupak zaključavanja prestaje u dva slučaja:

- kada se transakcija završi (COMMIT/ROLLBACK) i
- kada se kursor zatvori (logoff).

Postupak zaključavanja tabela i redova je najvažniji deo održavanja konzistentnosti i integriteta baze podataka korisnika.

Osnovna podela zaključavanja je nad:

- tabelama rečnika podataka - DDL (Data Definition Language),
- tabelama korisnika podataka - DML.

DDL način zaključavanja kontroliše pristup bazi podataka i izvodi se automatski nad tabelama rečnika podataka. Ovaj način zaključavanja upravlja sledećim SQL naredbama:

CREATE TABLE...,ALTER TABLE..., DROP TABLE...i dr.

DML način zaključavanja upravlja pristupom podacima u korisničkim tabelama.

Zaključavanje naročito dolazi do izražaja kada je u pitanju uporedna obrada transakcija. Transakcije se izvode uporedo sa drugim transakcijama u sistemu ako više transakcija zahteva isti slog baze podataka.

Mogu se definisati dva načina zaključavanja, i to:

- zajedničko (shared),
- isključivo (exclusive)).

Shared(s) zaključavanje je istovremeno zajedničko zaključavanje tabela da bi se obezbedio upit nad konzistentnim podacima cele tabele (bez transakcione obrade) ili reda (sa transakcionom obradom). Ovakvo zaključavanje može izvesti više korisnika. Ovaj način zaključavanja onemogućuje druge korisnike da vrše promene nad podacima i stavljaju exclusive zaključavanje, ali ih ne ograničava pri tom da vrše upite.

Exclusive zaključavanje je isključivo zaključavanje tabela ili redova da bi se omogućilo ekskluzivno unošenje promena podataka tako što se onemogućavaju drugi da istovremeno unose promene, tj. onemogućavaju drugi da istovremeno stave bilo kakvo zaključavanje nad istom tabelom.

Oporavak baze podataka

Jedan od veoma važnih elemenata, vezanih za server baze podataka, jeste mogućnost *oporavka baze podataka (recovery),* što predstavlja povratak baze podataka u stanje pre softverskog ili hardverskog otkaza sistema. Razlozi za otkaz sistema mogu biti usled greške u operativnom sistemu, greške u programiranju, greške u samom SUBP-u ili, pak, padanja glave diska, nestanka struje i dr.

Tehnika redundantnog pamćenja podataka i tehnika oporavka baze su veoma kompleksne. Poznato je da su do sada korišćene jednostavne procedure koje su se bazirale na periodičnom kopiranju baze podataka u neku arhivsku memoriju i svih transakcija koje su se u međuvremenu dogodile, kao i na jednostavnom dupliciranju baze podataka. Iako jednostavne, ove metode imaju čitav niz nedostataka.

Proces oporavka izvodi se u globalu u tri koraka:

- periodično kopiranje (dump) baze podataka na eksternu memoriju;
- zapisivanje promena baze podataka u žurnal (tzv.log), i to:
  - stara vrednost u before image i
  - nova vrednost u after image;

- ako je došlo do otkaza sistema zato što je:
  - sama baza podataka oštećena, kada se izvodi oporavljanje baze podataka na osnovu arhivske kopije, ili
- baza dovedena (nije oštećena) u neko nekonzistentno stanje i pomoću log-a se poništavaju sve nekorektne promene, a same transakcije koje su ih proizvele se ponove.

Da bi se omogućio oporavak baze podataka potrebno je, pre svega, razmotriti koje su to vrste otkaza:

- planirani ROLLBACK, tj. narušavanje integriteta u nekoj od transakcija koje program sam otkriva;
- otkazi u transakciji (lokalna greška, npr., deljenje nulom);
- sistemski otkaz koji utiče na sve transakcije, ali ne oštećuje bazu podataka (npr., nestanak struje);
- fizičko oštećenje baze podataka.

Otkazi u transakciji

Ako se transakcija završi pre naredbe COMMIT ili eksplicitni ROLLBACK, neophodno je da se izvrši automatski ROLLBACK. Proceduru ROLLBACK obavlja Recavery Manager, poništavajući sve promene unazad kroz log, dok se u log-u ne dostigne slog koji označava početak transakcije.

U toku poništavanja može doći do otkaza i u tom slučaju ROLLBACK procedura mora startovati ponovo, pa je i to jedan od uslova da transakcija bude što kraća.

Sistemski otkaz bez oštećenja baze podataka

Sistemski otkaz prouzrokuje prestanak rada celog sistema, pa se mora izvesti oporavak baze podataka pretraživanjem celog log-a i identifikovanjem transakcija koje su otpočete, a nisu završene.

Da bi se skratilo pretraživanje, uvode se tzv. tačke ispitivanja (checkpoint) na sledeći način:

- upisuje se sadržaj bafera za log na log;
- upisuje se "checkpoint record" na log;
- upisuje se sadržaj bafera baze podataka u bazu podataka;
- upisuje se adresa "checkpoint" rekord-a na "restart file".

Sadržaj bafera se prenosi tek kad se oni napune. "Checkpoint record" sadrži listu svih transakcija koje su aktivne u trenutku uzimanja "checkpoint-a" i za svaku transakciju adresu sloga u log-u kojem je poslednjem pristupio.

Algoritam oporavka izvodi se tako što se prvo formiraju liste: "undo", u kojoj su sve transakcije zapisane u "checkpoint "-u, i "redo", koja je u početku prazna.

Pretraživanje po log-u počinje od "checkpoint record "-a na sledeći način:

- svaka transakcija za koju se pronađe BEGIN TRANSACTION stavlja se u "undo" listu;
- svaka transakcija za koju se izvodi COMMIT smešta se u "redo" listu.

Ovakav način memorisanja omogućuje Recavery Manager-u da unazad izvršava "undo" transakciju, a potom unapred "redo" odgovarajuću transakciju.

Kako u intervalu upisivanja u samu bazu (Log Write-Ahead) i na log može doći do otkaza, a da bi se obezbedio oporavak baze podataka, prvo se vrši upisivanje na log.

Otkaz diska

Ako bi došlo do otkaza diska, baza podataka bi se oporavljala na osnovu arhivske kopije i log-a, obnavljajući sve transakcije od trenutka uzimanja kopije do otkaza. To je veoma dugotrajan proces i izvodi se, obično, kada nijedna transakcija nije aktivna.

Može se definisati i tzv. inkrementni dump, odnosno na kopiju se prenose samo oni slogovi koji su pretrpeli promene poslednjeg uzimanja kopije.

Na osnovu definisanog fizičkog dizajna (slika 4.1) u sledećem koraku pristupa se generisanju šeme baze podataka.

Aktivnost
3.2. Generisanje šeme baze podataka

Aktivnost *"3.2. Generisanje šeme baze podataka"* izvodi se na osnovu prethodno urađene aktivnosti *"3.1. Definisanje fizičkog dizajna"*, tj. definisanih elemenata za izabrani SUBP. Šemu baze podataka čine fizičke tabele, kolone i relacije, koje se, kao što je rečeno u prethodnom poglavlju, u CASE alatu automatski generišu iz logičkog modela. Takođe, u prethodnom poglavlju pokazani su i automatsko kreiranje default tipova podataka za svaku generisanu kolonu i način izmene specifikacija kolona (domen i validacija podataka).

Proces generisanja šeme baze podataka može se izvesti na dva načina:

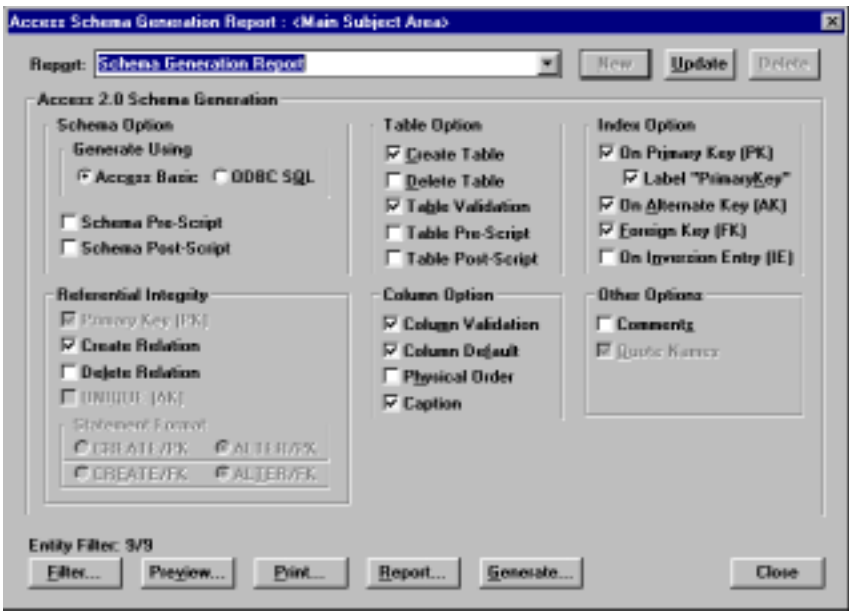
- direktnim inženjerstvom (forward engineering) i/ili
- inverznim inženjerstvom (reverse engineering).

Proces generisanja šeme baze podataka iz logičkog modela podataka naziva se *direktni inženjering*. Kada se generiše šema baze podataka, entiteti prelaze u tabele, atributi u kolone, a veze u relacije i definišu se referencijalni integritet, trigeri, procedure, indeksi i druge osobine koje podržava izabrani SUBP.

Dakle, da bi se generisala baza podataka potrebno je, prvo, izabrati odgovarajuću ciljnu platformu (SUBP) i potom se logovati na nju (slika 4.3). Kada korisnik loguje na izabranu platformu, ERwin kreira aktivnu bidirekcionu vezu sa sistemskim katalogom izabranog servera koja omogućava direktno kreiranje baze podataka. MS ACCESS koristiće kao test SUBP za generisanje šeme baze podataka za primer dokumenta "Karton isplata".

Na slici.4.18. prikazani su za MS ACCESS bazu podataka elementi šeme koja će se generisati. Šema je struktura baze podataka definisana DDL (Data Definition Language) skript-fajlom koji će u sledećem poglavlju biti detaljno objašnjen.

Inverzni inženjering predstavlja proces dobijanja fizičkog i logičkog modela iz postojeće fizičke baze podataka. Mnogi će se zapitati: "Zašto je to potrebno, kad već postoji kreirana baza koja odrađuje jedan posao". Mora se istaći da je najteža i najskuplja faza - održavanje baze podataka (videti aktivnost: *"4.3.Održavanje"*). Pod održavanjem se podrazumevaju dogradnja i proširivanje prema zahtevima korisnika. Obično, po završetku "izgradnje" aplikacije "zaboravlja" se na dokumentaciju, što za posledicu ima otežan proces održavanja baze podataka. Odlaskom programera iz firme, održavanje postaje problem. Da bi se to prevazišlo i da bi se mogla uraditi nadgradnja (uzimajući u obzir i zahteve standarda ISO 9000), postupkom inverznog inženjerstva, od fizičke baze podataka dolazi se do polaznog fizičkog i logičkog modela, tj. dokumentacije koja nedostaje.



Slika 4.18. Access šema za generisanje BP

ERwin omogućava inverzni inženjering sa dvanaest SQL SUBP upravljanja bazom podataka (AS/400, DB2, Informix, Ingres, NetWare SQL, ORACLE, Progress, Rdb, SQL Base, SQL Server, SYBASE i WATCOM SQL) i šest desktop-sistema za upravljanje bazom podataka (Microsoft Access, FoxPro, Clipper, dBase III, dBase IV i Paradox, prikazanih na slici 4.3).

Dakle, inverznim inženjerstvom iz postojeće fizičke baze podataka kreiraju se fizički i logički model i zatim uz pomoć CASE alata i editora redizajnira logički model podataka. Jedan od proizvoda ovog redizajna je i sistemska dokumentacija. Nakon kreiranja logičkog modela podataka može se, postupkom reinženjeringa poslovnih procesa, struktura baze prebaciti na drugu ciljnu platformu (drugi format baze podataka). U ERwin-u se može na dva načina izvršiti inverzni inženjering fizičke baze podataka:

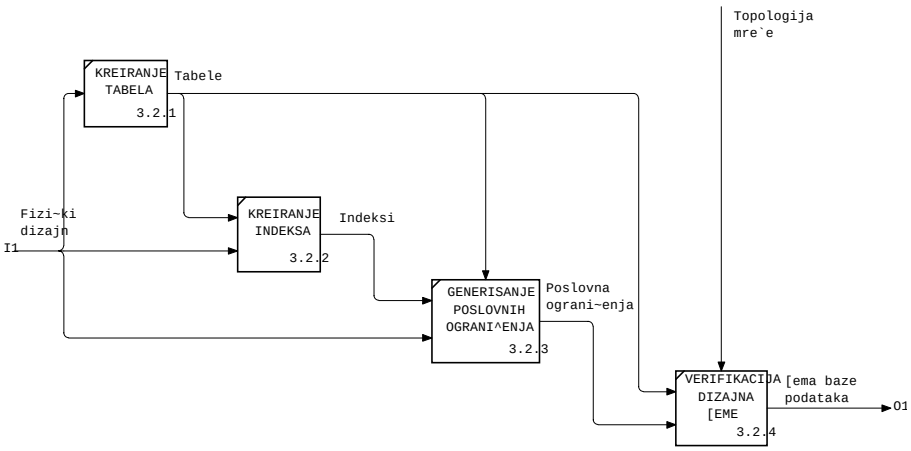
- direktnom vezom sa ciljnom platformom i
- otvaranjem i čitanjem SQL skript-datoteke.

No, bilo koji način da se izabere, automatski se kreira novi fizički model podataka. Ako se koristi *direktna veza* sa ciljnom platformom, SQL baza podržava deklaraciju prenesenih ključeva (FOREIGN KEYS). ERwin prepoznaje jake i slabe veze i podrazumevajuće "rolename" za generisani model podataka. Za DB2, SQL server i SYBASE, ERwin "izvlači" sve važne informacije o modelu podataka, izuzev podtipova, koji nisu podržani ni od jednog SQL sistema za upravljanje bazom podataka.

Ako se izabere opcija SQL skript-datoteke, onda ERwin generiše atribute gde primarni ključ nije u prvoj koloni. Takođe, može se pratiti čitav postupak inverznog inženjeringa baze, kao i "on-line" ispravljanje grešaka SQL skrpit-datoteke.

Na slici 4.19. prikazan je dekompozicioni dijagram za aktivnost *"3.2. Generisanje šeme baze podataka"* koja se sastoji od sledećih aktivnosti:

- Aktivnost 3.2.1. Kreiranje tabela,
- Aktivnost 3.2.2. Kreiranje indeksa,
- Aktivnost 3.2.3. Generisanje poslovnih pravila,
- Aktivnost 3.2.4. Verifikacija šeme baze podataka.



Slika 4.19. Dekompozicioni dijagram za aktivnost "3.2. Generisanje šeme baze podataka"

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 4.19.

Aktivnost
3.2.1. Kreiranje tabela

U okviru aktivnosti *"3.2.1. Kreiranje tabela"* kreiraju se tabele naredbom CREATE TABLE, koja definiše "praznu" tabelu sa nazivom i imenima kolona sa fizičkog modela podataka datog u ERwin-u. Podaci se kasnije unose INSERT naredbom ili iz razvijene korisničke aplikacije.

Opis tabela podrazumeva definisanje naziva tabele, naziva kolone (tip podatka i ograničenje), određivanje primarnog ključa i, po potrebi, definisanje indeksa po bilo kojoj koloni ili grupi kolona u tabeli.

Dakle, imena tabela i kolona se automatski preuzimaju iz fizičkog modela definisanog u ERwin-u, no ako se direktno u SUBP definišu imena moraju se poštovati sledeća uputstva:

- koristiti opisna imena za tabele, kolone, indekse i druge objekte;
- biti dosledan u korišćenju skraćenica;
- koristiti ista imena za opis istih tabela ili kolona;
- pisati nazive tabela ili kolona u jednini (predlog).

Sintaksa SQL naredbe za kreranje tabela ima sledeću strukturu:

```
CREATE TABLE tabela (kolona1 tip [ (velicina)] [index1] [, kolona2 tip [ (velicina)] [index2] [, ...]]
[, indexvisekol [, ...]])
```

Naredba CREATE TABLE koristi sledeće argumente.

Argument	Opis
tabela	Naziv tabele
kolona1, kolona2	Nazivi kolone ili kolona kojima se kreira nova tabela
tip	Tip podatka za kolonu u novoj tabeli
velicina	Veličina kolone u karakterima (za Text kolone)
index1, index2	Definisanje ograničenja za pojedine kolone index
indexvisekol	Definisanje ograničenja za više kolona index

Korišćenjem ERwin CASE alata za izabran desktop SUBP MS Access izgenerisana tabela RADNIK (videti tablice 4.1. i 4.2) ima sledeći izgled:

```
CREATE TABLE          (`Sifrar` TEXT (6),
`RADNIK`
                      `Prezime` TEXT (20),
                      `Ime` TEXT (20),
                      `Sifrarm` TEXT (2),
                      `JMBG` TEXT (13),
                      `rukov` TEXT (6),
                      `Datumz` DATETIME,
                      `Plata` FLOAT8,
                      `Stimul` FLOAT8,
                      `Sifrao` TEXT (2),
                      `Pol` TEXT (1),
                      `Vrsta` TEXT (1))
```

U naredbi CREATE TABLE specificiraju se:

- naziv tabele (npr.RADNIKA);
- imena kolona tabele RADNIK (sifrar, Prezime,...);
- tip podatka svake kolone.

Na osnovu generisanih tabela u sledećem koraku biće izvršeno generisanje indeksa.

Aktivnost
3.2.2. Kreiranje indeksa

U okviru aktivnosti *"3.2.2. Kreiranje indeksa"* definišu se indeksi naredbom CREATE INDEX za određenu tabelu nad jednom ili više kolona neke tabele. Indeks sadrži po jednu stavku za svaku različitu vrednost indeksirane kolone (ili kolona) u tabeli. U svakoj stavci indeksa pamte se fizičke adrese redova koji imaju datu vrednost u indeksiranoj koloni/kolonama.

Indeksi se implicitno koriste u sledećim slučajevima:

- kada se pretražuju tabele po vrednostima indeksiranih kolona (WHERE uslov) i
- kada se izdvajaju redovi tabele u redosledu vrednosti indesiranih kolona.

Indeks omogućava direktan pristup redovima i tako smanjuje vreme pristupa. Mogu se kreirati više indeksa za razne kombinacije kolona u istoj tabeli, ali svaki indeks uvećava vreme ažuriranja.

Dakle, na performanse baze podataka veoma utiče način na koji su podaci fizički smešteni, tj. memorisani. Kreiranje indeksa pomaže SUBP da locira specifične redove u bazi na isti način kao indekse u knjizi koji pomažu da se brzo lociraju specifične informacije.

Sintaksa kojom se kreira indeks je:

```
CREATE [UNIQUE], INDEX Index_ime
ON tabela (kolona [,<kolona2>,...])
```

Opcija UNIQUE obezbeđuje da tabela nikad ne sadrži redove sa jednakim vrednostima u kolonama koje su u indeksu.

Naredba CREATE INDEX koristi sledeće argumente:

Argument	Opis
Index_ime	Naziv indeksa koji se kreira
Tabela	Naziv tabele nad kojom se indeks formira
kolona	Nazivi kolone ili kolona nad kojima se indeks pravi

Da bi se dalje poboljšale performanse i da bi se omogućila jedinstvenost podataka, tj. da kolona sadrži jedinstvene vrednosti, treba kreirati jedinstven indeks (da svaka vrednost u koloni bude jedinstvena) korišćenjem opcije UNIQUE nad kolonom SIFRAR tabele RADNIK.

```
CREATE UNIQUE INDEX `PrimaryKey` ON `RADNIK` (`Sifrar`);
```

Na taj način se sprečava ubacivanje ili modifikovanje reda u kome vrednost ide u indeksirano polje, duplicirajući bilo koju vrednost polja.

Indeksa nad tabelom može biti koliko se želi, a može se i eksperimentisati da bi se našlo najpovoljnije rešenje. Preporučuje se da se male tabele ne indeksiraju, već da se to čini samo sa velikim tabelama.

Preporuke u radu sa indeksima su sledeće:

- tabele preko 200 redova indeksirati zbog poboljšanih performansi;
- zbog preglednosti, ne raditi više od tri indeksa po tabeli;
- indeksne kolone najčešće koristiti u WHERE ili JOIN klauzuli.

Za tabelu RADNIK generisani su sledeći indeksi:

```
CREATE UNIQUE INDEX `PrimaryKey`ON`RADNIK` (`Sifrar`);
CREATE UNIQUE INDEX `XAK2RADNIK` ON `RADNIK` (`JMBG`);
CREATE INDEX `XIF1RADNIK` ON `RADNIK` (`Sifrao`);
CREATE INDEX `XIF13RADNIK` ON `RADNIK` (`Sifrarm`);
CREATE INDEX `XIF6RADNIK` ON `RADNIK` (`rukov`);
```

Generisani indeksi za tabelu RADNIK definisani su u okviru aktivnosti: "3.1.3. Definisanje indeksa" (na slici 4.17. i tabeli 4.4).

Na osnovu prethodno izvedenih aktivnosti (slika 4.19) u sledećem koraku pristupa se generisanju poslovnih ograničenja.

Aktivnost
3.2.3. Generisanje poslovnih ograničenja

Kroz aktivnost *"3.2.3. Generisanje poslovnih ograničenja"* treba sagledati potrebu prilagođavanja definisanog fizičkog modela za konkretno realizovanu bazu podataka.

Ograničenja mogu biti definisana za tabele ili kolone i specificirana kao deo CREATE ili ALTER TABLE komande u okviru SQL-a. Svrha ograničenja je da se definiše rang validnih vrednosti. Svaka INSERT, UPDATE, ili DELETE naredba se proverava važećim ograničenjem. Ograničenje mora biti zadovoljeno da bi naredba uspela.

Generalno govoreći, ograničenjima se definišu domeni kolona, pravila referencijalnog integriteta (RI), pravila integriteta tabela, kao i dodatna pravila poslovanja.

Ograničenje se definiše ključnom reči CONSTRAINT i nazivom ime_ogranicenja kojim se specificira ime ograničenja (da li je u pitanju primarni ili preneseni ključ).

Sintaksa SQL za definisanje ograničenja je za:

```
CONSTRAINT ime_ogranicenja
{PRIMARY KEY (kolonapk1[, kolonapk2 [, ...]]) |
UNIQUE (kolonau1[, kolonau2 [, ...]]) |
FOREIGN KEY (kolonafk1 [, kolonafk2]...)
REFERENCES tabelafk (kolona1[,kolona2]...)
```

Ograničenje (CONSTRAINT) uslov ima sledeće argumente:

Argument	Opis
ime_ogranicenja	Naziv ograničenja
kolonapk1, kolonapk2	Naziv kolone ili kolona koje će biti definisane kao PRIMARY KEY
kolonau1, kolonau2	Naziv kolone ili kolona koje će biti definisane kao jedinstven ključ
kolonafk1, kolonafk2	Naziv prenesenog ključa kolone ili kolona
tabelafk	Naziv prenesene tabele koja ima kolonu ili kolone specificirane kao kolonafk
fkolona1, fkolona2	Nazivi kolone ili kolona u tabelafk koje specificiraju kolonafk1, kolonafk2

Mogu se definisti sledeća ograničenja za kolonu ili grupu kolona:

- ograničenja nad primarnim ključem,
- ograničenje UNIQUE,
- ograničenje NOT NULL,
- ograničenje nad prenesenim ključem,
- CHECK ograničenje.

Ograničenje nad primarnim ključem

Sintaksa vezana za definisanje ograničenja nad *primarnim ključem* ima sledeći izgled:

```
CONSTRAINT ime_ogranicenja
{PRIMARY KEY (kolonapk1[, kolonapk2 [, ...]])}
```

Ako je u pitanju primarni ključ, definiše se ime ograničenja kao, npr., za tabelu RADNIK, XPKRADNIK. U nastavku se definiše ključna reč PRIMARY KEY, kojom se određuje naziv kolone kao primarni ključ (Sifrar) tako da se dobije za primer tabele RADNIK sledeći oblik ograničenja:

```
CONSTRAINT `XPKRADNIK` PRIMARY KEY (`Sifrar`),
```

Dakle, primarni ključ je odabran kao jedinstven ključ, čija nijedna kolona ne sme da poprimi NULL vrednost ni u jednom redu. Kada se ovom klauzulom definiše primarni ključ tabele, SUBP u pozadini implicitno kreira jedinstven indeks i dodatna NOT NULL ograničenja nad svakom kolonom primarnog ključa.

Kompletna definicija tabele podrazumeva definisanje primarnog ključa. Mora se imati u vidu da se vrednosti kolona primarnog ključa nikada ne menjaju (ažuriraju), već postupak ažuriranja treba posmatrati kao brisanje reda i dodavanje reda sa novom vrednošću primarnog ključa.

Ograničenje UNIQUE

Ograničenje *UNIQUE* definiše jedinstven ključ tabele nad jednom ili više kolona, tj. svaki red mora imati različitu vrednost za kolonu, a svaka kolona mora biti deklarisana kao NOT NULL i ne mora biti primarni ključ.

Ograničenje NOT NULL

NOT NULL zabranjuje null vrednosti i se definiše isključivo kao ograničenje nad kolonom.

Ograničenje nad prenesenim ključem

Ograničenje nad prenesenim ključem, tzv. FOREIGN KEY ograničenje, predstavlja ograničenje za vrednost kolone ili grupe kolona koje postoje u drugoj tabeli. Preneseni ključ (foreign key) definisan je nad jednom ili više kolona (složeni ključ) i referencira se na primarni ili jedinstveni ključ neke tabele. Tabela u kojoj je definisan preneseni ključ je referencirajuća tabela ('dete'), a tabela na koju se ona referencira - referencirana tabela (tabela 'roditelj').

Takođe, ovom klauzulom se definišu pravila čuvanja referencijalnog integriteta (RI). Pravilo referencijalnog integriteta (RI) se iskazuje preko dva uslova za vrednost prenesenog ključa,

od kojih jedan uvek mora biti ispunjen. Ti uslovi glase:

- vrednost prenesenog ključa u referencirajućoj tabeli *mora* postojati kao vrednost ključa (primarnog ili jedinstvenog) u odgovarajućoj referenciranoj tabeli, ili
- vrednost neke kolone (ili više njih) prenesenog ključa *može* biti NULL, jer nije u sastavu primarnog ključa referencirajuće tabele.

Sintaksa za definisanje *prenesenog ključa* ima sledeći izgled:

```
CONSTRAINT ime_ogranicenja
{FOREIGN KEY (kolonafk1 [, kolonafk2]...)
REFERENCES korisnik.tabela (kolona[,kolona]...)}
```

FOREIGN KEY/REFERENCES ograničenje izvodi sledeće aktivnosti:

- odbacuje INSERT ili UPDATE ako odgovarajuće vrednosti ne postoje u tabeli primarnih ključeva;
- odbija DELETE ako će to narušiti REFERENCES ograničenje;
- mora se odnositi na PRIMARNI KLJUČ ili JEDINSTVENE (UNIQUE) kolone primarnih ključeva u odgovarajućim tabelama u kojima su definisani primarni ključevi;
- može se odnositi na PRIMARNI KLJUč tabele i ako nisu kolone ili grupe kolona specificirane u ograničenu (dozvoljena NULL vrednost);
- zahteva da je korisnik vlasnik tabele u kojoj su primarni ključevi, da ima, takođe, REFERENCE privilegije, ili da ima nivo kolone REFERENCE privilegije pri referenciranju kolona u tabeli u kojoj su primarni ključevi;
- zahteva da preneseni ključ kolona ima iste tipove podataka sa sprežućom primarnom tabelom.

Za dosadašnji primer preneseni ključ Sifrao iz tabele RADNIK referenciran je na tabelu ODELJENJE, gde je SIFRAO primarni ključ:

```
CONSTRAINT `zaposljava` FOREIGN KEY (`Sifrao`)
REFERENCES `ODELJENJE` (`Sifrao`));
```

Ime ograničenja 'zaposljava' vezano je za naziv relacije.

U primeru dokumenta "Karton isplata", za tabelu RADNIK, ako se izabere ODBC SQL opcija ograničenja (CONSTRAINT) u okviru Access šema Generation Report je:

```
CONSTRAINT `XPKRADNIK`
PRIMARY KEY (`Sifrar`),
CONSTRAINT `pripada`
FOREIGN KEY (`Sifrarm`)
REFERENCES `SIFRARM` (`Sifrarm`),
CONSTRAINT `rukovodi`
FOREIGN KEY (`rukov`)
REFERENCES `RADNIK` (`Sifrar`),
CONSTRAINT `zaposljava`
FOREIGN KEY (`Sifrao`)
REFERENCES `ODELJENJE` (`Sifrao`));
```

CHECK ograničenje

U CHECK ograničenju, za kolone, uslovi se moraju pozivati na kolone na koje se ograničenje već odnosilo. Za CHECK ograničenja tabele uslovi se mogu odnositi na više kolona. Po ANSI/ISO standardu, CHECK ograničenje je narušeno samo ako je izraz lažan, tj. istinite i nepoznate vrednosti ne narušavaju ograničenje.

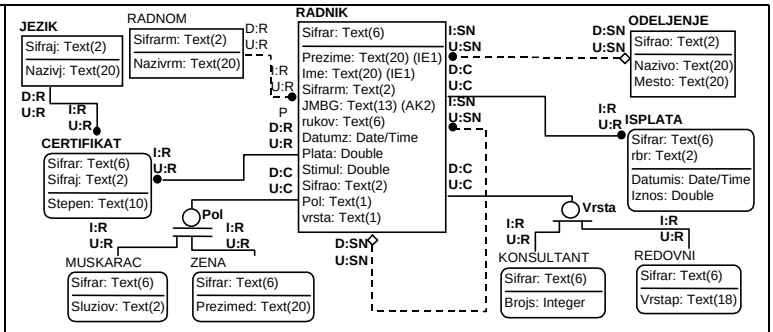
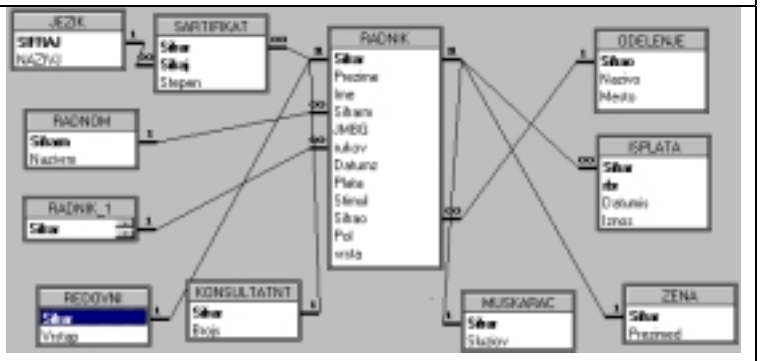
Prikaz ograničenja za Oracle SUBP za primer nad tabelom RADNIK ima sledeći izgled:

```
CREATE TABLE "RADNIK" (
 "Sifrar" VARCHAR2 (6) NOT NULL,
 "Prezime" VARCHAR2 (20) NOT NULL,
 "Ime" VARCHAR2 (20) NOT NULL,
 "Sifrarm" VARCHAR2 (2) NOT NULL,
 "JMBG" VARCHAR2 (13) NOT NULL,
 "rukov" VARCHAR2 (6) NOT NULL,
 "Datumz" DATE DEFAULT Date () NOT NULL
       CHECK (<=Date ()),
 "Plata"        FLOAT NOT NULL
 CHECK (>0),
 "Stimul" FLOAT DEFAULT 600 NOT NULL
       CHECK (stimul BETWEEN 500 AND 100
 "Sifrao" VARCHAR2 (2) NULL,
 "Pol" VARCHAR2 (1) DEFAULT M NOT NULL
 CHECK (Pol IN ('M', 'Z')),
 "vrsta" VARCHAR2 (1) DEFAULT 2,
 PRIMARY KEY ("Sifrar"),
 FOREIGN KEY ("Sifrarm")
 REFERENCES "SIFRARM",
 FOREIGN KEY ("rukov")
 REFERENCES "RADNIK",
 FOREIGN KEY ("Sifrao")
 REFERENCES "ODELJENJE");
```

U sledećem primeru biće generisana ograničenja za dokument "Karton isplata".

Generisanje poslovnih ograničenja za dokument "Karton isplata"

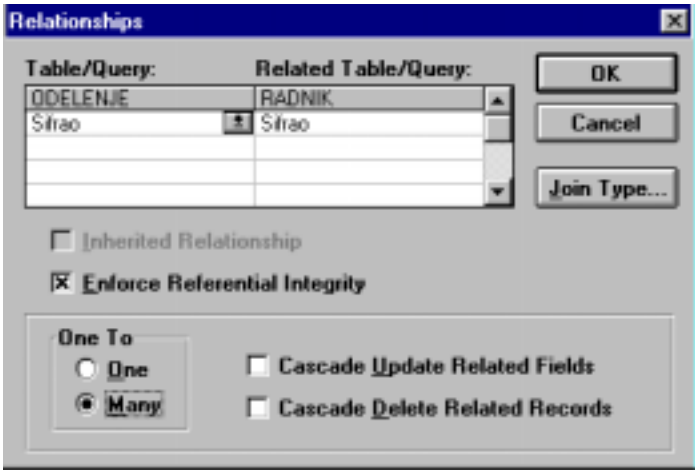
Da bi se lakše uočio način generisanja poslovnih ograničenja na slici 4.20. prikazani su fizički model podataka definisan u ERwin-u i model baze podataka generisan u MS ACCESS-u.

