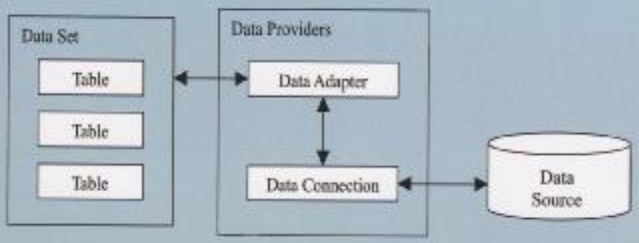
Univerzitet u Novom Sadu
Tehnički fakultet „Mihajlo Pupin"
Zrenjanin

Dr Biljana Radulović
Mr Ljubica Kazi
Mr Zoltan Kazi

INFORMACIONI SISTEMI



Odabrana poglavlja

UNIVERZITET U NOVOM SADU
TEHNIČKI FAKULTET »MIHAJLO PUPIN«
ZRENJANIN

Prof. dr Biljana Radulović
Mr Ljubica Kazi
Mr Zoltan Kazi

INFORMACIONI SISTEMI

- ODABRANA POGLAVLJA

(drugo ponovljeno izdanje)

Zrenjanin, decembar 2010.

Autori:
Prof. dr Biljana Radulović
Mr Ljubica Kazi
Mr Zoltan Kazi

Recenzenti:
prof. dr Dragica Radosav
doc. dr Branko Markoski

Izdavač:
Tehnički fakultet «Mihajlo Pupin», Zrenjanin

Za izdavača:
prof. dr Milan Pavlović, dekan

Nastavno-naučno veće
Tehničkog fakulteta «Mihajlo Pupin» u Zrenjaninu donelo je odluku 15.12.2010. godine da se rukopis ove knjige može koristiti kao udžbenik Fakulteta.

Tehnička obrada:
Ljubica Kazi
Zoltan Kazi

Štampa:
Grafopanonija D.O.O. Zrenjanin

Tiraž:
200 komada

CIP - Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

004 (075.8)
РАДУЛОВИЋ, Биљана
Informacioni sistemi : odabrana poglavlja / Biljana Radulović, Ljubica Kazi, Zoltan Kazi. - 2., ponovljeno izd. - Zrenjanin : Tehnički fakultet "Mihajlo Pupin", 2011 (Zrenjanin : Grafopanonija). II, 219 str.; 24cm
Tiraž 200. - Bibliografija

ISBN 978-86-7672-149-8
1. Кази, Љубица 2. Кази, Золтан
а) Информациони системи
COBISS.SR-ID 267513095

PREDGOVOR PRVOM IZDANJU

Ovaj materijal je namenjen slušaocima kurseva Projektovanje informacionih sistema, Informaciono-upravljački sistemi, Informacioni sistemi i Informacioni sistemi u obrazovanju kao delimični udžbenik. Nastao je kao rezultat višegodišnjeg rada autora u nastavi pomenutih kurseva. Kroz uputstva za izradu seminarskog rada studenata (konkretnog softverskog proizvoda) uz upotrebu CASE alata Power Designer, pokrivene su sledeće oblasti:

- *definicije osnovnih pojmova u oblasti projektovanja informacionih sistema,*
- *upravljanje razvojem softverskog proizvoda,*
- *definisanje metodologije projektovanja informacionih sistema i životnog ciklusa razvoja softvera,*
- *metoda funkcionalne dekompozicije – Strukturne Sistem Analize,*
- *projektovanja relacione baze podataka,*
- *objektno orijentisanog pristupa razvoju softvera primenom UML dijagrama,*
- *uputstva i primeri za programersku realizaciju.*

Namera autora je da u narednim izdanjima ovaj priručnik dobije izgled zaokružene celine – metodologije projektovanja informacionih sistema.

Zahvalnost za kritički pregled dugujemo recenzentima.

U Zrenjaninu, februar 2006.

PREDGOVOR DRUGOM IZDANJU

Drugo ponovljeno izdanje predstavlja reprint udžbenika iz 2006. godine. Namenjen je slušaocima kurseva Informacioni sistemi 1 i Informacioni sistemi 2 kao delimični udžbenik. Obuhvata odabrana teorijska poglavlja u oblasti projektovanja i razvoja informacionih sistema, uz ilustracije i primere korišćenjem razvojnih alata.

U Zrenjaninu, decembar 2010.

SADRŽAJ:

1. OSNOVI INFORMACIONIH SISTEMA

Teorijske osnove informacionih sistema i njegovog razvoja čini više zasebnih naučnih disciplina, među kojima su teorija sistema i informatika. Složenost informacionih sistema, njegova organska pripadnost organizacionom sistemu i zavisnost od ljudi kao korisnika i onih koji ga razvijaju zahteva prisustvo saznanja iz oblasti psihologije, organizacionih nauka, operacionih istraživanja i drugih.

Bez pretenzija ka detaljnijem prikazu rezultata navedenih oblasti, u nastavku će biti dat kratak pregled osnovnih pojmova, koncepata i pristupa, a u narednim poglavljima akcenat je dat na razvoju softverske podrške u informacionim sistemima.

1.1. OSNOVNI POJMOVI

INFORMATIKA

- nauka o informacionim sistemima koja se bavi izučavanjem zakonitosti razvoja i funkcionisanja informacionih sistema;
- vrsta delatnosti koja pokriva radne zadatke u praktičnih realizaciji i funkcionisanju informacionih sistema.

INFORMACIONI SISTEM (IS) - model realnog organizacionog sistema u kojem deluje. Sistem u kojem su elementi međusobno i sistem kao celina sa okruženjem povezani informacionim tokovima.

SISTEM – skup povezanih elemenata koji predstavljaju jasno izdvojenu celinu sa određenim ponašanjem, gde je funkcionisanje svakog elementa podređeno ostvarenju zajedničkih ciljeva. Ključne karakteristike sistema su: skup elemenata, postoje relacije između elemenata i interakcije sa okruženjem, hijerarhijska uređenost elemenata u strukturi, uređenost, organizovanost sa jasnom ulogom elemenata u doprinosu ciljevima celine, atributi kao karakteristike elemenata, strukture, funkcionisanja i sistema kao celine, karakteristike sistema kao celine (statičke i dinamičke), izdvojena celina od okruženja, funkcionisanje (pravila, funkcija, zakon), ponašanje – niz stanja, procesi, ulaz/izlaz, cilj.

ORGANIZACIONI SISTEM - sistemi čije osnovne elemente čine sociološki (ljudi povezani sociološkim vezama) i tehničko-tehnološki podsistemi (sistemi čiji su sastavni delovi i veze materijalno-fizičkog karaktera).

STRUKTURA SISTEMA - broj i karakter elemenata sistema, broj i karakter veza između elemenata sistema. Uloga veza u sistemima je omogućavanje prenošenja rezultata rada pojedinih elemenata sistema.

FUNKCIONISANJE SISTEMA - skup aktivnosti koje izvršavaju elementi sistema u određenom vremenu, a usmerene su ka realizaciji svrhe i ciljeva sistema. Produkt

funkcionisanja sistema predstavlja njegovu funkciju. Osnovne funkcionalne komponente organizacionih sistema nose informaciona, energetska i materijalna obeležja. U funkcionisanju organizacioni sistemi razmenom materije, energije i informacija ostvaruju ciljeve i svrhu postojanja putem internih strukturnih veza ili veza sa okruženjem.

PROCES – Proces se sastoji od aktivnosti rada sistema u toku vremena, produkata tog rada i izvršilaca (ljudska i materijalno-tehnička komponenta) rada. U organizacionim sistemima ne postoji ni jedan proces koji ne sadrži informacionu komponentu. Postoje i procesi koji imaju gotovo isključivo informacioni karakter. Razlikujemo:

- osnovne procese - ostvaruju svrhu postojanja sistema;
- procese održavanja - uklanjanje smetnji i snabdevanje resursima;
- razvojne procese - menjaju strukturu i funkcionisanje sistema radi obezbeđivanja prilagođavanja promenama u okruženju i uspešnijem ostvarivanju ciljeva u okviru osnovnih procesa;
- upravljački procesi - postavljanje ciljeva, usmeravanje, nadgledanje, odlučivanje, vrednovanje ostvarivanja ciljeva.

MODEL - materijalni ili idejni sistem koji je usled aktivnosti i delovanja subjekta modelovanja u nekoj relaciji (sličnost po spoljašnjim osobinama, strukturi, ponašanju, funkcijama) sa nekim originalnim sistemom i zamenjuje ga u ispitivanjima, u saopštavanju znanja ili u rešavanju problema. To je pojednostavljena predstava o relevantnim karakteristikama sistema, nastala analizom, apstrakcijom i sintezom.

SISTEMSKI MODEL - model sistema opisan putem sledećih karakteristika: skup aspekata posmatranja (kao kriterijum modelovanja), obeležja celine, okruženja i njihove interakcije, dekompozicija sistema na podsisteme i za svaki podsistem prikaz:

- interes (misija, svrha, ciljevi);
- struktura (podsistemi, elementi, veze);
- ponašanje (funkcije, procesi, algoritmi, operatori).

PODATAK - misaoni oblik i zapis o činjenici (postojanje određene stvarnosti) o nekoj pojavi, aktivnosti, događaju ili objektu - o vrsti atributa i vrednosti atributa koji opisuje te činjenice.

INFORMACIJA - bilo koja vrsta poruka koje se koriste za donošenje odluka ili izvršenje aktivnosti na ostvarenju donetih odluka. Sastoje se od podataka ili skupova podataka uređenih tako da kao celina predstavljaju iskaze sa određnim logičkim smislom. Informacije se dobijaju obradom podataka, ali i obradom informacija dobijenih prethodnim obradama.

Informacije imaju sledeće karakteristike:

- sintaksna svojstva informacije - svaki podatak ili informacija mogu se predstaviti određenim kombinacijama simbola iz nekog kodnog skupa, a pomoću posebnih pravila sintakse;
- semantičko svojstvo informacija - svaka informacija ima svoj smisao orjentisan prema korišćenju;
- kvantitativno svojstvo informacija - svaka informacija ima svoj opseg, količinu;

- pragmatičko svojstvo informacija - svaka informacija ima svoje korisnike;
- ekonomsko svojstvo informacija - svaka informacija ima svoju cenu koja se za nju plaća i daje određene ekonomske efekte kada se koristi.

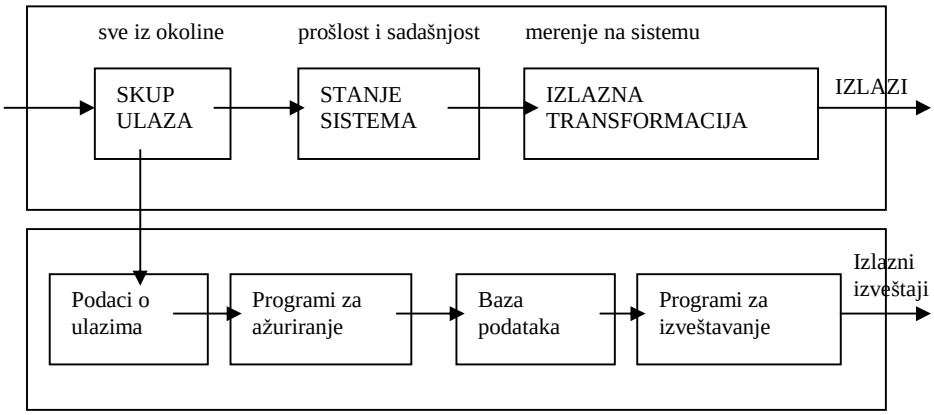
Neke od uloga informacija u organizacionim sistemima:

- daju prikaz, informacionu sliku realnih subjekata i objekata organizacionih sistema i njihovih okruženja, odraz njihovog postojanja i karakteristika stanja;
- mera eliminisane neodređenosti nekog procesa;
- mera raznovrsnosti kod izbora jednog od mogućih alternativnih rešenja;
- mera eliminisane nesređenosti tokom upravljanja funkcionisanjem nekog (nesređenog) procesa.

1.2. STRUKTURA INFORMACIONOG SISTEMA

Informacioni sistem se razvija za realni sistem, te je struktura realnog sistema osnova za modeliranje strukture IS. Pod realnim sistemom se podrazumeva stvarni dinamički sistem koji postoji u objektivnoj realnosti, odnosno to je i organizacioni sistem.

Struktura informacionog sistema (odnosno arhitektura IS), njegovi delovi i međusobne veze, kao i odnos prema realnom sistemu [LAZAREVIĆ i sar., 1988], bez obzira na to kojim softverom je izgrađen, prikazana je na sledećoj šemi:



Slika 1.1. Struktura informacionog sistema

U realan sistem ulaze materija, energija i podaci, kao i podaci o ulaznoj materiji i energiji. Stanje sistema opisuje prošlost i sadašnjost sistema, koje pod uticajem ulaza i procesa transformacije dovodi do promene stanja sistema i generisanja izlaza.

U informacionom sistemu ulaz predstavljaju podaci, kao i podaci o ulazima u realan sistem, kao njihova "slika". Ovi podaci se pomoću programa za ažuriranje unose u bazu podataka koja u svakom trenutku sadrži opis stanja realnog sistema. Prikaz stanja realog sistema izraženo podacima u bazi podataka vrši se pomoću programa za izveštavanje.

Stanje realnog sistema se menja zbog delovanja niza procesa (funkcija) u sistemu. Ove procese opisujemo MODELOM PROCESA u informacionom sistemu. Realizovan na računaru, model procesa je skup programa za ažuriranje. Stanje objekata realnog sistema u informacionom sistemu predstavljamo podacima o tim objektima gradeći MODEL PODATAKA. Baza podataka je fizička realizacija modela podataka.

MODEL RESURSA - obuhvata tehničke i kadrovske resurse organizacije koji su uključeni u funkcionisanje informacionog sistema.

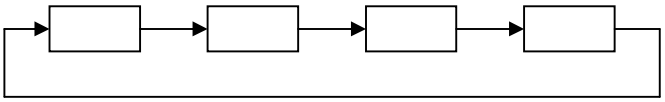
Projektovanje informacionog sistema je nalaženje relevantnog modela realnog sistema. Projektujući tri modela: model procesa, model podataka i model resursa gradi se model informacionog sistema.

1.3. PRISTUPI RAZVOJU INFORMACIONOG SISTEMA

Postoje različiti pristupi razvoju informacionih sistema, odnosno definisanju faza životnog ciklusa razvoja.

I GRUPA PRISTUPA RAZVOJU IS

1. LINEARAN



Slika 1.2. Linearni pristup razvoju informacionog sistema

Karakteristike:

- Sukcesivan sled faza u razvoju IS
- Nema povratka na prethodne faze
- Vrednovanje se dešava na kraju razvoja
- Korisnik na početku ne zna mogućnosti novog IS

Linearnim (konvencionalnim) modelom životnog ciklusa se razvoj informacionog sistema deli na faze:

1. planiranje razvoja (najčešće korišćena BSP - Business System Planning metoda kompanije IBM)
2. analiza zahteva
3. specifikacija informacionog sistema
4. projektovanje baze podataka
5. projektovanje programa
6. implementacija (kodiranje, testiranje)
7. uvodenje sistema u rad
8. održavanje sistema

i na kraju prestanak upotrebe. Ova metodologija razvoja informacionog sistema je došla u krizu zbog haosa koji se stvara prilikom dvostrukog održavanja sistema (paralelno

održavanje sistema i dokumentacije), kao i zbog predugog ciklusa razvoja informacionih sistema. Njen najveći nedostatak je što je korisnik upoznat sa informacionim sistemom tek na kraju njegove realizacije (fizičke implementacije), te su moguća velika odstupanja kreiranog informacionog sistema u odnosu na realni sistem i očekivanja njihovih korisnika.

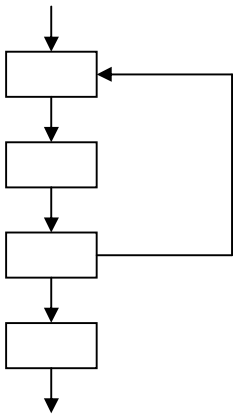
2. PROTOTIPSKI

Karakteristike:

- U ranoj fazi razvoja IS-a teži se razvoju prototipa (početne verzija koja se menja u skladu sa korisnikovim zahtevima)
- Postoji mogućnost povratka na prethodne faze u razvoju IS-a

Problemi:

- Uvođenje dokumantacije
- Integracija delova u jedinstven IS.



Slika 1.3. Prototipski pristup razvoju informacionog sistema

Faze kod protipskog razvoja IS:

1. Planiranje razvoja
2. Analiza i specifikacija zahteva
3. Izrada prototipa
4. Implementacija
5. Održavanje

Prototipski pristup razvoju IS predstavlja unapređenje u odnosu na konvencionalni pristup. Osnova ove metodologije je da se napravi što je moguće brže **prototip softvera**, da bi se na osnovu njega utvrdile potrebe korisnika. U odnosu na to da li se prototip dalje razvija i dopunjuje ili ne postoje *odbacivi* i *nadgradivi* prototipi. Odbacivi prototip služi samo za utvrđivanje potreba korisnika, odnosno za specificiranje zahteva. Kada se ove potrebe utvrde, prototip se odbacuje, a na osnovu njega se ponovo izrađuje ceo novi sistem.

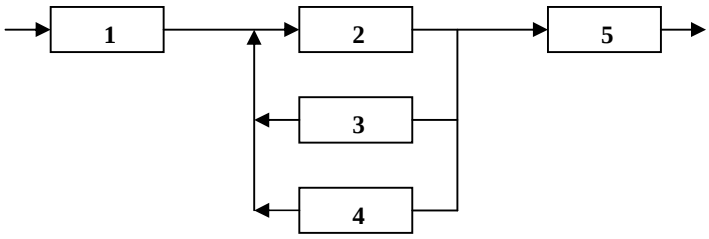
Ovakvi prototipi se obično rade koristeći jednostavne, ali ne i efikasne generatore aplikacija ili jezike četvrte generacije kojima je lako opisati sistem i brzo napraviti prototip koji približno radi ono što je zahtevano. Na osnovu odbacivog prototipa se vrši projektovanje novog sistema koristeći konkretne alate, klasične programske jezike ili neki od sistema za upravljanje bazom podataka. Za razliku od odbacivog prototipa nadgradivi prototipi se usavršavaju i koriguju i u nekoliko iteracija dovodeći do konačnog rešenja. Nedostatak prototipa predstavlja dokumentacija, jer se često dokumentacija ne piše ili je nepotpuna prilikom razvoja prototipa. Takode, prototip celog sistema se ne može napraviti, pa se zbog toga pravi više manjih prototipa (prototipi podsistema), koji mogu dovesti do haosa. Rešenje se nalazi u prvobitnom projektovanju zajedničke baze podataka za sve prototipe koja obezbeđuje koordinirani razvoj sistema. Ovakav način izrade prototipa je poznat kao Data Driven (vođen podacima) prototip.

Takođe, korisnici se aktivno uključuju u sam proces izrade prototipa, pa su i manja odstupanja od zahteva korisnika i potreba realnog sistema. Od korisnika se, u ovom slučaju, zahteva poznavanje metodologije projektovanja IS i osnova informaciono-komunikacionih tehnologija.

3. TRANSFORMACIONI

Karakteristike:

- Objedinjuje dobre osobine linearnog i prototipskog pristupa u razvoju IS-a.
- Zasniva se na postojanju alata za brzi razvoj IS-a.
- Specifikacija zahteva je na jeziku koji je blizak korisniku, a sam alat generiše programski kod.



Slika 1.4. Transformacioni pristup razvoju informacionog sistema

Faze su sledeće:

1. Analiza zahteva
2. Formalna specifikacija
3. Validacija specifikacije
4. Održavanje
5. Automatska optimizacija

Metoda koja predstavlja unapređenje prototipskog pristupa razvoju je operacioni ili transformacioni razvoj informacionog sistema. Suština ovog načina razvoja je u korišćenju alata koji na osnovu specifikacije problema automatski generišu softversko rešenje, tj. program u određenom programskom jeziku. Takvi alati se nazivaju **CASE alati (Computer Aided Software Engineering)**. Ceo proces se svodi na formulisanje analize zahteva u obliku u kojem se izrađuje formalna, izvršiva specifikacija, a na osnovu toga prototip programa. Validacija i održavanje modela se vrše nad formalnom specifikacijom, a radi dobijanja izvršnog koda programa, ponovno se pokreće proces automatske transformacije.

II GRUPA PRISTUPA RAZVOJU IS-a

Pristupi **TOP – DOWN** (od vrha ka dnu) i **BOTTOM-UP** (od dna ka vrhu) sastoje se u odluci odakle započeti razvoj informacionog sistema. Kao pravilan pristup koristi se top-down kod analize i projektovanja informacionog sistema (svrha postojanja sistema, cilj, definisanje celine i podsistema itd.), a bottom-up da se koristi kod implementacije i uvođenja IS deo po deo.

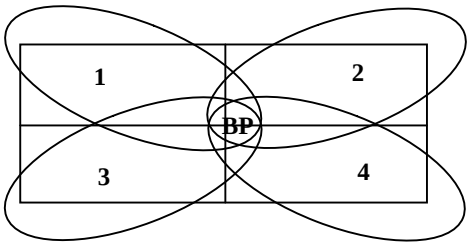
III GRUPA PRISTUPA RAZVOJU IS-a

Zasniva se na sledeće dve podele:

1. **STRUKTURNO** - projektovanje, dizajn i programiranje. Definišu se procesi i funkcije koje vrši sistem.
2. **OBJEKTNO** – atributi (osobine) i metode (procedure i funkcije). Pri razvoju IS-a treba definisati objekte u sistemu (elemente, veze između elemenata i sl.) i procedure i funkcije koje se dešavaju nad tim objektima, koje su veze između objekata. Realizacija funkcija i procedura se ostvaruje preko poziva odgovarajućih metoda.

IV GRUPA PRISTUPA RAZVOJU IS-a

1. **SISTEMSKI** (totalni) – svi podsistemi se istovremeno razvijaju i implementiraju. Zahteva izuzetno mnogo vremena i novca.
2. **PARCIJALNI** (ad hoc) – ne vodi računa o sistemskom pristupu jer se razvija i implementira samo jedna služba, sektor ili organizaciona jedinica. Komunikacija sa ostalim delovima IS-a i integracija sa globalnom bazom podataka se ostvaruje kad se ukaže potreba.
3. **MODULARNI** – udružuje oba ova pristupa. Definišu se MODEL PODATAKA i MODEL PROCESA na nivou cele organizacije.



Slika 1.5. Modularni pristup razvoju informacionog sistema

1.4. KLASIFIKACIJE INFORMACIONIH SISTEMA

Postoje različiti kriterijumi za klasifikaciju informacionih sistema. Navešćemo klasifikacije prema vrsti obrade, prema vrsti pruženih usluga i prema stepenu automatizacije [JAUKOVIĆ, 1992].

Informacioni sistemi se mogu klasifikovati prema vrsti obrade na one koji vrše obradu podataka:

- paketno;
- serijski;
- slučajno;
- interaktivno;
- u realnom vremenu.

Klasifikacija prema vrsti pruženih usluga:

- sistemi za računarske usluge opšte namene;
- sistemi za čuvanje i pretraživanje podataka;
- sistemi za komutaciju poruka;
- sistemi za upravljanje fizičkim procesima;
- sistemi za kontrolu i upozorenja;
- sistemi za obradu transakcija.

Prema stepenu automatizacije postoje:

- neautomatizovani informacioni sistemi;
- sistemi automatske obrade podataka;
- upravljački informacioni sistemi;
- izvršni informacioni sistemi;
- sistemi za podršku odlučivanju;
- ekspertni sistemi.

Osnove razvoja automatizovanih informacionih sistema:

- korišćenje zajedničkih računarskih sistema kao tehničke osnove,
- korišćenje zajedničkih podataka iz baze podataka,
- korišćenje jedinstvenog sistema obeležavanja kojim se povezuju podaci i ostali elementi informacionog sistema.

2. PROCES RAZVOJA INFORMACIONOG SISTEMA

2.1. RAZVOJ INFORMACIONOG SISTEMA

U razvoju informacionog sistema učestvuju mnogobrojne aktivnosti, koje se mogu grupisati u faze razvoja koje su opisane u prethodnom poglavlju. Prema [STANOJEVIC, 1986], aktivnosti projektovanja, uvođenja i funkcionisanja informacionih sistema su:

IZRADA STUDIJE OPRAVDANOSTI
- definisanje parametara za ocenu efekata od razvoja novog IS
- snimanje parametara za ocenu efekata
- procena troškova razvoja
- izrada studije opravdanosti

IZRADA PROGRAMA RAZVOJA I DEFINISANJA PROJEKTNIH ZADATAKA
- definisanje strukture organizacionog sistema
- razlaganje organizacionog sistema na podsisteme
- određivanje informacionih podsistema i redosleda razvoja
- priprema i izdavanje projektnih zadataka

SNIMANJE INFORMACIONIH PODSISTEMA, MODELIRANJE I ANALIZA
- snimanje tokova aktivnosti u organizacionim podsistemima
- snimanje tokova informacija u informacionim podsistemima
- izrada modela tokova informacija i njihova analiza

IZRADA IDEJNOG PROJEKTA INFORMACIONIH PODSISTEMA OBUHVAĆENIH PROJEKTNIM ZADATKOM
- definisanje logičke strukture skupova informacija
- definisanje rečima obrade informacija
- definisanje organizacije informacione osnove
- definisanje sistema orade
- definisanje organizacije i tehnologije procesa rada u informacionim podsistemima troškova realizacije
- određivanje ekonomske opravdanosti proizvodnje pojedinih informacija korisnika

IZBOR SISTEMA OBRADE
- izbor centralnog procesora i glavne memorije
- izbor periferne memorije
- izbor ulaznih i izlaznih uređaja i nosilaca
- izbor opreme za zahvatanje informacija i opreme za komuniciranje
- izbor sistemske programske podrške

IZRADA DETALJNOG PROJEKTA INFORMACIONIH PODSISTEMA

- formiranje i razrada dokumenata i drugih nosilaca izvornih informacija
- izrada fizičke organizacije informacione osnove
- razrada sistema prenosa informacija
- izrada ulazno-izlaznih postupaka
- definisanje programskih zahteva

PROGRAMIRANJE

- analiza programskih zahteva
- izrada programa, kompilacija i čišćenje grešaka
- testiranje programa
- formiranje programske biblioteke

PRIPREME ZA UVOĐENJE PROJEKTOVANIH SISTEMA

- obuka kadrova za rad na sistemu (tehn. oprema i programska podrška)
- obuka kadrovakorisnika za rad sa novom dokumentacijom
- priprema priručnika sa uputstvima i standardima za rad i održavanje sistema
- obuhvatanje i priprema ulaznih informacija za punjenje sistema
- izrada plana uvođenja

INSTALIRANJE SISTEMA, TESTIRANJE I PROBNI RAD

- instaliranje i testiranje sistema
- generisanje operativnog sistema
- konverzija i formiranje novih datoteka
- testiranje projekata
- probna obrada

FUNKCIONISANJE PROJEKTOVANIH INFORMACIONIH PODSISTEMA

- planiranje i terminiranje rada sistema
- održavanje programske biblioteke
- održavanje informacione osnove i datoteke
- održavanje i generisanje oeprativnog sistema
- rukovanje sistemom
- ulazno-izlazna kontrola
- priprema informacija.

Prema Bruegge i Dutoit ([BRUEGGE i sar., 2004], 618.str), IEEE 1074 je standard za razvoj procesa životnog ciklusa softvera. Proces je definisan kao skup aktivnosti koje se izvode u skladu sa određenom svrhom. Aktivnost je zadatak ili grupa podaktivnosti koje su zadate timu ili učesniku projekta radi ostvarivanja specifične namene. Zadaci koriste resurse da bi proizveli radni proizvod. Za svaki zadatak određeni su termini početka i završetka i kontrolne tačke radi praćenja toka izvršavanja. IEEE daje spisak od 17 procesa koji čine životni ciklus softvera, a grupiše ih u posebne celine nazvane grupe procesa. U narednoj tabeli dat je spisak grupa procesa, samih procesa i pripadajućih aktivnosti, u skladu sa IEEE1074 standardom.

GRUPA PROCESA	PROCESI	AKTIVNOSTI
Modelovanje životnog ciklusa	Izbor modela životnog ciklusa	određivanje aktivnosti u skladu sa konkretnim projektom
Upravljanje projektima	Inicijacija projekta	pridruživanje aktivnosti izabranom modelu životnog ciklusa softvera
		alociranje resursa za projekat
		ostvarivanje (establish) okruženja za razvoj projekta
		planiranje upravljanje projektom
	Monitoring i kontrola projekta	analiza rizika
		izvođenje planiranja (contingency)
		upravljanje projektom
		upravljanje zapisima (retain records)
		implementacija modela za izveštavanje o problemima
	Upravljanje kvalitetom softvera	planiranje upravljanja kvalitetom softvera
		definisanje metrika
		upravljanje kvalitetom softvera
		identifikacija potreba za unapređenjem kvaliteta
Pred-Razvoj (Pre-Development)	Istraživanje koncepta	identifikacija ideja i potreba
		formulisanje potencijalnih pristupa
		sprovođenje studije izvodljivosti (feasibility study)
		planiranje tranzicije (transition) sistema (ako je izvodljivo)
		detaljiziranje (refine) i finalizacija ideje ili potrebe
	Alokacija sistema	analiza funkcija
		razvoj arhitekture sistema
		dekompozicija sistemskih zahteva (system requirements)
Razvoj	Zahtevi (Requirements)	definisanje i razvoj softverskih zahteva (software requirements)

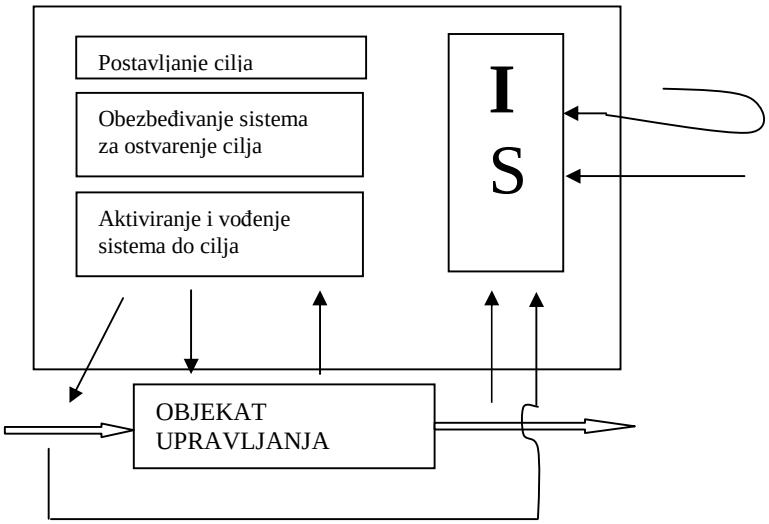
		definisanje zahteva za interfejsom
		određivanje prioriteta i intergracija softverskih zahteva
	Dizajn	izvođenje dizajna arhitekture
		dizajn baze podataka (po potrebi)
		dizajn interfejsa
		selekcija ili razvoj algoritama (po potrebi)
		izvođenje detaljnog dizajna
	Implementacija	kreiranje test podataka
		kreiranje izvornog koda
		generisanje objekt koda
		kreiranje radne (operating) okumentacije
		planiranje integracije
		izvođenje integracije
Post-Razvoj (Post-development)	Instalacija	planiranje instalacije
		distribucija softvera
		instalacija softvera
		prihvaanje softvera u radnom okruženju
	Korišćenje (operation) i podrška	korišćenje sistema
		obezbeđivanje tehničke podrške i pomoći (consulting)
		održavanje zapisa o zahtevima za pomoć
	Održavanje	ponovna primena životnog cilklusa softvera
	Prestanak korišćenja (retirement – »penzionisanje«)	obaveštavanje korisnika
		obezbeđivanje i sprovođenje paralelnog rada (po potrebi)
		prestanak korišćenja sistema
Integral (Cross-Development)	Verifikacija i validacija	planiranje verifikacije i validacije
		izvršavanje zadataka verifikacije i validacije
		prikupljanje i analiza metričkih podataka
		planiranje testiranja
		razvoj zahteva testiranja (develop test requirements)
		izvršavanje testiranja

	Upravljanje softverskom konfiguracijom	planiranje upravljanja konfiguracijom
		razvoj identifikacije konfiguracije
		izvođenje kontrole konfiguracije
		izvođenje izračunavanja statusa (status accounting)
	Razvoj dokumentacije (documentation development)	planiranje dokumentacije
		implementacija dokumentacije
		proizvodnja i distribucija dokumentacije
	Obuka	planiranje programa obuke
		razvoj materijala za obuku
		validacija programa treninga
		implementacija programa treninga

Tabela 2.1. Aktivnosti u procesu razvoja informacionog sistema

2.2. UPRAVLJANJE PROJEKTIMA

Projekat je: nacrt (plan), spisak aktivnosti (niz, red, resursi, sredina), cilj (da bi se uradio neki posao, rešio problem i sl.).



Slika 2.1. Šema upravljanja projektima u razvoju informacionog sistema

U oblasti razvoja informacionih sistema se javljaju 3 vrste projekata:

1. IDEJNI
2. GLAVNI
3. IZVOĐAČKI (DETALJNI)

IDEJNI projekat – Suština idejnog projekta je u predstavljanju postojećih problema trenutno aktivnog informacionog sistema i načelni prikaz mogućih rešenja. Sledi određivanje u kojoj meri će novi računarski podržani informacioni sistem poboljšati postojeći. To se izražava kroz odnos:

$$\frac{\text{T R O Š K O V I (za razvoj IS-a)}}{\text{D O B I T (ušteda, korist uvođenjem IS-a)}}$$

Glavni i izvođački projekti su skupi i troše puno resursa, pa je zbog toga veoma visoka cena njihovog realizovanja.

GLAVNI projekat - Suština glavnog projekta je u detaljnom prikazu postojećeg stanja, analizi i predlogu novog informacionog sistema, koncept novog rešenja i plan realizacije informacionog sistema.

U informacionim sistemima razlikujemo 2 nivoa projektovanja: LOGIČKO i FIZIČKO. Idejni i glavni projekti su na nivou logičkog a izvođački, tj. detaljni projekti na nivou fizičkog projektovanja.

IZVOĐAČKI projekat – Bavi se implementacijom (primenom, uvođenjem i korišćenjem) novog IS-a, bavi se konkretno alatima za razvoj IS-a, kadrovima, organizacijom, programskim zahtevima (strukturom programa, pravima pristupa, izgledom maski, radom sa maskama, načinom izrade, bazom podataka, izveštajima i sl.)

Za upravljanje projektima se koriste posebni programi – alati koji se jednim imenom zovu **CASE alati** – alati za softversko inženjerstvo. Naziv potiče od početnih slova sledećih reči:

Computer
Aided
Software
Engineering

CASE alati su najčešće »metodološki nezavisni« - podržavaju samo modele (Strukturna Sistem Analiza, Model Objekti Veze, UML) i neke transformacije iz jednog modela u drugi, a mogu se prema [LAZAREVIĆ i sar., 1993] koristiti u različitim metodologijama:

- **metodologije I generacije** (metode funkcionalne dekompozicije, strukturno projektovanje i strukturno programiranje),
- **metodologije II generacije** (metode funkcionalne dekompozicije, Model Objekti Veze (ER model), dvoslojna klijent – server arhitektura, relacione baze podataka, jezici IV generacije),
- **metodologije III generacije** (objektni pristup, objektno orijentisani modeli, alati i pristupi, troslojna (višeslojna) arhitektura distribuiranih softverskih komponenti, objektni jezici, i objektno – relacione baze podataka).

Primeri CASE alata: ARTIST (Fakultet organizacionih nauka - Beograd), BPWIN (SSA) i ERWIN (MOV, E-R-A), POWER DESIGNER, Oracle CASE, Rational Rose, Paradigm i sl.

Računarski podržan IS i aplikacije koje rade sa podacima nisu samo baza podataka i programi koji opslužuju tu bazu, već je neophodno da se podaci obrađuju od strane korisnika, koji u svojoj glavi stvaraju informaciju i koriste je za odlučivanje!

S obzirom da postoje dva nivoa projektovanja informacionog sistema (logičko i fizičko) i da je informacioni sistem model nekog realnog sistema, realni sistemi se opisuju kroz:

STATIKU – opisana je bazom podataka i modelom podataka. Jednoj logičkoj strukturi odgovara više fizičkih realizacija.

DINAMIKU – opisana je funkcijama i procesima realnog sistema, tj. modelom procesa informacionog sistema.

Razvoj informacionog sistema je, u stvari, prevođenje jednog nivoa informacionog sistema u novi informacioni sistem, a tim razvojem se mora upravljati. Pri tome se mora voditi računa o činjenici da se Project Management za informacione tehnologije u svetu veoma brzo razvija.

2.1.1. PROBLEMI U RAZVOJU INFORMACIONIH SISTEMA

Koji se **problemi** javljaju u razvoju informacionih sistema?

1. **Problem SLOŽENOSTI POSTOJEĆIH SISTEMA** (sektori, organizacione jedinice, službe i sl.). Rešava se na dva načina: podelom sistema na podsisteme (podceline) i podelom razvoja informacionog sistema na faze u postupku projektovanja.
2. **Problem KOMUNIKACIJE SA KORISNIKOM** (poznaje problematiku poslovnog i organizacionog sistema). Korisnik definiše svoje potrebe i zahteve, a projektanti definišu način na koji će se potrebe korisnika zadovoljiti. Ovaj problem rešava se korišćenjem zajedničkog, jednostavnog jezika simbola i grafičkih oblika koji su dovoljno jednostavni i razumljivi korisniku.
3. **Problem OGRANIČENJA** (novac, vreme). Rešava se tako što se za informacioni sistem utvrdi logička struktura celog sistema (model) a implementacija (uvođenje i primena) zatim sledi deo po deo (definišu se podsistemi za koje se određuju prioriteti implementacije).
4. **Problem UPRAVLJANJA RAZVOJEM**. U projektovanje informacionog sistema, kao deo projektantskog tima je uključeno najviše rukovodstvo organizacije, koje mora da da podršku za uvođenje novog informacionog sistema. Problem se javlja u slučaju kada je projekat urađen, projektantski tim je krenuo u realizaciju pojedinačnih zadataka a rukovodstvo organizacije se promeni ili se u toku vremena promeni organizaciona struktura organizacije. Time se menjaju

početni uslovi koji utiču na završetak posla. Problem se rešava postojanjem pisanog traga o zahtevima korisnika – SPECIFIKACIJA SISTEMA (Dokument kojim korisnik izražava svoje zahteve). Međutim kako korisnik ne poznaje nove mogućnosti informacionog sistema, specifikacija ne mora biti nepromenljiva i konačna jer korisnik ne mora da zna šta želi!

2.1.2. PRINCIPI U RAZVOJU INFORMACIONIH SISTEMA

1. **VREME** – potrebno je posmatrati promene u realnom sistemu u vremenu i omogućiti stabilnost informacionog sistema. Ovo se postiže ADAPTIBILNOŠĆU, tj. parametrizovanjem aplikacija, uopštavanjem realnih objekata, šifarnicima u bazi podataka.

2. **KORISNIK** – cilj projektovanja i implementacije IS-a su ZADOVOLJENJE SPECIFIKACIJE ZAHTEVA KORISNIKA koji zna šta želi a ne zna da to realizuje. Zadovoljenje ovog principa se postiže kroz odgovarajući pristup u projektovanju, kroz učešće rukovodstva u projektovanju i kroz obuku korisnika za rad u novom sistemu.

3. **PROJEKTOVANJE ODOZGO A IMPLEMENTACIJA SA DNA** – sistem se sagleda kao celina, podeli se na podsisteme, projektuje i zatim se implementira jedan po jedan podsistem sa uvažavanjem činjenice da podsistem nije izolovan, već pripada celini.

4. **EKONOMIČNOST** – obezbediti IS-u da zahteva minimum utroška vremena i novca u fazi PROJEKTOVANJA, IMPLEMENTACIJE I KORIŠĆENJA. Ovaj princip podrazumeva USER FRIENDLY korisnički interfejs, čime se povećava produktivnost rada korisnika sistema.

5. **STANDARDI** – međunarodni sistem standarda ISO 9000 - 3 Razvoj softvera u praksi, ISO/IEC 12207 Faze i aktivnosti životnog ciklusa razvoja softvera, Objektno orijentisani dizajn softvera i ISO/IEC 9126, Information Technology – Software Product Evaluation – Quality Characteristics and Guideline for their Use. Ovaj princip podrazumeva i definisanje standarda i propisa u konkretnoj oblasti primene sistema. Softveri IS-a trebaju da imaju podršku za standarde, a da nisu promenljivi u vremenu, trebaju da poseduju standardizovane maske, ekrane, izveštaje i sl.

DEKOMPOZICIJA SISTEMA vrši se na više načina:

a) funkcijama (procesima)
b) postojećom organizacionom strukturom
c) objektima
d) strukturom podataka

Udruživanjem a) i d) dobijamo logičku strukturu informacionog sistema. Formira se matrica SUŠTINSKIH PROCESA (sortiranih prema životnom ciklusu osnovnog objekta obrade) i KLASA PODATAKA (objekti). Grupisanjem preseka u ovoj matrici dobijaju se podsistemi postojećeg IS-a koji su suštinski za njegovo izvršavanje.

2.3. MODELOVANJE U RAZVOJU INFORMACIONOG SISTEMA

U razvoju rešenja problema, čovek koristi različite metode, tehnike i alate. "Modelovanje je dokazana i široko prihvaćena inženjerska tehnika." [BOOCH i sar., 2000]

U rešavanju različitih vrsta problema, čovek je još od davnina došao do sledećih ideja:

1. složen problem podeliti na manji broj problema (dekompozicija, hijerarhija, grupisanje);
2. pojednostaviti problem, uočavajući i rešavajući suštinu (apstrakcija);
3. predstaviti problem grafički, kako bi mogao da ga lakše sagleda u celosti (grafički prikazi);
4. razmeniti iskustva sa ostalim ljudima (jezik kao sredstvo).

Na ovome se i danas temelji modelovanje - na misaonim postupcima, jeziku i grafičkom prikazu kao sredstvu komunikacije i pomoći u savladavanju složenosti.

U nastavku je dato nekoliko najvažnijih principa modelovanja, prema [BOOCH i sar., 2000]:

1. Izbor koje modele kreirati ima suštinski uticaj na to kako je problem "napadnut" i kako je rešenje oblikovano.
2. Svaki model se može izraziti na više različitih nivoa preciznosti.
3. Najbolji modeli su povezani sa realnošću.
4. Nije dovoljan samo jedan model. Svaki netrivijalni sistem je najbolje obrađen kroz mali skup skoro nezavisnih modela.

Neki od problema složenosti u razvoju informacionih sistema i uopšte softvera su:

1. Na sistem se može gledati sa raznih aspekata.
2. Raznovrsnost tehnologije razvoja i samog implementacionog rešenja.
3. Složenost realnog sistema (procesi, podaci, dokumenti,veze, stanja i sl.).
4. Složenost implementacionog sistema (elementi, veze, stanja i sl.).
5. Brojne faze razvoja i rezultati pojedinih faza.
6. Dinamičnost promena realnog sistema na osnovu kojeg se izrađuje implementacioni sistem.

Prikazan je samo jedan deo problema koji treba da se reši kako bi se dobio implementacioni sistem koji:

1. odgovara korisnikovim zahtevima i potrebama i realnom sistemu;
2. efektivan i efikasan u korišćenju;
3. fleksibilan za dalji razvoj i prilagođavanje promenama.

Zašto modelujemo u razvoju informacionih sistema i uopšte softvera?

1. Da bi savladali složenost.
2. Da bi omogućili komunikaciju između različitih učesnika, pre svega korisnika i dizajnera implementacionog sistema.
3. Da bismo što lakše kontrolisali i upravljali razvojem složenog sistema, da bismo lakše planirali razvoj sistema.
4. Da bismo bolje razumeli sistem (pre svega složene sisteme).
5. Da bismo upravljali rizikom.
6. Da bismo lakše i jeftinije evaluirali osmišljena rešenja.
7. Da bismo dokumentovali odluke i specificirali strukturu i ponašanje sistema.

2.4. OPIS REALNOG SISTEMA

2.4.1. OPIS POSLA

Opis posla je tekst koji se sastavlja u toku procesa upoznavanja sa realnim sistemom i snimkom stanja postojećeg informacionog sistema. Da bi se sastavio opis posla potrebno je za posmatrani realan sistem uočiti sledeće:

1. Svrha postojanja sistema - namenu sistema, osnovnu funkciju, čemu koristi i zbog čega postoji taj sistem.
2. Objekti obrade - pošto ih može ih biti više i izdvajanje najvažnijeg, tj. osnovnog objekta obrade.

Svrhu postojanja sistema realan sistem ostvaruje izvršavanjem određenih aktivnosti. U okviru tih aktivnosti obrađuje se osnovni objekat obrade. Aktivnosti se mogu urediti hronološki, tako da slede jedna iza druge u odnosu na različite faze i stanja kroz koje prolazi osnovni objekat obrade od trenutka kad uđe u sistem do trenutka kad promenjen izlazi iz sistema. Taj niz aktivnosti koji se vezuje ze jedan objekat obrade naziva se životni ciklus tog objekta obrade. Naravno, može postojati više objekata obrade i više paralelnih nizova aktivnosti, tj. životnih ciklusa obrade. Za neke objekte obrade pojedine aktivnosti životnog ciklusa su zajedničke. Primer: U video klubu, prilikom izdavanja kopije filma se istovremeno, u okviru jedne aktivnosti, obrađuju dva objekta obrade - član kluba i kopija filma.

Opisom posla se određuju granice posmatranja sistema, pri čemu se jasno može iz teksta uočiti šta je okruženje sistema (drugi sistemi ili kategorije pojedinaca sa kojima dati sistem razmenjuje materiju, energiju i podatke radi izvršavanja aktivnosti). Isto tako se u tekstu mogu uočiti elementi realnog sistema (izvršioci aktivnosti – kadrovi, oprema, resursi itd.).

Opis posla ne treba da sadrži strukturu dokumenata, precizno opisane aktivnosti kroz pojedinačne operacije, poslovna pravila, ograničenja i sl. a može sadržati spisak hijerarhijski uređenih aktivnosti u sistemu.

Sadržaj teksta opisa posla:

- Svrha postojanja sistema,
- Objekti obrade,
- Životni ciklus svakog od objekta obrade hronološkim tekstualnim opisom aktivnosti realnog sistema i za te realne prateće informacione procese,
- U okviru tekstualnog opisa životnog ciklusa navode se elementi sistema i sistemi iz okruženja koji učestvuju u tim aktivnostima,
- Dokumentacija koja se koristi i/ili nastaje nekim informacionim procesom,
- Način arhiviranja podatka u toku izvršavanja informacionih procesa,
- Grafički prikaz hijerarhijski uređenih aktivnosti realnog sistema.

Stablo procesa je grafički prikaz hijerarhijski uređenih odnosa između aktivnosti realnog sistema. Procesi se u stablu uređuju prema osnovnom objektu obrade odnosno njegovom životnom ciklusu. Procesi se uređuju od opštijih ka detljnijim tako što je na vrhu opštiji proces koji se može izvršiti samo ukoliko se izvrše procesi, tj. aktivnosti koje ovaj obuhvata i koji su precizniji. Nepravilno uređenje stabla procesa je prema objektima obrade, prema organizacoj srukturi realnog sistema, prema sličnosti procesa (Primer: Prijava ispita na studijama i prijava prijemnog ispita – ne uopštavati u jedan proces).

2.4.2. SNIMAK STANJA

1. Postojeće stanje realnog sistema

Prikazati postojeće stanje realnog sistema, i naročito njegovog informacionog sistema. Navesti postojeći način obrade podataka, računarske programe, oprema, hardver, mrežu i planove razvoja realnog i informacionog sistema.

2. Spisak dokumenata

Navesti dokumenta koja se koriste u sistemu koja prate aktivnosti, i dokumenta koja se formiraju, primaju ili izdaju u okviru nekih aktivnosti. To su npr. razne knjige, evidencije, računi, fakture, trebovanja, otpremnice, liste, i drugi brojni izveštaji. Potrebno je dati izgled i strukturu dokumenata, kao i način popunjavanja - tipove i formate podataka za određena polja dokumenta.

3. Organizaciona struktura i povezanost sa drugim sistemima

Grafički prikazati celokupnu ili interna organizaciona struktura sistema u zavisnosti od granica posmatranja. Posmatrati i prikazati pripadnost dela koji se analizira celini i povezanost sistema sa okruženjem.

4. Radna mesta, odgovornosti i ovlašćenja

Dati opis radnih mesta i poslova iz kojih se može saznati deo aktivnosti koje se obavljaju u organizaciji. Naravno, ne projektuje se sistem na osnovu radnih mesta i organizacionih celina, jer je to promenljivo, ali može da pomogne za definisanje osnovne funkcije realnog

sistema, a na osnovu toga i funkcija budućeg softvera. Na osnovu radnih mesta, odgovornosti i ovlašćenja se određuju profili korisnika, odnosno prava pristupa nad funkcijama softvera.

5. Zakonska regulativa

Navesti zakone i propise po kojima se obavljaju poslovi i u kojima su definisani poslovi, aktivnosti, organizacione strukture, dokumentacija, njena struktura i postupci u korišćenju dokumentacije. Primer: Za fakultet - Zakon o Univerzitetu.

6. Lokalna pravila poslovanja

Navesti lokalna pravila poslovanja koja se odnose na opis načina izvršavanja aktivnosti koji važe na nivou date organizacije, a nisu precizirana zakonima i drugim opštim propisima za sve organizacije tog tipa. Definisane su npr. statutom te organizacije, raznim pravilnicima ili dokumentima sistema kvaliteta ISO 9000, 14000 itd. Nekad lokalna pravila poslovanja nisu strogo formalno dokumentovana, a ipak su neophodna. Svaka organizacija ima određenu fleksibilnost u okviru opštih propisa i zakona i tu definiše svoja pravila i parametre poslovanja. Primeri:

- Visina školarine – svaki fakultet određuje visinu školarine i način plaćanja (sve, rate, čekovi itd.).
- Broj ispita za uslov prilikom upisa naredne školske godine.
- Parametar - broj iznajmljenih video kaseta, a pravilo: Ako je broj kaseta veći od tri jedna se iznajmljuje besplatno.
- Glerija - formiranje novog reversa sa preostalim slikama nakon vraćanja najmanje jedne, pri čemu se stari revers proglašava nevažećim.

7. Problemi

Potrebno je anketom, intervjuom ili upitnikom uočiti probleme postojećeg informacionog sistema i kako to utiče na uspešnost funkcionisanja i izvršavanja osnovnih delatnosti realnog sistema. Koje greške, neusaglašenosti, neažurnosti, neefikasnosti se mogu javiti ili postoje u poslovanju zahvaljujući postojećem informacionom sistemu.

8. Potrebe

Potrebe su eksplicitno izraženi zahtevi za funkcijama softvera i njihovom načinu izvršavanja a vezani za konkretnu problematiku posmatranog informacionog sistema. Među tim zahtevima, mogu se naći opšti zahtevi za osobinama korisničkog interfejsa, koji mogu da važe za bilo koji drugi softver. Ove opšte zahteve vezane za korisnički interfejs nije potrebno navoditi, s obzirom da se podrazumevaju.

Primeri:

Potrebe

- Mogućnost razduživanja pojedinačnih slika sa zajedničkog reversa.

- Glavna knjiga (finansijsko knjigovodstvo) – Otvaranje i zatvaranje poslovne godine.
- Mogućnost štampanja izveštaja REVERS u Galeriji sa ili bez kolone »Vrednost slike«.

Opšte funkcije softvera
- Validacija podataka prilikom unosa.
- Štampanje različitih izveštaja.
- »User Friendly« korisnički intefejs.
- Pretrage podataka i sortiranje po jednom ili više kriterijuma.

9. Odluke

Koje se odluke donose u sistemu, na osnovu čega i kojih podataka. Jedan od osnovnih ciljeva informaciono-upravljačkih sistema je obezbeđivanje informacione osnove za donošenje odluka.

Softverska realizacija informacione podrške odlučivanju najčešće je u obliku parametarskih upita ili dijagrama i grafikona.

10. Automatizmi

Navesti šta sve treba program da radi automatski.
Primer:
- Automatsko zatvaranje poslovne godine, izrada bilansa stanja, bilansa uspeha, zaključnog lista i otvaranje nove poslovne godine sa prenosom svih konta koja imaju saldo (Finansijsko knjigovodstvo).
- Automatsko smeštanje slike na smeštajno mesto nakon što je vraćena od strane komitenta koji se razdužio.

11. Perspektive razvoja sistema

U toku planiranja razvoja informacionog sistema se definišu koraci razvoja po fazama koje obuhvataju softverske funkcije, hardverske platforme, softversku arhitekturu prema definisanim prioritetima i mogućnostima implementacije. Razgraničiti delove IS koji će biti odmah realizovani jer imaju najviši prioritet od onih koji će biti razvijani u kasnijim fazama.

Najčešće se u prvoj iteraciji razvija softverska podrška za osnovnu delatnost, a u kasnijim iteracijama za prateće aktivnosti (obrada kadrova, sredstava za rad, potrošnog materijala, obrada dokumentacije, organizacioni poslovi i sl.)

12. Organizacija rada

Reorganizacijom postojećeg informacionog sitema javljaju se sledeće organizacione implikacije:

- Potreba za novim radnim mestima za elektronsku obradu podataka ukoliko ista ne postoje,
- Dopuna opisa posla postojećih radnih mesta,
- Sistematizacija, usaglašavanje, pa čak i izrada novih poslovnih pravila i dokumentacije,
- Obaveza/mogućnost evidentiranja podataka koja do tada nije postojala,
- Obaveza/mogućnost za izvršavanje određenih aktivnosti u realnom sistemu.

Programi treba da budu dovoljno fleksibilni da ne primoravaju korisnika na izvršavanje određenih aktivnosti u realnom sistemu osim ukoliko to nije eksplicitni zahtev. Korisnik nejčešće ne želi da uvođenje novog IS-a utiče na bilo kakve organizacione promene, kao ni na navike u dosadašnjem radu. Ipak, projektanti novog IS-a treba da nastoje da unaprede postojeći IS, koji obuhvata pored softverske podrške i unapređenje postojeće organizacije i metdologije rada.

2.5. STRUKTURNA SISTEM ANALIZA

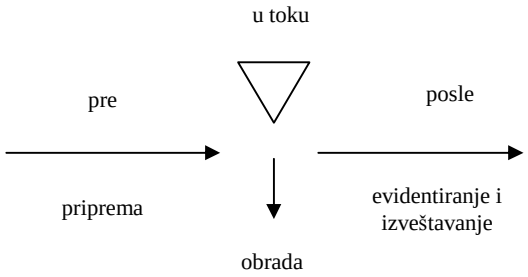
2.5.1. SISTEMSKA ANALIZA

Spoj **sistemska analiza** je misaoni postupak analize sistema sa sledećim koracima:

1. Izbor objekta istraživanja
2. Određivanje aspekata istraživanja objekta
3. Definisanje objekta istraživanja
4. Globalno ispitivanje objekta istraživanja
5. Razlaganje definisanog objekta istraživanja na elemente
6. Generisanje strukture definisanog objekta istraživanja
7. Definisanje kriterijuma istraživanja strukture
8. Istraživanje strukture
9. Istraživanje elemenata
10. Generisanje sistemskog modela
11. Ispitivanje i eksperimentisanje sa sistemskim modelom
12. Verifikovanje rezultata istraživanja
13. Odluka o ishodu procesa analize sistema
14. Primena sistemskog modela

U informacionim sistemima i teoriji sistema u postupku analize sistema posmatramo objekte i sisteme preko procesa koji se u njima dešavaju u vremenu sa postojećom organizacijom i sa određenom tehnologijom.

Primer - posmatramo realan sistem u vremenu (student - prijava ispita - studentska služba). Student predaje ispitnu prijavu, dokument koji ima odgovarajući sadržaj i strukturu, a u suštini on hoće da prijavi ispit postojećom tehnologijom i organizacijom rada. Međutim postoji mogućnost promene organizacione strukture, promene rasporeda radnih mesta i preraspodele službi i promena tehnologije prijavljivanja ispita (prijava, pošta, e-mail, web i sl.).



Slika 2.2. Šema upravljanja projektima u razvoju informacionog sistema

Teži se apstrakciji - izdvajanju procesa koji se dešavaju u sistemu, a koji su nepromenljivi u vremenu. Ove procese zovemo **suštinskim procesima** - to su procesi koji bi se odvijali u nekoj idealnoj tehnologiji, tj. oni koji ne bi imali prostorna i vremenska ograničenja i ne bi bili vezani za neku konkretnu tehnologiju i organizaciju. Suštinski procesi izražavaju svrhu postojanja sistema. Takođe je bitno **izdvojiti fizičke od informacionih procesa**.

Primer - pregled pacijenta na kod lekara (fizički proces) i evidentiranje rezultata pregleda (informacioni proces).

Svaki proces se sastoji u izvršavanju određenog skupa aktivnosti koje se mogu desiti pre, u toku i nakon izvršavanja nekih aktivnosti. Za izvršavanje procesa su potrebni neki podaci koje aktiviraju proces, zatim on vrši određenu obradu podataka i izdaje rezultate obrade podataka. U ovakvoj analizi se procesi posmatraju kao crne kutije pošto ne ulazimo u strukturu procesa. Prilikom analize sistema važno je uočiti kako su procesi međusobno povezani, kako utiču jedan na drugog, koji je redosled izvršavanja procesa i na koji način jedan proces predstavlja osnovu za rad drugog procesa.

2.5.2. OSNOVNI KONCEPTI STRUKTURNE SISTEM ANALIZE

Strukturna sistem analiza (SSA) je najzastupljenija i najpopularnija metoda za modeliranje procesa u projektovanju IS-a. Autori su Žordan (E. Yourdon) i Konstantin (L. Constantine) sa saradnicima, '70 –tih godina, ali je pretrpela izmene do danas - DeMarko (T. DeMarco), MakMenamin (S. McMenamin), Palmer (J. Palmer, Ros (D. Ross) i drugi.

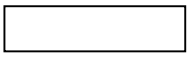
Služi za definisanje specifikacije zahteva korisnika prikazuje šta IS treba da radi, a ne kako i bazira se na funkcionalnoj dekompoziciji procesa uočenih u sistemu koji se analizira. Sadrži dokumentaciju i grafičke elemente koji treba da budu jasni i korisniku i projektantu.

Osnovni elementi SSA:

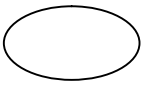
1. Dijagram toka podataka - DTP
2. Rečnik podataka - RP
3. Opis logike primitivnih procesa
   (Pseudo kod, Stabla i tabele odlučivanja, Dijagrami akcija)

2.5.3. DIJAGRAMI TOKA PODATAKA

Osnova dijagrama toka podataka (DTP) je grafički prikaz koji sadrži sledeće elemente:

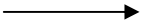


Interfejs kao elemenat predstavlja neki sistem iz okruženja sa kojim posmatrani sistem razmenjuje podatke i to putem tokova podataka (izvor i uvir podataka).



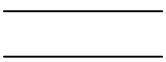
Proces (obrade podataka) su suštinski procesi nekog realnog sistema i mogu se grupisati u 4 vrste procesa:

1. Osnovni ili suštinski procesi
2. Upravljački procesi
3. Razvojni procesi
4. Procesi za održavanje

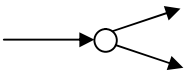


Tok podataka predstavlja apstaktni kanal putem kojeg teku podaci u kontinuiranom vremenu. Treba da je nezavistan od tehnologije i organizacije. Smisao tokova podataka je da omoguće rad procesa i nezavisni su od medija i tehnologije.

Obeležava se sa dva broja (prvi je broj procesa, a dugi o redosledu izvršavanja procesa).



Skladište podataka je tok podataka u mirovanju, i predstavlja apstrakciju koja služi za smeštanje podataka radi njegovog ponovnog korištenja. Procesi su zaduženi za smeštanje podataka u skladišta. Svi elementi SSA se moraju imenovati. To ime je neprekidan niz simbola. Poželjna su duga imena.

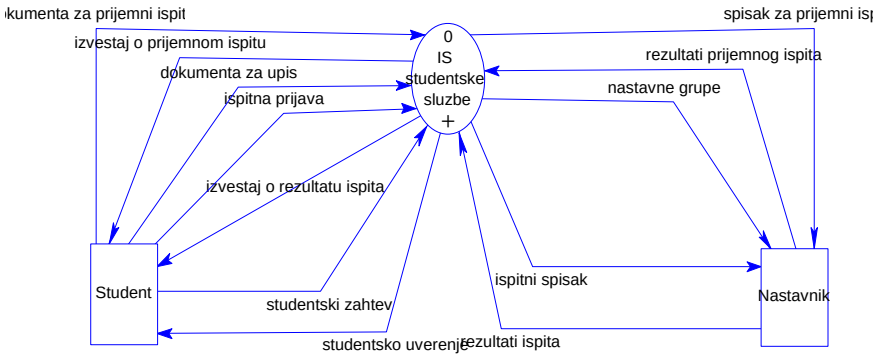


Konektor - spaja dva ili više tokova podataka u jedan ili razdvaja jedan tok podataka u nekoliko pojedinačnih.

2.5.4. FUNKCIONALNA DEKOMPOZICIJA

Funkcionalna dekompozicija se zasniva na principu "od vrha prema dnu" (top-down) po kojem rad na razvoju projekta IS treba započeti od identifikovanja funkcija sistema. Na funkcije sistema se gleda kao na grupu aktivnosti koje se konstantno obavljaju radi ispunjenja odredene svrhe postojanja realnog sistema koji se analizira.

Identifikovanjem funkcija odreduju se granice razvoja, tj. definiše se šta će iz realnog sistema biti obuhvaćeno projektom IS, a šta ne. Rezultat analize je model realnog sistema kojim se grafički, na dijagramu toka podataka (DTP), prikazuje šta sistem treba da radi. Suština metode Strukturne Sistem Analize je da realni sistem posmatra kao proces, koji transformiše ulazne informacije u izlazne informacije. Taj globalni proces (proces 0-og niova) se dalje dekomponuje na prostije (jednostavnije) procese, i postupak dekompozicije se odvija sve dok se ne dođe do niova primitivnih procesa – procesa čija se logika (tok odvijanja) može opisati na jednoj stranici A4 formata. Procese (obrade informacija) treba razlikovati od suštinskih procesa u jednom realnom sistemu (odvijanje nabavke, proizvodni proces, pružanje usluge, način realizacije gotovih proizvoda, planiranje strategije razvoja, implementacija sistema kvaliteta i dr.). S obzirom da je IS složen sistem, prikaz svih informacionih procesa na jednom DTP bio bi veoma nepregledan. Iz tog razloga, opisivanje analiziranog sistema se ostvaruje skupom hijerarhijski povezanih DTP-a. Na vrhu ove hijerarhije nalazi se tzv. **dijagram konteksta** kojim se prikazuje veza sistema sa okolinom (spoljnim objektima). Dijagram konteksta je DTP sa najvišim stepenom apstrakcije u opisivanju sistema i uobičajeno je da se na njemu nalazi samo jedan globalni proces, kojim se prikazuje IS koji se razvija. Zatim se u opisivanju analiziranog sistema pristupa postepenom uvodenju detalja, tako što se svaki globalni proces predstavlja (dekomponuje) novim dijagramom toka podataka na nižem nivou.



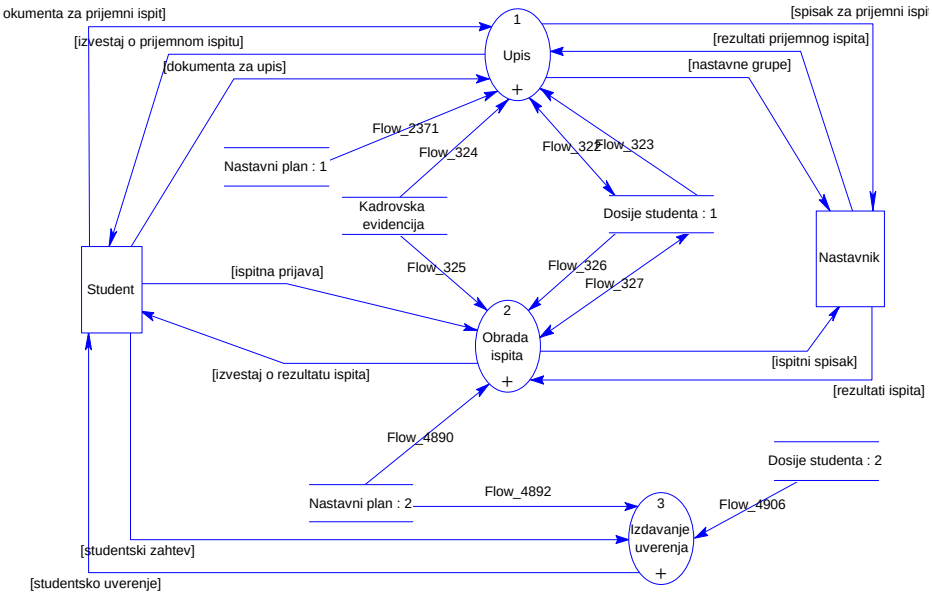
Slika 2.3. Dijagram konteksta SSA

Veze izmedu svih komponenti DTP se uspostavljaju imenovanim tokovima podataka. Pri tome, u dekompoziciji se mora poštovati osnovno pravilo - pravilo balansa tokova, po kojem svaki tok koji ulazi ili izlazi iz procesa na DTP višeg nivoa mora biti jednak sumi ulaznih ili izlaznih tokova sa DTP nižeg nivoa (tj. suma dekomponovanih tokova na nižem nivou hijerahije mora biti jednaka toku, sa višeg nivoa, koji se dekomponuje). Suština ovog pravila je da se dekompozicijom tokova podataka ne izgube podaci na nižim nivoima dijagrama, kao i da se ne doda neki novi tok podataka, koji nije bio zastupljen na višim nivoima hijerahije dijagrama tokova podataka. Radi lakšeg praćenja, vrlo je poželjno ispoštovati i pravilo 7 plus/minus 2 po kome na jednom DTP-u treba da bude prikazano od 5 do 9 procesa.

0. IS Studenstke službe
    1. Upis
        1.1. Evidentiranje kandidata

1.2. Formiranje spiskova za prijemni
1.3. Obrada rezultata prijemnog
1.4. Izveštavanje kandidata o rezultatima prijemnog
1.5. Upis godine
1.5.1. Upis na fakultet
1.5.2. Upis viših godina
2. Obrada ispita
2.1. Evidentiranje ispitnih prijava
2.2. Formiranje spiskova za ispit
2.3. Evidentiranje rezultata ispita
2.4. Izveštavanje o rezultatima ispita
3. Izdavanje uverenja
3.1. Izdavanje uverenja o statusu
3.2. Izdavanje uverenja o položenim ispitima
4. Diplomiranje
4.1. Evidentiranje prijave teme za diplomski
4.2. Formiranje zahteva za odobravanjem tema
4.3. Evidentiranje izvestaja o odobrenim temama
4.4. Zakazivanje odbrane diplomskog rada
4.5. Evidentiranje ocene diplomskog rada
4.6. Izdavanje uverenja o diplomiranju
4.7. Izdavanje diplome

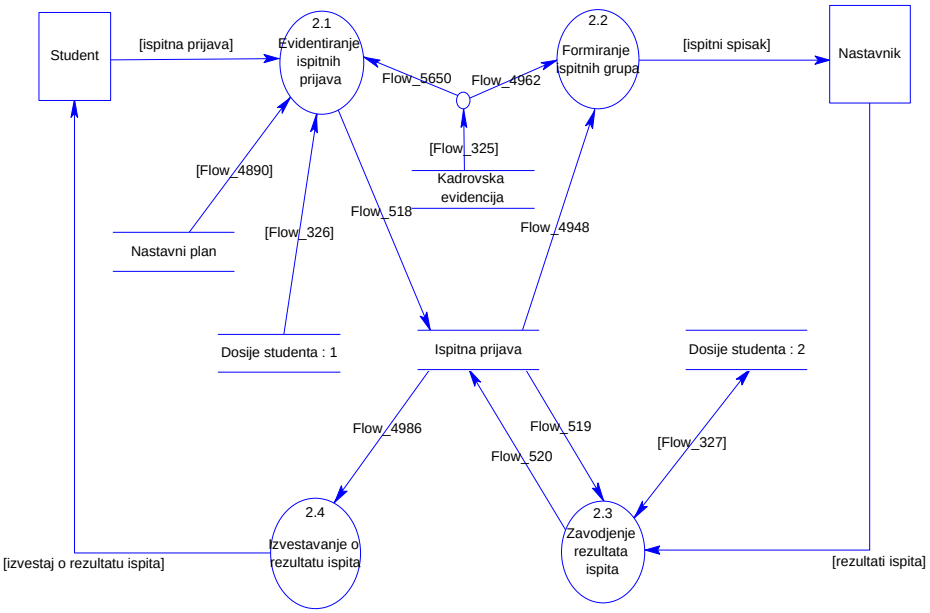
Svi procesi moraju pored naziva nositi i brojnu oznaku. U okviru istog hijerarhijskog nivoa brojne oznake se dodeljuju procesima po redosledu odvijanja. Procesi nižeg nivoa nasleđuju brojnu oznaku procesa koji se dekomponuje, proširenu s desne strane za redni broj procesa u okviru svog hijerarhijskog nivoa (tj. u okviru konkretnog DTP-a).



Slika 2.4. Dijagram toka podataka 1. nivo

Tako će oznake procesa nastalih dekompozicijom dijagrama konteksta biti po redosledu uključivanja: 1., 2., 3, ..., oznake procesa nastalih dekompozicijom procesa 1. biće 1.1., 1.2., 1.3., ..., oznake procesa nastalih dekompozicijom procesa 1.1.. biće 1.1.1., 1.1.2., 1.1.3., ... itd.

Dekompozicija se može obavljati sve do nivoa na kojem dalja dekompozicija procesa nije više moguća ili potrebna. Procesi koji se ne mogu dalje dekomponovati se nazivaju **primitivnim** ili **elementarnim**. Karakteristika primitivnih procesa je da se oni odvijaju sekvencijalno i sa poznatim redosledom aktivnosti.



Slika 2.5. Dijagram toka podataka 2. nivo

Za opis logike ovih procesa, tj. opis aktivnosti koje čine jedan proces, moraju se koristiti neki od specijalizovanih alata: dijagram toka programa, stabla odlučivanja, Nassi-Shneiderman-ov dijagram, pseudokod i drugi. Ipak, kao alat za opis procesa najčešće se koristi pseudokod, u kojem se logika procesa opisuje korišćenjem ključnih reči programskog jezika u kojem će se realizovati aplikacije.

2.5.5. REČNIK PODATAKA

Sastoji se iz tri vrste opisa:

1. Opis strukture i sadržaja svih tokova podataka i skladišta.

   {a, b, c} - unija komponenti - odgovarajuća struktura komponenata može da pojavi više puta.

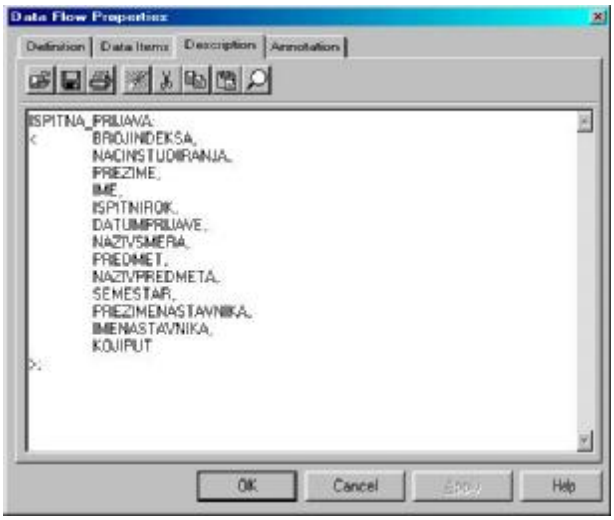
   [a, b, c] - ekskluzivna specifikacija komponenti – međusobno se isključuju strukture.

   <a, b, c> - agregacija komponenti - koje se obavezno pojavljuju tačno jedanput u strukturi.

   /a, b, c/ - neekskluzivna specijalizacija - označava da se u odgovarajućoj srtukturi pojavljuje bilo samo jedna, komponenta, bilo dve, bilo sve.

2. Opis elementarnih podataka (*Data Items*).
3. Opis domena (Domen je skup ispravnih vrednosti iz kojih elementarni podaci uzimaju vrednost – semantička vrednost (definišu je korisnici)).

Primeri:
```
OCENA_STUD: BYTE, >=5,<=10
OCENA: INT(2) IN (1,2,3,4,5)
SEMESTAR: INTEGER (2) BETWEEN 1,10
```



Slika 2.6. Struktura toka podataka

Primer strukture toka podataka “Dokumenta za upis”:

```
DOKUMENTA_ZA_UPIS:
<
      PREZIME,
      IME,
      IMERODITELJA,
      POL,
      DATUMRODJENJA,
      MESTORODJENJA,
      NAZIVMESTARODJENJA,
      NAZIVZEMLJERODJENJA,
      ULICAIBROJ,
      MESTO,
      NAZIVMESTA,
      NAZIVZEMLJE,
      TELEFON,
      DRZAVLJANSTVO,
      NACIONALNOST,
      [
      <
      BROJLICNEKARTE,
      JMBG
      >,
      BROJPASOSA
      ],
      VRSTASKOLE,
      NAZIVSKOLE,
      GODINAZAVRSETKA,
      PROSEK,
      DATUMUPISA,
      VRSTASTUDIJA,
      NAZIVSMERA,
      GODINA,
      SEMESTAR,
      RBRUPISA,
      NACINSTUDIRANJA
>.
```

2.5.6. PSEUDO KOD PRIMITIVNIH PROCESA

Pseudo kod je zapis koji koristi neke reči programskih jezika, ali je veći deo reči na prirodnom jeziku. Opisuju se primitivni procesi, i opisuje se šta taj proces radi.

Načini zapisa domena, je korišćenje logičkih uslova (sadrže neke logičke operatore i konstante). Drugi način je nabrajanje (navođenje iz nekog skupa CHAR – 'M' , 'c' , 'N')

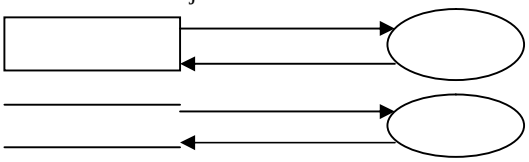
Logika procesa - Evidentiranje ISPITNE_PRIJAVE:

```
Preuzmi ISPITNU_PRIJAVU;
IF ne postoji POTVRDA_O_UPLATI THEN
       BEGIN
       izdvoj ISPITNU_PRIJAVU;
       obavesti STUDENTA;
       END
    ELSE Preuzmi POTVRDU_O_UPLATI
END IF;
IF ISPITNA_PRIJAVA nije korektno popunjena THEN
       BEGIN
       izdvoj ISPITNU_PRIJAVU;
       obavesti STUDENTA;
       END;
END IF;
Pronadji DOSIJE_STUDENTA (ISPITNA_PRIJAVA.BROJ_INDEKSA);
BRPRIJAVLJIVANJA:=0;
FOR EACH DOSIJE_STUDENTA.NAZIV_PREDMETA DO
IF ISPITNA_PRIJAVA.NAZIV_PREDMETA=DOSIJE_STUDENTA.NAZIV_PREDMETA
THEN
       BRPRIJAVLJIVANJA= BRPRIJAVLJIVANJA+1;
END FOR;
IF BRPRIJAVLJIVANJA<>ISPITNA_PRIJAVA.BRPRIJAVLJIVANJA THEN
       BEGIN
       izdvoj ISPITNU_PRIJAVU;
       obavesti STUDENTA;
       END;
   ELSE
       Pronadji Cenovnik;
       IF POTVRDA_O_UPLATI.IZNOS neodgovara cenovniku THEN
              BEGIN
              izdvoj ISPITNU_PRIJAVU;
              izdvoj POTVRDA_O_UPLATI;
              obavesti STUDENTA;
              END;
       Ubaci podatke sa ISPITNA_PRIJAVA u ISPITNE_PRIJAVE;
END IF.
```

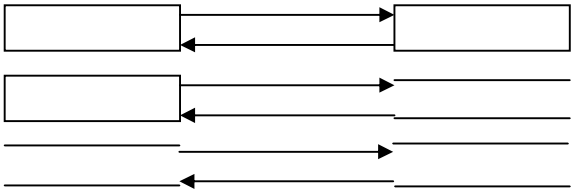
2.5.7. PRAVILA STRUKTURNE SISTEM ANALIZE

1. Više različitih elemenata postoji na jednom dijagramu.
2. Svi elementi dijagrama (DTP-a) moraju biti povezani.
3. Imenovanje elemenata (procesi imaju i broj) je obavezno. Broj procesa se odnosi na redosled izvršavanja u životnom ciklusu osnovnog objekta obrade i potrebno je voditi računa da se uskladi hronologija procesa i brojevi. Broj procesa označava mesto u stablu procesa.
4. Pravilo balansa (automatski podržan u CASE alatu, ali treba ga znati jer i pored toga što postoji automatizam može se narušiti).
   - Ukupno svi podaci koji ulaze i izlaze iz procesa moraju biti zastupljeni na dekompoziciji tog procesa.

   Jednostavnije: Pravilo je narušeno ako na dekompoziciji dodamo ili obrišemo tok koji je povezan sa interfejsom (internih tokova podataka može bii proizvoljno mnogo). Na većem nivou dekompozicije ne sme biti ni "viška" ni "manjka" podataka u eksternoj komunikaciji procesa. Interna komunikacija ne podleže pravilu balansa.
5. Pravila za crtanje dijagrama:
   - Ispravne veze između objekata



   - Neispravne veze između objekata



6. Svaki proces mora da ima bar jedan ulazni i bar jedan izlazni tok podataka.
7. Po pravilu, svako skladište takođe treba da ima barem jedan ulazni i barem jedan izlazni tok. Međutim, dozvoljava se da skladište nema ulazni tok, podrazumevajući da se formira i ažurira u nekom drugom sistemu (mada bi ga tada, možda, logičnije bilo prikazati kao tok od nekog interfejsa), odnosno da nema izlazni tok, podrazumevajući da posmatrani sistem formira i ažurira skladište koje se koristi u nekom drugom sistemu.
8. Pravilo 7+(-)2 - odnosi se na to da na jednom dijagramu toka podataka ma kog nivoa dekompozicije ne može biti više od 7+(-)2 (dakle 5-9) procesa. Ukoliko se javi potreba za više procesa, potrebno je udružiti procese u grupe i "detaljnije procese" prikazati na većem nivou dekompozicije.
9. Na 0.-om nivou dekompizicije ne bi trebalo da se prikažu skladišta podataka.
10. Procesi se povezuju samo preko skladišta podataka, a ne direktno.

2.5.8. HEURISTIKE STRUKTURNE SISTEM ANALIZE

U tekstu koji sledi navode se heuristike za metodu strukturne sistem analize, koje su rezultat višegodišnjeg rada autora u nastavi kursa *Informacioni sistemi*, kao i izrade seminarskih radova studenata. Ove heuristike su date u vidu šireg objašnjenja pojmova i elemenata strukturne sistem analize (SSA) kao i metode, njenih pravila i rezultata (Dijagrama toka podataka - DTP, rečnika podataka - RP) u kontekstu objašnjenja (sa pratećim primerima) elemenata gde su uočeni problemi i greške u primeni metode strukturne sistem analize za različite realne sisteme.

1. ELEMENTARNI PODACI – Predstavljaju relevantne podatke koji su u strukturi toka podataka i skladišta podataka, uz nastojanje da je ova struktura potpuna (Primer: često se zaboravljaju datumi na toku podataka i podaci o uspešnosti izvršene aktivnosti) u smislu da su svi neophodni elementarni podaci zastupljeni. Obavezno je da elementarni podaci imaju jedninu u nazivu i sam naziv elementarnog podatka mora biti potpun, tj. mora se znati kome pripada, odnosno na šta se precizno odnosi elementarni podatak
   Primer: *Datum* je pogrešan naziv elementarnog podatka, *datum_upisa_godine_studija* je primer potpuno definisanog naziva elementarnog podatka).
   Među elementarnim podacima treba razlikovati potencijalno od stvarnog.
   Primer: Razlikovati datum kada je trebalo da se izvrši aktivnost od datuma stvarne realizacije ili promene vrednosti osobine (atributa) nekog objekta u sistemu.
2. IMENOVANE STRUKTURE - U navođenju strukture tokova i skladišta podataka mogu se naći i tzv. imenovane strukture, ali se u daljem detaljnijem opisu mora dati struktura ovih "imenovanih struktura"
   Primer: Strukturu toka podataka *Ispitna_prijava* čine: Student, Nastavnik, Predmet, Ispitni_rok, gde su Student, Nastavnik, Predmet i Ispitni_rok nazivi imenovani struktura; npr. Student se sastoji od sledećih elementarnih podataka: ImeStudenta, PrezimeStudenta, BrojIndeksa, StatusStudiranja, AdresaStanovanja, gde je AdresaStanovanja opet imenovana struktura koja se sastoji od MestaStanovanja, Ulice i broja.
3. DOMEN ZA ELEMENTARNE PODATKE – Svaki elementarni podataka uzima vrednosti iz nekog skupa podataka i treba da je precizno definisan. Postoje dve vrste tih skupova vrednosti (matematičkim jezikom: domena): standardni (ili predefinisani - to su opšti tipovi podataka koji postoje u većini sistema za upravljanje relacionim bazama podataka i programskim jezicima) i tzv. korisnički (u užem smislu reči - domen, a tako se naziva i u CASE alatu Power Designer). Ove "tipove podataka" definiše "korisnik" CASE alata, odnosno sistem analitičar u procesu opisivanja ograničenja realnog sveta nad elementarnim podacima. Domen je podtip osnovnih (standardnih) tipova podataka, gde se definiše pravilo ograničenja, odnosno pravilo kojim se određuje uži skup vrednosti nego što je standardni tip. Pravilo je da domeni imaju množinu u nazivu.
   Primer: Standardni tip je Byte, a domen ili korisnički tip je *Ocene*, koji pripada standardnom tipu Byte sa ograničenjem 1..10. *Ocene_os_ss* su ocene u osnovnoj i srednjoj skoli sa karakteristikama Byte i 1..5, a *Ocene_vs_f* su ocene na visoj skoli i fakultetu, sa karakteristikama Byte i 5..10. *Statusi_finansiranja* su VarChar dužine 20 sa mogućim vrednostima: samofinansiranje, sufinansiranje, budžet.

4. STRUKTURA TOKA PODATAKA I SKLADIŠTA – Strukturu toka podataka i skladišta ne određuje samo skup elementarnih podataka već i njihovi odnosi, pre svega odnosi brojnosti i mogućnosti postojanja. Ovi odnosi se izražavaju različitim notacijama, pri čemu jedna od korišćenih sadrži sledeće simbole: ; , : < >, { }, //, [] Zagrade <...> označavaju da se između njih nalaze elementarni podaci ili imenovane podstrukture koje obavezno moraju imati vrednost i to samo jednu vrednost za dati tok podataka ili skladište
   Primer: Skladište podataka ***Dosije_studenta*** obavezno mora imati i to samo jednu vrednost Imena i Prezimena studenta, njegovog broja indeksa itd.
   Zagrade {...} označavaju da se elementarni podaci ili imenovane podstrukture sa datom strukturom mogu javiti jednom ili više puta u datom toku podataka ili skladištu. Najčešće se koriste u paru sa <> zagradama kao {<...>}.
   Primer: Tok podataka Narudžbenica (ili Podaci o naručivanju robe) kao zaglavlje ima po jedan primerak podataka o kupcu, dobavljaču, datumu naručivanja, a kao stavke, koje su najčešće tabelarnog prikaza i unosa daju se (za svaku stavku ista podstruktura): Redni_broj, naziv_robe, količina_robe. Dakle Narudžbenica: <Dobavlja;, Kupac, Datum_Narucivanja, {<redni_broj, naziv_robe, kolicina_robe>};
   Zagrade /.../ označavaju da podstruktura sadrži elemente u vidu unije - za pojedine elemente ili sve elemente mogu u konkretnim primercima tokova podataka biti zastupljeni podaci.
   Primer: Kod narudžbenice stavke mogu biti {<redni_broj, /šifra_robe, naziv_robe/, kolicina_robe>}. Dakle, dovoljno je navesti ili šifru robe ili naziv robe, a mogu i oba, što nije greška.
   Zagrade [...] označavaju da se unutar podstrukture elementi međusobno isključuju, dakle u jednom konkretnom slučaju ne mogu bizi zastupljeni podaci za sve grupe, već ili samo jedne ili druge...
   Primer: U skladištu podataka Dosije_studenta podaci o studentima iz zemlje sadrže sledeće elementarne podatke: JMBG, broj_lične_karte, Izdao_SUP. Podaci o stranim državljanima sadrže: Naziv_zemlje, Broj_pasoša. Dakle, u strukruri Dosijea_studenta nalazi se sledeća podstruktura: [<JMBG, broj_lične_karte, Izdao_SUP>, < Naziv_zemlje, Broj_pasoša >]
5. STABLO PROCESA – Prilikom uređivanja procesa u stablo, bitno je obratiti pažnju na kriterijum grupisanja aktivnosti u procese.
   - nije dobar kriterijum: grupisanje orijentisano na dokumente (*Primer*: Evidentiranje diploma, Evidentiranje prijave za upis, Evidentiranje uplatnice i sl.).
   - nije dobar kriterijum: grupisanje orijentisano na spoljne entitete (*Primer*: Poslovi vezani za dobavljača, Poslovi vezani za kupca, Poslovi vezani za Banku itd.).
   - nije dobar kriterijum: grupisanje orijentisano na poslove iste vrste (*Primer*: sva prijavljivanja ispita, jer nije svejedno koje su vrste ispiti: prijemni ili redovni, na osnovnim studijama ili poslediplomskim i sl.).
   - dobar kriterijum: udružiti sve poslove po logičkoj pripadnosti vrsti delatnosti (*Primer*: Osnovna, Upravljačka i Pomoćna delatnost) .
   - dobar kriterijum: svrha postojanja sistema i sve aktivnosti da se ona ostvari, šta je sve neophodno uraditi da bi se ostvario cilj sistema ili procesa (kada govorimo o određivanju njegovih podprocesa). (*Primer*: Svrha postojanja Zdravstvene ustanove (u odnosu na osnovnu delatnost) je pružanje zdravstvenih usluga, uključujući lečenje pacijenata, aktivnosti koje su potrebne da se ostvari osnovna delatnost je Uvođenje

pacijenata u evidenciju, Dijagnostika uključujući pregled, Sprovođenje terapije, Upućivanje u druge medicinske ustanove, Izdavanje uverenja).

- dobar kriterijum: životni ciklus osnovnog objekta obrade. Udružiti procese po fazama životnog ciklusa, ili barem na pripremnu, izvršnu i završnu fazu. (***Primer***: IS fakulteta- segment Obrada nastave. Pripremna faza: Prijemni ispit i upis studenata, Izvršna faza - Nastavni proces uključujući i ispite, Završna faza - Diplomiranje i izveštavanje)

6. PROCES KAO CRNA KUTIJA – U razmatranju ulaznih i izlaznih tokova podataka i skladišta i u i nastojanju da ovaj skup bude što potpuniji, za svaku proces ma kog nivoa dekompozicije, postavljamo sledeće pitanje: šta je izlaz iz datog procesa, koja je njegova svrha i rezultat (proizvod, usluga, aktivnost, ali šta tim povodom izlazi kao tok podataka)? Kada odgovorimo na ovo pitanje, prirodno se nadovezuje sledeće: "Šta nam je sve neophodno od podataka da bi realizovali taj izlaz?", u nastojanju da je skup ulaza što potpuniji. (***Primer***: Proces UpisGodine rezultuje Tokom podataka o upisanoj godini koji je usmeren ka studentu i ka skladištu podataka DosijeStudenta. Da bi ovaj proces uspešno izvršio svoj cilj mora konsultovati DosijeStudenta i proveriti broj položenih ispita i odgovarajuće nazive predmeta ali takođe mora proveriti i skladište PravilaRada gde su evidentirana pravila koja se odnose na uslov za upis naredne godine studija).
7. BROJ INTERFEJSA – Najčešće realni organizacioni sistemi predstavljaju izuzetno složene sisteme, posebno u delu komunikacije sa okruženjem. Po pravilu, organizacioni sistemi ma koje složenosti nemaju samo jedan sistem iz okruženja sa kojim komuniciraju (interfejs na dijagramu), već više. Kao heurističko pravilo se nameće potreba da na 0. nivoa ne postoji samo jedan, već barem dva interfejsa. Ukoliko nije odmah očigledno koji su to interfejsi, detaljnijim upoznavanjem sa realnim sistemom i sledeći ovu heuristiku dolazimo do drugih interfejsa. (***Primer***: Zubarska ordinacija - kao osnovni objekat obrade javlja se Pacijent; on je ujedno i glavni interfejs. Ipak, kada se uzmu u obzir i upravljačke i pomoćne delatnosti (koje se bave resursima datog sistema), dolazi se do novih interfejsa: Lekarska komora, Ministarstvo zdravlja, Dobavljači lekova i opreme, Tržište rada, Banka itd.)
8. ZAPOSLENI KAO INTERFEJS - u opisu delatnosti i životnog ciklusa osnovnog bjekta obrade često se spominju aktivnosti različitih radnih mesta. Ljudi koj rade na tim mestima predstavljaju izvođače aktivnosti (procesa) i kao takvi su deo tih procesa; ne smeju se postavljati kao okruženje (interfejs). Zaposleni koji izvršavaju ove procese se "kriju unutar" procesa i njih nigde ekplicitno ne prikazujemo. Ovde treba napomenuti razliku SSA prema Dijagramima slučajeva korišćenja (USE CASE) softvera u okviru UML-a. U SSA se govori o poslovnom sistemu i okruženju celog realnog organizacionog sistema, a u USE CASE dijagramima se govori o okruženju softvera, odnosno o Actor-ima (ulogama, odnosno profilima korisnika softvera). u slučaju USE CASE naravno da su različita radna mesta predstavljena kao Actori i "okruženje softvera" sa svojim definisanim pravima pristupa pojedinim softverskim funkcijama. Naravno kao Actori se predstavljaju i korisnici iz okruženja poslovnog sistema ukoliko to softverski sistem podrazumeva.
9. POVEZANOST PROCESA ISTOG NIVOA - Na istom nivou dekompozicije svi procesi treba da su preko skladišta podataka međusobno povezani. Nije dozvoljeno da postoji proces koji je nepovezan, izolovan od drugih procesa, sa svojim tokovima podataka i skladištima koje ne koristi ni jedan drugi proces.

10. DEKOMPOZICIJA - Predstavlja "razdvajanje celine na delove", odnosno opštijeg, šireg procesa na aktivnosti po hronologiji izvršavanja kojima se omogućuje ostvarivanje svrhe tog procesa. Nije dozvoljeno dekomponovati jedan proces na samo jedan podproces.
11. SLEDLJIVOST PODATAKA - Svi elementarni podaci na DTP imaju svoj izvor -mesto nastanka, mesto korišćenja i put protoka kroz sistem. Na taj način može se pratiti protok podataka do nivoa elementarnog podatka, njegovu sledljivost. Na taj način može se vršiti i provera elementarnih podataka - ne smeju se pojaviti bez prethodnog razloga i izvora.
12. POVRATNA VEZA - Heuristično pravilo nalaže da svaki ulazni tok podataka ima svoj izlazni i obrnuto. Ovo pravilo zasniva se na prirodnom ponašanju realnog sistema u dinamičkom i otvrenom odnosu prema okruženju - uzročno – posledična veza. Na spoljašnji impuls (u vidu protoka materije, energije i podataka) odgovaramo informacijom o našoj aktivnosti (proizvodnji, pružanju usluge i sl.) i pokreću se dalje aktivnosti u sistemu. Kao odgovor na spoljni stimulans, elementi realnog sisema reaguju aktivnostima i slanjem informacija okruženju.
13. ŠTA SU PROCESI-AKTIVNOSTI SISTEMA - Interakcija sa okruženjem rezultat je događaja u okruženju koji iniciraju interne događaje-aktivnosti. Realni sistem (organizacioni sistem koga modelujemo) ima interne procese kojima reaguje na spoljne događaje i ovo predstavlja jedan od načina određivanja procesa-aktivnosti sistema.
14. PODACI U SKLADIŠTU – Skladište podataka može sadržavati: činjenice (o objektima i događajima realnog sistema i okruženja), znanje i pravila o osobinama i ponašanju objekata, parametri rada sistema, opšte podatke.
15. SKLADIŠTE BEZ ULAZNOG TOKA PODATAKA – Po pravilu svako skladište podataka mora imati bar jedan ulaz i bar jedan izlaz. Međutim, moguće je koristiti tzv. deljeno skladište koje se puni u nekom drugom sistemu, a koristi u posmatranom sistemu. Radi potpunog poštovanja prethodno navedenog pravila, uvode se procesi preuzimanja podataka iz okruženja i punjenja skladišta. Ovi procesi nisu deo prirodnog životnog ciklusa osnovnog objekta obrade i deluju "veštački". Ukoliko bi posmatrani realni sistem bio posmatran u širem kontekstu okruženja, za ovim procesom ne bi bilo potrebe. Ipak, vrlo često se informacioni sistem i "obuhvat posla" radi samo za jedan segment delatnosti, tj. organizacionu jedinicu ili pojedinu organizaciju.
16. RELATIVNOST NAZIVA I BROJA SKLADIŠTA – važno je da naziv skladišta odslikava suštinu podataka koje sadrži i svoju namenu, odnosno naziv treba da je mnemonički. Ne treba težiti da postoji samo jedno skladište podataka za ceo sistem (skladište podataka nije isto što i baza podataka), a takođe ne treba poistovetiti skladište podataka i tabele relacione baze podataka.
17. SPISAK PROCESA – Pri određivanju procesa-aktivnosti datog realnog sistema, potrebno je izdvojiti šta su osnovni procesi posmatranog sistema. Prateći životni ciklus osnovnog objekta obrade dolazimo do određenih procesa-aktivnosti. Među njima posebno izdvojiti interne procese i procese u interakciji sa okruženjem, odnosno fizičke i informacione procese. Od interesa u SSA je prikaz informacionih procesa koji prate fizičke procese realnog sistema. Posebno treba razdvojiti OSNOVNE, PRATEĆE I UPRAVLJAČKE PROCESE. Nije potrebno posebno izdvajati i nazivati procese koji se odnose na prijem i evidentiranje dokumenata ili identifikaciju objekata (Primer: utvrđvanje identiteta Studenta ili Dobavljača putem lične karte i sl.). To su procesi koji

su sadržani u svakom procesu. Nazive procesa ne treba vezivati za radna mesta i organizacione jedinice.

Primer: fizički proces je pružanje usluge transporta - dakle sam fizički proces transporta, a informacioni proces koji ga prati je evidentiranje izvršene usluge transporta. U DTP ćemo proces nazvati : evidentiranje izvršene usluge transporta, jer se ovde bavimo dijagramom toka podataka i modelom informacionog sistema koji je slika fizičkog realnog sistema.

18. INTERPRETACIJA, DOPUNA I UOPŠTAVANJE - modelovanjem u SSA nastojimo da predstavimo suštinske procese realnog sistema, nezavisno od postojeće organizacije i tehnologije. Na putu ka opštem modelu najpre možemo izraditi model SSA preslikavanjem postojećeg realnog organizacionog sistema. Naredni model treba da predstavlja njegovo uopštavanje i izvlačenje jezgra aktivnosti koje bi postojale u ma kojoj tehnologiji. Takođe, prilikom modelovanja predstavljamo sistem kakav treba da bude, a ne kako trenutno (možda i nepotpuno i nepravilno) funkcioniše.
19. PRE, U TOKU I POSLE - Radi pravilnog nazivanja procesa u SSA možemo slediti sledeću heuristiku: pre u toku i posle fizičkog procesa (proizvodnja, pružanje usluge i sl.) dešavaju se informacioni procesi koji predstavljaju pripremu, evidentiranje i izveštavanje o urađenom fizičkom procesu.
20. SVRHA SSA je predstavljanje znanja o realnom sistemu:
    a) uopštavanjem sa sličnim sistemima (koji se bave istom delatnošću) možemo lakše izdvojiti suštinu funkcionisanja nezavisno od postojeće tehnologije / organizacije. Na taj način dobijamo univerzalan vanvremenski model za koji može biti implementiran informacioni sistem u bilokojoj tehnologiji.
    b) specifikacijom informacionih potreba realnog sistema pripremamo materijal za fazu modelovanja podataka, modelovanja sofverskog rešenja - dizajna softverskih funkcija, ekrana i izveštaja.
21. DOKUMENT - TOK PODATAKA - Više dokumenata koji se obrađuju jednim procesom čine jedan tok podataka. Prazan obrazac (ili pratan deo obrasca koji se u nekom trenutku ne popunjava nije deo toka) ili kopija nekog dokumenta nije tok podataka. Tok podataka obuhvata sve medijume, ali zapravo predstavlja apstrakciju (opšti kocept nezavisno od tehnologije) i esenciju (izdvaja suštinske podatke koje izvor saopštava uviru).
22. DUBINA DEKOMPOZICIJE I ODREĐIVANJE PRIMITIVNIH PROCESA -
    - procesi koji se ne mogu paralelno i nezavisno izvršavati su aktivnosti unutar primitivnih procesa. Njih ne prikazujemo na DTP, već u pseudo kodu.
    - procesi koji imaju jedan ulaz i jedan izlaz su najčešće primitivni procesi.
    - jedan primitivni proces preslikavamo u programu na jedan ekran.
    - logiku primitivnog procesa možemo opisati na jednoj strani A4 formata.
    - kada bi se dekompozicijom datog procesa dobili trivijalni procesi (Primer evidentiranje prezimena) koji nezavisno ne mogu egzistirati i čije izvršavanje ne može biti prekinuto u vremenu i rezultati rada evidentirani u skladištu podataka ponaosob, već isključivo u nizu, taj proces je primitivni. Ove "aktivnosti obavezno u nizu bez prekida" nisu procesi već interne aktivnosti primitivnog procesa i predstavljaju se pseudo kodom.
    - kada bi se dekompozicijom procesa dobile naredbe programskog jezika, taj proces nazivamo primitivnim.

- kada bi dekompozicijom procesa ušli u nivo implementacije u programu (primer: unos, brisanje, izmena, sortiranje, tabelarni prikaz - nisu primitivni procesi, već softverske funkcije).

23. OBJEKAT OBRADE I INTERFEJS – Objekat obrade u sistemu može biti materijalnog (Primer: proizvod) ili idejnog karaktera (Primer: znanje, usluga, zdravlje), a kao nosilac može se javiti pojedinac (Primer: Student, Pacijent i sl.) ili organizaciona tvorevina. Pojedinac u različitim fazama "obrade" može imati različite nazive, ali je to ipak sve vreme isti interfejs i može se na DTP tako predstaviti. Radi preciznijeg prikaza i veze prema kasnijim USE CASE modelima, moguće je prikazati ih kao različite, onoliko koliko ih ima. (Primer: Kandidat za studenta, student, apsolvent, diplomac, podnosilac molbe i sl.), međutim radi preglednosti dozvoljeno je da bude prikazan i kao jedan. Razlikujemo osnovne objekte obrade prema vrstama delatnosti - osnovna, pomoćna i upravljačka, a u okviru svake od navedenih može biti više objekata obrade. Osnovni objekat obrade određujemo prema osnovnoj delatnosti, onom objektu koji je u vezi sa svrhom postojanja realnog sistema.
24. GRANICE POSMATRANJA – U razvoju informacionog sistema granice obuhvata posla određuju se u početnoj fazi strateškog planiranja. Svaki organizacioni sistem je deo nekog šireg sistema. u složenim organizacionim sistemima potrebno je odrediti informacione podsisteme (npr. primenom metode BSP - Business System Planning koristeći posebne matrice) i po redosledu prioriteta pristupiti analizi i implementaciji. Informacioni podsistemi najčešće ne odgovaraju organizacionim podsistemima.
25. OPIS POSLA I SSA - primena metode SSA zasniva se na poznavanju načina funkcionisanja i protoka podataka realnog sistema. Sam opis aktivnosti životnog ciklusa osnovnog objekta obrade u tekstualno obliku kao osnov modeliranja predstavlja rizik, jer zahteva veliku preciznost kako bi rezultati SSA bili zadovoljavajući. Zato je proces primene metode SSA iterativan (više iteracija razvoja i dopune modela) i interaktivan sa zaposlenima realnog sistema koji imaju uvida u modele, učestvuju u njihovoj izradi i korekciji.
26. INTEGRITET SKLADIŠTA - Skladište podataka ne može u strukturi da sadrži elementarne podatke, koji nisu putem ulaznog toka podataka "dopremljeni" u to skladište. U CASE alatu Power Designer Process Analyst opcijom Data Store Auto Fill omogućeno je automatsko punjenje elementarnih podataka iz tokova podataka u skladišta i zabrana nezavisnog unosa elementarnih podataka.
27. DELJENO SKLADIŠTE PODATAKA – na prvom i daljim nivoima dekompozicije skladište podataka omogućuje vezu među procesima, kada jedan proces svoj rezultat rada smešta u skladište podataka, da bi služilo drugim procesima u sistemu radi čitanja ili korekcije-dopune novim elementarnim podacima. Skladište podataka treba da se crta na onom nivou gde ga koristi više procesa. Ukoliko na nekom nivou skladište koristi samo jedan proces, tada je to skladište interno na nivou dekompozicije datog procesa.
28. PARALELENOST MATERIJE, ENERGIJE I INFORMACIJE - U realan sistem ulaze: materija, energija i podaci, ali i podaci o materiji i energiji i to u obliku parametara ili deskriptivno (opisno).
29. REDUNDANSA - Na Dijagramima Toka Podataka je dozvoljena redundansa, a na Modelu Objekti Veze težimo njenoj minimizaciji. To znači, da se jedan elementarni podatak može pojaviti u različitim skladištima ili tokovima podataka DTP-a ali, naravno, ne i u okviru istog.

30. INTERFEJS je suštinski izvor/uvir podataka, a ne posrednik.
    Primer: sva komunikacija (pošta, fax, e-mail firme, telefoni) u nekom preduzeću se odvija preko tehničkog sekretara, ali nije sekretar izvor podataka (INTERFEJS) već npr. dobavljač, kupac, ministarstvo za finansije, banke i sl..
31. ISTOVREMENO I ISTI NIVO- Ako se neke aktivnosti istovremeno odvijaju to može biti jedan proces. Ne moraju svi procesi biti dekomponovani do istog nivoa.
32. SSA opisuje procese kao odgovor na pitanje ŠTA?, a ne KAKO? Na taj način treba opisati aktivnosti u sistemu, bez zalaženja u detalje implementacije.
33. DUGAČKI NAZIVI - Poželjna su duža imena (a ne skraćenice) u nazivima elemenata DTP-a (reči odvojene simbolom »_«).
34. RAZLIČITI NIVOI - Na istom dijagramu greškom se mogu naći procesi čije je mesto na dijagramima različite dubine. Vrlo pažljivo je potrebno odrediti da li je neki proces "sveobuhvatniji ili detaljniji" i odrediti mu pravilno mesto i odnos prema drugim procesima. (Primer: Proces Prijem robe i Kontrola pristigle robe su u odnosu nadređeni-podređeni).
35. Problemu određivanja "ŠTA PREDSTAVLJA I KAKO NAZVATI SKLADIŠTE PODATAKA" pristupamo empirijski. To je u realnom sistemu svaki dosije, kartoteka, knjiga evidencije i sl. (otuda simbol dve paralelne linije, kao pogled na knjigu "sa boka"). Takođe, to je znanje i pravila obavljanja delatnosti. Skladište podataka možemo posmatrati i kao tok podataka u mirovanju, sredstvo gde će se sadržaj toka podataka smestiti, radi kasnijeg korišćenja od strane nekog drugog procesa, sredstvo gde se smeštaju podaci u vremenskom prekidu aktivnosti, sredstvo za smeštanje podataka iste vrste radi zajedničke masovne obrade. U skladištu podataka se mogu smeštati podaci i iz više različitih tokova podataka. Ne treba zaboraviti da se sadržaj izlaznih podataka sa tokova čuva u vidu skladišta podataka (Primer: Druga kopija nekog dokumenta)
36. Među nazivima elementarnih podataka koji se nalaze na tokovima podataka mogu se naći HOMONIMI I SINONIMI. Posebno se ova situacija može javiti ukoliko dokumentacija realnog sistema nije u ovom smislu usaglašena, a prilikom definisanja strukture tokova podataka se "preslikavaju" nazivi iz dokumentacije. Zadatak u oblasti razvoja informacionog sistema (sistem analitičara i domenskog stručnjaka) sastoji se i u "usaglašavanju rečnika stručnih termina", odnosno termina koji se koriste za iste ili različite pojmove, dokumentovanjem u okviru posebnog dokumenta "Glossary of terms", odnosno dokumenta "Client requirements" ("Specifikacije zahteva korisnika"), rezultata sistemske analize. U ovom poslu usaglašavanja i dokumentovanja, potrebno je definisati jedinstvene nazive. Inače, s obzirom da se problematični elementarni podaci importuju iz SSA u razvoj modela modataka, rešavanje ovih problema se odlaže za fazu modelovanja podataka, što može predstavljati problem dizajneru baze podataka, koji ne mora toliko detaljno da poznaje problematiku realnog poslovnog sistema i ovakve greške dovode do slabijeg razumevanja i otežanog rada na modelovanju podataka
    Primer: Termini Predavač, Profesor, Nastavnik na fakultetu odnose se u kontekstu nastave na isti pojam, ali u kontekstu nastavnih zvanja imaju različito značenje - predavač i profesor- redovni, vanredni su nastavna zvanja radnog mesta nastavnik. Važno je u tzv. rečniku termina opisati šta se pod ovim terminima podrazumeva.

37. Obavezni su nazivi na tokovima podataka, kojima se razmenjuju podaci sa interfejsima. Nisu obavezni nazivi na internim tokovima podataka (od ili ka skladištu), ukoliko sadrže sve podatke kao i struktura datog skladišta.
38. Sistem koji se izučava i modeluje putem SSA deo je nekog šireg, opštijeg sistema, njegov element. Važno je procese posmatranog sistema analizirati u kontekstu okruženja i pripadnosti ovom širem sistemu. Na taj način lakše je odrediti ovakve interfejse i podprocese posmatranog sistema čija je osnovna uloga komunikacija sa datim interfejsima.
    Primer: Studentska služba pripada Fakultetu, Fakultet Univerzitetu, sistemu visokog školovanja, odnosno sistemu obrazovanja. Pored Studentske službe na fakultetu postoje i druge organizacione jedinice, koje predstavljaju neposredno okruženje, a posredno predstavljaju institucije Univerziteta itd. Pri analizi procesa treba uzeti u obzir kakva je razmena podataka između Studentske službe i ostalih entiteta iz okruženja, tj.interfejsa.
39. Na DTP-u 0. nivoa se ne crta interakcija između samih interfejsa (nemoguće je i nepravilno povezati dva interfejsa na DTP-u). U kontekstu životnog ciklusa događaja povodom osnovnog objekta obrade moguće su aktivnosti razmene podataka između samih interfejsa za posmatrani sistem, ali u DTP je od interesa samo kakva je interakcija svakog od interfejsa sa datim sistemom zasebno.
    Primer: razmena podataka između Kadrovske službe i Biblioteke nije od interesa za Studentsku službu pa se ne prikazuje na DTP-u 0. nivoa studentske službe. Ipak, informacioni sistem studentske službe fakulteta ne bi trebalo razvijati nezavisno od informacionog sistema Fakulteta.
40. Tokovi podataka ne smeju se nazivati kao procesi.
    Primer: nije dobro da se tok podataka naziva *Prijavljivanje_ispita*, već treba da njegov naziv bude: *Prijava_ispita- kao naziv stvarnog dokumenta*, ili *Podaci_o_prijavi_ispita- kao suštinski sadržaj tog dokumenta, odnosno toka podataka*.

2.6. PROJEKTOVANJE BAZA PODATAKA

Baza podataka je statička slika modela podataka jednog informacionog sistema. Tačno je da je model podataka a pogotovo njegov deo koji se odnosi na bazu podataka vrlo bitan za ceo informacioni sistem. Tačno je da bez dobro projektovane baze podataka informacioni sistem nikako ne može biti dobro projektovan, ali projektovanjem baze podataka još uvek nije rešen ceo problem. Baza podataka je statička slika modela podataka, odnosno slika informacionog sistema u jednom trenutku. Bazom podataka se ne mogu izraziti ograničenja modela podataka (bar ne kod klasičnih modela, kao što su hijerarhijski, mrežni i klasični relacioni), tako da je njih potrebno implementirati aplikacijom.

"Konzistentnost baze podataka obezbeđuju posebni mehanizmi u okviru sistema za upravljanje bazom podataka. Postoje dve grupe mehanizama za kontrolu konzistentnosti baze podataka. To su:

- **pasivni** mehanizmi, koji samo sprečavaju takve promene sadržaja baze podataka, koje bi narušile uslove integriteta i
- **aktivni** mehanizmi, koji održavaju propisane odnose između podataka o različitim entitetima tako, što menjaju podatke o jednim entitetima, kada se menjaju podaci o drugim." [MOGIN i sar., 1996]

Ovakva ograničenja su važila do početka 90-tih godina, sve do uvođenja aktivnih sistema baza podataka. Prvi pristupi uključivanja aktivnih komponenti sreću se dosta rano i u različitim oblastima. U programskim jezicima je to obrada izuzetaka (exceptions), a u sistemima veštačke inteligencije to su demoni (daemons). CODASYL (mrežni) model podataka podržava procedure, koje su memorisane u bazi podataka, a pozivaju se na sledeći način:

```
ON STORE CALL <procedura>;
ON ERROR DURING MODIFY CALL <procedura>.
```

"Relacioni sistemi omogućuju definisanje iskaza i trigera (Standardi SQL2-1992. i SQL3-1999.), definisanje pravila (INGRES Knowledge Management System) ili delimičnu podršku transakcija (Sybase – Transact SQL Enhancements). "[MARJANOVIĆ, 1993]

"Aktivnu bazu podataka čine baza podataka i skup aktivnih mehanizama za praćenje stanja baze podataka. Aktivni mehanizmi reaguju na nedozvoljene promene stanja, bilo na takve promene, koje traže neku akciju." [MOGIN i sar., 1996]

U SQL-92 standardu (ISO/IEC 9075:1992, »Information Technology – Database Languages – SQL« ili ANSI X3.135-1992, »Database Language SQL« definišu se naredbe za:

- deklarisanje integriteta domena putem opisa domena (naredba CREATE DOMAIN),
- deklarisanje ograničenja u okviru opisa šeme relacije (naredba CREATE TABLE), i
- deklarianje opštih ograničenja (naredba CREATE ASSERTION).

U standardu SQL3 se iskazi i trigeri definišu na sledeći način:

```
CREATE ASSERTION <naziv_iskaza>
       {BEFORE COMMIT |
       AFTER {INSERT | DELETE | UPDATE [OF <lista_kolona>]}
       ON <tabela>
       CHECK <predikat> [FOR EACH ROW]

CREATE TRIGER <naziv_trigera>
       {BEFORE | AFTER} {INSERT | DELETE | UPDATE [OF <lista_kolona>]}
       ON <tabela>
       [REFERENCING OLD AS <staro_ime>
                NEW AS <novo_ime>]
                [WHEN <predikat>]
                <lista_naredbi> [FOR EACH ROW OF <tabela>]
```

Na osnovu sintakse navedenih naredbi može se zaključiti da se događaji odnose isključivo na operacije nad bazom podataka, da uslovi proveravaju vrednosti atributa u bazi podataka i da su akcije iskazane u terminima operacija nad bazom. Takođe, moguće je naredbom CREATE TRIGGER referenciranje i starih i novih vrednosti u bazi, a opcija FOR EACH ROW omogućuje obradu cele tabele, a ne samo jedne torke.

INGRES DBMS poseduje komponentu za upravljanje znanjem – INGRES Knowledge Management, koja omogućuje definisanje pravila:

```
CREATE RULE <naziv_pravila>
        AFTER {INSERT |
               DELETE |
               UPDATE [<naziv_kolone>]} [,…]
        [ON | OF | FROM | INTO] <tabela>
        [REFERENCING    [OLD AS <staro_ime>]
                        [NEW AS <novo_ime>]]
        [WHERE <predikat>]
        EXECUTE PROCEDURE <naziv_proc>
        [(<parameter>=<vrednost>[,…])]
```

Najpoznatiji objektno orijentisani sistem za upravljanje bazom podataka, koji podržava Event – Condition – Action pravila je HiPAC (CCA/XAIT, Univ. Wisconsin). Pored njega tu su i: CPLEX (Harvard Univ.), ODE (AT&T Bell Labs). Najpoznatiji predstavnici aktivnih sistema, koji se razvijaju kao nadgradnja relacionih sistema su: POSTGRES (Univ. Ca Berkley) i Starburst (IBM Almaden).

Implementacija standarda za deklarativno definisanje ograničenja zavisi od konkretnog proizvođača sistema za upravljanje bazom podataka. U samom standardu definisana su tri nivoa pokrivenosti standarda od strane konkretnog SRBP-a:

- prvi nivo – *entry SQL*,
- drugi nivo – *intermediate SQL*,
- treći nivo – *full SQL*.

Najveći broj komercijalnih SRBP-a podržava samo prvi nivo implementacije SQL standarda sa nekim elementima drugog ili trećeg nivoa. Sistemi za upravljanje bazom podataka, koji ne podržavaju u potpunosti ni prvi nivo implementacije SQL standarda, deklaraciju uslova integriteta ograničavaju na proveru dužine i tipa podatka određenog obeležja, proveru da li domen obeležja može ili ne može da sadrži nula vrednosti i proveru jedinstvenoti ključa šeme relacije.

S obzirom da većina komercijalnih SRBP-a ne podržava veći deo deklarativne implementacije ograničenja u bazi podataka, to je potrebno omogućiti njihovu proceduralnu specifikaciju (pisanje procedura i funkcija i trigera). Procedure baze podataka su programi, koji se čuvaju u bazi podataka, i oni su po pravilu, kompleksni, kompilirani i optimizovani. Zato je bolje koristiti okidače (trigere). Izvršni deo okidača je korektan blok PL/SQL naredbi sa ugrađenom logikom provere ograničenja. SRBP-i ne čuvaju telo okidača u prevedenom obliku, kao što je to slučaj sa procedurama i funkcijama. Kada se okidač pokrene, SRBP učita deklaraciju okidača, prevede je, smesti u zajedničku radnu memoriju i izvrši.

Procesi i postupci koji se odvijaju u realnom sistemu (dinamika sistema) koji treba predstaviti modelom, informacionim sistemom, se opisuju **modelom procesa**. Model procesa je, u stvari, onaj deo informacionog sistema koji predstavlja skup aplikacija,

programa koji operišu bazom podataka. Projektovanje informacionog sistema nije nimalo jednostavan posao. Pod projektovanjem informacionog sistema ne podrazumeva izrada nekih ad-hoc softvera, kao što je vođenje evldencije video kaseta i korisnika u jednom video klubu, mada i programeri koji su radili ovakve poslove znaju do kakvih problema može doći prilikom definisanja korisničkog zahteva. Vrlo često se po izradi programa utvrdi da "to u stvari nije ono što je korisnik zahtevao?!". U svetu su čak razrađene metode kojima se od korisnika "izvlače" podaci potrebni za definisanje zahteva, pa i realizaciju informacionog sistema. Osim problema koji se odnose na definisanje zahteva, u projektovanju informacionog sistema javlja se i problem složenosti sistema. Da bi problem postao savladiv, mora se razložiti na više jednostavnijih. To se može uraditi na dva načina. Jedan od njih je podela razvoja informacionog sistema na faze.

Drugi način savladavanja složenosti je dekomponovanje sistema u manje podsisteme. I ovo dekomponovanje se može izvršiti na više načina. Posmatrajući preduzeće kao sistem za koji se najčešće i pravi informacioni sistem, dekompoziciju sistema je moguće uraditi na tri načina:

1. Dekompozicijom slstema na osnovu organizacione strukture
2. Funkcionalnom dekompozicijom
3. Objektnom dekompozicijom

Prvi način dekomponovanja sistema se vrši prema već postojećoj organizacionoj strukturi preduzeća. Na ovaj način se pravi veći broj podsistema i to na primer, za svako odeljenje preduzeća ili neku drugu funkcionalnu ili teritorijalnu celinu, zavisno od vrste organizacione strukture. Na žalost, ovakav način projektovanja informacionog sistema je vrlo nestabilan, iz jednog vrlo jednostavnog razloga - organizaciona struktura u preduzeću nije nešto što se projektuje jednom zauvek i nikada više. Organizaciona sturktura preduzeća se, naprotiv, vrlo često menja. U tom slučaju dolazi i do potrebe preprojektovanja celog informacionog sistema.

Bolji način dekomponovanja sistema je onaj po funkcionalnoj osnovi. Funkcije u preduzeću (planiranje, nabavka, proizvodnja (pružanje usluge), realizacija (prodaja gotovog proizvoda), kadrovi, sistem kvaliteta) su znatno stabilnije od njegove organizacione stukture, all ipak dovoljno nestabilne. I one se vrlo često menjaju kao posledica razvoja tehnologije, a pogotovo njihov način izvršavanja. Funkcionalna dekompozicija sistema se odlično uklapa i u koncept klasičnog programiranja jer je celokupan način razmišljanja algoritamski, ali se na taj način, ipak, ne otkriva suština sistema.

Najstabilnija dekompozicija sistema je ona koja je bazirana na objektima u sistemu i njihovim medusobnim vezama. Objekti u sistemu su vrlo stabilni i veze medu njima, kao i operacije koje se nad njima obavljaju su, takode, veoma stabilne. Praktično, menjaju se samo spoljni načini komponovanja, odnosno funkcije u ststemu. Ovaj način dekomponovanja se odlično uklapa i u objektno programiranje koje omogućava (između ostalog) i koncept nasleđivanja kojim se postiže mogućnost ponovnog korlšćenja već projektovanih stvari (reusability) a time se znatno povećava i produktivnost softvera - smanjuje se vreme potrebno za njegovu izradu. Najpoznatlji način za objektno dekomponovanje sistema je ERA model (Entity Relationship Atribute), ili ER model, a kod nas često

pominjan kao MOV (model objekti-veze) ili PMOV (prošireni model objekti-veze) LAZAREVIĆ i sar., 1993.

U sadašnjim trenucima najbolji način razvoja informacionih sistema te softvera, uopšte, je objektno orijentisana transformaciona metoda razvoja, dakle kombinacija operacione, transformacione metode i objektne dekompozicije sistema. Iz svega se može izvesti zaključak da ustvari, oblast baza podataka nije oblast koja postoji sama za sebe, već da je ona deo jedne šire problematike, problematike projektovanja informacionih sistema.

Baza podataka je samo osnova za izgradnju informacionog sistema, drugim rečima, njegov temelj. Koliko je slab temelj, toliko će biti slaba i cela konstrukcija (a možda i slabija). Zato baza podataka mora biti dobro projektovana da bi na osnovu nje moglo da bude izgradeno i sve ostalo. Ukoliko je model podataka korektno projektovan, na osnovu njega se može izgraditi i dobar model procesa, ali ako je model podataka slab, i sve ostalo će biti slabo, bez obzira na to koliko je konačan proizvod šaren, skladno obojen, sa zvučnim efektima ili bez njih, i da li radi u Windows okruženju ili van njega.

2.6.1. MODELI PODATAKA

Da bi se podaci, koji treba da uđu u bazu podataka, a prikupljaju se strukturnom sistemskom analizom realnog sistema, sredili i sistematizovano prikazali na način pogodan za realizaciju baze podataka koristi se informacioni **model podataka**. Model podataka je formalna apstrakcija, putem koje se realni svet preslikava u bazu podataka [MOGIN I SAR., 1996]. Modelom podataka se, preko skupa podataka i njihovih veza, prikazuje stanje realnog sistema u jednom određenom trenutku. Najpoznatiji način za prikazivanje modela podataka je **E-R model** (Entity-Relationship-Atribute) ili kod nas **MOV** (Model Objekti Veze obeležja). Autor je Peter Chen, sa Univerziteta MIT u Bostonu, 1978. godine.

2.6.2. GENERACIJE MODELA PODATAKA

Baza podataka predstavlja strukturu modela podataka, statički model sistema koji se želi predstaviti na računaru, odnosno podatke o objektima u sistemu i vezama između njih.

U istoriji razvoja modela podataka razlikuju se tri generacije modela podataka:

I GENERACIJA: Klasični programski jezici, sa relativno jednostavnim tipovima podataka i siromašnom semantikom. Ne mogu predstaviti realan sistem.

II GENERACIJA: Modeli konvencionalnih sistema za upravljanje bazom podataka – hijerarhijski, mrežni i relacioni. Poseduju znatno bogatiju semantiku od I generacije, strukture podataka i složenije tipove podataka. Nepotpuno opisuju realan sistem.

III GENERACIJA: Semantički bogati modeli – E-R-A ili MOV (Model objekti veze obeležja), SDM (Semantic Data Model) i drugi. Poseduju specifične koncepte za detaljan opis realnog sistema. Nedostatak je u tome što nisu potpuno realizovani na računaru (CASE alati).

OBJEKTNI MODEL PODATAKA – Nastao je početkom 90-tih godina XX veka, kao implementacija struktura podataka iz objektno – orijentisanih programskih jezika (C++, SmallTalk) u sisteme za rukovanje bazama podataka, u oblasti inženjerskog projektovanja (CAD sistemi), ili kao implementacija ugnježdenih tabela u okviru klasičnog relacionog modela podataka ([MOGIN i sar., 1996] kao i [ULMAN i sar., 2002]).

Rešenje koje se danas koristi u projektovanju baze podataka jeste da se specifikacija sistema uradi korišćenjem alata treće generacije (npr. Data Architect u Power Designer-u), a zatim sledi prevođenje u relacioni model koji pripada drugoj generaciji i za koji postoji veliki broj sistema za upravljanje bazom podataka (ORACLE, DB2, INGRES, PROGRESS, SyBase, DBase, Paradox, SQL Server, FoxPro, Access i sl.).

2.6.3. ER MODEL PODATAKA

OSNOVNI ELEMENTI MODELA PODATAKA

1. **Entiteti** – jedinice posmatranja, to su objekti realnog sistema, mogu biti konkretni ali i apstraktni i vezani su za interes korisnika. Entiteti mogu biti: živa bića, predmeti, događaji iz realnog sistema. Entiteti poseduju neka svojstva (osobine, obeležja) i mogu se postaviti u različite odnose sa drugim entitetima, a takođe, moguće je ustanoviti i hijerahiju nasleđivanja (IS-a hijerarhiju) u klasama tipova entiteta.
2. **Atributi** – obeležja, svojstva nekog objekta (entiteta) iste vrste. Vrednosti atributa odnosno obeležja su podaci.
3. Atributi u svakom trenutku imaju određenu, konkretnu vrednost, koju uzimaju iz određenog **domena** – skupa dozvoljenih vrednosti. Atributi se identifikuju na osnovu:
   - zahteva korisnika,
   - dokumentacije.
4. Atributi koji se navode za jednu klasu objekata treba da obezbede da jedan ili više atributa jednoznačno identifikuju svaki objekat u skupu pojava te klase objekata i zove(u) se **ključ** te klase objekata.

   «**Definicija 2.6.** Neka je $P=\{p_i \mid i=1,...,k\}$ skup pojava tipa entiteta $N(A_1,...,A_n)$, a $p[X]$ restrikcija pojave p na obeležje X. Obeležje X predstavlja ***ključ*** tipa entiteta $N(A_1,...,A_n)$, ako, za svaki skup P pojava tipa entiteta N, važe sledeća dva uslova:

   $1°\ (\forall\, p_i,p_j \in P)\ (p_i \neq p_j \Rightarrow p_i[X] \neq p_j[X])$ i
   $2°\ (\forall\, X' \subset X)\ (\neg 1°)$.» [MOGIN i sar., 1996]

   Prvi uslov se zove uslov jedinstvenosti, i kaže da u skupu pojava tipa entieta N ne mogu postojati dve iste pojave tipa entiteta (dve pojave sa istom vrednošću ključa). Drugi uslov se zove uslov minimalnosti, i njegova suština je da ako je ključ tipa entiteta složen tj. sastoji se iz više obeležja, onda nijedno od tih obeležja ne sme na sebe da preuzme ulogu primarnog ključa. Ključ mora biti jednoznačan, nepromenljiv i raspoloživ. Kriterijum za izbor

ključa između više kandidata jeste da je kratak, lak za pamćenje i prihvatljiv od strane korisnika.

5. Za realizaciju baze podataka nije dovoljno samo definisati entitete, atribute i njihove domene, te ključeve, već moramo prikazati i **veze između entiteta**, tj. objekata.

   Osnovni tipovi veze između klasa objekata:
   - **1 : 1 «jedan prema jedan»** - za svaki objekat iz jedne klase postoji samo jedan povezani objekat iz druge klase.
   - **1 : M «jedan prema više»** - svaki objekat iz jedne klase postoji više povezanih objekata iz druge klase, a iz te druge klase se svaki objekat može povezati samo sa jednim objektom ove druge klase.
   - **M : M «više prema više»** - svaki objekat iz jedne klase postoji više povezanih objekata iz druge klase, a iz te druge klase se svaki objekat može povezati opet sa više objekata ove druge klase.

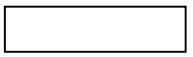
Formalni način prikazivanja entiteta:

E (a_1, a_2, ..., a_n)

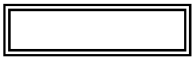
gde je: E – ime klase entiteta (skup svih sličnih objekata je klasa)

a_1, a_2, ..., a_n - atributi

Grafički simboli za prikaz entiteta u modelu objekti – veze – obeležja:

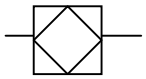


Entitet (jak tip ebjekta)

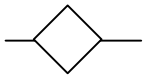


Slab tip objekta

Obeležje (atribut)



Agregacija (veza-objekat)
Poveznik



Veza između entiteta

Jak (fundamentalni) entitet je onaj koji ima odredenu osobinu ili skup osobina koje omogućavaju njegovo jednoznačno identifikovanje. Takva osobina (ili skup osobina) se naziva primarni ključ entiteta.

Slabi objekat je objekat koji nasleđuje ključ objekta s kojim je povezan i proširuje ga nekom svojom osobinom. Podtip je entitet koji nasleduje primarni ključ i sve neključne atribute jakog entiteta s kojim je povezan. U model podataka se uvodi zbog postojanja odredenih specifičnih osobina koje se definišu posebno za svaki uvedeni podtip ili zbog izdvajanja onih osobina koje se ređe koriste.

Agregacija je entitet koji poseduje složeni ključ, sastavljen od ključeva entiteta s kojima je on povezan.

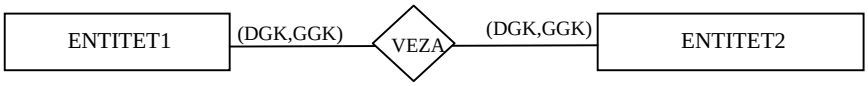
Agregacija (Mešoviti Tip Entitet – Poveznik) uvodi se u slučaju kada je potrebno uspostaviti vezu između veza, odnosno, kada su (ne nužno sve) pojave jednog tipa entiteta poveznika povezane sa pojavama nekog drugog tipa poveznika. Tada se povezani tipovi poveznika prevode u gerunde (odn. mešoviti tip entitet – poveznik). Ovaj koncept ER modela se koristi za modeliranje sledećih situacija kao na primer:

- ako je student upisao neki smer, onda on može prijavti polaganje ispita iz onih predmeta, koji se na tom smeru slušaju po nastavnom planu, ili
- radnik je osposobljen da radi na mašini, na mašini se može proizvesti deo, pa sledi - radnik na mašini, za koju je osposobljen, izrađuje one delove, koji se na toj mašini mogu proizvesti.

Osim svojim nazivom, entiteti se u posmatranom sistemu opisuju i preko svojih atributa.

Atributi su imenovana svojstva entiteta u sistemu. Obično definišu na osnovu pregleda interne i eksterne dokumentacije, upitnika, pregleda postojećeg softvera. Na dijagramima se grafički predstavaljaju kao ovali sa nazivom, koji su povezani s entitetom. Skup vrednosti koje neki atribut može da ima se zove domen atributa. **Domen** može biti unapred predefinisan tipom podatka koji konkretni programski jezik podržava i dužinom podatka, ali može biti i "semantički". Semantički domen je skup dozvoljenih vrednosti i odgovarajućih (potrebnih) ograničenja koje definiše sam projektant.

Grafički simbol za prikazivanje veze između entiteta u modelu objekti-veze:



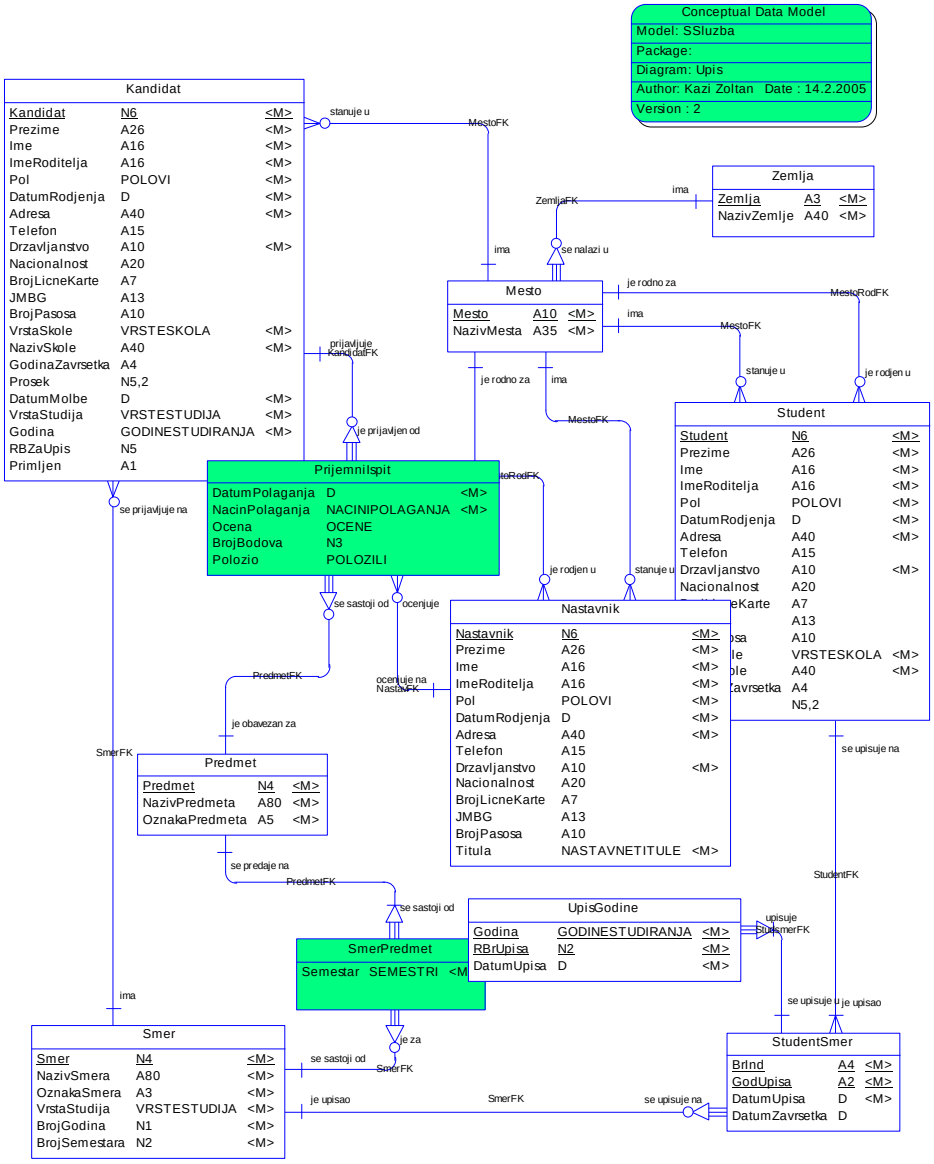
Slika 2.7. Osnovni elementi ER dijagrama

Kardinalnost veze predstavlja broj pojavljivanja entiteta s druge strane veze, s kojim je posmatrani entitet povezan. Prvi broj je najmanje mogući broj pojavljivanja drugog entiteta (tj. donja granica kardinalnosti i označava se sa DGK), a drugi najviši (tj. gornja granica kardinalnosti i označava se sa GGK), pri čemu oznaka "M" predstavlja nepoznat broj pojavljivanja.

Potrebno je uraditi po jedan podmodel za svaki primitivni proces iz strukturne sistem analize i to iz sledećih razloga:

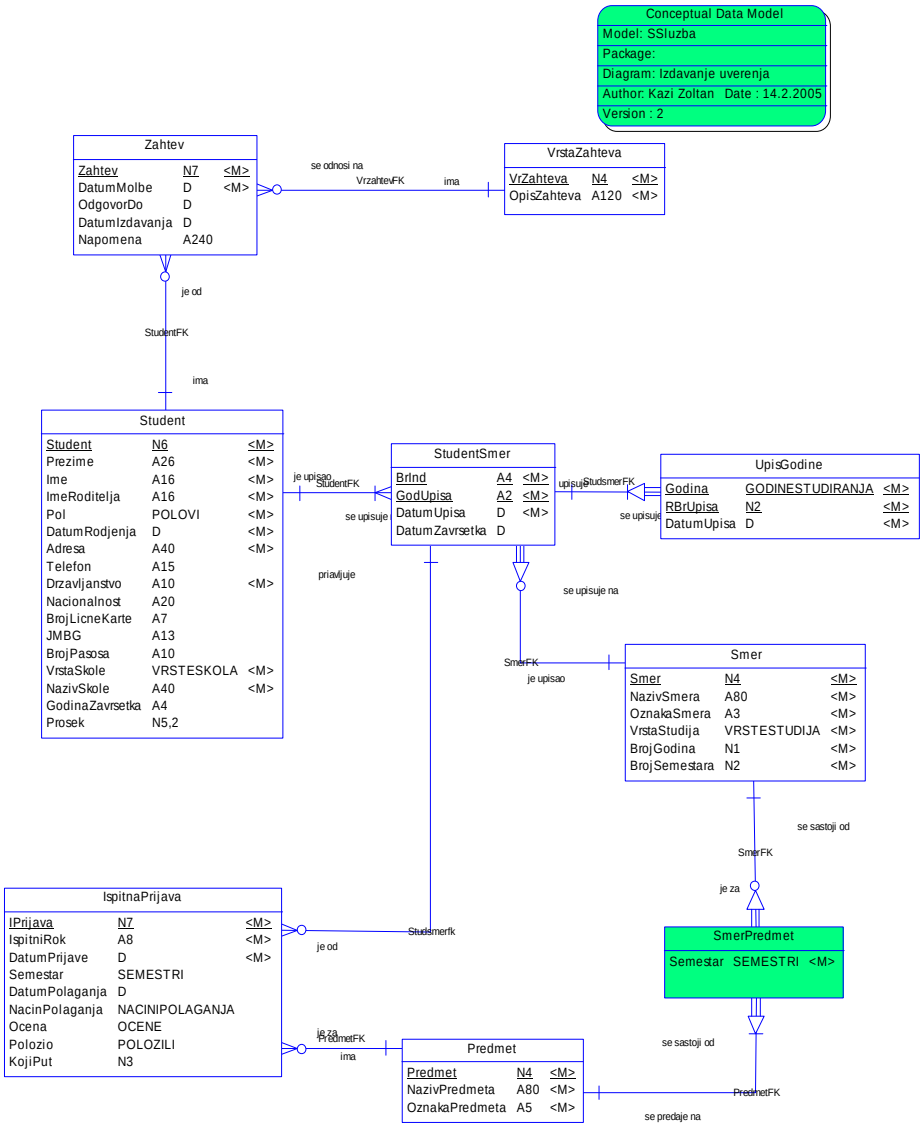
- Kontrola ispravnosti projektovanja globalnog modela podataka koji bi nastao integracijom ovih podmodela.
- Priprema i određivanje potrebnih podataka programeru koji će iskodirati odgovarajuće maske (ekrane) za ažuriranje, prikaz, pregled i pretragu podataka, te ekrane za izveštaje i sl.

E-R MODEL PODATAKA - dijagram "Studentska služba", urađen u CASE alatu Power Designer:



Slika 2.8. ER dijagram u CASE alatu

Primer podmodela za proces "Izdavanje uverenja":



Slika 2.9. Dijagram podmodela ER modela podataka

2.6.4. PRAVILA PROJEKTOVANJA BAZA PODATAKA

OGRANIČENJA

a. Vrednosna ograničenja

- Ograničenja domena - ograničenje elementarnog podatka na određen domen vrednosti.
- Složena ograničenja - dozvoljeni odnosi između vrednosti različitih atributa (npr. starost manja od 30g na može biti direktor).

b. Strukturna ograničenja

- Referencijalni integritet - odnos primarnog i njemu odgovarajućeg spoljnog ključa (mora svakom spoljnom ključu odgovarati jedan slog sa odgovarajućim primarnim ključem).
- Integritet ključa:
  - Not null
  - Unique

2.6.5. PRAVILA PREVOĐENJA ER MODELA U RELACIONI MODEL PODATAKA

Relacija 1:1 se obično prevodi u jednu tabelu koja ih obe sadrži; izuzetak je da su to dve tabele kada je tabela izuzetno opterećena kolonama (poljima) i podacima i kada se jedan deo tabele (kolona) znatno frekventnije koristi od drugih delova. Tada može dve ili više tabela.

Veza M:M se prevodi u 1:M i 1:M sa agregacijom izmedju ili ako veza ima vlastite osobine, onda je to objekat veza ili agregacija. Npr. student-predmet veza je M:M, ali ako bismo prepustili CASE alatu da reši ovu vezu, ne bismo mogli da postavimo vlastite osobine ove veze kao što je datum polaganja i ocena na tom ispitu. Agregacija se realizuje tako što se posebna tabela formira koja sadrži primarne ključeve od obe tabele koje agregira (npr. STUDENT-ISPITI), i ima još neka svoja obeležja, atribute koji su neključna.

Slika 2.10. Primertransformacije veze M:M u okviru dijagrama ER modela podataka

U okviru dijagrama (MOV), definišu se: atributi entiteta, tipovi podataka, odgovarajuća ograničenja.

Po kreiranju MOV-a može se pristupiti prevodenju strukture MOV-a u odgovarajući relacioni model. Za to postoji definisani skup pravila po kojima:

- svaki entitet postaje šema relacije (tj. tabela),
- atributi posmatranog entiteta postaju atributi relacija,
- atributi posmatranog entiteta postaju i identifikatori povezanih entiteta prema kojima je donja i gornja granica kardinalnosti preslikavanja jednaka 1,
- ključ relacije nastale od jakog entiteta je njegov identifikator,
- ključ relacije nastale od slabog entiteta je identifikator nadređenog jakog entiteta proširen sa jednim ili više atributa slabog entiteta,
- ključ relacije nastale od agregacije je složeni ključ koji se sastoji od identifikatora eniteta koji čine agregaciju.

Zahvaljujući savremenim CASE alatima ovaj postupak se obavlja automatizovano, primenom navedenih pravila. Dobijena relaciona šema je potpuno normalizovana. Ona ispunjava sve zahteve tzv. treće normalne forme i sadrži sve integritete koje konkretan sistem za upravljenje bazom podataka (SUBP) podržava, tj. referencijalni integritet, integritet ključa, integritet domena, integritet za operacije nad bazom podataka i sl.

Pre prevođenja E-R (MOV) modela u relacioni, potrebno je uraditi po jedan podmodel za svaki primitivni proces iz Strukturne Sistem Analize i to iz sledećih razloga:

- Kontrola ispravnosti projektovanja globalnog Modela Podataka koji bi nastao integracijom ovih podmodela.
- Priprema i određivanje potrebnih podataka programeru koji će iskodirati odgovarajuće maske (ekrane) za ažuriranje, prikaz, pregled i pretragu podataka, te ekrane za izveštaje i sl.

U fizičkom modelu podataka (Physical Data Model ili PDM) se vrši dodavanje i definisanje alternativnih (semantičkih) ključeva u budućim tabelama, koji treba da obezbede eliminisanje redundanse i nedozvoljenih unosa podataka, a potom i ograničenja nad podacima u vidu Referencijalnog integriteta.

Nakon generisanja relacione šeme pristupa se kreiranju baze podataka u izabranom SUBP, sledi definisanje aplikacije(a) u vidu projekta aplikacije, a zatim i izradi, testiranju primeni programa, odnosno njegovom održavanju.

2.6.6. NORMALNE FORME I NORMALIZACIJA BAZE PODATAKA

Baza podataka je statička slika realnog sistema, odnosno skup podataka o objektima u sistemu i načinu na koji su oni međusobno povezani. Relaciona baza podataka predstavlja takvu bazu podataka u kojoj su podaci organizovani u relacije (odnosno tabele). Međutim, sam način organizovanja podataka u relacije nije tako jednostavan i nikako ne može biti proizvoljan. Pitanje organizacije podataka u relacije je problem logičkog projektiovanja baze podataka. Relacija je, po matematičkoj definiciji podskup skupa Dekartovog proizvoda domena obeleža odnosno skup n-torki: R (a1, a2. .... an) gde su a1, a2. ..., an

atributi relacije. Atributi relacije ne mogu biti proizvoljno izabrani, jer u slučaju njihovog lošeg izbora može doći do velikih problema prilikom implementacije aplikacija za ažuriranje same baze podataka. Da ne bi dolazilo do takvih problema relacije se projektuju saglasno NORMALNIM FORMAMA. Normalne forme predstavljaju pravila o grupisanju atributa u relacije pri čemu se vodi računa o logičkim vezama i zavisnostima izmedu atributa.

Normalizacija podataka je proces podešavanja strukture baze podataka kako bi se izbegle anomalije dodavanja, brisanja i izmene u relacionoj bazi podataka.

Napomena: S obzirom da se projektovanje baze podataka odvija u CASE alatu, Power Designer, u 2 nivoa: logičkom (CONCEPTUAL DATA MODEL – CDM: odgovara E-R-A modelu, tj. MOV-u) i fizičkom (PHYSICAL DATA MODEL – PDM: odgovara relacionom modelu, tj. tabelama buduće baze), normalizacija se može vršiti u relacionom modelu, ali je mnogo praktičnije, poznavajući pravila za prevođenje MOV u relacioni model, izvršiti normalizaciju na logičkom (konceptualnom) nivou. Razlog je u tome što, naknadne izmene u logičkom modelu mogu izazvati velike promene u fizičkom, te se može desiti, da se normalizacija mora vršiti ponovo, na fizičkom nivou.

Prva normalna forma

Svi atributi relacije moraju biti iz prostih domena, tj. domeni svih atributa moraju biti atomarne vrednosti.

Tačnije, atribut relacije ne može biti iz složenog domena, odnosno ne može i on sam biti relacija. Ukoliko je atribut iz složenog domena, više nije reč o relacionom modelu, već o objektnom. Ako je ovaj uslov zadovoljen, relacija je u PRVOJ NORMALNOJ FORMI.

Međutim, objekti u realnom sistemu vrlo često imaju atribute iz složenih domena. Tačnije, vrlo često se kao atribut pojavljuje grupa sa ponavljanjem, odnosno niz nekih elementa. Reprezentativni primer je faktura koja sadrži neke osnovne atribute, kao što su: datum, kupac, mesto, adresa, telefon (i druge atribute koji opisuju fakturu u celini) i nekoliko stavki fakture koje se sastoje iz atributa kao što su: artikl, količina, jedinica mere, jedinična cena, ukupna cena. Stavke fakture se ponavljaju više puta, a osnovni atributi samo jednom. Ovakve tabele su nenormalizovane. Faktura, npr. se, ovakva kakva je, ne može predstaviti relacionim modelom. Rešenje se nalazi u ponavljanju osnovnih atributa za svaki red grupe sa ponavljanjem. Tako bi se faktura predstavila relacijom koja bi imala sve pomenute atribute. Osnovni atributi bi se nepotrebno ponavljali, ali bi se u ovom slučaju faktura bar mogla predstaviti relacijom. Rešenje nije optimalno, all zato postoje druga, treća, ... normalna forma.

Druga normalna forma

Dok se prva normalna forma bavi definicijom relacije i domenima atributa, druga, a i sve ostale normalne forme, se bavi odnosom i međuzavisnostima atributa, tj. zavisnostima atributa.

Primer: Ako postoji relacija (u MOV-u entitet) STUDENT (BRINDEKSA, IME, POSTA, MESTO, ADRESA, PREDMET, OCENA) iz nje možemo izdvojiti za primer dve funkcionalne zavisnosti:

Atribut BRINDEKSA određuje atribut IME, a složeni atribut BRINDEKSA, POSTA određuje atribut ADRESA. Naravno, ove dve zavisnosti nisu jedine u ovoj relaciji, ali su samo ove dve uzete za primer. Prva zavisnost je potpuna funkcionalna zavisnost, jer atribut IME zavisi od atributa BRINDEKSA, ali ne i od njegovih delova (jer je atribut BRINDEKSA prost). Iz ovoga se može izvesti zaključak da su sve funkcionalne zavisnosti od prostog atributa ujedno i potpune funkcionalne zavisnosti. Sa druge strane, zavisnost atributa ADRESA nije potpuno funkcionaina, jer atribut ADRESA zavisi funkcionalno i od samog atributa BROJINDEKSA koji je deo složenog atributa BRINDEKSA, POSTA. Na osnovu potpune funkcionalne zavisnosti definiše se i druga normalna forma. Relacija je u drugoj normalnoj formi ako je u prvoj i ako su svi njeni neključni atributi potpuno tunkcionalno zavisni od primarnog ključa te relacije.

Da li relacija STUDENT zadovoljava uslove postavljene definicijom druge normalne forme? U ovoj relaciji samo složeni atribut BRINDEKSA, PREDMET odreduje sve ostale atribute u relaciji. Dakle, on igra ulogu primarnog ključa, ali nisu svi atributi POTPUNO funkcionalno zavisni od njega. Atributi IME, POSTA, MESTO i ADRESA su zavisni samo od aiributa BRINDEKSA, koji je deo primarnog ključa. U čemu je rešenje? Relacija koja nije u drugoj normalnoj formi (ali je u prvoj) se dekomponuje u dve ili više relacija koje jesu u drugoj normalnoj formi, iz kojih se operacijom prirodnog spajanja može rekonstruisati početna relacija, i to tako da se sve nepotpune funkcionalne zavisnosti odvoje u posebne relacije. Tako bi se, primenom definicije druge normalne forme, relacija STUDENT dekomponovala u relacije:

STUDENT (BRINDEKSA, IME, POSTA, MESTO, ADRESA)
OCENA (BRINDEKSA, PREDMET, OCENA)

Ove ***dve relacije jesu u drugoj normalnoj formi jer su svi atributi u njima potpuno funkcionalno zavisni od svojih primarnih ključeva.***

Relacija je sigurno u drugoj normalnoj formi, ako je kandidat za njen primarni ključ prosto (elementarno) obeležje (na primer, JMBG, ili PTT, ili SIF_RADNIKA).
Relacije koje su u drugoj normalnoj formi još uvek nisu optimalne i kod njih još uvek mogu postojati anomalije ažuriranja (unos, brisanje i izmena) podataka.

Treća normalna forma

Konkretan primer se može videti u jednom od prethodnih redova. Primenom definicije druge normalne forme, dobijena je relacija STUDENT (BRINDEKSA, IME, POSTA, MESTO, ADRESA). Atribut BRINDEKSA je primarni ključ relacije STUDENT i, logično, funkcionalno određuje sve ostale atribute u relaciji. Međutim, ovaj atribut nije jedini koji određuje nešto u relaciji. Atribut POSTA funkcionalno odreduje atribut MESTO.

Treća normalna forma ne dozvoljava tranzitivne zavsnosti. Šta je loše u tranzitivnoj

zavisnosti? Ukoliko postoje tranzitivne zavisnosti, postoje i anomalije u ažuriranju. Na primer, zamislite se i odgovorite na pitanje: "Šta je potrebno uraditi da bi se u bazu podataka uneo podatak o nekom mestu?". Podaci o mestu, u ovom slučaju, su zavisni od podataka o studentu, što ne bi smelo da se desi. Dakle, ***relacija je u trećoj normalnoj formi, ako je u drugoj normalnoj formi i u njoj ne postoje tranzitivne zavisnosti neključnih atributa od ključa (primarnog ili alternativnog).***

Relacija koja nije u trećoj normalnoj formi (all jeste u drugoj) se dekomponuje u relacije koje jesu u trećoj normalnoj formi, a iz kojih se operacijom prirodnog spajanja može dobiti polazna relacija, i to tako da se odvoje tranzitivne zavisnosti. Tako bi se pomenuta relaclja STUDENT (BRINDEKSA, IME, POSTA, MESTO, ADRESA) dekomponovala na relacije: STUDENT (BRINDEKSA, IME, POSTA, ADRESA) i MESTO (POSTA, MESTO). Ove dve relacije jesu u trećoj normalnoj formi i iz njih se operacijom prirodnog spajanja preko atributa POSTA može rekonstruisati polazna relacija. Primena definicije treće normalne forme i normalizacija relacija saglasno njoj nije kraj - iako treća normalna forma ne dozvoljava tranzitivne zavisnosti neključnih atributa od ključa. Tranzitivne zavisnosti se i dalje mogu pojaviti izmedu samih ključnih atributa. Zbog toga se uvodi pojam Boye-Codd –ove normalne forme, ali o ovoj normalno formi detaljnije videti u [MOGIN i sar., 2000].

U prethodnom tekstu opisana je metoda normalizacije – **vertikalna dekompozicija**. Ona vertikalno deli relacije na osnovu operacije relacione algebre – projekcije. Da bi se očuvao princip dekompozicije bez gubitka informacije uvode se i pojmovi 4 Normalne Forme, koja se zasniva na višeznačnim zavisnostima atributa u šemi relacije, kao i 5 Normalne Forme, koja se, upravo, zasniva na zavisnostima spoja. Zavisnost spoja, kao što samo ime kaže, obezbeđuje rekonstrukciju polazne relacije na osnovu spoja dobijenih dekomponovanih relacija.

Osim metode dekompozicije, postoji i **metoda sinteze** [MOGIN i sar., 2000]. Ova metoda se zasniva na generisanju skupa šema relacija počev od polaznog skupa obeležja i skupa ograničenja (zadatog skupa zavisnosti: funkcionalnih, zavisnosti spoja, višeznačnih zavisnosti, koje zadaje projektant informacionog sistema). U svakom narednom koraku ovog postupka, šeme relacija se »sintetizuju« u odnosu na postavljena ograničenja.

2.6.7. OGRANIČENJA NAD BAZOM PODATAKA

Baza podataka mora u svakom trenutku zadovoljavati određeni skup zadatih uslova integriteta, odnosno ograničenja. Ta ograničenja su posledica: zakonske regulative u datoj oblasti, internih pravila poslovanja, uobičajenih normi ponašanja i slično. Ukoliko su svi uslovi ispunjeni, tada je baza podataka u konzistentnom stanju. Drugim rečima, za bazu podataka, koja je usaglašena sa promenama u realnom sistemu kaže se da je konzistentna [MOGIN i sar., 1996].

Osim strukture, model podataka poseduje i neka ograničenja, kao i operacije koje se vrše u slučaju narušavanja istih, da bi se baza podataka vratila u konzistentno, ispravno stanje.

Ograničenja modela podataka se mogu podeliti u dve velike grupe:

- vrednosna ograničenja,
- stukturna ograničenja.

Pod **vrednosnim ograničenjima** se podrazumevaju jednostavna ograničenja (ograničenja domena) i složena vrednosna ograničenja. Ograničenja domena su ona koja se tiču skupa (domena) iz kojih atribut uzima vrednosti. Na primer, atribut OCENA može uzimati vrednosti iz zatvorenog skupa celih brojeva (5, 6, 7, 8, 9, 10). Atribut BRINDEKSA može biti definisan ka broj (koji označava redni broj studenta u godini u kojoj se upisao na fakultet) iza koga sledi znak /, godina upisa fakulteta, a potom još jedan dvocigren broj (koji označava smer i status studenta). Tako bi podaci: 123/90-011, 101/97-022 bili regularni brojevi indeksa, a podaci kao: PERA/93-021, 12C/90-POT nikako ne bi smeli biti uneti u bazu podataka. Ova ograničenja su jednostavna i moguće je napraviti mehanizme provere za svaki od njih. Na žalost, većina relacionih ststema za upravljanje bazom podataka nemaju implementirane ovakve mehanizme. Složena vrednosna ograničenja se tiču odnosa između vrednosti različitlh atributa. Na primer, u modelu podataka može postojati ograničenje da ukoliko radnik ima manje od 30 godina, onda njegova funkcija u preduzeću ne može biti diirektor, takođe, ukoliko ukupna vrednost računa prelazi odredenu sumu (na primer 2000 dinara), onda ona mora biti odobrena od strane finansijskog dfrektora. Primera za složena vrednosna ograničenja ima mnogo u realnim sistemima i ona se nikako ne mogu izbeći. Ali, u klasičnim relacionim sistemima za upravljanje bazom podataka, ova ograničenja se ne mogu predstavlti bazom podataka. Kada bi bila jedina, vrednosna ograni čenja i ne bi predstavljala veliki problem.

Osim vrednosnih postoje i strukturna ograničenja. Ova ograničenja se odnose na veze između objekata u sistemu. Na primer, svaki radnik u preduzeću mora raditi u jednom i samo jednom odeljenju, a odeljenje može imati 0, 1 ili više radnika. Faktura mora imati bar jednu stavku, a stavka mora imati samo jednu fakturu kojoj pripada.

Kod sturkturnih ograničenja obavezno je pomenuti dva pravila integriteta:

- integritet entiteta (integritet ključa),
- referencijalni integritet.

Integritet ključa je pravilo po kome primarni ključ relacije obavezno mora imati vrednost, što znači da ne može biti definisan kao polje u koje se može, ali i ne mora upisati vrednost. Dakle, primarni ključ uvek mora imati konkretnu vrednost, i ta vrednost mora biti jedinstvena u posmatranoj relaciji. Znači da se u relaciji STUDENT (BRINDEKSA, IME itd.) ne sme pojaviti više od jedne n-torke sa određenom vrednošću primarnog ključa (u ovom slučaju je to atribut BRINDEKSA).

Referencijalni integritet se tiče odnosa izmedu spoljnog ključa u relaciji i primarnog ključa u nekoj drugoj relaciji, a koji uzima vrednosti iz istog domena kao i posmatrani spoljni ključ. Ovo pravilo integriteta nalaže da se svaka vrednost spoljnog ključa u tabeli mora pojaviti i kao vrednost primarnog ključa u tabeli gde je odredeni atribut primarni ključ. Ako, na primer, postoje relacije RADNIK (SIFRAD, IME, SIFODEL) i ODELJENJE (SIFODEL, NAZIV), onda u relaciji radnik svaka šifra odeljenja (SIFODEL) mora imati pojavljivanje u relaciji ODELJENJE, u atributu SIFODEL. Drugim rečima, radnik ne

može raditi u nepostojećem odeljenju. Ukoliko je za radika uneta šifra odeljenja 1234, onda u relaciji ODELJENJE mora postojati odeljenje sa istom šifrom. To znači, da se pri unosu novih podataka mora voditi računa o vrednosti spoljnog ključa, odnosno, mora se paziti da se ne ubaci radnik koji radi u nepostojećem odeljenju. Ali, to nije dovoljno. Pri brisanju podataka iz relacije ODELJENJE, mora se voditi računa o tome da se ne obriše odeljenje u kome rade neki radnici, jer će isti ostati vezani za nepostojeće odeljenje (jer je obrisano).

Jedan atribut, ill grupa njih mora igrati ulogu identifikatora entiteta, tj. buduće šeme relacije tabele u bazi podataka. Ovaj identifikator obezbeđuje jedinstvenost reda tabele i ni u jednom redu tabele njegova vrednost ne sme bitl izostavljena. ldentifikator relacije se drugačije zove KLJUČ RELACIJE i predstavlja prost ili složen atribut koji poseduje osobine jedinstvenosti i neredundantnosti. Osobina jedinstvenosti se odnosi na činjenicu da u relaciji (tabeli) ne smeju postojati dve iste n-torke (reda). To je omogućeno upravo uvođenjem ključa, čija se vrednost u relaciji pojavljuje samo jednom, pa se samim tim i svi redovi razlikuju, ako ni po čemu drugom, onda bar po vrednostima ključa. Osobina neredundantnosti podrazumeva da nijedan atribut iz grupe atributa ne sme da se izostavi, a da ova grupa ne izgubi osobinu jedinstvenosti. Prema tome, ključ relacije je najmanja moguća kombinacija atributa u relaciji koja poseduje osobinu jedinstvenosti. Složeni atribut (grupa atributa) koja poseduje osobinu jedinstvenosti, ali ne i osobinu neredundantnosti se naziva SUPERKLJUČ relacije.

U relaciji može postojati više atributa koji zadovoljavaju osobine ključa. Svi oni se zovu KANDIDATI ZA KLJUČ. Atribut koji se izabere da bude ključ relacije se naziva PRIMARNI KLJUČ, a ostali kandidati se nazivaju ALTERNATIVNI KLJUČEVI. SPOLJNI KLJUČ relacije je atribut koji u posmatranoj relaciji ne igra ulogu primarnog ključa, ali je zato u nekoj drugoj relaciji primarni ključ.