Fizički model primera dokumenta "Karton isplata" definisan u ERwin-u	
Model baze podataka primera dokumenta "Karton isplata" realizovan u MS ACCESS-u	

Slika 4.20. Fizički model podataka i model baze podataka za dokument "Karton isplata"

Generisanjem fizičkog modela iz ERwin-a u konkretni model baze podataka - MS ACCESS, automatski se uspostavljaju relacije između polja u selektovanim tabelama prema indeksima definisanim nad tim tabelama. *Relacija* se ovde definiše kao odnos između polja u

različitim tabelama baze podataka. Relacije između polja tabela se uspostavljaju po sličnosti (jednakosti) naziva, tipa i veličine polja. Na slici 4.21. prikazana je relacija između tabela: RADNIK i ODELJENJE.



Slika 4.21. MS ACCESS relacija između tabela: RADNIK i ODELJENJE sa slike 4.20.

U MS ACCESS-u uspostavljaju se sledeće relacije (One To):

- jedan prema jedan (*One-To-One*-1/1) tj. jednom redu iz jedne tabele pripada jedan red u drugoj tabeli (slika 4.21);
- jedan prema više (*One-To-Many*-1/m), tj. jednom redu iz jedne tabele pripada više redova iz druge tabele sa podacima istim kao u prvoj tabeli.

Ako je uključena opcija "*Enforce Referential Integrity*" omogućuje se obezbeđenje kontrole integriteta baze podataka u zavisnosti od generisane relacije iz ERwin-a u MS ACCESS, tj. omogućuje se definisanje kardinalnosti i referencijalnog integriteta (RI) relacija.

Kardinalnost relacija na nivou MS ACCESS-a je odnos *One* ili *Many* i to je odnos prve tabele prema drugoj. *One* je 1/1 odnos, a *Many* je 1/m odnos.

Tako relacija sa slike 4.21. glasi: Jednom odeljenju pripada više radnika (*One To Many*).

Referencijalni integritet na nivou MS ACCESS može biti (slika 4.21):

- Cascade Update Related Fields i
- Cascade Delete Related Records.

Tip relacije *Cascade Update Related Fields* omogućuje da se u slučaju izmene sadržaja polja u prvoj tabeli promene i vrednosti polja koja su u relaciji sa njima u drugoj tabeli.

Tip relacije *Cascade Delete Related Records* omogućuje da se u slučaju brisanja redova u jednoj tabeli izaziva brisanje redova iz druge tabele.

Tipovi relacija definisani u MS ACCESS-u su mnogo siromašniji po svojim mogućnostima nego što je to opisano u ERwin modelu podataka. Ova konstatacija dolazi do izražaja prilikom sprovođenja inverznog inženjeringa, jer je dobijeni model podataka mnogo siromašniji od željenog, pa ga treba dograđivati.

U okviru relacije se definiše i tip Join (slika 4.21, tipka *Join Type...).* Postoje tri tipa Join, između MS ACCESS tabela:

- biraju se samo redovi koji u veznim poljima imaju iste sadržaje (Iner Join);
- biraju se svi redovi iz prve tabele i samo oni redovi iz druge tabele čiji je sadržaj veznih polja jednak sa sadržajem u prvoj tabeli (Left Join);
- biraju se svi redovi iz druge tabele i samo oni redovi iz prve tabele čiji je sadržaj veznih polja jednak sadržaju veznih polja druge tabele (Right Join).

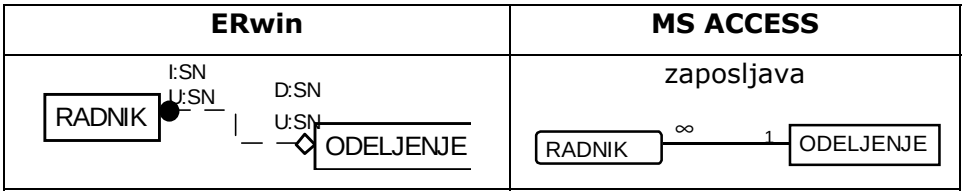
U poglavlju 4.3. u okviru aktivnosti *"3.3.3. Definisanje upita"* biće razmatrana detaljno problematika Join-a.

U daljem tekstu biće upoređene definisane kardinalnosti veza u ERwin-u i generisane kardinalnosti relacija u MS ACCESS-u, kao i referencijalni integritet (RI) veza u ERwin-u i generisani referencijalni integritet (RI) relacija u MS ACCESS-u, na primeru dokumenta "Karton isplata".

Generisanje relacije RADNIK-ODELJENJE

Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.22):

U Odeljenju je zaposleno više Radnika a Radnik radi u nula ili jednom Odeljenju.



Slika 4.22. Uporedni prikaz kardinađlnosti i referencijalnog integriteta u ERwin-u i MS ACCESS-u

Na slici 4.22. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-u i generisanih kardinalnosti i relacija u MS ACCESS-u.

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Set Null	Ne može se uneti Radnik bez 'roditelj' Odeljenje, sem ako preneseni ključ radnika ne setuje na NULL (I:SN)
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela ODELJENJE	Set Null	Radnik za koje je Odeljenje 'roditelj' setuje preneseni ključ Radnika na NULL (D:SN)
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Tabela ODELJENJE	Set Null	Prilikom izmena Radnici za koje je Odeljenje 'roditelj' setuju preneseni ključ Radnika na NULL (U:SN)
Tabela RADNIK	Set Null	Ne može se izmeniti RADNIK bez Odeljenja, sem ako preneseni ključ Radnika se ne setuje na NULL (U:SN

Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.22) pokazuje da opcija CASCADE nije uključena, što znači da se operacije menjanja i brisanja izvode nezavisno.

Generisanje relacije RADNIK-SIFRARM

Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.23):

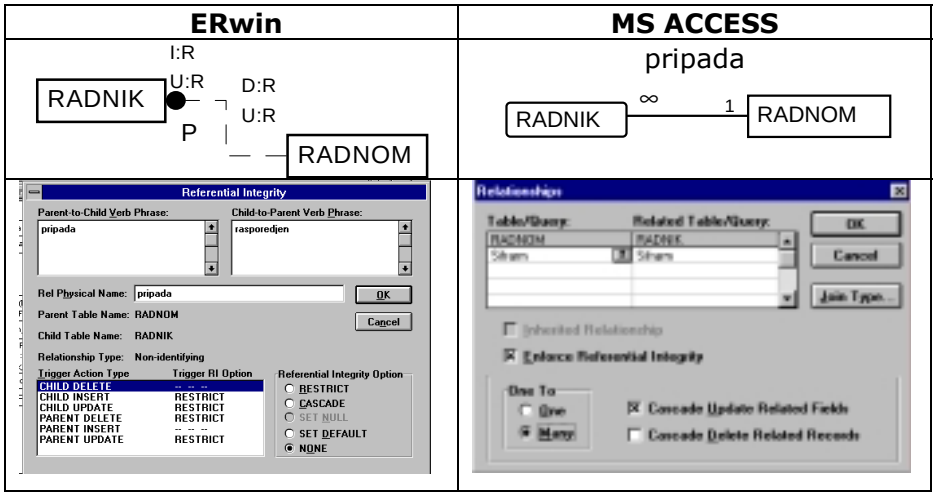
Na radno mesto mora biti raspoređen najmanje jedan Radnik, a Radnik zauzima isključuvo jedno radno mesto.

Na slici 4.23. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-

u i generisanih kardinalnosti i relacija u MS ACCESS-u.

Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.23) pokazuje da je opcija *Cascade Update Related Fields* uključena, što znači da promena vrednosti polja (Sifrarm) u tabeli SIFRARM izaziva promenu vrednosti polja (Sifrarm) i u tabeli RADNIK.

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:



Slika 4.23. Uporedni prikaz kardinalnosti i referencijalnog integriteta u ERwin i MS ACCESS-u

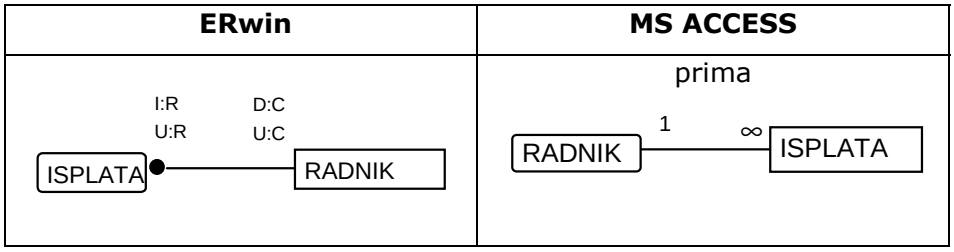
Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Restrict	Ne može se uneti Radnik (I:R) ako ne postoji 'roditelj' SIFRARM
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela SIFRARM	Restrict	Ne dozovoljava se brisanje SIFRARM(ako je poslednji 'roditelj' Radnik)
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Tabela SIFRARM	Restrict	Ne može se izmeniti SIFRARM (U:R) ako je vezan za Radnika
Tabela RADNIK	Restrict	Ne može se izmeniti Radnik (U:R) ako ne postoji Odeljenje

Generisanje relacije RADNIK-ISPLATA

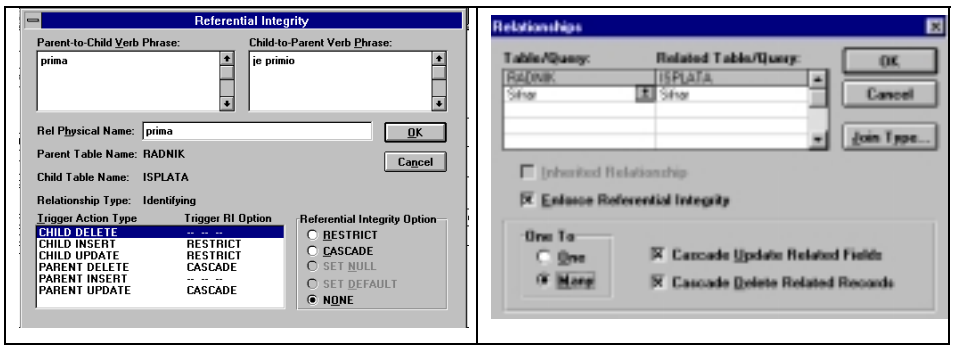
Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.24):

Radnik je primio više isplata, a jednu isplatu prima isključivo jedan Radnik.

Na slici 4.24. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-u i generisanih kardinalnosti i relacija u MS ACCESS-u.



alempije@beotel.yu



Slika 4.24. Uporedni prikaz kardinalnosti i referencijalnog integriteta u ERwin i MS ACCESS-u

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela ISPLATA	Restrict	Ne može se uneti Isplata (I:R) bez 'roditelj' Radnik
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Cascade	Brišu se sve Isplate za izbrisanog 'roditelj' Radnik (D:C)
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Cascade	Automatska izmena Isplata za izmene nad Radnikom (U:C)
Tabela ISPLATA	Restrict	Ne može se izmeniti Isplata (U:R) ako ne postoji Radnik

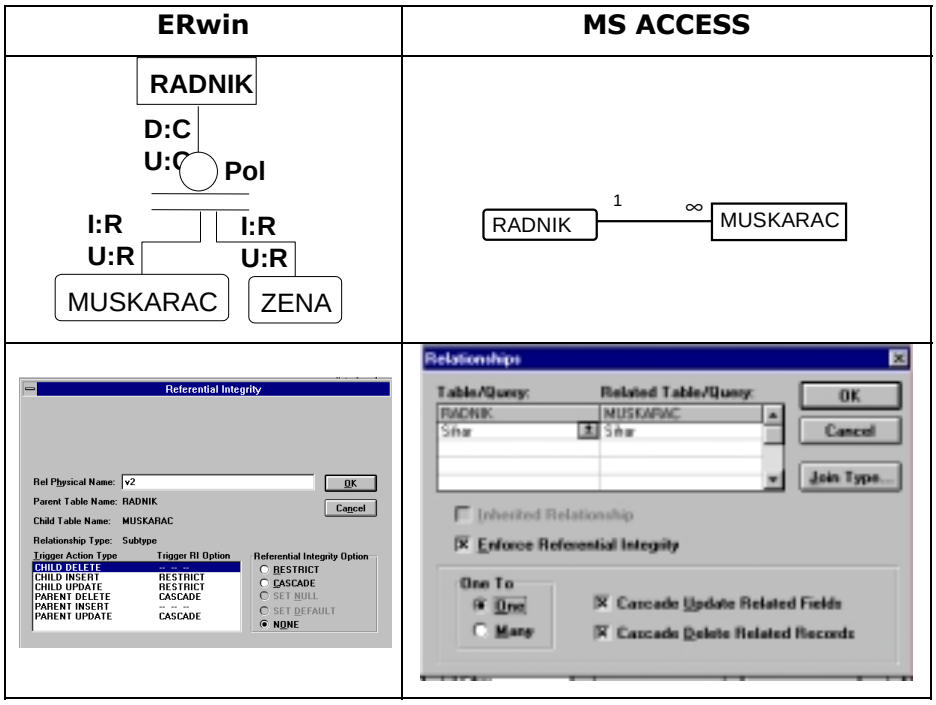
Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.24) pokazuje da je opcija *Cascade Update Related Fields* uključena, što znači da promena vrednosti polja (Sifrar) u tabeli RADNIK izaziva promenu vrednosti polja (Sifrar) i u tabeli ISPLATA, a uključena opcija *Cascade Delete Related Records* definiše automatsko brisanje redova tabele ISPLATA za izbrisan red tabele RADNIK po vrednosti primarnog ključa Sifrar.

Generisanje relacije RADNIK-MUSKARAC-ZENA

Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.25):

Radnik može biti ili muškarac ili žena.

Na slici 4.25. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-u i generisanih kardinalnosti i relacija u MS ACCESS-u.



Slika 4.25. Uporedni prikaz kardinalnosti i referencijalnog integriteta u ERwin i MS ACCESS-u

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela MUSKARAC		Ne može se uneti Muskarac bez unesenog 'roditelj' Radnika
Tabela ZENA		Ne može se uneti Zena bez unesenog 'roditelj' Radnika
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Cascade	Briše se primerak kategorije za izbrisanog Radnika kao 'roditelj'

Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.25) pokazuje da je opcija *Cascade Update Related Fields* uključena, što znači da promena vrednosti polja (Sifrar) u tabeli RADNIK izaziva promenu vrednosti polja (Sifrar) ili u tabeli MUSKARAC ili ZENA, a uključena opcija *Cascade Delete Related Records* definiše automatsko brisanje ILI reda tabele MUSKARAC ili ZENA za izbrisan red tabele RADNIK po vrednosti primarnog ključa Sifrar.

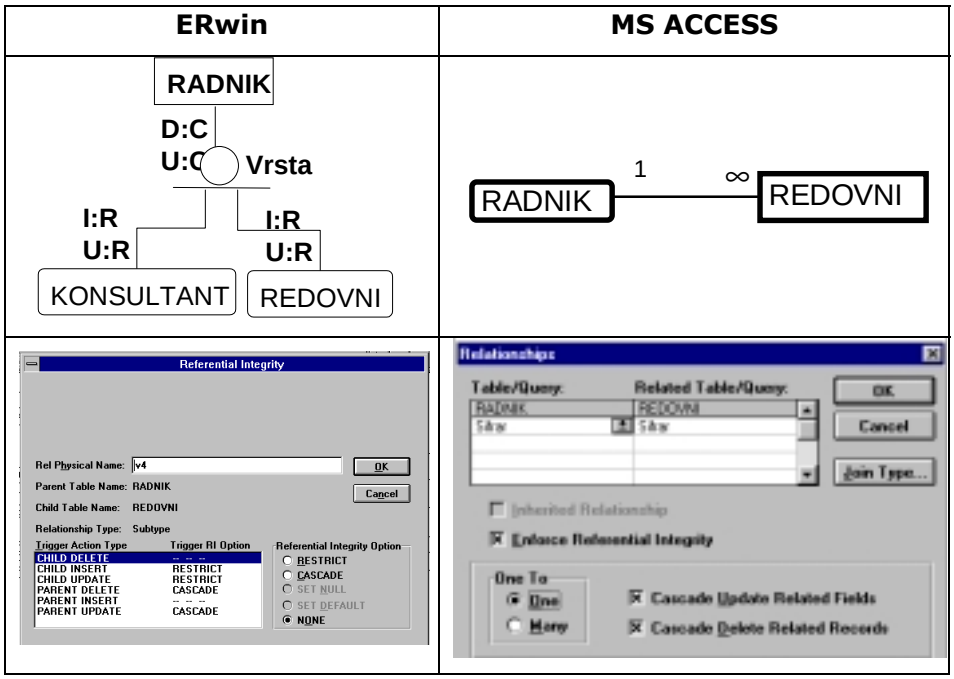
Generisanje relacije RADNIK-KONSULTANT-REDOVNI

Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.26):

Radnik može biti REDOVAN, KONSULTANT ili neki drugi tip radnika.

Na slici 4.26. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-u i generisanih kardinalnosti i relacija u MS ACCESS-u.

Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.25) pokazuje da je opcija *Cascade Update Related Fields* uključena, što znači da promena vrednosti polja (Sifrar) u tabeli RADNIK izaziva promenu vrednosti polja (Sifrar) ili u tabeli KONSULTANT ili REDOVNI, a uključena opcija *Cascade Delete Related Records* definiše automatsko brisanje reda tabele KONSULTANT ili REDOVNI za izbrisan red tabele RADNIK po vrednosti primarnog ključa Sifrar.



Slika 4.26. Uporedni prikaz kardinalnosti i referencijalnog integriteta u ERwin i MS ACCESS-u

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:

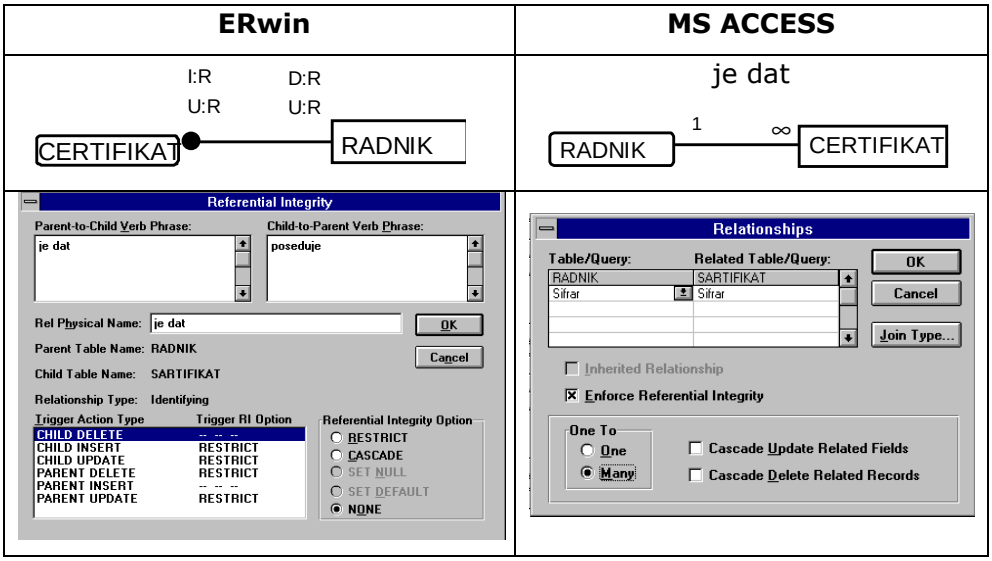
Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela KONSULTANT		Ne može se uneti Konsultant bez unesenog 'roditelj'
Tabela REDOVNI		Ne može se uneti Redovni bez unesenog 'roditelj'
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Cascade	Briše se primerak kategorije za izbrisanog Radnika kao 'roditelj'

Generisanje relacije RADNIK-CERTIFIKAT

Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.27):

Radnik može posedovati više certifikata, a certifikat poseduje tačno jedan radnik.

Na slici 4.27. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-u i generisanih kardinalnosti i relacija u MS ACCESS-u.



Slika 4.27. Uporedni prikaz kardinalnosti i referencijalnog integriteta u ERwin i MS ACCESS-u

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela CERTIFIKAT	Restrict	Ne može se uneti Certifikat (I:R) bez 'roditelj' Radnik
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Restrict	Ne može se obrisati Radnik (D:R) dok se ne izbrišu svi Certifikati
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Restrict	Ne može se izmeniti Radnik (U:R) dok se ne izvrši izmena nad tabelom Certifikati
Tabela CERTIFIKAT	Restrict	Ne može se izmeniti Certifikat (U:R) ako ne postoji Radnik

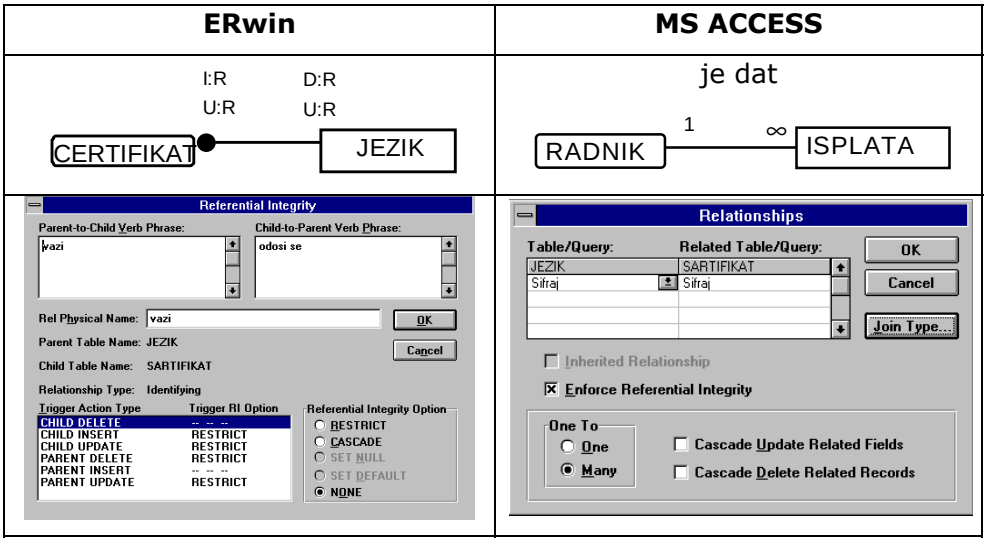
Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.27) pokazuje da opcija CASCADE nije uključena, što znači da se operacije menjanja i brisanja izvode nezavisno.

Generisanje relacije JEZIK-CERTIFIKAT

Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.28):

Jezik se odnosi na više certifikata a certifikat, se odnosi na tačno jedan jezik.

Na slici 4.28. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-u i generisanih kardinalnosti i relacija u MS ACCESS-u.



Slika 4.28. Uporedni prikaz kardinalnosti i referencijalnog integriteta u ERwin i MS ACCESS-u

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:

Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela CERTIFIKAT	Restrict	Ne može se uneti Certifikat (I:R) bez 'roditelj' Jezik
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela JEZIK	Restrict	Ne može se obrisati Jezik (D:R) dok se ne izbrišu svi Certifikati
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Tabela JEZIK	Restrict	Ne može se izmeniti Jezik (D:R) dok se ne izvrši izmena nad tabelom Certifikat
Tabela CERTIFIKAT	Restrict	Ne može se izmeniti Certifikat (U:R) ako ne postoji Jezik

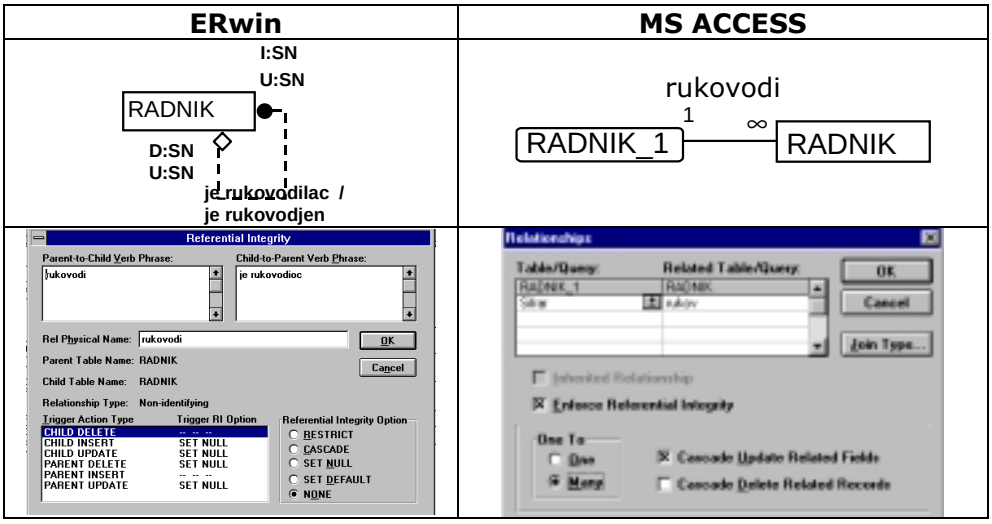
Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.28) pokazuje da opcija CASCADE nije uključena, što znači da se operacije menjanja i brisanja izvode nezavisno.

Generisanje relacije RADNIK-RADNIK

Kardinalnost veza u ERwin-u i kardinalnost relacija u MS ACCESS-u glasi (slika 4.29):

Radnik (šef) je rukovodilac nula ili jednom Radniku, a radnik (šef) rukovodi sa više radnika.

Na slici 4.29. dat je uporedni prikaz kardinalnosti veza i referencijalnog integriteta u ERwin-u i generisanih kardinalnosti i relacija u MS ACCESS-u.



Slika 4.29. Uporedni prikaz kardinalnosti i referencijalnog integriteta u ERwin i MS ACCESS-u

Referencijalni integritet fizičkog modela definisanog u ERwin-u ima sledeće operacije:

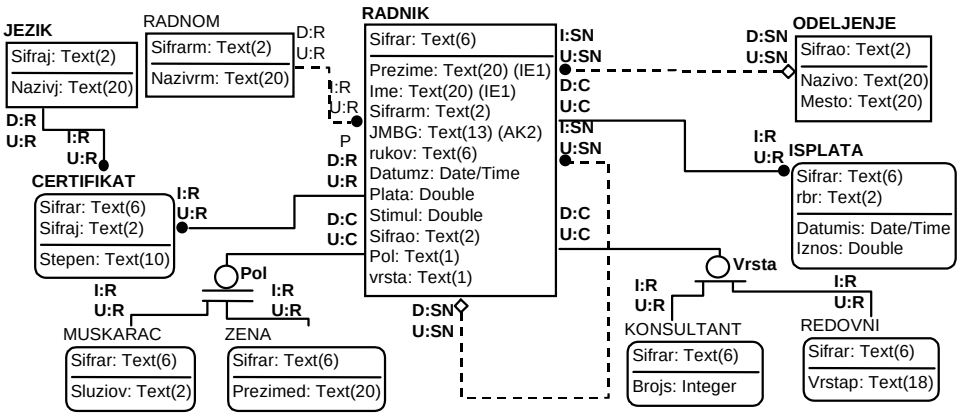
Unos (Insert) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK	Set Null	Ne može se uneti RADNIK bez 'roditelj' Rukov, sem ako preneseni ključ Radnika ne setuje na NULL (I:SN)
Brisanje (Delete) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK_1	Set Null	RADNIK za koje je Rukov 'roditelj' setuju preneseni ključ Rukov na NULL (D:SN)
Izmena (Update) omogućuje izvođenje sledećih akcija:		
Tabela RADNIK_1	Set Null	Prilikom izmena RADNIK za koje je Rukov 'roditelj' setuju preneseni ključ rukovodioca na NULL (U:SN)
Tabela RADNIK	Set Null	Ne može se izmeniti RADNIK bez Rukov, sem ako preneseni ključ rukovodioca ne setuje na NULL (U:SN)

Referencijalni integritet relacija na nivou MS ACCESS-a (slika 4.29) pokazuje da je opcija *Cascade Update Related Fields* uključena, što znači da promena vrednosti polja (Sifrar) u tabeli RADNIK_1 izaziva promenu vrednosti polja (Rukov) i u tabeli RADNIK.

Na osnovu prethodno izvedenih aktivnosti (slika 4.19) u sledećem koraku pristupa se verifikaciji dizajnirane šeme baze podataka.

Aktivnost
3.2.4. Verifikacija šeme baze podataka

Aktivnost *"3.2.4. Verifikacija šeme baze podataka"* verifikuje za dosadašnji primer, dokument "Karton isplata", gde se za kreirane tabele unose test-podaci.



Slika 4.30. Fizički model podataka za dokument "Karton isplata"

U sledećim tabelama biće prikazan SQL skript generisanih tabela u MS ACCESS-u, kao i izvršena verifikacija šeme unošenjem test-podataka. Na slici 4.30. prikazan je fizički model koji je korišćen da ERwin izgeneriše šeme baze podataka, tj. odgovarajuće tabele.

U tablici br 4.5. prikazana je kompletna šema za tabele: RADNIK i ODELJENJE.

Tablica 4.5.

RADNIK	ODELJENJE

CREATE TABLE `RADNIK` (
`Sifrar` TEXT (6),
`Prezime` TEXT (20),
`Ime` TEXT (20),
`Sifrarm` TEXT (2),
`JMBG` TEXT (13),
`rukov` TEXT (6),
`Datumz` DATETIME,
`Plata` FLOAT8,
`Stimul` FLOAT8,
`Sifrao` TEXT (2),
`Pol` TEXT (1),
`vrsta` TEXT (1),
CONSTRAINT `XPKRADNIK`
PRIMARY KEY (`Sifrar`),
CONSTRAINT `pripada`
FOREIGN KEY (`Sifrarm`)
REFERENCES `SIFRARM`
(`Sifrarm`),
CONSTRAINT `rukovodi`
FOREIGN KEY (`rukov`)
REFERENCES`RADNIK`
(`Sifrar`),
CONSTRAINT `zaposljava`
FOREIGN KEY (`Sifrao`)
REFERENCES`ODELJENJE`
(`Sifrao`));
CREATE UNIQUE INDEX
`PrimaryKey` ON `RADNIK`
(`Sifrar`);
CREATE UNIQUE INDEX
`XAK2RADNIK` ON `RADNIK`
(`JMBG`);
CREATE INDEX `XIF1RADNIK`
ON `RADNIK` (`Sifrao`);
CREATE INDEX `XIF13RADNIK`
ON `RADNIK` (`Sifrarm`);
CREATE INDEX `XIF6RADNIK`
ON `RADNIK` (`rukov`);
	CREATE TABLE `ODELJENJE`(
	`Sifrao` TEXT (2),
	`Nazivo` TEXT (20),
	`Mesto` TEXT (20),
	CONSTRAINT `XPKODELJENJE`
	PRIMARY KEY (`Sifrao`));
	CREATE UNIQUE INDEX
	`PrimaryKey` ON`ODELJENJE`
	(`Sifrao`);

Za kreiranu tabelu RADNIK verifikacija je izvršena unošenjem sledećih test-primeraka:

RADNIK

Sifra	Prezime	Ime	Sifr	JMBG:	rukov:	Datumz	Plata	Stim	Si	P	Vr
827369	STEVIC	ZORAN	01	1411952171033	827902	28/03/98	8000	600	2	M	2
827499	ALAGIC	MILAN	02	2503964345612	827698	28/03/98	1600	600	3	M	
827521	VUKIC	MILOS	02	1130497055432	827698	28/03/98	1250	600	3	M	2
827566	JOVIC	MIRA	03	1130497056435	827839	28/04/90	2975	700	2	Z	1
827654	MARTIC	ZORA	02	1140496055444	827698	12/06/88	1250	1000	3	Z	1
827698	BOBIC	IVAN	03	1405987055455	827839	15/11/89	2850	800	3	M	2
827782	CEBIC	GORAN	03	1203970654566	827839	11/06/80	2450	500	1	M	2
827788	SUSIC	ZORAN	04	1304954676755	827566	23/10/91	3000	650	2	M	2
827839	KLJAKIC	STEVA	05	1312952122344	827839	06/12/87	5000	600	1	M	2
827844	TUBIC	MIRA	02	1312954342122	827698	08/03/78	1500	600	3	Z	1
827876	ALIMPIC	PETAR	01	2503976343566	827788	09/05/92	1100	780	2	M	
827900	JAKIC	VLADA	01	1211965457231	827698	28/03/90	9500	900	3	M	
827902	FILIPIC	DRAGA	04	1210970534221	827566	20/11/96	3000	1000	2	M	2
827934	MILIC	DRAGA	01	0706956456465	827782	28/03/93	1300	790	1	M	2

Za kreiranu tabelu ODELJENJE verifikacija je izvršena unošenjem sledećih test-primeraka:

ODELJENJE

Sifrao	Nazivo	Mesto
10	PRIPREMA	PANCEVO
20	RAZVOJ	NOVI SAD
30	PRODAJA	BEOGRAD
40	PROIZVODNJA	PAZOVA

U tablici 4.6. prikazana je kompletna šema za tabele: SIFRARM i JEZIK.