2.6.8. HEURISTIČKA UPUTSTVA ZA PROJEKTOVANJE BAZE PODATAKA

CILJEVI PROJEKTOVANJA BAZE PODATAKA:

1. Minimum redundanse.
2. Optimalno prema programskoj podršci - minimum utroška vremena za izdvajanje podataka i memorije. Funkcionalnost.
3. Optimalno prema promenama u vremenu - pogodnost kasnijeg održavanja i intervencija. Fleksibilnost. Sistem treba da živi u vremenu uz minimalne ili nikakve intervencije nad bazom.
4. Ugrađivanje semantičkih odnosa realnog sveta - adekvatnost.
5. Ugrađivanje ograničenja (integriteta) u bazu podataka odnosno, ako je implementirano u samom SRBP-u (Sistem za rukovanje bazom podataka - DBMS), koristiti što više DEKLARATIVNOST SQL naredbi, ili mehanizme zaštite integriteta baze podataka (integritet domena, integritet torke relacije, referencijalni integritet).
   - da baza podataka "sama sebe" štiti od grešaka (unosa), a ne da zavisi od programa,
   - ugrađivanje lokalnih pravila u vidu kardinaliteta ili uopšte u odnosima, tj. ugrađivanje poslovnih pravila (loklanih pravila koja važe u određenoj organizaciji).

6. Potpunost modela podataka, tako da pokriva podacima sve procese koji su definisani u SSA, a pre svega primitivne procese, tj, da su sve informacione potrebe pokrivene, tj. svaki elementarni podatak iz SSA da ima adekvatan atribut u MOV, tj CDM-u. Dakle, pokrivenost potreba realnog sistema.
7. Smanjenje broja tabela, radi lakše manipulacije i fleksibilnijeg rešenja.
8. Uvođenje "reda", zbog anomalija ažuriranja.

MATERIJAL KOJI KORISTIMO:

1. TEKSTUALNI OPIS POSLOVA U SISTEMU.
2. MODEL PROCESA STRUKTURNE SISTEM ANALIZE - DTP i RP.
   a. Elementarni podaci - na osnovu njih dobijamo atribute za entitete.
   b. Odnosi elementarnih podataka (sintaksa pomoću zagrada) - elementarni podaci koji se javljaju unutar <> najčešće pripadaju jednom entitetu, {} govori o ponavljanju atributa, [..] IS-A, /../ moguće null vrednosti.
   c. Skladišta - dekomponuju se na tabele.
   d. Primitivni procesi - osnov za podšeme (podmodele) baze podataka.
   e. Poslovna pravila - postaju tabele sa parametrima relacionog modela ili se koriste deduktivni sistemi u sprezi sa relacionim bazama podataka.
   f. Domeni - postaju tzv. šifarničke tabele baze, odnosno zasebni entiteti ER modela.

POSTUPAK nije linearan. Opisan je kao niz koraka koje treba uraditi jednom ili više puta u iteracijama:

1. Kreiranje preliminarnog skupa entiteta i dodeljivanje atributa

a. Transformacija elementarnih podataka koji su preuzeti iz SSA, ukoliko u SSA nije izvršena korekcija:

- Elementarni podaci koji su množina postaju jednina.
- Elementarni podaci koji se nazivaju kao struktura i mogu se dalje dekomponovati na elementarne podatke, vrši se kreiranje novih elementarnih podataka koji čine tu strukturu.
- Nepotpuno imenovani elementarni podaci se dopunjuju dovoljno preciznim imenom.
- Neophodno je imati, za nedovoljno jasno imenovane elementarne podatke detaljniji opis (npr. u Annotation delu u CASE alatu), kako bi se potpuno razumelo značenje kako bi se moglo pravilno odrediti njegovo mesto.

b. Primena heuristika ([MOGIN i sar., 1996], str. 51):

ANALIZA TEKSTA OPISA POSLA: Imenice (subjekti, objekti) iz opisa posla postaju kandidati za entitete.

ČIJI ELEMENTARNI PODATAK: Na osnovu imena elementarnog podatka, određuje se koji je entitet kojem pripada, pa se kreira odgovarajući entitet i njemu dodeljuje ovaj i ostali atributi koji mu odgovaraju.

DEKOMPOZICIJA SKLADIŠTA PODATAKA: Sva skladišta podataka se preuzimaju iz DTP-a (SSA) i vrši se njihova dekompozicija na entitete, tako što se uočavaju odnosi

između elementarnih podataka na nivou svakog skladišta ponaosob (brojni odnosi, odnosi međuzavisnosti) i koriste se odnosi izraženi u navođenju strukture skladišta pomoću zagrada.

2. Transformacija preliminarnog skupa entiteta i atributa

a. Eliminisanje redundanse - atributi koji se ponavljaju u više entiteta se izdvajaju u posebnu tabelu. Jedan elementarni podatak bi trebalo da se memoriše samo u jednoj tabeli.

b. Transformacija elementarnih podataka

- Svaki elementarni podatak koji je preuzet iz SSA treba da bude pokriven nekim oblikom elementarnog podatka. Mogu biti uvedeni novi elementarni podaci koji pokrivaju postojeće importovane elementarne podatke i predstavljaju njegovo uopštenje.
- Izračunljivi elementarni podaci se (uglavnom) ne memorišu, već se od ostalih elementarnih podataka traži da li se može pokriti njihovo izračunavanje, ukoliko je postupak izračunavanja trivijalan. Ako takvi elementarni podaci ne postoje, dodaju se elementarni podaci koji su sirovi za ovaj izračunljivi podatak.
  Primeri: prosečna ocena studenta, prosečna plata radnika, broj studenata po smerovima.
- Rešavanje problema sinonima i homonima - potrebno je da ostane samo jedan termin za istovetno značenje. HOMONIMI - isti naziv a različito značenje (semantika). Na primer, Obeležje DATUM, može da znači Datum rođenja Radnika, ali i Datum zaposlenja Radnika, pa jedan naziv DATUM ima različito značenje.

c. Dodavanje elementarnih podataka

- Dodavanje identifikacionih obeležja - Korisno je da se identifikaciono obeležje zove ID_xxx, ili samo XXX, npr, Id_pacijenta, ili samo Pacijent (posebno je pogodno pri prevođenju ER modela u relacioni i migraciji ovih polja u drugu tabelu). Zaista, to polje je u toj tabeli predstavnik pacijenta, tj, njegov zastupnik, pa je lakše zbog razumevanja da nosi takav naziv.
- Dodavanje još nekih atributa za određene entitete, ukoliko se ukaže potreba, a nije definisana u SSA.

d. Apstrakcija

Uopštavanje više sličnih entiteta u jedan opšti. Pri tome se kreiraju novi, opšti atributi. Ovi konkretni slučajevi postaju vrste tog entiteta i kao takvi se javljaju u konkretnim slogovima, kao pojavni oblici. Potrebno je dodati još jednu kolonu u tom opštem entitetu, kojim se omogućuje razlikovanje objekata različite vrste.

Mora se voditi računa o tome da se tim uopštavanjem ne izgubi neki potreban podatak. *Primer*: ne mogu se svi datumi uopštiti u jedan elementarni podatak DATUM, jer nije svejedno da li je u pitanju DATUM IZDAVANJA ili DATUM VRACANJA u nekoj biblioteci.

Uopštavanje (generalizacija) se uglavnom vrši na nivou više entiteta. Može se vršiti i na nivou atributa, u situacijama kada je dozvoljeno i omogućeno da se svi pojedinačni slučajevi adekvatno pokriju. Npr. OCENA_MATEMATIKA i OCENA_SRPSKI se MORA apstrahovati u OCENA, NAZIV_PREDMETA i sl. kako bismo dobili fleksibilnije rešenje.

Postoji više REŠENJA kao projekta baze podataka, u zavisnosti od NIVOA APSTRAKCIJE.

e. Uočavanje odnosa (brojnih npr. 1:M i identifikacionih (funkcionalnih) zavisnosti) između elementarnih podataka i izdvajanje grupa elementarnih podataka iz entiteta u poseban entitet:

- uočavanje odnosa prema identifikacionom obeležju
- uočavanje međusobnih odnosa elementarnih podataka

f-a. Uočavanje i imenovanje odnosa (relacije- *Relationship*) između entiteta:

- odnosi celina - deo (npr. zaglavlje - stavke),
- odnosi opšte - specifično,
- odnosi stalno - promenljivo,
- odnosi statično- dinamično,
- odnosi potencijalno - stvarno,
- odnosi prošlo - sadašnje- buduće,
- odnosi jedan -više,
- odnosi pre- posle,
- odnosi sirovo- izračunljivo,
- stanja jednog objekta se menjaju u vremenu.

Bitno je: Uloge jednog entiteta se ostvaruju relacijama ka drugim entitetima.
Primer: Može se na jednom mestu memorisati tabela *Mesto*, a ona ima ulogu mesta rođenja, mesta stanovanja, mesta kao sedišta neke firme itd. i sve to u istoj tabeli. Njene uloge se izražavaju nazivima relacija: je rođen, stanuje ...

f-b. Izdvajanje tipova entiteta:
Razlikovati:

- Entitet koji opisuje dokument.
- Entitet koji opisuje činjenice o nekom objektu ili događaju.

Naš je cilj fleksibilno rešenje, tako da ne treba da težimo da memorišemo podatke pod nazivima dokumenata, jer je to promenljiva kategorija. Bolje je memorisati podatke o činjenicama o događajima ili objektima, (posebno se realizuju tzv. document management sistemi, pa i oni mogu biti kroz bazu podataka uključeni o model baze svakog informacionog sistema).
Razlikovati:

- tabelu šifarnik (osnovni podaci, matični podaci),
- tabelu znanje i pravila,
- tabelu povezivač,
- tabelu koja opisuje osnovnu dinamiku redovnog rada (pokriva obradu, tj. primitivne procese iz SSA) - dakle događaje i objekte obrade.

g. Kreiranje šifarnika

- Transformacija nabrojivog domena u šifarničku tabelu.
- Izdvajanje elementarnog podatka koji ima nabrojivi domen vrednosti koji se pri unosu često ponavlja - elementarni podatak koji ima nabrojiv skup vrednosti koji se

često unosi u originalnoj tabeli se izdvaja u posebnu tabelu šifarnik - memoriše se šifra i naziv te vrste podatka. Unosi se samo jednom, a koristi više puta, kad god treba, pozivom preko šifre se memoriše, tako da ukoliko se podatak iz šifarnika menja, menja se automatski svako njegovo pojavljivanje u tabelama gde je njegova šifra strani ključ.

- § Jedan podatak je opšti i koristiće se na više mesta, ako se izdvoji kao posebna tabela. *Primer*: tabela MESTO nastaje na osnovu polja: mesto rođenja i mesto stanovanja.

h. Određivanje kardinaliteta relacije

- § određivanje kardinaliteta svakog od preslikavanja.
- § voditi računa da se određivanje vrši u 2 koraka:
  1. Odnosi između elemenata realnog sistema (veće ograničavanje),
  2. Odnosi između tabela u bazi podataka (manje ograničavanje, ukoliko je moguće).

i. Određivanje primarnog ključa za svaku buduću tabelu BP

Kriterijum za izbor primarnog ključa (PK) među kandidatima za primarni ključ:

- § Jednoznačnost
- § Potpunost
- § Minimalnost

Postoje 3 varijante:

1. Prost brojač koji se sastoji od jednog polja za svaki entitet. Služi samo za ostvarivanje relacija u prevođenju u relacioni model (jednostavnije je da se prenosi samo jedno polje, a ne složeni primarni ključ). Ukoliko se izabere ovaj oblik PK, obavezno je kasnije u PDM-u (Power Designer Data Arhitect format modela relacione baze podataka) da se za svaku tabelu odredi *unique index* kojim se definiču alternativni ključevi, tj. kombinacije sadržinskih (neključnih) polja koje se ne smeju ponoviti.
2. Složeni primarni ključ (najčešće se sastoji od polja koja nose neko značenje). Obezbeđuje semantičku jednostvenost sloga, a nedostatak je složenost pri migraciji polja.
3. Tzv. govoreće šifre - jedan elementarni podatak čiji delovi imaju značenje. Npr. Broj indeksa ili JMBG.

j. Određivanje identifikacionih zavisnosti između entiteta. Određivanje tipova entiteta.
Identifikaciona zavisnost određuje tipove entiteta: Jak, Slab, Agregacija (Gerund).

3. Završna optimizacija

Vrši se u odnosu na:

- § performanse celokupnog sistema, zajedno sa programom. Npr. dozvoljavaju se izračunljiva polja, ukoliko se time postiže skraćenje vremena odziva sistema, na zadatu (SQL) komandu.
- § lakoću održavanja. Napraviti takvu organizaciju tabela koja zahteva kasnije što manje popravke i intervencije. Npr. sve što je moguće se uopštava i sažima, tako da se dobija fleksibilno i upravljivo rešenje.

Primeri:

- Mogu se podaci o ulazu ili izlazu neke materije objediniti da se čuvaju u jednoj tabeli, sa bulovim poljem (.T. ili .F.) koji opisuje da li je ulaz ili izlaz.

- Mogu se svi šifarnici objediniti da se čuvaju u jednoj tabeli. Potrebna je pored šifre i naziva još i kolona tip šifarnika.

4. Kreiranje podmodela

Kreiranje podmodela se radi kao izdvajanje podšeme celokupne šeme baze podataka, i to za svaki od primitivnih procesa se radi po jedna podšema baze podataka.
Podmodeli se rade iz 2 razloga:

- § Proveravanje da li su svi primitivni procesi pokriveni podacima (dakle, pomoću tih entiteta, tj. tabela će biti realizovan taj proces, po pitanju unosa i pregleda podataka).
- § Da li su na podmodelu svi entiteti međusobno povezani. Ako nisu, povezani su preko nekog drugog, pa i njega treba uključiti u podmodel, ili da se doda direktna veza između entiteta na celokupnoj šemi, kako ne i neki entitet ostao nepovezan.

5. Kreiranje fizičkog modela podataka (PDM, odgovara relacionom modelu)

Fizički model se u CASE alatu kreira automatski. U suštini se svodi na prevođenje MOV u RM, pri čemu se RM koji se dobije dodaju osobine koje su vezane za konkretni SUBP za koji se uradio fizički model. Kada je kreiran, proverava se:

- o Da li su elementarni podaci odgovarajućeg tipa podatka? Npr. voditi računa da neko polje ne postane memo, a trebalo bi da je string!
- o Da li je dobro izvršena migracija primarnog ključa? Da li je dobijen željeni relacioni model.
- o Da li elementarni podaci imaju jasni naziv? Preimenovanje elementarnih podataka u jasnije i potpunije nazive.
- o Da li primarni ključ obezbeđuje semantičku jedinstvenost podataka u datoj tabeli? Najčešće je primarni ključ "običan" brojač i ne obezbeđuje očuvanje semantičke jedinstvenosti sloga, u smislu da se može pod nekim drugim brojem uneti dati slog, tj, ostali neključni podaci. Da bi se to sprečilo, radi se kreiranje ***Unique index-a*** nad svakom od tabela, gde je to moguće!
- o Da li je obezbeđen integritet podataka, pre svega referencijalni integritet putem pravila za očuvanje referencijalnog integriteta?

POSEBNI OBLICI OBJEKATA I ODNOSA:

1. AGREGACIJA
- objekat-veza: kada veza ima vlastite atribute
- realizuje vezu M:M. Kada je već tako, zapitajte se da li možda takva veza možda ima vlastita obeležja.

2. ŠIFARNIK - Kada se neki podaci koriste u više procesa, oni se jedinstveno izdvajaju i povezuju sa okolnim tabelama sa onoliko relacija, koliko uloga ta tabela ima.

3. ODNOS 1:1 - Kada se radi o tabelama 1:1, treba se zapitati da li je to moguće objediniti u jednu tabelu. Najčešće se to preporučuje, osim u nekim slučajevima, npr:

- Kada je potrebno razlikovati potencijalno i stvarno, što je u odnosima 1:1 (Primer:. zahtev za članstvom i podaci o članu)
- Kada ne treba opterećivati jednu tabelu koja bi se češće koristila u odnosu na ovu drugu, čiji se podaci ređe koriste.
- Kada se jedan isti slog posećuje 2 ili više puta, i edituje.
- Jedna je tabela, kada se u suštini podaci unose u jednom trenutku (Primer: podaci o poslu i ugovoru).
- Pri situaciji kada su 2 tabele u odnosu 1:1, treba izabrati onu koja će drugoj predati primarni ključ (dominantna), kada se prevodi u relacioni model. To je ona tabela koja je "starija u vremenu", tj. tabela u koju se najpre unose podaci, a zaim tek u ovu drugu tabelu.

4. ODNOS 1:M - Primer: odnos celina-deo i zaglavlje-stavke.

5. Ne izdvajati šifarnik - Kada nije relevantno za dati sistem i mala je verovatnoća da će se pojaviti ista vrednost elementarnog podatka, tj. ima isuviše mnogo vrednosti elementarnog podatka. Primer: Naziv škole u tabeli sa podacima za kandidata za upis na Fakultet, na prijemnom ispitu. Moglo bi se posebno čuvati tabela ŠKOLA, gde bi se čuvala šifra, naziv, adresa itd, ali to npr. za Informacioni sistem Fakulteta nije relevantno i to pogotovo ako se retko (mala je verovatnoća ponavljanja) javljaju Kandidati iz iste škole. Dovoljno je svakome Kandidatu upisati u originalnoj tabeli u polju naziv škole odgovarajuću vrednost. Međutim, ukoliko bi bila veća verovatnoća ponavljanja, onda se izvlači napolje u poseban šifarnički entitet. Takođe, ukoliko je to polje po kojem se često pretražuje i unapred ima definisan moguć i dozvoljen skup vrednosti, izdvaja se u poseban šifarnik.

6. Potreba da se pamte stanja i promene u vremenu zahtevaju posebne tabele, koje su prema originalnoj u odnosu 1:M. Primer: Roba - Cenovnik, a ne da se polje cena nalazi u tabeli roba.

OBRATITI PAŽNJU (ČESTO SE GREŠI):

- Uočavanje elementarnih podataka koji su imenovane strukture.
- Uočavanje izračunljivih podataka.
- Određivanje primarnog ključa. Određivanje naziva PK.
- Veze 1:M izmedju elementarnih podataka.
- Uočavanje skupa vrednosti za elementarne podatke i potrebe za domenom. Domen postaje entitet ili ne, izdvajanje šifarnika i tabela za praćenje svakodnevne dinamike rada (evidencija objekata i događaja realnog sistema).
- Određivanje za preslikavanja u relacijama kardinaliteta i identifikacione zavisnosti.
- Odredjivanje koji se elementarni podaci nalaze u istom entitetu, a koji se izmeštaju u poseban entitet.
- Potpunost opisivanja entiteta elementarnim podacima.
- Iz naziva elementarnog podatka se ne može saznati njegovo značenje.
- Dobro uočiti stvarne veze izmedju entiteta. *Primer*: Ne popravlja se vozilo, već kvar i to prema radnom nalogu. Dakle, relacije su: vozilo-kvar-radni nalog-podaci o popravci.
- Vremenski sled zavisnosti entiteta.

- Homonimi i sinonimi.
- 1:1 odnosi.
- Šta je atribut, a šta entitet?

2.6.9. OBJEKTNI MODEL PODATAKA

Prema [LAZAREVIĆ i sar., 2003] i [MOGIN i sar., 1996] objektna orijentacija je pristup u kome se neki sistemi organizuju kao kolekcije međusobno povezanih objekata koji sarađujući ostvaruju postavljene ciljeve. Objektna orjentacija je danas dominantna u oblasti programiranja i programskih jezika, metodološkim pristupima razvoju softvera i informacionih sistema, postepeno preuzima primat i u bazama podataka.

Osnovni koncepti objektnog modela podataka su ***objekat*** i ***literal***. Pod objektom se podrazumeva entitet koji je sposoban da čuva svoja stanja i koji stavlja okolini na raspolaganje skup operacija preko kojih se ta stanja prikazuju ili menjaju. Literal je vrednost, tj. podatak koji se koristi u modelu i može biti prost, složen ili kolekcija. Razlika između objekta i literala je ta da objekat ima svoj identifikator a literal nema [LAZAREVIĆ i sar., 2003].

Objekti i literali se kategorizuju u tipove. Svi objekti istog tipa imaju isti skup stanja (osobina) i jedinstveno ponašanje (isti skup operacija). Konkretni objekat je pojavljivanje ili instanca datog tipa objekta, tj. ***klase objekta***. ***Stanje objekta*** je određeno vrednostima njegovih osobina, koji se zovu još i ***atributi*** objekta kao i vezama između tog objekta i drugih objekata u nekom sistemu. Ove osobine objekata se menjaju u vremenu. ***Ponašanje objekta*** se opisuje skupom operacija koje taj objekat izvršava ili koje se nad njim izvršavaju. Svaka operacija ima kao argument objekat kome je pridružena, može da ima listu ulaznih i izlaznih parametara a takođe može da vrati tipizovani rezultat. ***Baza podataka*** sadrži objekte koje stavlja na korišćenje većem broju korisnika, odnosno aplikacija. Baza podataka se opisuje preko svoje šeme u kojoj se definišu objekti čija se pojavljivanja čuvaju u bazi [LAZAREVIĆ i sar., 2003].

Svaki tip ima jednu ***specifikaciju*** i jednu ili više ***implementacija***. Specifikacije su eksterne karakteristike, vidljive korisniku tog tipa. To su operacije koje mogu biti pozvane, osobine kojima se može pristupati i to su izuzeci koji se mogu inicirati prilikom izvođenja operacije. Implementacija tipa definiše interne karakteristike objekata datog tipa i ona nije vidljiva korisniku. Odvajanjem specifikacije od implementacije tipa se ostvaruje veoma bitna karakteristika - ***učaurenje*** (***Encapsulation***) objekta. Specifikacija tipa se vrši preko koncepta interfejsa ili preko ***klase***. Interfejs se ne može direktno instancirati i preko njega se definišu operacije koje će naslediti korisnički definisani objekti. Klasa predstavlja apstraktno stanje i ponašanje tipa objekta i može se instancirati. Zapisuje se iskazom:

```
class Student{...};
```

Implementacija tipa se ostvaruje preko strukture podataka za opis stanja i skupa metoda preko kojih su implementirane operacije tipa objekta. Stanje pojavljivanja nekog tipa

objekta se iskazuje preko vrednosti njegovih osobina (atributa i veza). Ponašanje nekog tipa objekta predstavlja se skupom operacija. [LAZAREVIĆ i sar., 2003]

Primer klase je preuzet iz [LAZAREVIĆ i sar., 2003], str 174:

```
class Lice
{
            attribute string Ime;
            attribute date DatumRodjenja;
};
```

U objektnom modelu podataka postoje i dve vrste *nasleđivanja*:

- nasleđivanje ponašanja;
- nasleđivanje stanja;

Za nasleđivanje ponašanja se koristi veza nadtip-podtip (generalizacija-specijalizacija ili "IS A" veza). Prilikom nasleđivanja stanja podređena klasa nasleđuje celokupno stanje i ponašanje klase koju proširuje [LAZAREVIĆ i sar., 2003].

2.6.10. OBJEKTNO-RELACIONI MODEL PODATAKA

Prema [LAZAREVIĆ i sar., 2003] i [ULMAN i sar., 2002], d*va najvažnija modela podataka* na kojima se zasnivaju današnji sistemi za upravljanje bazama podataka su *relacioni* i *objektni model*. Struktura tabela u bazi podataka koje su kreirane na osnovu relacionog modela podataka imaju *ograničenja* prilikom kreiranja složenijih upita koji su potrebni za *kompleksniju obradu podataka*. Da bi se otklonio ovaj nedostatak relacionog modela proširen je konceptima objektnog modela podataka i tako se došlo do *objektno-relacionog* modela podataka.

Osnovno proširenje relacionog modela podataka se sastoji u tome da domeni atributa ne moraju biti samo osnovni predefinisani tipovi podataka, već i korisnički definisani složeni tipovi podataka koji mogu biti organizovani hijerarhijski. Na osnovu ove modifikacije proizilazi značajna činjenica da vrednost koja odgovara nekom atributu definisanom nad prethodno opisanim tipom podatka u okviru neke n-torke može biti čitava relacija. Ovakva relacija se naziva *ugnježđena relacija*.

Pored ovog proširenja, objektno-relacioni model podataka dozvoljava:

- mogućnost definisanja relacije direktno nad složenim tipom podataka koji se hijerarhijski može organizovati u generalizacije sa podrškom koncepta nasleđivanja;
- mogućnost definisanja metoda kojima se određuju operacije nad korisničkim tipovima;
- mogućnost da n-torke imaju ulogu objekata koji imaju svoje jedinstvene identifikatore;
- dva načina povezivanja podataka:

- konvencionalni - na osnovu vrednosti primarnih i spoljnih ključeva u povezanim tabelama;
- objektni - referenciranjem identifikatora povezanih n-torki.

Prethodno navedene karakteristike objektno-relacionog modela podataka *standardizovane* su SQL:1999 standardom, a izvršena je i integracija ovog modela u SQL (*Stuctured Query Language*) jezik za upite.

2.6.11. XML KAO MODEL PODATAKA

XML je, prema [DEVEDŽIĆ, 2004], World Wide Web Consortium (W3C) standard za razmenu podataka na Web-u. Osnovna ideja vezana za XML je da se difiniše standard kojim će se razmenjivati podaci na Web-u, ali tako da se sva pažnja usmeri na sadržaj, a ne na prikaz. XML je jezik koji nema predefinisan skup ključnih reči (elemenata i atributa), već je to jezik za definisanje drugih jezika, tj. metajezik. Međutim, sam po sebi XML nije dovoljan, već je u okviru XML tehnologije definisan skup standarda koji omogućuju njegovo korišćenje.

U [LAZAREVIĆ i sar., 2003]] i [LIGHT, 1997] je XML (*eXtensible Markup Language*) definisan kao jezik za označavanje strukture dokumenta unutar njegovog sadržaja, čija je prvenstvena namena standardizacija strukture i sadržaja dokumenata radi njihove efektivne razmene u različitim vrstama poslovnih i drugih elektronskih komunikacija. XML je jednostavan i fleksibilan jezik koji je zasnovan na standardnom uopštenom jeziku za označavanje (*Standard Generalized Markup Language* - SGML). Organizacija za standardizaciju Web tehnologija, World Wide Web Consortium (W3C) je 1996. godine počela razvoj XML-a sa osnovnim ciljem da se dobije jezik sa kombinacijom fleksibilnosti SGML-a i jednostavnosti HTML-a (*Hyper Text Markup Language*). Prva verzija XML-a je definisana 1998. godine. Za razliku od HTML-a koji prvenstveno opisuje način prikazivanja podataka u Internet stranici, XML opisuje podatke ali bez načina prikazivanja istih.

Primer XML dokumenta:

```
<?xml version="1.0" standalone="yes"?>
<DataSet1 xmlns="http://www.tempuri.org/DataSet1.xsd">
  <MESTO>
    <NAZIVMESTA>Beograd</NAZIVMESTA>
    <OZNAKAZEMLJE>YUG</OZNAKAZEMLJE>
    <PTT>11000</PTT>
  </MESTO>
  <MESTO>
    <NAZIVMESTA>Novi Sad</NAZIVMESTA>
    <OZNAKAZEMLJE>YUG</OZNAKAZEMLJE>
    <PTT>21000</PTT>
  </MESTO>
  <MESTO>
    <NAZIVMESTA>Zrenjanin</NAZIVMESTA>
    <OZNAKAZEMLJE>YUG</OZNAKAZEMLJE>
    <PTT>23000</PTT>
  </MESTO>
</DataSet1>
```

XML dokumenti su samoopisujuće platformski nezavisne tekstualne datoteke sa hijerarhijski uređenom strukturom koja se sastoji od elemenata, atributa i podataka tipa niza karaktera. Za specifikaciju tipova dokumenta se koristi *XML Schema* - XML dokument koji sadrži definicije prostih i složenih tipova, deklaracije elemenata, deklaracije atributa, modela grupa, grupa atributa, deklaracije notacija i drugo.

Primer XML Scheme:

```
<?xml version="1.0" standalone="yes"?>
<xs:schemaid="DataSet1"
targetNamespace="http://www.tempuri.org/DataSet1.xsd"
xmlns:mstns="http://www.tempuri.org/DataSet1.xsd"
xmlns="http://www.tempuri.org/DataSet1.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
attributeFormDefault="qualified" elementFormDefault="qualified">
<xs:element name="DataSet1" msdata:IsDataSet="true" msdata:Locale="sr-SP-
Latn">
     <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="MESTO">
          <xs:complexType>
            <xs:sequence>
       <xs:element name="NAZIVMESTA" type="xs:string" minOccurs="0" />
              <xs:element name="OZNAKAZEMLJE" type="xs:string" />
              <xs:element name="PTT" type="xs:int" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
    <xs:unique name="Constraint1" msdata:PrimaryKey="true">
      <xs:selector xpath=".//mstns:MESTO" />
      <xs:field xpath="mstns:OZNAKAZEMLJE" />
      <xs:field xpath="mstns:PTT" />
    </xs:unique>
  </xs:element>
</xs:schema>
```

U poslednje vreme se XML sve više tretira kao model podataka koji sistem predstavlja kao skup međusobno povezanih tipova dokumenata, a bazu kao kolekciju međusobno povezanih dokumenata koji su pojavljivanja definisanih tipova. Koristi se za povezivanje baza podataka sa ostalim izvorima podataka dostupnim preko Interneta. Takođe može da posluži za transformaciju jednog modela podataka u drugi, jer poseduje alate i metode preko kojih se to može izvršiti.

2.7. MODELOVANJE SOFTVERA U INFORMACIONOM SISTEMU

2.7.1. UML

Objektno orjentisani jezici modeliranja prvi put su se pojavili u periodu između 1970 i 1980 godine, kao eksperimenti u traženju novih pristupa i metoda analize i dizajna softverskog rešenja, koje bi pratilo već tada postojeće objektno-orjentisane programske jezike. Tada su korišćene metode strukturne analize, bazirane na funkcionalnoj dekompoziciji problema do nivoa zaokruženih celina koje se mogu transformisati u izvršni kod. Najveća raznolikost i broj objektno orjentisanih metoda javio se u periodu između 1989 i 1994 godine. Nakon 1990 godine započinje proces unifikacije metoda, pre svega tri vodeće metode: BOOCH metoda (Grady Booch), OOSE (Object Oriented Software Engineering) metoda (Ivar Jacobson) i OMT(Object Modelling Technique) metoda (James Rumbaugh). Naporima su se pridružili i drugi autori sa svojim metodama, kao i sve vodeće softverske i hardverske kuće (pre svega firma Rational, kao pokretač), tako da danas važeća verzija UML jezika 1.3 (prihvaćena u jesen 1999 godine) važi za standard koji prihvataju ovi proizvođači i u svoje postupke i proizvode ugrađuju elemente i pravila UML-jezika.

2.7.1.1. OBLASTI PRIMENE UML-a

UML je pre svega namenjen za prikazivanje konceptualnih i fizičkih karakteristika sistema. Pod sistemom se podrazumeva implementacioni sistem, tj. sistem koji je cilj razvoja i kasnije primene, dakle softverski sistem pre svega.

UML je jezik, što podrazumeva da ima svoje elemente rečnika i pravila primene. Naravno, jezik sam za sebe ne znači ništa, ukoliko se ne primenjuje kao sredstvo u okviru određene metodologije i oblikovanju rezultata pojedinih faza primene odgovarajuće metodologije. S obzirom da je UML pre svega ideja i rezultat rada firme Rational, ova firma je stvorila i metodologiju razvoja softvera, nazvanu "Rational Unified Software Development Process", u okviru koje se koristi ovaj jezik kao sredstvo. UML je dizajniran da bude fleksibilan, tako da je moguće primenjivati njegove koncepte i dijagrame u različitim fazama razvoja softverskog rešenja, od faze specifikacije zahteva korisnika i upoznavanja sa poslovnom semantikom realnog sistema, do dizajna sofverskog rešenja, implementacije, isporuke i testiranja.

UML je jezik pre svega:

1. *Vizualizacije* (sredstva su pre svega grafički simboli, ali u odrešenim segmentima su i tekstualni opisi kao delovi modela).
2. *Specifikacije* (sredstvo je dokumentovanja odluka specifikacije zahteva i dizajna, precizno, nedvosmisleno i potpuno daje osnov za dalji rad - programiranje, testiranje)
3. *Konstrukcije* (omogućuje direktno preslikavanje u izvorni kod).
4. *Dokumentovanja* (elementi omogućuju kreiranje dokumenata u svim fazama razvoja softvera i njihovim rezultatima).

UML se najviše primenjuje u semantičkim oblastima kao što su:

1. Informacioni sistemi preduzeća
2. Bankarski i finansijski servisni sistemi
3. Telekomunikacije
4. Transport
5. Odbrana
6. Nauka
7. Medicinska elektronika
8. Distribuirani web bazirani servisi.

UML nije namenjen samo za modelovanje softvera, već je ekspresivan dovoljno da bi se primenio i za modelovanje drugih sistema, npr. hardvera, organizacionih sistema i njihovih radnih tokova i sl.

UML je jezik koji je pre svega sredstvo modelovanja i dizajniranja velikih softverskih sistema. Njegove prednosti dolaze do izražaja u velikim sistemima distribuirane obrade, sa mnoštvom procesa obrade, najčešće paralelnih i sinhronih. Naravno, korišćenje UML-a kao jezika uz odgovarajući standardni proces razvoja softvera (Unified Software Development Process) daje oslonac kao zaokružena metodologija, proverena u praksi, čime se postiže potpunost, sledljivost i usaglašenost pojedinih rezultata faza razvoja (jer su sve pokrivene u određenom obliku), čime se znatno štede resursi i dobija na kvalitetu završnog proizvoda, jer se dizajneri usredsređuju na stručne i semantičke probleme, umesto na alat i metode razvoja.

2.7.1.2. OSNOVNI ELEMENTI UML-a

2.7.1.2.1. OSNOVNI PRINCIPI

Neki od osnovnih principa UML-a izraženi su kroz oblike i pravila korišćenja elemenata i modela:

1. Princip fleksibilnosti
2. Princip potpunosti
3. Princip adekvatnosti
4. Princip jednoznačnosti i preciznosti
5. Princip jasnosti

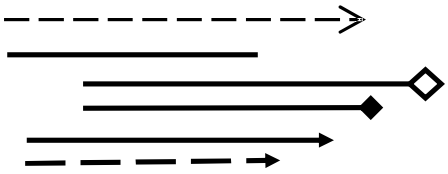
2.7.1.2.2. OSNOVNI KONCEPTI

Osnovni koncepti se mogu podeliti na:

1. Osnovne gradivne (vizualne - grafičke i tekstualne) elemente
2. Pravila
3. Opšte mehanizme

Osnovni gradivni elementi su:

1. "Stvari"
    1.1. Strukturalne stvari: klasa, interface, collaboration, use case, aktivna klasa, komponenta, čvor
    1.2. Stvari "ponašanja": interakcija, stanje
    1.3. Stvari grupisanja: paket
    1.4. Stvari notiranja (beleški): komentar
2. Relacije
    2.1. Relacija zavisnosti
    2.2. Relacija asocijacije
        2.3. Relacija agregacije
        2.4. Relacija kompozicije
    2.3. Relacija generalizacije
    2.4. Relacija realizacije



3. Dijagrami
    3.1. USE CASE DIAGRAM
    3.2. CLASS DIAGRAM
    3.3. OBJECT DIAGRAM
    3.4. STATECHART DIAGRAM
    3.5. ACTIVITY DIAGRAM
    3.6. COLLABORATION DIAGRAM
    3.7. SEQUENCE DIAGRAM
    3.8. COMPONENT DIAGRAM
    3.9. DEPLOYMENT DIAGRAM

Pravila se odnose na pravila imenovanja, pravila uloge elemenata, vidljivosti i mogućnosti povezivanja sa elementima itd.

Opšti mehanizmi su:

1. Mehanizam specifikacije, UML koristi:
    - grafičke elemente za vizualizaciju
    - tekstualne prikaze za detaljniji opis elemenata
2. Mehanizam detaljisanja, UML koristi specijalne grafičke detalje, za prikaz osobina pojedinih elemenata,kao što je vidljivost od strane drugih elemenata i slično.
3. Mehanizam podele,UML jasno deli:
    - uopšteno od konkretnog (class/object, association/link...)
    - interfejs od implementacije
4. Mehanizam proširenja, UML dozvoljava proširenja pomoću:
    - stereotip
    - pridružena vrednost
    - ograničenje.

Detalji izgleda i pravila načina korišćenja koncepata opisani su u [BOOCH i sar., 2000].

PRIMER GALERIJE

Primer projekta "Galerija" dat je u pojednostavljenom obliku, pre svega iz razloga što je cilj ovog priloga demostracija elemenata UML jezika, a ne prikaz kompletne projektne dokumentacije. Zato je opis problema dat pre svega kao opis životnog ciklusa osnovnog objekta obrade, da bi se mogla razumeti semantička oblast. Dijagrami su dati u pojednostavljenom obliku (izostavljeni neki elementi, prikazani su delovi dijagrama, ne cele šeme, dat je po jedan primer za svaku vrstu), uz izostavljanje detalja. Dakle, nije bilo nastojanje da se postigne preciznost i potpunost, već pre svega ilustracija primene elemenata UML jezika.

OPIS PROBLEMA

Organizacioni sistem koji je za potrebe ovog priloga analiziran je SAVREMENA GALERIJA UK EČKA Zrenjanin (u daljem tekstu Galerija). Opis problema je prikazan u skraćenom obliku i orjentisan je pre svega na prikaz delatnosti Galerije. Ostali elementi specifikacije problema su izostavljeni.

Svrha postojanja Galerije kao organizacije je formiranje fonda slika, čuvanje i ustupanje slika i drugih umetničkih predmeta i organizovanje i prezentacija umetničkih predmeta na raznim vrstama manifestacija.

Osnovni objekat obrade Galerije je umetnički predmet - slika, skulptura i slično. Umetnički predmeti koji pripadaju Galeriji su u vlasništvu Republike Srbije.

Životni ciklus slike u Galeriji počinje nabavkom. Galerija nabavlja sliku najčešće od autora - kupovinom, poklonom od autora ili preuzimanjem nakon Umetničke kolonije u Ečkoj. Naime, pravilo je da nakon održavanja Umetničke kolonije u Ečkoj svaki autor učesnik kolonije ostavlja Galeriji, koja je organizator Kolonije, po 2 slike za Umetnički fond.

Ukoliko je nabavka kupovinom, Galerija formira komisiju (direktor imenuje komisiju posebnim rešenjem) koja procenjuje stanje i vrednost slike i izveštaj o proceni dostavlja direktoru Galerije. Direktor donosi odluku o kupovini slike. Vlasnik slike potpisuje izjavu kojom slika prelazi u vlasništvo Galerije. Vrši se uplata iznosa vrednosti slike (dogovorene sa vlasnikom slike i procenjene od strane komisije). Prilikom preuzimanja slike vrši se procena stanja i vrednosti slike. Slika se opisuje određenim osobinama i tekstom opisa sadržaja slike, dodeljuje joj se signatura i inventarni broj, vrši se evidencija u inventarnoj knjizi slika i u knjizi ulaza slika. Takođe, slika se i fotografiše, pa se evidentira i odgovarajući broj negativa slike. Slici se dodeljuje trajno smeštajno mesto, kao mesto gde će biti odložena i prilikom izdavanja gde treba da bude uvek vraćena. U okviru jednog smeštajnog mesta postoji više depoa (blokova), tako da se slika smešta na samo jedan depo (blok) u okviru smeštajnog mesta. Ukoliko je potrebno, vrši se obrada slika - dodavanje rama, restauracija i konzervacija slika i podaci o ovim aktivnostima se uvode u dosije konzervacije i restauracije.

U toku "života" slike u Galeriji, dešavaju se razne aktivnosti:
1. Promet slika - izdavanje i vraćanje za razne potrebe i različitim komitentima.
2. Inventarisanje slika - provera brojnosti, smeštaja, stanja i vrednosti slika.

Izdavanje slika započinje najčešće usmenim zahtevom za određenim slikama. Slike se izdaju u svrhu izlaganja ili u svrhu eksternih popravki. Komitent bira jednu ili više slika, na osnovu kataloga slika ili nekog drugom izvora podataka. Komitent koji uzima sliku može biti pojedinac (autor slike, neki drugi pojedinac) ili organizacija (druga organizaciona jedinica Galerije, druge galerije, javne organizacije ili preduzeća). Ukoliko je u pitanju javna organizacija ili preduzeće, najčešće se prethodno potpisuje se ugovor o saradnji, čime se preciziraju uslovi saradnje i razmene. Najčešće su ove organizacije sponzori i donatori Galerije. Slike se najčešće izdaju na neograničen rok (iako je moguće da postoji i određen datum vraćanja slika), tako da se slike vraćaju Galeriji na njen zahtev, za potrebe novog izdavanja ili iz nekog drugog razloga. Pre izdavanja slike najčešće se vrši procena stanja i vrednosti slike, kako bi se odgovarajući podaci mogli uneti u revers i na taj način opisati obaveza komitenta. Prilikom primopredaje slika primalac - zaposleni u organizaciji koja je uzela slike ili pojedinac) potpisuje i prima revers za izdate slike. Time je u obavezi da slike čuva i vrati u stanju u kojem su izdate.

Naravno, slike mogu biti izdate i u svrhu popravke, tj. restauracije i to izvan Galerije, tako da se pod izdavanjem slika može smatrati i ovakva vrsta izdavanja slike sa njenog trajnog smeštajnog mesta.

Ukoliko komitent nije vratio slike, a želi da mu se izdaju nove slike formira se novi revers sa slikama koje su izdate u tom trenutku. Pri tome, postoji pravilo da se revers sa slikama koje su rpethodno bile izdate poništava i podaci o ranije izdatim slikama se objedinjuju (prepisuju) sa slikama na novom reversu. Tako je u svakom trenutku aktivan i važeći samo jedan revers.

Galerija najčešće usmeno (telefonom) ili pismeno (faxom) zahteva vraćanje slika ili komitenti vraćaju ili najave vraćanje slike. Zahtevanje vraćanja se radi iz više razloga: zbog izlaganja na izložbama koje organizuje Galerija, zbog izdavanja nekom drugom Komitentu (ukoliko je prethodno bila izdana na neodređeni rok) ili ukoliko je istekao rok za vraćanje slika. Prilikom vraćanja ne moraju sve slike biti vraćene. već samo jedan deo. Ukoliko je takva situacija, formira se novi revers sa preostalim slikama. U postojećem načinu rada ne postoji dokument kojim bi se revers sa vraćenim slikama poništio i evidentirale vraćene slike. U trenutku ili nakon primopredaje slika, najčešće se vrši procena stanja i vrednosti slike, kako bi u slučaju oštećenja mogao da se pokrene postupak obeštećenja slike. Postupak obeštećenja obuhvata odgovarajući zahtev za obeštećenjem sa naznačenim novčanim iznosom na ime obeštećenja određene slike.

Kada se vraćaju, slike se najpre donose u Upravu Galerije. Slike se vraćaju najčešće tako što vozač Galerije službenim vozilom prenese slike do Uprave Galerije. Tu se vrši njihova procena i ostali postupci. Neoštećene slike vozač Galerije odnosi do smeštajnih mesta, ukoliko su dislocirane u odnosu na zgradu Uprave. Svaka slika se vraća u odgovarajući depo na svom smeštajnom mestu.

Što se tiče inventarisanja slika, ono se vrši po pravilu jednom godišnje (ili u manjim periodima vremena) u cilju popisa slika - utvrđivanja prisutnosti slika na mestima gde bi se trenutno trebale nalaziti, njihovog stanja i ponekad i vrednosti.

Kraj životnog ciklusa slike predstavlja otpis ili otuđenje slike. Proces otpisa može biti iniciran lošim stanjem slike (najčešće zbog vlage). Otpis slike se vrši nakon procene stanja slike od strane imenovane komisije Galerije. Ukoliko je stanje takvo da slika ne može biti popravljena/restaurirana, donosi se odluka o otpisu slike i ona može biti otuđena iz Galerije. Po postojećim propisima, slika ne može biti otuđena (dakle ni prodata) osim u slučaju otpisa.

Sve procene stanja i vrednosti slika vrši posebno imenovana komisija od strane Direktora Galerije. Komisija je imenovana rešenjem, a o svom radu podnosi izveštaj. Na osnovu izveštaja Direktor donosi odluku o daljim aktivnostima nad slikom.

Galerija se bavi i organizovanjem različitih vrsta kulturnih manifestacija, najčešće izložbenog karaktera. Prilikom organizacije manifestacija, jedna od aktivnosti je kreiranje kataloga slika koje učestvuju na izložbi.

Problemi - Osnovni problemi postojećeg informacionog sistema su uobičajeni za manualne i delimično uvedene AOP informacione sisteme i ovde neće biti detaljnije prikazani.

Zahtevi - Najvažniji deo specifikacije zahteva odnosi se na osobine budućeg softverskog rešenja i obuhvat funkcija i podataka koji su prikazani u sistemskoj analizi kao postojeći i neophodni. Usvojen je inkrementalni pristup pri čemu se najpre razvija najvažniji skup funkcija, sa mogućnošću dalje nedogradnje. Najvažnije funkcije su: Obrada fonda slika - nabavka i inventarisanje i Obrada prometa slika - izdavanje i vraćanje slika.

2.7.1.3. PRIMERI UML DIJAGRAMA

UML dijagrami se mogu podeliti u odnosu na poglede koji ovi modeli imaju na sistem koji se razvija:

1. LOGIČKI
    1.1. STATIČKI - STRUKTURA (Class, Object)
    1.2. DINAMIČKI - PONAŠANJE (USE CASE, Interaction, Activity, Statechart)
2. IMPLEMENTACIONI (Component, Deployment).

Svaki dijagram se može predstaviti sa većim ili manjim stepenom detaljnosti, dubine prikaza.

2.7.1.4. USE CASE DIAGRAM

USE CASE diagram se sastoji od više slučajeva korišćenja, učesnika i relacija.

Pod slučajem korišćenja (USE CASE) se najčešće smatra:

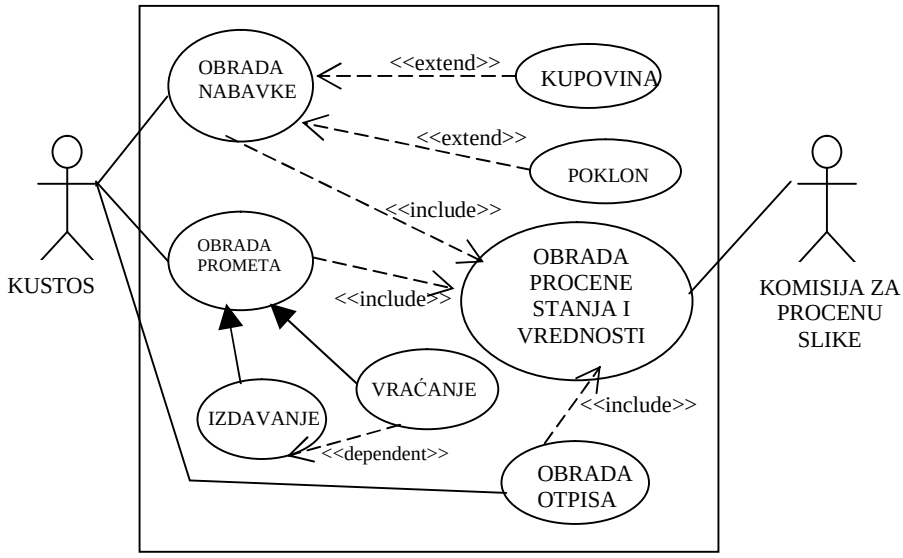
a. U FAZI MODELOVANJA POSLOVNOG DOMENA - poslovna funkcija, tj. usluga koja se nudi korisnicima usluga organizacionog sistema u celini,
b. U FAZI SPECIFIKACIJE ZAHTEVA KORISNIKA ZA FUNKCIJAMA SOFTVERA I FAZI DIZAJNA SOFTVERSKIH FUNKCIJA - softverska funkcija koja je na raspolaganju korisniku tog softvera, usluga koja daje određeni rezultat od vrednosti za korisnika.

Dakle, isti koncepti se koriste u različitim fazama razvoja softvera. U skladu sa tim razlikuje se i značenje ostalih elemenata na diagramu USE CASE:

1. ACTOR
a. Neposredno poslovno okruženje sa kojim poslovni sistem razmenjuje podatke.
b. Neposredni korisnik programa, dakle kategorija korisnika koja će imati direktan pristup softveru i u odnosu na čije potrebe su dizajnirane ove funkcije.

2. RELATIONSHIP
a. Podaci koji se razmenjuju sa poslovnim "teku" pomoću poruka koje su moguće pomoću relacije između actora i poslovne funkcije i inicirajuće aktivnosti koje pokreću te aktivnosti
b. Podaci koji se razmenjuju između neposrednog korisnika programa i samog programa.

Za specifikaciju slučajeva korišćenja koristi se grafički prikaz i prikaz svakog od slučaja korišćenja strukturiranim tekstom, kako bi se preciznije odredila njegova struktura. Grafički prikazi su u obliku:
1. USE CASE diagrama - prikaz slučaja korišćenja bez prikaza kako je realizovan, već samo šta treba da radi.
2. Interaction diagrama - implementacija slučaja korišćenja pomoću objekata i poruka.



Slika 2.11. USE CASE DIAGRAM

U primeru koji sledi prikazan je aspekt specifikacije softverskih funkcija, dok se aspekt specifikacije poslovnog domena na najbolji način može uraditi primenom DTP Strukturne sistem analize. Tako, Komitent i Vlasnik Slike kao okruženje poslovnog domena nemaju (iako bi možda mogli, ali u postojećem softverskom rešenju nije predviđeno) neposrednog "dodira" sa softverskim funkcijama (nemaju neposrednu mogućnost iniciranja softverskih funkcija, već na posredan način bivaju informisani - dokumentima i slično), tako da neće biti prikazani, već samo actori koji imaju neposrednu ulogu pokretanja softverskih funkcija. To su pojedine kategorije zaposlenih u organizacionom sistemu - Kustos i Komisija za procenu slike.

2.7.1.5. CLASS DIAGRAM (STATIC STRUCTURE DIAGRAM)

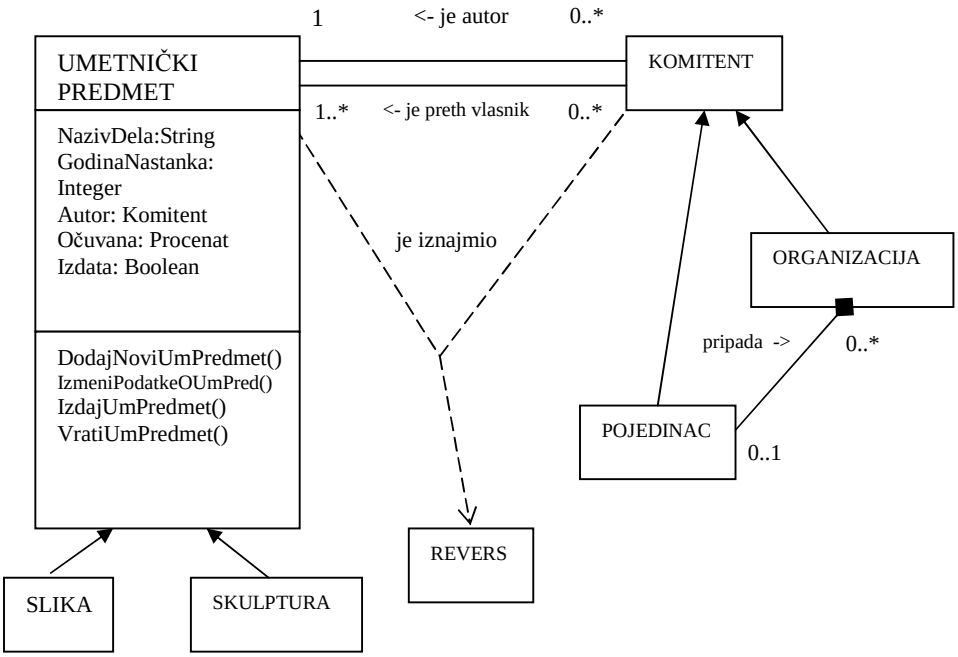
Dijagram klasa predstavlja sredstvo prikaza statičkih odnosno strukturalnih osobina sistema. Sastoji se od simbola klasa, interfejsa, "collaboration"-a i relacija.

Dijagram klasa se najčešće koristi:

1. Za prikaz rečnika posmatranog sistema, odnosno za određivanje granica sistema - šta sve sistem obuhvata. Ovde se pre svega klase odnose na određene vrste entiteta iz realnog sveta koji se obrađuje i potrebno je predstaviti njihovu semantiku - model entiteta i njihovih veza.
2. Za prikaz "collaboration"-a, odnosno skupa klasa, relacija i interfejsa koji realizuju pojedine slučajeve korišćenja. Ovde bi bile prikazane klase povezane statičkim vezama, a izabrane bi bile one koje realizuju određen slučaj korišćenja.
3. Kao sredstvo za prikaz šeme baze podataka - jezik class diagrama je bogatiji od jezika ERA dijagrama, tako da se može koristiti za prikaz šeme relacione ili objektne baze podataka. Ovde treba posebno obratiti pažnju na kardinalitete veza i atribute klasa koje su predstavnici tabela baze podataka. Za klase se za ovaj slučaj upotrebe postavlja <<persistent>> stereotip i one odgovaraju tabelama baze podataka.
4. Za prikaz softverskih komponenti implementacije softverskog rešenja i njihovih osobina i odnosa.

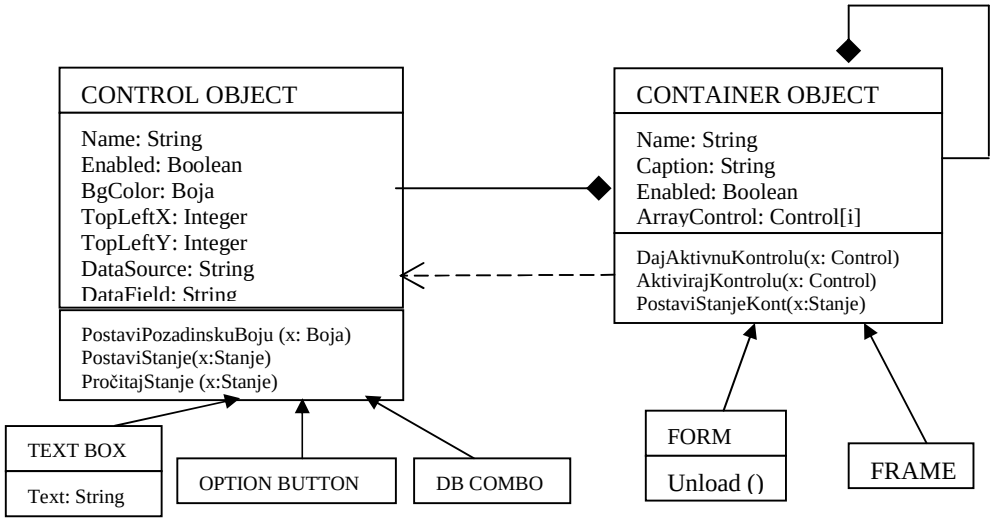
U primerima koji slede prikazane su primene navedene pod brojem 1 i 4 sa izdvojenim pojedinim interesantnim klasama, relevantnim atributima i operacijama. Za primenu u domenu "collaboration"-a postoje posebni collaboration diagrami, a za primenu u dizajniranju baze podataka nije dat primer, jer je urađen koristeći CASE alat koji podržava konceptualno i fizičko modeliranje podataka.

PRIMENA - REČNIK POSMATRANOG SISTEMA



Slika 2.12. CLASS DIAGRAM

PRIMENA - SOFTVERSKE KOMPONENTE REALIZACIJE (Elementi GUI)



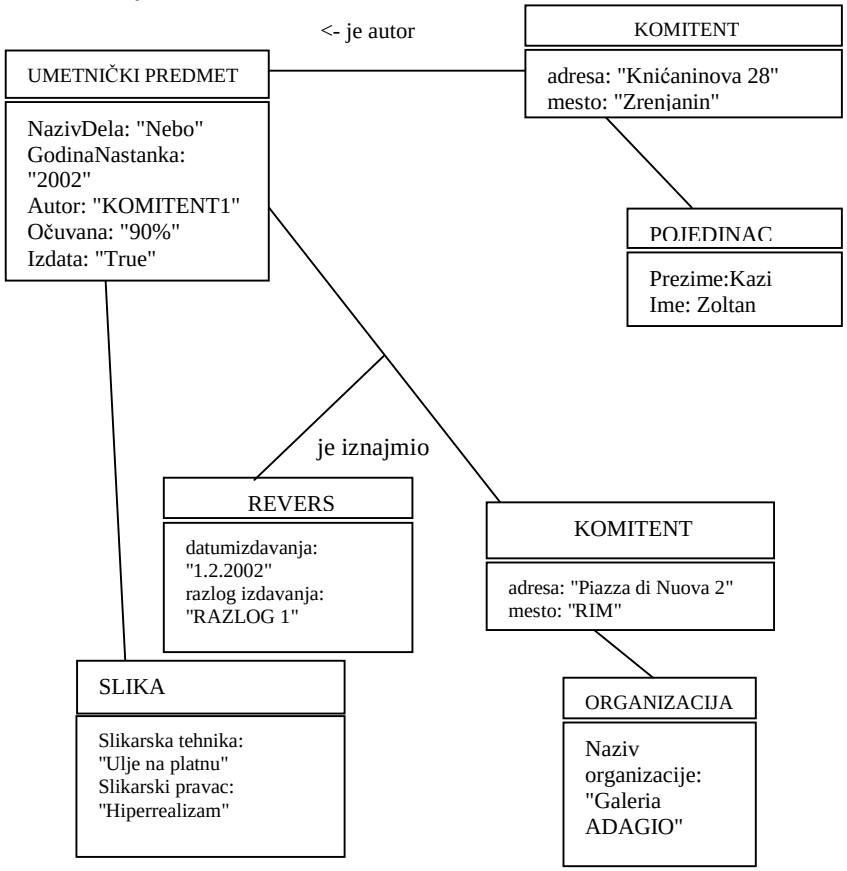
Slika 2.13. CLASS DIAGRAM

2.7.1.6. OBJECT DIAGRAM

Object diagram sadrži objekte, kao instance pojedinih klasa(pravougaonici) i linkove (pune linije), kao instance relacija dijagrama klasa.

U određenom izabranom trenutku, prikazuje pre svega vrednosti atributa klasa, čime su određena stanja tih objekata.

Najčešće se primenjuje za prikaz situacije nakon izvršene neke softverske funkcije (use case ili scenario). Dakle, misaono se "pokrene" izvršavanje neke softverske funkcije, i u nekom trenutku toka izvršavanja se "zamrzne" stanje klasa koje učestvuju, tj. dobije se situacija sa konkretnim instancama tih klasa i relacija. Primer pokazuje "situaciju" nakon izvršavanja USE CASE-a "Izdavanje umetničkog predmeta", dakle rezultat rada ove softverske funkcije.

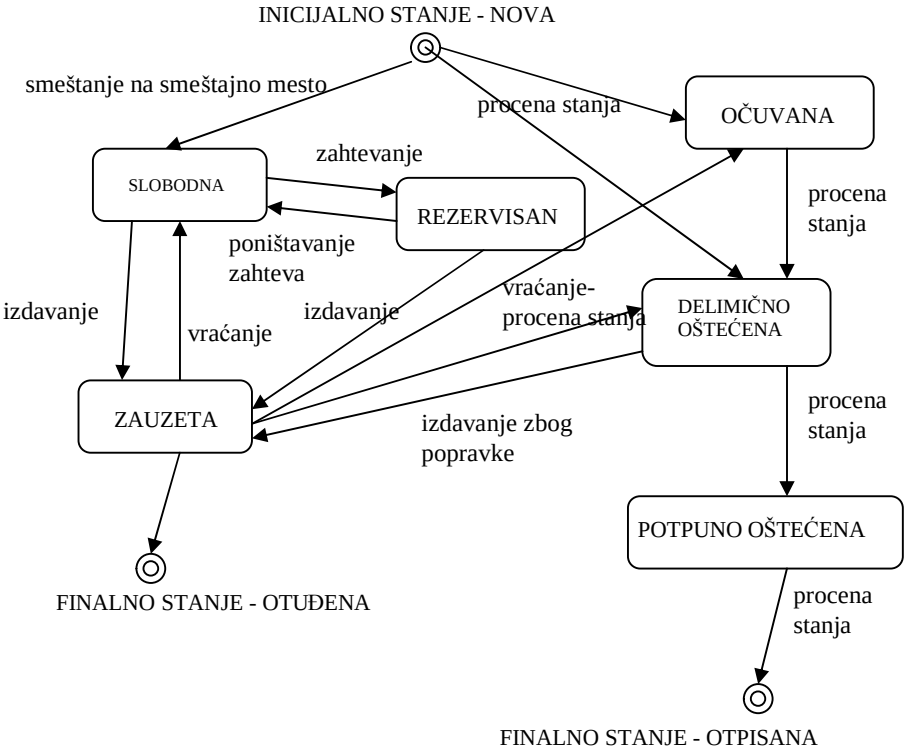


Slika 2.14. OBJECT DIAGRAM

2.7.1.7. STATECHART DIAGRAM

Statechart diagram (dijagram stanja) prikazuje promenu stanja (state) sistema ili pojedinih podsistema ili objekata, uslovljeni izvršavanjem akcija (action, activity) koje su pokrenute od strane događaja (event). Dakle, događaj izaziva akciju, a akcija menja stanje objekta. Prevođenje jednog stanja u drugo prikazuje linija tranzicije.

Dakle, na dijagramu stanja su prikazani, za izabrani domen posmatranja (najčešće jedna i to reaktivna (što znači klasa čiji objekti mogu da menjaju stanja u zavisnosti od inicirajućih događaja) klasa, odnosno stanja njenih objekata), stanja, tranzicije, akcije, aktivnosti i događaji.



Slika 2.15. STATECHART DIAGRAM

Tranzicije (transition) predstavljaju linije koje povezuju pojedina stanja. One predstavljaju prevođenje jednog stanja u drugo. Na linijama tranzicije se najčešće nalaze nazivi događaja, akcija i aktivnosti koje su uslovile prelaz iz jednog stanja u drugo. Akcija predstavlja atomarnu operaciju (dakle neprekidivu - uninterruptable, tj. nemoguće je podeliti na manje i nemoguće je prekinuti njeno izvršavanje u toku samog njenog izvršavanja) kojom se jedan objekat prevodi iz jednog stanja u drugo. Aktivnost (activity) predstavlja neatomarnu

operaciju, dakle aktivnost se može dekomponovati na akcije i prekidiva je (interruptable). Izabran je primer slike kao objekta. Slika generalno ima dve vrste stanja:
1. Po pitanju prometa - da li je slobodna (tj. na smeštajnom je mestu) ili je izdata (zauzeta).
2. Po pitanju stepena očuvanosti - da li je očuvana, delimično oštećena ili potpuno oštećena.

2.7.1.8. ACTIVITY DIAGRAM

Activity diagram prestavlja tok kontrole od aktivnosti do aktivnosti. Najčešće se koristi za prikaz:

1. Toka poslovnih procesa (Bussiness workflow)
2. "Algoritma" operacije neke klase (Class operation flowchart)
3. Ponašanja klasa u collaboration-u, tj. realizaciji jednog use case-a

Activity diagram se sastoji od sledećih elemenata:

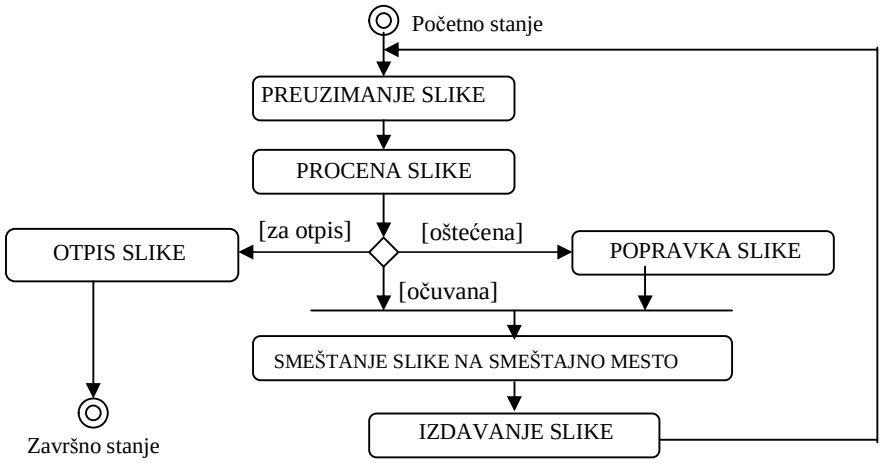
1. AKTIVNOST - Activity state, action state (oval)
2. TOK KONTROLE, VEZA - transition
3. GRANANJA - branch, fork
4. SPAJANJA GRANA - join

i opciono:

5. OBJEKTI I VEZE KA NJIMA - object i object flow
6. PODELA ODGOVORNOSTI NAD AKTIVNOSTIMA OD STRANE OBJEKATA - swimlane.

U narednim primerima je dat prikaz korišćenja activity diagrama koji odgovaraju 1. i 2. stavci. Ponašanje u collaboration-u se može videti kroz interaction diagrame, pre svega squence diagrame.

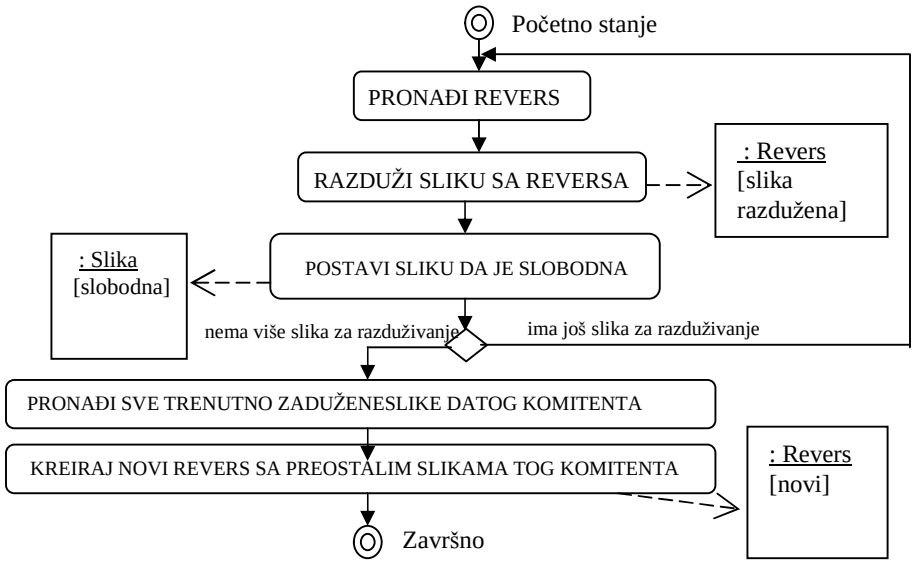
PRIMER BUSSINESS WORKFLOW



Slika 2.16. ACTIVITY DIAGRAM

PRIMER CLASS OPERATION FLOWCHART

Klasa: slika
Operacija: VratiSliku



Slika 2.17. ACTIVITY DIAGRAM

2.7.1.9. COLLABORATION DIAGRAM

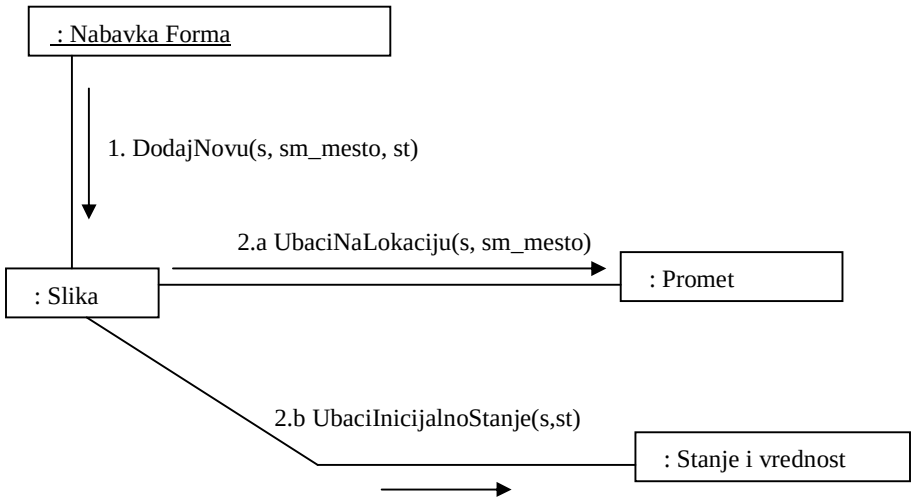
Collaboration diagram je jedan od dve vrste tzv. interaction diagrama. Služi za prikaz dinamičkih osobina sistema.

Fokus ove vrste diagrama je u prikazu strukturne organizacije objekata koji šalju i primaju poruke, u cilju izvršavanja zajedničkog zadatka- realizacije use case-a ili nekog ponašanja.

Diagram collaboration se sastoji od:

1. Objekata - imenovanih ili neimenovanih (anonimnih) instanci klasa
2. Linkova - instanci relacija
3. Poruka - pozivi operacija objekata(najčešće) ili prosleđivanje signala u obliku vrednosti. Nazivu poruke prethodi broj i dvotačka, čime se prikazuje redosled (vremenski sled) u izvršavanju niza poruka na celom diagramu.

Primer collaboration diagrama dat je za realizaciju use case-a nabavka slika, tj. evidentiranje nabavljene slike. (Primer je sličan primeru iz [STANOJEVIĆ i sar., 1999], str. 88).



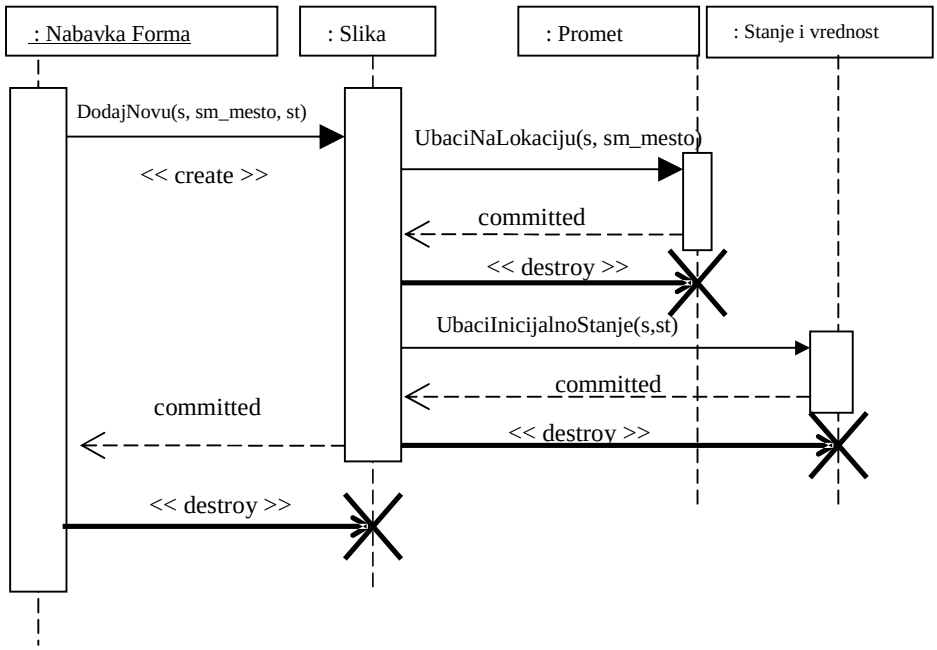
Slika 2.18. COLABORATION diagram

2.7.1.10. SEQUENCE DIAGRAM

Sequence diagram jedan je od dva tzv. interection diagrama i služi za predstavljanje dinamičkih osobina sistema ili njegovih delova. glavni fokus sequence diagrama je prikaz vremenskog sleda slanja i prijema poruka između objekata u cilju ostvarenja zajedničkog zadatka.

Elementi sequence diagrama su:

1. Objekti (najčešće anonimne instance klasa).
2. Poruke - pozivi operacija objekata ili prosleđivanje signala sa vrednostima.
3. Vremenske linije - Isprekidane linije koje prikazuju vremenski tok života objekta.
4. Linije programske kontrole - (prazan pravougaonik) prikazuju koji objekat i koliko dugo ima fokus i vlasnik je programske kontrole nad događajima u sistemu.



Slika 2.19. SEQUENCE DIAGRAM

Posebne vrste poruka su:

1. Call - poziv operacije (opciono je navođenje reči call ispred naziva operacije)
2. Return - vrednost koju vraća izvršena operacija
3. Send - slanje signala, vrednosti
4. Create - Kreiranje novog objekta
5. Destroy - uništavanje objekta

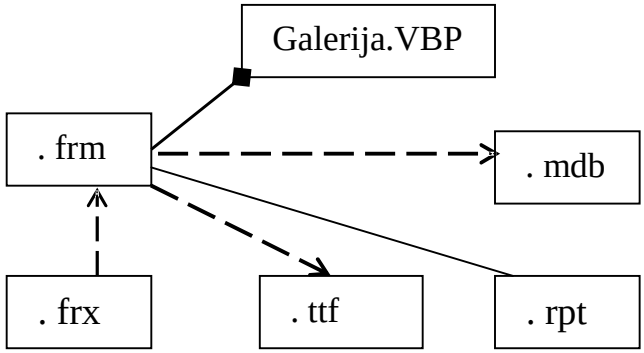
Pošto su collaboration diagrami i sequence diagrami izomorfni, tj. mogu se prevesti jedan u drugoga, dajem primer koji je dat kao collaboration diagram u obliku sequence diagrama, da bi se moglo izvršiti upoređivanje.

2.7.1.11. COMPONENT DIAGRAM

Component diagram se zasniva pre svega na elementima kao što su softverske komponente i relacije između njih.

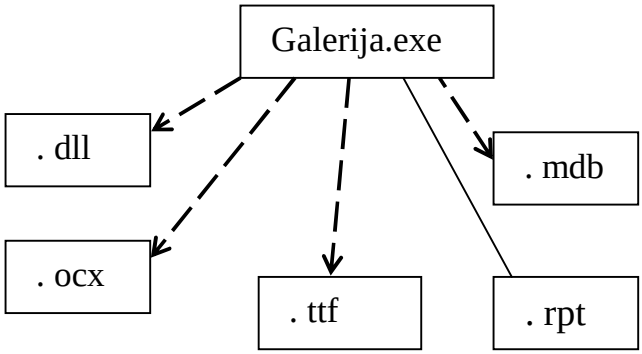
Najčešće je ova vrsta diagrama korišćena za:
1. Prikaz komponenti source koda, tj. fajlova koji čine source kod.
2. Prikaz komponenti koje čine izvršni ili distribucioni paket fajlova.

1. Komponente source koda
Projektni fajl - Galerija.VBP
Baze podataka - Galerija.mdb, Dinam.mdb, Korisnik.mdb
Forme - .frm, .frx
Izveštaji - .rpt
Fontovi - .ttf



Slika 2.20. COMPONENT DIAGRAM

2. Izvršni paket fajlova
Izvršna datoteka - Galerija.exe
Biblioteke - . dll, . ocx
Izveštaji - . rpt
Baze podataka - Galerija.mdb, Dinam.mdb, Korisnik.mdb
Fontovi - .ttf

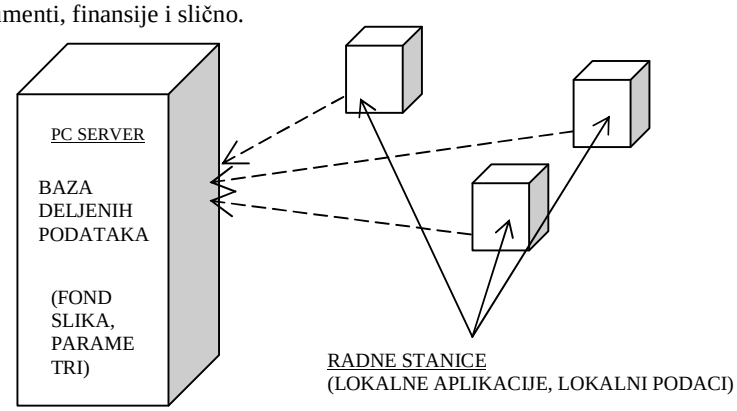


Slika 2.21. COMPONENT DIAGRAM

2.7.1.12. DEPLOYMENT DIAGRAM

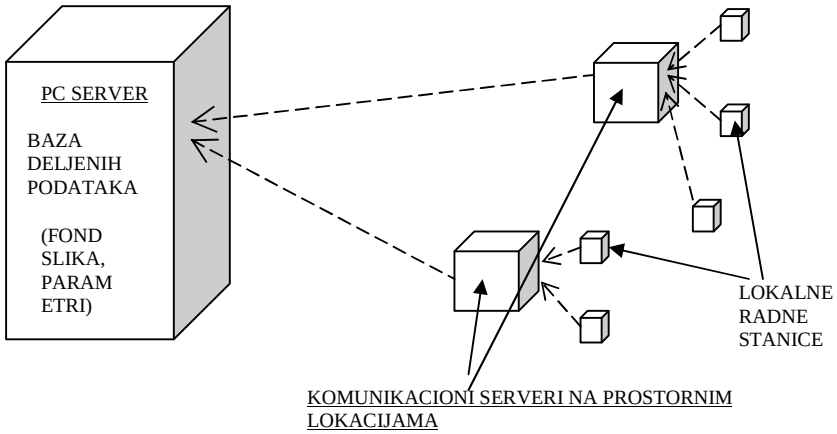
Deployment diagram se zasniva pre svega na elementima kao što su procesni čvorovi i relacije između njih. S obzirom da je ovim radom opisana postojeća aplikacija, koja je aplikacija za jednu mašinu, nema smisla da se ova vrsta diagrama razmatra. Naravno, u perspektivi je da se sistem razvija u pravcu proširenih client-server aplikacija, odnosno:

1. LAN računarska mreža na nivou zgrade uprave Galerije, sa aplikacijom koja bi bila proširena (u odnosu na broj obuhvaćenih funkcija) u odnosu na postojeću i obuhvatala evidenciju fonda i podatke o svim aktivnostima vezano za slike i ostale elemente, kao što su dokumenti, finansije i slično.



Slika 2.22. DEPLOYMENT DIAGRAM

2. WAN računarsku mrežu, koja bi povezala upravu sa prostorno udaljenim čvorovima - smeštajnim mestima fonda slika.



Slika 2.23. DEPLOYMENT DIAGRAM

Detaljno razmatranje deployment diagrama na ovom nivou je bespredmetno, jer nije izvršena detaljna specifikacija zahteva i ostalih dizajnerskih rešenja, koja mora da prethodi.

Napomena - Cilj ovog dela knjige je da se prikaže kratak pregled UML koncepata kroz pojedinačne primere primene tih koncepata na semantičkom domenu organizacionog sistema Savremena galerija Zrenjanin, odnosno Programa za vođenje evidencije fonda i prometa slika Galerije. Primeri nisu opterećeni bogatom anotacionom sintaksom UML-a, već samo najvažnijim konceptima i njihovim osobinama, a takođe nisu dati svi detalji koji bi bili neophodni u preciznom i potpunom UML projektu, već je bilo nastojanje da se na pojednostavljenim semantičkim situacijama prikaže svrha pojedinih UML koncepata. Nastojao se postići pre svega edukativan efekat-motivacioni pre svega, jer koncepti UML-a su jednostavni, ukoliko se prikaže na dovoljno ilustrativnom primeru njihova suština.

2.7.2. DIZAJN APLIKACIJE

Stručni tim koji učestvuju u izradi aplikacije sastoji se:

1. Bussiness analitičar - intervjuiše domenske stručnjake, klijente i zadatak mu je da razume problematiku domenske oblasti, potrebe (procesi, podaci) realnog sistema i zahteve korisnika i da to predstavi kroz odgovarajuću dokumentaciju. Takođe, radi i sa inžinjerima na implementaciji, praćenjem pokrivenosti uočenih potreba realnog sistema.
2. Sistem analitičar - modelira realan sistem na osnovu dokumentacije od strane bussiness analitičara.
3. Dizajner baze podataka – radi MOV i BP.
4. Dizajner ekrana - osmišljava izgled i funkcionisanje svih ekrana (user friendly).
5. Programer (Engineering staff).
6. Stručnjak za hardver (Engineering staff).
7. Stručnjak za mreže.

Osobine koje treba da imaju alati za programiranje:

- objektno orjentisani (biblioteke gotovih klasa i mogućnost pravljenja vlastitih);
- event driven (programiranje događaja);
- GUI okruženje za razvoj i izgled aplikacije.

Aplikacija za rad sa bazama podataka treba da sadrži:

- kontrole za rad sa bazama podataka i podešavanja osobina;
- tipovi formi: a)1:1, 1:M, b) šifrnici, procesi iz SSA, c) ažuriranje, pregled, pretraga, štampa;
- user friendly osobine;
- standardni (lokalni standardi na nivou aplikacije ili opšti standardi);
- izgled i ponašanja programa (raspored kontrola, stanja ekrana itd.)
- opredeljenja za vrste korisničkog interfejsa (KI):
  a) navike korisnika ili standardi,
  b) orijentacija na procese i događaje ili orijentacija na entitete i objekte.

Struktura glavnog menija aplikacije:

Šifarnici, Ograničenja, Obrada, Izveštaji, Upiti, Servis

- **Šifarnici** - ekrani za unošenje opštih podataka, koji se jednom unose, a kasnije koriste na ekranima Obrada.
- **Ograničenja** - ekrani za unos odluka koje predstavljaju ograničenja i pravila za rad.
- **Obrada** - ekrani koji odgovaraju primitivnim procesima iz strukturne sistem analize, a stavke menija procesima višeg nivoa. Dakle, kompletan meni *Obrada* odgovara stablu procesa iz SSA, a ekrani koji se pokreću odgovaraju primitivnim procesima.
- **Izveštaji** - ekrani za unošenje parametara za prikaz podataka i štampu standardnih ili lokalno potrebnih dokumenata (zvanični obrasci, pojedinačni podaci) i izveštaja (statistički podaci, podaci za period). Ovde se prikazuju izlazni tokovi podataka iz sistema prikazani u strukturnoj sistem analizi.
- **Upiti** - ekrani za unos parametara (jedan ili više kriterijuma istovremeno) za pretragu i prikaz podataka. Ovde je potrebno da se osmisle potrebni upiti sistemu, najfrekventniji
- **Servis** – Pomoć i uputstvo za korišćenje, About, Servis baze podataka (backup, brisanje, reindeksiranje, učitavanje starih podataka), Korisnik programa

PRIMER: ***IS studentska služba***

Šifarnici - Mesto, Vrsta finansiranja, Obrazovni profil, Nastavnik, Predmet itd.

Ograničenja - Upis (broj studenata koji je odobren za svaki profil za neku školsku godinu, broj ispita za upis u narednu godinu studija, nazivi ispita za uslov za upis u narednu godinu), Nastava (max i min broj studenata u jednoj nastavnoj grupi, max broj nastavnih časova u bloku jednog nastavnog predmeta), itd.

Obrada - Upis studenata: Evidentiranje prijava za prijemni, Formiranje spiska za polaganje prijemnih, Evidentiranje rezultata prijemnih ispita itd.

Izveštaji - Dokumenti (Uverenje o položenim ispitima, Spisak za prijemni ispit), Statistika (Prolaznost na ispitima u X ispitnom roku po predmentima itd.).

Upiti - Unapred uređeni upiti npr. Student - jedan ili više (Kriterijum pretrage: X prezime, X ime, X broj indeksa ili JMBG, Traženi podaci: godina studija, prosek ocena itd.) ili "custom" upiti (prikaži sve kandidate na prijemnom ispitu koji su iz Gimnazije u X ispitnom roku).

Servis - npr. za Korisnika programa (ime, prezime, korisničko ime, lozinka, status - privilegije).

2.7.3. DIZAJNIRANJE KORISNIČKOG INTERFEJSA

Pre početka postupka dizajniranja softvera potrebno je utvrditi ciljeve IS kao i ciljeve softvera (koncepcijski utvrditi unapred ciljeve i principe). Primer principa koji se postavljaju pre početka dizajniranja softvera: razvojnost, funkcionalnost, fleksibilnost (oprema, matični podaci), ekonomičnost, efikasnost, adekvatnost, pouzdanost, ergonomska usklađenost, jednostavnost, jezik komunikacije, informisanost, korektnost, unificiranost.

ŠTA JE KORISNIČKI INTERFEJS?

Komunikacija korisnik-računar treba da bude slika realnih potreba korisnika i njegovog posla, a ne slika vizije projektanta. Korisnici treba da urade neki posao, a računar treba da im u tome pomogne. Korisnik želi da koristi računarski sistem u svom poslu kao i svaki drugi alat, sa privikavanjem, ali da poveća efikasnost i ne remeti dotadašnji proces rada.

Interakcija je stil upravljanja. Korisnički interfejs je softver koji posreduje između čoveka i programa oblikujući računar u alat specifične namene. Detaljnije o korisničkom interfejsu videti u [MALBAŠKI, 2001].

VRSTE DIJALOGA

Korisnički interfejs obezbeđuje komunikaciju korisnika sa računarom putem dijaloga. Razlikuju se generalno dva tipa dijaloga:

SEKVENCIJALAN - korisnik unosi (obično putem tastature) komandu računaru šta sledeće treba da uradi. To je zahtev-odgovor komunikacija na nekom komandnom jeziku, koji ima strogu sintaksu (komande u obliku glagol-imenica). Dijalozi su funkcionalno orijentisani (funkcije sistema su u prvom planu). Jednostavniji su za projektovanje. Problemi: kruta sintaksa, sekvencijalnost...

ASINHRON - omogućava korisniku da u jednom trenutku izda bilo koju od dozvoljenih komandi. Pojavom miša, grafičkog prikaza i ikone se javlja kao dominantan. Pojam dijaloga sa direktnom manipulacijom je ustanovljen i zasnovan je na principu pokaži i pritisni. Objektno je koncipiran. Problemi: složeniji za projektovanje, ikone često ne daju dobru asocijaciju korisniku i pretrpanost interfejsa ikonama koje mogu da zbune korisnika.

U praksi, asinhroni dijalozi preovladavaju, ali ima elemenata i sekvencijalnosti.

2.7.3.1. KORISNIČKI INTERFEJS I ŽIVOTNI CIKLUS SOFTVERA

Zahvaljujući promeni stava projektanata da treba više pažnje obratiti na korisnički interfejs (program je proizvod, namenjen korisniku i tržištu), korisnik se više uključuje u proces izrade programa, tj. menja se životni ciklus softvera.

U praksi specifikacija zahteva se vrši tako da se najpre evidentiraju zahtevi za funkcionalnošću sistema, a zatim zahtevi za korisničkim interfejsom. Analiza rezultata ostalih faza vrši se kada se uradi početni prototip. Analiziraju se estetski izgled, stepen potrebnog memorisanja od strane krajnjeg korisnika, lakoća rada, ergonomičnost, funkcionalnost, stepen fizičke aktivnosti korisnika da bi se izvršio zadatak najfrekventnijeg posla, lokalnost (delovi interfejsa koji podležu kulturnim specifičnostima korisničke grupe) i internacionalnost, vernost realnom sistemu. Razvoj prototipa je iterativni proces. Pri tome se analiziraju razna stanja interfejsa i potrebno je postići što veću nezavisnost interfejsa od funkcionalnog dela. Potrebna je real-time povratna veza sa korisnikom-kada korisnik aktivira neku akciju treba da dobije indikaciju da li je zahtev prihvaćen i da čeka na odgovor.

2.7.3.2. IMPLEMENTACIJA KORISNIČKOG INTERFEJSA

Implementacija interfejsa se sprovodi u praksi pomoću:

- toolkit - skup elemenata interfejsa su implementirani u programskoj biblioteci. Uključuju se u interfejs pomoću programskih poziva. Za iskusne.
- interface builder - gotovi elementi interfejsa-pokaži i pritisni. Posle se definiše njihovo ponašanje. Za manje iskusne.
- user interface management system (Delphi, C++).

Trend ide ka automatizaciji svih faza razvoja korisničkog interfejsa, na osnovu modela zadataka, objekata rada, stilova prezentacije i dijaloga se generiše optimalni interfejs. Modeli se grade na osnovu specifikacije zahteva korisnika. U praksi je pokazano da ipak čovek formira izgled interfejsa na osnovu predloženih solucija.

Tako je u Visual Basic 5.0 prisutan ADD-In Manger kojim se uključuju dodatni alati za brzo i standardizovano pravljenje aplikacije. Tako se uključuju razni alati tipa Tools i Wizards, koji omogućuju brzo kreiranje formi uz neophodne manje prepravke, tj. prilagođavanja na konkretne zahteve korisnika. Jedan od njih je i Application Wizard koji kreira celu aplikaciju. Tu je i Data Form Wizard, koji kreira forme vezane za bazu: pojedinačni unos, tabela, pojedinačni+ tabela (master+detail->za povezivanje dveju tabela koje su vezane ključem).

2.7.3.3. PRINCIPI IZRADE KORISNIČKOG INTERFEJSA

Na ovom mestu se navode principi do kojih su autori došli u izradi seminarskih radova na kursu *Informacioni sistemi*. Inače, o opštim principima korisničkog interfejsa videti u [MALBAŠKI, 2001].

1. Razdvojiti funkcionalnost (svrha rada programa) i korisnički interfejs
2. Osnovna dva aspekta KI su upravljački (upravljački pult programa) i estetski (izgled, zvuk i sl.)
3. U svakom trenutku korisnik treba da jasno zna stanje programa (aktivne opcije, čekanje zbog snimanja i sl.)
4. Ekran ne sme biti pretrpan informacijama, funkcijama i bojama

5. Ekran treba da sadrži što više informacija
6. Put do informacija treba da je što kraći
7. Put do aktiviranja funkcija treba da je što kraći
8. Poruke i upozorenja uvek da su na istom mestu na ekranu
9. Program mora da ima help sistem
10. KI treba da bude podešljiv
11. Princip upravljanjem mišem - koristiti različite pokazivače miša (standardne Windows) za određena stanja i pozicije u programu.
12. Princip upravljanja ekranom - prikaz u uklanjanje ekrana i osvežavanje monitora u skladu sa potrebama
13. Principi rukovanja bojom - poslovne aplikacije ne smeju biti šarene, boja služi da određeni deo ekrana (u određenim stanjima programa) istakne (crvena-alarm; žuta-informacija, upozorenje), blink još više privlači pažnju, iste vrste poruka uvek istom bojom.
14. Principi rukovanja zvukom - zvuk se koristi u 2 smisla: 1 - kao alarm (u kombinaciji sa crvenom i blink), 2 – afirmativno (a- nakon dužeg posla koji računar radi samostalno, b- kod interaktivnog posla nakon segmenta koji računar radi samostalno)
15. **Za isti posao koji KI treba da omogući postoji više realizacija. Kreativnost je i umetnost proceniti koja varijanta najviše odgovara budućem korisniku.** (Primer: da li brisanje podatka raditi sa tasterom: Briši, sa tasterom sa ikonicom "kanta za otpatke", ili da se vrši prevlačenje: "drag and drop" sistem: korisnik uzme podatak i prevuče ga u kantu za otpatke). **Pri izboru varijante rukovodimo se sledećim faktorima korisnika: navike u radu (vrsta posla) i životu (lokalni običaji), znanje (stručno i o računarima), metodologija rada (standardna i lokalna), zamor (da je lako za korišćenje, što manje napora da bi se neki efekat sproveo) jezik komunikacije programa sa korisnikom.**

2.7.4. OSNOVNE FUNKCIJE U SOFTVERU INFORMACIONIH SISTEMA

1 - Prijavljivanje korisnika;
2 - Ažuriranje sadržaja neke tabele ili pogleda (1 tabela ili više, 1:M unos) kao realizacija unosa šifarnika ili pokrivanje procesa - dogadjaja realnog sistema;
3 - Prikaz alfanumeričkih podataka: tabelarno (Data Grid);
4 - Prikaz/unos slika (ukoliko ih ima);
5 - Arhiviranje podataka i manipulaciju sa fajlovima;
6 - Pretraživanje i prikaz podataka (sort, filter, upiti: jednostavni po ključnoj reči ili deo reči, po svim obeležjima ili složeni koji uključuju više kriterijuma);
7 - Izveštavanje (izveštaji) i unos većeg teksta;
8 - Prikaz statističkih i pojedinačnih podataka diagramima (Chart);
9 - Rad sa mapama.

NAJČEŠĆE PRATEĆE OPCIJE U SOFTVERU

1. Upozorenja, obaveštenja;
2. "Help" – sistem za pomoć;
3 "About" programa: autori, godina proizvodnje, registracija i sl.;
4. Forma za podešavanje rada programa “options”.

3. SOFTVERSKI ALATI I JEZICI ZA RAZVOJ INFORMACIONIH SISTEMA

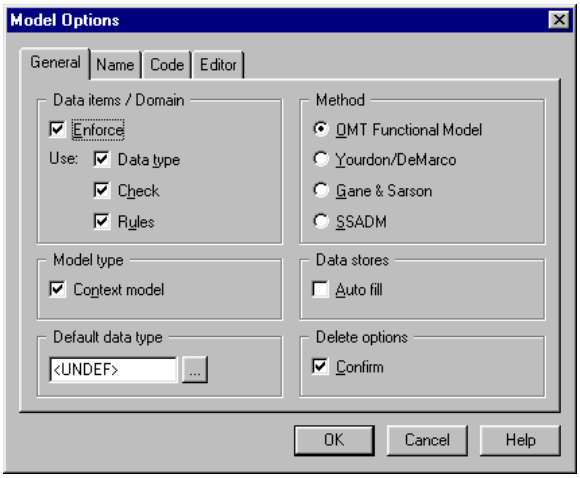
3.1. POWER DESIGNER

Power Designer je softverski paket firme Sybase namenjen analizi poslovnih i organizacionih sistema, projektovanju različitih modela informacionog sistema, dizajniranju softverskih delova IS, kao i dokumentovanju u razvoju IS.

3.1.1. PROCESS ANALYST

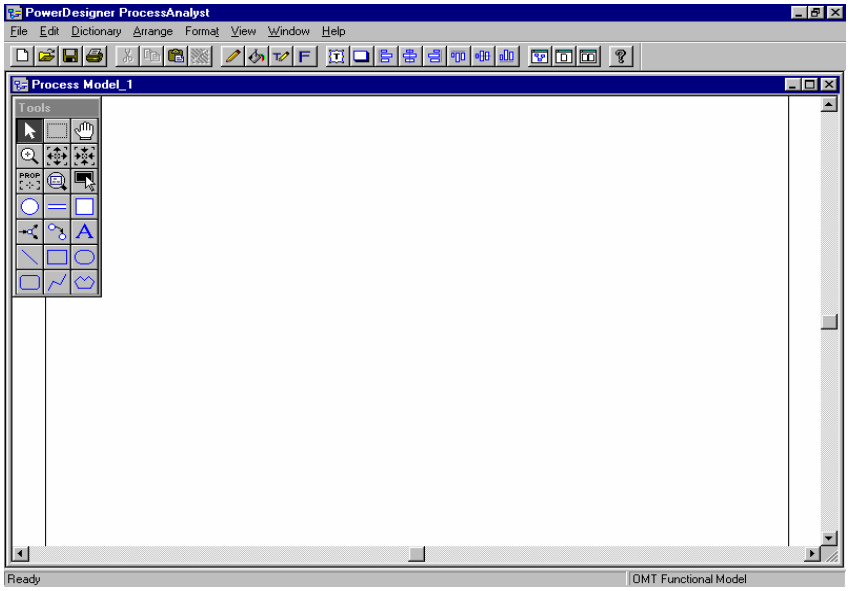
Process Analyst je alat za kreiranje modela procesa IS. Ovaj program je namenjen podršci kreiranju funkcionalnih modela koji odgovaraju različitim metodama: Object Modeling Technique (OMT), Yourdon/DeMarco, Gane & Sarson i Structured System Analysis and Design Methodology (SSADM). U okviru ovog alata moguće je vršiti dekompoziciju procesa, kreirati rečnik podataka, određivati i opisivati elementarne procese obrade podataka, kreirati i štampati izveštaje iz modela, kao i koristiti OLE tehnologiju za povezivanje vaših modela sa drugim aplikacijama.

Prvi korak u kreiranju novog modela procesa jeste kreiranje novog diagrama toka podataka (DTP). Izabrati opciju: File -> New Model.



Slika 3.1. Ekran za određivanje vrste modela procesa

Na kartici *General* izabrati Method: Yourdon/DeMarco za klasičnu SSA i otkačiti *Context model* u slučaju da je dijagram koji želimo nacrtati kontekst dijagram, kako bi njegova oznaka bila 0. Postaviti, takođe, *Auto fill* svojstvo na *True*, kako bi CASE alat automatski punio skladišta podataka elementarnim podacima.



Slika 3.2. Ekran sa prvim "praznim" DTP-om

Sledeći korak jeste podešavanje osobina celog modela procesa, izborom sledeće opcije iz glavnog menija ProccessAnalyst-a:

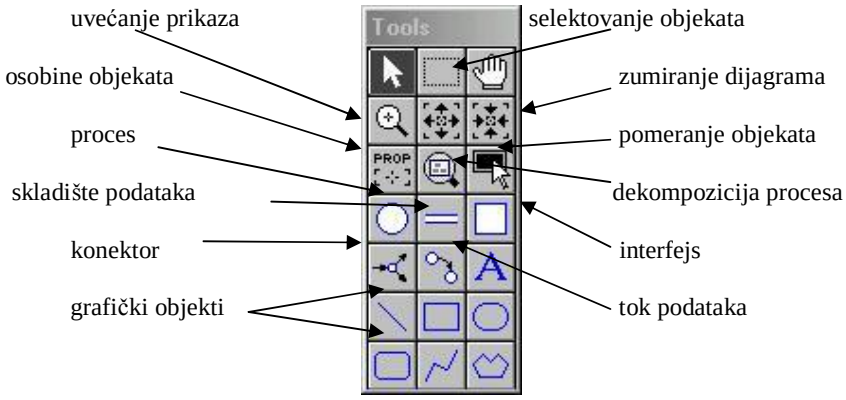
Dictionary -> Model Properties.

Na ovom ekranu se unose podaci o:

- nazivu i identifikatoru projekta (*Project Name & Code*);
- nazivu i identifikatoru modela (*Model Name & Code*);
- autoru (*Author*);
- verziji (*Version*);
- opisu dijagrama/modela (*Label*);
- datumu kreiranja modela (određuje sam CASE alat na osnovu sistemskog datuma).

Slika 3.3. Ekran sa osobinama modela procesa

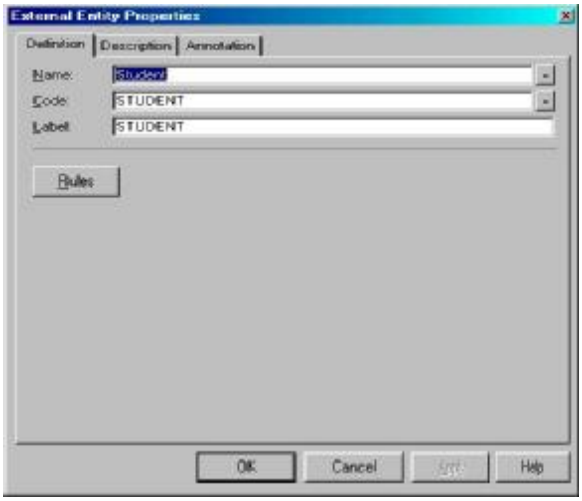
Paleta alata za modeliranje procesa sadrži objekte (*Tools*) za crtanje DTP-a ima sledeće opcije, tj. alate:



Slika 3.4. Ekran sa paletom alata za modeliranje procesa

3.1.1.1. CRTANJE DIJAGRAMA

Dijagrami tokova podataka se crtaju jednostavnim nanošenjem objekata sa palete alata (*Tools*) na radnu površinu dijagrama. Sledi definisanje osobina objekta (engl. *Properties*) preko dijaloga kao na slici ispod, koji se otvara dvostrukim klikom na izabrani objekat ili primenom alata *Properties* sa palete alata.



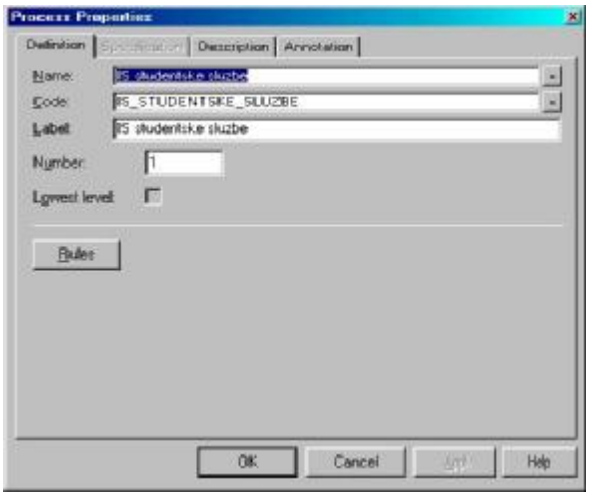
Slika 3.5. Ekran za definisanje osobina objekata modela

Bira se kartica *Definition* i upisuju sledeće osobine:

- Ime objekta (*Name*) - Imena su u ovom kontekstu naslovi ali ne identifikuju objekte, jer u modelu procesa, tj. njegovom rečniku podataka mogu postojati različiti objekti sa istim imenom. Ova činjenica je, naravno, u suprotnosti sa pravilom SSA da svaki objekat mora imati različit naziv. Inicijalno je maksimalna dužina naziva objekta 80 karaktera, uključujući velika, mala slova te karaktere. Korisnik CASE alata može promeniti ove «dozvoljene simbole».
- Kod objekta (*Code*) – kodna imena objekata jesu zapravo jedinstveni identifikatori svih objekata u projektu, modelu i rečniku podataka. Inicijalno, kodna imena imaju 80 karaktera maksimalno i uključuju samo velika slova.
- Oznaka objekta (*Label*) - Oznaka opisuje model ili objekt i obezbeđuje nešto više informacija nego ime ili kod. Labele su opcione, tako da polje može ostati nepopunjeno.
- U kartici *Description* se unose opisi modela, procesa, interfejsa. Opis obezbeđuje detaljne informacije o modelu ili objektima koje model sadrži.

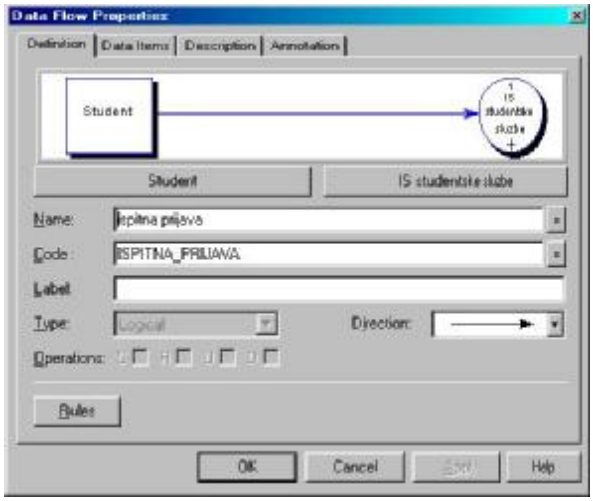
Na primer, opis spoljnjeg objekta Student u modelu Studentska služba glasi: “Osoba koja se u početku javlja studentskoj službi samo kao kandidat za upis na nekom od smerova fakulteta. Oni koji steknu uslove, postaju studenti. Zatim pohađjaju nastavu, polažu ispite i u toku svog školovanja postavljaju određene zahteve studentskoj službi”. Opis je slobodan tekst. Napomena *Anotation* sadrži napomene koje se tiču objekata na koje se odnosi. Napomena je, takođe, proizvoljan tekst od nekoliko redova.

Osobine procesa:



Slika 3.6. Ekran za određivanje osobina procesa

Prozor koji određuje osobine tokova podataka, za razliku od interfejsa i procesa, ima karticu *Data Items* u koju se unose elementarni podaci koji čine taj tok.



Slika 3.7. Ekran za određivanje osobina toka podataka

U zavisnosti od tipa toka podataka, proces može uraditi sledeće operacije (*Operations*) na objektima u skladištu podataka:

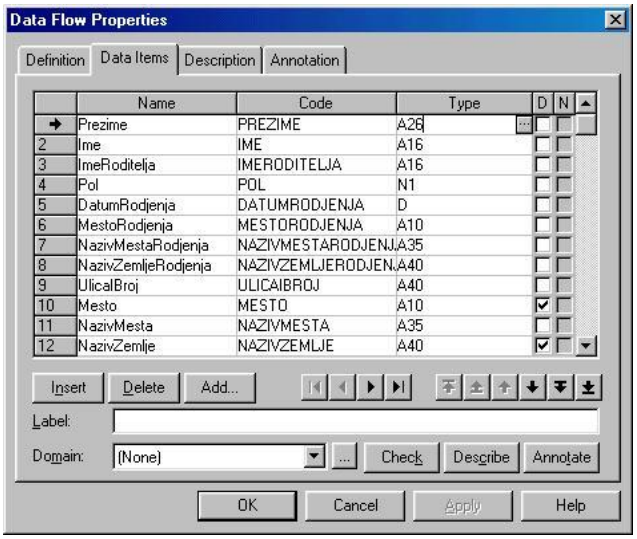
C - Kreiranje podataka u skladištu podataka (*Create*).

R - Čitanje podatka iz skladišta podataka (*Read*). Inicijalno za tokove koji izlaze iz skladišta podataka.

U - Ažuriranje podataka u skladištu (*Update*). Inicijalno za tokove koji ulaze u skladište podataka.

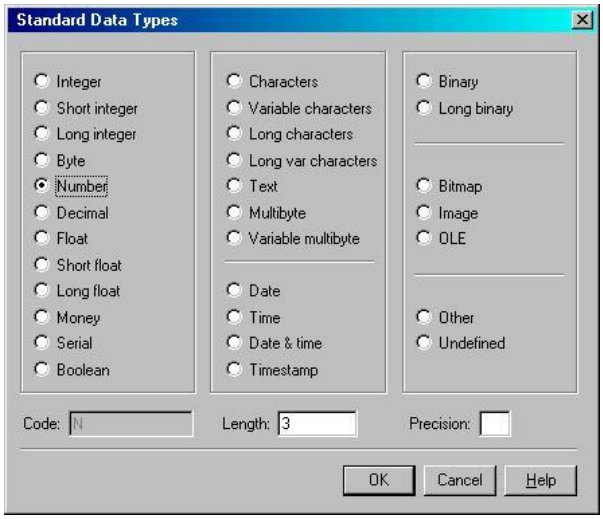
D - Brisanje podataka iz skladišta podataka (*Delete*).

Nakon definisanja naziva, koda i oznake toka podataka, pristupa se unosu elementarnih podataka koji su pored imena i koda određeni i tipom podatka (kolona *Type*):



Slika 3.8. Ekran za određivanje elementarnih podataka u okviru toka podataka

Tip podatka se određuje tako što se klikne na taster u polju *Type*, nakon čega se otvara sledeći prozor na kome se preko Option Button-a bira tip podatka, a za neke tipove unose i dužina (*Length*) i preciznost (*Precision*) - slika 3.9.



Slika 3.9. Ekran za izbor tipa podataka

Pregled tipova podataka:

Numerički tipovi podataka:
Integer – Celobrojni
Short Integer - Celobrojni (manjeg opsega)
Long Integer - Celobrojni (većeg opsega)
Byte – Celobrojni (nenegativni, najmanjeg opsega)
Number - Brojevi sa fiksnim decimalnim zarezom
Decimal - Brojevi sa fiksnim decimalnim zarezom
Float - Brojevi sa pokretnim zarezom
Short Float - Brojevi sa pokretnim zarezom (manjeg opsega)
Long Float - Brojevi sa pokretnim zarezom (većeg opsega)
Money - Brojevi sa fiksnim decimalnim zarezom – novčani tip podatka
Serial - Brojevi sa automatskim inkrementiranjem
Boolean – Logički tip podatka (true/false; yes/no; 1/0)

Karkter tip podataka:
Characters - Karakter string
Variable Characters - Karakter string
Long Characters - Karakter string
Long Var Characters - Karakter string
Text - Karakter string (Memo)
Multibyte - Multibajt karakter stringovi
Variable Mulitbyte - Multibajt karakter stringovi

Vremenski tip podatka:
Date - Datumski (Dan, mesec, godina)

Time – Vreme (Čas, minut, i sekunda)
Date & Time - Datum i vreme
Timestamp - Sistemski datum i vreme

Ostali tipovi podataka:
Binary - Binarni stringovi
Long Binary - Binarni stringovi
Image – Slike
Bitmap - Slike u bitmap formatu (BMP)
OLE - OLE linkovi
Drugo (Other) - Korisnički definisani tipovi
Nedefinisani (Undefined) - Još nedefinisani tipovi podataka

3.1.1.2. DEFINISANJE DOMENA

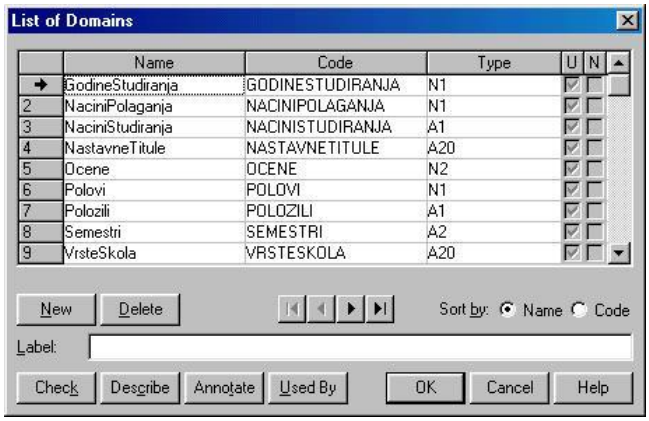
Domeni identifikujete tipove informacija u projektu. U PAM-u, je moguće pridružiti sledeće informacije domenu:

- Generički tip podatka, dužinu, i preciznost,
- Parametre provere (*Check*),
- Poslovna pravila (*Bussines Rules*).

Dužina i preciznost su osobine koje se ne primenjuju na sve tipove podataka. Šta više, u zavisnosti od tipa podatka, dužina može da označava maksimalan ili fiksni broj karaktera.

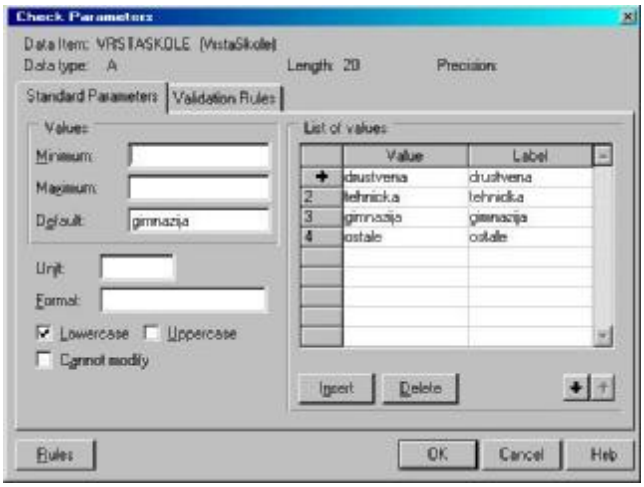
Za kreiranje korisnički definisanih (semantičkih) domena je potrebno izabrati opciju: Dictionary -> List of Domains. Pojaviće se lista domena (slika ispod).

Klikom na dugme *New* se pojavljuje nova linije u koju unosimo ime i kod novog domena, a zatim se bira tip podatka, eventualno dužina i preciznost, te ograničenja nad njim (*Check*).



Slika 3.10. Lista definisanih domena

Za uvođenje ograničenja nad elementarnim podacima usled specifičnosti realnog sistema koji se modelira ili zahteva korisnika se bira taster *Check* za selektovano polje, tj. elementarni podatak iz rečnika i otvara ekran (*Check Parameters*) kao na slici dole.



Slika 3.11. Ekran za određivanje ograničenja

Na ovom ekranu moguće je definisati:

- Vrednosti koje će elementarni podatak uzimati (*Value*).
- Oznake (*Label*) tih vrednosti.
- Minimalnu i maksimalnu vrednost opsega.
- Inicijalnu (*Default*) vrednost.
- Velika ili mala slova tekstualnih tipova podataka (*Lowercase* – mala, *Uppercase* – velika slova) itd.

Brisanje objekata - Ostvaruje se selektovanjem objekta i jednostavnim pritiskom tastera *Delete* na tastaturi ili izborom stavke iskačućeg menija na izabranom objektu: Edit -> Delete. Kada se briše simbol objekta, mora se naznačiti da li se jednostavno briše samo simbol iz grafičkog prikaza modela – dijagrama toka podataka ili se briše objekat iz rečnika. Ako je simbol samo grafički brisanje se vrši bez potvrde, ali ukoliko je simbol asociran (povezan) sa objektom u rečniku bira se između brisanja simbola vizuelno ili brisanje celokupnog objekta iz rečnika podataka.

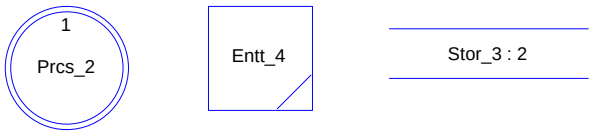
Kopiranje objekata - Postoje dva načna kopiranja objekata:

1. Dupliranje – kreiran je potpuno novi objekt u rečniku podataka koji ima identične osobine originalu, jedino se razlikuje kodno ime koje mora biti jedinstveno za sve objekte u

svim dijagramima, pomodelima i modelu. Na primer, ako dupliramo entitet Student, novi entitet imaće ime, tj. kod Student.

2. Postupak dupliranja se sastoji u izboru objekta na dijagramu i zatim: Edit -> Duplicate (ili CTRL+U)

3. Sinonimi - kreiranje dodatnih vizuelnih simbola za postojeći objekt čime se na različitim mestima u modelu, možemo poboljšati čitljivost i preglednost modela. Pri prikazivanju, simboli sinonima su različiti u zavisnosti od metode i tipa objekta:



Slika 3.12. Sinonimi objekata

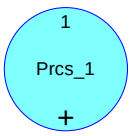
Za kreiranje sinonima objekta je potrebno odaberati objekat a zatim izabrati opciju: Edit -> Create a Synonym.

Za dekompoziciju procesa potrebno je odabrati dekompoziciju sa *Tools* palete i kliknuti na proces koji se želi dekomponovati ili prvo selektovati proces, otvoriti desnim tasterom miša iskačući meni i izabrati *Decompose* opciju. Pojavljuje se novi prozor za podprocese koji se crtaju i definišu na identičan način kao i nadređeni. Dekompozicija procesa se može izvršiti onoliko puta koliko to želi sistem analitičar.

Prelazak u viši nivo po hijerarhiji može se vršiti izborom: Dictionary -> Up One Level (ili Ctrl+A). Za dostizanje osnovnog (korenog) procesa, ponoviti ovu operaciju onoliko puta koliko je potrebno sve dok se osnovni proces ne pojavi.

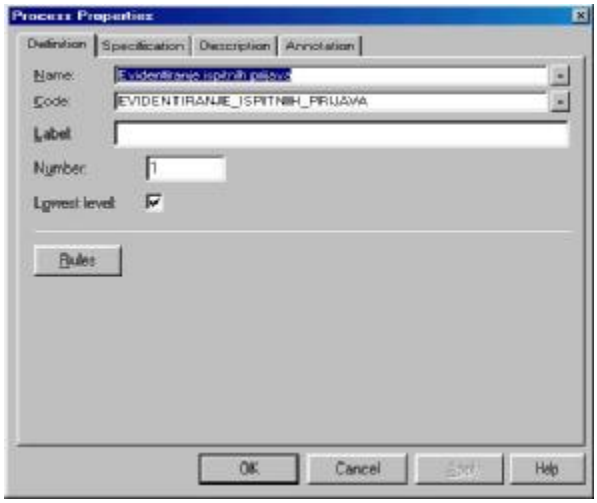
Identifikovanje dekomponovanog i nedekomponovanog procesa

Kada se koriste OMT ili Yourdon/DeMarco and Gane & Sarson metodologije, znak plus (+) označava da je proces dekomponovan, u suprotnom radi se o elementarnom, primitivnom procesu koji se ne može dalje dekomponovati:



Slika 3.13. Proces obrade podataka

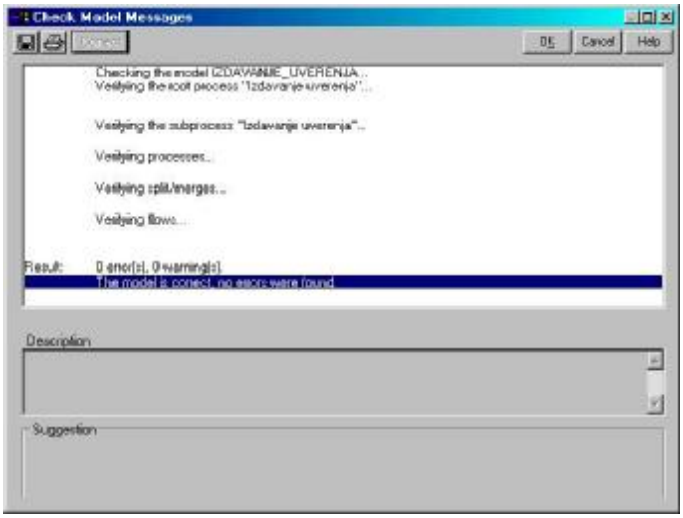
Označavanje primitivnog procesa se ostvaruje otvaranjem osobina tog procesa postavljanjem svojstva *Lowest level* (Najniži nivo) na vrednost *True* (sledeća slika).



Slika 3.14. Ekran za određivanje primitivnih procesa

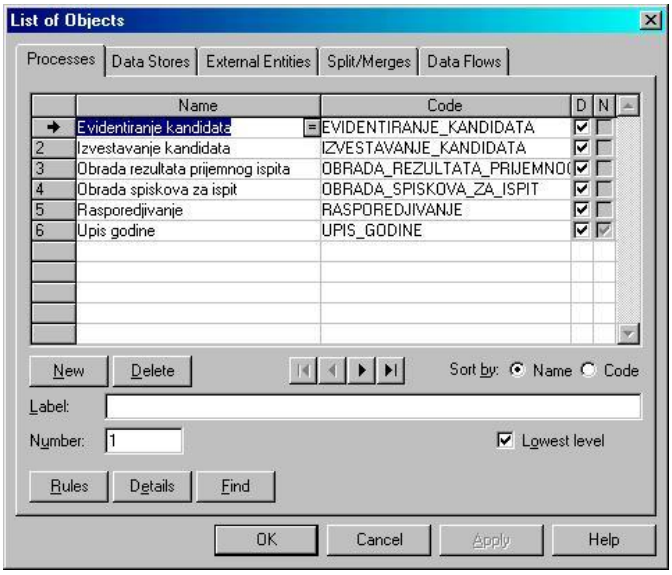
Provera ispravnosti nacrtanih dijagrama ne vrši sistem analitičar ručno, već to radi CASE alat, pomoću opcije: Dictionary –> Check Model (F4).

Važno je primetiti da se ovakva automatska provera korektnosti nacrtanih dijagrama vrši za trenutno otvoreni, a da bi se izvršila provera celog modela, mora se otvoriti kontekst dijagram. U slučaju je model procesa korektan i u skladu sa pravilima SSA dobiće se ekran kao na prethodnoj slici, u protivnom se ispisuju greške i upozorenja.



Slika 3.15. Provera ispravnosti modela procesa

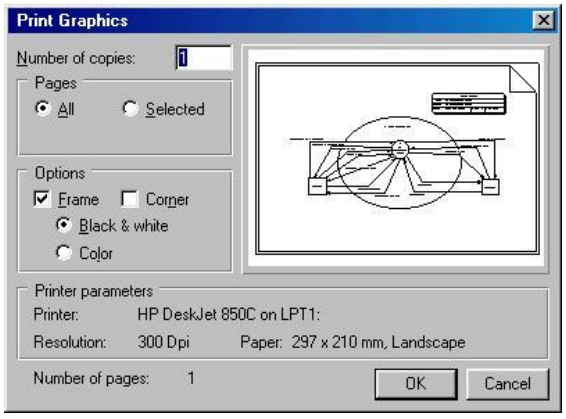
Ako postoje greške, one se otklanjaju izmenom dijagrama, njihovim elementima i rečnika podataka: Dictionary –> Objects i/ili: Dictionary –> List of Data Items.



Slika 3.16. Ekran za pristup objektima u rečniku podataka

3.1.1.3. KREIRANJE IZVEŠTAJA

Štampanje dijagrama tokova podataka - Ova opcija podrazumeva da je prethodno, u Windows-u, ispravno instaliran i podešen drajver za štampač. Samo štampanje dijagrama započinje ozborom sledeće opcije: File -> Print Graph. Pojaviće se dijalog prozor prikazan na slici 3.17.



Slika 3.17. Štampanje dijagrama

Kreiranje izveštaja CASE alata - ProcessAnalyst omogućava kreiranje celokupnog izveštaja o modelu koji smo uneli. Pri tome u izveštaj ulaze dijagrami tokova, opisi objekata, specifikacije primitivnih procesa, struktura skladišta i tokova podataka, detaljan opis elementarnih podataka. Šta će se sve nalaziti u izveštaju zavisi od toga koji tip izveštaja ćemo odabrati (standardni, potpuni, i liste) ili ćemo kreirati svoj sopstveni šablon za izveštaj. Kreiranje izveštaja se vrši na sledeći način: File -> Create Report.

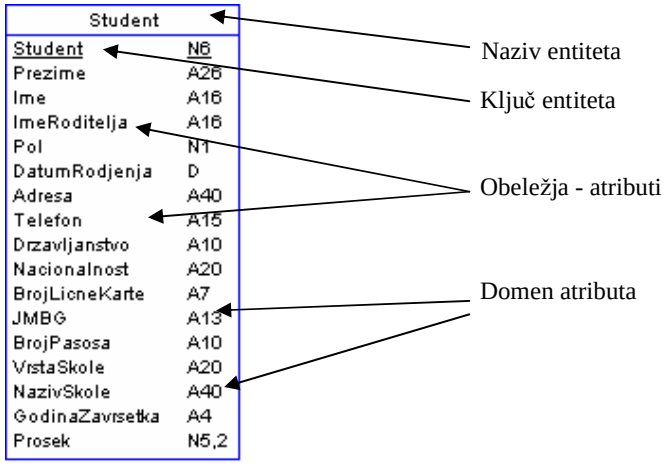
Pojaviće se dijalog prozor izgleda kao što je prikazano na narednoj slici (sledeća strana). Potrebno je odabrati jedan od ponuđenih šablona za izveštaje koji se razlikuju po obimu podataka koje uzmaju iz rečnika podataka.

Pregled funkcija tastera za kreiranje izveštaja ProccessAnalyst-a:

Print - Štampa izveštaj.
Save RTF - Snima izveštaj u Reach Text Format-u.
Preview - Mogućnost pregleda izveštaja.
Create - Kreiranje sopstvenog, u slučaju da smo upisali ime u polje Report Name.
Delete - Brisanje izveštaja.
Cancel - Odustajenje od akcije generisanja izveštaja.

3.1.2. DATA ARCHITECT

Grafički prikaz entiteta u CASE alatu Power Designer (Data Architect):



Slika 3.18. Grafički prikaz entiteta

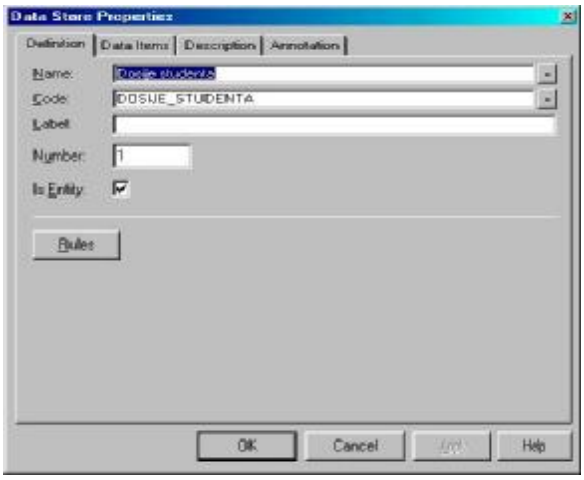
Prvi korak u kreiranju modela podataka jeste kreiranje novog modela u alatu DataArchitect i učitavanje modela procesa, kao rezultata rada u alatu ProcessAnalyst (.PAM) i to izborom opcije: File – Import – PowerDesigner ProcessAnalyst. Zatim se otvara prozor, koji nas pita

da izaberemo elemente modela procesa koji će biti učitani u DataArchitect. Celokupan rečnik podataka biće učitan, te svi elementarni podaci, već jednom definisani, domeni elementarnih podataka, skladišta i interna poslovna pravila, biće raspoloživi za dalje projektovanje baze podataka, na logičkom nivou.

Drugi korak u modeliranju podataka, je identifikovanje entiteta i određivanje koji će atribut (ili više njih) biti kandidat za ključ buduće šeme relacije (tabele u bazi podataka).

Heuristike za formiranje entiteta modela objekti veze obeležja:

Normalizacija skladišta - Izabrana ili sva skladišta se u ProcessAnalyst-u mogu proglasiti entitetima. Potrebno je na kartici Definition osobina određenog skladišta otkačiti (postaviti na logičku vrednost TRUE) opciju Is Entity. Prilikom učitavanja modela u DataArchitect, CASE alat će automatski generisati enititete od izabranih skladišta. Zatim je potrebno uočavati odnose između samih atributa i vršiti normalizaciju modela podataka.



Slika 3.19. Ekran za određivanje osobina entiteta

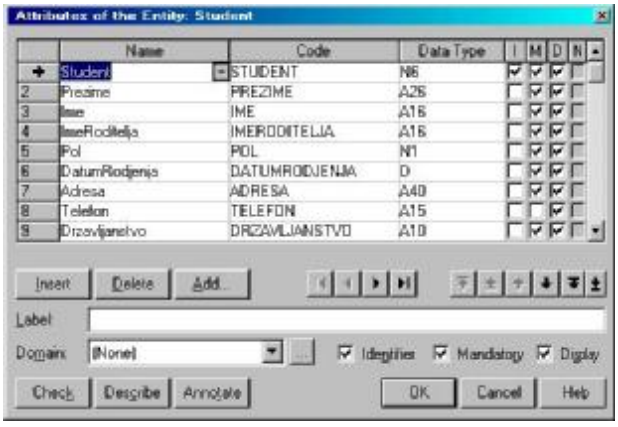
Pripadnost elementarnog podatka – Postavlja se pitanje: Čiji je elementarni podatak? Ako je isti ispravno uočen, definisan i imenovan (npr. ImeRadnika, PrezimeRadnika, AdresaRadnika, DatumRodjenjaRadnika i sl.), moguće je i formirati odgovarajuće entitete (u primeru - RADNIK).

Rečnik podataka – pravilno definisana struktura tokova podataka (pomoću {}, [], // i <>) i skladišta omogućava uočavanje klasa entiteta i veza između tih entiteta. Npr. Ispitni spisak se sastoji od 2 celine (2 entiteta): zaglavlje i telo (stavke) i sl.

```
ISPITNI_SPISAK:
<
ISPITNIROK,
       NAZIVSMERA,
       GODINA,
       SEMESTAR,
       PREDMET,
       NAZIVPREDMETA,
       DATUMPOLAGANJA,
       NACINPOLAGANJA,
       PREZIMENASTAVNIKA,
       IMENASTAVNIKA,
       PREZIMEISPITIVACA,
       IMEISPITIVACA,
       {<     REDNIBROJ,
              BROJINDEKSA,
              PREZIME,
              IME,
              NACINSTUDIRANJA,
              KOJIPUT
       >}
>;
```

Analizom opisa posla uočavaju se entiteti kao imenice ovog opisa, a glagoli kao veze između entiteta.

Prikaz ekrana za definisanje atributa entiteta:



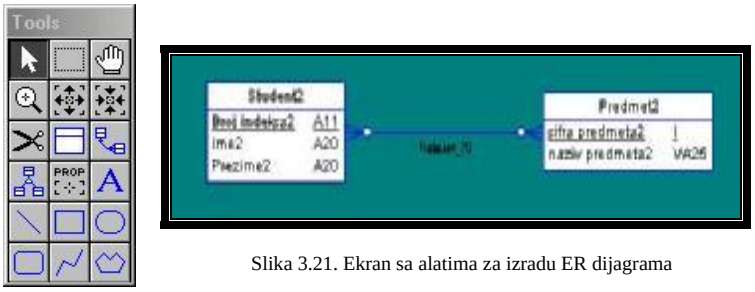
Slika 3.20. Ekran za određivanje osobina atributa

Dodavanje novog svojstva, tj. atributa u entitet, se vrši izborom opcije Add, pri čemu se otvara prozor sa listom svih elementarnih podataka i preuzima se jedno ili više podataka koja postaju atributi.

Značenje osobina atributa: Mandatory – obavezan podatak, isto što i Not Null vrednost atributa. Ako je postavljeno na TRUE, znači da konkretna vrednost (podatak) atributa ne

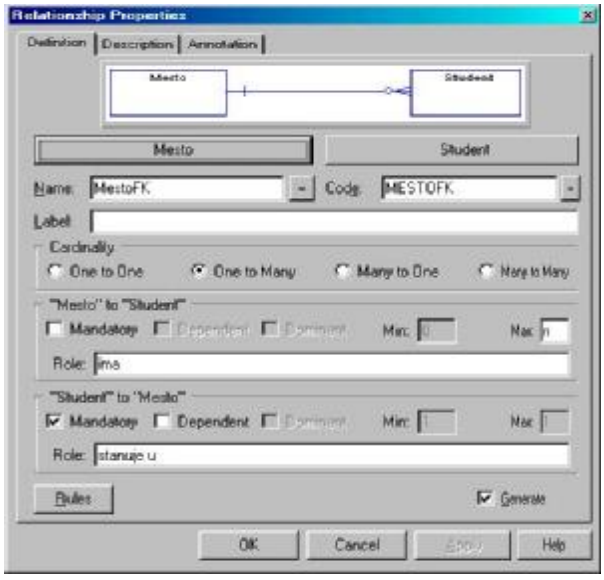
može biti nedefinisana (Null). Identifier je atirbut koji se proglašava za kandidata za primarni ključ entiteta. Uvek se nalazi na vrhu liste atributa entiteta. Display svojstvo, koje određuje, da li će atribut biti prikazan na ekranu ili ne.

U Power Designer-u se veza između 2 entiteta uspostavlja jednostavnim izborom opcije *Relationship* sa palete alata (*Tools*) i prevlačenjem iz površine jednog entiteta u površinu drugog entiteta, koji su u međusobnoj vezi:



Slika 3.21. Ekran sa alatima za izradu ER dijagrama

Sledi određivanje naziva veze (glagol, glagoslka konstrukcija koja proističe iz prirode same veze), vrste veze između entiteta (1:1, 1:M, M:M) i kardinaliteta preslikavanja preko ekrana *Relationship Properties*, koji se otvara izborom opcije *Properties* na tri načina ravnopravno (dvostruki klik na grafičkom prikazu veze, jednostruki klik koji daje iskačući meni ili izborom opcije sa palete alata).



Slika 3.22. Ekran za određivanje osobina procesa

Naziv veze se upisuje u polje *Name*, dok polje *Code* predstavlja jedinstveni identifikator objekta u celokupnom modelu podataka, analogno radu u Process Analystu. Tip veze se određuje preko polja (*option button*-a) *Cardinality*. Gornja granica kardinaliteta je određena tipom (vrstom) veze i može biti: 1,1 (veza jedan prema jedan); 1,n i n,1 (veza jedan prema više); n,n (veza više prema više).

Donju granicu kardinaliteta određuje polje *Mandatory* i to u 2 smera, posebno. U primeru kao na slici, postavljaju se uvek 2 pitanja koja određuju donju granicu kardinaliteta:

SMER MESTO PREMA STUDENTU: Da li za bilo koju pojavu prvog entiteta (levi na slici) mora postojati (u evidenciji u BP, a ne i u fizičkom, realnom sistemu) njemu zavisna pojava u drugom entitetu? Ako je odgovor potvrdan bira se *Mandatory* (*True*) u protivnom, ne.

SMER STUDENT PREMA MESTU: Da li za bilo koju pojavu drugog entiteta (desni na slici) mora postojati njemu zavisna pojava u prvomom entitetu? Ako je odgovor potvrdan bira se *Mandatory* (*True*) u protivnom, ne.

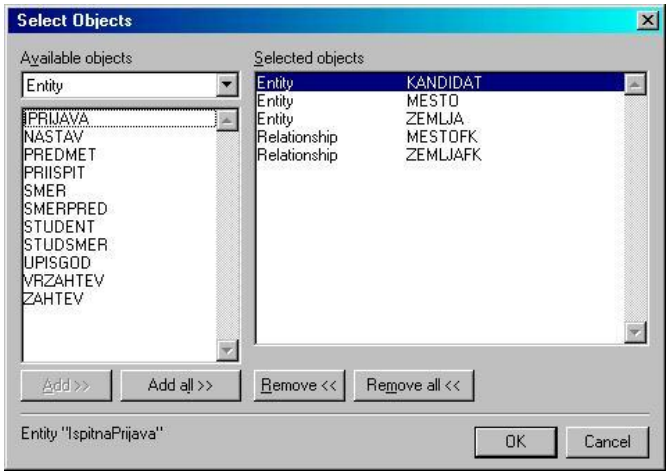
Primer: Zemlja ima mesto, mesto pripada zemlji (misli se na državu). Kada se pitamo za kardinalitet veze, tada se ne posmatra da li zemlja fizički mora da IMA grad (mesto) da bi postojala, nego da li je za evidenciju u tabeli zemalja neophodno da postoji i u tabeli mesta bar 1 mesto koje odgovara toj zemlji. Odgovor je Ne, tako da je za vezu Zemlje prema Mesta *Mandatory=False*.

Dependent znači identifikacionu zavisnost entiteta od entiteta, a prikazuje se na odgovarajućem preslikavanju. To se javlja kod slabih objekata i agregacija. To znači da će onaj entitet koji je dependent (identifikaciono zavistan) dobiti primarni ključ ovog drugog entiteta ali u funkciji dela njegovog primarnog ključa (entitet ima svoj ključ + dobijeni, koji zajedno čine složeni primarni ključ u slučaju JAK-SLAB objekat ili bez svog ključa ako je u pitanju AGREGACIJA).

Dominant se uključuje samo ako je veza 1:1. Dominantan je onaj koji je stariji u vremenu (opet u u evidenciji u BP, a ne i u fizičkom, realnom sistemu). Polje *Generate* se odnosi na to da li će od izabranog entiteta nastati tabela ili ne.

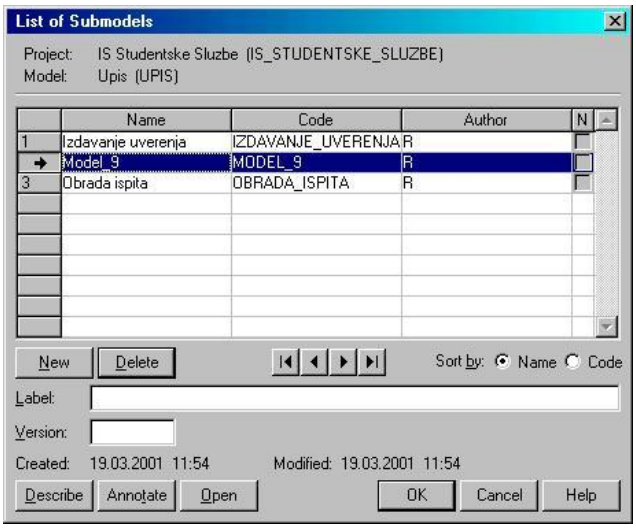
Napomena: U Power Designeru se kod svake relacije (1:M) ubacuje na strani M ključ iz strane 1, tako da nije potrebno ubacivati polje(a) koje učestvuje u relaciji. To ubacivanje se vrši kada se sa Konceptualnog prelazi na Fizički model (Generate Physical Model).

Konvencija: Primarni ključ treba u navođenju u tabeli unositi na prvom mestu (najviša pozicija u tabeli definicija). Ako smo uneli negde dole, pomera se naviše strelicama sa strane. U meniju Dictionary, se za svaki primitivan proces bira opcija: Dictionary -> Submodel -> New. Nakon otvaranja podmodela se dodaju, izborom (selektovanjem), entiteti potrebni za izvršavanje (rad, funkionisanje) tog primitivnog procesa. Prebacivanje izabranih entiteta u podmodel, vrši se tasterom Add.



Slika 3.23. Kreiranje podmodela

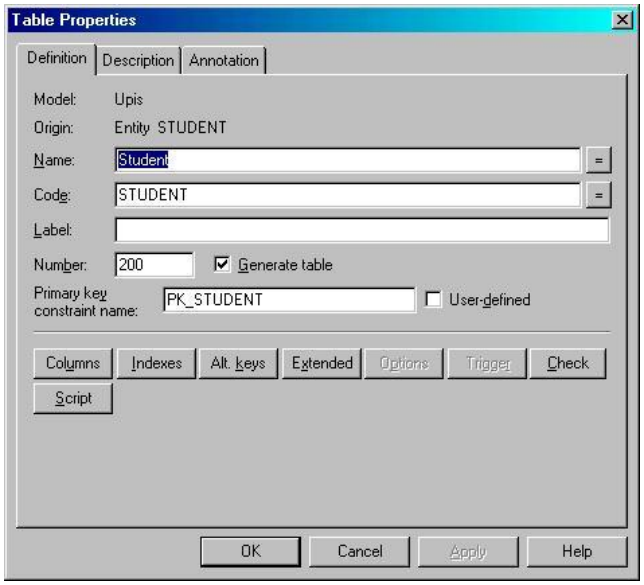
Kako se entiteti pojavljuju u prozoru Selected objects, automatski CASE alat dodaje i veze koje postoje između tih entiteta. Ukoliko se ne otvara automatski ovaj prozor (bio je selektovan neki entitet npr.), potrebno je pokrenuti opciju: Dictionary -> Submodel -> Add/Remove Objects, te će se otvoriti gore prikazani ekran. Imena podmodela su nazivi primitivnih procesa, a nakon izbora opcije: Dictionary -> Submodel -> List of Submodels, gde se određuju ime - Name i jedinstveni identifikator – Code, za određeni podmodel sa liste postojećih podmodela.



Slika 3.24. Izmena naziva podmodela

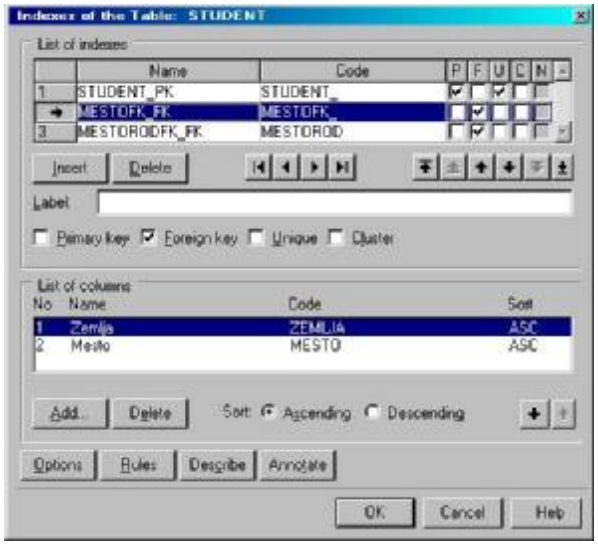
Semantički ključevi se u fizičkom modelu podataka, u DATA ARCHITECT-u dodaju iz razloga što je primarni ključ u vudi brojača (Sifra ili ID) dovoljan da se uspostavi veza između 2 entiteta i minimalan u pogledu dužine polja koja se prenosi u druge tabele (izbegava se nagomilavanje složenih primarnih ključeva), ali ne sprečava redundansu i višestruko memorisanje istih podataka. Ovo je veoma opasno za integritet podataka u bazi i može dovesti do grešaka koje se kasnije izuzetno teško otkrivaju i ispravljaju.

Postupak dodavanja semantičkih-alternativnih ključeva je sledeći: Dvostruki klik levim tasterom miša na entitetu, ili jednostruki desni, pa potom opcija Properties otvara osobine tog entiteta kao što je prikazano na narednoj slici.



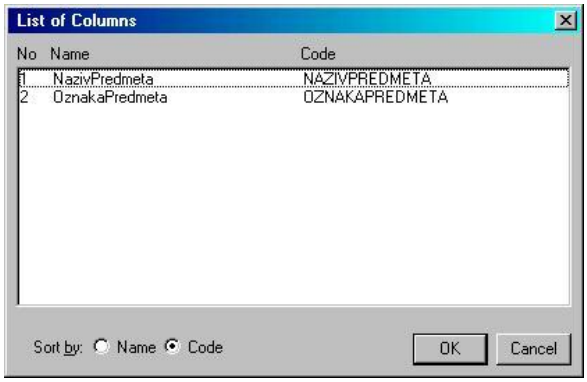
Slika 3.25. Ekran za određivanje osobina tabela

Sledi otvaranje dijaloga za definisanje indeksa, gde se opcijom Insert dodaje novi indeks koji će biti sastavljen iz nekoliko polja i koji će imati osobinu jedinstvenosti Unique = True, a neće biti primaran:



Slika 3.26. Ekran za određivanje osobina indeksa

Izbor polja (kolona) buduće tabele, koja čine semantički indeks, se vrši pomoću tastera <Ctrl> + Levi klik na ponuđene atribute entiteta:



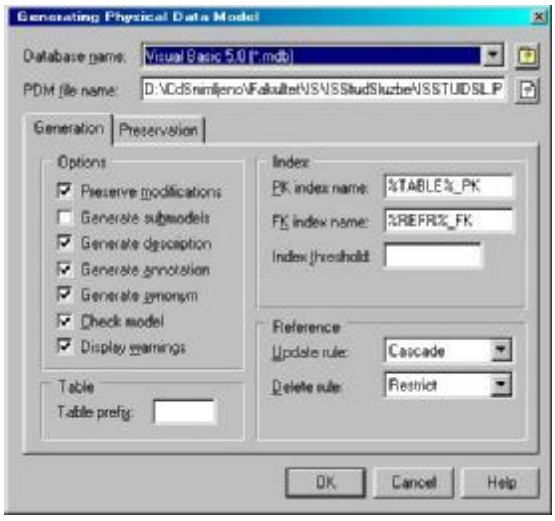
Slika 3.27. Izbor kolona za indeksa

Napomena: Ako je prvi korak u kreiranju modela podataka u DataArchitect-u, bio učitavanje modela procesa, kao rezultata rada u alatu ProcessAnalyst (.PAM), treba voditi računa da se atributi koji se proglase identifikatorom, tj. ključem, MORAJU prvo obrisati iz svih drugih entiteta, u kojima se nalaze.

3.1.2.1. PREVOĐENJE LOGIČKOG U FIZIČKI MODEL PODATAKA

Kod prevođenja logičkog u fizički model podataka: Dictionary -> Generate Physical Model, prvo se otvara prozor koji nas pita da li želimo da snimimo model ili ne, te se pojavljuje ekran koji prethodi postupku prevođenje. Tu se bira sledeće:

- Format buduće baze podataka (Database Name),
- Putanja fizičkog modela (PDM File Name),
- Elementi fičkog modela (Options),
- Stil naziva indeksnih fajlova (Index) i
- Referencijalni integritet (Reference).



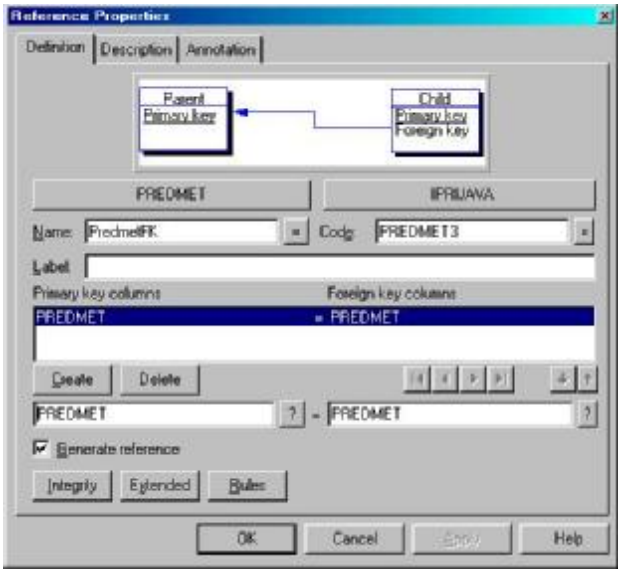
Slika 3.28. Kreiranje fizičkog modela podataka

U Data Architect-u se očuvanje referencijalnog integriteta određuje za svaku vezu između tabela pojedinačno (prikazano na nnarednoj slici):

Za operaciju UPDATE (dodavanje novih i izmena postojećih podataka u tabeli). Moguća su tri slučaja:

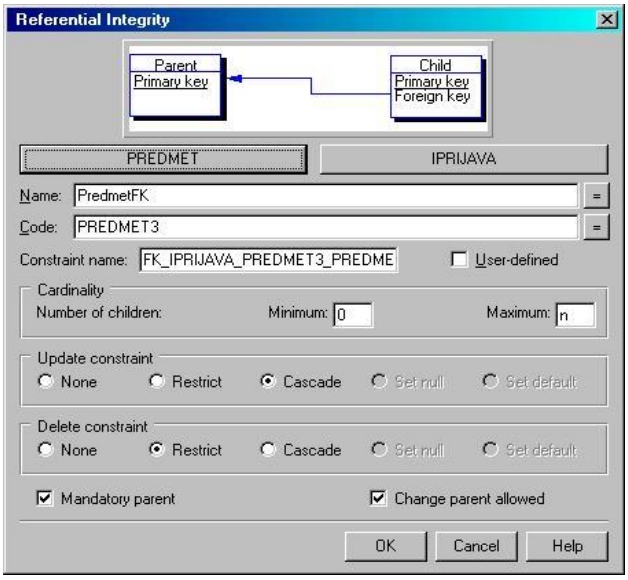
None – Ne postoji očuvanje referencijalnog ontegriteta. Ukoliko se u tabeli roditelja (strana 1) promeni vrednost primarnog ključa (slog, n-torka), ostaju viseći podaci u tabeli dete (strana M - više). Tim slogovima se više ne može pristupiti, jer imaju staru vrednost prim. ključa. NIKAKO NIJE DOBAR IZBOR.

Restrict – Restriktivno očuvanje referencijalnog ontegriteta. Ukoliko se u tabeli na strani 1 promeni vrednost primarnog ključa, a postoje na strana više vezani slogovi, BP ne dozvoljava promenu te vrednosti primarnog ključa. Samo u slučaju da ne postoji upotrebljen prim. ključ na strani više, moguće će biti promeniti vrednost prim. ključa.



Slika 3.29. Podešavanje osobina relacije

Cascade – Kaskadno očuvanje referencijalnog ontegriteta. Ukoliko se u tabeli na strani 1 promeni vrednost primarnog ključa, a postoje na strana više vezani slogovi, BP automatski vrši promenu te vrednosti na strani više.



Slika 3.30. Određivanje referencijalnog integriteta

Za operaciju DELETE (brisanje nepotrebnih podataka iz tabele). Takođe su moguća tri slučaja:

None – Ukoliko se u tabeli na strani 1 obriše podatak, ostaju "viseći podaci" u tabeli na strani M-više. Tim slogovima se više ne može pristupiti, jer imaju staru vrednost prim. ključa. TAKOĐE, NIKAKO NIJE DOBAR IZBOR.

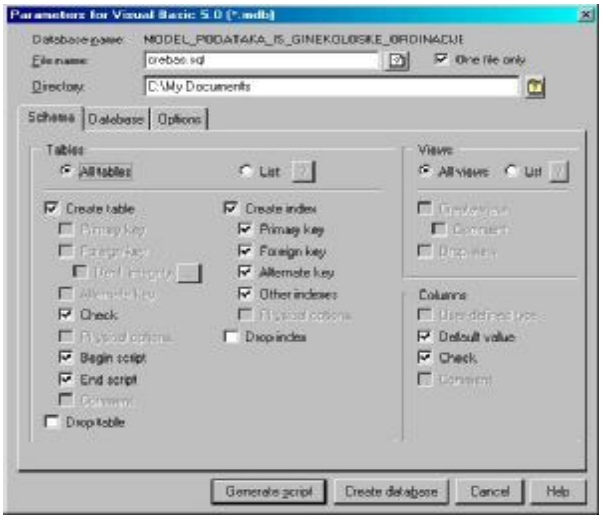
Restrict – Ukoliko se u tabeli na strani 1 obriše podatak, a postoje na strana više vezani slogovi, BP ne dozvoljava ovo brisanje jer je primarni ključ upotrebljen. Samo u slučaju da ne postoji upotrebljen primarni ključ na strani više, moguće je brisati podatak.

Cascade – Ukoliko se u tabeli na strani 1 briše vrednost primarnog ključa, a postoje na strana više vezani slogovi, BP automatski vrši brisanje svih slogova sa tom vrednošću i mnogo podataka na strani više može biti izgubljeno.

3.1.2.2. GENERISANJE BAZE PODATAKA

Fizički model podataka mora biti potpuno tehnički (sintaksno) ispravan, jer se u protivnom neće izvršiti generisanje baze. Ako se pojave greške, neophodno je prvo ih ispraviti i tek tada pristupiti generisanju baze podataka vrši na jednom nekoliko načina:

- Kreiranjem prazne baze podataka u nekom sistemu za upravljanje bazom podataka - SUBP (npr. Access, SQLServer), koja ne sadrži nijednu tabelu, pa se zatim iz Data Architect-a konektujemo na tu bazu opcijom: Database -> Connect i potom izaberemo: Database -> Generate Database;
- Snimanjem script fajla, koji u sebi sadrži SQL kod (sadrži naredbe za kreiranje polja, njihovih domena, indeksa, relacija među tabelama i sl.) i koji se zatim pokreće u nekom od SUBP-a (npr. FoxPro, DBase);
- Direktnim kreiranjem baze podataka (npr. Paradox, Delphi i sl.).



Slika 3.31. Kreiranje baze podataka

Napomena: U kartici Options, neposredno pre generisanja baze podataka, podesiti - ODBC (Open Database Connectivity – mehanizam za otvoreno povezivanje baza podataka različitih formata) Syntax = True.

Nakon generisanja baze podataka u izabranom SUBP, sledi definisanje aplikacije(a) u vidu projekta aplikacije, a zatim i izradi, testiranju, primeni programa, odnosno njegovom održavanju i dokumentovanju.

3.2. UPRAVLJANJE BAZAMA PODATAKA

3.2.1. TEHNOLOGIJE ZA RAD SA BAZAMA PODATAKA

Tehnologije pristupa bazama podataka opisane u [FORGEY i sar., 2002] su: ODBC, DAO, RDO, ODBCDIRECT, OLEDB, ADO, RDS i ADO.NET.

ODBC (*Open Database Connectivity*) je jedna od najčešće primenjivanih tehnologija za pristup podacima u bazama podataka, promovisana od strane *SQL Access Group* (SAG) 1992. godine. SAG je grupa osnovana 1989. godine kako bi se definisali i promovisali standardi koji bi programerima omogućili prenos lokalnih ili udaljenih sistema podataka, bez ponovnog kodiranja. Drugi cilj SAG-a je bio taj da se organizacijama i kompanijama omogući da centralizuju svoje baze podataka, kako se ne bi instalirale odvojene kopije baza podataka na svakom računaru klijenta. Ovu grupu su činili *IBM, Oracle, Microsoft, Hewlett-Packard, Sun, Digital* i *Informix*. 1990. godine je ova grupa izvršila kombinovanje potrebnih karakteristika u *Application Programming Interface* (API) koji bi omogućio aplikacijama da pristupaju različitim bazama podataka. Kompanije koje proizvode sisteme za rukovanje bazama podataka su od 1992. godine dodale svojim proizvodima podršku za ODBC i danas postoji preko 170 različitih tipova ODBC upravljačkih programa. ODBC omogućuje pristup širokom opsegu izvora podataka preko standardnog seta API-funkcija. Pomoću drajverskog upravljača prevodi naredbe i vraća rezultate aplikaciji. Ovaj mehanizam snabdeva programera jednim setom naredbi za opseg sistema koji ne zavise od platforme, baze podataka, sistema za upravljanje bazama podataka i programskog jezika. Pored svega navedenog, ODBC je kompleksan i neintuitivan za rad, a postoji i određena nekompatibilnost sa nekim drajverima.

DAO (*Data Acces Objects*) je tehnologija za pristup bazama podataka uvedena 1992. godine u *Microsoft Access 1.0*. Bazira se na *Microsoft*-ovom JET mehanizmu za rad sa bazama podataka koji omogućuje pristup eksternim ISAM izvorima podataka *dBase, Paradox, FoxPro, Btrieve* i ODBC. JET mehanizam koji je podržavao 16-bitne operacije tako da je bio primaran za *Microsoft Windows 3.x* okruženje. DAO je obezbeđivao mogućnost pristupa JET-ovim i ODBC-ovim izvorima opštim setom koda. Osnovne karakteristike DAO tehnologije su te da nema jednostavnu sintaksu, spor je, nedostajali su mu odgovarajući resursi i dizajniran je za aplikacije koje će raditi na stonom računaru, pa su i baza podataka i aplikacija morali biti na istoj mašini. JET je optimizovan za rad sa *Microsoft Access* bazom podataka, tako da je rad sa drugim izvorima imao za posledicu pad performansi., jer je bilo potrebno prevesti komande pre slanja u server baze podataka. Ipak,

i pored navedenih ograničenja dosta je bio u upotrebi jer je podržan za *Microsoft Office* i *Visual Basic* 3 koji su mogli koristiti ovu tehnologiju.

RDO (*Remote Data Objects*) je uveden 1995. godine sa pojavom 32-bitne verzije programskog jezika *Visual Basic 4*. Osnovna namena ove tehnike jeste pristup udaljenim bazama podataka ODBC, poput *Microsoft SQL Server* i *Oracle*, ali bez složene sintakse ODBC API-ja. Objektni model RDO je zasnovan na DAO i uključuje neke od najboljih karakteristika interfejsa i osnovnu funkcionalnost DAO, pri čemu se ne koristi JET mehanizam. Procesiranje podataka preko RDO je ubrzan korišćenjem inteligentnog kursora. Ovaj mehanizam je pokazao dobre performanse u pristupu bazama podataka koje nisu Accessove. U poređenju sa DAO ima manje manipulisanja upitima i setovima podataka, tako da je brži i zahteva manje dodatnog angažovanja. Ove dobre performanse proizilaze iz činjenice da RDO direktno komunicira sa ODBC API-jem, bez prolaska kroz neke druge slojeve. Ipak, nije bio dugo u upotrebi, jer su brzo uvedene druge tehnologije.

ODBCDIRECT je uveden 1997. godine u *Microsoft Visual Basic 5* kao alternativa za DAO, koja povezuje JET mehanizam i RDO tehnologiju. Omogućuje direktan pristup ODBC-ovim izvorima podataka, bez korišćenja *Microsoft JET Engine*. Koristi RDO-ovo jezgro, sa kojim je gotovo istovetan, ali sa DAO nazivima objekata. U poređenju sa DAO ima bolje performanse, resurse i pristup serveru, kao i bolje metode ažuriranja i upita. Loša strana je ta što ne može pristupati izvorima podataka koji nisu ODBC.

OLE DB (*Object Linking and Embedding for Database*) je *Microsoft* API uveden 1996. godine. Osnovna karakteristika ove tehnologije je univerzalni pristup podacima koji omogućuje komunikaciju sa relacionim i nerelacionim izvorima podataka, uključujući i one koji koriste COM (*Component Object Model*). Ova tehnologija obezbeđuje sve mogućnosti ODBC-a, ali je podeljena na dve komponente: korisnike i provajdere. Komponente korisnika koriste podatke, a provajderi komuniciraju sa podacima i prikazuju interfejs korisnicima. OLE DB se može koristiti i za različite tipove nerelacionih izvora podataka, kao što su: datoteke, grafici, elektronska pošta i dr. Ovo je danas jezgro *Microsoft*-ove tehnologije za rad sa bazama podataka. Direktan pristup OLE DB-u imaju samo programi napisani u C++ jeziku.

ADO (*ActiveX Data Objects*) je omot oko OLE DB, uveden 1996. godine kako bi se prevazišla složena sintaksa OLE DB-a. ADO ima slične osobine kao DAO i RDO, lako se uči i ima dobre performanse jer zahteva minimalan mrežni saobraćaj. ADO tehnologija koristi mogućnost OLE DB da pristupa izvorima podataka različitih sistema za rukovanje bazama podataka. Ovaj model je dizajniran tako da omogućuje nekonektovan način rada, što znači da se podatak nalazi u memoriji, a da pri tome ne postoji aktivna konekcija do baze podataka. Ovim se omogućuje rad sa podacima bez oslanjanja na mrežu, koja može biti nedostupna ili spora u kraćem ili dužem vremenskom periodu. ADO može do baze podataka uspostaviti i više konekcija, pri čemu se serverski resursi mogu proporcionalno oslobađati.

RDS (*Remote Data Source*) je veoma sličan ADO tehnologiji, ali je dizajniran tako da obezbedi OLE DB tehnologiju za Web aplikacije. Omogućuje manipulaciju nad podacima

na Web klijentu bez dodatnih poziva servera, oslobađajući tako resurse na serveru. Dizajniran je za nekonektovane izvore podataka više nego ADO.

ADO.NET je uveden 2000. godine sa pojavom *Microsoft Visual Studio .Net* razvojnog okruženja. Ovo je najnovija *Microsoft*-ova ADO i RDS tehnologija izgrađena na XML-u (Extensible Markup Language), industrijskom standardu za upravljanje podacima. XML-om može upravljati bilo koja aplikacija koja čita ovaj standard, bez obzira na platformu: *Windows, Linux, Unix* i sl. Takođe je upravljiv iz bilo kog programskog jezika: *C++, Delphi, Visual Basi*c i dr.