Tablica 4.6.

SIFRARM	JEZIK
CREATE TABLE `SIFRARM` (`Sifrarm` TEXT (2), `Nazivrm` TEXT (20), CONSTRAINT `XPKSIFRARM` PRIMARY KEY (`Sifrarm`)); CREATE UNIQUE INDEX `PrimaryKey` ON `SIFRARM` (`Sifrarm`);	CREATE TABLE `JEZIK` (`Sifraj` TEXT (2), `Nazivj` TEXT (20), CONSTRAINT `XPKJEZIK` PRIMARY KEY (`Sifraj`)); CREATE UNIQUE INDEX `PrimaryKey` ON `JEZIK` (`Sifraj`);

Za kreirane tabele SIFRARM i JEZIK verifikacija je izvršena unošenjem sledećih test-primeraka:

SIFRARM

Sifrarm	Nazivrm
01	PROJEKTANT
02	REFERENT
03	DIREKTOR
04	TEHNOLOG
05	GEN_DIR

JEZIK

Sifraj	Nazivj
01	ENGLESKI
02	NEMACKI
03	RUSKU
04	FRANCUSKI

U tablici 4.7.prikazana je kompletna šema za tabele: CERTIFIKAT i ISPLATA.

Tablica 4.7.

CERTIFIKAT	ISPLATA
CREATE TABLE `CERTIFIKAT` (`Sifrar` TEXT (6), `Sifraj` TEXT (2), `Stepen` TEXT (10), CONSTRAINT `XPKZNA JEZIK` PRIMARY KEY (`Sifrar`, `Sifraj`), CONSTRAINT `vazi` FOREIGN KEY (`Sifraj`) REFERENCES `JEZIK` (`Sifraj`), CONSTRAINT `je dat` FOREIGN KEY (`Sifrar`) REFERENCES `RADNIK` (`Sifrar`)); CREATE UNIQUE INDEX `PrimaryKey` ON `CERTIFIKAT` (`Sifrar`,`Sifraj`); CREATE INDEX `XIF3ZNA JEZIK` ON `CERTIFIKAT` (`Sifrar`); CREATE INDEX `XIF4ZNA JEZIK` ON `CERTIFIKAT` (`Sifraj`);	CREATE TABLE `ISPLATA` (`Sifrar` TEXT (6), `rbr` TEXT (2), `Datumis` DATETIME, `Iznos` FLOAT8, CONSTRAINT `XPKISPLATA` PRIMARY KEY (`Sifrar`,`rbr`), CONSTRAINT `prima` FOREIGN KEY (s ifrar) REFERENCES`RADNIK` (`Sifrar`)); CREATE UNIQUE INDEX `PrimaryKey` ON `ISPLATA` (`Sifrar`,`rbr`); CREATE INDEX `XIF2ISPLATA` ON `ISPLATA` (`Sifrar`);

Za kreirane tabele: CERTIFIKAT i ISPLATA verifikacija je izvršena unošenjem sledećih test--primeraka:

CERTIFIKAT

Sifrar	Sifraj	Stepen
827369	01	Cita
827369	02	Govori
827369	03	Pise
827654	01	Cita
827698	03	Cita

ISPLATA

Sifrar	rbr	Datumis	Iznos
827654	01	01/01/98	450
827654	02	08/03/98	1500
827698	01	01/02/98	2000
827698	02	01/06/98	1000

U tablici 4.8. prikazana je kompletna šema za tabele: REDOVNI i KONSULTANT.

Tablica 4.8.

REDOVNI	KONSULTANT
CREATE TABLE `REDOVNI` (`Sifrar` TEXT (6), `Vrstap` TEXT (18), CONSTRAINT `XPKE/8` PRIMARY KEY (`Sifrar`), CONSTRAINT `v4` FOREIGN KEY (`Sifrar`) REFERENCES`RADNIK` (`Sifrar`)); CREATE UNIQUE INDEX `PrimaryKey` ON `REDOVNI` (`Sifrar`);	CREATE TABLE `KONSULTANT` (`Sifrar` TEXT (6), `Brojs` SHORT, CONSTRAINT `XPKE/9` PRIMARY KEY (`Sifrar`), CONSTRAINT `v1` FOREIGN KEY (`Sifrar`) REFERENCES `RADNIK` (`Sifrar`)); CREATE UNIQUE INDEX`PrimaryKey` ON `KONSULTATNT` (`Sifrar`);

Za kreirane tabele: REDOVNI i KONSULTANT verifikacija je izvršena unošenjem sledećih test primeraka:

KONSULTANT

Sifrar	Brojs
827654	10
827698	5
827782	15
827788	12

REDOVNI

Sifrar	Vrstap
827369	01
827499	02
827521	01
827566	03

U tablici 4.9 prikazana je kompletna šema za tabele ZENA i MUSKARAC.

Tablica 4.9.

ZENA	MUSKARAC
CREATE TABLE `ZENA` (`Sifrar` TEXT (6), `Prezimed`TEXT (20), CONSTRAINT `XPKZENA` PRIMARY KEY (`Sifrar`), CONSTRAINT `v3` FOREIGN KEY (`Sifrar`) REFERENCES`RADNIK` (`Sifrar`)); CREATE UNIQUE INDEX `PrimaryKey` ON `ZENA` (`Sifrar`);	CREATE TABLE `MUSKARAC` (`Sifrar` TEXT (6), `Sluziov`TEXT (2), CONSTRAINT `XPKMUSKARAC` PRIMARY KEY (`Sifrar`), CONSTRAINT `v2` FOREIGN KEY (`Sifrar`) REFERENCES`RADNIK` (`Sifrar`)); CREATE UNIQUE INDEX `PrimaryKey` ON `MUSKARAC` (`Sifrar`);

Za kreirane tabele: ZENA i MUSKARAC verifikacija je izvršena unošenjem sledećih test--primeraka:

ZENA

Sifrar	Prezimed
827654	Vasic
827566	Savic
827844	Miric

MUSKARCI

Sifrar	Sluziov
827698	DA
827782	DA
827788	NE
827369	DA
827499	DA

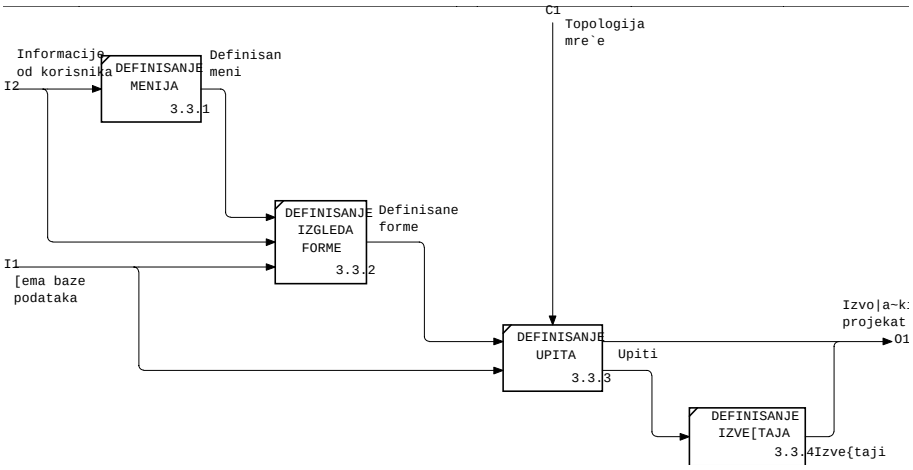
Na osnovu generisane šeme i verifikovanih tabela za fizički model sa slike 4.29. pristupa se izradi odgovarajuće aplikacije kroz aktivnost *"3.3 Specifikacija aplikacije"*, što je predmet daljih razmatranja.

Na osnovu prethodno izvedenih aktivnosti (slika 4.1) u sledećem koraku pristupa se izradi aplikacije.

Aktivnost
3.3. Izrada aplikacije

Aktivnost *"3.3. Izrada aplikacije"* izvodi se na osnovu prethodno urađene šeme baze podataka, kao i konkretnih zahteva budućih korisnika. Specifikacija forme se izvodi za sledeće aktivnosti (slika 4.31):

- Aktivnost 3.3.1. Definisanje menija,
- Aktivnost 3.3.2. Definisanje izgleda forme,
- Aktivnost 3.3.3. Definisanje upita i
- Aktivnost 3.3.4. Definisanje izveštaja.



Slika 4.31. Dekompozicioni dijagram za aktivnost "3.3. Izrada aplikacije"

Aktivnost *"3.3. Izrada aplikacije"* pravi se uz sve specifičnosti konkretnog SUBP. Međutim, u okviru ove aktivnosti treba opisati zajedničke postavke koje se moraju poštovati bez obzira na izabrani SUBP. Kada treba prikazati izgled odgovarajućih objekata, koristi se desktop SUBP MS ACCESS.

Izrada aplikacije, sa druge strane, neposredno je vezana i za klijent/server arhitekturu, i to u zavisno od toga da li je u pitanju dvoslojna ili troslojna arhitektura.

U dvoslojnoj klijent/server arhitekturi definiše se jedan server u kojoj se nalazi baze podataka i SUBP (videti glavu 4.2) i klijent-strana gde su definisane klijentove (korisničke) aplikacije, pa se izrada aplikacije izvodi na klijent-strani.

Troslojna arhitektura sadrži server kao u prethodnom slučaju, ali i tzv. aplikativni server koji sadrži zajedničke aplikacije koje napada klijent; treći sloj je klijent gde se definišu aplikacije koje su specifične za konkretnog korisnika i nalaze se na klijent-strani. Kako se i u jednom i drugom slučaju aplikacije odnose na korisnika, to se i definišu kao aplikacija klijent. Aplikacija klijent: radi sa redovima iz tabele; poseduje interfejs prema korisniku, tzv. GUI (Graphical User Interface); izvršava logiku aplikacije; proverava ispravnost ulaznih podataka; traži prijem podataka od servera.

Aplikacija klijent se razmatra sa aspekta:

- korišćenja same aplikacije klijent,

- odgovarajućeg razvojnog okruženja aplikacije klijent,
- korisničkog interfejsa u aplikacijama klijent.

Korišćenje aplikacije klijent podrazumeva da korisnici unose, pretražuju i analiziraju podatke, a administrator koristi servisne programe za održavanje servera i da projektanti izrađuju aplikacije klijent.

Razvojno okruženje aplikacije klijent obuhvata:

- unos podataka korišćenjem formi,
- direktnu transakcionu obradu pomoću formi,
- izradu upita i
- izradu izveštaja korišćenjem alata za razvoj aplikacija.

Unos podataka korišćenjem *formi* treba da omogući automatsku proveru pravila integriteta i pri tom, ako pravila integriteta nisu ispunjena, treba da ponudi listu dozvoljenih vrednosti, imajući u vidu i mogućnost multimedijalnog prikaza (npr., uputstva za rad sa formom).

Direktna transakciona obrada ili, kako se još definiše, Online transaction processing-OLTP, pod transakcijom podrazumeva operaciju za dodavanje ili ažuriranje podataka u bazi, gde aplikacija šalje zahtev za ažuriranje podataka serveru baze podataka, a server ažurira bazu i registruje transakciju (videti glavu 4.1).

Upiti i izveštaji su posebne aplikacije gde se može definisati izveštaj od više kolona ili, pak, jedan zapis na strani ili grafički izveštaji. Izveštaji se najčešće prave kao rezultat AD-HOCK upita.

Alati za razvoj aplikacija mogu biti jezici treće generacije: C++, ADA, COBOL ili jezici četvrte generacije: Oracle Forms, Card, Reports i Graphics, MS ACCESS korišćenjem - CASE alata.

Korisnički interfejsi u aplikacijama klijent mogu biti tekstualni korisnički interfejs (CUI - Character-based User Interface) ili grafički korisnički interfejs (GUI - Graphical User Interface).

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 4.31.

Aktivnost
3.3.1. Definisanje menija

Definisani meniji treba da prate scenario odvijanja aktivnosti budućeg korisnika. Za definisanje menija moraju se koristiti odgovarajuća pravila za strukturiranje kojima se definiše mogući redosled pozivanja operacija.

Meniji treba da se definišu na takav način:

- da svaki meni ima koncizan naslov na vrhu;
- da meniji koji zauzimaju ceo ekran budu balansirani;
- da se razdvajaju liste opcija na više celina;
- da se ograniči broj izbora u meniju na jedan ekran;
- da se razmisli o selekciji menija;
- da se omogući napuštanje menija bez izbora bilo koje opcije;
- da se koriste aktivne imenice za opis opcija menija;
- da se koriste nedvosmislene ikone;
- da se izbegava često naglašavanje;
- da se omogući korišćenje i malih i velikih slova,
- da se proveri tastatura pre upotrebe;
- da se izabere jasna, opštepoznata, kratka reč za komande;

- da se omogući korisniku da upotrebi više komandi u jednoj liniji;
- da se omogući korisniku da dobije listu komandi.

Definisanje menija i kretanje kroz aplikaciju treba da odražavaju logičan način rada korisnika aplikacije, stoga su oni povezani sa sledećom aktivnosti: *"3.3.2 Definisanje izgleda forme"* .

Aktivnost
3.3.2. Definisanje izgleda forme

Ekranske forme su osnovni tip objekata u većini SUBP i treba da omoguće korisniku predstavljanje podataka iz baze i unos podataka u bazu. Forme u sebi mogu imati veliki broj drugih objekata (kontrola). Većina SUBP, koji za osnovu imaju MS WINDOWS, podržava tzv. wizard metodologiju za kreiranje formi. Specifičnosti u izradi formi nisu predmet razmatranja ove knjige, pa se čitaoci upućuju na literaturu, u zavisnosti koji su SUBP izabrali. Ovde će biti definisane neke opšte postavke koje se moraju poštovati prilikom definisanja ekranskih formi.

Dakle, ekranske forme treba da ispune sledeće karakteristike:

- opšte kakrakteristike:
  - svaka forma mora imati naslov;
  - formi dati samo ono što korisniku treba;
  - obezbediti simetriju i balans na ekranu;
  - u slučaju pojavljivanja više formi, naznačiti u kojoj se formi korisnik nalazi;
  - omogućiti korišćenje malih i velikih slova u tekstu;
  - definisati praznu liniju između svakog paragrafa;
  - tekst poravnavati na levu stranu;
  - biti pažljiv sa skraćenicama i akronimima;
- tabele i liste:
  - redovi i kolone u listama moraju imati imena;
  - listeređati u prepoznatljivom redosledu;
  - koristiti vertikalne kolone, jer su preglednije za čitanje;
  - tekst u kolonama poravnavati nalevo, a brojeve nadesno;
  - svaka peta linija treba da bude prazna linija;
  - ostaviti bar dva mesta prazna između kolona;
  - numerisati liste počev od 1, a ne od 0;
  - razbiti niz slova u kraće podnizove;
- naglašavanje:
  - ne preterivati sa naglašavanjem;
  - treptanje i zvuke upotrebljavati samo kao alarm;
  - naglašeni tekst treba da bude razumljiv;
  - boje birati iz sredine duginog spektra;
  - biti dosledan u prikazima i naglašavanju;
- unos podataka:

- pri unosu podataka iz formulara kopirati izgled formulara;
- grupisati polja po kategorijama i staviti nazive;
- ne unositi unapred poznate podatke;
- u svakom polju obezbediti memorisanje polja;
- uvesti kad je moguće podrazumevanu (default) vrednost;
- omogućiti help na nivou polja;
- omogućiti slobodno kretanje među poljima za unos podataka;
- forma je jedinica prenosa podataka;
- omogućiti korisniku da odustane od unosa.

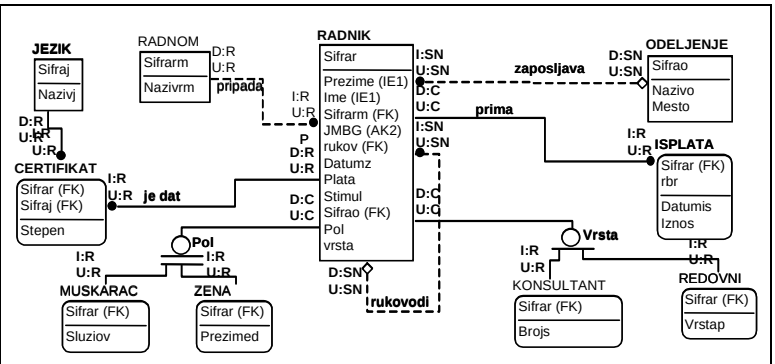
Na slici 4.32. prikazan je primer realizovane forme za dokument "Karton isplata". Forma je podeljena u jednu glavnu formu i dve podforme. U glavnoj formi definišu se ekranska polja o radnicima, kao što su: SIFRAR, PREZIME, IME, PLATA, STIMUL i DATUMZ koji odgovaraju definisanim kolonama u okviru tabele RADNIK. Ekransko polje SIFRAR odgovara koloni Sifrar u tabeli RADNIK koji je definisan kao primarni ključ.

U okviru glavne forme definisana su ekranska polja: SIFRAO i SIFRARM kao tzv. ComboBox za prenesene ključeve Sifrao i Sifrarm u tabeli RADNIK. ComboBox nudi listu za izbor iz tabela: SIFRARM i ODELJENJE, gde su Sifrao i Sifrarm primarni ključevi (videti sliku 4.32).

U okviru glavne forme realizovane su specijalizacije Pol i Vrsta. Prva specijalizacija Pol zahteva obavezan unos podatka o radniku, dok druga specijalizacija Vrsta (Tip zaposlenog sa slike 4.32) ne zahteva unos, već se ažuriranje radi po potrebi (default NULL opcija).

Definisane su i dve podforme vezane za: isplate radnika i stepen poznavanja jezika (slika 4.32). Podforma sadrži vezu sa glavnom formom. Veza između polja glavne forme i podforme ostvaruje se preko veznih polja sa glavne forme i veznih polja na podformi. Za podformu isplate radnika *vezno polje je Sifrar* koje je definisano u glavnoj formi i podformi a koje je prikazano i u fizičkom modelu podataka.

Podforma stepen poznavanja jezika, vezana je za glavnu formu preko polja Sifrar sa tabelom CERTIFIKAT. U okviru ove podforme je definisano ekransko polje JEZIK koji uspostavlja vezu sa tabelom JEZIK preko ComboBox za polje Sifraj.

Fizički model podataka za primer dokumenta "Karton isplata"	

Ekranska forma za dokument "Karton isplata"	

Slika 4.32. Ekranska forma za dokument "Karton isplata"

Na osnovu prethodno izvedenih aktivnosti (slika 4.31) u sledećem koraku izvodi se definisanje upita.

Aktivnost
3.3.3. Definisanje upita

U okviru aktivnosti "3.3.3. Definisanje upita" pokazan je na dosadašnjem primeru fizički realizovan model podataka u SUBP. Predmet posmatranja je set zajedničkih komandi i funkcija definisanih ISO standardom za SQL i realizovanih u različitim SUBP. Upiti su testirani u okviru MS ACCESS-a. Specifičnosti SQL za MS ACCESS nisu razmatrane već se čitaoci upućuju na literaturuŠ4Ć.

Za prikaz načina postavljanja upita koristi se deo modela baze podataka sa slike 4.20. prikazan na slici 4.33.

Slika 4.33. Model MS ACCESS baze podataka za tabele: RADNIK i ODELENJE

Pozivanje i dobijanje željenih podataka iz baze podataka su najčešće SQL operacije. Ovaj postupak se naziva QUERY (query znači pitanje ili upit), a izvršava se SELECT naredbama.

Osnovna podela upita je na:

- upiti nad jednom tabelom i
- upiti nad više tabela.

Upit nad jednom tabelom

Najjednostavniji način izdvajanja podataka je iz samo jedne tabele. Potrebna sintaksa za ovaj oblik upita ima sledeći izgled:

```
SELECT kolona [,kolona...]
 FROM tabela
```

SELECT izvlači redove i kolone iz jedne ili više tabela. U stvari, SELECT naredba je, sama po sebi, upit, a može biti i podupit kada je klauzula u drugoj naredbi.

SELECT klauzula se uvek prva upisuje i odmah iza nje sledi FROM klauzula i njome se pretražuju informacije iz baze podataka, implementirajući sve operatore relacione algebre. Klauzule moraju biti prikazane jedna iznad druge kada su korišćene zajedno.

Očigledno je da SELECT odgovara operaciji *projekcije* u relacionoj algebri. Iskaz FROM se može smatrati ekvivalentnim operaciji *Dekartovog proizvoda* (u relacionoj algebri), a iskaz WHERE (o čemu će biti više reči kasnije) ekvivalentnim operaciji *selekcije* u relacionoj algebri, jer *uslov* zadovoljavaju selektovane n-torke u rezultatu (videti prilog br. 1).

Prvo treba pogledati podatke u tabeli ODELJENJE, a potom i u tabeli RADNIK.

Ako se izlistaju svi redovi i sve kolone tabele ODELJENJE, upit koji će dati željeni rezultat je:

```
SELECT SIFRAO, NAZIVO, MESTO
 FROM ODELJENJE
```

Rezultat SQL upita je sledeći:

SIFRAO	NAZIVO	MESTO
10	PRIPREMA	PANCEVO
20	RAZVOJ	NOVI_SAD
30	PRODAJA	BEOGRAD
40	PROIZVODNJA	PAZOVA

U ovom primeru upita izlistana su imena svih kolona tabele ODELJENJE (SIFRAO, NAZIV, MESTO) SELECT naredbom.

Međutim, mnogo jednostavnije je da se izlistaju sve kolone neke tabele korišćenjem znaka "*", odnosno (SELECT *), umesto da se nabrajaju sve kolone tabele pojedinačno.

Na primeru tabele RADNIK to izgleda ovako:

```
SELECT *
 FROM RADNIK
```

Rezultat SQL upita je:

SIFRA	PREZIM	IME	SI	JMBG	RUKO	DATUMZ	PLAT	STIM	SI	P	V
827369	STEVIC	ZORAN	0	141195217103	827902	28/03/98	8000	600	2	M	2
827499	ALAGIC	MILAN	0	250396434561	827698	28/03/98	1600	600	3	M	
827521	VUKIC	MILOS	0	113049705543	827698	28/03/98	1250	600	3	M	2
827566	JOVIC	MIRA	0	113049705643	827839	28/04/90	2975	700	2	Z	1
827654	MARTIC	ZORA	0	114049605544	827698	12/06/88	1250	1000	3	Z	1
827698	BOBIC	IVAN	0	140598705545	827839	15/11/89	2850	800	3	M	2
827782	CEBIC	GORAN	0	120397065456	827839	11/06/80	2450	500	1	M	2
827788	SUSIC	ZORAN	0	130495467675	827566	23/10/91	3000	650	2	M	2
827839	KLJAKIC	STEVA	0	131295212234	827839	06/12/87	5000	600	1	M	2
827844	TUBIC	MIRA	0	131295434212	827698	08/03/78	1500	600	3	Z	1
827876	ALIMPIC	PETAR	0	250397634356	827788	09/05/92	1100	780	2	M	
827900	JAKIC	VLADA	0	121196545723	827698	28/03/90	9500	900	3	M	
827902	FILIPIC	DRAGA	0	121097053422	827566	20/11/96	3000	1000	2	M	2
827934	MILIC	DRAGA	0	070695645646	827782	28/03/93	1300	790	1	M	2

Ako korisnik ne želi da vidi sve kolone tabele, onda u SELECT klauzulu uključuje imena samo onih kolona koje želi da vidi, kao što je prikazano sledećim upitom.

```
SELECT NAZIVO, SIFRAO
 FROM ODELJENJE
```

Rezultat SQL upita je:

NAZIVO	SIFRA
PRIPREMA	10
RAZVOJ	20
PRODAJA	30
PROIZVODNJA	40

Eliminacija duplih redova - DISTINCT

Kako se u pojedinim kolonama ponavljaju podaci, to se opcijom DISTINCT prikazuju samo različiti podaci za izabranu kolonu.

Dograđena sintaksa naredbe SELECT ima sledeći izgled:

```
SELECT [DISTINCT] kolona [,kolona...]
 FROM tabela
```

Ključna reč DISTINCT će u sledećem upitu značiti:

"Pokaži mi sve različite (DISTINCT) vrste radnih mesta u tabeli RADNIK."

```
SELECT DISTINCT SIFRARM
 FROM RADNIK
```

Iako se u tabeli RADNIK nalazi ukupno četrnaest radnih mesta koja odgovaraju broju radnika, u ovoj tabeli se nalazi samo pet različitih radnih mesta.

Rezultat SQL upita je:

SIFRAR
01
02
03
04
05

Izbor specificiranih redova WHERE klauzula

S obzirom na to da SELECT klauzula omogućava da se dobiju željene kolone iz tabele, da bi se dobili određeni *redovi* iz tabele, potrebno je dodati WHERE klauzulu u SELECT naredbi.

Dograđena sintaksa naredbe SELECT ima sledeći izgled:

```
SELECT [DISTINCT] kolona [,kolona...]
FROM tabela
[WHERE uslov_selekcije]
```

WHERE klauzula odgovara operatoru *restrikcije* u relacionoj algebri. U WHERE klauzuli upoređuju se vrednosti kolona, literal-vrednosti, aritmetički izrazi ili funkcije.

Uslovi selekcije u WHERE klauzuli su:

- operatori poređenja (kao što su =, >, >=, <, <=),
- operatori ranga (BETWEEN i NOT BETWEEN),
- liste (IN, NOT IN),
- uzorci (LIKE i NOT LIKE),
- nepoznate vrednosti (IS NULL i IS NOT NULL),
- višestruki uslovi pretraživanja (AND,OR).

Operatori poređenja

Npr., treba selektovati kolone: Sifrar, Prezime, Ime, Plata, Sifrao iz tabele RADNIK za odeljenje 30 operatorom poređenja "=":

```
SELECT Sifrar, Prezime, Ime, Plata, Sifrao
 FROM RADNIK
 WHERE SIFRAO = '30'
```

Napomena: Ako su brojevi između navodnika to znači da su definisani kao tekst-podaci (pogledati sliku 4.6).

WHERE klauzula inicira SUBP da pronađe željeni podatak iz tabele, odnosno da pozove one redove koji odgovaraju postavljenom uslovu pretraživanja u ovoj klauzuli (WHERE SIFRAO= 30).

Rezultat SQL upita je:

Sifrar	Prezim	Ime	Plata	Sifrao
827499	ALAGIC	MILAN	16000	30
827521	VUKIC	MILOS	12500	30
827654	MARTIC	ZORA	12500	30
827698	BOBIC	IVAN	28500	30
827844	TUBIC	MIRA	15000	30
827900	JAKIC	VLADA	9500	30

Treba zatim posmatrati i primer selektovanja kolona: Sifrar, Prezime, Ime, Plata, Sifrao tabele RADNIK koji NE rade u odeljenju 30, operatorom negacije NOT i operatorom poređenja "=":

```
SELECT Sifrar, Prezime, Ime, Plata, Sifrao
FROM RADNIK
WHERE NOT ( SIFRAO = '30')
```

Rezultat SQL upita je:

Sifrar	Prezime	Ime	Plata:	Sifrao
827369	STEVIC	ZORAN	8000	20
827566	JOVIC	MIRA	29750	20
827782	CEBIC	GORAN	24500	10
827788	SUSIC	ZORAN	30000	20
827839	KLJAKIC	STEVAN	50000	10
827876	ALIMPIC	PETAR	11000	20
827902	FILIPIC	DRAGAN	30000	20
827934	MILIC	DRAGAN	13000	10

Operatori ranga *(BETWEEN i NOT BETWEEN)*

Operator BETWEEN omogućava da se izaberu redovi koji sadrže vrednosti u nekom rasponu, a koje je korisnik specificirao.

Na primer, treba videti listu svih zaposlenih Radnika, čija je plata u rasponu 12000 i 14000.

```
SELECT PREZIME, PLATA
 FROM RADNIK
 WHERE PLATA BETWEEN 12000 AND 14000
```

Rezultat SQL upita je:

PREZIM	PLATA
VUKIC	12500
MARTIC	12500
MILIC	13000

Operator NOT BETWEEN omogućava da se izaberu redovi koji su van vrednosti u nekom rasponu, a koje je korisnik specificirao.

Na primer, treba videti listu svih zaposlenih RADNIKa, čija plata NIJE u rasponu 12000 i 14000.

```
SELECT PREZIME, PLATA
 FROM RADNIK
 WHERE PLATA NOT BETWEEN 12000 AND 14000
```

Rezultat SQL upita je:

PREZIME	PLATA
STEVIC	8000
ALAGIC	16000
JOVIC	29750
BOBIC	28500
CEBIC	24500
SUSIC	30000
KLJAKIC	50000
TUBIC	15000
ALIMPIC	11000
JAKIC	9500
FILIPIC	30000

Operatori za liste (IN,NOT IN)

IN *operator* omogućava da se izaberu redovi koji sadrže vrednost koja je jednaka jednoj od vrednosti navedene liste.

Npr., treba pogledati sva odeljenja čiji je broj ili 10 ili 30.

```
SELECT *
 FROM ODELJENJE
 WHERE SIFRAO IN ('10','30')
```

Lista vrednosti se smešta u zagrade - npr. (10,30).

Za ovaj upit može se upotrebiti i OR konektor kao:

```
WHERE SIFRAO= '10' OR SIFRAO= '30'
```

i dobio bi se isti rezultat:

SIFRAO	NAZIVO	MESTO
10	PRIPREMA	PANCEVO
30	PRODAJA	BEOGRAD

NOT IN *operator* omogućava da se izaberu redovi koji NE sadrže vrednost koja je jednaka

jednoj od vrednosti navedene liste.

Na primer, pogledati sva odeljenja čiji broj nije 10 ili 30.

```
SELECT *
FROM ODELJENJE
WHERE SIFRAO NOT IN ('10', '30')
```

Rezultat SQL upita je:

Sifrao	Naziv	Mesto
20	RAZVOJ	NOVI_SAD
40	PROIZVODNJA	PAZOVA

Operatori za definisanje uzoraka (LIKE i NOT LIKE)

Korišćenjem LIKE *operatora* mogu se selektirati redovi koji odgovaraju uzorku nekog karaktera ili specificiranom broju. U primeru je dat spisak svih radnika koji imaju "U" kao drugo slovo u prezimenu.

```
SELECT PREZIME
 FROM RADNIK
 WHERE PREZIME LIKE "?U*"
```

U ovom primeru korišćen je SQL LIKE operator, da bi se SQL usmerio na pozivanje svih onih redova iz tabele RADNIK, čija kolona PREZIME sadrži vrednost koja odgovara LIKE klauzuli. U navedenom uzorku to je ('?U*'). Upitnik (?) označava poziciju jednog karaktera, a znak * označava bilo koji niz nula ili niz više karaktera.

Rezultat SQL upita je:

PREZIM
VUKIC
SUSIC
TUBIC

Treba, međutim, pokazati NOT LIKE operator i na primeru spiska svih Radnika koji nemaju "U" kao drugo slovo u prezimenu.

```
SELECT PREZIME
 FROM RADNIK
 WHERE PREZIME NOT LIKE '?U*'
```

Rezultat SQL upita je:

Prezime
STEVIC
ALAGIC
JOVIC
MARTIC
BOBIC
CEBIC
KLJAKIC
JAKIC
FILIPIC
MILIC

Definisanje nepoznatih vrednosti (IS NULL, IS NOT NULL)

Korišćenjem IS NULL *operatora* mogu se selektirati redovi koji su NULL. U primeru je dat spisak svih radnika kojima nije definisana kolona Vrsta.

```
SELECT Prezime,Ime,Vrsta
FROM RADNIK
WHERE VRSTA IS NULL;
```

Prezime	Ime	Vrsta:
ALAGIC	MILAN	
ALIMPIC	PETAR	
JAKIC	VLADA	

Korišćenjem IS NOT NULL *operatora* mogu se selektirati redovi koji su NOT NULL. U primeru je dat spisak svih radnika kojima je definisana kolona Vrsta.

```
SELECT Prezime, Ime, Vrsta
FROM RADNIK
WHERE VRSTA IS NOT NULL;
```

Prezime	Ime	Vrsta
STEVIC	ZORAN	2
VUKIC	MILOS	2
JOVIC	MIRA	1
MARTIC	ZORA	1
BOBIC	IVAN	2
CEBIC	GORAN	2
SUSIC	ZORAN	2
KLJAKIC	STEVAN	2
TUBIC	MIRA	1
FILIPIC	DRAGA	2
MILIC	DRAGA	2

Višestruki uslovi pretraživanja (AND,OR)

Višestruki uslov pretraživanja, tzv. AND *konektor* koristi se u okviru WHERE klauzule gde treba specificirati više od jednog uslova pretraživanja.