U sledećoj tabeli je dat prikaz osobina i razloga za i protiv svake od prethodno opisanih tehnologija za pristup bazama podataka [FORGEY i sar., 2002]:

Tehnologija	**Za**	**Protiv**
ODBC	Omogućuje vezu sa više različitih izvora podataka.	Složen rad sa ODBC API-jem.
DAO	Optimizovan za *Microsoft Access* baze podataka. Lako pristupa *Office*-ovim programima. Dobar za programe na stonom računaru. Lak rad.	Slabe performanse u radu sa bazama podataka koje nisu *Microsoft Access*. Kompleksna sintaksa.
RDO	Omogućuje daljinski pristup izvorima podataka poput *SQL Server* ili *Oracle*. Koristi jezgro funkcionalnosti DAO i brži je od DAO.	Zamenjen je brzo sa ADO-om.
ODBCDirect	Alternativni režim za DAO da se zaobiđe JET mehanizam i direktno komunicira sa ODBC.	Ne može se pristupati izvorima podataka koji nisu ODBC.
OLE DB	Omogućuje komunikaciju sa relacionim i nerelacionim bazama podataka korišćenjem COM-a. Ima sve karakteristike ODBC-a.	Izuzetno teško pisanje koda. Dostupan samo u programskom jeziku C++.
ADO	Obezbeđuje okruženje OLE DB tehnologije. Ima visoke performanse i lak je za korišćenje.	Od početka je dizajniran za konektovane arhitekture.
RDS	Koristi se u web aplikacijama za pristup bazama podataka kroz OLE DB tehnologiju. Primarno dizajniran za konektovanu arhitekturu.	Dostupan je samo Web aplikacijama.
ADO.NET	Dizajniran slično ADO-u i RDS-u. Podržava XML. Pojednostavljuje komunikaciju između različitih okruženja i porgramskih jezika.	Još nisu uočene.

Tabela 3.1. Uporedni prikaz tehnologija zapristup bazama podataka

3.2.4. FOXPRO

FoxPro je softverski proizvod kompanije Asthon Tate. Prvi proizvod ove softverske kuće je bio DBase II sistem koji je 1981. godine urađen za IBM PC kompatibilne računare. Usavršavanjem ovog softvera nastao je FoxBase sistem za upravljanje bazama podataka. Kompanija Microsoft je 1992. godine otkupila Ashton Tate i od tada se FoxPro razvija pod Windows okruženjem. Izvršeno je povezivanje ovog SUBP-a sa drugim sistemima za rad sa bazama podataka. Verzija FoxPro-a koja se pojavila na tržištu 1994. godine je proširena mogućnostima za rad sa objektima, razvijen je mehanizam za očuvanje referencijalnog integriteta. Daljim razvojem ovog sistema povećan je broj istovremeno otvorenih datoteka, automatske kontrole unosa podataka, alata za generatore maski, izveštaja, menija, rad u višekorisničkom okruženju, dodavanje novih tipova podataka, kreiranje složenih indeksa itd. Aktuelna verzija je FoxPro 8 [HENTZEN, 1996], [RADULOVIĆ, 1997].

3.2.3. SQL SERVER

SQL Server je sistem za upravljanje bazama podataka firme Microsoft koji je nastao od Sybase SQL Servera. Microsoft i Sybase su se udružili 1989. godine, oko razvoja servera baza podataka, razvijajući verziju SQL Servera za IBM OS/2. Windows NT verzija pojavljuje se 1993. godine, sa oznakom 4.2. Partnerstvo Microsoft-a i Sybase-a je raskinuto sa verzijom SQL Server 6.0. Od verzije 6.5 pa nadalje u pitanju je kompletan Microsoft-ov proizvod. Verzija 7 je praktično napisana ispočetka, dok je SQL Server 2000 poboljšana verzija prethodnog izdanja. SQL Server koristi upitni jezik koji je dijalekat SQL-a i naziva se Transact SQL, ili T-SQL. T-SQL je ANSI kompatibilan u smislu osnovnih SQL upita i operacija, koji su potpuno usklađeni sa standardom. Ipak, neka rešenja se mogu izvesti samo korišćenjem specifičnih T-SQL naredbi, a koje nisu deo standarda. Postoji šest varijanti MSSQL Servera: CE, Personal, Desktop Engine, Standard, Developer i Enterprise [WYNKOOP, 2000].

3.2.5. ORACLE

Oracle je prvi komercijalni sistem za upravljanje bazama podataka, proizveden od strane Oracle Corporation (Belmont, Kalifornija, SAD) 1979. godine. Prvobitna ideja tokom razvoja ovog SUBP-a je bila ta da se razvije softver koji će koristiti SQL jezik i imati mogućnost prenosivosti aplikacija i podataka na razne platforme. Ovo se postiglo u verziji ORACLE 3, koja se pojavila 1983. godine i koja je bila napisana u programskom jeziku C. U ovoj verziji je uveden rad sa transakcijama. Verzija 4 se pojavila 1984. godine za IBM računare sa MVS operativnim sistemom i verzija za personalne računare. Verzija 5 je uvela klijent/server arhitekturu, disribuirane upite, pristup podacima koji se nalaze na više lokacija. Verzija 6 je donela novine u vidu modula za upravljanje baferima, kreiranje rezervne kopije baze, zaključavanje na nivou sloga itd. Verzija 6.2 iz 1991. godine je bio prvi višekorisnički LAN SQL server baze podataka za razne operativne sisteme (OS/2, MacOS, Xenix, Banyan i Vines). Značajno poboljšanje je predstavljalo uvođenje PL/SQL procedurnog jezika. Verzija 7 uvodi uskladištene procedure baze podataka i mehanizam trigera, korisnički definisane funkcije itd. Oracle SUBP podržava rad i pristup podacima u

mreži, rad sa distibuiranim bazama podataka, sastoji se iz više programskih modula za različite namene. [BOBROWSKI, 1995], [RADULOVIĆ, 1997]

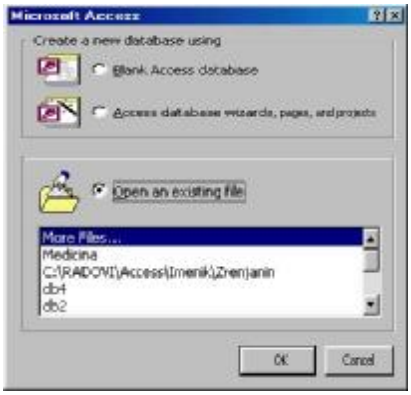
3.2.2. ACCESS

Microsoft Access je jedan od prvih sistema za rukovanje bazama podataka razvijen za operativni sistem Windows. Prva verzija Access 1.0 se pojavila na tržištu 1992. godine i bila je namenjena za kreiranje jednostavnih baza podataka i aplikacija. Ova verzija je imala ograničenje tako da baza podataka nije mogla da bude veća od 128MB. Uskoro je verzija Access 1.1 ovu granicu pomerila na 1GB. Sledeća verzija, Access 2.0, je bila namenjena za rad na operativnom sistemu Windows 3.1 i kasnijim verzijama. Uvedene su značajne novine u skoro svakoj oblasti, kao što su: velika proširenja nad objektima i modelima događaja, mogućnost pisanja procedura za događaje, kaskadno ažuriranje i brisanje podataka, novi tipovi i optimizacija upita, DAO model za manipulaciju sa podacima, OLE klijent podršku, mogućnost programiranja zaštitnih ograničenja itd. Polovinom '90-tih godina prošlog veka izlazi i verzija za operativni sistem Windows 95 pod nazivom Access 7.0. Početkom 1997. godine izlazi, verzija Access 97, zatim Access 2000, Access XP, a danas je aktuelna verzija Access 2003, koja koristi Access 2000 format baze podataka. Razvojem Access-a se u poslednjoj verziji ovog SUBP-a stiglo do stabilnog 32-bitnog sistema za upravljanje bazama podataka za izradu stonih i klijent/server aplikacija baza podataka, koje se izvršavaju pod Windows 9x, NT, 2000, XP i 2003 verzijama ovog operativnog sistema. Access poseduje podršku za tehnologije: ADO, DAO, OLEDB i ADP, kao i podršku za razvoj Internet stranica [JENNINGS, 2000], [RADULOVIĆ, 1997].

POKRETANJE MICROSOFT ACCESS-a

Po pokretanju *Microsoft Access*-a 2000 otvara se prozor (slika 3.32) kojim možete izabrati opcije:

- Kreiranje nove baze podataka (*Create a new database using*) - nudi opet 2 mogućnosti: kreiranje "prazne" baze – ručno i "automatsko" kreiranje baze uz pomoć posebnog programa – čarobnjaka (*Wizard*), koji su namenjeni početnicima i onima koji nemaju veće informatičko iskustvo.
- Otvaranje postojeće baze podataka (*Open an existing database file*) - izborom jedne od baza sa spiska otvataranih.



Slika 3.32. Pokretanje Access-a

Kreiranje nove baze će od korisnika tražiti da odredi naziv baze i putanju u kojoj će se ona nalaziti, a to je najčešće, zbog veličine samog fajla baze podataka, hard disk.

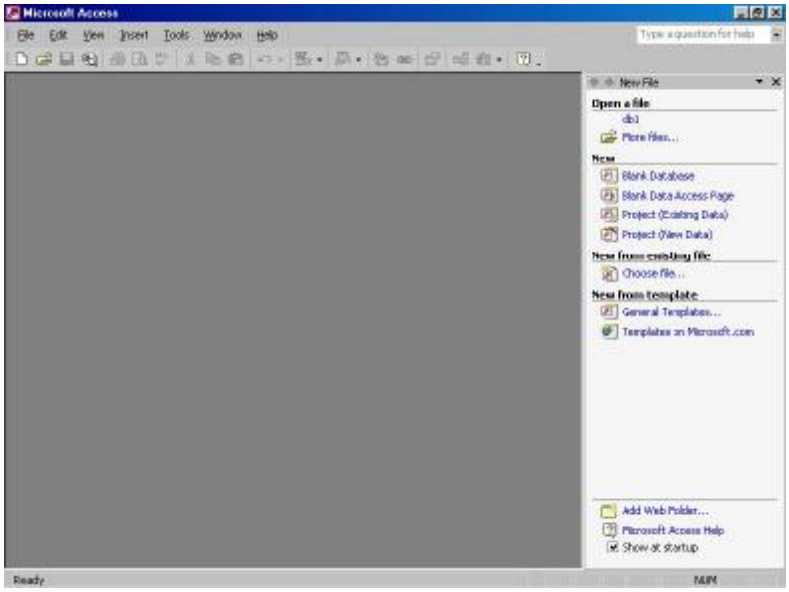
Nakon pokretanja *Microsoft Access*-a *XP* pojavljuje se prozor sa *Task Panel*-om (slika 3.33) sa desne strane sa opcijama:

- Kreiranje nove baze (*New*) - U okviru za kreiranje nove baze moguće je odabrati:
  - *Blank Database* - prazna baza podataka,
  - *Blank Data Access Page* – prazna dinamička stranica za Internet,
  - *Project (Existing Data)* – nov projekat nad postojećim podacima,
  - *Project (New Data)* – nov projekat nad novom bazom podataka.
- Otvaranje neke od postojećih baza podataka (*Open a file*)
- Kreiranje nove baze podataka iz postojećeg fajla (*New from existing file*)
- Kreiranje nove baze podataka pomoću šablona (*New from template*).

RADNO OKRUŽENJE

Izgled radnog grafičkog okruženja *Microsoft Access-a* prikazan je na sledećoj slici 3.33. Izdvajaju sledeći elementi:

- Naslovna linija (*Title bar*) - sadrži znak, naziv programa, dugme za minimiziranje ekrana programa na radnu liniju (engl. *Task Bar*), dugme za maksimiziranje ekrana programa naa celu površinu ekrana, kao i dugme za izlazak iz programa.
- Linija menija (*Meni bar*) - sadrži imena globalnih mogućnosti programa. Svaki od ovih menija se aktivira klikom na naziv menija, ili korišćenjem tastature - pritiskom na taster *Alt* i podvučeno slovo u određenom meniju. U meni liniji se nalaze sledeći meniji: *File, Edit, View, Insert, Tools, Window* i *Help.*Aktiviranjem jednog od ovih menija pojavljuju se padajući meniji sa mogućnostima za manipulaciju bazom podataka.
- Paleta alatki (*Toolbar*) - Ikone koje ovde postoje su standardne za sve aplikacije pod *Windows*-om *XP* pa neće biti posebno objašnjavane.
- Radno područje - Radno područje predstavlja centralni prozor. U radnom području se otvara, edituje, menja i manipuliše sa objektima baze podataka.
- Statusna linija (*Status Line*) - Statusna linija se nalazi na dnu ekrana. U statusnoj liniji se pojavljuju informacje o trenutnom načinu rada i trenutnim aktivnostima. Obično su informacije vezane za preciznija objašnjenja vezana za određene tastere. Objašnjenja se pojavljuju u jednoj rečenici.
- Panel zadataka (*Task Panel*) - Panel zadataka je novina u verziji *XP*. U zavisnosti od aktivnog objekta, razlikuju se opcije ovog panela. Po pokretanju programa pojavljuje se sa opcijama:
  - *Open a file* – sa listom fajlova koji su poslednji korišćeni,
  - *New* – sa opcijama za pokretanje nove baze ili projekta,
  - *New from existing file (template)* – pokretanje baze na osnovu postojeće (ili šablona).



Slika 3.33. Task Panel prozor

OSNOVNI OBJEKTI

Bilo koji način otvaranja baze podataka startuje osnovni prozor za rad u *Access*-u, prozor *DATABASE WINDOW* (slika 3.34). Podeljen je na nekoliko osnovnih celina koje omogućavaju rad sa osnovnim komponentama (objektima) *Access*-a:

- ***Tables* (tabele)** – organizacione celine, u relacionim BP, u koje se fizički smeštaju podaci.
- ***Queries* (upiti)** – mehanizam koji omogućava manipulaciju sa podacima iz tabela, prikuplja podatke koji se nalaze u različitim i povezanim tabelama i prikazuje ih korisniku u određenom obliku koji je pogodniji i korisniji nego u tabelama.
- ***Forms* (forme)** – obrasci, maske, ekrani – prozori koji služe za unos (izmenu, brisanje, pregled) podataka i uopšte komunikacija sa korisnikom.
- ***Reports* (izveštaji)** – obrasci za prezentaciju podataka i informacija na ekranu fleksibilnije, preglednije i upotrebljivije nego što je to u tabelama ili maskama i štampanje tih informacija na papiru.
- ***Macros* (makroi)** – korisnički definisan skup akcija za manipulaciju objektima (najčeće pokretanje prozora).
- ***Pages* (stranice)** – dinamičke Internet stranice koje prikazuju i ažuriraju podatke iz tabela u nekom od programa za pretraživanje (Internet Explorer ili Netscape Navigator npr.).
- ***Modules* (moduli)** – globalne procedure, funkcije ili promenljive pisane u programskom jeziku *Visual Basic for Aplications* (*VBA*) i koje važe na nivou celog projekta u Access-u.

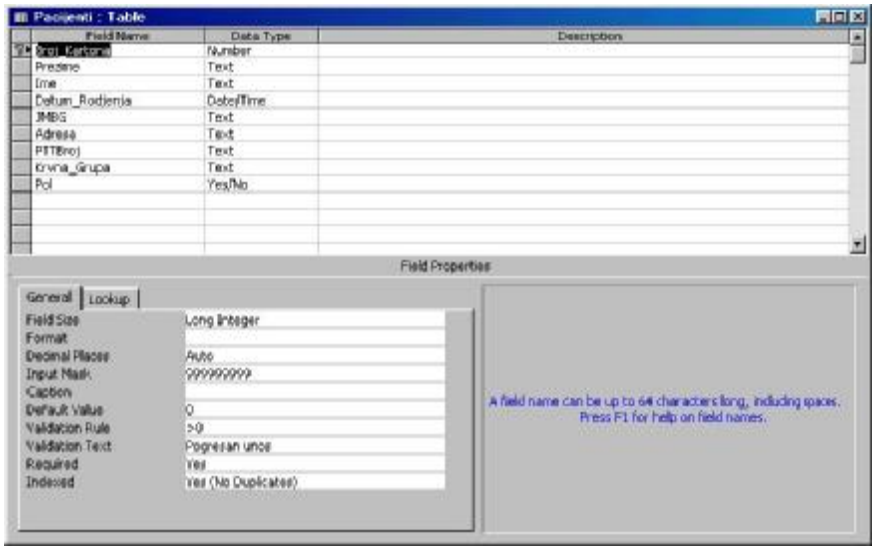
Slika 3.34. Database prozor

TABELE

Podaci su u bazama podataka smešteni u tabelama, koje se upotrebljavaju za predstavljanje koncepata, događaja i objekata iz realnog sistema (npr. RADNIK, KUPAC, PACIJENT itd.). Tabele se sastoje od redova (jedan slog je jedan red tabele sa konkretnim vrednostima podataka) i kolona (relevantnih osobina, atributi, obeležja realnih objekata). Jedna baza podataka koja opisuje realan sistem sadrži nekoliko tabela koje su međusobno povezne relacijama (vezama). Ovakva organizacija podataka čini Coddov relacioni model podataka. *Access* omogućava nekoliko načina kreiranja tabela:

- ***Design View*** – ova opcija se najčešće koristi (gotovo uvek) prilikom kreiranja tabela. Prozor koji se otvara izborom ove opcije je podeljen na dva dela:
  - Prvi, u kojem se sa tastature unose: naziv kolone (*Field Name*), tip podatka (*Data Type*) i opis te kolone (*Description*).
  - Drugi (u donjem delu prozora – *Field Properties*), služi za definisanje osobina pojedinačnih kolona:
    - *Field Size* – veličina polja,
    - *Format* – format u kojem se čuvaju podaci,
    - *Input Mask* – maska prikaza podataka,
    - *Caption* – naziv polja prilikom prikazivanja na formi,
    - *Default Value* – inicijalna vrednost polja,
    - *Validation Rule* – izraz koji definiše proveru tipa podatka,
    - *Validation Text* – tekst koji se pojavljuje ukoliko nije zadovoljen izraz iz polja *Validation Rule*,
    - *Required* – obavezan podatak. Ako je setovan na *Yes* onda se u izabrano polje (kolonu tabele) uvek mora uneti konkretni podatak,
    - *Allow Zero Length* – dozvoli dužinu nula. Ukoliko je setovan na *Yes* onda se u izabrano polje (kolonu tabele) ne mora uneti konkretni podatak – dužina nula,

- *Indexed* – definisanje da je izabrano polje indeks (ključ) tabele. Indeksi su polja kojima se ubrzava pretraživanje i sortiranje podataka. Indeks se može sastojati od jednog ali i od nekoliko polja tabele. *Primary* – je primarni indeks, tj. primarni ključ – jedinstveni identifikator svakog reda u tabeli.
- *Decimal Places* – mesto u polju gde se nalazi oznaka za decimale.



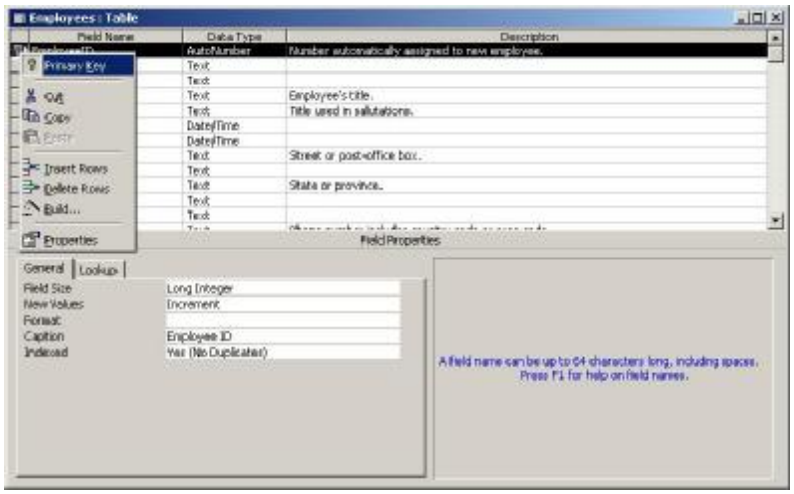
Slika 3.35. Kreiranje tabele

- ***Table Wizard*** – opcija koja početnicima preko programa čarobnjaka nudi biblioteku raznih tabela, kolona i podataka. Metodologija rada sa čarobnjacima je uglavnom ujednačena i odvija se preko nekoliko dijalog prozora (*Window*) u kojima se donosi odluka o uključenju pojedinih komponenti baze, ponuđenih objekata ili delova aplikacije u projekat. Sledeće opcije se javljaju u većini "čarobnjaka":
  - *Next* (sledeći korak) – ova opcija se bira po završetku rada, tj. izbora na aktuelnom dijalog prozoru i automatski se otvara naredni dijalog itd.
  - *Back* (prethodni korak) – opcija koja vraća korisnika na prethodni dijalog ukoliko želi da ispravi neki podatak ili izbor definisan na prethodnom prozoru itd.
  - *Finish* (kraj, završetak) – opcija sa kojom se završava rad sa čarobnjakom i on na osnovu izabranih podataka kreira željeni objekat. Ovu opciju je moguće koristiti i i pre poslednjeg dijalog prozora, ukoliko ne želite da menjate ni jedan od ponuđenih podataka na narednim prozorima itd.
  - *Cancel* (odustani) - opcija sa kojom se prekida rad sa čarobnjakom itd.
- ***Entering Data*** – opcija koja formira tabelu tako što se podaci unose u tabelu a tipove podataka određuje Access, dok je naziv kolone potrebno naknadno ispraviti (Field1, Field2, itd.) opcijom Design.

U procesu kreiranja tabele baze podataka, prilikom definisanja kolona (obeležja, atributa, polja) potrebno je znati sledeće:

- **Naziv kolone** (*Field Name*) – ne sme biti veći od 64 karaktera i ne smeju se koristiti sledeći karakteri: (.) - tačka, (!) – znak uzvika, ([,]) – uglaste zagrade, apostrof (') i karakter za prazno ( ) *SPACE*.
- **Tip podatka** (*Data Type*) – Access podržava sledeće tipove podataka:
  - *Text* – tekstualni tip max. dužine 255 karaktera,
  - *Memo* – tekstualni tip max. dužine 65535 karaktera,
  - *Number* – brojčani tip podatka (*Byte* 0-255, *Integer, Long Integer* – celobrojni, *Single, Double* – realni brojevi, *Decimal* – decimalni, *ReplicationID* – ceo broj koji se automatski uvećava kada korisnik unese novi red sa podacima u tabelu),
  - *Date/Time* – Datumski tip podatka u jednom od sledećih formata: DD.MM.GGG, MM.DD.GGGG, DD/MM/GGGG, MM/DD/GGGGG, DD-MM-GGGG, MM-DD-GGGG,
  - *Currency* – novčani tip podatka,
  - *Yes/No* – logički tip podatka (*Bool*-ov) koji može imati samo dve vrednosti 0 ili 1 tj. DA (Yes) ili NE (No).
  - *AutoNumber* – Celobrojni tip podatka koji se automatski uvećava.
- Kartica ***General*** služi za definisanje osobina pojedinačnih kolona (Field Size – dužina polja, Required – obavezno za unos)
- Kartica ***Lookup*** definiše tip kontrole u kojima će se prikazivati podaci: Label – nalepnica, polje za prikaz pod., *Text Box* – polje za unos pod., *List Box* – lista, *Combo Box* – kombinovano *Text* i *List*, *Check Box* – polje za potvrdu (Da/Ne).

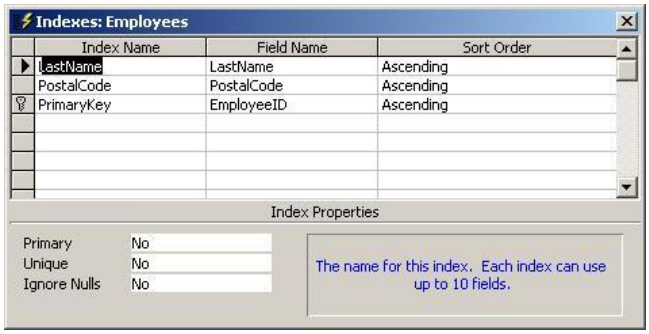
Za svaku tabelu mora postojati polje koje će jednoznačno određivati svaki slog. To polje se naziva **primarni ključ** - jedinstveni identifikator svakog reda (sloga) tabele. Formiranje primarnog ključa (slika 3.36) se vrši označavanjem imena polja desnim klikom miša i levim klikom na stavku ponuđenog menija - *Primary Key*.



Slika 3.36. Određivanje primarnog ključa tabele

Indeksi su polja kojima se ubrzava pretraživanje i sortiranje podataka, kao i očuvanje jedinstvenosti podataka na nivou tabele. Indeks se može sastojati od jednog ali i od nekoliko polja tabele. *Primary* – je primarni indeks, tj. primarni ključ tabele. Indeksi se mogu kreirati na dva načina:

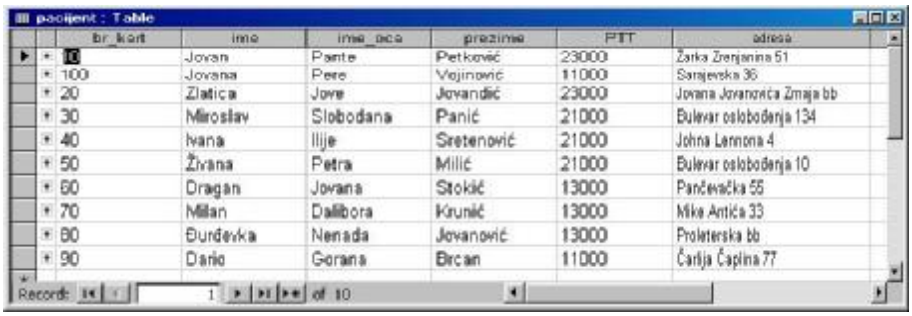
- U *General* sekciji prilikom kreiranja tabele,
- Korišćenjem opcije *View* – *Indexes* sa glavnog menija *Access*-a, otvara se dijalog prozor za definisanje indeksa nad trenutno otvorenom tabelom koja je u *Design* režimu rada.



Slika 3.37. Dijalog prozor za određivanje indeksa

Prva kolona ovog dijalog prozora se odnosi na naziv indeksa (*Index Name*), druga na ime polja iz tabele (*Field Name*) koje ulazi u sastav indeksa, a treća (*Sort Order*) na način sortiranja podataka u tom polju (koloni). Svaki indeks može sadržati maksimalno 10 polja.
Donji deo dojalog prozora se odnosi na osobine svakog pojedinačnog indeksa:

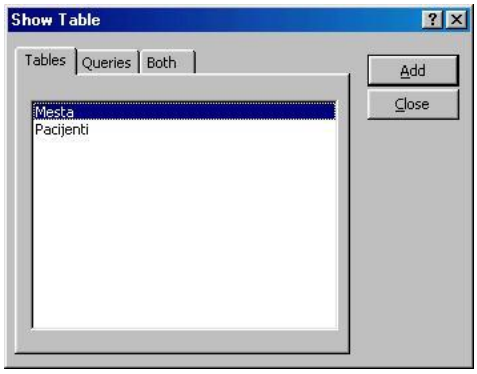
- *Primary* – ovu osobinu može imati samo jedan indeks koji se zove primarni ključ.
- *Unique* – obezbeđuje jedinstvenost slogova koji sadrže podatke iz datog indeksa. Ovo praktično znači da ne mogu postojati dva sloga sa istim vrednostima polja iz datog indeksa.
- *Ignore Nulls* – ukoliko indeks nije primarni ključ, dozvoljeno je da polja u indeksu mogu imati "Null" vrednosti, tj. da nije ništa uneto u to polje.



Slika 3.38. Tabela sa podacima

RELACIJE IZMEĐU TABELA

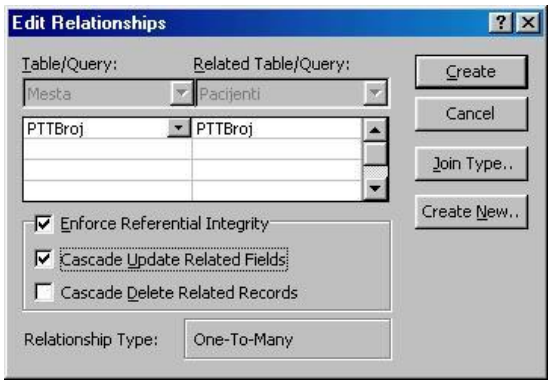
Access je relacioni sistem za rukovanje bazama podataka, tako da se u jednom trenutku mogu uzeti i koristiti podaci iz više tabela. Ovo je moguće povezivanjem tabela relacijama. Kada se kreiraju tabele i za svaku postavi primarni ključ (*Primary Key* - polje koje jedinstveno identifikuje svaki red tabele, tj. svaki konkretan podataka iz nekog realnog sistema), pristupa se spajanju, tj. povezivanju tabela. Opcija koju treba izabrati jeste: *Tools -> Relationships.*



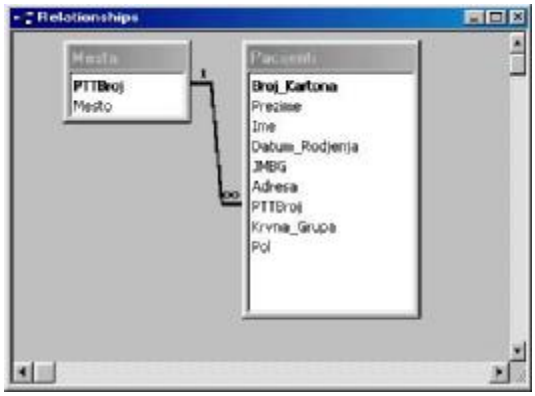
Slika 3.39. Dodavanje tabela u editor relacija

Otvara se prozor kao na slici gore, koji traži od nas da izaberemo tabele iz baze podataka koje želimo povezati. Prvo je potrebno kliknuti na naziv tabele, a zatim na taster *Add* koji ubacuje jednu po jednu tabelu u prozor *Relationships* koji je prikazan na slici ispod. Kada smo izvršili prebacivanje svih izabranih tabela, zatvaramo prerhodni prozor opcijom *Close.*

Da bi se bilo koja dva polja povezala relacijom, ne moraju imati isto ime, ali moraju sadržati podatke istog tipa! Samo stvaranje veza između tabela je prilično jednostavno. Prevuče se, mišem, polje iz jedne tabeli u odgovarajuće polje u drugoj tabeli. Smer prevlačenja polja ne određuje tip veze (jedan prema više (1-M), više prema jedan (M-1), jedan prema jedan (1-1), više prema više (M-M). Tip veze je uslovljen načinom na koji su određeni indeksi u tabelama koje se povezuju. Kada se otpusti levi taster miša, pojavljuje se sledeći ekran, koji prikazuje polja po kojima smo izvršili povezivanje i omogućuje dodatno podešavanje osobina relacije. Izbormo opcije *Create* prozor se zatvara i tada Access grafički prikazuje relaciju (slika 3.40).



Slika 3.40. Određivanje osobina relacija



Slika 3.41. Uspostavljena relacija između tabela

U prozoru za podešavanje osobina relacija se u gornjem levom delu nalaze nazivi tabela koje su povezane, a ispod toga nazivi polja pomoću kojih je uspostavljena relacija. U donjem delu ovog dijalog prozora se nalaze opcije za podešavanje pravila za očuvanje referencijalnog integriteta kao. U samom dnu prozora se nalazi opis vrste relacije. U desnom delu prozora se nalaze tasteri:

- *Create* – Završetak postupka kreiranje i podešavanja relacije,
- *Cancel* – Odustajanje od kreiranje i podešavanja relacije,
- *Join Type* – Vrsta spoja podataka u tabelama,
- *Create New* – Kreiranje nove relacije između bilo koje dve tabele u bazi podataka.

Join Type jeste podešavanje osobine relacije koji se odnosi na broj slogova koji je rezultat prikaza podataka nekim upitom u kojem se koriste ove dve tabele. Postoje tri mogućnosti:

- Uključivanje samo onih slogova iz obe tabele koje u polju pomoću kog se povezuju, imaju unetu istu vrednost u obe tabele.
- Uključivanje svih slogova sa strane jedan, relacije 1 prema više i samo onih slogova iz tabele sa strane više, koje u polju pomoću kog se povezuju tabele, imaju unetu vrednost koja odgovara nekoj iz tabele sa strane jedan.
- Uključivanje svih slogova sa strane više, relacije 1 prema više i samo onih slogova iz tabele sa strane jedan, koje u polju pomoću kog se povezuju tabele, imaju unetu vrednost koja odgovara nekoj iz tabele sa strane više.

Podeševanje referencijalnog integriteta se vrši uključivanjem opcije *Enforce Referential Integrity*. Time se omogućuje uključivanje preostalih opcija koje se odnose na očuvanje referencijalnog integriteta:

- *Cascade Update Related Fields* - kaskadna izmena vrednosti povezanih polja,
- *Cascade Delete Related Fields* - kaskadno brisanje povezanih slogova.

Referencijalni integritet jeste skup pravila koje se tiče odnosa između spoljnog ključa u jednoj tabeli i primarnog ključa u nekoj drugoj, a koji uzima vrednosti iz istog domena kao i posmatrani spoljni ključ. Ovo pravilo integriteta nalaže da se svaka vrednost spoljnog ključa u tabeli mora pojaviti i kao vrednost primarnog ključa u tabeli gde je odredeni atribut

primarni ključ. Ako, na primer, postoje tabele RADNIK (SIFRAD, IME, SIFODEL) i ODELJENJE (SIFODEL, NAZIV), onda u tabeli radnik svaka šifra odeljenja (SIFODEL) mora imati pojavljivanje u tabeli ODELJENJE, u atributu SIFODEL. Drugim rečima, radnik ne može raditi u nepostojećem odeljenju. Ukoliko je za radika uneta šifra odeljenja 1234, onda u relaciji ODELJENJE mora postojati odeljenje sa istom šifrom. To znači, da se pri unosu novih podataka mora voditi računa o vrednosti spoljnog ključa, odnosno, mora se paziti da se ne ubaci radnik koji radi u nepostojećem odeljenju. Ali, to nije dovoljno, pri brisanju podataka iz relacije ODELJENJE, mora se voditi računa o tome da se ne obriše odeljenje u kome rade neki radnici, jer će isti ostati vezani za nepostojeće odeljenje (jer je obrisano).

a) Ako ne želimo da postoji očuvanje jedinstva podataka ne treba izabrati opciju *ENFORCE REFERENTIAL INTEGRITY*. Tada se može desiti da ostanu viseći podaci u tabeli RADNIK (strana M - više). Tim slogovima se više ne može pristupiti, jer imaju staru vrednost prim. ključa. NIKAKO NIJE DOBAR IZBOR, jer dovodi do potpunog informativnog haosa u bazi podataka.
b) *CASCADE UPDATE RELATED FIELDS* nije izabrano (vrednost *False*=*Restrict*). Restriktivno očuvanje referencijalnog ontegriteta. Ukoliko se u tabeli na strani 1 (ODLEJENJE) promeni vrednost primarnog ključa, a postoje na strana više (RADNIK) vezani slogovi, *Access* ne dozvoljava promenu te vrednosti primarnog ključa. Samo u slučaju da ne postoji upotrebljen prim. ključ na strani više, moguće će biti promeniti vrednost primarnog ključa.
c) *CASCADE UPDATE RELATED FIELDS* je izabrano (vrednost *True*=*Cascade*). Kaskadno očuvanje referencijalnog ontegriteta. Ukoliko se u tabeli na strani 1 (ODELJENJE) promeni vrednost primarnog ključa, a postoje na strana više (RADNIK) vezani slogovi, *Access* automatski vrši promenu te vrednosti na strani više.
d) *CASCADE DLETE RELATED RECORDS* nije izabrano (vrednost *False*=*Restrict*). Ukoliko se u tabeli na strani 1 (ODELJENJE) obriše podatak, a postoje na strana više vezani slogovi, *Access* ne dozvoljava ovo brisanje jer je primarni ključ upotrebljen. Samo u slučaju da ne postoji upotrebljen prim. ključ na strani više, moguće je brisati podatak.
e) *CASCADE DLETE RELATED RECORDS* je izabrano (vrednost *True*=*Restrict*). Ukoliko se u tabeli na strani 1 briše vrednost primarnog ključa, a postoje na strana više vezani slogovi, *Access* automatski vrši brisanje svih slogova sa tom vrednošću i mnogo podataka na strani više može biti izgubljeno!?

FORME

Forme su prozori, tj. maske, ekrani koji predstavljaju osnovni objekat u *Access*-u za komunikaciju sa korsnikom. Imaju istu namenu kao i papirni obrasci za unos, izmenu ili brisanje podataka. Forme, takođe, omogućavaju prikaz, pretragu, pregled i štamopanje podataka. Osnovna karakteristika formi jeste ta da ona mora biti pregledna i jednostavna za korišćenje.

Prilikom kreiranja nove forme redosled radnji je isti kao i kod kreiranja tabele. Prvo je potrebno da u *Database Window* (prozoru) izaberete karticu *Form*. Ponuđena su dva osnovna načina kreiranja forme:

- *Design View* – kreira se blanko forma i za nju nije vezana nijedna tabela ili upit, a nije kreirana nijedna kontrola (vizuelni objekti za rad sa *Windows* radnim okruženjem, npr. prozor (*Window, Form*), taster (*Command Button*), oznake (*Label*), Polja za unos (*Text, Edit Box*) itd.). Ova opcija se preporučuje iskusnijim korisnicima Access-a.
- *Form Wizard* – program čarobnjak koji postepeno kreira formu. Koraci čarobnjaka su sledeći:
    - Izbor polja iz tabele koji će se prikazati na formi,
    - Izbor tipa forme: *Columnar* – polja rasporedjena u kolonama, *Tabular* – tabelarni prikaz polja, *DataSheet* – tabelarni prikaz polja, *Justified* – ravnomerno rasporedjena polja po formi sa max. iskorišćenjem prostora. Sledi izbor stila prikaza forme.
    - Posledniji korak je unos naziva forme koji je istovremeno i naslov (*Title*) forme i *Finish* – opcija kojom se završava kreiranje forme i ista se prikazuje korisniku na ekranu.

Slika 3.42. Forma za ažuriranje podataka

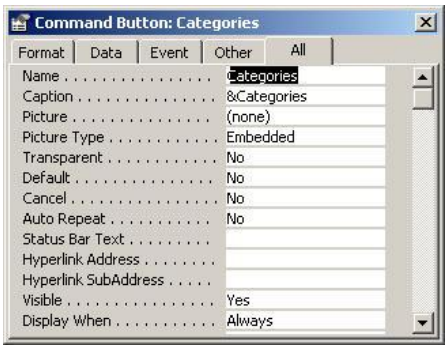
Kada je kreirana forma, na bilo koji od navedena dva načina, može se naknadno menjati, u smislu promene osobina forme ili postojećih kontrola na formi, kao i dodavanjem ili brisanjem kontrola sa forme. Da bi se vršile naknadne promene nad formom, potrebno je da se ista prethodni snimi. U *Database* prozoru se izabere opcija *Form*, pa nakon izbora naziva forme koja se želi menjati, izabere opcija *Design.* Tada se forma otvara u tzv. dizajn režimu rada i mogu se vršiti korekcije. Nakon izvršenih korekcija, forma se snima korišćenjem opcije *File - Save* sa glavnog menija *Access*-a, ikonice sa disketom ili prilikom napuštanja forme, nakon potvrde snimanja u odgovarajućem dijalog prozoru.

Kontrole se na formu nanose koristeći paletu (prozor) sa kontrolama. Ovaj prozor se uključuje preko stavke glavnog menija *Access*-a *View - Toolbox*. Na paleti sa kontrolama, na samom vrhu se nalaze dve ikonice – strelica, čijim uključivanjem se prelazi u režim podešavanja osobina nad postojećim objektima (mogućnost izbora objekta i podešavanja osobina) i druga ikonica čime se uključuje »čarobnjak za kontrole« odnosno mogućnost da kada se klikne na formu radi postavljanja neke kontrole, pokreće se program čarobnjak, čijim korišćenjem se podešavaju osobine date kontrole na lakši i brži način. Dakle, ovaj čarobnjak se aktivira samo u trenutku postavljanja kontrole na formu, a sva naknadna podešavasnja osobina se moraju izvršiti

»ručno«, kao što je objašnjeno u nastavku. Kontrola se na formu nanosi tako što se izabere sa palete sa kontrolama levim klikom miša, a zatim se klikne na formu na mestu gde se želi postaviti. Naravno, za svaku postavljenu kontrolu mogu se naknadno podesiti osobine.

Za svaku kontrolu, pa i samu formu, podešavanje osobina (ukoliko se radi »ručno«, a ne preko »čarobnjaka«) se vrši na sledeći način:

- Formu učitati u dizajn režimu rada,
- Objekat postaviti u fokus, levim klikom na njega (ukoliko je forma u pitanju, dovoljno je da se učita i da se na naslovnoj liniji forme klikne desnim tasterom da bi se otvorio odgovarajući meni),
- Desnim klikom na objekat otvara se meni, gde se bira opcija *Properties*,
- Izborom opcije *Properties* se otvara dijalog prozor gde se podešavaju osobine objekta,
- Snimanje promena, kao što je prethodno objašnjeno.



Slika 3.44. Podešavanje osobina kontrole

Svaki prozor *Properties* se sastoji od sledećih kartica, koje predstavljaju kategorizaciju osobina i mogućnost prikaza i promene njihovih vrednosti:

- *Format* – osobine koje se odnose na oblik, izgled objekta
- *Data* – osobine koje se odnose na vezu izvoru podataka (tabela,upit) koje prikazuje objekat
- *Event* – osobine koje se odnose na događaje koje se mogu dešavati nad datim objektom i odgovarajuće aktivnosti kao odgovor na njih,
- *Other* – ostale osobine
- *All* – sve osobine, objedinjene osobine sa svih ostalih kartica na jednom mestu, sortirano prema alfabetskom redosledu naziva osobina.

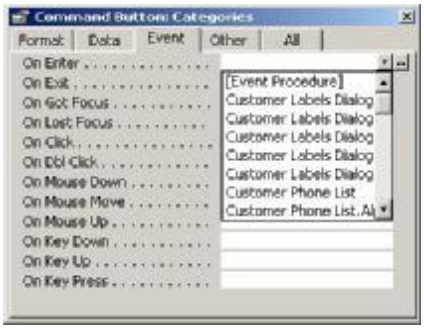
Bitno je napomenuti da se kod većine objekata nalazi skup zajedničkih i skup specifičnih osobina. Neke od najvažnijih zajedničkih osobina su: *Name* –jedinstveni naziv objekta, koji se najčešće koristi prilikom programiranja, *Caption* – natpis, tj. tekst koji piše na samom objektu i druge osobine. Bitno je razlikovati *Name* i *Caption*, iako mogu imati istu vrednost. Osobine podešavamo na dva načina:

1. Unosom vrednosti u odgovarajući prostor desno od naziva osobine
2. Izborom vrednosti koje se nude na odgovarajućoj padajućoj listi kraj naziva osobine.

Specifičnost u podešavanju osobina je vezana za događaje (*Event*). Naime, *Event* predstavlja mogući događaj, tj. dinamiku koja se može izvršiti nad objektom. Npr. nad *Command Button* kontrolom može da se izvrši događaj *Click*, tj. kada je pokazivač miša nad tom kontrolom i klikne se levim tasterom miša, dešava se događaj *Click* nad ovim

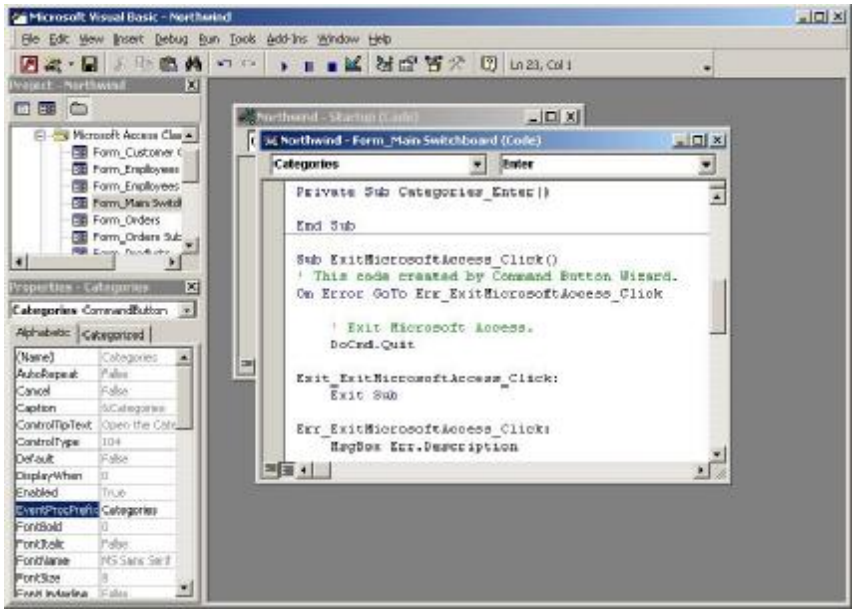
objektom. Ono što je moguće uraditi je podesiti za ovaj događaj kakav će biti odziv forme, tj. koja će se operacija izvršiti, a pokrenuta je nastupanjem ovog događaja. Dakle, potrebno je definisati aktivnosti koje će se izvršiti kada se desi događaj, npr. *Click* nad *Command Button*-om. Da bismo definisali akciju koja će se izvršiti po nastupanju nekog događaja, imamo sledeće mogućnosti:

- Pokretanje makroa, prethodno napisanog, ali ne mora da bude napisan samo za ovaj događaj, već se može pozivati i koristiti na više mesta,
- Pokretanje programskog koda (*Event Procedure*), posebno napisanog za ovaj događaj.



Slika 3.45. Određivanje akcije za izabrani događaj

Na slici 3.46. je prikazana lista sa mogućnostima izbora akcija koje treba da se izvrše kada se desi odgovarajući događaj. Prva stavka liste je uvek [*Event procedure*], čijim izborom, nakon čega treba da sledi klik na taster sa »...«, se omogućava otvaranje prostora za definisanje programskog koda za ovaj događaj (*Visual Basic Editor*).



Slika 3.46. Visual Basic Editor

Druga mogućnost je izbor makroa. Ispod stavke [*Event procedure*] uvek sledi lista svih makroa koji postoje sistemu (u mdb fajlu). Izborom naziva makroa izabrali smo da se taj makro izvrši prilikom nastupanja događaja nad tim objektom.

Ukoliko ne izaberemo nijednu od datih stavki, možemo samo kliknuti na taster sa »...« pored liste, čime nam se omogućuje korišćenje generatora koda, prema sledećem dijalog prozoru:

- Opcija *Expresion Builder* otvara dijalog prozor za kreiranje novog izraza koji će se izvršiti kada se desi dati događaj,
- Opcija *Macro Builder* otvara dijalog prozor za kreiranje novog makroa,
- Opcija *Code Builder* otvara *Visual Basic Editor* za pisanje programskog koda za dati događaj.

FORMA SA PODFORMOM

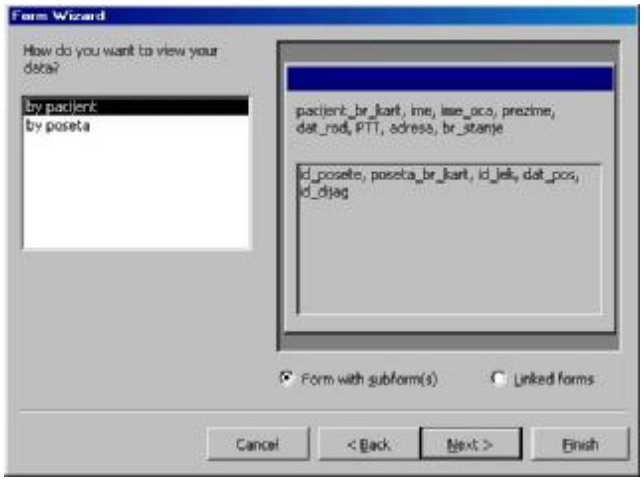
Forma sa podformom je namenjena unosu i pregledu podataka u (iz) dve ili više tabela koje su povezane relacijama, a imaju strukturu JEDNO ZAGLAVLJE – VIŠE STAVKI. U ovu grupu spadaju npr. dokumenti koji u sebi sadrže tabelarne podatke, odnosno podatke koji se po istoj strukturi unose više puta u okviru jedne celine (npr. karton pacijenta, otpremnice, prijemnice, računi, fakture, profakture, radni nalozi i sl.).

Slika 3.47. Forma sa podformom

Jednostavne forme za unos podataka, čije je kreiranje već spomenuto, ne omogućuju korisniku prirodan rad sa obrascima tipa JEDAN prema VIŠE. Zbog toga je neophodno kreirati složenu formu koja u sebi sadrži podformu (*Subform*). Postupak kreiranje složene forme u *Access*-u je najjednostavniji i najbrži "čarobnjakom". Koraci su sledeći:

- Izbor polja iz tabele koji će se prikazati u zaglavlju forme (strana JEDAN),
- Izbor polja iz tabele koji će se tabelarno prikazati u telu forme (strana VIŠE), ponovnim izborom, ovog puta druge tabele iz padajuće liste (isti dijalog prozor) i prebacivanje željenih kolona u desni prozor,

- Određivanje koja tabela ide u zaglavlje a koja u telo (podformu, mesto gde su prikazane »stavke«) forme,
- Izbor tipa podforme: ***Tabular*** – jednostavno tabelarno postavljanje polja za unos, ***DataSheet*** – posebna vrsta tabelarnog postavljanja polja za unos,
- Izbora stila prikaza forme, izbor pozadinske slike, boja i sl.
- Posledniji korak – unos naziva forme i podforme,
- ***Finish*** – opcija kojom se završava kreiranje formi i iste se prikazuju korisniku na ekranu.



Slika 3.48. Formiranje zaglavlja i tela forme sa podformom

Naravno, forma sa podformom se može kreirati i u dizajn režimu forme, bez korišćenja »čarobnjaka«. Postupak je sledeći:

1. Kreira se glavna forma, na jedan od prethodno opisanih načina za kreiranje jednostavnih formi (može i pomoću »čarobnjaka«). Ova forma treba da sadrži polja »zaglavlja«. Ova forma bi trebalo da je tipa »Columnar«.
2. Kreira se pomoćna forma, koja sadrži polja »stavki«. Ova forma bi trebalo da je tipa »***Tabular***«, da bi mogle da se prikažu više stavki.
3. Na glavnoj formi se izvrši modifikacija, tako što se proširi donja ivica forme da bi stala podforma. Na toj formi se doda sa palete sa kontrolama kontrola Subform i podese se za ovaj objekat osobine koje se odnose na to koju formu poziva ovaj objekat u ulozi subforme (to je dakle ta »tabelarna forma« koju smo kreirali u koraku 2.). Ovim je završen postupak kreiranja forme sa podformom.

UPITI

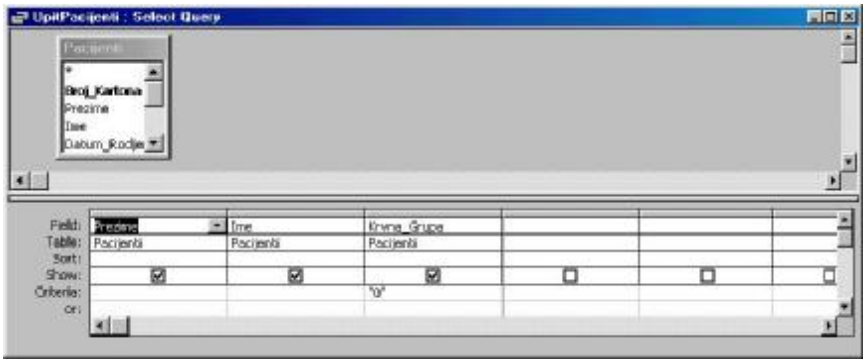
Prava snaga sitema za upravljanje bazama podataka jeste mogućnost da korisnici vide podatke u bazi onako kako oni žele u onom rasporedu koji im je u tom trenutku potreban. Način prikaza podataka u upitima može biti u vidu forme, tabele ili izveštaja. Upiti su neka

vrsta pitanja o podacima koji se nalaze rasporedjeni u tabelama. Npr. Koliko pacijenata imate u vašoj evidenciji? Ko je najstariji pacijent? i sl. Upiti su takodje mehanizmi za manipulaciju podacima.

Kreiranje novog upita je slično kao i kreiranje forme. Prvo je potrebno da u *Database Window* (prozoru) izaberete karticu *Queries*. Ponuđena su 2 osnovna načina kreiranja upita:

- *Design View* – otvara se prozor za ručno kreiranje upita. Ova opcija se preporučuje iskusnijim korisnicima *Access*-a.
- *Query Wizard* – program-čarobnjak koji automatski kreira upit. Koraci čarobnjaka su sledeći:
    - Izbor polja iz tabele koji će se prikazati u upitu,
    - Unos naziva upita koji je istovremeno i naslov (*Title*) upita,
    - *Finish* – opcija kojom se završava kreiranje upita.

Ako se želi izmena kreiranog upita može se pre opcije *Finish* izabrati opcija Modify the query design, i tada se otvara prozor koji omogućava kreiranje složenih upita sa parametrima. Ovom prozoru se može pristupiti i iz *Database Window* prozora tako što se prvo selektuje (mišem) kreirani upit i izabere opcija *Design:*



Slika 3.49. Kreiranje upita pomoću QBE

U ovom ekranu je moguće dodati kritetijume za izdvajanje podataka, menjati način sortiranja podataka, prikazivati i sklanjati kolone iz prikaza u upitu. Rezultat rada upita je prikazan tabelarno, slično kao što su i podaci raspoređeni u tabelama.

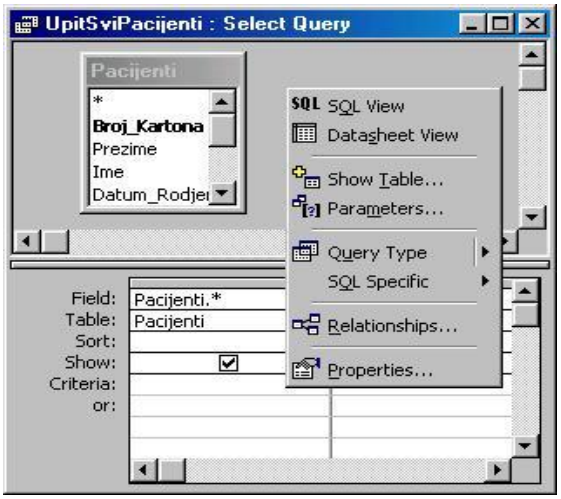
SQL

STRUCTURED QUERY LANGUAGE je jezik za postavljanje upita kojim možemo kreirati i menjati bazu podataka, tabele sa poljima i indeksima, a takođe i izdvajati podatke iz postojećih baza podataka, što je najčešća primena, posebno kod baza podataka koje sadrže veliku količinu podataka. SQL jezik je podržan od strane svih sistema za rukovanje bazama podataka, pa i *Access*-a.

Opšti oblik SQL naredbe jeste:

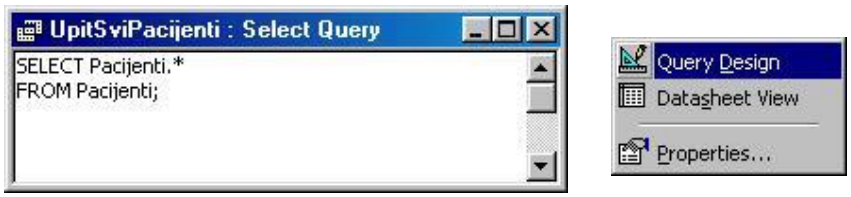
```
SELECT DISTINCT ime_kolone1, ime_kolone2...   (izdvajanje kolona)
FROM tabela1, tabela2...                      (iz tabela)
WHERE uslov_ili_kriterijum_pretrage           (za zadati uslov, kriterijum)
GROUP BY ime_kolone                           (grupisano po polju)
ORDER BY ime_kolone                           (sortirano po polju)
```

Access-ov *QBE* (*Query by Example*) editor za generisanje upita koji je prethodno opisan, automatski kreira SQL kod, kako korisnik koristi opcije editora. Upit koji prikazuje sve podatke o pacijentima može se kreirati »čarobnjakom« i u *QBE* editoru ima izgled, kao na slici 3.50.



Slika 3.50. Prelaz iz QBE editora u SQL prikaz upita

Za upite nastale pomoću čarobnjaka (*Wizard*) ili korišćenjem QBE editora (*Design View*) može se pogledati SQL kod koji je automatski generisan. Potrebno je da otvorimo postojeći upit u dizajn režimu rada (izborom: *Queries*-naziv upita-Design opcija), kada se otvara *QBE* editor. Na naslovnoj liniji prozora *QBE* editora potrebno je kliknuti desnim tasterom miša, pri čemu se otvara iskačući meni, gde je opcija *SQL View* na vrhu. Izborom ove opcije se jednostavno može preći u »SQL pogled upita« koji je za dati primer sledeći:

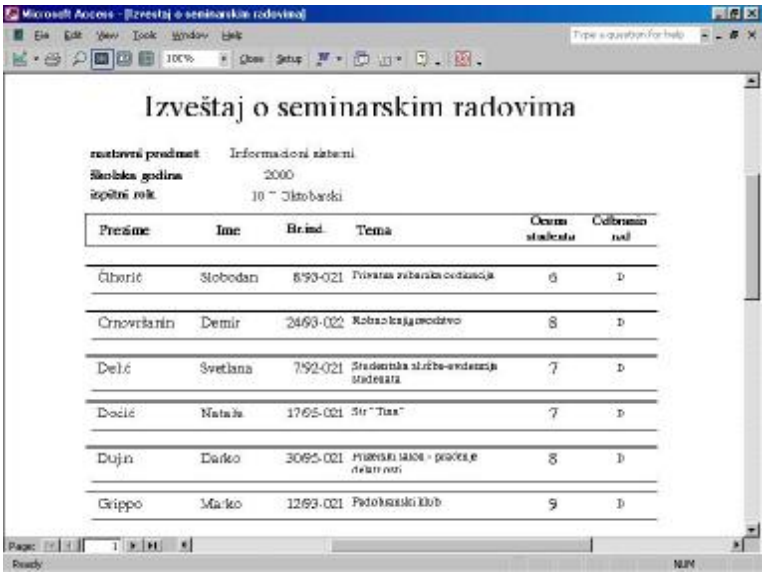


Slika 3.51. SQL prikaz upita

U ovom prozoru se potpuno ravnopravno kao i u *QBE* editoru mogu pisati ili menjati postojeći upiti zadavanjem SQL naredbe i njenih prametara. Povratak u *QBE* editor za upite se vrši, takođe, preko iskačućeg menija, opcijom *Query Design*. Pokretanje upita (izvršavanje) i u jednom i u drugom slučaju jeste opcija *Datasheet View*.

IZVEŠTAJI

Izveštaji omogućuju prikaz podataka u kao štampanih dokumenata, obrazaca, statističkih proračuna, zbirnih podataka, računa, obračuna itd. Izveštaji su slično upitima izvučeni podaci iz tabela i organizovani i prikazani na način oji je potreban korisniku programa. Kreiranje novog izveštaja je slično kao i kreiranje upita ili forme. Prvo je potrebno da u *Database Window* (prozoru) izaberete karticu *Reports*.



Slika 3.52. Primer izveštaja

Ponuđena su dva osnovna načina kreiranja izveštaja:

1. *Design View* – kreira se blanko izveštaj za koji nije vezana nijedna tabela ili upit, a nije kreirana nijedna kontrola za prikaz podataka na papiru. Ova opcija se preporučuje iskusnijim korisnicima *Access*-a.
2. *Report Wizard* – program čarobnjak koji postepeno kreira izveštaj. Koraci čarobnjaka su sledeći:
   - *Izbor polja* iz tabele koji će se prikazati na izveštaju,
   - Grupisanje podataka u izveštaju po izabranom polju (koloni).
   - Izbor polja po kojem će se sortirati podaci u izveštaju. Ako je vrednost polja ista, moguće je postaviti još 3 dodatna kriterijuma sortiranja podataka.

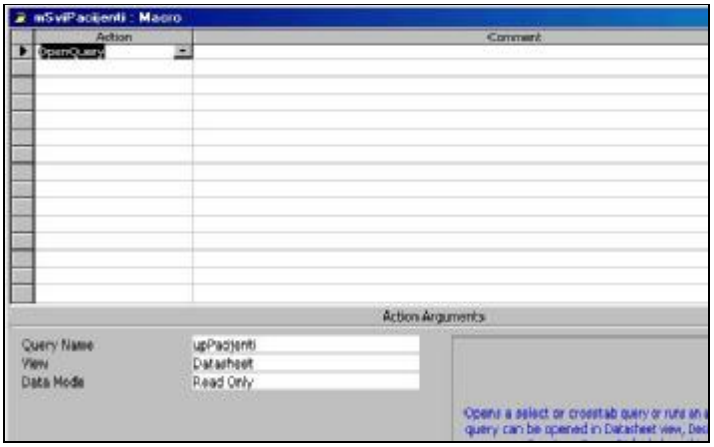
- Izbor tipa izveštaja: *Columnar* – polja rasporedjena u kolonama, *Tabular* – tabelarni prikaz polja, *Justified* – ravnomerno rasporedjena polja u izveštaju sa max. iskorišćenjem prostora.
- Izbora stila prikaza izveštaja,
- Posledniji korak – unos naziva izveštaja koji je istovremeno i naslov (*Title*) izveštaja,
- *Finish* – opcija kojom se završava kreiranje izveštaja.

MAKROI

Reč makro uglavnom u računarstvu znači objedinjavanje više akcija ili naredbi. Koriste se najviše za aktiviranje neke operacije ili događaja nad objektom. Pogodni su kao elementi brzog pravljenja prototipa aplikacija, pre svega kao neka vrsta zamene za «pravo programiranje», tj. pisanje koda u nekom programskom jeziku (u *Access*-u je to VBA – *Visual Basic for Applications*).

Makroi se kreiraju tako što se u *Database Window* prozoru izabere kartica *Macro*, pritisne dugme *New* i otvara radni prozor za kreiranje makroa (slika 3.53). Gornja polovina prozora je namenjena za izbor akcije(a), a donja polovina za definisanje argumenata akcija. Tabela za odabir akcije ima dve kolone:

- padajuću listu *Action* za izbor akcije,
- Vaš komentar, opis uz makro – *Comment.*



Slika 3.53. Formiranje makroa

Neke od najčešće korišćenih akcija u *Access-u*:

1. *Close* – zatvara prozor,
2. *OpenForm* – otvara formu,
3. *OpenQuery* – otvara upit,
4. *OpenReport* – otvara izveštaj,

5. *OpenTable* – otvara tabelu,
6. *Quit* – izlaz iz Access-a i snimanje svih promena,
7. *RunMacro* – pokreće drugi makro,
8. *RunSQL* – izvršava SQL upit,
9. *StopMacro* – zaustavlja izvršavanje makroa itd.

Podešavanje parametara akcije se najčešće svodi na izbor objekta na koji se odnosi akcija (npr. izbor izveštaja koja se otvara akcijom *OpenReport* i sl.) kao i načina na koji treba da se data akcija izvrši (npr. da li da se izveštaj prikaže na ekranu ili na štampaču i sl.). Na kraju formiranja makroa, klikne se na taster *Close* (X) i tada se pojavljuje prozor za upis naziva makroa, koji se zatim snima (*Save As*) u bazu podataka.

3.3. VISUAL BASIC.NET

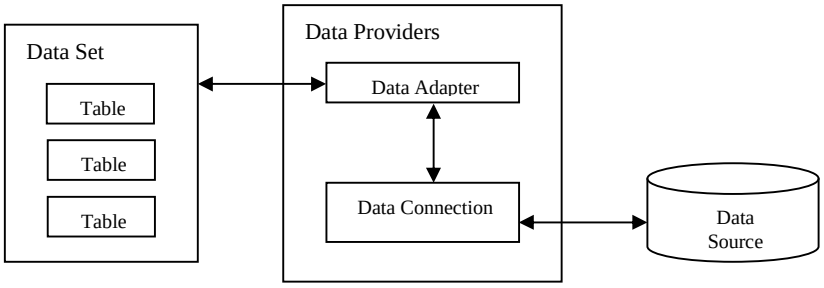
Microsoft Visual Basic.NET 2003 je razvojno okruženje za izradu različitih tipova aplikacija: konzolnih, Windows, Web, za mobilne uređaje, kao i različitih komponenti aplikacija – Web servisa, *ActiveX* komponenti i sl. Ove komponente se koriste za **implementaciju softverske podrške informacionih sistema**. *Visual Basic.NET* je sastavni deo *Microsftov*-og paketa za razvoj aplikacija *Visual Studio.NET*, koji sadrši još i razvojna okruženja: *Visual C++.NET* i *Visual C#.NET*.

Razvojno okruženje *Visual Basic.NET*, za rad sa bazama podataka, koristi *ADO.NET* tehnologiju, koja je uvedena u Microsoftova rešenja za razvoj aplikacija 2000. godine. ***ADO.NET* se bazira na XML standardu**. Jedna od najvažnijih karakteristika ovog standarda je ta da aplikacija koja čita ovaj standard može **upravljati podacima bez obzira na platformu** – bilo da je to *Windows*, *Unix*, *Linux* i dr. ili programski jezik – *C++*, *Visual Basic, Delphi, C#* i dr. [FORGEY i sar., 2002]

ADO.NET obezbeđuje **nekonektovani pristup podacima** u bazama podataka. Klijent-server aplikacije su tradicionalno morale da drže otvorene konekcije ka svojim bazama podataka ili da sopstvenim mehanizmima lokalno čuvaju podatke. [FORGEY i sar., 2002]

Glavne komponente ADO.NET-a su prema [FORGEY i sar., 2002] sledeće:

- ***DataProviders*** – komponente koje upravljaju podacima:
  - ***DataConnection*** – Otvara konekciju sa izvorom podataka, može biti *OleDataAdapter* ili *SQLDataAdapter*.
  - ***DataAdapter*** – Fizičko sredstvo komunikacije između izvora podataka i objekta *DataSet*. Svojstva *Command* objekta *DataAdaptera* su komande *SELECT, INSERT, UPDATE* i *DELETE*. Ovo su komandni objekti koji komuniciraju direktno sa izvorom podataka zbog manipulacije sa podacima.
- ***DataSet*** – memorijska prezentacija podataka potpuno nezavisna od od originalnog izvora podataka. Kada se napuni podacima, radi nezavisno od drugih objekata i ne treba mu konekcija do baze podataka.



Slika 3.54. Glavne komponente ADO.NET

DataSet je kolekcija tabela. Sastoji se od pet različitih tipova objekata:

- ***DataTable*** – Tabela u memoriji, ne može biti nula ili više tabela i sastoji se od više redova.
- ***DataRow*** – Red sa vrednostima aktuelnih podataka u *DataSet* objektu. Sastavni je deo kolekcije redova u objektu *DataRowCollection.*
- ***DataColumn*** – Objekat koji je deo kolekcije kolona a kreira se na osnovu šeme relacije tabele, tj. objekta *DataTable.*
- ***DataConstraint*** – Ograničenja za čuvanje integriteta podataka u *DataSet* objektu, određuje akciju koja će se preduzeti ukoliko podaci u zapisima budu ažurirani ili izbrisani.
- ***DataRelation*** – Relacija, veza jedne kolone tabele sa nekom drugom kolonom u drugoj tabeli podataka. Sve relacije u *DataSet* objektu čuvaju se u kolekciji *DataRelationCollection*.

Osnovni princip programiranja u *Microsoft Visul Basic.NET*-u je **programiranje vođeno događajima**. U ovakvom programiranju ne postoji jasan niz naredbi, procedura, fumkcija i potprograma koji su poređani od početka do kraja programa i koji se sekvencijalno ili po nekom logičkom redosledu izvršavaju, već se program gradi tako što se kreiraju grupe objekata koji imaju svoje osobine (svojstva) i događaje koji se mogu desiti nad tim objektima (procedure i funkcije). Korisnik je u interakciji sa ovim događajima objekata koji izvršavaju pojedine procedure ili funkcije tamo gde se iste pokrenu od strane korisnika. [HALVORSSON, 2002]

3.3.1. KREIRANJE APLIKACIJE

Novi projekat za izradu *Windows* aplikacije kao implementacije nekog dela informacionog sistema se vrši na sledeći način: Meni *File* → stavka *New Project* → izabrati *Project Type*: *Visual Basic Projects* → izabrati Templates: *Windows Application* → upisati u *Name* polje: IS Studentske službe.

Glavni alati koji su inicijalno vidljivi u razvojnom okruženju su [HALVORSSON, 2002]:

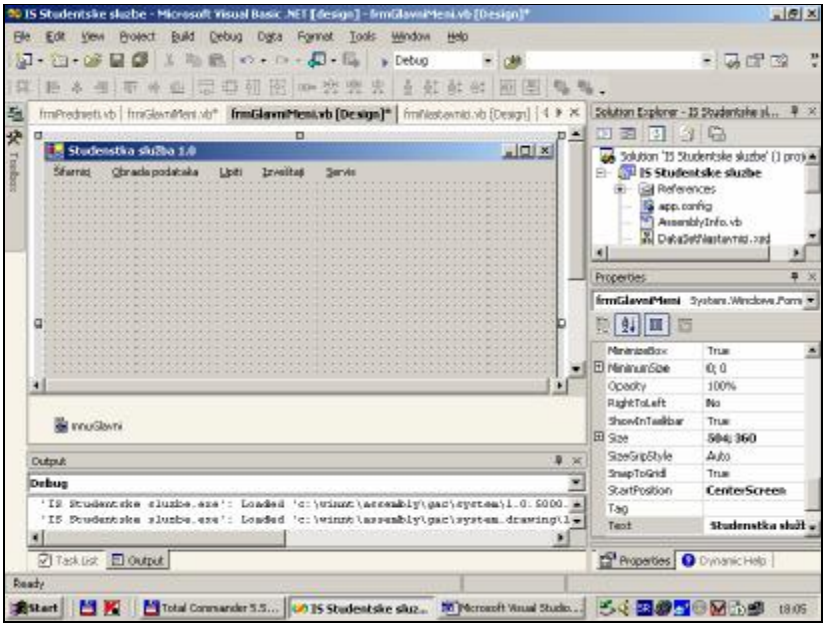
- *Solution Explorer* – kreiranje, pristup i izmena delova programa (npr. formi, modula itd.)

- *Properties* – podešavanje osobina, svojstava elemenata programa.
- Windows Form Designer – alat za rad sa objektom *Form* prilikom izrade aplikacije.
- *Toolbox* – paleta alata i kontrola koji se dodaju na nosioce objekata u programu.
- *Output* – poruke korisniku oizvršavanju programa, greškama i sl.
- *Dynamic Help* – sistem za pomoć programeru.

U okviru projekta koji je kreiran dodaju se *Form* objekti kao nosioci kontrola, komponenti i objekata za formiranje korisničkog interfejsa softvera i objekata za rad sa bazom podataka (*DataProviders*, *DataSet* i sl.).

3.3.2. KREIRANJE GLAVNOG MENIJA

Kreiranje glavnog menija aplikacije se sastoji iz dodavanja *Form* objekta sa padajućim menijem (kontrola *MainMenu*) u projekat aplikacije. Stavke menija treba da su: Šifarnici, Obrada podataka, Upiti, Izveštaji i Program. Forma treba da je fiksne veličine sa aktivnim tasterom za minimiziranje prozora (*Minimize Button*), neaktivnim tasterom za maksimiziranje prozora (*Minimize Button*) i aktivnim tasterom za izlaz iz aplikacije (*Close Button*). Izgled forme "Glavni meni" u režimu rada za izradu i kreiranje aplikacije (*Design*) je prikazan na sledećoj slici.



Slika 3.55. Kreiranje glavnog menija aplikacije

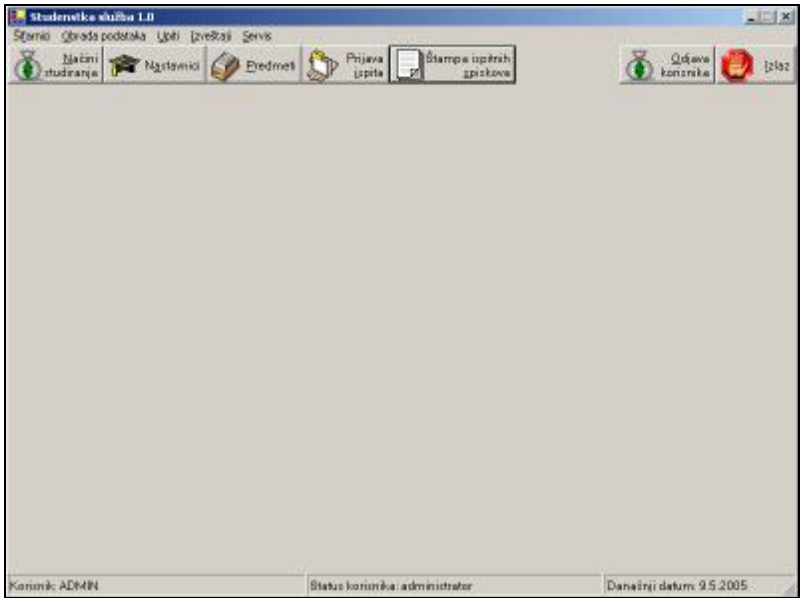
Podešavanja osobina objekta forma su sledeća:

- Objekat *Form1* → *Name* svojstvo → upisati *frmGlavniMeni*.
- *FormBorderStyle* → postaviti FixedSingle (forma fiksnih dimenzija, bez mogućnosti promene veličine forme u izvršnom režimu rada programa).
- *Text* svojstvo → upisati Studenstka služba 1.0 (naslovna linija prozora-forme).
- *MaximizeBox* → postaviti vrednost *False* (prozor glavnog menija se neće moći maksimizirati, već samo minimizirati).

U okviru određivanja osobina novog projekta potrebno je zadati početni objekat, a to će biti u našem primeru *frmGlavniMeni.* Izabrati: Meni *Project* → stavka menija *Properties* → *Startup Object* → izabrati *frmGlavniMeni.*

Kontrola *MainMenu1* → *Name* svojstvo → upisati *mnuGlavni.*
Polje *Type Here* → upisati redom: Šifarnici, Obrada podataka, Upiti, Izveštaji, Servis.

- stavka *MenuItem1* → *Name* svojstvo → upisati *mnuSifarnici*
- stavka *MenuItem2* → *Name* svojstvo → upisati *mnuObrada*
- stavka *MenuItem3* → *Name* svojstvo → upisati *mnuUpiti*
- stavka *MenuItem4* → *Name* svojstvo → upisati *mnuIzvestaji*
- stavka *MenuItem5* → *Name* svojstvo → upisati *mnuServis*



Slika 3.56. Glavni menija aplikacije

Sada je na formu potrebno dodati i nekoliko kontrola tipa *CommandButton* za aktiviranje funkcija programa koje će se najčešće koristiti. Statusna linija je objekat tipa *StatusBar*, koja će imati tri kartice sa podacima o prijavljenom korisniku i drugim važnim

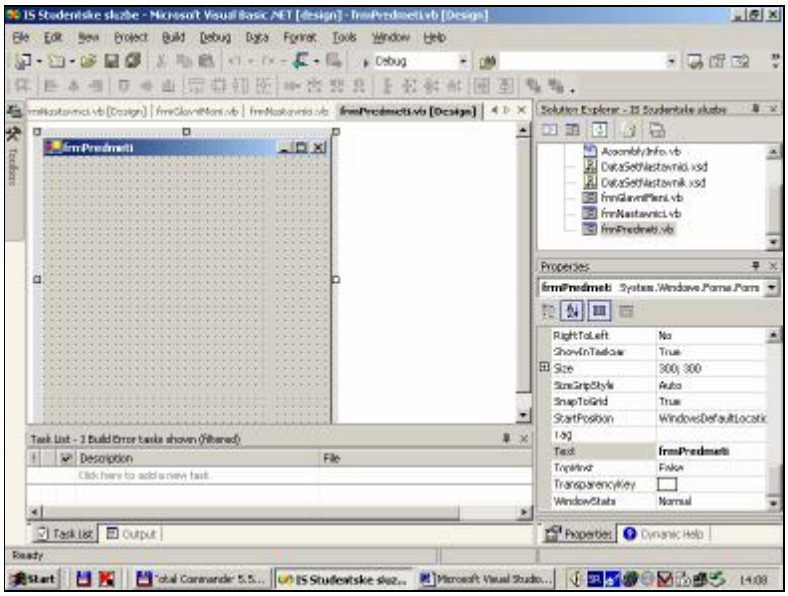
informacijama za pokrenuti program. Na slici 3.56. je prikazana forma glavnog menija u izvršnom režimu rada programa (*run-time*), koja sadrži sve osnovne elemente jedne *Windows* aplikacije.

Izlaz iz aplikacije korisnik ostvaruje izborom stavke menija Servis -> Izlaz ili klikom na taster Izlaz u glavnom meniju. Realizacija ove akcija se postiže pisanjem naredbe *End* na odgovarajućem događaju, a čije izvršavanje ima za posledicu završetak rada programa. Primer izvornog koda procedure *Click* događaja za taster Izlaz (*btnIzlaz*):

```
Private Sub btnIzlaz_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnIzlaz.Click
        End
End Sub
```

3.3.3. IZRADA ŠIFARNIČKE FORME

Šifarničke forme su produkt logičkog projektovanja baze podataka ili eksplicitnog zahteva korisnika za određenom vrstom podataka o nekim objektima ili sistemima. Jedan od šifarnika za primer IS Studentske službe jeste šifarnik predmeta koji se predaju na fakultetu. Da bi se kreirala forma koja će implementirati unos, ažuriranje i korišćenje podataka u okviru šifarničke tabele PREDMET, potrebno je sa glavnog menija *Visual Studio.NET* radnog okruženja, pokrenuti *Project* → *Add Windows Form*. U okviru dela *Templates* izabrati *Windows Form*. Uneti naziv forme u *Name* polju: *frmPredmeti.vb*, opcijom *Open* dobijamo novu formu (slika 3.57).

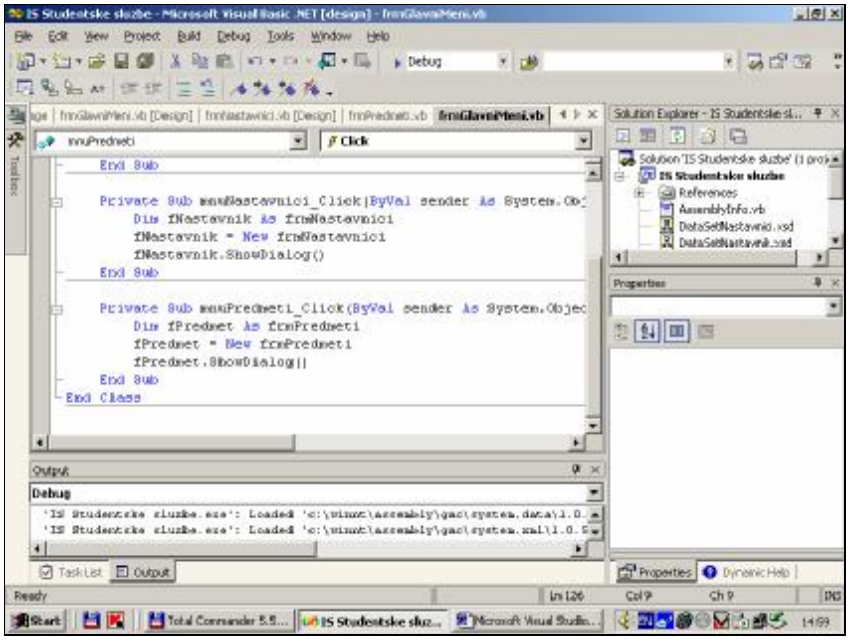


Slika 3.57. Kreiranje forme ŠIFARNIK PREDMETA

Povezivanje ove forme sa glavnim menijem se postiže tako što se u formi *frmGlavniMeni* doda nova stavka menija kucanjem u okviru objekta *mnuGlavni*. Inicijalni naziv stavke menija *MenuItem1* promeniti u *mnuPredmeti*. Otvariti prostor (2x klik levim tasterom miša) za unos naredbi za otvaranje forme (*Code Editor*). Upisati naredbe (slika 3.58) za deklarisanje objekta tipa klase *frmPredmeti*, instancirati objekat *fPredmeti* navedene klase forme i prikazati formu sa *ShowDialog* metodom:

```
Dim fPredmet As frmPredmeti
fPredmet = New frmPredmeti
fPredmet.ShowDialog()
```

Na sledećoj slici je prikazan otvoren editor koda za događaj klik u okviru stavke menija *mnuPredmeti*:



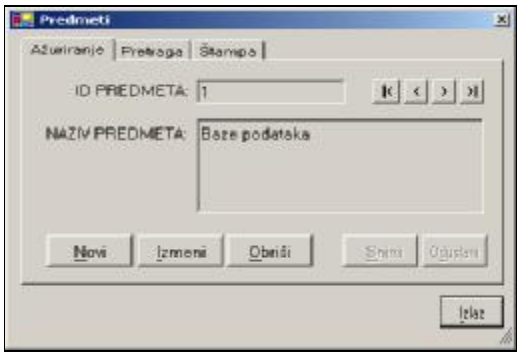
Slika 3.58. Pisanje naredbi u editoru koda

Pokretanje ovako kreirane jednostavne aplikacije se postiže nasledeće načine:

- glavni meni *Visual Studio.Net*-a, meni *Debug* → opcijom *Start,*
- pritiskom funkcijskog tastera F5,
- pritiskom tastera *Start* (>) sa palete najčešće korišćenih alata *Visual Studio.NET*-a.

Forma ŠIFARNICI → PREDMETI može sadržati tri kartice:

1. Za unos, ažuriranje, prikaz podataka:



Slika 3.59. Kartica za ažuriranje podataka na šifarničkoj formi

2. Za pretragu i pronalaženje željenog predmeta:

Slika 3.60. Kartica za pretragu podataka na šifarničkoj formi

3. Za štampu spiska šifarnika u vidu izveštaja:

Slika 3.61. Kartica za štampu spiska predmeta

Kartice na formi se dodaju pomoću komponente *TabControl*, čije svojstvo *Name* treba promeniti u *TabPredmeti*. Svojstvo *TabPages*, dodati sa tasterom *Add*, tri stavke:

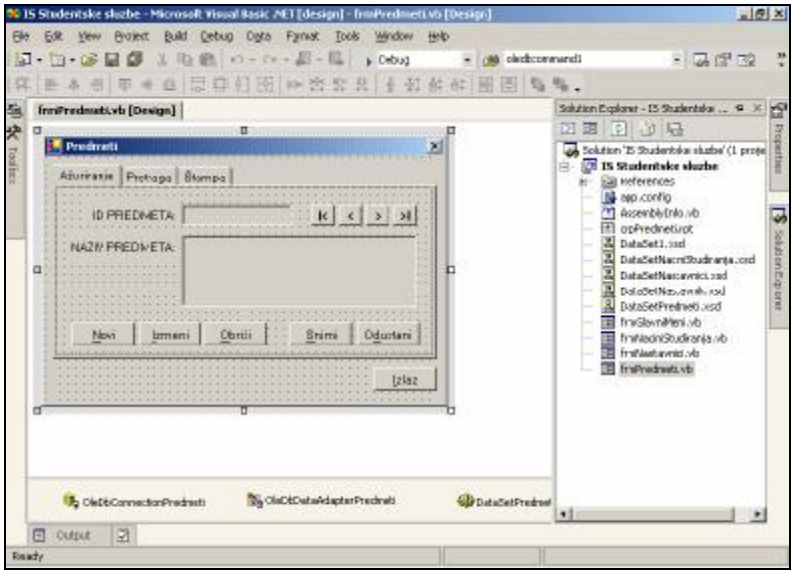
- *TabAzuriranje* – *Name* svojstvo, *Text* - upisati Ažuriranje;
- *TabPretraga* – *Name* svojstvo, *Text* - upisati Pretraga;
- *TabStampa* – *Name* svojstvo, *Text* - upisati Štampa.

Na prvu karticu, *TabAzuriranje*, dodati dve *TextBox* kontrole koje će služiti za unos, izmenu i prikaz podataka o predmetu. Postaviti:

- *txtIDPredmeta* – name svojstvo, umesto *TextBox1* i
- *txtNazivPredmeta* – name svojstvo, umesto *TextBox2*.

Promeniti *ReadOnly* osobine za oba *TextBox*-a u *True* (ili *Enabled=False*) – ova osobina *TextBox* kontrole onemogućuje korisnika da menja ili dodaje podatke o predmetima kada to nije dozvoljeno i sprečava ga da čini greške koje mogu dovesti do izgubljenih podataka zbog nepažnje.

Promeniti za *TextBox* - *txtNazivPredmeta* osobinu *MultiLine* u *True*, *MaxLength* u 80 (prikaz i unos teksta u više redova sa maksimalnom dužinom teksta od 80 karaktera).
Promeniti za *TextBox* - *txtIDPredmeta* osobinu *MaxLength* u 5.



Slika 3.62. Krieranje kartica na formi

Dodati sledeće komandne tastere - *Button* kontrole i upisati u *Name* svojstva:

- *btnIzlaz* – Kraj rada sa šifarnićkom formom i povratak u glavni meni.
- *btnNovi* – Dodavanje novih predmeta u bazu podataka.
- *btnIzmeni* – Izmena postojećih predmeta u bazi podataka.
- *btnObrisi* – Brisanje postojećih predmeta iz baze podataka.

- *btnSnimi* – Potvrda unosa novih, izmene i brisanja postojećih podataka. Postaviti svojstvo *Enabled=False* (neaktivan taster).
- *btnOdustani* – Odustajanje od unosa novih, izmene i brisanja postojećih podataka. Postaviti svojstvo *Enabled=False* (neaktivan taster).
- *btnPrethodni* – Prikaz prethodnog zapisa (sloga) o predmetima.
- *btnNaredni* – Prikaz sledećeg zapisa (sloga) o predmetima.
- *btnPrvi* – Prikaz prvog zapisa (sloga) o predmetima u bazi podataka.
- *btnPoslednji* – Prikaz poslednjeg zapisa (sloga) o predmetima u bazi podataka.

Dodati *DataGrid* kontrolu na drugu karticu – PRETRAGA. Odrediti sledeća svojstva *DataGrid* kontrole:

- naziv – *Name=DataGridPredmeti*;
- svojstvo *CaptionVisible=False* (Tabealarni prikaz bez posebne naslovne linije);
- svojstvo *ReadOnly=True* (Ne postoji mogućnost dodavanja novih podataka u tabelarni prikaz podataka o predmetima, pošto će se ova operacija nad podacima raditi na kartici AŽURIRANJE);

Na kartici ŠTAMPA se dodaju sledeće kontrole:

- *GroupBox* – svojstvo *Name=GroupBoxStampa* (okvir za logičku organizaciju kontrola i podataka na jednom delu forme, tj. ekrana);
- *RadioButton* – svojstvo *Name=radIzvestaj*,
- Dve kontrole tipa *Button*: *btnPregledPreStampe* i *btnStampa* (Izabrati za *Image* svojstvo sliku (.ico datoteku), *ImageAlign=MiddleLeft* – poravnanje slike levo, *TextAlign=MiddleRight* – poravnanje teksta desno na dugmetu).

Text svojstvo forme postaviti na Predmeti – ovaj tekst će biti ispisan u naslovnoj liniji prozora (*Title Bar*).

Stanje forme - *WindowState* treba da je *Normal*, *StartPosition* je *Center Screen* – forma će se uvek inicijalno prikazivati na sredini ekrana.

Dugme za maksimiranje prozora forme - *Maximize Box* postaviti na *False*.

Dodati na formu komandni taster (*Button1*) i promeniti mu naziv (*Name*) na *btnIzlaz*. Upisati u *Text* svojstvo tastera: Izlaz.

Otvoriti prostor za pisanje koda za događaje koji se mogu desiti nad tasterom Izlaz i upisati naredbu za izlaz sa forme:

```
Me.Dispose ()
```

kojim se objekat forma uništava u memoriji ili

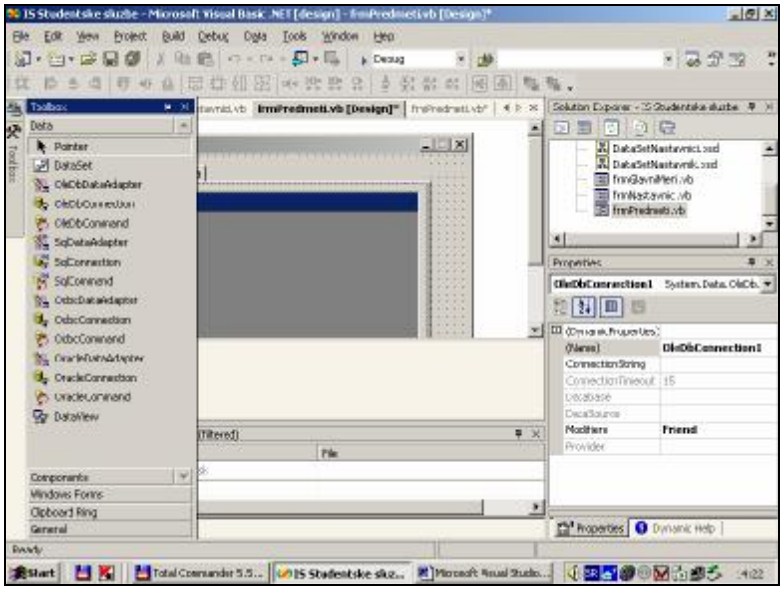
```
Me.Hide()
```

kojim se forma samo sklanja sa ekranskog prikaza, ali je i dalje učitana u memoriju.

3.3.4. KONEKCIJA SA BAZOM PODATAKA

Da bi se mogle realizovati funkcije prikaza, unosa i ažuriranja podataka, moraju se u daljem procesu razvoja forme kreirati objekti za rad sa izvorom podtaka - bazom podataka. Koraci u dodavanju objekata za konekciju sa bazom podataka su sledeći:

1. Postavljanje *OLEDBConnection* objekta za formu (jer je baza podataka kreirana u *Microsoft Access* sistemu za upravljanje bazama podataka). Sa palete alata – *ToolBox*, postaviti kontrolu *OLEDBConnection* radi ostvarivanja konekcije sa bazom podataka.



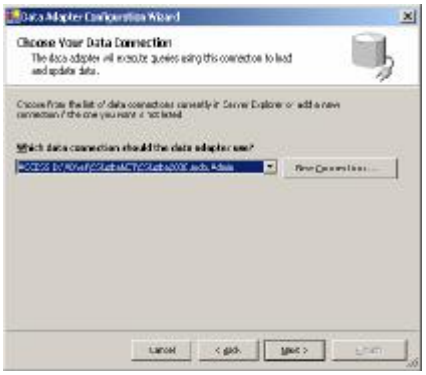
Slika 3.63. Krieranje kartica na formi

Nije neophodno promeniti naziv (*Name*) konekcije do baze podataka: *OLEDBConnection1*. Svojstvo ove konekcije *ConnectionString* postaviti tako što će se sa spiska izabrati *NewConnection* i zatim pristupiti izboru opcija pomoćnog ekrana. Potrebno je vratiti se se na karticu *Provider* i izabrati odgovarajući drajver za pristup bazi. Za *Access* je to *Microsoft Jet 4.0. OLE db Provider*. Opcijom *Next* se prelazi na karticu *Connection*. Izborom *Providera*, kartica *Connection* menja izgled. U okviru polja *Select or Enter Database Name* bira se putanja datoteke *Access* baze (mdb fajla). Nakon što se kompletna putanja i naziv fajla pojavi u polju, tasterom *Test Connection* se proverava da li je uspešno uspostavljena konekcija do baze podataka. Na ekranskoj kartici *Advanced* se podešavaju prava pristupa korisnika bazi podataka.

Nakon što je pritisnut taster *OK*, dobija se prozor za proveru da li se ipak želi postavljanje *Passworda* (šifre) na konekciju do baze podataka. Ukoliko se ne postavlja šifra, nakon izbora opcije *Don't include password*, vrši se povratak na formu.

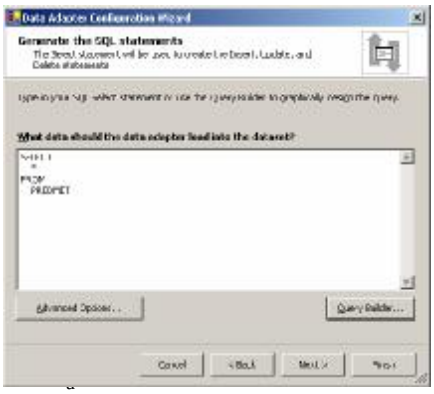
2. Postavljanje *OLEDBDataAdapter* objekta za formu - pojavljuje se *Wizard* sa sledećim koracima:

- Izbor postojeće konekcije do baze podataka (slika 3.64) ili kreiranje nove konekcije.
- Izbor načina pristupa podacima.
- Izbora podataka koje *Data Adapter* treba da poveže sa *Data Set* memorijskom kolekcijom podatka - opcija *Query Builder*.



Slika 3.64. Krieranje OLEDDBDataAdapter objekta

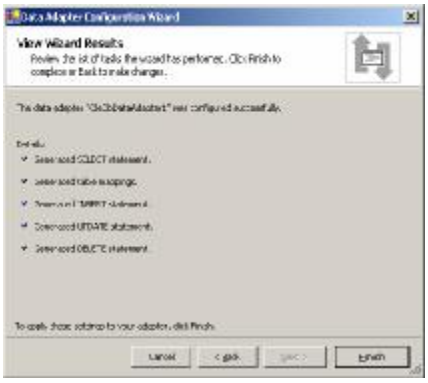
- Ručno pisanje SQL upita ili izbor jedne ili više tabela sa spiska, pri čemu se automatski formira SQL izraz sa mogućim izmenama skripta. Za primer šifarnika predmeta, izabrati tabelu PREDMET i dodati u donjem zapisu SQL izraza *. Sada je za naš primer formiran je SQL upit: *Select * From Premet* (slika 3.65).



3.65. SQL upit DataAdapter objekta

- Spisak SQL komandi koje će se kreirati za *Data Adapter*:
  - SELECT,
  - INSERT,
  - UPDATE,
  - DELETE,

  sa kreiranim mapiranjem tabelan (slika 3.66).
- Postupak kreiranja objekta *Data Adapter* se završava *Finish* opcijom.
- Na kraju se kreiranom *Data Se*t objektu menja ime, tj. svojstvo *Name* sa *OleDBDataAdapter1* na *OleDB DataAdapterPredmeti*.



Slika 3.66. Komande DataAdapter objekta

3. Dodavanje *DataSet* objekta za formu. *DataSet* objekat sadrži kolekciju podataka iz tabele ili upita. Nakon što je selektovana *OleDB DataAdapterPredmeti* kontrola, u okviru *Properies* prozora za adapter, pojavljuje se opcija *Generate Dataset*. Ovde postoje dve mogućnosti: izbor postojećih *DataSet*-ova ili definisanje novih (izbor). Naziv *DataSet*-a se unosi u okviru *New* stavke kao *DataSetPredmeti*.

4. Punjenje *DataSet*-a podacima preko *DataAdapter* provajdera izvršavanjem SELECT SQL komande. U okviru *Form Load* događaja za formu, piše se naredba:

```
OleDbDataAdapterPredmeti.Fill(DataSetPredmeti)
```

5. Povezivanje odgovarajućih kontrola sa izvorom podataka.

Promeniti *DataBindings* osobine ova dva *TextBox*-a u:

- *txtIDPredmeta* → Text svojstvo: DataSetPredmeti - PREDMET.IDPREDMETA,
- txtNazivPredmeta → Text svojstvo:
  DataSetPredmeti - PREDMET.NAZIVPREDMETA.

Za *DataGrid* na drugoj kartici Pretraga:

- svojstvo *DataSource*, izabrati: *DataSetPredmeti*;
- svojstvo *DataMember*, izabrati: *Predmet*.
- svojstvo *TableStyles*, dodati stil: *DataGridTableStylePredmeti* – name svojstvo stila tabele; *MappingName* izabrati: Predmet; *GridColumnStyles*: dodati dve kolone: *DataGridTextBoxColumnIDPredmeta* i *DataGridTextBoxColumnNazivPredmeta* (*Name* svojtva kolona tabelarnog prikaza podataka).
- Promeniti *HeaderText* u: ID PREDMETA i NAZIV PREDMETA (nazivi kolona u tabelarnom prikazu u okviru forme).
- Promeniti *MappingName* u: IDPREDMETA i NAZIVPREDMETA (nazivi kolona u bazi podataka).
- Kolona *Width* za: *DataGridTextBoxColumnIDPredmeta* je npr. 110 tačaka, a za *DataGridTextBoxColumnNazivPredmeta* je npr. 220 tačaka.

DODAVANJE NAREDBI ZA AŽURIRANJE PODATAKA

Naredbe za navigaciju među zapisima kolekcije podataka o predmetima (promena *Position* svojstva u *DataSet*-u):

```
Me.BindingContext(DataSetPredmeti, "PREDMET").Position += 1 'Naredni
Me.BindingContext(DataSetPredmeti, "PREDMET").Position -= 1 'Preth.
Me.BindingContext(DataSetPredmeti, "PREDMET").Position = 0 'Prvi
Me.BindingContext(DataSetPredmeti, "PREDMET").Position =
      DataSetPredmeti.Tables("PREDMET").Rows.Count - 1 'Poslednji
```

Naredba za početak dodavanja novog zapisa u *DataSetPredmeti* (metoda *AddNew*):

```
Me.BindingContext(DataSetPredmeti, "PREDMET").AddNew() 'Novi
```

Naredba za kraj dodavanja novog zapisa u *DataSetPredmeti* (metoda *EndCurrentEdit*):

```
Me.BindingContext(DataSetPredmeti,"PREDMET").EndCurrentEdit() 'Snimi
```

Blok naredbi za brisanje tekućeg zapisa iz *DataSetPredmeti* sadrđi i deklarisanje lokalne promenljive *slogPom* koja sadrđi vrednost pozicije tekućeg zapisa (sloga) *DataSet*-a kako bi se isti obrisao metodom *Delete*:

```
Dim slogPom As Integer
slogPom = Me.BindingContext(DataSetPredmeti, "PREDMET").Position
DataSetPredmeti.PREDMET.Rows(slogPom).Delete()
```

Odustajanje od unosa novog podatka i izmene tekućeg (metoda *CancelCurrentEdit*):

```
Me.BindingContext(DataSetPredmeti, "PREDMET").CancelCurrentEdit()
```

Elementi realizacije kartice za pretragu i tabelarni prikaz:

```
Dim brojReda, brojKaraktera As Integer
Dim podString As String

For brojReda = 0 To DataSetPredmeti.Tables("PREDMET").Rows.Count - 1
     DataGridPredmeti.UnSelect(brojReda)
Next brojReda

For brojReda = 0 To DataSetPredmeti.Tables("PREDMET").Rows.Count - 1
      brojKaraktera = Len(txtPronadji.Text)
      podString = Microsoft.VisualBasic.Left
      (DataGridPredmeti.Item(brojReda, 1), brojKaraktera)
       If UCase(txtPronadji.Text) = UCase(podString) Then
             DataGridPredmeti.Select(brojReda)
              DataGridPredmeti.CurrentRowIndex = brojReda
      End If
      Next brojReda
```

Pretraga podataka po zadatom kriterijumu - po nazivu predmeta i pronalaženje odgovarajućeg reda tabele u prikazu se postiže tako što se prvo vrši demarkiranje jednog ili više redova metodom *UnSelect DataGrid* objekta. Zatim se *For ... Next* ciklusom prolazi kroz sve redove tabelarnog prikaza (*DataGrid*-a) i vrši poređenje sadržaja *TextBox*-a i sadržaja ćelije tabele u koloni koja ima indeks (u ovom primeru 1, tj. druga kolona tabele) kolone u kojoj su smešteni podaci o nazivima predmeta. Ukoliko je ovaj sadržaj *TextBox*-a i odgovarajuće ćelije tabele isti, vrši se markiranje tog reda tabele postavljanjem svojstva *CurrentRowIndex DataGrid*-a na vrednost koju ima brojač ciklusa *brojReda*.

Treća kartica forme ŠIFARNICI → PREDMETI ima dve opcije, kao što se može videti na slici 3.61:

- Prikaz spiska svih predmeta u bazi podataka u tzv. PREGLEDU PRE ŠTAMPE (Print Preivew), kada korisnik ima vizuelnu predstavu celokupnog izveštaja i može ga ali i ne mora odštampati.

- Štampanje spiska svih predmeta iz baze podataka bez prvobitnog uvida u formu, izgled i sadržaj izveštaja (Print Report). Izbor ove opcije će rezultovati izvlačenjem izveštaja na tekućem štampaču.

Ova dva načina okretanja izveštaja zahtevaju da se dodaju i dva komandna tastera (*Button*): npr. *btnPrintPreivew* za pregled izveštaja pre štampe i *btnPrintReport*.

Rad sa izveštajima je detaljnije opisan u poglavlju 3.3.7. Ovde napominjemo da je potrebno kreirati i dokument izveštaja u npr. *Crystal Report* generatoru izveštaja i postaviti komponentu *CrystalReport Viewer* na posebnu formu za prikaz izveštaja. Zatim se *Viewer* povezuje sa *RPT* datotekom na disku. Tasteri na formi ŠIFARNICI → PREDMETI za pokretanje ove druge forme za prikaz izveštaja, u okviru procedure za *Click* događaj, sadrže sledeće naredbe za deklarisanje objekta tipa klase *frmStampaPredmeta*, instanciranje objekta *fIzvestajPredemti* navedene klase forme i prikazivanje forme sa *ShowDialog* metodom:

```
Dim fIzvestajPredemti As frmStampaPredmeta
fIzvestajPredemti = New frmStampaPredmeta
fIzvestajPredemti.ShowDialog()
```

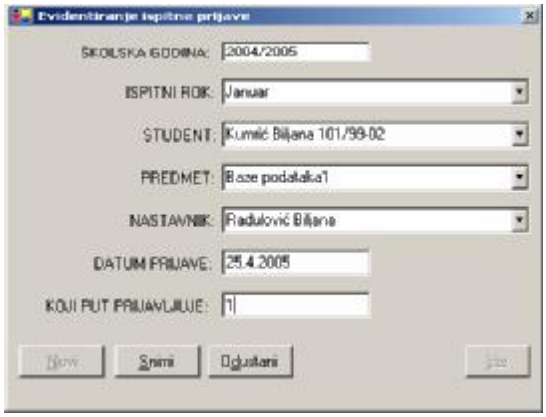
Na kraju postupka kreiranja šifarničke forme, potrebno je još omogućiti snimanje podataka iz *DataSet* objekta u bazu podataka (*Update* metoda *DataAdapter*-a), kao i izlaz sa forme i povratak u glavni meni aplikacije. Sledeće naredbe se pišu na *btnIzlaz* tasteru u okviru procedure za *Click* događaj:

```
If DataSetPredmeti.HasChanges Then
OleDbDataAdapterPredmeti.Update(DataSetPredmeti)
End If
Me.Close()
```

3.3.5. IMPLEMENTACIJA PROCESA ZA AŽURIRANJE

Implementcija procesa za ažuriranje je određena zahtevima korisnika, primitivnim procesom strukturne sistem analize 2.1. EVIDENTIRANJE ISPITNE PRIJAVE, njegovom logikom i poslovnim pravilima, te podmodelom ER modela podataka na osnovu kojeg je CASE alat kreirao podmodel šeme relacione baze podataka.

Objektnim pristupom razvoju softvera mogao bi se još kreirati i set UML dijagrama za detaljnije i preciznije određenje ovog ekrana, npr. USE CASE za opis softverskih funkcija, ACTIVITY diagram za opis logike procesa i poslovnih pravila i na kraju CLASS diagram koji na osnovu relacionog podmodela podataka definiše klase objekata, njihova obeležja, metode i veza između tih klasa objekata.
Prikaz izgleda forme za evidentiranje ispitne prijave je dat na sledećoj slici:



Slika 3.67. Forma za evidentiranje ispitne prijave

Ova forma sadrži sledeće kontrole:

- Tri *TextBox* komponente, za unos podataka sa tastature: *txtSkolskaGodina, txtDatumPrijave, txtKojiPut*;
- Četiri kontrole tipa *ComboBox* za izbor već postojećih podataka evidentiranih kroz šifarnike ili neke druge procese obrade podataka: *cmbRok, cmbStudent, cmbPredmet, cmbNastavnik*;
- Četiri kontrole tipa *Button* za operacije: dodavanja nove ispitne prijave, potvrdu snimanja te prijave u bazu podataka, odustanka od unosa tekuće prijave i izlaz sa forme i povratak u glavni meni aplikacije - *btnNovi, btnSnimi, btnOdustani, btnIzlaz.*
- Nekoliko kontrola tipa *Label* – oznaka, tj. natpisa sa leve strane svakog od polja za unos podataka.

Osim ovih komponenti, treba dodati i prethodno formiranu konekciju do baze podataka, četiri *DataAdapter*-a i četiri *DataSet*-a (slika 3.68):

1. *OleDbDataAdapterStudenti1* (Select * from Student) i *DataSetStudent1* – memorijska kolekcija podataka o studentima, koja će biti izvor podataka za *cmbStudent* (*DataSource=DataSetStudent1* i *DisplayMember* se generiše kodom u proceduri *frmPrijavaIspita_Load*:

```
OleDbDataAdapterStudenti1.Fill(DataSetStudent1)
Dim brsl As Integer
Dim student As String
For brsl = 0 To DataSetStudent1.Tables("STUDENT").Rows.Count - 1
     student = DataSetStudent1.STUDENT.Rows(brsl).Item(2) & " " &
      DataSetStudent1.STUDENT.Rows(brsl).Item(1) & " " &
      DataSetStudent1.STUDENT.Rows(brsl).Item(0)
     cmbStudent.Items.Add(student)
Next brsl
```

2. *OleDbDataAdapterPredmeti1* (Select * from Predmet) i *DataSetPredmet1* – memorijska kolekcija podataka o predmetima, koja će biti izvor podataka za

cmbPredmet (*DataSource=DataSetPredmet1* i *DisplayMember* se generiše kodom u proceduri *frmPrijavaIspita_Load*:

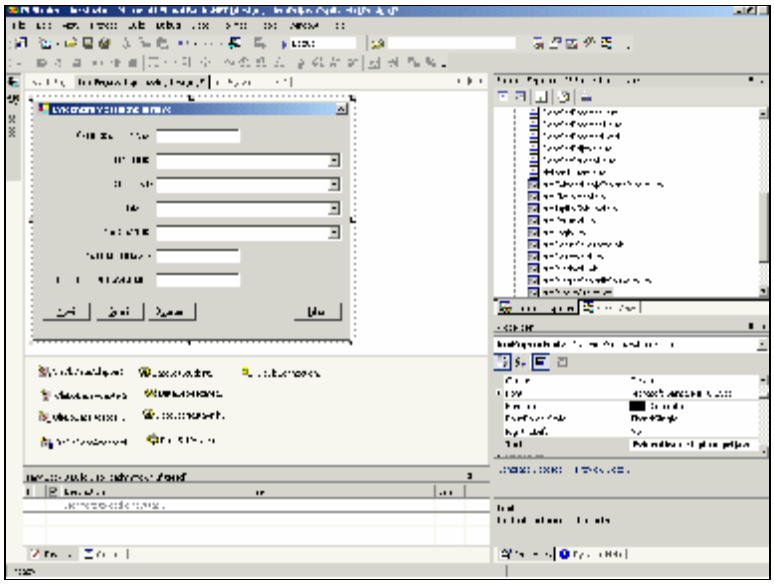
```
OleDbDataAdapterPredmeti1.Fill(DataSetPredmet1)
Dim predmet As String
For brsl = 0 To DataSetPredmet1.Tables("PREDMET").Rows.Count - 1
            predmet = DataSetPredmet1.PREDMET.Rows(brsl).Item(1)
            cmbPredmet.Items.Add(predmet)
Next brsl
```

3. *OleDbDataAdapterNastavnici1* (Select * from Nastavnik) i *DataSetNastavnik1* – memorijska kolekcija podataka o nastavnicima, koja će biti izvor podataka za *cmbNastavnik* (*DataSource= DataSetNastavnik1* i *DisplayMember* se generiše kodom u proceduri *frmPrijavaIspita_Load*:

```
OleDbDataAdapterNastavnici1.Fill(DataSetNastavnik1)
Dim nastavnik As String
For brsl = 0 To DataSetNastavnik1.Tables("NASTAVNIK").Rows.Count - 1
            nastavnik = DataSetNastavnik1.NASTAVNIK.Rows(brsl).
            Item(4) & " " & DataSetNastavnik1.NASTAVNIK.
            Rows(brsl).Item(2)
            cmbNastavnik.Items.Add(nastavnik)
Next brsl
```

4. *OleDbDataAdapterPrijava1* (Select * from Ispitna_prijava) i *DataSetPrijava1* – memorijska kolekcija podataka o ispitnim prijavama. *Adapter* objekat puni *Dataset* sledećom naredbom u proceduri *frmPrijavaIspita_Load*:

```
OleDbDataAdapter4.Fill(DataSetPrijava1)
```



Slika 3.68. Dizajn forme za evidentiranje ispitne prijave

Karakteristični elementi realizacije forme za evidentiranje ispitnih prijava su sledeći:

- Jedan od načina validacije podataka, na nivou polja, jeste sprečavanje korisnika da unese simbole koji nisu dozvoljeni domenom za taj tip podatka. U narednom listingu imamo naredbe koje će dozvoliti, korisniku, unos isključivo brojeva u okviru ***TextBox*** kontrole, koja prima numeričke celobrojne vrednosti:

```
Private Sub txtKojiPut_KeyPress(ByVal sender As Object, ByVal e As
      System.Windows.Forms.KeyPressEventArgs) Handles txtKojiPut.KeyPress
       If ((e.KeyChar < "0" Or e.KeyChar > "9") And e.KeyChar <> Chr(8))
      Then
           e.Handled = True
       End If
End Sub
```

- Taster Novi otvara polja za unos podataka i polja za izbor podataka iz skupa postojećih vrednosti kako bi korisnik mogao da unese podatke o novoj ispitnoj prijavi. Vrši se aktiviranje tastera Snimi i Odustani. Sledeći listing ilustruje naredbe koje su napisane u proceduri ***btnSnimi_Click***, a koje vrše dodavanje novog skupa podataka, tj. zapis u memorijsku kolekciju ***DataSetPrijava1***, nakon što su podaci uneti u za to predviđena polja. Deklariše se objekat tipa ***DataRow***, metodom ***NewRow Tables*** kolekcije ***DataSet***-a, dodaje se nov memorijski zapis (slog). Zatim se dodeljuju vrednosti obeležjima ***DataSet***-a, koja moraju odgovarati onima u tabeli baze podataka. Na kraju se metodom Add u okviru ***Rows*** kolekcije ***DataSet***-a vrši dodavanje zapisa u memorijsku kolekciju ispitnih prijava. Primer ilustruje i korišćenje MessageBox komponente za prikaz poruka i informacija korisniku.

```
Private Sub btnSnimi_Click(ByVal sender As System.Object, ByVal e As
      System.EventArgs) Handles btnSnimi.Click
       Dim myRow As DataRow
       Dim idprijave As Long
       idprijave = DataSetPrijava1.Tables("ISPITNA_PRIJAVA").Rows.Count + 1
       myRow = DataSetPrijava1.Tables("Ispitna_prijava").NewRow
       myRow("rbprijave") = idprijave
       myRow("idpredmeta") =
      DataSetPredmet1.PREDMET.Rows(cmbPredmet.SelectedIndex).Item(0)
       myRow("brojindeksa") =
      DataSetStudent1.STUDENT.Rows(cmbStudent.SelectedIndex).Item(0)
       myRow("idnastavnika") =
      DataSetNastavnik1.NASTAVNIK.Rows(cmbNastavnik.SelectedIndex).Item(1)
       myRow("ispitnirok") = cmbRok.Text
       myRow("skolskagodina") = 2005
       myRow("datumprijave") = Now.ToShortDateString
       myRow("polozio") = "ne"
       myRow("kojiput") = Val(txtKojiPut.Text)
       myRow("ponisten") = False
       DataSetPrijava1.Tables("Ispitna_prijava").Rows.Add(myRow)
       MessageBox.Show("Ispitna prijava je uspesno evidentirana.", "
      Informacija o unosu podataka", MessageBoxButtons.OK,
      MessageBoxIcon.Information)
End Sub
```

- Taster Odustani prikazuje korisniku poruku kako bi izvršio potvrdu željene akcije odustajanja od unosa nove ispitne prijave:

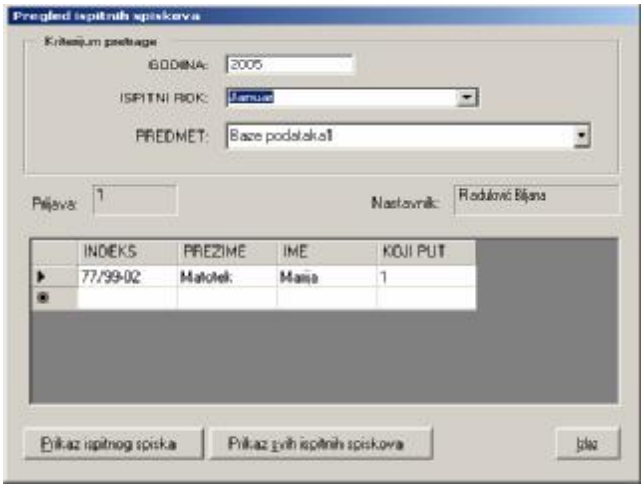
```
Private Sub btnOdustani_Click(ByVal sender As System.Object, ByVal e As
      System.EventArgs) Handles btnOdustani.Click
       While (1 = 1)
       If MessageBox.Show("Da li ste sigurni?", "Pitanje",
      MessageBoxButtons.YesNo, MessageBoxIcon.Question) = DialogResult.No
      Then Exit While
       End While
End Sub
```

- Na tasteru Izlaz moramo završiti transakciju unosa novih podataka, tako što ukoliko je bilo izmena u *DataSetPrijava1*, vršimo snimanje podataka iz *DataSet* objekta u bazu podataka (*Update* metoda *DataAdapter*-a):

```
Private Sub btnIzlaz_Click(ByVal sender As System.Object, ByVal e As
      System.EventArgs) Handles btnIzlaz.Click
       If DataSetPrijava1.HasChanges Then
           OleDbDataAdapterPrijava1.Update(DataSetPrijava1,
      "ISPITNA_PRIJAVA")
       End If
       Me.Close()
End Sub
```

3.3.6. REALIZACIJA UPITA PODATAKA

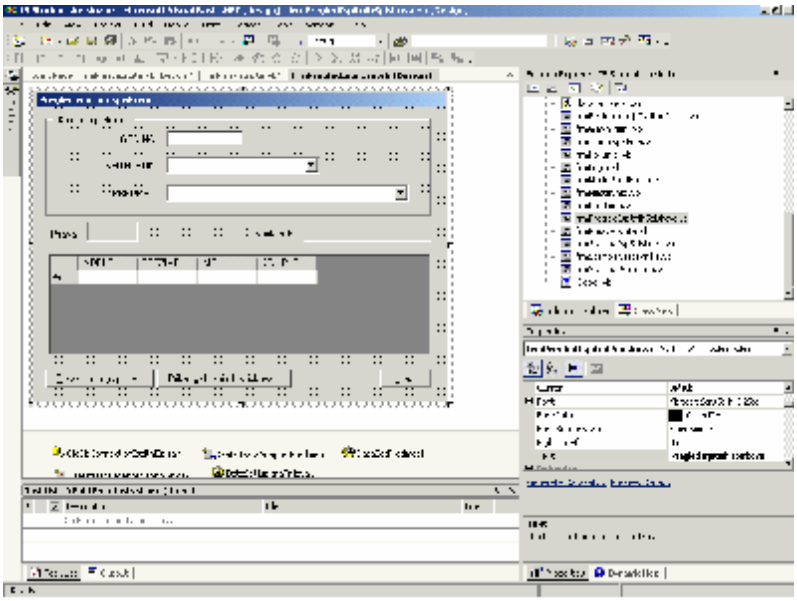
Forme za upite podataka (slika 3.69.) se koriste za pretragu, pronalaženje i izdvajanje podataka po nekom zadatom kriterijumu koji je utvrđen od strane korisnika informacionog sistema, a koji se ne mora obavezno nalaziti u okviru nekog od izveštaja a važan je za poslovanje realnog sistema, smanjuje vreme potrebno za obradu podataka, povećava produktivost i efikasnost u radu, a može se koristiti i za donošenje određenih odluka.



Slika 3.69. Forma za pregled ispitnih spiskova

Na slici 3.70. su prikazane komponente forme za upite podataka:

- Jedna *DataGrid* komponenta za prikaz traženih podataka – *DataGridPrijave*; svojstvo *DataSource*, izabrati: *DataSetPredmeti*; naziv, tj. *Name*= *DataGridPredmeti*; svojstvo *CaptionVisible=False* (Tabealarni prikaz bez posebne naslovne linije); svojstvo *ReadOnly=True* (Ne postoji mogućnost dodavanja novih podataka u tabelarni prikaz podataka o predmetima, ukoliko nije zahtev korisnika i izmena unetih podataka); Ostala relevantna svojstva: *DataMember*, *TableStyles*, *MappingName*, *GridColumnStyles* (4 kolone INDEKS, PREZIME; IME I KOJI PUT), *HeaderText i Width* podesiti kao što je pikazano na slici 3.70., a prema objašnjnju za povezivanje *DataGird* kontrole sa bazom podataka u odeljku 3.34.
- Jedna kontrola tipa *TextBox - txtGodina* za unos školske godine u kojoj se traže ispitne prijave, pošto računarski podržan informacioni sistem ima višegodišnju eksploataciju;
- Dve kontrole tipa *ComboBox* – za izbor naziva predmeta (cmbPredmet) i ispitnog roka (*cmbRok*) za koji će se u tabelarnom prikazu pojaviti podaci o ispitnim prijavama. U *cmbRok* uneti vrednosti: JANUAR, MART, APRIL, MAJ, JUN, SEPTEMBAR, OKTOBAR, NOVEMBAR i DECEMBAR, u okviru svojstva *List.*
- Četiri *Label* kontrole za prikaz ukupnog broja prijava u tom roku i prezimena i imena nastavnika koji će ispitivati studente.
- Komponente za povezivanje forme sa bazom podataka su: *OLEDBConnection1*, *OleDbDataAdapterPredmeti i DataSetPredmet1* za povezivanje *cmbPredmet* sa tabelom PREDMET u bazi podataka, *te OleDbDataAdapterPrijava* i *DataSetIspitnaPrijava1* za povezivanje *Grid* kontrole sa tabelom ISPITNA_PRIJAVA.



Slika 3.70. Dizajn forme za pregled ispitnih spiskova

- Tri tastera (*Button*): *btnIzlaz* – povrataka u glavni meni, *btnPrikaz* – Popunjavanje tabele podacima na osnovu kriterijuma za izdvajanje podataka (*OptionGroup* kontrola),

btnPrikazSvih – prikaz svih ispitnih prijava u bazi podataka (ova opcija je uključena u ovu formu samo kao primer implementacije).

Prilikom učitavanja forme u memoriju, na *frmPregledIspitnihSpiskova_Load* proceduri, potrebno je napuniti *DataSet* podacima *Fill* metodom *DataAdapter*-a, kako je to već i ranije opisano u prethodnim formama:

```
OleDbDataAdapterPredmeti.Fill(DataSetPredmet1)
```

Za taster *btnPrikaz,* u proceduri događaja *Click* su naredbe koje prvo prazne *DataSetIspitnaPrijava1* i *Label*-e od postojećih podataka, a zatim se menja *CommandText* svojstvo *SELECT* naredbe u *OleDbDataAdapterIspitniSpisak*, te se ponovo puni *DataSetIspitnaPrijava1* podacima, a *Label-e* primaju vrednosti iz *Tables* kolekcije *DataSetIspitnaPrijava1* za kolone koje po indeksu odgovaraju redosledu navođenja atributa tabele ISPITNA_PRIJAVA prilikom formiranja u *OleDbDataAdapterIspitniSpisak*. *Label3* prima vrednost *Count* svojstva *Rows* kolekcije iz *Tables* kolekcije *DataSetIspitnaPrijava1* (broj redova koja vraća uit je i broj ispitnih prijava).

```
DataSetIspitnaPrijava1.Clear()
Label2.Text = ""
Label3.Text = ""
OleDbDataAdapterIspitniSpisak.SelectCommand.CommandText =
  "SELECT ISPITNA_PRIJAVA.BROJINDEKSA, PREDMET.NAZIVPREDMETA,
   NASTAVNIK.IMENASTAVNIKA, NASTAVNIK.PREZIMENASTAVNIKA,
   ISPITNA_PRIJAVA.KOJIPUT, ISPITNA_PRIJAVA.ISPITNIROK, STUDENT.IME,
   STUDENT.PREZIME, ISPITNA_PRIJAVA.SKOLSKAGODINA FROM (((STUDENT INNER
   JOIN ISPITNA_PRIJAVA ON STUDENT.BROJINDEKSA =
   ISPITNA_PRIJAVA.BROJINDEKSA) INNER JOIN NASTAVNIK ON
   ISPITNA_PRIJAVA.IDNASTAVNIKA = NASTAVNIK.IDNASTAVNIKA) INNER JOIN
   PREDMET ON ISPITNA_PRIJAVA.IDPREDMETA = PREDMET.IDPREDMETA) WHERE
   ISPITNA_PRIJAVA.SKOLSKAGODINA=" & Val(txtGodina.Text) & " AND
   PREDMET.NAZIVPREDMETA='" & cmbPredmet.Text & "' AND
   ISPITNA_PRIJAVA.ISPITNIROK='" & cmbRok.Text & "'"
OleDbDataAdapterIspitniSpisak.Fill(DataSetIspitnaPrijava1)
Label2.Text = DataSetIspitnaPrijava1.Tables
  ("ISPITNA_PRIJAVA").Rows(Me.BindingContext(DataSetIspitnaPrijava1,
   "ISPITNA_PRIJAVA").Position).Item(7)
Label2.Text = Label2.Text + " " +
   DataSetIspitnaPrijava1.Tables("ISPITNA_PRIJAVA").Rows(Me.BindingConte
   xt(DataSetIspitnaPrijava1, "ISPITNA_PRIJAVA").Position).Item(8)
Label3.Text =
   DataSetIspitnaPrijava1.Tables("ISPITNA_PRIJAVA").Rows.Count
```

Kao što se može videti, u prethodnom listingu, SQL upit koji je formiran u *OleDbDataAdapterIspitniSpisak* provajderu, izdvaja sve ispitne prijave i on je ovde modifikovan, tako što je proširen *WHERE* klauzulom u kojoj se kao dodatni kriterijum dodaju vrednosti iz kontrola iz korisničkog interfejsa:

```
WHERE ISPITNA_PRIJAVA.SKOLSKAGODINA=" & Val(txtGodina.Text) & " AND
       PREDMET.NAZIVPREDMETA='" & cmbPredmet.Text & "' AND
       ISPITNA_PRIJAVA.ISPITNIROK='" & cmbRok.Text & "'"
```

Procedura *Click* za taster *btnSvi* sadrži isti kod kao i prethodno opisana za taster *btnPrikaz* ali je SQL upit takava da prikazuje podatke svih ispitnih prijava, bez *WHERE* klauzule.

```
DataSetIspitnaPrijava1.Clear()
Label2.Text = ""
Label3.Text = ""
OleDbDataAdapterIspitniSpisak.SelectCommand.CommandText = "SELECT
      ISPITNA_PRIJAVA.BROJINDEKSA, PREDMET.NAZIVPREDMETA,
      NASTAVNIK.IMENASTAVNIKA, NASTAVNIK.PREZIMENASTAVNIKA,
      ISPITNA_PRIJAVA.KOJIPUT, ISPITNA_PRIJAVA.ISPITNIROK,
      STUDENT.IME, STUDENT.PREZIME, ISPITNA_PRIJAVA.SKOLSKAGODINA
      FROM (((STUDENT INNER JOIN ISPITNA_PRIJAVA ON
      STUDENT.BROJINDEKSA = ISPITNA_PRIJAVA.BROJINDEKSA) INNER JOIN
      NASTAVNIK ON ISPITNA_PRIJAVA.IDNASTAVNIKA =
      NASTAVNIK.IDNASTAVNIKA) INNER JOIN PREDMET ON
      ISPITNA_PRIJAVA.IDPREDMETA = PREDMET.IDPREDMETA)"
OleDbDataAdapterIspitniSpisak.Fill(DataSetIspitnaPrijava1)
```

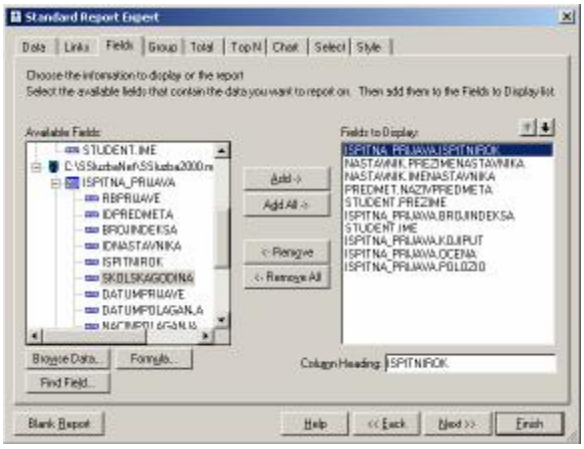
3.3.7. RAD SA IZVEŠTAJIMA

Kreiranje izveštaja u *Crystal Report* generatoru izveštaja ima nekoliko koraka:

1. Dodavanje novog izveštaja u *VisualBasic.NET* projekat. Izabrati u glavnnom meniju *Project* → *Add New Item*. Otvara se ekran kao na slici 3.71. Izabrati u *Templates* → *CrystalReport*. U polje *Name* upisati umesto *CrystalReport1* → *crpPredmeti*. Izabrati opciju *Open*.

Slika 3.71. Kreiranje izveštaja – dodavanje u projekat

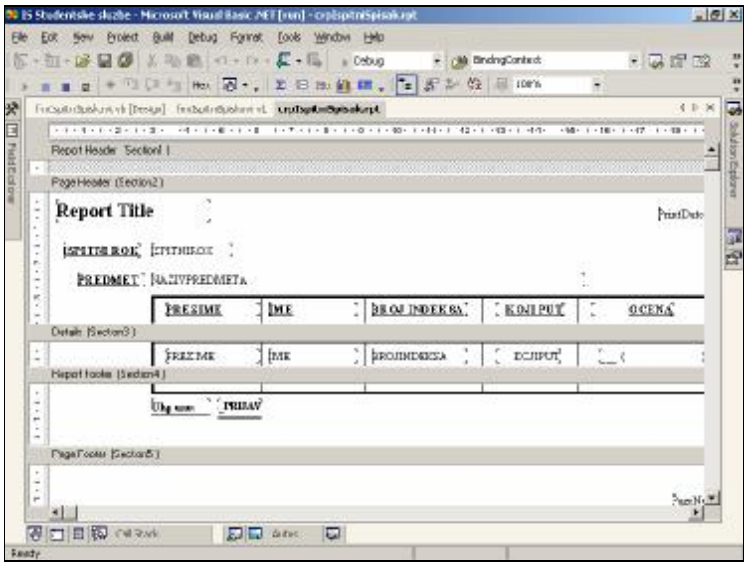
2. Izabrati tip izveštaja *STANDARD* (*Choose an Expert*).
3. Podesiti sledeće parametre: određivanje izvora podataka za prikaz u izveštaju (*Data* kartica). Izabrati *OLEDB*, zbog korišćenja *Access* SUBP-a. Odrediti putanju do datoteke baze podataka (*Database Name* polje). Nakon opcije *Finish*, otvara se naredni ekran:



Slika 3.72. Kreiranje izveštaja – izbor kolona tabele za izveštaj

U polju *Available Data Sources*, izabrati tabele i preneti u polje *Tables in Report*. Preći na sledeću karticu → *Fields*, *Next* opcijom. Odrediti polja tabele (kolone, obeležja) koja će se naći u izveštaju. Iz *Available Fields*, opcijom *Add* u *Fields To Display*, prebaciti kolone tabela koje će se prikazati u izveštaju.

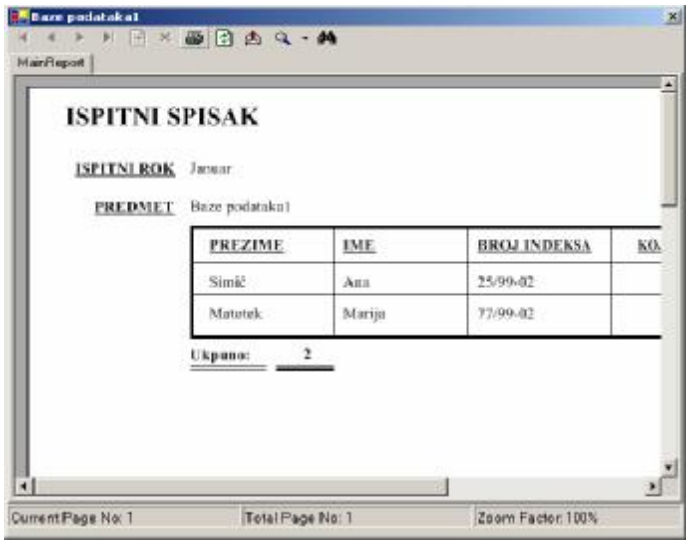
4. Preći na poslednju karticu *Style*, uneti naslov izveštaja (*Title* polje): ISPITNI SPISAK npr., stil može biti proizvoljan. Opcijom *Finish* završiti kreiranje izveštaja. Izveštaj u *Design* režimu rada ima ekran u Editoru izveštaja kao na slici 3.73.



Slika 3.73. Kreiranje izveštaja – Dizajn izveštaja

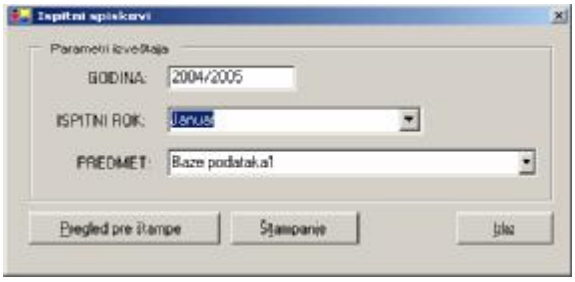
5. Opcijom *Insert* → *Text Field*, dodati u zaglavlje izveštaja (*Page Header* sekcija) podatke o organizaciji (Fakultet u našem primeru). *Report Title* se može menjati istom opcijom *Insert* → *Text Field*. U *Details* sekciji se nalaze *TextBox* kontrole koje će prikazivati podatke o studentima koji su prijavili neki ispit npr. (pošto je izabran izveštaj ISPITNI SPISAK) i to su podaci iz tabele u okviru baze podataka.

Sada je kreiran izveštaj *crpPredmeti.rpt* u okviru *Visual Basic.NET* projekta. Sledi kreiranje forme koja će služiti za prikazivanje izveštaja u programu. Dodati novu formu opcijom *Project* → *Add Windows Form.* Name svojstvo forme promeniti u *frmStampaPredmeta*. U *Text* svojstvo forme uneti: Spisak predmeta. Dodati *Crystal Report Viewer* kontrolu, sa imenom: *crpViewerPredmeti*, u *ReportSource* osobini izabrati putanju do datoteke izveštaja: npr. C:\SSluzbaNET\crpPredmeti.rpt. Za osobinu *DisplayGroupTree*, izabrati *False*.



Slika 3. 74. *Crystal Report Viewer* komponenta

Na kraju se može u projekat dodati i jednostavna forma (*frmIspitniSpiskovi*) za pokretanje ovog parametarskog izveštaja (slika 3.75.).



Slika 3. 75. Forma za pokretanje izveštaja

Forma za prikaz i pokretanja izveštaja ima tri tastera. Taster PREGLED PRE ŠTAMPE će pokrenuti *frmStampaPredmeta* i aktivirati *crpPredmeti*. Sa *frmIspitniSpiskovi* se prosleđuje *frmStampaPredmeta* globalna promenljiva *Stampaj* koja ima vrednost *False*. Globalne promenljive se deklarišu u *Module* odeljku projekta. Opcija je *Project* → *AddModule*. Name: Global.vb

```
Module Global
      Public Stampaj As Boolean ‘promenljiva bulovog tipa koja određuje način
                                ‘prikaza izveštaja: EKRAN ili ŠTAMPAČ
End Module
```

```
Private Sub btnPregledPreStampe_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnPregledPreStampe.Click
        Godina = txtGodina.Text
        Predmet = cmbPredmet.Text
        IspitniRok = cmbRok.Text
        Stampaj = False
        Dim fStampaIspSpiskova As frmStampaIspSpiskova
        fStampaIspSpiskova = New frmStampaIspSpiskova
        fStampaIspSpiskova.ShowDialog()
    End Sub
```

Taster ŠTAMPANJE će pokrenuti *frmStampaPredmeta* i aktivirati *crpPredmeti*. Sa *frmIspitniSpiskovi* se prosleđuje *frmStampaPredmeta* globalna promenljiva *Stampaj* koja ima vrednost *True*. Kod porcedure *Click* za taster ŠTAMPANJE:

```
Private Sub btnStampa_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnStampa.Click
        Godina = txtGodina.Text
        Predmet = cmbPredmet.Text
        IspitniRok = cmbRok.Text
        Stampaj = True
        Dim fStampaIspSpiskova As frmStampaIspSpiskova
        fStampaIspSpiskova = New frmStampaIspSpiskova
        fStampaIspSpiskova.ShowDialog()
End Sub
```

Promenljiva STAMPAJ određuje da li će se *CrystalReportViewer* prikazati na formi *frmStampaPredmeta* - ako je u pitanju PREGLED PRE ŠTAMPE ili ne i izvršiti samo štampanje izveštaja, bez prikaza forme na ekranu (*Me.Close* naredba), kao što se vidi u Load proceduri *frmStampaPredmeta*:

```
Private Sub frmStampaPredmeta_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        If Stampaj = True Then
            crpViewerPredmeti.PrintReport()
            Me.Close()
        End If
    End Sub
```

Forma sa *CrystalReportViewer* komponentom *frmStampaPredmeta* u *Load* proceduri za form objekat ima naredbu za postavljanje vrednosti *SelectionFormula* svojstva *CrystalReportViewer1* komponente kako bi se izvršilo štampanje samo onih ispitnih

spiskova koji odgovaraju parametrima koje je korisnik odabrao na formi za pokretanje izveštaja – *frmIspitniSpiskovi*.

```
Private Sub frmStampaIspSpiskova_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        CrystalReportViewer1.SelectionFormula  =  "{PREDMET.NAZIVPREDMETA}='"
       & Predmet & "' AND {ISPITNA_PRIJAVA.ISPITNIROK}='" & IspitniRok & "'"
        CrystalReportViewer1.RefreshReport()
        If Stampaj = True Then
            CrystalReportViewer1.PrintReport()
            Me.Close()
        End If
    End Sub
```

Ovi parametri štampe se prenose, u ovom primeru, globalnim promenljivama: *Predmet* i *IspitniRok*, koje takođe treba deklarisati kao globalne promenljive u modulu.

```
Module Global
      Public Stampaj As Boolean ‘promenljiva bulovog tipa koja određuje
                                ‘način prikaza izveštaja: EKRAN ili ŠTAMPAČ
      Public IspitniRok As Integer
      Public Predmet As String
End Module
```

3.4. RAZVOJ INTERNET APLIKACIJA

Rad sa udaljenim bazama podataka preko Internet stranica bazira se na HTTP protokolu. U osnovi ovog protokola je koncept rada sa statičkim stranicama, u kom autor stranice mora da je nakon kreiranja postavi na diskove Web servera.

Korisnik (klijent), koji želi da pokrene takvu stranicu, u Web čitaču unosi URL adresu stranice. Tada Web čitač šalje zahtev (HTTP *request*) Web serveru. Nakon što dobije zahtev, Web server provera da li se tražena stranica nalazi na resursima sa datotekama. Ako je pronađe, pakuje HTML kod u stranici pomoću TCP protokola, adresira pakete HTTP-om i vraća ih u mrežu. Ako server ne pronađe traženu datoteku, generiše sopstvenu HTML stranicu sa porukom o greški. [IVKOVIĆ i sar., 2005]

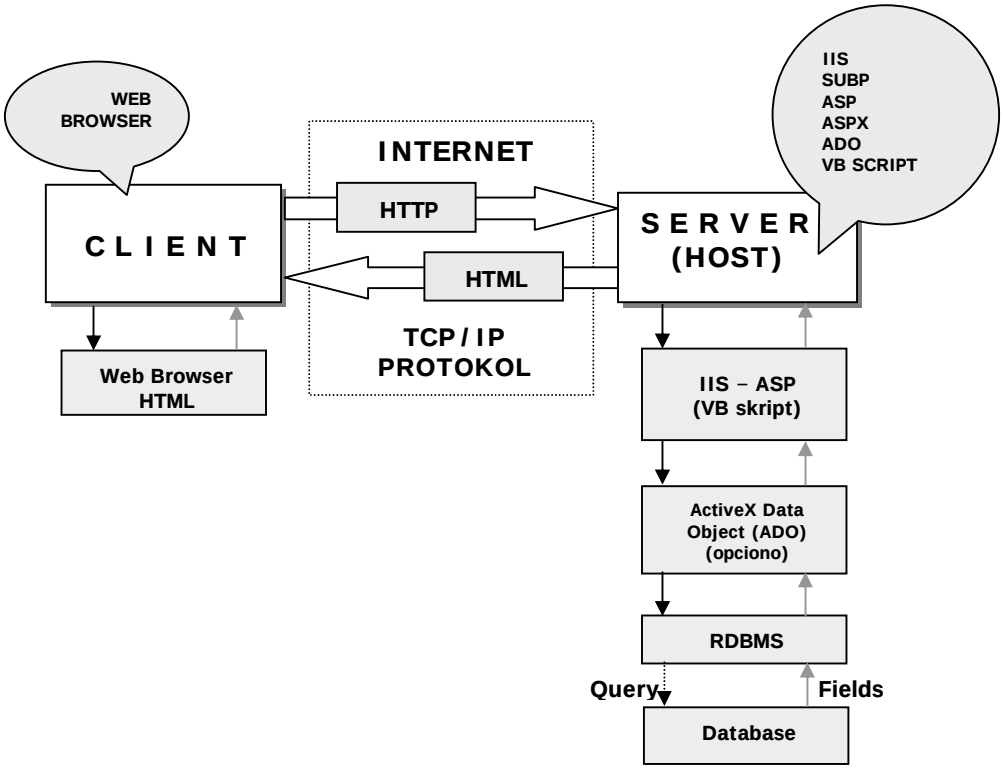
Web je funkcionisao 90.-tih godina prošlog veka upravo na ovakav način prikazujući samo statične HTML stranice, bez uzimanja u obzir informacija koje bi pristizale od korisnika. Uskoro se pojavila potreba sve većeg broja korisnika da se razvije tehnologija koja će prikazivati i izvršavati dinamičke stranice koje će menjati sadržaj u zavisnosti od zatheva korisnika. Ove dinamičke Internet stranice se baziraju na principu da se stranice ne prikazuje sve dok korisnik ne želi stvarno da je vidi i koristi. Ovaj način korišćenja Interneta je uveo mogućnost rada sa podacima koji se nalaze u bazama podataka. Razlika u izvršavanju dinamičkih Web stranica, u odnosu na statičke, jeste ta da se na Web serveru vrši kompajliranje dinamičke stranice, u koju su autori ugradili programske instrukcije napisane u nekom script jeziku izvedenom iz određenog programskog jezika. Ove stranice preko određenih programa na serveru i sistema za rukovanje bazama podataka mogu upisivati, čitati, menjati, brisati podatke iz relacione baze podataka. Kada se stranica

iskompajlira, u formi HTML stranice se preko TCP/IP protokola vraća nazad korisniku. [IVKOVIĆ i sar., 2005]

Najčešće korišćene tehnologije za izradu dinamičkih Internet stranica i aplikacija su:

- PHP (*Hypertext Preprocessor*);
- ASP (*Active Server Page*);
- JSP (*Java Server Pages*);

PHP – je script jezik koji se dodaje u HTML kod stranice. Po sintaksi je sličan programskim jezicima *C*, *Java* i *Perl*. Omogućuje kreiranje Web u trenutku kada čitač klijenta pozove PHP stranicu. Tada Web server šalje zahtev PHP procesoru (npr. *Apache Web Server*), koji čita dokument i izvršava iskaze "<?php" ili "<?" za početak i "?>" završetak umetnutog dela koda. PHP procesor tada upisuje dinamički generisani HTML u memoriju Web servera i sadržaj stranice prosleđuje klijentu. PHP može komunicirati sa relacionim sistemima za rukovanje bazama podataka, a najčešće je to MySQL. Stranica urađena u ovoj tehnologiji ima ekstenziju datoteke: *php*. PHP tehnologija se može koristiti na raznim platformama kao što su *Linux, Windows, FreeBSD* [IVKOVIĆ i sar., 2005].



Slika 3. 76. Šema izvršavanja dinamičkih Web stranica u radu sa bazama podataka

ASP – Tehnologija skriptovanja Web stranica slična PHP-u, koja je razvijena od strane *Microsoft*-a, 1996. i 1997. godine. Kod ASP stranica, izvršavanje je moguće samo ako je na Web serveru instaliran program *Internet Information Services* (IIS), koji vrši kompajliranje tih stranica. IIS, preko ADO konekcije i određenih softverskih komponenata, može vršiti komunikaciju sa sistemom za rukovanje bazama podataka. Na kraju se kao rezultat zahteva korisnika, klijentu vraća HTML stranica, koja može sadržati tekstualne i multimedijalne elemente kao i podatke koji su rezultat izvršavanja skriptova ili SQL upita nad bazom podataka. Ova stranica koja stiže nazad korisniku, kao odgovor na njegov zahtev ne sadrži izvorni kod, već samo HTML, tako da se brzo učitava u Web čitač.

Svaki deo HTML stranice unutar specijalnih tagova <% i %> sadrži strukture programskog jezika *Visual Basic*. ASP aplikacija se izrađuje sa sledećim ugrađenim objektima: *Server, Application, Session, Response, Request, ObjectContext* i *ASPError* [IVKOVIĆ i sar., 2005].

Stranica urađena u ovoj tehnologiji ima ekstenziju datoteke: *asp*. Primer izvornog koda Web stranice napisane u ASP tehnologiji:

```
<HTML>
  <HEAD> <TITLE> BaselogNet, autor: KAZI ZOLTAN </TITLE> </HEAD>
  <BODY> <FONT FACE="Verdana" SIZE="2">
    <H> <B> BaselogNET v1.0 </B> </H>
          <%                                                    VB
          Set objTextStream=colFileName.OpenAsTextStream(1,ascii)
          Set colBazaPut=objFileSysObj.GetFile
          ("C:\inetpub\wwwroot\baselognet\baza.txt")
          Set objBazaPut=colBazaPut.OpenAsTextStream(1,ascii)   SCRIPT
          While not objBazaPut.AtEndOfStream
                vPutanjaBaze=objBazaPut.ReadLine
          Wend
          %>
          <P> Baza podataka u kojoj se vrsi pretraga je: </P>
          <P> <% Response.Write(vPutanjaBaze) %> </P>
  </BODY>
</HTML>
```

JSP – U osnovi *Java Server Pages* tehnologije jeste ideja da se razdvoji rad Web dizajnera od Web programera. To su Web stranice koje su kao i PHP i ASP, tekstualne datoteke sa HTML tagovima, kojima se dodaju specijalni tagovi <% i %> identični onima u ASP stranicama. Realizacija se bazira na tome da JSP stranica jeste osnov za izradu HTML dokumenta kakav je specificiran u JSP stranici (tzv. servlet). Nakon generisanja servleta sa izvornim *java* kodom se prevodi u *Java*-bajt kod, a zatim izvršava u programu *Java*-virtuelna mašina na serveru. Proizvod ovog izvršavanja jeste čist HTML, koji se vraća klijentu. Da bi se u JSP tehnologiji gradile aplikacije, moguće je koristiti klase iz raspoloživih *Java* biblioteka i razne *Java* komponente [IVKOVIĆ i sar., 2005]. Stranica urađena u ovoj tehnologiji ima ekstenziju datoteke: *jsp*.

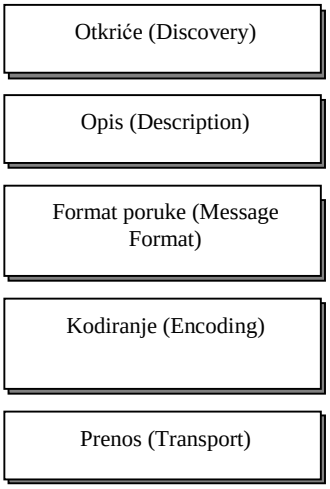
Slika 3.77. Dinamička ASPX stranica na Internetu za korišćenje podataka
iz baze na sajtu Tehničkog fakulteta "Mihajlo Pupin" u Zrenjaninu

Internet aplikacije, izrađene u prethodno navedenim tehnologijama, mogu koristiti sajtove koji besplatno hostuju baze podataka. Primer ovakvog sajta je sa lokacije: http://free-mysql.bizhostnet.com.

3.4.1. WEB SERVISI

Web servisi su nastali iz potrebe da se povežu kompleksna i heterogena okruženja hardverskih i softverskih platformi u računarskim mrežama i na Internetu, u cilju povezivanja različitih aplikacija. To je mehanizam koji omogućuje deljenje podataka između različitih aplikacija i platformi, koristeći određene osobine komponentnog razvoja softvera.

Web servisi su fleksibilne softverske komponente, koje se nalaze u računarskim mrežama ili na Internetu, na tačno određenim lokacijama i predstavljaju delove aplikacione logike koja rešava određeni problem ili izvršava određeni zadatak. Mogu se lako integrisati u delove drugih aplikacija, nezavisno od platforme na kojoj se izvršava aplikacija.



Slika 3.78. Šema elemenata za komunikaciju aplikacija sa Web servisom

Prednosti izrade aplikacija koje koriste Web servise su prema [IVKOVIĆ i sar., 2005] sledeće:

- jednostavnije korišćenje postojeće infrastrukture starije generacije sa novim tehnologijama;
- veća fleksibilnost i lakša nadogradnja sistema, bez potrebe reinženjeringa u slušaju promene obima posla;
- uštede u implementaciji.

Web servisima se može pristupiti korišćenjem standarnih Web protokola u bilo kom mrežnom okruženju. Izgrađeni su na već široko prihvaćenim standardima kao što su HTTP i XML.

Prethodna šema prikazuje osnovne gradivne elemente Web servisa koji se potrebni za ostvarivanje udaljene komunikacije između dve aplikacije.

Otkriće (*Discovery*) – Klijentskoj aplikaciji, kojoj treba pristup funkcionalnosti Web servisa, potreban je način da pronađe lokaciju udaljenog servisa, kroz proces koji se naziva *Discovery*. To se omogućuje kroz centralizovani direktorijum ili pomoću više različitih ad hoc metoda.

Opis (*Description*) – Kada se odredi lokacija pojedinačnog Web servisa, klijentu je potrebno dodvoljno informacija da bi adekvatno uspostavio interakciju sa Web servisom. Opis Web servisa sadrži strukturirane metapodake o interfejsu koji je predviđen da bude korišten od strane klijentske aplikacije, kao i pisanu dokumentaciju o Web servisu sa primerima korišćenja.

Format poruke (*Message Format*) – U cilju razmene podataka, klijent i server treba da se usaglase oko zajedničkog načina na koji će kodirati i formatirati poruke. Standardni način

kodiranja podataka treba da obezbedi da podaci kodirani kod klijenta budu pravilno interpretirani na serveru.

Kodiranje (*Encoding*) – Podaci koji treba da se prenose između klijenta i servera treba da budu kodirani u okviru tela poruke.

Prenos (*Transport*) – Kada je poruka formatirana i podaci postavljeni u telo poruke, oni treba da budu prosleđeni između kljienta i servera preko nekog transportnog protokola, kao što su TCP/IP i drugi.

3.4.2. DISTRIBUIRANE BAZE PODATAKA

Poslednjih godina, dostupnost baza podataka i računarskih mreža kao problem doveo je do razvoja nove oblasti – distribuiranih baza podataka. Distribuirana baza podataka je integrisana baza podataka koja je izgrađena na računarskoj mreži, a ne na pojedinačnom računaru. Podaci koji se nalaze u bazi podataka su memorisani na različitim lokacijama u okviru računarske mreže i programi koji se izvršavaju koriste podatke sa različitih lokacija. Baze podataka mogu uključivati različite sisteme za rukovanje bazama podataka, mogu biti implementirane u okviru različitih arhitektura koje distribuirano izvršavaju transakcije.

Prema Moginu, Lukoviću i Govedarici "*distribuirana baza podataka predstavlja takvu bazu podataka u kojoj su podaci fizički smešteni na najmanje dva servera baze podataka*", [MOGIN i sar., 2000] strana 482.

Distribuirani sistem za rukovanje bazama podataka (*Distributed Database Management System* - DDBMS) se može definisati kao softver koji omogućava upravljanje distribuiranom bazom podataka (DDB -*Distributed Database*) i čini funkcionisanje takvog sistema da deluje korisniku kao da je centralizovana baza podataka.

Osnovne odlike distribuiranih baza podataka su:

- podaci su memorisani na većem broju lokacija, a svaka lokacija predstavlja jedan računar;
- lokacije su povezane računarskom mrežom, pre nego multiprocesorskom konfiguracijom;
- distribuirana baza podataka je logička celina tj. jedinstvena baza podataka;
- DDBMS ima potpunu funkcionalnost DBMS-a;
- prema korisniku se distribuirana baza podataka prikazuje istovetno kao nedistribuirane baze podataka.

Prednosti distribuiranih baza podataka u odnosu na centralizovane baze podataka su:

- implementacija lokalne autonomije u okviru preduzeća čije su pojedini delovi prostorno dislocirani;
- unapređenje performansi (s obzirom da su podaci smešteni blizu mesta gde se koriste i upiti mogu biti podeljeni na više različitih lokacija i izvršavani paralelno);

- unapređena pouzdanost i dostupnost (dolazi do izražaja u slučaju otkaza pojedinačne lokacije);
- ekonomičnost;
- proširivost;
- mogućnost zajedničkog pristupa podacima – deljenje podataka (***shareability***).

Nedostaci sistema distribuiranih baza podataka su:

- kompleksnost (velika mogućnost grešaka u softveru);
- veliki troškovi (razvoj softvera može biti mnogo složeniji i s toga skuplji, a razmena poruka i dodatna računarska obrada uključuje povećanje troškova);
- distribucija kontrole (ne kontroliše samo jedan administrator sistem distribuirane baze podataka);
- manja bezbednost podataka (s obzirom da je sistem distribuiran, šanse za bezbednosnim greškama su veće);
- teške za izmenu (s obzirom da svaka lokacija ima sopstvenu kontrolu);
- nedovoljno iskustva kadrova za rad sa DDB (nedovoljno su zastupljena iskustva u oblasti razvoja distribuiranih baza podataka);

3.4.3. SEMANTIČKI WEB

Koncept semantičkog Web-a predstavlja novu fazu razvoja Interneta, koja treba da prevaziđe neke postojeće probleme. Naime, u prvoj fazi svog razvoja Web je bio zasnovan na korišćenju HTML-a koga su činile isključivo statički kreirane i povezane stranice. Međutim, da bi se povećalo iskorišćenje Web-a, kako bi se npr. mogle direktno koristiti baze podataka ili dinamički kreirati Web sadržaji, počeli su da se koriste različiti skript jezici koji su Web proširili sa ovim mogućnostima. Korišćenjem skript jezika implementiran je veliki broj različitih Web aplikacija, kao što su poslovne aplikacije koje koriste baze podataka. Vizuelni prikaz i korisnički interfejs koji je prvenstveno namenjen za komunikaciju čoveka i računara je ono što bi se moglo izdvojiti kao osnovna karkteristika i za dinamičke i za statičke sranice na Web-u. Rezultat oba pristupa je HTML sadržaj koji prikazuje Internet čitač. Bez obzira što su svi Web sadržaji u tekstualnom obliku (tj. HTML-u), u njima ne postoje definisane semantičke relacije. Postojeći Internet pretraživači (npr. Google), za zadati upit od nekoliko reči mogu da vrate kao rezultat pretrage reference na nekoliko stotina hiljada pa i nekoliko miliona stranica. Očigledno je potrebno nešto dodatno postojećem Web-u kako bi se ostvarila semantička interoperabilnost sadržaja [DEVEDŽIĆ, 2004].

WWW je zasnovan na HTTP protokolu i HTML jeziku za označavanje Web dokumenata. Kao takav, on nudi pristup distribuiranim, heterogenim i dosta nestrukturiranim informacijama na raznim serverima. Pa ipak, Web kakvim ga danas poznajemo, dizajniran je za korišćenje od strane ljudi. Takav, kakav je Web ne pruža korisnicima nikakvu pomoć u lociranju, filtriranju i korišćenju nepreglednog mora informacija - korisnici sve moraju da urade sami, od interakcije sa programom za pretraživanje i pregledanja ponuđenih informacija, sve do korišćenja Web formi. Bolje bi bilo kada bi semantika podataka bila eksplicitno predstavljena na Web-u, jer bi to omogućilo inteligentnim agentima i drugim

aplikacijama da automatski razumeju sadržaje Web dokumenata i sami rasuđuju o njima, rasterećujući korisnike u velikoj meri. Uz eksplicitno predstavljenu semantiku podataka i semantiku Web servisa, agenti bi mogli sami i da pokreću Web, odnosno da stupaju u interakciju sa njim u ime korisnika, a ne samo da pretražuju.

Koncept semantičkog Web-a uveo je Tim Berners Lee, kreator sadašnjeg Web-a. Semantički Web treba da podrži semantičku interoperabilnost sadržaja na Web-u. Ideja je da Web sadržaji ne budu samo čitljivi i razumljivi za čoveka, već da budu i mašinski razumljivi. Ovo znači potpuno odvajanje prikaza od sadržaja, ali i više od toga - semantičko označavanje sadržaja. Kako bi se ovo postiglo predlaže se korišćenje tehnika veštačke inteligencije za predstavljanje znanja kao osnovne tačke za buduću infrastrukturu. Ontologije se izdvajaju kao ključni mehanizam za predstavljanje znanja semantičkog Web-a.

Semantički Web se javlja i iz potrebe da se na Internet postave podaci i znanje koje mašine, tj. računari mogu razumeti isto tako dobro kao i ljudi. To se postiže tako što se na tim računarima izvršavaju programi-alati, koji mogu deliti podatke i vršiti obradu deljenih podataka, čak i ako su razvijani potpuno nezavisno jedan od drugog. U osnovi semantičkog Web-a je ideja da se na Internetu nalaze podaci definisani i povezani na način koji nije samo prezentacija i prikaz sadržaja, već automatizacija i integracija podataka, kao i njihovo ponovno korišćenje kroz različite aplikacije.

"*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*." [BERNERS LEE i sar., 2001]

Semantički Web (*Semantic Web*) predstavlja zajednički okvir, koji omogućuje podacima da budu deljeni i ponovo korišćeni u okviru aplikacija, organizacije i u okviru zajednice. Predstavlja zajednički projekat vođen od strane W3C (*WWW Consortium*) sa učešćem velikog broja istraživača i partnera iz industrije. Zasniva se na «Okviru za opis resursa» (*Resource Description framework* - RDF), koji integriše različite aplikacije koristeći XML za sintaksu i URL za imenovanje.

Da bi se obezbedilo postojanje semantičkog Web-a, potrebno je obezbediti širok spektar standarda na kojma se semantički Web bazira. Cela infrastruktura semantički Web je zasnovana na korišćenju standarda:

- za tekst (Unicode, za standardno korišćenje predstavljanja znakova u URI - Uniform Resource Identifier - ima šire značenje od URL-a. URI može da predstavlja: stvar kojoj se može pristupiti preko računarske mreže (HTML dokument), stvar kojoj se ne može pristupiti preko računarske mreže (čovek) i apstraktne koncepte koji fizički ne postoje (npr. pojam kreator Web stranice))
- na sledećem sloju se nalazi definicija XML-a kao ključnog jezika za postizanje interoperabilnosti na Webu. Korišćenjem XML-a definisani su ostali jezici Semantički Web.
- korišćenjem XML-a definisani su RDF (Resource Desription framework) I RDF schema (RDFS) - W3C standardi za opisivanje metapodataka i rečnika koncepata na Webu, koji su zasnovani na XML-u.

- poslednji napori W3C-a koji se odnose na standardizaciju ontoloških jezika se prvenstveno odnose na one jezike koji su zasnovani na RDF(S)-u.

W3C konzorcijum je 2004. godine objavio RDF okvir i preporučio OWL jezik (*Web Ontology Language*) za prezentovanje informacija i razmenu znanja na Web-u. OWL jezik se koristi za objavljivanje i deljenje skupa osnovnih reči, podržava naprednu pretragu Web-a, programske agente i upravljanje znanjem.

Razvoj semantičkog Web-a danas omogućuju dva ključna faktora. Jedna je sve veća produkcija ontologija, kao mašinski predstavljenih i mašinski razumljivih (od strane inteligentnih agenata na Webu) suštinskih znanja o raznim pojmovima, temama i oblastima. Drugi su XML tehnologije koje razvija W3C pod rukovodstvom Tim Berners Lee-a, a koje omogućuju predstavljanje i korišćenje ontologija na Web-u uz upotrebu univerzalnog XML formata. Ontologije nije lako razviti jer zahtevaju domensku ekspertizu, te ih je danas još uvek malo.