Na primer, pretpostavka je da se u dosadašnjem primeru želi dobiti lista DIREKTORA (03), koji zarađuju više od 28000 dinara.

```
SELECT PREZIME, SIFRARM, PLATA
 FROM RADNIK
 WHERE SIFRARM = '03'
 AND PLATA > 28000
```

Višestruki uslovi pretraživanja se vezuju za reč AND (SIFRARM = '03' AND PLATA > 28000). AND konektor znači da željeni podatak mora da "sretne" sve navedene uslove pretraživanja, pre nego što SQL da traženi red. U WHERE klauzulu pomoću AND konektora može se dodati neograničeni broj ovakvih uslova.

Rezultat SQL upita je:

PREZIME	SIFRARM	PLATA
JOVIC	03	29750
BOBIC	03	28500

Alternativni uslov pretraživanja, tzv. OR konektor selektuje redove koji udovoljavaju bilo kojem od više datih uslova.

Na primer, pretpostavka je da se u dosadašnjem primeru želi dobiti i lista DIREKTORA (03), *ili* svih onih koji zarađuju više od 28000 dinara.

```
SELECT PREZIME, SIFRARM, PLATA
FROM RADNIK
WHERE SIFRARM ='03'
OR PLATA > 28000
```

U ovom primeru uslov pretraživanja se povezuje sa rečju OR (SIFRARM = '03' OR PLATA > 28000). OR znači da ako naredni podatak udovoljava jednom od uslova, SQL će izdati željeni red.

Rezultat SQL upita je:

PREZIME	SIFRAR	PLATA
JOVIC	03	29750
BOBIC	03	28500
CEBIC	03	24500
SUSIC	04	30000
KLJAKIC	05	50000
FILIPIC	04	30000

Izračunavanje vrednosti u SELECT listi

Aritmetički izrazi se koriste za povezivanje imena kolona i konstantne numeričke vrednosti sa aritmetičkim operatorom. Npr., treba izlistati prezime, platu, stimulaciju i sumu plate i stimulacije za sve Radnike sa SIFRARM= '02'.

```
SELECT PREZIME,PLATA,STIMUL,PLATA+STIMUL
FROM RADNIK
WHERE SIFRARM = '02'
```

Može se primetiti da je aritmetički izraz (PLATA+STIMUL) prikazan kao nova kolona na dobijenoj tabeli. Mada ova kolona nije realna kolona (kolona iz baze), ona se može koristiti za sve postupke kao i bazne kolone.

Alijas kolona se koristi za posebno označavanje prikaza rezultata selektovane kolone, što je u sledećem primeru i pokazano.

```
SELECT PREZIME,PLATA,STIMUL,PLATA+STIMUL AS UKUPNO
FROM RADNIK
WHERE SIFRARM = '02'
```

Rezultat SQL upita je:

PREZIME	PLATA	STIMUL	UKUPN
ALAGIC	16000	600	16600
VUKIC	12500	600	13100
MARTIC	12500	1000	13500
TUBIC	15000	600	15600

Kreiranje nove tabele pomoću SELECT INTO

Ako se želi SELECT naredbom kreirati nova tabela, treba koristiti naredbu SELECT...INTO koja ima sledeći izgled:

```
SELECT [DISTINCT] kolona [,kolona...]
INTO nova_tabela
FROM tabela
[WHERE uslov-selekcije]
```

Na primer, treba kreirati tabelu PRIMANJA ako se iz tabele RADNIK selektuju prezime i suma plate i stimulacije za sve Radnike sa SIFRARM= '02'.

```
SELECT PREZIME,PLATA+STIMUL AS UKUPNO
INTO PRIMANJA
 FROM RADNIK
 WHERE SIFRARM = '02'
```

Novoformirana tabela ima sledeći izgled:

PRIMANJA

PREZIME	UKUPNO
ALAGIC	16600
VUKIC	13100
MARTIC	13500
TUBIC	15600

Sortiranje redova pomoću ORDER BY klauzule

Korišćenjem klauzule ORDER BY kontroliše se redosled prikazivanja redova. Ova klauzula se dodaje na kraju SELECT naredbe.

Dograđena sintaksa naredbe SELECT ima sledeći izgled:

```
SELECT [DISTINCT] kolona [,kolona...]
FROM tabela
[WHERE uslov-selekcije]
ORDER BY (izraz|pozicija)[ASC | DESC]
```

Na primer, ako se želi prikazati lista zaposlenih u odeljenju 30, ali da odgovarajući redovi budu prikazani po stavci plate, u rastućem redosledu treba napisati:

```
SELECT PLATA, SIFRARM, PREZIME
FROM RADNIK
WHERE SIFRAO= '30'
ORDER BY PLATA
```

ORDER BY klauzula prouzrokuje sortiranje redova u rastućem nizu (ASC), tako da je najmanja plata na prvom mestu liste. Rastući niz (ASC) postavljen je po defaultu i ovu klauzulu ne treba eksplicitno naglašavati.

Rezultat SQL upita je:

PLATA	SIFRAR	PREZIM
9500	01	JAKIC
12500	02	MARTIC
12500	02	VUKIC
15000	02	TUBIC
16000	02	ALAGIC
28500	03	BOBIC

Opadajući niz se definiše DESC *klauzulom.* Tako, ako bi se tražila lista zaposlenih u redosledu radnih mesta i u okviru toga redosleda prikazale njihove zarade u opadajućem nizu, trebalo bi koristiti DESC klauzulu.

```
SELECT SIFRARM,PLATA, PREZIME
FROM RADNIK
ORDER BY SIFRARM, PLATA DESC
```

Podaci u koloni SIFRARM su složeni po alfabetskom redosledu i nad njom se primenjuje ORDER BY klauzula. U tabeli se može videti da je redosled zaposlenih dat po opadajućoj mesečnoj zaradi.

Rezultat SQL upita je:

SIFRARM	PLATA	PREZIME
01	13000	MILIC
01	11000	ALIMPIC
01	9500	JAKIC
01	8000	STEVIC
02	16000	ALAGIC
02	15000	TUBIC
02	12500	MARTIC
02	12500	VUKIC
03	29750	JOVIC
03	28500	BOBIC
03	24500	CEBIC
04	30000	FILIPIC
04	30000	SUSIC
05	50000	KLJAKIC

Korišćenje GROUP BY klauzule

GROUP BY *klauzula* logički deli tabelu na grupe n-torki tako da u okviru jedne grupe sve n-torke imaju istu vrednost zadate kolone. Ovim se omogućuje da funkcije za dobijanje sumarnih informacija budu primenjene na svaku ovakvu grupu posebno, umesto na celu

tabelu.

Sintaksa ove klauzule ima sledeći izgled:

```
SELECT [DISTINCT] kolona [,kolona...]
FROM tabela
[WHERE uslov-selekcije]
[GROUP BY izraz {,izraz}]
ORDER BY (izraz|pozicija)[ASC | DESC]
```

Na primer, treba pogledati sledeći primer:

```
SELECT SIFRAO, MAX (PLATA)
FROM RADNIK
GROUP BY SIFRAO
```

Rezultat SQL upita je:

SIFRAO	MAX
10	50000
20	30000
30	28500

Svaki red dobijenog rezultata reprezentuje jednu grupu redova, memorisanu u tabeli RADNIK, tj. svaki je red tabele RADNIK stavljen u jednu od tri grupe.

Dakle, GROUP BY klauzula omogućuje dobijanje sumarnih informacija za svaku različitu vrednost kolone po kojoj se vrši grupisanje.

Korišćenje GROUP BY u okviru WHERE klauzule

Kada se koristi GROUP BY u okviru WHERE klauzule onda se redovi koji ne zadovoljavaju uslov u okviru WHERE klauzule eliminišu pre grupisanja. Npr., treba prikazati radno mesto, srednju aritmetičku vrednost i broj radnika čija je plata veća od 25000 i izvršiti grupisanje po SIFRARM.

```
SELECT SIFRARM, AVG (PLATA), COUNT (*)
FROM RADNIK
 WHERE PLATA > 25000
 GROUP BY SIFRARM;
```

Rezultat SQL upita je:

Sifrarm:	AVG	COUNT (*)
03	29125	2
04	30000	2
05	50000	1

Korišćenje kombinacije klauzule GROUP BY i ORDER BY

Dodavanjem ORDER BY klauzule u SELECT naredbi, kao rezultat, dobija se redosled prikazivanja redova u rastućem redu (GROUP BY mora biti ispred ORDER BY). Na primer, ako se posmatraju radnici koji rade u istim odeljenjima, treba ih definisati u grupe i onda naći srednju aritmetičku vrednost u svakoj od tih grupa i pri tom prikazati sve to u rastućem nizu.

```
SELECT SIFRARM, AVG (PLATA)
FROM RADNIK
GROUP BY SIFRARM
ORDER BY AVG (PLATA);
```

Rezultat SQL upita je:

Sifrarm:	AVG
01	10375
02	14000
03	27583
04	30000
05	50000

Korišćenje HAVING klauzule

Klauzula HAVING koristi se zajedno sa GROUP BY i za grupu n-torki i ima isti efekat kao WHERE klauzula za pojedinačne n-torke. Klauzula HAVING ima zadatak da specificira uslove pretraživanja u okviru GROUP BY klauzule.

Sintaksa ove klauzule ima sledeći izgled:

```
SELECT [DISTINCT] kolona [,kolona...]
FROM tabela
[WHERE uslov-selekcije]
[GROUP BY izraz {,izraz}]
[HAVING grupni uslov]
ORDER BY (izraz|pozicija)[ASC | DESC]
```

Treba prikazati šifru odeljenja i srednju aritmetičku vrednost za odeljenje koje ima više od tri zaposlena radnika u odeljenju korišćenjem HAVING klauzule.

```
SELECT SIFRAO, AVG (PLATA)
FROM RADNIK
GROUP BY SIFRAO
HAVING COUNT (*) > 3;
```

Rezultat SQL upita je:

Sifrao	AVG (PLATA)
20	18125.2
30	15666.7

Grupni upiti

Grupni upiti vraćaju rezultate koji su karakteristika neke grupe redova umesto jednog reda. Redovi koji zadovoljavaju uslov selekcije predstavljaju grupu nad kojom se izračunava jedna ili više grupnih funkcija.

Na primer, korišćenjem COUNT *funkcije* treba izračunati broj zaposlenih radnika u odeljenju 20:

```
SELECT COUNT (*)
FROM RADNIK
WHERE SIFRAO = '20'
```

Rezultat ovog upita je:

COUNT
6

Pored grupne funkcije COUNT, mogu se definisati i sledeće grupne funkcije nad *numeričkim vrednostima:*

- AVG - izračunava srednju aritmetičku vrednost;
- SUM - izračunava ukupni zbir;
- MIN - nalazi minimalnu vrednost;
- MAX - nalazi maksimalnu vrednost.

Funkcija AVG ([DISTINCT|ALL],n) izračunava srednju aritmetičku vrednost ignorišući null vrednosti.

Tako, ako se želi izračunati srednja aritmetička vrednost plata radnika, treba pisati:

```
SELECT AVG (PLATA)AS SREDNJA_ARIT_VRED
FROM RADNIK
```

Rezultat SQL upita je:

SREDNJA_ARIT_VRE
19350.066

Funkcija SUM ([DISTINCT|ALL]expr) izračunava ukupni zbir vrednosti izraza expr.

Tako, ako se želi izračunati ukupan zbir plata radnika za SIFRARM = '02', treba pisati:

```
SELECT SUM (PLATA)
FROM RADNIK
WHERE SIFRARM='02'
```

Rezultat SQL upita je:

SUM (PLATA)
56000

Funkcija MIN ([DISTINCT|ALL]expr) izračunava minimalnu vrednost izraza expr.

Tako, ako se želi izračunati najmanja plata radnika za SIFRARM = '02', treba pisati:

```
SELECT MIN (PLATA)
FROM RADNIK
WHERE SIFRARM='02'
```

Rezultat SQL upita je:

MIN (PLATA)
12500

Funkcija MAX (ŠDISTINCTđALLĆexpr) izračunava maksimalnu vrednost izraza expr.

Tako, ako se želi izračunati maksimalna plata radnika za SIFRARM = '02', treba pisati:

```
SELECT MAX (PLATA)
FROM RADNIK
WHERE SIFRARM='02'
```

Rezultat SQL upita je:

MAX (PLATA)
16000

Upiti nad više tabela

Upiti nad više tabela se realizuju tzv. ulaganjem jednog upita u drugi ili preko upita spajanja (join queries).

Postavljanje kompleksa upita u okiru postojećeg (PODUPITI)

Podupit je SELECT naredba ugnježdena sa drugom SELECT naredbom koja vraća odgovarajući međurezultat.

Ono što daje posebnu snagu SQL je i to što se može praviti čitav kompleks upita u okviru postojećih jednostavnih upita. Upiti se mogu koristiti da se dinamički napravi uslov pretraživanja za glavni upit.

Podupit sa povratkom jednog reda

Podupit sa povratkom jednog reda treba posmatrati kroz primer definisanja najmanje plate u tabeli ODELJENJE.

Podupiti se rešavaju u dva koraka.

Prvo se nalazi minimalna plata u tabeli RADNIK:

```
SELECT MIN (PLATA) FROM RADNIK
```

Rezultat SQL upita je:

MIN (PLATA)
8000

Drugi korak je da se nađe ime radnika i posao sa najmanjom platom u tabeli RADNIK korišćenjem ugnježdenog podupita:

```
SELECT PREZIME,SIFRARM,PLATA
FROM RADNIK
WHERE PLATA = (SELECT MIN (PLATA)
FROM RADNIK)
```

Rezultat SQL upita je:

PREZIME	SIFRARM	PLATA
STEVIC	01	8000

Za podupit sa povratkom jednog reda koriste se komparacioni ili logički operatori: =, <, >, <= i dr.

Podupit sa povratkom skupa vrednosti (više redova)

Podupit sa povratkom skupa vrednosti (više redova) definiše se ako je rezultat podupita skup vrednosti u WHERE delu osnovnog upita. To je, takođe, ugnježdeni podupit koji se definiše preko dva upitna bloka: osnovnog i skrivenog.

Pod pretpostavkom da treba napraviti listu svih zaposlenih sa istim zanimanjem kao radnik JOVIC u tabeli RADNIK, upit izgleda ovako:

```
SELECT PREZIME,SIFRARM
FROM RADNIK
WHERE SIFRARM=
(SELECT SIFRARM FROM RADNIK WHERE PREZIME = 'JOVIC')
```

Skriveni podupit se obrađuje pre glavnog upita, jer je rezultat podupita potreban da bi se našao rezultat glavnog upita. Druga SELECT naredba u ovom primeru, u okviru skrivenog podupita, izdvojila je vrednost 03, pošto je to zanimanje radnika JOVIC u tabeli RADNIK.

Rezultat SQL upita je:

PREZIM	SIFRAR
JOVIC	03
BOBIC	03
CEBIC	03

Na primer, ako se žele liste svih radnika koji *zarađuju više od proseka* treba napisati:

```
SELECT PREZIME,PLATA
 FROM RADNIK
 WHERE PLATA >
 (SELECT AVG (PLATA)
      FROM RADNIK)
```

Rezultat SQL upita je:

PREZIME	PLATA
JOVIC	29750
BOBIC	28500
CEBIC	24500
SUSIC	30000
KLJAKIC	50000
FILIPIC	30000

Kao što se u gornjim primerima vidi, i za podupite sa povratkom više redova mogu se koristiti komparacioni ili logički operatori: =, <, >, <= i dr.

Podupit sa operatorom IN

Podupit se može upotrebiti i umesto liste vrednosti iza *operatora* IN. U tom slučaju podupit može da vraća više redova, od kojih se svaki posmatra kao pojedina vrednost u listi.

Na primer, definisana lista Radnika u odeljenju 10 sa istim SIFRARM u odnosu na odeljenje 30 izgleda ovako:

```
SELECT PREZIME,SIFRARM
FROM RADNIK
WHERE SIFRAO= '10'
AND SIFRARM IN
(SELECT SIFRARM
FROM RADNIK
WHERE SIFRAO= '30')
```

Rezultat SQL upita je:

PREZIM	SIFRAR
CEBIC	03
MILIC	01

Podupit sa operatorom NOT IN

Način korišćenja operatora negacije NOT IN biće pokazan kroz sledeći primer. Pre svega, treba definisati listu Radnika u odeljenju 10 sa SIFRARM koji NE pripadaju odeljenju 30.

```
SELECT PREZIME,SIFRARM
 FROM RADNIK
 WHERE SIFRAO = '10'
 AND SIFRARM NOT IN
 (SELECT SIFRARM
 FROM RADNIK
 WHERE SIFRAO = '30')
```

Rezultat SQL upita je:

Prezime	Sifrarm
KLJAKIC	05

Korišćenje ANY ili ALL operatora

ANY *ili* ALL *operatori* se koriste za definisanje podupita kojima se vraća više od jednog reda. Ovi operatori se koriste u okviru WHERE ili HAVING klauzula u konjukciji sa logičkim operatorima (=, !=, >, >=, <,<=).

ANY *operator* vrši upoređivanje za svaku vrednost koja se vraća iz podupita.

Na primer, treba prikazati Radnike koji zarađuju više (>) od najniže plate (ANY) radnika iz odeljenja 30.

```
SELECT PREZIME,PLATA,SIFRAO
 FROM RADNIK
 WHERE PLATA > ANY
 (SELECT PLATA
 FROM RADNIK
```

```
WHERE SIFRAO = '30')
```

Rezultat SQL upita je:

PREZIME	PLATA	SIFRAO
ALAGIC	16000	30
VUKIC	12500	30
JOVIC	29750	20
MARTIC	12500	30
BOBIC	28500	30
CEBIC	24500	10
SUSIC	30000	20
KLJAKIC	50000	10
TUBIC	15000	30
ALIMPIC	11000	20
FILIPIC	30000	20
MILIC	13000	10

Podupit se izvodi u sledećim koracima:

1. nalazi sve plate u odeljenju 30.
2. selektira SVAKOG (ANY) radnika koji zarađuje više od najmanje plate u odeljenju 30.

Najnižu platu u odeljenju 30 ima JAKIC i ona iznosi 9500 dinara. Osnovni upit vraća radnike čija je plata veća od najniže plate u odeljenju 30. Dakle, ">ANY" definiše više od minimuma.

Opcija "=ANY" je ekvivalentna IN operatoru.

ALL *operator* vrši upoređivanje za sve vrednosti koje se vraćaju iz podupita.

1. nalazi sve plate u odeljenju 30.
2. selektira sve (ALL) radnike koji zarađuju više od NAJVIŠE plate u odeljenju 30.

Najveću platu u odeljenju 30 ima BOBIC i iznosi 28500 dinara. Osnovni upit vraća radnike čija je plata veća od NAJVEĆE plate u odeljenju 30. Dakle, ">ALL" definiše više od MAKSIMUMA. Supstitucija IN je "=ANY " i NOT IN za "!=ALL"

Upit kojim se prikazuju radnici koji zarađuju više (>)od SVIH (ALL) radnika iz odeljenja 30 ima sledeći izgled:

```
SELECT PREZIME,PLATA,SIFRAO
FROM RADNIK
WHERE PLATA > ALL
(SELECT PLATA
FROM RADNIK
WHERE SIFRAO = '30')
```

Rezultat SQL upita je:

PREZIME	PLATA	SIFRAO
JOVIC	29750	20
SUSIC	30000	20
KLJAKIC	50000	10
FILIPIC	30000	20

Ugnježdenje podupita korišćenjem HAVING klauzule

Ugnježdeni podupit mora obavezno koristiti HAVING *klauzulu,* jer se WHERE klauzula odnosi na pojedine redove, dok se sa HAVING izvodi grupisanje redova specificiranih u okviru GROUP BY klauzule. Na primer, treba prikazati odeljenja čija je srednja plata veća od srednje plate u odeljenju 30.

```
SELECT SIFRAO, AVG (PLATA)
FROM RADNIK
GROUP BY SIFRAO
HAVING AVG (PLATA) >
 (SELECT AVG (PLATA)
 FROM RADNIK
 WHERE SIFRAO = '30')
```

Rezultat SQL upita je:

SIFRAO	AVG
10	29166.66
20	18125.16

Višestepeni podupiti

Poseban predmet razmatranja u definisanju podupita su *višestepeni podupiti* kojima se definiše više uzastopnih upita.

Kao primer višestepenih podupita biće definisani sledeći primeri:

1. Treba izlistati imena radnika koji rade isti posao kao JOVIC ili imaju platu koja je veća od plate ili jednaka kao plata FILIPIC, i to u rastućem nizu SIFRARM i PLATA:

```
SELECT PREZIME, SIFRARM, SIFRAO, PLATA
 FROM RADNIK
 WHERE SIFRARM =
      (SELECT SIFRARM
      FROM RADNIK
      WHERE PREZIME=' JOVIC')
 OR PLATA >
 (SELECT PLATA
 FROM RADNIK
 WHERE PREZIME= 'FILIPIC')
      ORDER BY SIFRARM, PLATA
```

Rezultat SQL upita je:

PREZIME	SIFRARM	SIFRAO	PLATA
KLJAKIC	05	10	50000

2. Potrebno je selektovati PREZIME i SIFRARM radnika u odeljenju 10 SA ISTIM SIFRARM *kao BILO KO* u odeljenju RAZVOJ:

```
SELECT PREZIME, PLATA
 FROM RADNIK
 WHERE SIFRAO = '10'
 AND SIFRARM IN
 (SELECT SIFRARM
 FROM RADNIK
 WHERE SIFRAO =
 (SELECT SIFRAO
 FROM ODELJENJE
 WHERE NAZIVO = 'RAZVOJ'))
```

Rezultat SQL upita je:

PREZIME	PLATA
CEBIC	24500
MILIC	13000

3. Treba Izlistati SVE radnike koji zarađuju više od srednje vrednosti plata u sopstvenom odeljenju i to prikazati u rastućem redosledu SIFRAO.

```
SELECT SIFRAO, PREZIME, PLATA
 FROM RADNIK X
 WHERE PLATA >
 (SELECT AVG (PLATA)
 FROM RADNIK
 WHERE X.SIFRAO= SIFRAO)
 ORDER BY SIFRAO
```

Svakom redu tabele RADNIK dodeljuje se ALIAS X nakon čega se izvodi unutrašnji SELECT

gde se utvrđuje prosečna plata za odeljenje; ako je manja od plate iz glavnog selecta onda se prikazuje.

Rezultat SQL upita je:

SIFRAO	PREZIME	PLATA
10	KLJAKIC	50000
20	FILIPIC	30000
20	SUSIC	30000
30	BOBIC	28500
30	ALAGIC	16000

Povezivanje više tabela (JOIN)

Upiti koji se mogu realizovati ulaganjem upita mogu se dobiti i pomoću upita spajanja (JOIN), pa je način pisanja upita stvar izbora korisnika. JOIN korisniku dozvoljava da bira podatke iz dve ili više tabela i da kombinuje izabrane podatke u jednu rezultujuću tabelu. Dakle, ako se želi da u jednom upitu budu izdvojeni podaci iz više tabela, potrebno je tabele iz kojih se izdvajaju određeni podaci navesti u klauzuli FROM.

Sa stanovišta proceduralnog programiranja (korišćenje jezika FORTRAN, COBOL i dr.), procedura povezivanja dve tabele, biranje specifičnih polja iz specifičnih redova i sortiranje rezultata traže komplikovaniji program, pogotovu sa povećavanjem broja tabela koje se spajaju. To nije slučaj sa SQL, jer on radi neproceduralno, tj. kaže se KOJI se podatak želi, a ne KAKO da se dobije.

S obzirom na to, definisani su sledeći tipovi spajanja :

- spajanje na jednakost (Equal Join),
- korišćenje grupnih funkcuja u JOIN-u,
- spajanje na osnovu nejednakosti (Not-Equal Join),
- spajanje sa samim sobom (Self-Join),
- spoljno spajanje (Other Join).

Spajanje na jednakost (Equal Join)

Ukoliko treba saznati, na primer, gde radi radnik sa imenom ALAGIC, gledajući obe tabele može se videti da tabela RADNIK ne sadrži kolonu (MESTO) za lokaciju koju ima tabela ODELJENJE. Međutim, tabele: RADNIK i ODELJENJE imaju zajedničku kolonu koja sadrži broj odeljenja (SIFRAO).

Brojevi odeljenja su memorisani u obe tabele tako da su dozvoljene relacije između tabela: RADNIK i ODELJENJE. To se može postići korišćenjem JOIN upita gde se u FROM klauzuli navode imena tabela koje se ispituju, a u WHERE klauzuli imena odgovarajućih kolona (a to su kolone koje su zajedničke za obe tabele).

```
SELECT PREZIME,MESTO
FROM RADNIK,ODELJENJE
WHERE PREZIME = 'ALAGIC'
AND RADNIK.SIFRAO = ODELJENJE.SIFRAO
```

Ovaj uslov definiše odnos između tabela: RADNIK i ODELJENJE, tj. broj odeljenja (SIFRAO) u tabeli RADNIK odgovara broju odeljenja u tabeli ODELJENJE, što je omogućilo spajanje redova u zajednički.

WHERE klauzula sadrži i uslov PREZIME = 'ALAGIC', koji kaže da iz baze treba dohvatiti samo podatke za radnika ALAGIC-a. Tako se povezuje ALAGIC-ev red iz tabele RADNIK, koji sadrži vrednost 30 u polju SIFRAO, sa redom u tabeli ODELJENJE, koja, takođe, sadrži istu vrednost u polju SIFRAO. Kolone koje su listane SELECT klauzulom pokazuju da se iz baze uzima samo polje PREZIME iz tabele RADNIK i samo polje MESTO iz tabele ODELJENJE da bi se dobio željeni rezultat.

Rezultat SQL upita je:

PREZIME	MESTO
ALAGIC	BEOGRAD

Mogu se povezivati pojedini redovi kao u prethodnom primeru, delovi tabela ili cele tabele.

Ako se posmatra tzv. spajanje θ (videti prilog br.1) koje je najčešći slučaj, kada se izjednačavaju vrednosti kolona koje se navode u klauzuli WHERE, mogu se spojiti tabele RADNIK i ODELJENJE preko zajedničke kolone SIFRAO:

```
SELECT NAZIVO, PREZIME, SIFRARM,PLATA
FROM RADNIK, ODELJENJE
WHERE RADNIK.SIFRAO = ODELJENJE.SIFRAO
ORDER BY NAZIVO,PLATA DESC
```

Rezultat SQL upita je:

NAZIVO	PREZIME	SIFRA	PLATA
PRIPREMA	KLJAKIC	05	50000
PRIPREMA	CFBIC	03	24500
PRIPREMA	MILIC	01	13000
PRODAJA	BOBIC	03	28500
PRODAJA	ALAGIC	02	16000
PRODAJA	TUBIC	03	15000
PRODAJA	VUKIC	02	12500
PRODAJA	MARTIC	02	12500
PRODAJA	JAKIC	01	9500
RAZVOJ	SUSIC	04	30000
RAZVOJ	FILIPIC	04	30000
RAZVOJ	JOVIC	03	29750
RAZVOJ	ALIMPIC	01	11000
RAZVOJ	STEVIC	01	8000

Potpuno isti rezultat kao u prethodnom slučaju dobija se i prilikom definisanja spajanja korišćenjem opcije INNER JOIN definisane u MS ACCESS-u.

```
SELECT NAZIVO, PREZIME, SIFRARM, PLATA
FROM RADNIK
INNER JOIN ODELJENJE ON RADNIK.SIFRAO =
ODELJENJE.SIFRAO
```

Rezultat SQL upita je:

Naziv	Prezime:	Sifrar	Plata
PRIPREMA	CFBIC	03	2450
PRIPREMA	KLJAKIC	05	5000
PRIPREMA	MILIC	01	1300
RAZVOJ	STEVIC	01	8000
RAZVOJ	JOVIC	03	2975
RAZVOJ	SUSIC	04	3000
RAZVOJ	ALIMPIC	01	1100
RAZVOJ	FILIPIC	04	3000
PRODAJA	ALAGIC	02	1600
PRODAJA	VUKIC	02	1250
PRODAJA	MARTIC	02	1250
PRODAJA	BOBIC	03	2850
PRODAJA	TUBIC	02	1500
PRODAJA	JAKIC	01	9500

Korišćenje grupnih funkcuja u JOIN-u

Na primeru korišćenja grupne funkcije u JOIN-upitu biće izvršeno i grupisanje dveju kolona. U primeru će biti korišćene tri funkcije:

- SUM - sabira vrednosti polja definisanih u GROUP BY klauzuli;

- COUNT - broji redove (brojač) koji pripadaju svakoj od ovih grupa;
- AVG - nalazi prosečnu vrednost određenih polja u svakoj grupi.

Na primer, na kom je SIFRARM mestu i koliko je radnika radilo (COUNT) na svakom od poslova u svakom odeljenju, kao i kolike su suma (SUM) i prosečna plata (AVG) u ovako formiranim grupama?

```
SELECT NAZIVO,SIFRARM,SUM (PLATA),COUNT(*),AVG
 (PLATA)
 FROM RADNIK,ODELJENJE
 WHERE RADNIK.SIFRAO = ODELJENJE.SIFRAO
 GROUP BY NAZIVO,SIFRARM
```

Rezultat SQL upita je:

NAZIVO	SIFRAR	SUM	COUNT	AVG (PLATA)
PRIPREMA	01	13000	1	13000
PRIPREMA	03	24500	1	24500
PRIPREMA	05	50000	1	50000
PRODAJA	02	56000	4	14000
PRODAJA	03	28500	1	28500
RAZVOJ	01	19001	3	6333.67
RAZVOJ	03	29750	1	29750
RAZVOJ	04	60000	2	30000

U sledećem primeru biće prikazani poslovi koje obavljaju više ili najmanje dva radnika u svakom odeljenju korišćenjem HAVING klauzule.

```
SELECT NAZIVO,SIFRARM,SUM (PLATA)COUNT (*), AVG (PLATA)
 FROM RADNIK,ODELJENJE
 WHERE RADNIK.SIFRAO =ODELJENJE.SIFRAO
 GROUP BY NAZIVO,SIFRARM
 HAVING COUNT (*) >= 2
```

Rezultat SQL upita je:

NAZIVO	SIFRA	SUM	COUNT	AVG
PRODAJA	02	56000	4	14000
RAZVOJ	01	19001	3	6333.6
RAZVOJ	04	60000	2	30000

Spajanje tabele sa samom sobom (Self-Join)

Spajanje tabele sa samom sobom je primer upita nad rekurzivnim tabelama; na primer, prikaži: prezime i radno mesto svakog radnika koji ima pretpostavljenog kao i prezime i posao pretpostavljenog.

```
SELECT PODR.PREZIME,PODR.SIFRARM, PODR.RUKOV,
 NADR.SIFRAR AS SEF, NADR.PREZIME,
 NADR.SIFRARM
 FROM RADNIK PODR, RADNIK NADR
 WHERE PODR.RUKOV = NADR.SIFRAR
```

Da bi se realizovao ovaj upit, tabeli RADNIK se daju dva sinonima: PODR i NADR. Kada su potrebni podaci o radniku tabela RADNIK se referencira sinonimom PODR, a kada su potrebni podaci o neposrednim rukovodiocima - sinonimom NADR. Na taj način, pomoću navedenih sinonima, praktično, formiraju se dve virtuelne tabele sa identičnim sadržajem.

Rezultat SQL upita je:

PREZIME	SIFRAR	RUKOV	SIFRAO	PREZIM	SIFRA
SUSIC	04	827566	827566	JOVIC	03
FILIPIC	04	827566	827566	JOVIC	03
ALAGIC	02	827698	827698	BOBIC	03
VUKIC	02	827698	827698	BOBIC	03
MARTIC	02	827698	827698	BOBIC	03
TUBIC	02	827698	827698	BOBIC	03
JAKIC	01	827698	827698	BOBIC	03
MILIC	01	827782	827782	CEBIC	03
ALIMPIC	01	827788	827788	SUSIC	04
JOVIC	03	827839	827839	KLJAKIC	05
BOBIC	03	827839	827839	KLJAKIC	05
CEBIC	03	827839	827839	KLJAKIC	05
KLJAKIC	05	827839	827839	KLJAKIC	05
STEVIC	01	827902	827902	FILIPIC	04

Spoljno spajanje (Other Join)

Postoje dva tipa spoljnog spajanja između MS ACCESS tabela (slika 4.21, tipka *Join Type...)*.

Spoljnim spajanjem *Left Join* biraju se svi redovi iz prve tabele i samo oni redovi iz druge tabele čiji je sadržaj veznih polja jednak sa sadržajem u prvoj tabeli. Ovu konstataciju treba posmatrati kroz sledeći primer.

```
SELECT NAZIVO, PREZIME, SIFRARM, PLATA
FROM ODELJENJE LEFT JOIN RADNIK
ON RADNIK.SIFRAO = ODELJENJE.SIFRAO
```

Rezultat SQL upita je:

Nazivo	Prezime	Sifrarm:	Plata
PRIPREMA	CEBIC	03	24500
PRIPREMA	KLJAKIC	05	50000
PRIPREMA	MILIC	01	13000
RAZVOJ	STEVIC	01	8000
RAZVOJ	JOVIC	03	29750
RAZVOJ	SUSIC	04	30000
RAZVOJ	ALIMPIC	01	11000
RAZVOJ	FILIPIC	04	30000
PRODAJA	ALAGIC	02	16000
PRODAJA	VUKIC	02	12500
PRODAJA	MARTIC	02	12500
PRODAJA	BOBIC	03	28500
PRODAJA	TUBIC	02	15000
PRODAJA	JAKIC	01	9500
PROIZVODNJA			

Može se primetiti da odeljenje PROIZVODNJA nema radnike, ali da je i ono uzeto u razmatranje, jer je postavljen uslov da se uzimaju svi redovi prve tabele (ODELJENJE), bez obzira na to da li imaju sprezanje.

Spoljnim spajanjem *Right Join* biraju se svi redovi iz druge tabele i samo oni redovi iz prve tabele čiji je sadržaj veznih polja jednak sadržaju veznih polja druge tabele. Da bi se prikazao primer i za ovaj oblik spoljnog spajanja, samo treba promeniti redosled tabela, pa je sada tabela sa desne strane ODELJENJE.

```
SELECT NAZIVO, PREZIME, SIFRARM, PLATA
FROM RADNIK RIGHT JOIN ODELJENJE
ON RADNIK.SIFRAO = ODELJENJE.SIFRAO
```

Rezultat SQL upita je:

Nazivo	Prezime	Sifrarm	Plata:
PRIPREMA	CEBIC	03	24500
PRIPREMA	KLJAKIC	05	50000
PRIPREMA	MILIC	01	13000
RAZVOJ	STEVIC	01	8000
RAZVOJ	JOVIC	03	29750
RAZVOJ	SUSIC	04	30000
RAZVOJ	ALIMPIC	01	11000
RAZVOJ	FILIPIC	04	30000
PRODAJA	ALAGIC	02	16000
PRODAJA	VUKIC	02	12500
PRODAJA	MARTIC	02	12500
PRODAJA	BOBIC	03	28500
PRODAJA	TUBIC	02	15000
PRODAJA	JAKIC	01	9500
PROIZVODNJ			

Može se primetiti da odeljenje 40 PROIZVODNJA nema radnike, ali da je i ono uzeto u razmatranje, jer je postavljen uslov da se uzimaju svi redovi druge (ODELJENJE) tabele bez obzira na to da li imaju sprezanje.

Spajanje korišćenjem operatora UNION

Operator UNION nad dva SELECT BLOKA daje sve rezultujuće redove prvog select bloka i one rezultujuće redove drugog koji nisu među rezultujućim redovima prvog select bloka.

Na primer, u okviru tabele RADNIK , izdvojiti zajednička radna mesta za odeljenje 10 i 30.

```
SELECT SIFRARM
FROM RADNIK
WHERE Sifrao = '10'
UNION
SELECT SIFRARM
FROM RADNIK
WHERE Sifrao = '30'
```

Rezultat SQL upita je:

SIFRARM
01
02
03
05

Održavanje baze podataka

Operacije za održavanje su: INSERT, UPDATE, DELETE.

Ubacivanje redova u tabelu-INSERT

Naredba INSERT INTO koristi se za kreiranje upita koji dodaje podatke tabeli.

Naredba za dodavanje jednog sloga ima sledeću sintaksu:

```
INSERT INTO tabela [ (kolona,kolona,...),
VALUES (vrednost,vrednost,...)]
```

Za unos podataka naredba INSERT se može definisati u dve varijante:

- varijanta NE definisanjem kolone u koju se unose vrednosti preko VALUES

```
INSERT INTO ODELJENJE
VALUES ('50', 'NABAVKA', 'CACAK')
```

- varijanta SA definisanjem kolone u koju se unose vrednosti preko VALUES

```
INSERT INTO ODELJENJE
(sIFRAO, NAZIVO, MESTO)
VALUES ('50', 'NABAVKA', 'CACAK')
```

Druga varijanta omogućuje definisanje samo onih kolona za koje treba uneti podatke (NON NULL), ali redosled kolona mora da odgovara redosledu iza VALUES.

U INSERT naredbi se imenuje tabela (ODELJENJE), u koju se ubacuju red i lista vrednosti svih podataka.

Ako se želi dodati istovremeno više slogova, onda naredba INSERT INTO ima sledeću sintaksu:

```
INSERT INTO tabela [ (kolona,kolona,..),
 SELECT kolona, kolona... FROM tabela
```

Za definisani primer piše se:

```
INSERT INTO ODELJENJE ( SIFRAO, NAZIVO, MESTO )
SELECT '50', 'NABAVKA', 'CACAK' FROM RADNIK
```

Izmena vrednosti memorisanih u poljima-UPDATE

Naredba UPDATE koristi se za kreiranje upita sa kojim se može izvršiti promena vrednosti podataka u poljima tabele.

Sintaksa ove naredbe je:

```
UPDATE tabela [alias]
SET kolona [,kolona...]=[izraz, podupit]
[WHERE uslov]
```

Ako se uzme već poznata tabela RADNIK i doda svim radnicima sa zanimanjem 04 (PROJEKTANT) po 100 dinara povećanja zarade, to iziskuje izvršenje UPDATE naredbe, na sledeći način:

```
UPDATE RADNIK
SET PLATA = PLATA + 100
WHERE SIFRARM = '04'
```

Klauzula UPDATE imenuje tabelu koju treba menjati (UPDATE RADNIK). Klauzula SET izjednačava polje koje korisnik imenuje sa nekom vrednosti (SET PLATA = PLATA + 100). U klauzuli WHERE specificira se jedan red ili niz redova koje treba promeniti (WHERE SIFRARM = '04').

Posle izvedenih izmena, tabela RADNIK za radnike na SIFRARM mestu 04 sada izgleda:

```
SELECT PREZIME, SIFRARM, PLATA
FROM RADNIK
WHERE SIFRARM = '04'
```

Prezime	Sifrarm:	Plata:
SUSIC	04	30100
FILIPIC	04	30100

Dodavanje nove kolone postojećoj tabeli-ALTER TABLE

Naredba ALTER TABLE ADD znači dodavanje nove kolone postojećoj tabeli.

Sintaksa ove naredbe je:

```
ALTER TABLE naziv
ADD (tip kolone [ogranicenja,])
```

Na primer, da bi se proširila tabela RADNIK definisanjem određenih honorarnih poslova,

odnosno dodavanjem kolone SIF, prethodno treba kreirati tabelu HPOSAO pomoću komande CREATE TABLE.

```
CREATE TABLE HPOSAO (SIF TEXT (3),
NAZIVH TEXT (20),
CENAP FLOAT8);
```

Sledeći korak je da se pomoću INSERT naredbe napuni tabela HPOSAO.

```
INSERT INTO HPOSAO VALUES ('101','MASINE',96000);
INSERT INTO HPOSAO VALUES ('102','ALATI',82000);
INSERT INTO HPOSAO VALUES ('103','DELOVI',15000);
```

Sada se to može izlistati:

```
SELECT * FROM HPOSAO;
```

SIF	NAZIVH	CENAP
101	MASINE	96000
102	ALATI	82000
103	DELOVI	15000

Da bi se mogla dodati nova kolona SIF tabeli RADNIK, koristi se komanda ALTER TABLE:

```
ALTER TABLE RADNIK ADD SIF TEXT (3)
```

Ovom naredbom su definisani: tabela koju treba izmeniti (RADNIK), kolona koju treba dodati (SIF), tip podatka nove kolone (TEXT) i maksimalna dužina polja nove kolone TEXT (3).

Izlistana tabela RADNIK sa dodatom kolonom SIF izgleda ovako:

```
SELECT * FROM RADNIK;
```

SIF	Sifra	Prezime	Ime:	Si	JMBG:	rukov:	Datum	Plata	Stim	Si	P	V
	827369	STEVIC	ZORAN	0	141195217103	827902	28.3.98	8000	600	2	M	2
	827499	ALAGIC	MILAN	0	250396434561	827698	28.3.98	1600	600	3	M	
	827521	VUKIC	MILOS	0	113049705543	827698	28.3.98	1250	600	3	M	2
	827566	JOVIC	MIRA	0	113049705643	827839	28.4.90	2975	700	2	Z	1
	827654	MARTIC	ZORA	0	114049605544	827698	12.6.88	1250	1000	3	Z	1
	827698	BOBIC	IVAN	0	140598705545	827839	15.11.89	2850	800	3	M	2
	827782	CEBIC	GORAN	0	120397065456	827839	11.6.80	2450	500	1	M	2
	827788	SUSIC	ZORAN	0	130495467675	827566	23.10.91	3000	650	2	M	2
	827839	KLJAKIC	STEVA	0	131295212234	827839	6.12.87	5000	600	1	M	2
	827844	TUBIC	MIRA	0	131295434212	827698	8.3.78	1500	600	3	Z	1
	827876	ALIMPIC	PETAR	0	250397634356	827788	9.5.92	1100	780	2	M	
	827900	JAKIC	VLADA	0	121196545723	827698	28.3.90	9500	900	3	M	
	827902	FILIPIC	DRAGA	0	121097053422	827566	20.11.96	3000	1000	2	M	2
	827934	MILIC	DRAGA	0	070695645646	827782	28.3.93	1300	790	1	M	2

Kada je definisana nova kolona naredbom UPDATE treba pridružiti određene radnike koji imaju honorarni posao i to samo u odeljenju 20, za REFERENTA (02) sa honorarnim poslom čija je šifra 101.

```
UPDATE RADNIK
SET SIF = '101'
WHERE SIFRAO = '20'
OR SIFRARM = '02';
```

Nova tabela RADNIK pokazuje da li je postignut željeni rezultat.

```
SELECT * FROM RADNIK;
```

Sifrar	SIF	Prezime	Ime:	Si	JMBG:	rukov:	Datum	Plata	Stim	Si	P	V
827369	101	STEVIC	ZORAN	0	141195217103	827902	28.3.98	8000	600	2	M	2
827499	101	ALAGIC	MILAN	0	250396434561	827698	28.3.98	1600	600	3	M	
827521	101	VUKIC	MILOS	0	113049705543	827698	28.3.98	1250	600	3	M	2
827566	101	JOVIC	MIRA	0	113049705643	827839	28.4.90	2975	700	2	Z	1
827654	101	MARTIC	ZORA	0	114049605544	827698	12.6.88	1250	1000	3	Z	1
827698		BOBIC	IVAN	0	140598705545	827839	15.11.89	2850	800	3	M	2
827782		CEBIC	GORAN	0	120397065456	827839	11.6.80	2450	500	1	M	2
827788	101	SUSIC	ZORAN	0	130495467675	827566	23.10.91	3000	650	2	M	2
827839		KLJAKIC	STEVA	0	131295212234	827839	6.12.87	5000	600	1	M	2
827844	101	TUBIC	MIRA	0	131295434212	827698	8.3.78	1500	600	3	Z	1
827876	101	ALIMPIC	PETAR	0	250397634356	827788	9.5.92	1100	780	2	M	
827900		JAKIC	VLADA	0	121196545723	827698	28.3.90	9500	900	3	M	
827902	101	FILIPIC	DRAGA	0	121097053422	827566	20.11.96	3000	1000	2	M	2
827934		MILIC	DRAGA	0	070695645646	827782	28.3.93	1300	790	1	M	2

Ako treba pridružiti honorarni posao 102 svim radnicima koji nemaju nijedan posao (SIF IS NULL) onda se to piše ovako:

```
UPDATE RADNIK
SET SIF = '102'
WHERE SIF IS NULL;
```

Sa sledećim upitom utvrđuje se novo stanje tabele RADNIK.

```
SELECT * FROM RADNIK;
```

Sifra	SIF	Prezime	Ime:	Si	JMBG:	rukov:	Datum	Plata	Stim	Si	P	V
827369	101	STEVIC	ZORAN	0	141195217103	827902	28.3.98	8000	600	2	M	2
827499	101	ALAGIC	MILAN	0	250396434561	827698	28.3.98	1600	600	3	M	
827521	101	VUKIC	MILOS	0	113049705543	827698	28.3.98	1250	600	3	M	2
827566	101	JOVIC	MIRA	0	113049705643	827839	28.4.90	2975	700	2	Z	1
827654	101	MARTIC	ZORA	0	114049605544	827698	12.6.88	1250	1000	3	Z	1
827698	102	BOBIC	IVAN	0	140598705545	827839	15.11.89	2850	800	3	M	2
827782	102	CEBIC	GORAN	0	120397065456	827839	11.6.80	2450	500	1	M	2
827788	101	SUSIC	ZORAN	0	130495467675	827566	23.10.91	3000	650	2	M	2
827839	102	KLJAKIC	STEVA	0	131295212234	827839	6.12.87	5000	600	1	M	2
827844	101	TUBIC	MIRA	0	131295434212	827698	8.3.78	1500	600	3	Z	1
827876	101	ALIMPIC	PETAR	0	250397634356	827788	9.5.92	1100	780	2	M	
827900	102	JAKIC	VLADA	0	121196545723	827698	28.3.90	9500	900	3	M	
827902	101	FILIPIC	DRAGA	0	121097053422	827566	20.11.96	3000	1000	2	M	2
827934	102	MILIC	DRAGA	0	070695645646	827782	28.3.93	1300	790	1	M	2

Ova promena vezana za nova polja u tabeli RADNIK dozvoljava uspostavljanje relacija između tabela: RADNIK i HPOSAO.

```
SELECT PREZIME,SIFRARM,SIFRAO,NAZIVH
FROM RADNIK, HPOSAO
WHERE RADNIK.SIF = HPOSAO.SIF;
```

Prezime	Sifrar	Sifra	NAZIV
FILIPIC	04	20	MASIN
ALIMPIC	01	20	MASIN
TUBIC	02	30	MASIN
SUSIC	04	20	MASIN
MARTIC	02	30	MASIN
JOVIC	03	20	MASIN
VUKIC	02	30	MASIN
ALAGIC	02	30	MASIN
STEVIC	01	20	MASIN
MILIC	01	10	ALATI
JAKIC	01	30	ALATI
KLJAKIC	05	10	ALATI
CEBIC	03	10	ALATI
BOBIC	03	30	ALATI

Brisanje redova iz tabele-DELETE

Naredbom DELETE brišu se redovi iz jedne ili više tabela preko liste u okviru FROM dela rečenice, ali koji zadovoljavaju uslov definisan pod WHERE delom rečenice.

Sintaksa ove naredbe je:

```
DELETE FROM tabela
[WHERE uslov]
```

Kako u odeljenju 40 (tabela ODELJENJE) nema zaposlenih (tabela RADNIK) može se brisati na sledeći način:

```
DELETE FROM ODELJENJE
WHERE SIFRAO = '40'
```

Treba zatim proveriti da li je izbrisano odeljenje 40 iz tabele ODELJENJE

```
SELECT *
FROM ODELJENJE
```

Sifrao	Nazivo	Mesto
10	PRIPREMA	PANCEVO
20	RAZVOJ	NOVI_SAD
30	PRODAJA	BEOGRAD

DELETE FROM klauzula imenuje tabelu iz koje treba izbrisati jedan ili više redova (na primer, tabela ODELJENJE). WHERE klauzula nije obavezna; ako je nema, onda se brišu svi redovi iz tabele.

Ako postoji WHERE klauzula onda se redovi brišu pod nekim datim uslovom. U ovom primeru korišćenjem WHERE klauzule brišu se iz tabele ODELJENJE oni redovi čija je vrednost SIFRAO = 40.

Brisanje tabela-DROP TABLE

Ukoliko se želi izbrisati cela tabela, koristi se naredba DROP TABLE.

Sintaksa ove naredbe je:

```
DROP TABLE naziv tabele
```

Ako se želi izbrisati tabela HPOSAO, piše se:

```
DROP TABLE HPOSAO
```

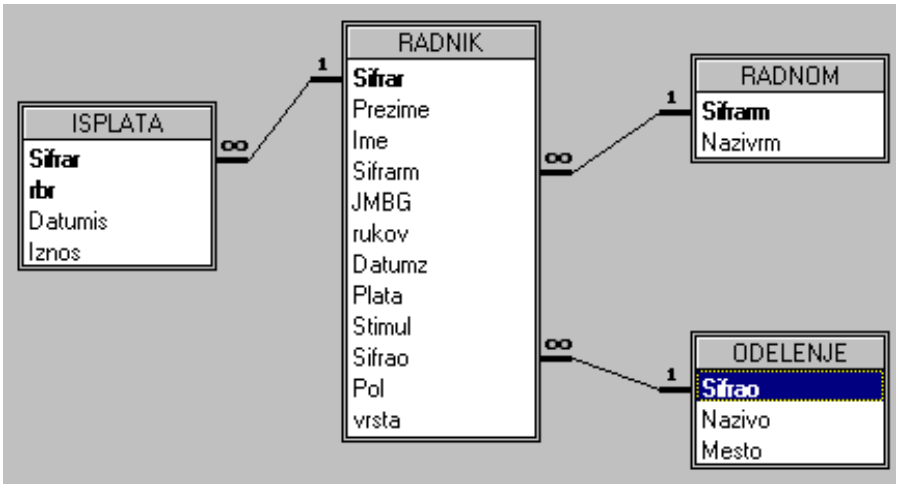
Na taj način briše se definicija tabele iz baze podataka zajedno sa podacima koje tabela sadrži; naredbu DROP može izvesti samo onaj koji je kreirao tabelu ili ima DBA privilegiju.

Na osnovu prethodno izvedenih aktivnosti (slika 4.31) u sledećem koraku potrebno je definisati izveštaj.

Aktivnost
3.3.4. Definisanje izveštaja

Kreiranje izveštaja izvodi se korišćenjem već definisanih i standardizovanih formi sa nadgradnjom.

Za primer izveštaja knjigovodstvu definisan je sledeći QBE upit:



Slika 4.34. Pogled na model baze podataka vezan za definisanje izveštaja

Definisani upit generiše SQL upit koji ima sledeći izgled:

```
SELECT DISTINCTROW SIFRARM.Nazivrm, RADNIK.Sifrar, RADNIK.Prezime,
RADNIK.Datumz, RADNIK.Plata, RADNIK.Stimul, [PLATA]+[STIMUL] AS uk,
ODELJENJE.Nazivo, RADNIK.Ime
FROM SIFRARM INNER JOIN ( (RADNIK INNER JOIN ODELJENJE ON RADNIK.Sifrao =
ODELJENJE.Sifrao) LEFT JOIN ISPLATA ON RADNIK.Sifrar = ISPLATA.Sifrar) ON
SIFRARM.Sifrarm = RADNIK.Sifrarm
GROUP BY SIFRARM.Nazivrm, RADNIK.Sifrar, RADNIK.Prezime, RADNIK.Datumz,
RADNIK.Plata, RADNIK.Stimul, [PLATA]+[STIMUL], ODELJENJE.Nazivo, RADNIK.Ime
```

Izvršenjem ovog upita dobija se sledeći rezultat:

NAZIVRM:	SIFRA	PREZIM	DATUM	PLAT	STIM	UK	NAZIV	IME:
DIREKTOR	827566	JOVIC	28.4.90	2975	700	30450	RAZVOJ	MIRA
DIREKTOR	827698	BOBIC	15.11.89	2850	800	29300	PRODAJA	IVAN
DIREKTOR	827782	CEBIC	11.6.80	2450	500	25000	PRIPREMA	GORAN
GEN_DIR	827839	KLJAKIC	6.12.87	5000	600	50600	PRIPREMA	STEVA
PROJEKTANT	827369	STEVIC	28.3.98	8000	700	8700	RAZVOJ	ZORAN
PROJEKTANT	827876	ALIMPIC	9.5.92	1100	780	11780	RAZVOJ	PETAR
PROJEKTANT	827900	JAKIC	28.3.90	9500	900	10400	PRODAJA	VLADA
PROJEKTANT	827934	MILIC	28.3.93	1300	790	13790	PRIPREMA	DRAGA
REFERENT	827499	ALAGIC	28.3.98	1600	600	16600	PRODAJA	MILAN
REFERENT	827521	VUKIC	28.3.98	1250	600	13100	PRODAJA	MILOS
REFERENT	827654	MARTIC	12.6.88	1250	1000	13500	PRODAJA	ZORA
REFERENT	827844	TUBIC	8.3.78	1500	600	15600	PRODAJA	MIRA
TEHNOLOG	827788	SUSIC	23.10.91	3000	650	30650	RAZVOJ	ZORAN
TEHNOLOG	827902	FILIPIC	20.11.96	3000	1000	31000	RAZVOJ	DRAGA

Korišćenjem objekta Report definiše se korisnički izveštaj koji može imati sledeći izgled:

Izvestaj knjigovodstvu

15-Jun-98

ODELJENJE	SIFRA RAD	PREZIME	DAT. ZAP	PLATA	STIM	UKUPNO
PRIPREMA						
DIREKTOR	**827782**	**CEBIC GORAN**	**11.6.80**	**24500**	**500**	25000
GEN_DIR	**827839**	**KLJAKIC STEVAN**	**6.12.87**	**50000**	**600**	50600
PROJEKTANT	**827934**	**MILIC DRAGAN**	**28.3.93**	**13000**	**790**	13790
		PRIPREMA UKUPNO		**87500**	**1890**	**89390**
PRODAJA						
DIREKTOR	**827698**	**BOBIC IVAN**	**15.11.89**	**28500**	**800**	29300
PROJEKTANT	**827900**	**JAKIC VLADAN**	**28.3.90**	**9500**	**900**	10400
REFERENT	**827499**	**ALAGIC MILAN**	**28.3.98**	**16000**	**600**	16600
		PRODAJA UKUPNO		**54000**	**2300**	**56300**
RAZVOJ						
DIREKTOR	**827566**	**JOVIC MIRA**	**28.4.90**	**29750**	**700**	30450
PROJEKTANT	**827369**	**STARCEV.ZORAN**	**28.3.98**	**8000**	**700**	8700
TEHNOLOG	**827788**	**SUSIC ZORAN**	**23.10.91**	**30000**	**650**	30650
			RAZVOJ UKUPNO	**67750**	**2050**	**69800**
			Grand Total:	**209250**	**8640**	**215490**

Slika 4.35. Izveštaj knjigovodstvu

Na ovaj način izvršeno je kompletiranje aktivnosti *"3. Aplikativno modeliranje"*, a u sledećem koraku je potrebno izvršiti implementaciju.

Aktivnost 4. Implementacija

Aktivnost 4.1. Uvođenje

Aktivnost 4.2. Testiranje

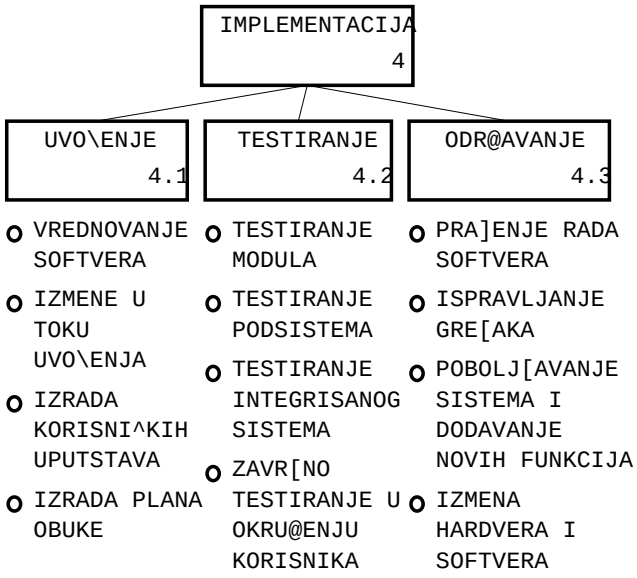
Aktivnost 4.3. Održavanje

Nakon izvedene aktivnosti *"3. Aplikativno modeliranje"* definisan je tok podataka "Radni IS" koji predstavlja ulaz u aktivnost *"4. Implementacija"* (slika 5.1).

Konsultant na ovom nivou treba da pomogne u uvođenju, testiranju i održavanju budućeg korisničkog softvera.

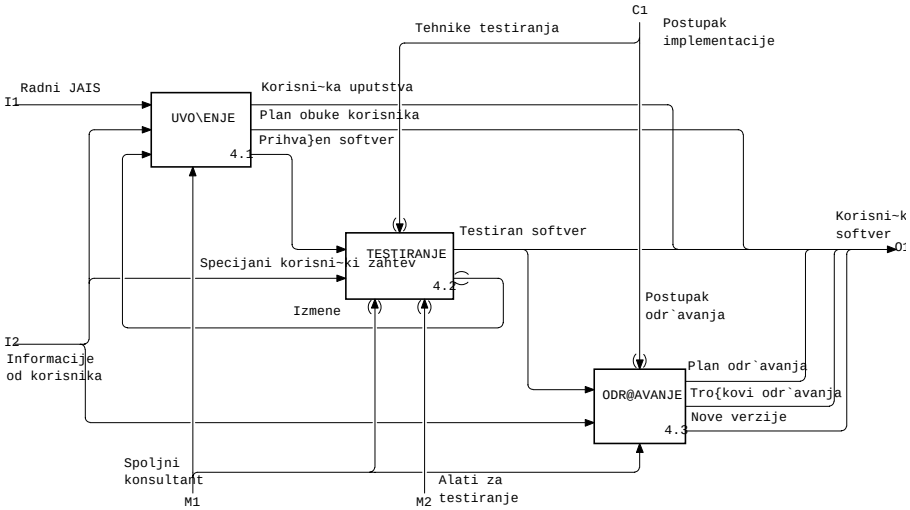
Na slici 5.1. prikazano je stablo aktivnosti za aktivnost *"4. Implementacija"*, koju čine sledeće aktivnosti:

- Aktivnost 4.1. Uvođenje,
- Aktivnost 4.2. Testiranje,
- Aktivnost 4.3. Održavanje.



Slika 5.1. Stablo aktivnosti za aktivnost "4. Implementacija"

Na slici 5.2. prikazan je dekompozicioni dijagram za aktivnost *"4. Implementacija"*, gde se definišu međusobne veze između aktivnosti datih na slici 5.1.



Slika 5.2. Dekompozicioni dijagram za aktivnost "4. Implementacija"

Aktivnost "*4. Implementacija*" izvodi se:

- na osnovu ulaznih informacija, definisanih kroz Radni JAIS-a (I1) prethodno urađen u aktivnosti "*3. Aplikativno modeliranje*" i Informacija od korisnika (I2);
- pod kontrolom "Postupak implementacije (C1);
- resursima vezanim za korišćenje alata za testiranje (M2) i uslugama spoljnog konsultanta (M1);
- kroz izlaz kao "Korisnički softver" (O1).

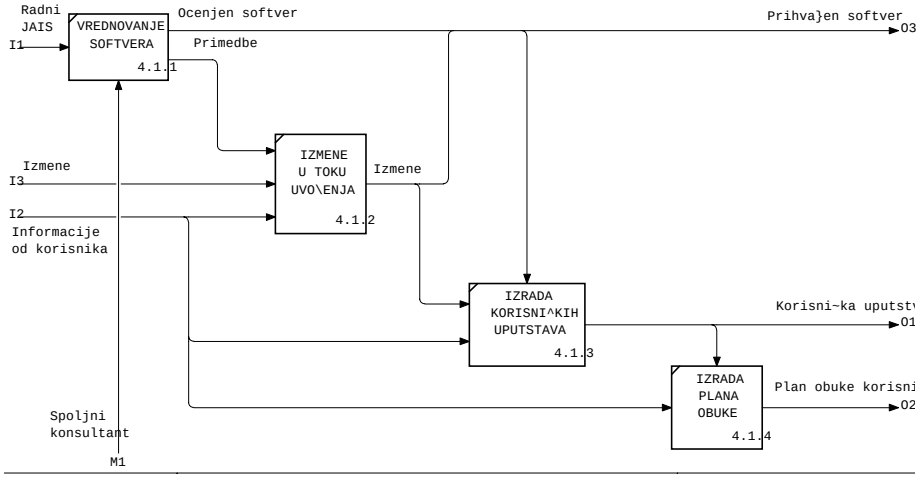
Mora se naglasiti da implementacija predstavlja priliku za sprovođenje principa reinženjeringa poslovnih procesa. Međusobne veze između podaktivnosti prikazane na slici 5.2. biće opisane u daljem tekstu.

Aktivnost
4.1. Uvođenje

Aktivnost *"4.1. Uvođenje"* izvodi se posredstvom sledećih aktivnosti (slika 3.1):

- Aktivnost 4.1.1. Vrednovanje softvera,
- Aktivnost 4.1.2. Izmene u toku uvođenja,
- Aktivnost 4.1.3. Izrada korisničkih uputstava,
- Aktivnost 4.1.4. Izrada plana obuke.

Na slici 5.3. prikazan je dekompozicioni dijagram za aktivnost *"4.1. Uvođenje"*, gde je definisan tok informacija počev od ulaznog dokumenta "Radni JAIS" preko nabrojanih aktivnosti (prikazanih na slici 5.3), do izlaza definisanog dokumentom Prihvaćen softver.



Slika 5.3. Dekompozicioni dijagram za aktivnost "4.1. Uvođenje"

Osnovni izlazi vezani za ovu aktivnost su:

- izmene u toku uvođenja,
- izrada korisničkih uputstava,
- plan obuke korisnika i dr.

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 5.3.

Aktivnost
4.1.1. Vrednovanje softvera

Vrednovanje softvera je podskup aktivnosti softverskog inženjerstva i može se smatrati sistemom, odnosno sistemom vrednovanja.

Da bi se izvršilo vrednovanje, treba *specificirati zahteve,* tj. definisati osnovne zahteve za funkcije i performanse i dati potrebna ograničenja i dr. Kao poseban element izdvajaju se *zahtevi za kvalitetom,* definisani kvantitativnim i kvalitativnim formulisanim zahtevima. To uslovljava i formulisanje *faktora kvaliteta* koji je odlika ili karakteristika određenog elementa. Kvantitativna mera se izražava preko tzv. *metrike kvaliteta*. Rezultat vrednovanja je *ocenjivanje* kao aktivnost primene određenog dokumentovanog kriterijuma ocenjivanja softverskog modula, paketa ili softverskog proizvoda radi određivanja prihvatljivosti ili puštanja u eksploataciju softverskog paketa ili proizvoda.

Osnovu procesa vrednovanja čine *zahtevi vrednovanja* koji se iskazuju preko:

- funkcionalnih zahteva, definisanih listom potrebnih funkcija određenih preko odgovarajućih težina;
- sadržaja korišćenja softverskog proizvoda;
- broja zadataka koje proizvod podržava;
- broja korisnika;
- profila korisnika (nivo eksperta, iskustvo, obučenost);
- dodeljivanja težina svakoj karakteristici kvaliteta.

U okviru aktivnosti "4.1.1. Vrednovanje" poštuju se principi definisani serijom standarda ISO/IEC 9126 Informacione tehnologije (Vrednovanje softvera - Karakteristike kvaliteta i smernice za njihovu upotrebu) i ISO 9000, vezanih za obezbeđenje kvaliteta softvera (Software Quality Assurance - SQA) i mogu se posmatrati kroz sledeće složene procese:

- analiza zahteva vrednovanja,
- specifikacija vrednovanja,
- projektovanje vrednovanja,
- primena vrednovanja i
- izveštavanje o vrednovanju.

U daljem tekstu detaljno će biti obrazloženi gore definisani složeni procesi.

Analiza zahteva vrednovanja

Složen proces *Analiza zahteva vrednovanja* u sebi sadrži sledeće procese:

- razvoj i utvrđivanje načina akvizicije,
- definisanje funkcionalnih zahteva,
- definisanje sadržaja korišćenja i
- definisanje zahteva sistema kvaliteta.

Razvoj i utvrđivanje načina akvizicije su veoma važan proces u kome se pronalazi i odabira prikladan način da bi se došlo do odgovora na postavljene zahteve. Primeri akvizicije mogu biti intervjui, upitnici, ankete... Može se napraviti i programski modul koji će podržati akviziciju.

Definisanje funkcionalnih zahteva se odnosi na identifikaciju korisnikovih funkcionalnih potreba, što uključuje izradu liste potrebnih funkcija i ukazivanje na njihovu respektivnu važnost. To se može uraditi dodeljivanjem težina ili klasa funkcijama.

Definisanje sadržaja korišćenja obuhvata: identifikaciju izvršenog zadatka (prirodu, proizvod dodeljen jednom zadatku, moguće korišćenje za nekoliko zadataka itd.), okruženje (trgovina, industrija, prosveta, klasa rizičnih proizvoda itd.), korisnike (jedan ili više, odeljenje, organizacija...), profil korisnika (nivo eksperta, iskustvo, obučenost itd.).

Definisanje sadržaja korišćenja se koristi za definisanje zahteva kvaliteta primenjivih na proizvod, ali i za vrstu vrednovanja koja će se koristiti tokom procesa izbora. To može dovesti i do modifikacije funkcionalnih zahteva proizvoda.