U bliskoj budućnosti, zahvaljujući razvoju semantičkog Web-a, mašine će značajno unaprediti svoju funkcionalnost tako što će biti u stanju da mnogo bolje "razumeju" i obrađuju podatke koje u današnjici mogu samo da prikazuju. Semantički Web, kao sistem za predstavljanje znanja, neće biti paralelni, novi Web, već predstavlja proširenje postojećeg, koji će omogućiti komunikaciju i zajednički rad ljudi i mašina, ali i mašina sa drugim mašinama. Kao i Internet, biće decentralizovan koliko god je to moguće. Više o semantičkom Web-u može se naći u [DEVEDŽIĆ, 2004] i [BERNERS LEE i sar., 2001].

3.4.4. MOBILNI AGENTI

Internet je u poslednjih deset godina u velikoj meri transformisao celokupno društvo, način obavljanja različitih oblika komunikacija, učenja, poslovanja i dr. Kao posledica sve bržeg širenja Weba, kao jednog od najznačajnijih servisa Interneta, dolazi i do otežanog pretraživanja informacija, što dalje uslovljava manje iskorišćenje mogućnosti koje Web sa sobom donosi. Kao rešenje ovog problema, u poslednje vreme, pojavljuju se mobilni-inteligentni agenti na Webu.

Mobilni agenti jesu programi koji samostalno, ili na osnovu zahteva korisnika Web-a, preuzimaju određene akcije u skladu sa ciljevima postavljenim od strane korisnika. Agenti imaju mogućnost filtriranja informacija, njihove obrade, kao i povezivanja mnoštva dostupnih podataka na Web-u.

Mobilni agenti predstavljaju posebne mrežne programe namenjene prvenstveno za pretragu informacija u ime korisnika i potom, kada se odgovarajuće informacije pronađu, obavljanje neke transakcije. Program-agent posećuje određene servere u lokalnoj ili svetskoj mreži i sakuplja određene podatke sa svakog diska ili se pomera sa jednog servera na drugi da bi izvršio predviđeni zadataka. Može vršiti pretraživanje i filtriranje ogromnih količina podataka, graditi indekse linkova ka delovima informacija koje se slažu sa kriterijumom pretraživanja, osmatra nove podatke koji se kreiraju dinamički - vremenom, obavlja aktivnosti umesto korisnika u oblasti elektornskog poslovanja. Po povratku sa mreže na

polazni server, mobilni agent može sačiniti izveštaje o prikupljenim podacima ili preneti tražene podatke.

Mobilni agenti imaju sposobnost da se pomeraju od jednog servera ka drugom i da prave nove agente u mreži, tako da jedna od primena ove tehnologije može biti u oblasti paralelnog procesiranja. Kada se za izvršavanje poslova zahteva dosta procesorskog vremena, izvršavanje se raspodeli na više procesora, a infrastruktura servera mobilnih agenata obezbeđuje jednostavan način da se takvi procesi podele. Jedna od oblasti primene mobilnih agenata jeste izrada distribuiranih aplikacija. Programer može da izvrši fleksibilnu distribuciju koda i da ne posmatra "svet" u okviru tradicionalne klijent-server paradigme. Umesto toga, programer može da pomera delove programa na različite servere i obavlja poslove na njima.

Postoje tri vrste uloge mobilnih agenata u distribuiranim aplikacijama:

- kao komunikacioni mehanizam;
- kao sredstvo za transport podataka između servera;
- kao okruženje za deljenje funkcionalnosti aplikacija.

Da bi mobilni agenti mogli da se premeštaju sa jednog na drugi računar u mreži, moraju biti zadovoljeni sledeći uslovi [DCAGLETS]:

- zajednički izvršni jezik koji moraju deliti serveri da bi agenti mogli da prelaze sa jednog na drugi host računar, i to u vrlo heterogenom okruženju platformi i operativnih sistema;
- nepromenljivost procesa koji se izvršavaju u agentima i koji moraju ostati nepromenjeni ili na neki način sačuvani kada agenti migriraju sa jednog servera na drugi;
- postojanje komunikacionog mehanizma između servera na kojima se nalaze agenti zbog njihovog prenosa kao što su npr. TCP/IP, SMTP, HTTP protokoli;
- bezbednost servera domaćina agenata i bezbednost samih agenata. Loše napisani agenti mogu biti uništeni ili promenjeni od strane drugih programa koji se nalaze na serverima, tj. host računarima. Naročito je osetljivo pitanje promene logike koja je ugrađena u agent i koja može biti poverljive sadržine.

Upotreba mobilnih agenata obuhvata i dosta sigurnosnih pitanja koja tek trebaju da budu rešena, jer se mobilni agenti mogu eksponencijalno reprodukovati i tako zauzeti kompletan skup računara koji na sebi imaju servere-domaćine agenata. Takođe, instalacija i kreiranje mobilnih agenata danas nije lak posao, jer nisu razvijeni alati za jednostavno pravljenje mobilnih agenata. Bez obzira na sposobnosti mobilnih agenata, stvarna efikasnost mobilnih softverskih agenata u trenutno postojećem okruženju Web-a je realno niska. U eri dolazeće generacije semantičkog Weba doći će do povećanja inteligentnosti Web-a i sa druge strane, kroz semantičko označavanje pojmova na Webu, njihovo skladištenje, kao i direktnu upotrebu mašina za zaključivanje i klasifikaciju podataka. Takođe, će doći do razumevanja označenih pojmova na Web-u, te će davno postavljeno pitanje o tome koliko su mobilni softverski agenti stvarno inteligentni, dobiti novu dimenziju, povezivanjem sa aktuelnim Web servisima i drugim Web tehnologijama, i bez sumnje postati vredni epiteta inteligentni [DEVEDŽIĆ, 2004].

4. PRIMERI

4.1. INFORMACIONI SISTEMI ORGANIZACIJA

Prikazani primeri informacionih sistema su rezultat rada autora sa studentima u izradi seminarskog rada u okviru predmeta: Informacioni sistemi, Informacioni sistemi u obrazovanju i informaciono-upravljački sistemi.

4.1.1. INFORMACIONI SISTEMI U ZDRAVSTVU

Posmatrajući savremene svetske tokove u razvoju organizacija možemo primetiti da se tehnološka sredstva koja se koriste u njihovom radu sve više usavršavaju i poboljšavaju sa ciljem olakšavanja i unapređivanja svakodnevnih poslova u radu. Povećava se obim neophodnih podataka i dinamika njihove promene i sve je više izražena potreba za izradom raznih izveštaja, analiza, statistika, evidencija, pregleda i sličnog. Zato, postojeće, klasične metode za vođenje evidencije i administrativnih poslova, kao što su prikupljanje, prenos, obrada i skladištenje podataka, postaju složene, spore, zamorne i skupe. Informacioni sistem ambulante za fizikalnu medicinu i rehabilitaciju se bavi ažuriranjem i arhiviranjem podataka, kao i izradom raznih dokumenata potrebnih za funkcionisanje ove ambulante primenom novih tehnologija i sredstava za automatsku obradu podataka kao što su: elektronski računari, matrični, inkjet i laserski štampači i sl. Izvor podataka je bila Ambulanta za fizikalnu medicinu i rehabilitaciju Zdravstvenog Centra "Vršac" iz Vršca.

Opis posla

Pacijent dolazi sa uputom, zdravstvenom knjižicom, medicinskom dokumentacijom (izveštajima lekara specijalista, raznim nalazima (laboratorijski, rentgentski...), na predlog lekara opšte prakse, ili medicine rada, ako je pacijent u radnom odnosu. Medicinski tehničar mu otvara zdravstveni karton, ako pacijent dolazi prvi put. Ako je već dolazio na pregled, tehničar pronalazi karton u kartoteci te službe koji je složen po godini kada su prvi put bili na pregledu. Zatim, tehničar zakazuje pregled kod jednog od fizijatara, upisuje pacijenta u listu čekanja i naplaćuje participaciju. Kada pacijent dođe na zakazani pregled, fizijatar mu uzima anamnezu. Nakon toga se uzima objektivni klinički nalaz (pregled pacijenta). Na osnovu anamneze, kliničkog pregleda i ostale medicinske dokumentacije postavlja se dijagnoza i određuje se odgovarajuća terapija, koja može biti: medikamentozna, ili fizikalna terapija. Fizikalna terapija se sprovodi na odeljenju fizikalne medicine i rehabilitacije. Medikamentozna terapija se sprovodi kod lekara koji je poslao pacijenta u ambulantu fizikalne medicine i rehabilitacije (lekar opšte prakse, ili medicine rada). Zatim se anamneza, klinički nalaz, dijagnoza i prepisana terapija upisuju u karton. Dijagnozu i prepisanu terapiju, fizijatar upisuje i na izveštaj lekara-specijaliste. Sa kartonom i izveštajem, pacijent se vraća kod medicinskog tehnicara, koji ga upućuje na dalje lečenje. Ako mu je određena fizikalna terapija, pacijent odlazi kod fizioterapeuta koji mu sprovodi fizikalni tretman. Fizikalna terapija podrazumeva sledeće procedure: termoterapija, elektroterapija, hidroterapija i kineziterapija. Nakon sprovedene terapije,

pacijent dolazi na kontrolu. Tada se ceo postupak ponavlja dok lečenje ne bude završeno. Kada lečenje bude završeno, fizijatar izdaje izveštaj o završenom lečenju.

Snimak stanja

Ambulanta nema potpuno automatizovanu obradu podataka. Pojedini dokumenti se i dalje obrađuju ručno. U ambulanti se koriste sledeći dokumenti:

- uput
- zdravstvena knjižica
- zdravstveni karton
- izveštaj lekara
- medicinska dokumentacija (izveštaji drugih lekara specijalista, nalazi)

Ambulanta fizikalne medicine i rehabilitacije se sastoji iz dva dela:

- ordinacije
- sale za primenu fizikalne terapije

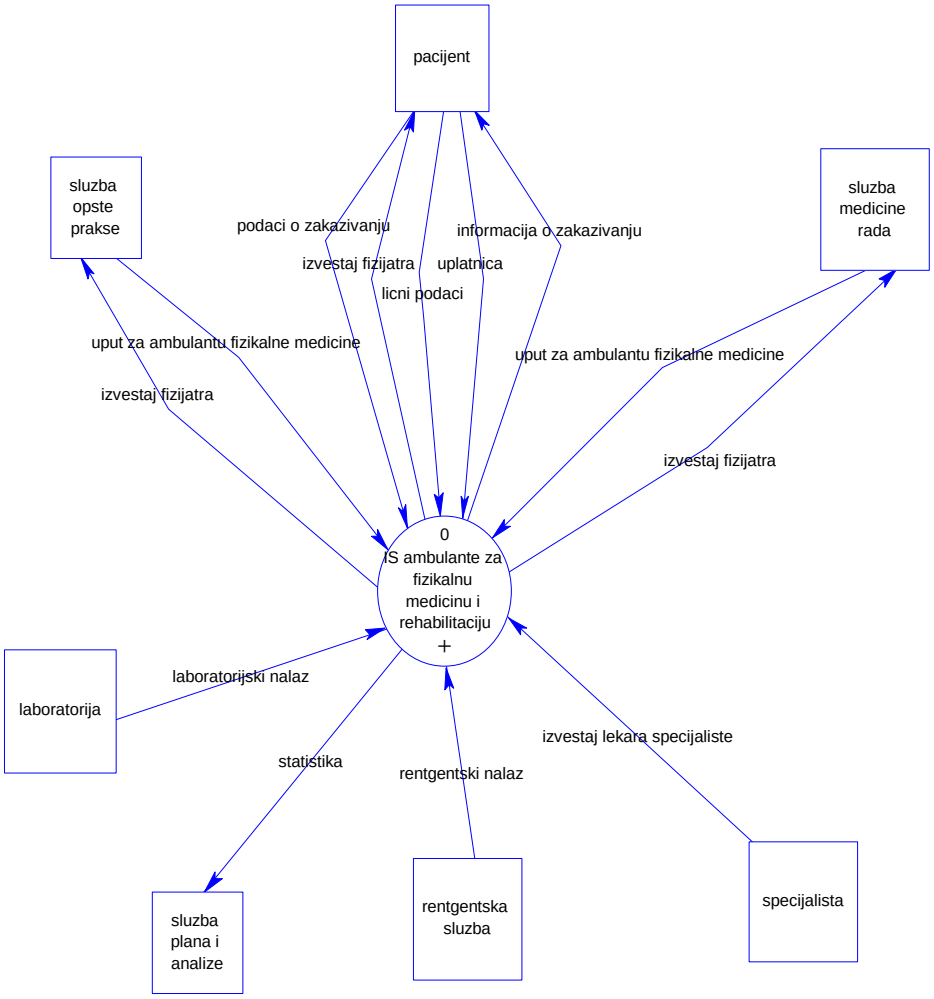
Ambulanta pripada odeljenju fizikalne medicine i rehabilitacije koje je u sklopu zdravstvenog centra. Pacijent može doći u ambulantu sa uputom nekog drugog odeljenja,ili drugog zdravstvenog centra. Takodje, može biti poslat na neko drugo odeljenje sa odeljenja fizikalne medicine i rehabilitacije, ili neki drugi zdravstveni centar. Dakle, rad ovog odeljenja je povezan, kako sa odeljenjima matičnog zdravstvenog centra, tako i sa drugim zdravstvenim ustanovama. U ambulanti rade četiri lekara-fizijatra, medicinska sestra i pomoćno osoblje-higijeničari. Fizijatri vrše pregled pacijenta, postavljaju dijagnozu.i usmeravaju pacijenta na dalje lečenje. Medicinska sestra vrši prijem pacijenata, zakazuje preglede i terapije i asistira fizijatrima, po potrebi. Pomoćno osoblje održava higijenu u ambulanti i, po potrebi, pomaže u transportu teže pokretnih i nepokretnih pacijenata. Najveću odgovornost i ovlašćenja imaju fizijatri. Rad u ambulanti se vrši prema Zakonu o zdravstvenom osiguranju. Lokalna pravila poslovanja su regulisana zahtevima uprave Zdravstvenog centra, kao i potrebama pacijenata. Adekvatna zdravstvena usluga mora biti pružena svakom pacijentu. Što se tiče radnog vremena, ono traje osam sati. Tri dana u nedelji, u popodnevnoj smeni, ambulantni rad obavlja po jedan fizijatar. Veliki problem u ambulanti je problem rada medicinske sestre. Kartoteka je obimna, a zbog nedostatka prostora, zdravstveni kartoni stariji od deset godina se uništavaju. Takođe, veliki je gubitak u vremenu prilikom traženja zdravstvenih kartona u kartoteci. Potrebe ambulante su:

- štampanje izveštaja lekara,
- pretraga i sortiranje zdravstvenih kartona i drugih dokumenata po jednom, ili više kriterijuma,
- automatizmi,
- lakoća korišćenja,
- razni statistički podaci.

Odluke u sistemu donosi lekar-fizijatar. On odlučuje o vrsti terapije koja će se promenjivati, kao i da li će se lečenje nastaviti u toj ustanovi, ili će se pacijent poslati u neku drugu zdravstvenu ustanovu. Program treba automatski da prijavljuje greške koje nastaju prilikom unosa raznih podataka. Uvođenje novog informacionog sistema sigurno bi dovelo do bolje organizacije rada u ambulanti. Recimo, medicinska sestra bi manje vremena trošila u radu u kartoteci, tako da bi više imala vremena za asistiranje fizijatru, kao i za rad sa pacijentima.

MODEL PROCESA

Dijagram konteksta

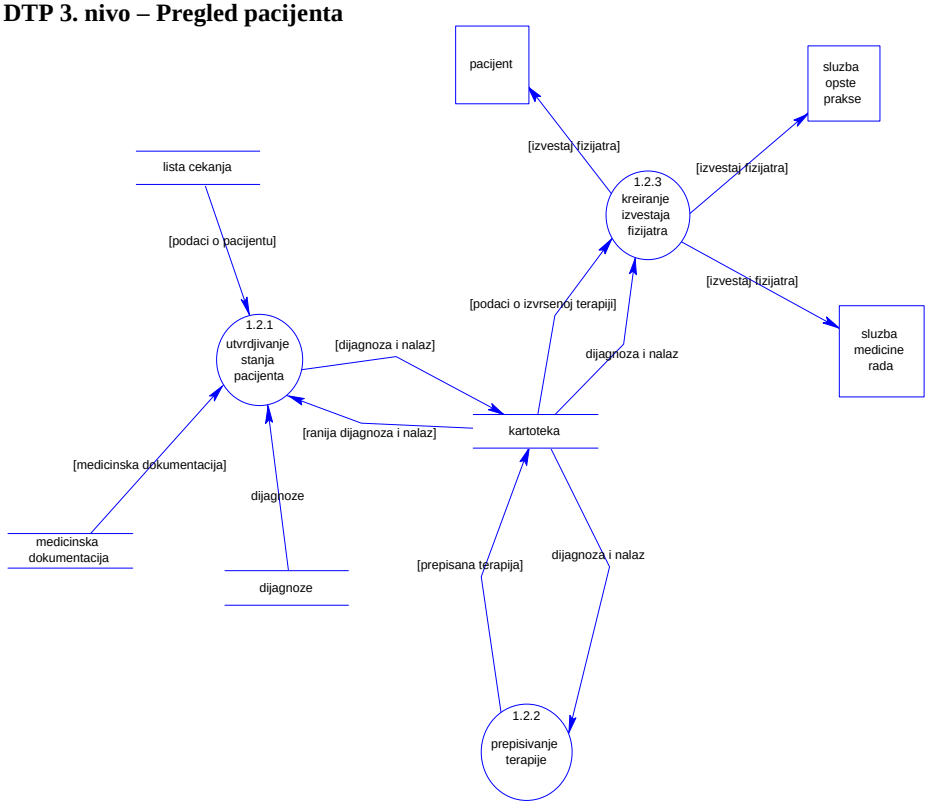


Slika 4.1. Dijagram konteksta za sistem Ambulanta

Stablo procesa

IS ambulante za fizikalnu medicinu i rehabilitaciju [0]
- Evidentiranje dokumenata [2]
  - evidentiranje medicinske dokumentacije [2.1]
    - evidentiranje izvestaja lekara specijalista [2.1.3]
    - evidentiranje laboratorijskih nalaza [2.1.1]
    - evidentiranje rentgentskih nalaza [2.1.2]
  - evidentiranje statistickih podataka [2.2]
- Vrsenje usluga [1]
  - pregled pacijenta [1.2]
    - kreiranje izvestaja fizijatra [1.2.3]
    - prepisivanje terapije [1.2.2]
    - utvrdjivanje stanja pacijenta [1.2.1]
  - prijem pacijenta [1.1]
    - naplata participacije [1.1.3]
    - otvaranje kartona [1.1.2]
    - zakazivanje pregleda [1.1.1]
  - vrsenje terapijskog lecenja [1.3]

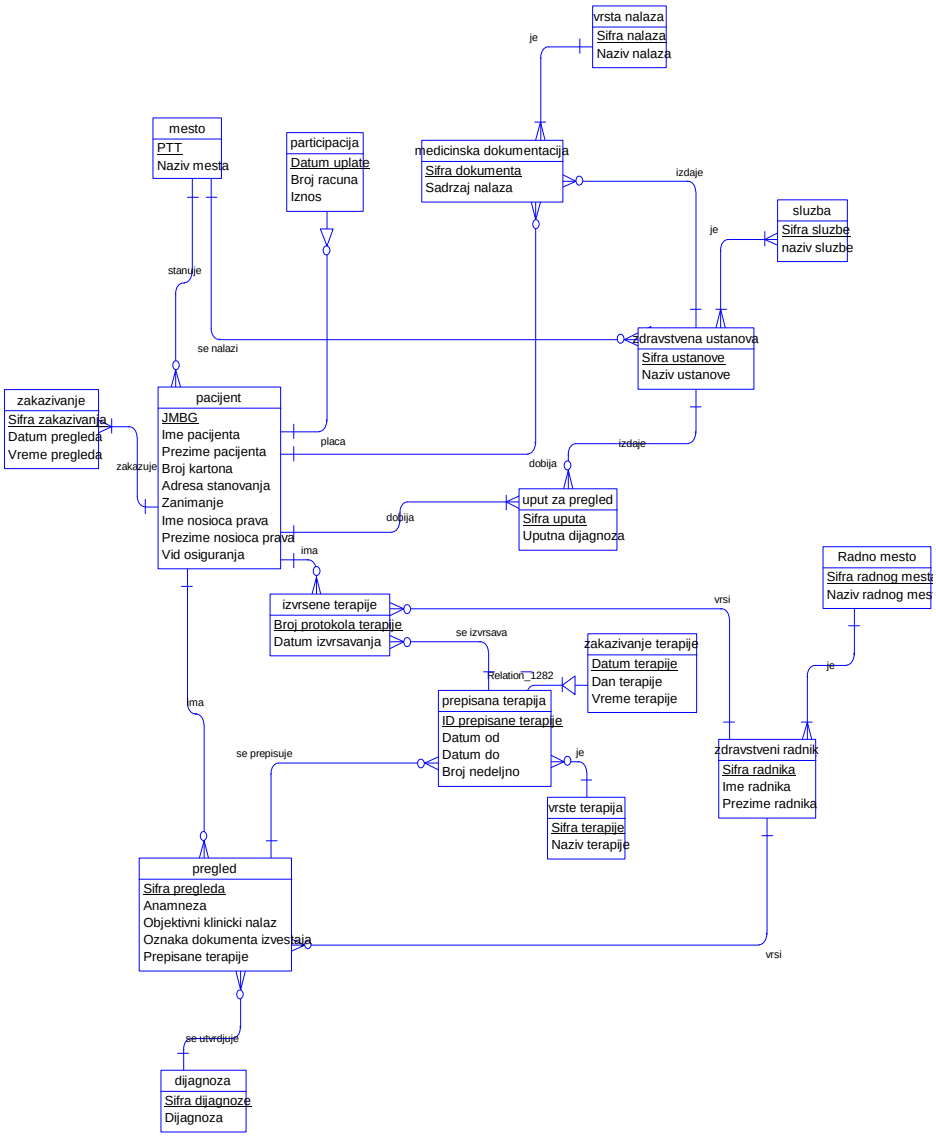
DTP 3. nivo – Pregled pacijenta



Slika 4.2. DTP 3. nivoa za sistem Ambulanta

MODEL PODATAKA

ER Dijagram (Conceptual Data Model):



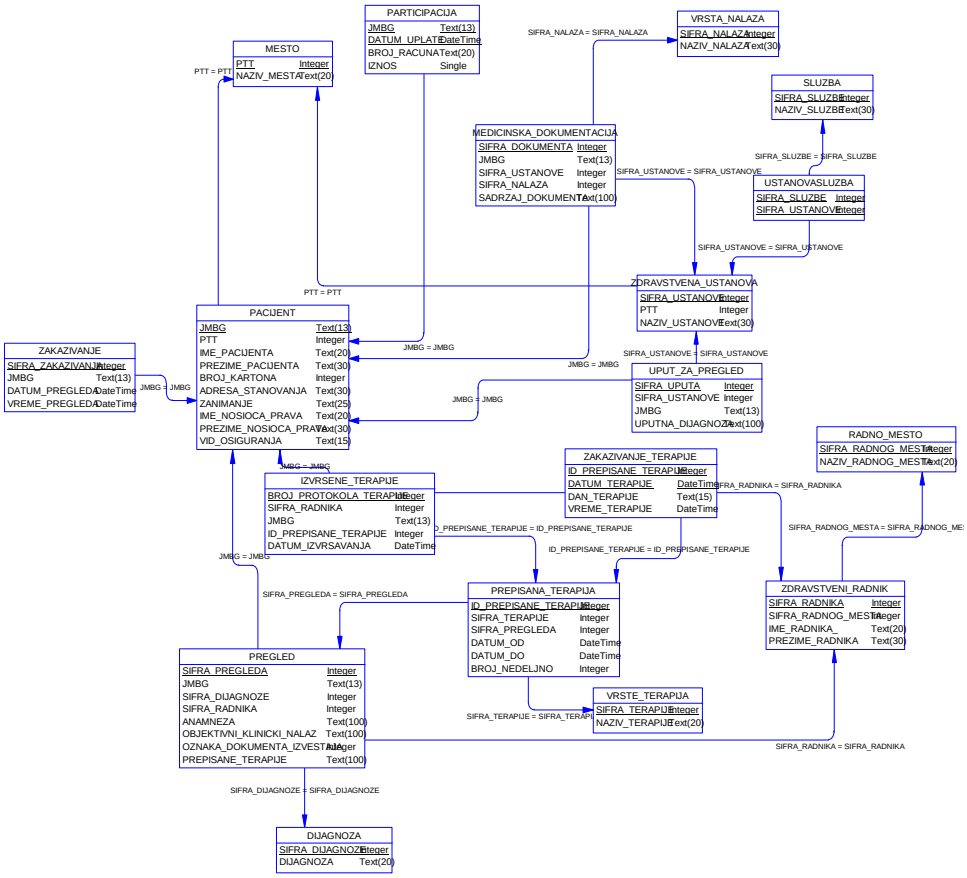
Slika 4.3. Dijagram ER modela podataka za sistem Ambulanta

Lista atributa:

Name	Code	Type
Adresa stanovanja	ADRESA_STANOVANJA	VA30
Anamneza	ANAMNEZA	TXT100
Broj kartona	BROJ_KARTONA	I
Broj nedeljno	BROJ_NEDELJNO	N1
Broj protokola terapije	BROJ_PROTOKOLA_TERAPIJE	I
Broj racuna	BROJ_RACUNA	VA20
Dan terapije	DAN_TERAPIJE	TXT15
Datum do	DATUM_DO	D
Datum izvrsavanja	DATUM_IZVRSAVANJA	D
Datum od	DATUM_OD	D
Datum pregleda	DATUM_PREGLEDA	D
Datum terapije	DATUM_TERAPIJE	D
Datum uplate	DATUM_UPLATE	D
Dijagnoza	DIJAGNOZA	VA20
ID prepisane terapije	ID_PREPISANE_TERAPIJE	I
Ime nosioca prava	IME_NOSIOCA_PRAVA	VA20
Ime pacijenta	IME_PACIJENTA	VA20
Ime radnika	IME_RADNIKA_	VA20
Iznos	IZNOS	N5,2
JMBG	JMBG	A13
Naziv mesta	NAZIV_MESTA	VA20
Naziv nalaza	NAZIV_NALAZA	VA30
Naziv radnog mesta	NAZIV_RADNOG_MESTA	VA20
naziv sluzbe	NAZIV_SLUZBE	VA30
Naziv terapije	NAZIV_TERAPIJE	VA20
Naziv ustanove	NAZIV_USTANOVE	VA30
Objektivni klinicki nalaz	OBJEKTIVNI_KLINICKI_NALAZ	TXT100
Oznaka dokumenta izvestaja	OZNAKA_DOKUMENTA_IZVESTAJA	I
Prepisane terapije	PREPISANE_TERAPIJE	TXT100
Prezime nosioca prava	PREZIME_NOSIOCA_PRAVA	VA30
Prezime pacijenta	PREZIME_PACIJENTA	VA30
Prezime radnika	PREZIME_RADNIKA	VA30
PTT	PTT	N5
Sadrzaj nalaza	SADRZAJ_DOKUMENTA	TXT100
Sifra dijagnoze	SIFRA_DIJAGNOZE	I
Sifra dokumenta	SIFRA_DOKUMENTA	I
Sifra nalaza	SIFRA_NALAZA	I
Sifra pregleda	SIFRA_PREGLEDA	I
Sifra radnika	SIFRA_RADNIKA	I
Sifra radnog mesta	SIFRA_RADNOG_MESTA	I
Sifra sluzbe	SIFRA_SLUZBE	I
Sifra terapije	SIFRA_TERAPIJE	I
Sifra uputa	SIFRA_UPUTA	I
Sifra ustanove	SIFRA_USTANOVE	I
Sifra zakazivanja	SIFRA_ZAKAZIVANJA	I
Uputna dijagnoza	UPUTNA_DIJAGNOZA	TXT100
Vid osiguranja	VID_OSIGURANJA	VA15
Vreme pregleda	VREME_PREGLEDA	T
Vreme terapije	VREME_TERAPIJE	T
Zanimanje	ZANIMANJE	VA25

Tabela 4.1. Spisak atributa ER modela podataka

Relacioni model podataka (Physical Data Model):



Slika 4.4. Dijagram relacionog modela podataka za sistem Ambulanta

OPIS PROGRAMA

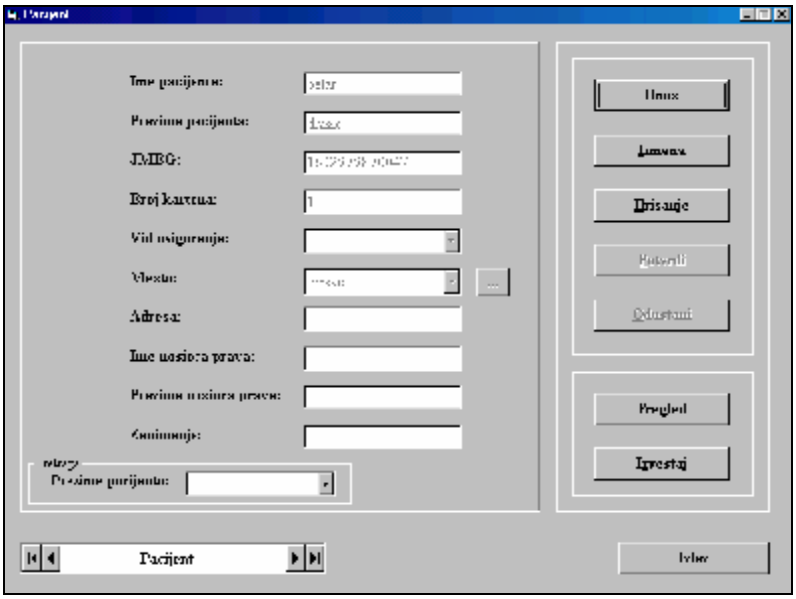
Izvršna verzija programa se kreira jednostavnim pokretanjem fajla SETUP.EXE, nakon čega se pokreće Wizard (eng. Čarobnjak) koji nas vodi kroz postupak instalacije programa. Program se može koristiti sa instaliranim Windows 9x, 2000, Me operativnim sistemom. Rezolucija ekrana koja je potrebna je 800x600 piksela. Softver izmedju ostalog zahteva da fontovi u Windows-u budu podešeni na veličinu: Small Fonts(Control Panel-Display-Settings).

Prilikom pokretanja programa, korisnik mora da se prijavi za rad u programu, tj. mora da unese korisničko ime i šifru (petar, petar). Svaki korisnik ima svoj status, koji može biti administratorski i korisnički. Program nudi različite opcije za rad u zavisnosti od statusa

korisnika. Nakon prijave korisnika, pojavljuje se glavni ekran koji u vrhu sadrži glavni meni sa nekoliko opcija: **šifarnici, obrada, upiti, izveštaji i servisi**.

Šifarnici - Svaki ekran iz menija šifarnici se sastoji iz tri ekranske kartice: ažuriranje, pregled i štampa. Na kartici ažuriranje se vrši ažuriranje, odnosno brisanje i izmena postojećih, ili unos novih podataka. Svaka akcija se mora potvrditi, ili se moze od nje odustati, pritiskom na taster potvrdi, odnosno odustani. Prilikom unosa novih podataka, korisnik može dodavati nove podatke iz drugog šifarnika, pritiskom na taster koji se nalazi pored polja za unos. Na kartici pregled, korisnik može videti sve postojeće podatke u datom šifarniku, a može i da se pozicionira na određeni podatak po određenom kriterijumu. Na kartici štampa korisnik može štampati sve podatke, ili može izabrati podatak koji se bira po određenom kriterijumu, pritiskom na taster štampanje. Pre štampanja se može izvršiti pregled dokumenta, pritiskom na taster pregled pre štampanja.

Obrada - Na svakom od ovih ekrana može da se vrši azuriranje, odnosno, unos novih, ili brisanje i izmena postojećih podataka. Svaka akcija zahteva potvrdu , koja se vrši pritiskom na taster potvrdi, ili se od akcije može odustati pritiskom na taster odustani. Svaki ekran obrade, takođe sadrzi taster pregled i taster izveštaj. Pritiskom na taster pregled, korisnik prelazi na ekran za postavljanje upita, odnosno na ekran za štampanje izveštaja, ako je pritisnuo taster izveštaj. Na ovim ekranima, takodje postoji i pretraga po određenom kriterijumu da bi korisnik mogao da vrši izmenu, ili brisanje postojećih korisnika. Napomena: na ekranu za preglede, u polju prepisane terapije, fizijatar treba da upiše vrste prepisanih terapija, kojim redosledom će biti primenjene i koliko puta nedeljno. Na ovom ekranu postoji taster zakazivanje, kojim se prelazi na ekran za zakazivanje terapije.



Slika 4.5. Evidentiranje pacijenta - softver za sistem Ambulanta

Upiti - Na ekranima za pretragu se vrši pretraga postojećih podataka po određenim kriterijumima koje korisnik sam bira prema potrebi. Redosled radnji je sledeći: na početku se bira kriterijum pretrage, zatim se u polje za pretragu unosi trazeni podatak i nakon toga korisnik pritiskom na taster prikaži započinje pretragu. Sledeći korak je sortiranje pronađenih podataka koji se mogu sortirati po više kriterijuma. U slučaju da korisnik želi da vidi sve postojeće podatke, on će kao kriterijum za pretragu da odabere opciju svi, a polje za pretragu ostavlja prazno.

Izveštaji - Sa ovih ekrana korisniku je omogućeno štampanje, kako svih tako i pojedinih podataka iz obrade. Korisnik, takođe može da pregleda izveštaj pre štampanja, pritiskom na taster **pregled pre štampanja.**

4.1.2. INFORMACIONI SISTEMI U ŠKOLSTVU

Opis posla

Školski dnevnik izdaje izdavačka kuća ovlašćena od strane Ministarstva Prosvete i Sporta Republike Srbije, čijom se odlukom i odobrava njegovo korišćenje u školama rešenjem broj 632-02-093/92-01 od 18.04/94. god. Vođenje dnevnika se poverava nastavniku-odeljenskom starešini na osnovu pravilnika izdatog u "Službenom glasniku Republike Srbije", br. 58/92, 64/92 i 14/2000. Učenici se upisuju u dnevnik na osnovu spiska upisanih učenika i dokumenata traženih pri upisu. Odeljenski starešina upisuje osnovne podatke o učenicima i nastavnicima, o udžbenicima koji se koriste za nastavu, o programu i planu rada odeljenskog starešine, odeljenske zajednice, saveta roditelja, zatim podatke o organizovanim ekskurzijama i društveno-korisnom radu učenika, kao i određivanje dežurnih učenika (redara) i disciplinskih mera. Odeljenski starešina vodi i evidenciju o sastancima odeljenskog veća, zajednice i saveta roditelja, evidenciju o dolascima roditelja na razgovore i sastanke. Obaveza odeljenskog starešine je i da na osnovu podataka iz dnevnika, na polugodištu i kraju školske godine, sumira rezultate i pravi izveštaje o uspehu učenika i izostancima učenika koje potom prezentuje nastavničkom veću škole. Izostanci učenika se pravdaju isključivo na osnovu opravdanja izdatih od strane nadležnih zdravstvenih organizacija ili odlukom odeljenskog starešine. Svakih sedam neopravdanih časova povlači smanjenje ocene za jedan stepen iz vladanja, odnosno za više od 21 neopravdan čas učenik se po pravilu ispisuje iz škole koju pohađa. Po pravilniku već nakon prvih sedam neopravdanih časova, odeljenski starešina je dužan da o tome obavesti učenikove roditelje. Nastavnici predavači (uključujući i odeljenskog starešinu) upisuju u dnevnik ocene učenika iz odgovarajućih predmeta na osnovu pravilnika, takođe vode evidenciju o odsutnim učenicima, evidenciju o održanim časovima (sa kratkim opisom nastavnih jedinica) i evidenciju o održanim pismenim i kontrolnim ispitima. Na kraju polugodišta predmetni nastavnik je u obavezi da na osnovu ocena iz čitavog polugodišta odredi zaključnu ocenu za svakog učenika. Ukoliko na kraju drugog polugodišta učenik ima jednu ili dve negativne zakjučne ocene, on se upućuje na popravni ispit iz odgovarajućih predmeta, što se takođe evidentira u dnevniku. Ukoliko učenik ima više od dve negativne zaključne ocene, on automatski obnavlja godinu bez mogućnosti polaganja predmeta na popravnom ispitu. Nakon završetka školske godine dnevnik se odlaže u arhivu škole.

Snimak stanja

Dokumenti:

- Pravilnik o ocenjivanju br. 110-010/94-02 izdat u "Službenom glasniku Republike Srbije", br. 42 od 26.6./94. Na osnovu ovog dokumenta nastavnici ocenjuju učenike.
- Spisak upisanih učenika koji sastavlja nastavničko veće škole na osnovu predatih dokumenata pri upisu u prvi / peti razred, odnosno njihovog uspeha u prethodnoj godini školovanja, a koji nastavnik – razredni starešina koristi za unos učenika u dnevnik.
- Opravdanje koje izdaje nadležna zdravstvena organizacija na zahtev učenika, a koji isti koristi za pravdanje časova koje je propustio zbog bolesti.
- Đačka knjižica, koju popunjava nastavnik – razredni starešina na osnovu podataka iz dnevnika, a sadrži ocene iz svih predmeta i informacije o izostancima i konačnom uspehu učenika. Ona se popunjava na kraju prvog i drugog polugodišta i predaje se učeniku na uvid. Njenu overu vrši nastavnik – razredni starešina uz saglasnost sa nastavničkim većem i direktorom škole.
- Diploma o uspehu se izdaje učeniku na kraju školske godine ukoliko je učenik ostvario odličan uspeh (5,0). Ona sadrži osnovne podatke o učeniku sa pohvalom nastavničkog veća. Overu vrši direktor škole.
- Operativni nastavni plan na osnovu koga nastavnik vrši upis nastavnih jedinica u dnevnik. Operativni plan za svaki predmet usvaja Ministarstvo Prosvete i Sporta Republike Srbije.

Zakonska regulativa - Korišćenje dnevnika u školama odobreno je odlukom Ministarstva Prosvete i Sporta Republike Srbije, rešenjem broj 632-02-093/92-01 od 18.04/94. god. Vođenje dnevnika (upis i ažuriranje podataka) propisano je pravilnikom izdatog u "Službenom glasniku Republike Srbije", br. 58/92, 64/92 i 14/2000.

Problemi - Nastavnici često zbog glomaznosti samog dnevnika upisuju ocene u pogrešne kolone ili vrste (drugim učenicima ili drugom predmetu) što se kasnije, ukoliko se utvrdi, mora prepravljati. Odeljenski starešina takođe može pogrešiti prilikom proračuna izostanaka i uspeha učenika, što se pokušava izbeći višestrukim ponavljanjem istih. Dnevnik mora biti van domašaja učenika u vremenu kada nastavnik nije u učionici, zbog mogućnosti upisa ili izmene podataka od strane učenika.

Potrebe - Brzo pravljenje izveštaja o uspehu učenika (pojedinačno i sumarno), o izostancima. Smanjenje mogućnosti greške prilikom unosa podataka (pre svega ocena i izostanaka). Onemogućavanje neovlašćene izmene ili unosa podataka.

Odluke - Odluke u vezi podataka iz dnevnika donosi nastavnik – razredni starešina, u saradnji sa nastavnicima – predavačima i nastavničkim većem. Na osnovu izostanaka odlučuje se o kaznenim merama prema učenicima, a na osnovu ocena o uspehu učenika.

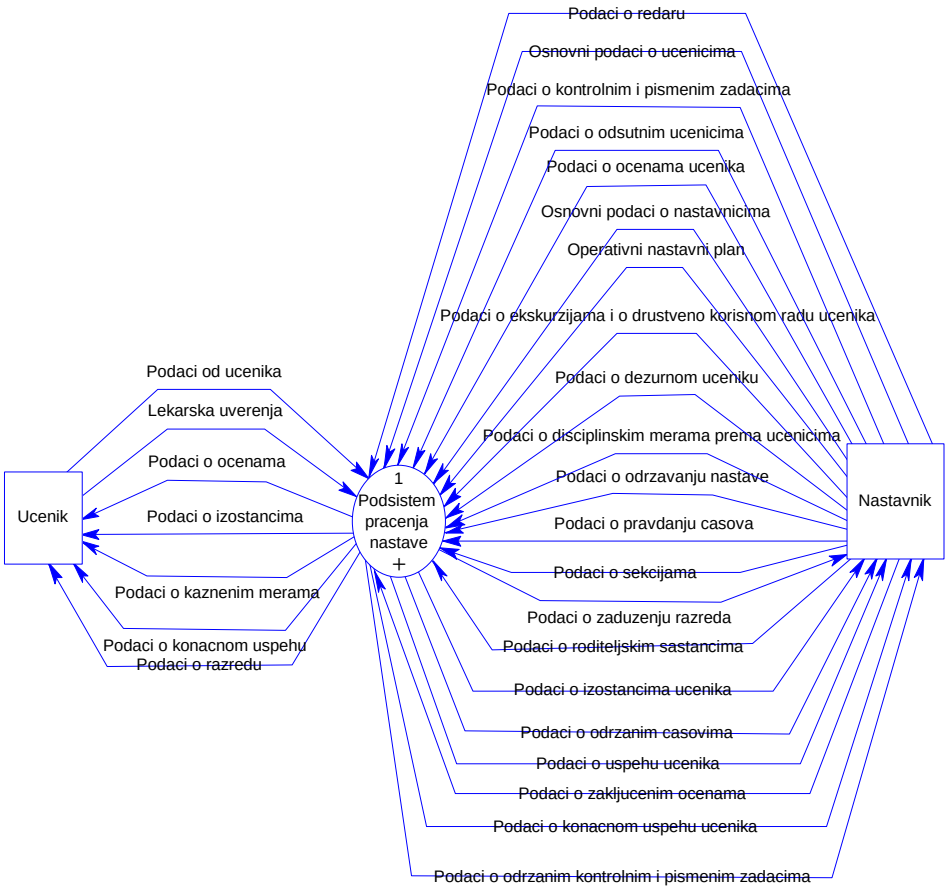
Automatizmi - Ograničenja u vezi raspona ocene (1 – 5). Nemogućnost zaključivanja ocene iz srpskog jezika i matematike učenicima sa manje od dve ocene sa pismenog ispita iz pomenutih predmeta ukoliko ne postoji napomena nastavnika. Onemogućavanje neovlašćenog manipulisanja podacima.

Kadrovi - Kadrovi koji koriste ovaj sistem su nastavnik – odeljenski starešina kao i svi ostali nastavnici – predavači koji drže nastavu odeljenju. Takođe, može ga koristiti i direktor za povremene kontrole rada nastavnika, uspeha učenika škole, itd.

Organizacija rada - Informacioni sistem Elektronski Školski Dnevnik ne menja postojeću organizaciju sistema. On samo olakšava rad, povećava efikasnost rada i donošenja odluka brže i tačnije.

MODEL PROCESA

Dijagram konteksta

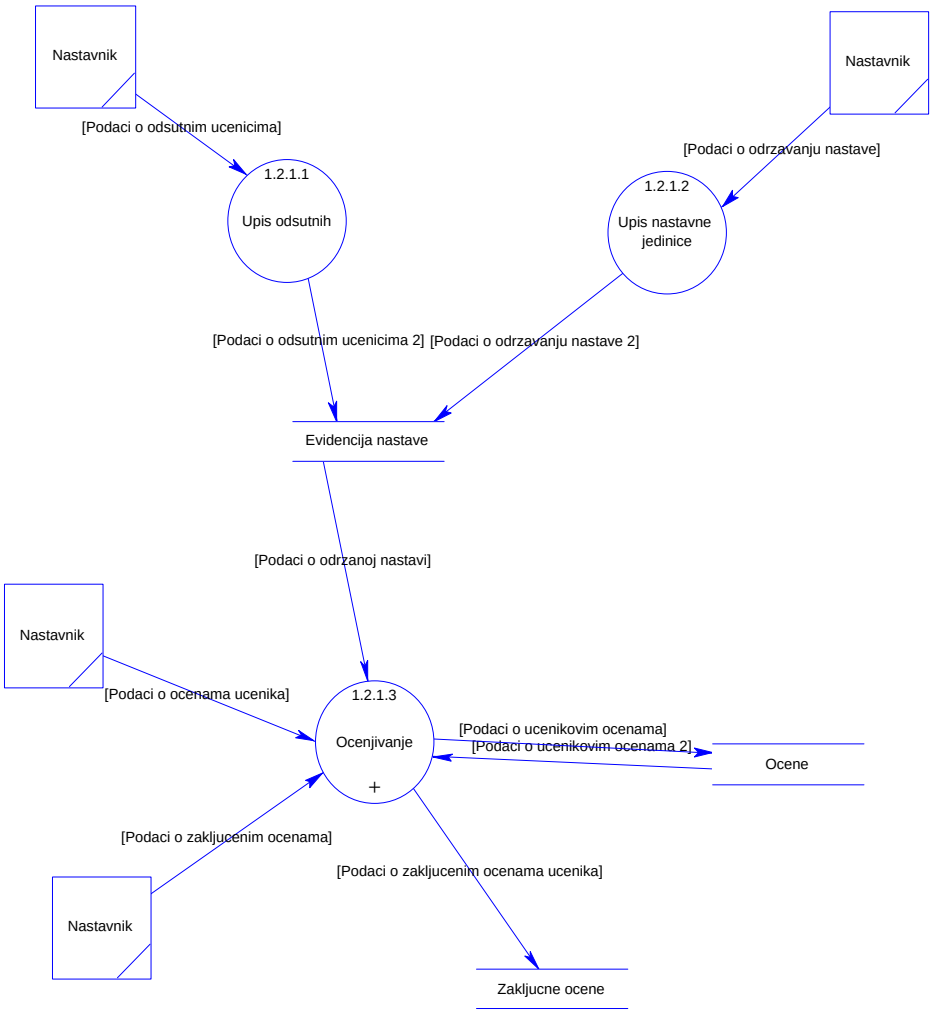


Slika 4.6. Dijagram konteksta za sistem Škola

Stablo procesa

- Podsistem pracenja nastave [1]
  - Izvestavanje [1.3]
    - Izvestavanje o izostancima ucenika [1.3.1]
    - Izvestavanje o nastavi [1.3.2]
      - Izvestavanje o formiranim odeljenjima [1.3.2.1]
      - Izvestavanje o odrzanim casovima [1.3.2.2]
      - Izvestavanje o odrzanim kontrolnim i pismenim zadacima [1.3.2.3]
    - Izvestavanje o uspehu ucenika [1.3.3]
      - Izvestavanje o kaznama [1.3.3.1]
      - Izvestavanje o uspehu [1.3.3.2]
        - Izvestavanje o konacnom uspehu ucenika [1.3.3.2.2]
        - Izvestavanje o trenutnom uspehu [1.3.3.2.1]
  - Izvodjenje nastave [1.2]
    - Evidencija o kontrolnim i pismenim zadacima [1.2.3]
    - Nastavne aktivnosti [1.2.1]
      - Ocenjivanje [1.2.1.3]
        - Ocenjivanje znanja ucenika na casu [1.2.1.3.1]
        - Zakljucivanje ocena [1.2.1.3.2]
      - Upis nastavne jedinice [1.2.1.2]
      - Upis odsutnih [1.2.1.1]
    - Pravdanje casova [1.2.2]
    - Van nastavne aktivnosti [1.2.5]
      - Evidencija aktivnosti ucenika [1.2.5.2]
        - Evidencija drustveno korisnog rada [1.2.5.2.1]
        - Evidencija sekcija [1.2.5.2.2]
      - Evidencija roditeljskih sastanaka [1.2.5.1]
      - Izricanje disciplinskih mera [1.2.5.3]
    - Zaduzenja ucenika [1.2.4]
      - Odredjivanje dezurnog ucenika [1.2.4.2]
      - Odredjivanje redara [1.2.4.1]
  - Priprema nastave [1.1]
    - Formiranje odeljenja [1.1.3]
      - Dodeljivamje zaduzenja nastavnika u odeljenju [1.1.3.2]
      - Dodeljivanje ucenika odeljenju [1.1.3.1]
    - Upis godine [1.1.2]
    - Usvajanje nastavnog plana i programa [1.1.1]

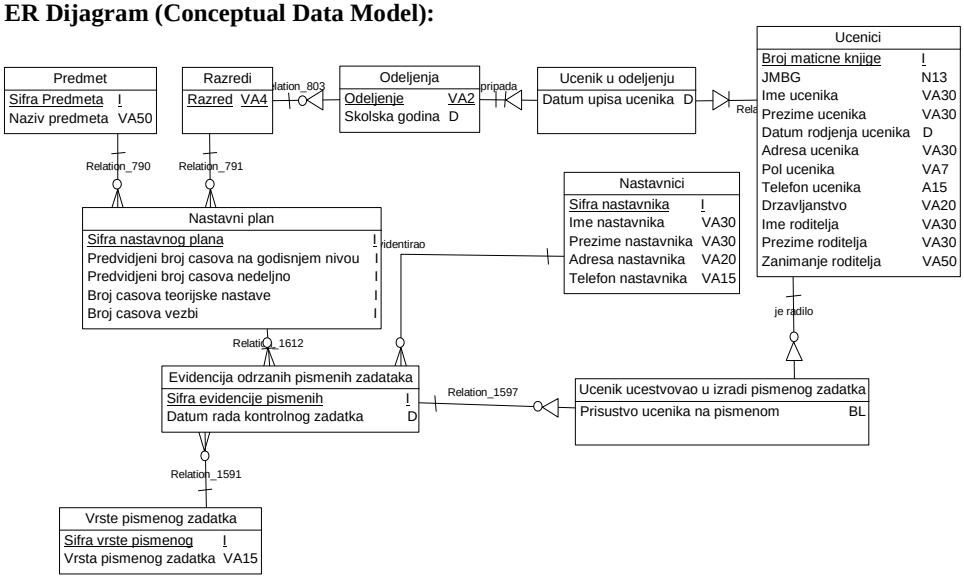
DTP 3. nivo - Nastavne aktivnosti



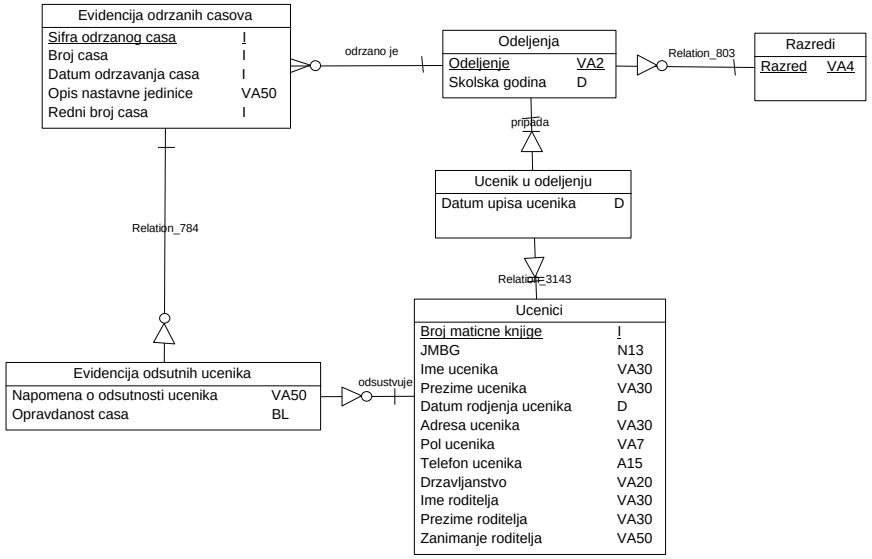
Slika 4.7. DTP 3.nivo za sistem Škola

MODEL PODATAKA

ER Dijagram (Conceptual Data Model):



Slika 4.8. Podmodel - Evidencija o kontrolnim i pismenim zadacima za sistem Škola



Slika 4.9. Podmodel - Izveštavanje o izostancima učenika za sistem Škola

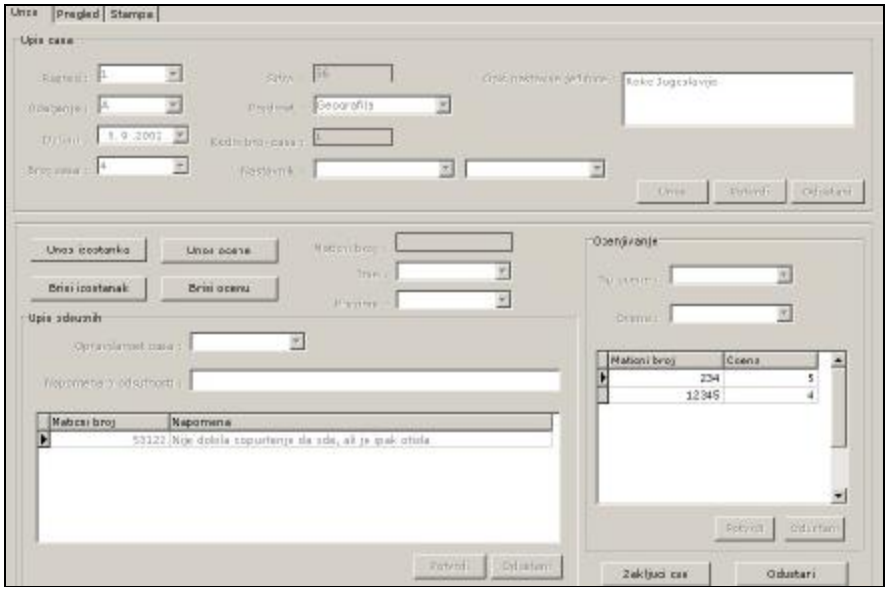
OPIS PROGRAMA

Za instalaciju programa potrebno je sa CD-a ili disketa pokrenuti fajl Setup.exe koji vrši automatsku instalaciju programa u folder koji bira sam korisnik tokom instalacije. Program se pokreće biranjem ikone programa na Desktopu, pozivom iz Start\Programs menija ili pokretanjem fajla ESD.exe iz foldera programa. Program je moguće koristiti na PC kompatibilnim računarima sa instaliranim WIN 98 (nije najpreporučljivije), WIN 2000 ili WIN XP operativnim sistemom i instaliranom podrškom za Borland DataBase Engine. Potrebna rezolucija ekrana je 1024x768 piksela u 32.000 boja minimalno. Softver, takođe, zahteva da u Windows-u budu podešeni fontovi na veličinu: Small Fonts.

Struktura aplikacije se zasniva na stvarnom stanju opisa posla i opštih zahteva. Početna forma je forma za unos podataka korisnika programa radi njegove identifikacije i postavljanja programa u određeni režim rada. Nakon identifikacije korisnika pojavljuje se glavna forma na kojoj se nalazi glavni meni programa sa pozivima svih dostupnih ekrana za tog korisnika. Struktura menija je sledeća:

Ekrani šifarnici služe za manipulaciju opštim (matičnim) podacima. Ovi podaci se uglavnom jednom unose, a višestruko koriste na ekranima za obradu. Sa ovih ekrana se mogu štampati spiskovi osnovnih podataka. Ova aplikacija sadrzi pet šifarnika i to: Mesto, Tip udžbenika, Vrste ocenjivanja, Uloga predmeta i Predmeti.

Ekrani za pregled podataka po odabranim kriterijumima. Korisnik zadaje parametre i dobija tabelarni pregled podataka. U ovoj aplikaciji možete pregledati održane časove za odabrani razred, odeljenje i predmet, zatim izostanke učenika i udžbenike koji se koriste u nastavi.



Slika 4.10. Ekran softvera - evidentiranje nastavnih aktivnosti

Aplikacija sadrži tri ekrana iz opcije menija Obrada: usvajanje nastavnog plana, nastavne aktivnosti i određivanje zaduženja učenika. Sa svakog ekrana obrade moguće je vršiti unos novih podataka, pregled podataka i štampu određenih izveštaja.

Ekran – Nastavne aktivnosti:

- Deo za unos podataka se sastoji iz tri dela: deo za upis časa, deo za unos izostanaka i deo za unos ocena učenika, čime je potpuno pokriven proces održavanja školskog časa u realnom sistemu. Nakon unosa podataka o samom času, nastavnik po potrebi vrši unos izostanaka i/ili ocena učenika. U slučaju pogrešnog upisa izostanka ili ocene moguće je brisanje istog/iste odabirom opcije «Briši izostanak», odnosno «Briši ocenu», selektovanjem pogrešno unetog podatka i potvrdom oodgovarajućim tasterom. Nakon završetka svih neophodnih unosa, vrši se zaključivanje časa pritiskom na taster «Zaključivanje časa» čime se omogućava unos sledećeg časa. Postoji i opcija za odustajanje od zaključivanja časa.
- Deo za pregled podataka zahteva biranje između opcije pregleda ocena učenika i pregleda održanih časova. Nakon izbora razreda, odeljenja, učenika i eventualno predmeta pritiskom na određene tastera prikazuju se zahtevani podaci. Za pregled ocena dobija se i vizuelni prikaz procentualne zastupljenosti ocena.
- Deo za štampanje omogućava štampanje spiskova ocena učenika i njihovih izostanaka. Nakon izbora traženih parametara, vrši se generisanje izveštaja koji se pre štampe moze i pogledati.

Ekran izveštaja omogućava izbor parametara potrebnih za generisanje izveštaja. Nakon izbora parametara odabirom odgovarajućeg tastera za štampu vrši se štampa traženog izveštaja. Odabirom tastera pregled pre štampe moguće je pregledati generisani izveštaj pre same štampe. Od ovih akcija je moguće i odustati.

4.1.3. INFORMACIONI SISTEM PROIZVODNE ORGANIZACIJE

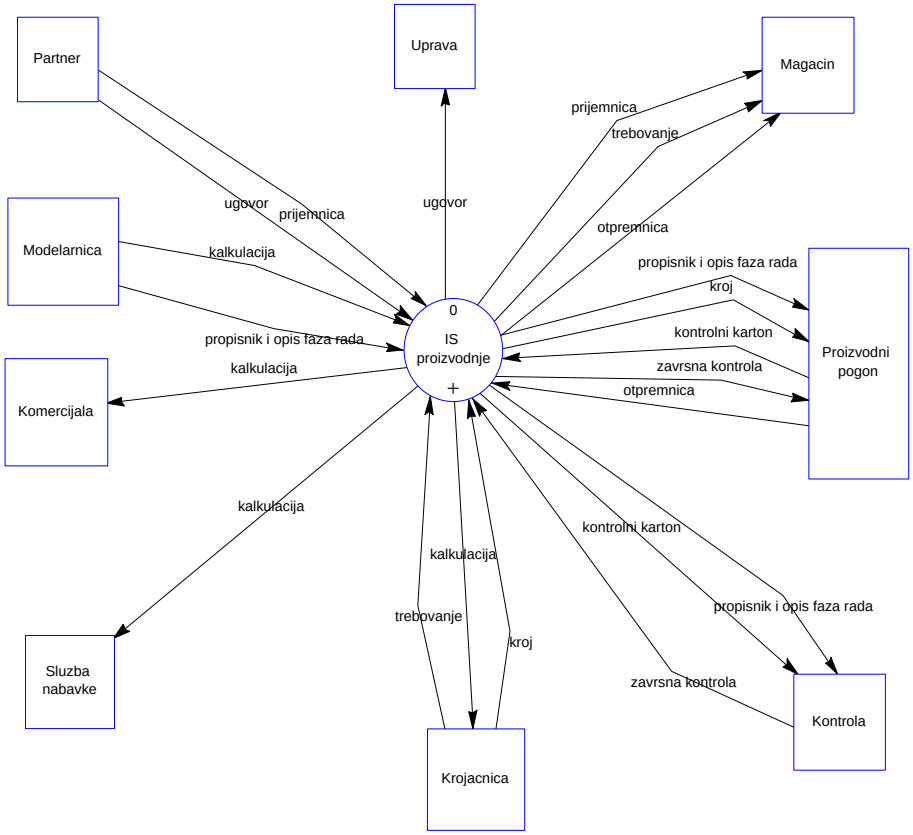
OPIS POSLA

Preduzeće za proizvodnju i promet miderske robe i kupaćih kostima "MIDERI" bavi se proizvodnjom ženskog i muškog donjeg rublja i drugih odevnih predmeta, za domaća i strana preduzeća. Proces proizvodnje u ovom preduzeću se odvija na sledeći način: Sklapa se ugovor sa domaćim ili inostranim klijentima. Ugovorom se precizira vrsta proizvoda, količina i rok isporučivanja. Materijal, koji je potreban za proizvodnju određenog proizvoda koji je specificiran ugovorom, se smešta u interni magacin na osnovu dokumenta o prijemu materijala - prijemnica. Proizvodni modeli se kreiraju u posebnom odeljenju koje se naziva modelarnica. Za svaki model se određuje utrošak materijala po jednom komadu. Dokument koji sadrži ove podatke se naziva kalkulacija i ona se izrađuje u četiri primerka. Jedan primerak kalkulacije ostaje u modelarnici, dok se ostali šalju u komercijalu, službi nabavke i obradi. Sledeći dokument koji se sastavlja u modelarnici jeste propisnik i opis faza rada. To je propratni list kojim se određuju faze rada i vreme izrade proizvoda koje je potrebno za svaku fazu proizvodnje. Ovaj dokument se izrađuje u dva primerka i oba se šalju proizvodnom odeljenju i to: jedan trakovođi i jedan međufaznoj kontroli.

Sledeća faza je krojačnica, koja na osnovu dokumenata koje dobija iz modelarnice, sastavlja krojeve za svaki model. Krojevi urađeni na taj način se šalju proizvodnji. Krojačnica koristi dokumenat koji se naziva trebovanje. U ovom dokumentu naznačava se vrsta materijala i neophodna količinu za rad i zatim se trebovanje šalje u magacin. Sledeća faza je sama proizvodnja. Trakovođa preuzima iz krojačnice iskrojene modele. Zatim, trakovođa upisuje u knjigu ulaza količinu iskrojenog materijala, kako bi raspodelila posao radnicama po fazama. Svaka radnica dobija kontrolni karton (za jedan model), u kome se navodi tačan opis posla po fazama koji svaka radnica treba da obavi. Međufazna kontrola vrši nadzor, nadgleda tok rada po fazama. Kada je model gotov, vrši se završna kontrola. Ukoliko model nema grešaka, onda se u knjigu ulaza upisuje broj gotovih modela. Svaki artikal ima svoj broj ili ime pod kojim se vodi. Ukoliko model ima grešaka, vrši se popravka i ponovo posle popravke vrši se završna kontrola. Gotovi model se šalju u interni magacin na osnovu otpremnice.

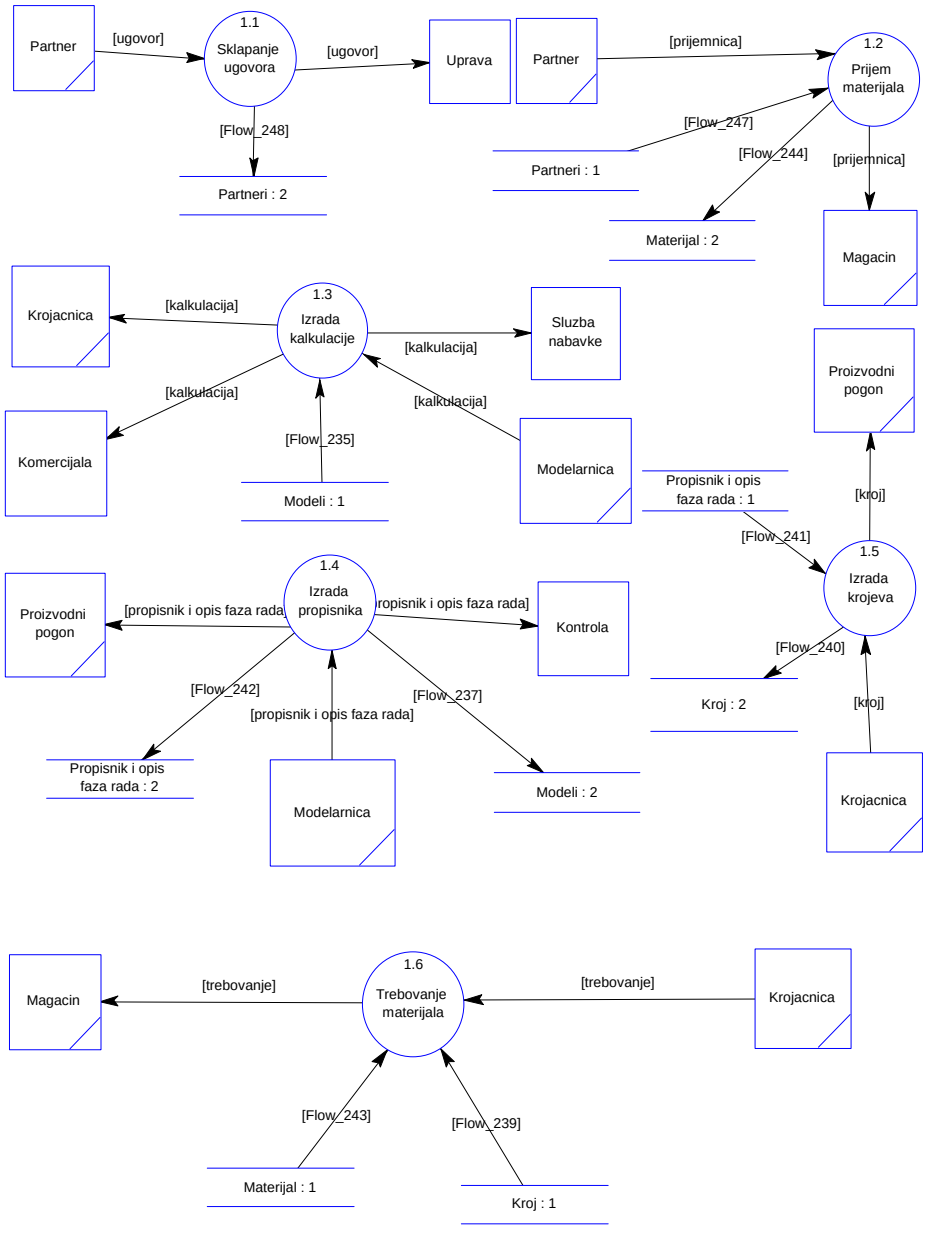
MODEL PROCESA

Dijagram konteksta



Slika 4.11. Kontekst dijagram za sistem Proizvodna organizacija

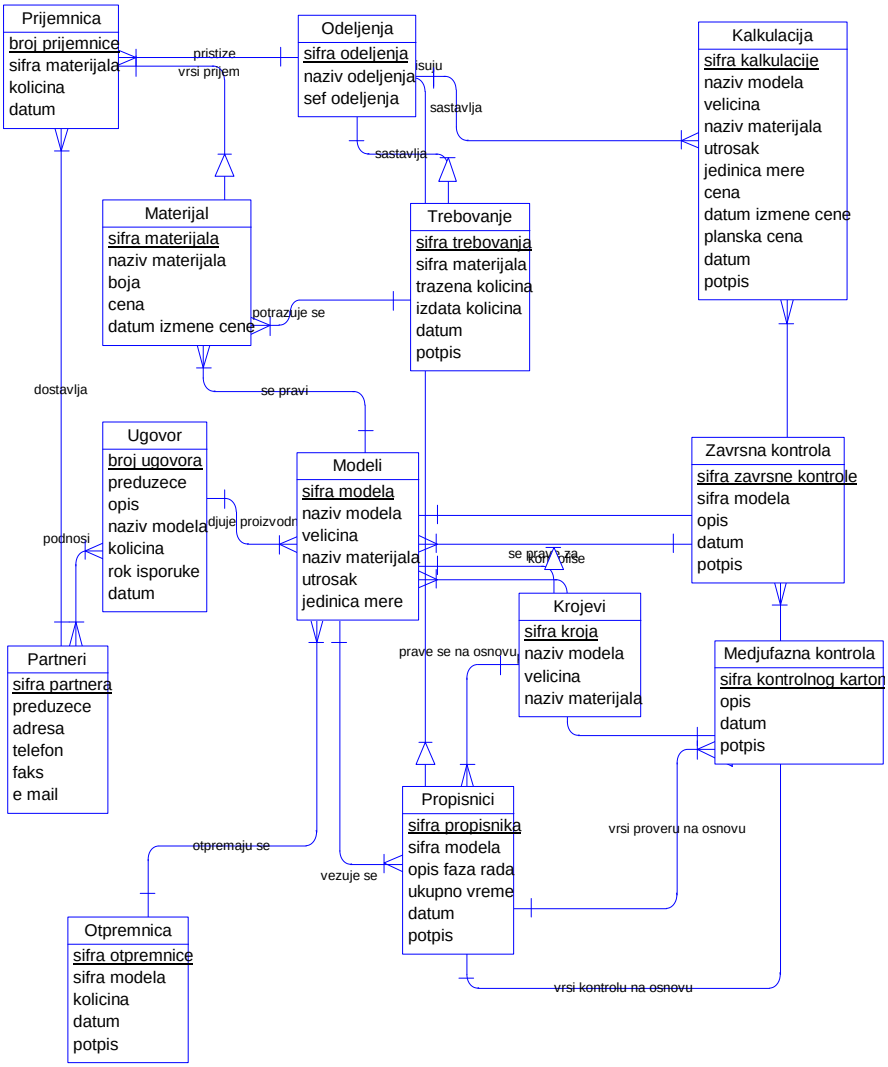
DTP 2. nivo – Priprema proizvodnje



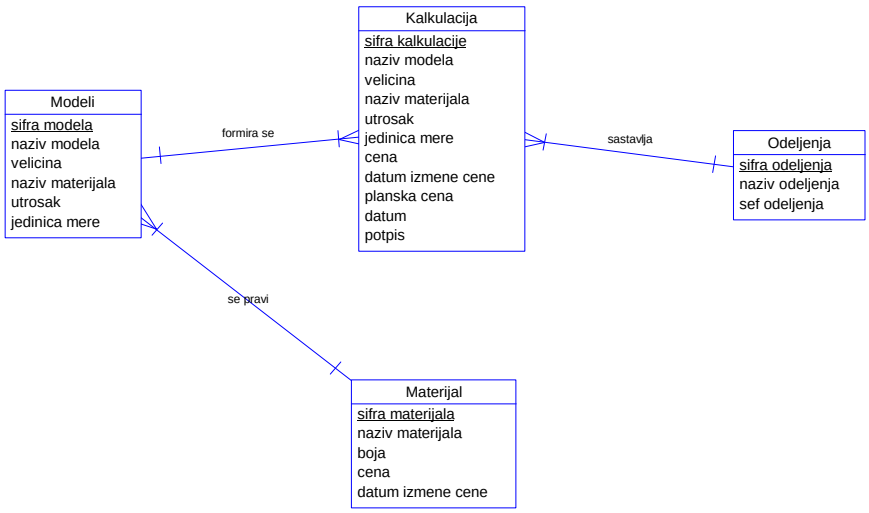
Slika 4.12. DTP 2. nivo za sistem Proizvodna organizacija

MODEL PODATAKA

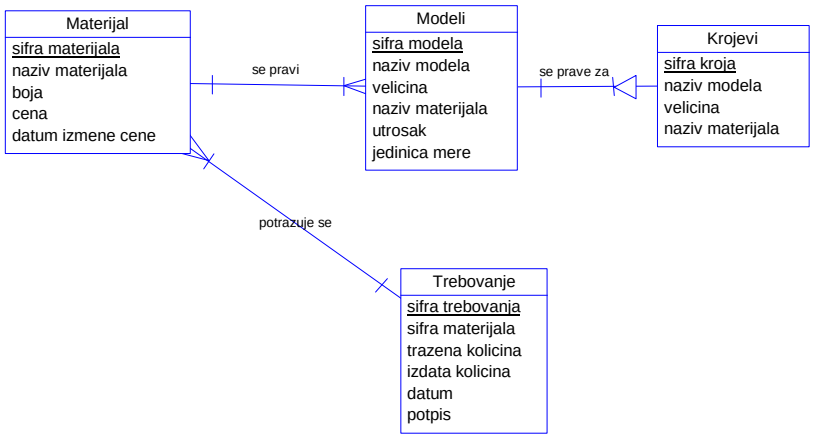
ER Dijagram (Conceptual Data Model):



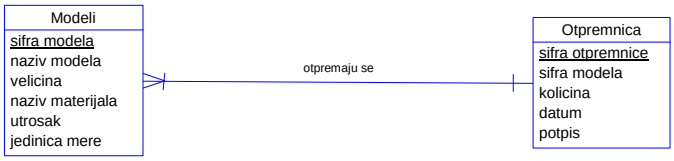
Slika 4.13. Dijagram ER modela podataka za sistem Proizvodna organizacija



Slika 4.14. Dijagram podmodela Izrada kalkulacije za sistem Proizvodna organizacija

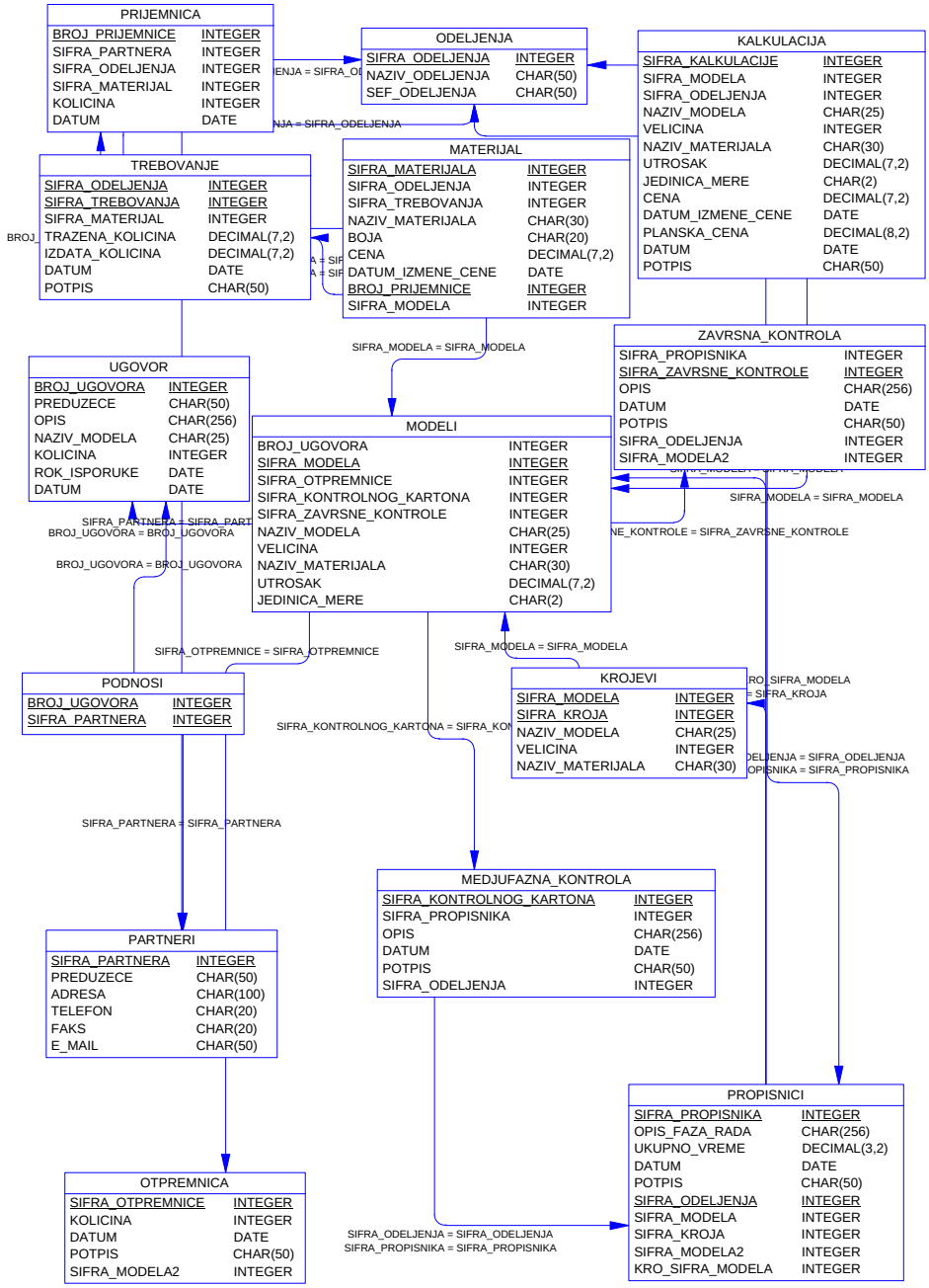


Slika 4.15. Dijagram podmodela Trebovanje materijala za sistem Proizvodna organizacija



Slika 4.16. Dijagram podmodela Skladištenje gotovih proizvoda za sistem Proizvodna organizacija

Relacioni model podataka (Physical Data Model):



Slika 4.17. Dijagram relacionog modela podataka za sistem Proizvodna organizacija

OPIS PROGRAMA

Program je moguće koristiti na PC kompatibilnim računarima sa instaliranim WIN98, WIN2000 ili WINXP operativnim sistemom. Za instalaciju i korišćenje programa na PC kompatibilnim računarima, neophodno je ispuniti sledeće hardverske zahteve: Procesor - Pentium na min 366MHz, Memorija: 128MB, Harddisk: 6.4MB, Rezolucija ekrana: 1024x768 piksela.

Struktura aplikacije se zasniva na stvarnom stanju opisa posla i opštih zahteva. Početna forma je forma za unos lozinke. Posle unosa lozinke, pojavljuje se glavna forma na kojoj se nalazi glavni meni programa sa pozivima svih dostupnih ekrana.

Slika 4.18. Softverska funkcija - Evidentiranje modela

Struktura menija izgleda ovako:

Šifarnici: Model, Kroj, Propisnik, Materijal, Odeljenja, Partneri, Izlaz.

Obrada: Proizvodni proces:

- Priprema proizvodnje
  - Sklapanje ugovora
  - Prijem materijala
  - Izrada kalkulacije

- Trebovanje materijala
- Proizvodnja
  - Medjufazna kontrola
  - Zavrsna kontrola
- Otpremanje proizvoda

Upiti: Model, Materijal, Ugovor
Servis: Pomoc, About.

Informacije o tipičnim ekranima date su u okviru help-a koji se instalira zajedno sa programom. Ekrani ″Š**ifarnici**″ služe za manipulaciju opštim podacima. Ovi podaci se uglavnom jednom unose, a višestruko se koriste na ekranima za obradu. Podaci se mogu unositi, menjati i brisati, mogu se praviti izveštaji na osnovu različitih kriterijuma (Grupni i Pojedinačni) i zatim se ti izveštaji mogu štampati. Ekrani ″**Obrade**″ **-** u ovom programu postoje sledeće grupe ekrana u okviru obrade: Proizvodni proces: Priprema proizvodnje (Sklapanje ugovora, Prijem materijala, Izrada kalkulacije i Trebovanje materijala), Proizvodnja (Medjufazna kontrola i Zavrsna kontrola) i Otpremanje proizvoda. Sa ekrane obrade omogućen je unos novih podataka, izbor traženih podataka i njihovo štampanje. Ekrani ″**Upiti**″ služe za pregled po odabranim kriterijumima. Korisnik zadaje određene parametre i dobija tabelaran pregled. U ovoj aplikaciji postoje sledeći ekrani za upite: Model, Materijal i Ugovor.