Definisanje zahteva kvaliteta primenjuje se na proizvod koji, uzimajući u obzir ISO/IEC 9126 karakteristike, treba da poseduje:

- funkcionalnost, tj. da izlazi budu korektni i tačni (treba da radi kako je predviđeno);
- postojanost, tj. ne sme da pada na greškama koje korisnik napravi;
- integritet (vezano za integritet baze podataka i referencijalni integritet);
- pouzdanost , tj. mora da radi svaki put isto prilikom korišćenja;
- dokumentovanost (vezano za korektno korišćenje CASE alata);
- prilagodljivost, koja treba da omogući da sami korisnici izvode izmene;
- preglednost, tj. ne treba da bude pretrpan informacijama;
- razumljivost (da omogući korisniku da zna šta treba da radi);
- validnost (vezano za zadovoljenje kriterijuma klijenta);
- pogodnost za održavanje;
- fleksibilnost, tj. lake izmene bez ulaganja velikih napora;
- prenosivost, tj. mogućnost rada sa druge platforme;
- kompatibilnost sa drugim programima (ODBC drajveri);
- efikasnost, tj. balansiranje između efikasnosti iskorišćenja resursa računara i čovekovih resursa;
- modularnost (zbog lake dogradnje i otklanjanja bagova);
- upotrebljivost, tj. mogućnost korišćenja delova programa u drugim aplikacijama.

Specifikacija vrednovanja

Specifikacija vrednovanja, kao složeni proces, u sebi sadrži sledeće procese:

- izbor metrike i indikatora,
- izbor karakteristika,
- definisanje nivoa ranga,
- definisanje kriterijuma ocenjivanja.

Izbor metrike se definiše za svaki zahtev. Metrika je, uopšteno posmatrano, numerička vrednost na skali. Može se koristiti ceo ili realan broj. Primeri za skale metrika su:

- 1...10 prirodni brojevi;
- 0...1 realni;
- procenat;
- "odličan, "dobar", "solidan", "loš";

- "da", "ne".

Definicija metrike treba da uključi i indikaciju o tome kako treba izvršiti merenje.

Izbor karakteristika uključuje identifikaciju specifičnog značenja karakteristika koje se primenjuju na proizvod u kontekstu na identifikovanog korisnika. Ako uzete karakteristike nisu dovoljne za precizan opis zahteva kvaliteta, mogu se koristiti i potkarakteristike. Rangiranje zahteva kvaliteta može se uraditi dodeljivanjem težine, ili klase.

Definisanje nivoa ranga, prema ISO/IEC 9126, određuje se u okviru četiri nivoa ranga koji odgovaraju mogućim vrednostima metrike. Ova aktivnost se odnosi na ucrtavanje ranga na metričku skalu.

Definisanje kriterijuma ocenjivanja se sastoji iz određivanja šta je prihvatljivo za razmatranje:

- rangiranje (koje je prethodno definisano),
- kontekst korišćenja proizvoda,
- dodatne mogućnosti, kao što su trošak, kašnjenje itd.

Projektovanje vrednovanja

Projektovanje vrednovanja, kao složeni proces, sastoji se iz povezivanja tehnika vrednovanja za karakteristike i nivoe softvera, kao i povezivanja modula vrednovanja za tehnike vrednovanja. Ovaj složeni proces kao izlaz daje plan vrednovanja.

Projektovanje vrednovanja čine procesi:

- specifikacija modula vrednovanja,
- ugradnja modula vrednovanja u biblioteku.

Specifikacija modula vrednovanja obuhvata sažimanje jedne ili više tehnika vrednovanja. Aktivnost obuhvata specificiranje zahtevanih informacija o: proizvodu i procesu, tehnikama vrednovanja, izlazu koji je rezultat primene tehnike, kao i procenjenom trošku primene modula vrednovanja. Jasno je da su tehnike vrednovanja i moduli vrednovanja u vezi sa karakteristikama softvera i nivoima vrednovanja. Posle određivanja tehnika vrednovanja, tj. identifikovanja koje karakteristike softvera treba razmatrati i koliko detaljno, uz tehnička ograničenja softvera, bira se i skup primenjivih modula vrednovanja iz biblioteke modula vrednovanja.

Ugradnja modula vrednovanja u biblioteku odnosi se na donošenje odluke o razvijanju novog modula i njegovo dodavanje u biblioteku.

Primena vrednovanja

Složeni proces *Primena vrednovanja* daje rezultate vrednovanja dobijenih kao izlazi iz procesa:

- merenje pomoću metrika,
- ocenjivanje poređenjem,
- izveštavanje merenja ocenjivanja na osnovu plana merenja definisanog u prethodnom procesu.

Izveštavanje o vrednovanju

Složeni proces - *Izveštavanje o vrednovanju* čine procesi:

- analiza rezultata vrednovanja,
- generisanje izveštaja.

Na osnovu rezultata dobijenih kao izlaz iz procesa *Primena vrednovanja* i analize tih rezultata stvaraju se razne vrste izveštaja koji predstavljaju izlaze iz kompletnog sistema vrednovanja.

Aktivnost
4.1.2. Izmene u toku uvođenja

Na osnovu definisanih primedbi izvode se izmene koje, ako se koriste CASE alati, ostaju kao trag aktuelnoj elektronskoj dokumentaciji o sprovedenim izmenama. Ako se izmene naprave u tabelama baze podataka, automatski se sprovode iste izmene i u modelu podataka.

Aktivnost
4.1.3. Izrada korisničkih uputstava

Korisnička uputstva mogu biti opšta uputstva za rad sa aplikacijom, kao i detaljna korisnička uputstva za svaki programski sistem. Pored papira, treba da imaju i dimenziju On-line dokumentacije. Dokumentacija mora da ima sledeće karakteristike:

- pri pisanju neophodni su jasni i koncizni izrazi;
- oslovljavanje korisnika treba da bude u drugom licu, uz korišćenje aktivnih glagola;
- pri opisu procedure treba upotrebljavati jednostavne glagole;
- procedure se moraju opisivati logičkim redom;
- ne treba upotrebljavati izraze iz žargona;
- treba izbegavati šale;
- dati mogućnost jednostavnog izbora i dr.

Aktivnost
4.1.4. Izrada plana obuke

Pretpostavka za izvođenje aktivnosti *"4.1.4. Izrada plana obuke"* je da su budući korisnici kompjuterski opismenjeni, kao što je to opisano u okviru aktivnosti "*1.3.2. Kadrovske potrebe*". Za ovu aktivnost se napravi plan obuke po prioritetima uvođenja pojedinih modula ili podsistema.

Na osnovu prethodno izvedenih aktivnosti (slika 5.2) u sledećem koraku treba izvesti testiranje.

Aktivnost
4.2. Testiranje

Procesom *"4.2.Testiranje"* testira se softver za konkretno korisničko okruženje, uz početni unos podataka, što je naročito bitno za testiranje softvera koji radi u mreži i za koji treba da se istestiraju elementi vezani za transakcionu obradu i odgovarajuća zaključavanja.

Testiranjem se ocenjuje valjanost programa, tj. testiraju se performanse programa i vrše korekcije programa da bi se njegove performanse prilagodile korisniku. Dakle, testiranje podrazumeva ocenjivanje odlika programa i njegovo revidiranje da bi se dostigli postavljeni ciljevi. Od iskustva programera i korisnika zavisi da li će se program pravilno oceniti.

Statistike govore da se kod ozbiljnih softverskih projekata oko 50% ukupnog vremena i napora troši na testiranje.

Od nivoa uključenosti budućeg korisnika prilikom izrade softverskog proizvoda zavisi i stepen njegovog uključenja u testiranje. Obično se definišu dva koncepta provere: validacija i verifikacija.

Validacija proverava da li proizvod zadovoljava spoljne kriterijume klijenta. *Verifikacija* proverava da li je proizvod napravljen kako treba.

Na slici 5.2. definisani su za aktivnost "4.2. Testiranje" resursi u obliku alata za testiranje i kao kontrola tehnike testiranja.

Alati za testiranje treba da omoguće planiranje, praćenje i realizaciju testiranja i da automatizuju pojedine faze testiranja.

Tehnike testiranja su podeljene na:

- tehniku crne kutije ili funkcionalnu tehniku;
- tehniku bele kutije ili strukturno testiranje sofvera.

Tehnika crne kutije omogućuje testiranje funkcionalne specifikacije programa, ne vodeći računa o unutrašnjoj tehnici i strukturi programa. Tehnika crne kutije je vezana za izbor test-primera i scenarija za što širi dijapazon ulaznih podataka.

Tehnika bele kutije svodi se na automatsko proveravanje programskih struktura, tokova podataka, logičke međuzavisnosti procedura i njihovih ulaza/izlaza i dr.

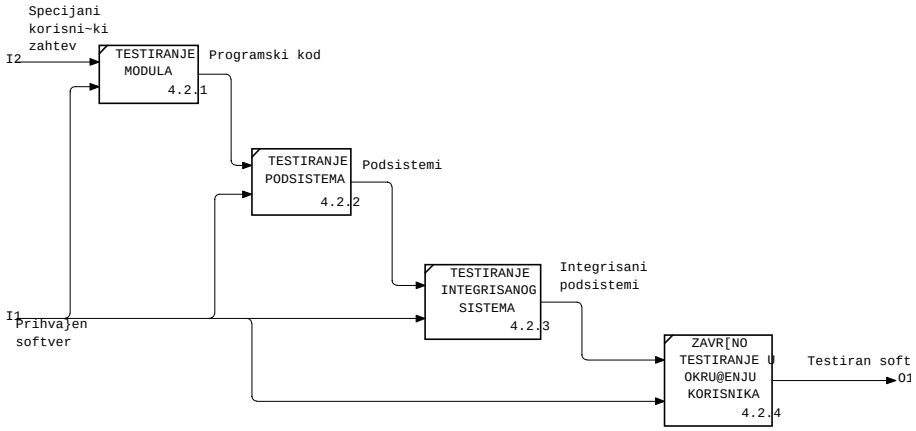
U principu, definišu se četiri strategije testiranja softvera, i to:

- demonstracija, gde se utvrđuje da li softver radi u skladu sa specifikacijom;
- destrukcija, kada se namerno ruši program, i to, obično, suprotno zahtevima;
- evaulacija, gde se testiranje izvodi u ranim fazama razvoja softvera;
- prevencija, gde se testiranje izvodi u ranim fazama razvoja softvera korišćenjem raznih alata za generisanje formalne specifikacije.

Aktivnost *"4.2. Testiranje"*, kao što je prikazano na slici 5.4, čine sledeće aktivnosti:

- Aktivnost 4.2.1. Testiranje modula,
- Aktivnost 4.2.2. Testiranje podsistema,
- Aktivnost 4.2.3. Testiranje integrisanog sistema,

- Aktivnost 4.2.4. Završno testiranje u okruženju korisnika.



Slika 5.4. Dekompozicioni dijagram za aktivnost "4.2. Testiranje"

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 5.4.

Aktivnost
4.2.1. Testiranje modula

Kako su moduli povezani sa nizom nezavisnih komponenti, to se zahteva provera svih specifikacija i konstrukcija vezanih za modul.

Aktivnost
4.2.2.Testiranje podsistema

Prilikom testiranja podsistema najčešće se pojavljuju problemi, jer su mnogi softverski inženjeri uključeni u izgradnju podsistema. Različite interpretacije specifikacija, obično, rezultiraju problemima, pa ih treba uklopiti. Problematika na ovom nivou se pojednostavljuje ako su korišćeni CASE alati gde je definisan jedinstven rečnik podataka i procesa i gde su veze prikazane na grafički način.

Osnovni koraci vezani za ovu aktivnost su:

- spajanje modula,
- definisanje internih interfejsa,
- testiranje rada grupe modula,
- testiranje veze sa eksternim interfejsima.

Aktivnost
4.2.3. Testiranje integrisanog sistema

Testiranje integrisanog sistema zahteva od budućeg korisnika da pripremi realne podatke za upotrebu. Pri tom se testiraju: funkcionalnost, performanse, restart, oporavak i funkcionisanje.

Aktivnost
4.2.4. Završno testiranje u okruženju korisnika

Završno testiranje u okruženju korisnika se, obično, zove alfa-testiranje ako je u pitanju jedan korisnik, a ako se proizvod distribuira na mnogo mesta ili mnogo pojedinaca tada je beta-testiranje. Test preuzimanja izvodi se u dva koraka:

- definisanjem zahteva i specifikacija i
- ispunjenjem uslova preuzimanja.

Na osnovu prethodno izvedenih aktivnosti (slika 5.2) u sledećem koraku biće izvedena aktivnost *"4.3. Održavanje".*

Aktivnost
4.3. Održavanje

Na nivou održavanja drastično se pokazuju sve manjkavosti vezane za loše urađene prethodno opisane faze. U zavisnosti od korektnosti rada u prethodnim fazama, programer troši u proseku od 20% do 80% svog radnog vremena za održavanje.

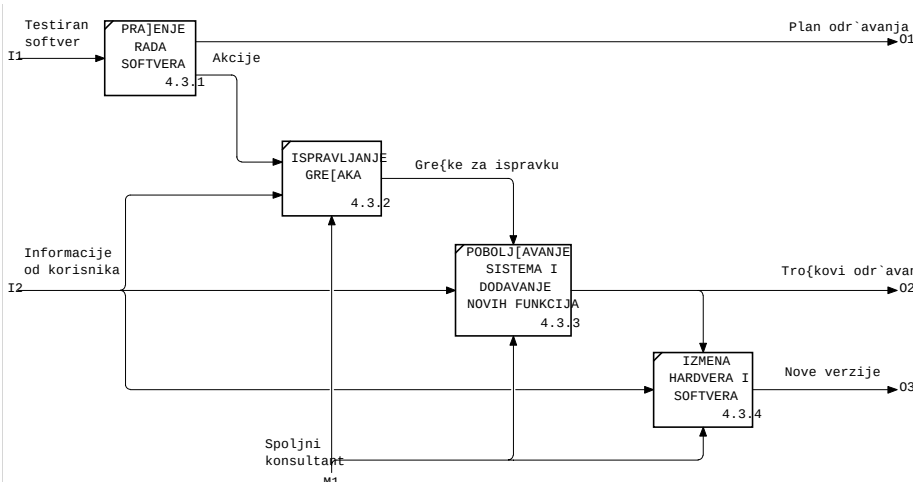
Osnovni problemi u održavanju su:

- nepridržavanje standarda,
- loša dokumentacija,
- nedostatak kadra,
- nepostojanje testova,
- nekorektno održavanje,
- odsustvo povratne veze,
- nepoznavanje troškova održavanja.

Aktivnost "4.3. Održavanje", kao što je prikazano na slici 5.5, čine sledeće aktivnosti:

- Aktivnost 4.3.1. Praćenje rada,
- Aktivnost 4.3.2. Ispravljanje grešaka,
- Aktivnost 4.3.3. Poboljšanje sistema i dodavanje novih funkcija,
- Aktivnost 4.3.4. Izmena hardvera i softvera.

U daljem tekstu detaljno će biti obrazložene aktivnosti prikazane na slici 5.5.



Slika 5.5. Dekompozicioni dijagram za aktivnost "4.3. Održavanje"

Aktivnost
4.3.1. Praćenje rada

Aktivnost *"4.3.1. Praćenje rada"* treba izvoditi kontinualno, sve dok ne bude potrebno da se izvede zamena, jer informacije stečene u toku tog praćenja omogućuju da se odredi vrsta promena, tj. na taj način se izvodi tzv. kontinualno usavršavanje. Mogu se na osnovnom nivou pratiti:

- sati u upotrebi,
- urađeni poslovi i transakcije,
- informacije o vremenu,
- učitavanja,
- registrovanje nedostataka i dr.

Ovakvo praćenje omogućava oporavak sistema od katastrofalnih grešaka. Stoga je ova aktivnost neposredno vezana i za sprovođenje korektivnih akcija. Tako, ako se prati vreme i primete varijacije u vremenu izvođenja transakcija, automatski se izvode korektivne akcije.

Aktivnost
4.3.2. Ispravljanje grešaka

U aktivnosti *"4.3.2. Ispravljanje grešaka"*, pored ispravljanja grešaka, prati se i odstupanje softvera, i vrši prilagođavanje korisniku i zadatku.

Aktivnost
4.3.3. Poboljšanje sistema i dodavanje novih funkcija

Aktivnost *"4.3.3. Poboljšanje sistema i dodavanje novih funkcija"* skoro uvek nastupa, jer korisnik ubrzo spozna nove mogućnosti sistema, pa ima nove zahteve vezane za poboljšanje sistema. Dodavanje novih funkcija ne mora da bude teško, ako su ispoštovani prethodno opisani koraci.

Aktivnost
4.3.4. Izmena hardvera i softvera

Aktivnost *"4.3.4. Izmena hardvera i softvera"* posmatra se kroz softverske izmene vezane za promenu SUBP, kao i hardverske izmene vezane za izgradnju mreže u okviru klijent/server-arhitekture.

Softverske izmene

Baza podataka i aplikacije, koje se na njoj temelje, podvrgnute su čestim promenama. Promene mogu biti izazvane razvojem opreme (npr., na tržištu se javlja efikasniji tip: memorije, terminala, procesora).

Češće promene su potrebne zbog razvoja korisničkog sistema (aplikacija), promena uslova i pravila poslovanja, novih zakonskih zahteva itd. Korisniku SUBP mora obezbediti što veću fleksibilnost, tako da funkcioniše kao nekakav štit između aplikacije i baze podataka.

SUBP poznaje fizičku i logičku strukturu baze podataka na jednoj strani i zahteve korisnikovih programa na drugoj strani, te deluje kao posrednik (interface), koji korisnikovom programu osigurava uvek isti pogled na podatke, bez obzira na eventualne promene u fizičkoj ili logičkoj strukturi baze podataka (data indepedence). Time se, sa jedne strane, postiže imunost korisnikovih programa na promene, koje doživljava baza podataka, a, sa druge, znatno se olakšava i pojednostavljuje održavanje korisnikovih programa.

Pojavom Interneta i otvorenih sistema omogućen je razvoj softvera, nezavisno od hardverske platforme, što dalje omogućuje razvoj na personalnim računarima i implementaciju na nekom drugom hardveru. Slične mogućnosti su i u vezi sa softverom. Ako je u okviru klijent/server-aplikacije razvijen ceo sistem na strani servera i klijenta, npr. u MS ACCESS-u, a treba na strani servera instalisati MS SQL SERVER, i ako je ispoštovana prethodno definisana modularnost, ovakvi poslovi se obavljaju preko vikenda. Ako korisnik u okviru održavanja sakuplja odgovarajuća iskustva, to će ovaj posao brže i bezbolnije obaviti.

Ceo ovaj posao zaokružuju na početku spomenuti CASE alati, koji treba automatski da registruju svaku izmenu i omoguće ažurno održavanje dokumentacije.

Hardverske izmene

Harverske izmene se odnose na nadgradnju mreže u okviru klijent/server-arhitekture, gde je mreža sredstvo za prenos podataka koje omogućuje razmenu poruka između aplikacije klijent (koja šalje, prima i analizira podatke) i servera baze podataka (koji obrađuje zahteve za pristup bazi).

Predmet izmena može biti:

- topologija mreže,
- hardver i softver za umreženi klijent/server-sistem,
- aplikacioni posrednik za rad sa bazom podataka,
- aktivnosti u sistemu klijent/server.

Topologija mreže može biti: zvezdasta (gde istovremena komunikacija zavisi od centralnog čvora), linijska (gde se izvodi komunikacija jedan po jedan između čvorova) i prstenasta (tj. istovremena komunikacija bez zavisnosti od centra čvora).

Hardver i softver za umreženi klijent/server-sistem vezani su za:

- mrežni hardver,
- mrežni softver i
- komunikacioni softver.

Mrežni hardver čine: mrežni adapter (koji čine jedna ili više utičnica za priključivanje kablova preko upletenih parica ili optičkih vlakana), bežični mrežni adapteri (visokofrekventni radio-signali), modemi i telefonske linije za regionalne mreže (WAN), spoljni modemi za velike udaljenosti.

Mrežni softver omogućuje: kontrolu pristupa mreži; upravljanje redovima čekanja na štampaču; dodeljivanje prostora za korisničke datoteke; rad sa elektronskom poštom.

Komunikacini softver omogućava da razni računari u mreži šalju i primaju pakete programa preko mreže za koju je definisan komunikacioni protokol, tj. jezik sporazumevanja računara, i to:

- TCP/IP - Transmission Control Program / Internet Program (UNIX),
- SPX/IPX - Sequenced Packet Exchange / Internet Package Exchange (NetWare),
- mrežna skretnica (komunikacioni most).

Aplikacioni posrednik za rad sa bazom podataka ili softverski sloj (Application middleware) omogućava komunikaciju između različitih komponenti distribuirane aplikacije; aktivan je i na klijentu i na serveru i pri tom prevodi poruke koje oni međusobno razmenjuju. Postoji i konverzacioni posrednik koji, prvo, omogućuje povezivanje, a potom razmenjivanje poruka, da bi u jednom trenutku poruka išla u jednom smeru (asinhrono), ali nije mnogo efikasan. Pozivanje udaljenih procedura (RPC-Remote Procedure Call) omogućuje jednostavni poziv procedura, i to procedura niskog nivoa, koje su jednostavne za upotrebu. API (Application Programming Interface) predstavlja specifičnu skupinu f-ja koji omogućavaju komunikaciju između aktivnosti koje se odvijaju na klijentima i onih na serverima vezanim sa ODBC-Open DataBase Connectivity i IDAPI-Integrated Database Application Programming Interface. API *programski interfejs* koristi se za različite tipove servera baze podataka i to je najmanji zajednički imenitelj.

Aktivnosti u sistemu klijent/server su zadatak čijim izvršenjem upravlja OS, a ako su u pitanju višeprocesni OS, onda se istovremeno upravlja sa više zadataka. Server baze podataka u klijent/server-sistemu radi pod višeprocesnim OS (UNIX, NT ili VMS). *Aktivnost klijent* izvršava aplikacije klijent koje se povezuju sa serverom baze podataka i pri tom se definiše zahtev za povezivanje, tj. ime korisnika i lozinka. *Aktivnost server* obrađuje zahteve koje šalje Aktivnost klijent i pri tom neposredno obrađuje proces koji ide od klijenta i upisuje podatke u datoteku sa podacima, tj. vodi datoteku dnevnika transakcija.

Prilog 1. Teoretske postavke relacionih modela

Struktura relacionih modela je jednostavna i prihvatljiva za svakog korisnika, jer relaciona baza podataka predstavlja skup tabela. Same relacije iz skupa tabela (baze podataka) generišu izlaz, takođe, u obliku jednostavne i lako prihvatljive tabele.

S druge strane, jednostavna je i formalno matematička interpretacija tabela, tj. tabela se može definisati kao matematička relacija.

Da bi se pristupilo razmatranju relacionih modela potrebno je dati definicije za:

- kartenzijanski (Dekartov) proizvod,
- relacije,
- domen relacije,
- stepen relacije,
- kardinalnost relacije,
- atribute relacije,
- ključeve,
- primarni ključ,
- spoljne ključeve,
- bazne relacije,
- izvedene relacije,
- šemu relacione baze podataka i
- null vrednost.

Kartezijanski proizvod od n skupova D1 x D2 x...Dn je skup svih mogućih uređenih n-torki

<d1,d2,....dn> tako da je $d1 \in D1$, $d2 \in D2$ ...$dn \in Dn$

Primer: Data su dva skupa brojeva:

D1 = {1,2,3,4} i D2 = {4,5}

D1 x D2 = {<1,4>,<1,5>,<2,4>,<2,5>,<3,4>,<3,5>,<4,4>,<4,5>

Relacija se definiše kao podskup Dekartovog proizvoda nad n-skupova, tj. podskup sadrži one n-torke Dekartovog proizvoda koje zadovoljavaju zadatu relaciju.

$R \subseteq D1 \times D2 \times ... \times Dn$

Primer: Definisati za prethodni primer nad skupovima D1 i D2 relaciju:

R: A x B ={ <d1,d2>| d1 = d2/2}

R = {<2,4>}

Domen relacije R definisan je okvirima skupova D1, D2,...Dn.

Stepen relacija je definisan brojem domena nad kojima je definisana neka relacija. Razlikuje se unarni stepen relacije nad jednim domenom, binarni nad dva i n-arni nad n-domena.

Kardinalnost relacije je broj n-torki u relaciji.

Po definiciji, Dekartov proizvod predstavlja skup uređenih n-torki i redosled elemenata u jednoj n-torki je bitan. Ako bi se vrednostima elemenata n-torki pridružila semantička imena domena, redosled elemenata u n-torkama postaje beznačajan.

Na primer, za definisane domene:

SIFRAR = {827369,827499,827521}
PREZIME= {STEVIC,ALAGIC,VUKIC}
RUKOV = {827902,827698,}

relacija se definiše ovako:

RADNIK $\subseteq$ SIFRARxPREZIMExRUKOV =
{<827369,STEVIC,827902>,
<827499,ALAGIC,827698>,
<827521,VUKIC,827698>}

Prvi element trojke uzima vrednost iz prvog, drugi iz drugog, a treći iz trećeg skupa.

Ako bi se vrednostima elemenata u n-torkama pridružila imena domena, redosled elemenata u n-torkama je beznačajan:

RADNIK $\subseteq$ SIFRAR x PREZIME x RUKOV =
{ <SIFRAR:827369,PREZIME:STEVIC, RUKOV:827902>,
<PREZIME:ALAGIC, SIFRAR:827499, RUKOV:827698>,
<RUKOV:827698,SIFRAR:827521,PREZIME:VUKIC>}

Atributi relacije

Atributi relacije su imenovani domeni sa imenom koje definiše ulogu domena u relaciji. Atributi relacije RADNIK su SIFRAR, PREZIME, RUKOV i dr.

Ovako definisan koncept atributa omogućuje predstavljanje relacija kao tabela, pa se relacija RADNIK može predstaviti sa:

SIFRAR	PREZIME	RUKOV
827369	STEVIC	827902
827499	ALAGIC	827698
827521	VUKIC	827698

Razlika između relacije i tabele je u tome što je relacija skup, a svaka tabela to nije. Stoga se definišu uslovi koje tabela mora da zadovolji da bi bila relacija:

- ne mogu postojati duple vrste tabela;
- redosled vrsta nema značaja;
- redosled kolona nema značaja;
- nisu dozvoljeni atributi ili grupe atributa sa ponavljanjem.

Prikazana tabela zadovoljava uslove da jedna tabela bude relacija. Ako je zadovoljen poslednji uslov, onda se kaže da se tabela nalazi u prvoj normalnoj formi.

Ključevi

Ključevi se definisu kao jedan ili više atributa čija vrednost jedinstveno identifikuje jednu n-torku u relaciji, tj. jedan red u tabeli. Ako je ključ definisan samo jednim atributom, onda je to prost ključ (npr., u relaciji RADNIK ključ je SIFRAR). Ako je ključ definisan sa više atributa, onda je to složeni ključ (npr., relacija CERTIFIKAT-ključ je SIFRAR, SIFRAJ).

Ako se definiše relacija R, onda se može reći da ključ relacije R predstavlja kolekcija K njenih atributa koja zadovoljava :

- osobinu jedinstvenosti i
- osobinu neredundantnosti.

Osobina *jedinstvenosti* je vezana za ograničenje da ne postoje bilo koje dve n-torke sa istom vrednošću K.

Osobina *neredundantnosti* se odnosi na gubljenje osobina jedinstvenosti ako se bilo koji atribut izostavi iz K.

U jednoj relaciji može postojati više različitih kolekcija K atributa koje zadovoljavaju definiciju ključa; sve se one nazivaju kandidati za ključ.

Primarni ključ je jedan od izabranih kandidata za ključ koji služi za identifikaciju n-torke relacije. Ostali kandidati za ključ postaju alternativni ključevi.

Atributi koji učestvuju u ključevima nazivaju se ključni atributi, dok se ostali nazivaju opisni atributi.

Spoljni ključ ili preneseni ključ je atribut ili grupa atributa u relaciji R, čija se vrednost koristi za povezivanje sa vrednošću primarnog ključa u nekoj relaciji R2. Spoljni ključevi i njima odgovarajući primarni ključevi definisani su nad istim domenom. Dakle, spoljni ključevi služe da se uspostave veze između relacija u relacionoj bazi podataka. Na primer, u relaciji RADNIK spoljni ključ SIFRAO vezuje sa relacijom ODELJENJE.

Bazna relacija je relacija koja se ne može izvesti iz ostalih relacija u relacionoj bazi podataka.

Izvedena relacija je relacija koja se izvodi iz skupa baznih i izvedenih relacija, i to preko operacija koje se definišu nad relacijama.

Kako se u tekstu koriste termini vezani za relacije, tabele i klasičnu obradu podataka, to se u sledećoj tabeli definišu njihove veze:

RELACIONA	TABELARNA	KLASICNA OBRADA
DOMEN	SKUP VREDNOSTI	TIP
ATRIBUT	KOLONA	POLJE
N-TORKA	RED	SLOG (RECORD)
PRIMARNI KLJUC	IDENTIFIKATOR REDA	JEDINSTVEN KLJUC
RELACIJA	TABELA	DATOTEKA
STEPEN	BROJ KOLONA	BR.POLJA U SLOGU
KARDINALNOST	BROJ REDOVA	BR.SLOGOVA U DAT.

Ekstenzija relacije je skup svih n-torki date relacije, odnosno predstavljanje tabele navođenjem svih vrsta. Tabela RADNIK predstavlja ekstenziju odgovarajućih relacija.

Intenzija relacije je generalizacija ekstenzije; ona je vremenski nepromenljiva i označava se na sledeći način:

RADNIK (#**SIFRAR**,PREZIME,RUKOV)

(Ispred zagrade je naziv relacije, u zagradi su navedeni atributi, a primarni ključ je obeležen masnim otiskom.)

Šema relacione baze podataka

Šema relacione baze podataka je predstavljanje strukture relacione baze kao skupa intenzija relacija.

RADNIK (#**SIFRAR**, PREZIME, RUKOV, DATUMZ, PLATA,
STIMUL,*SIFRAO)
ODELJENJE (#**SIFRAO**, NAZIVO, MESTO)

Null vrednosti

Null vrednosti se koriste da se označe još nepoznate vrednosti za neki atribut ili neprimenjivo svojstvo za neki objekat ili vezu koju predstavlja tabela (označavaće se znakom ?).

Relaciona algebra

Relaciona algebra definiše skup operacija pomoću kojih se, na proceduralan način, može dobiti željena relacija, tj.tabela iz skupa datih relacija.

Relacija se definiše kao podskup Dekartovog proizvoda i može se tretirati kao skup n-torki.

Za nivo relacione algebre operacije: unija, presek i diferencije nad relacijama R1 i R2 moraju zadovoljiti uslov kompatibilnosti (union compatible relations), tj. relacije R1 i R2 moraju imati isti broj atributa (isti stepen), a odgovarajući atributi su definisani nad istim domenom.

Na nivou relacione algebre, operacije su:

- unija,
- diferencija,
- presek,
- Dekartov proizvod,
- projekcija,

- selekcija (restrikcija),
- spajanje (join),
- deljenje,
- operacije sa null vrednostima.

Unija

Ako su date relacije R1 i R2 koje zadovoljavaju uslove kompatibilnosti, onda je rezultat operacije unije

R3 = R1 U R2

relacija R3 koja sadrži sve n-torke koje se pojavljuju bilo u R1, bilo u R2.

Diferencija

Ako su date relacije R1 i R2 koje zadovoljavaju uslov kompatibilnosti, onda rezultat operacije diferencije

R3 = R1 - R2

predstavljaju n-torke relacije R1, koje nisu istovremeno i n-torke relacije R2.

Presek

Ako su date relacije R1 i R2 koje zadovoljavaju uslov kompatibilnosti, onda je rezultat operacije preseka

R3 = R1 $\wedge$ R2

relacija R3 koja sadrži n-torke koje se pojavljuju u obe relacije R1 i R2.

Kada se izdvoje operacije unije i diferencije koje su pogodne za ažuriranje baze podataka, operacijom unije se mogu u neku relaciju dodati nove n-torke, a operacijom diferencije se mogu iz neke relacije izbaciti neželjene n-torke. Izmena vrednosti pojedinih atributa u nekoj relaciji vrši se uzastopnom primenom operacije diferencije, pomoću koje se izbaci cela n-torka u kojoj se nalazi posmatrani atribut, a zatim se operacijom unije "vraća" izbačena n-torka sa promenjenom vrednošću atributa.

Dekartov proizvod

Dekartov proizvod se može primeniti na bilo koje dve relacije. Rezultat je ove operacije

R3 = R1 x R2

relacija R3 čije su n-torke svi "parovi" koje čine jedna n-torka relacije R1 i jedna n-torka relacije R2.

Projekcija

Operacija projekcije se definise kao unarna operacija koja iz neke relacije selektuje skup navedenih atrubuta, odnosno "vadi" vertikalni podskup iz odgovarajuće tabele.

Formalno, projekcija se definiše na sledeći način:

Neka je R (A1,A2,...,An) relacija, a X podskup njenih atributa.

Ako se sa Y označi komplement {A1,A2,...An}-X rezultat operacije projekcije relacije R po atributima X glasi:

P[X], = {x | tako da postoji y da je <x,y> $\in$ R}

Operacijom projekcija eliminišu se duplikati n-torki.

Selekcija (Restrikcija)

Selekcija je unarna operacija koja iz date relacije selektuje n-torke koje zadovoljavaju zadati uslov ("vadi" horizontalni podskup tabele).

Formalno se definiše na sledeći način:

Dati su relacija R (A1,A2,...An) i predikat P, definisan nad njenim atributima. Rezultat operacije selekcije glasi:

S[P]R = {x | x $\in$ R i P (x)}

Spajanje (Join)

Spajanje je binarna operacija koja spaja dve relacije na taj način da se u rezultatu pojavljuju oni parovi n-torki jedne i druge relacije koji zadovoljavaju uslov zadat nad njihovim atributima.

Formalno se definiše na sledeći način:

Date su relacije R1 (A1,A2,...,An) i R2 (B1,B2,...,Bm) i predikat θ, definisan nad njihovim atributima.

Ako se sa X i Y obeleže skupovi atributa relacija R1 i R2, respektivno, rezultat operacije spajanja ovih relacija (tzv. θ-spajanje) glasi ovako:

R1[xθ]R2 = {<x,y> | x $\in$ R1 AND y $\in$ R2 AND θ (x,y)}

Oznaka xθ za operaciju spajanja ukazuje na činjenicu, očiglednu iz definicije θ spajanja, da ova operacija nije primitivna operacija relacione algebre, već da se može izvesti uzastopnom primenom operacije Dekartovog proizvoda (x) i selekcije po predikatu θ iz tako dobijene relacije.

Ako je predikat θ definisan sa Ak = Bj, s tim da su i atributi Ak i Bj definisani nad istim domenima, tada se takvo spajanje naziva ekvispajanje.

Očigledno je da se u rezultatu ekvispajanja uvek pojavljuju dve iste kolone. Ako se jedna od te dve kolone izbaci, takvo spajanje se naziva prirodno spajanje.

Deljenje

Deljenje je operacija pogodna za upite u kojima se javlja reč "svi".

Formalno se definiše na sledeći način:

Neka su A (X,Y) i B (Z) relacije, gde su X, Y i Z skupovi atributa takvi da su Y i Z jednakobrojni, a odgovarajući domeni su im jednaki.

Rezultat operacije deljenja je

A[Y) Z]B = R (X)

gde n-torka X uzima vrednosti iz A.X, a par <x,y> postoji u A za sve vrednosti y koje se pojavljuju u B (Z).

Operacija deljenja nije primitivna operacija relacione algebre, već se može izvesti na sledeći način:

A(X,Y)[Y) Z,B(Z) = P[X,A - P[X,((P[X,A x B)- A)

Objašnjenje:

P[X]A daje sve n-torke koje mogu da učestvuju u rezultatu.

P[X]A x B daje relaciju u kojoj se za svaku vrednost z iz B pojavljuju parovi <x,z> sa svim vrednostima x.

P[X]A x B) - A ne sadrži ni u jednom paru <x,z> one vrednosti x za koje u relaciji A, kao vrednosti y, postoje sve vrednosti z.

Kompletan izraz, prema tome, jeste relacija koja sadrži one n-torke x za koje postoje u paru <x,y>, kao vrednosti y, sve vrednosti z.

Relaciona algebra je proceduralni jezik. Pomoću operacija relacione algebre sačinjava se procedura koja dovodi do odgovora na postavljeni upit.

Mada je proceduralni jezik, relaciona algebra je znatno moćnija od klasičnih programskih jezika, koji su, takođe, proceduralni. Razlog za to je što operand relacione algebre predstavlja relaciju (cela tabela), a operand operacija sa datotekama u klasičnim jezicima je rekord (vrsta tabele).

Može se pokazati da se bilo koja relacija (bilo koji upit), izvodljiva iz skupa datih relacija, može dobiti procedurom (algoritmom) od tri koraka:

1. Dekartov proizvod svih relacija koje se koriste;
2. selekcija n-torki za koje se predikat selekcije sračunava u T;
3. projekcija rezultata po atributima koji se prikazuju.

Ova procedura se može iskazati jednim opštim izrazom relacione algebre:

P[A1, A2, ..., An, (S[P, (R1 x R2 x ... x Rm))

Međutim, ovakvo izvođenje operacija bilo bi veoma "skupo", jer bi rezultat Dekartovog proizvoda relacija R1, R2, ..., Rm bio relacija sa k1 x k2 x ... x km n-torki, gde su sa ki označene kardinalnosti odgovarajućih relacija. Zbog toga se i definiše operacija spajanja, koja istovremeno obavlja Dekartov proizvod i selekciju, smanjujući na taj način broj n-torki koji se pretražuje. Pored toga, očigledno je da su znatno efikasniji algoritmi sa više koraka, u kojima bi se prvo vršile sve selekcije koje se mogu izvesti na pojedinačnim relacijama, zatim projekcije, pa tek onda spajanja.

Operacije sa nula vrednostima

Navedene operacije relacione algebre nisu uzimale u obzir nula vrednosti, koje mogu postojati u relacijama. Podrazumevalo se da se vrednosti predikata sračunavaju na osnovu standardnih dvovrednosnih tablica istinitosti. Isto tako, bez dvoumljenja je bilo moguće da se odrede duplikati n-torki, odnosno kardinalnost pojedinih skupova. Pojavljivanje nula vrednosti u relacionoj bazi podataka zahteva da se proširi skup definisanih operacija koje bi na neki način uključile i nula vrednosti. Osnovne postavke za operacije sa nula vrednostima su sledeće:

(1) Tablice istinitosti trovrednosne logike:

AND	T	?	F
T	T	?	F
?	?	?	?
F	F	F	F

OR	T	?	F
T	T	T	T
?	T	?	?
F	T	?	F

NOT	
T	F
?	?
F	T

(Znak ? predstavlja nula vrednost, a može se u okviru tablica istinitosti čitati i kao "možda".)

(2) Sračunavanje aritmetičkih formula. Neka " označava neki od aritmetičkih operatora (+, -, *, /) i neka su x i y dve numeričke vrednosti. Vrednost aritmetičkog izraza "x " y" je, po definiciji, nula vrednost, ako bilo x, bilo y, bilo oba dobiju nula vrednost.

(3) Skupovi sa nula vrednostima. Da li kolekcija {1,2,3,?}, predstavlja skup? Ako ? uzme vrednost 1,2 ili 3, onda nije, ili se može tretirati kao skup sa kardinalnošću 3. Ako ? nije ni 1, ni 2, ni 3, onda gornja kolekcija predstavlja skup sa kardinalnošću 4. To pokazuje da postoje različite nula vrednosti: nula vrednost koja nije 1, nula vrednost koja nije 2, zatim nula vrednost koja nije ni 1, ni 2, i tako dalje. Operacije koje bi uzele u obzir različitost nula vrednosti bile bi veoma složene. Zbog toga se operacije definišu jednom vrstom nula vrednosti, a ostale nula vrednosti se tretiraju kao duplikati.

Imajući u vidu ove opšte postavke, primeri nekih ranije definisanih operacija na relacijama koje poseduju nula vrednosti izgledaju ovako:

UNIJA

R3 = R1 U R2

R1	A	B
	a	b
	a	?
	?	b
	?	?

R2	A	B
	x	y
	?	b
	?	?

R3	A	B
	a	b
	a	?
	?	b
	?	?
	x	y

DIFERENCIJA

R4 = R1 - R2

R4	A	B
	a	b
	a	?

DEKARTOV PROIZVOD

Dekartov proizvod ostaje neizmenjen.

SELEKCIJA

Operacija selekcije ostaje neizmenjena, selektuju se one n-torke za koje se odgovarajući predikat sračunava u T na osnovu trovrednosnih tablica istinitosti. Ova operacija se često zove i "TRUE SELECTION" (istinita selekcija).

S[A = a,R1

A	B
a	b
a	?

PROJEKCIJA

Uzima se u obzir da su nula vrednosti duplikati (jedna vrsta nula vrednosti):

P[A,R1

A
a
?

SPAJANJE

Operacija spajanja ostaje neizmenjena, jer operacije Dekartovog proizvoda i selekcije ostaju neizmenjene. U rezultatu se pojavljuju one n-torke za koje se predikat spajanja sračunava u T na osnovu trovrednosnih tablica istinitosti (TRUE TETA JOIN - istinito θ spajanje).

DODATNE OPERACIJE

Zbog postojanja nula vrednosti u bazi podataka, neophodno je da se definiše logička funkcija "IS_NULL", čiji argument može da bude neki skalarni izraz, a koja se sračunava u T ako je vrednost tog argumenta nula vrednost, a inače uzima vrednost F.

MOŽDA SELEKCIJA (MAYBE_SELECT)

Selektuju se one n-torke relacije za koje se predikat selekcije sračunava u nula vrednost na osnovu trovrednosnih tablica istinitosti.

MAYBE_SELECT[A]R3

A	B
?	b
?	?

MOŽDA SPAJANJE (MAYBE_JOIN)

U rezultatu spajanja se pojavljuju one n-torke za koje se predikat spajanja sračunava u nula vrednost na osnovu trovrednosnih tablica istinitosti.

Ra = Rb[MAYBE_JOIN A = D]Rc

R b	A	B	R c	C	D	E	Ra	A	B	C	D	E
	a1	b 1		c1	?	e1		a1	b 1	c1	?	e1
	a2	b 2		c2	d 2	e2		a1	b 1	c3	?	e3
	?	b 3		c3	?	e3		a2	b 2	c1	?	e1
								a2	b 2	c3	?	e3
								?	b 3	c1	?	e1
								?	b 3	c2	d 2	e2
								?	b 3	c3	?	e3

SPOLJNO SPAJANJE (OUTER_JOIN)

Neka su date relacije R1 (A,B) i R2 (C,D). Pretpostavka je da se vrši operacija ekvispajanja ovih relacija R1[A = C]R2. Ako je P[A]R1 # P[C]R2 (projekcije relacija po atributima spajanja su različite), tada će se u rezultatu spajanja "izgubiti" neke n-torke relacija R1 i R2. Ako se takvo gubljenje informacija ne želi, SPOLJNO_SPAJANJE ih može sačuvati na taj način što se u rezultat dodaju i ove n-torke, i to tako što za n-torke relacije R1 atributi C i D uzimaju nula vrednosti, a za n-torke relacije R2 atributi A i B uzimaju nula vrednosti.

Primer: R1[OUTER_JOIN A = C]R2

R 1	A	B	R 2	C	D	R 3	A	B	C	D
	1	2		3	4		2	1	2	2
	2	1		2	2		1	2	?	?
	4	?					?	?	3	4
							4	?	?	?

SPOLJNA UNIJA (OUTER_UNION)

Operacija unije se, kao što je rečeno, može izvesti samo nad relacijama koje zadovoljavaju kriterijume kompatibilnosti (da su istog stepena i da su im odgovarajući atributi definisani nad istim domenima). Dodavanjem novih atributa i postavljanjem njihovih vrednosti na nula vrednosti mogu se uvek dve nekompatibilne relacije učiniti kompatibilnim. Spoljna unija podrazumeva da su, na taj način, dve nekompatibilne relacije učinjene kompatibilnim, pa je tada izvršena operacija unije.

Primer: R3 = R1 OUTER_UNION R2

R 1	A	B	C	R 2	A	D	R 3	A	B	C	D
	a1	b 1	c1		a1	4		a1	b 1	c1	?
	a2	b 1	c2		a1	7		a2	b 1	c2	?
					a2	5		a1	?	?	4
								a1	?	?	7
								a2	?	?	5

Relacioni račun

Relacioni račun je neproceduralni način iskazivanja operacija, gde se pomoću konstrukcije predikatskog računa prvog reda, u kome su promenljive date kao n-torka relacija ili domeni relacija, koriste za definisanje osobina relacija koje se žele dobiti.

Relacioni račun n-torki

Relacioni račun n-torki je predikatski račun prvog reda u kome domeni promenljivih predstavljaju n-torke relacija date baze podataka.

Osnovni pojmovi predikatskog računa su:

- afirmativna rečenica, koja ima smisla i koja je istinita ili neistinita i naziva se sud;
- afirmativna rečenica, koja ima smisla i koja sadrži jedan ili više promenljivih parametara i postaje sud uvek kada parametri iz rečenice dobiju konkretnu vrednost, naziva se predikat; broj parametara u predikatu se naziva dužina predikata (primer predikata je x + y # 1);
- predikatski ili kvantifikatorski račun, kao matematička teorija, čiji su objekti formule koje predstavljaju predikate; simboli koji se koriste da označe neki sud nazivaju se atomskim formulama ili atomima. Atomi u relacionom računu n-torki su:
- A $\in$, R gde je x n-torka promenljiva, a R relacija, odnosno promenljiv x uzima vrednosti iz skupa n-torki relacije R;
- x.A θ y.B gde su x i y promenljive (n-torke), A i B su atributi relacija R1 i R2 iz čijih n-torki, respektivno, promenljive x i y uzimaju vrednosti (x $\in$ R1, y $\in$ R2), a θ je operacija poređenja definisana nad domenom atributa A i B (A i B moraju biti definisani nad istim domenom);
- x.A θ c gde su x, A i θ kao i u prethodnom stavu, a c je konstanta koja ima isti domen kao i A.

Formule se formiraju od atoma preko sledećih pravila:

- atom je formula;
- ako je P1 formula, tada su formule i NOT P1 i (P1);
- ako su P1 i P2 formule, tada su formule i P1 AND P2 i P1 OR P2;
- ako je P1(s) formula koja sadrži neku slobodnu promenljivu s, tada su i $\exists$ s (P1(s)) i $\forall$ s (P1(s), takođe, formule ($\exists$ - "postoji", egzistencijalni kvantifikator, $\forall$ - "za svako", univerzalni kvantifikator).

Jedna promenljiva u nekoj formuli se može pojaviti više puta. Promenljive mogu biti "slobodne" i "vezane". Vezana promenljiva u formuli je neka vrsta "prividne" (dummy) promenljive i ima ulogu da poveže promenljivu iza kvantifikatora sa promenljivima u zoni dejstva kvantifikatora. Drugim rečima, vezana promenljiva u je ($\exists$ u), ($\forall$ u) ili ($\exists$ u) A ili ($\forall$ u) A, gde je A formula u kojoj se pojavljuje u. Sve promenljive koje u nekoj formuli nisu vezane, slobodne su u toj formuli.

Na primer, u formuli: $\exists$x (x > 3), x je vezana promenljiva, pa je gornja formula ekvivalentna sa $\exists$y (y > 3).

U formuli: $\exists$x (x > 3) AND x < 0, prvo pojavljivanje promenljive x je vezano, a drugo slobodno, pa je ova formula ekvivalentna sa $\exists$y (y > 3) AND x < 0.

Neka su R1, R2, ..., Rn relacije u nekoj bazi podataka. Neka su A, B ..., C atributi ovih relacija, respektivno, i neka je f - formula. Opšti izraz relacionog računa n-torki je tada:

t $\in$ R1, u $\in$ R2, ..., v $\in$ Rn
t.A, u.B, ..., v.C GDE_JE f

(Prikazuju se vrednosti atributa A relacije R1, atributa B relacije R2, ... i atributa C relacije Rn za one n-torke koje zadovoljavaju uslov definisan formulom f.)

Relacioni račun domena

U relacionom računu domena promenljive uzimaju vrednosti iz nekih domena definisane relacione baze podataka. Ovde se, pored navedenih, definiše još jedna atomska formula, tzv. uslov članstva (membership condition). Uslov članstva ima sledeći oblik:

R (term, term, ...)

gde je R ime neke relacije, a svaki term ima oblik A:v, gde je A neki atribut relacije R, a v je ili promenljiva ili konstanta.

Uslov članstva se izačunava u T (TRUE) ako postoji n-torka u relaciji R, koja ima zadate vrednosti navedenih atributa.

Opšti izraz relacionog računa domena je:

x, y, ...,z GDE_JE f

gde su x,...,z promenljive, a f je formula koja uključuje i uslov članstva.

Relaciona algebra i relacioni račun (i račun n-torki i račun domena) fundamentalno su ekvivalentni jedno drugom. E.F Codd je pokazao da se svaki izraz relacionog računa može svesti na semantički ekvivalentan izraz u relacionoj algebri, a Ullman - da se svaki izraz u relacionoj algebri može svesti na izraz relacionih računa, pa se na osnovu toga može zaključiti da si ova dva formalizma logički ekvivalentna.

Relaciona baza podataka

Relaciona baza podataka stvara takav tip strukture koji se koristi za izražavanje odnosa između podataka u obliku jednostavnih dvodimenzionalnih tabela. Za razliku od većine drugih baza podataka (hijerarhijskih i mrežnih), relaciona baza podataka ima solidne teoretske osnove, za koje ima najveće zasluge dr E.F.Codd, koji je objavio prve rezultate svojih istraživanja već 1969. godine. Budućnost ovog sistema je velika, jer je za korisnika relacioni sistem znatno jednostavniji od ostalih.

U relacionom sistemu je baza podataka izražena skupom vremenski promenljivih "relacija". SUBP omogućava sve potrebne operacije sa relacijama i pri tom dozvoljava korišćenje najrazličitijih kombinacija postojećih relacija, da bi time korisniku omogućio traženje odgovora na sva pitanja koja imaju smisla s obzirom na sadržaj baze podataka.

Svaka relacija ima sva svojstva skupa. Osnovno svojstvo svakog skupa je da se elementi koje on sadrži međusobno razlikuju. Tako se i svi redovi relacije međusobno razlikuju. To znači da u relaciji uvek postoji neki atribut (ili neka kombinacija atributa), koji omogućava jednoznačnu identifikaciju svakog retka. Takav atribut (odnosno takva kombinacija atributa) koristi se kao "ključ" relacije.

Kolona (atribut) u relaciji je uvek homogena u tom smislu i sadrži samo vrednosti iz jednog domena. Redosled redova u relaciji nije važan, jer se redovi ne identifikuju na osnovu svog položaja već na osnovu ključa. Takođe, ni redosled kolona nije važan, jer smisao kolone nije određen relativnom pozicijom kolone u relaciji, već imenom atributa.

Prilog 2. Alati za projektovanje informacionih sistema i SUBP (CASE alati)

Razvoj informacione tehnologije karakteristiše zaostajanje softvera u odnosu na hardver. Pomenuti nedostatak softvera, koji se često naziva softverska kriza, nastaje zbog niske produktivnosti i visokih proizvodnih troškova.

Rešenje softverske krize je u iskorišćenju osobina inženjera proverenih u praksi, i to, pre svega, metodičnosti i operativne discipline. Kao rezultat nastaje softverski inženjering koji u sebi sadrži sistematizovane i koordinirane aktivnosti potrebne pri projektovanju, implementaciji, eksploataciji i održavanju softverskih proizvoda.

Dalji razvoj softverskih sistema na današnjem nivou mogućnosti računara i očekivanja korisnika, zahteva visokostručan rad i programiranje za svoju realizaciju. Pošto je ručno razvijanje softvera od najnižeg nivoa skupo i dugotrajno i sa ne uvek predvidivim rezultatima, postoji potreba da se razvoj softvera olakša, zbog čega je, pre više od dvadeset godina, nastalo softversko inženjerstvo kao disciplina.

Automatizacija softverskog inženjeringa na računaru se izvodi posebnim alatom, čiji je naziv CASE (*Computer Aided Software Engineering).*

Definicija CASE

Computer Aided Software Engineering (CASE) alati služe za automatizaciju softverskog inženjerstva i samim tim predstavljaju osnovni alat kojim se služi projektant informacionog sistema.

Prve definicije CASE alata su podrazumevale da predstavljaju sisteme čiji su ciljevi da definišu, integrišu i automatizuju što je moguće više faza u razvoju softvera. CASE alati omogućavaju da razvijanje softvera postane više inženjerska delatnost, a manje individualna umetnost i umeće.

Zavisno od toga koje faze projektovanja i implementacije CASE alati pokrivaju, oni se dele na CASE alate na višem i CASE alate na nižem nivou. CASE alati na višem nivou pokrivaju prve faze u proizvodnji softvera (analizu sistema i projektovanje), a CASE alati na nižem nivou pružaju pomoć u fazama programiranja.

CASE sistem predstavlja alat koji služi kao pomoć projektantu informacionih sistema. Od efikasnosti ovog alata može da zavisi kvalitet gotovog proizvoda (informacionog sistema), tako da je projektantu veoma važno da odabere pravi alat koji će ga zamenjivati u većini manuelnih poslova vezanih za projektovanje.

Od efikasnosti CASE alata može zavisiti kvalitet gotovog softverskog proizvoda. Uspešnim korišćenjem pravilno odabranog CASE alata može se:

- minimizirati vreme i trud (koštanje) razvoja softvera,
- višestruko povećati produktivnost u pisanju softvera,
- podići nivo kvaliteta,
- povećati pouzdanost,
- standardizovati proizvedeni softver.

Podele CASE alata

Pored navedene podele CASE alata na više (UPPER) i niže (LOWER), koja je ujedno i najšire prihvaćena, postoje i sledeće dve podele:

1. horizontalna - vremenski:
    - viši alati - za više faze životnog ciklusa (korisnički zahtevi i dizajn),
    - srednji alati - za srednje faze životnog ciklusa (implementacija - izrada),
    - niži alati -z a niže faze životnog ciklusa (podrška eksploataciji);
2. vertikalna - po funkciji:
    - alati za upravljanje, planiranje i procene,
    - tehnički alati (realizacija),
    - alati za podršku pojektu (skladišta, rečnici).

Integrisani CASE alati (Integrated CASE ili I-CASE) čine alate koji pokrivaju više vremenskih faza i više funkcija. Primer takvog alata je Oracle Designer/2000, koji u sebi sadrži čitav niz alata baziranih na integrisanom rečniku i koji pokriva sve faze izrade jednog informacionog sistema.

Alati mogu podržavati različite tehnike modeliranja poslovnih sistema, pa se i po tome mogu podeliti. Najrasprostanjenije tehnike modeliranja su:

- ER metodologija (ili proširena ER metodologija) po notaciji Chen-a, Bachman-a...;
- SSA (sistemska strukturna analiza za modeliranje funkcionalnog aspekta posmatranog sistema) po notaciji Yourdon/de Marco, Gane/Sarson, Ward/Mellor;
    - dijagrami toka programa (flow chart);
    - funkcionalne hijerarhije (dekompozicije poslovnih funkcija);
    - objektno-orijentisane metodologije, kao OMT i Grady Booch;

- dijagrami tranzicije stanja.

Kvalitetu CASE alata doprinosi činjenica da mogu podržavati više tehnika modeliranja.

Alati se mogu podeliti i po filozofiji rada sa njima na:

- alate sa ograničavajućom (restriktivnom) filozofijom, koji najviše pomažu novim korisnicima, jer ih ograničavaju na stvari koje prvo moraju da urade; npr., da nacrtaju kontekstni dijagram u SSA metodologiji, što pruža osećaj sigurnosti, mada, možda, malo guši kreativnost;
- alate sa vođenom filozofijom, koji su pogodni za korisnike višeg nivoa predznanja i intelektualne zadatke višeg nivoa, jer samo vode korisnika i asistiraju mu u biranju operacija;
- alate sa fleksibilnom filozofijom koji dopuštaju potpunu slobodu korisniku, čak i da generiše nekorektan dizajn; pogodni su za korisnike koji su već u potpunosti ovladali metodologijom i hoće da alat prati njihov kreativni proces stvaranja koji ide malo metodom sa vrha - nadole, a malo metodom sa dna - naviše. Ovakvi alati najviše vrše sintaksne provere interno i odmah po unosu, a kasnije provere ispravnosti (konzistentnosti) dizajna, i to isključivo na zahtev korisnika.

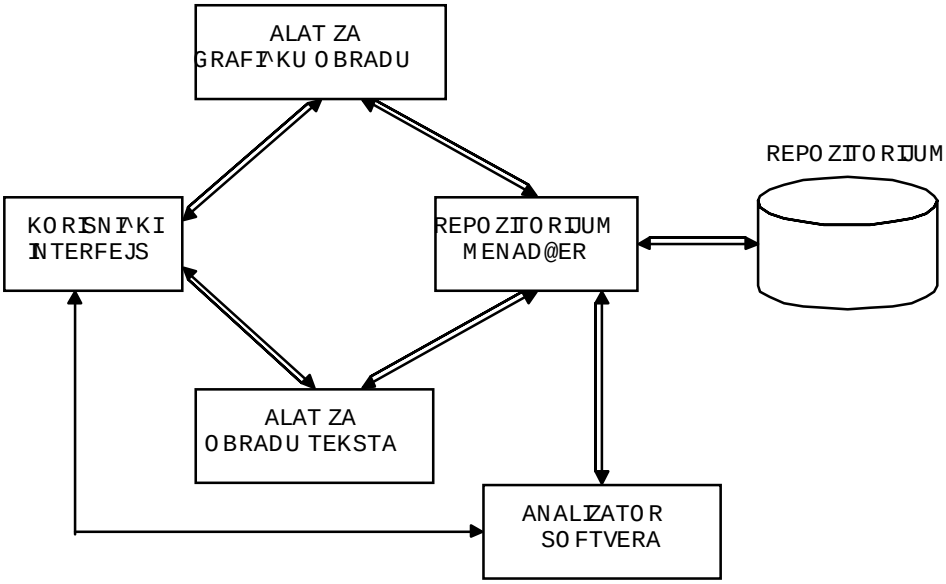
Dalje gledano, CASE alati mogu biti:

- jednokorisnički - projekti se vode na jednom mestu i samo jedan korisnik upotrebljava alat u jednom momentu;
- višekorisnički - omogućavaju komunkaciju i koordinaciju velikih timova za razvoj softvera; svima su dostupni planovi, statusni izveštaji, specifikacije, modifikacije, izvorni kod i testni podaci.

Postoji još veliki broj načina na koje se CASE alati mogu klasifikovati (na primer, po hardverskom okruženju i operativnom sistemu u kome rade, po jeziku na kome su razvijani ili po otvorenosti arhitekture), ali se ne odnose na funkcionalnost i mogućnosti nego pre na način razvoja samog alata.

Elementi CASE alata

Tipično CASE okruženje uključuje: repozitorijum, alate za grafičko predstavljanje, softver za definisanje teksta, softverski interfejs prema repozitorijumu, softver za procenu i interfejs prema korisniku kao što je prikazano na slici B.1.



Slika B.1. CASE arhitektura

Repozitorijum je aktivan rečnik podataka koji podržava definisanje različitih tipova objekata i njihovih veza. *Alati za grafičku obradu* omogućavaju razvijanje raznih tipova dijagrama, kao i ocenu kompletnosti dijagrama na osnovu unapred definisanih pravila.

Alat za obradu teksta omogućava definisanje naziva, sadržaja i detalja pojedinih stavki u repozitorijumu. *Korisnički interfejs* je interpreter koji određuje formu podataka koju treba uzeti (grafika ili tekst). *Analizator softvera* je ekspertni deo CASE i on analizira ulaze kako u dijagram tako i u repozitorijum, utvrđujući njihovu leksičku kompletnost, a i proveravajući kompatibilnost sa ostalim objektima u aplikaciji. Korisnički interfejs obezbeđuje interaktivnu i off-line obradu preko ekrana i izveštaja.

Idealni CASE treba da omogući potpunu automatizaciju kompletnog životnog ciklusa projekta informacionog sistema, obuhvatajući početnu inicijativu, nivo analize, rad na održavanju softverskog proizvoda sve do njegovog povlačenja. Takav CASE postaje žiža za sve poslove softverskog inženjerstva, pa se rad na razvoju softvera koncentriše na logički aspekt projektovanja. Upravo zbog toga idealni CASE treba da omogući: izradu arhitekture procesa organizacije, planiranje i praćenje projekta, grupni rad na razvoju softvera, aplikativno i ručno definisanje procedura, normalizaciju podataka, generisanje šeme baze podataka, generisanje koda u korisnički odabranom programskom jeziku, automatsko testiranje generisanog koda prema specifikaciji aplikacije i ekspertnu procenu savršenosti proizvoda, kao i način korekcije. Idealni CASE treba već u repozitorijumu da prepozna komponente koje su za ponovnu analizu, dizajn i kodiranje.

Po pravilu 40-20-40, koje važi za razvoj programskog sistema, 40% projektnog vremena otpada na analizu i modelovanje, 20% je odvojeno za programiranje, a preostalih 40% za testiranje. Trend dobavljača je da se eliminiše kodiranje, što čini samo jednu petinu vremena potrebnu za razvoj softverskog proizvoda. Sadašnji trenutak zahteva postojanje CASE alata koji će pokriti proces testiranja, a samim tim i skratiti vreme izrade softverskog proizvoda.

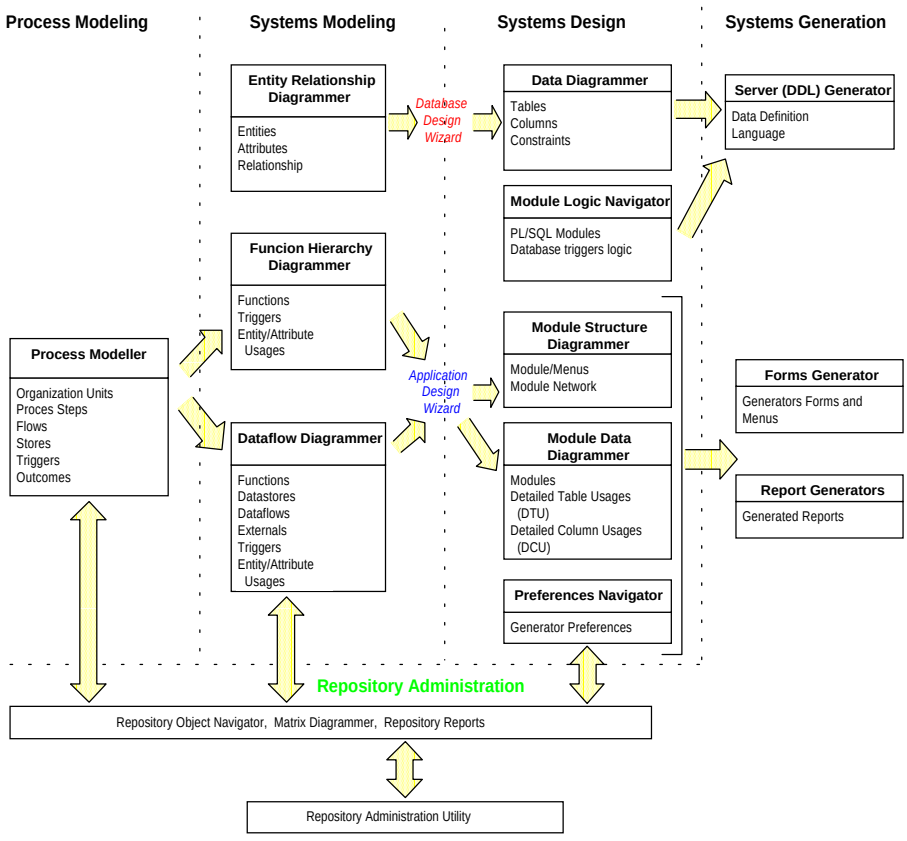
Budući CASE biće u mogućnosti da u ostatku 40% projektnog vremena identifikuje delove aplikacije koji mogu biti ponovo iskorišćeni. Iz navedenog se može pretpostaviti i put razvoja CASE alata, a to je efikasnije pokrivanje vremena za analizu i dizajn, kao i vremena potrebnog za testiranje.

ORACLE CASE Designer/2000

Odlika ORACLE CASE Designer/2000 alata je da su integrisani i da pokrivaju sve faze u razvoju sistema. Podržavaju koncept vodopada (Waterfall model), kao i "Spiral model" razvoja informacioNOg sistema. Svi alati za razvijanje dijagrama napisani su u C++ tako da je omogućeno korišćenje MS-Windows biblioteka. Koncepcija ORACLE CASE okruženja se bazira na jakim radnim stanicama u Windows ambijentu i skupu alata (modelera, dijagramera, utility-ja itd.) preko kojih se pristupa zajedničkom rečniku**.** Rečnik je baza podataka koja sadrži informacije potrebne u svim fazama razvoja sistema. Svaki podatak se u ovu bazu unosi samo jedanput, a onda se, po potrebi, referencira u svim slučajevima kada je to potrebno.

Karakteristika rečnika je da je višekorisnički i da se njegova konzistentnost stalno prati procedurama za održavanje referencijalnog integriteta i alatima za konsolidaciju. Različite klase korisnika mu pristupaju kroz različite alate u pojedinim fazama razvoja sistema.

Sledi prikaz arhitekture Designer/2000.