4.1.4. INFORMACIONI SISTEM SPORTSKE ORGANIZACIJE

Opis posla

Informacioni sistem fudbalskog kluba ima podsistem za vođenje evidencije i statistike igračkog kadra. Ovaj deo informacionog sistema obuhvata poslove evidentiranja igrača fudbalskog kluba, uvid u statističke podatke o aktivnostima igrača i kluba koji bi mogli biti od koristi za menadžere koji žele da se upoznaju sa karakteristikama i učinku igraca u toku sezone. Informacioni sistem se koristi za evidentiranje, arhiviranje i ažuriranje podataka o fudbalerima i utakmicama. Zadatak je da se olakša unos, uvid i arhivirajne podataka o igračkom kadru i vođenje baze podataka o igračima – članovima kluba.

Svaki fudbalski klub prilikom potpisivanja ugovora sa novim igračem mora da ima neki ustaljen proces i način sklapanja ugovora. Posle razmatranja svih pojedinosti koje se odnose na ugovor dolazi do potpisivanja ugovora ili odustajanja od saradnje. Ako dođe do saradnje klub sklapa ugovor sa igračem. Svaki taj ugovor je numerisan i pravi se u dva primerka i to jedan zadržava igrač, a drugi ostaje u evidenciji kluba. Sistem treba da omogući praćenje prisustva igrača na treninzima, kao i fizička spremnost igrača. Pre odigravanjna utakmice se vrši odabir igrača za utakmicu pomoću uvida u njihovu fizičku spremnost i prisustvo na treninzima. Posle odigrane utakmice službeno lice kluba evidentira podatke o utakmici i učinku igrača i pojedinačni parametri o igračima. Pored ovih funkcija koje su navedene ovaj informacioni sistem treba da omogući pregled rezultata kluba tokom sezone. Ali ne samo to, sistem bi trebao da omogući i uvid u pojedinačni učinak igrača u toku sezone ili učinak igrača tokom celog njegovog boravka u klubu. Što se tiče malađih kategorija igrača i

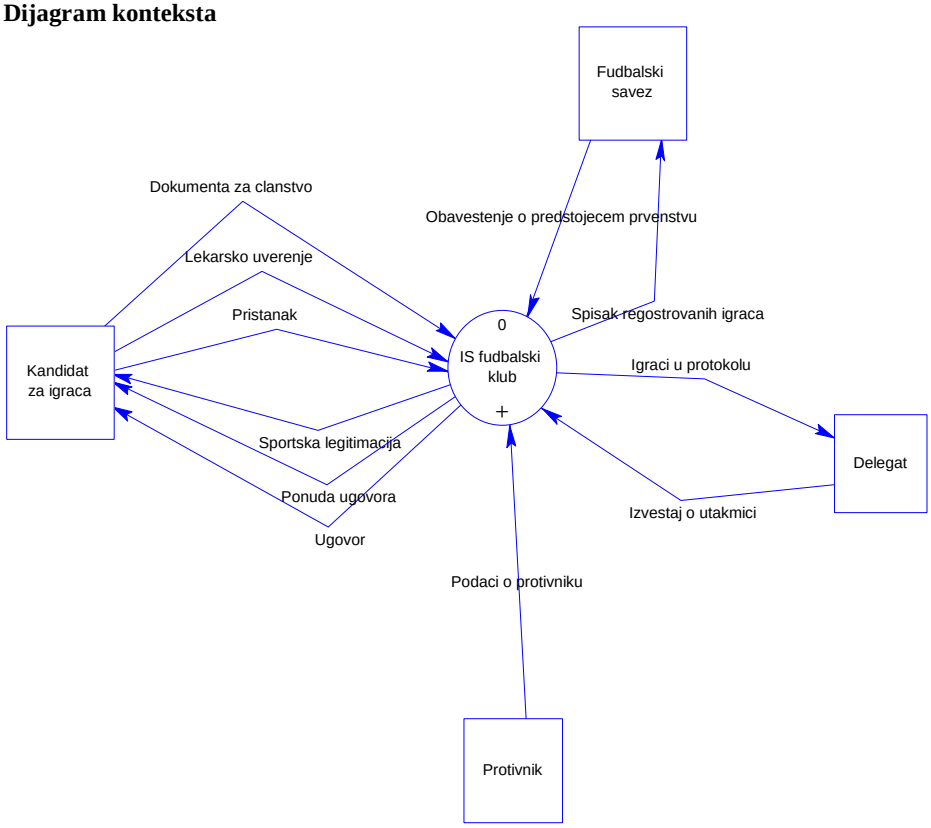
tu su predviđena mnoga olakšanja službenim licima koja se bave evidentiranjem igračkog kadra.

Snimak stanja

Dokumenti: - Svaki igrač potpisuje ugovor sa klubom koji se pravi u dva primerka i to jedan zadržava igrač, a drugi ostaje u evidenciji kluba. U okviru jedne lige nalazi se više klubova. Radna mesta su: menadžer, treneri i igrači. Zakoni i propisi kojima su definisane obaveze igrača i obaveze kluba proisticu iz zakona o sportu. Neki od igrača mogu da ne budu na raspolaganju zbog povrede ili zbog kaznenog kartona.Posle odigrane utakmice službeno lice kluba evidentira podatke o utakmici i učinku igrača. Uspešnost tj. ocenu za učinak na utakmici određuje trener. Informacioni sistem treba da omogući pregled rezultata kluba tokom sezone. Ali ne samo to, sistem bi trebao da omogući i uvid u pojedinačni učinak igrača u toku sezone ili učinak igrača tokom celog njegovog boravka u klubu.

MODEL PROCESA

Dijagram konteksta



Slika 4.19. Dijagram konteksta za sistem Sportska organizacija

Stablo procesa

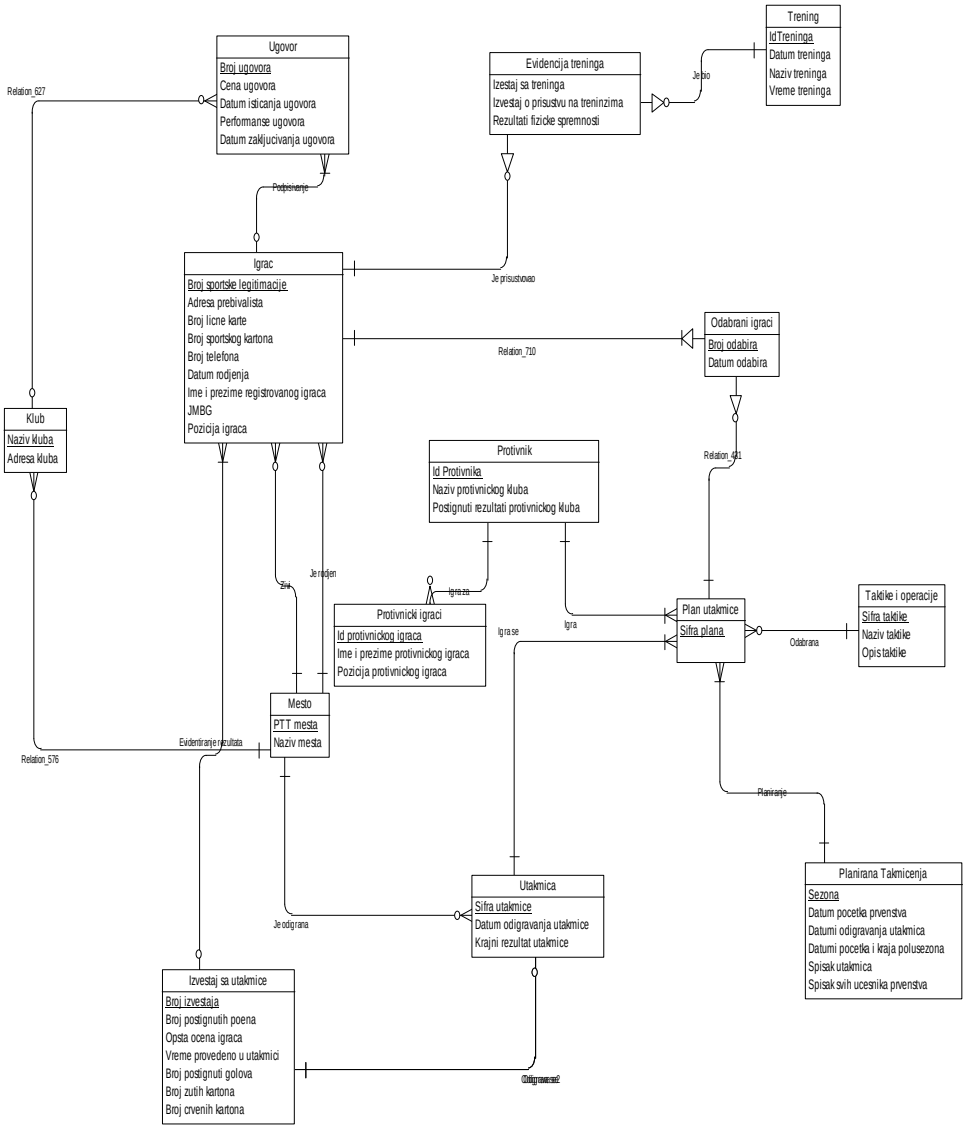
IS fudbalski klub [0]
- Uclanjivanje [1]
  - Odabir potencijalnih igraca [1.1]
  - Upisivanje u evidenciju [1.2]
- Trening [2]
- Takmicenje [3]
  - Evidentiranje najave takmicenja [3.1]
  - Evidentiranje rezultata utakmice [3.3]
  - Odabir igraca [3.2]
  - Planiranje razvoja utakmice [3.5]
  - Pracenje ucinka protivnika [3.7]
  - Rasporedjivanje igraca i primena taktike [3.6]

Struktura skladišta podataka - Evidencija igrača:

```
{
<
Adresa prebivalista igraca,
<
Broj licne karte igraca,
Broj sportske legitimacije igraca,
Broj sportskog kartona igraca,
Broj telefona igraca,
>
/
Broj ugovora,
Cena ugovora,
Datum isticanja ugovora,
/
<
Datum rodjenja igraca,
Ime i prezime registrovanog igraca,
JMBG igraca,
Mesto rodjenja igraca,
>
Naziv kluba,
Performanse ugovora,
Pozicija igraca,
Rezultati fizicke spremnosti,
>
}.
```

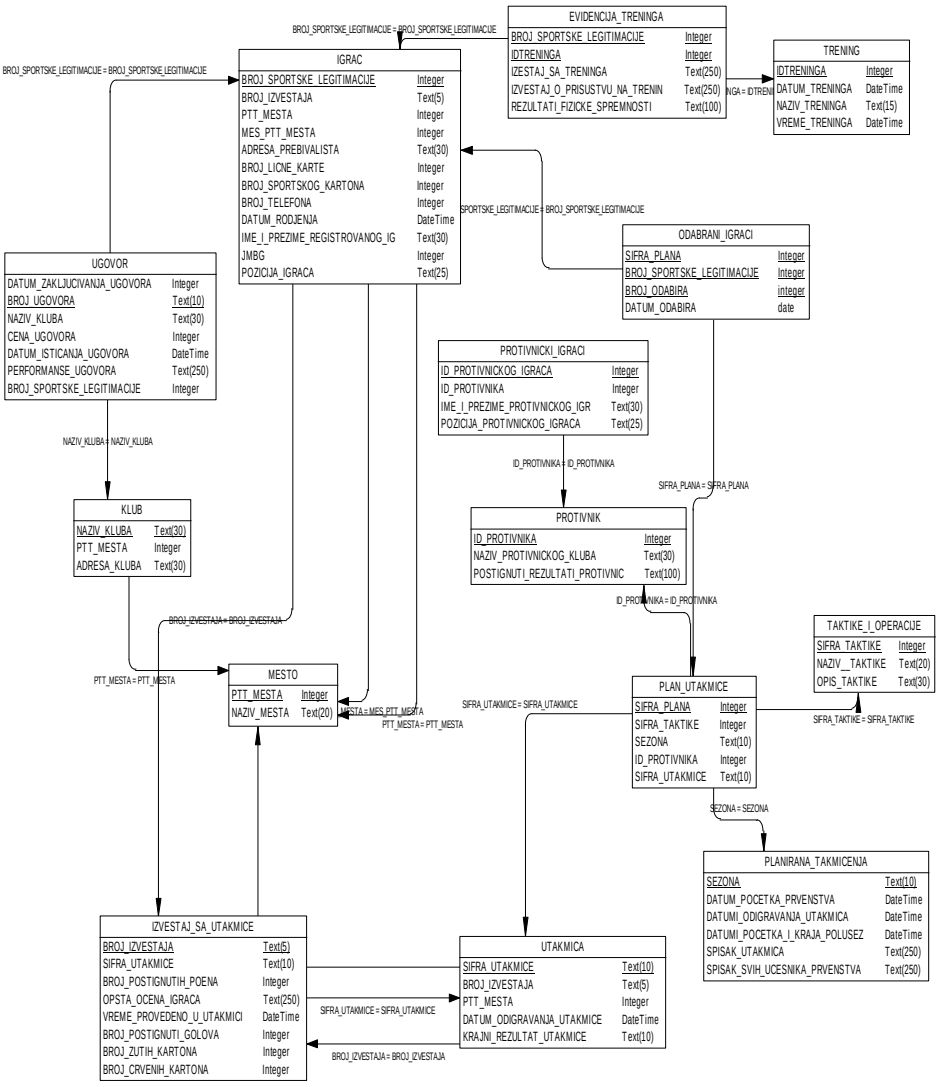
MODEL PODATAKA

ER Dijagram (Conceptual Data Model):



Slika 4.20. Dijagram ER modela podataka za sistem Sportska organizacija

Relacioni model podataka (Physical Data Model):

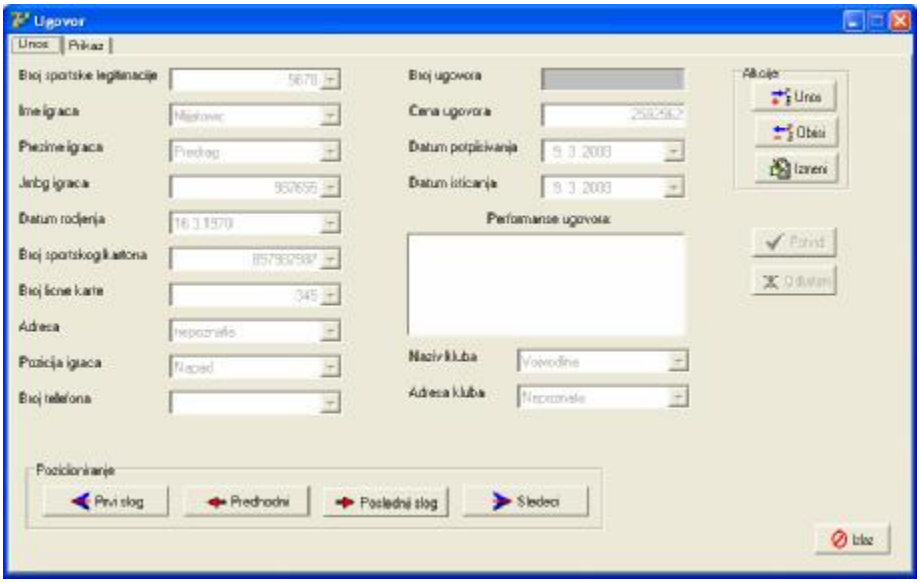


Slika 4.21. Dijagram relacionog modela podataka za sistem Sportska organizacija

OPIS PROGRAMA

Za instalaciju programa potrebno je sa CD-a ili disketa pokrenuti fajl Setup.exe koji vrši instalaciju programa u folder koji bira sam korisnik tokom instalacije. Program se pokreće sa radne povr[ine operativnog sistema (Desktop) ili pozivom iz Start\Programs menija ili pokretanjem fajla Fudbaldski klub.exe iz foldera u kom je program instaliran. Program je moguće koristiti na PC kompatibilnim računarima sa instaliranim WIN 98 (nije najpreporučljivije), WIN 2000 ili WIN XP operativnim sistemom i instaliranom podrškom za Borland DataBase Engine. Potrebna rezolucija ekrana je 1024x768 piksela u 32.000 boja minimalno. Softver, takođe, zahteva da u Windows-u budu podešeni fontovi na veličinu: Small Fonts.

Struktura aplikacije se zasniva na stvarnom stanju opisa posla i opštih zahteva. Početna forma je glavna forma na kojoj se nalazi glavni meni programa sa pozivima svih dostupnih ekrana za korisnika. Ekrani šifarnici sluze za manipulaciju opstim podacima. Ovi podaci se uglavnom jednom unose, a visestruko koriste na ekranima za obradu. Sa ovih ekrana se mogu stampati spiskovi osnovnih podataka. Ova aplikacija sadrzi pet sifarnika i to: Igrac, Mesto, Klub, Trening, Taktike.

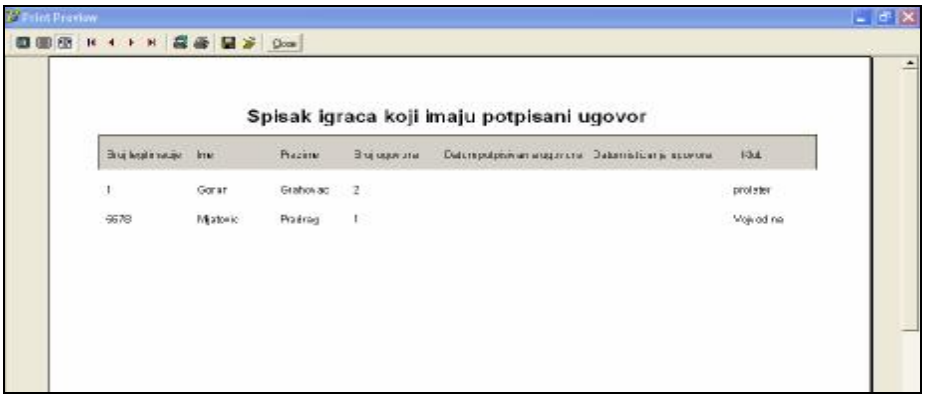


Slika 4.22. Softverska funkcija - Evidentiranje ugovora

U stavci menija Obrada imamo prikaz podataka o odredjenoj obradi podataka. U radu je uzet primer Ugovor, Evidencija treninga, Odabir igraca. Naravno, prikaz se vrsi preko dvei kartice i to su Unos i Prikaz. Unos omogucava korisniku da unese neke nove podatke, izmeni postojece i obrise neke postojece podatke, koji nisu potrebni. Podaci su dakle licni podaci o igracu , treningu, mestu, treningu, taktikama I klubu. Naravno, svi ovi podaci idu i

u prikaz - kartica Prikaz i naravno mozemo dobiti stampan izvestaj o svim podacima, kao i samo pregled tog izvestaja.

Upiti su prikazani kroz ekrane za pregled podataka po odabranim kriterijumima. Korisnik zadaje parametre i dobija tabelarni pregled podataka u donjem delu ekrana. U ovoj aplikaciji mozete izvrsiti pregled ugovora,treninga i igraca koji nemaju podpisan ugovor. Za pregled podataka po izabranim kriterijumima pritisnite odredenu opciju od ponudenih i pritisnite taster "Prikazi". Ukoliko nema trazenih podataka o tome ce korisnik biti obavesten porukom da podatak ne postoji u bazi. U suprotnom dobijamo prikaz na ekranu. Takode imamo dugme za izlazak iz prikaza.



Slika 4.23. Primer izveštaja za sistem Sportska organizacija

Ekran izvestaja omogucava izbor parametara potrebnih za generisanje izvestaja. Nakon izbora parametara odabirom odgovarajuceg tastera za stampu vrsi se stampa trazenog izvestaja. Odabirom tastera pregled pre stampe moguce je pregledati generisani izvestaj pre same stampe.

4.1.5. INFORMACIONI SISTEM MEDIJATEKE – VIDEO KLUB

OPIS POSLA

Osnovna svrha video kluba je pružanje usluge izdavanja video kaseta licima koja su članovi kluba. Vlasnik ili osoba zadužena za nabavku video kasete informiše se od distributera o novim naslovima. Vlasnik donosi odluku o izboru naslova koji će se nabaviti i broju komada (kopija). Naslovi koji se nabavljaju ne moraju da budu novi. Pri nabavci se uzima u obzir i spisak "Za nabavku" sa već postojećim naslovima, za koje su rashodovane trake. Video klub kao firma šalje narudžbenicu distributeru, a ovaj njima šalje predračun. Na osnovu predračuna video klub uplaćuje distributeru iznos sa predračuna, a distributer nakon izvršene uplate obaveštava vlasnika video kluba o terminu isporuke, nakon čega vlasnik preuzima video kasete u prostorijama distributera, pri čemu vlasnik video kluba dobija otpremnicu.

Svaki primerak nove video kasete u video klubu dobija svoj jedinstveni broj i uvodi se u evidenciju inventara (tu se beleži broj filma, naziv filma, režiser, glavni glumci, godina proizvodnje, zemlja u kojoj je snimljen, naziv distributera itd.). Kaseta se smešta na police na tačno određeno mesto prema žanru filma i abecednom redosledu naziva filma.

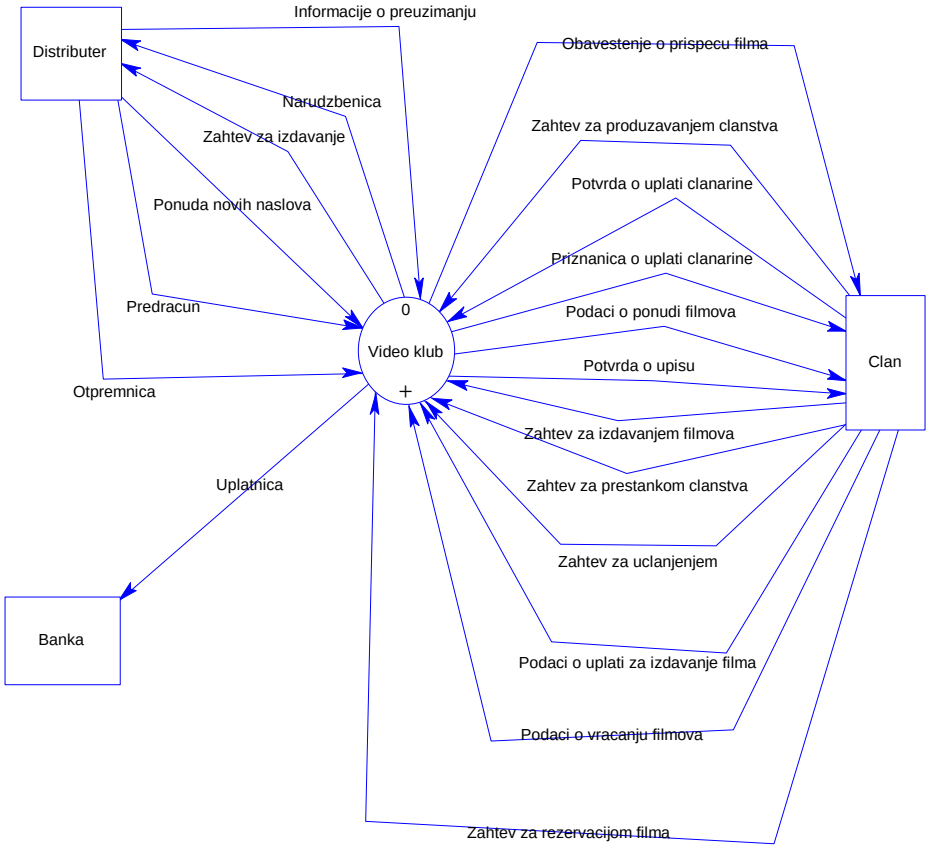
Takođe, na osnovu evidencije inventara formira se i katalog ponude filmova, pa se on ažurira prilikom svake nabavke novih naslova. Kada se kaseta izdaje, u posebnu svesku ("Evidencija izdavanja") se upisuje datum izdavanja, broj kasete, broj člana, prezime i ime člana i tu se član potpisuje (za svaki primerak kasete). Kada se kaseta vrati, evidentira se u svesci ("Evidencija izdavanja") u redu gde je evidentirano izdavanje u posebnoj koloni datum vraćanja i potpis člana koji je vratio kasetu. Kada se kaseta vrati, ne proverava se da li je traka premotana na početak ni da li je traka ispravna, već se kaseta odlaže na odgovarajuću policu. Ukoliko se kaseta vrati i član kluba ima pritužbe na kvalitet trake ili je eventualno dobio neispravnu traku, traka se preuzima i kasnije proverava, a članu se izdaje druga kopija trake sa novim datumom izdavanja (tako da mu se vreme izdavanja produžava). Ukoliko se pri proveri ustanovi da je traka neispravna ili oštećena, ona se rashoduje. U posebnoj svesci ("Rashodovane trake") se evidentira datum provere kvaliteta trake, broj video kasete, broj clana koji je poslednji koristio traku. U ovom slučaju cemo zanemariti situaciju u kojoj traku treba da obešteti poslednji korisnik trake. Istovremeno se evidentira na spisku ("Za nabavku") da treba nabaviti još jednu kopiju filma čija se traka rashoduje. Takođe, iz evidencije inventara se briše (precrta) video kaseta sa datim brojem, jer više nije u upotrebi. Da bi građanin postao član video kluba, potrebno je da prilikom učlanjenja prikaže ličnu kartu i uplati članarinu. Tada se članu dodeljuje jedinstveni broj člana i evidentiraju lični podaci člana u posebnoj svesci ("Evidencija članova").

Član prilikom učlanjenja dobija karticu sa svojim brojem. Prilikom svakog uplaćivanja radnik video kluba je dužan da članu izda priznanicu za uplaćenu sumu novca. Svako dalje plaćanje članarine obavlja se na godišnjem nivou. Prilikom prvog usluživanja u nekoj kalendarskoj godini, član je dužan da osim vrednosti koju plaća za izdavanje filmova plati i članarinu za tu godinu. Građanin prestaje da bude član video kluba ukoliko nije uplatio članarinu i potvrdio članstvo za odgovarajuću kalendarsku godinu. Gradjanin je dužan da vrati filmove u roku od 15 dana od prestanka članstva. Član video kluba nakon razmatranja kataloga odlučuje se za naslove. Ukoliko ne postoji ni jedna kopija slobodna kopija tog naslova, član kluba se odlučuje da rezerviše film čime se na listu čekanja upisuje broj člana, broj filma i datum rezervisanja. Ukoliko je bar jedna kopija filma vraćena, radnik video kluba je dužan da obavesti člana o prispeću filma.

Član na osnovu neposrednog izbora ili rezervacije filma preuzima video kasetu 1 ili više, i pri tome se evidentiraju u posebnoj svesci ("Evidencija izdavanja") odgovarajući podaci. Član se obavezuje da će kasete čuvati, da će prilikom vraćanja premotati trake na početak i da neće držati video kasete duže od 10 dana. Prilikom vraćanja filmova, radnik naplaćuje od člana vrednost iznajmljivanja jedne kasete za jedan dan, pomnoženo sa brojem kaseta i brojem dana koliko je član zadržao kasete. Ukoliko je član zadržao kasete duže od 10 dana, član je dužan da plati kaznu.

MODEL PROCESA

Dijagram konteksta

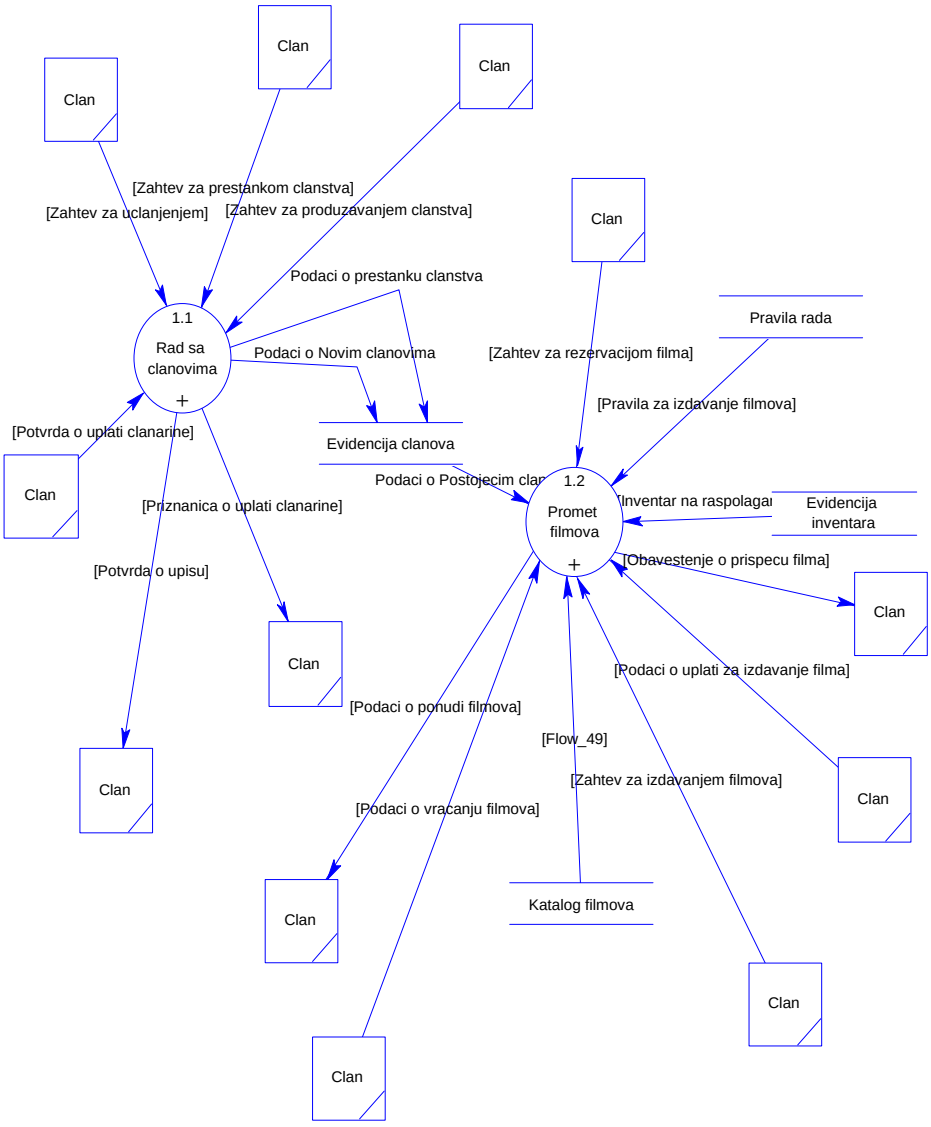


Slika 4.24. Dijagram konteksta za sistem Video klub

Stablo procesa

0. VIDEO KLUB
    1. OSNOVNA DELATNOST
        1.1. Rad sa clanovima
            1.1.1. Uclanjenje
                1.1.1.1. Upis novih clanova
                1.1.1.2. Produzavanje clanstva
                1.1.1.3. Prestanak clanstva
            1.1.2. Uplata clanarine
        1.2. Promet filmova
            1.2.1. Informisanje o ponudi filmova
            1.2.2. Izdavanje
            1.2.3. Vracanje
                1.2.3.1. Redovno vracanje
                1.2.3.2. Reklamacije
                1.2.3.3. Obrada zakasnjenja u vracanju
            1.2.4. Uplata izdavanja
                1.2.4.1. Redovna uplata
                1.2.4.2. Uplata kazni
            1.2.5. Rezervisanje
                1.2.5.1. Evidencija rezervacije
                1.2.5.2. Obavestavanje o prispeću
    2. UPRAVLJACKA DELATNOST
        2.1. Odredjivanje pravila rada
        2.2. Odlucivanje o nabavci
        2.3. Odlucivanje o rashodovanju
    3. PRATECA DELATNOST
        3.1. Nabavka filmova
            3.1.1. Informisanje o novim naslovima
            3.1.2. Izbor naslova za nabavku
            3.1.3. Narucivanje
            3.1.4. Placanje
            3.1.5. Preuzimanje
            3.1.6. Uvodjenje u inventar
            3.1.7. Kreiranje kataloga
        3.2. Kontrola kvaliteta kopija
            3.2.1. Provera ispravnosti
        3.2.2. Rashodovanje

DTP 2. nivo - Osnovna delatnost



Slika 4.25. DTP 2. nivo za sistem Video klub

Deo rečnika podataka - primer strukture toka podataka "Narudžbenica":

```
Narudzbenica:
<
 <
  naziv_kluba,
  ulica,
  broj_stana,
  Naziv_mesta,
 >
  datum_narucivanja,
      {
      <
      rb_stavke,
      naziv_filma,
      broj_kopija,
      kataloski_broj,
      >
      }
   izbor_nacina_placanja,
  <
   odgovorno_lice_prezime,
   odgovorno_lice_ime
  >
>.
```

ELEMENTARNI PODACI:

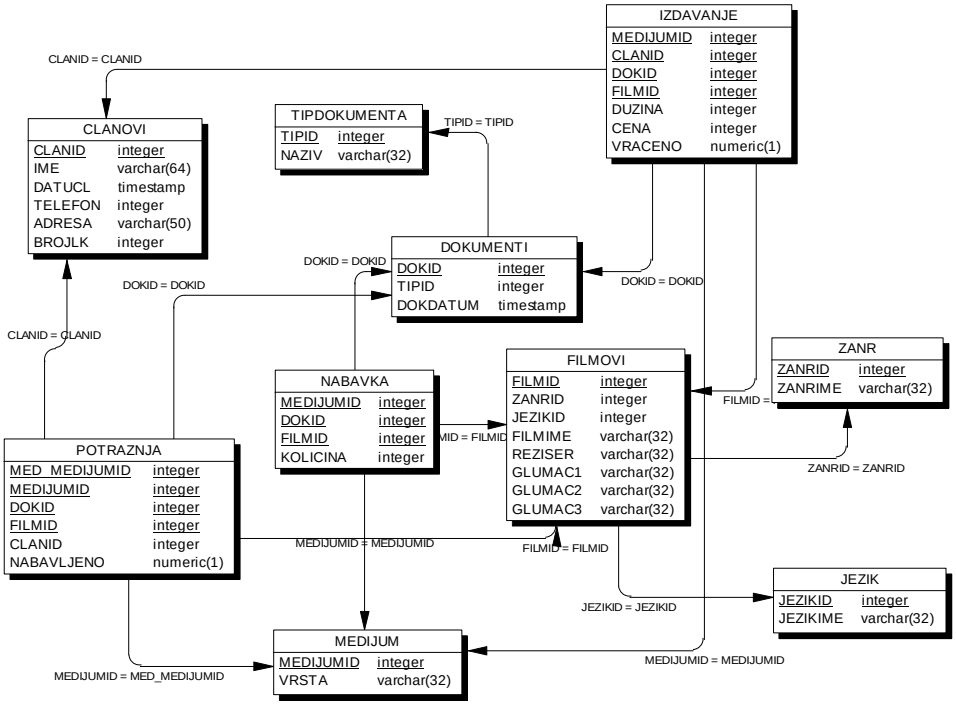
```
      Naziv_kluba: VarChar 30
      Ulica: VarChar 50
      Broj_stana:  Integer
      Naziv_mesta: VarChar 50
      Datum_narucivanja: Date
      RB_stavke: Integer
      Naziv_filma: VarChar 100
      Broj_kopija: Integer
      Kataloski broj: VarChar 20
      Izbor_nacina_placanja: NACIN PLACANJA
      Odgovorno_lice_prezime: VarChar 40
      Odgovorno_lice_ime: VarChar 30
```

DOMENI:

```
      Nacin_placanja: VarChar 20,
      Check: {gotovinski, ziro racun}
```

MODEL PODATAKA

Relacioni model podataka (Physical Data Model):



Slika 4.26. Dijagram relacionog modela podataka za sistem Video klub

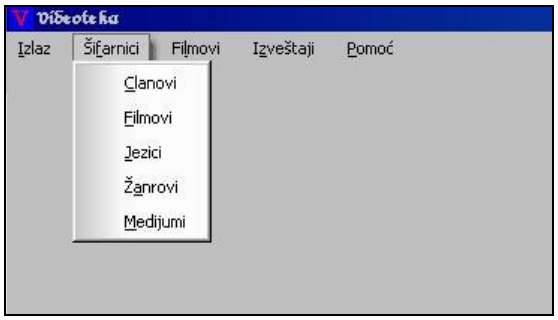
OPIS PROGRAMA

Instalacija - Instalacija programa se vrši na sledeći način: kreiramo direktorijum *Database* na disku c:\Database. Sa CD-a kopiramo fajl *Videoteka.GDB* (direktorijum Videoteka) u direktorijum c:\Database. Sa CD-a kopiramo fajl V*ideoteka.EXE* na proizvoljno mesto odakle želimo da startujemo program.

Korišćenje - Nakon startovanja aplikacije pojavljuje se meni prikazan na sledećoj slici. Opcije uvodnog ekrana realizuju sledeće funkcije:

- **Izlaz -** Završetak rada sa aplikacijom, zatvara se aplikacija i baza.
- **Šifarnici** – Ova opcija omogućava unosa, modifikacije i brisanja podataka vezanih za članove, filmove, jezike, žanrove i medijume.
- **Filmovi** – Ovom opcijom se otvara novi podmeni kojim se realizuju funkcije prijema, otpisivanja, iznajmljivanja, razduživanja i naručivanja filmova.
- **Izveštaji** – Otvara se podmeni za izbor tipa izveštaja kao što su: stanje filmova, izdati filmovi, istorija, otpisani filmovi i naručeni filmovi.

- **Pomoć** - Ovaj deo menija sadrži korisničko uputstvo i podatke o autorima.



Slika 4.27. Glavni meni softvera - Video klub

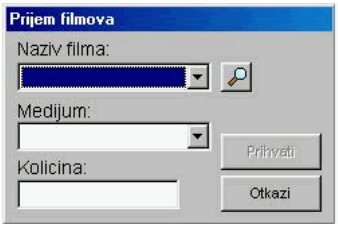
Opcije menija **Šifarnici** realizuju sledeće fukcije:

- **Članovi** - Opcija daje mogućnost učlanjenja novog člana u videoteku ako su nam poznati ime i prezime i broj lične karte, izmenu postojećih podataka, brisanje postojećih podataka i mogućnost odustajanja od bilo kakve radnje.
- **Filmovi** - Opcija daje mogućnost unosa novog filma gde su obavezni podaci o nazivu filma i jezik, mogućnost izmene podataka o već postojećem filmu i brisanje postojećih podataka.
- **Jezici** - Opcija daje mogućnost unosa i brisanja jezika.
- **Žanrovi** - Opcija daje mogućnost unosa i brisanja žanrova.
- **Medijumi**- Opcija daje mogućnost unosa i brisanja medijuma.

U okviru podmenija **Članovi** moguće je **upisivanje novog člana** - dugme Dodaj otvara prozor za upisivanje novog člana. Polja koja se obavezno moraju popuniti da bi opcija Prihvati bila dostupna su Ime i prezime člana i Broj L.K. Adresa i telefon se unose ukoliko su poznati, a ukoliko nisu mogu se ostaviti prazna polja. U polju Broj L.K. nije dozvoljeno upisivanje ovog podatka ukoliko takav već postoji (ako se pokuša upisati takav podatak pojaviće se prozor sa obaveštenjem o grešci). Ako su podaci korektno uneti dugme Prihvati je dostupno, što znači da će se pritiskom na taster ENTER ili na dugme Prihvati izvršiti upis novog člana. Ukoliko smo neki od dva obavezna polja pogrešno uneli ili tokom unosa podataka o članu odustali na raspolaganju nam je opcija Otkaži. **Izmenu** već postojećih podataka je moguće izvršiti tako što označimo polje koje želimo da izmenimo u tabeli Sifarnik članova klikom miša na njega, a potom aktiviramo dugme Izmeni u istoj tabeli .Na taj način moguće je vršiti izmene svih matičnih podateka (ime i prezime, adresa, telefon, broj l.k.) osim rednog broja koji program sam generiše. Brisanje člana iz evidencije je preko dugmeta Izbriši. **Brisanje člana** iz evidencije videoteke je moguće jedino ako nije (ili nije bio) zadužen ni sa jednim filmom,a do člana se dolazi jednostavnim klikom na člana u tabeli Šifarnici članova. Odustajanje od bilo kakve radnje je dugme Otkaži. **Pretraživanje** otvara prozor za unos teksta koji želimo da pronađemo u tabeli. Da bi tražena reč bila pronađena dovoljno je uneti barem jedno poznato slovo date reči, deo reči ili celu reč. Pretraživanje tabele Šifarnik članova moguće je izvršiti po imenu i prezimenu člana, adresi, broju lične karte, telefonu i rednom broju člana.

U okviru podmenija **Filmovi** moguće je :

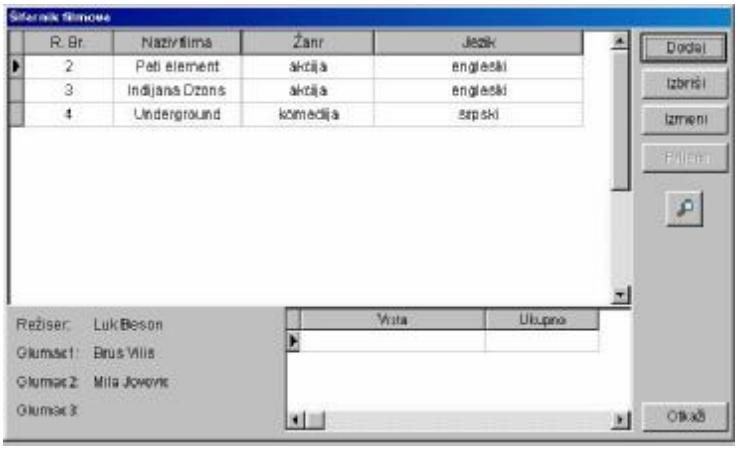
1. Izvršiti dodavanje novog filma - dugme Dodaj: Dodavanje filma je moguće je izvršiti ukoliko su nam poznati naziv filma i jezik, dok su ostala polja neobavezna za popunjavanje (žanr, glumac1, glumac2, glumac3, režiser). Ukoliko smo korektno popunili obavezna polja dugme Prihvati će biti dostupno. Ako smo nešto pogrešno uneli ili jednostavno želimo da odustanemo od unosa novog filma potrebno je kliknuti na dugme Otkaži.
2. Mogućnost izmene postojećih podataka o nazivu filmu - dugme Izmeni. Do naziva filma se dolazi na isti način kao i do člana, nakon čega se otkuca novi naziv filma i jednostavnim klikom na taster ENTER navedena izmena će biti zabeležena u tabeli.
3. Mogućnost brisanja postojećeg filma - dugme Izbriši: Dovoljno je u tabeli odabrati film koji se briše i kliknuti na taster Izbriši, nakon čega selektovani film nestaje se svim svojim podacima iz tabele Šifarnik filmova. PAŽNJA:Brisanje filma koji je izdat ili je to bio nije dozvoljeno. Ukoliko pokušamo da izbrišemo ovakav film pojaviće se upozorenje 'Brisanje nije dozvoljeno'.
4. Dugme sa nazivom Prijem omogućava unos količine odgovarajućeg medijuma filma koji je evidentiran u tabeli Šifarnik filmova.



Slika 4.28. Ekran za Prijem filmova u Video klubu

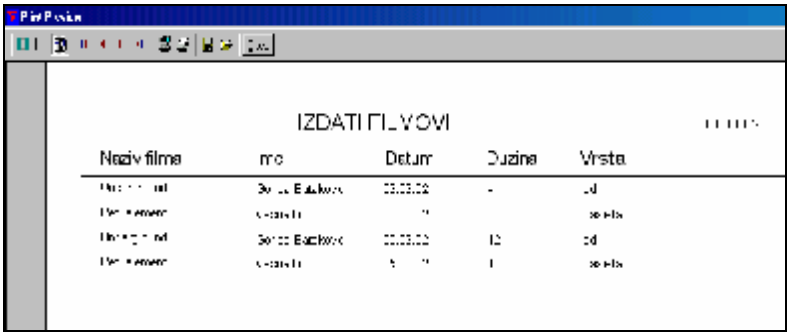
5. Odustajanja od bilo kakve radnje – kliknuti na dugme Otkaži.

6. Pretraživanje je moguće po nazivu filma, režiseru, glumcima, jeziku i žanru.



Slika 4.29. Šifarnik filmova Video kluba

Na sledećoj slici je prikazan izveštaj (prikaz Pregled pre štampe) koji omogućuje štampanje podataka o izdatim filmovima za određeni dan.



Slika 4.30. Izveštaj - Izdati filmovi video kluba

4.2. PRIMERI DIZAJNA KORISNIČKOG INTERFEJSA

U Visual Basic-u elementi korisničkog interfejsa se nazivaju kontrole (control). U grafičkom okruženju, osnovni elementi korisničkog interfejsa su (terminologija VB-a): radna površina (desktop), horizontalni meni, vertikalni meni, statusna linija, prozor, pozicione trake (scroll bar), softverski tasteri (komandni (Command), opcijski (Option), za podešavanje (Check, Toggle)), Okviri (za poruke - Message box, za reakciju - Message box, za unos - Input box, sa istorijom interakcije - history box, za spiskove - list box, za promenu direktorijuma - directory tree), Text box (linija unosa teksta), Combo box, List box, Grid, Tab Strip, Tree View, Timer, Picture box... i drugi. Generalno, elementi se mogu podeliti na kontejnere (oni koji sadrže druge) i jednostavne elemente.

U radu sa bazama podataka, najčešće se koriste sledeći elementi korisničkog interfejsa (VB): element za pristup bazi podataka (Data control), Combo Box sa listom koja čita podatke iz baze podataka, a pri izboru upisuje u bazu podataka (DBCombo), List Box sa vezom sa bazom podataka (DBList), DBgrid ili DataGrid (za tabelarno prikazivanje podataka iz baze podataka), Crystal Report kontrola (za omogućavanje veze sa izveštajima).

Podešavanja osobina Form objekata: font, pokazivač miša, pozicija forme, kreiranje, učitavanje i iščitavanje formi, aktiviranje i osvežavanje formi, postavljanje osobina na inicajlne (default) vrednosti nakon završetka rada sa nekom formom, redosled aktiviranja elemenata korisničkog interfejsa (Tab, Enter, strelica na dole), vizuelno predstavljanje aktivnog elementa (promena boje kada je u fokusu), vidljivost/nevidljivost pojedinih elemenata KI (npr. tastera).

Tipovi formi prema međuzavisnosti ponašanja:

1. MDI (multiple document interface) i MDI-child forme,
2. Modalne forme,
3. Obične SDI (single document interface).

4.2.1. VRSTE FORMI I ELEMENTI REALIZACIJE

1. Forma za podešavanje rada programa "Options"

Služi: za podešavanje ponašanja programa (font, pojavljivanje poruka, statusna linija i sl.). Elementi: TabStrip, PictureBox, Frame, ChechBox, OptionButton, Komandni tasteri.

2. Forma sa informacijama o programu i autorima "About"

Služi: za obaveštavanje o autoru i sl. Sadrži podatke:
1. Naziv aplikacije
2. Verzija
3. Platforma – 16, 32, 64 bit Windows okruženje
4. Copyright
5. Licenced to & Serial Number
6. Upozorenje o zaštićenosti zakonima
7. System info

Slika 4.31. Forma sa informacijama o programu

Elementi realizacije: PictureBox, labele, kommandni taster, linija.

3. MDI forma

Služi: sadrži druge forme, sadrži horizontalni padajući meni aplikacije. Elementi: meni, pozadinska slika. Tehnike: Kreiranje MDI forme, a ostale su MDI child=true. Modalne i nemodalne forme je nešto drugo. Modalna forma znači da korisnik mora nešto da uradi pre nego napusti tu formu, a nemodalna da ne mora. Kreiranje menija ("&" ispred znaka znači alt+taj znak->shortcut, - tj. minus je caption kada se želi odvojiti delove padajućeg menija linijom. Isto važi i za komandne tastere: Prvo slovo je veliko, ostala mala, ispred slova koje je shortcut (sa alt) se stavlja znak "&" i ono automatski biva podvučeno).

Slika 4.32. MDI forma

4. Forma za meni sa komandnim tasterima

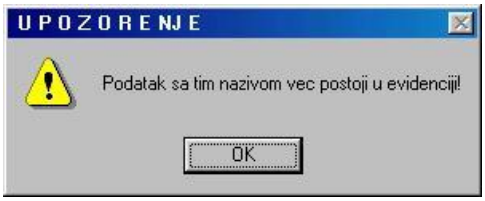
Služi: alternativno je za padajuće menije. Bolje je padajuće menije koristiti jer se odmah vidi struktura celog programa, a i svojevrstan je standard.
Elementi: komandni tasteri sa skraćenicama

Slika 4.33. Forma menija sa tasterima

Tehnike: postavljanje fokusa, promena boje, grafički tasteri, postavljanje skraćenica na tastere.

5. Upozorenja, obaveštenja

Služi: za alarme da je nešto pogrešno urađeno, za obaveštenja npr. o štampaču i upozorenja npr. da se naglasi da je nešto takvo i takvo, ili da treba nešto uraditi na neki način.
Elementi: labela, komandni tasteri,(option button kod ekrana za kraj)

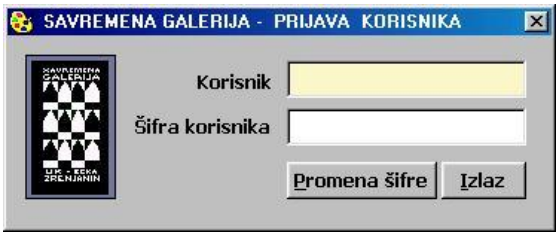


Slika 4.34. Forma sa upozorenjima i porukama korisniku

Tehnike: isticanje bojom, zvukom i blinkovanjem; javljaju se nakon što je izvršena neka akcija ili pre neke akcije-> provera uslova i nakon uslova treba prikazati poruku; koristi se MSGBOX, INPUTBOX ili obična forma.

6. Forma za prijavljivanje korisnika

Služi: za unos korisničkog imena i šifre, po kojoj se tome korisniku dodeljuju određena prava za korišćenje programa.
Elementi: labele, TextBox, komandni tasteri.



Slika 4.35. Forma za prijavu korisnika

Tehnike: korišćenje niza labela (labela(0), labela(1) itd.), korišćenje globalnih promenjivih (na nivou modula (forme)->Public, korišćenje api funkcije za očitavanje username, Korišćenje Space$ funkcije za pravljenje praznog stringa od fiksnog broja karaktera, Len, Left$, SetFocus na TextBox, Hide, unošenje šifre kao niz zvezdica (password char), Automatsko selektovanje teksta (Textbox1.Selstart=0, Textbox1.SetLength= Len(Textbox1.text)).

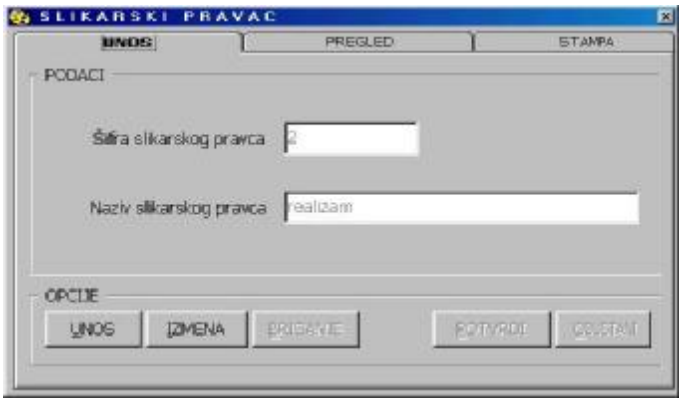
-> odakle se čita šifra: iz baze, pa se poredi sa bilokojim od korisnika u bazi, ili sa šifrom u .exe (kodu), ili sa šifrom u registry, ili se smesti na disketu, pa samo dok je inicijalna disketa u drajvu može da se koristi program.

-> da li će postojati Cancel dugme zavisi od toga da li postoji neki skup aktivnosti koje se mogu sprovoditi nad programom bezopasno.

7. Forma za ažuriranje sadržaja neke tabele ili pogleda (1 tabela ili više, 1:M unos)

Služi: za ažuriranje tabela iz baze, što znači unos (dodavanje novog sloga), brisanje (postojećeg sloga) i izmena podataka (postojećeg sloga).

Elementi: labele, text box, Combo, DBCombo, List, DBList, Option, Check, Komandni tasteri.



Slika 4.36. Forma za ažuriranje podataka

Tehnike:

- Pre svega, razlikuju se forme za ažuriranje matičnih podataka i tzv. korisničke forme koje služe osnovnoj svrsi informacionog sistema: za evidenciju računa, prijemnice, otpremnice i sl. jer korisničke forme koriste matične podatke u svojim DBCombovima.

- Za brisanje i izmenu prvo pronaći traženi slog, pa ga zatim izmeniti/obrisati. (koristiti bookmark ili šifru)

- Prilikom brisanja voditi računa da li je kaskadno brisanje podešeno u bazi ili nije-da li postaviti pitanje korisniku ili automatski izvršiti aktivnost. U svakom slučaju, za sve akcije je potrebno da se korisnik obavesti.

- Prilikom unosa i izmene šifre, voditi računa da ne postoji već slog sa istom šifrom u tabeli (kontrola u kodu ili prepustiti da SUBP tu reaguje).

- Kontrolisati vrednosti polja (domen) već pri unosu toga (svakog) polja, recimo pri napuštanju tog polja i prelasku na sledeće ili u toku kucanja (unosa), a ne na kraju kada se potvrđuje unos da se prijave sve greške!

- Pri kontroli kod neispravnog unosa ne sme da se prekine izvršavanje programa, već je potrebno obavestiti korisnika o grešci, postaviti fokus na pogrešno mesto (oboji drukčije, alarmira zvukom!).

- Kod unosa novog sloga, nakon potvrde unosa treba da je poslednje uneti slog zapravo i onaj koji se po potvrđivanju vidi kao tekući. (koristiti Bookmark=Lastmodified).

- Zgodno je prebacivati se sa kontrole na kontrolu sa tab-om, a pri unosu sa ENTER, da redosled bude u skladu sa redosledom unošenja, da bi se što brže unosilo, bez gledanja!!!

- Zgodno je da kontrola koja ima fokus bude drugačije obojena.

- Lepo je i korisno da se pri unošenju nove šifre matičnog podatke ovaj upiše automatski u text box, sa mogućnošću izmene, a da to bude automatski prvi sledeći nezauzeti broj

(poslednji po veličini +1 ili može i manji, koji nije bio zauzet. To može da bira korisnik, a ako ima takvih nejasnih situacija, svakako da se obavesti).

- Potrebno je umesto text boxova na korisničkim tabelama (NE matičnim podacima) koristiti Combo ili DBCombo boxove gde korisnik bira ime, a ne unosi (i pamti) šifru koja se zapravo unosi u korisničku tabelu.

- Potrebno je ukoliko, nedostaje neki podatak u kombo boxu, dodati ga u matičnim podacima, a zatim se vratiti na ovu formu i osvežiti je da bi se ponovo napunila lista combo boxa i da bi se ovaj novi podatak mogao iskoristiti.Zato je potrebno da su sve forme MDIchild prema nekoj MDI koja objedinjuje celu aplikaciju. Na taj način je moguće pozivati više formi odjednom.

- Potrebno je podatke koji se odnose kao 1:M (zaglavlje dokumenta i stavke dokumenta) unositi na jednoj formi. Pri tome treba omogućiti da se ova strana M više puta unosi, bez unošenja zaglavlja svaki put (jednom se unese, zadrži na ekranu i podrazumeva se da se unos na strani M odnosi na to zaglavlje.

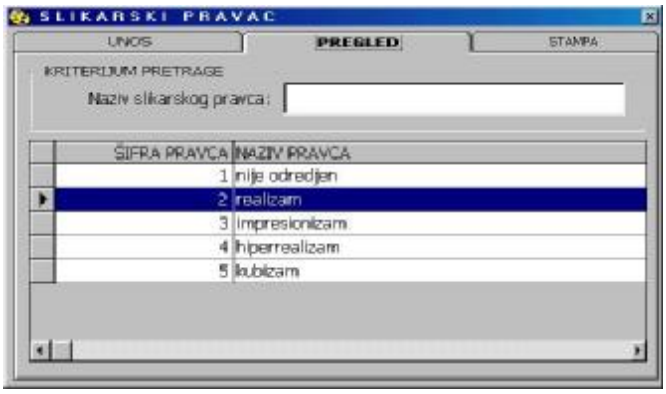
1. direktno tabela baze vezana na tekst boxove(baza proverava ispravnost za svaki tekst box i daje upozorenja-> prekida program sa radom): Text boxovi su vezani za Data kontrolu i odgovarajuće polje tabele
2. a) nije tabela vezana za tekst boxove, nego oni služe za unos vrednosti, a pritiskom na taster "Potvrdi" se dešava provera ispravnosti(poruke) i unos/izmena;
   b) nije tabela vezana za tekst boxove, nego oni služe za unos vrednosti, ali se nakon svakog unosa(svakog tastera) proverava da li je to polje ispravno uneto.
3. SQL upitom se ažurira tabela baze (briše, dodaje, menja).

Tehnike dodavanja sloga, editovanje, osvežavanje tabele, kretanje kroz tabelu (prvi, sledeći, zadnji), poništavanje akcije dodavanja, izmene, brisanja (u zavisnosti od stanja u kojem se nalazi forma), ispitivanje da li je prvi slog, poslednji, jedini, da li je prazna tabela, korišćenje promenljive kao instance objekta forme (dim f as NEw frmDataGrid, gde je frmDataGrid zapravo naziv forme, koja je univerzalna za tabelarni prikaz), korišćenje klasa i univerzalnih formi, a prosleđivanje samo podataka, korišćenje događaja kao što su data error, reposition, korišćenje promenljive screen, promena mouse pointera: default i hourglass, korišćenje defaultnih vrednosti (imenovanih konstanti-VBDefault), redni broj sloga kao apsolutna pozicija (absoluteposition), kolekcija slogova: Recordset, korišćenje case strukture, Validacija nakon što je korisnik izazvao neku akciju, a pre nego što se ona zaista izvršila (Validate), i nakon što se izvršila (Reposition), korišćenje with ... end with strukture.

8. Forma za tabelarni prikaz podataka

Služi: za prikaz svih ili većine podataka iz neke tabele ili upita, sortirano/ili filterisano ili baš po redosledu unosa.

Elementi: DBGrid, Combo za unos kolone po kojoj treba da je sortirano, Combo (polje, znak) za unos kriterijuma za filtriranje (izdvajanje bitnih po nekom kriterijumu) i text box za unos vrednosti za određeno polje, komandni tasteri (za potvrdi i sl.), Combo za veze (and, or, not itd.).



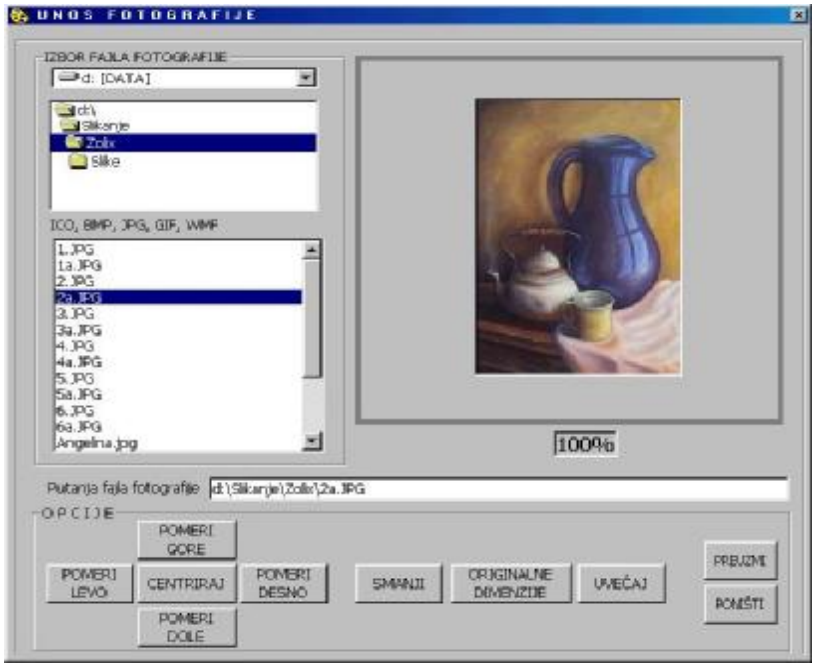
Slika 4.37. Forma za tabelarni prikaz i pretragu podataka

Tehnike: omogućiti sortiranje, filtriranje, osvežavanje, history sa vraćanjem, složene kriterijume za filter, Filter kao osobina recordseta (RS.filter=string_filtera, gde je string filtera slično kao iz WHERE za SQL), brže radi SQL, princip da sve što se radi i menja može i da se vrati (zapamti u promenljivima, kolekcijama (listi, bazi), requery (ponovo puni pokretanjem upita na kojem se zasniva), Sort kao osobina recordseta (uzima za vrednosti stringove), korišćenje picture boxa ili frejma da bi se sa većim brojem kontrola koje su na njih unesene zajednički manipulisalo, korišćenje reči a ne slika za komandne tastere, relativno pozicioniranje kontrola u odnosu na ekran, formu i sl. (Me.Height, Screen.Height i sl.), DBgrid omogućava brisanje/izmenu sloga, ali prvo se pita (Msgbox), Before Delete i cancel,pozivanje jednog događaja automatski kodom iz nekog drugog događaja (CMD1_click, na početku reda, bez call), Sortiranje tabele klikom na dbgrid, na zaglavlje (sortira kada se primeni kod za headclick)-> ASC običan click, DESC kada je click+CTRL.

9. Forma za rad sa slikama

Služi: za unos slika, za pregled slika i sl.
Elementi: Picture Box, Image Control, text box, komandni tasteri

Tehnike: Picture Box nema, a Image Box ima skaliranje (proporcionalno smenjenje i uvećanje slike koja se učita). Unos slika se može vršiti skeniranjem, pri čemu se dobije fajl koji se snima negde. Slike su potrebne u nekim informacionim sistemima radi identifikacije korisnika, mušterija, dobavljača, grafički rezultati nekih testova (snimci u medicini) i sl. Sliku treba vezati za neke tekstualne podatke, pa se naziv fajla i putanja do slike memorišu u bazi (u nekoj tabeli, uz šifru koja predstavlja vezu ka tekstualnim podacima) kao stringovi. Najbolje bi bilo sve fajlove (dokumente, slike i sl.) vezane za jednu osobu, organizaciju i sl. stavljati u jedan direktorijum. Zato bi bilo dobro da ove direktorijume pravi program, automatski sa pojavom novog partnera mu otvara vlastiti direktorijum koji bi imao naziv slično kao i šifra tog partnera. Ponekad je potrebno uvećanje slike ili delova slike; zato je potrebno odmah pri skeniranju da se slika uveliča i to pod visokom rezolucijom i takva snimi. Pogodno je koristiti MDI formu kao podlogu za uveličavanje (jer ima klizače).



Slika 4.38. Forma za rad sa slikama

10. Forma za arhiviranje podataka i manipulaciju sa fajlovima

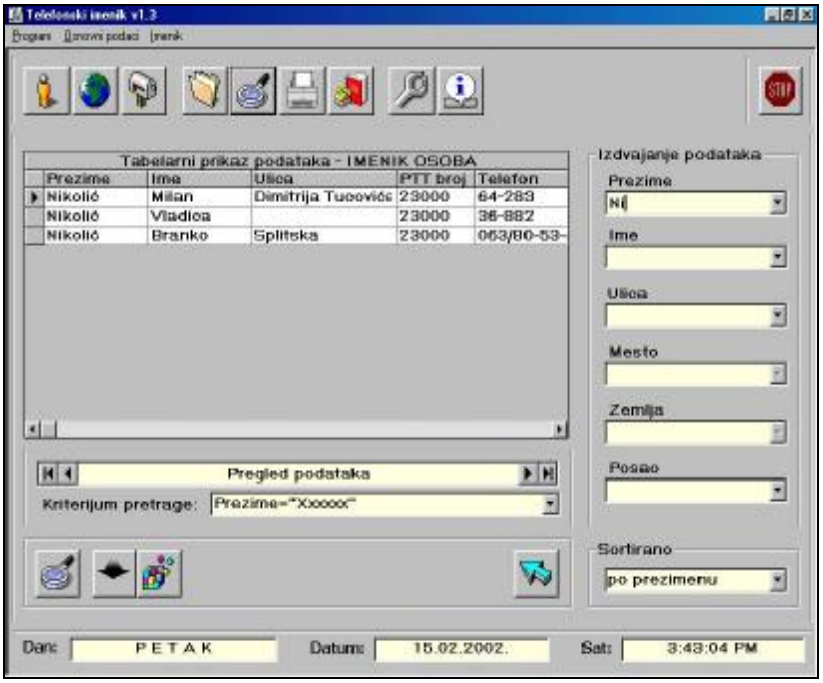
Služi: za smeštanje backup podataka, statističkih podataka, manipulaciju svim radnim fajlovima.
Elementi: Drive list, DirList, File List

Tehnike: povezati ove kontrole u funkcionalnu celinu. Da bi se fajlovi koji su u vezi sa nekim entitetom iz okruženja s njime povezali, treba da se smeštaju uvek u isti direktorijum. Najbolje bi bilo sve fajlove (dokumente, slike i sl.) vezane za jednu osobu, organizaciju i sl. stavljati u jedan direktorijum. Zato bi bilo dobro da ove direktorijume pravi program, automatski sa pojavom novog partnera mu otvara vlastiti direktorijum koji bi imao naziv slično kao i šifra tog partnera. Ipak, ta veza treba da je upisana u bazi kao naziv direktorijuma i apsolutna putanja. Svakako, pri svakom pokretanju programa treba proveriti da li su fajlovi koji su evidentirani u bazi nazivom i putanjom zaista tamo, da ne bi "pukao" program. Ukoliko nisu, omogućiti korisniku da ih pretražuje po medijima za memorisanje (diskete, hard disk, CD i sl.) iz vašeg programa. Backup treba raditi periodično, u pravilnim periodima. Najbolje je vezati backup uz izradu statističkih izveštaja.

11. Forma za pretraživanje i prikaz podataka

Služi: za pronalaženje željenig podataka (pojedinačnog ili grupa podataka).

Elementi: ZA UNOS KRITERIJUMA: combo, text box, list box, komandni taster, ZA PRIKAZ GRUPE: dbgrid, ZA PRIKAZ POJEDINAČNIH PODATAKA: text boxovi, ZA SUMARNE PODATKE: grafikoni, pie slice.



Slika 4.39. Forma za tabelarni prikaz i pretragu podataka

Tehnike: SQL, sort, filter, ključna reč ili deo reči, po svim obeležjima.

12. Forma za izveštavanje (izveštaji) i unos većeg teksta

Služi: za prikaz i štampu izveštaja, i većeg teksta
Elementi: Crystal report, Rich text

Tehnike: Crystal report je kontrola koja je veza korisničkog interfejsa i *.rpt fajla, koji se generiše u okviru editora za izveštaje, a vezuje se za određene tabele baze. Bitno je postići relativnost putanje rpt fajla, kao i relativnost putanje rpt fajla prema fajlu baze. Potrebno je kreirati najfrekventnije izveštaje, koji će se u realnim situacijama puniti realnim podacima iz baze. Takođe je poželjno postići univerzalnost izveštaja, tako da se mogu koristiti i praviti izveštaji po želji. Takođe, nekad je potrebno da se unosi tekst većeg obima. Zato nam služi Rich text , pomoću kojeg se može unositi i snimati kao txt ili rtf fajl.

Slika 4.40. Forma za rad sa izveštajima

13. Forma za dijagrame

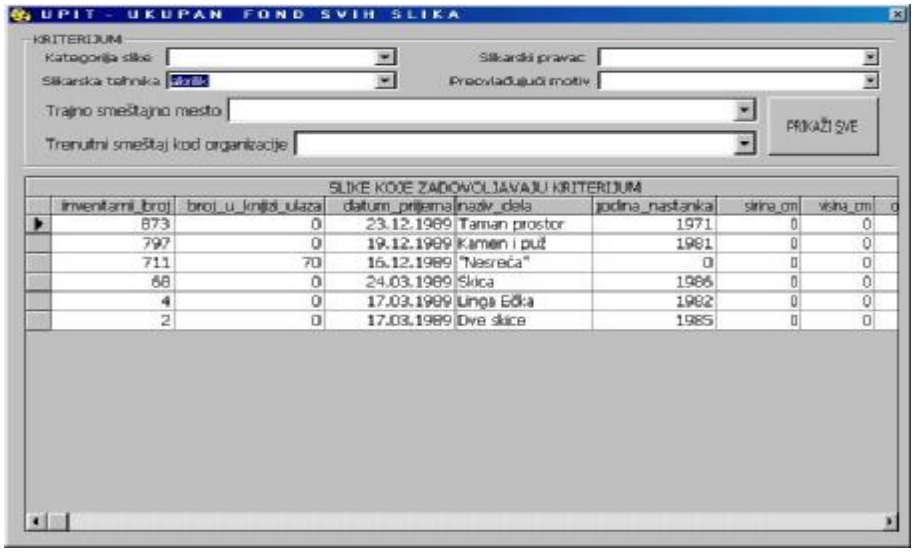
Služi: za prikaz veće količine podataka iste vrste, za prikaz statistički obrađenih podataka.
Elementi: MSChart.

Tehnike: forma sa ovom kontrolom se može upotrebljavati kao univerzalna forma. Takođe, oblici koji se mogu dobiti su različiti (dijagrami, pie slice i sl.).

14. Forma za upite

Služi: da se dobiju traženi podaci, po različitim kriterijumima.
Elementi: DbGrid, DBCombo, DbList, TextBox i sl.

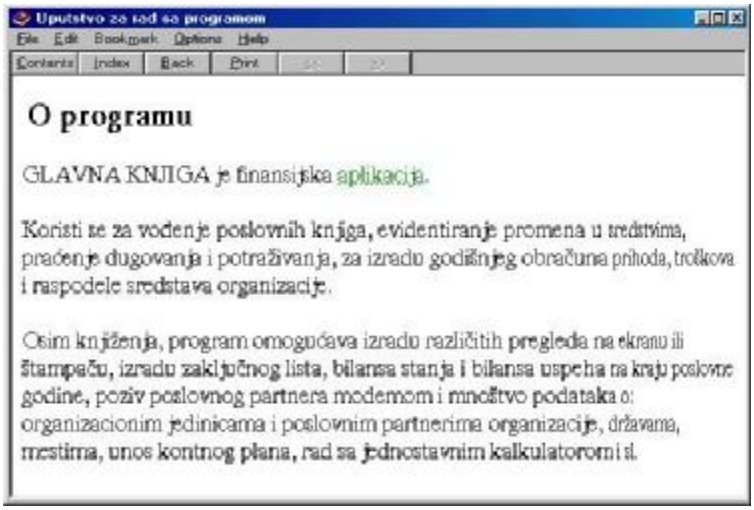
Tehnike: mogu se unositi upiti (loše), treba da se pamte najfrekventniji (za svaki poziv istog upita da se pamti koliko puta je pokretan), ili da postoji spisak upita (i to je loše). Bolja varijanta je da se napravi forma koja je korisnički orijentisana gde će biti moguće postavljati uslove upita tako što se radi formiranja kriterrijuma pretrage izabere: polje iz cele baze (svih tabela), znak, vrednost iz skupa svih unetih vrednosti za to polje, (ovaj postupak se može ponavljati, a između tih uređenih trojki se može stavljati bulov znak: and, or, not i slično) i izabere se traženi podatak (podaci) koji se traži po tom kriterijumu. Rezultat bi bio u DBGridu: kolone koje su izabrane i vrednosti (slogovi ili rezultati složenog upita).



Slika 4.41. Forma za upite nad podacima

15. Sistem za pomoć - Help sistem

Služi: za pomoć o korišćenju programa, kao i obaveštenje o trenutnom stanju programa
Elementi: Statusna linija, trake iznad prozora, tooltip svake kontrole, komandni taster sa Msgbox-om, *.hlp fajl, DBlist box sa svim pojmovima iz tabele sa objašnjenjima.



Slika 4.42. Forma sistema za pomoć

Tehnike: On-line help: statusna linija, trake iznad prozora, tooltip: u svakom trenutku da se zna koja kontrola čemu služi, u kom stanju je program, koji prozor je aktivan i čemu služi. Off-line help: komandni taster sa MSGBox-om, Hlp fajl, Chm fajl, DBList box na posebnoj formi za help. Najbolji off-line help je ako se napravi .hlp fajl (standardni Windows help fajl, koji u sebi sadrži i Word index-> listu nepoznatih pojmova, kao i procesno uputstvo u smislu: šta ako?). Da bismo sa svakog ekrana mogli dobiti pomoć, treba da imamo vezu sa delom .hlp fajla koji se na baš taj ekran odnosi.

5. LITERATURA

[BERNERS LEE i sar., 2001] Berners-Lee Tim, Hendler James, Lassila Ora: *The Semantic Web*, Scientific American, May 2001.

[BALABAN i sar., 1996] Balaban Neđo, Ristić Živan, Đurković Jovica: *Principi informatike,* Savremena administracija, Beograd, 1996.

[BOBROWSKI, 1995] Bobrowski M. Steven: *ORACLE 7 i obrada podataka po modelu klijent/server,* Mikro knjiga, Beograd, 1995.

[BOOCH i sar., 2000] Booch Grady, Rumbaugh James, Jacobson Ivar: *The Unified Modeling Language User Guide*, 2000.

[BRUEGGE i sar., 2004] Bruegge B., Dutoit A.: *Object-Oriented Software Engineering using UML Patterns and Java*, Pearson Education, Singapore, 2004.

[DEVEDŽIĆ, 2004] Devedžić Vladan (urednik): *Tehnologije inteligentnih sistema*, Univerzitet u Beogradu, Fakultet organizacionih nauka, Beograd, 2004.

[FORGEY i sar., 2002] Forgey Bill, Gosnell Denise, Reynolds Matthew: *Visual Basic.NET baze podataka*, CET Computer Equipment and Trade, Beograd, 2002.

[HALVORSSON, 2002] Halvorson Michael: *Microsoft Visual Basic.NET korak po korak*, CET Computer Equipment and Trade, Beograd, 2002.

[HENTZEN, 1996] Hentzen Whill: *Visual FoxPro 3.0*, Mikro knjiga, Beograd, 1996.

[IVKOVIĆ i sar., 2005] Ivković Miodrag, Milošević Slađana, Subić Zoran, Dobrilović Dalibor: *Elektronsko poslovanje e-Business*, Univerzitet u Novom Sadu, Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 2005.

[JAUKOVIĆ, 1992] Jauković Mihajlo, *Uvod u informacione sisteme*, Tehnička knjiga, Beograd, 1992.

[JENNINGS, 2000] Jennings Roger: *Vodić kroz Microsoft Access 2000,* CET Computer Equipment and Trade, Beograd, 2000.

[KAZI LJ., 2005] Kazi Ljubica: *Metodičko i organizaciono unapređenje kvaliteta univerzitetske nastave u okviru rzvoj integralnog softverskog sistema tehničkog fakulteta* - Magistarska teza, Tehnički fakultet “Mihajlo Pupin”, Zrenjanin, 2005.

[KAZI Z., 2005] Kazi Zoltan: *Korišćenje udaljenih baza podataka u sistemima automatskog rezonovanja* - Magistarska teza, Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 2005.

[LAZAREVIĆ i sar., 1988] Lazarević Branislav, Jovanović Vladan, Vučković Milica: *Projektovanje informacionih sistema I deo*, Naučna knjiga, Beograd, 1988.

[LAZAREVIĆ i sar., 1993] Lazarević Branislav, Nešković Siniša: *Objektno orijentisani transformacioni razvoj informacionih sistema*, Skripte, Fakultet organizacionih nauka, Beograd, 1993.

[LAZAREVIĆ i sar., 1997] Lazarević Branislav, Nešković Siniša: *Sistemsko – teorijska kritika objektno – orijentisanih pristupa razvoju* softvera, INFO – TECH, Vrnjačka Banja, 1997, str. 89-95.

[LAZAREVIĆ i sar., 1999] Lazarević Branislav, Nešković Siniša: *Nove arhitekture i pristupi objektno – orijentisanom razvoju softvera*, Časopis INFO SCIENCE 2-3/99. str. 21-26.

[LAZAREVIĆ i sar., 2003] Lazarević Branislav, Marjanović Zoran, Aničić Nenad, Babarogić Slađan: *Baze podataka,* Univerzitet u Beogradu, Fakultet organizacionih nauka, Beograd, 2003.

[LIGHT, 1997] Light R.: *Presenting XML*, Sams Publishing, 1997.

[MALBAŠKI, 2001] Malbaški Dušan: *Odabrana poglavlja Metoda programiranja*, Tehnički fakultet »Mihajlo Pupin«, Zrenjanin, 2001.

[MARINKOVIĆ, 1998] Marinković Rade: *Razvojni koncept metodologije projektovanja informacionog sistema za upravljanje proizvodnjom* – Magistarska teza, Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 1998.

[MARJANOVIĆ, 1993] Marjanović Zoran: *Aktivni sistemi baza podataka*, SYMOPIS 1993, str. 103-106.

[MIHAJLOVIĆ, 1993] Mihajlović Dragan: *Informacioni sistemi,* Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, 1993.

[MOGIN i sar., 1996] Mogin Pavle, Luković Ivan: *Principi baza podataka*, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, 1996.

[MOGIN i sar., 2000] Mogin Pavle, Luković Ivan, Govedarica Miro: *Principi projektovanja baza podataka*, Univerzitet u Novom Sadu, Edicija *Univerzitetski udžbenik,* 2000.

[RADULOVIĆ, 1997] Radulović, Biljana: *Projektovanje baza podataka u oblasti obrazovnog računarskog softvera*, Doktorska disertacija, Univerzitet u Novom Sadu, Tehnički fakultet "Mihajlo Pupin" Zrenjanin, 1997.

[SOTIROVIĆ i sar., 2004] Sotirović Velimir, Glušac Dragana, Jevtić Vesna, Eleven Erika: *Standardni softver PC-a u okruženju 2003*., Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 2004.

[STANOJEVIĆ, 1986] Stanojević Mićo: *Osnovi projektovanja informacionih* sistema, Naučna knjiga, Beograd, 1986.

[STANOJEVIĆ i sar., 1999] Stanojević Ivana, Surla Dušan: *Uvod u objedinjeni jezik modeliranja*, Novi Sad, 1999.

[ULMAN i sar., 2002] Ullman Jeffrey, Garcia – Molina Hector, Widom Jennifer, *Database Systems: The Complete Book*, Department of Computer Science, Stanford University, Prentice Hall, New Jersey, 2002.

[WYNKOOP, 2000] Wynkoop Stephen: *Vodič kroz SQL Server 7,* CET Computer Equipment and Trade, Beograd, 2000.

[ČERNIČEK, 2000] Černiček Ištvan: *Teorija sistema*, Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 2000.

[ČIP, 1995] Časopis ČIP, NIP TV Novosti, Beograd, broj. 6-10, 1994-1995.

[ISO 9000] 3 Smernice za primenu ISO 9001 u razvoju, isporuci i održavanju softvera, 1993.

[DCAGLETS] http://www.vistabonita.com/papers/DCAglets

[MYSQL] http://free-mysql.bizhostnet.com