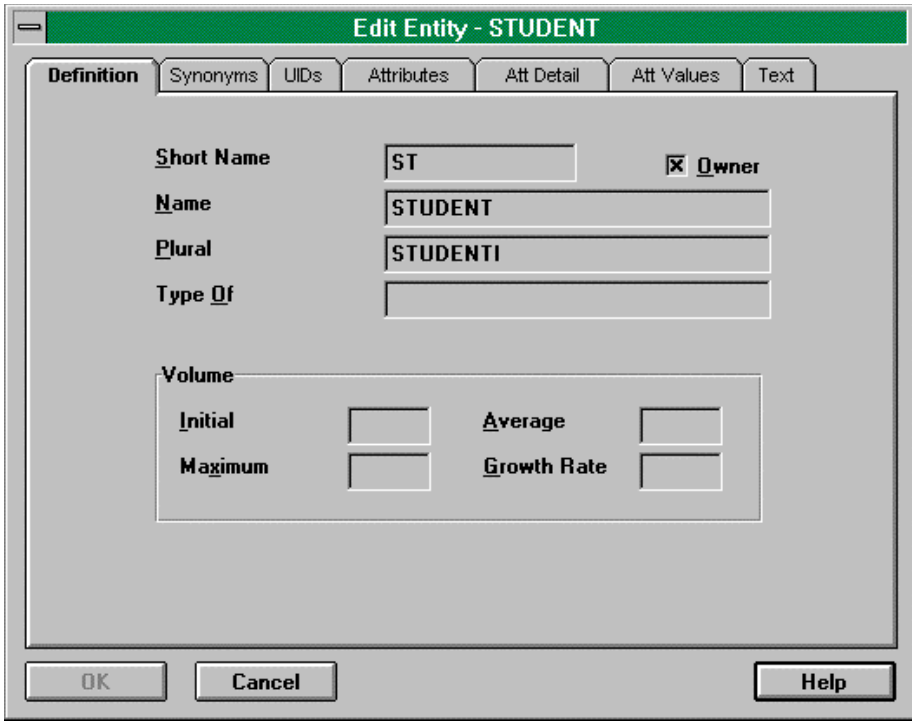


Slika B.2. Arhitektura Designer 2000

Entity Relationship Diagrammer (ERD)

Deo *Oracle* paketa *Designer/2000,* koji služi za modeliranje podataka, podržava sledeće koncepte: atribut, entitet, specijalizaciju i generalizaciju (podtipovi i nadtipovi), veze.

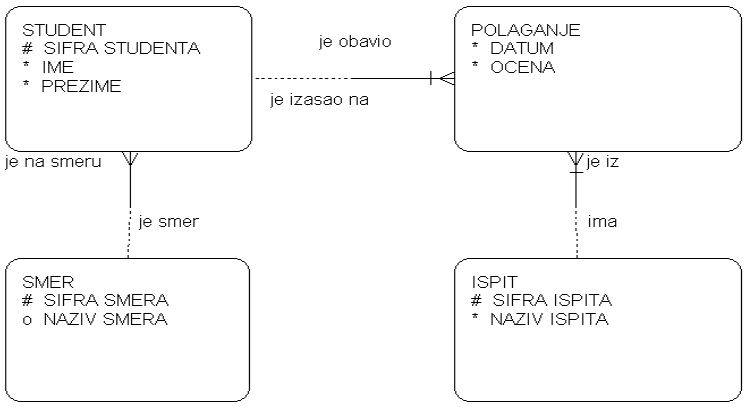
Pri kreiranju entiteta *ERD* zahteva samo njegovo ime, a svi ostali detalji vezani za entitet, kao i definisanje njegovih atributa, mogu se kasnije obaviti kroz opciju editovanja entiteta. Po želji se atributi mogu prikazivati u okviru entiteta na dijagramu i tada postoji konvencija da se: ispred atributa koji ulaze u sastav primarnog ključa stavlja znak "#", ispred atributa koji su obavezni oznaka "*" i ispred opcionalnih atributa oznaka "o", što doprinosi preglednosti dijagrama.



Slika B.3. ERD-*Ekranska forma za definisanje entiteta*

Pri modeliranju veza među entitetima zadaju se njihova opcionalnost (obaveznost veze se predstavlja punom linijom, a neobaveznost veze isprekidanom linijom (sa svake strane veze ponaosob) i njihova kardinalnost (veza sa gornjom granicom kardinalnosti 1 se predstavlja jednom tačkom dodira linije, koja predstavlja vezu i odgovarajućeg entiteta, dok se veza sa gornjom granicom kardinalnosti M predstavlja razgranatom linijom sa tri dodirne tačke).

Posebne vrste veza koje su podržane ovim *CASE* alatom su: prenosive i neprenosive veze (transferable i non-transferable), a odnose se na mogućnost prevezivanja jedne pojave entiteta sa drugom pojavom referentnog entiteta; spušteni ključevi, kada se kroz ovakvu vezu u ključ zavisnog entiteta uključuje kompletan ključ entiteta od koga egzistencijalno zavisi; rekurzivne veze, koje modeliraju vezu jedne pojave entiteta sa drugom pojavom istog entiteta; ekskluzivne veze, koje modeliraju situaciju u kojoj može u jednom trenutku postojati samo jedna od dve ili više veza koje polaze od jednog entiteta.



Slika B.4. ERD – *primer modela "objekti i veze" u grafičkom editoru*

Definisanje domena za *ERD* se vrši u *Repository Object Navigator*-u, alatu za direktan pristup rečniku podataka. Domeni definisani na taj način su dostupni i ostalim modulima *ORACLE*-ovog integrisanog *CASE*-a, a u *ERD*-u se koriste za definisanje atributa.

Svaki objekat u *ERD*-u ima karakteristiku *CREATE* koju projektant menja u *True* u trenutku kada je završio dizajniranje tog objekata. Na taj način se razlikuju objekti koji ulaze u generisanje izlaza od objekata koji ne ulaze, odnosno od onih koji su još u fazi dizajniranja. Prilikom aktiviranja analizatora grešaka, objekti čiji je *CREATE FLAG* jednak *False* se ne razmatraju.

BPwin (Bussines Process Windows)

BPwin alat za modeliranje i analizu složenih poslovnih procesa je zasnovan na konceptima standarda IDEF0. Zbog lakše identifikacije procesa i njihove optimizacije, kao i odbacivanja suvišnih procesa, nastala je metodologija modeliranja poslovnih procesa. Modeliranjem procesa se dobija uređena struktura sa jasno definisanim pravilima koji se procesi odvijaju i na koji način.

BPwin omogućava rigorozno testiranje konzistentnosti dijagrama. Dinamički objekti koji vode kroz model sprečavaju greške koje najčešće nastaju pri kreiranju modela.

BPwin podržava ABC (Activity Based Costing) sistem i ima interfejs ka ABC alatima namenjenim kompanijama koje menađžment strategiju baziraju na aktivnostima.

Kreiranje dokumentacije iz BPwin-ovih modela procesa je izvedeno na isti način kao i u svim alatima familije LogicWorks-a. Postoji potpuna sloboda u definisanju seta podataka iz modela aktivnosti, kao i u definisanju izgleda dokumentacije.

BPwin omogućava rad u grafičkom okruženju, uz punu podršku rada mišem, tako da je editovanje modela podataka izuzetno lako; dijagrami koji se dobijaju daju potpuni prikaz modela. Dekompozicija procesa se vrši direktno na dijagramu, a kretanje po nivoima dekompozicije podrazumeva prebacivanje sa dijagrama na dijagram. Pri radu na jednom modelu otvoreni su svi dijagrami iz tog modela.

ERwin

ERwin alat za modeliranje podataka je izgrađen na konceptima koji su postavljeni IDEF1X standardom.

ERwin/*ERX* je *CASE* alat namenjen modeliranju podataka *ER* (*Entity Relationship*) metodom, koja je podržana *IDEF1X* (*Integration DEFinition for information modeling*) metoda, a korisnik može opciono koristiti i *IE* (*Information Engineering*) metodu. Modeliranje podataka obuhvata dva aspekta: logičko definisanje modela i fizički dizajn baze. Korišćenje ovog alata podrazumeva istovremeni razvoj logičkog modela podataka i fizičko definisanje baze, što znači da nije potrebno nikakvo dodatno prebacivanje modela iz logičkog nivoa u fizički i obrnuto.

U bilo kom trenutku definisanja logičkog nivoa baze, pre nego što se počne sa fizičkom definicijom baze podataka, potrebno je odabrati ciljni server. To je *DBMS* na kome će model biti realizovan. Izbor ciljnog servera nije definitivna stvar, što znači da se ciljni server može menjati. *ERwin* insistira na nezavisnosti modela od *DBMS* platforme.

Server FRE (*Forward and Reverse Engineering*) omogućava živu vezu *ERwin*-ovog modela i baze podataka. Konekcija se vrši preko *ODBC drivera*. Na osnovu definisanog fizičkog modela podataka može se kreirati baza na odabranoj *DBMS* platformi. *ERwin* poseduje i modul za inverzni inženjering, koji čita sistemske kataloge postojeće baze podataka, pa na osnovu njih automatski kreira *ER* dijagram po metodologiji *IDEF1X* standarda.

Oba smera se svode na sinhronizaciju modela podataka sa bazom podataka i poređenje modela sa stanjem baze. Osim sa bazom podataka, *ERwin* može da poredi model sa *SQL* šemom ili drugim *ERwin* modelom. Model može selektivno da se sinhronizuje sa promenama u bazi podataka u oba smera. Omogućena je postepena nadogradnja baze podataka iz *ERwin*-ovog modela podataka, uz kontrolu fizičkih parametara baze podataka. Pri svakoj sinhronizaciji sa bazom, *ERwin* pruža detaljan izveštaj o svim promenama nastalim sinhronizacijom. Isti model podataka se može koristiti za generisanje više baza podataka na različitim *DBMS* platformama, ili za konvertovanje aplikacija sa jedne *DBMS* platforme na drugu.

ERwin/*ModelMart* (ili *ERwin*/*AOS*) proširen je na *ERwin*/*ERX* koji se standardno nalazi od *ERwin ver*, 2.6. i predstavlja prvi alat koji omogućava višekorisnički rad na modelima podataka (*Workgroup*). U tom slučaju je praćenje razvoja modela podataka znatno komplikovanije. Korisnici svoje promene na modelu pohranjuju u *SQL* bazu podataka, smeštenu na *Microsoft SQL*, *Sybase* ili *ORACLE* server, gde se vodi računa o zaključavanju modela, verzijama, ovlašćenjima korisnika i sl.

ERwin je razvio set specijalizovanih verzija, kao što su: *ERwin for Visual Basic*, *ERwin for Power Builder*, *ERwin for ORACLE*, *ERwin for SYBASE* i sl. U kreiranju specijalizovanih verzija ide se za tim da se pojača sprega sa proizvodom za koji je verzija specijalizovana. Tako, *ERwin for ORACLE* ima detaljniji rad sa *ORACLE* bazom, kao i interfejs ka *ORACLE CASE*-u. Rad sa ostalim *DBMS* platformama je na nivou *ERX* verzije. Verzije za *Power Builder* i *Visual Basic* imaju i proširenje koje gradi klijent-stranu aplikacije na navedenom alatu, bez obzira na *DBMS* u koji se baza pohranjuje.

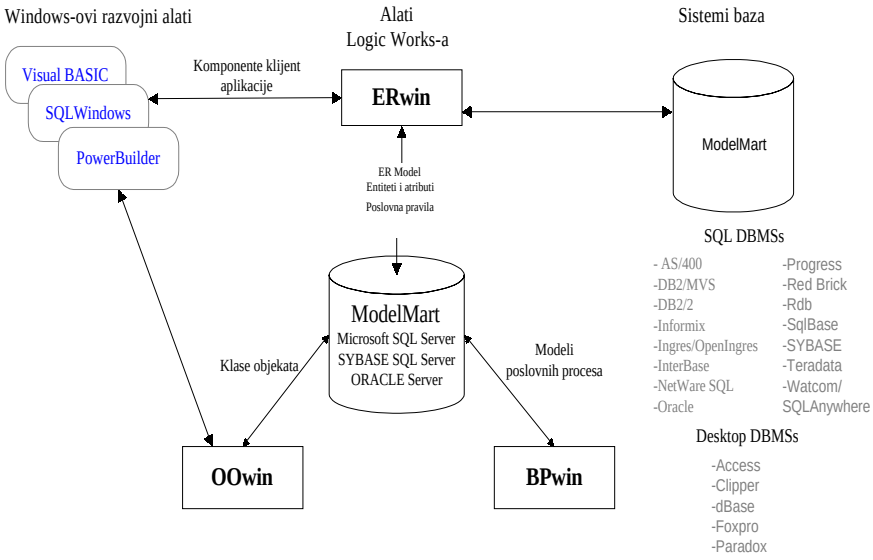
Verzija 2.6. *ERwin*-a obuhvata proširenje na *ERwin*/*OPEN* i predstavlja generalizaciju specijalizovanih verzija *ERwin*-a koje tretiraju klijent-stranu aplikacije. Osim izbora servera, korisnik bira i stranu klijenta. Ponuđeni su mu *Power Builder* i *Visual Basic*. U zavisnosti od izbora klijenta, korisnik na modelu može definisati koncepte vezane za klijent-stranu aplikacije koja se oslanja na bazu podataka koju modelira. Izbor *Target Client*-a ne zavisi od izbora *Target Servera*. *Logic works* najavljuje i još neka proširenja kada je u pitanju klijent-strana, npr. interfejs ka *Delphi*-ju.

ERwin/*Navigator* je jeftin alat koji pruža korisniku *read-only* pristup modelima podataka razvijenim u *Modelmart* okruženju. Ovi dijagrami se mogu pretraživati i štampati i može se iz njih kreirati dokumentacija. Uz ista ograničenja je podržana i klijent-strana razvoja

aplikacije. Promene napravljene na modelu se ne mogu sačuvati.

DataBOT je proširenje za *Visual Basic* koje omogućava visok nivo rada sa *Visual Basic*-om. *DataBOT* je alat koji ne zavisi od *ERwin*-a, ali se može nadograditi na *ERwin* i pojačati spregu *ERwin*-a i *Visual Basic*-a, bez obzira na to da li je u pitanju *ERwin*/*ERX*, *ERwin for Visual Basic* ili *ERwin*/*Navigator*.

Da bi pokrio proces testiranja, *Logic Works* izbacuje novi proizvod *TESTBytes* koji omogućava brzo i lako kreiranje test-podataka, koji su refleksija stvarnih podataka. Na taj način je uveliko skraćeno vreme testiranja.

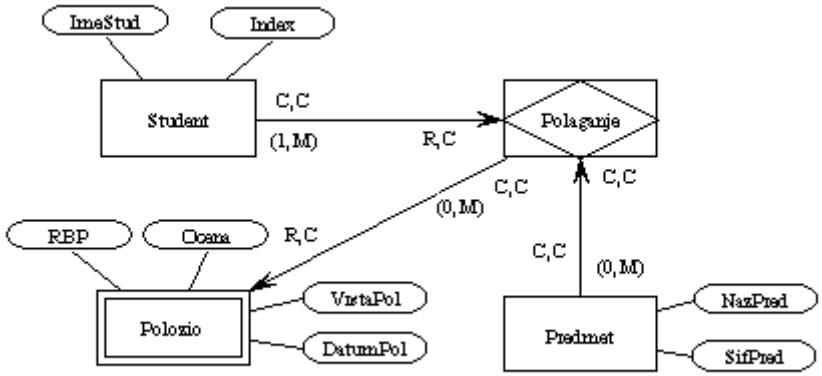


Slika B.5. Alati Logic Works-a

Artist

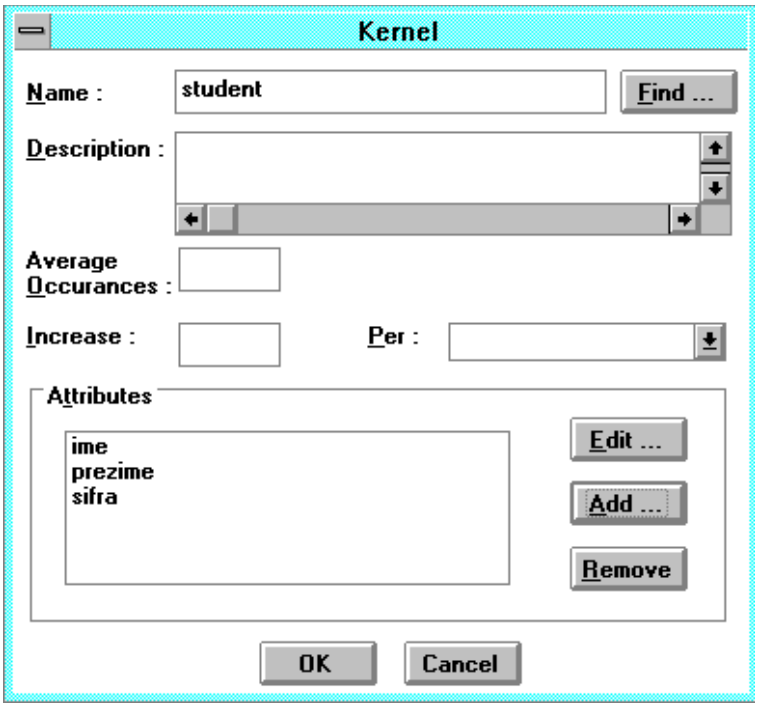
Artist CASE alat je razvijan da bi se pokrio celokupan razvoj informacionog sistema. Alat omogućava automatizaciju prve faze u razvoju informacionih sistema, odnosno planiranje razvoja informacionih sistema, koje se zasniva na *BSP* (*Business System Planning*) metodi. Pored toga, poseduje i model koji podržava modelovanje procesa (*SSA*).

Modul koji omogućava modelovanje podataka podržava semantički veoma bogat model, tako da se može modelirati neki skup podataka na veoma efektan i brz način. *Artist* podržava sledeće koncepte: jak entitet, slab entitet, agregaciju, specijalizaciju i generalizaciju (podtipove i nadtipove), vezu, identifikujuću vezu. Deo koncepata prikazan je na primeru (slika B.6).



Slika B.6. Artist – primer modela "objekti i veze" u grafičkom editoru

Izgled atributa i opis entiteta kroz formu dati su, takođe, na slici B.6. Nema komandi za dodavanje, brisanje i ispravku atributa na samoj formi za opis entiteta, već se to realizuje na formi koja se aktivira komandom *Attributes*, nakon čega se pojavljuje forma za unos.

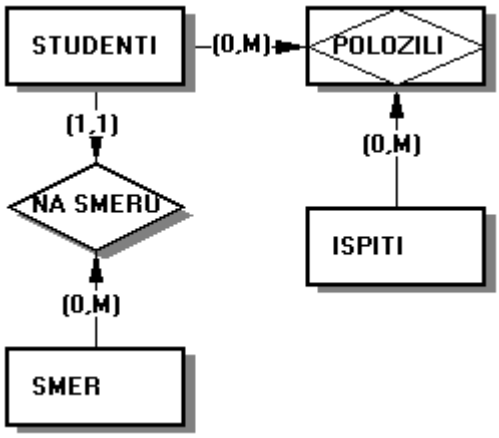


Slika B.7. Artist CASE – *ekranska forma za unos podataka o entitetu*

MagiCASE

MagiCASE je grafički orijentisan alat za modelovanje podataka. Zasnovan je na sledećim konceptima: atribut, jak entitet, slab entitet, agregacija, specijalizacija i generalizacija (podtipovi i nadtipovi), veza, identifikujuća veza.

U samom alatu može se menjati grafička prezentacija koncepata, čime je korisnik donekle oslobođen krutih stega metodologije, što predstavlja veliki podstrek u kreativnosti korisnika programa.

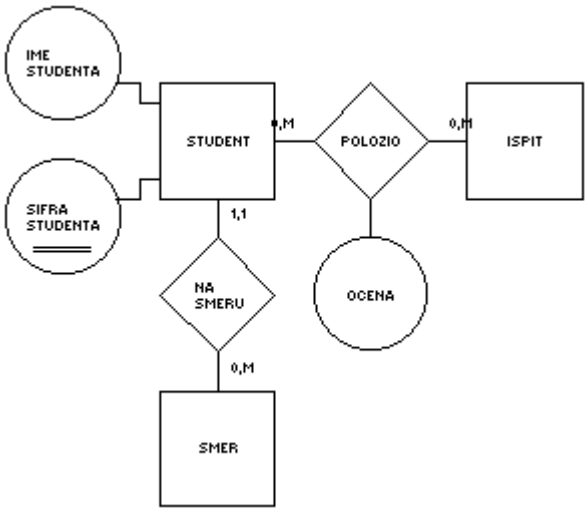


Slika B.8. MagiCASE –primer modela "objekti i veze" u grafičkom editoru

Kod MagiCASE-a u rečnik podataka uveden je i pojam domena, čime je omogućeno da se atributi, čiji su tipovi isti, mogu definisati preko prethodno kreiranih domena.

EasyCASE System Designer

EasyCASE System Designer je grafički orijentisan alat za razvoj sistema koji radi samo u *Microsoft Windows* okruženju. Pomoću skupa modula mogu se modelirati procesi po metodologijama: *Yourdan*/*DeMarco*, *Gane* & *Sarson*, *SSADM,* a, isto tako, i modeliranje podataka po metodologijama: *Chen*, *Martin Bachman*, *IDEF1X*, *Shlaer* & *Mellor*.

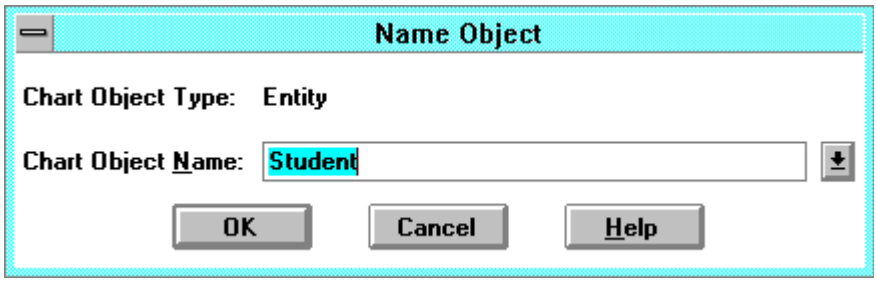


Slika B.9. EasyCASE – *primer modela "objekti i veze" u grafičkom editoru*

EasyCASE System Designer podržava sledeće koncepte: atributi (za koje se mora naglasiti da li je neki od njih običan atribut), primarni ključ, spoljni ključ, viševrednosni ili izvedeni atribut, jak entitet, slab entitet.

Deo koncepata dat je primerom na gornjoj slici.

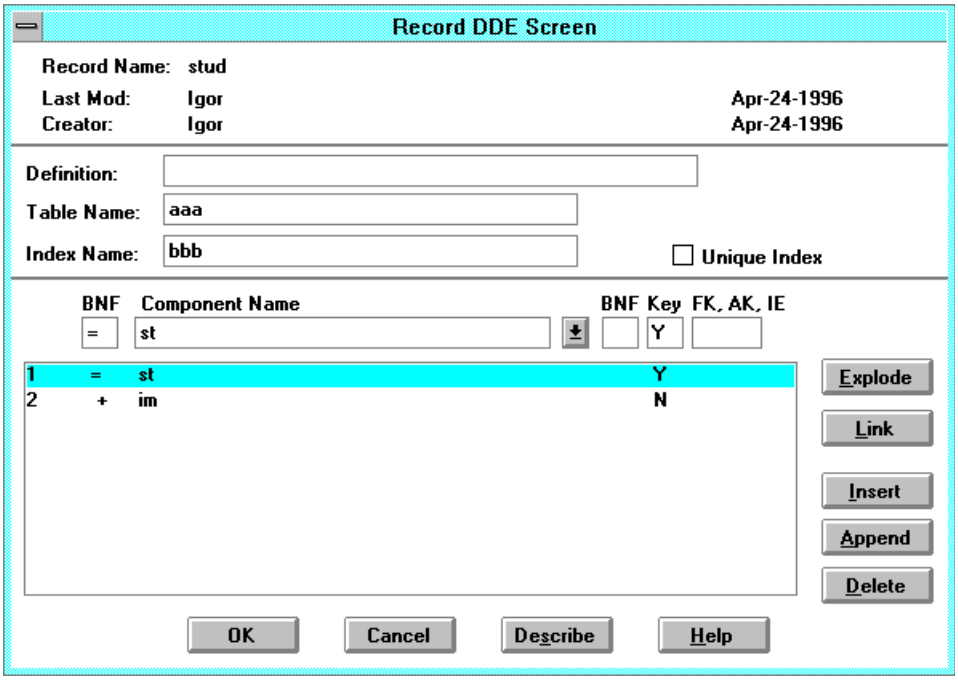
Na sledećoj slici data je forma za unos entiteta.



Slika B.10. EasyCASE – *ekranska forma za definisanje entiteta*

Atributi se u podmodelu moraju definisati odvojeno kao samostalan objekat, nezavisno od entiteta, nakon čega se atribut može povezati sa entitetom kome pripada. Definisanje spoljnog ključa kao tipa atributa predstavlja redundantni podatak, imajući u vidu da je pojavljivanje spoljnih ključeva definisano uspostavljenim vezama.

Drugi način kreiranja atributa je preko definisanja *child object*-a. Aktiviranjem određenog koncepta i definisanjem njegovog *child object*-a, kao *record*-a, ponuđeno je da se definišu i njegove komponente, odnosno atributi koncepta.



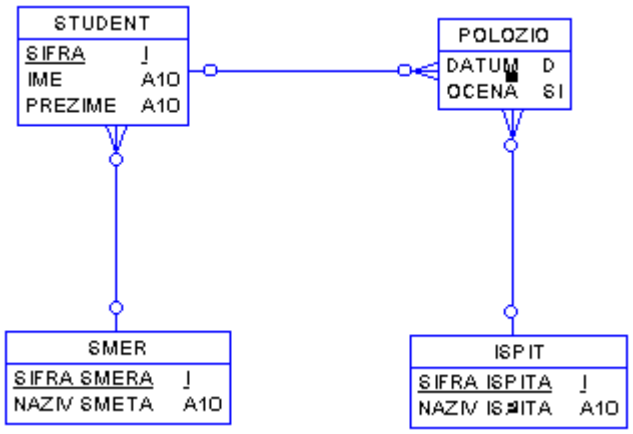
Slika B.11. EasyCASE – *ekranska forma za definisanje atributa*

Podržava generisanje šeme baze podataka za sledeće server baze: *ORACLE7*, *DB2*, *Rdb*, *Ingres*, *SQLServer*, *SQLBase*, *SYBASE*, *Progress*, kao i sisteme za upravljanje bazama podataka na personalnim računarima: *Clipper*, *FoxPro*, *dBASE*, *Paradox*.

S-Designor

S-Designor je alat za modeliranje podataka, grafički orijentisan, koji radi u okruženju *Microsoft Windows*.

Osnovni elementi na kojima počiva su: atributi, entiteti, veze. Deo koncepata koje podržava *S-Designor* dat je na slici B.12.



Slika B.12. S-Designor – *primer modela "objekti i veze" u grafičkom editoru*

Iako prividno siromašan (poseduje samo jedan koncept od objekata), mogu se iskazati sve vrste koncepata koje se pojavljuju u modelu objekti-veze. Prilikom definisanja entiteta i njegovih osobina grafičke prezentacije, rad sa njim je identičan, nezavisno od toga koji se koncept predstavlja. Naziv, atributi i opisi unose se na identičan način kod svih objekata, a razlike između njih postaju očigledne tek kada se opredeli za vrste veza kojima se određeni objekat povezuje sa ostatkom podmodela.

ADW (Application Development Workbench)

ADW je *KnowledgeWare*-ov *CASE* alat sa integrisanim delovima koji omogućava automatizaciju životnog ciklusa razvoja informacionog sistema.

Prvi deo *Planing* Workstation podržava definisanje objekata, kao i njihovo povezivanje preko asocijativnih matrica. Mogu se definisati organizaciona struktura i informacione potrebe, ciljevi, poslovne funkcije, kritični činioci uspeha. Postoji mogućnost utvrđivanja prioriteta projekata na osnovu postavljenih kritičnih činilaca. Sve informacije se smeštaju u centralnu enciklopediju kojoj mogu da pristupe svi delovi alata.

Drugi deo *Analisis Workstation,* koji služi za modelovanje procesa i podataka, sadrži čitav niz modula za crtanje dijagrama: dijagram sastavnih delova, dijagram toka podataka, *ER* dijagram, dijagram tipova informacija, kao i akcioni dijagram.

Deo *RAD Workstation* je alat za izradu prototipova i omogućava ispitivanje početnog dizajna aplikacija za krajnjeg korisnika.

Delovi *Design Workstation* i *Construction Workstation*, na osnovu prethodnih informacija koje su smeštene u enciklopediji, mogu izgenerisati generičke klijent/server-aplikacije relacionih i hijerarhijskih baza podataka (*Oracle7*, *Sybase SQL Server*, *DB2*, *DB2\2*, *IMS*, *VSAM)* i relacione baze koje podržavaju *ANSI* standard. Generatori na osnovu generičke aplikacije izrađuju klijent/server-aplikaciju za ciljni operativni sistem (*UNIX.Motif, OS/2 Presentation Manager, DOS/Windows3.x, AS/400 i MVS).*

Platinum

Platinum je integrisan skup alata sa mogućnošću da podrži proces razvoja informacionog sistema, prateći životni ciklus procesa. Pojedini delovi, odnosno moduli razvijeni su tako da podrže pojedine faze razvoja, a da, opet, čine jednu čvrstu celinu.

Platinum *Proces Continuum* pokriva aktivnosti upravljanja projektom razvoja informacionog sistema, što uključuje izbor metodologija, planiranje i praćenje projekta, kao i praćenje troškova uz mogućnost izveštavanja iz svih aktivnosti.

Platinum *Paradigm Plus* je esencijalni deo koji podržava fazu analize i dizajna. Alat je izgrađen tako da potpuno podrži objektnu analizu i objektno projektovanje. To je grafički orijentisan alat sa mogućnošću rada na raznim platformama i operativnim sistemima. Podržava identifikaciju poslovnih zahteva, izgradnju modela i komponenti aplikacije.

Faza konstrukcije podržana je preko sledećih alata:

- Platinum AionDS
- Platinum ObjectPro
- Platinum RuleServer
- Platinum SQL-Station Tool Suite

Veoma važna faza testiranja pokrivena je modulima:

- Platinum Final Exam Tool Suite
- Platinum Final Exam Internet Test

Slika B.13. Životni ciklus razvoja softverskog proizvoda

Očekivani pravci razvoja CASE alata

- *CASE* tehnologija će početi da isporučuje potpuno integrisani ambijent sistema ili skup komponenti koje su integrisane sa ostalim razvijanim softverskim proizvodima.
- Centralna pozicija tehnologije repozitorijuma ostaće potcenjena i uglavnom neprepoznatljiva van mesta koji su već svesni njenog značaja.
- Objektno oirjentisane metode jednostavno će zauzeti mesto postojećih metoda, jer su efikasnije; međutim, to će biti lagan proces.
- *CASE* tehnologija će zameniti *4GL* tehnologiju.

Konačno, očekuje se čvrsta veza izmeću *CASE* alata i proizvoda za razvoj desktop-aplikacija, kao što su: *PowerBuilder, MS SQL SERVER, MS ACCESS, Visual Basic* i drugi.

Literatura

1. dr Alempije Veljović, Integracija zahteva sistema kvaliteta u poslovanju preduzeća, Savezi inženjera i tehničara Jugoslavije, Beograd, 1996. godina
2. dr Branislav Lazarević: "Projektovanje informacionih sistema", interni materijal, FON, Beograd, 1990.
3. IT E 04-2-1, Sistemska analiza, podsetnik Intertrade, TOZD zastupništvo IBM, Izobrazevalni centar, Ljubljana, Ver.1.0, 1984.
4. IT FA 40-2-1, Planiranje informacionih sistema, podsetnik Intertrade, TOZD zastupnistvo IBM, Izobrazevalni centar, Ljubljana, Ver.1.0, 1984.
5. dr Vladimir R. Milačić, Sistem Analiza, Proizvodni informacioni sistem, Institut Mašinskog fakulteta, Odeljenje za primenu kompjutera, Beograd, 1974.
6. User Guide for BPwin.
7. User Guide for ERwin.
8. Veljović A. i dr., Projekat revizije po standardu ISO 9000:2000 i veza sa informacionim sistemom, Sojaprotein, Bečej, 2000.godina
9. Veljović A. i dr., Idejni projekat informacionog sistema programa A-85, interni materijal, 1989.
10. Veljović A. i dr., Detaljni projekat informacionog sistema programa A-85, interni materijal, 1990.
11. ERwin, Methods Guide, 1995.
12. Veljović A. Razvoj informacionog sistema kvaliteta, materijal za istoimeni seminar, Savez inžinjera i tehničara Jugoslavije, Beograd.
13. Veljović A. Kako definisati informacije potrebne računaru vezane za postavljanje sistema kvaliteta serije standarda JUS ISO 9000. Seminar, Jugoslovenska organizacija za standardizaciju i kvalitet (JUSK), Beograd, 20 - 21. decembar 1994. godine.
14. Veljović A. Gotova rešenja upravljanja kvalitetom za JUS ISO 9004 orjentisanih računarskoj obradi podataka. Seminar, Jugoslovenska organizacija za standardizaciju i kvalitet (JUSK), Beograd, 14 - 15. mart 1995. godine.
15. Veljović A. Naučite da kreirate informacioni sistem za upravljanje kvalitetom, Informativno instruktivni seminar, Savez inžinjera i tehničara Jugoslavije, Beograd, 22 - 23. juna 1995. godine.
16. Veljović A. Elementi osiguranja kvaliteta u ambijentu osvajanja novog proizvoda korišćenjem CASE alata, 21 JUPITER konferencija sa međunarodnim učešćem, 1. simpozijim KVALITET, strana 5.97-5.103., Beograd, februar 1995.
17. Veljović A., Dekomponovanje procesa petlje kvaliteta, Časopis za unapređenje kvaliteta Kvalitet, broj 5-6, strana 31-33, Poslovna politika, Beograd, 1995.